Package 'ctsemOMX'

July 7, 2020

Type Package

Title Continuous Time SEM - 'OpenMx' Based Functions

Version 1.0.2

Date 2020-7-7

Description Original 'ctsem' (continuous time structural equation modelling) functionality, based on the 'OpenMx' software, as described in Driver, Oud, Voelkle (2017) <doi:10.18637/jss.v077.i05>, with updated details in vignette. Combines stochastic differential equations representing latent processes with structural equation measurement models. These functions were split off from the main package of 'ctsem', as the main package uses the 'rstan' package as a backend now -- offering estimation options from max likelihood to Bayesian. There are nevertheless use cases for the wide format SEM style approach as offered here, particularly when there are no individual differences in observation timing and the number of individuals is large. For the main 'ctsem' package, see <https://cran.r-project.org/package=ctsem>.

License GPL-3

Depends R (>= 3.5.0), ctsem (>= 3.3.2), OpenMx (>= 2.9.0)

URL https://github.com/cdriveraus/ctsemOMX

Imports graphics, grDevices, Matrix, methods, plyr, stats, utils

Encoding UTF-8

LazyData true

ByteCompile true

Suggests knitr, testthat

VignetteBuilder knitr

RoxygenNote 7.1.1

NeedsCompilation no

Author Charles Driver [aut, cre, cph],
Manuel Voelkle [aut, cph],
Han