Package 'umx'

May 25, 2020

Version 4.0.0

Date 2020-05-25

Title Structural Equation and Twin Modeling in R

Maintainer Timothy C. Bates <timothy.c.bates@gmail.com>

License GPL-3

Language en-US

Encoding UTF-8

URL https://github.com/tbates/umx

Description Quickly create, run, and report structural equation and twin models. See '?umx' for help, and umx_open_CRAN_page(``umx") for NEWS.

Depends R (>= 3.5.0), OpenMx (>= 2.11.5)

Imports cowplot, DiagrammeR, lavaan, ggplot2, knitr, MASS, Matrix, methods, MuMIn, mvtnorm, nlme, polycor, R2HTML, RCurl, utils, xtable

Suggests cocor, devtools, gdata, Hmisc, spelling, testthat, rmarkdown, psych

BugReports https://github.com/tbates/umx/issues

LazyData true

RoxygenNote 7.1.0

NeedsCompilation no

Author Timothy C. Bates [aut, cre] (<https://orcid.org/0000-0002-1153-9007>), Gillespie Nathan [ctb], Brenton Wiernik [ctb], Joshua N. Pritikin [ctb], Michael C. Neale [ctb], Hermine Maes [ctb]

Repository CRAN

Date/Publication 2020-05-25 18:40:02 UTC

R topics documented:

deg2rad . . . 7
dl_from_dropbox . . . 8
docData . . . 9
extractAIC.MxModel . . . 10
Fischbein_wt . . . 11
FishersMethod . . . 12
GFF . . . 13
install.OpenMx . . . 15
iqdat . . . 17
loadings . . . 18
loadings.MxModel . . . 19
oddsratio . . . 20
plot.MxLISRELModel . . . 21
plot.MxModel . . . 22
plot.MxModelTwinMaker . . . 25
power.ACE.test . . . 27
print.oddsratio . . . 31
print.reliability . . . 32
qm . . . 33
rad2deg . . . 34
reliability . . . 34
residuals.MxModel . . . 35
RMSEA . . . 36
RMSEA.MxModel . . . 37
RMSEA.summary.mxmodel . . . 38
SE_from_p . . . 39
tmx_genotypic_effect . . . 40
tmx_is.identified . . . 41
tmx_show . . . 42
umx . . . 44
umx-deprecated . . . 47
umxACE . . . 47
umxACEcov . . . 55
umxACEv . . . 58
umxAlgebra . . . 65
umxAPA . . . 66
umxBrownie . . . 69
umxCI . . . 70
umxCI_boot . . . 72
umxCompare . . . 73
umxConfint . . . 75
umxCov2cor . . . 77
umxCP . . . 78
umxDiagnose . . . 83
umxDoC . . . 84
umxDoCp . . . 87

umxEFA . . . 88
umxEquate . . . 91
umxExamples . . . 93
umxExpCov . . . 99
umxExpMeans . . . 100
umxFactor . . . 101
umxFactorScores . . . 102
umxFitIndices . . . 104
umxFixAll . . . 105
umxGetParameters . . . 106
umxGxE . . . 108
umxGxEbiv . . . 111
umxGxE_window . . . 113
umxHetCor . . . 115
umxIP . . . 117
umxJiggle . . . 121
umxLabel . . . 122
umxLav2RAM . . . 124
umxMatrix . . . 128
umxMendelianRandomization . . . 130
umxMI . . . 132
umxModel . . . 133
umxModify . . . 134
umxParameters . . . 137
umxPath . . . 139
umxPlotACE . . . 143
umxPlotACEcov . . . 145
umxPlotACEv . . . 146
umxPlotCP . . . 147
umxPlotDoC . . . 149
umxPlotGxE . . . 151
umxPlotGxEbiv . . . 152
umxPlotIP . . . 154
umxPlotSexLim . . . 155
umxPlotSimplex . . . 157
umxPower . . . 158
umxRAM . . . 161
umxRAM2Lav . . . 168
umxReduce . . . 169
umxReduceACE . . . 170
umxReduceGxE . . . 171
umxRenameMatrix . . . 172
umxRotate . . . 173
umxRotate.MxModelCP . . . 174
umxRun . . . 176
umxSetParameters . . . 177
umxSexLim . . . 179
umxSimplex . . . 183

umxSummarizeTwinData . . . 187
umxSummary . . . 188
umxSummary.MxModel . . . 189
umxSummaryACE . . . 191
umxSummaryACEcov . . . 193
umxSummaryACEv . . . 194
umxSummaryCP . . . 196
umxSummaryDoC . . . 198
umxSummaryGxE . . . 200
umxSummaryGxEbiv . . . 202
umxSummaryIP . . . 203
umxSummarySexLim . . . 205
umxSummarySimplex . . . 207
umxSuperModel . . . 209
umxThresholdMatrix . . . 211
umxTwinMaker . . . 214
umxUnexplainedCausalNexus . . . 217
umxValues . . . 218
umxVersion . . . 219
umxWeightedAIC . . . 220
umx_aggregate . . . 221
umx_APA_pval . . . 222
umx_apply . . . 224
umx_array_shift . . . 225
umx_as_numeric . . . 225
umx_check . . . 226
umx_check_model . . . 227
umx_check_names . . . 229
umx_check_OS . . . 230
umx_check_parallel . . . 231
umx_cont_2_quantiles . . . 232
umx_cor . . . 234
umx_explode . . . 235
umx_explode_twin_names . . . 236
umx_file_load_pseudo . . . 237
umx_find_object . . . 238
umx_fun_mean_sd . . . 239
umx_get_bracket_addresses . . . 240
umx_get_checkpoint . . . 241
umx_get_options . . . 242
umx_grep . . . 243
umx_has_been_run . . . 244
umx_has_CIs . . . 245
umx_has_means . . . 246
umx_has_square_brackets . . . 247
umx_is_class . . . 248
umx_is_cov . . . 249
umx_is_endogenous . . . 250

umx_is_exogenous . . . 251
umx_is_MxData . . . 252
umx_is_MxMatrix . . . 252
umx_is_MxModel . . . 253
umx_is_numeric . . . 254
umx_is_ordered . . . 255
umx_is_RAM . . . 256
umx_long2wide . . . 257
umx_lower2full . . . 259
umx_make . . . 262
umx_make_fake_data . . . 263
umx_make_MR_data . . . 265
umx_make_raw_from_cov . . . 266
umx_make_sql_from_excel . . . 267
umx_make_TwinData . . . 268
umx_make_twin_data_nice . . . 273
umx_means . . . 274
umx_move_file . . . 275
umx_msg . . . 276
umx_names . . . 277
umx_open . . . 279
umx_open_CRAN_page . . . 280
umx_pad . . . 281
umx_paste_names . . . 282
umx_polychoric . . . 283
umx_polypairwise . . . 284
umx_polytriowise . . . 285
umx_print . . . 287
umx_read_lower . . . 288
umx_rename . . . 290
umx_rename_file . . . 291
umx_reorder . . . 293
umx_residualize . . . 294
umx_rot . . . 295
umx_round . . . 296
umx_r_test . . . 297
umx_scale . . . 298
umx_scale_wide_twin_data . . . 299
umx_score_scale . . . 300
umx_select_valid . . . 302
umx_set_auto_plot . . . 303
umx_set_auto_run . . . 304
umx_set_checkpoint . . . 305
umx_set_condensed_slots . . . 306
umx_set_cores . . . 307
umx_set_data_variance_check . . . 308
umx_set_mvn_optimization_options . . . 309
umx_set_optimizer . . . 310

umx_set_plot_file_suffix . . . 311
umx_set_plot_format . . . 312
umx_set_separator . . . 313
umx_set_silent . . . 313
umx_set_table_format . . . 314
umx_stack . . . 315
umx_standardize . . . 316
umx_string_to_algebra . . . 317
umx_str_chars . . . 318
umx_str_from_object . . . 319
umx_time . . . 320
umx_trim . . . 321
umx_var . . . 322
umx_wide2long . . . 324
umx_write_to_clipboard . . . 325
us_skinfold_data . . . 325
xmuHasSquareBrackets . . . 327
xmuLabel_Matrix . . . 328
xmuLabel_MATRIX_Model . . . 329
xmuLabel_RAM_Model . . . 331
xmuMakeDeviationThresholdsMatrices . . . 332
xmuMakeOneHeadedPathsFromPathList . . . 333
xmuMakeTwoHeadedPathsFromPathList . . . 334
xmuMaxLevels . . . 335
xmuMI . . . 336
xmuMinLevels . . . 337
xmuPropagateLabels . . . 338
xmuRAM2Ordinal . . . 339
xmuTwinSuper_Continuous . . . 340
xmuTwinUpgradeMeansToCovariateModel . . . 342
xmu_cell_is_on . . . 343
xmu_check_levels_identical . . . 345
xmu_check_needs_means . . . 346
xmu_check_variance . . . 348
xmu_CI_merge . . . 349
xmu_CI_stash . . . 350
xmu_clean_label . . . 351
xmu_data_missing . . . 352
xmu_data_swap_a_block . . . 353
xmu_describe_data_WLS . . . 355
xmu_DF_to_mxData_TypeCov . . . 356
xmu_dot_define_shapes . . . 358
xmu_dot_maker . . . 358
xmu_dot_make_paths . . . 359
xmu_dot_make_residuals . . . 361
xmu_dot_mat2dot . . . 362
xmu_dot_move_ranks . . . 365
xmu_dot_rank . . . 366

xmu_dot_rank_str . . . 367
xmu_extract_column . . . 368
xmu_get_CI . . . 369
xmu_lavaan_process_group . . . 370
xmu_make_bin_cont_pair_data . . . 371
xmu_make_mxData . . . 373
xmu_make_TwinSuperModel . . . 375
xmu_match.arg . . . 379
xmu_name_from_lavaan_str . . . 381
xmu_PadAndPruneForDefVars . . . 382
xmu_path2twin . . . 383
xmu_path_regex . . . 384
xmu_safe_run_summary . . . 386
xmu_set_sep_from_suffix . . . 388
xmu_show_fit_or_comparison . . . 389
xmu_simplex_corner . . . 390
xmu_standardize_ACE . . . 391
xmu_standardize_ACEcov . . . 392
xmu_standardize_ACEv . . . 393
xmu_standardize_CP . . . 394
xmu_standardize_IP . . . 395
xmu_standardize_RAM . . . 396
xmu_standardize_SexLim . . . 398
xmu_standardize_Simplex . . . 399
xmu_starts . . . 400
xmu_start_value_list . . . 402
xmu_twin_add_WeightMatrices . . . 403
xmu_twin_check . . . 404
xmu_twin_get_var_names . . . 406
xmu_twin_upgrade_selDvs2SelVars . . . 407

Index **409**

`deg2rad` *Convert Degrees to Degrees*

Description

Just a helper to multiply degrees by π and divide by 180 to get radians.

note: R's trig functions, e.g. `sin()` use Radians for input! 180 Degrees is equal to $2x\pi$ radians.

Usage

```
deg2rad(deg)
```

Arguments

`deg`	The value in degrees you wish to convert to radians

Value

- value in radians

See Also

- rad2deg(), sin()

Other Miscellaneous Functions: rad2deg()

Examples

```
deg2rad(180) # pi!
```

---

dl_from_dropbox *dl_from_dropbox*

Description

Download a file from Dropbox, given either the url, or the name and key

Usage

```
dl_from_dropbox(x, key = NULL)
```

Arguments

x	Either the file name, or full dropbox URL (see example below)
key	the code after s/ and before the file name in the dropbox url

Details

Improvements would include error handling...

Value

None

References

- https://thebiobucket.blogspot.kr/2013/04/download-files-from-dropbox.html

See Also

Other File Functions: umx_file_load_pseudo(), umx_make_sql_from_excel(), umx_move_file(), umx_open(), umx_rename_file(), umx_write_to_clipboard(), umx

Examples

```
## Not run:
dl_from_dropbox("https://dl.dropboxusercontent.com/s/7kauod48r9cfhwc/tinytwinData.rda")
dl_from_dropbox("tinytwinData.rda", key = "7kauod48r9cfhwc")

## End(Not run)
```

`docData` *Twin data for Direction of causation modelling*

Description

A dataset containing indicators for two traits `varA` and `varB`, each measured in MZ and DZ twins.

Usage

```
data(docData)
```

Format

A data frame 6 manifests for each of two twins in 1400 families of MZ and DZ twins

Details

It is designed to show off `umxDoC()` testing the hypothesis `varA` causes `varB`, `varB` causes `varA`, both cause each other.

- *zygosity* "MZFF", "DZFF", "MZMM", or "DZMM"
- *varA1_T1* Twin one's manifest 1 for varA
- *varA2_T1* Twin one's manifest 2 for varA
- *varA3_T1* Twin one's manifest 3 for varA
- *varB1_T1* Twin one's manifest 1 for varB
- *varB2_T1* Twin one's manifest 2 for varB
- *varB3_T1* Twin one's manifest 3 for varB
- *varA1_T2* Twin two's manifest 1 for varA
- *varA2_T2* Twin two's manifest 2 for varA
- *varA3_T2* Twin two's manifest 3 for varA
- *varB1_T2* Twin two's manifest 1 for varB
- *varB2_T2* Twin two's manifest 2 for varB
- *varB3_T2* Twin two's manifest 3 for varB

References

- N.A. Gillespie and N.G. Martin (2005). Direction of Causation Models. In *Encyclopedia of Statistics in Behavioral Science*, **1**, 496–499. Eds. Brian S. Everitt & David C. Howell

See Also

- umxDoC(), plot.MxModelDoC(), umxSummary.MxModelDoC(), umxModify()

Other datasets: Fischbein_wt, GFF, iqdat, umx, us_skinfold_data

Examples

```
data(docData)
str(docData)
mzData = subset(docData, zygosity %in% c("MZFF", "MZMM"))
dzData = subset(docData, zygosity %in% c("DZFF", "DZMM"))
par(mfrow = c(1, 2))  # 1 rows and 3 columns
plot(varA1_T2 ~varA1_T1, ylim = c(-4, 4), data = mzData, main="MZ")
tmp = round(cor.test(~varA1_T1 + varA1_T2, data = mzData)$estimate, 2)
text(x=-4, y=3, labels = paste0("r = ", tmp))
plot(varA1_T2 ~varA1_T1, ylim = c(-4, 4), data = dzData, main="DZ")
tmp = round(cor.test(~varA1_T1 + varA1_T2, data = dzData)$estimate, 2)
text(x=-4, y=3, labels = paste0("r = ", tmp))
par(mfrow = c(1, 1))  # back to as it was
```

---

extractAIC.MxModel *Extract AIC from MxModel*

Description

Returns the AIC for an OpenMx model. Original Author: Brandmaier

Usage

```
## S3 method for class 'MxModel'
extractAIC(fit, scale, k, ...)
```

Arguments

fit	an fitted mxModel() from which to get the AIC
scale	not used
k	not used
...	any other parameters (not used)

Value

- AIC value

References

- https://openmx.ssri.psu.edu/thread/931#comment-4858

See Also

- AIC(), umxCompare(), logLik()

Other Reporting functions: RMSEA.MxModel(), RMSEA.summary.mxmodel(), RMSEA(), loadings(), residuals.MxModel(), umxCI_boot(), umxCI(), umxCompare(), umxConfint(), umxExpCov(), umxExpMeans(), umxFitIndices(), umxPlotACEv(), umxRotate(), umxSummary.MxModel()

Examples

```
require(umx)
data(demoOneFactor)
manifests = names(demoOneFactor)
m1 = umxRAM("One Factor", data = demoOneFactor, type = "cov",
umxPath("G", to = manifests),
umxPath(var = manifests),
umxPath(var = "G", fixedAt = 1)
)
extractAIC(m1)
# -2.615998
AIC(m1)
```

Fischbein_wt *Weight data across time.*

Description

A dataframe containing correlations of weight for 66 females measured 6 times at 6-month intervals.

Usage

```
data(Fischbein_wt)
```

Format

A 6*6 correlation matrix based on n = 66 female subjects.

Details

- Weight1: Weight at time 1 (t0)
- Weight2: Weight at time 2 (t0 + 6 months)
- Weight3: Weight at time 3 (t0 + 12 months)
- Weight4: Weight at time 4 (t0 + 18 months)
- Weight5: Weight at time 5 (t0 + 24 months)
- Weight6: Weight at time 6 (t0 + 32 months)

Created as follows:

```
Fischbein_wt = umx_read_lower(file = "", diag = TRUE, names = paste0("Weight", 1:6), ensurePD= TRUE)
1.000
0.985 1.000
0.968 0.981 1.000
0.957 0.970 0.985 1.000
0.932 0.940 0.964 0.975 1.000
0.890 0.897 0.927 0.949 0.973 1.000
```

References

Fischbein, S. (1977). Intra-pair similarity in physical growth of monozygotic and of dizygotic twins during puberty. *Annals of Human Biology*, **4**. 417-430. https://doi.org/10.1080/03014467700002401

See Also

Other datasets: GFF, docData, iqdat, umx, us_skinfold_data

Examples

```
data(Fischbein_wt) # load the data
str(Fischbein_wt) # data.frame
as.matrix(Fischbein_wt) # convert to matrix
```

FishersMethod *Fishers Method of combining p-values.*

Description

FishersMethod implements R.A. Fisher's method for creating a meta-analytic p-value by combining a set of p-values from tests of the same hypothesis in independent samples,

Usage

```
FishersMethod(pvalues, ...)
```

Arguments

pvalues	A vector of p-values, e.g. c(.041, .183)
...	More p-values if you want to offer them up one by one instead of wrapping in a vector for pvalues

Value

- A meta-analytic p-value

References

- Fisher, R.A. (1925). *Statistical Methods for Research Workers*. Oliver and Boyd (Edinburgh). ISBN 0-05-002170-2. Fisher, R. A (1948). "Questions and answers #14". *The American Statistician*. **2**: 30–31. doi:10.2307/2681650. JSTOR 2681650. See also Stouffer's method for combining Z scores, which allows weighting. Stouffer, S. A. and Suchman, E. A. and DeVinney, L. C. and Star, S. A. and Williams, R. M. Jr. (1949) The American Soldier, Vol. 1 - Adjustment during Army Life. Princeton, Princeton University Press.

See Also

Other Miscellaneous Stats Helpers: `SE_from_p()`, `oddsratio()`, `reliability()`, `umxCov2cor()`, `umxHetCor()`, `umxWeightedAIC()`, `umx_apply()`, `umx_cor()`, `umx_means()`, `umx_r_test()`, `umx_round()`, `umx_scale()`, `umx_var()`, `umx`

Examples

```
FishersMethod(c(.041, .378))
```

GFF *Twin data: General Family Functioning, divorce, and well-being.*

Description

Measures of family functioning, happiness and related variables in twins, and their brothers and sisters. (see details)

Usage

```
data(GFF)
```

Format

A data frame with 1000 rows of twin-family data columns.

Details

Several scales in the data are described in van der Aa et al. (2010). General Family Functioning (GFF) refers to adolescents' evaluations general family health vs. pathology. It assesses problem solving, communication, roles within the household, affection, and control. GFF was assessed with a Dutch translation of the General Functioning sub-scale of the McMaster Family Assessment Device (FAD) (Epstein et al., 1983).

Family Conflict (FC) refers to adolescents' evaluations of the amount of openly expressed anger, aggression, and conflict among family members. Conflict sub-scale of the Family Environment Scale (FES) (Moos, 1974)

Quality of life in general (QLg) was assessed with the 10-step Cantril Ladder from best- to worst-possible life (Cantril, 1965).

- *zyg_6grp*: Six-level zygosity: MZMM, DZMM, MZFF, DZFF, DZMF, DZFM
- *zyg_2grp*: Two-level zygosity measure: 'MZ', 'DZ'
- *divorce*: Parental divorce status: 0 = No, 1 = Yes
- *sex_T1*: Sex of twin 1: 0 = "male", 1 = "female"
- *age_T1*: Age of twin 1 (years)
- *gff_T1*: General family functioning for twin 1
- *fc_T1*: Family conflict sub-scale of the FES
- *qol_T1*: Quality of life for twin 1
- *hap_T1*: General happiness for twin 1
- *sat_T1*: Satisfaction with life for twin 1
- *AD_T1*: Anxiety and Depression for twin 1
- *SOMA_T1*: Somatic complaints for twin 1
- *SOC_T1*: Social problems for twin 1
- *THOU_T1*: Thought disorder problems for twin 1
- *sex_T2*: Sex of twin 2
- *age_T2*: Age of twin 2
- *gff_T2*: General family functioning for twin 2
- *fc_T2*: Family conflict sub-scale of the FES
- *qol_T2*: Quality of life for twin 2
- *hap_T2*: General happiness for twin 2
- *sat_T2*: Satisfaction with life for twin 2
- *AD_T2*: Anxiety and Depression for twin 2
- *SOMA_T2*: Somatic complaints for twin 2
- *SOC_T2*: Social problems for twin 2
- *THOU_T2*: Thought disorder problems for twin 2
- *sex_Ta*: Sex of sib 1
- *age_Ta*: Age of sib 1
- *gff_Ta*: General family functioning for sib 1
- *fc_Ta*: Family conflict sub-scale of the FES
- *qol_Ta*: Quality of life for sib 1
- *hap_Ta*: General happiness for sib 1
- *sat_Ta*: Satisfaction with life for sib 1
- *AD_Ta*: Anxiety and Depression for sib 1
- *SOMA_Ta*: Somatic complaints for sib 1
- *SOC_Ta*: Social problems for sib 1
- *THOU_Ta*: Thought disorder problems for sib 1
- *sex_Ts*: Sex of sib 2

- *age_Ts*: Age of sib 2
- *gff_Ts*: General family functioning for sib 2
- *fc_Ts*: Family conflict sub-scale of the FES
- *qol_Ts*: Quality of life for sib 2
- *hap_Ts*: General happiness for sib 2
- *sat_Ts*: Satisfaction with life for sib 2
- *AD_Ts*: Anxiety and Depression for sib 2
- *SOMA_Ts*: Somatic complaints for sib 2
- *SOC_Ts*: Social problems for sib 2
- *THOU_Ts*: Thought disorder problems for sib 2

References

van der Aa, N., Boomsma, D. I., Rebollo-Mesa, I., Hudziak, J. J., & Bartels, M. (2010). Moderation of genetic factors by parental divorce in adolescents' evaluations of family functioning and subjective wellbeing. Twin Research and Human Genetics, 13(2), 143-162. doi:10.1375/twin.13.2.143

See Also

Other datasets: `Fischbein_wt`, `docData`, `iqdat`, `umx`, `us_skinfold_data`

Examples

```
# Twin 1 variables (end in '_T1')
data(GFF)
umx_names(GFF, "1$") # Just variables ending in 1 (twin 1)
str(GFF) # first few rows

m1 = umxACE(selDVs= "gff", sep = "_T",
mzData = subset(GFF, zyg_2grp == "MZ"),
dzData = subset(GFF, zyg_2grp == "DZ")
)
```

`install.OpenMx` *Install OpenMx, with choice of builds*

Description

You can install OpenMx, including the latest NPSOL-enabled build of OpenMx. Options are:

1. "NPSOL": Install from our repository (default): This is where we maintain binaries supporting parallel processing and NPSOL.
2. "travis": Install the latest travis built (MacOS only).
3. "CRAN": Install from CRAN.
4. "open travis build page": Open the list of travis builds in a browser window.

Usage

```
install.OpenMx(
  loc = c("NPSOL", "travis", "CRAN", "open travis build page", "UVa"),
  url = NULL,
  lib,
  repos = getOption("repos")
)
```

Arguments

loc	Version to get default is "NPSOL". "travis" (latest build),CRAN, list of builds.
url	Custom URL. On Mac, set this to "Finder" and the package selected in the Finder will be installed.
lib	Where to install the package.
repos	Which repository to use (ignored currently).

Value

None

References

- https://github.com/tbates/umx, https://tbates.github.io

See Also

umxVersion()

Other Miscellaneous Utility Functions: qm(), umxBrownie(), umxLav2RAM(), umxRAM2Lav(), umxVersion(), umx_array_shift(), umx_find_object(), umx_msg(), umx_open_CRAN_page(), umx_pad(), umx_print(), umx_score_scale(), umx

Examples

```
## Not run:
install.OpenMx() # gets the NPSOL version
install.OpenMx("NPSOL") # gets the NPSOL version explicitly
install.OpenMx("CRAN") # Get the latest CRAN version
install.OpenMx("open travis build page") # Open web page of travis builds

## End(Not run)
```

iqdat *Twin data: IQ measured longitudinally across 4 ages.*

Description

Measures of IQ across four ages in 261 pairs of identical twins and 301 pairs of fraternal (DZ) twins. (see details). It is used as data for the [umxSimplex()] examples.

Usage

```
data(iqdat)
```

Format

A data frame with 562 rows (twin families). Nine measures on each twin.

Details

- zygosity Zygosity (MZ or DZ)
- IQ_age1_T1 T1 IQ measured at age 1
- IQ_age2_T1 T1 IQ measured at age 2
- IQ_age3_T1 T1 IQ measured at age 3
- IQ_age4_T1 T1 IQ measured at age 4
- IQ_age1_T2 T2 IQ measured at age 1
- IQ_age2_T2 T2 IQ measured at age 2
- IQ_age3_T2 T2 IQ measured at age 3
- IQ_age4_T2 T2 IQ measured at age 4

References

Boomsma, D. I., Martin, N. G., & Molenaar, P. C. (1989). Factor and simplex models for repeated measures: application to two psychomotor measures of alcohol sensitivity in twins. *Behavior Genetics*, **19**, 79-96. Retrieved from <https://www.ncbi.nlm.nih.gov/pubmed/2712815>

See Also

[umxSimplex()]

Other datasets: Fischbein_wt, GFF, docData, umx, us_skinfold_data

Examples

```
data(iqdat)
str(iqdat)
par(mfrow = c(1, 3))  # 1 rows and 3 columns
plot(IQ_age4_T1 ~ IQ_age4_T2, ylim = c(50, 150), data = subset(iqdat, zygosity == "MZ"))
plot(IQ_age4_T1 ~ IQ_age4_T2, ylim = c(50, 150), data = subset(iqdat, zygosity == "DZ"))
plot(IQ_age1_T1 ~ IQ_age4_T2, data = subset(iqdat, zygosity == "MZ"))
par(mfrow = c(1, 1))  # back to as it was
```

`loadings` *loadings Generic loadings function to extract factor loadings from exploratory or confirmatory factor analyses.*

Description

See `loadings.MxModel` to access the loadings of OpenMx EFA models.

Usage

```
loadings(x, ...)
```

Arguments

`x`	an object from which to get loadings
`...`	additional parameters

Details

Base `loadings` handles `factanal()` objects.

Value

- matrix of loadings

References

- https://tbates.github.io, https://github.com/tbates/umx

See Also

Other Reporting functions: `RMSEA.MxModel()`, `RMSEA.summary.mxmodel()`, `RMSEA()`, `extractAIC.MxModel()`, `residuals.MxModel()`, `umxCI_boot()`, `umxCI()`, `umxCompare()`, `umxConfint()`, `umxExpCov()`, `umxExpMeans()`, `umxFitIndices()`, `umxPlotACEv()`, `umxRotate()`, `umxSummary.MxModel()`

loadings.MxModel *Extract factor loadings from an EFA (factor analysis).*

Description

loadings extracts the factor loadings from an EFA (factor analysis) model. It behaves equivalently to stats::loadings, returning the loadings from an EFA (factor analysis). However it does not store the rotation matrix.

Usage

```
## S3 method for class 'MxModel'
loadings(x, ...)
```

Arguments

x	A RAM model from which to get loadings.
...	Other parameters (currently unused)

Value

- loadings matrix

References

- https://github.com/tbates/umx, https://tbates.github.io

See Also

- factanal(), loadings()

Other Reporting Functions: umxAPA(), umxFactorScores(), umxGetParameters(), umxParameters(), umxReduce(), umx_aggregate(), umx_names(), umx_time(), umx

Examples

```
myVars <- c("mpg", "disp", "hp", "wt", "qsec")
m1 = umxEFA(name = "test", factors = 2, data = mtcars[, myVars])
loadings(m1)
```

`oddsratio` *Compute odds ratio (OR)*

Description

Returns the odds in each group, and the odds ratio. Takes the cases (n) and total N as a list of two numbers for each of two groups.

Usage

```
oddsratio(grp1 = c(n = 3, N = 10), grp2 = c(n = 1, N = 10), alpha = 0.05)
```

Arguments

`grp1`	either odds for group 1, or cases and total N , e.g c(n=3, N=10)
`grp2`	either odds for group 2, or cases and total N , e.g c(n=1, N=20)
`alpha`	for CI (default = 0.05)

Details

Returns a list of odds1, odds2, and OR + CI. Has a pretty-printing method so displays as:

```
Group 1 odds = 0.43
Group 2 odds = 0.11
          OR = 3.86 CI95[0.160, 3.64]
```

Value

- List of odds in group 1 and group2, and the resulting OR and CI

References

- https://github.com/tbates/umx, https://tbates.github.io

See Also

- `umx_r_test()`

Other Miscellaneous Stats Helpers: `FishersMethod()`, `SE_from_p()`, `reliability()`, `umxCov2cor()`, `umxHetCor()`, `umxWeightedAIC()`, `umx_apply()`, `umx_cor()`, `umx_means()`, `umx_r_test()`, `umx_round()`, `umx_scale()`, `umx_var()`, `umx`

Examples

```
oddsratio(grp1 = c(1, 10), grp2 = c(3, 10))
oddsratio(grp1 = c(3, 10), grp2 = c(1, 10))
oddsratio(grp1 = c(3, 10), grp2 = c(1, 10), alpha = .01)
```

`plot.MxLISRELModel` *Create and display a graphical path diagram for a LISREL model.*

Description

`plot.MxLISRELModel` produces SEM diagrams using `DiagrammeR::DiagrammeR()` (or a graphviz application) to create the image.

Usage

```
## S3 method for class 'MxLISRELModel'
plot(
  x = NA,
  std = FALSE,
  fixed = TRUE,
  means = TRUE,
  digits = 2,
  file = "name",
  labels = c("none", "labels", "both"),
  resid = c("circle", "line", "none"),
  strip_zero = TRUE,
  ...
)
```

Arguments

`x`	A LISREL `mxModel()` from which to make a path diagram
`std`	Whether to standardize the model (default = FALSE).
`fixed`	Whether to show fixed paths (defaults to TRUE)
`means`	Whether to show means or not (default = TRUE)
`digits`	The number of decimal places to add to the path coefficients
`file`	The name of the dot file to write: NA = none; "name" = use the name of the model
`labels`	Whether to show labels on the paths. both will show both the parameter and the label. ("both", "none" or "labels")
`resid`	How to show residuals and variances default is "circle". Options are "line" & "none"
`strip_zero`	Whether to strip the leading "0" and decimal point from parameter estimates (default = TRUE)
`...`	Optional parameters

Details

Note: By default, plots open in your browser (or plot pane if using RStudio).

Opening in an external editor/app

The underlying format is graphviz. If you use `umx_set_plot_format("graphviz")`, figures will open in a graphviz helper app (if installed). If you use graphviz, we try and use that app, but YOU HAVE TO INSTALL IT!

On MacOS, you may need to associate the '.gv' extension with your graphviz app. Find the .gv file made by plot, get info (cmd-I), then choose "open with", select graphviz.app (or OmniGraffle professional), then set "change all".

The commercial application "OmniGraffle" is great for editing these images.

References

- https://www.github.com/tbates/umx, https://en.wikipedia.org/wiki/DOT_(graph_description_language)

See Also

- umx_set_plot_format(), umx_set_auto_plot(), umx_set_plot_format(), plot.MxModel(), umxPlotACE(), umxPlotCP(), umxPlotIP(), umxPlotGxE()

Other umx S3 functions: plot.MxModel()

Other Plotting functions: plot.MxModel(), umxPlotACEcov(), umxPlotACEv(), umxPlotACE(), umxPlotCP(), umxPlotGxEbiv(), umxPlotGxE(), umxPlotIP(), umxPlotSexLim(), umxPlotSimplex(), umx

Examples

```
# plot()
# TODO get LISREL example model
# Figure out how to map its matrices to plot. Don't do without establishing demand.
```

---

plot.MxModel *Create and display a graphical path diagram for a model.*

Description

plot() produces SEM diagrams in graphviz format, and relies on DiagrammeR() (or a graphviz application) to create the image. *Note:* DiagrammeR is supported out of the box. By default, plots open in your browser.

Usage

```
## S3 method for class 'MxModel'
plot(
  x = NA,
  std = FALSE,
  fixed = TRUE,
  means = TRUE,
  digits = 2,
  file = "name",
  labels = c("none", "labels", "both"),
  resid = c("circle", "line", "none"),
  strip_zero = FALSE,
  splines = TRUE,
  min = NULL,
  same = NULL,
  max = NULL,
  ...
)
```

Arguments

x	An mxModel() from which to make a path diagram
std	Whether to standardize the model (default = FALSE).
fixed	Whether to show fixed paths (defaults to TRUE)
means	Whether to show means or not (default = TRUE)
digits	The number of decimal places to add to the path coefficients
file	The name of the dot file to write: NA = none; "name" = use the name of the model
labels	Whether to show labels on the paths. "none", "labels", or "both" (parameter + label).
resid	How to show residuals and variances default is "circle". Options are "line" & "none"
strip_zero	Whether to strip the leading "0" and decimal point from parameter estimates (default = FALSE)
splines	Whether to allow lines to curve: defaults to TRUE (nb: some models look better with FALSE)
min	optional list of objects to group at the top of the plot. Default (NULL) chooses automatically.
same	optional list of objects to group at the same rank in the plot. Default (NULL) chooses automatically.
max	optional list of objects to group at the bottom of the plot. Default (NULL) chooses automatically.
...	Optional parameters

Details

If you use umx_set_plot_format("graphviz"), they will open in a graphviz helper app (if installed). The commercial application "OmniGraffle" is great for editing these images. On unix and windows, plot() will create a pdf and open it in your default pdf reader.

If you use graphviz, we try and use that app, but YOU HAVE TO INSTALL IT!

MacOS note: On Mac, we will try and open an app: you may need to associate the '.gv' extension with the graphviz app. Find the .gv file made by plot, get info (cmd-I), then choose "open with", select graphviz.app (or OmniGraffle professional), then set "change all".

References

- https://www.github.com/tbates/umx, https://en.wikipedia.org/wiki/DOT_(graph_description_language)

See Also

- umx_set_plot_format(), plot.MxModel(), umxPlotACE(), umxPlotCP(), umxPlotIP(), umxPlotGxE()

Other umx S3 functions: plot.MxLISRELModel()

Other Plotting functions: plot.MxLISRELModel(), umxPlotACEcov(), umxPlotACEv(), umxPlotACE(), umxPlotCP(), umxPlotGxEbiv(), umxPlotGxE(), umxPlotIP(), umxPlotSexLim(), umxPlotSimplex(), umx

Examples

```
## Not run:
require(umx)
data(demoOneFactor)
manifests = names(demoOneFactor)
m1 = umxRAM("One Factor", data = demoOneFactor, type = "cov",
umxPath("G", to = manifests),
umxPath(var = manifests),
umxPath(var = "G", fixedAt = 1)
)
plot(m1)
plot(m1, std = TRUE, resid = "line", digits = 3, strip_zero = FALSE)

# ============================================================
# = With a growth model, demonstrate splines= false to force =
# = straight lines, and move "rank" of intercept object      =
# ============================================================

m1 = umxRAM("grow", data = myGrowthMixtureData,
umxPath(var = manifests, free = TRUE),
umxPath(means = manifests, fixedAt = 0),
umxPath(v.m. = c("intercept","slope")),
umxPath("intercept", with = "slope"),
umxPath("intercept", to = manifests, fixedAt = 1),
umxPath("slope", to = manifests, arrows = 1, fixedAt = c(0, 1, 2, 3, 4))
```

```
)

plot(m1, means=FALSE, std=TRUE, strip=TRUE, splines="FALSE", max="intercept")

## End(Not run) # end dontrun
```

`plot.MxModelTwinMaker` *Create and display a graphical path diagram for a path-based twin model.*

Description

Assumes the model has a group called "MZ" inside.

Usage

```
## S3 method for class 'MxModelTwinMaker'
plot(
  x = NA,
  std = FALSE,
  fixed = TRUE,
  means = TRUE,
  oneTwin = TRUE,
  sep = "_T",
  digits = 2,
  file = "name",
  labels = c("none", "labels", "both"),
  resid = c("circle", "line", "none"),
  strip_zero = FALSE,
  splines = TRUE,
  min = NULL,
  same = NULL,
  max = NULL,
  ...
)
```

Arguments

x	A umxTwinMaker() model from which to make a path diagram
std	Whether to standardize the model (default = FALSE)
fixed	Whether to show fixed paths (defaults to TRUE)
means	Whether to show means or not (default = TRUE)
oneTwin	(whether to plot a pair of twins, or just one (default = TRUE)
sep	The separator for twin variables ("_T")
digits	The number of decimal places to add to the path coefficients

file	The name of the dot file to write: NA = none; "name" = use the name of the model
labels	Whether to show labels on the paths. "none", "labels", or "both" (parameter + label).
resid	How to show residuals and variances default is "circle". Options are "line" & "none"
strip_zero	Whether to strip the leading "0" and decimal point from parameter estimates (default = FALSE)
splines	Whether to allow lines to curve: defaults to TRUE (nb: some models look better with FALSE)
min	optional list of objects to group at the top of the plot. Default (NULL) chooses automatically.
same	optional list of objects to group at the same rank in the plot. Default (NULL) chooses automatically.
max	optional list of objects to group at the bottom of the plot. Default (NULL) chooses automatically.
...	Optional parameters

Details

If you use umx_set_plot_format("graphviz"), they will open in a graphviz helper app (if installed). The commercial application "OmniGraffle" is great for editing these images. On unix and windows, plot() will create a pdf and open it in your default pdf reader.

See Also

- umx_set_plot_format(), plot.MxModel(), umxPlotACE(), umxPlotCP(), umxPlotIP(), umxPlotGxE()

Other Twin Modeling Functions: power.ACE.test(), umxACEcov(), umxACEv(), umxACE(), umxCP(), umxDoCp(), umxDoC(), umxGxE_window(), umxGxEbiv(), umxGxE(), umxIP(), umxRotate.MxModelCP(), umxSexLim(), umxSimplex(), umx

Examples

```
## Not run:
require(umx)
#
# =====================
# = Make an ACE model =
# =====================
# 1. Clean data: Add separator and scale
data(twinData)
tmp = umx_make_twin_data_nice(data=twinData, sep="", zygosity="zygosity", numbering=1:2)
tmp = umx_scale_wide_twin_data(varsToScale= c("wt", "ht"), sep= "_T", data= tmp)
mzData = subset(tmp, zygosity %in%  c("MZFF", "MZMM"))
dzData = subset(tmp, zygosity %in%  c("DZFF", "DZMM"))

# 2. Define paths: You only need the paths for one person:
```

```
paths = c(
umxPath(v1m0 = c("a1", 'c1', "e1")),
umxPath(means = c("wt")),
umxPath(c("a1", 'c1', "e1"), to = "wt", values=.2)
)
m1 = umxTwinMaker("test", paths, mzData = mzData, dzData= dzData)
plot(m1, std= TRUE, means= FALSE)
plot(m1, means=FALSE, std=TRUE, strip=TRUE, splines="FALSE", max="intercept")

## End(Not run) # end dontrun

# =================
# = An ACEv model =
# =================
# Not complete

paths = c(
umxPath(v1m0 = c("A1", 'C1', "E1")),
umxPath(v1m0 = c("A2", 'C2', "E2")),
umxPath(v.m0 = c("l1", 'l2')),
umxPath(v.m. = c("wt", "ht")),
umxPath(c("A1", 'C1', "E1"), to = "l1", values= .2),
umxPath(c("A2", 'C2', "E2"), to = "l2", values= .2),
umxPath(c("l1", 'l2'), to = c("wt", "ht"), values= .2)
)
```

---

power.ACE.test *Test the power of an ACE model to detect paths of interest.*

Description

power.ACE.test simulates a univariate ACE model (with nMZpairs= 2000 and MZ_DZ_ratio*nMZpairs DZ twins. It computes power to detect dropping one or more paths specified in drop=. The interface and functionality of this service are experimental and subject to change.

Usage

```
power.ACE.test(
  AA = 0.5,
  CC = 0,
  EE = NULL,
  update = c("a", "c", "a_after_dropping_c"),
  value = 0,
  n = NULL,
  MZ_DZ_ratio = 1,
  sig.level = 0.05,
  power = 0.8,
  method = c("ncp", "empirical"),
```

```
  search = FALSE,
  tryHard = c("yes", "no", "ordinal", "search"),
  digits = 2,
  optimizer = NULL,
  nSim = 4000
)
```

Arguments

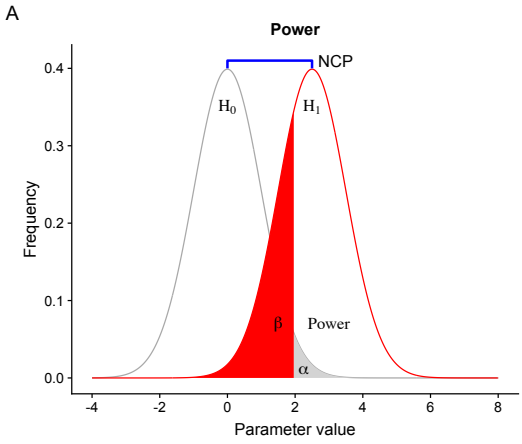
AA	Additive genetic variance (Default .5)
CC	Shared environment variance (Default 0)
EE	Unique environment variance. Leave NULL (default) to compute an amount summing to 1
update	Component to drop (Default "a", i.e., drop a)
value	Value to set dropped path to (Default 0)
n	If provided, solve at the given n (Default NULL)
MZ_DZ_ratio	MZ pairs per DZ pair (Default 1 = equal numbers.)
sig.level	alpha (p-value) Default = 0.05
power	Default = .8 (80 percent power, equal to 1 - Type II rate)
method	How to estimate power: Default = use non-centrality parameter ("ncp"). Alternative is "empirical"
search	Whether to return a search across power or just a point estimate (Default FALSE = point)
tryHard	Whether to tryHard to find a solution (default = "yes", alternatives are "no"...)
digits	Rounding for reporting parameters (default 2)
optimizer	If set, will switch the optimizer.
nSim	Total number of pairs to simulate in the models (default = 4000)

Details

Statistical power is the proportion of studies that, over the long run, one should expect to yield a statistically significant result given certain study characteristics such as sample size (N), the expected effect size (β), and the criterion for statistical significance (α).

A typical target for power is 80%. Much as the accepted critical p-value is .05, this has emerged as a trade off, in this case of resources required for more powerful studies against the cost of missing a true effect. People interested in truth discourage running studies with low power: A study with 20 percent power will fail to detect real effects 80% of the time. But even with zero power, the Type-I error rate remains a nominal 5% (and with any researcher degrees of freedom, perhaps much more than that). Low powered research, then, fails to detect true effects, and generates support for random false theories about as often. This sounds silly, but empirical rates are often as low as 20% (Button, et al., 2013).

Illustration of α, β, and power (1-β):



Value

`OpenMx::mxPower()` object

References

- Visscher, P.M., Gordon, S., Neale, M.C. (2008). Power of the classical twin design revisited: II detection of common environmental variance. *Twin Res Hum Genet*, **11**: 48-54. doi: 10.1375/twin.11.1.48
- Button, K. S., Ioannidis, J. P., Mokrysz, C., Nosek, B. A., Flint, J., Robinson, E. S., and Munafo, M. R. (2013). Power failure: why small sample size undermines the reliability of neuroscience. *Nature Reviews Neuroscience*, **14**, 365-376. doi: 10.1038/nrn3475

See Also

- `OpenMx::mxPower()`

Other Twin Modeling Functions: `plot.MxModelTwinMaker()`, `umxACEcov()`, `umxACEv()`, `umxACE()`, `umxCP()`, `umxDoCp()`, `umxDoC()`, `umxGxE_window()`, `umxGxEbiv()`, `umxGxE()`, `umxIP()`, `umxRotate.MxModelCP()`, `umxSexLim()`, `umxSimplex()`, `umx`

Examples

```
# ================================================
# = Power to detect a^2=.5 with equal MZ and DZ =
# ================================================
power.ACE.test(AA = .5, CC = 0, update = "a")
# Suggests n = 84 MZ and 94 DZ pairs.

## Not run:
# ================================
# = Show power across range of N =
# ================================
power.ACE.test(AA= .5, CC= 0, update = "a", search = TRUE)
```

```
# Salutary note: You need well fitting models with correct betas in the data
# for power to be valid.
# tryHard helps ensure this, as does the default nSim= 4000 pair data.
# Power is important to get right, so I recommend using tryHard = "yes" (the default)
power.ACE.test(AA= .5, CC= 0, update = "a")

# =====================
# = Power to detect C =
# =====================

# 102 of each of MZ and DZ pairs for 80% power.
power.ACE.test(AA= .5, CC= .3, update = "c")

# ========================================
# = Set 'a' to a fixed, but non-zero value =
# ========================================

power.ACE.test(update= "a", value= sqrt(.2), AA= .5, CC= 0)

# ======================================
# = Drop More than one parameter (A & C) =
# ======================================
# E vs AE: the hypothesis that twins show no familial similarity.
power.ACE.test(update = "a_after_dropping_c", AA= .5, CC= .3)

# ================================================
# = More power to detect A > 0 when more C present  =
# ================================================

power.ACE.test(update = "a", AA= .5, CC= .0)
power.ACE.test(update = "a", AA= .5, CC= .3)

# ================================================
# = More power to detect C > 0 when more A present?  =
# ================================================

power.ACE.test(update = "c", AA= .0, CC= .5)
power.ACE.test(update = "c", AA= .3, CC= .5)


# ============================================
# = Power with more DZs than MZs and vice versa =
# ============================================

# Power about the same: total pairs with 2 MZs per DZ = 692, vs. 707
power.ACE.test(MZ_DZ_ratio= 2/1, update= "a", AA= .3, CC= 0, method="ncp", tryHard="yes")
power.ACE.test(MZ_DZ_ratio= 1/2, update= "a", AA= .3, CC= 0, method="ncp", tryHard="yes")


# ===================================
# = Compare ncp and empirical methods =
# ===================================
```

```
# Compare to empirical mode: suggests 83.6 MZ and 83.6 DZ pairs

power.ACE.test(update= "a", AA= .5, CC= 0, method= "empirical")
# method= "empirical": For 80% power, you need 76 MZ and 76 DZ pairs
power.ACE.test(update= "a", AA= .5, CC= 0, method = "ncp")
# method = "ncp": For 80% power, you need 83.5 MZ and 83.5 DZ pairs

# ====================
# = Show off options =
# ====================
# 1. tryHard

power.ACE.test(update = "a", AA= .5, CC= 0, tryHard= "no")

# 2. toggle optimizer
power.ACE.test(update= "a", AA= .5, CC= 0, optimizer= "SLSQP")

# 3. How many twin pairs in the base simulated data?
power.ACE.test(update = "a", AA= .5, CC= 0)
power.ACE.test(update = "a", AA= .5, CC= 0, nSim= 20)


## End(Not run)
```

`print.oddsratio` *Print a scale "oddsratio" object*

Description

Print method for the `umx::oddsratio()` function.

Usage

```
## S3 method for class 'oddsratio'
print(x, digits = 3, ...)
```

Arguments

`x`	A `umx::oddsratio()` result.
`digits`	The rounding precision.
`...`	further arguments passed to or from other methods.

Value

- invisible oddsratio object (x).

See Also

- `print()`, `umx::oddsratio()`,

Examples

```
oddsratio(grp1 = c(1, 10), grp2 = c(3, 10))
oddsratio(grp1 = c(3, 10), grp2 = c(1, 10))
oddsratio(grp1 = c(3, 10), grp2 = c(1, 10), alpha = .01)
```

---

print.reliability *Print a scale "reliability" object*

Description

Print method for the umx::reliability() function.

Usage

```
## S3 method for class 'reliability'
print(x, digits = 4, ...)
```

Arguments

x	A umx::reliability() result.
digits	The rounding precision.
...	further arguments passed to or from other methods

Value

- invisible reliability object (x)

See Also

- print(), umx::reliability(),

Examples

```
# treat vehicle aspects as items of a test
data(mtcars)
reliability(cov(mtcars))
```

qm *qm*

Description

Quickmatrix function

Usage

```
qm(..., rowMarker = "|")
```

Arguments

...	the components of your matrix
rowMarker	mark the end of each row

Value

- matrix

References

http://www.sumsar.net/blog/2014/03/a-hack-to-create-matrices-in-R-matlab-style

See Also

Other Miscellaneous Utility Functions: install.OpenMx(), umxBrownie(), umxLav2RAM(), umxRAM2Lav(), umxVersion(), umx_array_shift(), umx_find_object(), umx_msg(), umx_open_CRAN_page(), umx_pad(), umx_print(), umx_score_scale(), umx

Examples

```
# simple example
qm(0, 1 |
   2, NA)
## Not run:
# clever example
M1 = M2 = diag(2)
qm(M1,c(4,5) | c(1,2),M2 | t(1:3))

## End(Not run)
```

rad2deg *Convert Radians to Degrees*

Description

Just a helper to multiply radians by 180 and divide by π to get degrees.

note: R's trig functions, e.g. `sin()` use Radians for input! There are $2x$ π radians in a circle.

Usage

```
rad2deg(rad)
```

Arguments

rad	The value in Radians you wish to convert

Value

- value in degrees

See Also

- `deg2rad()`, `sin()`

Other Miscellaneous Functions: `deg2rad()`

Examples

```
rad2deg(pi) #180 degrees
```

reliability *Report coefficient alpha (reliability)*

Description

Compute and report Coefficient alpha (extracted from Rcmdr to avoid its dependencies)

Usage

```
reliability(S)
```

Arguments

S	A square, symmetric, numeric covariance matrix

Value

None

References

- https://cran.r-project.org/package=Rcmdr

See Also

- [umx::print.reliability()],

Other Miscellaneous Stats Helpers: FishersMethod(), SE_from_p(), oddsratio(), umxCov2cor(), umxHetCor(), umxWeightedAIC(), umx_apply(), umx_cor(), umx_means(), umx_r_test(), umx_round(), umx_scale(), umx_var(), umx

Examples

```
# treat car data as items of a test
data(mtcars)
reliability(cov(mtcars))
```

---

residuals.MxModel *Get residuals from an MxModel*

Description

Return the residuals() from an OpenMx RAM model. You can format these (with digits), and suppress small values.

Usage

```
## S3 method for class 'MxModel'
residuals(object, digits = 2, suppress = NULL, reorder = NULL, ...)
```

Arguments

object	An fitted mxModel() from which to get residuals
digits	round to how many digits (default = 2)
suppress	smallest deviation to print out (default = NULL = show all)
reorder	optionally reorder the variables in the residuals matrix to show patterns
...	Optional parameters

Value

- matrix of residuals

References

- https://tbates.github.io, https://github.com/tbates/umx

See Also

Other Reporting functions: RMSEA.MxModel(), RMSEA.summary.mxmodel(), RMSEA(), extractAIC.MxModel(), loadings(), umxCI_boot(), umxCI(), umxCompare(), umxConfint(), umxExpCov(), umxExpMeans(), umxFitIndices(), umxPlotACEv(), umxRotate(), umxSummary.MxModel()

Examples

```
require(umx)
data(demoOneFactor)
manifests = names(demoOneFactor)

m1 = umxRAM("One Factor", data = demoOneFactor, type = "cov",
umxPath("G", to = manifests),
umxPath(var = manifests),
umxPath(var = "G", fixedAt = 1.0)
)

# ==================================
# = Show the residuals of the model =
# ==================================
residuals(m1)
# |   |x1   |x2    |x3   |x4    |x5 |
# |:--|:----|:-----|:----|:-----|:--|
# |x1 |.    |.     |0.01 |.     |.  |
# |x2 |.    |.     |0.01 |-0.01 |.  |
# |x3 |0.01 |0.01  |.    |.     |.  |
# |x4 |.    |-0.01 |.    |.     |.  |
# |x5 |.    |.     |.    |.     |.  |
# [1] "nb: You can zoom in on bad values with, e.g. suppress = .01, which
#      will hide values smaller than this. Use digits = to round"

residuals(m1, digits = 3)
residuals(m1, digits = 3, suppress = .005)
# residuals are returned as an invisible object you can capture in a variable
a = residuals(m1); a
```

RMSEA *Generic RMSEA function*

Description

See RMSEA.MxModel() to access the RMSEA of MxModels

Usage

```
RMSEA(x, ci.lower, ci.upper, digits)
```

Arguments

x	an object from which to get the RMSEA
ci.lower	the lower CI to compute
ci.upper	the upper CI to compute
digits	digits to show

Value

- RMSEA object containing value (and perhaps a CI)

See Also

Other Reporting functions: RMSEA.MxModel(), RMSEA.summary.mxmodel(), extractAIC.MxModel(), loadings(), residuals.MxModel(), umxCI_boot(), umxCI(), umxCompare(), umxConfint(), umxExpCov(), umxExpMeans(), umxFitIndices(), umxPlotACEv(), umxRotate(), umxSummary.MxModel()

RMSEA.MxModel *RMSEA function for MxModels*

Description

Return RMSEA and its confidence interval on a model.

Usage

```
## S3 method for class 'MxModel'
RMSEA(x, ci.lower = 0.05, ci.upper = 0.95, digits = 3)
```

Arguments

x	an mxModel() from which to get RMSEA
ci.lower	the lower CI to compute
ci.upper	the upper CI to compute
digits	digits to show (defaults to 3)

Value

- object containing the RMSEA and lower and upper bounds

References

- https://github.com/tbates/umx, https://github.com/simsem/semTools/wiki/Functions

See Also

Other Reporting functions: RMSEA.summary.mxmodel(), RMSEA(), extractAIC.MxModel(), loadings(), residuals.MxModel(), umxCI_boot(), umxCI(), umxCompare(), umxConfint(), umxExpCov(), umxExpMeans(), umxFitIndices(), umxPlotACEv(), umxRotate(), umxSummary.MxModel()

Examples

```
require(umx)
data(demoOneFactor)
manifests = names(demoOneFactor)

m1 = umxRAM("One Factor", data = demoOneFactor, type = "cov",
umxPath("G", to = manifests),
umxPath(var = manifests),
umxPath(var = "G", fixedAt = 1)
)
RMSEA(m1)
```

RMSEA.summary.mxmodel *RMSEA function for MxModels*

Description

Compute the confidence interval on RMSEA

Usage

```
## S3 method for class 'summary.mxmodel'
RMSEA(x, ci.lower = 0.05, ci.upper = 0.95, digits = 3)
```

Arguments

x	an mxModel() summary from which to get RMSEA
ci.lower	the lower CI to compute
ci.upper	the upper CI to compute
digits	digits to show (defaults to 3)

Value

- object containing the RMSEA and lower and upper bounds

References

- https://github.com/simsem/semTools/wiki/Functions, https://github.com/tbates/umx

See Also

Other Reporting functions: RMSEA.MxModel(), RMSEA(), extractAIC.MxModel(), loadings(), residuals.MxModel(), umxCI_boot(), umxCI(), umxCompare(), umxConfint(), umxExpCov(), umxExpMeans(), umxFitIndices(), umxPlotACEv(), umxRotate(), umxSummary.MxModel()

Examples

```
require(umx)
data(demoOneFactor)
manifests = names(demoOneFactor)

m1 = umxRAM("One Factor", data = demoOneFactor, type = "cov",
umxPath("G", to = manifests),
umxPath(var = manifests),
umxPath(var = "G", fixedAt = 1.0)
)

RMSEA(m1)
```

SE_from_p *Compute an SE from a beta and p value*

Description

SE_from_p takes beta and p, and returns an SE.

Usage

```
SE_from_p(beta = NULL, p = NULL, SE = NULL, lower = NULL, upper = NULL)
```

Arguments

beta	The effect size
p	The p-value for the effect
SE	Standard error
lower	Lower CI
upper	Upper CI

Value

- Standard error

See Also

- umxAPA()

Other Miscellaneous Stats Helpers: FishersMethod(), oddsratio(), reliability(), umxCov2cor(), umxHetCor(), umxWeightedAIC(), umx_apply(), umx_cor(), umx_means(), umx_r_test(), umx_round(), umx_scale(), umx_var(), umx

Examples

```
SE_from_p(beta = .0020, p = .780)
SE_from_p(beta = .0020, p = .01)
SE_from_p(beta = .0020, SE = 0.01)
umxAPA(.0020, p = .01)
```

`tmx_genotypic_effect` *Graphical display of genotypic effects.*

Description

`tmx_genotypic_effect` allows you to explore the concept of genotypic effect at a locus. With it, you can interactively explore the effects of allele frequency, additive variance, and dominance.

This function lets you explore the simplest two–allele system (`B` and `b`), with three possible genotypes, `BB`, `Bb`, and `bb`.

The point between the two homozygotes is `m` – the mean effect of the homozygous genotypes.

Parameter `a` is half the measured phenotypic difference between the homozygotes `BB` and `bb`. It corresponds to the additive effect of each additional `B` allele, relative to the `bb` phenotype.

Parameter `d` is the deviation of the heterozygote `Bb` phenotype from the homozygote mid-point `m`. It corresponds to the non-additive (dominance) effect of the `B` allele. The heterozygote phenotype may lie on either side of `m` and the sign of `d` will vary accordingly.

Old system from book ed 2:

Adapted from Mather and Jinks, 1977, p. 32). See book issue old-style nomenclature https://github.com/tbates/BGBook/issue

`u` = Frequency of the dominant allele (now = `p`). `v` = Frequency of the recessive allele (now = `q`).

`m` = midpoint between the two homozygotes `d` = half the difference between the two homozygote (now `a`)

`h` = deviation of the heterozygote from `m` (now = `d`)

New system:

`u` and `v` -> `p` and `q`

`d` and `h` -> `a` and `d`

See BGBook issue 23

Usage

```
tmx_genotypic_effect(p = 0.75, q = (1 - p), a = 0.5, d = 0, m = 0, show = TRUE)
```

Arguments

`p`	The frequency of the B allele (Default .5)
`q`	The frequency of the b allele (Default 1-p)
`a`	Half the difference between the two homozygote phenotypes (Default .5)
`d`	The deviation of the heterozygote from m (Default 0)
`m`	The value of the midpoint between the homozygotes (Default 0)
`show`	Whether to draw the plot or just return it (Default = TRUE)

Value

- optional plot

References

- Neale, M. C. (2005). Quantitative Genetics. In Encyclopedia of Life Sciences. New York: John Wiley & Sons, Ltd. pdf

See Also

Other Teaching and testing Functions: `tmx_is.identified()`, `umx`

Examples

```
library(umx);


# =========================
# = Pure additivity: d= 0 =
# =========================
tmx_genotypic_effect(p = .5, a = 1, d = 0, m = 0, show = TRUE);

# ============================
# = Complete dominance: a=d=1 =
# ============================
tmx_genotypic_effect(p = .5, q =.5, a = 1, d = 1, m = 0, show = TRUE);

# ==========================
# = Over dominance: a< d =1 =
# ==========================
tmx_genotypic_effect(p = .5, q =.5, a =.5, d = 1, m = 0)

p = tmx_genotypic_effect(p = .5, q = .5, a = 1, d = .5, m = 0, show = TRUE);
# p + ggplot2::geom_point() + ggplot2::geom_text(hjust = 0, nudge_x = 0.05)
# ggsave(paste0(base, "c03_genotypic_effect_by_gene_dose.pdf"), width = 4.6, height = 4.6)
```

---

tmx_is.identified *Test if a factor model is identified*

Description

Test if a factor model is identified by establishing if the number of variables is equal too or grater than the number of model parameters. See also `mxCheckIdentification()` for checking actual models.

Usage

```
tmx_is.identified(nVariables, nFactors)
```

Arguments

nVariables	the number of variables measured.
nFactors	the number of factors posited.

Value

- Binary

References

- https://github.com/tbates/umx, https://tbates.github.io

See Also

- mxCheckIdentification()

Other Teaching and testing Functions: tmx_genotypic_effect(), umx

Examples

```
tmx_is.identified(nVariables = 2, nFactors = 1) # FALSE
tmx_is.identified(nVariables = 3, nFactors = 1) # TRUE
tmx_is.identified(nVariables = 4, nFactors = 2) # FALSE
tmx_is.identified(nVariables = 5, nFactors = 2) # TRUE
```

---

tmx_show *Show matrices of RAM models in a easy-to-learn-from format.*

Description

A great way to learn about models is to look at the matrix contents. tmx_show is designed to do this in a way that makes it easy to process for users: The matrix contents are formatted as tables, and can even be displayed as tables in a web browser.

Usage

```
tmx_show(
  model,
  what = c("values", "free", "labels", "nonzero_or_free"),
  show = c("free", "fixed", "all"),
  matrices = c("S", "A", "M"),
  digits = 2,
  report = c("markdown", "inline", "html", "report"),
  na.print = "",
  zero.print = "."
)
```

Arguments

model	an mxModel() from which to show parameters.
what	legal options are "values" (default), "free", or "labels").
show	filter on what to show c("all", "free", "fixed").
matrices	to show (default is c("S", "A")). "thresholds" in beta.
digits	precision to report. Default = round to 2 decimal places.
report	How to report the results. "html" = open in browser.
na.print	How to display NAs (default = "")
zero.print	How to display 0 values (default = ".")

Details

The user can select which matrices to view, whether to show values, free, and/or labels, and the precision of rounding.

Value

None

References

- https://tbates.github.io

See Also

Other Teaching and Testing functions: umxDiagnose(), umxPower()

Examples

```
require(umx)
data(demoOneFactor)
manifests = names(demoOneFactor)
m1 = umxRAM("tmx_sh", data = demoOneFactor, type = "cov",
umxPath("G", to = manifests),
umxPath(var = manifests),
umxPath(var = "G", fixedAt = 1)
)

tmx_show(m1)
tmx_show(m1, digits = 3)
tmx_show(m1, matrices = "S")
tmx_show(m1, what = "free")
tmx_show(m1, what = "labels")
tmx_show(m1, what = "free", matrices = "A")
# tmx_show(m1, what = "free", matrices = "thresholds")
```

umx *Functions for Structural Equation Modeling in OpenMx*

Description

umx allows you to more easily build, run, modify, and report structural models, building on the OpenMx package. All core functions are organized into families, so they are easier to find (see "families" below under **See Also**)

All the functions have full-featured and well commented examples, some even have *figures*, so use the help, even if you think it won't help :-) Have a look, for example at umxRAM()

Check out NEWS about new features at news(package = "umx")

Details

Introductory working examples are below. You can run all demos with demo(umx) When I have a vignette, it will be: vignette("umx", package = "umx")

There is a helpful blog at https://tbates.github.io

(Only) if you want the bleeding-edge version:

devtools::install_github("tbates/umx")

References

- https://www.github.com/tbates/umx

See Also

Other Teaching and testing Functions: tmx_genotypic_effect(), tmx_is.identified()

Other Core Modeling Functions: umxAlgebra(), umxMatrix(), umxModify(), umxPath(), umxRAM(), umxRun(), umxSummary(), umxSuperModel()

Other Reporting Functions: loadings.MxModel(), umxAPA(), umxFactorScores(), umxGetParameters(), umxParameters(), umxReduce(), umx_aggregate(), umx_names(), umx_time()

Other Modify or Compare Models: umxEquate(), umxFixAll(), umxMI(), umxModify(), umxSetParameters(), umxUnexplainedCausalNexus()

Other Plotting functions: plot.MxLISRELModel(), plot.MxModel(), umxPlotACEcov(), umxPlotACEv(), umxPlotACE(), umxPlotCP(), umxPlotGxEbiv(), umxPlotGxE(), umxPlotIP(), umxPlotSexLim(), umxPlotSimplex()

Other Super-easy helpers: umxEFA(), umxMendelianRandomization()

Other Twin Modeling Functions: plot.MxModelTwinMaker(), power.ACE.test(), umxACEcov(), umxACEv(), umxACE(), umxCP(), umxDoCp(), umxDoC(), umxGxE_window(), umxGxEbiv(), umxGxE(), umxIP(), umxRotate.MxModelCP(), umxSexLim(), umxSimplex()

Other Twin Reporting Functions: umxPlotCP(), umxPlotDoC(), umxReduceACE(), umxReduceGxE(), umxReduce(), umxSummarizeTwinData(), umxSummaryACEcov(), umxSummaryACEv(), umxSummaryACE(),

umxSummaryCP(), umxSummaryDoC(), umxSummaryGxEbiv(), umxSummaryGxE(), umxSummaryIP(), umxSummarySexLim(), umxSummarySimplex()

Other Twin Data functions: umx_long2wide(), umx_make_TwinData(), umx_make_twin_data_nice(), umx_residualize(), umx_scale_wide_twin_data(), umx_wide2long()

Other Get and set: umx_get_checkpoint(), umx_get_options(), umx_set_auto_plot(), umx_set_auto_run(), umx_set_checkpoint(), umx_set_condensed_slots(), umx_set_cores(), umx_set_data_variance_check(), umx_set_mvn_optimization_options(), umx_set_optimizer(), umx_set_plot_file_suffix(), umx_set_plot_format(), umx_set_separator(), umx_set_silent(), umx_set_table_format()

Other Check or test: umx_check_names(), umx_is_class(), umx_is_endogenous(), umx_is_exogenous(), umx_is_numeric(), umx_is_ordered()

Other Data Functions: umxFactor(), umxHetCor(), umx_as_numeric(), umx_cont_2_quantiles(), umx_lower2full(), umx_make_MR_data(), umx_make_TwinData(), umx_make_fake_data(), umx_make_raw_from_cov(), umx_polychoric(), umx_polypairwise(), umx_polytriowise(), umx_read_lower(), umx_rename(), umx_reorder(), umx_select_valid(), umx_stack()

Other File Functions: dl_from_dropbox(), umx_file_load_pseudo(), umx_make_sql_from_excel(), umx_move_file(), umx_open(), umx_rename_file(), umx_write_to_clipboard()

Other String Functions: umx_explode_twin_names(), umx_explode(), umx_grep(), umx_names(), umx_paste_names(), umx_rot(), umx_str_chars(), umx_str_from_object(), umx_trim()

Other Miscellaneous Stats Helpers: FishersMethod(), SE_from_p(), oddsratio(), reliability(), umxCov2cor(), umxHetCor(), umxWeightedAIC(), umx_apply(), umx_cor(), umx_means(), umx_r_test(), umx_round(), umx_scale(), umx_var()

Other Miscellaneous Utility Functions: install.OpenMx(), qm(), umxBrownie(), umxLav2RAM(), umxRAM2Lav(), umxVersion(), umx_array_shift(), umx_find_object(), umx_msg(), umx_open_CRAN_page(), umx_pad(), umx_print(), umx_score_scale()

Other datasets: Fischbein_wt, GFF, docData, iqdat, us_skinfold_data

Other Advanced Model Building Functions: umxJiggle(), umxLabel(), umxThresholdMatrix(), umxValues()

Other xmu internal not for end user: umxModel(), umxRenameMatrix(), umxTwinMaker(), umx_APA_pval(), umx_fun_mean_sd(), umx_get_bracket_addresses(), umx_make(), umx_standardize(), umx_string_to_algebra(), xmuHasSquareBrackets(), xmuLabel_MATRIX_Model(), xmuLabel_Matrix(), xmuLabel_RAM_Model(), xmuMI(), xmuMakeDeviationThresholdsMatrices(), xmuMakeOneHeadedPathsFromPathList(), xmuMakeTwoHeadedPathsFromPathList(), xmuMaxLevels(), xmuMinLevels(), xmuPropagateLabels(), xmuRAM2Ordinal(), xmuTwinSuper_Continuous(), xmuTwinUpgradeMeansToCovariateModel(), xmu_CI_merge(), xmu_CI_stash(), xmu_DF_to_mxData_TypeCov(), xmu_PadAndPruneForDefVars(), xmu_cell_is_on(), xmu_check_levels_identical(), xmu_check_needs_means(), xmu_check_variance(), xmu_clean_label(), xmu_data_missing(), xmu_data_swap_a_block(), xmu_describe_data_WLS(), xmu_dot_make_paths(), xmu_dot_make_residuals(), xmu_dot_maker(), xmu_dot_move_ranks(), xmu_dot_rank_str(), xmu_extract_column(), xmu_get_CI(), xmu_lavaan_process_group(), xmu_make_TwinSuperModel(), xmu_make_bin_cont_pair_data(), xmu_make_mxData(), xmu_match.arg(), xmu_name_from_lavaan_str(), xmu_path2twin(), xmu_path_regex(), xmu_safe_run_summary(), xmu_set_sep_from_suffix(), xmu_show_fit_or_comparison(), xmu_simplex_corner(), xmu_standardize_ACEcov(), xmu_standardize_ACEv(), xmu_standardize_ACE(), xmu_standardize_CP(), xmu_standardize_IP(), xmu_standardize_RAM(), xmu_standardize_SexLim(), xmu_standardize_Simplex(), xmu_start_value_list(), xmu_starts(), xmu_twin_add_WeightMatrices(), xmu_twin_check(), xmu_twin_get_var_names(), xmu_twin_upgrade_selDvs2SelVars()

Examples

```
require("umx")
data(demoOneFactor)
manifests = names(demoOneFactor)
m1 = umxRAM("One Factor", data = demoOneFactor, type="cov",
umxPath("G", to = manifests),
umxPath(var = manifests),
umxPath(var = "G"  , fixedAt= 1)
)

# umx added informative labels, created starting values,
# Ran you model (if autoRun is on), and displayed a brief summary
# including a comparison if you modified a model...!

# Let's get some journal-ready fit information for standardized parameters

umxSummary(m1, std = TRUE)
# You can get the coefficients of an MxModel with coef(), just like for lm etc.
coef(m1)

# But with more control using "parameters", for example just the G loadings
# above .3, rounded to 2-digits.
parameters(m1, thresh="above", b=.3, pattern = "G_to.*", digits = 2)

# ==================
# = Model updating =
# ==================
# We'll use umxModify to modify the model...
# Can we set the loading of x1 on G to zero? (nope...)
m2 = umxModify(m1, "G_to_x1", name = "no_effect_of_g_on_X1", comparison = TRUE)

# note1: umxSetParameters can do this with some additional flexibility
# note2 "comparison = TRUE" above is the same as calling
# umxCompare, like this
umxCompare(m1, m2)


# ========================
# = Confidence intervals =
# ========================

# umxSummary() will show these, but you can also use the confint() function
confint(m1) # OpenMx's SE-based confidence intervals


## Not run:
# umxConfint formats everything you need nicely, and allows adding CIs (with parm=)
umxConfint(m1, parm = 'all', run = TRUE) # likelihood-based CIs

# And make a Figure and open in browser
plot(m1, std = TRUE)
```

```
# If you just want the .dot code returned set file = NA
plot(m1, std = TRUE, file = NA)

## End(Not run)
```

---

umx-deprecated *Deprecated. May already stop() code and ask to be updated. May be dropped entirely in future.*

Description

xmuMakeThresholdsMatrices should be replaced with umxThresholdMatrix()

umxTryHard is deprecated: use umxRun() instead

stringToMxAlgebra is deprecated: please use umx_string_to_algebra() instead

genEpi_EvalQuote is deprecated: please use mxEvalByName() instead

umxReportCIs is deprecated: please use umxCI() instead

replace umxReportFit with umxSummary()

Replace umxGraph_RAM with plot()

Replace tryHard with mxTryHard()

Replace standardizeRAM with umx_standardize()

Arguments

...	the old function's parameters (now stripped out to avoid telling people how to do it the wrong way :-)

References

- https://tbates.github.io, https://github.com/tbates/umx

---

umxACE *Build and run a 2-group Cholesky twin model (uni-variate or multi-variate)*

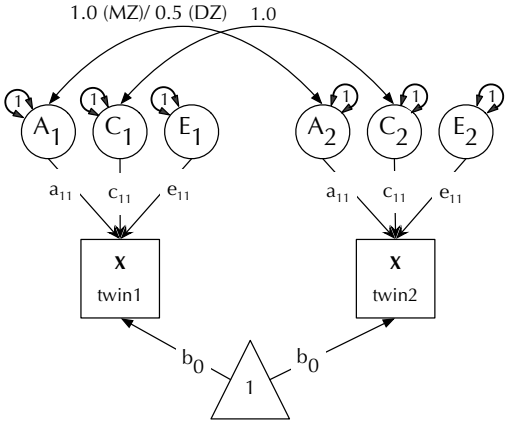
Description

Implementing a core task in twin modeling, umxACE models the genetic and environmental structure of one or more phenotypes (measured variables) using the Cholesky ACE model (Neale and Cardon, 1996).

Classical twin modeling uses the genetic and environmental differences among pairs of monozygotic (MZ) and di-zygotic (DZ) twins reared together.

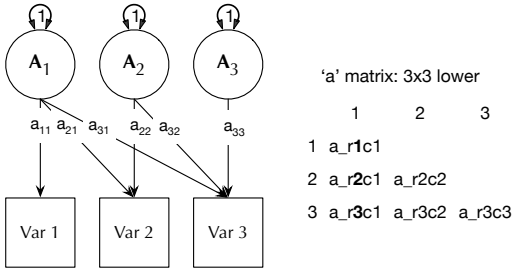
umxACE implements a 2-group model to capture these data and represent the phenotypic variance as a sum of Additive genetic, unique environmental (E) and, optionally, either common or shared-environment (C) or non-additive genetic effects (D).

The following figure shows how the ACE model appears as a path diagram (for one variable):



umxACE allows multivariate analyses, and this brings us to the Cholesky part of the model.

This model creates as many latent A C and E variables as there are phenotypes, and, moving from left to right, decomposes the variance in each manifest into successively restricted factors. The following figure shows how the ACE model appears as a path diagram:



In this model, the variance-covariance matrix of the raw data is recovered as the product of the lower Cholesky and its transform.

This Cholesky or lower-triangle decomposition allows a model which is both sure to be solvable, and also to account for all the variance (with some restrictions) in the data.

This figure also contains the key to understanding how to modify models that umxACE produces. read the "Matrices and Labels in ACE model" section in details below...

NOTE: Scroll down to details for how to use the function, a figure and multiple examples.

Usage

```
umxACE(
  name = "ACE",
  selDVs,
  selCovs = NULL,
  dzData = NULL,
  mzData = NULL,
  sep = NULL,
  data = NULL,
  zyg = "zygosity",
  type = c("Auto", "FIML", "cov", "cor", "WLS", "DWLS", "ULS"),
  numObsDZ = NULL,
  numObsMZ = NULL,
  boundDiag = 0,
  allContinuousMethod = c("cumulants", "marginals"),
  autoRun = getOption("umx_auto_run"),
  intervals = FALSE,
  tryHard = c("no", "yes", "ordinal", "search"),
  optimizer = NULL,
  residualizeContinuousVars = FALSE,
  dzAr = 0.5,
  dzCr = 1,
  weightVar = NULL,
  equateMeans = TRUE,
  addStd = TRUE,
  addCI = TRUE
)
```

Arguments

name	The name of the model (defaults to"ACE").
selDVs	The variables to include from the data: preferably, just "dep" not c("dep_T1", "dep_T2").
selCovs	(optional) covariates to include from the data (do not include sep in names)
dzData	The DZ dataframe.
mzData	The MZ dataframe.
sep	The separator in twin variable names, often "_T", e.g. "dep_T1". Simplifies selDVs.
data	If provided, dzData and mzData are treated as levels of zyg to select() MZ and DZ data sets (default = NULL)
zyg	If data provided, this column is used to select rows by zygosity (Default = "zygosity")
type	Analysis method one of c("Auto", "FIML", "cov", "cor", "WLS", "DWLS", "ULS")
numObsDZ	Number of DZ twins: Set this if you input covariance data.

numObsMZ	Number of MZ twins: Set this if you input covariance data.
boundDiag	Numeric lbound for diagonal of the a, c, and e matrices. Defaults to 0 since umx version 1.8
allContinuousMethod	"cumulants" or "marginals". Used in all-continuous WLS data to determine if a means model needed.
autoRun	Whether to run the model (default), or just to create it and return without running.
intervals	Whether to run mxCI confidence intervals (default = FALSE)
tryHard	Default ('no') uses normal mxRun. "yes" uses mxTryHard. Other options: "ordinal", "search"
optimizer	Optionally set the optimizer (default NULL does nothing).
residualizeContinuousVars	Not yet implemented.
dzAr	The DZ genetic correlation (defaults to .5, vary to examine assortative mating).
dzCr	The DZ "C" correlation (defaults to 1: set to .25 to make an ADE model).
weightVar	If provided, a vector objective will be used to weight the data. (default = NULL).
equateMeans	Whether to equate the means across twins (defaults to TRUE).
addStd	Whether to add the algebras to compute a std model (defaults to TRUE).
addCI	Whether to add intervals to compute CIs (defaults to TRUE).

Details

Covariates umxACE handles covariates by modelling them in the means. On the plus side, there is no distributional assumption for this method. A downside of this approach is that all covariates must be non-NA, thus dropping any rows where one or more covariates are missing. This can waste data. See also umx_residualize().

Data Input The function flexibly accepts raw data, and also summary covariance data (in which case the user must also supple numbers of observations for the two input data sets).

The type parameter can select how you want the model data treated. "FIML" is the normal treatment. "cov" and "cor" will turn raw data into cor data for analysis, or check that you've provided cor data as input.

Types "WLS", "DWLS", and "ULS" will process raw data into WLS data of these types.

The default, "Auto" will treat data as the type they are provided as.

Ordinal Data In an important capability, the model transparently handles ordinal (binary or multilevel ordered factor data) inputs, and can handle mixtures of continuous, binary, and ordinal data in any combination. An experimental feature is under development to allow Tobit modeling.

The function also supports weighting of individual data rows. In this case, the model is estimated for each row individually, then each row likelihood is multiplied by its weight, and these weighted likelihoods summed to form the model-likelihood, which is to be minimized. This feature is used in the non-linear GxE model functions.

Additional features The umxACE function supports varying the DZ genetic association (defaulting to .5) to allow exploring assortative mating effects, as well as varying the DZ "C" factor from 1 (the

default for modeling family-level effects shared 100% by twins in a pair), to .25 to model dominance effects.

Matrices and Labels in ACE model

Matrices 'a', 'c', and 'e' contain the path loadings of the Cholesky ACE factor model.

So, labels relevant to modifying the model are of the form "a_r1c1", "c_r1c1" etc.

Variables are in rows, and factors are in columns. So to drop the influence of factor 2 on variable 3, you would say:

`m2 = umxModify(m1, update = "c_r3c2")`

Less commonly-modified matrices are the mean matrix `expMean`. This has 1 row, and the columns are laid out for each variable for twin 1, followed by each variable for twin 2.

So, in a model where the means for twin 1 and twin 2 had been equated (set = to T1), you could make them independent again with this script:

`m1$top$expMean$labels[1,4:6] = c("expMean_r1c4","expMean_r1c5","expMean_r1c6")`

note: Only one of C or D may be estimated simultaneously. This restriction reflects the lack of degrees of freedom to simultaneously model C and D with only MZ and DZ twin pairs (Eaves et al. 1978, p267).

Value

- `mxModel()` of subclass mxModel.ACE

References

- Eaves, L. J., Last, K. A., Young, P. A., & Martin, N. G. (1978). Model-fitting approaches to the analysis of human behaviour. *Heredity*, **41**, 249-320. https://www.nature.com/articles/hdy1978101.pdf

See Also

- `umxPlotACE()`, `umxSummaryACE()`, `umxModify()`

Other Twin Modeling Functions: `plot.MxModelTwinMaker()`, `power.ACE.test()`, `umxACEcov()`, `umxACEv()`, `umxCP()`, `umxDoCp()`, `umxDoC()`, `umxGxE_window()`, `umxGxEbiv()`, `umxGxE()`, `umxIP()`, `umxRotate.MxModelCP()`, `umxSexLim()`, `umxSimplex()`, `umx`

Examples

```
require(umx)
# ============================
# = How heritable is height? =
# ============================

# 1. Height in metres has a tiny variance, and this makes optimising hard.
#    We'll scale it by 10x to make the Optimizer's task easier.
data(twinData) # ?twinData from Australian twins.
twinData[, c("ht1", "ht2")] = twinData[, c("ht1", "ht2")] * 10

# 2. Make mz & dz data.frames (no need to select variables: umx will do this)
```

```
mzData = twinData[twinData$zygosity %in% "MZFF", ]
dzData = twinData[twinData$zygosity %in% "DZFF", ]



# 3. Built & run the model, controlling for age in the means model
m1 = umxACE(selDVs = "ht", selCovs = "age", sep = "", dzData = dzData, mzData = mzData)

# sidebar: umxACE figures out variable names using sep:
#    e.g. selVars = "wt" + sep= "_T" -> "wt_T1" "wt_T2"

umxSummary(m1, std = FALSE) # un-standardized

# tip 1: set report = "html" and umxSummary prints the table to your browser!
# tip 2: plot works for umx: Get a figure of the model and parameters
# plot(m1) # Also, look at the options for ?plot.MxModel.


# ============================
# = Model, with 2 covariates =
# ============================

# Create another covariate: cohort
twinData$cohort1 = twinData$cohort2 =twinData$part
mzData = twinData[twinData$zygosity %in% "MZFF", ]
dzData = twinData[twinData$zygosity %in% "DZFF", ]

# 1. def var approach
m2 = umxACE(selDVs = "ht", selCovs = c("age", "cohort"), sep = "", dzData = dzData, mzData = mzData)

# 2. Residualized approach: remove height variance accounted-for by age.
FFdata = twinData[twinData$zygosity %in% c("MZFF", "DZFF"), ]
FFdata = umx_residualize("ht", "age", suffixes = 1:2, data = FFdata)
mzData = FFdata[FFdata$zygosity %in% "MZFF", ]
dzData = FFdata[FFdata$zygosity %in% "DZFF", ]
m3 = umxACE(selDVs = "ht", sep = "", dzData = dzData, mzData = mzData)

# ============================================================
# = ADE: Evidence for dominance ? (DZ correlation set to .25) =
# ============================================================
m2 = umxACE(selDVs = "ht", sep = "", dzData = dzData, mzData = mzData, dzCr = .25)
umxCompare(m2, m1) # ADE is better
umxSummary(m2, comparison = m1)
# nb: Although summary is smart enough to print d, the underlying
#     matrices are still called a, c & e.

# tip: try umxReduce(m1) to automatically build and compare ACE, ADE, AE, CE
# including conditional probabilities!

# ================================================
# = WLS example using diagonal weight least squares =
# ================================================
```

```
m3 = umxACE(selDVs = "ht", sep = "", dzData = dzData, mzData = mzData,
type = "DWLS", allContinuousMethod='marginals'
)


# ==============================
# = Univariate model of weight =
# ==============================

# Things to note:

# 1. Weight has a large variance, and this makes solution finding very hard.
# Here, we residualize the data for age, which also scales weight and height.

data(twinData)
tmp = umx_residualize(c("wt", "ht"), cov = "age", suffixes= c(1, 2), data = twinData)
mzData = tmp[tmp$zygosity %in% "MZFF", ]
dzData = tmp[tmp$zygosity %in% "DZFF", ]

# tip: You might also want transform variables
# tmp = twinData$wt1[!is.na(twinData$wt1)]
# car::powerTransform(tmp, family="bcPower"); hist(tmp^-0.6848438)
# twinData$wt1 = twinData$wt1^-0.6848438
# twinData$wt2 = twinData$wt2^-0.6848438

# 4. note: the default boundDiag = 0 lower-bounds a, c, and e at 0.
#    Prevents mirror-solutions. If not desired: set boundDiag = NULL.

m2 = umxACE(selDVs = "wt", dzData = dzData, mzData = mzData, sep = "", boundDiag = NULL)

# A short cut (which is even shorter for "_T" twin data with "MZ"/"DZ" data in zygosity column is:
m1 = umxACE(selDVs = "wt", sep = "", data = twinData,
dzData = c("DZMM", "DZFF", "DZOS"), mzData = c("MZMM", "MZFF"))
# |   |   a1|c1 |   e1|
# |:--|----:|:--|----:|
# |wt | 0.93|.  | 0.38|

# tip: umx_make_twin_data_nice() will make data into this nice format for you!

# ======================
# = MODEL MODIFICATION =
# ======================
# We can modify this model, e.g. test shared environment.
# Set comparison to modify, and show effect in one step.

m2 = umxModify(m1, update = "c_r1c1", name = "no_C", comparison = TRUE)
#*tip* call umxModify(m1) with no parameters, and it will print the labels available to fix!
# nb: You can see parameters of any model with parameters(m1)

# ========================================================
# = Well done! Now you can make modify twin models in umx =
# ========================================================
```

```
# ====================================
# = Bivariate height and weight model =
# ====================================
data(twinData)
# We'll scale height (ht1 and ht2) and weight
twinData = umx_scale_wide_twin_data(data = twinData, varsToScale = c("ht", "wt"), sep = "")
mzData = twinData[twinData$zygosity %in% c("MZFF", "MZMM"),]
dzData = twinData[twinData$zygosity %in% c("DZFF", "DZMM", "DZOS"), ]
m1 = umxACE(selDVs = c("ht", "wt"), sep = '', dzData = dzData, mzData = mzData)
umxSummary(m1)

# ==================
# = Ordinal example =
# ==================
require(umx)
data(twinData)
twinData= umx_scale_wide_twin_data(data=twinData,varsToScale=c("wt"),sep="")
# Cut BMI column to form ordinal obesity variables
obLevels = c('normal', 'overweight', 'obese')
cuts = quantile(twinData[, "bmi1"], probs = c(.5, .2), na.rm = TRUE)
twinData$obese1=cut(twinData$bmi1, breaks=c(-Inf,cuts,Inf), labels=obLevels)
twinData$obese2=cut(twinData$bmi2, breaks=c(-Inf,cuts,Inf), labels=obLevels)
# Make the ordinal variables into umxFactors
ordDVs = c("obese1", "obese2")
twinData[, ordDVs] = umxFactor(twinData[, ordDVs])
mzData = twinData[twinData$zygosity %in% "MZFF", ]
dzData = twinData[twinData$zygosity %in% "DZFF", ]
str(mzData) # make sure mz, dz, and t1 and t2 have the same levels!

# Data-prep done - here's the model and summary!
m1 = umxACE(selDVs = "obese", dzData = dzData, mzData = mzData, sep = '')

# And controlling age (otherwise manifests appearance as latent C)
m1 = umxACE(selDVs = "obese", selCov= "age", dzData = dzData, mzData = mzData, sep = '')
# umxSummary(m1)

# ============================================
# = Bivariate continuous and ordinal example =
# ============================================
data(twinData)
twinData= umx_scale_wide_twin_data(data=twinData,varsToScale="wt",sep= "")
# Cut BMI column to form ordinal obesity variables
obLevels   = c('normal', 'overweight', 'obese')
cuts       = quantile(twinData[, "bmi1"], probs = c(.5, .2), na.rm = TRUE)
twinData$obese1=cut(twinData$bmi1,breaks=c(-Inf,cuts,Inf),labels=obLevels)
twinData$obese2=cut(twinData$bmi2,breaks=c(-Inf,cuts,Inf),labels=obLevels)
# Make the ordinal variables into mxFactors
ordDVs = c("obese1", "obese2")
twinData[, ordDVs] = umxFactor(twinData[, ordDVs])
mzData = twinData[twinData$zygosity %in% "MZFF",]
dzData = twinData[twinData$zygosity %in% "DZFF",]
mzData = mzData[1:80,] # just top 80 so example runs in a couple of secs
dzData = dzData[1:80,]
```

```
m1 = umxACE(selDVs= c("wt","obese"), dzData= dzData, mzData= mzData, sep='')

# And controlling age
m1 = umxACE(selDVs = c("wt","obese"), selCov= "age", dzData = dzData, mzData = mzData, sep = '')

# ======================================
# = Mixed continuous and binary example =
# ======================================
require(umx)
data(twinData)
twinData= umx_scale_wide_twin_data(data= twinData,varsToScale= "wt", sep="")
# Cut to form category of 20% obese subjects
# and make into mxFactors (ensure ordered is TRUE, and require levels)
obLevels   = c('normal', 'obese')
cuts       = quantile(twinData[, "bmi1"], probs = .2, na.rm = TRUE)
twinData$obese1= cut(twinData$bmi1, breaks=c(-Inf,cuts,Inf), labels=obLevels)
twinData$obese2= cut(twinData$bmi2, breaks=c(-Inf,cuts,Inf), labels=obLevels)
ordDVs = c("obese1", "obese2")
twinData[, ordDVs] = umxFactor(twinData[, ordDVs])

selDVs = c("wt", "obese")
mzData = twinData[twinData$zygosity %in% "MZFF",]
dzData = twinData[twinData$zygosity %in% "DZFF",]
m1 = umxACE(selDVs = selDVs, dzData = dzData, mzData = mzData, sep = '')
umxSummary(m1)

# ==================================
# Example with covariance data only =
# ==================================

require(umx)
data(twinData)
twinData= umx_scale_wide_twin_data(data=twinData, varsToScale= "wt", sep="")
selDVs = c("wt1", "wt2")
mz = cov(twinData[twinData$zygosity %in%  "MZFF", selDVs], use = "complete")
dz = cov(twinData[twinData$zygosity %in%  "DZFF", selDVs], use = "complete")
m1 = umxACE(selDVs=selDVs, dzData=dz, mzData=mz, numObsDZ=569, numObsMZ=351)
umxSummary(m1)
plot(m1)
```

---

umxACEcov *Run a Cholesky with covariates that are random (in the expected covariance matrix)*

Description

Often, researchers include covariates in 2-group Cholesky `umxACE()` twin models. The umxACEcov 'random' option models the covariates in the expected covariance matrix, thus allowing all data

to be preserved. The downside is that this method has a strong assumption of multivariate normality. Covariates like age, which are perfectly correlated in twins cannot be used. Covariates like sex, which are ordinal, violate the normality assumption.

Usage

```
umxACEcov(
  name = "ACEcov",
  selDVs,
  selCovs,
  dzData,
  mzData,
  sep = NULL,
  type = c("Auto", "FIML", "cov", "cor", "WLS", "DWLS", "ULS"),
  allContinuousMethod = c("cumulants", "marginals"),
  dzAr = 0.5,
  dzCr = 1,
  addStd = TRUE,
  addCI = TRUE,
  boundDiag = 0,
  equateMeans = TRUE,
  bVector = FALSE,
  autoRun = getOption("umx_auto_run"),
  tryHard = c("no", "yes", "ordinal", "search"),
  optimizer = NULL
)
```

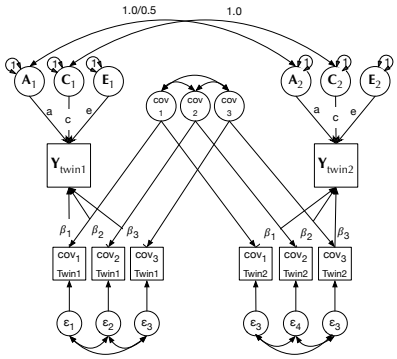
Arguments

name	The name of the model (defaults to"ACE").
selDVs	The variables to include from the data (do not include sep).
selCovs	The covariates to include from the data (do not include sep).
dzData	The DZ dataframe.
mzData	The MZ dataframe.
sep	Separator text between basename for twin variable names. Often "_T". Used to expand selDVs into full column names, i.e., "dep" –> c("dep_T1", "dep_T2").
type	Analysis method one of c("Auto", "FIML", "cov", "cor", "WLS", "DWLS", "ULS")
allContinuousMethod	"cumulants" or "marginals". Used in all-continuous WLS data to determine if a means model needed.
dzAr	The DZ genetic correlation (defaults to .5, vary to examine assortative mating).
dzCr	The DZ "C" correlation (defaults to 1: set to .25 to make an ADE model).
addStd	Whether to add the algebras to compute a std model (defaults to TRUE).
addCI	Whether to add intervals to compute CIs (defaults to TRUE).

boundDiag	= Whether to bound the diagonal of the a, c, and e matrices.
equateMeans	Whether to equate the means across twins (defaults to TRUE).
bVector	Whether to compute row-wise likelihoods (defaults to FALSE).
autoRun	Whether to run the model (default), or just to create it and return without running.
tryHard	Default ('no') uses normal mxRun. "yes" uses mxTryHard. Other options: "ordinal", "search"
optimizer	optionally set the optimizer. Default (NULL) does nothing.

Details

The following figure shows how the ACE model with random covariates appears as a path diagram:



Value

- mxModel() of subclass mxModel.ACEcov

References

Neale, M. C., & Martin, N. G. (1989). The effects of age, sex, and genotype on self-report drunkenness following a challenge dose of alcohol. *Behavior Genetics*, **19**, 63-78. doi:https://doi.org/10.1007/BF01065884.

Schwabe, I., Boomsma, D. I., Zeeuw, E. L., & Berg, S. M. (2015). A New Approach to Handle Missing Covariate Data in Twin Research : With an Application to Educational Achievement Data. *Behavior Genetics*, **46**, 583-95. doi:https://doi.org/10.1007/s10519-015-9771-1.

See Also

Other Twin Modeling Functions: plot.MxModelTwinMaker(), power.ACE.test(), umxACEv(), umxACE(), umxCP(), umxDoCp(), umxDoC(), umxGxE_window(), umxGxEbiv(), umxGxE(), umxIP(), umxRotate.MxModelCP(), umxSexLim(), umxSimplex(), umx

Examples

```
## Not run:
# ========================================
# = BMI, can't use Age as a random covariate =
# ========================================
require(umx)
data(twinData)
# Replicate age to age1 & age2
twinData$age1 = twinData$age2 = twinData$age
mzData = subset(twinData, zygosity == "MZFF")
dzData = subset(twinData, zygosity == "DZFF")

# ============================================================================
# = Trying to use identical var (like age) as a random cov is ILLEGAL =
# ============================================================================
m1 = umxACEcov(selDVs = "bmi", selCovs = "age", dzData = dzData, mzData = mzData, sep = "")

# ========================================================
# = Use an lm-based age-residualisation approach instead =
# ========================================================

resid_data = umx_residualize("bmi", "age", suffixes = 1:2, twinData)
mzData = subset(resid_data, zygosity == "MZFF")
dzData = subset(resid_data, zygosity == "DZFF")
m2     = umxACE("resid", selDVs = "bmi", dzData = dzData, mzData = mzData, sep = "")

# Univariate BMI without covariate of age for comparison
mzData = subset(twinData, zygosity == "MZFF")
dzData = subset(twinData, zygosity == "DZFF")
m3 = umxACE("raw_bmi", selDVs = "bmi", dzData = dzData, mzData = mzData, sep = "")

# ============================================================================
# = A bivariate example (need a dataset with a VIABLE COVARIATE to do this) =
# ============================================================================
selDVs  = c("ht", "wt") # Set the DV
selCovs = c("income") # Set the COV
selVars = umx_paste_names(selDVs, covNames = selCovs, sep = "", sep = 1:2)
# 80 rows so example runs fast on CRAN
mzData = subset(twinData, zygosity == "MZFF", selVars)[1:80, ]
dzData = subset(twinData, zygosity == "DZFF", selVars)[1:80, ]
m1 = umxACEcov(selDVs = selDVs, selCovs = selCovs,
   dzData = dzData, mzData = mzData, sep = "", autoRun = TRUE
)

## End(Not run)
```

---

umxACEv *Build and run 2-group uni- or multi-variate ACE models based on VARIANCE (not paths).*

Description

A common task in twin modeling involves using the genetic and environmental differences between large numbers of pairs of mono-zygotic (MZ) and di-zygotic (DZ) twins reared together to model the genetic and environmental structure of one, or, typically, several phenotypes (measured behaviors).

Usage

```
umxACEv(
  name = "ACEv",
  selDVs,
  selCovs = NULL,
  sep = NULL,
  dzData,
  mzData,
  dzAr = 0.5,
  dzCr = 1,
  type = c("Auto", "FIML", "cov", "cor", "WLS", "DWLS", "ULS"),
  allContinuousMethod = c("cumulants", "marginals"),
  data = NULL,
  zyg = "zygosity",
  weightVar = NULL,
  numObsDZ = NULL,
  numObsMZ = NULL,
  addStd = TRUE,
  addCI = TRUE,
  boundDiag = NULL,
  equateMeans = TRUE,
  bVector = FALSE,
  covMethod = c("fixed", "random"),
  autoRun = getOption("umx_auto_run"),
  tryHard = c("no", "yes", "ordinal", "search"),
  optimizer = NULL
)
```

Arguments

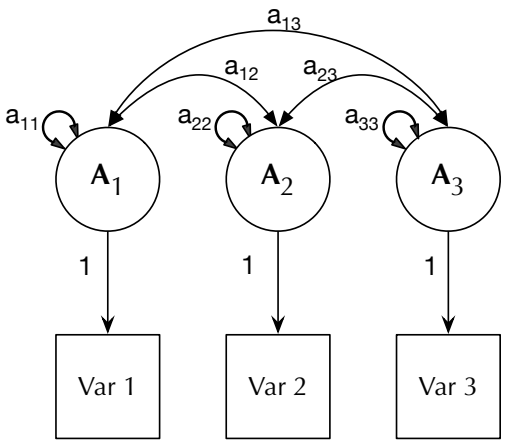
name	The name of the model (defaults to"ACE").
selDVs	The variables to include from the data: preferably, just "dep" not c("dep_T1", "dep_T2").
selCovs	(optional) covariates to include from the data (do not include sep in names)
sep	The separator in twin var names, often "_T" in vars like "dep_T1". Simplifies selDVs.
dzData	The DZ dataframe.
mzData	The MZ dataframe.
dzAr	The DZ genetic correlation (defaults to .5, vary to examine assortative mating).

dzCr	The DZ "C" correlation (defaults to 1: set to .25 to make an ADE model).
type	Analysis method one of c("Auto", "FIML", "cov", "cor", "WLS", "DWLS", "ULS").
allContinuousMethod	"cumulants" or "marginals". Used in all-continuous WLS data to determine if a means model needed.
data	If provided, dzData and mzData are treated as valid levels of zyg to select() data sets (default = NULL)
zyg	If data provided, this column is used to select rows by zygosity (Default = "zygosity")
weightVar	= If provided, a vector objective will be used to weight the data. (default = NULL).
numObsDZ	= Number of DZ twins: Set this if you input covariance data.
numObsMZ	= Number of MZ twins: Set this if you input covariance data.
addStd	Whether to add the algebras to compute a std model (defaults to TRUE).
addCI	Whether to add intervals to compute CIs (defaults to TRUE).
boundDiag	= Numeric lbound for diagonal of the a, c, and e matrices. Default = NULL (no bound)
equateMeans	Whether to equate the means across twins (defaults to TRUE).
bVector	Whether to compute row-wise likelihoods (defaults to FALSE).
covMethod	How to treat covariates: "fixed" (default) or "random".
autoRun	Whether to run the model (default), or just to create it and return without running.
tryHard	Default ('no') uses normal mxRun. "yes" uses mxTryHard. Other options: "ordinal", "search"
optimizer	Optionally set the optimizer (default NULL does nothing).

Details

The ACE variance-based model decomposes phenotypic variance into additive genetic (A), unique environmental (E) and, optionally, either common environment (shared-environment, C) or non-additive genetic effects (D). Scroll down to details for how to use the function, a figure and multiple examples.

The following figure shows the A components of a trivariate ACEv model:



NOTE: This function does not use the Cholesky decomposition. Instead it directly models variance. This ensures unbiased type-I error rates. It means that occasionally estimates of variance may be negative. This should be used as an occasion to inspect you model choices and data. `umxACEv` can be used as a base model to validate the ACE Cholesky model, a core model in behavior genetics (Neale and Cardon, 1992).

Data Input The function flexibly accepts raw data, and also summary covariance data (in which case the user must also supple numbers of observations for the two input data sets).

Ordinal Data In an important capability, the model transparently handles ordinal (binary or multi-level ordered factor data) inputs, and can handle mixtures of continuous, binary, and ordinal data in any combination.

The function also supports weighting of individual data rows. In this case, the model is estimated for each row individually, then each row likelihood is multiplied by its weight, and these weighted likelihoods summed to form the model-likelihood, which is to be minimized. This feature is used in the non-linear GxE model functions.

Additional features The umxACEv function supports varying the DZ genetic association (defaulting to .5) to allow exploring assortative mating effects, as well as varying the DZ "C" factor from 1 (the default for modeling family-level effects shared 100% by twins in a pair), to .25 to model dominance effects.

note: Only one of C or D may be estimated simultaneously. This restriction reflects the lack of degrees of freedom to simultaneously model C and D with only MZ and DZ twin pairs (Eaves et al. 1978 p267).

Value

- `mxModel()` subclass `mxModel.ACE`

References

- Eaves, L. J., Last, K. A., Young, P. A., & Martin, N. G. (1978). Model-fitting approaches to the analysis of human behaviour. *Heredity*, **41**, 249-320. https://www.nature.com/articles/hdy1978101.pdf

See Also

Other Twin Modeling Functions: plot.MxModelTwinMaker(), power.ACE.test(), umxACEcov(), umxACE(), umxCP(), umxDoCp(), umxDoC(), umxGxE_window(), umxGxEbiv(), umxGxE(), umxIP(), umxRotate.MxModelCP(), umxSexLim(), umxSimplex(), umx

Examples

```
# ==============================
# = Univariate model of weight =
# ==============================
require(umx)
data(twinData) # ?twinData from Australian twins.

# Things to note: ACE model of weight will return a NEGATIVE variance in C.
#  This is exactly why we have ACEv! It suggests we need a different model
#  In this case: ADE.
# Other things to note:
# 1. umxACEv can figure out variable names: provide "sep", and selVars.
#    Function generates: "wt" -> "wt1" "wt2"
# 2. umxACEv picks the variables it needs from the data.

mzData = twinData[twinData$zygosity %in% "MZFF", ]
dzData = twinData[twinData$zygosity %in% "DZFF", ]
m1 = umxACEv(selDVs = "wt", sep = "", dzData = dzData, mzData = mzData)

# A short cut (which is even shorter for "_T" twin data with "MZ"/"DZ" data in zygosity column is:
m1 = umxACEv(selDVs = "wt", sep = "", dzData = "MZFF", mzData = "DZFF", data = twinData)
# ========================================================
# = Evidence for dominance ? (DZ correlation set to .25) =
# ========================================================
m2 = umxACEv("ADE", selDVs = "wt", sep = "", dzData = dzData, mzData = mzData, dzCr = .25)
# note: the underlying matrices are still called A, C, and E.
# I catch this in the summary table, so columns are labeled A, D, and E.
# However, currently, the plot will say A, C, E.

# We can modify this model, dropping dominance component (still called C),
# and see a comparison:
m3 = umxModify(m2, update = "C_r1c1", comparison = TRUE, name="AE")
# ========================================================
# = Well done! Now you can make modify twin models in umx =
# ========================================================

# ============================
# = How heritable is height? =
# ============================
#
# Note: Height has a small variance. umx can typically picks good starts,
#    but scaling is advisable.
#
require(umx)
data(twinData) # ?twinData from Australian twins.
```

```
# height var is very small: move from m to cm to increase.
twinData[,c("ht1", "ht2")]= twinData[,c("ht1", "ht2")]*100
mzData <- twinData[twinData$zygosity %in% "MZFF", ]
dzData <- twinData[twinData$zygosity %in% "DZFF", ]
m1 = umxACEv(selDVs = "ht", sep = "", dzData = dzData, mzData = mzData)
umxSummary(m1, std = FALSE) # unstandardized
# tip: with report = "html", umxSummary can print the table to your browser!
plot(m1)

# ========================================================
# = Evidence for dominance ? (DZ correlation set to .25) =
# ========================================================
m2 = umxACEv("ADE", selDVs = "ht", sep="", dzData = dzData, mzData = mzData, dzCr = .25)
umxCompare(m2, m1) # Is ADE better?
umxSummary(m2, comparison = m1) # nb: though this is ADE, matrices are still called A,C,E

# We can modify this model, dropping shared environment, and see a comparison:
m3 = umxModify(m2, update = "C_r1c1", comparison = TRUE, name = "AE")

# ====================================
# = Bivariate height and weight model =
# ====================================

data(twinData)
twinData[,c("ht1", "ht2")]= twinData[,c("ht1", "ht2")]*100
mzData = twinData[twinData$zygosity %in% c("MZFF", "MZMM"), ]
dzData = twinData[twinData$zygosity %in% c("DZFF", "DZMM", "DZOS"), ]
mzData = mzData[1:80, ] # Quicker run to keep CRAN happy
dzData = dzData[1:80, ]
m1 = umxACEv(selDVs = c("ht", "wt"), sep = '', dzData = dzData, mzData = mzData)

# ==================
# = Ordinal example =
# ==================
require(umx)
data(twinData)
# Cut bmi column to form ordinal obesity variables
ordDVs = c("obese1", "obese2")
selDVs = c("obese")
obesityLevels = c('normal', 'overweight', 'obese')
cutPoints = quantile(twinData[, "bmi1"], probs = c(.5, .2), na.rm = TRUE)
twinData$obese1 = cut(twinData$bmi1, breaks = c(-Inf, cutPoints, Inf), labels = obesityLevels)
twinData$obese2 = cut(twinData$bmi2, breaks = c(-Inf, cutPoints, Inf), labels = obesityLevels)
# Make the ordinal variables into mxFactors (ensure ordered is TRUE, and require levels)
twinData[, ordDVs] = mxFactor(twinData[, ordDVs], levels = obesityLevels)
mzData = twinData[twinData$zygosity %in% "MZFF", ][1:80,] # 80 pairs for speed
dzData = twinData[twinData$zygosity %in% "DZFF", ][1:80,]
str(mzData) # make sure mz, dz, and t1 and t2 have the same levels!
m1 = umxACEv(selDVs = selDVs, dzData = dzData, mzData = mzData, sep = '')
umxSummary(m1)

# ============================================
# = Bivariate continuous and ordinal example =
```

```
# ========================================
data(twinData)
selDVs = c("wt", "obese")
# Cut bmi column to form ordinal obesity variables
ordDVs = c("obese1", "obese2")
obesityLevels = c('normal', 'overweight', 'obese')
cutPoints = quantile(twinData[, "bmi1"], probs = c(.5, .2), na.rm = TRUE)
twinData$obese1 = cut(twinData$bmi1, breaks = c(-Inf, cutPoints, Inf), labels = obesityLevels)
twinData$obese2 = cut(twinData$bmi2, breaks = c(-Inf, cutPoints, Inf), labels = obesityLevels)

# Make the ordinal variables into mxFactors (ensure ordered is TRUE, and require levels)
twinData[, ordDVs] = mxFactor(twinData[, ordDVs], levels = obesityLevels)

# umxACEv can trim out unused variables on its own
mzData = twinData[twinData$zygosity %in% "MZFF", ]
dzData = twinData[twinData$zygosity %in% "DZFF", ]
m1 = umxACEv(selDVs = selDVs, dzData = dzData, mzData = mzData, sep = '')
plot(m1)

# ======================================
# = Mixed continuous and binary example =
# ======================================
require(umx)
data(twinData)
# Cut to form category of 20% obese subjects
# and make into mxFactors (ensure ordered is TRUE, and require levels)
cutPoints = quantile(twinData[, "bmi1"], probs = .2, na.rm = TRUE)
obesityLevels = c('normal', 'obese')
twinData$obese1 = cut(twinData$bmi1, breaks = c(-Inf, cutPoints, Inf), labels = obesityLevels)
twinData$obese2 = cut(twinData$bmi2, breaks = c(-Inf, cutPoints, Inf), labels = obesityLevels)
ordDVs = c("obese1", "obese2")
twinData[, ordDVs] = mxFactor(twinData[, ordDVs], levels = obesityLevels)

selDVs = c("wt", "obese")
mzData = twinData[twinData$zygosity %in% "MZFF", ]
dzData = twinData[twinData$zygosity %in% "DZFF", ]
## Not run:
m1 = umxACEv(selDVs = selDVs, dzData = dzData, mzData = mzData, sep = '')
umxSummary(m1)

## End(Not run)

# ==================================
# Example with covariance data only =
# ==================================

require(umx)
data(twinData)
selDVs = c("wt")
mz = cov(twinData[twinData$zygosity %in% "MZFF", tvars(selDVs, "")], use = "complete")
dz = cov(twinData[twinData$zygosity %in% "DZFF", tvars(selDVs, "")], use = "complete")
m1 = umxACEv(selDVs = selDVs, sep= "", dzData = dz, mzData= mz, numObsDZ= 569, numObsMZ= 351)
umxSummary(m1, std = FALSE)
```

umxAlgebra *A simple wrapper for mxAlgebra with name as the first parameter for more readable compact code.*

Description

umxAlgebra is a wrapper for mxAlgebra which has the name parameter first in order.

Usage

```
umxAlgebra(
  name = NA,
  expression,
  dimnames = NA,
  ...,
  joinKey = as.character(NA),
  joinModel = as.character(NA),
  verbose = 0L,
  initial = matrix(as.numeric(NA), 1, 1),
  recompute = c("always", "onDemand"),
  fixed = "deprecated_use_recompute"
)
```

Arguments

name	The name of the algebra (Default = NA). Note the different order compared to mxAlgebra!
expression	The algebra
dimnames	Dimnames of the algebra
...	Other parameters
joinKey	See mxAlgebra documentation
joinModel	See mxAlgebra documentation
verbose	Quiet or informative
initial	See mxAlgebra documentation
recompute	See mxAlgebra documentation
fixed	= See mxAlgebra documentation

Value

- mxAlgebra()

See Also

- umxMatrix()

Other Core Modeling Functions: umxMatrix(), umxModify(), umxPath(), umxRAM(), umxRun(), umxSummary(), umxSuperModel(), umx

Examples

```
x = umxAlgebra("circ", 2 * pi)
class(x$formula)
x = mxAlgebra(name = "circ", 2 * pi)
class(x$formula) # "call"
```

---

umxAPA *Creates nicely formatted journal-style summaries of models, p-values, data-frames and much more.*

Description

umxAPA creates summaries from a range of inputs. Use it for reporting lm models, effects, and summarizing data.

1. Given an stats::lm() model, umxAPA will return a formatted effect, including 95% CI in square brackets e.g.: umxAPA(lm(mpg~wt,data=mtcars),"wt") yields: β = -5.344 [-6.486, -4.203], p< 0.001. here "wt" restricts the output to just the named effect.
2. This also works for t.test(), stats::glm(), cor.test(), and others as I come across them.
3. Get a CI from obj=beta and se=se : umxAPA(-0.30,.03) returns β = -0.3 [-0.36, -0.24]
4. Back out an SE from b and CI: umxAPA(-0.030,c(-0.073,0.013)) returns β = -0.03, se = 0.02
5. Given only a number as obj, will be treated as a p-value, and returned in APA format.
6. Given a dataframe, umxAPA will return a table of correlations with the mean and SD of each variable as the last row. e.g.: umxAPA(mtcars[,c("cyl", "wt", "mpg",)] yields a table of correlations, means and SDs thus:

	cyl	wt	mpg
cyl	1	0.78	-0.85
wt	0.78	1	-0.87
mpg	-0.85	-0.87	1
mean_sd	6.19 (1.79)	3.22 (0.98)	20.09 (6.03)

Usage

```
umxAPA(
  obj = .Last.value,
  se = NULL,
```

```
  p = NULL,
  std = FALSE,
  digits = 2,
  use = "complete",
  min = 0.001,
  addComparison = NA,
  report = c("markdown", "html"),
  lower = TRUE,
  test = c("Chisq", "LRT", "Rao", "F", "Cp"),
  SEs = TRUE,
  means = TRUE
)
```

Arguments

obj	A model (e.g. lm(), lme(), glm(), t.test()), beta-value, or data.frame
se	If obj is a beta, se treated as standard-error (returning a CI). If obj is a model, used to select effect of interest (blank for all effects). Finally, set se to the CI c(lower, upper), to back out the SE.
p	If obj is a beta, use p-value to compute SE (returning a CI).
std	Whether to report std betas (re-runs model on standardized data).
digits	How many digits to round output.
use	If obj is a data.frame, how to handle NAs (default = "complete")
min	For a p-value, the smallest value to report numerically (default .001)
addComparison	For a p-value, whether to add "</=" default (NA) adds "<" if necessary
report	What to return (default = 'markdown'). Use 'html' to open a web table.
lower	Whether to not show the lower triangle of correlations for a data.frame (Default TRUE)
test	If obj is a glm, which test to use to generate p-values options = "Chisq", "LRT", "Rao", "F", "Cp"
SEs	Whether or not to show correlations with their SE (Default TRUE)
means	Whether or not to show means in a correlation table (Default TRUE)

Value

- string

References

- https://github.com/tbates/umx, https://my.ilstu.edu/~jhkahn/apastats.html

See Also

SE_from_p()

Other Reporting Functions: loadings.MxModel(), umxFactorScores(), umxGetParameters(), umxParameters(), umxReduce(), umx_aggregate(), umx_names(), umx_time(), umx

Examples

```
# ========================================
# = Report lm (regression/anova) results =
# ========================================
umxAPA(lm(mpg ~ wt + disp, mtcars)) # All parameters
umxAPA(lm(mpg ~ wt + disp, mtcars), "disp") # Just disp effect
umxAPA(lm(mpg ~ wt + disp, mtcars), std = TRUE) # Standardize effects

# glm example
df = mtcars
df$mpg_thresh = 0
df$mpg_thresh[df$mpg>16] = 1
m1 = glm(mpg_thresh ~ wt + gear,data = df, family = binomial)
umxAPA(m1)

# A t-Test
m1 = t.test(1:10, y = c(7:20))
umxAPA(m1)

# ========================================================
# = Summarize a DATA FRAME: Correlations + Means and SDs =
# ========================================================
umxAPA(mtcars[,1:3])
umxAPA(mtcars[,1:3], digits = 3)
umxAPA(mtcars[,1:3], lower = FALSE)
## Not run:
umxAPA(mtcars[,1:3], report = "html")

## End(Not run)

# ================================================
# = CONFIDENCE INTERVAL text from effect and se =
# ================================================
umxAPA(.4, .3) # parameter 2 interpreted as SE

# Input beta and CI, and back out the SE
umxAPA(-0.030, c(-0.073, 0.013), digits = 3)

# ====================
# = Format a p-value =
# ====================
umxAPA(.0182613)
umxAPA(.000182613)
umxAPA(.000182613,  addComparison=FALSE)

# ========================
# = Report a correlation =
# ========================
data(twinData)
tmp = subset(twinData, zygosity %in% c("MZFF", "MZMM"))
m1 = cor.test(~ wt1 + wt2, data = tmp)
```

```
umxAPA(m1)

# ====================
# = Report a t-test =
# ====================
m1 = t.test(extra ~ group, data = sleep)
umxAPA(m1)
```

---

umxBrownie *A recipe Easter-egg for umx*

Description

How to cook steak.

Usage

```
umxBrownie()
```

Details

Equipment matters. You should buy a heavy cast-iron skillet, and a digital internal thermometer. Preferably cook over a gas flame.

note: Cheaper cuts like blade steak can come out fine.

A great reference is The Food Lab by Kenji Alt Lopez. https://www.amazon.co.uk/Food-Lab-Cooking-Through-Science/dp/0393081087.

References

- The Food Lab

See Also

- omxBrownie()

Other Miscellaneous Utility Functions: install.OpenMx(), qm(), umxLav2RAM(), umxRAM2Lav(), umxVersion(), umx_array_shift(), umx_find_object(), umx_msg(), umx_open_CRAN_page(), umx_pad(), umx_print(), umx_score_scale(), umx

Examples

```
umxBrownie()
```

umxCI *Add (and, optionally, run) confidence intervals to a structural model.*

Description

umxCI adds OpenMx::mxCI() calls for requested (default all) parameters in a model, runs these CIs if necessary, and reports them in a neat summary.

Usage

```
umxCI(
  model = NULL,
  which = c("ALL", NA, "list of your making"),
  remove = FALSE,
  run = c("no", "yes", "if necessary", "show"),
  interval = 0.95,
  type = c("both", "lower", "upper"),
  showErrorCodes = TRUE
)
```

Arguments

model	The mxModel() you wish to report mxCI()s on
which	What CIs to add: c("ALL", NA, "list of your making")
remove	= FALSE (if set, removes existing specified CIs from the model)
run	Whether or not to compute the CIs. Valid values = "no" (default), "yes", "if necessary". 'show' means print the intervals if computed, or list their names if not.
interval	The interval for newly added CIs (defaults to 0.95)
type	The type of CI (defaults to "both", options are "lower" and "upper")
showErrorCodes	Whether to show errors (default == TRUE)

Details

umxCI also reports if any problems were encountered. The codes are standard OpenMx errors and warnings

- 1: The final iterate satisfies the optimality conditions to the accuracy requested, but the sequence of iterates has not yet converged. NPSOL was terminated because no further improvement could be made in the merit function (Mx status GREEN)
- 2: The linear constraints and bounds could not be satisfied. The problem has no feasible solution.
- 3: The nonlinear constraints and bounds could not be satisfied. The problem may have no feasible solution.
- 4: The major iteration limit was reached (Mx status BLUE).

- 6: The model does not satisfy the first-order optimality conditions to the required accuracy, and no improved point for the merit function could be found during the final linesearch (Mx status RED)
- 7: The function derivatives returned by funcon or funobj appear to be incorrect.
- 9: An input parameter was invalid.

If run = "no", the function simply adds the CI requests, but returns the model without running them.

Value

- mxModel()

References

- https://www.github.com/tbates/umx

See Also

- stats::confint(), umxConfint(), umxCI(), umxModify()

Other Reporting functions: RMSEA.MxModel(), RMSEA.summary.mxmodel(), RMSEA(), extractAIC.MxModel(), loadings(), residuals.MxModel(), umxCI_boot(), umxCompare(), umxConfint(), umxExpCov(), umxExpMeans(), umxFitIndices(), umxPlotACEv(), umxRotate(), umxSummary.MxModel()

Examples

```
require(umx)
data(demoOneFactor)
manifests = names(demoOneFactor)

m1 = umxRAM("One Factor", data = demoOneFactor, type = "cov",
umxPath("G", to = manifests),
umxPath(var = manifests),
umxPath(var = "G", fixedAt = 1)
)
m1$intervals # none yet - empty list()
m1 = umxCI(m1)
m1$intervals # $G_to_x1
m1 = umxCI(m1, remove = TRUE) # remove CIs from the model and return it

# ========================
# = A twin model example =
# ========================
data(twinData)
mzData <- subset(twinData, zygosity == "MZFF")
dzData <- subset(twinData, zygosity == "DZFF")
m1 = umxACE(selDVs = c("bmi1","bmi2"), dzData = dzData, mzData = mzData)
## Not run:
umxCI(m1, run = "show") # show what will be requested
umxCI(m1, run = "yes") # actually compute the CIs
# Don't force update of CIs, but if they were just added, then calculate them
umxCI(m1, run = "if necessary")
```

```
m1 = umxCI(m1, remove = TRUE) # remove them all
m1$intervals # none!
# Show what parameters are available to get CIs on
umxParameters(m1)
# Request a CI by label:
m1 = umxCI(m1, "a_r1c1", run = "yes")

## End(Not run)
```

---

umxCI_boot *umxCI_boot*

Description

Compute boot-strapped Confidence Intervals for parameters in an `mxModel()` The function creates a sampling distribution for parameters by repeatedly drawing samples with replacement from your data and then computing the statistic for each redrawn sample.

Usage

```
umxCI_boot(
  model,
  rawData = NULL,
  type = c("par.expected", "par.observed", "empirical"),
  std = TRUE,
  rep = 1000,
  conf = 95,
  dat = FALSE,
  digits = 3
)
```

Arguments

model	is an optimized mxModel
rawData	is the raw data matrix used to estimate model
type	is the kind of bootstrap you want to run. "par.expected" and "par.observed" use parametric Monte Carlo bootstrapping based on your expected and observed covariance matrices, respectively. "empirical" uses empirical bootstrapping based on rawData.
std	specifies whether you want CIs for unstandardized or standardized parameters (default: std = TRUE)
rep	is the number of bootstrap samples to compute (default = 1000).
conf	is the confidence value (default = 95)
dat	specifies whether you want to store the bootstrapped data in the output (useful for multiple analyses, such as mediation analysis)
digits	rounding precision

Value

- expected covariance matrix

References

- https://openmx.ssri.psu.edu/thread/2598 Original written by https://openmx.ssri.psu.edu/users/bwiernik

See Also

- umxExpMeans(), umxExpCov()

Other Reporting functions: RMSEA.MxModel(), RMSEA.summary.mxmodel(), RMSEA(), extractAIC.MxModel(), loadings(), residuals.MxModel(), umxCI(), umxCompare(), umxConfint(), umxExpCov(), umxExpMeans(), umxFitIndices(), umxPlotACEv(), umxRotate(), umxSummary.MxModel()

Examples

```
## Not run:
require(umx)
data(demoOneFactor)
manifests = names(demoOneFactor)

m1 = umxRAM("One Factor", data = demoOneFactor, type = "cov",
umxPath("G", to = manifests),
umxPath(var = manifests),
umxPath(var = "G", fixedAt = 1.0)
)

umxCI_boot(m1, type = "par.expected")

## End(Not run)
```

---

umxCompare *Print a comparison table of one or more mxModel()s, formatted nicely.*

Description

umxCompare compares two or more mxModel()s. It has several nice features:

1. It supports direct control of rounding, and reports p-values rounded to APA style.
2. It reports the table in your preferred format (default is markdown, options include latex).
3. Table columns are arranged to make for easy comparison for readers.
4. report = 'inline', will provide an English sentence suitable for a paper.
5. report = "html" opens a web table in your browser to paste into a word processor.

Note: If you leave comparison blank, it will just give fit info for the base model

Usage

```
umxCompare(
  base = NULL,
  comparison = NULL,
  all = TRUE,
  digits = 3,
  report = c("markdown", "inline", "html"),
  compareWeightedAIC = FALSE,
  file = "tmp.html"
)
```

Arguments

base	The base mxModel() for comparison
comparison	The model (or list of models) which will be compared for fit with the base model (can be empty)
all	Whether to make all possible comparisons if there is more than one base model (defaults to T)
digits	rounding for p-values etc.
report	"markdown" (default), "inline" (a sentence suitable for inclusion in a paper), or "html". create a web table and open your default browser. (handy for getting tables into Word, and other text systems!)
compareWeightedAIC	Show the Wagenmakers AIC weighted comparison (default = FALSE)
file	file to write html too if report = "html" (defaults to "tmp.html")

References

- https://www.github.com/tbates/umx

See Also

- mxCompare(), umxSummary(), umxRAM(),

Other Reporting functions: RMSEA.MxModel(), RMSEA.summary.mxmodel(), RMSEA(), extractAIC.MxModel(), loadings(), residuals.MxModel(), umxCI_boot(), umxCI(), umxConfint(), umxExpCov(), umxExpMeans(), umxFitIndices(), umxPlotACEv(), umxRotate(), umxSummary.MxModel()

Examples

```
require(umx)
data(demoOneFactor)
manifests = names(demoOneFactor)

m1 = umxRAM("One Factor", data = demoOneFactor, type = "cov",
umxPath("G", to = manifests),
umxPath(var = manifests),
umxPath(var = "G", fixedAt = 1)
```

```
)

m2 = umxModify(m1, update = "G_to_x2", name = "drop_path_2_x2")
umxCompare(m1, m2)
umxCompare(m1, m2, report = "inline") # Add English-sentence descriptions
## Not run:
umxCompare(m1, m2, report = "html") # Open table in browser

## End(Not run)
m3 = umxModify(m2, update = "G_to_x3", name = "drop_path_2_x2_and_3")
umxCompare(m1, c(m2, m3))
umxCompare(m1, c(m2, m3), compareWeightedAIC = TRUE)
umxCompare(c(m1, m2), c(m2, m3), all = TRUE)
```

`umxConfint` *Get confidence intervals from a umx model*

Description

Implements confidence interval function for umx models.

Usage

```
umxConfint(
  object,
  parm = c("existing", "all", "or one or more labels", "smart"),
  wipeExistingRequests = TRUE,
  level = 0.95,
  run = FALSE,
  showErrorCodes = FALSE,
  optimizer = c("SLSQP", "NPSOL", "CSOLNP", "current")
)
```

Arguments

`object`	An `mxModel()`, possibly already containing `mxCI()`s that have been `mxRun()` with intervals = TRUE))
`parm`	Which parameters to get confidence intervals for. Can be "existing", "all", or one or more parameter names.
`wipeExistingRequests`	Whether to remove existing CIs when adding new ones (ignored if parm = 'existing').
`level`	The confidence level required (default = .95)
`run`	Whether to run the model (defaults to FALSE)
`showErrorCodes`	(default = FALSE)
`optimizer`	For difficult CIs, trying other optimizers can help!

Details

Note: By default, requesting new CIs wipes the existing ones. To keep these, set wipeExistingRequests = FALSE.

Note: `confint()` is an OpenMx function which will return SE-based CIs.

Because these can take time to run, by default only CIs already computed will be reported. Set run = TRUE to run new CIs. If `parm` is empty, and `run = FALSE`, a message will alert you to set `run = TRUE`.

Value

- `mxModel()`

References

- https://www.github.com/tbates/umx

See Also

- `stats::confint()`, `OpenMx::mxSE()`, `umxCI()`, `OpenMx::mxCI()`

Other Reporting functions: `RMSEA.MxModel()`, `RMSEA.summary.mxmodel()`, `RMSEA()`, `extractAIC.MxModel()`, `loadings()`, `residuals.MxModel()`, `umxCI_boot()`, `umxCI()`, `umxCompare()`, `umxExpCov()`, `umxExpMeans()`, `umxFitIndices()`, `umxPlotACEv()`, `umxRotate()`, `umxSummary.MxModel()`

Examples

```
require(umx)
data(demoOneFactor)

manifests = names(demoOneFactor)
m1 = umxRAM("OneFactor", data = demoOneFactor, type = "cov",
umxPath(from = "G", to = manifests),
umxPath(var = manifests),
umxPath(var = "G", fixedAt = 1)
)

m1 = umxConfint(m1, run = TRUE) # There are no existing CI requests...

## Not run:
# Add a CI request for "G_to_x1", run, and report. Save with this CI computed
m2 = umxConfint(m1, parm = "G_to_x1", run = TRUE)

# Just print out any existing CIs
umxConfint(m2)

# CI requests added for free matrix parameters. User prompted to set run = TRUE
m3 = umxConfint(m1, "all")

# Run the requested CIs
m3 = umxConfint(m3, run = TRUE)
```

```
# Run CIs for free one-headed (asymmetric) paths in RAM model.
#   note: Deletes other existing requests,
tmp = umxConfint(m1, parm = "A", run = TRUE)

# Wipe existing CIs, add G_to_x1
tmp = umxConfint(m1, parm = "G_to_x1", run = TRUE, wipeExistingRequests = TRUE)

# For some twin models, a "smart" mode is implemented
# note: only implemented for umxCP so far
m2 =  umxConfint(m1, "smart")

## End(Not run)
```

---

umxCov2cor *Convert a covariance matrix into a correlation matrix*

Description

A version of cov2cor() that forces upper and lower triangles to be *identical* (rather than nearly identical)

Usage

```
umxCov2cor(x)
```

Arguments

x	something that cov2cor can work on (matrix, df, etc.)

Value

- A correlation matrix

References

- https://www.github.com/tbates/umx

See Also

cov2cor()

Other Miscellaneous Stats Helpers: FishersMethod(), SE_from_p(), oddsratio(), reliability(), umxHetCor(), umxWeightedAIC(), umx_apply(), umx_cor(), umx_means(), umx_r_test(), umx_round(), umx_scale(), umx_var(), umx

Examples

```
umxCov2cor(cov(mtcars[,1:5]))
```

umxCP *umxCP: Build and run a Common pathway twin model*

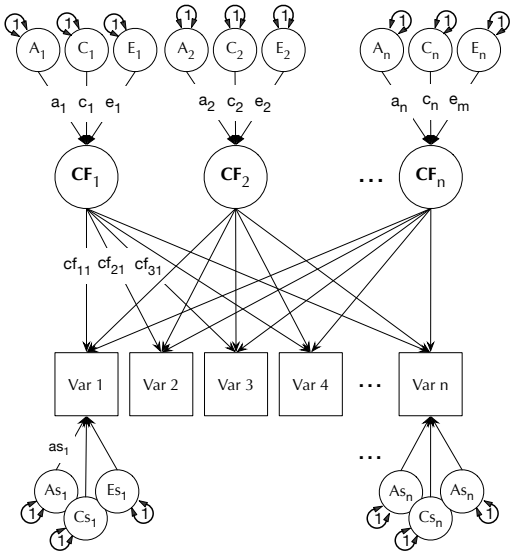
Description

Make a 2-group Common Pathway twin model (Common-factor common-pathway multivariate model).

The common-pathway model provides a powerful tool for theory-based decomposition of genetic and environmental differences.

umxCP supports this with pairs of mono-zygotic (MZ) and di-zygotic (DZ) twins reared together to model the genetic and environmental structure of multiple phenotypes (measured behaviors).

Common-pathway path diagram:



As can be seen, each phenotype also by default has A, C, and E influences specific to that phenotype.

Features include the ability to include more than one common pathway, to use ordinal data.

note: The function `umx_set_mvn_optimization_options()` allows users to see and set `mvnRelEps` and `mvnMaxPointsA` mvnRelEps defaults to .005. For ordinal models, you might find that '0.01' works better.

Usage

```
umxCP(
  name = "CP",
  selDVs,
  selCovs = NULL,
  dzData = NULL,
  mzData = NULL,
```

```
  sep = NULL,
  nFac = 1,
  type = c("Auto", "FIML", "cov", "cor", "WLS", "DWLS", "ULS"),
  data = NULL,
  zyg = "zygosity",
  allContinuousMethod = c("cumulants", "marginals"),
  correlatedA = FALSE,
  dzAr = 0.5,
  dzCr = 1,
  autoRun = getOption("umx_auto_run"),
  tryHard = c("no", "yes", "ordinal", "search"),
  optimizer = NULL,
  equateMeans = TRUE,
  weightVar = NULL,
  bVector = FALSE,
  boundDiag = 0,
  addStd = TRUE,
  addCI = TRUE,
  numObsDZ = NULL,
  numObsMZ = NULL,
  freeLowerA = FALSE,
  freeLowerC = FALSE,
  freeLowerE = FALSE
)
```

Arguments

name	The name of the model (defaults to "CP").
selDVs	The variables to include. omit sep in selDVs, i.e., just "dep" not c("dep_T1", "dep_T2").
selCovs	basenames for covariates
dzData	The DZ dataframe.
mzData	The MZ dataframe.
sep	(required) The suffix for twin 1 and twin 2, often "_T".
nFac	How many common factors (default = 1)
type	One of "Auto", "FIML", "cov", "cor", "WLS", "DWLS", "ULS"
data	If provided, dzData and mzData are treated as valid levels of zyg to select() data sets (default = NULL)
zyg	If data provided, this column is used to select rows by zygosity (Default = "zygosity")
allContinuousMethod	"cumulants" or "marginals". Used in all-continuous WLS data to determine if a means model needed.
correlatedA	?? (default = FALSE).
dzAr	The DZ genetic correlation (defaults to .5, vary to examine assortative mating).

dzCr	The DZ "C" correlation (defaults to 1: set to .25 to make an ADE model).
autoRun	Whether to run the model (default), or just to create it and return without running.
tryHard	Default ('no') uses normal mxRun. "yes" uses mxTryHard. Other options: "ordinal", "search"
optimizer	optionally set the optimizer (default NULL does nothing).
equateMeans	Whether to equate the means across twins (defaults to TRUE).
weightVar	If provided, a vector objective will be used to weight the data. (default = NULL).
bVector	Whether to compute row-wise likelihoods (defaults to FALSE).
boundDiag	= Numeric lbound for diagonal of the a_cp, c_cp, & e_cp matrices. Set = NULL to ignore.
addStd	Whether to add the algebras to compute a std model (defaults to TRUE).
addCI	Whether to add the interval requests for CIs (defaults to TRUE).
numObsDZ	= not yet implemented: Ordinal Number of DZ twins: Set this if you input covariance data.
numObsMZ	= not yet implemented: Ordinal Number of MZ twins: Set this if you input covariance data.
freeLowerA	(ignore): Whether to leave the lower triangle of A free (default = FALSE).
freeLowerC	(ignore): Whether to leave the lower triangle of C free (default = FALSE).
freeLowerE	(ignore): Whether to leave the lower triangle of E free (default = FALSE).

Details

Like the umxACE() model, the CP model decomposes phenotypic variance into Additive genetic, unique environmental (E) and, optionally, either common or shared-environment (C) or non-additive genetic effects (D).

Unlike the Cholesky, these factors do not act directly on the phenotype. Instead latent A, C, and E influences impact on one or more latent factors which in turn account for variance in the phenotypes (see Figure).

Data Input Currently, the umxCP function accepts only raw data. This may change in future versions.

Ordinal Data

In an important capability, the model transparently handles ordinal (binary or multi-level ordered factor data) inputs, and can handle mixtures of continuous, binary, and ordinal data in any combination.

Additional features

The umxCP function supports varying the DZ genetic association (defaulting to .5) to allow exploring assortative mating effects, as well as varying the DZ "C" factor from 1 (the default for modeling family-level effects shared 100% by twins in a pair), to .25 to model dominance effects.

Matrices and Labels in CP model

A good way to see which matrices are used in umxCP is to run an example model and plot it.

All the shared matrices are in the model "top".

Matrices topas, topcs, and top$es contain the path loadings specific to each variable on their diagonals.

So, to see the 'as' values, labels, or free states, you can say:

```
m1$top$as$values
```

```
m1$top$as$free
```

```
m1$top$as$labels
```

Labels relevant to modifying the specific loadings take the form "as_r1c1", "as_r2c2" etc.

The common-pathway loadings on the factors are in matrices topa_cp, topc_cp, top$e_cp.

The common factors themselves are in the matrix top$cp_loadings (an nVar * 1 matrix)

Less commonly-modified matrices are the mean matrix expMean. This has 1 row, and the columns are laid out for each variable for twin 1, followed by each variable for twin 2. So, in a model where the means for twin 1 and twin 2 had been equated (set = to T1), you could make them independent again with this line:

```
m1$top$expMean$labels[1,4:6] = c("expMean_r1c4","expMean_r1c5","expMean_r1c6")
```

For a deep-dive, see xmu_make_TwinSuperModel()

Value

- mxModel()

References

- https://www.github.com/tbates/umx

See Also

- umxSummaryCP(), umxPlotCP(). See umxRotate.MxModelCP() to rotate the factor loadings of a umxCP() model. See umxACE() for more examples of twin modeling. plot() and umxSummary() work for all twin models, e.g., umxIP(), umxCP(), umxGxE(), and umxACE().

Other Twin Modeling Functions: plot.MxModelTwinMaker(), power.ACE.test(), umxACEcov(), umxACEv(), umxACE(), umxDoCp(), umxDoC(), umxGxE_window(), umxGxEbiv(), umxGxE(), umxIP(), umxRotate.MxModelCP(), umxSexLim(), umxSimplex(), umx

Examples

```
## Not run:
# ============================================================
# = Run a 3-factor Common pathway twin model of 6 traits =
# ============================================================
require(umx)
umx_set_optimizer("SLSQP")
data(GFF)
mzData = subset(GFF, zyg_2grp == "MZ")
dzData = subset(GFF, zyg_2grp == "DZ")
selDVs = c("gff", "fc", "qol", "hap", "sat", "AD")
```

```
m1 = umxCP("new", selDVs = selDVs, sep = "_T", nFac = 3,
dzData = dzData, mzData = mzData, tryHard = "yes")

# Shortcut using "data ="
selDVs = c("gff", "fc", "qol", "hap", "sat", "AD")
m1 = umxCP(selDVs = selDVs, nFac = 3, data=GFF, zyg="zyg_2grp")

# ===================
# = Do it using WLS =
# ===================
m2 = umxCP("new", selDVs = selDVs, sep = "_T", nFac = 3, optimizer = "SLSQP",
dzData = dzData, mzData = mzData, tryHard = "ordinal",
type= "DWLS", allContinuousMethod='marginals'
)

# ================================================
# = Find and test dropping of shared environment  =
# ================================================
# Show all labels for C parameters
umxParameters(m1, patt = "^c")
# Test dropping the 9 specific and common-factor C paths
m2 = umxModify(m1, regex = "(cs_.*$)|(c_cp_)", name = "dropC", comp = TRUE)
umxSummaryCP(m2, comparison = m1, file = NA)
umxCompare(m1, m2)

# ======================================
# = Mixed continuous and binary example =
# ======================================
data(GFF)
# Cut to form umxFactor 20% depressed  DEP
cutPoints = quantile(GFF[, "AD_T1"], probs = .2, na.rm = TRUE)
ADLevels  = c('normal', 'depressed')
GFF$DEP_T1 = cut(GFF$AD_T1, breaks = c(-Inf, cutPoints, Inf), labels = ADLevels)
GFF$DEP_T2 = cut(GFF$AD_T2, breaks = c(-Inf, cutPoints, Inf), labels = ADLevels)
ordDVs = c("DEP_T1", "DEP_T2")
GFF[, ordDVs] = umxFactor(GFF[, ordDVs])

selDVs = c("gff","fc","qol","hap","sat","DEP")
mzData = subset(GFF, zyg_2grp == "MZ")
dzData = subset(GFF, zyg_2grp == "DZ")

# umx_set_optimizer("NPSOL")
# umx_set_mvn_optimization_options("mvnRelEps", .01)
m1 = umxCP(selDVs = selDVs, sep = "_T", nFac = 3, dzData = dzData, mzData = mzData)
m2 = umxModify(m1, regex = "(cs_r[3-5]|c_cp_r[12])", name = "dropC", comp= TRUE)

# Do it using WLS
m3 = umxCP(selDVs = selDVs, sep = "_T", nFac = 3, dzData = dzData, mzData = mzData,
tryHard = "ordinal", type= "DWLS")
# TODO umxCPL fix WLS here
# label at row 1 and column 1 of matrix 'top.binLabels'' in model 'CP3fac' : object 'Vtot'

# Correlated factors example
```

```
data(GFF)
mzData = subset(GFF, zyg_2grp == "MZ")
dzData = subset(GFF, zyg_2grp == "DZ")
selDVs = c("gff", "fc", "qol", "hap", "sat", "AD")
m1 = umxCP("new", selDVs = selDVs, sep = "_T", dzData = dzData, mzData = mzData,
nFac = 3, correlatedA = TRUE, tryHard = "yes")

## End(Not run)
```

---

umxDiagnose *Diagnose problems in a model - this is a work in progress.*

Description

The goal of this function **WILL BE** (not currently functional) is to diagnose problems in a model and return suggestions to the user. It is a work in progress, and of no use as yet.

Usage

```
umxDiagnose(model, tryHard = FALSE, diagonalizeExpCov = FALSE)
```

Arguments

model	an mxModel() to diagnose
tryHard	whether I should try and fix it? (defaults to FALSE)
diagonalizeExpCov	Whether to diagonalize the ExpCov

Details

Best diagnostics are:

1. Observed data variances and means
2. Expected variances and means
3. Difference of these?

Try * diagonalizeExpCov diagonal. * umx_is_ordered()

more tricky - we should really report the variances and the standardized thresholds.

The guidance would be to try starting with unit variances and thresholds that are within +/- 2SD of the mean. bivariate outliers %p option

Value

- helpful messages and perhaps a modified model

References

- https://tbates.github.io, https://github.com/tbates/umx

See Also

Other Teaching and Testing functions: tmx_show(), umxPower()

Examples

```
require(umx)
data(demoOneFactor)
manifests = names(demoOneFactor)

m1 = umxRAM("OneFactor", data = demoOneFactor, type= "cov",
umxPath("G", to = manifests),
umxPath(var = manifests),
umxPath(var = "G", fixedAt = 1)
)
m1 = mxRun(m1)
umxSummary(m1, std = TRUE)
umxDiagnose(m1)
```

---

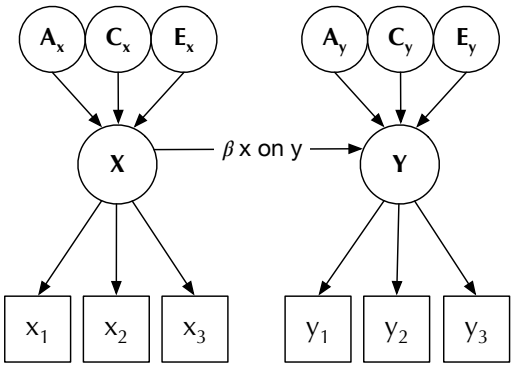
umxDoC *Build and run a 2-group Direction of Causation twin models.*

Description

Testing causal claims is often difficult due to an inability to conduct experimental randomization of traits and situations to people. When twins are available, even when measured on a single occasion, the pattern of cross-twin cross-trait correlations can (given distinguishable modes of inheritance for the two traits) falsify causal hypotheses.

umxDoC implements a 2-group model to form latent variables for each of two traits, and allows testing whether trait 1 causes trait 2, vice-versa, or even reciprocal causation.

The following figure shows how the DoC model appears as a path diagram (for two latent variables X and Y, each with three indicators). Note: For pedagogical reasons, only the model for 1 twin is shown, and only one DoC pathway drawn.



Usage

```
umxDoC(
  name = "DoC",
  var1Indicators,
  var2Indicators,
  mzData = NULL,
  dzData = NULL,
  sep = "_T",
  causal = TRUE,
  autoRun = getOption("umx_auto_run"),
  intervals = FALSE,
  tryHard = c("no", "yes", "ordinal", "search"),
  optimizer = NULL
)
```

Arguments

name	The name of the model (defaults to "DOC").
var1Indicators	variables defining latent trait 1
var2Indicators	variables defining latent trait 2
mzData	The MZ dataframe
dzData	The DZ dataframe
sep	The separator in twin variable names, default = "_T", e.g. "dep_T1".
causal	whether to add the causal paths (default TRUE)
autoRun	Whether to run the model (default), or just to create it and return without running.
intervals	Whether to run mxCI confidence intervals (default = FALSE)
tryHard	Default ('no') uses normal mxRun. "yes" uses mxTryHard. Other options: "ordinal", "search"
optimizer	Optionally set the optimizer (default NULL does nothing).

Details

To be added.

Value

- `mxModel()` of subclass MxModelDoC

References

- N.A. Gillespie and N.G. Martin (2005). Direction of Causation Models. In *Encyclopedia of Statistics in Behavioral Science*, **1**. 496–499. Eds. Brian S. Everitt & David C. Howell.

See Also

- plot.MxModelDoC(), umxSummary.MxModelDoC(), umxModify()

Other Twin Modeling Functions: plot.MxModelTwinMaker(), power.ACE.test(), umxACEcov(), umxACEv(), umxACE(), umxCP(), umxDoCp(), umxGxE_window(), umxGxEbiv(), umxGxE(), umxIP(), umxRotate.MxModelCP(), umxSexLim(), umxSimplex(), umx

Examples

```
## Not run:
# ========================
# = Does Rain cause Mud? =
# ========================

# ======================================
# = 2. Define manifests for var 1 and 2 =
# ======================================
var1 = paste0("varA", 1:3)
var2 = paste0("varB", 1:3)

# ================
# = 1. Load Data =
# ================
data(docData)
docData = umx_scale_wide_twin_data(c(var1, var2), docData, sep= "_T")
mzData  = subset(docData, zygosity %in% c("MZFF", "MZMM"))
dzData  = subset(docData, zygosity %in% c("DZFF", "DZMM"))

# ======================================================
# = 2. Make the non-causal (Cholesky) and causal models =
# ======================================================
Chol = umxDoC(var1= var1, var2= var2, mzData= mzData, dzData= dzData, causal= FALSE)
DoC  = umxDoC(var1= var1, var2= var2, mzData= mzData, dzData= dzData, causal= TRUE)

# ================================================
# = Make the directional models by modifying DoC =
# ================================================
a2b   = umxModify(DoC, "a2b", free = TRUE, name = "a2b"); summary(a2b)
b2a   = umxModify(DoC, "b2a", free = TRUE, name = "b2a"); summary(b2a)
Recip = umxModify(DoC, c("a2b", "b2a"), free = TRUE, name = "Recip"); summary(Recip)

var1 = paste0("SOS", 1:8)
var2 = paste0("Vocab", 1:10)
Chol = umxDoC(var1= var1, var2= var2,mzData= mzData, dzData= dzData, causal= FALSE)
DoC  = umxDoC(var1= var1, var2= var2, mzData= mzData, dzData= dzData, causal= TRUE)
a2b  = umxModify(DoC, "a2b", free = TRUE, name = "a2b")
b2a  = umxModify(DoC, "b2a", free = TRUE, name = "b2a")
Recip= umxModify(DoC, c("a2b", "b2a"), free = TRUE, name = "Recip")
umxCompare(Chol, c(a2b, b2a, Recip))


## End(Not run)
```

umxDoCp *Make a direction of causation model based on umxPath statements*

Description

Makes a direction of causation model with umxPath() statements

Usage

```
umxDoCp(
  var1Indicators,
  var2Indicators,
  mzData = NULL,
  dzData = NULL,
  sep = "_T",
  causal = TRUE,
  name = "DoC",
  autoRun = getOption("umx_auto_run"),
  intervals = FALSE,
  tryHard = c("no", "yes", "ordinal", "search"),
  optimizer = NULL
)
```

Arguments

var1Indicators	The indicators of trait 1
var2Indicators	The indicators of trait 2
mzData	The MZ twin dataframe
dzData	The DZ twin dataframe
sep	(Default "_T")
causal	(Default TRUE)
name	= "DoC"
autoRun	Default: getOption("umx_auto_run")_
intervals	Whether to run intervals (Default FALSE)
tryHard	Default "no" (valid = "yes", "ordinal", "search")
optimizer	Whether to set this for this run (Default no))

Details

See also umxDoC()

Value

- [A direction of causation model with umxPath() statements.

See Also

- umxDoC()

Other Twin Modeling Functions: plot.MxModelTwinMaker(), power.ACE.test(), umxACEcov(), umxACEv(), umxACE(), umxCP(), umxDoC(), umxGxE_window(), umxGxEbiv(), umxGxE(), umxIP(), umxRotate.MxModelCP(), umxSexLim(), umxSimplex(), umx

Examples

```
## Not run:
# ================
# = 1. Load Data =
# ================
data(docData)
var1 = paste0("varA", 1:3)
var2 = paste0("varB", 1:3)
tmp = umx_scale_wide_twin_data(varsToScale= c(var1, var2), sep= "_T", data= docData)
mzData = subset(docData, zygosity %in% c("MZFF", "MZMM"))
dzData = subset(docData, zygosity %in% c("DZFF", "DZMM"))
m1 = umxDoCp(var1, var2, mzData= mzData, dzData= dzData, sep = "_T", causal= TRUE)


## End(Not run)
```

---

umxEFA *FIML-based Exploratory Factor Analysis (EFA)*

Description

Perform full-information maximum-likelihood factor analysis on a data matrix.

Usage

```
umxEFA(
  x = NULL,
  factors = NULL,
  data = NULL,
  scores = c("none", "ML", "WeightedML", "Regression"),
  minManifests = NA,
  rotation = c("varimax", "promax", "none"),
  return = c("model", "loadings"),
  report = c("markdown", "html"),
  summary = FALSE,
  name = "efa",
  digits = 2,
```

```
  n.obs = NULL,
  covmat = NULL
)
```

Arguments

x	Either 1: data, 2: Right-hand-side ~ formula , 3: Vector of variable names, or 4: Name for the model.
factors	Either number of factors to request or a vector of factor names.
data	A dataframe you are modeling.
scores	Type of scores to produce, if any. The default is none, "Regression" gives Thompson's scores. Other options are 'ML', 'WeightedML', Partial matching allows these names to be abbreviated.
minManifests	The least number of variables required to return a score for a participant (Default = NA).
rotation	A rotation to perform on the loadings (default = "varimax" (orthogonal))
return	by default, the resulting MxModel is returned. Say "loadings" to get a fact.anal object.
report	Report as markdown to the console, or open a table in browser ("html")
summary	run `umxSummary()` on the underlying umxRAM model? (Default = FALSE)
name	A name for your model (default = efa)
digits	rounding (default = 2)
n.obs	Number of observations in if covmat provided (default = NA)
covmat	Covariance matrix of data you are modeling (not implemented)

Details

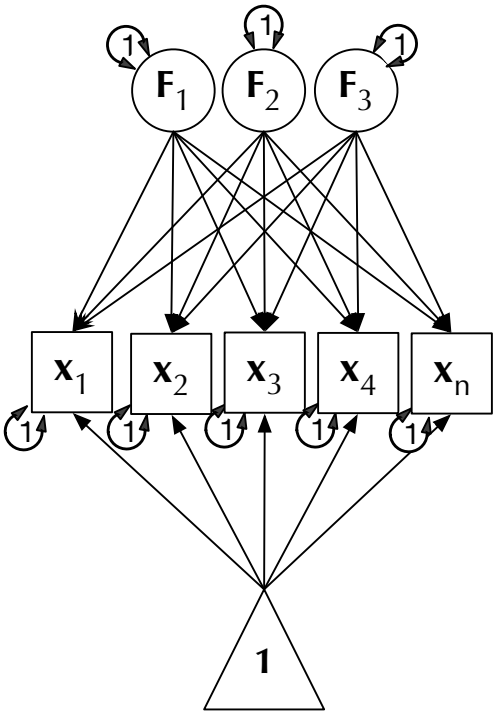
As in `factanal()`, you need only specify the number of factors and offer up some manifest data, e.g:

```
umxEFA(factors = 2,data = mtcars)
```

Equivalently, you can also give a list of factor names:

```
umxEFA(factors = c("g","v"),data = mtcars)
```

The factor model is implemented as a structural equation model, e.g.



You can request `scores` from the model. Unlike factanal, these can cope with missing data.

You can also rotate the factors using any rotation function.

In an EFA, all items may load on all factors.

Should work with rotations provided in `library("GPArotation")` and `library("psych")`, e.g

Orthogonal: "varimax", "quartimax", "bentlerT", "equamax", "varimin", "geominT" and "bifactor"
Oblique: "Promax", "promax", "oblimin", "simplimax", "bentlerQ", "geominQ", "biquartimin" and "cluster"

For identification we need m^2 degrees of freedom. We get m(m+1)/2 from fixing factor variances to 1 and covariances to 0. We get another m(m-1)/2 degrees of freedom by fixing the upper-right hand corner of the factor loadings component of the A matrix at 0.

To aid optimization, manifest residual variances are `lbounded` at 0.

EFA reports standardized loadings: to do this, we scale the data.

note: Bear in mind that factor scores are indeterminate (can be rotated to an infinity of equivalent solutions).

Thanks to @ConorDolan for code implementing the rotation matrix and other suggestions!

Value

- EFA `mxModel()`

References

- https://github.com/tbates/umx,

Hendrickson, A. E. and White, P. O. (1964). Promax: a quick method for rotation to orthogonal oblique structure. *British Journal of Statistical Psychology*, **17**, 65–70. doi: 10.1111/j.2044-8317.1964.tb00244.x.

Kaiser, H. F. (1958). The varimax criterion for analytic rotation in factor analysis. *Psychometrika*, **23**, 187–200. doi: 10.1007/BF02289233.

See Also

- factanal(), mxFactorScores()

Other Super-easy helpers: umxMendelianRandomization(), umx

Examples

```
## Not run:
myVars <- c("mpg", "disp", "hp", "wt", "qsec")
m1 = umxEFA(mtcars[, myVars], factors =   2, rotation = "promax")
loadings(m1)

# Formula interface in base-R factanal
m2 = factanal(~ mpg + disp + hp + wt + qsec, factors = 2, rotation = "promax", data = mtcars)
loadings(m2)

# Formula interface in umxEFA
m2 = factanal(~ mpg + disp + hp + wt + qsec, factors = 2, rotation = "promax", data = mtcars)
loadings(m2)

# Return a loadings object
x = umxEFA(mtcars[, myVars], factors = 2, return = "loadings")
names(x)

m1 = umxEFA(myVars, factors = 2, data = mtcars, rotation = "promax")
m1 = umxEFA(name = "named", factors = "g", data = mtcars[, myVars])
m1 = umxEFA(name = "by_number", factors = 2, rotation = "promax", data = mtcars[, myVars])
x = umxEFA(name = "score", factors = "g", data = mtcars[, myVars], scores= "Regression")

## End(Not run)
```

---

umxEquate *umxEquate: Equate two or more paths*

Description

In addition to dropping or adding parameters, a second common task in modeling is to equate parameters. umx provides a convenience function to equate parameters by setting one or more parameters (the "slave" set) equal to one or more "master" parameters. These parameters are picked out via their labels, and setting two or more parameters to have the same value is accomplished by setting the slave(s) to have the same label(s) as the master parameters, thus constraining them to take the same value during model fitting.

Usage

```
umxEquate(
  model,
  master,
  slave,
  free = c(TRUE, FALSE, NA),
  verbose = FALSE,
  name = NULL,
  autoRun = FALSE,
  tryHard = c("no", "yes", "ordinal", "search"),
  comparison = TRUE
)
```

Arguments

model	An mxModel() within which to equate parameters
master	A list of "master" labels to which slave labels will be equated
slave	A list of slave labels which will be updated to match master labels, thus equating the parameters
free	Should parameter(s) initially be free? (default = TRUE)
verbose	Whether to give verbose feedback (default = TRUE)
name	name for the returned model (optional: Leave empty to leave name unchanged)
autoRun	Whether to run the model (default), or just to create it and return without running.
tryHard	Default ('no') uses normal mxRun. "yes" uses mxTryHard. Other options: "ordinal", "search"
comparison	Compare the new model to the old (if updating an existing model: default = TRUE)

Details

note: In addition to using this method to equating parameters, you can also equate one parameter to another by setting its label to the "square bracket" address of the master, e.g. "a[r,c]".

Tip: To find labels of free parameters use umxGetParameters() with free = TRUE

Tip: To find labels by name, use the regex parameter of umxGetParameters()

Value

- mxModel()

References

- https://www.github.com/tbates/umx

See Also

umxModify(), umxCompare()

Other Modify or Compare Models: umxFixAll(), umxMI(), umxModify(), umxSetParameters(), umxUnexplainedCausalNexus(), umx

Examples

```
require(umx)
data(demoOneFactor)
manifests = names(demoOneFactor)
m1 = umxRAM("One Factor", data = demoOneFactor, type = "cov",
umxPath("G", to = manifests),
umxPath(var = manifests),
umxPath(var = "G", fixedAt = 1)
)
# By default, umxEquate just equates master and slave labels
m2 = umxEquate(m1, master = "G_to_x1", slave = "G_to_x2", name = "Eq x1 x2 loadings")
# Set autoRun = TRUE and comparison = TRUE to run and output a comparison
m2 = umxEquate(m1, autoRun = TRUE, comparison = TRUE, name = "Eq x1 x2",
     master = "G_to_x1", slave = "G_to_x2"
)
```

---

umxExamples *Example code from Twin Research and Human Genetics Paper on umx*

Description

This is the example code used in our Twin Research and Human Genetics Paper on umx

Usage

```
umxExamples()
```

References

- Bates, T. C., Neale, M. C., & Maes, H. H. (2019). umx: A library for Structural Equation and Twin Modelling in R. *Twin Research and Human Genetics*, **22**, 27-41. DOI: https://doi.org/10.1017/thg.2019.2

See Also

- umx()

Examples

```
## Not run:

# ============================================================================
# = Example code from Twin Research and Human Genetics Paper on umx(model) =
# ============================================================================

# Installing umx can be done using the R-code:
install.packages("umx")
# load as usual
library("umx")

# The current package version can be shown with:
umxVersion("umx")

# Get the latest NPSOL and multi-core build of OpenMx
install.OpenMx("NPSOL")

# Bleeding edge version of OpenMx for MacOS
install.OpenMx("travis")

# ============
# = CFA Code =
# ============

# Load the umx library (this is assumed in subsequent examples
library("umx")

# Load demo data consisting of 5 correlated variables, x1:x5
data(demoOneFactor)

# Create a list of the manifest variables for use in specifying the model
manifests = paste0("x", 1:5) # 'x1', 'x2', ...'x5'

# Create model cfa1, with name 'CFA', data demoOneFactor, and the CFA paths.

cfa1 = umxRAM("CFA", data = demoOneFactor,
# Create latent variable 'G', with fixed variance of 1 and mean of 0
umxPath(v1m0 = "G"),
# Create 5 manifest variables, x1:x5, with free variance and mean
umxPath(v.m. = manifests),
# Create 1-headed paths from G to each of the manifests
umxPath("G", to = manifests)
)

# ====================
# = Parameter labels =
# ====================

x = umxLabel(mxMatrix(name="means", "Full", ncol = 2, nrow = 2))
x$labels
```

```
# ========
# = Plot =
# ========

plot(cfa1, means = FALSE, fixed = TRUE)
plot(cfa1, std = TRUE, digits = 3, resid= 'line')


m1 = umxRAM("play", data = c("A", "B", "C"),
umxPath(unique.pairs = c("A", "B", "C"))
)

# ================================================
# = Inspecting model parameters and residuals. =
# ================================================

# Show parameters, below .1, with label containing `x2'
parameters(cfa1, "above", .5, pattern= "x2")

residuals(cfa1, suppress = .005)

# ==================================
# = Modifying and comparing models =
# ==================================

# Variable names in the Duncan data
dimnames = c("RespOccAsp", "RespEduAsp", "RespParAsp", "RespIQ", "RespSES",
             "FrndOccAsp", "FrndEduAsp", "FrndParAsp", "FrndIQ", "FrndSES")
# lower-triangle of correlations among these variables
tmp = c(
0.6247,
0.2137, 0.2742,
0.4105, 0.4043, 0.1839,
0.3240, 0.4047, 0.0489, 0.2220,
0.3269, 0.3669, 0.1124, 0.2903, 0.3054,
0.4216, 0.3275, 0.0839, 0.2598, 0.2786, 0.6404,
0.0760, 0.0702, 0.1147, 0.1021, 0.0931, 0.2784, 0.1988,
0.2995, 0.2863, 0.0782, 0.3355, 0.2302, 0.5191, 0.5007,  0.2087,
0.2930, 0.2407, 0.0186, 0.1861, 0.2707, 0.4105, 0.3607, -0.0438, 0.2950
)

# Use the umx_lower2full function to create a full correlation matrix
duncanCov = umx_lower2full(tmp, diag = FALSE, dimnames = dimnames)

# Turn the duncan data into an mxData object for the model
duncanCov = mxData(duncanCov, type = "cov", numObs = 300)

respondentFormants   = c("RespSES", "FrndSES", "RespIQ", "RespParAsp")
friendFormants       = c("FrndSES", "RespSES", "FrndIQ", "FrndParAsp")
latentAspiration     = c("RespLatentAsp", "FrndLatentAsp")
respondentOutcomeAsp = c("RespOccAsp", "RespEduAsp")
friendOutcomeAsp     = c("FrndOccAsp", "FrndEduAsp")
```

```
duncan1 = umxRAM("Duncan", data = duncanCov,
# Working from the left of the model, as laid out in the figure, to right...

# 1. Add all distinct paths between variables to allow the
# exogenous manifests to covary with each other.
umxPath(unique.bivariate = c(friendFormants, respondentFormants)),

# 2. Add variances for the exogenous manifests,
# These are assumed to be error-free in this model,
# and are fixed at their known value).
umxPath(var = c(friendFormants, respondentFormants), fixedAt = 1),

# 3. Paths from IQ, SES, and parental aspiration
# to latent aspiration for Respondents:
umxPath(respondentFormants, to = "RespLatentAsp"),
# And same for friends
umxPath(friendFormants,     to = "FrndLatentAsp"),

# 4. Add residual variance for the two aspiration latent traits.
umxPath(var = latentAspiration),

# 5. Allow the latent traits each influence the other.
# This is done using fromEach, and the values are
# bounded to improve stability.
# note: Using one-label would equate these 2 influences
umxPath(fromEach = latentAspiration, lbound = 0, ubound = 1),

# 6. Allow latent aspiration to affect respondent's
# occupational & educational aspiration.
# note: firstAt = 1 is used to provide scale to the latent variables.
umxPath("RespLatentAsp", to = respondentOutcomeAsp, firstAt = 1),

# And their friends
umxPath("FrndLatentAsp", to = friendOutcomeAsp, firstAt = 1),

# 7. Finally, on the right hand side of figure, we add
# residual variance for the endogenous manifests.
umxPath(var = c(respondentOutcomeAsp, friendOutcomeAsp))
)

# ====================
# = Modifying models =
# ====================

# Collect a list of paths to drop
pathList = c("RespLatentAsp_to_FrndLatentAsp", "FrndLatentAsp_to_RespLatentAsp")

# Modify the model duncan1, requesting a comparison table:
duncan2 = umxModify(duncan1, update = pathList, name = "No_influence", comparison = TRUE)

# An example using regex, to drop all paths beginning "G_to_"
cfa2 = umxModify(cfa1, regex = "^G_to.*")
```

```
# ====================
# = Comparing models =
# ====================

umxCompare(duncan1, duncan2, report = "inline")

# To open the output as an html table in a browser, say:
umxCompare(duncan1, duncan2, report = "html")

# ============================
# = Equating model parameters =
# ============================

parameters(duncan1, pattern = "IQ_to_")


duncan3 = umxModify(duncan1, name = "Equate IQ effect", comparison = TRUE,
master = "RespIQ_to_RespLatentAsp",
update = "FrndIQ_to_FrndLatentAsp"
)

# ================
# = ACE examples =
# ================

require(umx);
# open the built in dataset of Australian height and weight twin data
data("twinData")
selDVs = c("wt")
dz = twinData[twinData$zygosity == "DZFF", ]
mz = twinData[twinData$zygosity == "MZFF", ]

ACE1 = umxACE(selDVs = selDVs, dzData = dz, mzData = mz, sep = "")
ACE2 = umxModify(ACE1, update = "c_r1c1", name = "dropC")
umxSummary(ACE1, std = FALSE, report = 'html', digits = 3, comparison = ACE2)
parameters(ACE1)

ACE2 = umxModify(ACE1, update = "c_r1c1", name = "dropC")

# ================================
# = Example Common Pathway model =
# ================================

# load twin data built into umx
data("twinData")

# Selecting the 'ht' and 'wt' variables
selDVs = c("ht", "wt")
mzData = subset(twinData, zygosity == "MZFF",)
dzData = subset(twinData, zygosity == "DZFF",)

# Run and report a common-pathway model
```

```
CP1 = umxCP(selDVs = selDVs, dzData = dzData, mzData = mzData, suffix = "")

paths = c("c_cp_r1c1", "cs_r1c1", "cs_r2c2")
CP2 = umxModify(CP1, update = paths, name = "dropC", comparison = TRUE)

CP2 = umxModify(CP1, regex = "(^cs_)|(^c_cp_)", name = "dropC")
umxSummary(CP2, comparison = CP1)

# ====================================
# = Example Gene x environment model =
# ====================================

data("twinData")
twinData$age1 = twinData$age2 = twinData$age
# Define the DV and definition variables
selDVs  = c("bmi1", "bmi2")
selDefs = c("age1", "age2")
selVars = c(selDVs, selDefs)

# Create datasets
mzData  = subset(twinData, zygosity == "MZFF")
dzData  = subset(twinData, zygosity == "DZFF")

# Build, run and report the GxE model using selected DV and moderator
# umxGxE will remove and report rows with missing data in definition variables.
GE1 = umxGxE(selDVs = selDVs, selDefs = selDefs,
  dzData = dzData, mzData = mzData,  dropMissingDef = TRUE)

# Shift the legend to the top right
umxSummary(GE1, location = "topright")

# plot standardized and raw output in separate graphs
umxSummary(GE1, separateGraphs = TRUE)

GE2 = umxModify(GE1, update = "am_r1c1", comparison = TRUE)
umxReduce(GE1)

# ================================
# = Example GxE windowed analysis =
# ================================

require(umx);
data("twinData")
mod     = "age"
selDVs  = c("bmi1", "bmi2")

# select the younger cohort of twins
tmpTwin = twinData[twinData$cohort == "younger", ]
# Drop twins with missing moderator
tmpTwin = tmpTwin[!is.na(tmpTwin[mod]), ]
mzData  = subset(tmpTwin, zygosity == "MZFF", c(selDVs, mod))
dzData  = subset(tmpTwin, zygosity == "DZFF", c(selDVs, mod))
# toggle autoplot off, so we don't plot every level of the moderator
```

```
umx_set_auto_plot(FALSE)
umxGxE_window(selDVs = selDVs, moderator = mod, mzData = mzData, dzData = dzData)
umx_set_auto_plot(TRUE)


## End(Not run)
```

---

umxExpCov *Get the expected vcov matrix*

Description

Extract the expected covariance matrix from an mxModel()

Usage

```
umxExpCov(object, latents = FALSE, manifests = TRUE, digits = NULL, ...)
```

Arguments

object	an mxModel() to get the covariance matrix from
latents	Whether to select the latent variables (defaults to TRUE)
manifests	Whether to select the manifest variables (defaults to TRUE)
digits	precision of reporting. NULL (Default) = no rounding.
...	extra parameters (to match vcov())

Value

- expected covariance matrix

References

- https://openmx.ssri.psu.edu/thread/2598 Original written by https://openmx.ssri.psu.edu/users/bwiernik

See Also

- umxRun(), umxCI_boot()

Other Reporting functions: RMSEA.MxModel(), RMSEA.summary.mxmodel(), RMSEA(), extractAIC.MxModel(), loadings(), residuals.MxModel(), umxCI_boot(), umxCI(), umxCompare(), umxConfint(), umxExpMeans(), umxFitIndices(), umxPlotACEv(), umxRotate(), umxSummary.MxModel()

Examples

```
require(umx)
data(demoOneFactor)
manifests = names(demoOneFactor)
m1 = umxRAM("One Factor", data = demoOneFactor, type = "cov",
umxPath("G", to = manifests),
umxPath(var = manifests),
umxPath(var = "G", fixedAt = 1)
)#'
vcov(m1) # supplied by OpenMx
umxExpCov(m1, digits = 3)
```

---

umxExpMeans *Extract the expected means matrix from an* mxModel()

Description

Extract the expected means matrix from an mxModel()

Usage

```
umxExpMeans(model, manifests = TRUE, latents = NULL, digits = NULL)
```

Arguments

model	an mxModel() to get the means from
manifests	Whether to select the manifest variables (defaults to TRUE)
latents	Whether to select the latent variables (defaults to TRUE)
digits	precision of reporting. Default (NULL) will not round at all.

Value

- expected means

References

- https://openmx.ssri.psu.edu/thread/2598

See Also

Other Reporting functions: RMSEA.MxModel(), RMSEA.summary.mxmodel(), RMSEA(), extractAIC.MxModel(), loadings(), residuals.MxModel(), umxCI_boot(), umxCI(), umxCompare(), umxConfint(), umxExpCov(), umxFitIndices(), umxPlotACEv(), umxRotate(), umxSummary.MxModel()

Examples

```
require(umx)
data(demoOneFactor)
manifests = names(demoOneFactor)
m1 = umxRAM("One Factor", data = demoOneFactor,
umxPath("G", to = manifests),
umxPath(var = manifests),
umxPath(var = "G", fixedAt = 1)
)

umxExpMeans(m1)
umxExpMeans(m1, digits = 3)
```

---

umxFactor *umxFactor*

Description

A convenient version of mxFactor() supporting the common case in which the factor levels are those in the variable.

Usage

```
umxFactor(
  x = character(),
  levels = NULL,
  labels = levels,
  exclude = NA,
  ordered = TRUE,
  collapse = FALSE,
  verbose = FALSE,
  sep = NA
)
```

Arguments

x	A variable to recode as an mxFactor (see mxFactor())
levels	(default NULL). Like factor() but UNLIKE mxFactor(), unique values will be used if levels not specified.
labels	= levels (see mxFactor())
exclude	= NA (see mxFactor())
ordered	= TRUE By default return an ordered mxFactor
collapse	= FALSE (see mxFactor())
verbose	Whether to tell user about such things as coercing to factor
sep	If twin data are being used, the string that separates the base from twin index will try and ensure factor levels same across all twins.

Value

- mxFactor()

References

- https://github.com/tbates/umx, https://tbates.github.io

See Also

- umxFactanal(), mxFactor()

Other Data Functions: umxHetCor(), umx_as_numeric(), umx_cont_2_quantiles(), umx_lower2full(), umx_make_MR_data(), umx_make_TwinData(), umx_make_fake_data(), umx_make_raw_from_cov(), umx_polychoric(), umx_polypairwise(), umx_polytriowise(), umx_read_lower(), umx_rename(), umx_reorder(), umx_select_valid(), umx_stack(), umx

Examples

```
umxFactor(letters)
umxFactor(letters, verbose = TRUE) # report coercions
umxFactor(letters, ordered = FALSE) # non-ordered factor like factor(x)
# Dataframe example:
x = umx_factor(mtcars[,c("cyl", "am")], ordered = FALSE); str(x)
# =================
# = Twin example: =
# =================
data(twinData)
tmp = twinData[, c("bmi1", "bmi2")]
tmp$bmi1[tmp$bmi1 <= 22] = 22
tmp$bmi2[tmp$bmi2 <= 22] = 22
# remember to factor _before_ breaking into MZ and DZ groups
x = umxFactor(tmp, sep = ""); str(x)
xmu_check_levels_identical(x, "bmi", sep="")

# Simple example to check behavior
x = round(10 * rnorm(1000, mean = -.2))
y = round(5 * rnorm(1000))
x[x < 0] = 0; y[y < 0] = 0
jnk = umxFactor(x); str(jnk)
df  = data.frame(x = x, y = y)
jnk = umxFactor(df); str(jnk)
```

---

umxFactorScores *Return factor scores from a model as an easily consumable dataframe.*

Description

umxFactorScores takes a model, and computes factors scores using the selected method (one of 'ML', 'WeightedML', or 'Regression') It is a simple wrapper around mxFactorScores. For missing data, you must specify the least number of variables allowed for a score (subjects with fewer than minManifests will return a score of NA.

Usage

```
umxFactorScores(
  model,
  type = c("ML", "WeightedML", "Regression"),
  minManifests = NA,
  return = c("Scores", "StandardErrors")
)
```

Arguments

model	The model from which to generate scores.
type	Method of computing the score ('ML', 'WeightedML', or 'Regression').
minManifests	The minimum number of variables not NA to return a score for a participant (Default = ask).
return	What to return (defaults to "Scores", which is what most users want, but can return "StandardErrors" on each score.

Value

- dataframe of scores.

References

- https://github.com/tbates/umx, https://tbates.github.io

See Also

- mxFactorScores()

Other Reporting Functions: loadings.MxModel(), umxAPA(), umxGetParameters(), umxParameters(), umxReduce(), umx_aggregate(), umx_names(), umx_time(), umx

Examples

```
m1 = umxEFA(mtcars, factors = 2)
x = umxFactorScores(m1, type = 'Regression', minManifests = 3)

# ============================================================================
# = histogram of F1 and plot of F1 against F2 showing they are orthogonal =
# ============================================================================
hist(x$F1)
plot(F1 ~ F2, data = x)

## Not run:
m1 = umxEFA(mtcars, factors = 1)
x = umxFactorScores(m1, type = 'Regression', minManifests = 3)
x

## End(Not run)
```

umxFitIndices *Get additional fit-indices for a model with umxFitIndices*

Description

Computes a variety of fit indices. Originated in this thread: http://openmx.ssri.psu.edu/thread/765

Usage

```
umxFitIndices(model, ...)
```

Arguments

model	The mxModel for which you want fit indices.
...	Additional parameters passed to summary.MxModel.

Details

Note: This function is currently not robust across multi-group designs or definition variables. It is designed to provide residual-based fit indices (SRMR, CRMR, SMAR, CMAR, etc.) and less-often reported fit indices where Reviewer 2 wants something other than CFA/TLI/RMSEA.

Fit information reported includes:

Model characteristics: numObs, estimated parameters, observed statistics, observed summary statistics, -2*log(Likelihood), degrees of freedom

Chi-squared test: Chi, ChiDoF, p (of Chi), ChiPerDoF,

Noncentrality-based indices: RMSEA, RMSEACI, RMSEANull, RMSEAClose (p value), independenceRMSEA, NCP, NCPCI, F0, F0CI, Mc (aka NCI, MFI)

Comparative fit indices: TLI (aka NNFI), CFI, IFI, PRATIO, PCFI

Residual-based indices: RMR, SRMR, SRMR_mplus, CRMR, MAR, SMAR, SMAR_mplus, CMAR

Information-theory criteria (computed using chi-square or -2LL; df or parameters penalties) AIC, AICc, BIC, SABIC, CAIC, BCC ECVI, ECVICI, MECVI, MECVICI

LISREL and other early fit indices (we recommend not reporting these) GFI, AGFI, PGFI, GH, NFI, PNFI, RFI

Want more? *Open an Issue* at GitHub.

Value

List of fit statistics

Author(s)

Brenton M. Wiernik, Athanassios Protopapas, Paolo Ghisletta, Markus Brauer

See Also

Other Reporting functions: RMSEA.MxModel(), RMSEA.summary.mxmodel(), RMSEA(), extractAIC.MxModel(), loadings(), residuals.MxModel(), umxCI_boot(), umxCI(), umxCompare(), umxConfint(), umxExpCov(), umxExpMeans(), umxPlotACEv(), umxRotate(), umxSummary.MxModel()

Examples

```
## Not run:
library(umx)
data(demoOneFactor)
latents = c("G")
manifests = names(demoOneFactor)
m1 = umxRAM("One Factor",
data = mxData(cov(demoOneFactor), type = "cov", numObs = 500),
umxPath(latents, to = manifests),
umxPath(var = manifests),
umxPath(var = latents, fixedAt = 1)
)
umxFitIndices(m1)

# And with raw data
m2 = umxRAM("m1", data = demoOneFactor,
umxPath(latents, to = manifests),
umxPath(v.m. = manifests),
umxPath(v1m0 = latents)
)
umxFitIndices(m1, refModels = mxRefModels(m2, run = TRUE))

## End(Not run)
```

---

umxFixAll *umxFixAll: Fix all free parameters*

Description

Fix all free parameters in a model using omxGetParameters()

Usage

```
umxFixAll(model, name = "_fixed", run = FALSE, verbose = FALSE)
```

Arguments

model	an mxModel() within which to fix free parameters
name	optional new name for the model. if you begin with a _ it will be made a suffix
run	whether to fix and re-run the model, or just return it (defaults to FALSE)
verbose	whether to mention how many paths were fixed (default is FALSE)

Value

- the fixed mxModel()

References

- https://tbates.github.io, https://github.com/tbates/umx

See Also

Other Modify or Compare Models: umxEquate(), umxMI(), umxModify(), umxSetParameters(), umxUnexplainedCausalNexus(), umx

Examples

```
require(umx)
data(demoOneFactor)
manifests = names(demoOneFactor)

m1 = umxRAM("OneFactor", data = demoOneFactor, type = "cov",
umxPath("G", to = manifests),
umxPath(var = manifests),
umxPath(var = "G", fixedAt = 1)
)
m2 = umxFixAll(m1, run = TRUE, verbose = TRUE)
mxCompare(m1, m2)
```

---

umxGetParameters *Get parameters from a model, with support for pattern matching!*

Description

umxGetParameters retrieves parameter labels from a model, like omxGetParameters(). However, it is supercharged with regular expressions, so you can get labels that match a pattern.

Usage

```
umxGetParameters(
  inputTarget,
  regex = NA,
  free = NA,
  fetch = c("values", "free", "lbound", "ubound", "all"),
  verbose = FALSE
)
```

Arguments

inputTarget	An object to get parameters from: could be a RAM mxModel()
regex	A regular expression to filter the labels. Default (NA) returns all labels. If vector, treated as raw labels to find.
free	A Boolean determining whether to return only free parameters.
fetch	What to return: "values" (default) or "free", "lbound", "ubound", or "all"
verbose	How much feedback to give

Details

In addition, if regex contains a vector, this is treated as a list of raw labels to search for, and return if all are found. *note*: To return all labels, just leave regex as is.

References

- https://www.github.com/tbates/umx

See Also

omxGetParameters(), parameters()

Other Reporting Functions: loadings.MxModel(), umxAPA(), umxFactorScores(), umxParameters(), umxReduce(), umx_aggregate(), umx_names(), umx_time(), umx

Examples

```
require(umx)
data(demoOneFactor)
manifests = names(demoOneFactor)
m1 = umxRAM("One Factor", data = demoOneFactor, type = "cov",
umxPath("G", to = manifests),
umxPath(var = manifests),
umxPath(var = "G", fixedAt = 1)
)

# Show all parameters
umxGetParameters(m1)
umxGetParameters(m1, free = TRUE)  # Only free parameters
umxGetParameters(m1, free = FALSE) # Only fixed parameters
# Complex regex pattern
umxGetParameters(m1, regex = "x[1-3]_with_x[2-5]", free = TRUE)
```

umxGxE *umxGxE: Implements ACE models with moderation of paths, e.g. by SES.*

Description

Make a 2-group GxE (moderated ACE) model (Purcell, 2002). GxE interaction studies test the hypothesis that the strength of genetic (or environmental) influence varies parametrically (usually linear effects on path estimates) across levels of environment. umxGxE allows detecting, testing, and visualizing G xE (or C or E x E) interaction forms.

Usage

```
umxGxE(
  name = "G_by_E",
  selDVs,
  selDefs,
  dzData,
  mzData,
  sep = NULL,
  data = NULL,
  zyg = "zygosity",
  digits = 3,
  lboundACE = NA,
  lboundM = NA,
  dropMissingDef = TRUE,
  dzAr = 0.5,
  dzCr = 1,
  autoRun = getOption("umx_auto_run"),
  tryHard = c("no", "yes", "ordinal", "search"),
  optimizer = NULL
)
```

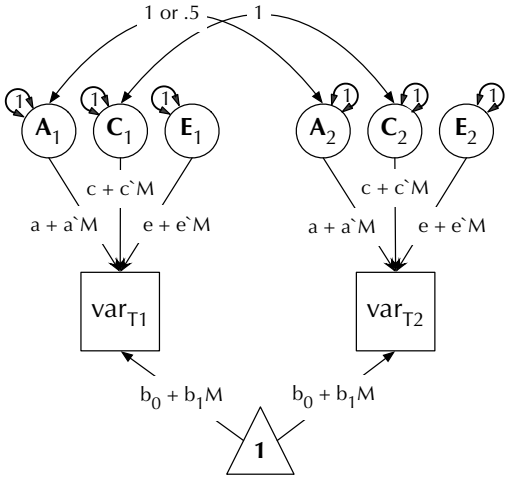
Arguments

name	The name of the model (default= "G_by_E")
selDVs	The dependent variable (e.g. "IQ")
selDefs	The definition variable (e.g. "SES")
dzData	The DZ dataframe containing the Twin 1 and Twin 2 DV and moderator (4 columns)
mzData	The MZ dataframe containing the Twin 1 and Twin 2 DV and moderator (4 columns)
sep	How to expand selDVs into full names, i.e., "_T" makes "var" -> "var_T1" and "var_T2"
data	If provided, dzData and mzData are treated as valid levels of zyg to select() data sets (default = NULL)

zyg	If data provided, this column is used to select rows by zygosity (Default = "zygosity")
digits	Rounding precision for tables (default 3)
lboundACE	If not NA, then lbound the main effects at this value (default = NA, can help to set this to 0)
lboundM	If not NA, then lbound the moderator effects at this value (default = NA, can help to set this to 0)
dropMissingDef	Whether to automatically drop missing def var rows for the user (default = TRUE). You get a polite note.
dzAr	The DZ genetic correlation (defaults to .5, vary to examine assortative mating).
dzCr	The DZ "C" correlation (defaults to 1: set to .25 to make an ADE model).
autoRun	Optionally run the model (default), or just to create it and return without running.
tryHard	Optionally tryHard to get the model to converge (Default = 'no'). "yes" uses mxTryHard. Other options: "ordinal", "search".
optimizer	Optionally set the optimizer (default NULL does nothing)

Details

The following figure the GxE model as a path diagram:



Value

- GxE `mxModel()`

References

- Purcell, S. (2002). Variance components models for gene-environment interaction in twin analysis. *Twin Research*, **6**, 554-571. DOI: 10.1375/twin.5.6.554

See Also

umxGxE_window(), umxReduce(), umxSummary()

Other Twin Modeling Functions: plot.MxModelTwinMaker(), power.ACE.test(), umxACEcov(), umxACEv(), umxACE(), umxCP(), umxDoCp(), umxDoC(), umxGxE_window(), umxGxEbiv(), umxIP(), umxRotate.MxModelCP(), umxSexLim(), umxSimplex(), umx

Examples

```
require(umx)
data(twinData)
umx_set_optimizer("SLSQP")
twinData$age1 = twinData$age2 = twinData$age
selDVs  = "bmi"
selDefs = "age"
mzData  = subset(twinData, zygosity == "MZFF")[1:100,]
dzData  = subset(twinData, zygosity == "DZFF")[1:100,]
m1 = umxGxE(selDVs= "bmi", selDefs= "age", sep= "", dzData= dzData, mzData= mzData, tryHard= "yes")

## Not run:
# Select the data on the fly with data= and zygosity levels
m1 = umxGxE(selDVs= "bmi", selDefs= "age", sep="", dzData= "DZFF", mzData= "MZFF", data= twinData)

# ================================================================
# = example with Twins having different values of the moderator =
# ================================================================

twinData$age1 = twinData$age2 = twinData$age
tmp = twinData
tmp$age2 = tmp$age2 +rnorm(n=length(tmp$age2))
selDVs  = "bmi"
selDefs = "age"
mzData = subset(tmp, zygosity == "MZFF")
dzData = subset(tmp, zygosity == "DZFF")
m1 = umxGxE(selDVs= "bmi", selDefs= "age", sep= "", dzData= dzData, mzData= mzData, tryHard= "yes")

# ===================================
# = Controlling output of umxSummary =
# ===================================
umxSummaryGxE(m1)
umxSummary(m1, location = "topright")
umxSummary(m1, separateGraphs = TRUE)

m2 = umxModify(m1, regex = "am_.*", comparison = TRUE, tryHard = "yes")

# umxReduce knows how to test all relevant hypotheses for GxE models,
# reporting these in a nice table.
umxReduce(m1)

## End(Not run)
```

umxGxEbiv *Purcell (2002) Bivariate GxE model: Suitable when twins differ on the moderator.*

Description

GxE interaction models test the hypothesis that the strength of genetic and environmental influences vary parametrically across levels of a measured environment.

Usage

```
umxGxEbiv(
  name = "GxEbiv",
  selDVs,
  selDefs,
  dzData,
  mzData,
  sep = NULL,
  lboundACE = NA,
  lboundM = NA,
  dropMissingDef = FALSE,
  autoRun = getOption("umx_auto_run"),
  tryHard = c("no", "yes", "ordinal", "search"),
  optimizer = NULL
)
```

Arguments

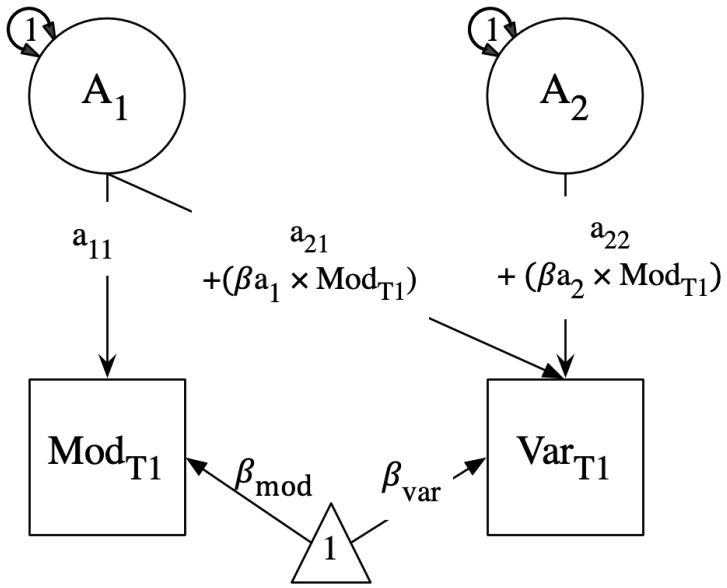
name	The name of the model (defaults to "GxEbiv")
selDVs	The dependent variable (e.g. IQ)
selDefs	The definition variable (e.g. socioeconomic status)
dzData	The DZ dataframe containing the Twin 1 and Twin 2 DV and moderator (4 columns)
mzData	The MZ dataframe containing the Twin 1 and Twin 2 DV and moderator (4 columns)
sep	Expand variable base names, i.e., "_T" makes var -> var_T1 and var_T2
lboundACE	If !NA, then lbound the main effects at this value (default = NA)
lboundM	If !NA, then lbound the moderators at this value (default = NA)
dropMissingDef	Whether to automatically drop missing def var rows for the user (gives a warning) default = FALSE
autoRun	Whether to run the model (default), or just to create it and return without running.
tryHard	Default ('no') uses normal mxRun. "yes" uses mxTryHard. Other options: "ordinal", "search"
optimizer	Optionally set the optimizer (default NULL does nothing)

Details

Whereas univariate `umxGxE()` models assume the twins share the moderator, or have zero correlation on the moderator, `umxGxEbiv()` allows testing moderation in cases where members of a twin pair differ on the moderator, (Purcell, 2002; van der Sluis et al., 2012).

This is the same model we teach at Boulder.

The following figure shows this bivariate GxE model as a path diagram (Twin 1 shown). Whereas the univariate model incorporates the moderator in the means model, the bivariate model incorporates the moderator as a first class variable, with its own ACE structure, shared pathways to the trait of interest, and the ability to moderate both specific and shared A, C, and E, influences on the trait of interest.



Twin 1 and twin 2 A, C, and E latent traits are connected in the standard fashion, with the covariance of the T1 and T2 latent genetic traits set to .5 for DZ and 1.0 for MZ pairs. For the sake of clarity, C, and E paths are omitted here. These mirror those for A.

Value

- GxEbiv `mxModel()`

References

- Purcell, S. (2002). Variance components models for gene-environment interaction in twin analysis. *Twin Research*, **6**, 554-571. doi:10.1375/twin.5.6.554.
- van der Sluis, S., Posthuma, D., & Dolan, C. V. (2012). A note on false positives and power in G x E modelling of twin data. *Behavior Genetics*, **42**, 170-186. doi:10.1007/s10519-011-9480-3.

See Also

- plot(), umxSummary(), umxReduce()

Other Twin Modeling Functions: plot.MxModelTwinMaker(), power.ACE.test(), umxACEcov(), umxACEv(), umxACE(), umxCP(), umxDoCp(), umxDoC(), umxGxE_window(), umxGxE(), umxIP(), umxRotate.MxModelCP(), umxSexLim(), umxSimplex(), umx

Examples

```
require(umx)
data(twinData)
selDVs  = "wt"
selDefs = "ht"
df = umx_scale_wide_twin_data(twinData, varsToScale = c("ht", "wt"), sep = "")
mzData  = subset(df, zygosity %in%  c("MZFF", "MZMM"))
dzData  = subset(df, zygosity %in%  c("DZFF", "DZMM", "DZOS"))

## Not run:
m1 = umxGxEbiv(selDVs = selDVs, selDefs = selDefs,
dzData = dzData, mzData = mzData, sep = "", dropMissingDef = TRUE)

# Plot Moderation
umxSummaryGxEbiv(m1)
umxSummary(m1, location = "topright")
umxSummary(m1, separateGraphs = FALSE)
m2 = umxModify(m1, update = c("cBeta2_r1c1", "eBeta1_r1c1", "eBeta2_r1c1"), comparison = TRUE)
#
# TODO: teach umxReduce to test all relevant hypotheses for umxGxEbiv
umxReduce(m1)

## End(Not run)
```

---

umxGxE_window *Implement the moving-window form of GxE analysis.*

Description

Make a 2-group GxE (moderated ACE) model using LOSEM. In GxE interaction studies, typically, the hypothesis that the strength of genetic influence varies parametrically (usually linear effects on path estimates) across levels of environment. Of course, the function linking genetic influence and context is not necessarily linear, but may react more steeply at the extremes, or take other, unknown forms. To avoid obscuring the underlying shape of the interaction effect, local structural equation modeling (LOSEM) may be used, and GxE_window implements this. LOSEM is a non-parametric, estimating latent interaction effects across the range of a measured moderator using a windowing function which is walked along the context dimension, and which weights subjects near the center of the window highly relative to subjects far above or below the window center. This allows detecting and visualizing arbitrary GxE (or CxE or ExE) interaction forms.

Usage

```
umxGxE_window(
  selDVs = NULL,
  moderator = NULL,
  mzData = mzData,
  dzData = dzData,
  sep = NULL,
  weightCov = FALSE,
  target = NULL,
  width = 1,
  plotWindow = FALSE,
  return = c("estimates", "last_model")
)
```

Arguments

selDVs	The dependent variables for T1 and T2, e.g. c("bmi_T1", "bmi_T2")
moderator	The name of the moderator variable in the dataset e.g. "age", "SES" etc.
mzData	Dataframe containing the DV and moderator for MZ twins
dzData	Dataframe containing the DV and moderator for DZ twins
sep	(optional) separator, e.g. "_T" which will be used expand base names into full variable names: e.g.: 'bmi' –> c("bmi_T1", "bmi_T2")
weightCov	Whether to use cov.wt matrices or FIML default = FALSE, i.e., FIML
target	A user-selected list of moderator values to test (default = NULL = explore the full range)
width	An option to widen or narrow the window from its default (of 1)
plotWindow	whether to plot what the window looks like
return	whether to return the last model (useful for specifiedTargets) or the list of estimates (default = "estimates")

Value

- Table of estimates of ACE along the moderator

References

- Hildebrandt, A., Wilhelm, O, & Robitzsch, A. (2009) Complementary and competing factor analytic approaches for the investigation of measurement invariance. *Review of Psychology*, **16**, 87–107.

Briley, D.A., Harden, K.P., Bates, T.C., Tucker-Drob, E.M. (2015). Nonparametric Estimates of Gene x Environment Interaction Using Local Structural Equation Modeling. *Behavior Genetics*, **45**, 581-96. doi 10.1007/s10519-015-9732-8

See Also

umxGxE()

Other Twin Modeling Functions: plot.MxModelTwinMaker(), power.ACE.test(), umxACEcov(), umxACEv(), umxACE(), umxCP(), umxDoCp(), umxDoC(), umxGxEbiv(), umxGxE(), umxIP(), umxRotate.MxModelCP(), umxSexLim(), umxSimplex(), umx

Examples

```
library(umx);
# ==============================
# = 1. Open and clean the data =
# ==============================
# umxGxE_window takes a data.frame consisting of a moderator and two DV columns: one for each twin.
# The model assumes two groups (MZ and DZ). Moderator can't be missing
mod = "age" # The full name of the moderator column in the dataset
selDVs = c("bmi1", "bmi2") # The DV for twin 1 and twin 2
data(twinData) # Dataset of Australian twins, built into OpenMx
# The twinData consist of two cohorts: "younger" and "older".
# zygosity is a factor. levels =  MZFF, MZMM, DZFF, DZMM, DZOS.
# Delete missing moderator rows
twinData = twinData[!is.na(twinData[mod]), ]
mzData = subset(twinData, zygosity == "MZFF", c(selDVs, mod))
dzData = subset(twinData, zygosity == "DZFF", c(selDVs, mod))

# ========================
# = 2. Run the analyses! =
# ========================
# Run and plot for specified windows (in this case just 1927)
umxGxE_window(selDVs = selDVs, moderator = mod, mzData = mzData, dzData = dzData,
target = 40, plotWindow = TRUE)

## Not run:
# Run with FIML (default) uses all information
umxGxE_window(selDVs = "bmi", sep="", moderator = "age", mzData = mzData, dzData = dzData)

# Run creating weighted covariance matrices (excludes missing data)
umxGxE_window(selDVs = "bmi", sep="", moderator= "age", mzData = mzData, dzData = dzData,
weightCov = TRUE)

## End(Not run)
```

---

umxHetCor *Create a matrix of correlations for variables of diverse types (binary, ordinal, continuous)*

Description

umxHetCor is a helper to:

1. return just the correlations from John Fox's polycor::hetcor function
2. If you give it a covariance matrix, return the nearest positive-definite correlation matrix.

Usage

```
umxHetCor(
  data,
  ML = FALSE,
  use = c("pairwise.complete.obs", "complete.obs"),
  treatAllAsFactor = FALSE,
  verbose = FALSE,
  return = c("correlations", "hetcor object"),
  std.err = FALSE
)
```

Arguments

data	A data.frame() of columns for which to compute heterochoric correlations. OR an existing covariance matrix.
ML	Whether to use Maximum likelihood computation of correlations (default = FALSE)
use	How to handle missing data: Default= "pairwise.complete.obs". Alternative ="complete.obs".
treatAllAsFactor	Whether to treat all columns as factors, whether they are or not (Default = FALSE)
verbose	How much to tell the user about what was done.
return	Return just the correlations (default) or the hetcor object (contains, method, SEs etc.)
std.err	Compute the SEs? (default = FALSE)

Value

- A matrix of correlations

See Also

Other Data Functions: umxFactor(), umx_as_numeric(), umx_cont_2_quantiles(), umx_lower2full(), umx_make_MR_data(), umx_make_TwinData(), umx_make_fake_data(), umx_make_raw_from_cov(), umx_polychoric(), umx_polypairwise(), umx_polytriowise(), umx_read_lower(), umx_rename(), umx_reorder(), umx_select_valid(), umx_stack(), umx

Other Miscellaneous Stats Helpers: FishersMethod(), SE_from_p(), oddsratio(), reliability(), umxCov2cor(), umxWeightedAIC(), umx_apply(), umx_cor(), umx_means(), umx_r_test(), umx_round(), umx_scale(), umx_var(), umx

Examples

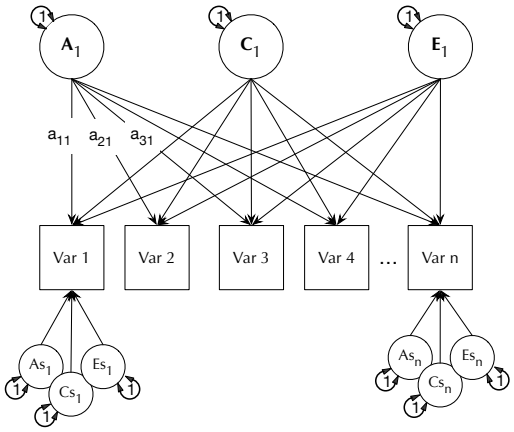
```
umxHetCor(mtcars[,c("mpg", "am")])
umxHetCor(mtcars[,c("mpg", "am")], treatAllAsFactor = TRUE, verbose = TRUE)
```

---

umxIP *umxIP: Build and run an Independent pathway twin model*

Description

Make a 2-group Independent Pathway twin model (Common-factor independent-pathway multivariate model). The following figure shows the IP model diagrammatically:



As can be seen, each phenotype also by default has A, C, and E influences specific to that phenotype.

Features of the model include the ability to include add more one set of independent pathways, different numbers of pathways for a, c, and e, as well the ability to use ordinal data, and different fit functions, e.g. WLS.

note: The function `umx_set_mvn_optimization_options()` allows users to see and set `mvnRelEps` and `mvnMaxPointsA` mvnRelEps defaults to .005. For ordinal models, you might find that '0.01' works better.

Usage

```
umxIP(
  name = "IP",
  selDVs,
  dzData,
  mzData,
  sep = NULL,
  nFac = c(a = 1, c = 1, e = 1),
  data = NULL,
  zyg = "zygosity",
  type = c("Auto", "FIML", "cov", "cor", "WLS", "DWLS", "ULS"),
```

```
  allContinuousMethod = c("cumulants", "marginals"),
  dzAr = 0.5,
  dzCr = 1,
  correlatedA = FALSE,
  numObsDZ = NULL,
  numObsMZ = NULL,
  autoRun = getOption("umx_auto_run"),
  tryHard = c("no", "yes", "ordinal", "search"),
  optimizer = NULL,
  equateMeans = TRUE,
  weightVar = NULL,
  addStd = TRUE,
  addCI = TRUE,
  freeLowerA = FALSE,
  freeLowerC = FALSE,
  freeLowerE = FALSE
)
```

Arguments

name	The name of the model (defaults to "IP").
selDVs	The base names of the variables to model. note: Omit suffixes - just "dep" not c("dep_T1", "dep_T2")
dzData	The DZ dataframe.
mzData	The MZ dataframe.
sep	The suffix for twin 1 and twin 2. e.g. selDVs= "dep", sep= "_T" -> c("dep_T1", "dep_T2")
nFac	How many common factors for a, c, and e. If one number is given, applies to all three.
data	If provided, dzData and mzData are treated as levels of zyg to select() MZ and DZ data sets (default = NULL)
zyg	If data provided, this column is used to select rows by zygosity (Default = "zygosity")
type	Analysis method one of c("Auto", "FIML", "cov", "cor", "WLS", "DWLS", "ULS")
allContinuousMethod	"cumulants" or "marginals". Used in all-continuous WLS data to determine if a means model needed.
dzAr	The DZ genetic correlation (defaults to .5, vary to examine assortative mating).
dzCr	The DZ "C" correlation (defaults to 1: set to .25 to make an ADE model).
correlatedA	Whether factors are allowed to correlate (not implemented yet: FALSE).
numObsDZ	= For cov data, the number of DZ pairs.
numObsMZ	= For cov data, the number of MZ pairs.
autoRun	Whether to run and return the model (default), or just to create and return without running.

tryHard	Whether to tryHard (default 'no' uses normal mxRun). options: "mxTryHard", "mxTryHardOrdinal", or "mxTryHardWideSearch"
optimizer	optionally set the optimizer (default NULL does nothing).
equateMeans	Whether to equate the means across twins (defaults to TRUE).
weightVar	If a weighting variable is provided, a vector objective will be used to weight the data. (default = NULL).
addStd	Whether to add algebras for a standardized model (defaults to TRUE).
addCI	Whether to add CIs (defaults to TRUE).
freeLowerA	ignore: Whether to leave the lower triangle of A free (default = FALSE).
freeLowerC	ignore: Whether to leave the lower triangle of C free (default = FALSE).
freeLowerE	ignore: Whether to leave the lower triangle of E free (default = FALSE).

Details

Like the umxACE() model, the CP model decomposes phenotypic variance into Additive genetic, unique environmental (E) and, optionally, either common or shared-environment (C) or non-additive genetic effects (D).

Unlike the Cholesky, these factors do not act directly on the phenotype. Instead latent A, C, and E influences impact on one or more latent common factors which, in turn, account for variance in the phenotypes (see Figure).

Data Input Currently, umxIP accepts only raw data. This may change in future versions. You can choose other fit functions, e.g. WLS.

Ordinal Data

In an important capability, the model transparently handles ordinal (binary or multi-level ordered factor data) inputs, and can handle mixtures of continuous, binary, and ordinal data in any combination.

Additional features

umxIP supports varying the DZ genetic association (defaulting to .5) to allow exploring assortative mating effects, as well as varying the DZ "C" factor from 1 (the default for modeling family-level effects shared 100% by twins in a pair), to .25 to model dominance effects.

Matrices and Labels in IP model

A good way to see which matrices are used in umxIP is to run an example model and plot it.

All the shared matrices are in the model "top".

Matrices as, cs, and es contain the path loadings specific to each variable on their diagonals.

To see the 'as' values, you can simply execute:

m1$top#as$values

m1$top#as$labels

m1$top#as$free

Labels relevant to modifying the specific loadings take the form "as_r1c1", "as_r2c2" etc.

The independent-pathway loadings on the manifests are in matrices a_ip, c_ip, e_ip.

Less commonly-modified matrices are the mean matrix expMean. This has 1 row, and the columns are laid out for each variable for twin 1, followed by each variable for twin 2.

So, in a model where the means for twin 1 and twin 2 had been equated (set = to T1), you could make them independent again with this line:

`m1$top$expMean$labels[1,4:6] = c("expMean_r1c4","expMean_r1c5","expMean_r1c6")`

Value

- mxModel()

References

- https://www.github.com/tbates/umx

See Also

- plot(), umxSummary() work for IP, CP, GxE, SAT, and ACE models.

Other Twin Modeling Functions: plot.MxModelTwinMaker(), power.ACE.test(), umxACEcov(), umxACEv(), umxACE(), umxCP(), umxDoCp(), umxDoC(), umxGxE_window(), umxGxEbiv(), umxGxE(), umxRotate.MxModelCP(), umxSexLim(), umxSimplex(), umx

Examples

```
## Not run:
require(umx)
data(GFF)
mzData <- subset(GFF, zyg_2grp == "MZ")
dzData <- subset(GFF, zyg_2grp == "DZ")
selDVs = c("gff","fc","qol","hap","sat","AD") # These will be expanded into "gff_T1" "gff_T2" etc.
m1 =    umxIP(selDVs = selDVs, sep = "_T", dzData = dzData, mzData = mzData)

# WLS example: Use "marginals" method to enable all continuous data with missingness.
m3 = umxIP(selDVs = selDVs, sep = "_T", dzData = dzData, mzData = mzData,
type = "DWLS", allContinuousMethod='marginals')
# omit missing to enable default WLS method to work on all continuous data
dzD = na.omit(dzData[, tvars(selDVs, "_T")])
mzD = na.omit(dzData[, tvars(selDVs, "_T")])
m4 = umxIP(selDVs = selDVs, sep = "_T", dzData = dzD, mzData = mzD, type = "DWLS")

# ======================================================================
# = Try with a non-default number of a, c, and e independent factors =
# ======================================================================
nFac = c(a = 2, c = 1, e = 1)
m2 = umxIP(selDVs = selDVs, sep = "_T", dzData = dzData, mzData = mzData, nFac = nFac,
tryHard = "yes")
umxCompare(m1, m2)

## End(Not run)
```

umxJiggle *umxJiggle*

Description

umxJiggle takes values in a matrix and jiggles them

Usage

```
umxJiggle(matrixIn, mean = 0, sd = 0.1, dontTouch = 0)
```

Arguments

matrixIn	an mxMatrix() to jiggle the values of
mean	the mean value to add to each value
sd	the sd of the jiggle noise
dontTouch	A value, which, if found, will be left as-is (defaults to 0)

Value

- mxMatrix()

References

- https://www.github.com/tbates/umx

See Also

Other Advanced Model Building Functions: umxLabel(), umxThresholdMatrix(), umxValues(), umx

Examples

```
## Not run:
mat1 = umxJiggle(mat1)

## End(Not run)
```

umxLabel *umxLabel: Add labels to a RAM model, matrix, or path*

Description

umxLabel adds labels to things, be it an: mxModel() (RAM or matrix based), an mxPath(), or an mxMatrix() This is a core function in umx: Adding labels to paths opens the door to umxEquate(), as well as omxSetParameters()

Usage

```
umxLabel(
  obj,
  suffix = "",
  baseName = NA,
  setfree = FALSE,
  drop = 0,
  labelFixedCells = TRUE,
  jiggle = NA,
  boundDiag = NA,
  verbose = FALSE,
  overRideExisting = FALSE,
  name = NULL
)
```

Arguments

obj	An mxModel() (RAM or matrix based), mxPath(), or mxMatrix()
suffix	String to append to each label (might be used to distinguish, say male and female submodels in a model)
baseName	String to prepend to labels. Defaults to NA ("")
setfree	Whether to label only the free paths (defaults to FALSE)
drop	The value to fix "drop" paths to (defaults to 0)
labelFixedCells	= TRUE
jiggle	How much to jiggle values in a matrix or list of path values
boundDiag	Whether to bound the diagonal of a matrix
verbose	How much feedback to give the user (default = FALSE)
overRideExisting	= FALSE
name	Optional new name if given a model. Default (NULL) does not rename model.

Value

- mxModel()

References

- https://www.github.com/tbates/umx

See Also

Other Advanced Model Building Functions: umxJiggle(), umxThresholdMatrix(), umxValues(), umx

Examples

```
# ============================================================
# = Show how OpenMx models are not labeled, and then add labels =
# ============================================================
require(umx)
data(demoOneFactor)
latents  = c("G")
manifests = names(demoOneFactor)
m1 = mxModel("One Factor", type = "RAM",
manifestVars = manifests, latentVars = latents,
mxPath(from = latents  , to = manifests),
mxPath(from = manifests, arrows = 2),
mxPath(from = latents  , arrows = 2, free = FALSE, values = 1.0),
mxData(cov(demoOneFactor), type = "cov", numObs=500)
)

umxGetParameters(m1) # Default "matrix address" labels, i.e "One Factor.S[2,2]"
m1 = umxLabel(m1)
umxGetParameters(m1, free = TRUE) # Informative labels: "G_to_x1", "x4_with_x4", etc.

# ============================================================================
# = Create a new model, with suffixes added to paths, and model renamed =
# ============================================================================
m2 = umxLabel(m1, suffix= "_male", overRideExisting= TRUE, name = "male")
umxGetParameters(m2, free = TRUE) # suffixes added

# ============================
# = Example Labeling a matrix =
# ============================
a = umxLabel(mxMatrix(name = "a", "Full", 3, 3, values = 1:9))
a$labels
a = umxLabel(mxMatrix(name = "a", "Full", 3, 3, values = 1:9), baseName="bob")
a$labels
# note: labels with "data." in the name are left untouched!
a = mxMatrix(name = "a", "Full", 1,3, labels = c("data.a", "test", NA))
a$labels
umxLabel(a, verbose = TRUE)
umxLabel(a, verbose = TRUE, overRideExisting = FALSE)
umxLabel(a, verbose = TRUE, overRideExisting = TRUE)
```

umxLav2RAM *Convert lavaan string to a umxRAM model*

Description

Takes a lavaan syntax string and creates the matching one or more umxRAM() models.

If data are provided, a umxRAM() model is returned.

If more than one group is found, a umxSuperModel() is returned.

This function is at the alpha quality stage, and **should be expected to have bugs**. Several features are not yet supported. Let me know if you would like them.

Usage

```
umxLav2RAM(
  model = NA,
  data = "auto",
  group = NULL,
  group.equal = NULL,
  name = NA,
  lavaanMode = c("sem", "lavaan"),
  std.lv = FALSE,
  suffix = "",
  comparison = TRUE,
  type = c("Auto", "FIML", "cov", "cor", "WLS", "DWLS", "ULS"),
  allContinuousMethod = c("cumulants", "marginals"),
  autoRun = getOption("umx_auto_run"),
  tryHard = c("no", "yes", "ordinal", "search"),
  verbose = FALSE,
  optimizer = NULL,
  std = FALSE,
  printTab = TRUE
)
```

Arguments

model	A lavaan syntax string, e.g. "A~~B"
data	Data to add to model (defaults to auto, which is just sketch mode)
group	= Column to use for multi-group (default = NULL)
group.equal	= what to equate across groups. Default (NULL) means no equates. See details for what we might implement in future.
name	Model name (can also add name in # commented first line)
lavaanMode	Auto-magical path settings for cfa/sem (default) or no-defaults ("lavaan")
std.lv	= FALSE Whether to set var of latents to 1 (default FALSE). nb. Toggles fix first.

suffix	String to append to each label (useful if model will be used in a multi-group model)
comparison	Compare the new model to the old (if updating an existing model: default = TRUE)
type	One of "Auto", "FIML", "cov", "cor", "WLS", "DWLS", "ULS"
allContinuousMethod	"cumulants" or "marginals". Used in all-continuous WLS data to determine if a means model needed.
autoRun	Whether to run the model (default), or just to create it and return without running.
tryHard	Default ('no') uses normal mxRun. "yes" uses mxTryHard. Other options: "ordinal", "search"
verbose	Whether to tell the user what latents and manifests were created etc. (Default = FALSE)
optimizer	optionally set the optimizer (default NULL does nothing)
std	Whether to print estimates. Defaults to FALSE ("raw"), TRUE = "std", for no parameter table use NULL.
printTab	= TRUE (more for debugging)

Details

Uses the defaults of `lavaan::sem`

- int.ov.free = TRUE
- int.lv.free = FALSE
- auto.fix.first = TRUE (unless std.lv = TRUE)
- auto.fix.single = TRUE
- auto.var = TRUE
- auto.cov.lv.x = TRUE
- auto.th = TRUE
- auto.delta = TRUE
- auto.cov.y = TRUE
- fixed.x = FALSE (not standard in `lavaan::sem`, but needed for RAM)

Lavaan is well documented. For quick reference, some common symbols in lavaan strings are:

"=~"	lhs (Latent) is manifested by rhs
"~"	lhs "is regressed on" (<-) rhs
"~~"	lhs covaries with rhs
"~ 1"	lhs has mean
":="	lhs is defined by rhs (see `OpenMx::mxAlgebra()`)
"=="	lhs is constrained == to rhs (see `OpenMx::mxConstraint()`)

Naming of multiple groups

When multiple groups are found the groups are named `name_grouplevel` White space is replaced with "_" and illegal characters are replaced with "x"

note: Options for group.equal. In future, we might implement (but have not as yet):

1. c("loadings"
2. "intercepts"
3. "means"
4. "regressions"
5. "residuals"
6. "covariances"

Value

- list of `umxPath()`s

See Also

- `umxRAM()`

Other Miscellaneous Utility Functions: `install.OpenMx()`, `qm()`, `umxBrownie()`, `umxRAM2Lav()`, `umxVersion()`, `umx_array_shift()`, `umx_find_object()`, `umx_msg()`, `umx_open_CRAN_page()`, `umx_pad()`, `umx_print()`, `umx_score_scale()`, `umx`

Examples

```
## Not run:
# auto-data, print table, return umxRAM model
m1 = umxLav2RAM("y ~ x", printTab= TRUE)

lav = "y ~ x1 + 2.4*x2 + x3"
tmp = umxLav2RAM(lav, data = "auto", printTab= FALSE)

# Add labels to parameters, e.g. "x3_loading" as a loading for x3->x1
tmp = umxLav2RAM("x1 ~ x3_loading*x3")
umx_print(tmp$A$labels)
# |   |x1       |x3         |
# |:--|:--------|:----------|
# |x1 |x1_to_x1 |x3_loading |
# |x3 |x1_to_x3 |x3_to_x3   |

# Fix values, e.g. x2 -> y fixed at 2.4
tmp = umxLav2RAM("y ~ x1 + 2.4*x2; s =~ 0*y11 + 1*y12 + 2*y13 + 3*y14")

tmp = umxLav2RAM("L =~ X1 + X2; L ~ Y")
plot(tmp, min=c("L", "Y"))

# Factor model showing auto-addition of correlations among exogenous latents
# and auto-residuals on manifests
```

```
data("HS.ability.data", package = "OpenMx")

cov(HS.ability.data[, c("visual"  , "cubes"   , "flags")])
cov(HS.ability.data[, c("paragrap", "sentence", "wordm")])
cov(HS.ability.data[, c("addition", "counting", "straight")])

HS = "spatial =~ visual   + cubes    + flags
      verbal  =~ paragrap + sentence + wordm
      speed   =~ addition + counting + straight"

m1 = umxRAM(HS, data = umx_scale(HS.ability.data))

# Multiple groups
m1 = umxRAM(HS, data = umx_scale(HS.ability.data), group = "school")

# More examples

lav = " # Moderated mediation
gnt ~ a*cb
INT ~ b1*gnt + b2*cn + b3*cngn + c*cb

indirect := a*b1
direct := c

ab3 := a * b3
loCN := a * b1 + ab3 * -0.5
hiCN := a * b1 + ab3 * 0.5
"
tmp = umxRAM(lav)
# plot showing ability to influence layout with max min same groupings
plot(tmp, max = c("cb", "cn", "cngn"), same = "gnt", min= "INT")

# Algebra: e.g. b1^2
m1 = umxRAM("x1~b1*x2; B1_sq := b1^2", data = demoOneFactor)
m1$B1_sq$result # = 0.47

# Model with constraints and labeled parameters
lav = "
y ~ b1*x1 + b2*x2 + b3*x3
# constraints
b1 == (b2 + b3)^2
b1 > exp(b2 + b3)"

tmp = umxLav2RAM(lav)

namedStr = "  # my name
y ~x"
m1 = umxRAM(namedStr)

# Formative factor
# lavaanify("f5 <~ z1 + z2 + z3 + z4")

## End(Not run)
```

umxMatrix *Make a mxMatrix with automatic labels. Also takes name as the first parameter for more readable code.*

Description

umxMatrix is a wrapper for mxMatrix which labels cells buy default, and has the name parameter first in order.

Usage

```
umxMatrix(
  name = NA,
  type = "Full",
  nrow = NA,
  ncol = NA,
  free = FALSE,
  values = NA,
  labels = TRUE,
  lbound = NA,
  ubound = NA,
  byrow = getOption("mxByrow"),
  baseName = NA,
  dimnames = NA,
  condenseSlots = getOption("mxCondenseMatrixSlots"),
  ...,
  joinKey = as.character(NA),
  joinModel = as.character(NA),
  jiggle = NA
)
```

Arguments

name	The name of the matrix (Default = NA). Note the different order compared to mxMatrix!
type	The type of the matrix (Default = "Full")
nrow	Number of rows in the matrix: Must be set
ncol	Number of columns in the matrix: Must be set
free	Whether cells are free (Default FALSE)
values	The values of the matrix (Default NA)
labels	Either whether to label the matrix (default TRUE), OR a vector of labels to apply.
lbound	Lower bounds on cells (Defaults to NA)

ubound	Upper bounds on cells (Defaults to NA)
byrow	Whether to fill the matrix down columns or across rows first (Default = getOption('mxByrow')
baseName	Set to override the default (which is to use the matrix name as the prefix).
dimnames	NA
condenseSlots	Whether to save memory by NULLing out unused matrix elements, like labels, ubound etc. Default = getOption('mxCondenseMatrixSlots')
...	Additional parameters (!! not currently supported by umxMatrix)
joinKey	See mxMatrix documentation: Defaults to as.character(NA)
joinModel	See mxMatrix documentation: Defaults to as.character(NA)
jiggle	= NA passed to umxLabel to jiggle start values (default does nothing)

Value

- mxMatrix()

References

- https://github.com/tbates/umx, https://tbates.github.io

See Also

- xmu_simplex_corner(), mxMatrix(), umxLabel(), umxRAM()

Other Core Modeling Functions: umxAlgebra(), umxModify(), umxPath(), umxRAM(), umxRun(), umxSummary(), umxSuperModel(), umx

Examples

```
# ============================================================================
# = 1. Showing how name is first parameter, and how cells are labelled by default. =
# ============================================================================
umxMatrix("test", "Full", 2, 2)$labels
#      [,1]        [,2]
# [1,] "test_r1c1" "test_r1c2"
# [2,] "test_r2c1" "test_r2c2"

# ============================================================
# = 2. Over-ride default (matrix name) as prefix for labels =
# ============================================================
umxMatrix("test", "Full", 2, 2, baseName = "bob")$labels # bob_r1c1


# ========================================
# = 3. User-provided labels are left as-is =
# ========================================
umxMatrix("foo", "Lower", nrow=2, ncol=2, labels= c(NA, "beta1", NA))
#      [,1]    [,2]
# [1,] NA      NA
```

```
# [2,] "beta1" NA
```

`umxMendelianRandomization` *Build a SEM implementing the equivalent of 2-stage least squares regression*

Description

`umxTwoStage` implementing the Structural Equation Model equivalent of a 2SLS regression. For ease of learning, the function is modeled closely on the `sem::tsls()`.

Usage

```
umxMendelianRandomization(
  formula = Y ~ X,
  instruments = ~qtl,
  data,
  subset,
  weights,
  contrasts = NULL,
  name = "tsls",
  ...
)
```

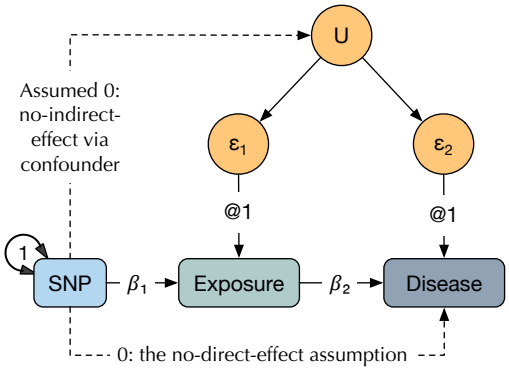
Arguments

`formula`	The structural equation to be estimated (default = Y ~ X). A constant is implied if not explicitly deleted.
`instruments`	A one-sided formula specifying instrumental variables (default = qtl).
`data`	Frame containing the variables in the model.
`subset`	(optional) vector specifying a subset of observations to be used in fitting the model.
`weights`	(optional) vector of weights to be used in the fitting process (not supported) If specified should be a non-negative numeric vector with one entry for each observation, to be used to compute weighted 2SLS estimates.
`contrasts`	an optional list (not supported)
`name`	for the model (default = "tsls")
`...`	arguments to be passed along. (not supported)

Details

The example is a Mendelian Randomization analysis showing the utility of SEM over two-stage regression.

The following figure shows how the ACE model appears as a path diagram:



Value

- mxModel()

References

- – Fox, J. (1979) Simultaneous equation models and two-stage least-squares. In Schuessler, K. F. (ed.) *Sociological Methodology*, Jossey-Bass.
- Greene, W. H. (1993) *Econometric Analysis*, Second Edition, Macmillan.

See Also

- umx_make_MR_data(), sem::tsls(), umxRAM()

Other Super-easy helpers: umxEFA(), umx

Examples

```
library(umx)


# ====================================
# = Mendelian Randomization analysis =
# ====================================

## Not run:
# Note: in practice: many more subjects are desirable - this just to let example run fast
df = umx_make_MR_data(1000)
m1 = umxTwoStage(Y ~ X, instruments = ~ qtl, data = df)
parameters(m1)
plot(m1, means = FALSE, min="") # help DiagrammaR layout the plot.
m2 = umxModify(m1, "qtl_to_X", comparison=TRUE, tryHard="yes", name="QTL_affects_X") # yip
```

```
m3 = umxModify(m1, "X_to_Y"  , comparison=TRUE, tryHard="yes", name="X_affects_Y") # nope
plot(m3, means = FALSE)

# Errant analysis using ordinary least squares regression (WARNING this result is CONFOUNDED!!)
m1 = lm(Y ~ X    , data = df); coef(m1) # incorrect .35 effect of X on Y
m1 = lm(Y ~ X + U, data = df); coef(m1) # Controlling U reveals the true 0.1 beta weight
#
#
df = umx_make_MR_data(1e5)
m1 = umxMendelianRandomization(Y ~ X, instruments = ~ qtl, data = df)
coef(m1)

# ======================
# = Now with sem::tsls =
# ======================
# library(sem) # may require you to install X11
m2 = sem::tsls(formula = Y ~ X, instruments = ~ qtl, data = df)
coef(m2)
m3 = tsls(formula = Y ~ X, instruments = ~ qtl, data = (df[1, "qtl"] = NA))

## End(Not run)
```

---

umxMI *Report modifications which would improve fit.*

Description

This function uses the mechanical modification-indices approach to detect single paths which, if added or dropped, would improve fit.

Usage

```
umxMI(
  model = NA,
  matrices = NA,
  full = TRUE,
  numInd = NA,
  typeToShow = "both",
  decreasing = TRUE
)
```

Arguments

model	An mxModel() for which to report modification indices
matrices	which matrices to test. The default (NA) will test A & S for RAM models
full	Change in fit allowing all parameters to move. If FALSE only the parameter under test can move.

numInd	How many modifications to report. Use -1 for all. Default (NA) will report all over 6.63 (p = .01)
typeToShow	Whether to shown additions or deletions (default = "both")
decreasing	How to sort (default = TRUE, decreasing)

Details

Notes:

1. Runs much faster with full = FALSE (but this does not allow the model to re-fit around the newly- freed parameter).
2. Compared to mxMI, this function returns top changes, and also suppresses the run message.
3. Finally, of course: see the requirements for (legitimate) post-hoc modeling in `mxMI()` You are almost certainly doing better science when testing competing models rather than modifying a model to fit.

References

- https://www.github.com/tbates/umx

See Also

- `mxMI()`

Other Modify or Compare Models: `umxEquate()`, `umxFixAll()`, `umxModify()`, `umxSetParameters()`, `umxUnexplainedCausalNexus()`, `umx`

Examples

```
require(umx)
data(demoOneFactor)
manifests = names(demoOneFactor)
m1 = umxRAM("One Factor", data = demoOneFactor, type = "cov",
umxPath("G", to = manifests),
umxPath(var = manifests),
umxPath(var = "G", fixedAt = 1)
)
# umxMI(m1, full=FALSE)
```

---

umxModel *Catches users typing umxModel instead of umxRAM.*

Description

Catches a common typo, moving from mxModel to umx.

Usage

```
umxModel(...)
```

Arguments

...	Anything. We're just going to throw an error.

Value

None

References

- https://github.com/tbates/umx, https://tbates.github.io

See Also

- umxRAM(), mxModel()

Other xmu internal not for end user: umxRenameMatrix(), umxTwinMaker(), umx_APA_pval(), umx_fun_mean_sd(), umx_get_bracket_addresses(), umx_make(), umx_standardize(), umx_string_to_algebra(), umx, xmuHasSquareBrackets(), xmuLabel_MATRIX_Model(), xmuLabel_Matrix(), xmuLabel_RAM_Model(), xmuMI(), xmuMakeDeviationThresholdsMatrices(), xmuMakeOneHeadedPathsFromPathList(), xmuMakeTwoHeadedPathsFromPathList(), xmuMaxLevels(), xmuMinLevels(), xmuPropagateLabels(), xmuRAM2Ordinal(), xmuTwinSuper_Continuous(), xmuTwinUpgradeMeansToCovariateModel(), xmu_CI_merge(), xmu_CI_stash(), xmu_DF_to_mxData_TypeCov(), xmu_PadAndPruneForDefVars(), xmu_cell_is_on(), xmu_check_levels_identical(), xmu_check_needs_means(), xmu_check_variance(), xmu_clean_label(), xmu_data_missing(), xmu_data_swap_a_block(), xmu_describe_data_WLS(), xmu_dot_make_paths(), xmu_dot_make_residuals(), xmu_dot_maker(), xmu_dot_move_ranks(), xmu_dot_rank_str(), xmu_extract_column(), xmu_get_CI(), xmu_lavaan_process_group(), xmu_make_TwinSuperModel(), xmu_make_bin_cont_pair_data(), xmu_make_mxData(), xmu_match.arg(), xmu_name_from_lavaan_str(), xmu_path2twin(), xmu_path_regex(), xmu_safe_run_summary(), xmu_set_sep_from_suffix(), xmu_show_fit_or_comparison(), xmu_simplex_corner(), xmu_standardize_ACEcov(), xmu_standardize_ACEv(), xmu_standardize_ACE(), xmu_standardize_CP(), xmu_standardize_IP(), xmu_standardize_RAM(), xmu_standardize_SexLim(), xmu_standardize_Simplex(), xmu_start_value_list(), xmu_starts(), xmu_twin_add_WeightMatrices(), xmu_twin_check(), xmu_twin_get_var_names(), xmu_twin_upgrade_selDvs2SelVars()

Examples

```
## Not run:
umxModel()

## End(Not run)
```

---

umxModify *umxModify: Add, set, or drop model paths by label.*

Description

umxModify allows you to modify, re-run and summarize an mxModel(), all in one line of script.

Usage

```
umxModify(
  lastFit,
  update = NULL,
  master = NULL,
  regex = FALSE,
  free = FALSE,
  value = 0,
  newlabels = NULL,
  freeToStart = NA,
  name = NULL,
  verbose = FALSE,
  intervals = FALSE,
  comparison = FALSE,
  autoRun = getOption("umx_auto_run"),
  tryHard = c("no", "yes", "ordinal", "search")
)
```

Arguments

lastFit	The mxModel() you wish to update and run.
update	What to update before re-running. Can be a list of labels, a regular expression (set regex = TRUE) or an object such as mxCI etc.
master	If you set master, then the labels in update will be equated (slaved) to those provided in master.
regex	Whether or not update is a regular expression (default FALSE). If you provide a string, it overrides the contents of update, and sets regex to TRUE.
free	The state to set "free" to for the parameters whose labels you specify (defaults to free = FALSE, i.e., fixed)
value	The value to set the parameters whose labels you specify too (defaults to 0)
newlabels	If not NULL, used as a replacement set of labels (can be regular expression). value and free are ignored!
freeToStart	Whether to update parameters based on their current free-state. free = c(TRUE, FALSE, NA), (defaults to NA - i.e, not checked)
name	The name for the new model
verbose	How much feedback to give
intervals	Whether to run confidence intervals (see mxRun())
comparison	Whether to run umxCompare() on the new and old models.
autoRun	Whether to run the model (default), or just to create it and return without running.
tryHard	Default ('no') uses normal mxRun. "yes" uses mxTryHard. Other options: "ordinal", "search"

Details

You can add paths, or other model elements, set path values (default is 0), or replace labels. As an example, this one-liner drops a path labelled "Cs", and returns the updated model:

```
fit2 = umxModify(fit1,update = "Cs",name = "newModelName",comparison = TRUE)
```

Regular expressions are a powerful feature: they let you drop collections of paths by matching patterns for instance, this would match labels containing either "Cs" or "Cr":

```
fit2 = umxModify(fit1, regex = "C\[sr\]", name = "drop_Cs_and_Cr", comparison = TRUE)
```

You may find it easier to be more explicit. Like this:

```
fit2 = umxSetParameters(fit1, labels = c("Cs", "Cr"), values = 0, free = FALSE, name = "newName")
fit2 = mxRun(fit2)
summary(fit2)
```

Note: A (minor) limitation is that you cannot simultaneously set value to 0 AND relabel cells (because the default value is 0, so it is ignored when using newlabels).

Value

- mxModel()

References

- https://github.com/tbates/umx

See Also

Other Core Modeling Functions: umxAlgebra(), umxMatrix(), umxPath(), umxRAM(), umxRun(), umxSummary(), umxSuperModel(), umx

Other Modify or Compare Models: umxEquate(), umxFixAll(), umxMI(), umxSetParameters(), umxUnexplainedCausalNexus(), umx

Examples

```
require(umx)

# First we'll just build a 1-factor model
umx_set_optimizer("SLSQP")
data(demoOneFactor)
manifests = names(demoOneFactor)

m1 = umxRAM("One Factor", data = demoOneFactor, type = "cov",
umxPath("G", to = manifests),
umxPath(var = manifests),
umxPath(var = "G", fixedAt = 1)
)

# 1. Drop the path to x1 (also updating the name so it's
```

```
#    self-explanatory, and get a fit comparison
m2 = umxModify(m1, update = "G_to_x1", name = "drop_X1", comparison = TRUE)

## Not run:
# 2. Add the path back (setting free = TRUE)
m2 = umxModify(m1, update = "G_to_x1", free= TRUE, name = "addback_X1", comparison = TRUE)
# 3. Fix a value at a non-zero value
m3 = umxModify(m1, update = "G_to_x1", value = .35, name = "fix_G_x1_at_35", comp = TRUE)
# You can add objects to models. For instance this would add a path (overwriting the existing one)
# (thanks Johannes!)
m3 = umxModify(m1, umxPath("G", with = "x1"), name= "addedPath")

# Use regular expression to drop multiple paths: e.g. G to x3, x4, x5
m3 = umxModify(m1, regex = "^G_to_x[3-5]", name = "tried_hard", comp = TRUE, tryHard="yes")

# Same, but don't autoRun
m2 = umxModify(m1, regex  = "^G_to_x[3-5]", name = "no_G_to_x3_5", autoRun = FALSE)

# Re-write a label
newLabel = "A_rose_by_any_other_name"
newModelName = "model_doth_smell_as_sweet"
m2 = umxModify(m1, update = "G_to_x1", newlabels= newLabel, name = newModelName, comparison = TRUE)
# Change labels in 2 places
labsToUpdate = c("G_to_x1", "G_to_x2")
newLabel = "G_to_1_or_2"
m2 = umxModify(m1, update = labsToUpdate, newlabels= newLabel, name = "equated", comparison = TRUE)

# Advanced!
# Regular expressions let you use pieces of the old names in creating new ones!
searchString = "G_to_x([0-9])"
newLabel = "loading_for_path\\1" # use value in regex group 1
m2 = umxModify(m1, regex = searchString, newlabels= newLabel, name = "grep", comparison = TRUE)

## End(Not run)
```

---

umxParameters *Display path estimates from a model, filtering by name and value.*

Description

Often you want to see the estimates from a model, and often you don't want all of them. umxParameters() helps in this case, allowing you to select parameters matching a name filter, and also to only show parameters above or below a certain value.

If pattern is a vector, each regular expression is matched, and all unique matches to the whole vector are returned.

Usage

```
umxParameters(
  x,
  thresh = c("all", "above", "below", ">", "<", "NS", "sig"),
  b = NULL,
  pattern = ".*",
  std = FALSE,
  digits = 2
)

parameters(
  x,
  thresh = c("all", "above", "below", ">", "<", "NS", "sig"),
  b = NULL,
  pattern = ".*",
  std = FALSE,
  digits = 2
)
```

Arguments

x	an mxModel() or model summary from which to report parameter estimates.
thresh	optional: Filter out estimates 'below' or 'above' a certain value (default = "all").
b	Combine with thresh to set a minimum or maximum for which estimates to show.
pattern	Optional string to match in the parameter names. Default '.*' matches all. regex() allowed!
std	Standardize output: NOT IMPLEMENTED YET
digits	Round to how many digits (2 = default).

Details

It is on my TODO list to implement filtering by significance, and to add standardizing.

Value

- list of matching parameters, filtered by name and value

References

- https://github.com/tbates/umx, https://tbates.github.io

See Also

- umxGetParameters(), umxSummary(), namez()

Other Reporting Functions: loadings.MxModel(), umxAPA(), umxFactorScores(), umxGetParameters(), umxReduce(), umx_aggregate(), umx_names(), umx_time(), umx

Examples

```
require(umx)
data(demoOneFactor)
manifests = names(demoOneFactor)
m1 = umxRAM("OneFactor", data = demoOneFactor,
umxPath(from = "G", to = manifests), # factor loadings
umxPath(v.m. = manifests),           # residual variance
umxPath(v1m0 = "G")                  # standardized latent
)
# Parameters with values below .1
umxParameters(m1, "below", .1)
# Parameters with values above .5
umxParameters(m1, "above", .5)
# Parameters with values below .1 and containing "_to_" in their label
umxParameters(m1, "below", .1, "_to_")
```

---

umxPath *Easier (and powerful) specification of paths in SEM.*

Description

The goal of this function is to enable quick-to-write, quick-to-read, flexible path descriptions for RAM models in OpenMx.

It introduces the following new words to our vocabulary for describing paths: **with**, **var**, **cov**, **means**, **v1m0**, **v0m0, v.m0**, **v.m.**, **fixedAt**, **freeAt**, **firstAt**, **unique.bivariate**, **unique.pairs**, **fromEach**, **Cholesky**, **defn**, **forms**.

The new preposition "with" means you no-longer need set arrows = 2 on covariances. Instead, you can say:

`umxPath(A,with = B)` instead of `mxPath(from = A,to = B,arrows = 2)`.

Specify a variance for A with

`umxPath(var = "A")`.

This is equivalent to `mxPath(from = "A",to = "A",arrows = 2)`.

Of course you can use vectors anywhere:

`umxPath(var = c('N','E','O'))`

To specify a mean, you just say

`umxPath(mean = "A")`, which is equivalent to `mxPath(from = "one",to = "A")`.

To fix a path at a value, you can say:

`umxPath(var = "A",fixedAt = 1)` .

instead of `mxPath(from = A,to = A,arrows = 2,free = FALSE,values = 1)`

The common task of creating a variable with variance fixed at 1 and mean at 0 is done thus:

`umxPath(v1m0 = "A")`

For free variance and means use:

`umxPath(v.m. = "A")`

umxPath exposes "unique.bivariate" and "unique.pairs" so you don't have to remember how to fill in connect = in mxPath (you can still use connect if you wish).

So, to create paths creates A<->A, B<->B, and A->B, you would say:

`umxPath(unique.pairs = c('A',"B"))`

To create paths A<->B, B<->C, and A<->C, you would say: `umxPath(unique.bivariate = c('A',"B","C"))`

`umxPath(fromEach = c('A',"B","C"))` Creates one-headed arrows on the all.bivariate pattern

Setting up a latent trait, you can scale with a fixed first path thus:

`umxPath("A",to = c("B","C","D"),firstAt = 1)`

This is equivalent to `mxPath(from = A,to = c(B,C,D),free = c(F,T,T),values = c(1,.5,.4))`.

To create Cholesky-pattern connections:

`umxPath(Cholesky = c("A1","A2"),to c("var1","var2"))`

Usage

```
umxPath(
  from = NULL,
  to = NULL,
  with = NULL,
  var = NULL,
  cov = NULL,
  means = NULL,
  v1m0 = NULL,
  v.m. = NULL,
  v0m0 = NULL,
  v.m0 = NULL,
  fixedAt = NULL,
  freeAt = NULL,
  firstAt = NULL,
  unique.bivariate = NULL,
  unique.pairs = NULL,
  fromEach = NULL,
  forms = NULL,
  Cholesky = NULL,
  defn = NULL,
  connect = c("single", "all.pairs", "all.bivariate", "unique.pairs",
    "unique.bivariate"),
  arrows = 1,
  free = TRUE,
  values = NA,
  labels = NA,
  lbound = NA,
  ubound = NA,
  hasMeans = NULL
)
```

Arguments

from	One or more source variables e.g "A" or c("A","B")
to	One or more target variables for one-headed paths, e.g "A" or c("A","B").
with	2-headed path <–> from 'from' to 'with'.
var	Equivalent to setting 'from' and 'arrows' = 2. nb: from, to, and with must be left empty.
cov	Convenience to allow 2 variables to covary (equivalent to 'from' and 'with'). nb: leave from, to, etc. empty
means	equivalent to "from = 'one', to = x. nb: from, to, with and var must be left empty (their default).
v1m0	variance of 1 and mean of zero in one call.
v.m.	variance and mean, both free.
v0m0	variance and mean, both fixed at zero.
v.m0	variance free, mean fixed at zero.
fixedAt	Equivalent to setting "free = FALSE, values = x" nb: free and values must be left empty (their default)
freeAt	Equivalent to setting "free = TRUE, values = x" nb: free and values must be left empty (their default)
firstAt	first value is fixed at this (values passed to free are ignored: warning if not a single TRUE)
unique.bivariate	equivalent to setting from, and "connect = "unique.bivariate", arrows = 2". nb: from, to, and with must be left empty (their default)
unique.pairs	equivalent to setting "connect = "unique.pairs", arrows = 2" (don't use from, to, or with)
fromEach	Like all.bivariate, but with one head arrows. 'to' can be set.
forms	Build a formative variable. 'from' variables form the latent. Latent variance is fixed at 0. Loading of path 1 is fixed at 1. unique.bivariate between 'from' variables.
Cholesky	Treat **Cholesky** variables as latent and **to** as measured, and connect as in an ACE model.
defn	Implements a definition variable as a latent with zero variance & mean and labeled 'data.defVar'
connect	as in mxPath - nb: from and to must also be set.
arrows	as in mxPath - nb: from and to must also be set.
free	whether the value is free to be optimised
values	default value list
labels	labels for each path
lbound	lower bounds for each path value
ubound	upper bounds for each path value
hasMeans	Used in 'forms' case to know whether the data have means or not.

Details

This function returns a standard mxPath, but gives new options for specifying the path. In addition to the normal "from" and "to", it adds specialised parameters for variances (var), two headed paths (with) and means (mean). There are also new terms to describe fixing values: "fixedAt" and "fixFirst".

Finally, (in future) it will allow sem-style "A->B" string specification.

Value

- 1 or more mxPath()s

References

- https://tbates.github.io

See Also

- mxPath()

Other Core Modeling Functions: umxAlgebra(), umxMatrix(), umxModify(), umxRAM(), umxRun(), umxSummary(), umxSuperModel(), umx

Examples

```
# ========================================
# = Examples of each path type, and option =
# ========================================

umxPath("A", to = "B") # One-headed path from A to B
umxPath("A", to = "B", fixedAt = 1) # same, with value fixed @1
umxPath("A", to = c("B", "C"), fixedAt = 1:2) # same, with more than 1 value
umxPath("A", to = c("B","C"), firstAt = 1) # Fix only the first path, others free
umxPath(var = "A") # Give a variance to A
umxPath(var = "A", fixedAt = 1) # Give A variance, fixed at 1
umxPath(means = c("A","B")) # Create a means model for A: from = "one", to = "A"
umxPath(v1m0 = "A") # Give "A" variance and a mean, fixed at 1 and 0 respectively
umxPath(v.m. = "A") # Give "A" variance and a mean, leaving both free.
umxPath(v0m0 = "W", label = c(NA, "data.W"))
umxPath("A", with = "B") # using with: same as "to = B, arrows = 2"
umxPath("A", with = "B", fixedAt = .5) # 2-head path fixed at .5
umxPath("A", with = c("B", "C"), firstAt = 1) # first covariance fixed at 1
umxPath(cov = c("A", "B"))  # Covariance A <-> B
umxPath(defn = "mpg") # create latent called def_mpg, with 0 mean * var, and label = "data.mpg"
umxPath(fromEach = c('a','b'), to = c('c','d')) # a->c, a<->d, b<->c, b<->d
umxPath(unique.bivariate = c('a','b','c')) # bivariate paths a<->b, a<->c, b<->c etc.
umxPath(unique.pairs = letters[1:3]) # all distinct pairs: a<->a, a<->b, a<->c, b<->b, etc.
umxPath(Cholesky = c("A1","A2"), to = c("m1", "m2")) # Cholesky

## Not run:
# A worked example
```

```
data(demoOneFactor)
manifests = names(demoOneFactor)
m1 = umxRAM("One Factor", data = demoOneFactor, type= "cov",
umxPath("G", to = manifests),
umxPath(var = manifests),
umxPath(var = "G", fixedAt = 1.0)
)
umxSummary(m1, std = TRUE)
require(umx)
#

# ====================
# = Cholesky example =
# ====================
# ============================================================================
# = 3-factor Cholesky (A component of a 5-variable 3-factor ACE model) =
# ============================================================================
latents   = paste0("A", 1:3)
manifests = names(demoOneFactor)
m1 = umxRAM("Chol", data = demoOneFactor, type = "cov",
umxPath(Cholesky = latents, to = manifests),
umxPath(var = manifests),
umxPath(var = latents, fixedAt = 1)
)
plot(m1, splines= FALSE)

# ============================================================
# = Definition variable example.Not much use at present,  =
# = as def vars are not readily used in RAM models...     =
# = Working on something rational and intuitive.          =
# ============================================================
data(mtcars)
m1 = umxRAM("manifest", data = mtcars,
 umxPath(v.m. = "mpg"),
 umxPath(defn = "mpg")
)


## End(Not run)
```

---

umxPlotACE *Make a graphical display of an ACE model*

Description

plot method for umxACE() models. Make a graphical display of an ACE model

Usage

```
umxPlotACE(
  x = NA,
  file = "name",
  digits = 2,
  means = FALSE,
  std = TRUE,
  strip_zero = TRUE,
  ...
)
```

Arguments

x	mxModel() to plot (created by umxACE in order to inherit the MxModelACE class)
file	The name of the dot file to write: NA = none; "name" = use the name of the model
digits	How many decimals to include in path loadings (default is 2)
means	Whether to show means paths (default is FALSE)
std	Whether to standardize the model (default is TRUE)
strip_zero	Whether to strip the leading "0" and decimal point from parameter estimates (default = TRUE)
...	Additional (optional) parameters

Value

- optionally return the dot code

References

- https://www.github.com/tbates/umx

See Also

- plot(), umxSummary() work for IP, CP, GxE, SAT, and ACE models.
- umxACE()

Other Plotting functions: plot.MxLISRELModel(), plot.MxModel(), umxPlotACEcov(), umxPlotACEv(), umxPlotCP(), umxPlotGxEbiv(), umxPlotGxE(), umxPlotIP(), umxPlotSexLim(), umxPlotSimplex(), umx

Examples

```
require(umx)
data(twinData)
mzData = subset(twinData, zygosity == "MZFF")
dzData = subset(twinData, zygosity == "DZFF")
```

```
m1 = umxACE("plotACE example", selDVs = "bmi", dzData = dzData, mzData = mzData, sep = "")
plot(m1, std = FALSE) # don't standardize
```

---

umxPlotACEcov *Make a graphical display of an ACE model with covariates.*

Description

Make a graphical display of an ACE model with covariates.

Usage

```
umxPlotACEcov(
  x = NA,
  file = "name",
  digits = 2,
  means = FALSE,
  std = TRUE,
  strip_zero = TRUE,
  ...
)
```

Arguments

x	mxModel() to plot (created by umxACE in order to inherit the MxModelACE class)
file	The name of the dot file to write: NA = none; "name" = use the name of the model
digits	How many decimals to include in path loadings (default is 2)
means	Whether to show means paths (default is FALSE)
std	Whether to standardize the model (default is TRUE)
strip_zero	Whether to strip the leading "0" and decimal point from parameter estimates (default = TRUE)
...	Additional (optional) parameters

Value

- optionally return the dot code

References

- https://tbates.github.io

See Also

- plot(), umxSummary() work for IP, CP, GxE, SAT, and ACE models.
- umxACE()

Other Plotting functions: plot.MxLISRELModel(), plot.MxModel(), umxPlotACEv(), umxPlotACE(), umxPlotCP(), umxPlotGxEbiv(), umxPlotGxE(), umxPlotIP(), umxPlotSexLim(), umxPlotSimplex(), umx

Examples

```
require(umx)
# BMI ?twinData from Australian twins.
# Cohort 1 Zygosity 1 == MZ females 3 == DZ females
data(twinData)

# Pick the variables. We will use base names (i.e., "bmi") and set suffix.
selDVs  = c("bmi")
selCovs = c("ht")
selVars = umx_paste_names(c(selDVs, selCovs), sep = "", suffixes= 1:2)
# Just top few pairs so example runs quickly
mzData = subset(twinData, zygosity == "MZFF", selVars)[1:100, ]
dzData = subset(twinData, zygosity == "DZFF", selVars)[1:100, ]
m1 = umxACEcov(selDVs= selDVs, selCovs= selCovs, dzData= dzData, mzData= mzData, sep= "")
plot(m1)
plot(m1, std = FALSE) # don't standardize
```

---

umxPlotACEv *Produce a graphical display of an ACE variance-components twin model*

Description

Plots an ACE model graphically, opening the result in the browser (or a graphviz application).

Usage

```
umxPlotACEv(
  x = NA,
  file = "name",
  digits = 2,
  means = FALSE,
  std = TRUE,
  strip_zero = TRUE,
  ...
)
```

Arguments

x	umxACEv() model to plot.
file	The name of the dot file to write: Default ("name") = use the name of the model. NA = don't plot.
digits	How many decimals to include in path loadings (default = 2)
means	Whether to show means paths (default = FALSE)
std	Whether to standardize the model (default = FALSE)
strip_zero	Whether to strip the leading "0" and decimal point from parameter estimates (default = TRUE)
...	Additional (optional) parameters

Value

- optionally return the dot code

References

- https://www.github.com/tbates/umx

See Also

Other Plotting functions: plot.MxLISRELModel(), plot.MxModel(), umxPlotACEcov(), umxPlotACE(), umxPlotCP(), umxPlotGxEbiv(), umxPlotGxE(), umxPlotIP(), umxPlotSexLim(), umxPlotSimplex(), umx

Other Reporting functions: RMSEA.MxModel(), RMSEA.summary.mxmodel(), RMSEA(), extractAIC.MxModel(), loadings(), residuals.MxModel(), umxCI_boot(), umxCI(), umxCompare(), umxConfint(), umxExpCov(), umxExpMeans(), umxFitIndices(), umxRotate(), umxSummary.MxModel()

Examples

```
require(umx)
data(twinData)
mzData <- subset(twinData, zygosity == "MZFF")
dzData <- subset(twinData, zygosity == "DZFF")
m1 = umxACEv(selDVs = "bmi", dzData = dzData, mzData = mzData, sep = "")
plot(m1, std = FALSE) # don't standardize
```

---

umxPlotCP *Draw and display a graphical figure of Common Pathway model*

Description

Options include digits (rounding), showing means or not, and which output format is desired.

Usage

```
umxPlotCP(
  x = NA,
  means = FALSE,
  std = TRUE,
  digits = 2,
  showFixed = TRUE,
  file = "name",
  format = c("current", "graphviz", "DiagrammeR"),
  SEstyle = FALSE,
  strip_zero = TRUE,
  ...
)
```

Arguments

x	The Common Pathway mxModel() to display graphically
means	Whether to show means paths (defaults to FALSE)
std	Whether to standardize the model (defaults to TRUE)
digits	How many decimals to include in path loadings (defaults to 2)
showFixed	Whether to graph paths that are fixed but != 0 (default = TRUE)
file	The name of the dot file to write: NA = none; "name" = use the name of the model
format	= c("current", "graphviz", "DiagrammeR")
SEstyle	report "b (se)" instead of "b [lower, upper]" when CIs are found (Default FALSE)
strip_zero	Whether to strip the leading "0" and decimal point from parameter estimates (default = TRUE)
...	Optional additional parameters

Value

- Optionally return the dot code

References

- https://tbates.github.io

See Also

- plot(), umxSummary() work for IP, CP, GxE, SAT, and ACE models.
- umxCP()

Other Plotting functions: plot.MxLISRELModel(), plot.MxModel(), umxPlotACEcov(), umxPlotACEv(), umxPlotACE(), umxPlotGxEbiv(), umxPlotGxE(), umxPlotIP(), umxPlotSexLim(), umxPlotSimplex(), umx

Other Twin Reporting Functions: umxPlotDoC(), umxReduceACE(), umxReduceGxE(), umxReduce(), umxSummarizeTwinData(), umxSummaryACEcov(), umxSummaryACEv(), umxSummaryACE(), umxSummaryCP(), umxSummaryDoC(), umxSummaryGxEbiv(), umxSummaryGxE(), umxSummaryIP(), umxSummarySexLim(), umxSummarySimplex(), umx

Examples

```
## Not run:
require(umx)
umx_set_optimizer("SLSQP")
data(GFF)
mzData = subset(GFF, zyg_2grp == "MZ")
dzData = subset(GFF, zyg_2grp == "DZ")
selDVs = c("gff", "fc", "qol", "hap", "sat", "AD")
m1 = umxCP("new", selDVs = selDVs, sep = "_T",
dzData = dzData, mzData = mzData, nFac = 3
)
# m1 = mxTryHardOrdinal(m1)
umxPlotCP(m1)
plot(m1) # No need to remember a special name: plot works fine!

## End(Not run)
```

---

umxPlotDoC *Plot a Direction of Causation Model.*

Description

Summarize a fitted model returned by umxDoC(). Can control digits, report comparison model fits, optionally show the *Rg* (genetic and environmental correlations), and show confidence intervals. the report parameter allows drawing the tables to a web browser where they may readily be pasted into, e.g. Word.

Usage

```
umxPlotDoC(
  x = NA,
  means = FALSE,
  std = TRUE,
  digits = 2,
  showFixed = TRUE,
  file = "name",
  format = c("current", "graphviz", "DiagrammeR"),
  SEstyle = FALSE,
  strip_zero = FALSE,
  ...
)
```

Arguments

x	a umxDoC() model to display graphically
means	Whether to show means paths (defaults to FALSE)
std	Whether to standardize the model (defaults to TRUE)
digits	How many decimals to include in path loadings (defaults to 2)
showFixed	Whether to graph paths that are fixed but != 0 (default = TRUE)
file	The name of the dot file to write: NA = none; "name" = use the name of the model
format	= c("current", "graphviz", "DiagrammeR")
SEstyle	report "b (se)" instead of "b [lower, upper]" when CIs are found (Default FALSE)
strip_zero	Whether to strip the leading "0" and decimal point from parameter estimates (default = TRUE)
...	Other parameters to control model summary.

Details

See documentation for other umx models here: umxSummary().

Value

- Optionally return the dot code

References

- https://tbates.github.io

See Also

- umxDoC(), umxSummary.MxModelDoC(), umxModify()

Other Twin Reporting Functions: umxPlotCP(), umxReduceACE(), umxReduceGxE(), umxReduce(), umxSummarizeTwinData(), umxSummaryACEcov(), umxSummaryACEv(), umxSummaryACE(), umxSummaryCP(), umxSummaryDoC(), umxSummaryGxEbiv(), umxSummaryGxE(), umxSummaryIP(), umxSummarySexLim(), umxSummarySimplex(), umx

Examples

```
## Not run:
# ================
# = 1. Load Data =
# ================
data(docData)
mzData = subset(docData, zygosity %in% c("MZFF", "MZMM"))
dzData = subset(docData, zygosity %in% c("DZFF", "DZMM"))

# ======================================
# = 2. Define manifests for var 1 and 2 =
```

```
# ====================================
var1 = paste0("varA", 1:3)
var2 = paste0("varB", 1:3)

# ======================================================
# = 2. Make the non-causal (Cholesky) and causal models =
# ======================================================
Chol= umxDoC(var1= var1, var2= var2, mzData= mzData, dzData= dzData, causal= FALSE)
DoC = umxDoC(var1= var1, var2= var2, mzData= mzData, dzData= dzData, causal= TRUE)

# ================================================
# = Make the directional models by modifying DoC =
# ================================================
a2b = umxModify(DoC, "a2b", free = TRUE, name = "A2B")
plot(a2b)


## End(Not run)
```

`umxPlotGxE` *Plot the results of a GxE univariate test for moderation of ACE components.*

Description

Plot GxE results (univariate environmental moderation of ACE components). Options include plotting the raw and standardized graphs separately, or in a combined panel. You can also set the label for the x axis (xlab), and choose the location of the legend.

Usage

```
umxPlotGxE(
  x,
  xlab = NA,
  location = "topleft",
  separateGraphs = FALSE,
  acergb = c("red", "green", "blue", "black"),
  ...
)
```

Arguments

x	A fitted `umxGxE()` model to plot
xlab	String to use for the x label (default = NA, which will use the variable name)
location	Where to plot the legend (default = "topleft") see ?legend for alternatives like bottomright
separateGraphs	(default = FALSE)
acergb	Colors to use for plot c(a = "red", c = "green", e = "blue", tot = "black")
...	Optional additional parameters

Value

None

References

- https://tbates.github.io

See Also

- plot(), umxSummary() work for IP, CP, GxE, SAT, and ACE models.
- umxGxE()

Other Plotting functions: plot.MxLISRELModel(), plot.MxModel(), umxPlotACEcov(), umxPlotACEv(), umxPlotACE(), umxPlotCP(), umxPlotGxEbiv(), umxPlotIP(), umxPlotSexLim(), umxPlotSimplex(), umx

Examples

```
## Not run:
require(umx)
data(twinData)
twinData$age1 = twinData$age2 = twinData$age
mzData = subset(twinData, zygosity == "MZFF")
dzData = subset(twinData, zygosity == "DZFF")
m1= umxGxE(selDVs= "bmi", selDefs= "age", dzData= dzData, mzData= mzData, sep="", tryHard="yes")
plot(m1)
# Directly call the umx function
umxPlotGxE(x = m1, xlab = "SES", separateGraphs = TRUE, location = "topleft")


## End(Not run)
```

---

umxPlotGxEbiv *Plot the results of a GxE univariate test for moderation of ACE components.*

Description

Plot GxE results (univariate environmental moderation of ACE components). Options include plotting the raw and standardized graphs separately, or in a combined panel. You can also set the label for the x axis (xlab), and choose the location of the legend.

Usage

```
umxPlotGxEbiv(x, xlab = NA, location = "topleft", separateGraphs = FALSE, ...)
```

Arguments

x	A fitted umxGxEbiv() model to plot
xlab	String to use for the x label (default = NA, which will use the variable name)
location	Where to plot the legend (default = "topleft") see ?legend for alternatives like bottomright
separateGraphs	(default = FALSE)
...	Optional additional parameters

Value

None

References

- https://tbates.github.io

See Also

- plot(), umxSummary() work for IP, CP, GxE, SAT, and ACE models.
- umxGxEbiv()

Other Plotting functions: plot.MxLISRELModel(), plot.MxModel(), umxPlotACEcov(), umxPlotACEv(), umxPlotACE(), umxPlotCP(), umxPlotGxE(), umxPlotIP(), umxPlotSexLim(), umxPlotSimplex(), umx

Examples

```
require(umx)
data(twinData)
## Not run:
selDVs  = "wt"; selDefs = "ht"
df = umx_scale_wide_twin_data(twinData, varsToScale = c("ht", "wt"), suffix = "")
mzData  = subset(df, zygosity %in%  c("MZFF", "MZMM"))
dzData  = subset(df, zygosity %in%  c("DZFF", "DZMM", "DZOS"))

m1 = umxGxEbiv(selDVs = selDVs, selDefs = selDefs,
dzData = dzData, mzData = mzData, sep = "", dropMissingDef = TRUE)
# Plot Moderation
plot(m1)
umxPlotGxEbiv(m1, xlab = "wt", separateGraphs = TRUE, location = "topleft")

## End(Not run)
```

umxPlotIP *Draw a graphical figure for a Independent Pathway model*

Description

Options include digits (rounding), showing means or not, standardization, and which output format is desired.

Usage

```
umxPlotIP(
  x = NA,
  file = "name",
  digits = 2,
  means = FALSE,
  std = TRUE,
  format = c("current", "graphviz", "DiagrammeR"),
  SEstyle = FALSE,
  strip_zero = TRUE,
  ...
)
```

Arguments

x	The umxIP() model to plot
file	The name of the dot file to write: NA = none; "name" = use the name of the model
digits	How many decimals to include in path loadings (defaults to 2)
means	Whether to show means paths (defaults to FALSE)
std	whether to standardize the model (defaults to TRUE)
format	= c("current", "graphviz", "DiagrammeR")
SEstyle	report "b (se)" instead of "b [lower, upper]" (Default)
strip_zero	Whether to strip the leading "0" and decimal point from parameter estimates (default = TRUE)
...	Optional additional parameters

Value

- optionally return the dot code

References

- https://tbates.github.io

See Also

- plot(), umxSummary() work for IP, CP, GxE, SAT, and ACE models.
- umxIP()

Other Plotting functions: plot.MxLISRELModel(), plot.MxModel(), umxPlotACEcov(), umxPlotACEv(), umxPlotACE(), umxPlotCP(), umxPlotGxEbiv(), umxPlotGxE(), umxPlotSexLim(), umxPlotSimplex(), umx

Examples

```
## Not run:
plot(model)
umxPlotIP(model, file = NA)

## End(Not run)
```

---

umxPlotSexLim *Draw and display a graphical figure of a Sex limitation model*

Description

Will plot a graphical figure for a sex limitation model. Options include digits (rounding), showing means or not, and which output format is desired.

Usage

```
umxPlotSexLim(
  x = NA,
  file = "name",
  digits = 2,
  means = FALSE,
  std = TRUE,
  format = c("current", "graphviz", "DiagrammeR"),
  SEstyle = FALSE,
  strip_zero = TRUE,
  ...
)
```

Arguments

x	mxModel() to display graphically
file	The name of the dot file to write: NA = none; "name" = use the name of the model
digits	How many decimals to include in path loadings (defaults to 2)
means	Whether to show means paths (defaults to FALSE)

std	Whether to standardize the model (defaults to TRUE)
format	= c("current", "graphviz", "DiagrammeR")
SEstyle	report "b (se)" instead of "b [lower, upper]" (Default)
strip_zero	Whether to strip the leading "0" and decimal point from parameter estimates (default = TRUE)
...	Optional additional parameters

Value

- Optionally return the dot code

References

- https://tbates.github.io

See Also

- umxSexLim(), umxSummarySexLim()

Other Plotting functions: plot.MxLISRELModel(), plot.MxModel(), umxPlotACEcov(), umxPlotACEv(), umxPlotACE(), umxPlotCP(), umxPlotGxEbiv(), umxPlotGxE(), umxPlotIP(), umxPlotSimplex(), umx

Examples

```
## Not run:
require(umx)
umx_set_optimizer("SLSQP")
data("us_skinfold_data")
# Rescale vars
us_skinfold_data[, c('bic_T1', 'bic_T2')] = us_skinfold_data[, c('bic_T1', 'bic_T2')]/3.4
us_skinfold_data[, c('tri_T1', 'tri_T2')] = us_skinfold_data[, c('tri_T1', 'tri_T2')]/3
us_skinfold_data[, c('caf_T1', 'caf_T2')] = us_skinfold_data[, c('caf_T1', 'caf_T2')]/3
us_skinfold_data[, c('ssc_T1', 'ssc_T2')] = us_skinfold_data[, c('ssc_T1', 'ssc_T2')]/5
us_skinfold_data[, c('sil_T1', 'sil_T2')] = us_skinfold_data[, c('sil_T1', 'sil_T2')]/5

# Data for each of the 5 twin-type groups
mzmData = subset(us_skinfold_data, zyg == 1)
mzfData = subset(us_skinfold_data, zyg == 2)
dzmData = subset(us_skinfold_data, zyg == 3)
dzfData = subset(us_skinfold_data, zyg == 4)
dzoData = subset(us_skinfold_data, zyg == 5)

# ============================
# = Run univariate example =
# ============================
m1 = umxSexLim(selDVs = "bic", sep = "_T", A_or_C = "A", autoRun= FALSE,
mzmData = mzmData, dzmData = dzmData,
mzfData = mzfData, dzfData = dzfData,
dzoData = dzoData
)
```

```
m1 = mxTryHard(m1)
umxPlotSexLim(m1)
plot(m1) # no need to remember a special name: plot works fine!

## End(Not run)
```

`umxPlotSimplex` *Draw and display a graphical figure of a simplex model*

Description

Options include digits (rounding), showing means or not, and which output format is desired.

Usage

```
umxPlotSimplex(
  x = NA,
  file = "name",
  digits = 2,
  means = FALSE,
  std = TRUE,
  format = c("current", "graphviz", "DiagrammeR"),
  strip_zero = TRUE,
  ...
)
```

Arguments

`x`	The `umxSimplex()` model to display graphically
`file`	The name of the dot file to write: NA = none; "name" = use the name of the model
`digits`	How many decimals to include in path loadings (defaults to 2)
`means`	Whether to show means paths (defaults to FALSE)
`std`	Whether to standardize the model (defaults to TRUE)
`format`	= c("current", "graphviz", "DiagrammeR")
`strip_zero`	Whether to strip the leading "0" and decimal point from parameter estimates (default = TRUE)
`...`	Optional additional parameters

Value

- Optionally return the dot code

See Also

- plot(), umxSummary() work for IP, CP, GxE, SAT, simplex, ACEv, or ACE model.
- umxSimplex()

Other Plotting functions: plot.MxLISRELModel(), plot.MxModel(), umxPlotACEcov(), umxPlotACEv(), umxPlotACE(), umxPlotCP(), umxPlotGxEbiv(), umxPlotGxE(), umxPlotIP(), umxPlotSexLim(), umx

Examples

```
## Not run:
data(iqdat)
mzData = subset(iqdat, zygosity == "MZ")
dzData = subset(iqdat, zygosity == "DZ")
selDVs = c("IQ_age1", "IQ_age2", "IQ_age3", "IQ_age4")
m1 = umxSimplex(selDVs = selDVs, sep = "_T", dzData = dzData, mzData = mzData)
# plot(m1)

## End(Not run)
```

---

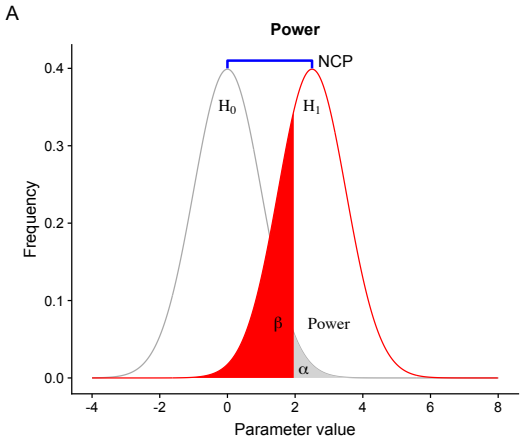
umxPower *Test power to detect specified path values in a model.*

Description

umxPower takes an input model (the model of the true data), and tests power (or determines n) to detect dropping (or changing the value) a path in this true model.

A typical target for power is 80%. Much as the accepted critical p-value is .05, this has emerged as a trade off, in this case of resources required for more powerful studies against the cost of missing a true effect. People interested in truth discourage running studies with low power: A study with 20 percent power will fail to detect real effects 80% of the time. But even with zero power, the Type-I error rate remains a nominal 5% (and with any researcher degrees of freedom, perhaps much more than that). Low powered research, then, fails to detect true effects, and generates support for random false theories about as often. This sounds silly, but empirical rates are often as low as 20% (Button, et al., 2013).

Illustration of α, β, and power (1-β):



Usage

```
umxPower(
  trueModel,
  update = NULL,
  n = NULL,
  power = NULL,
  sig.level = 0.05,
  value = 0,
  method = c("ncp", "empirical"),
  explore = FALSE,
  digits = 2,
  silent = TRUE
)
```

Arguments

trueModel	The model with the parameters at values you expect in the population.
update	The parameter(s) to drop
n	How many subjects? (Default = NULL)
power	Default = NULL (conventional level = .8)
sig.level	Default = .05
value	Value of dropped parameter (default = 0)
method	"ncp" (default) or "empirical"
explore	Whether to tabulate the range of n or effect size (if n specified). Default = FALSE.
digits	Rounding precision for reporting result.
silent	Suppress model runs printouts to console (TRUE)

Value

power table

References

- tutorials

See Also

- umxRAM()

Other Teaching and Testing functions: tmx_show(), umxDiagnose()

Examples

```
# ================================================
# = Power to detect correlation of .3 in 200 people =
# ================================================

# 1 Make some data
tmp = umx_make_raw_from_cov(qm(1, .3| .3, 1), n=2000, varNames= c("X", "Y"), empirical= TRUE)

# 2. Make model of true XY correlation of .3
m1 = umxRAM("corXY", data = tmp,
   umxPath("X", with = "Y"),
   umxPath(var = c("X", "Y"))
)
# 3. Test power to detect .3 versus 0, with n= 90 subjects
umxPower(m1, "X_with_Y", n= 90)

# ####################
# # Estimating power #
# ####################
#
#    method = ncp
#         n = 90
#     power = 0.83
# sig.level = 0.05
# statistic = LRT

# ================================================
# = Tabulate Power across a range of values of  n =
# ================================================
umxPower(m1, "X_with_Y", explore = TRUE)

## Not run:

# ====================================
# = Examples with method = empirical  =
# ====================================

# Power to detect r = .3 given n=90
umxPower(m1, "X_with_Y", n = 90, method = "empirical")
# power is .823
# Test using cor.test doing the same thing.
pwr::pwr.r.test(r = .3, n = 90)
#           n = 90
```

```
#           r = 0.3
#   sig.level = 0.05
#       power = 0.827
# alternative = two.sided

# Power search for detectable effect size, given n = 90
umxPower(m1, "X_with_Y", n= 90, method = "empirical", explore = TRUE)

# Search X_with_Y:power relationship for n=90
# |    | X_with_Y | power | lower | upper |
# |:---|:---------|:------|:------|:------|
# | 1  | 0.03     | 0.27  | 0.15  | 0.44  |
# | 2  | 0.03     | 0.32  | 0.20  | 0.48  |
# | 3  | 0.04     | 0.38  | 0.26  | 0.53  |
# | 4  | 0.04     | 0.45  | 0.33  | 0.57  |
# | 5  | 0.04     | 0.51  | 0.41  | 0.61  |
# | 6  | 0.05     | 0.58  | 0.49  | 0.66  |
# | 7  | 0.05     | 0.64  | 0.57  | 0.71  |
# | 8  | 0.06     | 0.70  | 0.64  | 0.75  |
# | 9  | 0.06     | 0.75  | 0.69  | 0.80  |
# | 10 | 0.06     | 0.80  | 0.74  | 0.85  |
# | 11 | 0.07     | 0.84  | 0.77  | 0.88  |
# | 12 | 0.07     | 0.87  | 0.80  | 0.92  |
# | 13 | 0.08     | 0.90  | 0.83  | 0.94  |
# | 14 | 0.08     | 0.92  | 0.85  | 0.96  |
# | 15 | 0.08     | 0.94  | 0.87  | 0.97  |
# | 16 | 0.09     | 0.95  | 0.89  | 0.98  |
# | 17 | 0.09     | 0.96  | 0.91  | 0.98  |
# | 18 | 0.10     | 0.97  | 0.92  | 0.99  |
# | 19 | 0.10     | 0.98  | 0.93  | 0.99  |
# | 20 | 0.10     | 0.98  | 0.94  | 0.99  |


## End(Not run)
```

---

umxRAM *Easier path-based SEM modeling.*

Description

umxRAM expedites creation of structural equation models, still without doing invisible things to the model. It supports umxPath() but also lavaan-style string specification of models: lavaan's scripting language has become a lingua franca for SEM books, so supporting this improves science learning.

Here's a path example that models miles per gallon (mpg) as a function of weight (wt) and engine displacement (disp) using the widely used mtcars data set.

```
m1 = umxRAM("tim", data = mtcars,
umxPath(c("wt", "disp"), to = "mpg"),
```

```
umxPath("wt", with = "disp"),
umxPath(v.m. = c("wt", "disp", "mpg"))
)
```

As you can see, most of the work is done by umxPath(). umxRAM wraps these paths up, takes the data = input, and then internally sets up all the labels and start values for the model, runs it, and calls umxSummary(), and plot.MxModel().

Try it, or one of the several models in the examples at the bottom of this page.

A common error is to include data in the main list, a bit like saying lm(y ~ x + df) instead of lm(y ~ x, data = df).

nb: Because it uses the presence of a variable in the data to detect if a variable is latent or not, umxRAM needs data at build time.

String Syntax

Here is an example using lavaan syntax (for more, see umxLav2RAM())

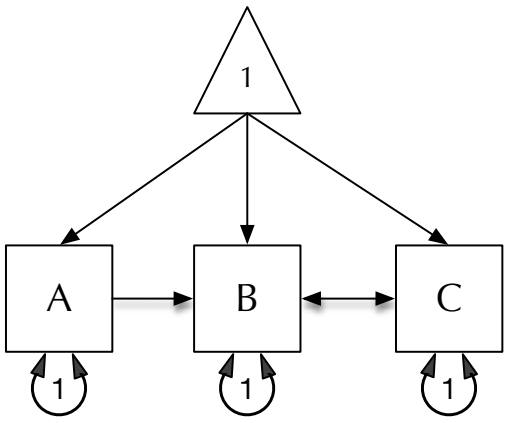
```
m1 = umxRAM("mpg ~ wt + disp", data = mtcars)
```

Sketch mode

If you are at the "sketching" stage of theory consideration, umxRAM supports a simple vector of manifest names to work with.

```
m1 = umxRAM("sketch", data = c("A", "B", "C"),
umxPath("A", to = "B"),
umxPath("B", with = "C"),
umxPath(v.m. = c("A", "B", "C"))
)
```

Will create this figure:



Usage

```
umxRAM(
  model = NA,
  ...,
  data = NULL,
  name = NA,
  group = NULL,
  group.equal = NULL,
  suffix = "",
  comparison = TRUE,
  type = c("Auto", "FIML", "cov", "cor", "WLS", "DWLS", "ULS"),
  allContinuousMethod = c("cumulants", "marginals"),
  autoRun = getOption("umx_auto_run"),
  tryHard = c("no", "yes", "ordinal", "search"),
  std = FALSE,
  refModels = NULL,
  remove_unused_manifests = TRUE,
  independent = NA,
  setValues = TRUE,
  optimizer = NULL,
  verbose = FALSE,
  std.lv = FALSE,
  lavaanMode = c("sem", "lavaan"),
  printTab = FALSE,
  show = "deprecated"
)
```

Arguments

model	A model to update (or set to string to use as name for new model)
...	umxPaths, mxThreshold objects, etc.
data	data for the model. Can be an mxData() or a data.frame
name	A friendly name for the model
group	(optional) Column name to use for a multi-group model (default = NULL)
group.equal	In multi-group models, what to equate across groups (default = NULL)
suffix	String to append to each label (useful if model will be used in a multi-group model)
comparison	Compare the new model to the old (if updating an existing model: default = TRUE)
type	One of "Auto", "FIML", "cov", "cor", "WLS", "DWLS", "ULS"
allContinuousMethod	"cumulants" or "marginals". Used in all-continuous WLS data to determine if a means model needed.
autoRun	Whether to run the model (default), or just to create it and return without running.

tryHard	Default ('no') uses normal mxRun. "yes" uses mxTryHard. Other options: "ordinal", "search"
std	Whether to show standardized estimates, raw (NULL print fit only)
refModels	pass in reference models if available. Use FALSE to suppress computing these if not provided.
remove_unused_manifests	Whether to remove variables in the data to which no path makes reference (defaults to TRUE)
independent	Whether the model is independent (default = NA)
setValues	Whether to generate likely good start values (Defaults to TRUE)
optimizer	optionally set the optimizer (default NULL does nothing)
verbose	Whether to tell the user what latents and manifests were created etc. (Default = FALSE)
std.lv	Whether to auto standardize latent variables when using string syntax (default = FALSE)
lavaanMode	Defaults when building out string syntax default = "sem" (alternative is "lavaan", with very few defaults)
printTab	(for string input, whether to output a table of paths (FALSE)
show	Deprecated

Details

Comparison with OpenMx and mxModel

umxRAM differs from mxModel in the following ways:

1. You don't need to set type = "RAM".
2. You don't need to list manifestVars (they are detected from path usage).
3. You don't need to list latentVars (detected as anything in paths but not in mxData).
4. You don't need to create mxData when you already have a data.frame.
5. You add data with data = (as elsewhere in R, e.g. lm()).
6. You don't need to add labels: paths are automatically labelled "a_to_b" etc.
7. You don't need to set start values, they will be done for you.
8. You don't need to mxRun the model: it will run automatically, and print a summary.
9. You don't need to run summary: with autoRun=TRUE, it will print a summary.
10. You get a plot of the model with estimates on the paths, including multiple groups.
11. Less typing: umxPath() offers powerful verbs to describe paths.
12. Supports a subset of lavaan string input.

Start values. Currently, manifest variable means are set to the observed means, residual variances are set to 80% of the observed variance of each variable, and single-headed paths are set to a positive starting value (currently .9). *note*: The start-value strategy is subject to improvement, and will be documented in the help for umxRAM().

Comparison with other software

Black-box software, defaults, and automatic addition of paths.

Some SEM software does a lot of behind-the-scenes defaulting and path addition. If you want this, I'd say use umxRAM with lavaan string input.

Value

- mxModel()

References

- https://tbates.github.io, https://github.com/tbates/umx

See Also

umxPath(), umxSummary(), plot(), parameters(), umxSuperModel(), umxLav2RAM()

Other Core Modeling Functions: umxAlgebra(), umxMatrix(), umxModify(), umxPath(), umxRun(), umxSummary(), umxSuperModel(), umx

Examples

```
# ============================================
# = 1. Here's a simple example with raw data =
# ============================================
mtcars$litres = mtcars$disp/61.02
m1 = umxRAM("tim", data = mtcars,
umxPath(c("wt", "litres"), to = "mpg"),
umxPath("wt", with = "litres"),
umxPath(v.m. = c("wt", "litres", "mpg"))
)

# 2. Use parameters to see the parameter estimates and labels
parameters(m1)

# And umxSummary to get standardized parameters, CIs etc from the run model.
umxSummary(m1, std=TRUE)
# |name           | Std.Estimate| Std.SE|CI                   |
# |:--------------|------------:|------:|:--------------------|
# |wt_to_mpg      |        -0.54|   0.17|-0.54 [-0.89, -0.2]  |
# |disp_to_mpg    |        -0.36|   0.18|-0.36 [-0.71, -0.02] |
# |mpg_with_mpg   |         0.22|   0.07|0.22 [0.08, 0.35]    |
# |wt_with_wt     |         1.00|   0.00|1 [1, 1]             |
# |b1             |         0.89|   0.04|0.89 [0.81, 0.96]    |
# |disp_with_disp |         1.00|   0.00|1 [1, 1]             |

## Not run:
# 3. Of course you can plot the model
plot(m1)
plot(m1, std=TRUE, means=FALSE)
plot(m1, std = TRUE, means=FALSE, strip= TRUE, resid = "line")
```

```
# ================================================
# = lavaan string example (more at ?umxLav2RAM) =
# ================================================
m1 = umxRAM(data = mtcars, "#modelName
 mpg ~ wt + disp")


# ======================
# = A multi-group model =
# ======================

m1 = umxRAM("tim", data = mtcars, group = "am",
umxPath(c("wt", "litres"), to = "mpg"),
umxPath("wt", with = "litres"),
umxPath(v.m. = c("wt", "litres", "mpg"))
)
# In this model, all parameters are equated across the two groups.

# ====================================
# = A cov model, with steps laid out =
# ====================================

# *note*: The variance of displacement is in cubic inches and is very large.
# to help the optimizer, one might, say, multiply disp *.016 to work in litres
tmp = mtcars; tmp$disp= tmp$disp *.016

# We can just give the raw data and ask for it to be made into type cov:
m1 = umxRAM("tim", data = tmp, type="cov",
umxPath(c("wt", "disp"), to = "mpg"),
umxPath("wt", with = "disp"),
umxPath(var = c("mpg", "wt", "disp"))
)

# (see ?umxPath for more nifty options making paths...)

# ========================================
# = umxRAM can also accept mxData as data =
# ========================================
# For convenience, list up the manifests you will be using

selVars = c("mpg", "wt", "disp")
tmp = mtcars; tmp$disp= tmp$disp *.016
myCov = mxData(cov(tmp[, selVars]), type = "cov", numObs = nrow(mtcars) )

m1 = umxRAM("tim", data = myCov,
umxPath(c("wt", "disp"), to = "mpg"),
umxPath("wt", with = "disp"),
umxPath(var = selVars)
)


# ======================
```

```
# = umxRAM supports WLS =
# =======================

# 1. Run an all-continuous WLS model
 mw = umxRAM("raw", data = mtcars[, c("mpg", "wt", "disp")],
type = "WLS", allContinuousMethod = "cumulants",
  umxPath(var = c("wt", "disp", "mpg")),
  umxPath(c("wt", "disp"), to = "mpg"),
  umxPath("wt", with = "disp"),
     umxPath(var = c("wt", "disp", "mpg"))
 )
# 2. Switch to marginals to support means
 mw = umxRAM("raw", data = mtcars[, c("mpg", "wt", "disp")],
type = "WLS", allContinuousMethod= "marginals",
  umxPath(var = c("wt", "disp", "mpg")),
  umxPath(c("wt", "disp"), to = "mpg"),
  umxPath("wt", with = "disp"),
     umxPath(var = c("wt", "disp", "mpg"))
 )


# ================================
# = Using umxRAM in Sketch mode =
# ================================
# No data needed: just list variable names!
# Resulting model will be plotted automatically
m1 = umxRAM("what does unique pairs do, I wonder", data = c("A", "B", "C"),
   umxPath(unique.pairs = c("A", "B", "C"))
)

m1 = umxRAM("ring around the rosey", data = c("B", "C"),
  umxPath(fromEach = c("A", "B", "C"))
)

m1 = umxRAM("fromEach with to", data = c("B", "C"),
   umxPath(fromEach = c("B", "C"), to= "D")
)

m1 = umxRAM("CFA_sketch", data = paste0("x", 1:4),
umxPath("g", to = paste0("x", 1:4)),
umxPath(var = paste0("x", 1:4)),
umxPath(v1m0 = "g")
)

# ================================================
# = This is an example of using your own labels:  =
#   umxRAM will not over-ride them                =
# ================================================
m1 = umxRAM("tim", data = mtcars, type="cov",
umxPath(c("wt", "disp"), to = "mpg"),
umxPath(cov = c("wt", "disp"), labels = "b1"),
umxPath(var = c("wt", "disp", "mpg"))
)
```

```
omxCheckEquals(m1$S$labels["disp", "wt"], "b1") # label preserved
m1$S$labels
#      mpg             wt            disp
# mpg  "mpg_with_mpg"  "mpg_with_wt" "disp_with_mpg"
# wt   "mpg_with_wt"   "wt_with_wt"  "b1"
# disp "disp_with_mpg" "b1"          "disp_with_disp"
parameters(m1)


## End(Not run)
```

---

umxRAM2Lav *Convert a RAM model to a lavaan string*

Description

Takes an OpenMx RAM model and creates the corresponding lavaan syntax string.

This function is at the alpha quality stage, and **should be expected to have bugs**. Also likely to change functionality and even parameters as new features are supported (e.g. groups) and lavaan-style strings exported. Several features are not yet supported. Let me know if you would like them.

Usage

```
umxRAM2Lav(model)
```

Arguments

model	an OpenMx RAM model

Value

A lavaan syntax string, e.g. "A~~B"

See Also

- [umxLav2RAM()], [umxRAM()]

Other Miscellaneous Utility Functions: install.OpenMx(), qm(), umxBrownie(), umxLav2RAM(), umxVersion(), umx_array_shift(), umx_find_object(), umx_msg(), umx_open_CRAN_page(), umx_pad(), umx_print(), umx_score_scale(), umx

Examples

```
umxRAM2Lav(umxLav2RAM("x ~ y", autoRun = FALSE, printTab = FALSE, lavaanMode = "lavaan"))
```

umxReduce *Reduce models, and report the results.*

Description

Given a umx model (currently umxACE and umxGxE are supported - ask for more!) umxReduce will conduct a formalised reduction process. It will also report Akaike weights are also reported showing relative support across models.

Specialized functions are called for different type of input:

1. **GxE model reduction** For umxGxE() models umxReduceGxE() is called.
2. **ACE model reduction** For umxACE() models,umxReduceACE() is called.

umxReduce reports the results in a table. Set the format of the table with umx_set_table_format(), or set report= "html" to open a table for pasting into a word processor.

umxReduce is a work in progress, with more automatic reductions coming as demand emerges. I am thinking for RAM models to drop NS paths, and report that test.

Usage

```
umxReduce(
  model,
  report = c("markdown", "inline", "html"),
  baseFileName = "tmp",
  ...
)
```

Arguments

model	The mxModel() which will be reduced.
report	How to report the results. "html" = open in browser
baseFileName	(optional) custom filename for html output (defaults to "tmp")
...	Other parameters to control model summary

References

- Wagenmakers, E.J., & Farrell, S. (2004). AIC model selection using Akaike weights. *Psychonomic Bulletin and Review*, **11**, 192-196. doi:

See Also

umxReduceGxE(), umxReduceACE()

Other Reporting Functions: loadings.MxModel(), umxAPA(), umxFactorScores(), umxGetParameters(), umxParameters(), umx_aggregate(), umx_names(), umx_time(), umx

Other Twin Reporting Functions: umxPlotCP(), umxPlotDoC(), umxReduceACE(), umxReduceGxE(), umxSummarizeTwinData(), umxSummaryACEcov(), umxSummaryACEv(), umxSummaryACE(), umxSummaryCP(),

umxSummaryDoC(), umxSummaryGxEbiv(), umxSummaryGxE(), umxSummaryIP(), umxSummarySexLim(), umxSummarySimplex(), umx

---

umxReduceACE *Reduce an ACE model.*

Description

This function can perform model reduction on umxACE() models, testing dropping A and C, as well as an ADE or ACE model, displaying the results in a table, and returning the best model.

Usage

```
umxReduceACE(
  model,
  report = c("markdown", "inline", "html", "report"),
  baseFileName = "tmp",
  intervals = TRUE,
  ...
)
```

Arguments

model	an ACE or ADE mxModel() to reduce
report	How to report the results. "html" = open in browser
baseFileName	(optional) custom filename for html output (defaults to "tmp")
intervals	Recompute CIs (if any included) on the best model (default = TRUE)
...	Other parameters to control model summary

Details

It is designed for testing univariate models. You can offer up either the ACE or ADE base model.

Suggestions for more sophisticated automation welcomed!

Value

Best fitting model

References

- Wagenmakers, E.J., & Farrell, S. (2004). AIC model selection using Akaike weights. *Psychonomic Bulletin and Review*, **11**, 192-196. doi:

See Also

umxReduceGxE(), umxReduce()

Other Twin Reporting Functions: umxPlotCP(), umxPlotDoC(), umxReduceGxE(), umxReduce(), umxSummarizeTwinData(), umxSummaryACEcov(), umxSummaryACEv(), umxSummaryACE(), umxSummaryCP(), umxSummaryDoC(), umxSummaryGxEbiv(), umxSummaryGxE(), umxSummaryIP(), umxSummarySexLim(), umxSummarySimplex(), umx

Examples

```
## Not run:
data(twinData)
mzData <- subset(twinData, zygosity == "MZFF")
dzData <- subset(twinData, zygosity == "DZFF")
m1 = umxACE(selDVs = "bmi", dzData = dzData, mzData = mzData, sep = "")
m2 = umxReduce(m1)
umxSummary(m2)
m1 = umxACE(selDVs = "bmi", dzData = dzData, mzData = mzData, sep = "", dzCr = .25)
m2 = umxReduce(m1)


## End(Not run)
```

---

umxReduceGxE *Reduce a GxE model.*

Description

This function can perform model reduction for umxGxE() models, testing dropping a,c & e, as well as c & c, a & a‘ etc.

It reports the results in a table. Set the format of the table with umx_set_table_format(). Or set report = "html" to open a table for pasting into a word processor.

In addition to printing a table, the function returns the preferred model.

Usage

```
umxReduceGxE(
  model,
  report = c("markdown", "inline", "html", "report"),
  baseFileName = "tmp_gxe",
  tryHard = c("no", "yes", "ordinal", "search"),
  ...
)
```

Arguments

model	An mxModel() to reduce.
report	How to report the results. "html" = open in browser.
baseFileName	(optional) custom filename for html output (defaults to "tmp").
tryHard	Default ('no') uses normal mxRun. "yes" uses mxTryHard. Other options: "ordinal", "search"
...	Other parameters to control model summary.

Value

best model

References

- Wagenmakers, E.J., & Farrell, S. (2004). AIC model selection using Akaike weights. *Psychonomic Bulletin and Review*, **11**, 192-196. doi:.

See Also

umxReduceACE(), umxReduce()

Other Twin Reporting Functions: umxPlotCP(), umxPlotDoC(), umxReduceACE(), umxReduce(), umxSummarizeTwinData(), umxSummaryACEcov(), umxSummaryACEv(), umxSummaryACE(), umxSummaryCP(), umxSummaryDoC(), umxSummaryGxEbiv(), umxSummaryGxE(), umxSummaryIP(), umxSummarySexLim(), umxSummarySimplex(), umx

Examples

```
## Not run:
model = umxReduce(model)

## End(Not run)
```

---

umxRenameMatrix *Rename a umxMatrix (in a model)*

Description

Rename a umxMatrix, including updating its labels

Usage

```
umxRenameMatrix(x, matrixName, name)
```

Arguments

x	A model or matrix
matrixName	Name of the matrix
name	The new name

Value

- updated matrix or model with updated matrix in it.

See Also

Other xmu internal not for end user: umxModel(), umxTwinMaker(), umx_APA_pval(), umx_fun_mean_sd(), umx_get_bracket_addresses(), umx_make(), umx_standardize(), umx_string_to_algebra(), umx, xmuHasSquareBrackets(), xmuLabel_MATRIX_Model(), xmuLabel_Matrix(), xmuLabel_RAM_Model(), xmuMI(), xmuMakeDeviationThresholdsMatrices(), xmuMakeOneHeadedPathsFromPathList(), xmuMakeTwoHeadedPathsFromPathList(), xmuMaxLevels(), xmuMinLevels(), xmuPropagateLabels(), xmuRAM2Ordinal(), xmuTwinSuper_Continuous(), xmuTwinUpgradeMeansToCovariateModel(), xmu_CI_merge(), xmu_CI_stash(), xmu_DF_to_mxData_TypeCov(), xmu_PadAndPruneForDefVars(), xmu_cell_is_on(), xmu_check_levels_identical(), xmu_check_needs_means(), xmu_check_variance(), xmu_clean_label(), xmu_data_missing(), xmu_data_swap_a_block(), xmu_describe_data_WLS(), xmu_dot_make_paths(), xmu_dot_make_residuals(), xmu_dot_maker(), xmu_dot_move_ranks(), xmu_dot_rank_str(), xmu_extract_column(), xmu_get_CI(), xmu_lavaan_process_group(), xmu_make_TwinSuperModel(), xmu_make_bin_cont_pair_data(), xmu_make_mxData(), xmu_match.arg(), xmu_name_from_lavaan_str(), xmu_path2twin(), xmu_path_regex(), xmu_safe_run_summary(), xmu_set_sep_from_suffix(), xmu_show_fit_or_comparison(), xmu_simplex_corner(), xmu_standardize_ACEcov(), xmu_standardize_ACEv(), xmu_standardize_ACE(), xmu_standardize_CP(), xmu_standardize_IP(), xmu_standardize_RAM(), xmu_standardize_SexLim(), xmu_standardize_Simplex(), xmu_start_value_list(), xmu_starts(), xmu_twin_add_WeightMatrices(), xmu_twin_check(), xmu_twin_get_var_names(), xmu_twin_upgrade_selDvs2SelVars()

Examples

```
data(twinData) # ?twinData from Australian twins.
twinData[, c("ht1", "ht2")] = twinData[, c("ht1", "ht2")] * 10
mzData = twinData[twinData$zygosity %in% "MZFF", ]
dzData = twinData[twinData$zygosity %in% "DZFF", ]
m1  = umxACE(selDVs= "ht", sep= "", dzData= dzData, mzData= mzData, autoRun= FALSE)
tmp = umxRenameMatrix(m1$top, matrixName = "a", name="hello")
umx_check(tmp$hello$labels == "hello_r1c1") # new is there
umx_check(is.null(tmp$a))                   # old is gone
```

---

umxRotate *Generic SEM factor model loading rotation function*

Description

See umxRotate.MxModelCP() to rotate the factor loadings of a umxCP() model

Usage

```
umxRotate(
  model,
  rotation = c("varimax", "promax"),
  tryHard = "yes",
  freeLoadingsAfter = TRUE,
  verbose = TRUE
)
```

Arguments

model	a model to rotate
rotation	name of the rotation.
tryHard	Default ("yes") is to tryHard
freeLoadingsAfter	Whether to keep the rotated loadings fixed (Default, free them again)
verbose	print detail about the rotation

Value

- Rotated solution

See Also

Other Reporting functions: RMSEA.MxModel(), RMSEA.summary.mxmodel(), RMSEA(), extractAIC.MxModel(), loadings(), residuals.MxModel(), umxCI_boot(), umxCI(), umxCompare(), umxConfint(), umxExpCov(), umxExpMeans(), umxFitIndices(), umxPlotACEv(), umxSummary.MxModel()

---

umxRotate.MxModelCP *Rotate a CP solution*

Description

Rotate a CP solution. Should work with rotations provided in library("GPArotation") and library("psych"), e.g

Orthogonal: "varimax", "quartimax", "bentlerT", "equamax", "varimin", "geominT" and "bifactor"

Oblique: "Promax", "promax", "oblimin", "simplimax", "bentlerQ", "geominQ", "biquartimin" and "cluster"

Usage

```
## S3 method for class 'MxModelCP'
umxRotate(
  model,
  rotation = c("varimax", "promax"),
  tryHard = "yes",
  freeLoadingsAfter = TRUE,
  verbose = TRUE
)
```

Arguments

model	a umxCP() model to rotate.
rotation	name of the rotation.
tryHard	Default ("yes") is to tryHard.
freeLoadingsAfter	return the model with factor loadings free (default) or fixed in the new locations.
verbose	print detail about the rotation

Details

This works by taking the common-pathways loadings matrix from a solved umxCP() model, rotating these, placing them back into the loadings matrix, re-estimating the model with the parameters fixed at this rotation, then return the new model.

Value

- Rotated solution.

See Also

- umxCP()

Other Twin Modeling Functions: plot.MxModelTwinMaker(), power.ACE.test(), umxACEcov(), umxACEv(), umxACE(), umxCP(), umxDoCp(), umxDoC(), umxGxE_window(), umxGxEbiv(), umxGxE(), umxIP(), umxSexLim(), umxSimplex(), umx

Examples

```
# Rotate a CP solution(param)
# Common pathway model rotation
## Not run:
library(umx)
# Fit 3 factor CPM
data(GFF)
selDVs = c("gff", "fc", "qol", "hap", "sat", "AD")
m1 = umxCP(selDVs = selDVs, nFac = 2, data = data, tryHard = "yes")
m2 = umxRotate(m1, rotation = "varimax",  tryHard = "yes")
```

```
## End(Not run)
```

---

umxRun *umxRun: Run an mxModel*

Description

umxRun is a version of mxRun() which can run also set start values, labels, and run multiple times It can also calculate the saturated and independence likelihoods necessary for most fit indices. **Note** this is not needed for umxRAM models or twin models - it is just a convenience to get base OpenMx models to run.

Usage

```
umxRun(
  model,
  n = 1,
  calc_SE = TRUE,
  calc_sat = TRUE,
  setValues = FALSE,
  setLabels = FALSE,
  intervals = FALSE,
  comparison = NULL
)
```

Arguments

model	The mxModel() you wish to run.
n	The maximum number of times you want to run the model trying to get a code green run (defaults to 1)
calc_SE	Whether to calculate standard errors (ignored when n = 1) for the summary (if you use mxCI() or umxCI(), you can turn this off)
calc_sat	Whether to calculate the saturated and independence models (for raw mxData() mxModel()s) (defaults to TRUE - why would you want anything else?)
setValues	Whether to set the starting values of free parameters (default = FALSE)
setLabels	Whether to set the labels (default = FALSE)
intervals	Whether to run mxCI confidence intervals (default = FALSE) intervals = FALSE
comparison	Whether to run umxCompare() after umxRun

Value

- mxModel()

References

- https://www.github.com/tbates/umx

See Also

Other Core Modeling Functions: umxAlgebra(), umxMatrix(), umxModify(), umxPath(), umxRAM(), umxSummary(), umxSuperModel(), umx

Examples

```
require(umx)
data(demoOneFactor)
latents  = c("G")
manifests = names(demoOneFactor)
m1 = mxModel("One Factor", type = "RAM",
manifestVars = manifests, latentVars = latents,
mxPath(from = latents  , to = manifests),
mxPath(from = manifests, arrows = 2),
mxPath(from = latents  , arrows = 2, free = FALSE, values = 1.0),
mxData(cov(demoOneFactor), type = "cov", numObs=500)
)

m1 = umxRun(m1) # just run: will create saturated model if needed
## Not run:
m1 = umxRun(m1, setValues = TRUE, setLabels = TRUE) # set start values and label all parameters
umxSummary(m1, std = TRUE)
m1 = mxModel(m1, mxCI("G_to_x1")) # add one CI
m1 = mxRun(m1, intervals = TRUE)
residuals(m1, run = TRUE) # get CIs on all free parameters
confint(m1) # OpenMx's SE-based CIs
umxConfint(m1, run = TRUE) # get likelihood-based CIs on all free parameters
m1 = umxRun(m1, n = 10) # re-run up to 10 times if not green on first run

## End(Not run)
```

---

umxSetParameters *Change or fix parameters (e.g. their values, labels, bounds, ..) in a model.*

Description

umxSetParameters is used to alter values, and other parameter properties in an mxModel(). A common use is setting new values and changing parameters from free to false. *Note*: If you just want to modify and re-run a model, you probably want umxModify().

Usage

```
umxSetParameters(
  model,
  labels,
  free = NULL,
  values = NULL,
```

```
  newlabels = NULL,
  lbound = NULL,
  ubound = NULL,
  indep = FALSE,
  strict = TRUE,
  name = NULL,
  regex = FALSE,
  test = FALSE
)
```

Arguments

model	an mxModel() to WITH
labels	= labels to find
free	= new value for free
values	= new values
newlabels	= newlabels
lbound	= value for lbound
ubound	= value for ubound
indep	= whether to look in indep models
strict	whether to complain if labels not found
name	= new name for the returned model
regex	patterns to match for labels (or if TRUE, use labels as regular expressions)
test	Just show what you would do? (defaults to FALSE)

Details

Using umxSetParameters, you use labels= to select the parameters you want to update. You can set their free/fixed state with free=, and set new values with values = . Likewise for bounds.

umxSetParameters supports pattern matching (regular expressions) to select labels. Set regex= to a regular expression matching the labels you want to select. e.g. "G_to_.*" would match "G_to_anything".

Details Internally, umxSetParameters is equivalent to a call to omxSetParameters where you have the ability to generate a pattern-based label list, and, because this can create duplicate labels, we also call omxAssignFirstParameters() to equate the start values for parameters which now have identical labels.

Value

- mxModel()

References

- https://github.com/tbates/umx, https://tbates.github.io

See Also

- umxModify(), umxLabel()

Other Modify or Compare Models: umxEquate(), umxFixAll(), umxMI(), umxModify(), umxUnexplainedCausalNexus(), umx

Examples

```
require(umx)
data(demoOneFactor)
latents  = c("G")
manifests = names(demoOneFactor)
m1 = umxRAM("One Factor", data = mxData(demoOneFactor[1:80,], type = "raw"),
umxPath(from = latents, to = manifests),
umxPath(v.m. = manifests),
umxPath(v1m0 = latents)
)
parameters(m1)
# Match all labels
umxSetParameters(m1, regex = "^", newlabels= "m1_", test = TRUE)
# Change path to x1 to x2, equating these two paths
m2 = umxSetParameters(m1, "G_to_x1", newlabels= "G_to_x2", test = FALSE)
parameters(m2)
```

---

umxSexLim *Multivariate sex limitation twin model*

Description

Multivariate twin analysis allowing for sex limitation (factors operate differently in males vs. females) based on a correlated factors model. With 5-groups of twins, this model allows for both Quantitative and Qualitative Sex-Limitation.

Quantitative differences refer to different amounts of phenotypic variance produced by the same A, C, or E components when operating in one sex compared to the other sex.

Qualitative differences refer to phenotypic variance attributable to an A, C, or E component which operates in one sex one but not in the other.

The correlation approach ensures that variable order does not affect the ability of the model to account for DZOS data.

1. Nonscalar Sex Limitation

Allow quantitative (distinct male and female paths) and qualitative sex differences on A or C. Allows distinct between variable correlations (Ra, Rc and Re) for males and for females. Male-Female correlations also free (Rao or Rco free in DZO group).

2. Scalar Sex Limitation

Quantitative sex differences only (distinct Male and female paths). Just one set of Ra, Rc and Re between variables (same for males and females)

3. Homogeneity

This is the model assumed by the basic ACE model: equal variance components in both sexes. Different means may be allowed for males and females.

Usage

```
umxSexLim(
  name = "sexlim",
  selDVs,
  mzmData,
  dzmData,
  mzfData,
  dzfData,
  dzoData,
  sep = NA,
  A_or_C = c("A", "C"),
  sexlim = c("Nonscalar", "Scalar", "Homogeneity"),
  dzAr = 0.5,
  dzCr = 1,
  autoRun = getOption("umx_auto_run"),
  tryHard = c("no", "yes", "ordinal", "search"),
  optimizer = NULL
)
```

Arguments

name	The name of the model (Default = "sexlim")
selDVs	BASE NAMES of the variables in the analysis. You MUST provide sep.
mzmData	Dataframe containing the MZ male data.
dzmData	Dataframe containing the DZ male data.
mzfData	Dataframe containing the MZ female data.
dzfData	Dataframe containing the DZ female data.
dzoData	Dataframe containing the DZ opposite-sex data (be sure and get in right order).
sep	Suffix used for twin variable naming. Allows using just the base names in selVars.
A_or_C	Whether to model sex-limitation on A or on C. (Defaults to "A").
sexlim	Which model type: "Nonscalar" (default), "Scalar", or "Homogeneity".
dzAr	The DZ genetic correlation (defaults to .5, vary to examine assortative mating).
dzCr	The DZ "C" correlation (defaults to 1: set to .25 to make an ADE model).
autoRun	Whether to mxRun the model (default TRUE: the estimated model will be returned).
tryHard	Default ('no') uses normal mxRun. "yes" uses mxTryHard. Other options: "ordinal", "search"
optimizer	optionally set the optimizer. Default (NULL) does nothing.

Details

A or C

Due to limitations on the degrees of freedom allowed by the twin model, we can model qualitative sex differences for only one of A or C at a time.

notes: There is a half-way house model of heterogeneity in which a, c, and e components are scaled by a scalar constant in one sex.

General restrictions: Assumes means and variances can be equated across birth order within zygosity groups.

Value

- `mxModel()` of subclass mxModel.CFSexLim

References

- Neale et al. (2006). Multivariate genetic analysis of sex-lim and GxE interaction. *Twin Research & Human Genetics*, **9**, pp. 481–489.

See Also

`umxSummarySexLim()`, `umxPlotSexLim()`

Other Twin Modeling Functions: `plot.MxModelTwinMaker()`, `power.ACE.test()`, `umxACEcov()`, `umxACEv()`, `umxACE()`, `umxCP()`, `umxDoCp()`, `umxDoC()`, `umxGxE_window()`, `umxGxEbiv()`, `umxGxE()`, `umxIP()`, `umxRotate.MxModelCP()`, `umxSimplex()`, `umx`

Examples

```
# ========================
# = Load and Process Data =
# ========================
## Not run:
require(umx)
data("us_skinfold_data")
# Rescale vars
us_skinfold_data[, c('bic_T1', 'bic_T2')] = us_skinfold_data[, c('bic_T1', 'bic_T2')]/3.4
us_skinfold_data[, c('tri_T1', 'tri_T2')] = us_skinfold_data[, c('tri_T1', 'tri_T2')]/3
us_skinfold_data[, c('caf_T1', 'caf_T2')] = us_skinfold_data[, c('caf_T1', 'caf_T2')]/3
us_skinfold_data[, c('ssc_T1', 'ssc_T2')] = us_skinfold_data[, c('ssc_T1', 'ssc_T2')]/5
us_skinfold_data[, c('sil_T1', 'sil_T2')] = us_skinfold_data[, c('sil_T1', 'sil_T2')]/5

# Data for each of the 5 twin-type groups
mzmData = subset(us_skinfold_data, zyg == 1)
mzfData = subset(us_skinfold_data, zyg == 2)
dzmData = subset(us_skinfold_data, zyg == 3)
dzfData = subset(us_skinfold_data, zyg == 4)
dzoData = subset(us_skinfold_data, zyg == 5)

umxSummarizeTwinData(us_skinfold_data, selVars="bic",zyg="zyg", sep="_T",
MZFF=2, DZFF=4, MZMM=1, DZMM=3, DZOS=5
)

# ========================
# = Run univariate example =
```

```
# ==========================

m1 = umxSexLim(selDVs = "bic", sep = "_T", A_or_C = "A", tryHard = "yes",
mzmData = mzmData, dzmData = dzmData,
mzfData = mzfData, dzfData = dzfData,
dzoData = dzoData
)

# Drop qualitative sex limitation
m1a = umxModify(m1, regex = "^Rao_", value=1, name = "no_qual", comparison = TRUE)


# Equate a, ac, and try ace across m & f in scalar model
m1b = umxModify(m1a, regex = "^a[fm]_", newlabels="a_", name = "eq_a_no_qual", comparison = TRUE)
m1c = umxModify(m1b, regex = "^c[fm]_", newlabels="c_", name = "eq_ac_no_qual", comparison = TRUE)
m1d = umxModify(m1c, regex = "^e[fm]_", newlabels="e_", name = "eq_ace_no_qual", comparison = TRUE)
umxCompare(m1, c(m1a, m1b, m1c, m1d))

# ============================
# = Scalar Sex Limitation =
# ============================

m2 = umxSexLim(selDVs = "bic", sep = "_T", sexlim = "Scalar", tryHard = "yes",
mzmData = mzmData, dzmData = dzmData,
mzfData = mzfData, dzfData = dzfData,
dzoData = dzoData
)

# Show our manual drop of qualitative is the same as umxSexLim with sexlim= "scalar"s
umxCompare(m1a, m2)

# ===============
# = Homogeneity =
# ===============

m3 = umxSexLim(selDVs = "bic", sep = "_T", sexlim = "Homogeneity", tryHard = "yes",
mzmData = mzmData, dzmData = dzmData,
mzfData = mzfData, dzfData = dzfData,
dzoData = dzoData
)
umxCompare(m1, c(m2, m3))

# ========================================
# = Bivariate example with manual reduction =
# ========================================
m1 = umxSexLim(selDVs = c("bic", "tri"), sep = "_T", A_or_C = "A", tryHard="yes",
mzmData = mzmData, dzmData = dzmData,
mzfData = mzfData, dzfData = dzfData,
dzoData = dzoData
)

# Scalar sex limitation (same correlation among components for m and f)
m2 = umxSexLim(selDVs = c("bic", "tri"), sep = "_T",
```

```
A_or_C = "A", tryHard="yes", sexlim="Scalar",
mzmData = mzmData, dzmData = dzmData,
mzfData = mzfData, dzfData = dzfData,
dzoData = dzoData
)
# Drop qualitative sex limitation
#  Distinct af and am (& c & e), but shared Ra (& Rc & Re) between variables
#   i.e., same correlations for males and females.
m1a = umxModify(m1 , regex = "^Ra[mfo]_", newlabels="^Ra_", name = "no_qual_a", comparison = TRUE)
m1b = umxModify(m1a, regex = "^Rc[mfo]_", newlabels="^Rc_", name = "no_qual_ac", comparison = TRUE)
m1c = umxModify(m1b, regex = "^Re[mfo]_", newlabels="^Re_", name = "no_qual_ace", comparison = TRUE)
umxCompare(m1, c(m1a, m1b, m1c, m2))

# In one smart regular expression
m2 = umxModify(m1, regex = "^R([ace])[fmo]_", newlabels = "R\\1_",
  name = "scalar", comparison = TRUE)

# Equate a, ac, and try ace across m & f in scalar model
m2a = umxModify(m2 , regex = "^a[fm]_", newlabels="a_", name = "eq_a_no_qual"  , comparison = TRUE)
m2b = umxModify(m2a, regex = "^c[fm]_", newlabels="c_", name = "eq_ac_no_qual" , comparison = TRUE)
m2c = umxModify(m2b, regex = "^e[fm]_", newlabels="e_", name = "eq_ace_no_qual", comparison = TRUE)
umxCompare(m1, c(m1a, m1b, m1c, m1d))

# ============================
# = Run multi-variate example =
# ============================
# Variables for Analysis
selDVs = c('ssc','sil','caf','tri','bic')
selDVs = c('ssc','tri','bic')
m1 = umxSexLim(selDVs = selDVs, sep = "_T", A_or_C = "A", tryHard = "yes",
mzmData = mzmData, dzmData = dzmData,
 mzfData = mzfData, dzfData = dzfData, dzoData = dzoData
)

m2 = umxSexLim(selDVs = selDVs, sep = "_T", A_or_C = "A", sexlim = "Nonscalar",
tryHard = "yes",
mzmData = mzmData, dzmData = dzmData,
 mzfData = mzfData, dzfData = dzfData, dzoData = dzoData
)

# umxSummary(m1)
# summary(m1)
# summary(m1)$Mi

## End(Not run)
```

---

umxSimplex *Build and run a simplex twin model (not ready for use!)*

Description

The simplex model provides a powerful tool for theory-based decomposition of genetic and environmental differences. umxSimplex makes a 2-group simplex twin model.

Usage

```
umxSimplex(
  name = "simplex",
  selDVs,
  dzData,
  mzData,
  sep = NULL,
  equateMeans = TRUE,
  dzAr = 0.5,
  dzCr = 1,
  addStd = TRUE,
  addCI = TRUE,
  autoRun = getOption("umx_auto_run"),
  tryHard = c("no", "yes", "ordinal", "search"),
  optimizer = NULL
)
```

Arguments

name	The name of the model (defaults to "simplex")
selDVs	The BASENAMES of the variables i.e., c(obese), not c(obese_T1, obese_T2)
dzData	The DZ dataframe
mzData	The MZ dataframe
sep	The string preceding the final numeric twin identifier (often "_T") Combined with selDVs to form the full var names, i.e., just "dep" –> c("dep_T1", "dep_T2")
equateMeans	Whether to equate the means across twins (defaults to TRUE).
dzAr	The DZ genetic correlation (default = .5. Vary to examine assortative mating).
dzCr	The DZ "C" correlation (defaults = 1. To make an ADE model, set = .25).
addStd	Whether to add the algebras to compute a std model (default = TRUE).
addCI	Whether to add the interval requests for CIs (default = TRUE).
autoRun	Whether to run the model (default), or just to create it and return without running.
tryHard	Default ('no') uses normal mxRun. "yes" uses mxTryHard. Other options: "ordinal", "search"
optimizer	Optionally set the optimizer (default NULL does nothing).

Details

This code is beta quality: **not** for publication use. It will be completed by Boulder 2020.

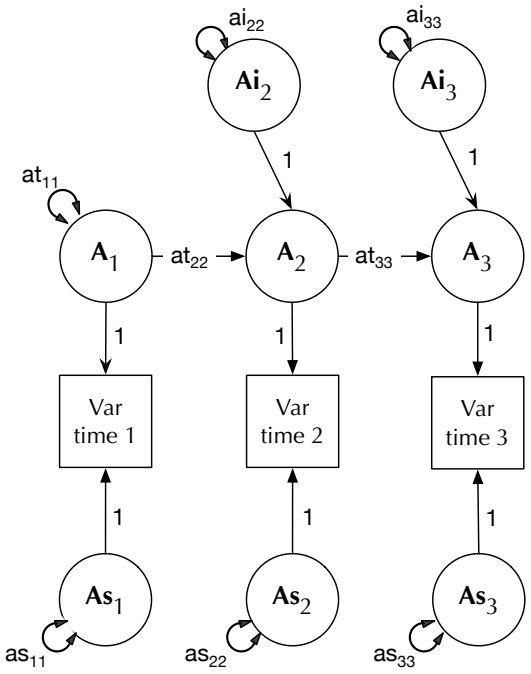
The simplex model decomposes phenotypic variance into Additive genetic, unique environmental (E) and, optionally, either common or shared-environment (C) or non-additive genetic effects (D).

In the simplex model, these influences are modeled as a combination of:

- Innovations at a given time (`ai` `ci` and `ei` matrices).
- Influences transmitted from previous time (`at`, `ct`, and `et` matrices).
- Influences specific to a single time (`as`, `cs`, `es`).

These combine to explain the causes of variance in the phenotype (see Figure).

Simplex path diagram:



Data Input Currently, the umxSimplex function accepts only raw data.

Ordinal Data In an important capability, the model transparently handles ordinal (binary or multi-level ordered factor data) inputs, and can handle mixtures of continuous, binary, and ordinal data in any combination.

Additional features The `umxSimplex` function supports varying the DZ genetic association (defaulting to .5) to allow exploring assortative mating effects, as well as varying the DZ "C" factor from 1 (the default for modeling family-level effects shared 100% by twins in a pair), to .25 to model dominance effects.

Matrices and Labels in the simplex model A good way to see which matrices are used in umxSummary is to run an example model and plot it.

The loadings specific to each time point are contained on the diagonals of matrices `as`, `cs`, and `es`. So labels relevant to modifying these are of the form "as_r1c1", "as_r2c2" etc.

All the shared matrices are in the model "top". So to see the 'as' values, you can simply execute:

`m1$top$as$values`

The transmitted loadings are in matrices `at`, `ct`, `et`.

The innovations are in the matrix `ai`, `ci`, and `ei`.

Less commonly-modified matrices are the mean matrix `expMean`. This has 1 row, and the columns are laid out for each variable for twin 1, followed by each variable for twin 2.

Thus, in a model where the means for twin 1 and twin 2 had been equated (set = to T1), you could make them independent again with this script:

`m1$top$expMean$labels[1,4:6] = c("expMean_r1c4","expMean_r1c5","expMean_r1c6")`

Value

- `mxModel()`

References

- https://www.github.com/tbates/umx

See Also

- `umxACE()` for more examples of twin modeling, `plot()`, `umxSummary()` work for IP, CP, GxE, SAT, and ACE models.

Other Twin Modeling Functions: `plot.MxModelTwinMaker()`, `power.ACE.test()`, `umxACEcov()`, `umxACEv()`, `umxACE()`, `umxCP()`, `umxDoCp()`, `umxDoC()`, `umxGxE_window()`, `umxGxEbiv()`, `umxGxE()`, `umxIP()`, `umxRotate.MxModelCP()`, `umxSexLim()`, `umx`

Examples

```
## Not run:
data(iqdat)
mzData = subset(iqdat, zygosity == "MZ")
dzData = subset(iqdat, zygosity == "DZ")
baseVars = c("IQ_age1", "IQ_age2", "IQ_age3", "IQ_age4")
m1= umxSimplex(selDVs= baseVars, dzData= dzData, mzData= mzData, sep= "_T", tryHard= "yes")

umxSummary(m1)
parameters(m1, patt = "^s")
m2 = umxModify(m1, regex = "as_r1c1", name = "no_as", comp = TRUE)
umxCompare(m1, m2)

# ============================
# = Test a 3 time-point model =
# ============================
m1 = umxSimplex(selDVs = paste0("IQ_age", 1:3),
dzData = dzData, mzData = mzData, sep = "_T", tryHard = "yes")

## End(Not run)
```

umxSummarizeTwinData *Summarize twin data*

Description

Produce a summary of wide-format twin data, showing the number of individuals, the mean and SD for each trait, and the correlation for each twin-type.

Set MZ and DZ to summarize the two-group case.

Usage

```
umxSummarizeTwinData(
  data = NULL,
  selVars = NULL,
  sep = "_T",
  zyg = "zygosity",
  MZ = NULL,
  DZ = NULL,
  MZFF = "MZFF",
  DZFF = "DZFF",
  MZMM = "MZMM",
  DZMM = "DZMM",
  DZOS = "DZOS",
  digits = 2,
  report = c("markdown", "html")
)
```

Arguments

data	The twin data.
selVars	Collection of variables to report on, e.g. c("wt", "ht").
sep	The separator string that will turn a variable name into a twin variable name, default= "_T" for wt_T1 and wt_T2.
zyg	The zygosity variable in the dataset, default = "zygosity".
MZ	Set level in zyg corresponding to MZ for two group case (defaults to using 5-group case).
DZ	Set level in zyg corresponding to DZ for two group case (defaults to using 5-group case).
MZFF	The level of zyg corresponding to MZ FF pairs: default= "MZFF".
DZFF	The level of zyg corresponding to DZ FF pairs: default= "DZFF".
MZMM	The level of zyg corresponding to MZ MM pairs: default= "MZMM".
DZMM	The level of zyg corresponding to DZ MM pairs: default= "DZMM".
DZOS	The level of zyg corresponding to DZ OS pairs: default= "DZOS".
digits	Rounding precision of the report (default 2).
report	What to return (default = 'markdown'). Use 'html' to open a web table.

Value

- formatted table, e.g. in markdown.

References

- https://github.com/tbates/umx

See Also

- umxAPA()

Other Twin Reporting Functions: umxPlotCP(), umxPlotDoC(), umxReduceACE(), umxReduceGxE(), umxReduce(), umxSummaryACEcov(), umxSummaryACEv(), umxSummaryACE(), umxSummaryCP(), umxSummaryDoC(), umxSummaryGxEbiv(), umxSummaryGxE(), umxSummaryIP(), umxSummarySexLim(), umxSummarySimplex(), umx

Examples

```
data(twinData)
umxSummarizeTwinData(twinData, sep = "", selVars = c("wt", "ht"))
MZs = c("MZMM", "MZFF"); DZs = c("DZFF","DZMM", "DZOS")
umxSummarizeTwinData(twinData, sep = "", selVars = c("wt", "ht"), MZ = MZs, DZ = DZs)
```

---

umxSummary *Shows a compact, publication-style, summary of umx models*

Description

Report the fit of a OpenMx model or specialized model class (such as ACE, CP etc.) in a compact form suitable for reporting in a journal.

See documentation for RAM models summary here: umxSummary.MxModel().

View documentation on the ACE model subclass here: umxSummaryACE().

View documentation on the ACEv model subclass here: umxSummaryACEv().

View documentation on the IP model subclass here: umxSummaryIP().

View documentation on the CP model subclass here: umxSummaryCP().

View documentation on the GxE model subclass here: umxSummaryGxE().

Usage

```
umxSummary(model, ...)
```

Arguments

model	The mxModel() whose fit will be reported
...	Other parameters to control model summary

See Also

Other Core Modeling Functions: umxAlgebra(), umxMatrix(), umxModify(), umxPath(), umxRAM(), umxRun(), umxSuperModel(), umx

---

umxSummary.MxModel *Shows a compact, publication-style, summary of a RAM model*

Description

Report the fit of a model in a compact form suitable for a journal. Alerts you when model fit is worse than accepted criterion (TLI >= .95 and RMSEA <= .06; (Hu & Bentler, 1999; Yu, 2002).

Usage

```
## S3 method for class 'MxModel'
umxSummary(
  model,
  refModels = NULL,
  std = FALSE,
  digits = 2,
  report = c("markdown", "html"),
  filter = c("ALL", "NS", "SIG"),
  SE = TRUE,
  RMSEA_CI = FALSE,
  ...,
  matrixAddresses = FALSE
)
```

Arguments

model	The mxModel() whose fit will be reported
refModels	Saturated models if needed for fit indices (see example below: If NULL will be competed on demand. If FALSE will not be computed. Only needed for raw data.
std	If TRUE, model is standardized (Default FALSE, NULL means "don't show").
digits	How many decimal places to report (Default 2)
report	If "html", then show results in browser (alternative = "markdown")
filter	whether to show significant paths (SIG) or NS paths (NS) or all paths (ALL)
SE	Whether to compute SEs... defaults to TRUE. In rare cases, you might need to turn off to avoid errors.
RMSEA_CI	Whether to compute the CI on RMSEA (Defaults to FALSE)
...	Other parameters to control model summary
matrixAddresses	Whether to show "matrix address" columns (Default = FALSE)

Details

Note: For some (multi-group) models, you will need to fall back on summary()

CIs and Identification This function uses the standard errors reported by OpenMx to produce the CIs you see in umxSummary These are used to derive confidence intervals based on the formula 95%CI = estimate +/- 1.96*SE)

Sometimes they appear NA. This often indicates a model which is not identified (see http://davidakenny.net/cm/identify.htm). This can include empirical under-identification - for instance two factors that are essentially identical in structure. use mxCheckIdentification() to check identification.

One or more paths estimated at or close to zero suggests that fixing one or two of these to zero may fix the standard error calculation, and alleviate the need to estimate likelihood-based or bootstrap CIs

If factor loadings can flip sign and provide identical fit, this creates another form of under-identification and can break confidence interval estimation. Fixing a factor loading to 1 and estimating factor variances can help here.

Value

- parameterTable returned invisibly, if estimates requested

References

- Hu, L., & Bentler, P. M. (1999). Cutoff criteria for fit indexes in covariance structure analysis: Conventional criteria versus new alternatives. *Structural Equation Modeling*, **6**, 1-55.
- Yu, C.Y. (2002). Evaluating cutoff criteria of model fit indices for latent variable models with binary and continuous outcomes. University of California, Los Angeles, Los Angeles. Retrieved from https://www.statmodel.com/download/Yudissertation.pdf

https://tbates.github.io

See Also

- umxRun()

Other Reporting functions: RMSEA.MxModel(), RMSEA.summary.mxmodel(), RMSEA(), extractAIC.MxModel(), loadings(), residuals.MxModel(), umxCI_boot(), umxCI(), umxCompare(), umxConfint(), umxExpCov(), umxExpMeans(), umxFitIndices(), umxPlotACEv(), umxRotate()

Examples

```
require(umx)
data(demoOneFactor)
manifests = names(demoOneFactor)
m1 = umxRAM("One Factor", data = demoOneFactor, type = "cov",
umxPath("G", to = manifests),
umxPath(var = manifests),
umxPath(var = "G", fixedAt = 1)
)
umxSummary(m1, std = TRUE)
```

```
# output as latex
umx_set_table_format("latex")
umxSummary(m1, std = TRUE)
umx_set_table_format("markdown")
# output as raw
umxSummary(m1, std = FALSE)

# switch to a raw data model
m1 = umxRAM("One Factor", data = demoOneFactor[1:100, ],
umxPath("G", to = manifests),
umxPath(v.m. = manifests),
umxPath(v1m0 = "G")
)
umxSummary(m1, std = TRUE, filter = "NS")
```

---

umxSummaryACE *Shows a compact, publication-style, summary of a umx Cholesky ACE model*

Description

Summarize a fitted Cholesky model returned by umxACE(). Can control digits, report comparison model fits, optionally show the Rg (genetic and environmental correlations), and show confidence intervals. the report parameter allows drawing the tables to a web browser where they may readily be copied into non-markdown programs like Word.

Usage

```
umxSummaryACE(
  model,
  digits = 2,
  file = getOption("umx_auto_plot"),
  comparison = NULL,
  std = TRUE,
  showRg = FALSE,
  CIs = TRUE,
  report = c("markdown", "html"),
  returnStd = FALSE,
  extended = FALSE,
  zero.print = ".",
  show,
  ...
)
```

Arguments

model	an mxModel() to summarize.
digits	round to how many digits (default = 2).

file	The name of the dot file to write: "name" = use the name of the model. Defaults to NA = do not create plot output.
comparison	you can run mxCompare on a comparison model (NULL).
std	Whether to standardize the output (default = TRUE).
showRg	= whether to show the genetic correlations (FALSE).
CIs	Whether to show Confidence intervals if they exist (TRUE).
report	If "html", then open an html table of the results.
returnStd	Whether to return the standardized form of the model (default = FALSE).
extended	how much to report (FALSE).
zero.print	How to show zeros (".")
show	std, raw etc. Not implemented for umxACE yet.
...	Other parameters to control model summary.

Details

See documentation for other umx models here: umxSummary().

Value

- optional mxModel()

References

- https://tbates.github.io, https://github.com/tbates/umx

See Also

- umxACE(), plot.MxModelACE(), umxModify()

Other Twin Reporting Functions: umxPlotCP(), umxPlotDoC(), umxReduceACE(), umxReduceGxE(), umxReduce(), umxSummarizeTwinData(), umxSummaryACEcov(), umxSummaryACEv(), umxSummaryCP(), umxSummaryDoC(), umxSummaryGxEbiv(), umxSummaryGxE(), umxSummaryIP(), umxSummarySexLim(), umxSummarySimplex(), umx

Examples

```
require(umx)
data(twinData)
selDVs = c("bmi1", "bmi2")
mzData <- subset(twinData, zygosity == "MZFF")
dzData <- subset(twinData, zygosity == "DZFF")
m1 = umxACE(selDVs = selDVs, dzData = dzData, mzData = mzData)
umxSummary(m1)
## Not run:
umxSummaryACE(m1, file = NA);
umxSummaryACE(m1, file = "name", std = TRUE)
stdFit = umxSummaryACE(m1, returnStd = TRUE);

## End(Not run)
```

umxSummaryACEcov *Present results of a twin ACE-model with covariates in table and graphical forms.*

Description

Summarize a Cholesky model with covariates, as returned by umxACEcov()

Usage

```
umxSummaryACEcov(
  model,
  digits = 2,
  showRg = FALSE,
  std = TRUE,
  comparison = NULL,
  CIs = TRUE,
  zero.print = ".",
  report = c("markdown", "html"),
  file = getOption("umx_auto_plot"),
  returnStd = FALSE,
  extended = FALSE,
  ...
)
```

Arguments

model	a umxACEcov() model to summarize
digits	round to how many digits (default = 2)
showRg	= whether to show the genetic correlations (FALSE)
std	= whether to show the standardized model (TRUE)
comparison	you can run mxCompare on a comparison model (NULL)
CIs	Whether to show Confidence intervals if they exist (TRUE)
zero.print	How to show zeros (".")
report	If "html", then open an html table of the results.
file	The name of the dot file to write: NA = none; "name" = use the name of the model
returnStd	Whether to return the standardized form of the model (default = FALSE)
extended	how much to report (FALSE)
...	Other parameters to control model summary

Value

- optional mxModel()

References

- https://tbates.github.io, https://github.com/tbates/umx

See Also

- umxACEcov()

Other Twin Reporting Functions: umxPlotCP(), umxPlotDoC(), umxReduceACE(), umxReduceGxE(), umxReduce(), umxSummarizeTwinData(), umxSummaryACEv(), umxSummaryACE(), umxSummaryCP(), umxSummaryDoC(), umxSummaryGxEbiv(), umxSummaryGxE(), umxSummaryIP(), umxSummarySexLim(), umxSummarySimplex(), umx

Examples

```
require(umx)
data(twinData)
selDVs = c("bmi1", "bmi2")
mzData = subset(twinData, zygosity == "MZFF")
dzData = subset(twinData, zygosity == "DZFF")
m1 = umxACE(selDVs = selDVs, dzData = dzData, mzData = mzData)
## Not run:
umxSummaryACE(m1, file = NA)
umxSummaryACE(m1, file = "name", std = TRUE)
stdFit = umxSummaryACE(m1, returnStd = TRUE)

## End(Not run)
```

---

umxSummaryACEv *Shows a compact, publication-style, summary of a variance-based Cholesky ACE model.*

Description

Summarize a fitted Cholesky model returned by umxACEv(). Can control digits, report comparison model fits, optionally show the Rg (genetic and environmental correlations), and show confidence intervals. the report parameter allows drawing the tables to a web browser where they may readily be copied into non-markdown programs like Word.

Usage

```
umxSummaryACEv(
  model,
  digits = 2,
  file = getOption("umx_auto_plot"),
  comparison = NULL,
  std = TRUE,
  showRg = FALSE,
  CIs = TRUE,
```

```
  report = c("markdown", "html"),
  returnStd = FALSE,
  extended = FALSE,
  zero.print = ".",
  show = c("std", "raw"),
  ...
)
```

Arguments

model	an mxModel() to summarize
digits	round to how many digits (default = 2)
file	The name of the dot file to write: "name" = use the name of the model. Defaults to NA = no plot.
comparison	you can run mxCompare on a comparison model (NULL)
std	Whether to standardize the output (default = TRUE)
showRg	= whether to show the genetic correlations (FALSE)
CIs	Whether to show Confidence intervals if they exist (TRUE)
report	If "html", then open an html table of the results
returnStd	Whether to return the standardized form of the model (default = FALSE)
extended	how much to report (FALSE)
zero.print	How to show zeros (".")
show	Here to support being called from generic xmu_safe_run_summary. User should ignore: can be c("std", "raw")
...	Other parameters to control model summary

Details

See documentation for other umx models here: umxSummary().

Value

- optional mxModel()

References

- https://tbates.github.io, https://github.com/tbates/umx

See Also

- umxACEv()

Other Twin Reporting Functions: umxPlotCP(), umxPlotDoC(), umxReduceACE(), umxReduceGxE(), umxReduce(), umxSummarizeTwinData(), umxSummaryACEcov(), umxSummaryACE(), umxSummaryCP(), umxSummaryDoC(), umxSummaryGxEbiv(), umxSummaryGxE(), umxSummaryIP(), umxSummarySexLim(), umxSummarySimplex(), umx

Examples

```
require(umx)
data(twinData)
mzData = subset(twinData, zygosity == "MZFF")
dzData = subset(twinData, zygosity == "DZFF")
m1 = umxACEv(selDVs = "bmi", sep = "", dzData = dzData, mzData = mzData)
umxSummary(m1, std = FALSE)
## Not run:
umxSummary(m1, file = NA);
umxSummary(m1, file = "name", std = TRUE)
stdFit = umxSummary(m1, returnStd = TRUE)

## End(Not run)
```

---

umxSummaryCP *Present the results of a Common-pathway twin model in table and graphical form*

Description

Summarizes a Common-Pathway model, as returned by umxCP()

Usage

```
umxSummaryCP(
  model,
  digits = 2,
  std = TRUE,
  CIs = FALSE,
  showRg = FALSE,
  comparison = NULL,
  report = c("markdown", "html"),
  file = getOption("umx_auto_plot"),
  returnStd = FALSE,
  ...
)
```

Arguments

model	A fitted umxCP() model to summarize
digits	Round to how many digits (default = 2)
std	Whether to show the standardized model (TRUE) (ignored: used extended = TRUE to get unstandardized)
CIs	Confidence intervals (default FALSE)
showRg	Whether to show the genetic correlations (default FALSE)
comparison	Run mxCompare on a comparison model (default NULL)

report	Print tables to the console (as 'markdown'), or open in browser ('html')
file	The name of the dot file to write: NA = none; "name" = use the name of the model
returnStd	Whether to return the standardized form of the model (default = FALSE)
...	Optional additional parameters

Value

- optional mxModel()

References

- https://www.github.com/tbates/umx, https://tbates.github.io

See Also

- umxCP(), plot(), umxSummary() work for IP, CP, GxE, SAT, and ACE models.

Other Twin Reporting Functions: umxPlotCP(), umxPlotDoC(), umxReduceACE(), umxReduceGxE(), umxReduce(), umxSummarizeTwinData(), umxSummaryACEcov(), umxSummaryACEv(), umxSummaryACE(), umxSummaryDoC(), umxSummaryGxEbiv(), umxSummaryGxE(), umxSummaryIP(), umxSummarySexLim(), umxSummarySimplex(), umx

Examples

```
## Not run:
require(umx)
umx_set_optimizer("SLSQP")
data(twinData)
twinData$wt1 = twinData$wt1/10
twinData$wt2 = twinData$wt2/10
selDVs = c("ht", "wt")
mzData = subset(twinData, zygosity == "MZFF")
dzData = subset(twinData, zygosity == "DZFF")
umx_set_auto_plot(FALSE) # turn off autoplotting for CRAN
m1 = umxCP(selDVs = selDVs, dzData = dzData, mzData = mzData, sep = "", optimizer = "SLSQP")
umxSummaryCP(m1, file = NA) # Suppress plot creation with file
umxSummary(m1, file = NA)   # Generic summary is the same
stdFit = umxSummaryCP(m1, digits = 2, std = TRUE, file = NA, returnStd = TRUE);
umxSummary(m1, std = FALSE, showRg = TRUE, file = NA);
umxSummary(m1, std = FALSE, file = NA)

# =================
# = Print example =
# =================
umxSummary(m1, file = "Figure 3", std = TRUE)

# =================
# = Confint example =
# =================
m1 = umxConfint(m1, "smart", run = FALSE);
```

```
m1 = umxConfint(m1, "smart", run = TRUE);
umxSummary(m1, CIs = TRUE, file = NA);

## End(Not run)
```

---

umxSummaryDoC *Shows a compact, publication-style, summary of a umx Direction of Causation model*

Description

Summarize a fitted model returned by umxDoC(). Can control digits, report comparison model fits, optionally show the Rg (genetic and environmental correlations), and show confidence intervals. the report parameter allows drawing the tables to a web browser where they may readily be copied into non-markdown programs like Word.

Usage

```
umxSummaryDoC(
  model,
  digits = 2,
  comparison = NULL,
  std = TRUE,
  showRg = FALSE,
  CIs = TRUE,
  report = c("markdown", "html"),
  file = getOption("umx_auto_plot"),
  returnStd = FALSE,
  zero.print = ".",
  ...
)
```

Arguments

model	a fitted umxDoC() model to summarize.
digits	round to how many digits (default = 2).
comparison	Run mxCompare on a comparison model (default NULL)
std	Whether to standardize the output (default = TRUE).
showRg	= whether to show the genetic correlations (FALSE).
CIs	Whether to show Confidence intervals if they exist (TRUE).
report	Print tables to the console (as 'markdown'), or open in browser ('html')
file	The name of the dot file to write: "name" = use the name of the model. Defaults to NA = do not create plot output.
returnStd	Whether to return the standardized form of the model (default = FALSE).
zero.print	How to show zeros (".")
...	Other parameters to control model summary.

Details

See documentation for other umx models here: umxSummary().

Value

- optional mxModel()

See Also

- umxDoC(), plot.MxModelDoC(), umxModify(), umxCP(), plot(), umxSummary() work for IP, CP, GxE, SAT, and ACE models.

Other Twin Reporting Functions: umxPlotCP(), umxPlotDoC(), umxReduceACE(), umxReduceGxE(), umxReduce(), umxSummarizeTwinData(), umxSummaryACEcov(), umxSummaryACEv(), umxSummaryACE(), umxSummaryCP(), umxSummaryGxEbiv(), umxSummaryGxE(), umxSummaryIP(), umxSummarySexLim(), umxSummarySimplex(), umx

Examples

```
## Not run:
# ================
# = 1. Load Data =
# ================
umx_set_auto_plot(FALSE) # turn off autoplotting for CRAN
data(docData)
mzData = subset(docData, zygosity %in% c("MZFF", "MZMM"))
dzData = subset(docData, zygosity %in% c("DZFF", "DZMM"))

# =======================================
# = 2. Define manifests for var 1 and 2 =
# =======================================
var1 = paste0("varA", 1:3)
var2 = paste0("varB", 1:3)

# ========================================================
# = 2. Make the non-causal (Cholesky) and causal models =
# ========================================================
Chol= umxDoC(var1= var1, var2= var2, mzData= mzData, dzData= dzData, causal= FALSE)
DoC = umxDoC(var1= var1, var2= var2, mzData= mzData, dzData= dzData, causal= TRUE)

# ================================================
# = Make the directional models by modifying DoC =
# ================================================
A2B = umxModify(DoC, "a2b", free = TRUE, name = "A2B")
A2B = umxModify(DoC, "a2b", free = TRUE, name = "A2B", comp=TRUE)
B2A = umxModify(DoC, "b2a", free = TRUE, name = "B2A", comp=TRUE)
umxCompare(B2A, A2B)


## End(Not run)
```

umxSummaryGxE *Summarize a GxE model*

Description

Summarize a genetic moderation model, as returned by umxGxE(). Prints graphs of A, C, and E, standardized and raw.

Usage

```
umxSummaryGxE(
  model = NULL,
  digits = 2,
  xlab = NA,
  location = "topleft",
  separateGraphs = FALSE,
  file = getOption("umx_auto_plot"),
  returnStd = NULL,
  std = NULL,
  reduce = FALSE,
  CIs = NULL,
  report = c("markdown", "html"),
  show = NULL,
  ...
)
```

Arguments

model	A fitted umxGxE() model to summarize
digits	round to how many digits (default = 2)
xlab	label for the x-axis of plot
location	default = "topleft"
separateGraphs	If TRUE, both std and raw plots in one figure (default FALSE)
file	The name of the dot file to write: NA = none; "name" = use the name of the model
returnStd	Whether to return the standardized form of the model (default = FALSE)
std	Whether to show the standardized model (not implemented! TRUE)
reduce	Whether run and tabulate a complete model reduction...(Defaults to FALSE)
CIs	Confidence intervals (FALSE)
report	"markdown" or "html" = open a browser for copyable tables
show	not doing anything yet (required for all summary functions)
...	Optional additional parameters

Details

Note: see also umxReduce() which knows how to reduce a GxE model.

Value

- optional mxModel()

References

- https://github.com/tbates/umx, https://tbates.github.io

See Also

- umxGxE(), umxReduce(), plot(), [umxSummary)] all work for IP, CP, GxE, and ACE models.

[umxSummary)]: R:umxSummary)

Other Twin Reporting Functions: umxPlotCP(), umxPlotDoC(), umxReduceACE(), umxReduceGxE(), umxReduce(), umxSummarizeTwinData(), umxSummaryACEcov(), umxSummaryACEv(), umxSummaryACE(), umxSummaryCP(), umxSummaryDoC(), umxSummaryGxEbiv(), umxSummaryIP(), umxSummarySexLim(), umxSummarySimplex(), umx

Examples

```
# The total sample has been subdivided into a young cohort,
# aged 18-30 years, and an older cohort aged 31 and above.
# Cohort 1 Zygosity is coded as follows 1 == MZ females 2 == MZ males
# 3 == DZ females 4 == DZ males 5 == DZ opposite sex pairs
require(umx)
data(twinData)
twinData$age1 = twinData$age2 = twinData$age
selDVs  = c("bmi1", "bmi2")
selDefs = c("age1", "age2")
selVars = c(selDVs, selDefs)
mzData  = subset(twinData, zygosity == "MZFF", selVars)
dzData  = subset(twinData, zygosity == "DZMM", selVars)
# Exclude cases with missing Def
mzData = mzData[!is.na(mzData[selDefs[1]]) & !is.na(mzData[selDefs[2]]),]
dzData = dzData[!is.na(dzData[selDefs[1]]) & !is.na(dzData[selDefs[2]]),]
## Not run:
m1 = umxGxE(selDVs = selDVs, selDefs = selDefs, dzData = dzData, mzData = mzData)
# Plot Moderation
umxSummaryGxE(m1)
umxSummaryGxE(m1, location = "topright")
umxSummaryGxE(m1, separateGraphs = FALSE)

## End(Not run)
```

umxSummaryGxEbiv *Summarize a bivariate GxE twin model*

Description

umxSummaryGxEbiv summarizes a bivariate moderation model, as returned by umxGxEbiv().

Usage

```
umxSummaryGxEbiv(
  model = NULL,
  digits = 2,
  xlab = NA,
  location = "topleft",
  separateGraphs = FALSE,
  file = getOption("umx_auto_plot"),
  comparison = NULL,
  std = NULL,
  reduce = FALSE,
  CIs = NULL,
  report = c("markdown", "html"),
  returnStd = NULL,
  ...
)
```

Arguments

model	A fitted umxGxEbiv() model to summarize
digits	round to how many digits (default = 2)
xlab	label for the x-axis of plot
location	default = "topleft"
separateGraphs	Std and raw plots in separate graphs? (default = FALSE)
file	The name of the dot file to write: NA = none; "name" = use the name of the model
comparison	mxCompare model with this model if offered up (default = NULL).
std	Whether to show the standardized model (not implemented! TRUE)
reduce	Whether to run and tabulate a complete model reduction...(Defaults to FALSE)
CIs	Confidence intervals (FALSE)
report	markdown or html (html opens in browser)
returnStd	Whether to return the standardized form of the model (default = FALSE)
...	Optional additional parameters

Value

- optional mxModel()

References

- https://github.com/tbates/umx, https://tbates.github.io

See Also

- umxGxEbiv(), plot(), umxSummary() work for IP, CP, GxE, and ACE models.

Other Twin Reporting Functions: umxPlotCP(), umxPlotDoC(), umxReduceACE(), umxReduceGxE(), umxReduce(), umxSummarizeTwinData(), umxSummaryACEcov(), umxSummaryACEv(), umxSummaryACE(), umxSummaryCP(), umxSummaryDoC(), umxSummaryGxE(), umxSummaryIP(), umxSummarySexLim(), umxSummarySimplex(), umx

Examples

```
data(twinData)
df = umx_scale_wide_twin_data(twinData, varsToScale = c("ht", "wt"), sep = "")
mzData  = subset(df, zygosity %in% c("MZFF", "MZMM"))
dzData  = subset(df, zygosity %in% c("DZFF", "DZMM", "DZOS"))

## Not run:
m1 = umxGxEbiv(selDVs = "wt", selDefs = "ht",
dzData = dzData, mzData = mzData, sep = "", dropMissingDef = TRUE)
# Plot Moderation
umxSummary(m1)
umxSummary(m1, location = "topright")
umxSummary(m1, separateGraphs = FALSE)

## End(Not run)
```

---

umxSummaryIP — *Present the results of an independent-pathway twin model in table and graphical form*

Description

Summarize a Independent Pathway model, as returned by umxIP()

Usage

```
umxSummaryIP(
  model,
  digits = 2,
  file = getOption("umx_auto_plot"),
  std = TRUE,
  showRg = FALSE,
```

```
  comparison = NULL,
  CIs = FALSE,
  returnStd = FALSE,
  report = c("markdown", "html"),
  ...
)
```

Arguments

model	A fitted umxIP() model to summarize
digits	round to how many digits (default = 2)
file	The name of the dot file to write: NA = none; "name" = use the name of the model
std	= Whether to show the standardized model (TRUE)
showRg	= whether to show the genetic correlations (FALSE)
comparison	Whether to run mxCompare on a comparison model (NULL)
CIs	Confidence intervals (F)
returnStd	Whether to return the standardized form of the model (default = FALSE)
report	how to display the results ("html" will open in browser as table)
...	Optional additional parameters

Value

- optional mxModel()

References

- https://github.com/tbates/umx, https://tbates.github.io

See Also

- umxIP(), plot(), umxSummary() work for IP, CP, GxE, SAT, and ACE models.

Other Twin Reporting Functions: umxPlotCP(), umxPlotDoC(), umxReduceACE(), umxReduceGxE(), umxReduce(), umxSummarizeTwinData(), umxSummaryACEcov(), umxSummaryACEv(), umxSummaryACE(), umxSummaryCP(), umxSummaryDoC(), umxSummaryGxEbiv(), umxSummaryGxE(), umxSummarySexLim(), umxSummarySimplex(), umx

Examples

```
require(umx)
data(GFF) # family function and well-being data
mzData <- subset(GFF, zyg_2grp == "MZ")
dzData <- subset(GFF, zyg_2grp == "DZ")
selDVs = c("hap", "sat", "AD") # These will be expanded into "hap_T1" "hap_T2" etc.
m1 = umxIP(selDVs = selDVs, sep = "_T", dzData = dzData, mzData = mzData)
umxSummaryIP(m1)
plot(m1)
```

```
## Not run:
umxSummaryIP(m1, digits = 2, file = "Figure3", showRg = FALSE, CIs = TRUE);

## End(Not run)
```

`umxSummarySexLim` *Shows a compact, publication-style, summary of a umx Sex Limitation model*

Description

Summarize a fitted Cholesky model returned by `umxSexLim()`. Can control digits, report comparison model fits, optionally show the Rg (genetic and environmental correlations), and show confidence intervals. The report parameter allows drawing the tables to a web browser where they may readily be copied into non-markdown programs like Word.

Usage

```
umxSummarySexLim(
  model,
  digits = 2,
  file = getOption("umx_auto_plot"),
  comparison = NULL,
  std = TRUE,
  showRg = FALSE,
  CIs = TRUE,
  report = c("markdown", "html"),
  extended = FALSE,
  zero.print = ".",
  show = c("std", "raw"),
  returnStd = FALSE,
  ...
)
```

Arguments

`model`	a `umxSexLim()` model to summarize
`digits`	round to how many digits (default = 2)
`file`	The name of the dot file to write: "name" = use the name of the model. Defaults to NA = do not create plot output
`comparison`	you can run mxCompare on a comparison model (NULL)
`std`	Whether to standardize the output (default = TRUE)
`showRg`	= whether to show the genetic correlations (FALSE)
`CIs`	Whether to show Confidence intervals if they exist (T)
`report`	If "html", then open an html table of the results

extended	how much to report (FALSE)
zero.print	How to show zeros (".")
show	Here to support being called from generic xmu_safe_run_summary. User should ignore: can be c("std", "raw")
returnStd	Whether to return the standardized form of the model (default = FALSE)
...	Other parameters to control model summary

Details

See documentation for summary functions for other types of umx model here: umxSummary().

Value

- optional mxModel()

References

- https://tbates.github.io, https://github.com/tbates/umx

See Also

- umxSexLim(), umxPlotSexLim()

Other Twin Reporting Functions: umxPlotCP(), umxPlotDoC(), umxReduceACE(), umxReduceGxE(), umxReduce(), umxSummarizeTwinData(), umxSummaryACEcov(), umxSummaryACEv(), umxSummaryACE(), umxSummaryCP(), umxSummaryDoC(), umxSummaryGxEbiv(), umxSummaryGxE(), umxSummaryIP(), umxSummarySimplex(), umx

Examples

```
## Not run:
# ======================================================
# = Beta: Should be good to use for Boulder/March 2020 =
# ======================================================

# ============================================
# = Run Qualitative Sex Differences ACE model =
# ============================================

# =========================
# = Load and Process Data =
# =========================
require(umx)
umx_set_optimizer("SLSQP")
data("us_skinfold_data")
# rescale vars
us_skinfold_data[, c('bic_T1', 'bic_T2')] = us_skinfold_data[, c('bic_T1', 'bic_T2')]/3.4
us_skinfold_data[, c('tri_T1', 'tri_T2')] = us_skinfold_data[, c('tri_T1', 'tri_T2')]/3
us_skinfold_data[, c('caf_T1', 'caf_T2')] = us_skinfold_data[, c('caf_T1', 'caf_T2')]/3
us_skinfold_data[, c('ssc_T1', 'ssc_T2')] = us_skinfold_data[, c('ssc_T1', 'ssc_T2')]/5
us_skinfold_data[, c('sil_T1', 'sil_T2')] = us_skinfold_data[, c('sil_T1', 'sil_T2')]/5
```

```
# Variables for Analysis
selDVs = c('ssc','sil','caf','tri','bic')
# Data for each of the 5 twin-type groups
mzmData = subset(us_skinfold_data, zyg == 1)
mzfData = subset(us_skinfold_data, zyg == 2)
dzmData = subset(us_skinfold_data, zyg == 3)
dzfData = subset(us_skinfold_data, zyg == 4)
dzoData = subset(us_skinfold_data, zyg == 5)

# ======================
# = Bivariate example =
# ======================

selDVs = c('tri','bic')
m1 = umxSexLim(selDVs = selDVs, sep = "_T", A_or_C = "A", tryHard = "yes",
mzmData = mzmData, dzmData = dzmData,
mzfData = mzfData, dzfData = dzfData,
dzoData = dzoData
)
umxSummary(m1, file = NA);

# ===============
# = Switch to C =
# ===============
m1 = umxSexLim(selDVs = selDVs, sep = "_T", A_or_C = "C", tryHard = "yes",
mzmData = mzmData, dzmData = dzmData,
mzfData = mzfData, dzfData = dzfData,
dzoData = dzoData
)

## End(Not run)
```

---

umxSummarySimplex *Shows a compact, publication-style, summary of a Simplex model.*

Description

Summarize a fitted Simplex model returned by umxSimplex(). Can control digits, report comparison model fits, optionally show the Rg (genetic and environmental correlations), and show confidence intervals. the report parameter allows drawing the tables to a web browser where they may readily be copied into non-markdown programs like Word.

Usage

```
umxSummarySimplex(
  model,
  digits = 2,
  file = getOption("umx_auto_plot"),
  comparison = NULL,
```

```
  std = TRUE,
  showRg = FALSE,
  CIs = TRUE,
  report = c("markdown", "html"),
  returnStd = FALSE,
  extended = FALSE,
  zero.print = ".",
  show = c("std", "raw"),
  ...
)
```

Arguments

model	an mxModel() to summarize
digits	round to how many digits (default = 2)
file	The name of the dot file to write: "name" = use the name of the model. Defaults to NA = no plot.
comparison	you can run mxCompare on a comparison model (default = NULL)
std	Whether to standardize the output (default = TRUE)
showRg	(T/F) Whether to show the genetic correlations (default = FALSE)
CIs	Whether to show Confidence intervals if they exist (default = TRUE)
report	If "html", then open an html table of the results (default = 'markdown')
returnStd	Whether to return the standardized form of the model (default = FALSE)
extended	how much to report (default = FALSE)
zero.print	How to show zeros (default = ".")
show	Here to support being called from generic xmu_safe_run_summary. User should ignore: can be c("std", "raw")
...	Other parameters to control model summary

Details

See documentation for other umx models here: umxSummary().

Value

- optional mxModel()

References

- https://tbates.github.io, https://github.com/tbates/umx

See Also

- umxSimplex()

Other Twin Reporting Functions: umxPlotCP(), umxPlotDoC(), umxReduceACE(), umxReduceGxE(), umxReduce(), umxSummarizeTwinData(), umxSummaryACEcov(), umxSummaryACEv(), umxSummaryACE(), umxSummaryCP(), umxSummaryDoC(), umxSummaryGxEbiv(), umxSummaryGxE(), umxSummaryIP(), umxSummarySexLim(), umx

Examples

```
## Not run:
# 4 time model
# Select Data
data(iqdat)
mzData <- subset(iqdat, zygosity == "MZ")
dzData <- subset(iqdat, zygosity == "DZ")
vars = c("IQ_age1", "IQ_age2", "IQ_age3", "IQ_age4")
m1= umxSimplex(selDVs= vars, sep= "_T", dzData= dzData, mzData= mzData, tryHard= "mxTryHard")
umxSummary(m1, file = NA);

## End(Not run)
```

---

umxSuperModel *Make a multi-group model*

Description

umxSuperModel takes 1 or more models and wraps them in a supermodel with a mxFitFunctionMultigroup() fit function that minimizes the sum of the fits of the sub-models.

Usage

```
umxSuperModel(
  name = "top",
  ...,
  autoRun = getOption("umx_auto_run"),
  tryHard = c("no", "yes", "ordinal", "search"),
  std = FALSE
)
```

Arguments

name	The name for the container model (default = 'top')
...	Models forming the multiple groups contained in the supermodel.
autoRun	Whether to run the model (default), or just to create it and return without running.
tryHard	Default ('no') uses normal mxRun. "yes" uses mxTryHard. Other options: "ordinal", "search"
std	Show standardized parameters, raw (default), or just the fit indices (null)

Value

- mxModel()

References

- https://github.com/tbates/umx, https://tbates.github.io

See Also

- mxFitFunctionMultigroup(), umxRAM()

Other Core Modeling Functions: umxAlgebra(), umxMatrix(), umxModify(), umxPath(), umxRAM(), umxRun(), umxSummary(), umx

Examples

```
## Not run:
library(umx)
# Create two sets of data in which X & Y correlate ~ .4 in both datasets.
manifests = c("x","y")
tmp = umx_make_TwinData(nMZpairs = 100, nDZpairs = 150,
AA = 0, CC = .4, EE = .6, varNames = manifests)

# Group 1
grp1   = tmp[tmp$zygosity == "MZ", manifests]
g1Data = mxData(cov(grp1), type = "cov", numObs = nrow(grp1), means=umx_means(grp1))

# Group 2
grp2   = tmp[tmp$zygosity == "DZ", manifests]
g2Data = mxData(cov(grp2), type = "cov", numObs = nrow(grp2), means=umx_means(grp2))


# Model 1 (could add auto=FALSE if you don't want to run this as it is being built)
m1 = umxRAM("m1", data = g1Data,
umxPath("x", to = "y", labels = "beta"),
umxPath(var = manifests, labels = c("Var_x", "Resid_y_grp1")),
umxPath(means = manifests, labels = c("Mean_x", "Mean_y"))
)

# Model 2
m2 = umxRAM("m2", data = g2Data,
umxPath("x", to = "y", labels = "beta"),
umxPath(var = manifests, labels=c("Var_x", "Resid_y_grp2")),
umxPath(means = manifests, labels=c("Mean_x", "Mean_y"))
)

# Place m1 and m2 into a supermodel, and autoRun it
# NOTE: umxSummary is only semi-smart/certain enough to compute saturated models etc
# and report multiple groups correctly.

m3 = umxSuperModel('top', m1, m2)

umxSummary(m3, std= TRUE)

# |name         | Std.Estimate| Std.SE|CI                |
# |:------------|------------:|------:|:-----------------|
# |beta         |         0.51|   0.05|0.51 [0.41, 0.61] |
# |Var_x        |         1.00|   0.00|1 [1, 1]          |
# |Resid_y_grp1 |         0.74|   0.05|0.74 [0.64, 0.84] |
# |beta         |         0.50|   0.05|0.5 [0.41, 0.6]   |
```

```
# |Var_x        |         1.00|   0.00|1 [1, 1]          |
# |Resid_y_grp2 |         0.75|   0.05|0.75 [0.65, 0.84] |

summary(m3)


## End(Not run)
```

---

umxThresholdMatrix *Create the threshold matrix needed for modeling ordinal data.*

Description

High-level helper for ordinal modeling. Creates, labels, and sets smart-starts for this complex matrix. Big time saver!

Usage

```
umxThresholdMatrix(
  df,
  selDVs = NULL,
  sep = NULL,
  method = c("Mehta", "allFree"),
  threshMatName = "threshMat",
  l_u_bound = c(NA, NA),
  droplevels = FALSE,
  verbose = FALSE
)
```

Arguments

df	The data being modeled (to allow access to the factor levels and quantiles within these for each variable)
selDVs	The variable names. Note for twin data, just the base names, which sep will be used to fill out.
sep	(e.g. "_T") Required for wide (twin) data. It is used to break the base names our from their numeric suffixes.
method	How to implement the thresholds: Mehta, (1 free thresh for binary, first two fixed for ordinal) or "allFree"
threshMatName	name of the matrix which is returned. Defaults to "threshMat" - best not to change it.
l_u_bound	c(NA, NA) by default, you can use this to bound the first (base) threshold.
droplevels	Whether to drop levels with no observed data (defaults to FALSE)
verbose	How much to say about what was done. (defaults to FALSE)

Details

We often need to model ordinal data: sex, low-med-hi, depressed/normal, etc., A useful conceptual strategy to handle these data is to build a standard model for normally-varying data and then to threshold this normal distribution to generate the observed data. Thus an observation of "depressed" is modeled as a high score on the latent normally distributed trait, with thresholds set so that only scores above this threshold (1-minus the number of categories) reach the criteria for the diagnosis.

Making this work can require fixing the first 2 thresholds of ordinal data, or fixing both the mean and variance of a latent variable driving binary data, in order to estimate its one-free parameter: where to place the single threshold separating low from high cases.

The function returns a 3-item list consisting of:

1. A thresholdsAlgebra (named `threshMatName`)
2. A matrix of deviations for the thresholds (`deviations_for_thresh`)
3. A lower matrix of ones (`lowerOnes_for_thresh`)

Twin Data

With twin data, make sure to provide the **full names** for twin data... this is not standard I know...

For twins (the function currently handles only pairs), the thresholds are equated for both twins using labels:

$labels

```
obese_T1          obese_T2
```

dev_1 "obese_dev1" "obese_dev1"

Value

- list of thresholds matrix, deviations, lowerOnes

References

- https://tbates.github.io, https://github.com/tbates/umx

See Also

Other Advanced Model Building Functions: `umxJiggle()`, `umxLabel()`, `umxValues()`, `umx`

Examples

```
# ============================
# = Simple non-twin examples =
# ============================

# data: 1 2-level ordered factor
x = data.frame(ordered(rbinom(100,1,.5))); names(x) = c("x")

tmp = umxThresholdMatrix(x, selDVs = "x")
```

```
# The lower ones matrix (all fixed)
tmp[[1]]$values
tmp[[1]]$free

# The deviations matrix
tmp[[2]]$values
tmp[[2]]$labels # note: for twins, labels will be equated across twins

# The algebra that adds the deviations to create thresholds:
tmp[[3]]$formula

# Example of a warning to not omit the variable names
# tmp = umxThresholdMatrix(x)
# Just a polite message, but for coding safety, I recommend calling
# umxThresholdMatrix with the names of the variables in the model.
#   Next time, please include selDVs (AND you MUST include sep if this is a twin model!!)

# One ordered factor with 5-levels
x = cut(rnorm(100), breaks = c(-Inf,.2,.5, .7, Inf)); levels(x) = 1:5
x = data.frame(ordered(x)); names(x) <- c("x")
tmp = umxThresholdMatrix(x, selDVs = "x")
tmp[[2]]$name
tmp[[2]]$free # last one is free.. (method = Mehta)

tmp = umxThresholdMatrix(x, selDVs = "x", l_u_bound= c(-1,1))
tmp[[2]]$lbound # bounds applied to base threshold

# ================================
# = Binary example with twin data =
# ================================
# ================================================================
# = Create a series of binary and ordinal columns to work with =
# ================================================================
data(twinData)

# Make "obese" variable with ~20% subjects categorised as obese
obesityLevels   = c('normal', 'obese')
cutPoints       = quantile(twinData[, "bmi1"], probs = .2, na.rm = TRUE)
twinData$obese1 = cut(twinData$bmi1, breaks = c(-Inf, cutPoints, Inf), labels = obesityLevels)
twinData$obese2 = cut(twinData$bmi2, breaks = c(-Inf, cutPoints, Inf), labels = obesityLevels)
# Step 2: Make the ordinal variables into umxFactors (ordered, with the levels found in the data)
selVars = c("obese1", "obese2")
twinData[, selVars] = umxFactor(twinData[, selVars])

# Example 1
# use verbose = TRUE to see informative messages
tmp = umxThresholdMatrix(twinData, selDVs = selVars, sep = "", verbose = TRUE)


# ====================================
# = Ordinal (n categories > 2) example =
# ====================================
# Repeat for three-level weight variable
```

```
obesityLevels = c('normal', 'overweight', 'obese')
cutPoints = quantile(twinData[, "bmi1"], probs = c(.4, .7), na.rm = TRUE)
twinData$obeseTri1 = cut(twinData$bmi1, breaks = c(-Inf, cutPoints, Inf), labels = obesityLevels)
twinData$obeseTri2 = cut(twinData$bmi2, breaks = c(-Inf, cutPoints, Inf), labels = obesityLevels)
selDVs = "obeseTri"; selVars = tvars(selDVs, sep = "", suffixes = 1:2)
twinData[, selVars] = umxFactor(twinData[, selVars])
tmp = umxThresholdMatrix(twinData, selDVs = selVars, sep = "", verbose = TRUE)


# ========================================================
# = Mix of all three kinds example (and a 4-level trait) =
# ========================================================
obesityLevels = c('underWeight', 'normal', 'overweight', 'obese')
cutPoints = quantile(twinData[, "bmi1"], probs = c(.25, .4, .7), na.rm = TRUE)
twinData$obeseQuad1 = cut(twinData$bmi1, breaks = c(-Inf, cutPoints, Inf), labels = obesityLevels)
twinData$obeseQuad2 = cut(twinData$bmi2, breaks = c(-Inf, cutPoints, Inf), labels = obesityLevels)
selVars = c("obeseQuad1", "obeseQuad2")
twinData[, selVars] = mxFactor(twinData[, selVars], levels = obesityLevels)

selDVs =c("bmi", "obese", "obeseTri", "obeseQuad")
tmp = umxThresholdMatrix(twinData, selDVs = tvars(selDVs, sep= ""), sep = "", verbose = TRUE)
# The lower ones matrix (all fixed)
tmp[[1]]$values
# The deviations matrix
tmp[[2]]$values
tmp[[2]]$labels # note labels are equated across twins
# Check to be sure twin-1 column labels same as twin-2
tmp[[2]]$labels[,2]==tmp[[2]]$labels[,4]

# The algebra that assembles these into thresholds:
tmp[[3]]$formula
# ================================
# = Example with method = allFree =
# ================================

tmp = umxThresholdMatrix(twinData, selDVs = tvars(selDVs, sep= ""), sep = "", method = "allFree")
all(tmp[[2]]$free)
```

---

umxTwinMaker *Make a twin model from the model describing just one person*

Description

xmu_path2twin takes a collection of paths describing the model for 1 person and returns a completed twin model. This consists of a umxSuperModel() containing MZ and DZ umxRAM() models.

Pass into umxTwinMaker:

1. A list of paths making up the twin 1 model

2. In t1_t2links, a vector describing the component relationships connecting twin 1 to twin 2 (The default here is 1 and .5 for the a, and, for c and e are 1 and 0 in both groups, respectively.

Details

Some rules. All labels are expanded with a twin suffix: so "var1" -> "var1_T1" etc. so you provide the person-model using just the base name (and tell umxTwinMaker how to expand it by providing a separator string).

Rule 2: The latent a, c, and e latent variables must be labelled to match the base name given in t1_t2links. To avoid clashes, variables must not match the numbered variables in t1_t2links - by default names like "a1" are reserved for ace.

Usage

```
umxTwinMaker(
  name = "m1",
  paths,
  t1_t2links = list(a = c(1, 0.5), c = c(1, 1), e = c(0, 0)),
  mzData = NULL,
  dzData = NULL,
  sep = "_T"
)
```

Arguments

name	The name for the resulting umxSuperModel() (Default "m1")
paths	A vector of umxPath()s describing one person
t1_t2links	base name (and values) of paths that covary between T1 and T2. Default: c('a'=c(1,.5), 'c'=c(1,1), 'e'=c(0,0))
mzData	Data for MZ twins
dzData	Data for DZ twins
sep	The separator used to create twin 1 and 2 names (Default "_T")

Value

- umxSuperModel()

References

- tutorials, tutorials

See Also

- umxRAM(), umxSuperModel()

Other xmu internal not for end user: umxModel(), umxRenameMatrix(), umx_APA_pval(), umx_fun_mean_sd(), umx_get_bracket_addresses(), umx_make(), umx_standardize(), umx_string_to_algebra(), umx, xmuHasSquareBrackets(), xmuLabel_MATRIX_Model(), xmuLabel_Matrix(), xmuLabel_RAM_Model(), xmuMI(), xmuMakeDeviationThresholdsMatrices(), xmuMakeOneHeadedPathsFromPathList(),

xmuMakeTwoHeadedPathsFromPathList(), xmuMaxLevels(), xmuMinLevels(), xmuPropagateLabels(), xmuRAM2Ordinal(), xmuTwinSuper_Continuous(), xmuTwinUpgradeMeansToCovariateModel(), xmu_CI_merge(), xmu_CI_stash(), xmu_DF_to_mxData_TypeCov(), xmu_PadAndPruneForDefVars(), xmu_cell_is_on(), xmu_check_levels_identical(), xmu_check_needs_means(), xmu_check_variance(), xmu_clean_label(), xmu_data_missing(), xmu_data_swap_a_block(), xmu_describe_data_WLS(), xmu_dot_make_paths(), xmu_dot_make_residuals(), xmu_dot_maker(), xmu_dot_move_ranks(), xmu_dot_rank_str(), xmu_extract_column(), xmu_get_CI(), xmu_lavaan_process_group(), xmu_make_TwinSuperModel(), xmu_make_bin_cont_pair_data(), xmu_make_mxData(), xmu_match.arg(), xmu_name_from_lavaan_str(), xmu_path2twin(), xmu_path_regex(), xmu_safe_run_summary(), xmu_set_sep_from_suffix(), xmu_show_fit_or_comparison(), xmu_simplex_corner(), xmu_standardize_ACEcov(), xmu_standardize_ACEv(), xmu_standardize_ACE(), xmu_standardize_CP(), xmu_standardize_IP(), xmu_standardize_RAM(), xmu_standardize_SexLim(), xmu_standardize_Simplex(), xmu_start_value_list(), xmu_starts(), xmu_twin_add_WeightMatrices(), xmu_twin_check(), xmu_twin_get_var_names(), xmu_twin_upgrade_selDvs2SelVars()

Examples

```
## Not run:
# We'll make some ACE models, but first, let's clean up the twinData
# set for analysis
# 1. Add a separator to the twin variable names (with sep = "_T")
# 2. Scale the data so it's easier for the optimizer.
data(twinData)
tmp = umx_make_twin_data_nice(data=twinData, sep="", zygosity="zygosity", numbering=1:2)
tmp = umx_scale_wide_twin_data(varsToScale= c("wt", "ht"), sep= "_T", data= tmp)
mzData = subset(tmp, zygosity %in%  c("MZFF", "MZMM"))
dzData = subset(tmp, zygosity %in%  c("DZFF", "DZMM"))

# ================
# = An ACE model =
# ================
# Define paths: You only need the paths for one person:
paths = c(
umxPath(v1m0 = c("a1", 'c1', "e1")),
umxPath(means = c("wt")),
umxPath(c("a1", 'c1', "e1"), to = "wt", values=.2)
)
m1 = umxTwinMaker("test", paths, mzData = mzData, dzData= dzData)
plot(m1, std= TRUE, means= FALSE)
m2 = umxACE(selDVs="wt", mzData = mzData, dzData=dzData, sep="_T")


# =====================
# = Bivariate example =
# =====================
latents = paste0(rep(c("a", "c", "e"), each = 2), 1:2)
biv = c(
umxPath(v1m0 = latents),
umxPath(mean = c("wt", "ht")),
umxPath(fromEach = c("a1", 'c1', "e1"), to = c("ht", "wt")),
umxPath(c("a2", 'c2', "e2"), to = "wt")
)
```

```
tmp= umxTwinMaker(paths= biv, mzData = mzData, dzData= dzData)
plot(tmp, means=FALSE)

# How to use latents other than a, c, and e: define in t1_t2links
paths = c(
umxPath(v1m0 = c("as1", 'c1', "e1")),
umxPath(means = c("wt")),
umxPath(c("as1", 'c1', "e1"), to = "wt", values=.2)
)
m1 = umxTwinMaker("test", paths, mzData = mzData, dzData= dzData,
t1_t2links = list('as'=c(1, .5), 'c'=c(1, 1), 'e'=c(0, 0))
)


## End(Not run)
```

---

umxUnexplainedCausalNexus *umxUnexplainedCausalNexus*

Description

umxUnexplainedCausalNexus report the effect of a change (delta) in a variable (from) on an output (to)

Usage

```
umxUnexplainedCausalNexus(from, delta, to, model = NULL)
```

Arguments

from	A variable in the model for which you want to compute the effect of a change.
delta	A the amount to simulate changing 'from' by.
to	The dependent variable that you want to watch changing.
model	The model containing variables from and to.

References

- https://www.github.com/tbates/umx/

See Also

- mxCheckIdentification(), mxCompare()

Other Modify or Compare Models: umxEquate(), umxFixAll(), umxMI(), umxModify(), umxSetParameters(), umx

Examples

```
## Not run:
umxUnexplainedCausalNexus(from="yrsEd", delta = .5, to = "income35", model)

## End(Not run)
```

---

umxValues *umxValues: Set values in RAM model, matrix, or path*

Description

For models to be estimated, it is essential that path values start at credible values. umxValues takes on that task for you. umxValues can set start values for the free parameters in both RAM and Matrix mxModel()s. It can also take an mxMatrix as input. It tries to be smart in guessing starts from the values in your data and the model type.

Usage

```
umxValues(obj = NA, sd = NA, n = 1, onlyTouchZeros = FALSE)
```

Arguments

obj	The RAM or matrix mxModel(), or mxMatrix() that you want to set start values for.
sd	Optional Standard Deviation for start values
n	Optional Mean for start values
onlyTouchZeros	Don't alter parameters that appear to have already been started (useful for speeding umxModify())

Details

note: If you give umxValues a numeric input, it will use obj as the mean, and return a list of length n, with sd = sd.

Value

- mxModel() with updated start values

References

- https://www.github.com/tbates/umx, https://tbates.github.io

See Also

- Core functions:

Other Advanced Model Building Functions: umxJiggle(), umxLabel(), umxThresholdMatrix(), umx

Examples

```
require(umx)
data(demoOneFactor)
latents = c("G")
manifests = names(demoOneFactor)

# ============================================================================
# = Make an OpenMx model (which will lack start values and labels..) =
# ============================================================================
m1 = mxModel("One Factor", type = "RAM",
manifestVars = manifests, latentVars = latents,
mxPath(from = latents  , to = manifests),
mxPath(from = manifests, arrows = 2),
mxPath(from = latents  , arrows = 2, free = FALSE, values = 1.0),
mxData(cov(demoOneFactor), type = "cov", numObs=500)
)
mxEval(S, m1) # default variances are jiggled away from near-zero
# Add start values to the model
m1 = umxValues(m1)
mxEval(S, m1) # plausible variances
umx_print(mxEval(S,m1), 3, zero.print = ".") # plausible variances
umxValues(14, sd = 1, n = 10) # Return vector of length 10, with mean 14 and sd 1
```

---

umxVersion *Get or print the version of umx, along with detail from OpenMx and general system info.*

Description

umxVersion returns the version information for umx, and for OpenMx and R. Essential for bug-reports! This function can also test for a minimum version.

Usage

```
umxVersion(model = NULL, min = NULL, verbose = TRUE, return = "umx")
```

Arguments

model	Optional to show optimizer in this model
min	Optional minimum version string to test for, e.g. '2.7.0' (Default = NULL).
verbose	= TRUE
return	Which package (umx or OpenMx) to 'return' version info for (Default = umx).

Value

- mxModel()

References

- https://github.com/tbates/umx, https://tbates.github.io

See Also

- packageVersion(), install.OpenMx()

Other Miscellaneous Utility Functions: install.OpenMx(), qm(), umxBrownie(), umxLav2RAM(), umxRAM2Lav(), umx_array_shift(), umx_find_object(), umx_msg(), umx_open_CRAN_page(), umx_pad(), umx_print(), umx_score_scale(), umx

Examples

```
x = umxVersion(); x
```

---

umxWeightedAIC *AIC weight-based conditional probabilities.*

Description

Returns the best model by AIC, and computes the probabilities according to AIC weight-based conditional probabilities (Wagenmakers & Farrell, 2004).

Usage

```
umxWeightedAIC(models, digits = 2)
```

Arguments

models	a list of models to compare.
digits	(default 2)

Value

- Best model

References

- Wagenmakers E.J., Farrell S. (2004), 192-196. AIC model selection using Akaike weights. *Psychonomic Bulletin and Review*. **11**, 192-196. https://www.ncbi.nlm.nih.gov/pubmed/15117008

See Also

- AIC()

Other Miscellaneous Stats Helpers: FishersMethod(), SE_from_p(), oddsratio(), reliability(), umxCov2cor(), umxHetCor(), umx_apply(), umx_cor(), umx_means(), umx_r_test(), umx_round(), umx_scale(), umx_var(), umx

Examples

```
l1 = lm(mpg~ wt + disp, data=mtcars)
l2 = lm(mpg~ wt, data=mtcars)
umxWeightedAIC(models = list(l1, l2))
```

`umx_aggregate` *Convenient formula-based cross-tabs & built-in summary functions*

Description

A common task is preparing summary tables, aggregating over some grouping factor. Like mean and sd of age, by sex. R's `aggregate()` function is useful and powerful, allowing xtabs based on a formula.

umx_aggregate makes using it a bit easier. In particular, it has some common functions for summarizing data built-in, like "mean (sd)" (the default).

```
umx_aggregate(mpg ~ cyl,data = mtcars,what = "mean_sd")
```

cyl	mpg
4 (n = 11)	26.66 (4.51)
6 (n = 7)	19.74 (1.45)
8 (n = 14)	15.1 (2.56)

Usage

```
umx_aggregate(
  formula = DV ~ condition,
  data = df,
  what = c("mean_sd", "n"),
  digits = 2,
  report = c("markdown", "html", "txt")
)
```

Arguments

`formula`	The aggregation formula. e.g., DV ~ condition.
`data`	frame to aggregate (defaults to df for common case)
`what`	function to use. Default reports "mean (sd)".
`digits`	to round results to.
`report`	Format for the table: Default is markdown.

Value

- table

References

- https://github.com/tbates/umx, https://tbates.github.io

See Also

- umx_apply(), aggregate()

Other Reporting Functions: loadings.MxModel(), umxAPA(), umxFactorScores(), umxGetParameters(), umxParameters(), umxReduce(), umx_names(), umx_time(), umx

Examples

```
# ====================================
# = Basic use, compare with aggregate =
# ====================================
aggregate(mpg ~ cyl, FUN = mean, na.rm = TRUE, data = mtcars)
umx_aggregate(mpg ~ cyl, data = mtcars)

# ============================================
# = Use different (or user-defined) functions =
# ============================================
umx_aggregate(mpg ~ cyl, data = mtcars, what = "n")
umx_aggregate(mpg ~ cyl, data = mtcars, what = function(x){sum(!is.na(x))})

# turn off markdown
umx_aggregate(mpg ~ cyl, data = mtcars, report = "txt")

# ==========================================
# = More than one item on the left hand side =
# ==========================================
umx_aggregate(cbind(mpg, qsec) ~ cyl, data = mtcars, digits = 3)
# Transpose table
t(umx_aggregate(cbind(mpg, qsec) ~ cyl, data = mtcars))

## Not run:
umx_aggregate(cbind(moodAvg, mood) ~ condition, data = study1)

## End(Not run)
```

---

umx_APA_pval *Round p-values according to APA guidelines*

Description

umx_APA_pval formats p-values, rounded in APA style. So you get '< .001' instead of .000000002 or 1.00E-09.

You probably would be better off using umxAPA(), which handles many more object types.

You set the precision with digits. Optionally, you can add '=' '<' etc. The default for addComparison (NA) adds these when needed.

Usage

```
umx_APA_pval(p, min = 0.001, digits = 3, addComparison = NA)
```

Arguments

p	The p-value to round
min	Values below min will be reported as "< min"
digits	Number of decimals to which to round (default = 3)
addComparison	Whether to add '=' '<' etc. (NA adds when needed)

Value

- p-value formatted in APA style

See Also

- umxAPA(), round()

Other xmu internal not for end user: umxModel(), umxRenameMatrix(), umxTwinMaker(), umx_fun_mean_sd(), umx_get_bracket_addresses(), umx_make(), umx_standardize(), umx_string_to_algebra(), umx, xmuHasSquareBrackets(), xmuLabel_MATRIX_Model(), xmuLabel_Matrix(), xmuLabel_RAM_Model(), xmuMI(), xmuMakeDeviationThresholdsMatrices(), xmuMakeOneHeadedPathsFromPathList(), xmuMakeTwoHeadedPathsFromPathList(), xmuMaxLevels(), xmuMinLevels(), xmuPropagateLabels(), xmuRAM2Ordinal(), xmuTwinSuper_Continuous(), xmuTwinUpgradeMeansToCovariateModel(), xmu_CI_merge(), xmu_CI_stash(), xmu_DF_to_mxData_TypeCov(), xmu_PadAndPruneForDefVars(), xmu_cell_is_on(), xmu_check_levels_identical(), xmu_check_needs_means(), xmu_check_variance(), xmu_clean_label(), xmu_data_missing(), xmu_data_swap_a_block(), xmu_describe_data_WLS(), xmu_dot_make_paths(), xmu_dot_make_residuals(), xmu_dot_maker(), xmu_dot_move_ranks(), xmu_dot_rank_str(), xmu_extract_column(), xmu_get_CI(), xmu_lavaan_process_group(), xmu_make_TwinSuperModel(), xmu_make_bin_cont_pair_data(), xmu_make_mxData(), xmu_match.arg(), xmu_name_from_lavaan_str(), xmu_path2twin(), xmu_path_regex(), xmu_safe_run_summary(), xmu_set_sep_from_suffix(), xmu_show_fit_or_comparison(), xmu_simplex_corner(), xmu_standardize_ACEcov(), xmu_standardize_ACEv(), xmu_standardize_ACE(), xmu_standardize_CP(), xmu_standardize_IP(), xmu_standardize_RAM(), xmu_standardize_SexLim(), xmu_standardize_Simplex(), xmu_start_value_list(), xmu_starts(), xmu_twin_add_WeightMatrices(), xmu_twin_check(), xmu_twin_get_var_names(), xmu_twin_upgrade_selDvs2SelVars()

Examples

```
umx_APA_pval(.052347)
umx_APA_pval(1.23E-3)
umx_APA_pval(1.23E-4)
umx_APA_pval(c(1.23E-3, .5))
umx_APA_pval(c(1.23E-3, .5), addComparison = TRUE)
```

umx_apply *umx_apply*

Description

Tries to make apply more readable. so "mean of x by columns", instead of "of x, by 2, mean" Other functions to think of include: cumsum(), rowSums(), colMeans(), etc.

Usage

```
umx_apply(FUN, of, by = c("columns", "rows"), ...)
```

Arguments

FUN	The function to apply.
of	The dataframe to work with.
by	Apply the function to columns or to rows (default = "columns")
...	optional arguments to FUN, e.g., na.rm = TRUE.

Value

- object

References

- https://tbates.github.io, https://github.com/tbates/umx

See Also

- umx_aggregate()

Other Miscellaneous Stats Helpers: FishersMethod(), SE_from_p(), oddsratio(), reliability(), umxCov2cor(), umxHetCor(), umxWeightedAIC(), umx_cor(), umx_means(), umx_r_test(), umx_round(), umx_scale(), umx_var(), umx

Examples

```
umx_apply(mean, mtcars, by = "columns")
umx_apply("mean", of = mtcars, by = "columns")
tmp = mtcars[1:3,]; tmp[1,1]=NA
umx_apply("mean", by = "rows", of = tmp)
umx_apply("mean", by = "rows", of = tmp, na.rm = TRUE)
```

umx_array_shift *Like the php array_shift function: shifts an item off the beginning of a list*

Description

Returns x[1]. Has the SIDE EFFECT of assigning x to x[2:end] in the container environment.

Usage

```
umx_array_shift(x)
```

Arguments

x	the vector to shift

Value

- first item of x

See Also

Other Miscellaneous Utility Functions: `install.OpenMx()`, `qm()`, `umxBrownie()`, `umxLav2RAM()`, `umxRAM2Lav()`, `umxVersion()`, `umx_find_object()`, `umx_msg()`, `umx_open_CRAN_page()`, `umx_pad()`, `umx_print()`, `umx_score_scale()`, `umx`

Examples

```
x = c("Alice", "Bob", "Carol")
umx_array_shift(x) # returns "Alice"
x # now only 2 items (altered in containing environment)
```

umx_as_numeric *umx_as_numeric*

Description

Convert each column of a dataframe to numeric

Usage

```
umx_as_numeric(df, which = NULL, force = FALSE)
```

Arguments

df	A [data.frame()] to convert
which	which columns to convert (default (null) selects all)
force	Whether to force conversion to numeric for non-numeric columns (defaults to FALSE)

Value

- data.frame

References

- <https://www.github.com/tbates/umx>

See Also

Other Data Functions: umxFactor(), umxHetCor(), umx_cont_2_quantiles(), umx_lower2full(), umx_make_MR_data(), umx_make_TwinData(), umx_make_fake_data(), umx_make_raw_from_cov(), umx_polychoric(), umx_polypairwise(), umx_polytriowise(), umx_read_lower(), umx_rename(), umx_reorder(), umx_select_valid(), umx_stack(), umx

Examples

```
df = mtcars
# make mpg into string, and cyl into a factor
df$mpg = as.character(df$mpg)
df$cyl = factor(df$cyl)

df = umx_as_numeric(df); str(df) # mpg not touched
df = umx_as_numeric(df, force=TRUE); str(df) # mpg coerced back to numeric
## Not run:
# coercing a real string will cause NAs
df$mpg = c(letters[1:16]); str(df) # replace mpg with letters.
df = umx_as_numeric(df, force=TRUE); str(df)

## End(Not run)
```

---

umx_check *umx_check*

Description

Check that a test evaluates to TRUE. If not, stop, warn, or message the user

Usage

```
umx_check(
  boolean.test,
  action = c("stop", "warning", "message"),
  message = "check failed",
  ...
)
```

Arguments

boolean.test	test evaluating to TRUE or FALSE.
action	One of "stop" (the default), "warning", or "message".
message	what to tell the user when boolean.test is FALSE.
...	extra text will be pasted after the messages.

Value

- boolean

See Also

Other Test: umx_check_OS(), umx_check_model(), umx_check_names(), umx_check_parallel(), umx_has_CIs(), umx_has_been_run(), umx_has_means(), umx_has_square_brackets(), umx_is_MxData(), umx_is_MxMatrix(), umx_is_MxModel(), umx_is_RAM(), umx_is_cov()

Examples

```
umx_check(length(1:3)==3, "message", "item must have length == 3", "another comment", "and another")
umx_check(1==2, "message", "one must be 2", ". Another comment", "and another")
```

---

umx_check_model *Check for required features in an OpenMx.*

Description

Allows the user to straight-forwardly require a specific model type (i.e., "RAM", "LISREL", etc.), whether or not the model has data, if it has been run or not. You can also test whether is has a means model or not and (in future) test if it has submodels.

Usage

```
umx_check_model(
  obj,
  type = NULL,
  hasData = NULL,
  beenRun = NULL,
```

```
  hasMeans = NULL,
  checkSubmodels = FALSE,
  callingFn = "a function"
)
```

Arguments

obj	an object to check
type	what type the model must be, i.e., "RAM", "LISREL", etc. (defaults to not checking NULL)
hasData	whether the model should have data or not (defaults to not checking NULL)
beenRun	whether the model has been run or not (defaults to not checking NULL)
hasMeans	whether the model should have a means model or not (defaults to not checking NULL)
checkSubmodels	whether to check submodels (not implemented yet) (default = FALSE)
callingFn	= Name of the calling function to help the user locate the error.

Value

- boolean

References

- <https://www.github.com/tbates/umx>

See Also

Other Test: umx_check_OS(), umx_check_names(), umx_check_parallel(), umx_check(), umx_has_CIs(), umx_has_been_run(), umx_has_means(), umx_has_square_brackets(), umx_is_MxData(), umx_is_MxMatrix(), umx_is_MxModel(), umx_is_RAM(), umx_is_cov()

Examples

```
require(umx)
data(demoOneFactor)
manifests = names(demoOneFactor)
m1 = umxRAM("check_model_ex", data = demoOneFactor, type = "cov",
umxPath("G", to = manifests),
umxPath(var = manifests),
umxPath(var = "G", fixedAt = 1)
)#'
umx_check_model(m1) # TRUE, this is a model
umx_check_model(m1, type = "RAM") # equivalent to umx_is_RAM()
umx_check_model(m1, hasData = TRUE)


## Not run:
umx_check_model(m1, hasMeans = TRUE)
umx_check_model(m1, beenRun = FALSE)
```

```
# Model with no data
m1 = umxRAM("x ~~ .3*y", autoRun = FALSE)
umx_check_model(m1, hasData = TRUE)

## End(Not run)
```

---

umx_check_names *Check if a request name exists in a dataframe or related object*

Description

Check if a list of names are in the [namez()] of a dataframe (or the [dimnames()] of a matrix), or the names of the observed data of an [mzData()]

Usage

```
umx_check_names(
  namesNeeded,
  data = NA,
  die = TRUE,
  no_others = FALSE,
  intersection = FALSE,
  message = ""
)
```

Arguments

namesNeeded	list of variable names to find (a dataframe is also allowed)
data	data.frame, matrix, or mxData to search in for names (default NA)
die	whether to die if the check fails (default TRUE)
no_others	Whether to test that the data contain no columns in addition to those in namesNeeded (default FALSE)
intersection	Show the intersection of names
message	Some helpful text to append when dieing.

References

- <https://www.github.com/tbates/umx>

See Also

Other Test: umx_check_OS(), umx_check_model(), umx_check_parallel(), umx_check(), umx_has_CIs(), umx_has_been_run(), umx_has_means(), umx_has_square_brackets(), umx_is_MxData(), umx_is_MxMatrix(), umx_is_MxModel(), umx_is_RAM(), umx_is_cov()

Other Check or test: umx_is_class(), umx_is_endogenous(), umx_is_exogenous(), umx_is_numeric(), umx_is_ordered(), umx

Examples

```
require(umx)
data(demoOneFactor) # "x1" "x2" "x3" "x4" "x5"
umx_check_names(c("x1", "x2"), demoOneFactor)
umx_check_names(c("x1", "x2"), as.matrix(demoOneFactor))
umx_check_names(c("x1", "x2"), cov(demoOneFactor[, c("x1","x2")]))
umx_check_names(c("x1", "x2"), mxData(demoOneFactor, type="raw"))
umx_check_names(c("z1", "x2"), data = demoOneFactor, die = FALSE)
umx_check_names(c("x1", "x2"), data = demoOneFactor, die = FALSE, no_others = TRUE)
umx_check_names(c("x1","x2","x3","x4","x5"), data = demoOneFactor, die = FALSE, no_others = TRUE)
# no request
umx_check_names(c(), data = demoOneFactor, die = FALSE, no_others = TRUE)

## Not run:
# An example error from vars that don't exist in the data
umx_check_names(c("bad_var_name", "x2"), data = demoOneFactor, die = TRUE)

## End(Not run)
```

---

umx_check_OS *umx_check_OS*

Description

Check what OS we are running on (current default is OS X). Returns a boolean. Optionally warn or die on failure of the test

Usage

```
umx_check_OS(
  target = c("OSX", "SunOS", "Linux", "Windows"),
  action = c("ignore", "warn", "die")
)
```

Arguments

target	Which OS(s) you wish to check for (default = "OSX")
action	What to do on failure of the test: nothing (default), warn or die

Value

- TRUE if on the specified OS (else FALSE)

References

- https://github.com/tbates/umx, https://tbates.github.io

See Also

Other Test: umx_check_model(), umx_check_names(), umx_check_parallel(), umx_check(), umx_has_CIs(), umx_has_been_run(), umx_has_means(), umx_has_square_brackets(), umx_is_MxData(), umx_is_MxMatrix(), umx_is_MxModel(), umx_is_RAM(), umx_is_cov()

Examples

```
umx_check_OS()
```

---

umx_check_parallel *Check if OpenMx is using OpenMP, test cores, and get timings*

Description

Shows how many cores you are using, and runs a test script so user can check CPU usage.

Usage

```
umx_check_parallel(
  nCores = c(1, imxGetNumThreads()),
  testScript = NULL,
  rowwiseParallel = TRUE,
  nSubjects = 1000
)
```

Arguments

nCores	How many cores to run (defaults to c(1, max). -1 = all available.
testScript	A user-provided script to run (NULL)
rowwiseParallel	Whether to parallel-ize rows (default) or gradient computation
nSubjects	Number of rows to model (Default = 1000) Reduce for quicker runs.

Details

Some historical (starting 2017-09-06) speeds on my late 2015 iMac, 3.3 GHz Quad-core i7 desktop.

date	type	Cores	Time	
2019-06-13	v2.13.2 (OpenMP git)	1 core	01 min, 11 sec	(NPSOL)
2019-06-13	v2.13.2 (OpenMP git)	4 core	00 min, 22 sec	(NPSOL)
2019-06-13	v2.13.2 (OpenMP git)	6 core	00 min, 21 sec	(NPSOL)
2018-10-14	v2.11.5 (OpenMP on CRAN)	4 cores	00 min, 36 sec	Δ:-39.598)
2018-09-17	v2.11.3	1	01 min, 31 sec	
2018-09-17	v2.11.3	4	00 min, 30.6 sec	Δ: -61.49)
2017-10-16	v2.7.18-9	1	01 min, 07.30 sec	
2017-10-16	v2.7.18-9	4	00 min, 22.63 sec	Δ: -44.68)

2017-10-16	Clang OpenMP	1	01 min, 08.38 sec	
2017-10-16	Clang OpenMP	4	00 min, 24.89 sec	Δ: -43.49)
2017-09-07	Clang OpenMP	1	01 min, 12.90 sec	
2017-09-07	Clang OpenMP	4	00 min, 32.20 sec	Δ: -40.70
2017-09-07	Clang notOpenMP	1	01 min, 09.90 sec	
2017-09-07	TRAVIS	1	01 min, 06.20 sec	
2017-09-07	TRAVIS	4	00 min, 21.10 sec	Δ: -45.00

Value

None

References

- https://tbates.github.io, https://github.com/tbates/umx

See Also

Other Test: umx_check_OS(), umx_check_model(), umx_check_names(), umx_check(), umx_has_CIs(), umx_has_been_run(), umx_has_means(), umx_has_square_brackets(), umx_is_MxData(), umx_is_MxMatrix(), umx_is_MxModel(), umx_is_RAM(), umx_is_cov()

Examples

```
## Not run:
# On a fast machine, takes a minute with 1 core
umx_check_parallel()

## End(Not run)
```

---

umx_cont_2_quantiles *umx_cont_2_quantiles*

Description

Recode a continuous variable into n-quantiles (default = deciles (10 levels)). It returns an mxFactor(), with the levels labeled with the max value in each quantile (i.e., open on the left-side). quantiles are labeled "quantile1" "quantile2" etc.

Usage

```
umx_cont_2_quantiles(
  x,
  nlevels = NULL,
  type = c("mxFactor", "ordered", "unordered"),
  verbose = FALSE,
  returnCutpoints = FALSE
)
```

Arguments

x	a variable to recode as ordinal (email maintainer("umx") if you'd like this upgraded to handle df input)
nlevels	How many bins or levels (at most) to use (i.e., 10 = deciles)
type	what to return (Default is "mxFactor") options: "ordered" and "unordered")
verbose	report the min, max, and decile cuts used (default = FALSE)
returnCutpoints	just return the cutpoints, for use directly

Details

Note: Redundant quantiles are merged. i.e., if the same score identifies all deciles up to the fourth, then these will be merged into one bin, labeled "quantile4".

Value

- recoded variable as an mxFactor()

References

- https://github.com/tbates/umx, https://tbates.github.io

See Also

Other Data Functions: umxFactor(), umxHetCor(), umx_as_numeric(), umx_lower2full(), umx_make_MR_data(), umx_make_TwinData(), umx_make_fake_data(), umx_make_raw_from_cov(), umx_polychoric(), umx_polypairwise(), umx_polytriowise(), umx_read_lower(), umx_rename(), umx_reorder(), umx_select_valid(), umx_stack(), umx

Examples

```
x = umx_cont_2_quantiles(rnorm(1000), nlevels = 10, verbose = TRUE)
x = data.frame(x)
str(x); levels(x)
table(x)
## Not run:
ggplot2::qplot(x$x)
y = mxDataWLS(x, type = "WLS")

## End(Not run)

# ============================
# = Use with twin variables =
# ============================

data(twinData)
x = twinData
cuts  = umx_cont_2_quantiles(rbind(x$wt1, x$wt2) , nlevels = 10, returnCutpoints = TRUE)
x$wt1 = umx_cont_2_quantiles(x$wt1, nlevels = cuts) # use same for both...
```

```
x$wt2 = umx_cont_2_quantiles(x$wt2, nlevels = cuts) # use same for both...
str(x[, c("wt1", "wt2")])

# More examples

x = umx_cont_2_quantiles(mtcars[, "mpg"], nlevels = 5) # quintiles
x = umx2ord(mtcars[, "mpg"], nlevels = 5) # using shorter alias
x = umx_cont_2_quantiles(mtcars[, "cyl"], nlevels = 10) # more levels than integers exist
x = umx_cont_2_quantiles(rbinom(10000, 1, .5), nlevels = 2)
```

---

umx_cor *Report correlations and their p-values*

Description

For reporting correlations and their p-values in a compact table. Handles rounding, and skipping non-numeric columns.

Usage

```
umx_cor(
  X,
  df = nrow(X) - 2,
  use = c("pairwise.complete.obs", "complete.obs", "everything", "all.obs",
    "na.or.complete"),
  digits = 2,
  type = c("r and p-value", "smart")
)
```

Arguments

X	a matrix or dataframe
df	the degrees of freedom for the test
use	how to handle missing data (defaults to pairwise complete)
digits	rounding of answers
type	Unused argument for future directions

Details

To compute heterochoric correlations, see umxHetCor().

note: The Hmisc package has a more robust function called rcorr.

Value

- Matrix of correlations and p-values

References

- https://www.github.com/tbates/umx

See Also

umxHetCor

Other Miscellaneous Stats Helpers: FishersMethod(), SE_from_p(), oddsratio(), reliability(), umxCov2cor(), umxHetCor(), umxWeightedAIC(), umx_apply(), umx_means(), umx_r_test(), umx_round(), umx_scale(), umx_var(), umx

Examples

```
tmp = myFADataRaw[1:8,1:8]
umx_cor(tmp)
tmp$x1 = letters[1:8] # make one column non-numeric
umx_cor(tmp)
```

---

umx_explode *Explode a string (Like the php function* explode*)*

Description

Takes a string and returns an array of delimited strings (by default, each single character)

Usage

```
umx_explode(delimiter = character(), string)
```

Arguments

delimiter	what to break the string on. Default is empty string ""
string	an character string, e.g. "dog"

Value

- a vector of strings, e.g. c("d", "o", "g")

References

- https://tbates.github.io, http://php.net/manual/en/function.explode.php

See Also

Other String Functions: umx_explode_twin_names(), umx_grep(), umx_names(), umx_paste_names(), umx_rot(), umx_str_chars(), umx_str_from_object(), umx_trim(), umx

Examples

```
umx_explode("", "dog") # "d" "o" "g"
umx_explode(" ", "cats and dogs") # [1] "cats" "and"  "dogs"
```

---

umx_explode_twin_names

Break twin variable names (BMI_T1, BMI_T2) into base variable names (BMI, "_T", 1:2)

Description

Break names like Dep_T1 into a list of base names, a separator, and a vector of twin indexes. e.g.: c("Dep_T1", "Dep_T2", "Anx_T1", "Anx_T2") will become:

list(baseNames = c("Dep", "Anx"), sep = "_T", twinIndexes = c(1,2))

Usage

```
umx_explode_twin_names(df, sep = "_T")
```

Arguments

df	vector of names or data.frame containing the data
sep	text constant separating name from numeric 1:2 twin index.

Value

- list(baseNames, sep, twinIndexes)

See Also

[umx_paste_names()]

Other String Functions: umx_explode(), umx_grep(), umx_names(), umx_paste_names(), umx_rot(), umx_str_chars(), umx_str_from_object(), umx_trim(), umx

Examples

```
require(umx)
data("twinData")
umx_explode_twin_names(twinData, sep = "")
umx_explode_twin_names(twinData, sep = NULL)

# Ignore this: just a single-character/single variable test case
x = round(10 * rnorm(1000, mean = -.2))
y = round(5 * rnorm(1000))
x[x < 0] = 0; y[y < 0] = 0
umx_explode_twin_names(data.frame(x_T1 = x, x_T2 = y), sep = "_T")
umx_explode_twin_names(data.frame(x_T11 = x, x_T22 = y), sep = "_T")
umx_explode_twin_names(c("x_T11", "x_T22"), sep = "_T")
```

umx_file_load_pseudo *Read in files from pseudocons.*

Description

Read in PRS scored files from pseudocons.

1. Read the file
2. Break it into pseudo and real rows
3. Clean-up by deleting the pseudo suffix
4. Rename NT vars with a suffix
5. Merge files on ID and return

	ID	FID	BMIS1	BMIS2	BMIS3	BMIS4	...
1	1234501	12345	-0.032	-0.77	-0.40	-3.87	...
2	1234501-pseudo-1	12345	0.117	-0.66	-0.33	-4.08	...

Usage

```
umx_file_load_pseudo(fn, bp, suffix = "_NT", chosenp = "S5")
```

Arguments

fn	The filename
bp	The path to the folder containing the file
suffix	to add to the NT columns (Default = "_NT")
chosenp	The suffix (pvalue) we desire to use (Default = "S5")

Value

- dataframe of real and pseudo PRS columns

See Also

Other File Functions: dl_from_dropbox(), umx_make_sql_from_excel(), umx_move_file(), umx_open(), umx_rename_file(), umx_write_to_clipboard(), umx

Examples

```
## Not run:
basepath = "~/Dropbox/2016 (1). project EA/2018/EA3/"
tmp = umx_file_load_pseudo("PRS_EA3_R9_autosomes_HRC1.1_pseudo.txt", bp = bp)
str(tmp)
head(tmp[, c("BMIS4", "BMIS4_NT")]
```

```
## End(Not run)
```

---

umx_find_object *umx_find_object*

Description

Find objects of a given class, whose name matches a search string. The string (pattern) is grep-enabled, so you can match wild-cards

Usage

```
umx_find_object(pattern = ".*", requiredClass = "MxModel")
```

Arguments

pattern	the pattern that matching objects must contain
requiredClass	the class of object that will be matched

Value

- a list of objects matching the class and name

References

-

See Also

Other Miscellaneous Utility Functions: install.OpenMx(), qm(), umxBrownie(), umxLav2RAM(), umxRAM2Lav(), umxVersion(), umx_array_shift(), umx_msg(), umx_open_CRAN_page(), umx_pad(), umx_print(), umx_score_scale(), umx

Examples

```
## Not run:
umx_find_object("^m[0-9]") # mxModels beginning "m1" etc.
umx_find_object("", "MxModel") # all MxModels

## End(Not run)
```

umx_fun_mean_sd *Summarizing functions used in umx_aggregate and for umxAPA*

Description

Miscellaneous functions that are handy in summary and other tasks where you might otherwise have to craft a custom nameless functions. e.g.

Usage

```
umx_fun_mean_sd(x, na.rm = TRUE, digits = 2)
```

Arguments

x	input
na.rm	How to handle missing (default = TRUE = remove)
digits	Rounding (default = 2)

Details

- umx_fun_mean_sd(): returns "mean (SD)" of x.

note: if a factor is given, then the mode is returned instead of the mean and SD.

Value

- function result

References

- https://github.com/tbates/umx, https://tbates.github.io

See Also

Other xmu internal not for end user: umxModel(), umxRenameMatrix(), umxTwinMaker(), umx_APA_pval(), umx_get_bracket_addresses(), umx_make(), umx_standardize(), umx_string_to_algebra(), umx, xmuHasSquareBrackets(), xmuLabel_MATRIX_Model(), xmuLabel_Matrix(), xmuLabel_RAM_Model(), xmuMI(), xmuMakeDeviationThresholdsMatrices(), xmuMakeOneHeadedPathsFromPathList(), xmuMakeTwoHeadedPathsFromPathList(), xmuMaxLevels(), xmuMinLevels(), xmuPropagateLabels(), xmuRAM2Ordinal(), xmuTwinSuper_Continuous(), xmuTwinUpgradeMeansToCovariateModel(), xmu_CI_merge(), xmu_CI_stash(), xmu_DF_to_mxData_TypeCov(), xmu_PadAndPruneForDefVars(), xmu_cell_is_on(), xmu_check_levels_identical(), xmu_check_needs_means(), xmu_check_variance(), xmu_clean_label(), xmu_data_missing(), xmu_data_swap_a_block(), xmu_describe_data_WLS(), xmu_dot_make_paths(), xmu_dot_make_residuals(), xmu_dot_maker(), xmu_dot_move_ranks(), xmu_dot_rank_str(), xmu_extract_column(), xmu_get_CI(), xmu_lavaan_process_group(), xmu_make_TwinSuperModel(), xmu_make_bin_cont_pair_data(), xmu_make_mxData(), xmu_match.arg(), xmu_name_from_lavaan_str(), xmu_path2twin(), xmu_path_regex(), xmu_safe_run_summary(), xmu_set_sep_from_suffix(), xmu_show_fit_or_comparison(), xmu_simplex_corner(), xmu_standardize_ACEcov()

xmu_standardize_ACEv(), xmu_standardize_ACE(), xmu_standardize_CP(), xmu_standardize_IP(), xmu_standardize_RAM(), xmu_standardize_SexLim(), xmu_standardize_Simplex(), xmu_start_value_list(), xmu_starts(), xmu_twin_add_WeightMatrices(), xmu_twin_check(), xmu_twin_get_var_names(), xmu_twin_upgrade_selDvs2SelVars()

Examples

```
umxAPA(mtcars[,1:3]) # uses umx_fun_mean_sd
```

---

umx_get_bracket_addresses

Get bracket-style addresses from an mxMatrix

Description

Sometimes you want these :-) This also allows you to change the matrix name: useful for using mxMatrix addresses in an mxAlgebra.

Usage

```
umx_get_bracket_addresses(mat, free = NA, newName = NA)
```

Arguments

mat	an mxMatrix to get address labels from
free	how to filter on free (default = NA: take all)
newName	= NA

Value

- a list of bracket style labels

References

- https://tbates.github.io, https://github.com/tbates/umx

See Also

Other xmu internal not for end user: umxModel(), umxRenameMatrix(), umxTwinMaker(), umx_APA_pval(), umx_fun_mean_sd(), umx_make(), umx_standardize(), umx_string_to_algebra(), umx, xmuHasSquareBrackets(), xmuLabel_MATRIX_Model(), xmuLabel_Matrix(), xmuLabel_RAM_Model(), xmuMI(), xmuMakeDeviationThresholdsMat, xmuMakeOneHeadedPathsFromPathList(), xmuMakeTwoHeadedPathsFromPathList(), xmuMaxLevels(), xmuMinLevels(), xmuPropagateLabels(), xmuRAM2Ordinal(), xmuTwinSuper_Continuous(), xmuTwinUpgradeMeansToCovariateModel(), xmu_CI_merge(), xmu_CI_stash(), xmu_DF_to_mxData_TypeCov(), xmu_PadAndPruneForDefVars(), xmu_cell_is_on(), xmu_check_levels_identical(), xmu_check_needs_means(), xmu_check_variance(), xmu_clean_label(), xmu_data_missing(), xmu_data_swap_a_block(), xmu_describe_data_WLS(), xmu_dot_make_paths(), xmu_dot_make_residuals(), xmu_dot_maker(),

xmu_dot_move_ranks(), xmu_dot_rank_str(), xmu_extract_column(), xmu_get_CI(), xmu_lavaan_process_group()
xmu_make_TwinSuperModel(), xmu_make_bin_cont_pair_data(), xmu_make_mxData(), xmu_match.arg(),
xmu_name_from_lavaan_str(), xmu_path2twin(), xmu_path_regex(), xmu_safe_run_summary(),
xmu_set_sep_from_suffix(), xmu_show_fit_or_comparison(), xmu_simplex_corner(), xmu_standardize_ACEcov()
xmu_standardize_ACEv(), xmu_standardize_ACE(), xmu_standardize_CP(), xmu_standardize_IP(),
xmu_standardize_RAM(), xmu_standardize_SexLim(), xmu_standardize_Simplex(), xmu_start_value_list(),
xmu_starts(), xmu_twin_add_WeightMatrices(), xmu_twin_check(), xmu_twin_get_var_names(),
xmu_twin_upgrade_selDvs2SelVars()

Examples

```
require(umx)
data(demoOneFactor)
manifests = names(demoOneFactor)
m1 = umxRAM("get_add_ex", data = demoOneFactor, type = "cov",
umxPath("G", to = manifests),
umxPath(var = manifests),
umxPath(var = "G", fixedAt = 1)
)#'
umx_get_bracket_addresses(m1$matrices$A, free= TRUE)
```

---

umx_get_checkpoint *Get or set checkpointing for a model*

Description

Get the checkpoint status for a model or global options

Usage

```
umx_get_checkpoint(model = NULL)
```

Arguments

model	an optional model to get options from

Value

None

References

- https://tbates.github.io

See Also

Other Get and set: umx_get_options(), umx_set_auto_plot(), umx_set_auto_run(), umx_set_checkpoint(),
umx_set_condensed_slots(), umx_set_cores(), umx_set_data_variance_check(), umx_set_mvn_optimization_opt
umx_set_optimizer(), umx_set_plot_file_suffix(), umx_set_plot_format(), umx_set_separator(),
umx_set_silent(), umx_set_table_format(), umx

Examples

```
umx_get_checkpoint() # current global default
require(umx)
data(demoOneFactor)
manifests = names(demoOneFactor)
m1 = umxRAM("One Factor", data = demoOneFactor, type = "cov",
umxPath("G", to = manifests),
umxPath(var = manifests),
umxPath(var = "G", fixedAt = 1)
)#' m1 = umx_set_checkpoint(interval = 2, model = m1)
umx_get_checkpoint(model = m1)
```

---

umx_get_options *umx_get_options*

Description

Show the umx options. Useful for beginners to discover, or people like me to remember :-)

Usage

```
umx_get_options()
```

Value

- message

See Also

Other Get and set: umx_get_checkpoint(), umx_set_auto_plot(), umx_set_auto_run(), umx_set_checkpoint(), umx_set_condensed_slots(), umx_set_cores(), umx_set_data_variance_check(), umx_set_mvn_optimization_opt umx_set_optimizer(), umx_set_plot_file_suffix(), umx_set_plot_format(), umx_set_separator(), umx_set_silent(), umx_set_table_format(), umx

Examples

```
umx_get_options()
```

umx_grep *Search for text*

Description

Search names if given a data.frame, or strings if given a vector of strings.

Usage

```
umx_grep(
  df,
  grepString,
  output = c("both", "label", "name"),
  ignore.case = TRUE,
  useNames = FALSE
)
```

Arguments

df	The data.frame() or string to search.
grepString	the search string.
output	the column name, the label, or both (default).
ignore.case	whether to be case sensitive or not (default TRUE = ignore case).
useNames	whether to search the names as well as the labels (for SPSS files with label metadata).

Details

The namez function is more flexible. A handy feature of umx_grep is that it can search the labels of data imported from SPSS.

nb: To simply grep for a pattern in a string use R's built-in grep() functions, e.g.: grepl("^NA\\[0-9]", "NA.3")

Value

- list of matched column names and/or labels.

References

- https://www.github.com/tbates/umx

See Also

- namez(), umx_aggregate(), grep()

Other String Functions: umx_explode_twin_names(), umx_explode(), umx_names(), umx_paste_names(), umx_rot(), umx_str_chars(), umx_str_from_object(), umx_trim(), umx

Examples

```
umx_grep(mtcars, "hp", output="both", ignore.case= TRUE)
umx_grep(c("hp", "ph"), "hp")
umx_grep(mtcars, "^h.*", output="both", ignore.case= TRUE)
## Not run:
umx_grep(spss_df, "labeltext", output = "label")
umx_grep(spss_df, "labeltext", output = "name")

## End(Not run)
```

---

umx_has_been_run *umx_has_been_run*

Description

check if an mxModel has been run or not

Usage

```
umx_has_been_run(model, stop = FALSE)
```

Arguments

model	The mxModel() you want to check has been run
stop	Whether to stop if the model has not been run (defaults to FALSE)

Value

- boolean

References

- https://www.github.com/tbates/umx

See Also

Other Test: umx_check_OS(), umx_check_model(), umx_check_names(), umx_check_parallel(), umx_check(), umx_has_CIs(), umx_has_means(), umx_has_square_brackets(), umx_is_MxData(), umx_is_MxMatrix(), umx_is_MxModel(), umx_is_RAM(), umx_is_cov()

Examples

```
require(umx)
data(demoOneFactor)
manifests = names(demoOneFactor)
m1 = umxRAM("has_been_run_example", data = demoOneFactor, type = "cov",
umxPath("G", to = manifests),
umxPath(var = manifests),
```

```
umxPath(var = "G", fixedAt = 1)
)#'
umx_has_been_run(m1)
```

---

umx_has_CIs *umx_has_CIs*

Description

A utility function to return a binary answer to the question "does this mxModel() have confidence intervals?"

Usage

```
umx_has_CIs(model, check = c("both", "intervals", "output"))
```

Arguments

model	The mxModel() to check for presence of CIs
check	What to check for: "intervals" requested, "output" present, or "both". Defaults to "both"

Value

- TRUE or FALSE

References

- https://www.github.com/tbates/umx

See Also

Other Test: umx_check_OS(), umx_check_model(), umx_check_names(), umx_check_parallel(), umx_check(), umx_has_been_run(), umx_has_means(), umx_has_square_brackets(), umx_is_MxData(), umx_is_MxMatrix(), umx_is_MxModel(), umx_is_RAM(), umx_is_cov()

Examples

```
require(umx)
data(demoOneFactor)
manifests = names(demoOneFactor)
m1 = umxRAM("_has_CI_ex", data = demoOneFactor, type = "cov",
umxPath("g", to = manifests),
umxPath(var = manifests),
umxPath(var = "g", fixedAt = 1.0)
)

umx_has_CIs(m1) # FALSE: no CIs and no output
m1 = mxModel(m1, mxCI("g_to_x1"))
```

```
umx_has_CIs(m1, check = "intervals") # TRUE intervals set
umx_has_CIs(m1, check = "output")  # FALSE not yet run
m1 = mxRun(m1)
umx_has_CIs(m1, check = "output")  # Still FALSE: Set and Run
## Not run:
m1 = mxRun(m1, intervals = TRUE)
umx_has_CIs(m1, check = "output")  # TRUE: Set, and Run with intervals = T
umxSummary(m1)

## End(Not run)
```

---

umx_has_means *umx_has_means*

Description

A utility function to return a binary answer to the question "does this mxModel() have a means model?"

Usage

```
umx_has_means(model)
```

Arguments

model	The mxModel() to check for presence of means

Value

- TRUE or FALSE

References

- https://www.github.com/tbates/umx

See Also

Other Test: umx_check_OS(), umx_check_model(), umx_check_names(), umx_check_parallel(), umx_check(), umx_has_CIs(), umx_has_been_run(), umx_has_square_brackets(), umx_is_MxData(), umx_is_MxMatrix(), umx_is_MxModel(), umx_is_RAM(), umx_is_cov()

Examples

```
require(umx)
data(demoOneFactor)
manifests = names(demoOneFactor)
m1 = umxRAM("has_means_ex", data = demoOneFactor, type = "cov",
umxPath("G", to = manifests),
```

```
umxPath(var = manifests),
umxPath(var = "G", fixedAt = 1)
)#'
umx_has_means(m1)
m1 <- mxModel(m1,
mxPath(from = "one", to = manifests),
mxData(demoOneFactor[1:100,], type = "raw")
)
umx_has_means(m1)
m1 = mxRun(m1)
umx_has_means(m1)
```

---

umx_has_square_brackets

Check if a label contains square brackets

Description

Helper function to check if a label has square brackets, e.g. "A[1,1]"

Usage

```
umx_has_square_brackets(input)
```

Arguments

input	The label to check for square brackets (string input)

Value

- boolean

References

- https://www.github.com/tbates/umx

See Also

Other Test: umx_check_OS(), umx_check_model(), umx_check_names(), umx_check_parallel(), umx_check(), umx_has_CIs(), umx_has_been_run(), umx_has_means(), umx_is_MxData(), umx_is_MxMatrix(), umx_is_MxModel(), umx_is_RAM(), umx_is_cov()

Examples

```
umx_has_square_brackets("[hello]")
umx_has_square_brackets("goodbye")
```

umx_is_class *Check if variables in a dataframe are in a list of classes.*

Description

Checks the class of each column in a dataframe, seeing if they are %in% a list of classes. Returns a vector of TRUE and FALSE, or, if all ==TRUE, a single binary (the default).

Usage

```
umx_is_class(df, classes = NULL, all = TRUE)
```

Arguments

df	A dataframe to check
classes	vector of valid classes, e.g. numeric
all	Whether to return a single all() Boolean or each column individually.

Value

- Boolean or Boolean vector

References

- https://github.com/tbates/umx, https://tbates.github.io

See Also

- umx_is_numeric()

Other Check or test: umx_check_names(), umx_is_endogenous(), umx_is_exogenous(), umx_is_numeric(), umx_is_ordered(), umx

Examples

```
umx_is_class(mtcars) # report class list
# Are the variables in mtcars type character?
umx_is_class(mtcars, "character") # FALSE
# They're all numeric data
umx_is_class(mtcars, "numeric") # TRUE
# Show the test-result for each variable in mtcars
umx_is_class(mtcars, "numeric") # TRUE
# Are they _either_ a char OR a num?
umx_is_class(mtcars, c("character", "numeric"))
# Is zygosity a factor (note we don't drop = F to keep as dataframe)
umx_is_class(twinData[,"zygosity", drop=FALSE], classes = "factor")
umx_is_class(mtcars$mpg) # report class of this column (same as class(mpg))
```

umx_is_cov *umx_is_cov*

Description

test if a data frame, matrix or mxData is type cov or cor, or is likely to be raw...

Usage

```
umx_is_cov(data = NULL, boolean = FALSE, verbose = FALSE)
```

Arguments

data	dataframe to test
boolean	whether to return the type ("cov") or a boolean (default = string)
verbose	How much feedback to give (default = FALSE)

Value

- "raw", "cor", or "cov", (or if boolean, then T | F)

References

- <https://www.github.com/tbates/umx>

See Also

Other Test: umx_check_OS(), umx_check_model(), umx_check_names(), umx_check_parallel(), umx_check(), umx_has_CIs(), umx_has_been_run(), umx_has_means(), umx_has_square_brackets(), umx_is_MxData(), umx_is_MxMatrix(), umx_is_MxModel(), umx_is_RAM()

Examples

```
df = cov(mtcars)
umx_is_cov(df)
df = cor(mtcars)
umx_is_cov(df)
umx_is_cov(mxData(df[1:3,1:3], type= "cov", numObs = 200))
umx_is_cov(df, boolean = TRUE)
umx_is_cov(mtcars, boolean = TRUE)
```

umx_is_endogenous *List endogenous variables in a model*

Description

Return a list of all the endogenous variables (variables with at least one incoming single-arrow path) in a model.

Usage

```
umx_is_endogenous(model, manifests_only = TRUE)
```

Arguments

model	an mxModel() from which to get endogenous variables
manifests_only	Whether to check only manifests (default = TRUE)

Value

- list of endogenous variables

References

- https://tbates.github.io, https://github.com/tbates/umx

See Also

Other Check or test: umx_check_names(), umx_is_class(), umx_is_exogenous(), umx_is_numeric(), umx_is_ordered(), umx

Examples

```
require(umx)
data(demoOneFactor)
m1 = umxRAM("umx_is_endogenous", data = demoOneFactor, type = "cov",
umxPath("g", to = names(demoOneFactor)),
umxPath(var = "g", fixedAt = 1),
umxPath(var = names(demoOneFactor))
)
umx_is_endogenous(m1, manifests_only = TRUE)
umx_is_endogenous(m1, manifests_only = FALSE)
```

umx_is_exogenous *umx_is_exogenous*

Description

Return a list of all the exogenous variables (variables with no incoming single-arrow path) in a model.

Usage

```
umx_is_exogenous(model, manifests_only = TRUE)
```

Arguments

model	an mxModel() from which to get exogenous variables
manifests_only	Whether to check only manifests (default = TRUE)

Value

- list of exogenous variables

References

- https://tbates.github.io, https://github.com/tbates/umx

See Also

Other Check or test: umx_check_names(), umx_is_class(), umx_is_endogenous(), umx_is_numeric(), umx_is_ordered(), umx

Examples

```
require(umx)
data(demoOneFactor)
m1 = umxRAM("One Factor", data = demoOneFactor, type = "cov",
umxPath("g", to = names(demoOneFactor)),
umxPath(var = "g", fixedAt = 1),
umxPath(var = names(demoOneFactor))
)
umx_is_exogenous(m1, manifests_only = TRUE)
umx_is_exogenous(m1, manifests_only = FALSE)
```

umx_is_MxData *Check if an object is an mxData object*

Description

Is the input an MxData?

Usage

```
umx_is_MxData(x)
```

Arguments

x	An object to test for being an MxData object

Value

- Boolean

References

- <https://www.github.com/tbates/umx>

See Also

Other Test: umx_check_OS(), umx_check_model(), umx_check_names(), umx_check_parallel(), umx_check(), umx_has_CIs(), umx_has_been_run(), umx_has_means(), umx_has_square_brackets(), umx_is_MxMatrix(), umx_is_MxModel(), umx_is_RAM(), umx_is_cov()

Examples

```
umx_is_MxData(mtcars)
umx_is_MxData(mxData(mtcars, type= "raw"))
umx_is_MxData(mxData(cov(mtcars), type= "cov", numObs = 73))
umx_is_MxData(mxDataWLS(na.omit(twinData[, c("wt1", "wt2")]), type= "WLS"))
```

umx_is_MxMatrix *umx_is_MxMatrix*

Description

Utility function returning a binary answer to the question "Is this an OpenMx mxMatrix?"

Usage

```
umx_is_MxMatrix(obj)
```

Arguments

obj	an object to be tested to see if it is an OpenMx mxMatrix()

Value

- Boolean

References

- https://www.github.com/tbates/umx

See Also

Other Test: umx_check_OS(), umx_check_model(), umx_check_names(), umx_check_parallel(), umx_check(), umx_has_CIs(), umx_has_been_run(), umx_has_means(), umx_has_square_brackets(), umx_is_MxData(), umx_is_MxModel(), umx_is_RAM(), umx_is_cov()

Examples

```
x = mxMatrix(name = "eg", type = "Full", nrow = 3, ncol = 3, values = .3)
if(umx_is_MxMatrix(x)){
message("nice OpenMx matrix!")
}
```

---

umx_is_MxModel *umx_is_MxModel*

Description

Utility function returning a binary answer to the question "Is this an OpenMx model?"

Usage

```
umx_is_MxModel(obj, listOK = FALSE)
```

Arguments

obj	An object to be tested to see if it is an OpenMx mxModel()
listOK	Is it acceptable to pass in a list of models? (Default = FALSE)

Value

- Boolean

References

- https://www.github.com/tbates/umx

See Also

Other Test: umx_check_OS(), umx_check_model(), umx_check_names(), umx_check_parallel(), umx_check(), umx_has_CIs(), umx_has_been_run(), umx_has_means(), umx_has_square_brackets(), umx_is_MxData(), umx_is_MxMatrix(), umx_is_RAM(), umx_is_cov()

Examples

```
m1 = mxModel("test")
if(umx_is_MxModel(m1)){
message("nice OpenMx model!")
}
if(umx_is_MxModel(list(m1,m1), listOK = TRUE)){
message("nice list of OpenMx models!")
}
```

---

umx_is_numeric *Check if variables in a dataframe are numeric*

Description

Checks across columns of a dataframe, return a vector of TRUE and FALSE, or, if all ==TRUE, a single binary (the default).

Usage

```
umx_is_numeric(df, all = TRUE)
```

Arguments

df	A dataframe to check
all	Whether to return a single all() Boolean or each column individually.

Value

- Boolean or Boolean vector

References

- https://github.com/tbates/umx, https://tbates.github.io

See Also

- umx_is_class()

Other Check or test: umx_check_names(), umx_is_class(), umx_is_endogenous(), umx_is_exogenous(), umx_is_ordered(), umx

Examples

```
umx_is_numeric(mtcars) # TRUE
umx_is_numeric(mtcars, all=FALSE) # vector of TRUE
```

---

umx_is_ordered *Test if one or more variables in a dataframe are ordered*

Description

Return the names of any ordinal variables in a dataframe

Usage

```
umx_is_ordered(
  df,
  names = FALSE,
  strict = TRUE,
  binary.only = FALSE,
  ordinal.only = FALSE,
  continuous.only = FALSE,
  summaryObject = FALSE
)
```

Arguments

df	A data.frame() or mxData() to look in for ordinal variables (if you offer a matrix or vector, it will be upgraded to a dataframe)
names	whether to return the names of ordinal variables, or a binary (T,F) list (default = FALSE)
strict	whether to stop when unordered factors are found (default = TRUE)
binary.only	only count binary factors (2-levels) (default = FALSE)
ordinal.only	only count ordinal factors (3 or more levels) (default = FALSE)
continuous.only	use with names = TRUE to get the names of the continuous variables
summaryObject	whether to return a nice summary object. Overrides other settings (FALSE)

Value

- vector of variable names or Booleans

References

- https://www.github.com/tbates/umx

See Also

Other Check or test: umx_check_names(), umx_is_class(), umx_is_endogenous(), umx_is_exogenous(), umx_is_numeric(), umx

Examples

```
x = data.frame(ordered(rbinom(100,1,.5))); names(x) = c("x")
umx_is_ordered(x, summaryObject= TRUE) # all ordered factors including binary
tmp = mtcars

tmp$cyl = ordered(mtcars$cyl) # ordered factor
tmp$vs = ordered(mtcars$vs) # binary factor
umx_is_ordered(tmp) # true/false
umx_is_ordered(tmp, strict=FALSE)
umx_is_ordered(tmp, names = TRUE)
umx_is_ordered(tmp, names = TRUE, binary.only = TRUE)
umx_is_ordered(tmp, names = TRUE, ordinal.only = TRUE)
umx_is_ordered(tmp, names = TRUE, continuous.only = TRUE)
umx_is_ordered(tmp, continuous.only = TRUE)

x = umx_is_ordered(tmp, summaryObject= TRUE)

isContinuous = !umx_is_ordered(tmp)
## Not run:
# nb: By default, unordered factors cause a message...
tmp$gear = factor(mtcars$gear) # Unordered factor
umx_is_ordered(tmp)
umx_is_ordered(tmp, strict = FALSE) # compare: no warning

# also: not designed to work on single variables...
umx_is_ordered(tmp$cyl)
# Do this instead...
umx_is_ordered(tmp[, "cyl", drop= FALSE])

## End(Not run)
```

---

umx_is_RAM *umx_is_RAM*

Description

Utility function returning a binary answer to the question "Is this a RAM model?"

Usage

```
umx_is_RAM(obj)
```

Arguments

obj	an object to be tested to see if it is an OpenMx RAM mxModel()

Value

- Boolean

References

- https://www.github.com/tbates/umx

See Also

Other Test: umx_check_OS(), umx_check_model(), umx_check_names(), umx_check_parallel(), umx_check(), umx_has_CIs(), umx_has_been_run(), umx_has_means(), umx_has_square_brackets(), umx_is_MxData(), umx_is_MxMatrix(), umx_is_MxModel(), umx_is_cov()

Examples

```
require(umx)
data(demoOneFactor)
manifests = names(demoOneFactor)
m1 = umxRAM("is_RAM_ex", data = demoOneFactor, type = "cov",
umxPath("G", to = manifests),
umxPath(var = manifests),
umxPath(var = "G", fixedAt = 1)
)

if(umx_is_RAM(m1)){
message("nice RAM model!")
}
if(!umx_is_RAM(m1)){
message("model needs to be a RAM model")
}
```

---

umx_long2wide *Take a long twin-data file and make it wide (one family per row)*

Description

umx_long2wide merges on famID. Family members are ordered by twinID.

twinID is equivalent to birth order. Up to 10 twinIDs are allowed (family order).

Note: Not all data sets have an order column, but it is essential to rank subjects correctly.

You might start off with a TWID which is a concatenation of a familyID and a 2 digit twinID

Generating famID and twinID as used by this function

You can capture the last 2 digits with the mod function: twinID = df$TWID %% 100

You can *drop* the last 2 digits with integer div: famID = df$TWID %/% 100

Note: The functions assumes that if zygosity or any passalong variables are NA in the first family member, they are NA everywhere. i.e., it does not hunt for values that are present elsewhere to try and self-heal missing data.

Usage

```
umx_long2wide(
  data,
  famID = NA,
  twinID = NA,
  zygosity = NA,
  vars2keep = NA,
  passalong = NA,
  twinIDs2keep = NA
)
```

Arguments

data	The original (long-format) data file
famID	The unique identifier for members of a family
twinID	The twinID. Typically 1, 2, 50 51, etc...
zygosity	Typically MZFF, DZFF MZMM, DZMM DZOS
vars2keep	= The variables you wish to analyse (these will be renamed with paste0("_T", twinID)
passalong	= Variables you wish to pass-through (keep, even though not twin vars)
twinIDs2keep	= If NA (the default) all twinIDs are kept, else only those listed here. Useful to drop sibs.

Value

- dataframe in wide format

References

- https://github.com/tbates/umx, https://tbates.github.io

See Also

- merge()

Other Twin Data functions: umx_make_TwinData(), umx_make_twin_data_nice(), umx_residualize(), umx_scale_wide_twin_data(), umx_wide2long(), umx

Examples

```
## Not run:
# ================================================
# = First make a long format file for the demo =
# ================================================
data(twinData)
tmp = twinData[, -2]
tmp$twinID1 = 1; tmp$twinID2 = 2
long = umx_wide2long(data = tmp, sep = "")
str(long)
```

```
# 'data.frame': 7616 obs. of  11 variables:
#  $ fam     : int  1 2 3 4 5 6 7 8 9 10 ...
#  $ zyg     : int  1 1 1 1 1 1 1 1 1 1 ...
#  $ part    : int  2 2 2 2 2 2 2 2 2 2 ...
#  $ cohort  : chr  "younger" "younger" "younger" "younger" ...
#  $ zygosity: Factor w/ 5 levels "MZFF","MZMM",..: 1 1 1 1 1 1 1 1 1 1 ...
#  $ wt      : int  58 54 55 66 50 60 65 40 60 76 ...
#  $ ht      : num  1.7 1.63 1.65 1.57 1.61 ...
#  $ htwt    : num  20.1 20.3 20.2 26.8 19.3 ...
#  $ bmi     : num  21 21.1 21 23 20.7 ...
#  $ age     : int  21 24 21 21 19 26 23 29 24 28 ...
#  $ twinID  : num  1 1 1 1 1 1 1 1 1 1 ...

# OK. Now to demo long2wide...

# Keeping all columns
wide = umx_long2wide(data= long, famID= "fam", twinID= "twinID", zygosity= "zygosity")
namez(wide) # some vars, like part, should have been passed along instead of made into "part_T1"

# =====================================
# = Demo requesting specific vars2keep =
# =====================================

# Just keep bmi and wt
wide = umx_long2wide(data= long, famID= "fam", twinID= "twinID",
    zygosity = "zygosity", vars2keep = c("bmi", "wt")
)

namez(wide)
# "fam" "twinID" "zygosity" "bmi_T1" "wt_T1" "bmi_T2" "wt_T2"

# ==================
# = Demo passalong =
# ==================
# Keep bmi and wt, and pass through 'cohort'
wide = umx_long2wide(data= long, famID= "fam", twinID= "twinID", zygosity= "zygosity",
vars2keep = c("bmi", "wt"), passalong = "cohort"
)
namez(wide)


## End(Not run)
```

---

umx_lower2full *Convert lower-only matrix data to full (or enforce symmetry on a full matrix)*

Description

Takes a vector of the lower-triangle of cells in a matrix as you might read-in from a journal article), OR a matrix (for instance from a "lower" [mxMatrix()], and returns a full matrix, copying the lower triangle into the upper.

Usage

```
umx_lower2full(lower.data, diag = NULL, byrow = TRUE, dimnames = NULL)
```

Arguments

lower.data	An [mxMatrix()]
diag	A boolean specifying whether the lower.data includes the diagonal
byrow	Whether the matrix is to be filled by row or by column (default = TRUE)
dimnames	Optional dimnames for the matrix (defaults to NULL)

Details

note: Can also take lower data presented in the form of a data.frame. Note also, if presented with a full matrix, the function will return a matrix with symmetry enforced. Can be handy when you have a "nearly-symmetrical" matrix (with differences in the tenth decimal place).

Value

- [mxMatrix()]

References

- <https://www.github.com/tbates/umx>

See Also

Other Data Functions: umxFactor(), umxHetCor(), umx_as_numeric(), umx_cont_2_quantiles(), umx_make_MR_data(), umx_make_TwinData(), umx_make_fake_data(), umx_make_raw_from_cov(), umx_polychoric(), umx_polypairwise(), umx_polytriowise(), umx_read_lower(), umx_rename(), umx_reorder(), umx_select_valid(), umx_stack(), umx

Examples

```
# 1. Test with a vector in byrow = TRUE order)
tmp = c(
1.0000,
0.6247, 1.0000,
0.3269, 0.3669, 1.0000,
0.4216, 0.3275, 0.6404, 1.0000,
0.2137, 0.2742, 0.1124, 0.0839, 1.0000,
0.4105, 0.4043, 0.2903, 0.2598, 0.1839, 1.0000,
0.3240, 0.4047, 0.3054, 0.2786, 0.0489, 0.2220, 1.0000,
```

```
0.2930, 0.2407, 0.4105, 0.3607, 0.0186, 0.1861, 0.2707,  1.0000,
0.2995, 0.2863, 0.5191, 0.5007, 0.0782, 0.3355, 0.2302,  0.2950, 1.0000,
0.0760, 0.0702, 0.2784, 0.1988, 0.1147, 0.1021, 0.0931, -0.0438, 0.2087, 1.000
)
x = umx_lower2full(tmp, diag = TRUE)
# check
isSymmetric(x)

# 2. Test with matrix input
tmpn = c("ROccAsp", "REdAsp", "FOccAsp", "FEdAsp", "RParAsp",
         "RIQ", "RSES", "FSES", "FIQ", "FParAsp")
tmp = matrix(nrow = 10, ncol = 10, byrow = TRUE, dimnames = list(tmpn,tmpn), data =
c(1.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,  0.0000, 0.0000, 0,
0.6247, 1.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,  0.0000, 0.0000, 0,
0.3269, 0.3669, 1.0000, 0.0000, 0.0000, 0.0000, 0.0000,  0.0000, 0.0000, 0,
0.4216, 0.3275, 0.6404, 1.0000, 0.0000, 0.0000, 0.0000,  0.0000, 0.0000, 0,
0.2137, 0.2742, 0.1124, 0.0839, 1.0000, 0.0000, 0.0000,  0.0000, 0.0000, 0,
0.4105, 0.4043, 0.2903, 0.2598, 0.1839, 1.0000, 0.0000,  0.0000, 0.0000, 0,
0.3240, 0.4047, 0.3054, 0.2786, 0.0489, 0.2220, 1.0000,  0.0000, 0.0000, 0,
0.2930, 0.2407, 0.4105, 0.3607, 0.0186, 0.1861, 0.2707,  1.0000, 0.0000, 0,
0.2995, 0.2863, 0.5191, 0.5007, 0.0782, 0.3355, 0.2302,  0.2950, 1.0000, 0,
0.0760, 0.0702, 0.2784, 0.1988, 0.1147, 0.1021, 0.0931, -0.0438, 0.2087, 1)
)
x = umx_lower2full(tmp, diag= TRUE)
isSymmetric(x)

# 3. Test with lower-vector, no diagonal.
tmp = c(
0.6247,
0.3269, 0.3669,
0.4216, 0.3275, 0.6404,
0.2137, 0.2742, 0.1124, 0.0839,
0.4105, 0.4043, 0.2903, 0.2598, 0.1839,
0.3240, 0.4047, 0.3054, 0.2786, 0.0489, 0.2220,
0.2930, 0.2407, 0.4105, 0.3607, 0.0186, 0.1861, 0.2707,
0.2995, 0.2863, 0.5191, 0.5007, 0.0782, 0.3355, 0.2302,  0.2950,
0.0760, 0.0702, 0.2784, 0.1988, 0.1147, 0.1021, 0.0931, -0.0438, 0.2087
)
umx_lower2full(tmp, diag = FALSE)

# An example with byrow = FALSE

ldiag = c(
1, -.17, -.22, -.19, -.12, .81, -.02, -.26, -.2, -.15,
1, .11, .2, .21, -.01, .7, .1, .7, .1, .17, .22,
1, .52, .68, -.12, .09, .49, .27, .46,
1, .5, -.06, .17, .26, .80, .31,
1, -.1, .19, .36, .23, .42,
1, .02, -19, -.06, -.06,
1, .1, .18, .27,
1, .51, .7,
1, .55,
1)
```

```
umx_lower2full(tmp, byrow = FALSE, diag = TRUE)
```

---

umx_make *"make" the umx package using devtools: release to CRAN etc.*

Description

Easily run devtools "install", "release", "win", "examples" etc.

Usage

```
umx_make(
 what = c("quick_install", "install_full", "spell", "run_examples", "check", "win",
    "rhub", "release", "travisCI", "sitrep"),
  pkg = "~/bin/umx",
  check = TRUE,
  run = FALSE,
  start = NULL,
  spelling = "en_US"
)
```

Arguments

what	whether to "install", "release" to CRAN, check on "win", "check", "rhub", "spell" check, or check "examples"))
pkg	the local path to your package. Defaults to my path to umx.
check	Whether to run check on the package before release (default = TRUE).
run	= If what is "examples", whether to also run examples marked don't run. (default FALSE)
start	If what is "examples", which function to start from (default (NULL) = beginning).
spelling	Whether to check spelling before release (default = "en_US": set NULL to not check).

Value

None

References

- https://github.com/tbates/umx, https://tbates.github.io

See Also

Other xmu internal not for end user: umxModel(), umxRenameMatrix(), umxTwinMaker(), umx_APA_pval(), umx_fun_mean_sd(), umx_get_bracket_addresses(), umx_standardize(), umx_string_to_algebra(), umx, xmuHasSquareBrackets(), xmuLabel_MATRIX_Model(), xmuLabel_Matrix(), xmuLabel_RAM_Model(), xmuMI(), xmuMakeDeviationThresholdsMatrices(), xmuMakeOneHeadedPathsFromPathList(), xmuMakeTwoHeadedPathsFromPathList(), xmuMaxLevels(), xmuMinLevels(), xmuPropagateLabels(), xmuRAM2Ordinal(), xmuTwinSuper_Continuous(), xmuTwinUpgradeMeansToCovariateModel(), xmu_CI_merge(), xmu_CI_stash(), xmu_DF_to_mxData_TypeCov(), xmu_PadAndPruneForDefVars(), xmu_cell_is_on(), xmu_check_levels_identical(), xmu_check_needs_means(), xmu_check_variance(), xmu_clean_label(), xmu_data_missing(), xmu_data_swap_a_block(), xmu_describe_data_WLS(), xmu_dot_make_paths(), xmu_dot_make_residuals(), xmu_dot_maker(), xmu_dot_move_ranks(), xmu_dot_rank_str(), xmu_extract_column(), xmu_get_CI(), xmu_lavaan_process_group(), xmu_make_TwinSuperModel(), xmu_make_bin_cont_pair_data(), xmu_make_mxData(), xmu_match.arg(), xmu_name_from_lavaan_str(), xmu_path2twin(), xmu_path_regex(), xmu_safe_run_summary(), xmu_set_sep_from_suffix(), xmu_show_fit_or_comparison(), xmu_simplex_corner(), xmu_standardize_ACEcov(), xmu_standardize_ACEv(), xmu_standardize_ACE(), xmu_standardize_CP(), xmu_standardize_IP(), xmu_standardize_RAM(), xmu_standardize_SexLim(), xmu_standardize_Simplex(), xmu_start_value_list(), xmu_starts(), xmu_twin_add_WeightMatrices(), xmu_twin_check(), xmu_twin_get_var_names(), xmu_twin_upgrade_selDvs2SelVars()

Examples

```
## Not run:
umx_make(what = "q"))        # Quick install
umx_make(what = "install"))  # Just installs the package
umx_make(what = "examples")) # Run the examples
umx_make(what = "spell"))    # Spell check the documents
umx_make(what = "check"))    # Run R CMD check
umx_make(what = "win"))      # Check on win-builder
umx_make(what = "release"))  # Release to CRAN

## End(Not run)
```

---

umx_make_fake_data *umx_make_fake_data*

Description

This function takes as argument an existing dataset, which must be either a matrix or a data frame. Each column of the dataset must consist either of numeric variables or ordered factors. When one or more ordered factors are included, then a heterogeneous correlation matrix is computed using John Fox's polycor package. Pairwise complete observations are used for all covariances, and the exact pattern of missing data present in the input is placed in the output, provided a new sample size is not requested. Warnings from the polycor::hetcor function are suppressed.

Usage

```
umx_make_fake_data(
  dataset,
  digits = 2,
  n = NA,
  use.names = TRUE,
  use.levels = TRUE,
  use.miss = TRUE,
  mvt.method = "eigen",
  het.ML = FALSE,
  het.suppress = TRUE
)
```

Arguments

dataset	The original dataset of which to make a simulacrum
digits	= Round the data to the requested digits (default = 2)
n	Number of rows to generate (NA = all rows in dataset)
use.names	Whether to name the variables (default = TRUE)
use.levels	= Whether to use existing levels (default = TRUE)
use.miss	Whether to have data missing as in original (defaults to TRUE)
mvt.method	= Passed to hetcor (default = "eigen")
het.ML	= Passed to hetcor (default = FALSE)
het.suppress	Passed to hetcor (default = TRUE)

Value

- new dataframe

See Also

[OpenMx::mxGenerateData()]

Other Data Functions: umxFactor(), umxHetCor(), umx_as_numeric(), umx_cont_2_quantiles(), umx_lower2full(), umx_make_MR_data(), umx_make_TwinData(), umx_make_raw_from_cov(), umx_polychoric(), umx_polypairwise(), umx_polytriowise(), umx_read_lower(), umx_rename(), umx_reorder(), umx_select_valid(), umx_stack(), umx

Examples

```
fakeCars = umx_make_fake_data(mtcars)
```

umx_make_MR_data *Simulate Mendelian Randomization data*

Description

umx_make_MR_data returns a dataset containing 4 variables: A variable of interest (Y), a putative cause (X), a qtl (quantitative trait locus) influencing X, and a confounding variable (U) affecting both X and Y.

Usage

```
umx_make_MR_data(
  nSubjects = 1000,
  Vqtl = 0.02,
  bXY = 0.1,
  bUX = 0.5,
  bUY = 0.5,
  pQTL = 0.5,
  seed = 123
)
```

Arguments

nSubjects	Number of subjects in sample
Vqtl	Variance of QTL affecting causal variable X (Default 0.02)
bXY	Causal effect of X on Y (Default 0.1)
bUX	Confounding effect of confounder 'U' on X (Default 0.5)
bUY	Confounding effect of confounder 'U' on Y (Default 0.5)
pQTL	Decreaser allele frequency (Default 0.5)
seed	value for the random number generator (Default 123)

Details

The code to make these Data. Modified from Dave Evans 2016 Boulder workshop talk.

Value

- data.frame

See Also

umx_make_TwinData

Other Data Functions: umxFactor(), umxHetCor(), umx_as_numeric(), umx_cont_2_quantiles(), umx_lower2full(), umx_make_TwinData(), umx_make_fake_data(), umx_make_raw_from_cov(), umx_polychoric(), umx_polypairwise(), umx_polytriowise(), umx_read_lower(), umx_rename(), umx_reorder(), umx_select_valid(), umx_stack(), umx

Examples

```
df = umx_make_MR_data(10000)
str(df)
## Not run:
m1 = umxTwoStage(Y ~ X, ~qtl, data = df)
plot(m1)

## End(Not run)
```

---

umx_make_raw_from_cov *Turn a cov matrix into raw data*

Description

A wrapper for MASS::mvrnorm() to simplify turning a covariance matrix into matching raw data.

Usage

```
umx_make_raw_from_cov(covMat, n, means = 0, varNames = NULL, empirical = FALSE)
```

Arguments

covMat	A covariance matrix
n	How many rows of data to return
means	the means of the raw data (default = 0)
varNames	default uses "var1", "var2"
empirical	(passed to mvrnorm) Default = FALSE

Value

- data.frame

References

- https://tbates.github.io, https://github.com/tbates/umx

See Also

- cov2cor(), MASS::mvrnorm()

Other Data Functions: umxFactor(), umxHetCor(), umx_as_numeric(), umx_cont_2_quantiles(), umx_lower2full(), umx_make_MR_data(), umx_make_TwinData(), umx_make_fake_data(), umx_polychoric(), umx_polypairwise(), umx_polytriowise(), umx_read_lower(), umx_rename(), umx_reorder(), umx_select_valid(), umx_stack(), umx

Examples

```
covData <- matrix(nrow=6, ncol=6, byrow=TRUE, dimnames=list(paste0("v", 1:6), paste0("v", 1:6)),
  data = c(0.9223099, 0.1862938, 0.4374359, 0.8959973, 0.9928430, 0.5320662,
           0.1862938, 0.2889364, 0.3927790, 0.3321639, 0.3371594, 0.4476898,
           0.4374359, 0.3927790, 1.0069552, 0.6918755, 0.7482155, 0.9013952,
           0.8959973, 0.3321639, 0.6918755, 1.8059956, 1.6142005, 0.8040448,
           0.9928430, 0.3371594, 0.7482155, 1.6142005, 1.9223567, 0.8777786,
           0.5320662, 0.4476898, 0.9013952, 0.8040448, 0.8777786, 1.3997558)
)

myData = umx_make_raw_from_cov(covData, n = 100, means = 1:6)
umxAPA(myData)
covMat = matrix(c(1, .3, .3, 1), nrow=2)
tmp= umx_make_raw_from_cov(covMat, n=10, varNames= c("x", "y"))
cov(tmp)
tmp= umx_make_raw_from_cov(covMat, n=10, varNames= c("x", "y"), empirical= TRUE)
cov(tmp)
tmp= umx_make_raw_from_cov(qm(1, .3| .3, 1), n=10, varNames= c("x", "y"))
cov(tmp)
```

`umx_make_sql_from_excel` *Convert an excel spreadsheet in a text file on sql statements.*

Description

Unlikely to be of use to anyone but the package author :-)

Usage

```
umx_make_sql_from_excel(theFile = "Finder")
```

Arguments

`theFile`	The xlsx file to read. Default = "Finder")

Details

On OS X, by default, the file selected in the front-most Finder window will be chosen. If it is blank, a choose file dialog will be thrown.

Read an xlsx file and convert into SQL insert statements (placed on the clipboard) On MacOS, the function can access the current front-most Finder window.

The file name should be the name of the test. Columns should be headed: itemText direction scale type [optional response options]

The SQL fields generated are: itemID, test, native_item_number, item_text, direction, scale, format, author

tabbedPlus: list scored from 0 to n-1

tabbedVertPlus: tabbed, but vertical lay-out

number 2+2\<itemBreak\>min='0' max='7' step='1'

5fm Scored 1-5, anchored: Strongly Disagree | Disagree | Neutral | Agree | Strongly Agree

intro (not) scored, and sequenced as item 0

Value

None

References

- https://www.github.com/tbates/umx

See Also

Other File Functions: dl_from_dropbox(), umx_file_load_pseudo(), umx_move_file(), umx_open(), umx_rename_file(), umx_write_to_clipboard(), umx

Examples

```
## Not run:
# An example Excel spreadsheet
# local uncompiled path
fp = system.file("inst/extdata", "GQ6.sql.xlsx", package = "umx")
# installed path
fp = system.file("extdata", "GQ6.sql.xlsx", package = "umx")
umx_open(fp)
umx_make_sql_from_excel() # Using file selected in front-most Finder window
umx_make_sql_from_excel("~/Desktop/test.xlsx") # provide a path

## End(Not run)
```

---

umx_make_TwinData *Simulate twin data with control over A, C, and E parameters, as well as moderation of A.*

Description

Makes MZ and DZ twin data, optionally with moderated A. By default, the three variance components must sum to 1.

See examples for how to use this: it is pretty flexible.

If you provide 2 varNames, they will be used for twin 1 and twin 2. If you provide one, it will be expanded to var_T1 and var_T2

You supply the number of pairs of each zygosity that wish to simulate (nMZpairs, nDZpairs), along with the values of AA, CC,and EE.

Note, if you want a power calculator, see mxPower().

Shortcuts

You can omit nDZpairs. You can also give any two of A, C, or E and the function produce the missing parameter that makes `A+C+E == 1`.

Moderation

Univariate GxE Data To simulate data for `umxGxE`, offer up a list of the average, min and max values for `AA`, i.e., c(avg = .5, min = 0, max = 1).

`umx_make_TwinData` will then return moderated heritability, with average value = avg, and swinging down to min and up to max across 3-SDs of the moderator.

Bivariate GxE Data

To simulate data with a moderator that is not shared by both twins. Moderated heritability is specified via the bivariate relationship (AA, CC, EE) and two moderators in each component. AA = list(a11 = .4, a12 = .1, a22 = .15) CC = list(c11 = .2, c12 = .1, c22 = .10) EE = list(e11 = .4, e12 = .3, e22 = .25) Amod = list(Beta_a1 = .025, Beta_a2 = .025) Cmod = list(Beta_c1 = .025, Beta_c2 = .025) Emod = list(Beta_e1 = .025, Beta_e2 = .025)

Usage

```
umx_make_TwinData(
  nMZpairs,
  nDZpairs = nMZpairs,
  AA = NULL,
  CC = NULL,
  EE = NULL,
  DD = NULL,
  varNames = "var",
  MZr = NULL,
  DZr = MZr,
  dzAr = 0.5,
  scale = FALSE,
  mean = 0,
  sd = 1,
  nThresh = NULL,
  sum2one = TRUE,
  bivAmod = NULL,
  bivCmod = NULL,
  bivEmod = NULL,
  seed = NULL,
  empirical = FALSE
)
```

Arguments

`nMZpairs`	Number of MZ pairs to simulate
`nDZpairs`	Number of DZ pairs to simulate (defaults to nMZpairs)
`AA`	value for A variance. NOTE: See options for use in GxE and Bivariate GxE
`CC`	value for C variance.

EE	value for E variance.
DD	value for E variance.
varNames	name for variables (defaults to 'var')
MZr	If MZr and DZr are set (default = NULL), the function returns dataframes of the request n and correlation.
DZr	Set to return dataframe using MZr and Dzr (Default NULL)
dzAr	DZ Ar (default .5)
scale	Whether to scale output to var=1 mean=0 (Default FALSE)
mean	mean for traits (default = 0) (not applied to moderated cases)
sd	sd of traits (default = 1) (not applied to moderated cases)
nThresh	If supplied, use as thresholds and return mxFactor output? (default is not to)
sum2one	Whether to enforce AA + CC + EE summing the one (default = TRUE)
bivAmod	Used for Bivariate GxE data: list(Beta_a1 = .025, Beta_a2 = .025)
bivCmod	Used for Bivariate GxE data: list(Beta_c1 = .025, Beta_c2 = .025)
bivEmod	Used for Bivariate GxE data: list(Beta_e1 = .025, Beta_e2 = .025)
seed	Allows user to set.seed() if wanting reproducible dataset
empirical	Passed to mvrnorm

Value

- list of mzData and dzData dataframes containing T1 and T2 plus, if needed M1 and M2 (moderator values)

References

- https://github.com/tbates/umx, https://tbates.github.io

See Also

- umxACE(), umxGxE(), umxGxEbiv()

Other Twin Data functions: umx_long2wide(), umx_make_twin_data_nice(), umx_residualize(), umx_scale_wide_twin_data(), umx_wide2long(), umx

Other Data Functions: umxFactor(), umxHetCor(), umx_as_numeric(), umx_cont_2_quantiles(), umx_lower2full(), umx_make_MR_data(), umx_make_fake_data(), umx_make_raw_from_cov(), umx_polychoric(), umx_polypairwise(), umx_polytriowise(), umx_read_lower(), umx_rename(), umx_reorder(), umx_select_valid(), umx_stack(), umx

Examples

```
# ======================================================================
# = Basic Example, with all elements of std univariate data specified =
# ======================================================================
tmp = umx_make_TwinData(nMZpairs = 10000, AA = .30, CC = .00, EE = .70)
# Show dataframe with 20,000 rows and 3 variables: var_T1, var_T2, and zygosity
str(tmp)
```

```
# ====================================
# = How to consume the built datasets =
# ====================================
mzData = tmp[tmp$zygosity == "MZ", ]
dzData = tmp[tmp$zygosity == "DZ", ]
str(mzData); str(dzData);
cov(mzData[, c("var_T1","var_T2")])
cov(dzData[, c("var_T1","var_T2")])
umxAPA(mzData[,c("var_T1","var_T2")])

# Prefer to work in path coefficient values? (little a?)
tmp = umx_make_TwinData(2000, AA = .7^2, CC = .0)
mzData = tmp[tmp$zygosity == "MZ", ]
dzData = tmp[tmp$zygosity == "DZ", ]
m1 = umxACE(selDVs="var", sep="_T", mzData= mzData, dzData= dzData)

# Examine correlations
cor(mzData[,c("var_T1","var_T2")])
cor(dzData[,c("var_T1","var_T2")])

# Example with D (left un-modeled in ACE)
tmp = umx_make_TwinData(nMZpairs = 500, AA = .4, DD = .2, CC = .2)
m1 = umxACE(selDVs="var", data = tmp, mzData= "MZ", dzData= "DZ")
# |    |   a1|   c1|   e1|
# |:---|----:|----:|----:|
# |var | 0.86| 0.24| 0.45|

m1 = umxACE(selDVs="var", data = tmp, mzData= "MZ", dzData= "DZ", dzCr=.25)
# |    |  a1|d1 |   e1|
# |:---|---:|:--|----:|
# |var | 0.9|.  | 0.44|


# =============
# = Shortcuts =
# =============

# Omit nDZpairs (equal numbers of both by default)
tmp = umx_make_TwinData(nMZpairs = 100, AA = 0.5, CC = 0.3) # omit any one of A, C, or E (sums to 1)
cov(tmp[tmp$zygosity == "DZ", c("var_T1","var_T2")])

# Not limited to unit variance
tmp = umx_make_TwinData(100, AA = 3, CC = 2, EE = 3, sum2one = FALSE)
cov(tmp[tmp$zygosity == "MZ", c("var_T1","var_T2")])

# Output can be scaled
tmp = umx_make_TwinData(100, AA = .7, CC = .1, scale = TRUE)
cov(tmp[tmp$zygosity == "MZ", c("var_T1","var_T2")])

## Not run:

# ===============
```

```
# = GxE Example =
# ===============

AA = c(avg = .5, min = .1, max = .8)
tmp = umx_make_TwinData(nMZpairs = 140, nDZpairs = 240, AA = AA, CC = .35, EE = .65, scale= TRUE)
mzData = tmp[tmp$zygosity == "MZ", ]
dzData = tmp[tmp$zygosity == "DZ", ]
m1 = umxGxE(selDVs = "var", selDefs = "M", sep = "_T", mzData = mzData, dzData = dzData)

# =====================
# = Threshold Example =
# =====================
tmp = umx_make_TwinData(100, AA = .6, CC = .2, nThresh = 3)
str(tmp)
umx_polychoric(subset(tmp, zygosity=="MZ", c("var_T1", "var_T2")))$polychorics
# Running model with 7 parameters
#           var_T1    var_T2
# var_T1 1.0000000 0.7435457
# var_T2 0.7435457 1.0000000


# ========================
# = Just use MZr and DZr =
# ========================
tmp = umx_make_TwinData(100, MZr = .86, DZr = .60, varNames = "IQ")
umxAPA(subset(tmp, zygosity == "MZ", c("IQ_T1", "IQ_T2")))
umxAPA(subset(tmp, zygosity == "DZ", c("IQ_T1", "IQ_T2")))
m1 = umxACE(selDVs= "IQ", data = tmp)
# TODO tmx_ examples of unmodeled D etc.

# Bivariate GxSES example (see umxGxEbiv)

AA   = list(a11 = .4, a12 = .1, a22 = .15)
CC   = list(c11 = .2, c12 = .1, c22 = .10)
EE   = list(e11 = .4, e12 = .3, e22 = .25)
Amod = list(Beta_a1 = .025, Beta_a2 = .025)
Cmod = list(Beta_c1 = .025, Beta_c2 = .025)
Emod = list(Beta_e1 = .025, Beta_e2 = .025)
tmp = umx_make_TwinData(5000, AA =AA, CC = CC, EE = EE,
bivAmod = Amod, bivCmod =Cmod, bivEmod =Emod)
str(tmp)
# 'data.frame': 10000 obs. of  7 variables:
#  $ defM_T1 : num  0.171 0.293 -0.173 0.238 -0.73 ...
#  $ defM_T2 : num  0.492 -0.405 -0.696 -0.829 -0.858 ...
#  $ M_T1    : num  0.171 0.293 -0.173 0.238 -0.73 ...
#  $ var_T1  : num  0.011 0.1045 0.5861 0.0583 1.0225 ...
#  $ M_T2    : num  0.492 -0.405 -0.696 -0.829 -0.858 ...
#  $ var_T2  : num  -0.502 -0.856 -0.154 0.065 -0.268 ...
#  $ zygosity: Factor w/ 2 levels "MZ","DZ": 1 1 1 1 1 1 1 1 1 1 ...

# TODO tmx example showing how moderation of A introduces heteroscedasticity in a regression model:
# More residual variance at one extreme of the x axis (moderator)
# m1 = lm(var_T1~ M_T1, data = x);
```

```
# x = rbind(tmp[[1]], tmp[[2]])
# plot(residuals(m1)~ x$M_T1, data=x)

## End(Not run)
```

`umx_make_twin_data_nice` *Convert a twin dataset into umx standard format.*

Description

`umx_make_twin_data_nice` is a function to convert your twin data into a format used across `umx`. Specifically:

1. Existing column for zygosity is renamed to "zygosity".
2. `sep` is set to "_T"
3. The twinID is is set to sequential digits, i.e. 1,2...

Usage

```
umx_make_twin_data_nice(
  data,
  sep,
  zygosity,
  numbering,
  labelNumericZygosity = FALSE,
  levels = 1:5,
  labels = c("MZFF", "MZMM", "DZFF", "DZMM", "DZOS")
)
```

Arguments

`data`	a `data.frame()` to check/convert.
`sep`	existing separator string (will be updated to "_T").
`zygosity`	existing zygosity column name (will be renamed `zygosity`).
`numbering`	existing twin sequence string (will be updated to _T1, _T2, _T3).
`labelNumericZygosity`	If TRUE numeric zygosity levels will be set to labels.
`levels`	legal levels of zygosity (ignored if labelNumericZygosity = FALSE (default 1:5)
`labels`	labels for each zyg level c("MZFF", "MZMM", "DZFF", "DZMM", "DZOS").

Value

- `data.frame()`

References

- tutorials, tbates/umx

See Also

- umx_wide2long(), umx_long2wide(),

Other Twin Data functions: umx_long2wide(), umx_make_TwinData(), umx_residualize(), umx_scale_wide_twin_data(), umx_wide2long(), umx

Examples

```
data(twinData)
tmp = twinData
tmp$zygosity=NULL
tmp = umx_make_twin_data_nice(twinData, sep="", numbering = 1:5, zyg="zygosity")
namez(tmp, "zyg")
levels(tmp$zygosity)
```

---

umx_means *umx_means*

Description

Helper to get means from a df that might contain ordered or string data. Factor means are set to "ordVar"

Usage

```
umx_means(df, ordVar = 0, na.rm = TRUE)
```

Arguments

df	a dataframe of raw data from which to get variances.
ordVar	value to return for the means of factor data = 0
na.rm	passed to mean - defaults to "na.rm"

Value

- frame of means

See Also

Other Miscellaneous Stats Helpers: FishersMethod(), SE_from_p(), oddsratio(), reliability(), umxCov2cor(), umxHetCor(), umxWeightedAIC(), umx_apply(), umx_cor(), umx_r_test(), umx_round(), umx_scale(), umx_var(), umx

Examples

```
tmp = mtcars[,1:4]
tmp$cyl = ordered(mtcars$cyl) # ordered factor
tmp$hp  = ordered(mtcars$hp)  # binary factor
umx_means(tmp, ordVar = 0, na.rm = TRUE)
```

---

umx_move_file *Move files*

Description

On OS X, umx_move_file can access the current front-most Finder window. The file moves are fast and, because you can use regular expressions, powerful.

Usage

```
umx_move_file(
  baseFolder = NA,
  regex = NULL,
  fileNameList = NA,
  destFolder = NA,
  test = TRUE,
  overwrite = FALSE
)
```

Arguments

baseFolder	The folder to search in. If set to "Finder" (and you are on OS X) it will use the current front-most Finder window. If it is blank, a choose folder dialog will be thrown.
regex	match string choosing which files are selected in baseFolder
fileNameList	List of files to move
destFolder	Folder to move files into
test	Boolean determining whether to change the names, or just report on what would have happened
overwrite	Boolean determining whether to overwrite files or not (default = FALSE (safe))

Value

None

See Also

file.rename(), regex()

Other File Functions: dl_from_dropbox(), umx_file_load_pseudo(), umx_make_sql_from_excel(), umx_open(), umx_rename_file(), umx_write_to_clipboard(), umx

Examples

```
## Not run:
base = "~/Desktop/"
dest = "~/Music/iTunes/iTunes Music/Music/"
umx_move_file(baseFolder = base, fileNameList = toMove, destFolder = dest, test= TRUE)

# ============================================================
# = Move all files in downloads ending in ".jpeg" to Desktop =
# ============================================================
umx_move_file(baseFolder = "~/Downloads/", regex=".jpeg",
destFolder = "~/Desktop/", test= TRUE)

## End(Not run)
```

---

umx_msg *Print the name and compact contents of variable.*

Description

Helper function to ease debugging with console notes like: "ObjectName = \<Object Value\>". This is primarily useful for inline debugging, where seeing, e.g., "nVar = 3" can be useful. The ability to say `umx_msg(nVar)` makes this easy.

Usage

```
umx_msg(x)
```

Arguments

x	the thing you want to pretty-print

Value

- NULL

References

- https://tbates.github.io, https://github.com/tbates/umx

See Also

Other Miscellaneous Utility Functions: install.OpenMx(), qm(), umxBrownie(), umxLav2RAM(), umxRAM2Lav(), umxVersion(), umx_array_shift(), umx_find_object(), umx_open_CRAN_page(), umx_pad(), umx_print(), umx_score_scale(), umx

Examples

```
a = "brian"
umx_msg(a)
b = c("brian", "sally", "jane")
umx_msg(b)
umx_msg(mtcars)
```

---

umx_names *umx_names*

Description

Convenient equivalent of running grep on names, with value = TRUE and ignore.case = TRUE.

Plus:umx_names can handle dataframes, a model, list of models, model summary, or a vector of strings as input.

In these cases, it will search column names, parameter or summary output names, or the literal string values themselves respectively.

In addition, umx_names can do replacement of a found string (see examples). It can also collapse the result (using paste0)

Note: namez (with a z) is a shortcut for umx_names, which makes it easy to replace where you would otherwise use names.

You can learn more about the matching options (like inverting the selection etc.) in the help for base-R grep.

Usage

```
umx_names(
  df,
  pattern = ".*",
  replacement = NULL,
  ignore.case = TRUE,
  perl = FALSE,
  value = TRUE,
  fixed = FALSE,
  useBytes = FALSE,
  invert = FALSE,
  global = FALSE,
  collapse = c("as.is", "vector", "formula")
)
```

Arguments

df	dataframe (or other objects, or a list of models) from which to get names.
pattern	Used to find only matching names (supports grep/regular expressions)

replacement	If not NULL, replaces the found string. Use backreferences ("\1" to "\9") to refer to (subexpressions).
ignore.case	default = TRUE (opposite default to grep)
perl	Should Perl-compatible regexps be used? Default = FALSE
value	Return matching elements themselves (TRUE) or their indices (FALSE) default = TRUE (opposite default to grep)
fixed	= FALSE (grep option If TRUE, pattern is a string to be matched as is. Overrides all conflicting arguments.)
useBytes	= FALSE logical. grep option. If TRUE, matching is by byte rather than by character.
invert	Return indices or values for elements that do not match (default = FALSE).
global	replace all instances in each strong, or just the first (Default).
collapse	"as.is" leaves alone. as.vector formats as pasteable code, i.e., "c('a', 'b')", not "a" "b" (default NULL), etc.

Value

- vector of matches

References

- https://tbates.github.io, https://github.com/tbates/umx

See Also

- Base-R pattern matching functions: grep(). And umx_check_names() to check for existence of names in a dataframe.

Other Reporting Functions: loadings.MxModel(), umxAPA(), umxFactorScores(), umxGetParameters(), umxParameters(), umxReduce(), umx_aggregate(), umx_time(), umx

Other String Functions: umx_explode_twin_names(), umx_explode(), umx_grep(), umx_paste_names(), umx_rot(), umx_str_chars(), umx_str_from_object(), umx_trim(), umx

Examples

```
# Names from a dataframe, with character matching
umx_names(mtcars, "mpg") # only "mpg" matches this

# Easy-to-type alias "namez"
namez(mtcars, "mpg")

# Use a regular expression to match a pattern
namez(mtcars, "r[ab]") # "drat", "carb"
namez(mtcars, "^d") # vars beginning with 'd' = "disp", drat

# Use this function to replace text in names!
umx_names(mtcars, "mpg", replacement = "hello") # "mpg" replaced with "hello"
```

```
# ========================================================================
# = Using the custom collapse option to quote each item, and wrap in c() =
# ========================================================================
namez(mtcars, "m", collapse = "vector") # Paste-able R-code for a vector

# Other options passed to R's grep command
umx_names(mtcars, "mpg" , invert = TRUE)  # Non-matches (instead of matches)
umx_names(mtcars, "disp", value  = FALSE) # Return indices of matches
umx_names(mtcars, "^d"  , fixed  = TRUE)  # Vars containing literal '^d' (none...)

# ======================================
# = Examples using built-in GFF dataset =
# ======================================

# Just show phenotypes for Twin 1
umx_names(GFF, "_T1$") # twin 1
# "zyg" "sex1" "age_T1" "gff_T1" "fc_T1" "qol_T1" "hap_T1"...

umx_names(GFF, "2$") # names ending in 2
umx_names(GFF, "[^12bs]$") # doesn't end in `1`, `2`, `b`, or `s`
# "zyg_6grp" "zyg_2grp" "divorce"
umx_names(mxData(twinData[, c("wt1", "wt2")], type= "raw"))
umx_names(mxData(cov(twinData[, c("wt1", "wt2")], use="comp"), type= "cov", numObs= 1000))
umx_names(mxDataWLS(na.omit(twinData[, c("wt1", "wt2")]), type= "WLS"))

namez(umxMatrix("bob", "Full", 3,3)$labels)
```

---

umx_open *Open a file or folder*

Description

Open a file or folder. Works on OS X, mostly on windows, and hopefully on unix.

Usage

```
umx_open(filepath = getwd())
```

Arguments

filepath	The file to open

Details

NOTE: Your filepath is shQuote()'d by this function.

Value

None

References

- https://github.com/tbates/umx, https://tbates.github.io

See Also

Other File Functions: dl_from_dropbox(), umx_file_load_pseudo(), umx_make_sql_from_excel(), umx_move_file(), umx_rename_file(), umx_write_to_clipboard(), umx

Examples

```
## Not run:
umx_open() # Default is to open working directory getwd()
umx_open("~/bin/umx/R/misc_and_utility copy.r")

## End(Not run)
```

---

umx_open_CRAN_page *Open the CRAN page for a package*

Description

On MacOS, this function opens the CRAN page for a package. Useful for looking up documentation, checking you have an up-to-date version, showing the package to people etc.

Usage

```
umx_open_CRAN_page(package = "umx")
```

Arguments

package	An R package name.

Value

None

See Also

Other Miscellaneous Utility Functions: install.OpenMx(), qm(), umxBrownie(), umxLav2RAM(), umxRAM2Lav(), umxVersion(), umx_array_shift(), umx_find_object(), umx_msg(), umx_pad(), umx_print(), umx_score_scale(), umx

Examples

```
## Not run:
umx_open_CRAN_page("umx")

## End(Not run)
```

umx_pad *Pad an Object with NAs*

Description

This function pads an R object (list, data.frame, matrix, atomic vector) with NAs. For matrices, lists and data.frames, this occurs by extending each (column) vector in the object.

Usage

```
umx_pad(x, n)
```

Arguments

x	An R object (list, data.frame, matrix, atomic vector).
n	The final length of each object.

Value

- padded object

References

- https://github.com/kevinushey/Kmisc/tree/master/man

See Also

Other Miscellaneous Utility Functions: install.OpenMx(), qm(), umxBrownie(), umxLav2RAM(), umxRAM2Lav(), umxVersion(), umx_array_shift(), umx_find_object(), umx_msg(), umx_open_CRAN_page(), umx_print(), umx_score_scale(), umx

Examples

```
umx_pad(1:3, 4)
umx_pad(1:3, 3)
```

umx_paste_names *Concatenate base variable names with suffixes to create wide-format variable names (i.e twin-format)*

Description

It's easier to work with base names, rather than the twice-as-long hard-to-typo list of column names. umx_paste_names adds suffixes to names so you can work with that nice short list. So, you provide bmi, and you get back fully specified family-wise names: c("bmi_T1","bmi_T2")

note: tvars is a shortcut for umx_paste_names

Usage

```
umx_paste_names(
  varNames,
  sep = "",
  suffixes = 1:2,
  covNames = NULL,
  prefix = NULL
)
```

Arguments

varNames	a list of *base* names, e.g c("bmi", "IQ")
sep	A string separating the name and the twin suffix, e.g. "_T" (default is "")
suffixes	a list of terminal suffixes differentiating the twins default = 1:2)
covNames	a list of *base* names for covariates (to be sorted last in list), e.g c("age", "sex")
prefix	a string to prepend to each label, e.g "mean" -> "mean_age" "mean_sex"

Details

Method 1: *Use complete suffixes*

You can provide complete suffixes like "_T1" and "_T2". This has the benefit of being explicit and very general:

```
umx_paste_names(c("var1", "var2"), suffixes = c("_T1", "_T2"))
```

Note: for quick typing, tvars is an alias for umx_paste_names

Method 2: *Use sep and a suffix vector.*

Alternatively, you can use sep to add a constant like "_T" after each basename, along with a vector of suffixes. This has the benefit of showing what is varying: This is then suffixed with e.g. "1", "2".

```
umx_paste_names(c("var1", "var2"), sep = "_T", suffixes = 1:2)
```

Working with covariates

If you are using umxACEcov(), you **need** to keep all the covariates at the end of the list. Here's how:

```
umx_paste_names(c("var1", "var2"), cov = c("cov1"), sep = "_T", suffixes = 1:2)
```

note: in conventional twin models, the expCov matrix is T1 vars, followed by T2 vars. For covariates, you want T1vars, T2 vars, T1 covs, T2 covs. This is what covNames accomplishes.

Value

- vector of suffixed var names, i.e., c("v1_T1", "v2_T1", "v1_T2", "v2_T2", "cov_T1", "cov_T2")

References

- https://tbates.github.io, https://github.com/tbates/umx

See Also

namez() umx_explode_twin_names()

Other String Functions: umx_explode_twin_names(), umx_explode(), umx_grep(), umx_names(), umx_rot(), umx_str_chars(), umx_str_from_object(), umx_trim(), umx

Examples

```
# two styles doing the same thing: first is more general
umx_paste_names("bmi", suffixes = c("_T1", "_T2"))
umx_paste_names("bmi", sep = "_T", suffixes = 1:2)
varNames = umx_paste_names(c("N", "E", "O", "A", "C"), "_T", 1:2)
umx_paste_names(c("IQ", "C"), cov = c("age"), sep = "_T", suffixes = 1:2)
umx_paste_names(c("IQ", "C"), cov = c("age"), sep = "_T", prefix= "mean_")
# For quick-typing, tvars is an alias for umx_paste_names
tvars(c("IQ", "C"), cov = "age", sep = "_T", prefix= "mean_")
tvars("IQ")
```

---

umx_polychoric *FIML-based polychoric, polyserial, and Pearson correlations*

Description

Compute polychoric/polyserial/Pearson correlations with FIML in OpenMx

Usage

```
umx_polychoric(
  data,
  useDeviations = TRUE,
  tryHard = c("no", "yes", "ordinal", "search")
)
```

Arguments

data	Dataframe
useDeviations	Whether to code the mode using deviation thresholds (default = TRUE)
tryHard	'no' uses normal mxRun (default), "yes" uses mxTryHard, and others used named versions: "mxTryHardOrdinal", "mxTryHardWideSearch"

Value

- list of output and diagnostics. matrix of correlations = $polychorics

References

- https://doi.org/10.3389/fpsyg.2016.00528

See Also

Other Data Functions: umxFactor(), umxHetCor(), umx_as_numeric(), umx_cont_2_quantiles(), umx_lower2full(), umx_make_MR_data(), umx_make_TwinData(), umx_make_fake_data(), umx_make_raw_from_cov(), umx_polypairwise(), umx_polytriowise(), umx_read_lower(), umx_rename(), umx_reorder(), umx_select_valid(), umx_stack(), umx

Examples

```
tmp = mtcars
tmp$am = umxFactor(mtcars$am)
tmp$vs = umxFactor(mtcars$vs)
tmp = umx_scale(tmp)
x = umx_polychoric(tmp[, c("am", "vs")], tryHard = "yes")
x$polychorics
cor(mtcars[, c("am", "vs")])
```

---

umx_polypairwise *FIML-based Pairwise polychoric, polyserial, and Pearson correlations*

Description

Compute polychoric/polyserial/Pearson correlations with FIML in OpenMx

Usage

```
umx_polypairwise(
  data,
  useDeviations = TRUE,
  printFit = FALSE,
  use = "any",
  tryHard = c("no", "yes", "ordinal", "search")
)
```

Arguments

data	Dataframe
useDeviations	Whether to code the mode using deviation thresholds (default = TRUE)
printFit	Whether to print information about the fit achieved (default = FALSE)
use	parameter (default = "any")
tryHard	'no' uses normal mxRun (default), "yes" uses mxTryHard, and others used named versions: "mxTryHardOrdinal", "mxTryHardWideSearch"

Value

- matrix of correlations

References

- https://doi.org/10.3389/fpsyg.2016.00528

See Also

Other Data Functions: umxFactor(), umxHetCor(), umx_as_numeric(), umx_cont_2_quantiles(), umx_lower2full(), umx_make_MR_data(), umx_make_TwinData(), umx_make_fake_data(), umx_make_raw_from_cov(), umx_polychoric(), umx_polytriowise(), umx_read_lower(), umx_rename(), umx_reorder(), umx_select_valid(), umx_stack(), umx

Examples

```
umx_set_optimizer("SLSQP")
tmp = mtcars
tmp$am = umxFactor(mtcars$am)
tmp$vs = umxFactor(mtcars$vs)
tmp = umx_scale(tmp)
x = umx_polypairwise(tmp[, c("hp", "mpg", "am", "vs")], tryHard = "yes")
x$R
cov2cor(x$R)
cor(mtcars[, c("hp", "mpg", "am", "vs")])
```

---

umx_polytriowise *FIML-based trio-based polychoric, polyserial, and Pearson correlations*

Description

Compute polychoric/polyserial/Pearson correlations with FIML in OpenMx.

Usage

```
umx_polytriowise(
  data,
  useDeviations = TRUE,
  printFit = FALSE,
  use = "any",
  tryHard = c("no", "yes", "ordinal", "search")
)
```

Arguments

data	Dataframe
useDeviations	Whether to code the mode using deviation thresholds (default = TRUE)
printFit	Whether to print information about the fit achieved (default = FALSE)
use	parameter (default = "any")
tryHard	'no' uses normal mxRun (default), "yes" uses mxTryHard, and others used named versions: "mxTryHardOrdinal", "mxTryHardWideSearch"

Value

- matrix of correlations

References

- https://doi.org/10.3389/fpsyg.2016.00528

See Also

Other Data Functions: umxFactor(), umxHetCor(), umx_as_numeric(), umx_cont_2_quantiles(), umx_lower2full(), umx_make_MR_data(), umx_make_TwinData(), umx_make_fake_data(), umx_make_raw_from_cov(), umx_polychoric(), umx_polypairwise(), umx_read_lower(), umx_rename(), umx_reorder(), umx_select_valid(), umx_stack(), umx

Examples

```
tmp = mtcars
tmp$am = umxFactor(mtcars$am)
tmp$vs = umxFactor(mtcars$vs)
tmp = umx_scale(tmp)
x = umx_polytriowise(tmp[, c("hp", "mpg", "am", "vs")], tryHard = "yes")
x$R
cor(mtcars[, c("hp", "mpg", "am", "vs")])
```

umx_print *Print tables in a range of formats (markdown default, see umx_set_table_format() for other formats) or as a web browser table.*

Description

To aid interpretability of printed tables from OpenMx (and elsewhere) you can change how NA and zero appear, and suppressing values below a certain cut-off. By default, Zeros have the decimals suppressed, and NAs are suppressed altogether.

Usage

```
umx_print(
  x,
  digits = getOption("digits"),
  quote = FALSE,
  na.print = "",
  zero.print = "0",
  justify = "none",
  file = c(NA, "tmp.html"),
  suppress = NULL,
  append = FALSE,
  sortableDF = TRUE,
  both = TRUE,
  ...
)
```

Arguments

x	A data.frame to print (matrices will be coerced to data.frame)
digits	The number of decimal places to print (getOption("digits"))
quote	Parameter passed to print (FALSE)
na.print	String to replace NA with ("")
zero.print	String to replace 0.000 with ("0")
justify	Parameter passed to print (defaults to "none")
file	whether to write to a file (defaults to NA (no file). Use "tmp.html" to open table in browser.
suppress	minimum numeric value to print (NULL = print all values, no matter how small)
append	If html, is this appended to file? (FALSE)
sortableDF	If html, is table sortable? (TRUE)
both	If html, is table also printed as markdown? (TRUE)
...	Optional parameters for print

Value

- A dataframe of text

See Also

umx_msg(), umx_set_table_format()

Other Miscellaneous Utility Functions: install.OpenMx(), qm(), umxBrownie(), umxLav2RAM(), umxRAM2Lav(), umxVersion(), umx_array_shift(), umx_find_object(), umx_msg(), umx_open_CRAN_page(), umx_pad(), umx_score_scale(), umx

Examples

```
umx_print(mtcars[1:10,], digits = 2, zero.print = ".", justify = "left")
umx_print(mtcars[1,1:2], digits = 2, zero.print = "")
## Not run:
umx_print(mtcars[1:10,], file = "html")
umx_print(mtcars[1:10,], file = "tmp.html")

## End(Not run)
```

---

umx_read_lower *Read lower-triangle of data matrix from console or file*

Description

umx_read_lower will read a lower triangle of data, either from the console, or from file, and return a full matrix, optionally coerced to positive definite. This is useful, especially when copying data from a paper that includes just the lower triangle of a correlation matrix.

Usage

```
umx_read_lower(file = "", diag = TRUE, names = NULL, ensurePD = FALSE)
```

Arguments

file	Path to a file to read (Default "" will read from user input)
diag	Whether the data include the diagonal. Defaults to TRUE
names	The default names for the variables. Defaults to as.character(paste("X", 1:n, sep=""))
ensurePD	Whether to coerce the resultant matrix to positive definite (Defaults to FALSE)

Value

- matrix()

References

- https://github.com/tbates/umx, https://tbates.github.io

See Also

Other Data Functions: umxFactor(), umxHetCor(), umx_as_numeric(), umx_cont_2_quantiles(), umx_lower2full(), umx_make_MR_data(), umx_make_TwinData(), umx_make_fake_data(), umx_make_raw_from_cov(), umx_polychoric(), umx_polypairwise(), umx_polytriowise(), umx_rename(), umx_reorder(), umx_select_valid(), umx_stack(), umx

Examples

```
## Not run:
require(umx) # for umxRAM
IQtests = c("brainstorm", "matrix", "moral", "shopping", "typing")
allCols = c("C", IQtests, "avgIQ", "maxIQ", "video")

df = umx_read_lower(file = "", diag = FALSE)
0.38
0.86 0.30
0.42 0.12 0.27
0.66 0.21 0.38 0.18
0.80 0.13 0.50 0.25 0.43
0.19 0.11 0.19 0.12 -0.06 0.22
0.27 0.09 0.33 0.05 -0.04 0.28 .73
0.52 0.17 0.38 0.37 0.39 0.44 0.18 0.13

dimnames(df) = list(allCols, allCols) # manually add

df = umx_read_lower(file = "", diag = FALSE, names = allCols, ensurePD= TRUE)
0.38
0.86 0.30
0.42 0.12 0.27
0.66 0.21 0.38 0.18
0.80 0.13 0.50 0.25 0.43
0.19 0.11 0.19 0.12 -0.06 0.22
0.27 0.09 0.33 0.05 -0.04 0.28 .73
0.52 0.17 0.38 0.37 0.39 0.44 0.18 0.13


round(df, 2)

m1 = umxRAM("wooley", data = mxData(df, type="cov", numObs = 90),
umxPath("g", to = IQtests),
umxPath(var = "g", fixedAt= 1),
umxPath(var = IQtests)
)
summary(m1)

## End(Not run)
```

umx_rename *umx_rename*

Description

Returns a dataframe with variables renamed as desired.

Usage

```
umx_rename(
  data,
  from = NULL,
  to = NULL,
  regex = NULL,
  test = FALSE,
  old = "deprecated",
  replace = "deprecated"
)
```

Arguments

data	The dataframe in which to rename variables
from	List of existing names that will be found and replaced by the contents of replace. (optional: Defaults to NULL).
to	If used alone, a named collection of c(oldName = "newName") pairs. OR, if "from" is a list of existing names, the list of new names) OR, if "regex" is a regular expression, the replace string)
regex	Regular expression with matches will be replaced using replace as the replace string. (Optional: Defaults to NULL).
test	Whether to report a "dry run", not changing anything. (Default = FALSE).
old	deprecated: use from
replace	deprecated: use to

Details

Unlike similar functions in other packages, it checks that the variables exist, and that the new names do not.

Importantly, it also supports regular expressions. This allows you to find and replace text based on patterns and replacements. so to change "replacement" to "in place", grep=re(place)ment, replace= in \\1.

note:To use replace list, you must say c(old = "new"), not c(old -> "new")

Value

- dataframe with columns renamed.

See Also

namez to filter (and replace) names, Also umx_check_names to check for existence of names in a dataframe.

Other Data Functions: umxFactor(), umxHetCor(), umx_as_numeric(), umx_cont_2_quantiles(), umx_lower2full(), umx_make_MR_data(), umx_make_TwinData(), umx_make_fake_data(), umx_make_raw_from_cov(), umx_polychoric(), umx_polypairwise(), umx_polytriowise(), umx_read_lower(), umx_reorder(), umx_select_valid(), umx_stack(), umx

Examples

```
tmp = mtcars

tmp = umx_rename(tmp, to = c(cyl = "cylinder"))
# let's check cyl has been changed to cylinder...
namez(tmp, "c")

# Alternate style: from->to, first with a test-run
# Dry run
tmp = umx_rename(tmp, from = "disp", to = "displacement", test= TRUE)
# Actually do it
tmp = umx_rename(tmp, from = c("disp"), to = c("displacement"))
umx_check_names("displacement", data = tmp, die = TRUE)
namez(tmp, "disp")

# This will warn that "disp" does not exist (anymore)
new = c("auto", "displacement", "rear_axle_ratio")
tmp = umx_rename(tmp, from = c("am", "disp", "drat"), to = new)
namez(tmp, "a") # still updated am to auto (and rear_axle_ratio)

# Test using regex (in this case to revert "displacement" to "disp")
tmp = umx_rename(tmp, regex = "lacement", to = "", test= TRUE)
tmp = umx_rename(tmp, regex = "lacement", to = "") # revert to disp
umx_names(tmp, "^d") # all names beginning with a d

# advanced: checking deprecated format handled...
tmp = umx_rename(tmp, old = c("am", "disp", "drat"), replace = new)
```

---

umx_rename_file *Rename files*

Description

Rename files. On OS X, the function can access the current front-most Finder window. The file renaming is fast and, because you can use regular expressions, powerful.

Usage

```
umx_rename_file(
  findStr = NA,
  replaceStr = NA,
  baseFolder = "Finder",
  ignoreSuffix = TRUE,
  listPattern = NULL,
  test = TRUE,
  overwrite = FALSE
)
```

Arguments

findStr	The (regex) string to find, i.e., "cat"
replaceStr	The (regex) replacement string "\1 are not dogs"
baseFolder	The folder to search in. If set to "Finder" (and you are on OS X) it will use the current front-most Finder window. If it is blank, a choose folder dialog will be thrown.
ignoreSuffix	Whether to ignore (don't search in) the suffix (filetype like .mpg) TRUE.
listPattern	A pre-filter for files
test	Boolean determining whether to change files on disk, or just report on what would have happened (Defaults to test = TRUE)
overwrite	Boolean determining if an existing file will be overwritten (Defaults to the safe FALSE)

Value

None

See Also

Other File Functions: dl_from_dropbox(), umx_file_load_pseudo(), umx_make_sql_from_excel(), umx_move_file(), umx_open(), umx_write_to_clipboard(), umx

Examples

```
## Not run:
# "Season 01" --> "S01" in current folder in MacOS Finder
umx_rename_file("[Ss]eason +([0-9]+)", replaceStr="S\1", baseFolder = "Finder", test = TRUE)

## End(Not run)
```

umx_reorder *Reorder or drop variables from a correlation/covariance matrix.*

Description

Reorder the variables in a correlation matrix. Can also remove one or more variables from a matrix using this function.

Usage

```
umx_reorder(old, newOrder, force = FALSE)
```

Arguments

old	a square matrix of correlation or covariances to reorder
newOrder	Variables you want in the order you wish to have
force	Just assume input is value (default = FALSE)

Value

- the re-ordered/resized matrix

References

- <https://www.github.com/tbates/umx>

See Also

Other Data Functions: umxFactor(), umxHetCor(), umx_as_numeric(), umx_cont_2_quantiles(), umx_lower2full(), umx_make_MR_data(), umx_make_TwinData(), umx_make_fake_data(), umx_make_raw_from_cov(), umx_polychoric(), umx_polypairwise(), umx_polytriowise(), umx_read_lower(), umx_rename(), umx_select_valid(), umx_stack(), umx

Examples

```
oldMatrix = cov(mtcars)
umx_reorder(oldMatrix, newOrder = c("mpg", "cyl", "disp")) # first 3
umx_reorder(oldMatrix, newOrder = c("hp", "disp", "cyl")) # subset and reordered
umx_reorder(oldMatrix, "hp") # edge-case of just 1-var
```

umx_residualize *Easily residualize variables in long or wide dataframes, returning them changed in-place.*

Description

Residualize one or more variables residualized against covariates, and return a complete dataframe with residualized variable in place. Optionally, this also works on wide (i.e., twin) data. Just supply suffixes to identify the paired-wide columns (see examples).

Usage

```
umx_residualize(var, covs = NULL, suffixes = NULL, data)
```

Arguments

var	The base name of the variable you want to residualize. Alternatively, a regression formula() containing var on the lhs, and covs on the rhs
covs	Covariates to residualize on.
suffixes	Suffixes that identify the variable for each twin, i.e. c("_T1", "_T2") Up to you to check all variables are present!
data	The dataframe containing all the variables

Details

In R, residuals for a variable can be found with the following statement:

`tmp <-residuals(lm(var ~ cov1 + cov2,data = data,na.action = na.exclude))`

This tmp variable could then be written over the old data:

umx_residualize obviates the user having to build the lm, set na.action, or replace the data. In addition, it has the powerful feature of operating on a list of variables, and of operating on wide data, expanding the var name using a set of variable-name suffixes.

Value

- dataframe with var residualized in place (i.e under its original column name)

References

- https://tbates.github.io, https://github.com/tbates/umx

See Also

Other Twin Data functions: umx_long2wide(), umx_make_TwinData(), umx_make_twin_data_nice(), umx_scale_wide_twin_data(), umx_wide2long(), umx

Examples

```
# Residualize mpg on cylinders and displacement
r1 = umx_residualize("mpg", c("cyl", "disp"), data = mtcars)
r2 = residuals(lm(mpg ~ cyl + disp, data = mtcars, na.action = na.exclude))
all(r1$mpg == r2)

# ============================
# = Use the formula interface =
# ============================
r1 = umx_residualize(mpg ~ cyl + I(cyl^2) + disp, data = mtcars)

# validate against using lm
r2 = residuals(lm(mpg ~ cyl + I(cyl^2) + disp, data = mtcars, na.action = na.exclude))
all(r1$mpg == r2)

# ============================================================
# = Residualize twin data (i.e. wide or "1 family per row") =
# ============================================================
# Make some toy "twin" data to demonstrate with
tmp = mtcars
tmp$mpg_T1  = tmp$mpg_T2  = tmp$mpg
tmp$cyl_T1  = tmp$cyl_T2  = tmp$cyl
tmp$disp_T1 = tmp$disp_T2 = tmp$disp

covs = c("cyl", "disp")
tmp= umx_residualize(var="mpg", covs=covs, suffixes=c("_T1","_T2"), data = tmp)
str(tmp[1:5, 12:17])

# ==================================
# = Residualize several DVs at once =
# ==================================
df1 = umx_residualize(c("mpg", "hp"), cov = c("cyl", "disp"), data = tmp)
df2 = residuals(lm(hp ~ cyl + disp, data = tmp, na.action = na.exclude))
all(df1$hp == df2)
```

---

umx_rot *Rotate a vector*

Description

umx_rot rotates the items of a vector (1 place, by default). So: c(1,2,3) -> c(2,3,1)

Usage

```
umx_rot(vec, na.last = FALSE)
```

Arguments

vec	vector to rotate
na.last	Whether to set the last value to NA (default = FALSE)

Value

- mxModel()

References

- https://tbates.github.io

See Also

Other String Functions: umx_explode_twin_names(), umx_explode(), umx_grep(), umx_names(), umx_paste_names(), umx_str_chars(), umx_str_from_object(), umx_trim(), umx

Examples

```
umx_rot(1:10)
umx_rot(c(3,4,5,6,7))
# [1] 4 5 6 7 3
```

---

umx_round *umx_round*

Description

A version of round() which works on dataframes that contain non-numeric data (or data that cannot be coerced to numeric) Helpful for dealing with table output that mixes numeric and string types.

Usage

```
umx_round(df, digits = getOption("digits"), coerce = FALSE)
```

Arguments

df	a dataframe to round in
digits	how many digits to round to (defaults to getOption("digits"))
coerce	whether to make the column numeric if it is not (default = FALSE)

Value

- mxModel()

References

- https://www.github.com/tbates/umx

See Also

Other Miscellaneous Stats Helpers: FishersMethod(), SE_from_p(), oddsratio(), reliability(), umxCov2cor(), umxHetCor(), umxWeightedAIC(), umx_apply(), umx_cor(), umx_means(), umx_r_test(), umx_scale(), umx_var(), umx

Examples

```
head(umx_round(mtcars, coerce = FALSE))
head(umx_round(mtcars, coerce = TRUE))
```

---

umx_r_test *Test the difference between correlations for significance.*

Description

umx_r_test is a wrapper around the cocor test of difference between correlations.

Usage

```
umx_r_test(
  data = NULL,
  vars = vars,
  alternative = c("two.sided", "greater", "less")
)
```

Arguments

data	The dataset.
vars	Four variables forming the two pairs of columns: "j & k" and "h & m".
alternative	A two (default) or one-sided (greater less) test.

Details

Currently umx_r_test handles the test of whether r.jk and r.hm differ in magnitude. i.e, two non-overlapping (no variable in common) correlations in the same dataset. In the future it will be expanded to handle overlapping correlations, and to take correlation matrices as input.

Value

cocor result.

See Also

Other Miscellaneous Stats Helpers: FishersMethod(), SE_from_p(), oddsratio(), reliability(), umxCov2cor(), umxHetCor(), umxWeightedAIC(), umx_apply(), umx_cor(), umx_means(), umx_round(), umx_scale(), umx_var(), umx

Examples

```
# Is the correlation of mpg with cylinder count different from that
# obtaining between disp and hp?
vars = c("mpg", "cyl", "disp", "hp")
umx_r_test(mtcars, vars)
```

umx_scale *Scale data columns, skipping non-scalable columns*

Description

umx_scale applies scale to the columns of a data.frame. By default it scales all numeric columns, and is smart enough to skip non-scalable columns (strings, factors, etc.).

You can also select which columns to convert. This is useful when you want to avoid numeric columns which are actually factors.

note: By default, the attributes which scale adds ("scaled:center" and "scaled:scale" removed to leave nice numeric columns. Set attr= TRUE to preserve these.

Usage

```
umx_scale(
  df,
  varsToScale = NULL,
  coerce = FALSE,
  attr = FALSE,
  verbose = FALSE
)
```

Arguments

df	A dataframe to scale (or a numeric vector)
varsToScale	(leave blank to scale all)
coerce	Whether to coerce non-numerics to numeric (Defaults to FALSE.
attr	to strip off the attributes scale creates (FALSE by default)
verbose	Whether to report which columns were scaled (default FALSE)

Value

- new dataframe with scaled variables

References

- https://www.github.com/tbates/umx

See Also

umx_scale_wide_twin_data scale

Other Miscellaneous Stats Helpers: FishersMethod(), SE_from_p(), oddsratio(), reliability(), umxCov2cor(), umxHetCor(), umxWeightedAIC(), umx_apply(), umx_cor(), umx_means(), umx_r_test(), umx_round(), umx_var(), umx

Examples

```
data(twinData)
df = umx_scale(twinData, varsToScale = c("wt1", "wt2"))
df = umx_scale(twinData,  attr= TRUE)
plot(wt1 ~ wt2, data = df)
```

`umx_scale_wide_twin_data` *Scale wide twin data*

Description

Scale wide data across all twins. You offer up a list of variables to scale, e.g. c("DEP", "bmi") and the separator (e.g. `sep = "_T"`) and twin suffixes e.g. 1:2 that paste together to make complete variable names: e.g. "DEP_T1" and "DEP_T2".

Usage

```
umx_scale_wide_twin_data(varsToScale, sep, data, twins = 1:2)
```

Arguments

`varsToScale`	The base names of the variables ("weight" etc.)
`sep`	The suffix that distinguishes each case, e.g. "_T")
`data`	A wide dataframe
`twins`	Legal digits following sep (default 1:2)

Value

- dataframe with varsToScale standardized

References

- https://www.github.com/tbates/umx

See Also

umx_scale

Other Twin Data functions: `umx_long2wide()`, `umx_make_TwinData()`, `umx_make_twin_data_nice()`, `umx_residualize()`, `umx_wide2long()`, `umx`

Examples

```
data(twinData)
df = umx_scale_wide_twin_data(data = twinData, varsToScale = c("ht", "wt"), sep = "")
plot(wt1 ~ wt2, data = df)
```

umx_score_scale *Score a psychometric scale by summing normal and reversed items.*

Description

Use this function to generate scores as the appropriate sum of responses to the normal and reversed items in a scale.

Items must be named on the pattern baseN, where base is the string common to all item (column) names and N is the item number in the scale.

pos and rev are vectors of the item numbers for the normal and reverse-scored item numbers.

To reverse items, the function uses max and min as the lowest and highest possible response scores to compute how to reverse items.

note: min defaults to 1.

Usage

```
umx_score_scale(
  base = NULL,
  pos = NULL,
  rev = NULL,
  min = 1,
  max = NULL,
  data = NULL,
  score = c("totals", "mean", "max"),
  name = NULL,
  na.rm = FALSE
)
```

Arguments

base	String common to all item names.
pos	The positive-scored item numbers.
rev	The reverse-scored item numbers.
min	Min possible score (default = 1). Not implemented for values other than 1 so far...
max	Max possible score for an item (to compute how to reverse items).
data	The data frame
score	Whether to compute the score totals, mean, or max (default = "totals")
name	= name of the scale to be returned. Defaults to "base_score"
na.rm	Whether to delete NAs when computing scores (Default = TRUE) Note: Choice affects mean!

Details

In the presence of NAs, score= "mean" and score = "totals" both return NA unless na.rm = TRUE. score = "max", ignores NAs no matter what.

Value

- scores

See Also

Other Miscellaneous Utility Functions: install.OpenMx(), qm(), umxBrownie(), umxLav2RAM(), umxRAM2Lav(), umxVersion(), umx_array_shift(), umx_find_object(), umx_msg(), umx_open_CRAN_page(), umx_pad(), umx_print(), umx

Examples

```
library(psych)
data(bfi)

# ==============================
# = Score Agreeableness totals =
# ==============================

# Handscore subject 1
# A1(Reversed) + A2 + A3 + A4 + A5
#      (6+1)-2 + 4  + 3  + 4  + 4  = 20

tmp = umx_score_scale("A", pos = 2:5, rev = 1, max = 6, data= bfi, name = "A")
tmp[1, namez(tmp, "A",ignore.case=FALSE)]
#  A1 A2 A3 A4 A5  A
#  2  4  3  4  4  20

# ================================================================================
# = Note: (as of a fix in 2020-05-08) items not reversed in the returned data set =
# ================================================================================
tmp = umx_score_scale("A", pos = 1, rev = 2:5, max = 6, data= bfi, name = "A")
tmp[1, namez(tmp, "A",ignore.case=FALSE)]
#   A1 A2 A3 A4 A5   A
#   2   4  3  4  4 = 15

tmp = umx_score_scale("A", pos = 2:5, rev = 1, max = 6, data= bfi, name = "A", score="mean")
tmp$A[1] # subject 1 mean = 4

# ==========================================
# = How does mean react to a missing value? =
# ==========================================
tmpDF = bfi
tmpDF[1, "A1"] = NA
tmp = umx_score_scale("A", pos = 2:5, rev = 1, max = 6, data= tmpDF, name = "A", score="mean")
tmp$A[1] # NA: (na.rm defaults to FALSE)

tmp = umx_score_scale("A", pos = 2:5, rev = 1, max = 6, data= tmpDF,
```

```
     name = "A", score="mean", na.rm=TRUE)
tmp$A[1] # 3.75

# ===============
# = Score = max =
# ===============
tmp = umx_score_scale("A", pos = 2:5, rev = 1, max = 6, data= bfi, name = "A", score="max")
tmp$A[1] # subject 1 max = 5 (the reversed item 1)

tmp = umx_score_scale("E", pos = c(3,4,5), rev = c(1,2), max = 6, data= tmp)
tmp$E_score[1] # default scale name

# Using @BillRevelle's psych package: More diagnostics, including alpha
scores= psych::scoreItems(items = bfi, min = 1, max = 6, keys = list(
E = c("-E1","-E2", "E3", "E4", "E5"),
A = c("-A1", "A2", "A3", "A4", "A5")
))
summary(scores)
scores$scores[1,]
#  E   A
# 3.8 4.0

# Compare output
# (note, by default psych::scoreItems replaces NAs with the sample median...)
RevelleE = as.numeric(scores$scores[,"E"]) * 5
all(RevelleE == tmp[,"E_score"], na.rm = TRUE)
```

`umx_select_valid` *Update NA values in one column with valid entries from another*

Description

Merge valid entries from two columns

Usage

```
umx_select_valid(col1, col2, bothways = FALSE, data)
```

Arguments

`col1`	name of the first column
`col2`	name of the second column
`bothways`	Whether to replace from 1 to 2 as well as from 2 to 1
`data`	The dataframe containing the two columns.

Value

- Updated dataframe

See Also

- within()

Other Data Functions: umxFactor(), umxHetCor(), umx_as_numeric(), umx_cont_2_quantiles(), umx_lower2full(), umx_make_MR_data(), umx_make_TwinData(), umx_make_fake_data(), umx_make_raw_from_cov(), umx_polychoric(), umx_polypairwise(), umx_polytriowise(), umx_read_lower(), umx_rename(), umx_reorder(), umx_stack(), umx

Examples

```
tmp = mtcars
tmp$newDisp = tmp$disp
tmp$disp[c(1,3,6)] = NA
anyNA(tmp$disp) # column has NAs
tmp = umx_select_valid("disp", "newDisp", data = tmp)
anyNA(tmp$disp) # column repaired
```

---

umx_set_auto_plot *umx_set_auto_plot*

Description

Set autoPlot default for models like umxACE umxGxE etc.

Usage

```
umx_set_auto_plot(autoPlot = NULL, silent = FALSE)
```

Arguments

autoPlot	If TRUE, sets the umx_auto_plot option. Else returns the current value of umx_auto_plot
silent	If TRUE, no message will be printed.

Value

- Current umx_auto_plot setting
- existing value

References

- https://tbates.github.io, https://github.com/tbates/umx

See Also

Other Get and set: umx_get_checkpoint(), umx_get_options(), umx_set_auto_run(), umx_set_checkpoint(), umx_set_condensed_slots(), umx_set_cores(), umx_set_data_variance_check(), umx_set_mvn_optimization_opt
umx_set_optimizer(), umx_set_plot_file_suffix(), umx_set_plot_format(), umx_set_separator(), umx_set_silent(), umx_set_table_format(), umx

Examples

```
library(umx)
umx_set_auto_plot() # print current state
old = umx_set_auto_plot(silent = TRUE) # store existing value
old
umx_set_auto_plot(TRUE)   # set to on (internally stored as "name")
umx_set_auto_plot(FALSE)  # set to off (internally stored as NA)
umx_set_auto_plot(old)    # reinstate
```

`umx_set_auto_run` *umx_set_auto_run*

Description

Set autoRun default for models like umxACE umxGxE etc.

Usage

```
umx_set_auto_run(autoRun = NA, silent = FALSE)
```

Arguments

autoRun	If TRUE or FALSE, sets the umx_auto_run option. Else returns the current value of umx_auto_run
silent	If TRUE, no message will be printed.

Value

- Current umx_auto_run setting

References

- https://tbates.github.io, https://github.com/tbates/umx

See Also

Other Get and set: umx_get_checkpoint(), umx_get_options(), umx_set_auto_plot(), umx_set_checkpoint(), umx_set_condensed_slots(), umx_set_cores(), umx_set_data_variance_check(), umx_set_mvn_optimization_opt umx_set_optimizer(), umx_set_plot_file_suffix(), umx_set_plot_format(), umx_set_separator(), umx_set_silent(), umx_set_table_format(), umx

Examples

```
library(umx)
umx_set_auto_run() # print existing value
old = umx_set_auto_run(silent = TRUE) # store existing value
umx_set_auto_run(FALSE)  # set to FALSE
umx_set_auto_run(old)    # reinstate
```

umx_set_checkpoint *umx_set_checkpoint*

Description

Set the checkpoint status for a model or global options

Usage

```
umx_set_checkpoint(
  interval = 1,
  units = c("evaluations", "iterations", "minutes"),
  prefix = "",
  directory = getwd(),
  model = NULL
)
```

Arguments

interval	How many units between checkpoints: Default = 1. A value of zero sets always to 'No' (i.e., do not checkpoint all models during optimization)
units	units to count in: Default unit is 'evaluations' ('minutes' is also legal)
prefix	string prefix to add to all checkpoint filenames (default = "")
directory	a directory, i.e "~/Desktop" (defaults to getwd())
model	(optional) model to set options in (default = NULL)

Value

- mxModel if provided

References

- https://tbates.github.io, https://github.com/tbates/umx

See Also

Other Get and set: umx_get_checkpoint(), umx_get_options(), umx_set_auto_plot(), umx_set_auto_run(), umx_set_condensed_slots(), umx_set_cores(), umx_set_data_variance_check(), umx_set_mvn_optimization_opt
umx_set_optimizer(), umx_set_plot_file_suffix(), umx_set_plot_format(), umx_set_separator(), umx_set_silent(), umx_set_table_format(), umx

Examples

```
## Not run:
umx_set_checkpoint(interval = 1, "evaluations", dir = "~/Desktop/")
# Turn off checkpointing with interval = 0
umx_set_checkpoint(interval = 0)
umx_set_checkpoint(2, "evaluations", prefix="SNP_1")
require(umx)
data(demoOneFactor)
manifests = names(demoOneFactor)
m1 = umxRAM("One Factor", data = demoOneFactor, type = "cov",
umxPath("G", to = manifests),
umxPath(var = manifests),
umxPath(var = "G", fixedAt = 1)
)
m1 = umx_set_checkpoint(model = m1)
m1 = mxRun(m1)
umx_checkpoint(0)

## End(Not run)
```

`umx_set_condensed_slots` *umx_set_condensed_slots*

Description

Sets whether newly-created mxMatrices are to be condensed (set to NULL if not being used) or not.

Usage

```
umx_set_condensed_slots(state = NA, silent = FALSE)
```

Arguments

`state`	what state (TRUE or FALSE) to set condensed slots (default NA returns current value).
`silent`	If TRUE, no message will be printed.

Value

- current value of condensed slots

References

- https://tbates.github.io, https://github.com/tbates/umx

See Also

Other Get and set: umx_get_checkpoint(), umx_get_options(), umx_set_auto_plot(), umx_set_auto_run(), umx_set_checkpoint(), umx_set_cores(), umx_set_data_variance_check(), umx_set_mvn_optimization_options(umx_set_optimizer(), umx_set_plot_file_suffix(), umx_set_plot_format(), umx_set_separator(), umx_set_silent(), umx_set_table_format(), umx

Examples

```
library(umx)
umx_set_condensed_slots() # print
old = umx_set_condensed_slots(silent = TRUE) # store the existing state
umx_set_condensed_slots(TRUE) # update globally
umx_set_condensed_slots(old) # set back
```

---

umx_set_cores *umx_set_cores*

Description

set the number of cores (threads) used by OpenMx

Usage

```
umx_set_cores(cores = NA, model = NULL, silent = FALSE)
```

Arguments

cores	number of cores to use. NA (the default) returns current value. "-1" will set to imxGetNumThreads().
model	an (optional) model to set. If left NULL, the global option is updated.
silent	If TRUE, no message will be printed.

Value

- number of cores

References

- https://tbates.github.io, https://github.com/tbates/umx

See Also

Other Get and set: umx_get_checkpoint(), umx_get_options(), umx_set_auto_plot(), umx_set_auto_run(), umx_set_checkpoint(), umx_set_condensed_slots(), umx_set_data_variance_check(), umx_set_mvn_optimizatio umx_set_optimizer(), umx_set_plot_file_suffix(), umx_set_plot_format(), umx_set_separator(), umx_set_silent(), umx_set_table_format(), umx

Examples

```
library(umx)
manifests = c("mpg", "disp", "gear")
m1 <- mxModel("ind", type = "RAM",
manifestVars = manifests,
mxPath(from = manifests, arrows = 2),
mxPath(from = "one", to = manifests),
mxData(mtcars[, manifests], type = "raw")
)
umx_set_cores() # print current value
oldCores <- umx_set_cores(silent = TRUE)  # store existing value
umx_set_cores(imxGetNumThreads()) # set to max
umx_set_cores(-1); umx_set_cores() # set to max
m1 = umx_set_cores(1, m1)  # set m1 usage to 1 core
umx_set_cores(model = m1)  # show new value for m1
umx_set_cores(oldCores)    # reinstate old global value
```

---

umx_set_data_variance_check

umx_set_data_variance_check

Description

Set default for data checking in models like umxACE umxGxE etc.

Usage

```
umx_set_data_variance_check(minVar = NULL, maxVarRatio = NULL, silent = FALSE)
```

Arguments

minVar	Set the threshold at which to warn user about variables with too-small variance. Else returns the current value of umx_minVar
maxVarRatio	Set the option for threshold at which to warn user variances differ too much. Else returns the current value of umx_maxVarRatio
silent	If TRUE, no message will be printed.

Value

- list of umx_minVar and umx_maxVarRatio settings

See Also

xmu_check_variance which uses these to check sanity in the variances of a data frame.

Other Get and set: umx_get_checkpoint(), umx_get_options(), umx_set_auto_plot(), umx_set_auto_run(), umx_set_checkpoint(), umx_set_condensed_slots(), umx_set_cores(), umx_set_mvn_optimization_options(), umx_set_optimizer(), umx_set_plot_file_suffix(), umx_set_plot_format(), umx_set_separator(), umx_set_silent(), umx_set_table_format(), umx

Examples

```
library(umx)
umx_set_data_variance_check() # print current state
old = umx_set_data_variance_check(silent = TRUE) # store existing value
umx_set_data_variance_check(minVar = .01)
umx_set_data_variance_check(maxVarRatio = 500)
umx_set_data_variance_check(minVar = old$minVar, maxVarRatio = old$maxVarRatio) # reinstate
```

---

umx_set_mvn_optimization_options

Set options that affect optimization in OpenMx

Description

umx_set_mvn_optimization_options provides access to get and set options affecting optimization.

Usage

```
umx_set_mvn_optimization_options(
  opt = c("mvnRelEps", "mvnMaxPointsA"),
  value = NULL,
  model = NULL,
  silent = FALSE
)
```

Arguments

opt	default returns current values of the options listed. Currently "mvnRelEps" and "mvnMaxPointsA".
value	If not NULL, the value to set the opt to (can be a list of length(opt))
model	A model for which to set the optimizer. Default (NULL) sets the optimizer globally.
silent	If TRUE, no message will be printed.

Details

note: For mvnRelEps, values between .0001 to .01 are conventional. Smaller values slow optimization.

Value

- current values if no value set.

References

- https://tbates.github.io, https://github.com/tbates/umx

See Also

Other Get and set: umx_get_checkpoint(), umx_get_options(), umx_set_auto_plot(), umx_set_auto_run(), umx_set_checkpoint(), umx_set_condensed_slots(), umx_set_cores(), umx_set_data_variance_check(), umx_set_optimizer(), umx_set_plot_file_suffix(), umx_set_plot_format(), umx_set_separator(), umx_set_silent(), umx_set_table_format(), umx

Examples

```
umx_set_mvn_optimization_options() # print the existing state(s)
umx_set_mvn_optimization_options("mvnRelEps") # show this one
## Not run:
umx_set_mvn_optimization_options("mvnRelEps", .01) # update globally

## End(Not run)
```

---

umx_set_optimizer *Set the optimizer in OpenMx*

Description

umx_set_optimizer provides an easy way to get and set the default optimizer.

Usage

```
umx_set_optimizer(opt = NA, model = NULL, silent = FALSE)
```

Arguments

opt	default (NA) returns current value. Current alternatives are "NPSOL" "SLSQP" and "CSOLNP".
model	A model for which to set the optimizer. Default (NULL) sets the optimizer globally.
silent	If TRUE, no message will be printed.

Value

- current optimizer if nothing requested to be set.

References

- https://tbates.github.io, https://github.com/tbates/umx

See Also

Other Get and set: umx_get_checkpoint(), umx_get_options(), umx_set_auto_plot(), umx_set_auto_run(), umx_set_checkpoint(), umx_set_condensed_slots(), umx_set_cores(), umx_set_data_variance_check(), umx_set_mvn_optimization_options(), umx_set_plot_file_suffix(), umx_set_plot_format(), umx_set_separator(), umx_set_silent(), umx_set_table_format(), umx

Examples

```
library(umx)
umx_set_optimizer() # print the existing state
old = umx_set_optimizer(silent = TRUE) # store the existing state
umx_set_optimizer("SLSQP") # update globally
umx_set_optimizer(old) # set back
```

---

umx_set_plot_file_suffix

Set output suffix used in umx plot (structural diagrams) files to disk

---

Description

Set output file suffix (default = "gv", alternative is "dot"). If you call this with no value, it will return the current setting. If you call it with TRUE, it toggles the setting.

Usage

```
umx_set_plot_file_suffix(umx.plot.suffix = NULL, silent = FALSE)
```

Arguments

umx.plot.suffix	the suffix for plots files (if empty, returns the current value of umx.plot.format). If "TRUE", then toggles
silent	If TRUE, no message will be printed.

Value

- Current umx.plot.suffix setting

References

- https://tbates.github.io, https://github.com/tbates/umx

See Also

Other Get and set: umx_get_checkpoint(), umx_get_options(), umx_set_auto_plot(), umx_set_auto_run(), umx_set_checkpoint(), umx_set_condensed_slots(), umx_set_cores(), umx_set_data_variance_check(), umx_set_mvn_optimization_options(), umx_set_optimizer(), umx_set_plot_format(), umx_set_separator(), umx_set_silent(), umx_set_table_format(), umx

Examples

```
umx_set_plot_file_suffix() # print current state
old = umx_set_plot_file_suffix(silent = TRUE) # store current value
umx_set_plot_file_suffix("dot")
umx_set_plot_file_suffix("gv")
umx_set_plot_file_suffix(old) # reinstate
```

umx_set_plot_format *Set output format of plots (structural diagrams) in umx*

Description

Set output format of plots (default = "DiagrammeR", alternative is "graphviz"). If you call this with no value, it will return the current setting. If you call it with TRUE, it toggles the setting.

Usage

```
umx_set_plot_format(umx.plot.format = NULL, silent = FALSE)
```

Arguments

umx.plot.format	format for plots (if empty, returns the current value of umx.plot.format). If "TRUE", then toggles
silent	If TRUE, no message will be printed.

Value

- Current umx.plot.format setting

References

- https://tbates.github.io, https://github.com/tbates/umx

See Also

Other Get and set: umx_get_checkpoint(), umx_get_options(), umx_set_auto_plot(), umx_set_auto_run(), umx_set_checkpoint(), umx_set_condensed_slots(), umx_set_cores(), umx_set_data_variance_check(), umx_set_mvn_optimization_options(), umx_set_optimizer(), umx_set_plot_file_suffix(), umx_set_separator(), umx_set_silent(), umx_set_table_format(), umx

Examples

```
library(umx)
umx_set_plot_format() # print current state
old = umx_set_plot_format(silent = TRUE) # store current value
umx_set_plot_format("graphviz")
umx_set_plot_format("DiagrammeR")
umx_set_plot_format(old) # reinstate
```

umx_set_separator *Set the separator*

Description

Set umx_default_separator (used in CI\[low sep high\]). Default = ","

Usage

```
umx_set_separator(umx_default_separator = NULL, silent = FALSE)
```

Arguments

umx_default_separator	separator for CIs etc. (if empty, returns the current value)
silent	If TRUE, no message will be printed.

Value

- Current umx_default_separator

See Also

Other Get and set: umx_get_checkpoint(), umx_get_options(), umx_set_auto_plot(), umx_set_auto_run(), umx_set_checkpoint(), umx_set_condensed_slots(), umx_set_cores(), umx_set_data_variance_check(), umx_set_mvn_optimization_options(), umx_set_optimizer(), umx_set_plot_file_suffix(), umx_set_plot_format(), umx_set_silent(), umx_set_table_format(), umx

Examples

```
library(umx)
umx_set_separator() # show current state
old = umx_set_separator(silent=TRUE) # store existing value
umx_set_separator("|")
umxAPA(.3, .2)
umx_set_separator(old)    # reinstate
```

umx_set_silent *Print anything when running a model?*

Description

Sets a umx property "silent" to TRUE or FALSE. This is fed to OpenMx::mxRun() inside functions like umxRAM(). If TRUE, then the progress messages from model runs are suppressed. Useful for power simulations etc.

Usage

```
umx_set_silent(value = NA, silent = FALSE)
```

Arguments

value	If TRUE mxRun is silent in most umx Models. Empty returns the current value.
silent	If TRUE, no message will be printed.

Value

- Current silent value

References

- https://tbates.github.io, https://github.com/tbates/umx

See Also

Other Get and set: umx_get_checkpoint(), umx_get_options(), umx_set_auto_plot(), umx_set_auto_run(), umx_set_checkpoint(), umx_set_condensed_slots(), umx_set_cores(), umx_set_data_variance_check(), umx_set_mvn_optimization_options(), umx_set_optimizer(), umx_set_plot_file_suffix(), umx_set_plot_format(), umx_set_separator(), umx_set_table_format(), umx

Examples

```
library(umx)
old = umx_set_silent() # print & store existing value
umx_set_silent(FALSE, silent = TRUE) # set to FALSE
umx_set_silent(old)   # reinstate
umx_set_silent() # print existing value
```

---

umx_set_table_format *umx_set_table_format*

Description

Set knitr.table.format default (output style for tables). Legal values are "latex", "html", "markdown", "pandoc", and "rst".

Usage

```
umx_set_table_format(knitr.table.format = NULL, silent = FALSE)
```

Arguments

knitr.table.format	format for tables (if empty, returns the current value of knitr.table.format)
silent	If TRUE, no message will be printed.

Value

- Current knitr.table.format setting

References

- https://tbates.github.io, https://github.com/tbates/umx

See Also

Other Get and set: umx_get_checkpoint(), umx_get_options(), umx_set_auto_plot(), umx_set_auto_run(), umx_set_checkpoint(), umx_set_condensed_slots(), umx_set_cores(), umx_set_data_variance_check(), umx_set_mvn_optimization_options(), umx_set_optimizer(), umx_set_plot_file_suffix(), umx_set_plot_format(), umx_set_separator(), umx_set_silent(), umx

Examples

```
library(umx)
umx_set_table_format() # show current state
old = umx_set_table_format() # store existing value
umx_set_table_format("latex")
umx_set_table_format("html")
umx_set_table_format("markdown")
umx_set_table_format("") # get available options
umx_set_table_format(old)    # reinstate
```

---

umx_stack *Stack data like stack() does, with more control.*

Description

Operates like stack(), but can preserve ("passalong") other variables on each row, and allows the user control over the values and group column names for ease of use.

Usage

```
umx_stack(x, select, passalong, valuesName = "values", groupName = "ind")
```

Arguments

x	a dataframe containing twin data.
select	The variables to stack (wide 2 long)
passalong	Variables to preserve on each row (e.g. age)
valuesName	The name for the new stacked column (default = "values")
groupName	The name for the column containing the grouping variable (default = "ind")

Value

- long-format dataframe

See Also

Other Data Functions: umxFactor(), umxHetCor(), umx_as_numeric(), umx_cont_2_quantiles(), umx_lower2full(), umx_make_MR_data(), umx_make_TwinData(), umx_make_fake_data(), umx_make_raw_from_cov(), umx_polychoric(), umx_polypairwise(), umx_polytriowise(), umx_read_lower(), umx_rename(), umx_reorder(), umx_select_valid(), umx

Examples

```
# Base-R stack function
df = stack(mtcars, select = c("disp", "hp"), drop=FALSE)

# umx_stack, with additional variables passed along
df= umx_stack(mtcars, select= c("disp", "hp"), passalong= "mpg")
str(df) # ind is a factor, with levels select
ggplot2::qplot(x = mpg, y= values, color=ind, data = df)
df= umx_stack(mtcars, select= c("disp", "hp"), passalong= "mpg")
ggplot2::qplot(x = mpg, y= values, group="ind", data = df)
```

---

umx_standardize *Return a standardized version of a Structural Model*

Description

Return the standardized version of a model (such as ACE, CP etc.)

Versions exist for RAM, ACE, ACEv, ACEcov, IP, CP and GxE models.

Usage

```
umx_standardize(model, ...)
```

Arguments

model	The mxModel() whose fit will be reported.
...	Other parameters.

Details

umx_standardize takes umx models, including RAM and twin models, and returns a standardized version.

See Also

Other xmu internal not for end user: umxModel(), umxRenameMatrix(), umxTwinMaker(), umx_APA_pval(), umx_fun_mean_sd(), umx_get_bracket_addresses(), umx_make(), umx_string_to_algebra(), umx, xmuHasSquareBrackets(), xmuLabel_MATRIX_Model(), xmuLabel_Matrix(), xmuLabel_RAM_Model(), xmuMI(), xmuMakeDeviationThresholdsMatrices(), xmuMakeOneHeadedPathsFromPathList(), xmuMakeTwoHeadedPathsFromPathList(), xmuMaxLevels(), xmuMinLevels(), xmuPropagateLabels(), xmuRAM2Ordinal(), xmuTwinSuper_Continuous(), xmuTwinUpgradeMeansToCovariateModel(), xmu_CI_merge(), xmu_CI_stash(), xmu_DF_to_mxData_TypeCov(), xmu_PadAndPruneForDefVars(), xmu_cell_is_on(), xmu_check_levels_identical(), xmu_check_needs_means(), xmu_check_variance(), xmu_clean_label(), xmu_data_missing(), xmu_data_swap_a_block(), xmu_describe_data_WLS(), xmu_dot_make_paths(), xmu_dot_make_residuals(), xmu_dot_maker(), xmu_dot_move_ranks(), xmu_dot_rank_str(), xmu_extract_column(), xmu_get_CI(), xmu_lavaan_process_group(), xmu_make_TwinSuperModel(), xmu_make_bin_cont_pair_data(), xmu_make_mxData(), xmu_match.arg(), xmu_name_from_lavaan_str(), xmu_path2twin(), xmu_path_regex(), xmu_safe_run_summary(), xmu_set_sep_from_suffix(), xmu_show_fit_or_comparison(), xmu_simplex_corner(), xmu_standardize_ACEcov(), xmu_standardize_ACEv(), xmu_standardize_ACE(), xmu_standardize_CP(), xmu_standardize_IP(), xmu_standardize_RAM(), xmu_standardize_SexLim(), xmu_standardize_Simplex(), xmu_start_value_list(), xmu_starts(), xmu_twin_add_WeightMatrices(), xmu_twin_check(), xmu_twin_get_var_names(), xmu_twin_upgrade_selDvs2SelVars()

---

umx_string_to_algebra *Convert a string to an OpenMx algebra*

Description

This is useful use to quickly and easily insert values from R variables into the string (using paste() and rep() etc.), then parse the string as an mxAlgebra argument.

Usage

```
umx_string_to_algebra(algString, name = NA, dimnames = NA)
```

Arguments

algString	a string to turn into an algebra
name	of the returned algebra
dimnames	of the returned algebra

Details

A use case is including a matrix exponent (that is A %% *A* %% A %*% A...) with a variable exponent.

Value

- mxAlgebra()

References

- https://www.github.com/tbates/umx

See Also

Other xmu internal not for end user: umxModel(), umxRenameMatrix(), umxTwinMaker(), umx_APA_pval(), umx_fun_mean_sd(), umx_get_bracket_addresses(), umx_make(), umx_standardize(), umx, xmuHasSquareBrackets(), xmuLabel_MATRIX_Model(), xmuLabel_Matrix(), xmuLabel_RAM_Model(), xmuMI(), xmuMakeDeviationThresholdsMatrices(), xmuMakeOneHeadedPathsFromPathList(), xmuMakeTwoHeadedPathsFromPathList(), xmuMaxLevels(), xmuMinLevels(), xmuPropagateLabels(), xmuRAM2Ordinal(), xmuTwinSuper_Continuous(), xmuTwinUpgradeMeansToCovariateModel(), xmu_CI_merge(), xmu_CI_stash(), xmu_DF_to_mxData_TypeCov(), xmu_PadAndPruneForDefVars(), xmu_cell_is_on(), xmu_check_levels_identical(), xmu_check_needs_means(), xmu_check_variance(), xmu_clean_label(), xmu_data_missing(), xmu_data_swap_a_block(), xmu_describe_data_WLS(), xmu_dot_make_paths(), xmu_dot_make_residuals(), xmu_dot_maker(), xmu_dot_move_ranks(), xmu_dot_rank_str(), xmu_extract_column(), xmu_get_CI(), xmu_lavaan_process_group(), xmu_make_TwinSuperModel(), xmu_make_bin_cont_pair_data(), xmu_make_mxData(), xmu_match.arg(), xmu_name_from_lavaan_str(), xmu_path2twin(), xmu_path_regex(), xmu_safe_run_summary(), xmu_set_sep_from_suffix(), xmu_show_fit_or_comparison(), xmu_simplex_corner(), xmu_standardize_ACEcov(), xmu_standardize_ACEv(), xmu_standardize_ACE(), xmu_standardize_CP(), xmu_standardize_IP(), xmu_standardize_RAM(), xmu_standardize_SexLim(), xmu_standardize_Simplex(), xmu_start_value_list(), xmu_starts(), xmu_twin_add_WeightMatrices(), xmu_twin_check(), xmu_twin_get_var_names(), xmu_twin_upgrade_selDvs2SelVars()

Examples

```
## Not run:
alg = umx_string_to_algebra(paste(rep("A", nReps), collapse = " %*% "), name = "test_case")

## End(Not run)
```

---

umx_str_chars *Select desired characters from a string*

Description

umx_str_chars returns desired characters of a string

Usage

```
umx_str_chars(what, which)
```

Arguments

what	A string
which	which chars to select out.

Value

- Array of selected characters

References

- tutorials, tutorials

See Also

- umx_explode()

Other String Functions: umx_explode_twin_names(), umx_explode(), umx_grep(), umx_names(), umx_paste_names(), umx_rot(), umx_str_from_object(), umx_trim(), umx

Examples

```
umx_str_chars("myFpassUword", c(3,8))
```

---

umx_str_from_object *Return variable name as a string*

Description

Utility to return an object's name as a string

Usage

```
umx_str_from_object(x)
```

Arguments

x	an object

Value

- name as string

References

- https://www.github.com/tbates/umx

See Also

Other String Functions: umx_explode_twin_names(), umx_explode(), umx_grep(), umx_names(), umx_paste_names(), umx_rot(), umx_str_chars(), umx_trim(), umx

Examples

```
umx_str_from_object(mtcars)
# "mtcars"
```

umx_time *umx_time*

Description

A function to compactly report how long a model took to execute. Comes with some preset styles User can set the format with C-style string formatting.

Usage

```
umx_time(
  x = NA,
  formatStr = c("simple", "std", "custom %H %M %OS3"),
  tz = "GMT",
  autoRun = TRUE
)
```

Arguments

x	A mxModel() or list of models for which to display elapsed time, or 'start' or 'stop'
formatStr	A format string, defining how to show the time (defaults to human readable)
tz	time zone in which the model was executed (defaults to "GMT")
autoRun	If TRUE (default), run the model if it appears not to have been.

Details

The default time format is "simple", which gives only the biggest unit used. i.e., "x seconds" for times under 1 minute. "std" shows time in the format adopted in OpenMx 2.0 e.g. "Wall clock time (HH:MM:SS.hh): 00:00:01.16"

If a list of models is provided, time deltas will also be reported.

If instead of a model the key word "start" is given in x, a start time will be recorded. "stop" gives the time since "start" was called (and clears the timer)

If a model has not been run, umx_time will run it for you.

Value

- invisible time string

References

- https://www.github.com/tbates/umx

See Also

Other Reporting Functions: loadings.MxModel(), umxAPA(), umxFactorScores(), umxGetParameters(), umxParameters(), umxReduce(), umx_aggregate(), umx_names(), umx

Examples

```
require(umx)
umx_time('stop') # alert user stop called when not yet started...
umx_time('stop')
umx_time('start')
data(demoOneFactor)
latents  = c("G")
manifests = names(demoOneFactor)
myData = mxData(cov(demoOneFactor), type = "cov", numObs=500)
m1 = umxRAM("umx_time_example", data = myData,
umxPath(from = latents, to = manifests),
umxPath(var = manifests),
umxPath(var = latents, fixedAt = 1)
)
umx_time(m1) # report time from mxModel
m2 = umxRun(m1)
umx_time(c(m1, m2)) # print comparison table
umx_time('stop') # report the time since timer last started, and restart
umx_time('stop') # report the time since timer was restarted.
```

---

umx_trim *Trim whitespace surrounding a string.*

Description

Returns string without leading or trailing whitespace, like the php function. See also built-in base::trimws() does the same.

Usage

```
umx_trim(string, removeThis = NULL)
```

Arguments

string	to trim
removeThis	if not NULL then this regular expression is removed wherever found in 'string'

Value

- string

References

- https://tbates.github.io, https://github.com/tbates/umx

See Also

base::trimws()

Other String Functions: umx_explode_twin_names(), umx_explode(), umx_grep(), umx_names(), umx_paste_names(), umx_rot(), umx_str_chars(), umx_str_from_object(), umx

Examples

```
umx_trim(" dog") # "dog"
trimws(" dog ", "l") # added by R in v 3.3.0
umx_trim("dog ") # "dog"
umx_trim("\t dog \n") # "dog"
umx_trim("xlsx dog.xlsx", "\\.?xlsx ?") # "dog"
```

---

umx_var *Get variances from a df that might contain some non-numeric columns*

Description

Pass in any dataframe and get variances despite some non-numeric columns. Cells involving these non-numeric columns are set to ordVar (default = 1).

Usage

```
umx_var(
  df,
  format = c("full", "diag", "lower"),
  use = c("complete.obs", "pairwise.complete.obs", "everything", "all.obs",
    "na.or.complete"),
  ordVar = 1,
  digits = NULL,
  strict = TRUE,
  allowCorForFactorCovs = FALSE
)
```

Arguments

df	A dataframe of raw data from which to get variances.
format	to return: options are c("full", "diag", "lower"). Defaults to full, but this is not implemented yet.
use	Passed to cov() - defaults to "complete.obs" (see param default for other options).
ordVar	The value to return at any ordinal columns (defaults to 1).
digits	digits to round output to (Ignored if NULL). Set for easy printing.
strict	Whether to allow non-ordered factors to be processed (default = FALSE (no)).
allowCorForFactorCovs	When ordinal data are present, use heterochoric correlations in affected cells, in place of covariances.

Value

- mxModel()

References

- https://tbates.github.io

See Also

Other Miscellaneous Stats Helpers: FishersMethod(), SE_from_p(), oddsratio(), reliability(), umxCov2cor(), umxHetCor(), umxWeightedAIC(), umx_apply(), umx_cor(), umx_means(), umx_r_test(), umx_round(), umx_scale(), umx

Examples

```
tmp     = mtcars[,1:4]
tmp$cyl = ordered(mtcars$cyl) # ordered factor
tmp$hp  = ordered(mtcars$hp)  # binary factor
umx_var(tmp, format = "diag", ordVar = 1, use = "pair")
tmp2 = tmp[, c(1, 3)]
umx_var(tmp2, format = "diag")
umx_var(tmp2, format = "full")

data(myFADataRaw)
df = myFADataRaw[,c("z1", "z2", "z3")]
df$z1 = mxFactor(df$z1, levels = c(0, 1))
df$z2 = mxFactor(df$z2, levels = c(0, 1))
df$z3 = mxFactor(df$z3, levels = c(0, 1, 2))
umx_var(df, format = "diag")
umx_var(df, format = "full", allowCorForFactorCovs=TRUE)

# Ordinal/continuous mix
data(twinData)
twinData= umx_scale_wide_twin_data(data=twinData,varsToScale="wt",sep= "")
# Cut BMI column to form ordinal obesity variables
obLevels   = c('normal', 'overweight', 'obese')
cuts       = quantile(twinData[, "bmi1"], probs = c(.5, .8), na.rm = TRUE)
twinData$obese1=cut(twinData$bmi1,breaks=c(-Inf,cuts,Inf),labels=obLevels)
twinData$obese2=cut(twinData$bmi2,breaks=c(-Inf,cuts,Inf),labels=obLevels)
# Make the ordinal variables into mxFactors
ordDVs = c("obese1", "obese2")
twinData[, ordDVs] = umxFactor(twinData[, ordDVs])
varStarts = umx_var(twinData[, c(ordDVs, "wt1", "wt2")],
format= "diag", ordVar = 1, use = "pairwise.complete.obs")
```

umx_wide2long *Change data family data from wide (2 twins per row) to long format.*

Description

Just detects the data columns for twin 1, and twin 2, then returns them stacked on top of each other (rbind) with the non-twin specific columns copied for each as well.

Note, zygosity codings differ among labs. One scheme uses 1 = MZFF, 2 = MZMM, 3 = DZFF, 4 = DZMM, 5 = DZOS or DZFM, 6 = DZMF, with 9 = unknown, and then 50, 51,... for siblings.

Typically, OS twins are ordered Female/Male.

Usage

```
umx_wide2long(data, sep = "_T", verbose = FALSE)
```

Arguments

data	a dataframe containing twin data.
sep	the string between the var name and twin suffix, i.e., var_T1 = _T
verbose	Report the non-twin and twin columns (default = FALSE).

Value

- long-format dataframe

See Also

Other Twin Data functions: umx_long2wide(), umx_make_TwinData(), umx_make_twin_data_nice(), umx_residualize(), umx_scale_wide_twin_data(), umx

Examples

```
long = umx_wide2long(data = twinData, sep = "")
long = umx_wide2long(data = twinData, sep = "", verbose = TRUE)
str(long)
str(twinData)
```

umx_write_to_clipboard *umx_write_to_clipboard*

Description

umx_write_to_clipboard writes data to the clipboard

Usage

```
umx_write_to_clipboard(x)
```

Arguments

x	something to paste to the clipboard

Details

Works on Mac. Let me know if it fails on windows or Unix.

Value

None

See Also

Other File Functions: dl_from_dropbox(), umx_file_load_pseudo(), umx_make_sql_from_excel(), umx_move_file(), umx_open(), umx_rename_file(), umx

Examples

```
## Not run:
umx_write_to_clipboard("hello")

## End(Not run)
```

us_skinfold_data *Anthropometric data on twins*

Description

A dataset containing height, weight, BMI, and skin-fold fat measures in several hundred US twin families participating in the MCV Cardiovascular Twin Study (PI Schieken). Biceps and Triceps are folds above and below the upper arm (holding arm palm upward), Calf (fold on the calf muscle), Subscapular (fold over the shoulder blade), Suprailiacal (fold between the hip and ribs).

Usage

```
data(us_skinfold_data)
```

Format

A data frame with 53940 twin families (1 per row) each twin measured on 10 variables.

Details

- *fan* FamilyID (t1=male,t2=female)
- *zyg* Zygosity 1:mzm, 2:mzf, 3:dzm, 4:dzf, 5:dzo
- *ht_T1* Height of twin 1 (cm)
- *wt_T1* Weight of twin 1 (kg)
- *bmi_T1* BMI of twin 1
- *bml_T1* log BMI of twin 1
- *bic_T1* Biceps Skinfold of twin 1
- *caf_T1* Calf Skinfold of twin 1
- *ssc_T1* Subscapular Skinfold of twin 1
- *sil_T1* Suprailiacal Skinfold of twin 1
- *tri_T1* Triceps Skinfold of twin 1
- *ht_T2* Height of twin 2
- *wt_T2* Weight of twin 2
- *bmi_T2* BMI of twin 2
- *bml_T2* log BMI of twin 2
- *bic_T2* Biceps Skinfold of twin 2
- *caf_T2* Calf Skinfold of twin 2
- *ssc_T2* Subscapular Skinfold of twin 2
- *sil_T2* Suprailiacal Skinfold of twin 2
- *tri_T2* Triceps Skinfold of twin 2

References

Moskowitz, W. B., Schwartz, P. F., & Schieken, R. M. (1999). Childhood passive smoking, race, and coronary artery disease risk: the MCV Twin Study. Medical College of Virginia. *Archives of Pediatrics and Adolescent Medicine*, **153**, 446-453. https://www.ncbi.nlm.nih.gov/pubmed/10323623

See Also

Other datasets: Fischbein_wt, GFF, docData, iqdat, umx

Examples

```
data(us_skinfold_data)
str(us_skinfold_data)
par(mfrow = c(1, 2))  # 1 rows and 3 columns
plot(ht_T1 ~ht_T2, ylim = c(130, 165), data = subset(us_skinfold_data, zyg == 1))
plot(ht_T1 ~ht_T2, ylim = c(130, 165), data = subset(us_skinfold_data, zyg == 3))
par(mfrow = c(1, 1))  # back to as it was
```

---

xmuHasSquareBrackets *xmuHasSquareBrackets*

Description

Tests if an input has square brackets

Usage

```
xmuHasSquareBrackets(input)
```

Arguments

input	an input to test

Value

- TRUE/FALSE

See Also

Other xmu internal not for end user: umxModel(), umxRenameMatrix(), umxTwinMaker(), umx_APA_pval(), umx_fun_mean_sd(), umx_get_bracket_addresses(), umx_make(), umx_standardize(), umx_string_to_algebra(), umx, xmuLabel_MATRIX_Model(), xmuLabel_Matrix(), xmuLabel_RAM_Model(), xmuMI(), xmuMakeDeviationThreshold xmuMakeOneHeadedPathsFromPathList(), xmuMakeTwoHeadedPathsFromPathList(), xmuMaxLevels(), xmuMinLevels(), xmuPropagateLabels(), xmuRAM2Ordinal(), xmuTwinSuper_Continuous(), xmuTwinUpgradeMeansToCovariateModel(), xmu_CI_merge(), xmu_CI_stash(), xmu_DF_to_mxData_TypeCov(), xmu_PadAndPruneForDefVars(), xmu_cell_is_on(), xmu_check_levels_identical(), xmu_check_needs_means(), xmu_check_variance(), xmu_clean_label(), xmu_data_missing(), xmu_data_swap_a_block(), xmu_describe_data_WLS(), xmu_dot_make_paths(), xmu_dot_make_residuals(), xmu_dot_maker(), xmu_dot_move_ranks(), xmu_dot_rank_str(), xmu_extract_column(), xmu_get_CI(), xmu_lavaan_process_group() xmu_make_TwinSuperModel(), xmu_make_bin_cont_pair_data(), xmu_make_mxData(), xmu_match.arg(), xmu_name_from_lavaan_str(), xmu_path2twin(), xmu_path_regex(), xmu_safe_run_summary(), xmu_set_sep_from_suffix(), xmu_show_fit_or_comparison(), xmu_simplex_corner(), xmu_standardize_ACEcov() xmu_standardize_ACEv(), xmu_standardize_ACE(), xmu_standardize_CP(), xmu_standardize_IP(), xmu_standardize_RAM(), xmu_standardize_SexLim(), xmu_standardize_Simplex(), xmu_start_value_list(), xmu_starts(), xmu_twin_add_WeightMatrices(), xmu_twin_check(), xmu_twin_get_var_names(), xmu_twin_upgrade_selDvs2SelVars()

Examples

```
xmuHasSquareBrackets("A[1,2]")
```

`xmuLabel_Matrix` *xmuLabel_Matrix (not a user function)*

Description

This function will label all the free parameters in an `mxMatrix()`

Usage

```
xmuLabel_Matrix(
  mx_matrix = NA,
  baseName = NA,
  setfree = FALSE,
  drop = 0,
  jiggle = NA,
  boundDiag = NA,
  suffix = "",
  verbose = TRUE,
  labelFixedCells = FALSE,
  overRideExisting = FALSE
)
```

Arguments

`mx_matrix`	an mxMatrix
`baseName`	A base name for the labels NA
`setfree`	Whether to set free cells FALSE
`drop`	What values to drop 0
`jiggle`	= whether to jiggle start values
`boundDiag`	set diagonal element lbounds to this numeric value (default = NA = ignore)
`suffix`	a string to append to each label
`verbose`	how much feedback to give
`labelFixedCells`	= FALSE
`overRideExisting`	Whether to overRideExisting (Default FALSE)

Details

End users should just call umxLabel()

Purpose: label the cells of an mxMatrix Detail: Defaults to the handy "name_r1c1" where name is the matrix name, and r1c1 = row 1 col 1. Use case: You should not use this: call umxLabel umx:::xmuLabel_Matrix(mxMatrix("Lower", 3, 3, values = 1, name = "a", byrow = TRUE), jiggle = .05, boundDiag = NA); umx:::xmuLabel_Matrix(mxMatrix("Full" , 3, 3, values = 1, name = "a", byrow = TRUE)); umx:::xmuLabel_Matrix(mxMatrix("Symm" , 3, 3, values = 1, name = "a", byrow = TRUE), jiggle = .05, boundDiag = NA); umx:::xmuLabel_Matrix(mxMatrix("Full" , 1, 1, values = 1, name = "a", labels= "data.a")); umx:::xmuLabel_Matrix(mxMatrix("Full" , 1, 1, values = 1, name = "a", labels= "data.a"), overRideExisting = TRUE); umx:::xmuLabel_Matrix(mxMatrix("Full" , 1, 1, values = 1, name = "a", labels= "test"), overRideExisting = TRUE); See also: fit2 = omxSetParameters(fit1, labels = "a_r1c1", free = FALSE, value = 0, name = "drop_a_row1_c1")

Value

- The labeled mxMatrix()

See Also

Other xmu internal not for end user: umxModel(), umxRenameMatrix(), umxTwinMaker(), umx_APA_pval(), umx_fun_mean_sd(), umx_get_bracket_addresses(), umx_make(), umx_standardize(), umx_string_to_algebra(), umx, xmuHasSquareBrackets(), xmuLabel_MATRIX_Model(), xmuLabel_RAM_Model(), xmuMI(), xmuMakeDeviationThresholdsMatrices(), xmuMakeOneHeadedPathsFromPathList(), xmuMakeTwoHeadedPathsFromPa xmuMaxLevels(), xmuMinLevels(), xmuPropagateLabels(), xmuRAM2Ordinal(), xmuTwinSuper_Continuous(), xmuTwinUpgradeMeansToCovariateModel(), xmu_CI_merge(), xmu_CI_stash(), xmu_DF_to_mxData_TypeCov(), xmu_PadAndPruneForDefVars(), xmu_cell_is_on(), xmu_check_levels_identical(), xmu_check_needs_means(), xmu_check_variance(), xmu_clean_label(), xmu_data_missing(), xmu_data_swap_a_block(), xmu_describe_data_WLS(), xmu_dot_make_paths(), xmu_dot_make_residuals(), xmu_dot_maker(), xmu_dot_move_ranks(), xmu_dot_rank_str(), xmu_extract_column(), xmu_get_CI(), xmu_lavaan_process_group() xmu_make_TwinSuperModel(), xmu_make_bin_cont_pair_data(), xmu_make_mxData(), xmu_match.arg(), xmu_name_from_lavaan_str(), xmu_path2twin(), xmu_path_regex(), xmu_safe_run_summary(), xmu_set_sep_from_suffix(), xmu_show_fit_or_comparison(), xmu_simplex_corner(), xmu_standardize_ACEcov() xmu_standardize_ACEv(), xmu_standardize_ACE(), xmu_standardize_CP(), xmu_standardize_IP(), xmu_standardize_RAM(), xmu_standardize_SexLim(), xmu_standardize_Simplex(), xmu_start_value_list(), xmu_starts(), xmu_twin_add_WeightMatrices(), xmu_twin_check(), xmu_twin_get_var_names(), xmu_twin_upgrade_selDvs2SelVars()

---

xmuLabel_MATRIX_Model *xmuLabel_MATRIX_Model (not a user function)*

Description

This function will label all the free parameters in a (non-RAM) OpenMx mxModel() nb: We don't assume what each matrix is for. Instead, the function just sticks labels like "a_r1c1" into each cell i.e., matrix-name + _ + r + rowNumber + c + colNumber

Usage

```
xmuLabel_MATRIX_Model(model, suffix = "", verbose = TRUE)
```

Arguments

model	a matrix-style mxModel to label
suffix	a string to append to each label
verbose	how much feedback to give

Details

End users should just call umxLabel()

Value

- The labeled mxModel()

See Also

Other xmu internal not for end user: umxModel(), umxRenameMatrix(), umxTwinMaker(), umx_APA_pval(), umx_fun_mean_sd(), umx_get_bracket_addresses(), umx_make(), umx_standardize(), umx_string_to_algebra(), umx, xmuHasSquareBrackets(), xmuLabel_Matrix(), xmuLabel_RAM_Model(), xmuMI(), xmuMakeDeviationThresholds xmuMakeOneHeadedPathsFromPathList(), xmuMakeTwoHeadedPathsFromPathList(), xmuMaxLevels(), xmuMinLevels(), xmuPropagateLabels(), xmuRAM2Ordinal(), xmuTwinSuper_Continuous(), xmuTwinUpgradeMeansToCovariateModel(), xmu_CI_merge(), xmu_CI_stash(), xmu_DF_to_mxData_TypeCov(), xmu_PadAndPruneForDefVars(), xmu_cell_is_on(), xmu_check_levels_identical(), xmu_check_needs_means(), xmu_check_variance(), xmu_clean_label(), xmu_data_missing(), xmu_data_swap_a_block(), xmu_describe_data_WLS(), xmu_dot_make_paths(), xmu_dot_make_residuals(), xmu_dot_maker(), xmu_dot_move_ranks(), xmu_dot_rank_str(), xmu_extract_column(), xmu_get_CI(), xmu_lavaan_process_group() xmu_make_TwinSuperModel(), xmu_make_bin_cont_pair_data(), xmu_make_mxData(), xmu_match.arg(), xmu_name_from_lavaan_str(), xmu_path2twin(), xmu_path_regex(), xmu_safe_run_summary(), xmu_set_sep_from_suffix(), xmu_show_fit_or_comparison(), xmu_simplex_corner(), xmu_standardize_ACEcov() xmu_standardize_ACEv(), xmu_standardize_ACE(), xmu_standardize_CP(), xmu_standardize_IP(), xmu_standardize_RAM(), xmu_standardize_SexLim(), xmu_standardize_Simplex(), xmu_start_value_list(), xmu_starts(), xmu_twin_add_WeightMatrices(), xmu_twin_check(), xmu_twin_get_var_names(), xmu_twin_upgrade_selDvs2SelVars()

Examples

```
require(umx)
data(demoOneFactor)
m2 <- mxModel("label_ex",
mxMatrix("Full", 5, 1, values = 0.2, free = TRUE, name = "A"),
mxMatrix("Symm", 1, 1, values = 1.0, free = FALSE, name = "L"),
mxMatrix("Diag", 5, 5, values = 1.0, free = TRUE, name = "U"),
mxAlgebra(A %*% L %*% t(A) + U, name = "R"),
mxExpectationNormal("R", dimnames = names(demoOneFactor)),
mxFitFunctionML(),
mxData(cov(demoOneFactor), type = "cov", numObs=500)
```

```
)
m3 = umx:::xmuLabel_MATRIX_Model(m2)
m4 = umx:::xmuLabel_MATRIX_Model(m2, suffix = "male")
# explore these with omxGetParameters(m4)
```

`xmuLabel_RAM_Model` *xmuLabel_RAM_Model (not a user function)*

Description

This function will label all the free parameters in a RAM `mxModel()`

Usage

```
xmuLabel_RAM_Model(
  model,
  suffix = "",
  labelFixedCells = TRUE,
  overRideExisting = FALSE,
  verbose = FALSE,
  name = NULL
)
```

Arguments

`model`	a RAM mxModel to label
`suffix`	a string to append to each label
`labelFixedCells`	Whether to labelFixedCells (Default TRUE)
`overRideExisting`	Whether to overRideExisting (Default FALSE)
`verbose`	how much feedback to give
`name`	Add optional name parameter to rename returned model (default = leave it along)

Details

End users should just call `umxLabel()`

Value

- The labeled `mxModel()`

See Also

Other xmu internal not for end user: umxModel(), umxRenameMatrix(), umxTwinMaker(), umx_APA_pval(), umx_fun_mean_sd(), umx_get_bracket_addresses(), umx_make(), umx_standardize(), umx_string_to_algebra(), umx, xmuHasSquareBrackets(), xmuLabel_MATRIX_Model(), xmuLabel_Matrix(), xmuMI(), xmuMakeDeviationThresho xmuMakeOneHeadedPathsFromPathList(), xmuMakeTwoHeadedPathsFromPathList(), xmuMaxLevels(), xmuMinLevels(), xmuPropagateLabels(), xmuRAM2Ordinal(), xmuTwinSuper_Continuous(), xmuTwinUpgradeMeansToCovariateModel(), xmu_CI_merge(), xmu_CI_stash(), xmu_DF_to_mxData_TypeCov(), xmu_PadAndPruneForDefVars(), xmu_cell_is_on(), xmu_check_levels_identical(), xmu_check_needs_means(), xmu_check_variance(), xmu_clean_label(), xmu_data_missing(), xmu_data_swap_a_block(), xmu_describe_data_WLS(), xmu_dot_make_paths(), xmu_dot_make_residuals(), xmu_dot_maker(), xmu_dot_move_ranks(), xmu_dot_rank_str(), xmu_extract_column(), xmu_get_CI(), xmu_lavaan_process_group() xmu_make_TwinSuperModel(), xmu_make_bin_cont_pair_data(), xmu_make_mxData(), xmu_match.arg(), xmu_name_from_lavaan_str(), xmu_path2twin(), xmu_path_regex(), xmu_safe_run_summary(), xmu_set_sep_from_suffix(), xmu_show_fit_or_comparison(), xmu_simplex_corner(), xmu_standardize_ACEcov() xmu_standardize_ACEv(), xmu_standardize_ACE(), xmu_standardize_CP(), xmu_standardize_IP(), xmu_standardize_RAM(), xmu_standardize_SexLim(), xmu_standardize_Simplex(), xmu_start_value_list(), xmu_starts(), xmu_twin_add_WeightMatrices(), xmu_twin_check(), xmu_twin_get_var_names(), xmu_twin_upgrade_selDvs2SelVars()

Examples

```
require(umx); data(demoOneFactor)
# raw but no means
m1 <- mxModel("label_ex", mxData(demoOneFactor, type = "raw"), type="RAM",
manifestVars = "x1", latentVars= "G",
umxPath("G", to = "x1"),
umxPath(var = "x1"),
umxPath(var = "G", fixedAt = 1)
)
xmuLabel_RAM_Model(m1)
```

`xmuMakeDeviationThresholdsMatrices` *Make a deviation-based mxRAMObjective for ordinal models.*

Description

Purpose: return a mxRAMObjective(A = "A", S = "S", F = "F", M = "M", thresholds = "thresh"), mxData(df, type = "raw") use-case see: umxMakeThresholdMatrix

Usage

```
xmuMakeDeviationThresholdsMatrices(df, droplevels, verbose)
```

Arguments

df	a dataframe
droplevels	whether to droplevels or not
verbose	how verbose to be

Value

- list of matrices

See Also

Other xmu internal not for end user: umxModel(), umxRenameMatrix(), umxTwinMaker(), umx_APA_pval(), umx_fun_mean_sd(), umx_get_bracket_addresses(), umx_make(), umx_standardize(), umx_string_to_algebra(), umx, xmuHasSquareBrackets(), xmuLabel_MATRIX_Model(), xmuLabel_Matrix(), xmuLabel_RAM_Model(), xmuMI(), xmuMakeOneHeadedPathsFromPathList(), xmuMakeTwoHeadedPathsFromPathList(), xmuMaxLevels(), xmuMinLevels(), xmuPropagateLabels(), xmuRAM2Ordinal(), xmuTwinSuper_Continuous(), xmuTwinUpgradeMeansToCovariateModel(), xmu_CI_merge(), xmu_CI_stash(), xmu_DF_to_mxData_TypeCov(), xmu_PadAndPruneForDefVars(), xmu_cell_is_on(), xmu_check_levels_identical(), xmu_check_needs_means(), xmu_check_variance(), xmu_clean_label(), xmu_data_missing(), xmu_data_swap_a_block(), xmu_describe_data_WLS(), xmu_dot_make_paths(), xmu_dot_make_residuals(), xmu_dot_maker(), xmu_dot_move_ranks(), xmu_dot_rank_str(), xmu_extract_column(), xmu_get_CI(), xmu_lavaan_process_group(), xmu_make_TwinSuperModel(), xmu_make_bin_cont_pair_data(), xmu_make_mxData(), xmu_match.arg(), xmu_name_from_lavaan_str(), xmu_path2twin(), xmu_path_regex(), xmu_safe_run_summary(), xmu_set_sep_from_suffix(), xmu_show_fit_or_comparison(), xmu_simplex_corner(), xmu_standardize_ACEcov(), xmu_standardize_ACEv(), xmu_standardize_ACE(), xmu_standardize_CP(), xmu_standardize_IP(), xmu_standardize_RAM(), xmu_standardize_SexLim(), xmu_standardize_Simplex(), xmu_start_value_list(), xmu_starts(), xmu_twin_add_WeightMatrices(), xmu_twin_check(), xmu_twin_get_var_names(), xmu_twin_upgrade_selDvs2SelVars()

---

xmuMakeOneHeadedPathsFromPathList

xmuMakeOneHeadedPathsFromPathList

Description

Make one-headed paths

Usage

```
xmuMakeOneHeadedPathsFromPathList(sourceList, destinationList)
```

Arguments

sourceList	A sourceList
destinationList	A destinationList

Value

- added items

See Also

Other xmu internal not for end user: umxModel(), umxRenameMatrix(), umxTwinMaker(), umx_APA_pval(), umx_fun_mean_sd(), umx_get_bracket_addresses(), umx_make(), umx_standardize(), umx_string_to_algebra(), umx, xmuHasSquareBrackets(), xmuLabel_MATRIX_Model(), xmuLabel_Matrix(), xmuLabel_RAM_Model(), xmuMI(), xmuMakeDeviationThresholdsMatrices(), xmuMakeTwoHeadedPathsFromPathList(), xmuMaxLevels(), xmuMinLevels(), xmuPropagateLabels(), xmuRAM2Ordinal(), xmuTwinSuper_Continuous(), xmuTwinUpgradeMeansToCovariateModel(), xmu_CI_merge(), xmu_CI_stash(), xmu_DF_to_mxData_TypeCov(), xmu_PadAndPruneForDefVars(), xmu_cell_is_on(), xmu_check_levels_identical(), xmu_check_needs_means(), xmu_check_variance(), xmu_clean_label(), xmu_data_missing(), xmu_data_swap_a_block(), xmu_describe_data_WLS(), xmu_dot_make_paths(), xmu_dot_make_residuals(), xmu_dot_maker(), xmu_dot_move_ranks(), xmu_dot_rank_str(), xmu_extract_column(), xmu_get_CI(), xmu_lavaan_process_group(), xmu_make_TwinSuperModel(), xmu_make_bin_cont_pair_data(), xmu_make_mxData(), xmu_match.arg(), xmu_name_from_lavaan_str(), xmu_path2twin(), xmu_path_regex(), xmu_safe_run_summary(), xmu_set_sep_from_suffix(), xmu_show_fit_or_comparison(), xmu_simplex_corner(), xmu_standardize_ACEcov(), xmu_standardize_ACEv(), xmu_standardize_ACE(), xmu_standardize_CP(), xmu_standardize_IP(), xmu_standardize_RAM(), xmu_standardize_SexLim(), xmu_standardize_Simplex(), xmu_start_value_list(), xmu_starts(), xmu_twin_add_WeightMatrices(), xmu_twin_check(), xmu_twin_get_var_names(), xmu_twin_upgrade_selDvs2SelVars()

---

xmuMakeTwoHeadedPathsFromPathList

xmuMakeTwoHeadedPathsFromPathList

Description

Make two-headed paths

Usage

```
xmuMakeTwoHeadedPathsFromPathList(pathList)
```

Arguments

pathList	A list of paths

Value

- added items

See Also

Other xmu internal not for end user: umxModel(), umxRenameMatrix(), umxTwinMaker(), umx_APA_pval(), umx_fun_mean_sd(), umx_get_bracket_addresses(), umx_make(), umx_standardize(), umx_string_to_algebra(), umx, xmuHasSquareBrackets(), xmuLabel_MATRIX_Model(), xmuLabel_Matrix(), xmuLabel_RAM_Model(), xmuMI(), xmuMakeDeviationThresholdsMatrices(), xmuMakeOneHeadedPathsFromPathList(), xmuMaxLevels(), xmuMinLevels(), xmuPropagateLabels(), xmuRAM2Ordinal(), xmuTwinSuper_Continuous(), xmuTwinUpgradeMeansToCovariateModel(), xmu_CI_merge(), xmu_CI_stash(), xmu_DF_to_mxData_TypeCov(), xmu_PadAndPruneForDefVars(), xmu_cell_is_on(), xmu_check_levels_identical(), xmu_check_needs_means(), xmu_check_variance(), xmu_clean_label(), xmu_data_missing(), xmu_data_swap_a_block(), xmu_describe_data_WLS(), xmu_dot_make_paths(), xmu_dot_make_residuals(), xmu_dot_maker(), xmu_dot_move_ranks(), xmu_dot_rank_str(), xmu_extract_column(), xmu_get_CI(), xmu_lavaan_process_group(), xmu_make_TwinSuperModel(), xmu_make_bin_cont_pair_data(), xmu_make_mxData(), xmu_match.arg(), xmu_name_from_lavaan_str(), xmu_path2twin(), xmu_path_regex(), xmu_safe_run_summary(), xmu_set_sep_from_suffix(), xmu_show_fit_or_comparison(), xmu_simplex_corner(), xmu_standardize_ACEcov(), xmu_standardize_ACEv(), xmu_standardize_ACE(), xmu_standardize_CP(), xmu_standardize_IP(), xmu_standardize_RAM(), xmu_standardize_SexLim(), xmu_standardize_Simplex(), xmu_start_value_list(), xmu_starts(), xmu_twin_add_WeightMatrices(), xmu_twin_check(), xmu_twin_get_var_names(), xmu_twin_upgrade_selDvs2SelVars()

`xmuMaxLevels` *xmuMaxLevels*

Description

Get the max levels from df

Usage

```
xmuMaxLevels(df, what = c("value", "name"))
```

Arguments

df	Dataframe to search through
what	Either "value" or "name" (of the max-level column)

Value

- max number of levels in frame

See Also

Other xmu internal not for end user: umxModel(), umxRenameMatrix(), umxTwinMaker(), umx_APA_pval(), umx_fun_mean_sd(), umx_get_bracket_addresses(), umx_make(), umx_standardize(), umx_string_to_algebra(), umx, xmuHasSquareBrackets(), xmuLabel_MATRIX_Model(), xmuLabel_Matrix(), xmuLabel_RAM_Model(), xmuMI(), xmuMakeDeviationThresholdsMatrices(), xmuMakeOneHeadedPathsFromPathList(), xmuMakeTwoHeadedPathsFromPathList(), xmuMinLevels(), xmuPropagateLabels(), xmuRAM2Ordinal(), xmuTwinSuper_Continuous(), xmuTwinUpgradeMeansToCovariateModel(), xmu_CI_merge(),

xmu_CI_stash(), xmu_DF_to_mxData_TypeCov(), xmu_PadAndPruneForDefVars(), xmu_cell_is_on(), xmu_check_levels_identical(), xmu_check_needs_means(), xmu_check_variance(), xmu_clean_label(), xmu_data_missing(), xmu_data_swap_a_block(), xmu_describe_data_WLS(), xmu_dot_make_paths(), xmu_dot_make_residuals(), xmu_dot_maker(), xmu_dot_move_ranks(), xmu_dot_rank_str(), xmu_extract_column(), xmu_get_CI(), xmu_lavaan_process_group(), xmu_make_TwinSuperModel(), xmu_make_bin_cont_pair_data(), xmu_make_mxData(), xmu_match.arg(), xmu_name_from_lavaan_str(), xmu_path2twin(), xmu_path_regex(), xmu_safe_run_summary(), xmu_set_sep_from_suffix(), xmu_show_fit_or_comparison(), xmu_simplex_corner(), xmu_standardize_ACEcov(), xmu_standardize_ACEv(), xmu_standardize_ACE(), xmu_standardize_CP(), xmu_standardize_IP(), xmu_standardize_RAM(), xmu_standardize_SexLim(), xmu_standardize_Simplex(), xmu_start_value_list(), xmu_starts(), xmu_twin_add_WeightMatrices(), xmu_twin_check(), xmu_twin_get_var_names(), xmu_twin_upgrade_selDvs2SelV

Examples

```
xmuMaxLevels(mtcars) # NA = no ordinal vars
xmuMaxLevels(umxFactor(mtcars))
xmuMaxLevels(umxFactor(mtcars), what = "name")
```

---

xmuMI *xmuMI (not for end users)*

Description

A function to compute and report modifications which would improve fit. You will probably use umxMI() instead

Usage

```
xmuMI(model, vector = TRUE)
```

Arguments

model	an mxModel() to derive modification indices for
vector	= Whether to report the results as a vector default = TRUE

See Also

Other xmu internal not for end user: umxModel(), umxRenameMatrix(), umxTwinMaker(), umx_APA_pval(), umx_fun_mean_sd(), umx_get_bracket_addresses(), umx_make(), umx_standardize(), umx_string_to_algebra(), umx, xmuHasSquareBrackets(), xmuLabel_MATRIX_Model(), xmuLabel_Matrix(), xmuLabel_RAM_Model(), xmuMakeDeviationThresholdsMatrices(), xmuMakeOneHeadedPathsFromPathList(), xmuMakeTwoHeadedPathsFromPa xmuMaxLevels(), xmuMinLevels(), xmuPropagateLabels(), xmuRAM2Ordinal(), xmuTwinSuper_Continuous(), xmuTwinUpgradeMeansToCovariateModel(), xmu_CI_merge(), xmu_CI_stash(), xmu_DF_to_mxData_TypeCov(), xmu_PadAndPruneForDefVars(), xmu_cell_is_on(), xmu_check_levels_identical(), xmu_check_needs_means(), xmu_check_variance(), xmu_clean_label(), xmu_data_missing(), xmu_data_swap_a_block(), xmu_describe_data_WLS(), xmu_dot_make_paths(), xmu_dot_make_residuals(), xmu_dot_maker(), xmu_dot_move_ranks(), xmu_dot_rank_str(), xmu_extract_column(), xmu_get_CI(), xmu_lavaan_process_group()

xmu_make_TwinSuperModel(), xmu_make_bin_cont_pair_data(), xmu_make_mxData(), xmu_match.arg(), xmu_name_from_lavaan_str(), xmu_path2twin(), xmu_path_regex(), xmu_safe_run_summary(), xmu_set_sep_from_suffix(), xmu_show_fit_or_comparison(), xmu_simplex_corner(), xmu_standardize_ACEcov(), xmu_standardize_ACEv(), xmu_standardize_ACE(), xmu_standardize_CP(), xmu_standardize_IP(), xmu_standardize_RAM(), xmu_standardize_SexLim(), xmu_standardize_Simplex(), xmu_start_value_list(), xmu_starts(), xmu_twin_add_WeightMatrices(), xmu_twin_check(), xmu_twin_get_var_names(), xmu_twin_upgrade_selDvs2SelVars()

---

xmuMinLevels *xmuMinLevels*

Description

Get the min levels from df

Usage

```
xmuMinLevels(df, what = c("value", "name"))
```

Arguments

df	Dataframe to search through
what	Either "value" or "name" (of the min-level column)

Value

- min number of levels in frame

See Also

Other xmu internal not for end user: umxModel(), umxRenameMatrix(), umxTwinMaker(), umx_APA_pval(), umx_fun_mean_sd(), umx_get_bracket_addresses(), umx_make(), umx_standardize(), umx_string_to_algebra(), umx, xmuHasSquareBrackets(), xmuLabel_MATRIX_Model(), xmuLabel_Matrix(), xmuLabel_RAM_Model(), xmuMI(), xmuMakeDeviationThresholdsMatrices(), xmuMakeOneHeadedPathsFromPathList(), xmuMakeTwoHeadedPathsFromPathList(), xmuMaxLevels(), xmuPropagateLabels(), xmuRAM2Ordinal(), xmuTwinSuper_Continuous(), xmuTwinUpgradeMeansToCovariateModel(), xmu_CI_merge(), xmu_CI_stash(), xmu_DF_to_mxData_TypeCov(), xmu_PadAndPruneForDefVars(), xmu_cell_is_on(), xmu_check_levels_identical(), xmu_check_needs_means(), xmu_check_variance(), xmu_clean_label(), xmu_data_missing(), xmu_data_swap_a_block(), xmu_describe_data_WLS(), xmu_dot_make_paths(), xmu_dot_make_residuals(), xmu_dot_maker(), xmu_dot_move_ranks(), xmu_dot_rank_str(), xmu_extract_column(), xmu_get_CI(), xmu_lavaan_process_group(), xmu_make_TwinSuperModel(), xmu_make_bin_cont_pair_data(), xmu_make_mxData(), xmu_match.arg(), xmu_name_from_lavaan_str(), xmu_path2twin(), xmu_path_regex(), xmu_safe_run_summary(), xmu_set_sep_from_suffix(), xmu_show_fit_or_comparison(), xmu_simplex_corner(), xmu_standardize_ACEcov(), xmu_standardize_ACEv(), xmu_standardize_ACE(), xmu_standardize_CP(), xmu_standardize_IP(), xmu_standardize_RAM(), xmu_standardize_SexLim(), xmu_standardize_Simplex(), xmu_start_value_list(), xmu_starts(), xmu_twin_add_WeightMatrices(), xmu_twin_check(), xmu_twin_get_var_names(), xmu_twin_upgrade_selDvs2SelV

Examples

```
xmuMinLevels(mtcars) # NA = no ordinal vars
xmuMinLevels(umxFactor(mtcars))
xmuMinLevels(umxFactor(mtcars), what = "name")
```

`xmuPropagateLabels` *xmuPropagateLabels (not a user function)*

Description

You should be calling `umxLabel()`. This function is called by xmuLabel_MATRIX_Model

Usage

```
xmuPropagateLabels(model, suffix = "", verbose = TRUE)
```

Arguments

`model`	a model to label
`suffix`	a string to append to each label
`verbose`	whether to say what is being done

Value

- `mxModel()`

See Also

Other xmu internal not for end user: `umxModel()`, `umxRenameMatrix()`, `umxTwinMaker()`, `umx_APA_pval()`, `umx_fun_mean_sd()`, `umx_get_bracket_addresses()`, `umx_make()`, `umx_standardize()`, `umx_string_to_algebra()`, `umx`, `xmuHasSquareBrackets()`, `xmuLabel_MATRIX_Model()`, `xmuLabel_Matrix()`, `xmuLabel_RAM_Model()`, `xmuMI()`, `xmuMakeDeviationThresholdsMatrices()`, `xmuMakeOneHeadedPathsFromPathList()`, `xmuMakeTwoHeadedPathsFromPathList()`, `xmuMaxLevels()`, `xmuMinLevels()`, `xmuRAM2Ordinal()`, `xmuTwinSuper_Continuous()`, `xmuTwinUpgradeMeansToCovariateModel()`, `xmu_CI_merge()`, `xmu_CI_stash()`, `xmu_DF_to_mxData_TypeCov()`, `xmu_PadAndPruneForDefVars()`, `xmu_cell_is_on()`, `xmu_check_levels_identical()`, `xmu_check_needs_means()`, `xmu_check_variance()`, `xmu_clean_label()`, `xmu_data_missing()`, `xmu_data_swap_a_block()`, `xmu_describe_data_WLS()`, `xmu_dot_make_paths()`, `xmu_dot_make_residuals()`, `xmu_dot_maker()`, `xmu_dot_move_ranks()`, `xmu_dot_rank_str()`, `xmu_extract_column()`, `xmu_get_CI()`, `xmu_lavaan_process_group()`, `xmu_make_TwinSuperModel()`, `xmu_make_bin_cont_pair_data()`, `xmu_make_mxData()`, `xmu_match.arg()`, `xmu_name_from_lavaan_str()`, `xmu_path2twin()`, `xmu_path_regex()`, `xmu_safe_run_summary()`, `xmu_set_sep_from_suffix()`, `xmu_show_fit_or_comparison()`, `xmu_simplex_corner()`, `xmu_standardize_ACEcov()`, `xmu_standardize_ACEv()`, `xmu_standardize_ACE()`, `xmu_standardize_CP()`, `xmu_standardize_IP()`, `xmu_standardize_RAM()`, `xmu_standardize_SexLim()`, `xmu_standardize_Simplex()`, `xmu_start_value_list()`, `xmu_starts()`, `xmu_twin_add_WeightMatrices()`, `xmu_twin_check()`, `xmu_twin_get_var_names()`, `xmu_twin_upgrade_selDvs2Sel`

Examples

```
require(umx)
data(demoOneFactor)
latents  = c("G")
manifests = names(demoOneFactor)
m1 = mxModel("propage_example", type = "RAM",
manifestVars = manifests, latentVars = latents,
mxPath(from = latents  , to = manifests),
mxPath(from = manifests, arrows = 2),
mxPath(from = latents  , arrows = 2, free = FALSE, values = 1.0),
mxData(cov(demoOneFactor), type = "cov", numObs=500)
)

m1 = umx:::xmuPropagateLabels(m1, suffix = "MZ")
```

---

xmuRAM2Ordinal *xmuRAM2Ordinal*

Description

xmuRAM2Ordinal: Convert a RAM model whose data contain ordinal variables to a threshold-based model

Usage

```
xmuRAM2Ordinal(model, verbose = TRUE, name = NULL)
```

Arguments

model	An RAM model to add thresholds too.
verbose	Tell the user what was added and why (Default = TRUE).
name	= A new name for the modified model. Default (NULL) = leave it as is).

Value

- mxModel()

See Also

- umxRAM()

Other xmu internal not for end user: umxModel(), umxRenameMatrix(), umxTwinMaker(), umx_APA_pval(), umx_fun_mean_sd(), umx_get_bracket_addresses(), umx_make(), umx_standardize(), umx_string_to_algebra(), umx, xmuHasSquareBrackets(), xmuLabel_MATRIX_Model(), xmuLabel_Matrix(), xmuLabel_RAM_Model(), xmuMI(), xmuMakeDeviationThresholdsMatrices(), xmuMakeOneHeadedPathsFromPathList(), xmuMakeTwoHeadedPathsFromPathList(), xmuMaxLevels(), xmuMinLevels(), xmuPropagateLabels(), xmuTwinSuper_Continuous(), xmuTwinUpgradeMeansToCovariateModel(), xmu_CI_merge(), xmu_CI_stash(), xmu_DF_to_mxData_TypeCov(), xmu_PadAndPruneForDefVars(), xmu_cell_is_on(),

xmu_check_levels_identical(), xmu_check_needs_means(), xmu_check_variance(), xmu_clean_label(), xmu_data_missing(), xmu_data_swap_a_block(), xmu_describe_data_WLS(), xmu_dot_make_paths(), xmu_dot_make_residuals(), xmu_dot_maker(), xmu_dot_move_ranks(), xmu_dot_rank_str(), xmu_extract_column(), xmu_get_CI(), xmu_lavaan_process_group(), xmu_make_TwinSuperModel(), xmu_make_bin_cont_pair_data(), xmu_make_mxData(), xmu_match.arg(), xmu_name_from_lavaan_str(), xmu_path2twin(), xmu_path_regex(), xmu_safe_run_summary(), xmu_set_sep_from_suffix(), xmu_show_fit_or_comparison(), xmu_simplex_corner(), xmu_standardize_ACEcov(), xmu_standardize_ACEv(), xmu_standardize_ACE(), xmu_standardize_CP(), xmu_standardize_IP(), xmu_standardize_RAM(), xmu_standardize_SexLim(), xmu_standardize_Simplex(), xmu_start_value_list(), xmu_starts(), xmu_twin_add_WeightMatrices(), xmu_twin_check(), xmu_twin_get_var_names(), xmu_twin_upgrade_selDvs2Sel

Examples

```
## Not run:
data(twinData)
# Cut to form category of 20% obese subjects
obesityLevels   = c('normal', 'obese')
cutPoints       = quantile(twinData[, "bmi1"], probs = .2, na.rm = TRUE)
twinData$obese1 = cut(twinData$bmi1, breaks = c(-Inf, cutPoints, Inf), labels = obesityLevels)
twinData$obese2 = cut(twinData$bmi2, breaks = c(-Inf, cutPoints, Inf), labels = obesityLevels)
ordDVs = c("obese1", "obese2")
twinData[, ordDVs] = umxFactor(twinData[, ordDVs])
mzData = twinData[twinData$zygosity %in% "MZFF",]
m1 = umxRAM("tim", data = mzData,
umxPath("bmi1", with = "bmi2"),
umxPath(v.m.= c("bmi1", "bmi2"))
)

m1 = umxRAM("tim", data = mzData,
umxPath("obese1", with = "obese2"),
umxPath(v.m.= c("obese1", "obese2"))
)

## End(Not run)
```

---

xmuTwinSuper_Continuous

Create core of twin model for all-continuous data.

Description

Sets up top, MZ and DZ submodels with a means model, data, and expectation for all-continuous data. called by xmu_make_TwinSuperModel().

Usage

```
xmuTwinSuper_Continuous(
  name = NULL,
  fullVars,
```

```
  fullCovs = NULL,
  sep,
  mzData,
  dzData,
  equateMeans,
  type,
  allContinuousMethod,
  nSib
)
```

Arguments

name	The name of the supermodel
fullVars	Full Variable names (wt_T1)
fullCovs	Full Covariate names (age_T1)
sep	default "_T"
mzData	An mxData object containing the MZ data
dzData	An mxData object containing the DZ data
equateMeans	Whether to equate the means across twins (default TRUE)
type	type
allContinuousMethod	allContinuousMethod
nSib	nSib

Value

- A twin model

See Also

- xmu_make_TwinSuperModel()

Other xmu internal not for end user: umxModel(), umxRenameMatrix(), umxTwinMaker(), umx_APA_pval(), umx_fun_mean_sd(), umx_get_bracket_addresses(), umx_make(), umx_standardize(), umx_string_to_algebra(), umx, xmuHasSquareBrackets(), xmuLabel_MATRIX_Model(), xmuLabel_Matrix(), xmuLabel_RAM_Model(), xmuMI(), xmuMakeDeviationThresholdsMatrices(), xmuMakeOneHeadedPathsFromPathList(), xmuMakeTwoHeadedPathsFromPathList(), xmuMaxLevels(), xmuMinLevels(), xmuPropagateLabels(), xmuRAM2Ordinal(), xmuTwinUpgradeMeansToCovariateModel(), xmu_CI_merge(), xmu_CI_stash(), xmu_DF_to_mxData_TypeCov(), xmu_PadAndPruneForDefVars(), xmu_cell_is_on(), xmu_check_levels_identical(), xmu_check_needs_means(), xmu_check_variance(), xmu_clean_label(), xmu_data_missing(), xmu_data_swap_a_block(), xmu_describe_data_WLS(), xmu_dot_make_paths(), xmu_dot_make_residuals(), xmu_dot_maker(), xmu_dot_move_ranks(), xmu_dot_rank_str(), xmu_extract_column(), xmu_get_CI(), xmu_lavaan_process_group(), xmu_make_TwinSuperModel(), xmu_make_bin_cont_pair_data(), xmu_make_mxData(), xmu_match.arg(), xmu_name_from_lavaan_str(), xmu_path2twin(), xmu_path_regex(), xmu_safe_run_summary(), xmu_set_sep_from_suffix(), xmu_show_fit_or_comparison(), xmu_simplex_corner(), xmu_standardize_ACEcov(), xmu_standardize_ACEv(), xmu_standardize_ACE(), xmu_standardize_CP(), xmu_standardize_IP(), xmu_standardize_RAM(), xmu_standardize_SexLim(), xmu_standardize_Simplex(), xmu_start_value_list(), xmu_starts(), xmu_twin_add_WeightMatrices(), xmu_twin_check(), xmu_twin_get_var_names(), xmu_twin_upgrade_selDvs2SelVars()

Examples

```
# xmuTwinSuper_Continuous(name="twin_super", selVars = selVars, selCovs = selCovs,
#    mzData = mzData, dzData = dzData, equateMeans = TRUE, type = type,
#    allContinuousMethod = allContinuousMethod, nSib= nSib, sep = "_T"
# )
```

`xmuTwinUpgradeMeansToCovariateModel`

Not for end-users: Add a means model with covariates to a twin model

---

Description

Does the following to `model` (i.e., a umx top/MZ/DZ supermodel):

1. Change `top.expMeans` to `top.intercept`.
2. Create `top.meansBetas` for beta weights in rows (of covariates) and columns for each variable.
3. Add matrices for each twin's data.cov vars (matrixes are called `T1DefVars`).
4. Switch `mxExpectationNormal` in each data group to point to the local `expMean`.
5. Add "expMean" algebra to each data group.

- `grp.expMean` sums `top.intercept` and grp.DefVars %*% top.meansBetas for each twin.

Usage

```
xmuTwinUpgradeMeansToCovariateModel(model, fullVars, fullCovs, sep)
```

Arguments

`model`	The `umxSuperModel()` we are modifying (must have `MZ` `DZ` and `top` submodels)
`fullVars`	the FULL names of manifest variables
`fullCovs`	the FULL names of definition variables
`sep`	How twin variable names have been expanded, e.g. "_T".

Details

In umx models with no covariates, means live in `top$expMean`

Value

- model, now with means model extended to covariates.

See Also

- called by xmuTwinSuper_Continuous()

Other xmu internal not for end user: umxModel(), umxRenameMatrix(), umxTwinMaker(), umx_APA_pval(), umx_fun_mean_sd(), umx_get_bracket_addresses(), umx_make(), umx_standardize(), umx_string_to_algebra(), umx, xmuHasSquareBrackets(), xmuLabel_MATRIX_Model(), xmuLabel_Matrix(), xmuLabel_RAM_Model(), xmuMI(), xmuMakeDeviationThresholdsMatrices(), xmuMakeOneHeadedPathsFromPathList(), xmuMakeTwoHeadedPathsFromPathList(), xmuMaxLevels(), xmuMinLevels(), xmuPropagateLabels(), xmuRAM2Ordinal(), xmuTwinSuper_Continuous(), xmu_CI_merge(), xmu_CI_stash(), xmu_DF_to_mxData_TypeCov(), xmu_PadAndPruneForDefVars(), xmu_cell_is_on(), xmu_check_levels_identical(), xmu_check_needs_means(), xmu_check_variance(), xmu_clean_label(), xmu_data_missing(), xmu_data_swap_a_block(), xmu_describe_data_WLS(), xmu_dot_make_paths(), xmu_dot_make_residuals(), xmu_dot_maker(), xmu_dot_move_ranks(), xmu_dot_rank_str(), xmu_extract_column(), xmu_get_CI(), xmu_lavaan_process_group(), xmu_make_TwinSuperModel(), xmu_make_bin_cont_pair_data(), xmu_make_mxData(), xmu_match.arg(), xmu_name_from_lavaan_str(), xmu_path2twin(), xmu_path_regex(), xmu_safe_run_summary(), xmu_set_sep_from_suffix(), xmu_show_fit_or_comparison(), xmu_simplex_corner(), xmu_standardize_ACEcov(), xmu_standardize_ACEv(), xmu_standardize_ACE(), xmu_standardize_CP(), xmu_standardize_IP(), xmu_standardize_RAM(), xmu_standardize_SexLim(), xmu_standardize_Simplex(), xmu_start_value_list(), xmu_starts(), xmu_twin_add_WeightMatrices(), xmu_twin_check(), xmu_twin_get_var_names(), xmu_twin_upgrade_selDvs2SelVars()

Examples

```
## Not run:
data(twinData) # ?twinData from Australian twins.
twinData[, c("ht1", "ht2")] = twinData[, c("ht1", "ht2")] * 10
mzData = twinData[twinData$zygosity %in% "MZFF", ]
dzData = twinData[twinData$zygosity %in% "DZFF", ]
# m1 = umxACE(selDVs= "ht", sep= "", dzData= dzData, mzData= mzData, autoRun= FALSE)
# m2 = xmuTwinUpgradeMeansToCovariateModel(m1, fullVars = c("ht1", "ht2"),
#  fullCovs = c("age1", "sex1", "age2", "sex2"), sep = "")
#

## End(Not run)
```

---

xmu_cell_is_on *Return whether a cell is in a set location of a matrix*

Description

Helper to determine is a cell is in a set location of a matrix or not. Left is useful for, e.g. twin means matrices.

Usage

```
xmu_cell_is_on(
  r,
```

```
  c,
  where = c("diag", "lower", "lower_inc", "upper", "upper_inc", "any", "left"),
  mat = NULL
)
```

Arguments

r	which row the cell is on.
c	which column the cell is in.
where	the location (any, diag, lower or upper (or _inc) or left).
mat	(optionally) provide matrix to check dimensions against r and c.

Value

- mxModel()

References

- https://github.com/tbates/umx, https://tbates.github.io

See Also

- umxLabel()

Other xmu internal not for end user: umxModel(), umxRenameMatrix(), umxTwinMaker(), umx_APA_pval(), umx_fun_mean_sd(), umx_get_bracket_addresses(), umx_make(), umx_standardize(), umx_string_to_algebra(), umx, xmuHasSquareBrackets(), xmuLabel_MATRIX_Model(), xmuLabel_Matrix(), xmuLabel_RAM_Model(), xmuMI(), xmuMakeDeviationThresholdsMatrices(), xmuMakeOneHeadedPathsFromPathList(), xmuMakeTwoHeadedPathsFromPathList(), xmuMaxLevels(), xmuMinLevels(), xmuPropagateLabels(), xmuRAM2Ordinal(), xmuTwinSuper_Continuous(), xmuTwinUpgradeMeansToCovariateModel(), xmu_CI_merge(), xmu_CI_stash(), xmu_DF_to_mxData_TypeCov(), xmu_PadAndPruneForDefVars(), xmu_check_levels_identical(), xmu_check_needs_means(), xmu_check_variance(), xmu_clean_label(), xmu_data_missing(), xmu_data_swap_a_block(), xmu_describe_data_WLS(), xmu_dot_make_paths(), xmu_dot_make_residuals(), xmu_dot_maker(), xmu_dot_move_ranks(), xmu_dot_rank_str(), xmu_extract_column(), xmu_get_CI(), xmu_lavaan_process_group(), xmu_make_TwinSuperModel(), xmu_make_bin_cont_pair_data(), xmu_make_mxData(), xmu_match.arg(), xmu_name_from_lavaan_str(), xmu_path2twin(), xmu_path_regex(), xmu_safe_run_summary(), xmu_set_sep_from_suffix(), xmu_show_fit_or_comparison(), xmu_simplex_corner(), xmu_standardize_ACEcov(), xmu_standardize_ACEv(), xmu_standardize_ACE(), xmu_standardize_CP(), xmu_standardize_IP(), xmu_standardize_RAM(), xmu_standardize_SexLim(), xmu_standardize_Simplex(), xmu_start_value_list(), xmu_starts(), xmu_twin_add_WeightMatrices(), xmu_twin_check(), xmu_twin_get_var_names(), xmu_twin_upgrade_selDvs2SelV

Examples

```
xmu_cell_is_on(r = 3, c = 3, "lower")
xmu_cell_is_on(r = 3, c = 3, "lower_inc")
xmu_cell_is_on(r = 3, c = 3, "upper")
xmu_cell_is_on(r = 3, c = 3, "upper_inc")
xmu_cell_is_on(r = 3, c = 3, "diag")
xmu_cell_is_on(r = 2, c = 3, "diag")
```

```
xmu_cell_is_on(r = 3, c = 3, "any")
a_cp = umxMatrix("a_cp", "Lower", 3, 3, free = TRUE, values = 1:6)
xmu_cell_is_on(r = 3, c = 3, "left", mat = a_cp)
## Not run:
# test stopping
xmu_cell_is_on(r=4,c = 3, "any", mat = a_cp)

## End(Not run)
```

`xmu_check_levels_identical` *xmu_check_levels_identical*

Description

Just checks that the factor levels for twins 1 and 2 are the same

Usage

```
xmu_check_levels_identical(df, selDVs, sep, action = c("stop", "ignore"))
```

Arguments

`df`	data.frame containing the data
`selDVs`	base names of variables (without suffixes)
`sep`	text-constant separating base variable names the twin index (1:2)
`action`	if unequal levels found: c("stop", "ignore")

Value

None

See Also

Other xmu internal not for end user: `umxModel()`, `umxRenameMatrix()`, `umxTwinMaker()`, `umx_APA_pval()`, `umx_fun_mean_sd()`, `umx_get_bracket_addresses()`, `umx_make()`, `umx_standardize()`, `umx_string_to_algebra()`, `umx`, `xmuHasSquareBrackets()`, `xmuLabel_MATRIX_Model()`, `xmuLabel_Matrix()`, `xmuLabel_RAM_Model()`, `xmuMI()`, `xmuMakeDeviationThresholdsMatrices()`, `xmuMakeOneHeadedPathsFromPathList()`, `xmuMakeTwoHeadedPathsFromPathList()`, `xmuMaxLevels()`, `xmuMinLevels()`, `xmuPropagateLabels()`, `xmuRAM2Ordinal()`, `xmuTwinSuper_Continuous()`, `xmuTwinUpgradeMeansToCovariateModel()`, `xmu_CI_merge()`, `xmu_CI_stash()`, `xmu_DF_to_mxData_TypeCov()`, `xmu_PadAndPruneForDefVars()`, `xmu_cell_is_on()`, `xmu_check_needs_means()`, `xmu_check_variance()`, `xmu_clean_label()`, `xmu_data_missing()`, `xmu_data_swap_a_block()`, `xmu_describe_data_WLS()`, `xmu_dot_make_paths()`, `xmu_dot_make_residuals()`, `xmu_dot_maker()`, `xmu_dot_move_ranks()`, `xmu_dot_rank_str()`, `xmu_extract_column()`, `xmu_get_CI()`, `xmu_lavaan_process_group()`, `xmu_make_TwinSuperModel()`, `xmu_make_bin_cont_pair_data()`, `xmu_make_mxData()`, `xmu_match.arg()`, `xmu_name_from_lavaan_str()`, `xmu_path2twin()`, `xmu_path_regex()`, `xmu_safe_run_summary()`, `xmu_set_sep_from_suffix()`,

xmu_show_fit_or_comparison(), xmu_simplex_corner(), xmu_standardize_ACEcov(), xmu_standardize_ACEv(), xmu_standardize_ACE(), xmu_standardize_CP(), xmu_standardize_IP(), xmu_standardize_RAM(), xmu_standardize_SexLim(), xmu_standardize_Simplex(), xmu_start_value_list(), xmu_starts(), xmu_twin_add_WeightMatrices(), xmu_twin_check(), xmu_twin_get_var_names(), xmu_twin_upgrade_selDvs2SelV

Examples

```
require(umx)
data(twinData)
baseNames = c("bmi")
selDVs = umx_paste_names(baseNames, "", 1:2)
tmp = twinData[, selDVs]
tmp$bmi1[tmp$bmi1 <= 22] = 22
tmp$bmi2[tmp$bmi2 <= 22] = 22
xmu_check_levels_identical(umxFactor(tmp, sep = ""), selDVs = baseNames, sep = "")
## Not run:
xmu_check_levels_identical(umxFactor(tmp), selDVs = baseNames, sep = "")

## End(Not run)
```

---

xmu_check_needs_means *Check data to see if model needs means.*

Description

Check data to see if model needs means.

Usage

```
xmu_check_needs_means(
  data,
  type = c("Auto", "FIML", "cov", "cor", "WLS", "DWLS", "ULS"),
  allContinuousMethod = c("cumulants", "marginals")
)
```

Arguments

data	mxData() to check.
type	of the data requested by the model.
allContinuousMethod	How data will be processed if used for WLS.

Value

- T/F

See Also

- xmu_make_mxData()

Other xmu internal not for end user: umxModel(), umxRenameMatrix(), umxTwinMaker(), umx_APA_pval(), umx_fun_mean_sd(), umx_get_bracket_addresses(), umx_make(), umx_standardize(), umx_string_to_algebra(), umx, xmuHasSquareBrackets(), xmuLabel_MATRIX_Model(), xmuLabel_Matrix(), xmuLabel_RAM_Model(), xmuMI(), xmuMakeDeviationThresholdsMatrices(), xmuMakeOneHeadedPathsFromPathList(), xmuMakeTwoHeadedPathsFromPathList(), xmuMaxLevels(), xmuMinLevels(), xmuPropagateLabels(), xmuRAM2Ordinal(), xmuTwinSuper_Continuous(), xmuTwinUpgradeMeansToCovariateModel(), xmu_CI_merge(), xmu_CI_stash(), xmu_DF_to_mxData_TypeCov(), xmu_PadAndPruneForDefVars(), xmu_cell_is_on(), xmu_check_levels_identical(), xmu_check_variance(), xmu_clean_label(), xmu_data_missing(), xmu_data_swap_a_block(), xmu_describe_data_WLS(), xmu_dot_make_paths(), xmu_dot_make_residuals(), xmu_dot_maker(), xmu_dot_move_ranks(), xmu_dot_rank_str(), xmu_extract_column(), xmu_get_CI(), xmu_lavaan_process_group(), xmu_make_TwinSuperModel(), xmu_make_bin_cont_pair_data(), xmu_make_mxData(), xmu_match.arg(), xmu_name_from_lavaan_str(), xmu_path2twin(), xmu_path_regex(), xmu_safe_run_summary(), xmu_set_sep_from_suffix(), xmu_show_fit_or_comparison(), xmu_simplex_corner(), xmu_standardize_ACEcov(), xmu_standardize_ACEv(), xmu_standardize_ACE(), xmu_standardize_CP(), xmu_standardize_IP(), xmu_standardize_RAM(), xmu_standardize_SexLim(), xmu_standardize_Simplex(), xmu_start_value_list(), xmu_starts(), xmu_twin_add_WeightMatrices(), xmu_twin_check(), xmu_twin_get_var_names(), xmu_twin_upgrade_selDvs2Sel\

Examples

```
xmu_check_needs_means(mtcars, type = "Auto")
xmu_check_needs_means(mtcars, type = "FIML")
# xmu_check_needs_means(mtcars, type = "cov")
# xmu_check_needs_means(mtcars, type = "cor")

# TRUE - marginals means means
xmu_check_needs_means(mtcars, type = "WLS", allContinuousMethod= "marginals")
xmu_check_needs_means(mtcars, type = "ULS", allContinuousMethod= "marginals")
xmu_check_needs_means(mtcars, type = "DWLS", allContinuousMethod= "marginals")

# ================================
# = Provided as an mxData object =
# ================================
tmp = mxData(mtcars, type="raw")
xmu_check_needs_means(tmp, type = "FIML") # TRUE
xmu_check_needs_means(tmp, type = "ULS", allContinuousMethod= "cumulants") #FALSE
# TRUE - means with marginals
xmu_check_needs_means(tmp, type = "WLS", allContinuousMethod= "marginals")
tmp = mxData(cov(mtcars), type="cov", numObs= 100)
# Should catch this can't be type FIML
xmu_check_needs_means(tmp) # FALSE
tmp = mxData(cov(mtcars), means = umx_means(mtcars), type="cov", numObs= 100)
xmu_check_needs_means(tmp) # TRUE

# ======================
# = One var is a factor =
# ======================
tmp = mtcars
```

```
tmp$cyl = factor(tmp$cyl)
xmu_check_needs_means(tmp, allContinuousMethod= "cumulants") # TRUE
xmu_check_needs_means(tmp, allContinuousMethod= "marginals") # TRUE - always has means
```

`xmu_check_variance` *Check the minimum variance in data frame*

Description

Check that each variable exceeds a minimum variance and all are on compatible scales. Let the user know what to do if not.

Usage

```
xmu_check_variance(
  data,
  minVar = umx_set_data_variance_check(silent = T)$minVar,
  maxVarRatio = umx_set_data_variance_check(silent = T)$maxVarRatio
)
```

Arguments

`data`	the data frame to check
`minVar`	Minimum allowed variance in variables before warning user variances differ too much.
`maxVarRatio`	Maximum allowed ratio of variance in data before warning user variances differ too much.

Value

None

See Also

Other xmu internal not for end user: `umxModel()`, `umxRenameMatrix()`, `umxTwinMaker()`, `umx_APA_pval()`, `umx_fun_mean_sd()`, `umx_get_bracket_addresses()`, `umx_make()`, `umx_standardize()`, `umx_string_to_algebra()`, `umx`, `xmuHasSquareBrackets()`, `xmuLabel_MATRIX_Model()`, `xmuLabel_Matrix()`, `xmuLabel_RAM_Model()`, `xmuMI()`, `xmuMakeDeviationThresholdsMatrices()`, `xmuMakeOneHeadedPathsFromPathList()`, `xmuMakeTwoHeadedPathsFromPathList()`, `xmuMaxLevels()`, `xmuMinLevels()`, `xmuPropagateLabels()`, `xmuRAM2Ordinal()`, `xmuTwinSuper_Continuous()`, `xmuTwinUpgradeMeansToCovariateModel()`, `xmu_CI_merge()`, `xmu_CI_stash()`, `xmu_DF_to_mxData_TypeCov()`, `xmu_PadAndPruneForDefVars()`, `xmu_cell_is_on()`, `xmu_check_levels_identical()`, `xmu_check_needs_means()`, `xmu_clean_label()`, `xmu_data_missing()`, `xmu_data_swap_a_block()`, `xmu_describe_data_WLS()`, `xmu_dot_make_paths()`, `xmu_dot_make_residuals()`, `xmu_dot_maker()`, `xmu_dot_move_ranks()`, `xmu_dot_rank_str()`, `xmu_extract_column()`, `xmu_get_CI()`, `xmu_lavaan_process_group()`, `xmu_make_TwinSuperModel()`, `xmu_make_bin_cont_pair_data()`, `xmu_make_mxData()`, `xmu_match.arg()`, `xmu_name_from_lavaan_str()`, `xmu_path2twin()`, `xmu_path_regex()`, `xmu_safe_run_summary()`, `xmu_set_sep_from_suffix()`,

xmu_show_fit_or_comparison(), xmu_simplex_corner(), xmu_standardize_ACEcov(), xmu_standardize_ACEv(), xmu_standardize_ACE(), xmu_standardize_CP(), xmu_standardize_IP(), xmu_standardize_RAM(), xmu_standardize_SexLim(), xmu_standardize_Simplex(), xmu_start_value_list(), xmu_starts(), xmu_twin_add_WeightMatrices(), xmu_twin_check(), xmu_twin_get_var_names(), xmu_twin_upgrade_selDvs2Sel\

Examples

```
data(twinData)
xmu_check_variance(twinData[, c("wt1", "ht1", "wt2", "ht2")])
twinData[,c("ht1", "ht2")]= twinData[,c("ht1", "ht2")] * 100
xmu_check_variance(twinData[, c("wt1", "ht1", "wt2", "ht2")])
```

---

xmu_CI_merge *xmu_CI_merge*

Description

if you compute some CIs in one model and some in another (copy of the same model, perhaps to get some parallelism), this is a simple helper to kludge them together.

Usage

```
xmu_CI_merge(m1, m2)
```

Arguments

m1	first copy of the model
m2	second copy of the model

Value

- [mxModel()]

References

- <https://www.github.com/tbates/umx>

See Also

Other xmu internal not for end user: umxModel(), umxRenameMatrix(), umxTwinMaker(), umx_APA_pval(), umx_fun_mean_sd(), umx_get_bracket_addresses(), umx_make(), umx_standardize(), umx_string_to_algebra(), umx, xmuHasSquareBrackets(), xmuLabel_MATRIX_Model(), xmuLabel_Matrix(), xmuLabel_RAM_Model(), xmuMI(), xmuMakeDeviationThresholdsMatrices(), xmuMakeOneHeadedPathsFromPathList(), xmuMakeTwoHeadedPathsFromPathList(), xmuMaxLevels(), xmuMinLevels(), xmuPropagateLabels(), xmuRAM2Ordinal(), xmuTwinSuper_Continuous(), xmuTwinUpgradeMeansToCovariateModel(), xmu_CI_stash(), xmu_DF_to_mxData_TypeCov(), xmu_PadAndPruneForDefVars(), xmu_cell_is_on(), xmu_check_levels_identical(), xmu_check_needs_means(), xmu_check_variance(), xmu_clean_label(), xmu_data_missing(), xmu_data_swap_a_block(), xmu_describe_data_WLS(), xmu_dot_make_paths(),

xmu_dot_make_residuals(), xmu_dot_maker(), xmu_dot_move_ranks(), xmu_dot_rank_str(), xmu_extract_column(), xmu_get_CI(), xmu_lavaan_process_group(), xmu_make_TwinSuperModel(), xmu_make_bin_cont_pair_data(), xmu_make_mxData(), xmu_match.arg(), xmu_name_from_lavaan_str(), xmu_path2twin(), xmu_path_regex(), xmu_safe_run_summary(), xmu_set_sep_from_suffix(), xmu_show_fit_or_comparison(), xmu_simplex_corner(), xmu_standardize_ACEcov(), xmu_standardize_ACEv(), xmu_standardize_ACE(), xmu_standardize_CP(), xmu_standardize_IP(), xmu_standardize_RAM(), xmu_standardize_SexLim(), xmu_standardize_Simplex(), xmu_start_value_list(), xmu_starts(), xmu_twin_add_WeightMatrices(), xmu_twin_check(), xmu_twin_get_var_names(), xmu_twin_upgrade_selDvs2Sel\

Examples

```
## Not run:
xmu_CI_merge(m1, m2)

## End(Not run)
```

`xmu_CI_stash` *Stash the CI values of a model as strings in the values of the model*

Description

Stash formatted CIs (e.g. ".1 [-.1, .3]") as strings, *overwriting* the parameter values of the model.

Usage

```
xmu_CI_stash(model, digits = 3, dropZeros = FALSE, stdAlg2mat = TRUE)
```

Arguments

`model`	An `mxModel()` to get CIs from.
`digits`	rounding.
`dropZeros`	makes strings for failed CIs?
`stdAlg2mat`	treat std as algebra: stash in non std matrix.

Details

I might change this to a lookup-function that gets a CI string if one exists.

Value

- `mxModel()`

References

- https://github.com/tbates/umx

See Also

- umxConfint(), xmu_get_CI()

Other xmu internal not for end user: umxModel(), umxRenameMatrix(), umxTwinMaker(), umx_APA_pval(), umx_fun_mean_sd(), umx_get_bracket_addresses(), umx_make(), umx_standardize(), umx_string_to_algebra(), umx, xmuHasSquareBrackets(), xmuLabel_MATRIX_Model(), xmuLabel_Matrix(), xmuLabel_RAM_Model(), xmuMI(), xmuMakeDeviationThresholdsMatrices(), xmuMakeOneHeadedPathsFromPathList(), xmuMakeTwoHeadedPathsFromPathList(), xmuMaxLevels(), xmuMinLevels(), xmuPropagateLabels(), xmuRAM2Ordinal(), xmuTwinSuper_Continuous(), xmuTwinUpgradeMeansToCovariateModel(), xmu_CI_merge(), xmu_DF_to_mxData_TypeCov(), xmu_PadAndPruneForDefVars(), xmu_cell_is_on(), xmu_check_levels_identical(), xmu_check_needs_means(), xmu_check_variance(), xmu_clean_label(), xmu_data_missing(), xmu_data_swap_a_block(), xmu_describe_data_WLS(), xmu_dot_make_paths(), xmu_dot_make_residuals(), xmu_dot_maker(), xmu_dot_move_ranks(), xmu_dot_rank_str(), xmu_extract_column(), xmu_get_CI(), xmu_lavaan_process_group(), xmu_make_TwinSuperModel(), xmu_make_bin_cont_pair_data(), xmu_make_mxData(), xmu_match.arg(), xmu_name_from_lavaan_str(), xmu_path2twin(), xmu_path_regex(), xmu_safe_run_summary(), xmu_set_sep_from_suffix(), xmu_show_fit_or_comparison(), xmu_simplex_corner(), xmu_standardize_ACEcov(), xmu_standardize_ACEv(), xmu_standardize_ACE(), xmu_standardize_CP(), xmu_standardize_IP(), xmu_standardize_RAM(), xmu_standardize_SexLim(), xmu_standardize_Simplex(), xmu_start_value_list(), xmu_starts(), xmu_twin_add_WeightMatrices(), xmu_twin_check(), xmu_twin_get_var_names(), xmu_twin_upgrade_selDvs2Sel\

---

xmu_clean_label *Remove illegal characters from labels*

Description

Replaces . with _ in labels - e.g. from lavaan where . is common.

Usage

```
xmu_clean_label(label, replace = "_")
```

Arguments

label	A label to clean.
replace	character to replace . with (default = _)

Value

- legal label string

See Also

- umxLabel()

Other xmu internal not for end user: umxModel(), umxRenameMatrix(), umxTwinMaker(), umx_APA_pval(), umx_fun_mean_sd(), umx_get_bracket_addresses(), umx_make(), umx_standardize(), umx_string_to_algebra(), umx, xmuHasSquareBrackets(), xmuLabel_MATRIX_Model(), xmuLabel_Matrix(), xmuLabel_RAM_Model(),

xmuMI(), xmuMakeDeviationThresholdsMatrices(), xmuMakeOneHeadedPathsFromPathList(), xmuMakeTwoHeadedPathsFromPathList(), xmuMaxLevels(), xmuMinLevels(), xmuPropagateLabels(), xmuRAM2Ordinal(), xmuTwinSuper_Continuous(), xmuTwinUpgradeMeansToCovariateModel(), xmu_CI_merge(), xmu_CI_stash(), xmu_DF_to_mxData_TypeCov(), xmu_PadAndPruneForDefVars(), xmu_cell_is_on(), xmu_check_levels_identical(), xmu_check_needs_means(), xmu_check_variance(), xmu_data_missing(), xmu_data_swap_a_block(), xmu_describe_data_WLS(), xmu_dot_make_paths(), xmu_dot_make_residuals(), xmu_dot_maker(), xmu_dot_move_ranks(), xmu_dot_rank_str(), xmu_extract_column(), xmu_get_CI(), xmu_lavaan_process_group(), xmu_make_TwinSuperModel(), xmu_make_bin_cont_pair_data(), xmu_make_mxData(), xmu_match.arg(), xmu_name_from_lavaan_str(), xmu_path2twin(), xmu_path_regex(), xmu_safe_run_summary(), xmu_set_sep_from_suffix(), xmu_show_fit_or_comparison(), xmu_simplex_corner(), xmu_standardize_ACEcov(), xmu_standardize_ACEv(), xmu_standardize_ACE(), xmu_standardize_CP(), xmu_standardize_IP(), xmu_standardize_RAM(), xmu_standardize_SexLim(), xmu_standardize_Simplex(), xmu_start_value_list(), xmu_starts(), xmu_twin_add_WeightMatrices(), xmu_twin_check(), xmu_twin_get_var_names(), xmu_twin_upgrade_selDvs2SelV

Examples

```
xmu_clean_label("data.var", replace = "_")
xmu_clean_label("my.var.lab", replace = "_")
```

---

xmu_data_missing *Drop rows with missing definition variables*

Description

Definition variables can't be missing. This function helps with that.

Usage

```
xmu_data_missing(
  data,
  selVars,
  sep = NULL,
  dropMissingDef = TRUE,
  hint = "data"
)
```

Arguments

data	The dataframe to check for missing variables
selVars	The variables to check for missingness
sep	A sep if this is twin data and selVars are baseNames (default NULL)
dropMissingDef	Whether to drop the rows, or just stop (TRUE)
hint	info for message to user ("data")

Value

- data with missing rows dropped

See Also

- complete.cases()

Other xmu internal not for end user: umxModel(), umxRenameMatrix(), umxTwinMaker(), umx_APA_pval(), umx_fun_mean_sd(), umx_get_bracket_addresses(), umx_make(), umx_standardize(), umx_string_to_algebra(), umx, xmuHasSquareBrackets(), xmuLabel_MATRIX_Model(), xmuLabel_Matrix(), xmuLabel_RAM_Model(), xmuMI(), xmuMakeDeviationThresholdsMatrices(), xmuMakeOneHeadedPathsFromPathList(), xmuMakeTwoHeadedPathsFromPathList(), xmuMaxLevels(), xmuMinLevels(), xmuPropagateLabels(), xmuRAM2Ordinal(), xmuTwinSuper_Continuous(), xmuTwinUpgradeMeansToCovariateModel(), xmu_CI_merge(), xmu_CI_stash(), xmu_DF_to_mxData_TypeCov(), xmu_PadAndPruneForDefVars(), xmu_cell_is_on(), xmu_check_levels_identical(), xmu_check_needs_means(), xmu_check_variance(), xmu_clean_label(), xmu_data_swap_a_block(), xmu_describe_data_WLS(), xmu_dot_make_paths(), xmu_dot_make_residuals(), xmu_dot_maker(), xmu_dot_move_ranks(), xmu_dot_rank_str(), xmu_extract_column(), xmu_get_CI(), xmu_lavaan_process_group(), xmu_make_TwinSuperModel(), xmu_make_bin_cont_pair_data(), xmu_make_mxData(), xmu_match.arg(), xmu_name_from_lavaan_str(), xmu_path2twin(), xmu_path_regex(), xmu_safe_run_summary(), xmu_set_sep_from_suffix(), xmu_show_fit_or_comparison(), xmu_simplex_corner(), xmu_standardize_ACEcov(), xmu_standardize_ACEv(), xmu_standardize_ACE(), xmu_standardize_CP(), xmu_standardize_IP(), xmu_standardize_RAM(), xmu_standardize_SexLim(), xmu_standardize_Simplex(), xmu_start_value_list(), xmu_starts(), xmu_twin_add_WeightMatrices(), xmu_twin_check(), xmu_twin_get_var_names(), xmu_twin_upgrade_selDvs2Sel

Examples

```
tmp = mtcars;
tmp[1,]; tmp[1, "wt"] = NA
tmp = xmu_data_missing(tmp, selVars = "wt", sep= NULL, dropMissingDef= TRUE, hint= "mtcars")
dim(mtcars)
dim(tmp)

## Not run:
tmp = xmu_data_missing(tmp, selVars = "wt", sep= NULL, dropMissingDef= FALSE, hint= "mtcars")

## End(Not run)
```

---

xmu_data_swap_a_block *Data helper function to swap blocks of data from one set of columns to another.*

Description

Swap a block of rows of a dataset between two sets of variables (typically twin 1 and twin 2)

Usage

```
xmu_data_swap_a_block(theData, rowSelector, T1Names, T2Names)
```

Arguments

theData	A data frame to swap within.
rowSelector	Rows to swap between first and second set of columns.
T1Names	The first set of columns.
T2Names	The second set of columns.

Value

- dataframe

See Also

- subset()

Other xmu internal not for end user: umxModel(), umxRenameMatrix(), umxTwinMaker(), umx_APA_pval(), umx_fun_mean_sd(), umx_get_bracket_addresses(), umx_make(), umx_standardize(), umx_string_to_algebra(), umx, xmuHasSquareBrackets(), xmuLabel_MATRIX_Model(), xmuLabel_Matrix(), xmuLabel_RAM_Model(), xmuMI(), xmuMakeDeviationThresholdsMatrices(), xmuMakeOneHeadedPathsFromPathList(), xmuMakeTwoHeadedPathsFromPathList(), xmuMaxLevels(), xmuMinLevels(), xmuPropagateLabels(), xmuRAM2Ordinal(), xmuTwinSuper_Continuous(), xmuTwinUpgradeMeansToCovariateModel(), xmu_CI_merge(), xmu_CI_stash(), xmu_DF_to_mxData_TypeCov(), xmu_PadAndPruneForDefVars(), xmu_cell_is_on(), xmu_check_levels_identical(), xmu_check_needs_means(), xmu_check_variance(), xmu_clean_label(), xmu_data_missing(), xmu_describe_data_WLS(), xmu_dot_make_paths(), xmu_dot_make_residuals(), xmu_dot_maker(), xmu_dot_move_ranks(), xmu_dot_rank_str(), xmu_extract_column(), xmu_get_CI(), xmu_lavaan_process_group(), xmu_make_TwinSuperModel(), xmu_make_bin_cont_pair_data(), xmu_make_mxData(), xmu_match.arg(), xmu_name_from_lavaan_str(), xmu_path2twin(), xmu_path_regex(), xmu_safe_run_summary(), xmu_set_sep_from_suffix(), xmu_show_fit_or_comparison(), xmu_simplex_corner(), xmu_standardize_ACEcov(), xmu_standardize_ACEv(), xmu_standardize_ACE(), xmu_standardize_CP(), xmu_standardize_IP(), xmu_standardize_RAM(), xmu_standardize_SexLim(), xmu_standardize_Simplex(), xmu_start_value_list(), xmu_starts(), xmu_twin_add_WeightMatrices(), xmu_twin_check(), xmu_twin_get_var_names(), xmu_twin_upgrade_selDvs2Sel\

Examples

```
test = data.frame(
a = paste0("a", 1:10),
b = paste0("b", 1:10),
c = paste0("c", 1:10),
d = paste0("d", 1:10), stringsAsFactors = FALSE)
xmu_data_swap_a_block(test, rowSelector = c(1,2,3,6), T1Names = "b", T2Names = "c")
xmu_data_swap_a_block(test, rowSelector = c(1,2,3,6), T1Names = c("a","c"), T2Names = c("b","d"))
```

`xmu_describe_data_WLS` *Determine whether a dataset will need weights and summary statistics for the means if used with mxFitFunctionWLS*

Description

Given either a data.frame or an mxData of type raw, this function determines whether `mxFitFunctionWLS()` will generate expectations for means.

Usage

```
xmu_describe_data_WLS(
  data,
  allContinuousMethod = c("cumulants", "marginals"),
  verbose = FALSE
)
```

Arguments

`data`	The raw data being used in a `mxFitFunctionWLS()` model.
`allContinuousMethod`	the method used to process data when all columns are continuous (default = "cumulants")
`verbose`	Whether or not to report diagnostics.

Details

All-continuous models processed using the "cumulants" method lack means, while all continuous processed with allContinuousMethod = "marginals" will have means.

When data are not all continuous, allContinuousMethod is ignored, and means are modeled.

Value

- list describing the data.

See Also

- `mxFitFunctionWLS()`, `omxAugmentDataWithWLSSummary()`

Other xmu internal not for end user: `umxModel()`, `umxRenameMatrix()`, `umxTwinMaker()`, `umx_APA_pval()`, `umx_fun_mean_sd()`, `umx_get_bracket_addresses()`, `umx_make()`, `umx_standardize()`, `umx_string_to_algebra()`, `umx`, `xmuHasSquareBrackets()`, `xmuLabel_MATRIX_Model()`, `xmuLabel_Matrix()`, `xmuLabel_RAM_Model()`, `xmuMI()`, `xmuMakeDeviationThresholdsMatrices()`, `xmuMakeOneHeadedPathsFromPathList()`, `xmuMakeTwoHeadedPathsFromPathList()`, `xmuMaxLevels()`, `xmuMinLevels()`, `xmuPropagateLabels()`, `xmuRAM2Ordinal()`, `xmuTwinSuper_Continuous()`, `xmuTwinUpgradeMeansToCovariateModel()`, `xmu_CI_merge()`, `xmu_CI_stash()`, `xmu_DF_to_mxData_TypeCov()`, `xmu_PadAndPruneForDefVars()`, `xmu_cell_is_on()`, `xmu_check_levels_identical()`, `xmu_check_needs_means()`, `xmu_check_variance()`,

xmu_clean_label(), xmu_data_missing(), xmu_data_swap_a_block(), xmu_dot_make_paths(), xmu_dot_make_residuals(), xmu_dot_maker(), xmu_dot_move_ranks(), xmu_dot_rank_str(), xmu_extract_column(), xmu_get_CI(), xmu_lavaan_process_group(), xmu_make_TwinSuperModel(), xmu_make_bin_cont_pair_data(), xmu_make_mxData(), xmu_match.arg(), xmu_name_from_lavaan_str(), xmu_path2twin(), xmu_path_regex(), xmu_safe_run_summary(), xmu_set_sep_from_suffix(), xmu_show_fit_or_comparison(), xmu_simplex_corner(), xmu_standardize_ACEcov(), xmu_standardize_ACEv(), xmu_standardize_ACE(), xmu_standardize_CP(), xmu_standardize_IP(), xmu_standardize_RAM(), xmu_standardize_SexLim(), xmu_standardize_Simplex(), xmu_start_value_list(), xmu_starts(), xmu_twin_add_WeightMatrices(), xmu_twin_check(), xmu_twin_get_var_names(), xmu_twin_upgrade_selDvs2SelV

Examples

```
# ====================================
# = All continuous, data.frame input =
# ====================================

tmp =xmu_describe_data_WLS(mtcars, allContinuousMethod= "cumulants", verbose = TRUE)
tmp$hasMeans # FALSE - no means with cumulants
tmp =xmu_describe_data_WLS(mtcars, allContinuousMethod= "marginals")
tmp$hasMeans # TRUE we get means with marginals

# ==========================
# = mxData object as input =
# ==========================
tmp = mxData(mtcars, type="raw")
xmu_describe_data_WLS(tmp, allContinuousMethod= "cumulants", verbose = TRUE)$hasMeans # FALSE
xmu_describe_data_WLS(tmp, allContinuousMethod= "marginals")$hasMeans  # TRUE

# ======================================
# = One var is a factor: Means modeled =
# ======================================
tmp = mtcars
tmp$cyl = factor(tmp$cyl)
xmu_describe_data_WLS(tmp, allContinuousMethod= "cumulants")$hasMeans # TRUE - always has means
xmu_describe_data_WLS(tmp, allContinuousMethod= "marginals")$hasMeans # TRUE
```

---

xmu_DF_to_mxData_TypeCov

Convert a dataframe into a cov mxData object

Description

xmu_DF_to_mxData_TypeCov converts a dataframe into mxData() with type="cov" and nrow = numObs and optionally adding means.

Usage

```
xmu_DF_to_mxData_TypeCov(
  df,
  columns = NA,
  use = c("complete.obs", "everything", "all.obs", "na.or.complete",
    "pairwise.complete.obs")
)
```

Arguments

df	the dataframe to covert to an mxData type cov object.
columns	= Which columns to keep (default is all).
use	= Default is "complete.obs".

Value

- mxData() of type = cov

References

- https://github.com/tbates/umx, https://tbates.github.io

See Also

Other xmu internal not for end user: umxModel(), umxRenameMatrix(), umxTwinMaker(), umx_APA_pval(), umx_fun_mean_sd(), umx_get_bracket_addresses(), umx_make(), umx_standardize(), umx_string_to_algebra(), umx, xmuHasSquareBrackets(), xmuLabel_MATRIX_Model(), xmuLabel_Matrix(), xmuLabel_RAM_Model(), xmuMI(), xmuMakeDeviationThresholdsMatrices(), xmuMakeOneHeadedPathsFromPathList(), xmuMakeTwoHeadedPathsFromPathList(), xmuMaxLevels(), xmuMinLevels(), xmuPropagateLabels(), xmuRAM2Ordinal(), xmuTwinSuper_Continuous(), xmuTwinUpgradeMeansToCovariateModel(), xmu_CI_merge(), xmu_CI_stash(), xmu_PadAndPruneForDefVars(), xmu_cell_is_on(), xmu_check_levels_identica xmu_check_needs_means(), xmu_check_variance(), xmu_clean_label(), xmu_data_missing(), xmu_data_swap_a_block(), xmu_describe_data_WLS(), xmu_dot_make_paths(), xmu_dot_make_residuals(), xmu_dot_maker(), xmu_dot_move_ranks(), xmu_dot_rank_str(), xmu_extract_column(), xmu_get_CI(), xmu_lavaan_process_group(), xmu_make_TwinSuperModel(), xmu_make_bin_cont_pair_data(), xmu_make_mxData(), xmu_match.arg(), xmu_name_from_lavaan_str(), xmu_path2twin(), xmu_path_regex(), xmu_safe_run_summary(), xmu_set_sep_from_suffix(), xmu_show_fit_or_comparison(), xmu_simplex_corner(), xmu_standardize_ACEcov(), xmu_standardize_ACEv(), xmu_standardize_ACE(), xmu_standardize_CP(), xmu_standardize_IP(), xmu_standardize_RAM(), xmu_standardize_SexLim(), xmu_standardize_Simplex(), xmu_start_value_list(), xmu_starts(), xmu_twin_add_WeightMatrices(), xmu_twin_check(), xmu_twin_get_var_names(), xmu_twin_upgrade_selDvs2SelVars()

Examples

```
xmu_DF_to_mxData_TypeCov(mtcars, c("mpg", "hp"))
```

xmu_dot_define_shapes *Helper to make the list of vars and their shapes for a graphviz string*

Description

Helper to make a graphviz rank string is a function which.

Usage

```
xmu_dot_define_shapes(latents, manifests, preOut = "")
```

Arguments

latents	list of latent variables
manifests	list of manifest variables
preOut	"" by default.

Value

string

See Also

- [xmu_dot_rank()]

Other Graphviz: xmu_dot_make_paths(), xmu_dot_make_residuals(), xmu_dot_maker(), xmu_dot_mat2dot(), xmu_dot_rank()

Examples

```
xmu_dot_define_shapes(c("as1"), c("E", "N"))
```

xmu_dot_maker *Internal umx function to help plotting graphviz*

Description

Helper to print a digraph to file and open it

Usage

```
xmu_dot_maker(model, file, digraph, strip_zero = TRUE)
```

Arguments

model	An mxModel() to get the name from
file	Either "name" (use model name) or a file name
digraph	Graphviz code for a model
strip_zero	Whether to remove the leading "0." in digits in the diagram

Value

- optionally returns the digraph text.

See Also

Other xmu internal not for end user: umxModel(), umxRenameMatrix(), umxTwinMaker(), umx_APA_pval(), umx_fun_mean_sd(), umx_get_bracket_addresses(), umx_make(), umx_standardize(), umx_string_to_algebra(), umx, xmuHasSquareBrackets(), xmuLabel_MATRIX_Model(), xmuLabel_Matrix(), xmuLabel_RAM_Model(), xmuMI(), xmuMakeDeviationThresholdsMatrices(), xmuMakeOneHeadedPathsFromPathList(), xmuMakeTwoHeadedPathsFromPathList(), xmuMaxLevels(), xmuMinLevels(), xmuPropagateLabels(), xmuRAM2Ordinal(), xmuTwinSuper_Continuous(), xmuTwinUpgradeMeansToCovariateModel(), xmu_CI_merge(), xmu_CI_stash(), xmu_DF_to_mxData_TypeCov(), xmu_PadAndPruneForDefVars(), xmu_cell_is_on(), xmu_check_levels_identical(), xmu_check_needs_means(), xmu_check_variance(), xmu_clean_label(), xmu_data_missing(), xmu_data_swap_a_block(), xmu_describe_data_WLS(), xmu_dot_make_paths(), xmu_dot_make_residuals(), xmu_dot_move_ranks(), xmu_dot_rank_str(), xmu_extract_column(), xmu_get_CI(), xmu_lavaan_process_group(), xmu_make_TwinSuperModel(), xmu_make_bin_cont_pair_data(), xmu_make_mxData(), xmu_match.arg(), xmu_name_from_lavaan_str(), xmu_path2twin(), xmu_path_regex(), xmu_safe_run_summary(), xmu_set_sep_from_suffix(), xmu_show_fit_or_comparison(), xmu_simplex_corner(), xmu_standardize_ACEcov(), xmu_standardize_ACEv(), xmu_standardize_ACE(), xmu_standardize_CP(), xmu_standardize_IP(), xmu_standardize_RAM(), xmu_standardize_SexLim(), xmu_standardize_Simplex(), xmu_start_value_list(), xmu_starts(), xmu_twin_add_WeightMatrices(), xmu_twin_check(), xmu_twin_get_var_names(), xmu_twin_upgrade_selDvs2Sel

Other Graphviz: xmu_dot_define_shapes(), xmu_dot_make_paths(), xmu_dot_make_residuals(), xmu_dot_mat2dot(), xmu_dot_rank()

---

xmu_dot_make_paths *xmu_dot_make_paths (not for end users)*

Description

Makes graphviz paths

Usage

```
xmu_dot_make_paths(
  mxMat,
  stringIn,
  heads = NULL,
  fixed = TRUE,
```

```
  comment = "More paths",
  showResiduals = TRUE,
  labels = "labels",
  digits = 2
)
```

Arguments

mxMat	An mxMatrix
stringIn	Input string
heads	1 or 2 arrows (default NULL - you must set this)
fixed	Whether show fixed values or not (defaults to TRUE)
comment	A comment to include
showResiduals	Whether to show residuals
labels	show labels on the path? ("none", "labels", "both")
digits	how many digits to report

Value

- string

See Also

Other xmu internal not for end user: umxModel(), umxRenameMatrix(), umxTwinMaker(), umx_APA_pval(), umx_fun_mean_sd(), umx_get_bracket_addresses(), umx_make(), umx_standardize(), umx_string_to_algebra(), umx, xmuHasSquareBrackets(), xmuLabel_MATRIX_Model(), xmuLabel_Matrix(), xmuLabel_RAM_Model(), xmuMI(), xmuMakeDeviationThresholdsMatrices(), xmuMakeOneHeadedPathsFromPathList(), xmuMakeTwoHeadedPathsFromPathList(), xmuMaxLevels(), xmuMinLevels(), xmuPropagateLabels(), xmuRAM2Ordinal(), xmuTwinSuper_Continuous(), xmuTwinUpgradeMeansToCovariateModel(), xmu_CI_merge(), xmu_CI_stash(), xmu_DF_to_mxData_TypeCov(), xmu_PadAndPruneForDefVars(), xmu_cell_is_on(), xmu_check_levels_identical(), xmu_check_needs_means(), xmu_check_variance(), xmu_clean_label(), xmu_data_missing(), xmu_data_swap_a_block(), xmu_describe_data_WLS(), xmu_dot_make_residuals(), xmu_dot_maker(), xmu_dot_move_ranks(), xmu_dot_rank_str(), xmu_extract_column(), xmu_get_CI(), xmu_lavaan_process_group(), xmu_make_TwinSuperModel(), xmu_make_bin_cont_pair_data(), xmu_make_mxData(), xmu_match.arg(), xmu_name_from_lavaan_str(), xmu_path2twin(), xmu_path_regex(), xmu_safe_run_summary(), xmu_set_sep_from_suffix(), xmu_show_fit_or_comparison(), xmu_simplex_corner(), xmu_standardize_ACEcov(), xmu_standardize_ACEv(), xmu_standardize_ACE(), xmu_standardize_CP(), xmu_standardize_IP(), xmu_standardize_RAM(), xmu_standardize_SexLim(), xmu_standardize_Simplex(), xmu_start_value_list(), xmu_starts(), xmu_twin_add_WeightMatrices(), xmu_twin_check(), xmu_twin_get_var_names(), xmu_twin_upgrade_selDvs2Sel

Other Graphviz: xmu_dot_define_shapes(), xmu_dot_make_residuals(), xmu_dot_maker(), xmu_dot_mat2dot(), xmu_dot_rank()

xmu_dot_make_residuals *xmu_dot_make_residuals (not for end users)*

Description

xmu_dot_make_residuals (not for end users)

Usage

```
xmu_dot_make_residuals(
  mxMat,
  latents = NULL,
  fixed = TRUE,
  digits = 2,
  resid = c("circle", "line")
)
```

Arguments

mxMat	An A or S mxMatrix
latents	Optional list of latents to alter location of circles (defaults to NULL)
fixed	Whether to show fixed values or not
digits	How many digits to report
resid	How to show residuals and variances default is "circle". Other option is "line"

Value

- list of variance names and variances

See Also

Other xmu internal not for end user: umxModel(), umxRenameMatrix(), umxTwinMaker(), umx_APA_pval(), umx_fun_mean_sd(), umx_get_bracket_addresses(), umx_make(), umx_standardize(), umx_string_to_algebra(), umx, xmuHasSquareBrackets(), xmuLabel_MATRIX_Model(), xmuLabel_Matrix(), xmuLabel_RAM_Model(), xmuMI(), xmuMakeDeviationThresholdsMatrices(), xmuMakeOneHeadedPathsFromPathList(), xmuMakeTwoHeadedPathsFromPathList(), xmuMaxLevels(), xmuMinLevels(), xmuPropagateLabels(), xmuRAM2Ordinal(), xmuTwinSuper_Continuous(), xmuTwinUpgradeMeansToCovariateModel(), xmu_CI_merge(), xmu_CI_stash(), xmu_DF_to_mxData_TypeCov(), xmu_PadAndPruneForDefVars(), xmu_cell_is_on(), xmu_check_levels_identical(), xmu_check_needs_means(), xmu_check_variance(), xmu_clean_label(), xmu_data_missing(), xmu_data_swap_a_block(), xmu_describe_data_WLS(), xmu_dot_make_paths(), xmu_dot_maker(), xmu_dot_move_ranks(), xmu_dot_rank_str(), xmu_extract_column(), xmu_get_CI(), xmu_lavaan_process_group(), xmu_make_TwinSuperModel(), xmu_make_bin_cont_pair_data(), xmu_make_mxData(), xmu_match.arg(), xmu_name_from_lavaan_str(), xmu_path2twin(), xmu_path_regex(), xmu_safe_run_summary(), xmu_set_sep_from_suffix(), xmu_show_fit_or_comparison(), xmu_simplex_corner(), xmu_standardize_ACEcov(), xmu_standardize_ACEv(), xmu_standardize_ACE(), xmu_standardize_CP(),

xmu_standardize_IP(), xmu_standardize_RAM(), xmu_standardize_SexLim(), xmu_standardize_Simplex(), xmu_start_value_list(), xmu_starts(), xmu_twin_add_WeightMatrices(), xmu_twin_check(), xmu_twin_get_var_names(), xmu_twin_upgrade_selDvs2SelVars()

Other Graphviz: xmu_dot_define_shapes(), xmu_dot_make_paths(), xmu_dot_maker(), xmu_dot_mat2dot(), xmu_dot_rank()

`xmu_dot_mat2dot` *Return dot code for paths in a matrix*

Description

Return dot code for paths in a matrix is a function which walks the rows and cols of a matrix. At each free cell, it creates a dot-string specifying the relevant path, e.g.:

`ai1 -> var1 [label=".35"]`

Its main use is to correctly generate paths (and their sources and sink objects) without depending on the label of the parameter.

It is highly customizable:

1. You can specify which cells to inspect, e.g. "lower".
2. You can choose how to interpret path direction, from = "cols".
3. You can choose the label for the from to ends of the path (by default, the matrix name is used).
4. Offer up a list of from and toLabel which will be indexed into for source and sink
5. You can set the number of arrows on a path (e.g. both).
6. If type is set, then sources and sinks added manifests and/or latents output (p)

Finally, you can pass in previous output and new paths will be concatenated to these.

Usage

```
xmu_dot_mat2dot(
  x,
  cells = c("diag", "lower", "lower_inc", "upper", "upper_inc", "any", "left"),
  from = c("rows", "cols"),
  fromLabel = NULL,
  toLabel = NULL,
  showFixed = FALSE,
  arrows = c("forward", "both", "back"),
  fromType = NULL,
  toType = NULL,
  digits = 2,
  model = NULL,
  SEstyle = FALSE,
  p = list(str = "", latents = c(), manifests = c())
)
```

Arguments

x	a umxMatrix() to make paths from.
cells	which cells to process: "any" (default), "diag", "lower", "upper". "left" is the left half (e.g. in a twin means matrix)
from	one of "rows", "columns"
fromLabel	= NULL. NULL = use matrix name (default). If one, if suffixed with index, length() > 1, index into list. "one" is special.
toLabel	= NULL. NULL = use matrix name (default). If one, if suffixed with index, length() > 1, index into list.
showFixed	= FALSE.
arrows	"forward" "both" or "back"
fromType	one of "latent" or "manifest" NULL (default) = don't accumulate new names.
toType	one of "latent" or "manifest" NULL (default) = don't accumulate new names.
digits	to round values to (default = 2).
model	If you want to get CIs, you can pass in the model (default = NULL).
SEstyle	If TRUE, CIs shown as "b(SE)" ("b [l,h]" if FALSE (default)). Ignored if model NULL.
p	input to build on. list(str = "", latents = c(), manifests = c())

Value

- list(str = "", latents = c(), manifests = c())

See Also

- plot()

Other Graphviz: xmu_dot_define_shapes(), xmu_dot_make_paths(), xmu_dot_make_residuals(), xmu_dot_maker(), xmu_dot_rank()

Examples

```
# Make a lower 3 * 3 value= 1:6 (1, 4, 6 on the diag)
a_cp = umxMatrix("a_cp", "Lower", 3, 3, free = TRUE, values = 1:6)

# Get dot strings for lower triangle (default from and to based on row and column number)
out = xmu_dot_mat2dot(a_cp, cells = "lower", from = "cols", arrows = "both")
cat(out$str) # a_cp1 -> a_cp2 [dir = both label="2"];

# one arrow (the default = "forward")
out = xmu_dot_mat2dot(a_cp, cells = "lower", from = "cols")
cat(out$str) # a_cp1 -> a_cp2 [dir = forward label="2"];

# label to (rows) using var names

out = xmu_dot_mat2dot(a_cp, toLabel= paste0("v", 1:3), cells = "lower", from = "cols")
```

```
umx_msg(out$str) # a_cp1 -> v2 [dir = forward label="2"] ...

# First call also inits the plot struct
out = xmu_dot_mat2dot(a_cp, from = "rows", cells = "lower", arrows = "both", fromType = "latent")
out = xmu_dot_mat2dot(a_cp, from = "rows", cells = "diag",
toLabel= "common", toType = "manifest", p = out)
umx_msg(out$str); umx_msg(out$manifests); umx_msg(out$latents)

# ================================
# = Add found sinks to manifests =
# ================================
out = xmu_dot_mat2dot(a_cp, from= "rows", cells= "diag",
toLabel= c('a','b','c'), toType= "manifest");
umx_msg(out$manifests)

# ================================
# = Add found sources to latents =
# ================================
out = xmu_dot_mat2dot(a_cp, from= "rows", cells= "diag",
toLabel= c('a','b','c'), fromType= "latent");
umx_msg(out$latents)


# ========================
# = Label a means matrix =
# ========================

tmp = umxMatrix("expMean", "Full", 1, 4, free = TRUE, values = 1:4)
out = xmu_dot_mat2dot(tmp, cells = "left", from = "rows",
fromLabel= "one", toLabel= c("v1", "v2")
)
cat(out$str)

## Not run:
# ============================================
# = Get a string which includes CI information =
# ============================================
data(demoOneFactor)
latents = c("g"); manifests = names(demoOneFactor)
m1 = umxRAM("xmu_dot", data = demoOneFactor, type = "cov",
umxPath(latents, to = manifests),
umxPath(var = manifests),
umxPath(var = latents, fixedAt = 1.0)
)
m1 = umxCI(m1, run= "yes")
out = xmu_dot_mat2dot(m1$A, from = "cols", cells = "any",
      toLabel= paste0("x", 1:5), fromType = "latent", model= m1);
umx_msg(out$str); umx_msg(out$latents)


## End(Not run)
```

xmu_dot_move_ranks *xmu_dot_move_ranks (not for end users)*

Description

Variables will be moved from any existing rank to the new one. Setting a rank to "" will clear it.

Usage

```
xmu_dot_move_ranks(
  min = NULL,
  same = NULL,
  max = NULL,
  old_min,
  old_same,
  old_max
)
```

Arguments

min	vars to group at top of plot
same	vars to group at the same level
max	vars to group at bottom of plot
old_min	vars to group at top of plot
old_same	vars to group at the same level
old_max	vars to group at bottom of plot

Value

- list(min=min, same=same, max=max)

See Also

Other xmu internal not for end user: umxModel(), umxRenameMatrix(), umxTwinMaker(), umx_APA_pval(), umx_fun_mean_sd(), umx_get_bracket_addresses(), umx_make(), umx_standardize(), umx_string_to_algebra(), umx, xmuHasSquareBrackets(), xmuLabel_MATRIX_Model(), xmuLabel_Matrix(), xmuLabel_RAM_Model(), xmuMI(), xmuMakeDeviationThresholdsMatrices(), xmuMakeOneHeadedPathsFromPathList(), xmuMakeTwoHeadedPathsFromPathList(), xmuMaxLevels(), xmuMinLevels(), xmuPropagateLabels(), xmuRAM2Ordinal(), xmuTwinSuper_Continuous(), xmuTwinUpgradeMeansToCovariateModel(), xmu_CI_merge(), xmu_CI_stash(), xmu_DF_to_mxData_TypeCov(), xmu_PadAndPruneForDefVars(), xmu_cell_is_on(), xmu_check_levels_identical(), xmu_check_needs_means(), xmu_check_variance(), xmu_clean_label(), xmu_data_missing(), xmu_data_swap_a_block(), xmu_describe_data_WLS(), xmu_dot_make_paths(), xmu_dot_make_residuals(), xmu_dot_maker(), xmu_dot_rank_str(), xmu_extract_column(), xmu_get_CI(), xmu_lavaan_process_group(), xmu_make_TwinSuperModel(), xmu_make_bin_cont_pair_data(), xmu_make_mxData(), xmu_match.arg(), xmu_name_from_lavaan_str(), xmu_path2twin(), xmu_path_regex(), xmu_safe_run_summary(), xmu_set_sep_from_suffix(),

xmu_show_fit_or_comparison(), xmu_simplex_corner(), xmu_standardize_ACEcov(), xmu_standardize_ACEv(), xmu_standardize_ACE(), xmu_standardize_CP(), xmu_standardize_IP(), xmu_standardize_RAM(), xmu_standardize_SexLim(), xmu_standardize_Simplex(), xmu_start_value_list(), xmu_starts(), xmu_twin_add_WeightMatrices(), xmu_twin_check(), xmu_twin_get_var_names(), xmu_twin_upgrade_selDvs2Sel\

Examples

```
old_min = c("min1", "min2")
old_same = c("s1", "s2")
old_max = paste0("x", 1:3)

# Add L1 to min
xmu_dot_move_ranks(min = "L1", old_min= old_min, old_same= old_same, old_max= old_max)

# Move min1 to max
xmu_dot_move_ranks(max = "min1", old_min= old_min, old_same= old_same, old_max= old_max)

# Clear min
xmu_dot_move_ranks(min = "", old_min= old_min, old_same= old_same, old_max= old_max)
```

---

xmu_dot_rank *Helper to make a graphviz rank string*

Description

Given a list of names, this filters the list, and returns a graphviz string to force them into the given rank. e.g. "rank=same; as1;"

Usage

```
xmu_dot_rank(vars, pattern, rank)
```

Arguments

vars	a list of strings
pattern	regular expression to filter vars
rank	"same", "max", "min"

Value

string

See Also

- xmu_dot_define_shapes()

Other Graphviz: xmu_dot_define_shapes(), xmu_dot_make_paths(), xmu_dot_make_residuals(), xmu_dot_maker(), xmu_dot_mat2dot()

Examples

```
xmu_dot_rank(c("as1"), "^[ace]s[0-9]+$", "same")
```

---

xmu_dot_rank_str *xmu_dot_rank_str (not for end users)*

Description

xmu_dot_rank_str (not for end users)

Usage

```
xmu_dot_rank_str(min = NULL, same = NULL, max = NULL)
```

Arguments

min	vars to group at top of plot
same	vars to group at the same level
max	vars to group at bottom of plot

Value

- GraphViz rank string

See Also

Other xmu internal not for end user: umxModel(), umxRenameMatrix(), umxTwinMaker(), umx_APA_pval(), umx_fun_mean_sd(), umx_get_bracket_addresses(), umx_make(), umx_standardize(), umx_string_to_algebra(), umx, xmuHasSquareBrackets(), xmuLabel_MATRIX_Model(), xmuLabel_Matrix(), xmuLabel_RAM_Model(), xmuMI(), xmuMakeDeviationThresholdsMatrices(), xmuMakeOneHeadedPathsFromPathList(), xmuMakeTwoHeadedPathsFromPathList(), xmuMaxLevels(), xmuMinLevels(), xmuPropagateLabels(), xmuRAM2Ordinal(), xmuTwinSuper_Continuous(), xmuTwinUpgradeMeansToCovariateModel(), xmu_CI_merge(), xmu_CI_stash(), xmu_DF_to_mxData_TypeCov(), xmu_PadAndPruneForDefVars(), xmu_cell_is_on(), xmu_check_levels_identical(), xmu_check_needs_means(), xmu_check_variance(), xmu_clean_label(), xmu_data_missing(), xmu_data_swap_a_block(), xmu_describe_data_WLS(), xmu_dot_make_paths(), xmu_dot_make_residuals(), xmu_dot_maker(), xmu_dot_move_ranks(), xmu_extract_column(), xmu_get_CI(), xmu_lavaan_process_group(), xmu_make_TwinSuperModel(), xmu_make_bin_cont_pair_data(), xmu_make_mxData(), xmu_match.arg(), xmu_name_from_lavaan_str(), xmu_path2twin(), xmu_path_regex(), xmu_safe_run_summary(), xmu_set_sep_from_suffix(), xmu_show_fit_or_comparison(), xmu_simplex_corner(), xmu_standardize_ACEcov(), xmu_standardize_ACEv(), xmu_standardize_ACE(), xmu_standardize_CP(), xmu_standardize_IP(), xmu_standardize_RAM(), xmu_standardize_SexLim(), xmu_standardize_Simplex(), xmu_start_value_list(), xmu_starts(), xmu_twin_add_WeightMatrices(), xmu_twin_check(), xmu_twin_get_var_names(), xmu_twin_upgrade_selDvs2Sel

Examples

```
xmu_dot_rank_str(min = "L1", same = c("x1", "x2"), max = paste0("e", 1:3))
```

xmu_extract_column *Get on or more columns from mzData or regular data.frame*

Description

same effect as df[,col] but works for mxData() and check the names are present

Usage

```
xmu_extract_column(data, col, drop = FALSE)
```

Arguments

data	mxData or data.frame
col	the name(s) of the column(s) to extract
drop	whether to drop the structure of the data.frame when extracting one column

Value

- column of data

See Also

Other xmu internal not for end user: umxModel(), umxRenameMatrix(), umxTwinMaker(), umx_APA_pval(), umx_fun_mean_sd(), umx_get_bracket_addresses(), umx_make(), umx_standardize(), umx_string_to_algebra(), umx, xmuHasSquareBrackets(), xmuLabel_MATRIX_Model(), xmuLabel_Matrix(), xmuLabel_RAM_Model(), xmuMI(), xmuMakeDeviationThresholdsMatrices(), xmuMakeOneHeadedPathsFromPathList(), xmuMakeTwoHeadedPathsFromPathList(), xmuMaxLevels(), xmuMinLevels(), xmuPropagateLabels(), xmuRAM2Ordinal(), xmuTwinSuper_Continuous(), xmuTwinUpgradeMeansToCovariateModel(), xmu_CI_merge(), xmu_CI_stash(), xmu_DF_to_mxData_TypeCov(), xmu_PadAndPruneForDefVars(), xmu_cell_is_on(), xmu_check_levels_identical(), xmu_check_needs_means(), xmu_check_variance(), xmu_clean_label(), xmu_data_missing(), xmu_data_swap_a_block(), xmu_describe_data_WLS(), xmu_dot_make_paths(), xmu_dot_make_residuals(), xmu_dot_maker(), xmu_dot_move_ranks(), xmu_dot_rank_str(), xmu_get_CI(), xmu_lavaan_process_group(), xmu_make_TwinSuperModel(), xmu_make_bin_cont_pair_data(), xmu_make_mxData(), xmu_match.arg(), xmu_name_from_lavaan_str(), xmu_path2twin(), xmu_path_regex(), xmu_safe_run_summary(), xmu_set_sep_from_suffix(), xmu_show_fit_or_comparison(), xmu_simplex_corner(), xmu_standardize_ACEcov(), xmu_standardize_ACEv(), xmu_standardize_ACE(), xmu_standardize_CP(), xmu_standardize_IP(), xmu_standardize_RAM(), xmu_standardize_SexLim(), xmu_standardize_Simplex(), xmu_start_value_list(), xmu_starts(), xmu_twin_add_WeightMatrices(), xmu_twin_check(), xmu_twin_get_var_names(), xmu_twin_upgrade_selDvs2Sel

Examples

```
xmu_extract_column(mtcars, "wt")
xmu_extract_column(mxData(mtcars, type = "raw"), "wt")
xmu_extract_column(mxData(mtcars, type = "raw"), "wt", drop=TRUE)
xmu_extract_column(mxData(mtcars, type = "raw"), c("wt", "mpg"))
```

xmu_get_CI *Look up and report CIs for free parameters*

Description

Look up CIs for free parameters in a model, and return as APA-formatted text string. If std are available, then these are reported.

Usage

```
xmu_get_CI(
  model,
  label,
  prefix = "top.",
  suffix = "_std",
  digits = 2,
  SEstyle = FALSE,
  verbose = FALSE
)
```

Arguments

model	an mxModel() to get CIs from
label	the label of the cell to interrogate for a CI, e.g. "ai_r1c1"
prefix	The submodel to look in (default = "top.")
suffix	The suffix for algebras when standardized (default = "_std")
digits	= 2
SEstyle	If TRUE, report "b(se)" instead of b CI95[l,u] (default = FALSE)
verbose	= FALSE

Value

- the CI string, e.g. ".73[-.20, .98]" or .73(.10)

References

- https://tbates.github.io, https://github.com/tbates/umx

See Also

Other xmu internal not for end user: umxModel(), umxRenameMatrix(), umxTwinMaker(), umx_APA_pval(), umx_fun_mean_sd(), umx_get_bracket_addresses(), umx_make(), umx_standardize(), umx_string_to_algebra(), umx, xmuHasSquareBrackets(), xmuLabel_MATRIX_Model(), xmuLabel_Matrix(), xmuLabel_RAM_Model(), xmuMI(), xmuMakeDeviationThresholdsMatrices(), xmuMakeOneHeadedPathsFromPathList(), xmuMakeTwoHeadedPathsFromPathList(), xmuMaxLevels(), xmuMinLevels(), xmuPropagateLabels(), xmuRAM2Ordinal(), xmuTwinSuper_Continuous(), xmuTwinUpgradeMeansToCovariateModel(),

xmu_CI_merge(), xmu_CI_stash(), xmu_DF_to_mxData_TypeCov(), xmu_PadAndPruneForDefVars(), xmu_cell_is_on(), xmu_check_levels_identical(), xmu_check_needs_means(), xmu_check_variance(), xmu_clean_label(), xmu_data_missing(), xmu_data_swap_a_block(), xmu_describe_data_WLS(), xmu_dot_make_paths(), xmu_dot_make_residuals(), xmu_dot_maker(), xmu_dot_move_ranks(), xmu_dot_rank_str(), xmu_extract_column(), xmu_lavaan_process_group(), xmu_make_TwinSuperModel(), xmu_make_bin_cont_pair_data(), xmu_make_mxData(), xmu_match.arg(), xmu_name_from_lavaan_str(), xmu_path2twin(), xmu_path_regex(), xmu_safe_run_summary(), xmu_set_sep_from_suffix(), xmu_show_fit_or_comparison(), xmu_simplex_corner(), xmu_standardize_ACEcov(), xmu_standardize_ACEv(), xmu_standardize_ACE(), xmu_standardize_CP(), xmu_standardize_IP(), xmu_standardize_RAM(), xmu_standardize_SexLim(), xmu_standardize_Simplex(), xmu_start_value_list(), xmu_starts(), xmu_twin_add_WeightMatrices(), xmu_twin_check(), xmu_twin_get_var_names(), xmu_twin_upgrade_selDvs2Sel

Examples

```
## Not run:
require(umx); data(demoOneFactor)
manifests = names(demoOneFactor)

m1 = umxRAM("get_CI_example", data = demoOneFactor, type = "cov",
umxPath("G", to = manifests),
umxPath(var = manifests),
umxPath(var = "G", fixedAt = 1)
)
m1 = umxCI(m1, run= "yes")

# Get CI by parameter label
xmu_get_CI(model= m1, "x1_with_x1")
xmu_get_CI(model= m1, "x1_with_x1", SEstyle=TRUE, digits=3)

# prefix (submodel) and suffix (e.g. std) are ignored if not needed
xmu_get_CI(model= m1, "x1_with_x1", prefix = "top.", suffix = "_std")

xmu_get_CI(fit_IP, label = "ai_r1c1", prefix = "top.", suffix = "_std")
xmu_get_CI(fit_IP, label = "ai_r1c1", prefix = "top.", SEstyle = TRUE, suffix = "_std")

## End(Not run)
```

---

xmu_lavaan_process_group

Process table of paths to model

Description

Process a set of lavaan tables rows forming a group (Model). Returns empty arrays if no rows matching the requested group are found.

Usage

```
xmu_lavaan_process_group(tab, groupNum)
```

Arguments

tab	a parameter table
groupNum	group number to filter table on

Value

- list(plist=plist, latents = latents, manifests = manifests)

See Also

- umxLav2RAM()

Other xmu internal not for end user: umxModel(), umxRenameMatrix(), umxTwinMaker(), umx_APA_pval(), umx_fun_mean_sd(), umx_get_bracket_addresses(), umx_make(), umx_standardize(), umx_string_to_algebra(), umx, xmuHasSquareBrackets(), xmuLabel_MATRIX_Model(), xmuLabel_Matrix(), xmuLabel_RAM_Model(), xmuMI(), xmuMakeDeviationThresholdsMatrices(), xmuMakeOneHeadedPathsFromPathList(), xmuMakeTwoHeadedPathsFromPathList(), xmuMaxLevels(), xmuMinLevels(), xmuPropagateLabels(), xmuRAM2Ordinal(), xmuTwinSuper_Continuous(), xmuTwinUpgradeMeansToCovariateModel(), xmu_CI_merge(), xmu_CI_stash(), xmu_DF_to_mxData_TypeCov(), xmu_PadAndPruneForDefVars(), xmu_cell_is_on(), xmu_check_levels_identical(), xmu_check_needs_means(), xmu_check_variance(), xmu_clean_label(), xmu_data_missing(), xmu_data_swap_a_block(), xmu_describe_data_WLS(), xmu_dot_make_paths(), xmu_dot_make_residuals(), xmu_dot_maker(), xmu_dot_move_ranks(), xmu_dot_rank_str(), xmu_extract_column(), xmu_get_CI(), xmu_make_TwinSuperModel(), xmu_make_bin_cont_pair_data(), xmu_make_mxData(), xmu_match.arg(), xmu_name_from_lavaan_str(), xmu_path2twin(), xmu_path_regex(), xmu_safe_run_summary(), xmu_set_sep_from_suffix(), xmu_show_fit_or_comparison(), xmu_simplex_corner(), xmu_standardize_ACEcov(), xmu_standardize_ACEv(), xmu_standardize_ACE(), xmu_standardize_CP(), xmu_standardize_IP(), xmu_standardize_RAM(), xmu_standardize_SexLim(), xmu_standardize_Simplex(), xmu_start_value_list(), xmu_starts(), xmu_twin_add_WeightMatrices(), xmu_twin_check(), xmu_twin_get_var_names(), xmu_twin_upgrade_selDvs2Sel\

Examples

```
tab = lavaan::lavaanify("y~x")
xmu_lavaan_process_group(tab, groupNum = 1)
xmu_lavaan_process_group(tab, groupNum = 0)
```

---

xmu_make_bin_cont_pair_data

Make pairs of bin & continuous columns to represent censored data

---

Description

Takes a dataframe of left-censored variables (vars with a floor effect) and does two things to it: 1. It creates new binary (1/0) copies of each column (with the suffix "bin"). These contain 0 where the variable is below the minimum and NA otherwise. 2. In each existing variable, it sets all instances of min for that var to NA

Usage

```
xmu_make_bin_cont_pair_data(data, vars = NULL, suffixes = NULL)
```

Arguments

data	A [data.frame()] to convert
vars	The variables to process
suffixes	Suffixes if the data are family (wide, more than one persona on a row)

Value

- copy of the dataframe with new binary variables and censoring

See Also

Other xmu internal not for end user: umxModel(), umxRenameMatrix(), umxTwinMaker(), umx_APA_pval(), umx_fun_mean_sd(), umx_get_bracket_addresses(), umx_make(), umx_standardize(), umx_string_to_algebra(), umx, xmuHasSquareBrackets(), xmuLabel_MATRIX_Model(), xmuLabel_Matrix(), xmuLabel_RAM_Model(), xmuMI(), xmuMakeDeviationThresholdsMatrices(), xmuMakeOneHeadedPathsFromPathList(), xmuMakeTwoHeadedPathsFromPathList(), xmuMaxLevels(), xmuMinLevels(), xmuPropagateLabels(), xmuRAM2Ordinal(), xmuTwinSuper_Continuous(), xmuTwinUpgradeMeansToCovariateModel(), xmu_CI_merge(), xmu_CI_stash(), xmu_DF_to_mxData_TypeCov(), xmu_PadAndPruneForDefVars(), xmu_cell_is_on(), xmu_check_levels_identical(), xmu_check_needs_means(), xmu_check_variance(), xmu_clean_label(), xmu_data_missing(), xmu_data_swap_a_block(), xmu_describe_data_WLS(), xmu_dot_make_paths(), xmu_dot_make_residuals(), xmu_dot_maker(), xmu_dot_move_ranks(), xmu_dot_rank_str(), xmu_extract_column(), xmu_get_CI(), xmu_lavaan_process_group(), xmu_make_TwinSuperModel(), xmu_make_mxData(), xmu_match.arg(), xmu_name_from_lavaan_str(), xmu_path2twin(), xmu_path_regex(), xmu_safe_run_summary(), xmu_set_sep_from_suffix(), xmu_show_fit_or_comparison(), xmu_simplex_corner(), xmu_standardize_ACEcov(), xmu_standardize_ACEv(), xmu_standardize_ACE(), xmu_standardize_CP(), xmu_standardize_IP(), xmu_standardize_RAM(), xmu_standardize_SexLim(), xmu_standardize_Simplex(), xmu_start_value_list(), xmu_starts(), xmu_twin_add_WeightMatrices(), xmu_twin_check(), xmu_twin_get_var_names(), xmu_twin_upgrade_selDvs2Sel

Examples

```
df = xmu_make_bin_cont_pair_data(mtcars, vars = c("mpg"))
str(df)
df[order(df$mpg), c(1,12)]
# Introduce a floor effect
tmp = mtcars; tmp$mpg[tmp$mpg<=15]=15
tmp$mpg_T1 = tmp$mpg_T2 = tmp$mpg
df = xmu_make_bin_cont_pair_data(tmp, vars = c("mpg"), suffixes = c("_T1", "_T2"))
df[order(df$mpg), 12:15]
```

xmu_make_mxData *Upgrade a dataframe to an mxData type.*

Description

xmu_make_mxData is an internal function to upgrade a dataframe to mxData. It can also drop variables and rows from the dataframe. The most common use will be to give it a dataframe, and get back an mxData object of type raw, cov, cor (WLS is just raw).

Usage

```
xmu_make_mxData(
  data = NULL,
  type = c("Auto", "FIML", "cov", "cor", "WLS", "DWLS", "ULS"),
  manifests = NULL,
  numObs = NULL,
  fullCovs = NULL,
  dropMissingDef = TRUE,
  verbose = FALSE
)
```

Arguments

data	A data.frame() or mxData()
type	What data type is wanted out c("Auto", "FIML", "cov", "cor", 'WLS', 'DWLS', 'ULS')
manifests	If set, only these variables will be retained.
numObs	Only needed if you pass in a cov/cor matrix wanting this to be upgraded to mxData
fullCovs	Covariate names if any (NULL = none) These are checked by dropMissingDef
dropMissingDef	Whether to automatically drop missing def var rows for the user (default = TRUE). You get a polite note.
verbose	If verbose, report on columns kept and dropped (default FALSE)

Value

- mxData()

See Also

Other xmu internal not for end user: umxModel(), umxRenameMatrix(), umxTwinMaker(), umx_APA_pval(), umx_fun_mean_sd(), umx_get_bracket_addresses(), umx_make(), umx_standardize(), umx_string_to_algebra(), umx, xmuHasSquareBrackets(), xmuLabel_MATRIX_Model(), xmuLabel_Matrix(), xmuLabel_RAM_Model(), xmuMI(), xmuMakeDeviationThresholdsMatrices(), xmuMakeOneHeadedPathsFromPathList(), xmuMakeTwoHeadedPathsFromPathList(), xmuMaxLevels(), xmuMinLevels(), xmuPropagateLabels(),

xmuRAM2Ordinal(), xmuTwinSuper_Continuous(), xmuTwinUpgradeMeansToCovariateModel(), xmu_CI_merge(), xmu_CI_stash(), xmu_DF_to_mxData_TypeCov(), xmu_PadAndPruneForDefVars(), xmu_cell_is_on(), xmu_check_levels_identical(), xmu_check_needs_means(), xmu_check_variance(), xmu_clean_label(), xmu_data_missing(), xmu_data_swap_a_block(), xmu_describe_data_WLS(), xmu_dot_make_paths(), xmu_dot_make_residuals(), xmu_dot_maker(), xmu_dot_move_ranks(), xmu_dot_rank_str(), xmu_extract_column(), xmu_get_CI(), xmu_lavaan_process_group(), xmu_make_TwinSuperModel(), xmu_make_bin_cont_pair_data(), xmu_match.arg(), xmu_name_from_lavaan_str(), xmu_path2twin(), xmu_path_regex(), xmu_safe_run_summary(), xmu_set_sep_from_suffix(), xmu_show_fit_or_comparison(), xmu_simplex_corner(), xmu_standardize_ACEcov(), xmu_standardize_ACEv(), xmu_standardize_ACE(), xmu_standardize_CP(), xmu_standardize_IP(), xmu_standardize_RAM(), xmu_standardize_SexLim(), xmu_standardize_Simplex(), xmu_start_value_list(), xmu_starts(), xmu_twin_add_WeightMatrices(), xmu_twin_check(), xmu_twin_get_var_names(), xmu_twin_upgrade_selDvs2Sel\

Examples

```
# ========================
# = Continuous ML example =
# ========================
data(mtcars)
tmp = xmu_make_mxData(data= mtcars, type = "Auto"); # class(tmp); # "MxDataStatic"
# names(tmp$observed) # "mpg"  "cyl"  "disp"
manVars = c("mpg", "cyl", "disp")
tmp = xmu_make_mxData(data= mtcars, type = "Auto", manifests = manVars);
tmp$type == "raw" # TRUE

# ==============================
# = All continuous WLS example =
# ==============================
tmp = xmu_make_mxData(data= mtcars, type = "WLS" , manifests = manVars, verbose= TRUE)
tmp$type == "raw" # TRUE (WLS is triggered by the fit function, not the data type)

# ============================
# = Missing data WLS example =
# ============================
tmp = mtcars; tmp[1, "mpg"] = NA # add NA
tmp = xmu_make_mxData(data= tmp, type = "WLS", manifests = manVars, verbose= TRUE)

# ========================
# = already mxData example =
# ========================
m1 = umxRAM("auto", data = mxData(mtcars, type = "raw"),
umxPath(var= "wt"),
umxPath(mean=  "wt")
)

# ========================
# = Cov and cor examples =
# ========================
tmp = xmu_make_mxData(data= mtcars, type = "cov", manifests = c("mpg", "cyl"))
tmp = xmu_make_mxData(data= mtcars, type = "cor", manifests = c("mpg", "cyl"))
tmp = xmu_make_mxData(data= cov(mtcars[, c("mpg", "cyl")]),
        type = "cov", manifests = c("mpg", "cyl"), numObs=200)
```

```
# mxData input examples
tmp = mxData(cov(mtcars[, c("mpg", "cyl")]), type = "cov", numObs= 100)
xmu_make_mxData(data= tmp, type = "cor", manifests = c("mpg", "cyl")) # consume mxData
xmu_make_mxData(data= tmp, type = "cor", manifests = c("mpg"))      # trim existing mxData
xmu_make_mxData(data= tmp, type = "cor") # no manifests specified (use all)
xmu_make_mxData(data= tmp, manifests = c("mpg", "cyl")) # auto

# ======================
# = Pass string through =
# ======================
xmu_make_mxData(data= c("a", "b", "c"), type = "Auto")
```

---

xmu_make_TwinSuperModel

Helper to make a basic top, MZ, and DZ model.

Description

xmu_make_TwinSuperModel makes basic twin model containing top, MZ, and DZ models. It intelligently handles thresholds for ordinal data, and means model for covariates matrices in the twin models if needed.

It's the replacement for xmu_assemble_twin_supermodel approach.

Usage

```
xmu_make_TwinSuperModel(
  name = "twin_super",
  mzData,
  dzData,
  selDVs,
  selCovs = NULL,
  sep = NULL,
  type = c("Auto", "FIML", "cov", "cor", "WLS", "DWLS", "ULS"),
  allContinuousMethod = c("cumulants", "marginals"),
  numObsMZ = NULL,
  numObsDZ = NULL,
  nSib = 2,
  equateMeans = TRUE,
  weightVar = NULL,
  bVector = FALSE,
  dropMissingDef = TRUE,
  verbose = FALSE
)
```

Arguments

name	for the supermodel
mzData	Dataframe containing the MZ data
dzData	Dataframe containing the DZ data
selDVs	List of manifest base names (e.g. BMI, NOT 'BMI_T1') (OR, you don't set "sep", the full variable names)
selCovs	List of covariate base names (e.g. age, NOT 'age_T1') (OR, you don't set "sep", the full variable names)
sep	string used to expand selDVs into selVars, i.e., "_T" to expand BMI into BMI_T1 and BMI_T2 (optional but STRONGLY encouraged)
type	One of 'Auto','FIML','cov', 'cor', 'WLS','DWLS', or 'ULS'. Auto tries to react to the incoming mxData type (raw/cov).
allContinuousMethod	"cumulants" or "marginals". Used in all-continuous WLS data to determine if a means model needed.
numObsMZ	Number of MZ observations contributing (for summary data only)
numObsDZ	Number of DZ observations contributing (for summary data only)
nSib	Number of members per family (default = 2)
equateMeans	Whether to equate T1 and T2 means (default = TRUE).
weightVar	If provided, a vector objective will be used to weight the data. (default = NULL).
bVector	Whether to compute row-wise likelihoods (defaults to FALSE).
dropMissingDef	Whether to automatically drop missing def var rows for the user (default = TRUE). You get a polite note.
verbose	(default = FALSE)

Details

xmu_make_TwinSuperModel is used in twin models (e.g.umxCP(), umxACE() and umxACEv() and will be added to the other models: umxGxE(), umxIP(), simplifying code maintenance.

It takes mzData and dzData, a list of the selDVs to analyse and optional selCovs (as well as sep and nSib), along with other relevant information such as whether the user wants to equateMeans. It can also handle a weightVar.

If covariates are passed in these are included in the means model (via a call to xmuTwinUpgradeMeansToCovariateModel.

Modeling

Matrices created

top model

For raw and WLS data, top contains a expMeans matrix (if needed). For summary data, the top model contains only a name.

For ordinal data, top gains top.threshMat (from a call to umxThresholdMatrix()).

For covariates, top stores the intercepts matrix and a betaDef matrix. These are then used to make expMeans in MZ and DZ.

MZ and DZ models

MZ and DZ contain the data, and an expectation referencing top.expCovMZ and top.expMean, and, vector = bVector. For continuous raw data, MZ and DZ contain OpenMx::mxExpectationNormal() and OpenMx::mxFitFunctionML(). For WLS these the fit function is switched to OpenMx::mxFitFunctionWLS() with appropriate type and allContinuousMethod.

For binary, a constraint and algebras are included to constrain Vtot (A+C+E) to 1.

If a weightVar is detected, these columns are used to create a row-weighted MZ and DZ models.

If equateMeans is TRUE, then the Twin-2 vars in the mean matrix are equated by label with Twin-1.

Decent starts are guessed from the data. varStarts is computed as sqrt(variance)/3 of the DVs and meanStarts as the variable means. For raw data, a check is made for ordered variables. For Binary variables, means are fixed at 0 and total variance (A+C+E) is fixed at 1. For ordinal variables, the first 2 thresholds are fixed.

Where needed, e.g. continuous raw data, top adds a means matrix "expMean". For ordinal data, top adds a umxThresholdMatrix().

If binary variables are present, matrices and a constraint to hold A+C+E == 1 are added to top.

If a weight variable is offered up, an mzWeightMatrix will be added.

Data handling

In terms of data handling, xmu_make_TwinSuperModel was primarily designed to take data.frames and process these into mxData. It can also, however, handle cov and mxData input.

It can process data into all the types supported by mxData.

Raw data input with a target of cov or cor type requires the numObsMZ and numObsDZ to be set.

Type "WLS", "DWLS", or "ULS", data remain raw, but are handled as WLS in the OpenMx::mxFitFunctionWLS().

Unused columns are dropped.

If you pass in raw data, you can't request type cov/cor yet. Will work on this if desired.

Value

- mxModel()s for top, MZ and DZ.

See Also

Other xmu internal not for end user: umxModel(), umxRenameMatrix(), umxTwinMaker(), umx_APA_pval(), umx_fun_mean_sd(), umx_get_bracket_addresses(), umx_make(), umx_standardize(), umx_string_to_algebra(), umx, xmuHasSquareBrackets(), xmuLabel_MATRIX_Model(), xmuLabel_Matrix(), xmuLabel_RAM_Model(), xmuMI(), xmuMakeDeviationThresholdsMatrices(), xmuMakeOneHeadedPathsFromPathList(), xmuMakeTwoHeadedPathsFromPathList(), xmuMaxLevels(), xmuMinLevels(), xmuPropagateLabels(), xmuRAM2Ordinal(), xmuTwinSuper_Continuous(), xmuTwinUpgradeMeansToCovariateModel(), xmu_CI_merge(), xmu_CI_stash(), xmu_DF_to_mxData_TypeCov(), xmu_PadAndPruneForDefVars(), xmu_cell_is_on(), xmu_check_levels_identical(), xmu_check_needs_means(), xmu_check_variance(), xmu_clean_label(), xmu_data_missing(), xmu_data_swap_a_block(), xmu_describe_data_WLS(), xmu_dot_make_paths(), xmu_dot_make_residuals(), xmu_dot_maker(), xmu_dot_move_ranks(), xmu_dot_rank_str(), xmu_extract_column(), xmu_get_CI(), xmu_lavaan_process_group(), xmu_make_bin_cont_pair_data(), xmu_make_mxData(), xmu_match.arg(), xmu_name_from_lavaan_str(), xmu_path2twin(), xmu_path_regex(), xmu_safe_run_summary(), xmu_set_sep_from_suffix(),

xmu_show_fit_or_comparison(), xmu_simplex_corner(), xmu_standardize_ACEcov(), xmu_standardize_ACEv(), xmu_standardize_ACE(), xmu_standardize_CP(), xmu_standardize_IP(), xmu_standardize_RAM(), xmu_standardize_SexLim(), xmu_standardize_Simplex(), xmu_start_value_list(), xmu_starts(), xmu_twin_add_WeightMatrices(), xmu_twin_check(), xmu_twin_get_var_names(), xmu_twin_upgrade_selDvs2SelV

Examples

```
# ==============
# = Continuous =
# ==============
library(umx)
data(twinData)
twinData = umx_scale(twinData, varsToScale= c('ht1','ht2'))
mzData = twinData[twinData$zygosity %in%  "MZFF",]
dzData = twinData[twinData$zygosity %in%  "DZFF",]
m1= xmu_make_TwinSuperModel(mzData=mzData, dzData=dzData, selDVs=c("wt","ht"), sep="", nSib=2)
names(m1) # "top" "MZ"  "DZ"
class(m1$MZ$fitfunction)[[1]] == "MxFitFunctionML"

# ====================
# = With a covariate =
# ====================

m1= xmu_make_TwinSuperModel(mzData=mzData, dzData=dzData,
selDVs= "wt", selCovs= "age", sep="", nSib=2)
m1$top$intercept$labels
m1$MZ$expMean

# ===============
# = WLS example =
# ===============
m1=xmu_make_TwinSuperModel(mzData=mzData, dzData=dzData,selDVs=c("wt","ht"),sep="",type="WLS")
class(m1$MZ$fitfunction)[[1]] == "MxFitFunctionWLS"
m1$MZ$fitfunction$type =="WLS"
# Check default all-continuous method
m1$MZ$fitfunction$continuousType == "cumulants"

# Choose non-default type (DWLS)
m1= xmu_make_TwinSuperModel(mzData= mzData, dzData= dzData,
selDVs= c("wt","ht"), sep="", type="DWLS")
m1$MZ$fitfunction$type =="DWLS"
class(m1$MZ$fitfunction)[[1]] == "MxFitFunctionWLS"

# Switch WLS method
m1 = xmu_make_TwinSuperModel(mzData= mzData, dzData= dzData, selDVs= c("wt","ht"), sep= "",
  type = "WLS", allContinuousMethod = "marginals")
m1$MZ$fitfunction$continuousType == "marginals"
class(m1$MZ$fitfunction)[[1]] == "MxFitFunctionWLS"


# ============================================
# = Bivariate continuous and ordinal example =
```

```
# ==========================================
data(twinData)
selDVs = c("wt", "obese")
# Cut BMI column to form ordinal obesity variables
ordDVs          = c("obese1", "obese2")
obesityLevels   = c('normal', 'overweight', 'obese')
cutPoints       = quantile(twinData[, "bmi1"], probs = c(.5, .2), na.rm = TRUE)
twinData$obese1 = cut(twinData$bmi1, breaks = c(-Inf, cutPoints, Inf), labels = obesityLevels)
twinData$obese2 = cut(twinData$bmi2, breaks = c(-Inf, cutPoints, Inf), labels = obesityLevels)
# Make the ordinal variables into mxFactors (ensure ordered is TRUE, and require levels)
twinData[, ordDVs] = umxFactor(twinData[, ordDVs])
mzData = twinData[twinData$zygosity %in%  "MZFF",]
dzData = twinData[twinData$zygosity %in%  "DZFF",]
m1 = xmu_make_TwinSuperModel(mzData= mzData, dzData= dzData, selDVs= selDVs, sep="", nSib= 2)
names(m1) # "top" "MZ"  "DZ"

# ==============
# = One binary =
# ==============
data(twinData)
cutPoints       = quantile(twinData[, "bmi1"], probs = .2, na.rm = TRUE)
obesityLevels   = c('normal', 'obese')
twinData$obese1 = cut(twinData$bmi1, breaks = c(-Inf, cutPoints, Inf), labels = obesityLevels)
twinData$obese2 = cut(twinData$bmi2, breaks = c(-Inf, cutPoints, Inf), labels = obesityLevels)
ordDVs = c("obese1", "obese2")
twinData[, ordDVs] = umxFactor(twinData[, ordDVs])
selDVs = c("wt", "obese")
mzData = twinData[twinData$zygosity %in% "MZFF",]
dzData = twinData[twinData$zygosity %in% "DZFF",]
m1 = xmu_make_TwinSuperModel(mzData= mzData, dzData= dzData, selDVs= selDVs, sep= "", nSib= 2)

# ========================================
# = Cov data (calls xmuTwinSuper_CovCor) =
# ========================================

data(twinData)
mzData =cov(twinData[twinData$zygosity %in% "MZFF", tvars(c("wt","ht"), sep="")], use="complete")
dzData =cov(twinData[twinData$zygosity %in% "DZFF", tvars(c("wt","ht"), sep="")], use="complete")
m1 = xmu_make_TwinSuperModel(mzData= mzData, dzData= dzData, selDVs= "wt", sep= "",
nSib= 2, numObsMZ = 100, numObsDZ = 100, verbose=TRUE)
class(m1$MZ$fitfunction)[[1]] =="MxFitFunctionML"
dimnames(m1$MZ$data$observed)[[1]]==c("wt1", "wt2")
```

`xmu_match.arg` *Select first item in list of options, while being flexible about choices.*

Description

Like a smart version of `match.arg()`: Handles selecting parameter options when default is a list. Unlike `match.arg()` `xmu_match.arg` allows items not in the list.

Usage

```
xmu_match.arg(x, option_list, check = TRUE)
```

Arguments

x	the value chosen (may be the default option list)
option_list	A vector of valid options
check	Whether to check that single items are in the list. Set false to accept abbreviations (defaults to TRUE)

Value

- one validated option

References

- https://www.github.com/tbates/umx

See Also

- match.arg()

Other xmu internal not for end user: umxModel(), umxRenameMatrix(), umxTwinMaker(), umx_APA_pval(), umx_fun_mean_sd(), umx_get_bracket_addresses(), umx_make(), umx_standardize(), umx_string_to_algebra(), umx, xmuHasSquareBrackets(), xmuLabel_MATRIX_Model(), xmuLabel_Matrix(), xmuLabel_RAM_Model(), xmuMI(), xmuMakeDeviationThresholdsMatrices(), xmuMakeOneHeadedPathsFromPathList(), xmuMakeTwoHeadedPathsFromPathList(), xmuMaxLevels(), xmuMinLevels(), xmuPropagateLabels(), xmuRAM2Ordinal(), xmuTwinSuper_Continuous(), xmuTwinUpgradeMeansToCovariateModel(), xmu_CI_merge(), xmu_CI_stash(), xmu_DF_to_mxData_TypeCov(), xmu_PadAndPruneForDefVars(), xmu_cell_is_on(), xmu_check_levels_identical(), xmu_check_needs_means(), xmu_check_variance(), xmu_clean_label(), xmu_data_missing(), xmu_data_swap_a_block(), xmu_describe_data_WLS(), xmu_dot_make_paths(), xmu_dot_make_residuals(), xmu_dot_maker(), xmu_dot_move_ranks(), xmu_dot_rank_str(), xmu_extract_column(), xmu_get_CI(), xmu_lavaan_process_group(), xmu_make_TwinSuperModel(), xmu_make_bin_cont_pair_data(), xmu_make_mxData(), xmu_name_from_lavaan_str(), xmu_path2twin(), xmu_path_regex(), xmu_safe_run_summary(), xmu_set_sep_from_suffix(), xmu_show_fit_or_comparison(), xmu_simplex_corner(), xmu_standardize_ACEcov(), xmu_standardize_ACEv(), xmu_standardize_ACE(), xmu_standardize_CP(), xmu_standardize_IP(), xmu_standardize_RAM(), xmu_standardize_SexLim(), xmu_standardize_Simplex(), xmu_start_value_list(), xmu_starts(), xmu_twin_add_WeightMatrices(), xmu_twin_check(), xmu_twin_get_var_names(), xmu_twin_upgrade_selDvs2SelV

Examples

```
option_list = c("default", "par.observed", "empirical")

xmu_match.arg("par.observed", option_list)
xmu_match.arg("allow me", option_list, check = FALSE)
xmu_match.arg(option_list, option_list)
option_list = c(NULL, "par.observed", "empirical")
 # fails with NULL!!!!!
xmu_match.arg(option_list, option_list)
```

```
option_list = c(NA, "par.observed", "empirical")
xmu_match.arg(option_list, option_list) # use NA instead
option_list = c(TRUE, FALSE, NA)
xmu_match.arg(option_list, option_list) # works with non character
# An example of checking a bad item and stopping
## Not run:
xmu_match.arg("bad", option_list)

## End(Not run)
```

`xmu_name_from_lavaan_str` *Find name for model*

Description

Use name if provided. If first line contains a #, uses this line as name. Else use default.

Usage

```
xmu_name_from_lavaan_str(lavaanString = NULL, name = NA, default = "m1")
```

Arguments

`lavaanString`	A model string, possibly with # model name on line 1.
`name`	A desired model name (optional).
`default`	A default name if nothing else found.

Value

- A name string

References

- https://github.com/tbates/umx, https://tbates.github.io

See Also

- `umxRAM()`

Other xmu internal not for end user: `umxModel()`, `umxRenameMatrix()`, `umxTwinMaker()`, `umx_APA_pval()`, `umx_fun_mean_sd()`, `umx_get_bracket_addresses()`, `umx_make()`, `umx_standardize()`, `umx_string_to_algebra()`, `umx`, `xmuHasSquareBrackets()`, `xmuLabel_MATRIX_Model()`, `xmuLabel_Matrix()`, `xmuLabel_RAM_Model()`, `xmuMI()`, `xmuMakeDeviationThresholdsMatrices()`, `xmuMakeOneHeadedPathsFromPathList()`, `xmuMakeTwoHeadedPathsFromPathList()`, `xmuMaxLevels()`, `xmuMinLevels()`, `xmuPropagateLabels()`, `xmuRAM2Ordinal()`, `xmuTwinSuper_Continuous()`, `xmuTwinUpgradeMeansToCovariateModel()`, `xmu_CI_merge()`, `xmu_CI_stash()`, `xmu_DF_to_mxData_TypeCov()`, `xmu_PadAndPruneForDefVars()`, `xmu_cell_is_on()`, `xmu_check_levels_identical()`, `xmu_check_needs_means()`, `xmu_check_variance()`,

xmu_clean_label(), xmu_data_missing(), xmu_data_swap_a_block(), xmu_describe_data_WLS(), xmu_dot_make_paths(), xmu_dot_make_residuals(), xmu_dot_maker(), xmu_dot_move_ranks(), xmu_dot_rank_str(), xmu_extract_column(), xmu_get_CI(), xmu_lavaan_process_group(), xmu_make_TwinSuperModel(), xmu_make_bin_cont_pair_data(), xmu_make_mxData(), xmu_match.arg(), xmu_path2twin(), xmu_path_regex(), xmu_safe_run_summary(), xmu_set_sep_from_suffix(), xmu_show_fit_or_comparison(), xmu_simplex_corner(), xmu_standardize_ACEcov(), xmu_standardize_ACEv(), xmu_standardize_ACE(), xmu_standardize_CP(), xmu_standardize_IP(), xmu_standardize_RAM(), xmu_standardize_SexLim(), xmu_standardize_Simplex(), xmu_start_value_list(), xmu_starts(), xmu_twin_add_WeightMatrices(), xmu_twin_check(), xmu_twin_get_var_names(), xmu_twin_upgrade_selDvs2SelV

Examples

```
"m1" == xmu_name_from_lavaan_str("x~~x")
"bob" == xmu_name_from_lavaan_str(name = "bob")
"my_model" == xmu_name_from_lavaan_str("# my model")
```

---

xmu_PadAndPruneForDefVars

Where all data are missing for a twin, add default values for definition variables, allowing the row to be kept

---

Description

Replaces NAs in definition slots with the mean for that variable ONLY where all data are missing for that twin.

Usage

```
xmu_PadAndPruneForDefVars(
  df,
  varNames,
  defNames,
  suffixes,
  highDefValue = 99,
  rm = c("drop_missing_def", "pad_with_mean")
)
```

Arguments

df	The dataframe to process
varNames	list of names of the variables being analysed
defNames	list of covariates
suffixes	that map names on columns in df (i.e., c("T1", "T2"))
highDefValue	What to replace missing definition variables (covariates) with. Default = 99
rm	= how to handle missing values in the varNames. Default is "drop_missing_def", "pad_with_mean")

Value

- dataframe

See Also

Other xmu internal not for end user: umxModel(), umxRenameMatrix(), umxTwinMaker(), umx_APA_pval(), umx_fun_mean_sd(), umx_get_bracket_addresses(), umx_make(), umx_standardize(), umx_string_to_algebra(), umx, xmuHasSquareBrackets(), xmuLabel_MATRIX_Model(), xmuLabel_Matrix(), xmuLabel_RAM_Model(), xmuMI(), xmuMakeDeviationThresholdsMatrices(), xmuMakeOneHeadedPathsFromPathList(), xmuMakeTwoHeadedPathsFromPathList(), xmuMaxLevels(), xmuMinLevels(), xmuPropagateLabels(), xmuRAM2Ordinal(), xmuTwinSuper_Continuous(), xmuTwinUpgradeMeansToCovariateModel(), xmu_CI_merge(), xmu_CI_stash(), xmu_DF_to_mxData_TypeCov(), xmu_cell_is_on(), xmu_check_levels_identical, xmu_check_needs_means(), xmu_check_variance(), xmu_clean_label(), xmu_data_missing(), xmu_data_swap_a_block(), xmu_describe_data_WLS(), xmu_dot_make_paths(), xmu_dot_make_residuals(), xmu_dot_maker(), xmu_dot_move_ranks(), xmu_dot_rank_str(), xmu_extract_column(), xmu_get_CI(), xmu_lavaan_process_group(), xmu_make_TwinSuperModel(), xmu_make_bin_cont_pair_data(), xmu_make_mxData(), xmu_match.arg(), xmu_name_from_lavaan_str(), xmu_path2twin(), xmu_path_regex(), xmu_safe_run_summary(), xmu_set_sep_from_suffix(), xmu_show_fit_or_comparison(), xmu_simplex_corner(), xmu_standardize_ACEcov(), xmu_standardize_ACEv(), xmu_standardize_ACE(), xmu_standardize_CP(), xmu_standardize_IP(), xmu_standardize_RAM(), xmu_standardize_SexLim(), xmu_standardize_Simplex(), xmu_start_value_list(), xmu_starts(), xmu_twin_add_WeightMatrices(), xmu_twin_check(), xmu_twin_get_var_names(), xmu_twin_upgrade_selDvs2SelVars()

Examples

```
## Not run:
data(twinData)
sum(is.na(twinData$ht1))
df = xmu_PadAndPruneForDefVars(twinData, varNames = "ht", defNames = "wt", c("1", "2"))

## End(Not run)
```

---

xmu_path2twin *Re-name variables in umxPaths to twin versions*

Description

xmu_path2twin takes a collection of paths that use base variable names, and returns a model with twin names.

Usage

```
xmu_path2twin(paths, thisTwin = 1, sep = "_T")
```

Arguments

paths	A collection of paths using base variable names.
thisTwin	The twin we are making (i.e., "_T1", or "_T2")
sep	The separator (default "_T")

Details

A path like a to b will be returned as a_T1 to b_T1.

Value

- list of relabeled paths

See Also

- umxTwinMaker(), umxRAM()

Other xmu internal not for end user: umxModel(), umxRenameMatrix(), umxTwinMaker(), umx_APA_pval(), umx_fun_mean_sd(), umx_get_bracket_addresses(), umx_make(), umx_standardize(), umx_string_to_algebra(), umx, xmuHasSquareBrackets(), xmuLabel_MATRIX_Model(), xmuLabel_Matrix(), xmuLabel_RAM_Model(), xmuMI(), xmuMakeDeviationThresholdsMatrices(), xmuMakeOneHeadedPathsFromPathList(), xmuMakeTwoHeadedPathsFromPathList(), xmuMaxLevels(), xmuMinLevels(), xmuPropagateLabels(), xmuRAM2Ordinal(), xmuTwinSuper_Continuous(), xmuTwinUpgradeMeansToCovariateModel(), xmu_CI_merge(), xmu_CI_stash(), xmu_DF_to_mxData_TypeCov(), xmu_PadAndPruneForDefVars(), xmu_cell_is_on(), xmu_check_levels_identical(), xmu_check_needs_means(), xmu_check_variance(), xmu_clean_label(), xmu_data_missing(), xmu_data_swap_a_block(), xmu_describe_data_WLS(), xmu_dot_make_paths(), xmu_dot_make_residuals(), xmu_dot_maker(), xmu_dot_move_ranks(), xmu_dot_rank_str(), xmu_extract_column(), xmu_get_CI(), xmu_lavaan_process_group(), xmu_make_TwinSuperModel(), xmu_make_bin_cont_pair_data(), xmu_make_mxData(), xmu_match.arg(), xmu_name_from_lavaan_str(), xmu_path_regex(), xmu_safe_run_summary(), xmu_set_sep_from_suffix(), xmu_show_fit_or_comparison(), xmu_simplex_corner(), xmu_standardize_ACEcov(), xmu_standardize_ACEv(), xmu_standardize_ACE(), xmu_standardize_CP(), xmu_standardize_IP(), xmu_standardize_RAM(), xmu_standardize_SexLim(), xmu_standardize_Simplex(), xmu_start_value_list(), xmu_starts(), xmu_twin_add_WeightMatrices(), xmu_twin_check(), xmu_twin_get_var_names(), xmu_twin_upgrade_selDvs2Sel

Examples

```
twin1PathList = c(
umxPath(v1m0 = c("a1", 'c1', "e1")),
umxPath(fromEach = c("a1", 'c1', "e1"), to = "NFC3", values=.2)
)
xmu_path2twin(twin1PathList, thisTwin = 2)
```

---

xmu_path_regex *Re-name variables umxPaths to twin versions*

Description

xmu_path2twin takes a collection of umxPath()s (use base variable names), and returns a model for both twins (and using the expanded variable names).

Usage

```
xmu_path_regex(input, pattern = NA, replacement = NA, ignore = "one")
```

Arguments

input	vector of path labels
pattern	= pattern to match and replace
replacement	= replacement string
ignore	Labels to ignore (reserved words like "one")

Details

A path like a to b will be returned as a_T1 to b_T1.

Value

- renamed paths

References

- tutorials, tutorials

See Also

- xmu_path2twin(), umxTwinMaker()

Other xmu internal not for end user: umxModel(), umxRenameMatrix(), umxTwinMaker(), umx_APA_pval(), umx_fun_mean_sd(), umx_get_bracket_addresses(), umx_make(), umx_standardize(), umx_string_to_algebra(), umx, xmuHasSquareBrackets(), xmuLabel_MATRIX_Model(), xmuLabel_Matrix(), xmuLabel_RAM_Model(), xmuMI(), xmuMakeDeviationThresholdsMatrices(), xmuMakeOneHeadedPathsFromPathList(), xmuMakeTwoHeadedPathsFromPathList(), xmuMaxLevels(), xmuMinLevels(), xmuPropagateLabels(), xmuRAM2Ordinal(), xmuTwinSuper_Continuous(), xmuTwinUpgradeMeansToCovariateModel(), xmu_CI_merge(), xmu_CI_stash(), xmu_DF_to_mxData_TypeCov(), xmu_PadAndPruneForDefVars(), xmu_cell_is_on(), xmu_check_levels_identical(), xmu_check_needs_means(), xmu_check_variance(), xmu_clean_label(), xmu_data_missing(), xmu_data_swap_a_block(), xmu_describe_data_WLS(), xmu_dot_make_paths(), xmu_dot_make_residuals(), xmu_dot_maker(), xmu_dot_move_ranks(), xmu_dot_rank_str(), xmu_extract_column(), xmu_get_CI(), xmu_lavaan_process_group(), xmu_make_TwinSuperModel(), xmu_make_bin_cont_pair_data(), xmu_make_mxData(), xmu_match.arg(), xmu_name_from_lavaan_str(), xmu_path2twin(), xmu_safe_run_summary(), xmu_set_sep_from_suffix(), xmu_show_fit_or_comparison(), xmu_simplex_corner(), xmu_standardize_ACEcov(), xmu_standardize_ACEv(), xmu_standardize_ACE(), xmu_standardize_CP(), xmu_standardize_IP(), xmu_standardize_RAM(), xmu_standardize_SexLim(), xmu_standardize_Simplex(), xmu_start_value_list(), xmu_starts(), xmu_twin_add_WeightMatrices(), xmu_twin_check(), xmu_twin_get_var_names(), xmu_twin_upgrade_selDvs2Sel

Examples

```
xmu_path_regex(c("a", "one", "b"), pattern = "$", replacement = "_T1")
# "a_T1" "one"  "b_T1"
```

`xmu_safe_run_summary` *Safely run and summarize a model*

Description

The main benefit is that it returns the model, even if it can't be run.

The function will run the model if requested, wrapped in `tryCatch()` to avoid throwing an error. If summary = TRUE then `umxSummary()` is requested (again, wrapped in try).

note: If `autoRun` is logical, then it over-rides `summary` to match `autoRun`. This is useful for easy use `umxRAM()` and twin models.

Usage

```
xmu_safe_run_summary(
  model1,
  model2 = NULL,
  autoRun = TRUE,
  tryHard = c("no", "yes", "ordinal", "search"),
  summary = !umx_set_silent(silent = TRUE),
  std = "default",
  comparison = TRUE,
  digits = 3
)
```

Arguments

`model1`	The model to attempt to run and summarize.
`model2`	Optional second model to compare with model1.
`autoRun`	Whether to run or not (default = TRUE) Options are FALSE and "if needed".
`tryHard`	Default ('no') uses normal mxRun. "yes" uses mxTryHard. Other options: "ordinal", "search"
`summary`	Whether to print model summary (default = autoRun).
`std`	What to print in summary. "default" = the object's summary default. FALSE = raw, TRUE = standardize, NULL = omit parameter table.
`comparison`	Toggle to allow not making comparison, even if second model is provided (more flexible in programming).
`digits`	Rounding precision in tables and plots

Value

- `mxModel()`

See Also

- mxTryHard()

Other xmu internal not for end user: umxModel(), umxRenameMatrix(), umxTwinMaker(), umx_APA_pval(), umx_fun_mean_sd(), umx_get_bracket_addresses(), umx_make(), umx_standardize(), umx_string_to_algebra(), umx, xmuHasSquareBrackets(), xmuLabel_MATRIX_Model(), xmuLabel_Matrix(), xmuLabel_RAM_Model(), xmuMI(), xmuMakeDeviationThresholdsMatrices(), xmuMakeOneHeadedPathsFromPathList(), xmuMakeTwoHeadedPathsFromPathList(), xmuMaxLevels(), xmuMinLevels(), xmuPropagateLabels(), xmuRAM2Ordinal(), xmuTwinSuper_Continuous(), xmuTwinUpgradeMeansToCovariateModel(), xmu_CI_merge(), xmu_CI_stash(), xmu_DF_to_mxData_TypeCov(), xmu_PadAndPruneForDefVars(), xmu_cell_is_on(), xmu_check_levels_identical(), xmu_check_needs_means(), xmu_check_variance(), xmu_clean_label(), xmu_data_missing(), xmu_data_swap_a_block(), xmu_describe_data_WLS(), xmu_dot_make_paths(), xmu_dot_make_residuals(), xmu_dot_maker(), xmu_dot_move_ranks(), xmu_dot_rank_str(), xmu_extract_column(), xmu_get_CI(), xmu_lavaan_process_group(), xmu_make_TwinSuperModel(), xmu_make_bin_cont_pair_data(), xmu_make_mxData(), xmu_match.arg(), xmu_name_from_lavaan_str(), xmu_path2twin(), xmu_path_regex(), xmu_set_sep_from_suffix(), xmu_show_fit_or_comparison(), xmu_simplex_corner(), xmu_standardize_ACEcov(), xmu_standardize_ACEv(), xmu_standardize_ACE(), xmu_standardize_CP(), xmu_standardize_IP(), xmu_standardize_RAM(), xmu_standardize_SexLim(), xmu_standardize_Simplex(), xmu_start_value_list(), xmu_starts(), xmu_twin_add_WeightMatrices(), xmu_twin_check(), xmu_twin_get_var_names(), xmu_twin_upgrade_selDvs2Sel

Examples

```
## Not run:
m1 = umxRAM("tim", data = mtcars,
umxPath(c("wt", "disp"), to = "mpg"),
umxPath("wt", with = "disp"),
umxPath(v.m. = c("wt", "disp", "mpg"))
)
m2 = umxModify(m1, "wt_to_mpg")

# Summary ignored if run is false
xmu_safe_run_summary(m1, autoRun = FALSE, summary = TRUE)
# Run, no summary
xmu_safe_run_summary(m1, autoRun = TRUE, summary = FALSE)
# Default summary is just fit string
xmu_safe_run_summary(m1, autoRun = TRUE, summary = TRUE)
# Show std parameters
xmu_safe_run_summary(m1, autoRun = TRUE, summary = TRUE, std = TRUE)
# Run + Summary + comparison
xmu_safe_run_summary(m1, m2, autoRun = TRUE, summary = TRUE)
# Run + Summary + no comparison
xmu_safe_run_summary(m1, m2, autoRun = TRUE, summary = TRUE, std = TRUE, comparison= FALSE)


## End(Not run)
```

`xmu_set_sep_from_suffix` *Just a helper to cope with deprecated suffix lying around.*

Description

Returns either suffix or sep, with a deprecation warning if suffix is set.

Usage

```
xmu_set_sep_from_suffix(sep, suffix)
```

Arguments

sep	The separator (if suffix != 'deprecated', then this is returned).
suffix	The suffix, defaults to 'deprecated'.

Value

- sep

See Also

Other xmu internal not for end user: umxModel(), umxRenameMatrix(), umxTwinMaker(), umx_APA_pval(), umx_fun_mean_sd(), umx_get_bracket_addresses(), umx_make(), umx_standardize(), umx_string_to_algebra(), umx, xmuHasSquareBrackets(), xmuLabel_MATRIX_Model(), xmuLabel_Matrix(), xmuLabel_RAM_Model(), xmuMI(), xmuMakeDeviationThresholdsMatrices(), xmuMakeOneHeadedPathsFromPathList(), xmuMakeTwoHeadedPathsFromPathList(), xmuMaxLevels(), xmuMinLevels(), xmuPropagateLabels(), xmuRAM2Ordinal(), xmuTwinSuper_Continuous(), xmuTwinUpgradeMeansToCovariateModel(), xmu_CI_merge(), xmu_CI_stash(), xmu_DF_to_mxData_TypeCov(), xmu_PadAndPruneForDefVars(), xmu_cell_is_on(), xmu_check_levels_identical(), xmu_check_needs_means(), xmu_check_variance(), xmu_clean_label(), xmu_data_missing(), xmu_data_swap_a_block(), xmu_describe_data_WLS(), xmu_dot_make_paths(), xmu_dot_make_residuals(), xmu_dot_maker(), xmu_dot_move_ranks(), xmu_dot_rank_str(), xmu_extract_column(), xmu_get_CI(), xmu_lavaan_process_group(), xmu_make_TwinSuperModel(), xmu_make_bin_cont_pair_data(), xmu_make_mxData(), xmu_match.arg(), xmu_name_from_lavaan_str(), xmu_path2twin(), xmu_path_regex(), xmu_safe_run_summary(), xmu_show_fit_or_comparison(), xmu_simplex_corner(), xmu_standardize_ACEcov(), xmu_standardize_ACEv(), xmu_standardize_ACE(), xmu_standardize_CP(), xmu_standardize_IP(), xmu_standardize_RAM(), xmu_standardize_SexLim(), xmu_standardize_Simplex(), xmu_start_value_list(), xmu_starts(), xmu_twin_add_WeightMatrices(), xmu_twin_check(), xmu_twin_get_var_names(), xmu_twin_upgrade_selDvs2SelV

Examples

```
xmu_set_sep_from_suffix(sep = "_T", suffix = "deprecated")
```

xmu_show_fit_or_comparison *Show model logLik of model or print comparison table*

Description

Just a helper to show the logLik of a model or print a comparison table.

Usage

```
xmu_show_fit_or_comparison(model, comparison = NULL, digits = 2)
```

Arguments

model	an mxModel() to report on
comparison	If not NULL, used as comparison model
digits	(default = 2)

Value

None

See Also

- umxSummary()

Other xmu internal not for end user: umxModel(), umxRenameMatrix(), umxTwinMaker(), umx_APA_pval(), umx_fun_mean_sd(), umx_get_bracket_addresses(), umx_make(), umx_standardize(), umx_string_to_algebra(), umx, xmuHasSquareBrackets(), xmuLabel_MATRIX_Model(), xmuLabel_Matrix(), xmuLabel_RAM_Model(), xmuMI(), xmuMakeDeviationThresholdsMatrices(), xmuMakeOneHeadedPathsFromPathList(), xmuMakeTwoHeadedPathsFromPathList(), xmuMaxLevels(), xmuMinLevels(), xmuPropagateLabels(), xmuRAM2Ordinal(), xmuTwinSuper_Continuous(), xmuTwinUpgradeMeansToCovariateModel(), xmu_CI_merge(), xmu_CI_stash(), xmu_DF_to_mxData_TypeCov(), xmu_PadAndPruneForDefVars(), xmu_cell_is_on(), xmu_check_levels_identical(), xmu_check_needs_means(), xmu_check_variance(), xmu_clean_label(), xmu_data_missing(), xmu_data_swap_a_block(), xmu_describe_data_WLS(), xmu_dot_make_paths(), xmu_dot_make_residuals(), xmu_dot_maker(), xmu_dot_move_ranks(), xmu_dot_rank_str(), xmu_extract_column(), xmu_get_CI(), xmu_lavaan_process_group(), xmu_make_TwinSuperModel(), xmu_make_bin_cont_pair_data(), xmu_make_mxData(), xmu_match.arg(), xmu_name_from_lavaan_str(), xmu_path2twin(), xmu_path_regex(), xmu_safe_run_summary(), xmu_set_sep_from_suffix(), xmu_simplex_corner(), xmu_standardize_ACEcov(), xmu_standardize_ACEv(), xmu_standardize_ACE(), xmu_standardize_CP(), xmu_standardize_IP(), xmu_standardize_RAM(), xmu_standardize_SexLim(), xmu_standardize_Simplex(), xmu_start_value_list(), xmu_starts(), xmu_twin_add_WeightMatrices(), xmu_twin_check(), xmu_twin_get_var_names(), xmu_twin_upgrade_selDvs2Sel

Examples

```
## Not run:
xmu_show_fit_or_comparison(model, comparison, digits=3)

## End(Not run)
```

`xmu_simplex_corner` *Internal function to help building simplex models*

Description

internal function to help building simplex models is a function which

Usage

```
xmu_simplex_corner(x, start = 0.9)
```

Arguments

x	size of matrix, or an umxMatrix() of which to free the bottom triangle.
start	a default start value for the freed items.

Value

- umxMatrix()

See Also

- umxMatrix()

Other xmu internal not for end user: umxModel(), umxRenameMatrix(), umxTwinMaker(), umx_APA_pval(), umx_fun_mean_sd(), umx_get_bracket_addresses(), umx_make(), umx_standardize(), umx_string_to_algebra(), umx, xmuHasSquareBrackets(), xmuLabel_MATRIX_Model(), xmuLabel_Matrix(), xmuLabel_RAM_Model(), xmuMI(), xmuMakeDeviationThresholdsMatrices(), xmuMakeOneHeadedPathsFromPathList(), xmuMakeTwoHeadedPathsFromPathList(), xmuMaxLevels(), xmuMinLevels(), xmuPropagateLabels(), xmuRAM2Ordinal(), xmuTwinSuper_Continuous(), xmuTwinUpgradeMeansToCovariateModel(), xmu_CI_merge(), xmu_CI_stash(), xmu_DF_to_mxData_TypeCov(), xmu_PadAndPruneForDefVars(), xmu_cell_is_on(), xmu_check_levels_identical(), xmu_check_needs_means(), xmu_check_variance(), xmu_clean_label(), xmu_data_missing(), xmu_data_swap_a_block(), xmu_describe_data_WLS(), xmu_dot_make_paths(), xmu_dot_make_residuals(), xmu_dot_maker(), xmu_dot_move_ranks(), xmu_dot_rank_str(), xmu_extract_column(), xmu_get_CI(), xmu_lavaan_process_group(), xmu_make_TwinSuperModel(), xmu_make_bin_cont_pair_data(), xmu_make_mxData(), xmu_match.arg(), xmu_name_from_lavaan_str(), xmu_path2twin(), xmu_path_regex(), xmu_safe_run_summary(), xmu_set_sep_from_suffix(), xmu_show_fit_or_comparison(), xmu_standardize_ACEcov(), xmu_standardize_ACEv(), xmu_standardize_ACE(), xmu_standardize_CP(), xmu_standardize_IP(), xmu_standardize_RAM(), xmu_standardize_SexLim(), xmu_standardize_Simplex(), xmu_start_value_list(), xmu_starts(), xmu_twin_add_WeightMatrices(), xmu_twin_check(), xmu_twin_get_var_names(), xmu_twin_upgrade_selDvs2SelVars()

Examples

```
x = umxMatrix('test', 'Full', nrow = 4, ncol = 4)
xmu_simplex_corner(x, start = .9)
# See how we have a diag free, but offset 1-down?
umx_print( xmu_simplex_corner(x, start = .9)$values, zero=".")
```

---

xmu_standardize_ACE *xmu_standardize_ACE*

Description

Standardize an ACE model *BUT* you probably want umx_standardize().

Usage

```
xmu_standardize_ACE(model, ...)
```

Arguments

model	an umxACE() model to standardize
...	Other options

Value

- Standardized ACE umxACE() model

References

- https://tbates.github.io, https://github.com/tbates/umx

See Also

Other xmu internal not for end user: umxModel(), umxRenameMatrix(), umxTwinMaker(), umx_APA_pval(), umx_fun_mean_sd(), umx_get_bracket_addresses(), umx_make(), umx_standardize(), umx_string_to_algebra(), umx, xmuHasSquareBrackets(), xmuLabel_MATRIX_Model(), xmuLabel_Matrix(), xmuLabel_RAM_Model(), xmuMI(), xmuMakeDeviationThresholdsMatrices(), xmuMakeOneHeadedPathsFromPathList(), xmuMakeTwoHeadedPathsFromPathList(), xmuMaxLevels(), xmuMinLevels(), xmuPropagateLabels(), xmuRAM2Ordinal(), xmuTwinSuper_Continuous(), xmuTwinUpgradeMeansToCovariateModel(), xmu_CI_merge(), xmu_CI_stash(), xmu_DF_to_mxData_TypeCov(), xmu_PadAndPruneForDefVars(), xmu_cell_is_on(), xmu_check_levels_identical(), xmu_check_needs_means(), xmu_check_variance(), xmu_clean_label(), xmu_data_missing(), xmu_data_swap_a_block(), xmu_describe_data_WLS(), xmu_dot_make_paths(), xmu_dot_make_residuals(), xmu_dot_maker(), xmu_dot_move_ranks(), xmu_dot_rank_str(), xmu_extract_column(), xmu_get_CI(), xmu_lavaan_process_group(), xmu_make_TwinSuperModel(), xmu_make_bin_cont_pair_data(), xmu_make_mxData(), xmu_match.arg(), xmu_name_from_lavaan_str(), xmu_path2twin(), xmu_path_regex(), xmu_safe_run_summary(), xmu_set_sep_from_suffix(), xmu_show_fit_or_comparison(), xmu_simplex_corner(), xmu_standardize_ACEcov(), xmu_standardize_ACEv(), xmu_standardize_CP(), xmu_standardize_IP(), xmu_standardize_RAM(), xmu_standardize_SexLim(), xmu_standardize_Simplex(), xmu_start_value_list(), xmu_starts(), xmu_twin_add_WeightMatrices(), xmu_twin_check(), xmu_twin_get_var_names(), xmu_twin_upgrade_selDvs2Sel

Examples

```
require(umx)
data(twinData)
selDVs = c("bmi1", "bmi2")
mzData <- twinData[twinData$zygosity %in% "MZFF", selDVs][1:80,] # 80 pairs for speed
dzData <- twinData[twinData$zygosity %in% "DZFF", selDVs][1:80,]
m1  = umxACE(selDVs = selDVs, dzData = dzData, mzData = mzData)
std = xmu_standardize_ACE(m1)
```

`xmu_standardize_ACEcov`

xmu_standardize_ACEcov

Description

Standardize an ACE model with covariates

Usage

```
xmu_standardize_ACEcov(model, ...)
```

Arguments

model	an umxACEcov() model to standardize
...	Other options

Value

- Standardized umxACEcov() model

References

- https://tbates.github.io, https://github.com/tbates/umx

See Also

Other xmu internal not for end user: umxModel(), umxRenameMatrix(), umxTwinMaker(), umx_APA_pval(), umx_fun_mean_sd(), umx_get_bracket_addresses(), umx_make(), umx_standardize(), umx_string_to_algebra(), umx, xmuHasSquareBrackets(), xmuLabel_MATRIX_Model(), xmuLabel_Matrix(), xmuLabel_RAM_Model(), xmuMI(), xmuMakeDeviationThresholdsMatrices(), xmuMakeOneHeadedPathsFromPathList(), xmuMakeTwoHeadedPathsFromPathList(), xmuMaxLevels(), xmuMinLevels(), xmuPropagateLabels(), xmuRAM2Ordinal(), xmuTwinSuper_Continuous(), xmuTwinUpgradeMeansToCovariateModel(), xmu_CI_merge(), xmu_CI_stash(), xmu_DF_to_mxData_TypeCov(), xmu_PadAndPruneForDefVars(), xmu_cell_is_on(), xmu_check_levels_identical(), xmu_check_needs_means(), xmu_check_variance(), xmu_clean_label(), xmu_data_missing(), xmu_data_swap_a_block(), xmu_describe_data_WLS(), xmu_dot_make_paths(), xmu_dot_make_residuals(), xmu_dot_maker(), xmu_dot_move_ranks(), xmu_dot_rank_str(), xmu_extract_column(), xmu_get_CI(), xmu_lavaan_process_group(),

xmu_make_TwinSuperModel(), xmu_make_bin_cont_pair_data(), xmu_make_mxData(), xmu_match.arg(), xmu_name_from_lavaan_str(), xmu_path2twin(), xmu_path_regex(), xmu_safe_run_summary(), xmu_set_sep_from_suffix(), xmu_show_fit_or_comparison(), xmu_simplex_corner(), xmu_standardize_ACEv(), xmu_standardize_ACE(), xmu_standardize_CP(), xmu_standardize_IP(), xmu_standardize_RAM(), xmu_standardize_SexLim(), xmu_standardize_Simplex(), xmu_start_value_list(), xmu_starts(), xmu_twin_add_WeightMatrices(), xmu_twin_check(), xmu_twin_get_var_names(), xmu_twin_upgrade_selDvs2SelV

Examples

```
require(umx)
data(twinData)
twinData$age1 = twinData$age2 = twinData$age
selDVs  = c("bmi")
selCovs = c("ht") # silly example
selVars = umx_paste_names(c(selDVs, selCovs), sep = "", suffixes= 1:2)
mzData = subset(twinData, zyg == 1, selVars)[1:80, ]
dzData = subset(twinData, zyg == 3, selVars)[1:80, ]
m1 = umxACEcov(selDVs = selDVs, selCovs = selCovs, dzData = dzData, mzData = mzData,
 sep = "", autoRun = TRUE)
fit = xmu_standardize_ACEcov(m1)
```

---

xmu_standardize_ACEv *Standardize an ACE variance components model (ACEv)*

Description

xmu_standardize_ACE allows umx_standardize to standardize an ACE variance components model.

Usage

```
xmu_standardize_ACEv(model, ...)
```

Arguments

model	An umxACEv() model to standardize.
...	Other parameters.

Value

- A standardized umxACEv() model.

References

- https://tbates.github.io, https://github.com/tbates/umx

See Also

Other xmu internal not for end user: umxModel(), umxRenameMatrix(), umxTwinMaker(), umx_APA_pval(), umx_fun_mean_sd(), umx_get_bracket_addresses(), umx_make(), umx_standardize(), umx_string_to_algebra(), umx, xmuHasSquareBrackets(), xmuLabel_MATRIX_Model(), xmuLabel_Matrix(), xmuLabel_RAM_Model(), xmuMI(), xmuMakeDeviationThresholdsMatrices(), xmuMakeOneHeadedPathsFromPathList(), xmuMakeTwoHeadedPathsFromPathList(), xmuMaxLevels(), xmuMinLevels(), xmuPropagateLabels(), xmuRAM2Ordinal(), xmuTwinSuper_Continuous(), xmuTwinUpgradeMeansToCovariateModel(), xmu_CI_merge(), xmu_CI_stash(), xmu_DF_to_mxData_TypeCov(), xmu_PadAndPruneForDefVars(), xmu_cell_is_on(), xmu_check_levels_identical(), xmu_check_needs_means(), xmu_check_variance(), xmu_clean_label(), xmu_data_missing(), xmu_data_swap_a_block(), xmu_describe_data_WLS(), xmu_dot_make_paths(), xmu_dot_make_residuals(), xmu_dot_maker(), xmu_dot_move_ranks(), xmu_dot_rank_str(), xmu_extract_column(), xmu_get_CI(), xmu_lavaan_process_group(), xmu_make_TwinSuperModel(), xmu_make_bin_cont_pair_data(), xmu_make_mxData(), xmu_match.arg(), xmu_name_from_lavaan_str(), xmu_path2twin(), xmu_path_regex(), xmu_safe_run_summary(), xmu_set_sep_from_suffix(), xmu_show_fit_or_comparison(), xmu_simplex_corner(), xmu_standardize_ACEcov(), xmu_standardize_ACE(), xmu_standardize_CP(), xmu_standardize_IP(), xmu_standardize_RAM(), xmu_standardize_SexLim(), xmu_standardize_Simplex(), xmu_start_value_list(), xmu_starts(), xmu_twin_add_WeightMatrices(), xmu_twin_check(), xmu_twin_get_var_names(), xmu_twin_upgrade_selDvs2SelV

Examples

```
require(umx)
data(twinData)
selDVs = c("bmi")
mzData <- twinData[twinData$zygosity %in% "MZFF",][1:80,] # 80 pairs for speed
dzData <- twinData[twinData$zygosity %in% "DZFF",][1:80,]
m1  = umxACEv(selDVs = selDVs, sep="", dzData = dzData, mzData = mzData)
std = umx_standardize(m1)
```

---

xmu_standardize_CP *Function to standardize a common pathway model*

Description

You probably want umx_standardize(). This function simply inserts the standardized CP components into the ai ci ei and as cs es matrices

Usage

```
xmu_standardize_CP(model, ...)
```

Arguments

model	an umxCP() model to standardize
...	Other options

Value

- standardized umxCP() model

References

- https://tbates.github.io, https://github.com/tbates/umx

See Also

Other xmu internal not for end user: umxModel(), umxRenameMatrix(), umxTwinMaker(), umx_APA_pval(), umx_fun_mean_sd(), umx_get_bracket_addresses(), umx_make(), umx_standardize(), umx_string_to_algebra(), umx, xmuHasSquareBrackets(), xmuLabel_MATRIX_Model(), xmuLabel_Matrix(), xmuLabel_RAM_Model(), xmuMI(), xmuMakeDeviationThresholdsMatrices(), xmuMakeOneHeadedPathsFromPathList(), xmuMakeTwoHeadedPathsFromPathList(), xmuMaxLevels(), xmuMinLevels(), xmuPropagateLabels(), xmuRAM2Ordinal(), xmuTwinSuper_Continuous(), xmuTwinUpgradeMeansToCovariateModel(), xmu_CI_merge(), xmu_CI_stash(), xmu_DF_to_mxData_TypeCov(), xmu_PadAndPruneForDefVars(), xmu_cell_is_on(), xmu_check_levels_identical(), xmu_check_needs_means(), xmu_check_variance(), xmu_clean_label(), xmu_data_missing(), xmu_data_swap_a_block(), xmu_describe_data_WLS(), xmu_dot_make_paths(), xmu_dot_make_residuals(), xmu_dot_maker(), xmu_dot_move_ranks(), xmu_dot_rank_str(), xmu_extract_column(), xmu_get_CI(), xmu_lavaan_process_group(), xmu_make_TwinSuperModel(), xmu_make_bin_cont_pair_data(), xmu_make_mxData(), xmu_match.arg(), xmu_name_from_lavaan_str(), xmu_path2twin(), xmu_path_regex(), xmu_safe_run_summary(), xmu_set_sep_from_suffix(), xmu_show_fit_or_comparison(), xmu_simplex_corner(), xmu_standardize_ACEcov(), xmu_standardize_ACEv(), xmu_standardize_ACE(), xmu_standardize_IP(), xmu_standardize_RAM(), xmu_standardize_SexLim(), xmu_standardize_Simplex(), xmu_start_value_list(), xmu_starts(), xmu_twin_add_WeightMatrices(), xmu_twin_check(), xmu_twin_get_var_names(), xmu_twin_upgrade_selDvs2SelV

Examples

```
## Not run:
selDVs = c("gff", "fc", "qol", "hap", "sat", "AD")
m1 = umxCP(selDVs = selDVs, nFac = 3, data=GFF, zyg="zyg_2grp")
m2 = xmu_standardize_CP(m1)

## End(Not run)
```

---

xmu_standardize_IP *non-user: Standardize an IP model*

Description

You probably want umx_standardize(). This function simply copies the standardized IP components into the ai ci ei and as cs es matrices

Usage

```
xmu_standardize_IP(model, ...)
```

Arguments

model	an umxIP() model to standardize
...	Other options

Value

- standardized IP umxIP() model

References

- https://tbates.github.io, https://github.com/tbates/umx

See Also

Other xmu internal not for end user: umxModel(), umxRenameMatrix(), umxTwinMaker(), umx_APA_pval(), umx_fun_mean_sd(), umx_get_bracket_addresses(), umx_make(), umx_standardize(), umx_string_to_algebra(), umx, xmuHasSquareBrackets(), xmuLabel_MATRIX_Model(), xmuLabel_Matrix(), xmuLabel_RAM_Model(), xmuMI(), xmuMakeDeviationThresholdsMatrices(), xmuMakeOneHeadedPathsFromPathList(), xmuMakeTwoHeadedPathsFromPathList(), xmuMaxLevels(), xmuMinLevels(), xmuPropagateLabels(), xmuRAM2Ordinal(), xmuTwinSuper_Continuous(), xmuTwinUpgradeMeansToCovariateModel(), xmu_CI_merge(), xmu_CI_stash(), xmu_DF_to_mxData_TypeCov(), xmu_PadAndPruneForDefVars(), xmu_cell_is_on(), xmu_check_levels_identical(), xmu_check_needs_means(), xmu_check_variance(), xmu_clean_label(), xmu_data_missing(), xmu_data_swap_a_block(), xmu_describe_data_WLS(), xmu_dot_make_paths(), xmu_dot_make_residuals(), xmu_dot_maker(), xmu_dot_move_ranks(), xmu_dot_rank_str(), xmu_extract_column(), xmu_get_CI(), xmu_lavaan_process_group(), xmu_make_TwinSuperModel(), xmu_make_bin_cont_pair_data(), xmu_make_mxData(), xmu_match.arg(), xmu_name_from_lavaan_str(), xmu_path2twin(), xmu_path_regex(), xmu_safe_run_summary(), xmu_set_sep_from_suffix(), xmu_show_fit_or_comparison(), xmu_simplex_corner(), xmu_standardize_ACEcov(), xmu_standardize_ACEv(), xmu_standardize_ACE(), xmu_standardize_CP(), xmu_standardize_RAM(), xmu_standardize_SexLim(), xmu_standardize_Simplex(), xmu_start_value_list(), xmu_starts(), xmu_twin_add_WeightMatrices(), xmu_twin_check(), xmu_twin_get_var_names(), xmu_twin_upgrade_selDvs2SelV

Examples

```
## Not run:
model = xmu_standardize_IP(model)

## End(Not run)
```

---

xmu_standardize_RAM *Standardize a Structural Model (not for end users)*

Description

You probably want umx_standardize(), not this.

Usage

```
xmu_standardize_RAM(model, ...)
```

Arguments

model	The mxModel() you wish to standardize
...	Other options

Details

xmu_standardize_RAM takes a RAM-style model, and returns standardized version.

References

- https://github.com/tbates/umx

See Also

Other xmu internal not for end user: umxModel(), umxRenameMatrix(), umxTwinMaker(), umx_APA_pval(), umx_fun_mean_sd(), umx_get_bracket_addresses(), umx_make(), umx_standardize(), umx_string_to_algebra(), umx, xmuHasSquareBrackets(), xmuLabel_MATRIX_Model(), xmuLabel_Matrix(), xmuLabel_RAM_Model(), xmuMI(), xmuMakeDeviationThresholdsMatrices(), xmuMakeOneHeadedPathsFromPathList(), xmuMakeTwoHeadedPathsFromPathList(), xmuMaxLevels(), xmuMinLevels(), xmuPropagateLabels(), xmuRAM2Ordinal(), xmuTwinSuper_Continuous(), xmuTwinUpgradeMeansToCovariateModel(), xmu_CI_merge(), xmu_CI_stash(), xmu_DF_to_mxData_TypeCov(), xmu_PadAndPruneForDefVars(), xmu_cell_is_on(), xmu_check_levels_identical(), xmu_check_needs_means(), xmu_check_variance(), xmu_clean_label(), xmu_data_missing(), xmu_data_swap_a_block(), xmu_describe_data_WLS(), xmu_dot_make_paths(), xmu_dot_make_residuals(), xmu_dot_maker(), xmu_dot_move_ranks(), xmu_dot_rank_str(), xmu_extract_column(), xmu_get_CI(), xmu_lavaan_process_group(), xmu_make_TwinSuperModel(), xmu_make_bin_cont_pair_data(), xmu_make_mxData(), xmu_match.arg(), xmu_name_from_lavaan_str(), xmu_path2twin(), xmu_path_regex(), xmu_safe_run_summary(), xmu_set_sep_from_suffix(), xmu_show_fit_or_comparison(), xmu_simplex_corner(), xmu_standardize_ACEcov(), xmu_standardize_ACEv(), xmu_standardize_ACE(), xmu_standardize_CP(), xmu_standardize_IP(), xmu_standardize_SexLim(), xmu_standardize_Simplex(), xmu_start_value_list(), xmu_starts(), xmu_twin_add_WeightMatrices(), xmu_twin_check(), xmu_twin_get_var_names(), xmu_twin_upgrade_selDvs2SelV

Examples

```
require(umx)
data(demoOneFactor)
manifests = names(demoOneFactor)

m1 = umxRAM("std_ex", data = demoOneFactor, type = "cov",
umxPath("G", to = manifests),
umxPath(var = manifests),
umxPath(var = "G", fixedAt = 1.0)
)

m1 = xmu_standardize_RAM(m1)
m1 = umx_standardize(m1)
```

```
umxSummary(m1)
```

`xmu_standardize_SexLim` *Standardize a SexLim model*

Description

`xmu_standardize_SexLim` would move standardized Sexlim values into raw cells, but can't as these are algebras.

Usage

```
xmu_standardize_SexLim(model, ...)
```

Arguments

`model`	an `umxSexLim()` model to standardize
`...`	Other options

Value

- standardized `umxSexLim()` model

See Also

Other xmu internal not for end user: `umxModel()`, `umxRenameMatrix()`, `umxTwinMaker()`, `umx_APA_pval()`, `umx_fun_mean_sd()`, `umx_get_bracket_addresses()`, `umx_make()`, `umx_standardize()`, `umx_string_to_algebra()`, `umx`, `xmuHasSquareBrackets()`, `xmuLabel_MATRIX_Model()`, `xmuLabel_Matrix()`, `xmuLabel_RAM_Model()`, `xmuMI()`, `xmuMakeDeviationThresholdsMatrices()`, `xmuMakeOneHeadedPathsFromPathList()`, `xmuMakeTwoHeadedPathsFromPathList()`, `xmuMaxLevels()`, `xmuMinLevels()`, `xmuPropagateLabels()`, `xmuRAM2Ordinal()`, `xmuTwinSuper_Continuous()`, `xmuTwinUpgradeMeansToCovariateModel()`, `xmu_CI_merge()`, `xmu_CI_stash()`, `xmu_DF_to_mxData_TypeCov()`, `xmu_PadAndPruneForDefVars()`, `xmu_cell_is_on()`, `xmu_check_levels_identical()`, `xmu_check_needs_means()`, `xmu_check_variance()`, `xmu_clean_label()`, `xmu_data_missing()`, `xmu_data_swap_a_block()`, `xmu_describe_data_WLS()`, `xmu_dot_make_paths()`, `xmu_dot_make_residuals()`, `xmu_dot_maker()`, `xmu_dot_move_ranks()`, `xmu_dot_rank_str()`, `xmu_extract_column()`, `xmu_get_CI()`, `xmu_lavaan_process_group()`, `xmu_make_TwinSuperModel()`, `xmu_make_bin_cont_pair_data()`, `xmu_make_mxData()`, `xmu_match.arg()`, `xmu_name_from_lavaan_str()`, `xmu_path2twin()`, `xmu_path_regex()`, `xmu_safe_run_summary()`, `xmu_set_sep_from_suffix()`, `xmu_show_fit_or_comparison()`, `xmu_simplex_corner()`, `xmu_standardize_ACEcov()`, `xmu_standardize_ACEv()`, `xmu_standardize_ACE()`, `xmu_standardize_CP()`, `xmu_standardize_IP()`, `xmu_standardize_RAM()`, `xmu_standardize_Simplex()`, `xmu_start_value_list()`, `xmu_starts()`, `xmu_twin_add_WeightMatrices()`, `xmu_twin_check()`, `xmu_twin_get_var_names()`, `xmu_twin_upgrade_selDvs2Sel`

Examples

```
## Not run:
model = xmu_standardize_SexLim(model)

## End(Not run)
```

`xmu_standardize_Simplex`

Standardize a Simplex twin model

Description

xmu_standardize_Simplex

Usage

```
xmu_standardize_Simplex(model, ...)
```

Arguments

model	an umxSimplex() model to standardize
...	Other options

Value

- Standardized Simplex umxSimplex() model

References

- https://tbates.github.io, https://github.com/tbates/umx

See Also

Other xmu internal not for end user: umxModel(), umxRenameMatrix(), umxTwinMaker(), umx_APA_pval(), umx_fun_mean_sd(), umx_get_bracket_addresses(), umx_make(), umx_standardize(), umx_string_to_algebra(), umx, xmuHasSquareBrackets(), xmuLabel_MATRIX_Model(), xmuLabel_Matrix(), xmuLabel_RAM_Model(), xmuMI(), xmuMakeDeviationThresholdsMatrices(), xmuMakeOneHeadedPathsFromPathList(), xmuMakeTwoHeadedPathsFromPathList(), xmuMaxLevels(), xmuMinLevels(), xmuPropagateLabels(), xmuRAM2Ordinal(), xmuTwinSuper_Continuous(), xmuTwinUpgradeMeansToCovariateModel(), xmu_CI_merge(), xmu_CI_stash(), xmu_DF_to_mxData_TypeCov(), xmu_PadAndPruneForDefVars(), xmu_cell_is_on(), xmu_check_levels_identical(), xmu_check_needs_means(), xmu_check_variance(), xmu_clean_label(), xmu_data_missing(), xmu_data_swap_a_block(), xmu_describe_data_WLS(), xmu_dot_make_paths(), xmu_dot_make_residuals(), xmu_dot_maker(), xmu_dot_move_ranks(), xmu_dot_rank_str(), xmu_extract_column(), xmu_get_CI(), xmu_lavaan_process_group(), xmu_make_TwinSuperModel(), xmu_make_bin_cont_pair_data(), xmu_make_mxData(), xmu_match.arg(), xmu_name_from_lavaan_str(), xmu_path2twin(), xmu_path_regex(), xmu_safe_run_summary(), xmu_set_sep_from_suffix(), xmu_show_fit_or_comparison(), xmu_simplex_corner(), xmu_standardize_ACEcov()

xmu_standardize_ACEv(), xmu_standardize_ACE(), xmu_standardize_CP(), xmu_standardize_IP(), xmu_standardize_RAM(), xmu_standardize_SexLim(), xmu_start_value_list(), xmu_starts(), xmu_twin_add_WeightMatrices(), xmu_twin_check(), xmu_twin_get_var_names(), xmu_twin_upgrade_selDvs2Sel

Examples

```
## Not run:
data(iqdat)
mzData = subset(iqdat, zygosity == "MZ")
dzData = subset(iqdat, zygosity == "DZ")
m1  = umxSimplex(selDVs = paste0("IQ_age", 1:4), sep = "_T",
dzData = dzData, mzData = mzData, tryHard = "yes")
std = xmu_standardize_Simplex(m1)

## End(Not run)
```

---

xmu_starts *Helper providing boilerplate start values for means and variance in twin models*

Description

xmu_starts can handle several common/boilerplate situations in which means and variance start values are used in twin models.

Usage

```
xmu_starts(
  mzData,
  dzData,
  selVars = selVars,
  sep = NULL,
  equateMeans = NULL,
  nSib = 2,
  varForm = c("Cholesky"),
  SD = TRUE,
  divideBy = 3
)
```

Arguments

mzData	Data for MZ pairs.
dzData	Data for DZ pairs.
selVars	Variable names: If sep = NULL, then treated as full names for both sibs.
sep	All the variables full names.
equateMeans	(NULL)

nSib	How many subjects in a family.
varForm	currently just "Cholesky" style.
SD	= TRUE (FALSE = variance, not SD).
divideBy	= 3 (A,C,E) 1/3rd each. Use 1 to do this yourself post-hoc.

Value

- varStarts and meanStarts

See Also

Other xmu internal not for end user: umxModel(), umxRenameMatrix(), umxTwinMaker(), umx_APA_pval(), umx_fun_mean_sd(), umx_get_bracket_addresses(), umx_make(), umx_standardize(), umx_string_to_algebra(), umx, xmuHasSquareBrackets(), xmuLabel_MATRIX_Model(), xmuLabel_Matrix(), xmuLabel_RAM_Model(), xmuMI(), xmuMakeDeviationThresholdsMatrices(), xmuMakeOneHeadedPathsFromPathList(), xmuMakeTwoHeadedPathsFromPathList(), xmuMaxLevels(), xmuMinLevels(), xmuPropagateLabels(), xmuRAM2Ordinal(), xmuTwinSuper_Continuous(), xmuTwinUpgradeMeansToCovariateModel(), xmu_CI_merge(), xmu_CI_stash(), xmu_DF_to_mxData_TypeCov(), xmu_PadAndPruneForDefVars(), xmu_cell_is_on(), xmu_check_levels_identical(), xmu_check_needs_means(), xmu_check_variance(), xmu_clean_label(), xmu_data_missing(), xmu_data_swap_a_block(), xmu_describe_data_WLS(), xmu_dot_make_paths(), xmu_dot_make_residuals(), xmu_dot_maker(), xmu_dot_move_ranks(), xmu_dot_rank_str(), xmu_extract_column(), xmu_get_CI(), xmu_lavaan_process_group(), xmu_make_TwinSuperModel(), xmu_make_bin_cont_pair_data(), xmu_make_mxData(), xmu_match.arg(), xmu_name_from_lavaan_str(), xmu_path2twin(), xmu_path_regex(), xmu_safe_run_summary(), xmu_set_sep_from_suffix(), xmu_show_fit_or_comparison(), xmu_simplex_corner(), xmu_standardize_ACEcov(), xmu_standardize_ACEv(), xmu_standardize_ACE(), xmu_standardize_CP(), xmu_standardize_IP(), xmu_standardize_RAM(), xmu_standardize_SexLim(), xmu_standardize_Simplex(), xmu_start_value_list(), xmu_twin_add_WeightMatrices(), xmu_twin_check(), xmu_twin_get_var_names(), xmu_twin_upgrade_selDvs2SelV

Examples

```
data(twinData)
selDVs = c("wt", "ht")
mzData = twinData[twinData$zygosity %in%  "MZFF", ]
dzData = twinData[twinData$zygosity %in%  "DZFF", ]

round(sqrt(var(dzData[,tvars(selDVs, "")], na.rm=TRUE)/3),3)

tmp = xmu_starts(mzData, dzData, selVars = selDVs, sep= "",
equateMeans = TRUE, varForm = "Cholesky")
tmp

round(var(dzData[,tvars(selDVs, "")], na.rm=TRUE)/3,3)
tmp = xmu_starts(mzData, dzData, selVars = selDVs, sep= "",
equateMeans = TRUE, varForm = "Cholesky", SD= FALSE)

tmp

# one variable
tmp = xmu_starts(mzData, dzData, selVars = "wt", sep= "",
```

```
equateMeans = TRUE, varForm = "Cholesky", SD= FALSE)

# Ordinal/continuous mix
data(twinData)
twinData= umx_scale_wide_twin_data(data=twinData,varsToScale="wt",sep= "")
# Cut BMI column to form ordinal obesity variables
obLevels = c('normal', 'overweight', 'obese')
cuts     = quantile(twinData[, "bmi1"], probs = c(.5, .8), na.rm = TRUE)
twinData$obese1= cut(twinData$bmi1,breaks=c(-Inf,cuts,Inf),labels=obLevels)
twinData$obese2= cut(twinData$bmi2,breaks=c(-Inf,cuts,Inf),labels=obLevels)
# Make the ordinal variables into mxFactors
ordDVs = c("obese1", "obese2")
twinData[, ordDVs] = umxFactor(twinData[, ordDVs])
mzData = twinData[twinData$zygosity %in% "MZFF",]
dzData = twinData[twinData$zygosity %in% "DZFF",]
tmp = xmu_starts(mzData, dzData, selVars = c("wt","obese"), sep= "",
 nSib= 2, equateMeans = TRUE, varForm = "Cholesky", SD= FALSE)

tmp = xmu_starts(mxData(mzData, type="raw"), mxData(mzData, type="raw"),
   selVars = c("wt","obese"), sep= "", nSib= 2, equateMeans = TRUE,
  varForm = "Cholesky", SD= FALSE)
```

`xmu_start_value_list` *Make start values*

Description

Purpose: Create startvalues for OpenMx paths use cases umx:::xmuStart_value_list(1) umxValues(1) # 1 value, varying around 1, with sd of .1 umxValues(1, n=letters) # length(letters) start values, with mean 1 and sd .1 umxValues(100, 15) # 1 start, with mean 100 and sd 15

Usage

```
xmu_start_value_list(mean = 1, sd = NA, n = 1)
```

Arguments

`mean`	the mean start value
`sd`	the sd of values
`n`	how many to generate

Value

- start value list

See Also

Other xmu internal not for end user: umxModel(), umxRenameMatrix(), umxTwinMaker(), umx_APA_pval(), umx_fun_mean_sd(), umx_get_bracket_addresses(), umx_make(), umx_standardize(), umx_string_to_algebra(), umx, xmuHasSquareBrackets(), xmuLabel_MATRIX_Model(), xmuLabel_Matrix(), xmuLabel_RAM_Model(), xmuMI(), xmuMakeDeviationThresholdsMatrices(), xmuMakeOneHeadedPathsFromPathList(), xmuMakeTwoHeadedPathsFromPathList(), xmuMaxLevels(), xmuMinLevels(), xmuPropagateLabels(), xmuRAM2Ordinal(), xmuTwinSuper_Continuous(), xmuTwinUpgradeMeansToCovariateModel(), xmu_CI_merge(), xmu_CI_stash(), xmu_DF_to_mxData_TypeCov(), xmu_PadAndPruneForDefVars(), xmu_cell_is_on(), xmu_check_levels_identical(), xmu_check_needs_means(), xmu_check_variance(), xmu_clean_label(), xmu_data_missing(), xmu_data_swap_a_block(), xmu_describe_data_WLS(), xmu_dot_make_paths(), xmu_dot_make_residuals(), xmu_dot_maker(), xmu_dot_move_ranks(), xmu_dot_rank_str(), xmu_extract_column(), xmu_get_CI(), xmu_lavaan_process_group(), xmu_make_TwinSuperModel(), xmu_make_bin_cont_pair_data(), xmu_make_mxData(), xmu_match.arg(), xmu_name_from_lavaan_str(), xmu_path2twin(), xmu_path_regex(), xmu_safe_run_summary(), xmu_set_sep_from_suffix(), xmu_show_fit_or_comparison(), xmu_simplex_corner(), xmu_standardize_ACEcov(), xmu_standardize_ACEv(), xmu_standardize_ACE(), xmu_standardize_CP(), xmu_standardize_IP(), xmu_standardize_RAM(), xmu_standardize_SexLim(), xmu_standardize_Simplex(), xmu_starts(), xmu_twin_add_WeightMatrices(), xmu_twin_check(), xmu_twin_get_var_names(), xmu_twin_upgrade_selDvs2SelV

`xmu_twin_add_WeightMatrices`

Add weight matrices to twin models.

Description

Add weight models (MZw, DZw) with matrices (e.g. mzWeightMatrix) to a twin model, and update `mxFitFunctionMultigroup`. This yields a weighted model with vector objective.

To weight objective functions in OpenMx, you specify a container model that applies the weights m1 is the model with no weights, but with "vector = TRUE" option added to the FIML objective. This option makes FIML return individual likelihoods for each row of the data (rather than a single -2LL value for the model) You then optimize weighted versions of these likelihoods by building additional models containing weight data and an algebra that multiplies the likelihoods from the first model by the weight vector.

Usage

```
xmu_twin_add_WeightMatrices(model, mzWeights = NULL, dzWeights = NULL)
```

Arguments

`model`	umx-style twin model
`mzWeights`	data for MZ weights matrix
`dzWeights`	data for DZ weights matrix

Value

- model

See Also

Other xmu internal not for end user: umxModel(), umxRenameMatrix(), umxTwinMaker(), umx_APA_pval(), umx_fun_mean_sd(), umx_get_bracket_addresses(), umx_make(), umx_standardize(), umx_string_to_algebra(), umx, xmuHasSquareBrackets(), xmuLabel_MATRIX_Model(), xmuLabel_Matrix(), xmuLabel_RAM_Model(), xmuMI(), xmuMakeDeviationThresholdsMatrices(), xmuMakeOneHeadedPathsFromPathList(), xmuMakeTwoHeadedPathsFromPathList(), xmuMaxLevels(), xmuMinLevels(), xmuPropagateLabels(), xmuRAM2Ordinal(), xmuTwinSuper_Continuous(), xmuTwinUpgradeMeansToCovariateModel(), xmu_CI_merge(), xmu_CI_stash(), xmu_DF_to_mxData_TypeCov(), xmu_PadAndPruneForDefVars(), xmu_cell_is_on(), xmu_check_levels_identical(), xmu_check_needs_means(), xmu_check_variance(), xmu_clean_label(), xmu_data_missing(), xmu_data_swap_a_block(), xmu_describe_data_WLS(), xmu_dot_make_paths(), xmu_dot_make_residuals(), xmu_dot_maker(), xmu_dot_move_ranks(), xmu_dot_rank_str(), xmu_extract_column(), xmu_get_CI(), xmu_lavaan_process_group(), xmu_make_TwinSuperModel(), xmu_make_bin_cont_pair_data(), xmu_make_mxData(), xmu_match.arg(), xmu_name_from_lavaan_str(), xmu_path2twin(), xmu_path_regex(), xmu_safe_run_summary(), xmu_set_sep_from_suffix(), xmu_show_fit_or_comparison(), xmu_simplex_corner(), xmu_standardize_ACEcov(), xmu_standardize_ACEv(), xmu_standardize_ACE(), xmu_standardize_CP(), xmu_standardize_IP(), xmu_standardize_RAM(), xmu_standardize_SexLim(), xmu_standardize_Simplex(), xmu_start_value_list(), xmu_starts(), xmu_twin_check(), xmu_twin_get_var_names(), xmu_twin_upgrade_selDvs2SelVars()

Examples

```
tmp = umx_make_twin_data_nice(data=twinData, sep="", zygosity="zygosity", numbering= 1:2)
m1  = umxACE(selDVs = "wt", data = tmp, dzData = "DZFF", mzData = "MZFF", autoRun= FALSE)
m1$MZ$fitfunction$vector= TRUE

tmp = xmu_twin_add_WeightMatrices(m1,
mzWeights= rnorm(nrow(m1$MZ$data$observed)),
dzWeights= rnorm(nrow(m1$DZ$data$observed))
 )
```

---

xmu_twin_check *Check basic aspects of input for twin models.*

Description

Check that DVs are in the data, that the data have rows, set the optimizer if requested.

Usage

```
xmu_twin_check(
  selDVs,
  dzData = dzData,
  mzData = mzData,
  sep = NULL,
  enforceSep = TRUE,
  nSib = 2,
```

```
  numObsMZ = NULL,
  numObsDZ = NULL,
  optimizer = NULL
)
```

Arguments

selDVs	Variables used in the data.
dzData	The DZ twin data.
mzData	The MZ twin data.
sep	Separator between base-name and numeric suffix when creating variable names, e.g. "_T"
enforceSep	Whether to require sep to be set, or just warn if it is not (Default = TRUE: enforce).
nSib	How many people per family? (Default = 2).
numObsMZ	set if data are not raw.
numObsDZ	set if data are not raw.
optimizer	Set by name (if you want to change it).

Value

None

References

- https://github.com/tbates/umx, https://tbates.github.io

See Also

Other xmu internal not for end user: umxModel(), umxRenameMatrix(), umxTwinMaker(), umx_APA_pval(), umx_fun_mean_sd(), umx_get_bracket_addresses(), umx_make(), umx_standardize(), umx_string_to_algebra(), umx, xmuHasSquareBrackets(), xmuLabel_MATRIX_Model(), xmuLabel_Matrix(), xmuLabel_RAM_Model(), xmuMI(), xmuMakeDeviationThresholdsMatrices(), xmuMakeOneHeadedPathsFromPathList(), xmuMakeTwoHeadedPathsFromPathList(), xmuMaxLevels(), xmuMinLevels(), xmuPropagateLabels(), xmuRAM2Ordinal(), xmuTwinSuper_Continuous(), xmuTwinUpgradeMeansToCovariateModel(), xmu_CI_merge(), xmu_CI_stash(), xmu_DF_to_mxData_TypeCov(), xmu_PadAndPruneForDefVars(), xmu_cell_is_on(), xmu_check_levels_identical(), xmu_check_needs_means(), xmu_check_variance(), xmu_clean_label(), xmu_data_missing(), xmu_data_swap_a_block(), xmu_describe_data_WLS(), xmu_dot_make_paths(), xmu_dot_make_residuals(), xmu_dot_maker(), xmu_dot_move_ranks(), xmu_dot_rank_str(), xmu_extract_column(), xmu_get_CI(), xmu_lavaan_process_group(), xmu_make_TwinSuperModel(), xmu_make_bin_cont_pair_data(), xmu_make_mxData(), xmu_match.arg(), xmu_name_from_lavaan_str(), xmu_path2twin(), xmu_path_regex(), xmu_safe_run_summary(), xmu_set_sep_from_suffix(), xmu_show_fit_or_comparison(), xmu_simplex_corner(), xmu_standardize_ACEcov(), xmu_standardize_ACEv(), xmu_standardize_ACE(), xmu_standardize_CP(), xmu_standardize_IP(), xmu_standardize_RAM(), xmu_standardize_SexLim(), xmu_standardize_Simplex(), xmu_start_value_list(), xmu_starts(), xmu_twin_add_WeightMatrices(), xmu_twin_get_var_names(), xmu_twin_upgrade_selDvs2SelVars()

Examples

```
library(umx)
data(twinData)
mzData = subset(twinData, zygosity == "MZFF")
dzData = subset(twinData, zygosity == "MZFF")
xmu_twin_check(selDVs = c("wt", "ht"), dzData = dzData, mzData = mzData,
sep = "", enforceSep = TRUE)
xmu_twin_check(selDVs = c("wt", "ht"), dzData = dzData, mzData = mzData,
sep = "", enforceSep = FALSE)
xmu_twin_check(selDVs = c("wt", "ht"), dzData = dzData, mzData = mzData,
sep = "", enforceSep = TRUE, nSib = 2, optimizer = NULL)

## Not run:
# TODO xmu_twin_check: move to a test file:
# 1. stop on no rows
xmu_twin_check("Generativity", twinData[NULL,], twinData[NULL,], sep="_T")
# Error in xmu_twin_check("Generativity", twinData[NULL, ], twinData[NULL,  :
#   Your DZ dataset has no rows!

# 2. Stop on a NULL sep  = NULL IFF enforceSep = TRUE
xmu_twin_check(selDVs = c("wt", "ht"), dzData = dzData, mzData = mzData, enforceSep = TRUE)
# 3. stop on a factor with sep = NULL

## End(Not run)
```

`xmu_twin_get_var_names`

Not for user: pull variable names from a twin model

Description

Barely useful, but justified perhaps by centralizing trimming the "_T1" off, and returning just twin 1.

Usage

```
xmu_twin_get_var_names(
  model,
  source = c("expCovMZ", "observed"),
  trim = TRUE,
  twinOneOnly = TRUE
)
```

Arguments

`model`	A model to get the variables from
`source`	Whether to access the dimnames of the "expCovMZ" or the names of the "observed" data (will include covariates)

trim	Whether to trim the suffix (TRUE)
twinOneOnly	Whether to return on the names for twin 1 (i.e., unique names)

Value

- variable names from twin model

See Also

Other xmu internal not for end user: umxModel(), umxRenameMatrix(), umxTwinMaker(), umx_APA_pval(), umx_fun_mean_sd(), umx_get_bracket_addresses(), umx_make(), umx_standardize(), umx_string_to_algebra(), umx, xmuHasSquareBrackets(), xmuLabel_MATRIX_Model(), xmuLabel_Matrix(), xmuLabel_RAM_Model(), xmuMI(), xmuMakeDeviationThresholdsMatrices(), xmuMakeOneHeadedPathsFromPathList(), xmuMakeTwoHeadedPathsFromPathList(), xmuMaxLevels(), xmuMinLevels(), xmuPropagateLabels(), xmuRAM2Ordinal(), xmuTwinSuper_Continuous(), xmuTwinUpgradeMeansToCovariateModel(), xmu_CI_merge(), xmu_CI_stash(), xmu_DF_to_mxData_TypeCov(), xmu_PadAndPruneForDefVars(), xmu_cell_is_on(), xmu_check_levels_identical(), xmu_check_needs_means(), xmu_check_variance(), xmu_clean_label(), xmu_data_missing(), xmu_data_swap_a_block(), xmu_describe_data_WLS(), xmu_dot_make_paths(), xmu_dot_make_residuals(), xmu_dot_maker(), xmu_dot_move_ranks(), xmu_dot_rank_str(), xmu_extract_column(), xmu_get_CI(), xmu_lavaan_process_group(), xmu_make_TwinSuperModel(), xmu_make_bin_cont_pair_data(), xmu_make_mxData(), xmu_match.arg(), xmu_name_from_lavaan_str(), xmu_path2twin(), xmu_path_regex(), xmu_safe_run_summary(), xmu_set_sep_from_suffix(), xmu_show_fit_or_comparison(), xmu_simplex_corner(), xmu_standardize_ACEcov(), xmu_standardize_ACEv(), xmu_standardize_ACE(), xmu_standardize_CP(), xmu_standardize_IP(), xmu_standardize_RAM(), xmu_standardize_SexLim(), xmu_standardize_Simplex(), xmu_start_value_list(), xmu_starts(), xmu_twin_add_WeightMatrices(), xmu_twin_check(), xmu_twin_upgrade_selDvs2SelVars()

Examples

```
data(twinData) # ?twinData from Australian twins.
twinData[, c("ht1", "ht2")] = twinData[, c("ht1", "ht2")] * 10
mzData = twinData[twinData$zygosity %in% "MZFF", ]
dzData = twinData[twinData$zygosity %in% "DZFF", ]
m1 = umxACE(selDVs= "ht", sep= "", dzData= dzData, mzData= mzData, autoRun= FALSE)
selVars = xmu_twin_get_var_names(m1, source = "expCovMZ", trim = TRUE, twinOneOnly = TRUE) # "ht"
umx_check(selVars == "ht")
xmu_twin_get_var_names(m1, source= "expCovMZ", trim= FALSE, twinOneOnly= FALSE) #"ht1" "ht2"
selVars = xmu_twin_get_var_names(m1, source= "observed", trim= TRUE, twinOneOnly= TRUE)# "ht"
nVar = length(selVars)
umx_check(nVar==1)
```

---

xmu_twin_upgrade_selDvs2SelVars

Upgrade selDvs to SelVars

Description

Just a helper to go from "wt" to "wt_T1" contingent on sep not being null

Usage

```
xmu_twin_upgrade_selDvs2SelVars(selDVs, sep, nSib)
```

Arguments

selDVs	with wt or wt_T1
sep	either "" etc., or NULL
nSib	wideness of data

Value

list of wt_T1 wt_T2 etc.

See Also

- umx

Other xmu internal not for end user: umxModel(), umxRenameMatrix(), umxTwinMaker(), umx_APA_pval(), umx_fun_mean_sd(), umx_get_bracket_addresses(), umx_make(), umx_standardize(), umx_string_to_algebra(), umx, xmuHasSquareBrackets(), xmuLabel_MATRIX_Model(), xmuLabel_Matrix(), xmuLabel_RAM_Model(), xmuMI(), xmuMakeDeviationThresholdsMatrices(), xmuMakeOneHeadedPathsFromPathList(), xmuMakeTwoHeadedPathsFromPathList(), xmuMaxLevels(), xmuMinLevels(), xmuPropagateLabels(), xmuRAM2Ordinal(), xmuTwinSuper_Continuous(), xmuTwinUpgradeMeansToCovariateModel(), xmu_CI_merge(), xmu_CI_stash(), xmu_DF_to_mxData_TypeCov(), xmu_PadAndPruneForDefVars(), xmu_cell_is_on(), xmu_check_levels_identical(), xmu_check_needs_means(), xmu_check_variance(), xmu_clean_label(), xmu_data_missing(), xmu_data_swap_a_block(), xmu_describe_data_WLS(), xmu_dot_make_paths(), xmu_dot_make_residuals(), xmu_dot_maker(), xmu_dot_move_ranks(), xmu_dot_rank_str(), xmu_extract_column(), xmu_get_CI(), xmu_lavaan_process_group(), xmu_make_TwinSuperModel(), xmu_make_bin_cont_pair_data(), xmu_make_mxData(), xmu_match.arg(), xmu_name_from_lavaan_str(), xmu_path2twin(), xmu_path_regex(), xmu_safe_run_summary(), xmu_set_sep_from_suffix(), xmu_show_fit_or_comparison(), xmu_simplex_corner(), xmu_standardize_ACEcov(), xmu_standardize_ACEv(), xmu_standardize_ACE(), xmu_standardize_CP(), xmu_standardize_IP(), xmu_standardize_RAM(), xmu_standardize_SexLim(), xmu_standardize_Simplex(), xmu_start_value_list(), xmu_starts(), xmu_twin_add_WeightMatrices(), xmu_twin_check(), xmu_twin_get_var_names()

Examples

```
xmu_twin_upgrade_selDvs2SelVars("wt", NULL, 2)
```

Index

∗Topic **datasets**
docData, 9
Fischbein_wt, 11
GFF, 13
iqdat, 17
us_skinfold_data, 325

aggregate(), *221, 222*
AIC(), *11*, *220*

base::trimws(), *321, 322*

colMeans(), *224*
complete.cases(), *353*
confint(), *76*
cor.test(), *66*
cov(), *322*
cov2cor(), *77*, *266*
cumsum(), *224*

data.frame, *67*
data.frame(), *116*, *243*, *255*, *273*, *373*
deg2rad, 7, *34*
deg2rad(), *34*
DiagrammeR(), *22*
DiagrammeR::DiagrammeR(), *21*
dl_from_dropbox, 8, *45*, *237*, *268*, *275*, *280*, *292*, *325*
docData, 9, *12*, *15*, *17*, *45*, *326*

extractAIC.MxModel, 10, *18*, *36–39*, *71*, *73, 74*, *76*, *99, 100*, *105*, *147*, *174*, *190*

factanal(), *18, 19*, *89*, *91*
factor(), *101*
file.rename(), *275*
Fischbein_wt, *10*, 11, *15*, *17*, *45*, *326*
FishersMethod, 12, *20*, *35*, *39*, *45*, *77*, *116*, *220*, *224*, *235*, *274*, *296–298*, *323*
formula(), *294*

GFF, *10*, *12*, 13, *17*, *45*, *326*
glm(), *67*
grep, *277*
grep(), *243*, *278*

install.OpenMx, 15, *33*, *45*, *69*, *126*, *168*, *220*, *225*, *238*, *276*, *280, 281*, *288*, *301*
install.OpenMx(), *220*
iqdat, *10*, *12*, *15*, 17, *45*, *326*

lm(), *67*, *164*
lme(), *67*
loadings, *11*, *18*, 18, *36–39*, *71*, *73, 74*, *76*, *99, 100*, *105*, *147*, *174*, *190*
loadings(), *19*
loadings.MxModel, *18*, 19, *44*, *67*, *103*, *107*, *138*, *169*, *222*, *278*, *320*
logLik(), *11*

MASS::mvrnorm(), *266*
match.arg(), *379, 380*
matrix(), *288*
merge(), *258*
mxAlgebra(), *65*, *317*
mxCheckIdentification(), *41, 42*, *190*, *217*
mxCI(), *70*, *75*, *176*
mxCompare(), *74*, *217*
mxData(), *163*, *176*, *255*, *346*, *356, 357*, *368*, *373*
mxEvalByName(), *47*
mxFactor(), *101, 102*, *232, 233*
mxFactorScores(), *91*, *103*
mxFitFunctionMultigroup(), *209, 210*
mxFitFunctionWLS(), *355*
mxMatrix(), *121, 122*, *129*, *218*, *253*, *328, 329*
mxMI(), *133*
mxModel, *104*
mxModel(), *10*, *21*, *23*, *35*, *37, 38*, *43*, *51*, *57*, *61*, *70–76*, *81*, *83*, *85*, *90*, *92*, *99,*

100, *105–107*, *109*, *112*, *120*, *122*, *131*, *132*, *134–136*, *138*, *144*, *145*, *148*, *155*, *165*, *169*, *170*, *172*, *176–178*, *181*, *186*, *188*, *189*, *191–193*, *195*, *197*, *199*, *201*, *203*, *204*, *206*, *208*, *209*, *218*, *219*, *244–246*, *250*, *251*, *253*, *256*, *296*, *316*, *320*, *323*, *329–331*, *336*, *338*, *339*, *344*, *350*, *359*, *369*, *377*, *386*, *389*, *397*
mxPath(), *122*, *142*
mxPower(), *268*
mxRun(), *75*, *135*, *176*
mxTryHard(), *47*, *387*

names, *277*
namez, *291*
namez *(umx_names)*, 277
namez(), *138*, *243*, *283*

oddsratio, *13*, 20, *35*, *39*, *45*, *77*, *116*, *220*, *224*, *235*, *274*, *296–298*, *323*
omxAssignFirstParameters(), *178*
omxAugmentDataWithWLSSummary(), *355*
omxBrownie(), *69*
omxGetParameters(), *106*, *107*
omxSetParameters(), *122*
OpenMx::mxAlgebra(), *125*
OpenMx::mxCI(), *70*, *76*
OpenMx::mxConstraint(), *125*
OpenMx::mxExpectationNormal(), *377*
OpenMx::mxFitFunctionML(), *377*
OpenMx::mxFitFunctionWLS(), *377*
OpenMx::mxPower(), *29*
OpenMx::mxRun(), *313*
OpenMx::mxSE(), *76*

packageVersion(), *220*
parameters *(umxParameters)*, 137
parameters(), *107*, *165*
paste0, *277*
plot *(plot.MxModel)*, 22
plot(), *22*, *24*, *26*, *47*, *81*, *113*, *120*, *144*, *146*, *148*, *152*, *153*, *155*, *158*, *165*, *186*, *197*, *199*, *201*, *203*, *204*, *363*
plot.MxLISRELModel, 21, *24*, *44*, *144*, *146–148*, *152*, *153*, *155*, *156*, *158*
plot.MxModel, *22*, 22, *44*, *144*, *146–148*, *152*, *153*, *155*, *156*, *158*
plot.MxModel(), *22*, *24*, *26*, *162*
plot.MxModelACE *(umxPlotACE)*, 143
plot.MxModelACE(), *192*
plot.MxModelACEcov *(umxPlotACEcov)*, 145
plot.MxModelACEv *(umxPlotACEv)*, 146
plot.MxModelCP *(umxPlotCP)*, 147
plot.MxModelDoC *(umxPlotDoC)*, 149
plot.MxModelDoC(), *10*, *86*, *199*
plot.MxModelGxE *(umxPlotGxE)*, 151
plot.MxModelGxEbiv *(umxPlotGxEbiv)*, 152
plot.MxModelIP *(umxPlotIP)*, 154
plot.MxModelSexLim *(umxPlotSexLim)*, 155
plot.MxModelSimplex *(umxPlotSimplex)*, 157
plot.MxModelTwinMaker, 25, *29*, *44*, *51*, *57*, *62*, *81*, *86*, *88*, *110*, *113*, *115*, *120*, *175*, *181*, *186*
power.ACE.test, *26*, 27, *44*, *51*, *57*, *62*, *81*, *86*, *88*, *110*, *113*, *115*, *120*, *175*, *181*, *186*
print(), *31*, *32*
print.oddsratio, 31
print.reliability, 32

qm, *16*, 33, *45*, *69*, *126*, *168*, *220*, *225*, *238*, *276*, *280*, *281*, *288*, *301*

rad2deg, *8*, 34
rad2deg(), *8*
regex(), *138*, *275*
regular expressions, *290*
reliability, *13*, *20*, 34, *39*, *45*, *77*, *116*, *220*, *224*, *235*, *274*, *296–298*, *323*
replacement, *277*
residuals(), *35*
residuals.MxModel, *11*, *18*, 35, *37–39*, *71*, *73*, *74*, *76*, *99*, *100*, *105*, *147*, *174*, *190*
RMSEA, *11*, *18*, *36*, 36, *38*, *39*, *71*, *73*, *74*, *76*, *99*, *100*, *105*, *147*, *174*, *190*
RMSEA.MxModel, *11*, *18*, *36*, *37*, 37, *39*, *71*, *73*, *74*, *76*, *99*, *100*, *105*, *147*, *174*, *190*
RMSEA.MxModel(), *36*
RMSEA.summary.mxmodel, *11*, *18*, *36–38*, 38, *71*, *73*, *74*, *76*, *99*, *100*, *105*, *147*, *174*, *190*
round(), *223*
rowSums(), *224*

SE_from_p, *13*, *20*, *35*, 39, *45*, *77*, *116*, *220*, *224*, *235*, *274*, *296–298*, *323*
SE_from_p(), *67*
sem::tsls(), *130, 131*
shQuote(), *279*
sin(), *7, 8*, *34*
stack(), *315*
stats::confint(), *71*, *76*
stats::glm(), *66*
stats::lm(), *66*
subset(), *354*
summary(), *190*
summary.MxModel, *104*
summaryAPA *(*umxAPA*)*, 66

t.test(), *66, 67*
tmx_genotypic_effect, 40, *42*, *44*
tmx_is.identified, *41*, 41, *44*
tmx_show, 42, *84*, *160*
tryCatch(), *386*
tvars *(*umx_paste_names*)*, 282

umx, *8*, *10*, *12, 13*, *15–17*, *19, 20*, *22*, *24*, *26*, *29*, *33*, *35*, *39*, *41, 42*, 44, *51*, *57*, *62*, *66, 67*, *69*, *77*, *81*, *86*, *88*, *91*, *93*, *102, 103*, *106, 107*, *110*, *113*, *115, 116*, *120, 121*, *123*, *126*, *129*, *131*, *133, 134*, *136*, *138*, *142*, *144*, *146–150*, *152, 153*, *155, 156*, *158*, *165*, *168–173*, *175*, *177*, *179*, *181*, *186*, *188, 189*, *192*, *194, 195*, *197*, *199*, *201*, *203, 204*, *206*, *208*, *210*, *212*, *215*, *217, 218*, *220*, *222–226*, *229*, *233*, *235–243*, *248*, *250, 251*, *254*, *256*, *258*, *260*, *263–266*, *268*, *270*, *274–276*, *278*, *280, 281*, *283–286*, *288, 289*, *291–294*, *296–299*, *301*, *303–305*, *307, 308*, *310–320*, *322–327*, *329, 330*, *332–339*, *341*, *343–345*, *347–349*, *351*, *353–355*, *357*, *359–361*, *365*, *367–369*, *371–373*, *377*, *380, 381*, *383–385*, *387–392*, *394–399*, *401*, *403–405*, *407, 408*
umx(), *93*
umx-deprecated, 47
umx2ord *(*umx_cont_2_quantiles*)*, 232
umx::oddsratio(), *31*
umx::reliability(), *32*
umx_aggregate, *19*, *44*, *67*, *103*, *107*, *138*, *169*, 221, *278*, *320*
umx_aggregate(), *224*, *243*
umx_APA_pval, *45*, *134*, *173*, *215*, 222, *239, 240*, *263*, *317, 318*, *327*, *329, 330*, *332–339*, *341*, *343–345*, *347–349*, *351*, *353–355*, *357*, *359–361*, *365*, *367–369*, *371–373*, *377*, *380, 381*, *383–385*, *387–392*, *394–399*, *401*, *403–405*, *407, 408*
umx_apply, *13*, *20*, *35*, *39*, *45*, *77*, *116*, *220*, 224, *235*, *274*, *296–298*, *323*
umx_apply(), *222*
umx_array_shift, *16*, *33*, *45*, *69*, *126*, *168*, *220*, 225, *238*, *276*, *280, 281*, *288*, *301*
umx_as_numeric, *45*, *102*, *116*, 225, *233*, *260*, *264–266*, *270*, *284–286*, *289*, *291*, *293*, *303*, *316*
umx_check, 226, *228, 229*, *231, 232*, *244–247*, *249*, *252–254*, *257*
umx_check_model, *227*, 227, *229*, *231, 232*, *244–247*, *249*, *252–254*, *257*
umx_check_names, *45*, *227, 228*, 229, *231, 232*, *244–254*, *256, 257*, *291*
umx_check_names(), *278*
umx_check_OS, *227–229*, 230, *232*, *244–247*, *249*, *252–254*, *257*
umx_check_parallel, *227–229*, *231*, 231, *244–247*, *249*, *252–254*, *257*
umx_checkpoint *(*umx_set_checkpoint*)*, 305
umx_cont_2_quantiles, *45*, *102*, *116*, *226*, 232, *260*, *264–266*, *270*, *284–286*, *289*, *291*, *293*, *303*, *316*
umx_cor, *13*, *20*, *35*, *39*, *45*, *77*, *116*, *220*, *224*, 234, *274*, *296–298*, *323*
umx_explode, *45*, 235, *236*, *243*, *278*, *283*, *296*, *319*, *322*
umx_explode(), *319*
umx_explode_twin_names, *45*, *235*, 236, *243*, *278*, *283*, *296*, *319*, *322*
umx_explode_twin_names(), *283*
umx_factor *(*umxFactor*)*, 101
umx_file_load_pseudo, *8*, *45*, 237, *268*, *275*, *280*, *292*, *325*
umx_find_object, *16*, *33*, *45*, *69*, *126*, *168*, *220*, *225*, 238, *276*, *280, 281*, *288*, *301*

umx_fun_mean_sd, *45*, *134*, *173*, *215*, *223*, 239, *240*, *263*, *317, 318*, *327*, *329, 330*, *332–339*, *341*, *343–345*, *347–349*, *351*, *353–355*, *357*, *359–361*, *365*, *367–369*, *371–373*, *377*, *380, 381*, *383–385*, *387–392*, *394–399*, *401*, *403–405*, *407, 408*
umx_fun_mean_sd(), *239*
umx_get_bracket_addresses, *45*, *134*, *173*, *215*, *223*, *239*, 240, *263*, *317, 318*, *327*, *329, 330*, *332–339*, *341*, *343–345*, *347–349*, *351*, *353–355*, *357*, *359–361*, *365*, *367–369*, *371–373*, *377*, *380, 381*, *383–385*, *387–392*, *394–399*, *401*, *403–405*, *407, 408*
umx_get_checkpoint, *45*, 241, *242*, *303–305*, *307, 308*, *310–315*
umx_get_options, *45*, *241*, 242, *303–305*, *307, 308*, *310–315*
umx_grep, *45*, *235, 236*, 243, *278*, *283*, *296*, *319*, *322*
umx_has_been_run, *227–229*, *231, 232*, 244, *245–247*, *249*, *252–254*, *257*
umx_has_CIs, *227–229*, *231, 232*, *244*, 245, *246, 247*, *249*, *252–254*, *257*
umx_has_means, *227–229*, *231, 232*, *244, 245*, 246, *247*, *249*, *252–254*, *257*
umx_has_square_brackets, *227–229*, *231, 232*, *244–246*, 247, *249*, *252–254*, *257*
umx_is_class, *45*, *229*, 248, *250, 251*, *254*, *256*
umx_is_class(), *254*
umx_is_cov, *227–229*, *231, 232*, *244–247*, 249, *252–254*, *257*
umx_is_endogenous, *45*, *229*, *248*, 250, *251*, *254*, *256*
umx_is_exogenous, *45*, *229*, *248*, *250*, 251, *254*, *256*
umx_is_MxData, *227–229*, *231, 232*, *244–247*, *249*, 252, *253, 254*, *257*
umx_is_MxMatrix, *227–229*, *231, 232*, *244–247*, *249*, *252*, 252, *254*, *257*
umx_is_MxModel, *227–229*, *231, 232*, *244–247*, *249*, *252, 253*, 253, *257*
umx_is_numeric, *45*, *229*, *248*, *250, 251*, 254, *256*
umx_is_numeric(), *248*
umx_is_ordered, *45*, *229*, *248*, *250, 251*, *254*, 255
umx_is_ordered(), *83*
umx_is_RAM, *227–229*, *231, 232*, *244–247*, *249*, *252–254*, 256
umx_long2wide, *45*, 257, *270*, *274*, *294*, *299*, *324*
umx_long2wide(), *274*
umx_lower2full, *45*, *102*, *116*, *226*, *233*, 259, *264–266*, *270*, *284–286*, *289*, *291*, *293*, *303*, *316*
umx_make, *45*, *134*, *173*, *215*, *223*, *239, 240*, 262, *317, 318*, *327*, *329, 330*, *332–339*, *341*, *343–345*, *347–349*, *351*, *353–355*, *357*, *359–361*, *365*, *367–369*, *371–373*, *377*, *380, 381*, *383–385*, *387–392*, *394–399*, *401*, *403–405*, *407, 408*
umx_make_fake_data, *45*, *102*, *116*, *226*, *233*, *260*, 263, *265, 266*, *270*, *284–286*, *289*, *291*, *293*, *303*, *316*
umx_make_MR_data, *45*, *102*, *116*, *226*, *233*, *260*, *264*, 265, *266*, *270*, *284–286*, *289*, *291*, *293*, *303*, *316*
umx_make_MR_data(), *131*
umx_make_raw_from_cov, *45*, *102*, *116*, *226*, *233*, *260*, *264, 265*, 266, *270*, *284–286*, *289*, *291*, *293*, *303*, *316*
umx_make_sql_from_excel, *8*, *45*, *237*, 267, *275*, *280*, *292*, *325*
umx_make_twin_data_nice, *45*, *258*, *270*, 273, *294*, *299*, *324*
umx_make_TwinData, *45*, *102*, *116*, *226*, *233*, *258*, *260*, *264–266*, 268, *274*, *284–286*, *289*, *291*, *293, 294*, *299*, *303*, *316*, *324*
umx_means, *13*, *20*, *35*, *39*, *45*, *77*, *116*, *220*, *224*, *235*, 274, *296–298*, *323*
umx_move_file, *8*, *45*, *237*, *268*, 275, *280*, *292*, *325*
umx_msg, *16*, *33*, *45*, *69*, *126*, *168*, *220*, *225*, *238*, 276, *280, 281*, *288*, *301*
umx_msg(), *288*
umx_names, *19*, *44, 45*, *67*, *103*, *107*, *138*, *169*, *222*, *235, 236*, *243*, 277, *283*, *296*, *319, 320*, *322*
umx_open, *8*, *45*, *237*, *268*, *275*, 279, *292*, *325*

umx_open_CRAN_page, *16*, *33*, *45*, *69*, *126*, *168*, *220*, *225*, *238*, *276*, 280, *281*, *288*, *301*
umx_pad, *16*, *33*, *45*, *69*, *126*, *168*, *220*, *225*, *238*, *276*, *280*, 281, *288*, *301*
umx_paste_names, *45*, *235, 236*, *243*, *278*, 282, *296*, *319*, *322*
umx_polychoric, *45*, *102*, *116*, *226*, *233*, *260*, *264–266*, *270*, 283, *285, 286*, *289*, *291*, *293*, *303*, *316*
umx_polypairwise, *45*, *102*, *116*, *226*, *233*, *260*, *264–266*, *270*, *284*, 284, *286*, *289*, *291*, *293*, *303*, *316*
umx_polytriowise, *45*, *102*, *116*, *226*, *233*, *260*, *264–266*, *270*, *284, 285*, 285, *289*, *291*, *293*, *303*, *316*
umx_print, *16*, *33*, *45*, *69*, *126*, *168*, *220*, *225*, *238*, *276*, *280, 281*, 287, *301*
umx_r_test, *13*, *20*, *35*, *39*, *45*, *77*, *116*, *220*, *224*, *235*, *274*, *296*, 297, *298*, *323*
umx_r_test(), *20*
umx_read_lower, *45*, *102*, *116*, *226*, *233*, *260*, *264–266*, *270*, *284–286*, 288, *291*, *293*, *303*, *316*
umx_rename, *45*, *102*, *116*, *226*, *233*, *260*, *264–266*, *270*, *284–286*, *289*, 290, *293*, *303*, *316*
umx_rename_file, *8*, *45*, *237*, *268*, *275*, *280*, 291, *325*
umx_reorder, *45*, *102*, *116*, *226*, *233*, *260*, *264–266*, *270*, *284–286*, *289*, *291*, 293, *303*, *316*
umx_residualize, *45*, *258*, *270*, *274*, 294, *299*, *324*
umx_residualize(), *50*
umx_rot, *45*, *235, 236*, *243*, *278*, *283*, 295, *319*, *322*
umx_round, *13*, *20*, *35*, *39*, *45*, *77*, *116*, *220*, *224*, *235*, *274*, 296, *297, 298*, *323*
umx_scale, *13*, *20*, *35*, *39*, *45*, *77*, *116*, *220*, *224*, *235*, *274*, *296, 297*, 298, *323*
umx_scale_wide_twin_data, *45*, *258*, *270*, *274*, *294*, 299, *324*
umx_score_scale, *16*, *33*, *45*, *69*, *126*, *168*, *220*, *225*, *238*, *276*, *280, 281*, *288*, 300
umx_select_valid, *45*, *102*, *116*, *226*, *233*, *260*, *264–266*, *270*, *284–286*, *289*, *291*, *293*, 302, *316*
umx_set_auto_plot, *45*, *241, 242*, 303, *304, 305*, *307, 308*, *310–315*
umx_set_auto_plot(), *22*
umx_set_auto_run, *45*, *241, 242*, *303*, 304, *305*, *307, 308*, *310–315*
umx_set_checkpoint, *45*, *241, 242*, *303, 304*, 305, *307, 308*, *310–315*
umx_set_condensed_slots, *45*, *241, 242*, *303–305*, 306, *307, 308*, *310–315*
umx_set_cores, *45*, *241, 242*, *303–305*, *307*, 307, *308*, *310–315*
umx_set_data_variance_check, *45*, *241, 242*, *303–305*, *307*, 308, *310–315*
umx_set_mvn_optimization_options, *45*, *241, 242*, *303–305*, *307, 308*, 309, *310–315*
umx_set_mvn_optimization_options(), *78*, *117*
umx_set_optimizer, *45*, *241, 242*, *303–305*, *307, 308*, *310*, 310, *311–315*
umx_set_plot_file_suffix, *45*, *241, 242*, *303–305*, *307, 308*, *310*, 311, *312–315*
umx_set_plot_format, *45*, *241, 242*, *303–305*, *307, 308*, *310, 311*, 312, *313–315*
umx_set_plot_format(), *22*, *24*, *26*
umx_set_separator, *45*, *241, 242*, *303–305*, *307, 308*, *310–312*, 313, *314, 315*
umx_set_silent, *45*, *241, 242*, *303–305*, *307, 308*, *310–313*, 313, *315*
umx_set_table_format, *45*, *241, 242*, *303–305*, *307, 308*, *310–314*, 314
umx_set_table_format(), *169*, *171*, *287, 288*
umx_stack, *45*, *102*, *116*, *226*, *233*, *260*, *264–266*, *270*, *284–286*, *289*, *291*, *293*, *303*, 315
umx_standardize, *45*, *134*, *173*, *215*, *223*, *239, 240*, *263*, 316, *318*, *327*, *329, 330*, *332–339*, *341*, *343–345*, *347–349*, *351*, *353–355*, *357*, *359–361*, *365*, *367–369*, *371–373*, *377*, *380, 381*, *383–385*, *387–392*, *394–399*, *401*, *403–405*, *407, 408*
umx_standardize(), *47*, *391*, *394–396*
umx_str_chars, *45*, *235, 236*, *243*, *278*, *283*,

296, 318, *319*, *322*
umx_str_from_object, *45*, *235*, *236*, *243*, *278*, *283*, *296*, *319*, 319, *322*
umx_string_to_algebra, *45*, *134*, *173*, *215*, *223*, *239*, *240*, *263*, *317*, 317, *327*, *329*, *330*, *332–339*, *341*, *343–345*, *347–349*, *351*, *353–355*, *357*, *359–361*, *365*, *367–369*, *371–373*, *377*, *380*, *381*, *383–385*, *387–392*, *394–399*, *401*, *403–405*, *407*, *408*
umx_string_to_algebra(), *47*
umx_time, *19*, *44*, *67*, *103*, *107*, *138*, *169*, *222*, *278*, 320
umx_trim, *45*, *235*, *236*, *243*, *278*, *283*, *296*, *319*, 321
umx_update_OpenMx *(*install.OpenMx*)*, 15
umx_var, *13*, *20*, *35*, *39*, *45*, *77*, *116*, *220*, *224*, *235*, *274*, *296–298*, 322
umx_wide2long, *45*, *258*, *270*, *274*, *294*, *299*, 324
umx_wide2long(), *274*
umx_write_to_clipboard, *8*, *45*, *237*, *268*, *275*, *280*, *292*, 325
umxACE, *26*, *29*, *44*, 47, *57*, *62*, *81*, *86*, *88*, *110*, *113*, *115*, *120*, *175*, *181*, *186*
umxACE(), *55*, *80*, *81*, *119*, *143*, *144*, *146*, *169*, *170*, *191*, *192*, *270*, *376*, *391*
umxACEcov, *26*, *29*, *44*, *51*, 55, *62*, *81*, *86*, *88*, *110*, *113*, *115*, *120*, *175*, *181*, *186*
umxACEcov(), *193*, *194*, *283*, *392*
umxACEv, *26*, *29*, *44*, *51*, *57*, 58, *81*, *86*, *88*, *110*, *113*, *115*, *120*, *175*, *181*, *186*
umxACEv(), *147*, *194*, *195*, *376*, *393*
umxAlgebra, *44*, 65, *129*, *136*, *142*, *165*, *177*, *189*, *210*
umxAPA, *19*, *44*, 66, *103*, *107*, *138*, *169*, *222*, *278*, *320*
umxAPA(), *39*, *188*, *222*, *223*
umxBrownie, *16*, *33*, *45*, 69, *126*, *168*, *220*, *225*, *238*, *276*, *280*, *281*, *288*, *301*
umxCI, *11*, *18*, *36–39*, 70, *73*, *74*, *76*, *99*, *100*, *105*, *147*, *174*, *190*
umxCI(), *47*, *71*, *76*, *176*
umxCI_boot, *11*, *18*, *36–39*, *71*, 72, *74*, *76*, *99*, *100*, *105*, *147*, *174*, *190*
umxCI_boot(), *99*
umxCompare, *11*, *18*, *36–39*, *71*, *73*, 73, *76*, *99*, *100*, *105*, *147*, *174*, *190*
umxCompare(), *11*, *93*
umxConfint, *11*, *18*, *36–39*, *71*, *73*, *74*, 75, *99*, *100*, *105*, *147*, *174*, *190*
umxConfint(), *71*, *351*
umxCov2cor, *13*, *20*, *35*, *39*, *45*, 77, *116*, *220*, *224*, *235*, *274*, *296–298*, *323*
umxCP, *26*, *29*, *44*, *51*, *57*, *62*, 78, *86*, *88*, *110*, *113*, *115*, *120*, *175*, *181*, *186*, *197*
umxCP(), *81*, *148*, *173*, *175*, *196*, *199*, *376*, *394*, *395*
umxDiagnose, *43*, 83, *160*
umxDoC, *26*, *29*, *44*, *51*, *57*, *62*, *81*, 84, *88*, *110*, *113*, *115*, *120*, *175*, *181*, *186*
umxDoC(), *9*, *10*, *87*, *88*, *149*, *150*, *198*, *199*
umxDoCp, *26*, *29*, *44*, *51*, *57*, *62*, *81*, *86*, 87, *110*, *113*, *115*, *120*, *175*, *181*, *186*
umxEFA, *44*, 88, *131*
umxEquate, *44*, 91, *106*, *133*, *136*, *179*, *217*
umxEquate(), *122*
umxExampleCode_TRHGpaper *(*umxExamples*)*, 93
umxExamples, 93
umxExpCov, *11*, *18*, *36–39*, *71*, *73*, *74*, *76*, 99, *100*, *105*, *147*, *174*, *190*
umxExpCov(), *73*
umxExpMeans, *11*, *18*, *36–39*, *71*, *73*, *74*, *76*, *99*, 100, *105*, *147*, *174*, *190*
umxExpMeans(), *73*
umxFactanal *(*umxEFA*)*, 88
umxFactanal(), *102*
umxFactor, *45*, 101, *116*, *226*, *233*, *260*, *264–266*, *270*, *284–286*, *289*, *291*, *293*, *303*, *316*
umxFactorScores, *19*, *44*, *67*, 102, *107*, *138*, *169*, *222*, *278*, *320*
umxFitIndices, *11*, *18*, *36–39*, *71*, *73*, *74*, *76*, *99*, *100*, 104, *147*, *174*, *190*
umxFixAll, *44*, *93*, 105, *133*, *136*, *179*, *217*
umxGetParameters, *19*, *44*, *67*, *103*, 106, *138*, *169*, *222*, *278*, *320*
umxGetParameters(), *92*, *138*
umxGxE, *26*, *29*, *44*, *51*, *57*, *62*, *81*, *86*, *88*, 108, *113*, *115*, *120*, *175*, *181*, *186*
umxGxE(), *81*, *112*, *115*, *151*, *152*, *169*, *171*, *200*, *201*, *270*, *376*
umxGxE_window, *26*, *29*, *44*, *51*, *57*, *62*, *81*, *86*, *88*, *110*, *113*, 113, *120*, *175*, *181*, *186*
umxGxE_window(), *110*

umxGxEbiv, *26*, *29*, *44*, *51*, *57*, *62*, *81*, *86*, *88*, *110*, 111, *115*, *120*, *175*, *181*, *186*, *203*
umxGxEbiv(), *112*, *153*, *202*, *270*
umxHetCor, *13*, *20*, *35*, *39*, *45*, *77*, *102*, 115, *220*, *224*, *226*, *233*, *235*, *260*, *264–266*, *270*, *274*, *284–286*, *289*, *291*, *293*, *296–298*, *303*, *316*, *323*
umxHetCor(), *234*
umxIP, *26*, *29*, *44*, *51*, *57*, *62*, *81*, *86*, *88*, *110*, *113*, *115*, 117, *175*, *181*, *186*, *204*
umxIP(), *81*, *154, 155*, *203, 204*, *376*, *396*
umxJiggle, *45*, 121, *123*, *212*, *218*
umxLabel, *45*, *121*, 122, *212*, *218*
umxLabel(), *129*, *179*, *329–331*, *338*, *344*, *351*
umxLav2RAM, *16*, *33*, *45*, *69*, 124, *168*, *220*, *225*, *238*, *276*, *280, 281*, *288*, *301*
umxLav2RAM(), *162*, *165*, *371*
umxMatrix, *44*, *66*, 128, *136*, *142*, *165*, *177*, *189*, *210*
umxMatrix(), *66*, *363*, *390*
umxMendelianRandomization, *44*, *91*, 130
umxMI, *44*, *93*, *106*, 132, *136*, *179*, *217*
umxMI(), *336*
umxModel, *45*, 133, *173*, *215*, *223*, *239, 240*, *263*, *317, 318*, *327*, *329, 330*, *332–339*, *341*, *343–345*, *347–349*, *351*, *353–355*, *357*, *359–361*, *365*, *367–369*, *371–373*, *377*, *380, 381*, *383–385*, *387–392*, *394–399*, *401*, *403–405*, *407, 408*
umxModify, *44*, *66*, *93*, *106*, *129*, *133*, 134, *142*, *165*, *177*, *179*, *189*, *210*, *217*
umxModify(), *10*, *51*, *71*, *86*, *93*, *150*, *177*, *179*, *192*, *199*, *218*
umxParameters, *19*, *44*, *67*, *103*, *107*, 137, *169*, *222*, *278*, *320*
umxParameters(), *137*
umxPath, *44*, *66*, *129*, *136*, 139, *165*, *177*, *189*, *210*
umxPath(), *87, 88*, *126*, *161, 162*, *164, 165*, *215*, *384*
umxPlot *(*plot.MxModel*)*, 22
umxPlotACE, *22*, *24*, *44*, 143, *146–148*, *152, 153*, *155, 156*, *158*
umxPlotACE(), *22*, *24*, *26*, *51*
umxPlotACEcov, *22*, *24*, *44*, *144*, 145, *147, 148*, *152, 153*, *155, 156*, *158*
umxPlotACEv, *11*, *18*, *22*, *24*, *36–39*, *44*, *71*, *73, 74*, *76*, *99, 100*, *105*, *144*, *146*, 146, *148*, *152, 153*, *155, 156*, *158*, *174*, *190*
umxPlotCP, *22*, *24*, *44*, *144*, *146, 147*, 147, *150*, *152, 153*, *155, 156*, *158*, *169*, *171, 172*, *188*, *192*, *194, 195*, *197*, *199*, *201*, *203, 204*, *206*, *208*
umxPlotCP(), *22*, *24*, *26*, *81*
umxPlotDoC, *44*, *149*, 149, *169*, *171, 172*, *188*, *192*, *194, 195*, *197*, *199*, *201*, *203, 204*, *206*, *208*
umxPlotGxE, *22*, *24*, *44*, *144*, *146–148*, 151, *153*, *155, 156*, *158*
umxPlotGxE(), *22*, *24*, *26*
umxPlotGxEbiv, *22*, *24*, *44*, *144*, *146–148*, *152*, 152, *155, 156*, *158*
umxPlotIP, *22*, *24*, *44*, *144*, *146–148*, *152, 153*, 154, *156*, *158*
umxPlotIP(), *22*, *24*, *26*
umxPlotMxModelTwinMaker *(*plot.MxModelTwinMaker*)*, 25
umxPlotSexLim, *22*, *24*, *44*, *144*, *146–148*, *152, 153*, *155*, 155, *158*
umxPlotSexLim(), *181*, *206*
umxPlotSimplex, *22*, *24*, *44*, *144*, *146–148*, *152, 153*, *155, 156*, 157
umxPower, *43*, *84*, 158
umxRAM, *44*, *66*, *129*, *136*, *142*, 161, *177*, *189*, *210*
umxRAM(), *44*, *74*, *124*, *126*, *129*, *131*, *134*, *160*, *164*, *210*, *214, 215*, *313*, *339*, *381*, *384*, *386*
umxRAM2Lav, *16*, *33*, *45*, *69*, *126*, 168, *220*, *225*, *238*, *276*, *280, 281*, *288*, *301*
umxReduce, *19*, *44*, *67*, *103*, *107*, *138*, *149, 150*, 169, *171, 172*, *188*, *192*, *194, 195*, *197*, *199*, *201*, *203, 204*, *206*, *208*, *222*, *278*, *320*
umxReduce(), *110*, *113*, *171, 172*, *201*
umxReduceACE, *44*, *149, 150*, *169*, 170, *172*, *188*, *192*, *194, 195*, *197*, *199*, *201*, *203, 204*, *206*, *208*
umxReduceACE(), *169*, *172*
umxReduceGxE, *44*, *149, 150*, *169*, *171*, 171, *188*, *192*, *194, 195*, *197*, *199*, *201*, *203, 204*, *206*, *208*
umxReduceGxE(), *169*, *171*

umxRenameMatrix, *45*, *134*, 172, *215*, *223*, *239*, *240*, *263*, *317*, *318*, *327*, *329*, *330*, *332–339*, *341*, *343–345*, *347–349*, *351*, *353–355*, *357*, *359–361*, *365*, *367–369*, *371–373*, *377*, *380*, *381*, *383–385*, *387–392*, *394–399*, *401*, *403–405*, *407*, *408*
umxRotate, *11*, *18*, *36–39*, *71*, *73*, *74*, *76*, *99*, *100*, *105*, *147*, 173, *190*
umxRotate.MxModelCP, *26*, *29*, *44*, *51*, *57*, *62*, *81*, *86*, *88*, *110*, *113*, *115*, *120*, 174, *181*, *186*
umxRotate.MxModelCP(), *81*, *173*
umxRun, *44*, *66*, *129*, *136*, *142*, *165*, 176, *189*, *210*
umxRun(), *47*, *99*, *190*
umxSetParameters, *44*, *93*, *106*, *133*, *136*, 177, *217*
umxSexLim, *26*, *29*, *44*, *51*, *57*, *62*, *81*, *86*, *88*, *110*, *113*, *115*, *120*, *175*, 179, *186*
umxSexLim(), *156*, *205*, *206*, *398*
umxSimplex, *26*, *29*, *44*, *51*, *57*, *62*, *81*, *86*, *88*, *110*, *113*, *115*, *120*, *175*, *181*, 183
umxSimplex(), *157*, *158*, *207*, *208*, *399*
umxSummarizeTwinData, *44*, *149*, *150*, *169*, *171*, *172*, 187, *192*, *194*, *195*, *197*, *199*, *201*, *203*, *204*, *206*, *208*
umxSummary, *44*, *66*, *129*, *136*, *142*, *165*, *177*, 188, *210*
umxSummary(), *47*, *74*, *81*, *89*, *110*, *113*, *120*, *138*, *144*, *146*, *148*, *150*, *152*, *153*, *155*, *158*, *162*, *165*, *186*, *192*, *195*, *197*, *199*, *203*, *204*, *206*, *208*, *386*, *389*
umxSummary.MxModel, *11*, *18*, *36–39*, *71*, *73*, *74*, *76*, *99*, *100*, *105*, *147*, *174*, 189
umxSummary.MxModel(), *188*
umxSummary.MxModelACE *(*umxSummaryACE*)*, 191
umxSummary.MxModelACEcov *(*umxSummaryACEcov*)*, 193
umxSummary.MxModelACEv *(*umxSummaryACEv*)*, 194
umxSummary.MxModelCP *(*umxSummaryCP*)*, 196
umxSummary.MxModelDoC *(*umxSummaryDoC*)*, 198
umxSummary.MxModelDoC(), *10*, *86*, *150*
umxSummary.MxModelGxE *(*umxSummaryGxE*)*, 200
umxSummary.MxModelGxEbiv *(*umxSummaryGxEbiv*)*, 202
umxSummary.MxModelIP *(*umxSummaryIP*)*, 203
umxSummary.MxModelSexLim *(*umxSummarySexLim*)*, 205
umxSummary.MxModelSimplex *(*umxSummarySimplex*)*, 207
umxSummary.MxRAMModel *(*umxSummary.MxModel*)*, 189
umxSummaryACE, *44*, *149*, *150*, *169*, *171*, *172*, *188*, 191, *194*, *195*, *197*, *199*, *201*, *203*, *204*, *206*, *208*
umxSummaryACE(), *51*, *188*
umxSummaryACEcov, *44*, *149*, *150*, *169*, *171*, *172*, *188*, *192*, 193, *195*, *197*, *199*, *201*, *203*, *204*, *206*, *208*
umxSummaryACEv, *44*, *149*, *150*, *169*, *171*, *172*, *188*, *192*, *194*, 194, *197*, *199*, *201*, *203*, *204*, *206*, *208*
umxSummaryACEv(), *188*
umxSummaryCP, *45*, *149*, *150*, *169*, *171*, *172*, *188*, *192*, *194*, *195*, 196, *199*, *201*, *203*, *204*, *206*, *208*
umxSummaryCP(), *81*, *188*
umxSummaryDoC, *45*, *149*, *150*, *170–172*, *188*, *192*, *194*, *195*, *197*, 198, *201*, *203*, *204*, *206*, *208*
umxSummaryGxE, *45*, *149*, *150*, *170–172*, *188*, *192*, *194*, *195*, *197*, *199*, 200, *203*, *204*, *206*, *208*
umxSummaryGxE(), *188*
umxSummaryGxEbiv, *45*, *149*, *150*, *170–172*, *188*, *192*, *194*, *195*, *197*, *199*, *201*, 202, *204*, *206*, *208*
umxSummaryIP, *45*, *149*, *150*, *170–172*, *188*, *192*, *194*, *195*, *197*, *199*, *201*, *203*, 203, *206*, *208*
umxSummaryIP(), *188*
umxSummarySexLim, *45*, *149*, *150*, *170–172*, *188*, *192*, *194*, *195*, *197*, *199*, *201*, *203*, *204*, 205, *208*
umxSummarySexLim(), *156*, *181*
umxSummarySimplex, *45*, *149*, *150*, *170–172*, *188*, *192*, *194*, *195*, *197*, *199*, *201*, *203*, *204*, *206*, 207
umxSuperModel, *44*, *66*, *129*, *136*, *142*, *165*, *177*, *189*, 209

umxSuperModel(), *124*, *165*, *214*, *215*, *342*
umxThresholdMatrix, *45*, *121*, *123*, 211, *218*
umxThresholdMatrix(), *47*, *376*, *377*
umxTwinMaker, *45*, *134*, *173*, 214, *223*, *239*, *240*, *263*, *317*, *318*, *327*, *329*, *330*, *332–339*, *341*, *343–345*, *347–349*, *351*, *353–355*, *357*, *359–361*, *365*, *367–369*, *371–373*, *377*, *380*, *381*, *383–385*, *387–392*, *394–399*, *401*, *403–405*, *407*, *408*
umxTwinMaker(), *25*, *384*, *385*
umxTwoStage *(*umxMendelianRandomization*)*, 130
umxUnexplainedCausalNexus, *44*, *93*, *106*, *133*, *136*, *179*, 217
umxValues, *45*, *121*, *123*, *212*, 218
umxVersion, *16*, *33*, *45*, *69*, *126*, *168*, 219, *225*, *238*, *276*, *280*, *281*, *288*, *301*
umxVersion(), *16*
umxWeightedAIC, *13*, *20*, *35*, *39*, *45*, *77*, *116*, 220, *224*, *235*, *274*, *296–298*, *323*
us_skinfold_data, *10*, *12*, *15*, *17*, *45*, 325

vcov(), *99*
vcov.MxModel *(*umxExpCov*)*, 99

within(), *303*

xmu_cell_is_on, *45*, *134*, *173*, *216*, *223*, *239*, *240*, *263*, *317*, *318*, *327*, *329*, *330*, *332–339*, *341*, *343*, 343, *345*, *347–349*, *351–355*, *357*, *359–361*, *365*, *367*, *368*, *370–372*, *374*, *377*, *380*, *381*, *383–385*, *387–392*, *394–399*, *401*, *403–405*, *407*, *408*
xmu_check_levels_identical, *45*, *134*, *173*, *216*, *223*, *239*, *240*, *263*, *317*, *318*, *327*, *329*, *330*, *332–338*, *340*, *341*, *343*, *344*, 345, *347–349*, *351–355*, *357*, *359–361*, *365*, *367*, *368*, *370–372*, *374*, *377*, *380*, *381*, *383–385*, *387–392*, *394–399*, *401*, *403–405*, *407*, *408*
xmu_check_needs_means, *45*, *134*, *173*, *216*, *223*, *239*, *240*, *263*, *317*, *318*, *327*, *329*, *330*, *332–338*, *340*, *341*, *343–345*, 346, *348*, *349*, *351–355*, *357*, *359–361*, *365*, *367*, *368*, *370–372*, *374*, *377*, *380*, *381*, *383–385*, *387–392*, *394–399*, *401*, *403–405*, *407*, *408*
xmu_check_variance, *45*, *134*, *173*, *216*, *223*, *239*, *240*, *263*, *317*, *318*, *327*, *329*, *330*, *332–338*, *340*, *341*, *343–345*, *347*, 348, *349*, *351–355*, *357*, *359–361*, *365*, *367*, *368*, *370–372*, *374*, *377*, *380*, *381*, *383–385*, *387–392*, *394–399*, *401*, *403–405*, *407*, *408*
xmu_CI_merge, *45*, *134*, *173*, *216*, *223*, *239*, *240*, *263*, *317*, *318*, *327*, *329*, *330*, *332–339*, *341*, *343–345*, *347*, *348*, 349, *351–355*, *357*, *359–361*, *365*, *367*, *368*, *370–372*, *374*, *377*, *380*, *381*, *383–385*, *387–392*, *394–399*, *401*, *403–405*, *407*, *408*
xmu_CI_stash, *45*, *134*, *173*, *216*, *223*, *239*, *240*, *263*, *317*, *318*, *327*, *329*, *330*, *332–339*, *341*, *343–345*, *347–349*, 350, *352–355*, *357*, *359–361*, *365*, *367*, *368*, *370–372*, *374*, *377*, *380*, *381*, *383–385*, *387–392*, *394–399*, *401*, *403–405*, *407*, *408*
xmu_clean_label, *45*, *134*, *173*, *216*, *223*, *239*, *240*, *263*, *317*, *318*, *327*, *329*, *330*, *332–338*, *340*, *341*, *343–345*, *347–349*, *351*, 351, *353*, *354*, *356*, *357*, *359–361*, *365*, *367*, *368*, *370–372*, *374*, *377*, *380*, *382–385*, *387–392*, *394–399*, *401*, *403–405*, *407*, *408*
xmu_data_missing, *45*, *134*, *173*, *216*, *223*, *239*, *240*, *263*, *317*, *318*, *327*, *329*, *330*, *332–338*, *340*, *341*, *343–345*, *347–349*, *351*, *352*, 352, *354*, *356*, *357*, *359–361*, *365*, *367*, *368*, *370–372*, *374*, *377*, *380*, *382–385*, *387–392*, *394–399*, *401*, *403–405*, *407*, *408*
xmu_data_swap_a_block, *45*, *134*, *173*, *216*, *223*, *239*, *240*, *263*, *317*, *318*, *327*, *329*, *330*, *332–338*, *340*, *341*, *343–345*, *347–349*, *351–353*, 353, *356*, *357*, *359–361*, *365*, *367*, *368*, *370–372*, *374*, *377*, *380*, *382–385*, *387–392*, *394–399*, *401*, *403–405*,

407, 408
xmu_describe_data_WLS, *45*, *134*, *173*, *216*, *223*, *239, 240*, *263*, *317, 318*, *327*, *329, 330*, *332–338*, *340, 341*, *343–345*, *347–349*, *351–354*, 355, *357*, *359–361*, *365*, *367, 368*, *370–372*, *374*, *377*, *380*, *382–385*, *387–392*, *394–399*, *401*, *403–405*, *407, 408*
xmu_DF_to_mxData_TypeCov, *45*, *134*, *173*, *216*, *223*, *239, 240*, *263*, *317, 318*, *327*, *329, 330*, *332–339*, *341*, *343–345*, *347–349*, *351–355*, 356, *359–361*, *365*, *367, 368*, *370–372*, *374*, *377*, *380, 381*, *383–385*, *387–392*, *394–399*, *401*, *403–405*, *407, 408*
xmu_dot_define_shapes, 358, *359, 360*, *362, 363*, *366*
xmu_dot_define_shapes(), *366*
xmu_dot_make_paths, *45*, *134*, *173*, *216*, *223*, *239, 240*, *263*, *317, 318*, *327*, *329, 330*, *332–338*, *340, 341*, *343–345*, *347–349*, *351–354*, *356–359*, 359, *361–363*, *365–368*, *370–372*, *374*, *377*, *380*, *382–385*, *387–392*, *394–399*, *401*, *403–405*, *407, 408*
xmu_dot_make_residuals, *45*, *134*, *173*, *216*, *223*, *239, 240*, *263*, *317, 318*, *327*, *329, 330*, *332–338*, *340, 341*, *343–345*, *347, 348*, *350–354*, *356–360*, 361, *363*, *365–368*, *370–372*, *374*, *377*, *380*, *382–385*, *387–392*, *394–399*, *401*, *403–405*, *407, 408*
xmu_dot_maker, *45*, *134*, *173*, *216*, *223*, *239, 240*, *263*, *317, 318*, *327*, *329, 330*, *332–338*, *340, 341*, *343–345*, *347, 348*, *350–354*, *356–358*, 358, *360–363*, *365–368*, *370–372*, *374*, *377*, *380*, *382–385*, *387–392*, *394–399*, *401*, *403–405*, *407, 408*
xmu_dot_mat2dot, *358–360*, *362*, 362, *366*
xmu_dot_move_ranks, *45*, *134*, *173*, *216*, *223*, *239*, *241*, *263*, *317, 318*, *327*, *329, 330*, *332–338*, *340, 341*, *343–345*, *347, 348*, *350–354*, *356, 357*, *359–361*, 365, *367, 368*, *370–372*, *374*, *377*, *380*, *382–385*, *387–392*, *394–399*, *401*, *403–405*, *407, 408*
xmu_dot_rank, *358–360*, *362, 363*, 366
xmu_dot_rank_str, *45*, *134*, *173*, *216*, *223*, *239*, *241*, *263*, *317, 318*, *327*, *329, 330*, *332–338*, *340, 341*, *343–345*, *347, 348*, *350–354*, *356, 357*, *359–361*, *365*, 367, *368*, *370–372*, *374*, *377*, *380*, *382–385*, *387–392*, *394–399*, *401*, *403–405*, *407, 408*
xmu_extract_column, *45*, *134*, *173*, *216*, *223*, *239*, *241*, *263*, *317, 318*, *327*, *329, 330*, *332–338*, *340, 341*, *343–345*, *347, 348*, *350–354*, *356, 357*, *359–361*, *365*, *367*, 368, *370–372*, *374*, *377*, *380*, *382–385*, *387–392*, *394–399*, *401*, *403–405*, *407, 408*
xmu_get_CI, *45*, *134*, *173*, *216*, *223*, *239*, *241*, *263*, *317, 318*, *327*, *329, 330*, *332–338*, *340, 341*, *343–345*, *347, 348*, *350–354*, *356, 357*, *359–361*, *365*, *367, 368*, 369, *371, 372*, *374*, *377*, *380*, *382–385*, *387–392*, *394–399*, *401*, *403–405*, *407, 408*
xmu_get_CI(), *351*
xmu_lavaan_process_group, *45*, *134*, *173*, *216*, *223*, *239*, *241*, *263*, *317, 318*, *327*, *329, 330*, *332–338*, *340, 341*, *343–345*, *347, 348*, *350–354*, *356, 357*, *359–361*, *365*, *367, 368*, *370*, 370, *372*, *374*, *377*, *380*, *382–385*, *387–392*, *394–399*, *401*, *403–405*, *407, 408*
xmu_make_bin_cont_pair_data, *45*, *134*, *173*, *216*, *223*, *239*, *241*, *263*, *317, 318*, *327*, *329, 330*, *332–338*, *340, 341*, *343–345*, *347, 348*, *350–354*, *356, 357*, *359–361*, *365*, *367, 368*, *370, 371*, 371, *374*, *377*, *380*, *382–385*, *387–391*, *393–399*, *401*, *403–405*, *407, 408*
xmu_make_mxData, *45*, *134*, *173*, *216*, *223*, *239*, *241*, *263*, *317, 318*, *327*, *329, 330*, *332–338*, *340, 341*, *343–345*, *347, 348*, *350–354*, *356, 357*, *359–361*, *365*, *367, 368*, *370–372*, 373, *377*, *380*, *382–385*, *387–391*, *393–399*, *401*, *403–405*, *407, 408*

xmu_make_mxData(), *347*
xmu_make_TwinSuperModel, *45*, *134*, *173*, *216*, *223*, *239*, *241*, *263*, *317, 318*, *327*, *329, 330*, *332–338*, *340, 341*, *343–345*, *347, 348*, *350–354*, *356, 357*, *359–361*, *365*, *367, 368*, *370–372*, *374*, 375, *380*, *382–385*, *387–391*, *393–399*, *401*, *403–405*, *407, 408*
xmu_make_TwinSuperModel(), *81*, *340, 341*
xmu_match.arg, *45*, *134*, *173*, *216*, *223*, *239*, *241*, *263*, *317, 318*, *327*, *329, 330*, *332–338*, *340, 341*, *343–345*, *347, 348*, *350–354*, *356, 357*, *359–361*, *365*, *367, 368*, *370–372*, *374*, *377*, 379, *382–385*, *387–391*, *393–399*, *401*, *403–405*, *407, 408*
xmu_name_from_lavaan_str, *45*, *134*, *173*, *216*, *223*, *239*, *241*, *263*, *317, 318*, *327*, *329, 330*, *332–338*, *340, 341*, *343–345*, *347, 348*, *350–354*, *356, 357*, *359–361*, *365*, *367, 368*, *370–372*, *374*, *377*, *380*, 381, *383–385*, *387–391*, *393–399*, *401*, *403–405*, *407, 408*
xmu_PadAndPruneForDefVars, *45*, *134*, *173*, *216*, *223*, *239, 240*, *263*, *317, 318*, *327*, *329, 330*, *332–339*, *341*, *343–345*, *347–349*, *351–355*, *357*, *359–361*, *365*, *367, 368*, *370–372*, *374*, *377*, *380, 381*, 382, *384, 385*, *387–392*, *394–399*, *401*, *403–405*, *407, 408*
xmu_path2twin, *45*, *134*, *173*, *216*, *223*, *239*, *241*, *263*, *317, 318*, *327*, *329, 330*, *332–338*, *340, 341*, *343–345*, *347, 348*, *350–354*, *356, 357*, *359–361*, *365*, *367, 368*, *370–372*, *374*, *377*, *380*, *382, 383*, 383, *385*, *387–391*, *393–399*, *401*, *403–405*, *407, 408*
xmu_path2twin(), *385*
xmu_path_regex, *45*, *134*, *173*, *216*, *223*, *239*, *241*, *263*, *317, 318*, *327*, *329, 330*, *332–338*, *340, 341*, *343–345*, *347, 348*, *350–354*, *356, 357*, *359–361*, *365*, *367, 368*, *370–372*, *374*, *377*, *380*, *382–384*, 384, *387–391*, *393–399*, *401*, *403–405*, *407, 408*
xmu_safe_run_summary, *45*, *134*, *173*, *216*, *223*, *239*, *241*, *263*, *317, 318*, *327*, *329, 330*, *332–338*, *340, 341*, *343–345*, *347, 348*, *350–354*, *356, 357*, *359–361*, *365*, *367, 368*, *370–372*, *374*, *377*, *380*, *382–385*, 386, *388–391*, *393–399*, *401*, *403–405*, *407, 408*
xmu_set_sep_from_suffix, *45*, *134*, *173*, *216*, *223*, *239*, *241*, *263*, *317, 318*, *327*, *329, 330*, *332–338*, *340, 341*, *343–345*, *347, 348*, *350–354*, *356, 357*, *359–361*, *365*, *367, 368*, *370–372*, *374*, *377*, *380*, *382–385*, *387*, 388, *389–391*, *393–399*, *401*, *403–405*, *407, 408*
xmu_show_fit_or_comparison, *45*, *134*, *173*, *216*, *223*, *239*, *241*, *263*, *317, 318*, *327*, *329, 330*, *332–338*, *340, 341*, *343, 344*, *346, 347*, *349–354*, *356, 357*, *359–361*, *366–368*, *370–372*, *374*, *378*, *380*, *382–385*, *387, 388*, 389, *390, 391*, *393–399*, *401*, *403–405*, *407, 408*
xmu_simplex_corner, *45*, *134*, *173*, *216*, *223*, *239*, *241*, *263*, *317, 318*, *327*, *329, 330*, *332–338*, *340, 341*, *343, 344*, *346, 347*, *349–354*, *356, 357*, *359–361*, *366–368*, *370–372*, *374*, *378*, *380*, *382–385*, *387–389*, 390, *391*, *393–399*, *401*, *403–405*, *407, 408*
xmu_simplex_corner(), *129*
xmu_standardize_ACE, *45*, *134*, *173*, *216*, *223*, *240, 241*, *263*, *317, 318*, *327*, *329, 330*, *332–338*, *340, 341*, *343, 344*, *346, 347*, *349–354*, *356, 357*, *359–361*, *366–368*, *370–372*, *374*, *378*, *380*, *382–385*, *387–390*, 391, *393–398*, *400, 401*, *403–405*, *407, 408*
xmu_standardize_ACEcov, *45*, *134*, *173*, *216*, *223*, *239*, *241*, *263*, *317, 318*, *327*, *329, 330*, *332–338*, *340, 341*, *343, 344*, *346, 347*, *349–354*, *356, 357*, *359–361*, *366–368*, *370–372*, *374*, *378*, *380*, *382–385*, *387–391*, 392, *394–399*, *401*, *403–405*, *407, 408*

xmu_standardize_ACEv, *45*, *134*, *173*, *216*, *223*, *240, 241*, *263*, *317, 318*, *327*, *329, 330*, *332–338*, *340, 341*, *343, 344*, *346, 347*, *349–354*, *356, 357*, *359–361*, *366–368*, *370–372*, *374*, *378*, *380*, *382–385*, *387–391*, *393*, 393, *395–398*, *400, 401*, *403–405*, *407, 408*

xmu_standardize_CP, *45*, *134*, *173*, *216*, *223*, *240, 241*, *263*, *317, 318*, *327*, *329, 330*, *332–338*, *340, 341*, *343, 344*, *346, 347*, *349–354*, *356, 357*, *359–361*, *366–368*, *370–372*, *374*, *378*, *380*, *382–385*, *387–391*, *393, 394*, 394, *396–398*, *400, 401*, *403–405*, *407, 408*

xmu_standardize_IP, *45*, *134*, *173*, *216*, *223*, *240, 241*, *263*, *317, 318*, *327*, *329, 330*, *332–338*, *340, 341*, *343, 344*, *346, 347*, *349–354*, *356, 357*, *359, 360*, *362*, *366–368*, *370–372*, *374*, *378*, *380*, *382–385*, *387–391*, *393–395*, 395, *397, 398*, *400, 401*, *403–405*, *407, 408*

xmu_standardize_RAM, *45*, *134*, *173*, *216*, *223*, *240, 241*, *263*, *317, 318*, *327*, *329, 330*, *332–338*, *340, 341*, *343, 344*, *346, 347*, *349–354*, *356, 357*, *359, 360*, *362*, *366–368*, *370–372*, *374*, *378*, *380*, *382–385*, *387–391*, *393–396*, 396, *398*, *400, 401*, *403–405*, *407, 408*

xmu_standardize_SexLim, *45*, *134*, *173*, *216*, *223*, *240, 241*, *263*, *317, 318*, *327*, *329, 330*, *332–338*, *340, 341*, *343, 344*, *346, 347*, *349–354*, *356, 357*, *359, 360*, *362*, *366–368*, *370–372*, *374*, *378*, *380*, *382–385*, *387–391*, *393–397*, 398, *400, 401*, *403–405*, *407, 408*

xmu_standardize_Simplex, *45*, *134*, *173*, *216*, *223*, *240, 241*, *263*, *317, 318*, *327*, *329, 330*, *332–338*, *340, 341*, *343, 344*, *346, 347*, *349–354*, *356, 357*, *359, 360*, *362*, *366–368*, *370–372*, *374*, *378*, *380*, *382–385*, *387–391*, *393–398*, 399, *401*, *403–405*, *407, 408*

xmu_start_value_list, *45*, *134*, *173*, *216*, *223*, *240, 241*, *263*, *317, 318*, *327*, *329, 330*, *332–338*, *340, 341*, *343, 344*, *346, 347*, *349–354*, *356, 357*, *359, 360*, *362*, *366–368*, *370–372*, *374*, *378*, *380*, *382–385*, *387–391*, *393–398*, *400, 401*, 402, *404, 405*, *407, 408*

xmu_starts, *45*, *134*, *173*, *216*, *223*, *240, 241*, *263*, *317, 318*, *327*, *329, 330*, *332–338*, *340, 341*, *343, 344*, *346, 347*, *349–354*, *356, 357*, *359, 360*, *362*, *366–368*, *370–372*, *374*, *378*, *380*, *382–385*, *387–391*, *393–398*, *400*, 400, *403–405*, *407, 408*

xmu_twin_add_WeightMatrices, *45*, *134*, *173*, *216*, *223*, *240, 241*, *263*, *317, 318*, *327*, *329, 330*, *332–338*, *340, 341*, *343, 344*, *346, 347*, *349–354*, *356, 357*, *359, 360*, *362*, *366–368*, *370–372*, *374*, *378*, *380*, *382–385*, *387–391*, *393–398*, *400, 401*, *403*, 403, *405*, *407, 408*

xmu_twin_check, *45*, *134*, *173*, *216*, *223*, *240, 241*, *263*, *317, 318*, *327*, *329, 330*, *332–338*, *340, 341*, *343, 344*, *346, 347*, *349–354*, *356, 357*, *359, 360*, *362*, *366–368*, *370–372*, *374*, *378*, *380*, *382–385*, *387–391*, *393–398*, *400, 401*, *403, 404*, 404, *407, 408*

xmu_twin_get_var_names, *45*, *134*, *173*, *216*, *223*, *240, 241*, *263*, *317, 318*, *327*, *329, 330*, *332–338*, *340, 341*, *343, 344*, *346, 347*, *349–354*, *356, 357*, *359, 360*, *362*, *366–368*, *370–372*, *374*, *378*, *380*, *382–385*, *387–391*, *393–398*, *400, 401*, *403–405*, 406, *408*

xmu_twin_upgrade_selDvs2SelVars, *45*, *134*, *173*, *216*, *223*, *240, 241*, *263*, *317, 318*, *327*, *329, 330*, *332–338*, *340, 341*, *343, 344*, *346, 347*, *349–354*, *356, 357*, *359, 360*, *362*, *366–368*, *370–372*, *374*, *378*, *380*, *382–385*, *387–391*, *393–398*, *400, 401*, *403–405*, *407*, 407

xmuHasSquareBrackets, *45*, *134*, *173*, *215*, *223*, *239, 240*, *263*, *317, 318*, 327,

329, *330*, *332–339*, *341*, *343–345*, *347–349*, *351*, *353–355*, *357*, *359–361*, *365*, *367–369*, *371–373*, *377*, *380*, *381*, *383–385*, *387–392*, *394–399*, *401*, *403–405*, *407*, *408*

xmuLabel_Matrix, *45*, *134*, *173*, *215*, *223*, *239*, *240*, *263*, *317*, *318*, *327*, 328, *330*, *332–339*, *341*, *343–345*, *347–349*, *351*, *353–355*, *357*, *359–361*, *365*, *367–369*, *371–373*, *377*, *380*, *381*, *383–385*, *387–392*, *394–399*, *401*, *403–405*, *407*, *408*

xmuLabel_MATRIX_Model, *45*, *134*, *173*, *215*, *223*, *239*, *240*, *263*, *317*, *318*, *327*, *329*, 329, *332–339*, *341*, *343–345*, *347–349*, *351*, *353–355*, *357*, *359–361*, *365*, *367–369*, *371–373*, *377*, *380*, *381*, *383–385*, *387–392*, *394–399*, *401*, *403–405*, *407*, *408*

xmuLabel_RAM_Model, *45*, *134*, *173*, *215*, *223*, *239*, *240*, *263*, *317*, *318*, *327*, *329*, *330*, 331, *333–339*, *341*, *343–345*, *347–349*, *351*, *353–355*, *357*, *359–361*, *365*, *367–369*, *371–373*, *377*, *380*, *381*, *383–385*, *387–392*, *394–399*, *401*, *403–405*, *407*, *408*

xmuMakeDeviationThresholdsMatrices, *45*, *134*, *173*, *215*, *223*, *239*, *240*, *263*, *317*, *318*, *327*, *329*, *330*, *332*, 332, *334–339*, *341*, *343–345*, *347–349*, *351–355*, *357*, *359–361*, *365*, *367–369*, *371–373*, *377*, *380*, *381*, *383–385*, *387–392*, *394–399*, *401*, *403–405*, *407*, *408*

xmuMakeOneHeadedPathsFromPathList, *45*, *134*, *173*, *215*, *223*, *239*, *240*, *263*, *317*, *318*, *327*, *329*, *330*, *332*, *333*, 333, *335–339*, *341*, *343–345*, *347–349*, *351–355*, *357*, *359–361*, *365*, *367–369*, *371–373*, *377*, *380*, *381*, *383–385*, *387–392*, *394–399*, *401*, *403–405*, *407*, *408*

xmuMakeTwoHeadedPathsFromPathList, *45*, *134*, *173*, *216*, *223*, *239*, *240*, *263*, *317*, *318*, *327*, *329*, *330*, *332–334*, 334, *335–339*, *341*, *343–345*, *347–349*, *351–355*, *357*, *359–361*, *365*, *367–369*, *371–373*, *377*, *380*, *381*, *383–385*, *387–392*, *394–399*, *401*, *403–405*, *407*, *408*

xmuMaxLevels, *45*, *134*, *173*, *216*, *223*, *239*, *240*, *263*, *317*, *318*, *327*, *329*, *330*, *332–335*, 335, *336–339*, *341*, *343–345*, *347–349*, *351–355*, *357*, *359–361*, *365*, *367–369*, *371–373*, *377*, *380*, *381*, *383–385*, *387–392*, *394–399*, *401*, *403–405*, *407*, *408*

xmuMI, *45*, *134*, *173*, *215*, *223*, *239*, *240*, *263*, *317*, *318*, *327*, *329*, *330*, *332–335*, 336, *337–339*, *341*, *343–345*, *347–349*, *351–355*, *357*, *359–361*, *365*, *367–369*, *371–373*, *377*, *380*, *381*, *383–385*, *387–392*, *394–399*, *401*, *403–405*, *407*, *408*

xmuMinLevels, *45*, *134*, *173*, *216*, *223*, *239*, *240*, *263*, *317*, *318*, *327*, *329*, *330*, *332–336*, 337, *338*, *339*, *341*, *343–345*, *347–349*, *351–355*, *357*, *359–361*, *365*, *367–369*, *371–373*, *377*, *380*, *381*, *383–385*, *387–392*, *394–399*, *401*, *403–405*, *407*, *408*

xmuPropagateLabels, *45*, *134*, *173*, *216*, *223*, *239*, *240*, *263*, *317*, *318*, *327*, *329*, *330*, *332–337*, 338, *339*, *341*, *343–345*, *347–349*, *351–355*, *357*, *359–361*, *365*, *367–369*, *371–373*, *377*, *380*, *381*, *383–385*, *387–392*, *394–399*, *401*, *403–405*, *407*, *408*

xmuRAM2Ordinal, *45*, *134*, *173*, *216*, *223*, *239*, *240*, *263*, *317*, *318*, *327*, *329*, *330*, *332–338*, 339, *341*, *343–345*, *347–349*, *351–355*, *357*, *359–361*, *365*, *367–369*, *371*, *372*, *374*, *377*, *380*, *381*, *383–385*, *387–392*, *394–399*, *401*, *403–405*, *407*, *408*

xmuTwinSuper_Continuous, *45*, *134*, *173*, *216*, *223*, *239*, *240*, *263*, *317*, *318*, *327*, *329*, *330*, *332–339*, 340, *343–345*, *347–349*, *351–355*, *357*, *359–361*, *365*, *367–369*, *371*, *372*, *374*, *377*, *380*, *381*, *383–385*, *387–392*, *394–399*, *401*, *403–405*, *407*, *408*

xmuTwinSuper_Continuous(), *343*

xmuTwinUpgradeMeansToCovariateModel, *45*, *134*, *173*, *216*, *223*, *239*, *240*,

263, *317, 318*, *327*, *329, 330*,
332–339, *341*, 342, *344, 345*,
347–349, *351–355*, *357*, *359–361*,
365, *367–369*, *371, 372*, *374*, *377*,
380, 381, *383–385*, *387–392*,
394–399, *401*, *403–405*, *407, 408*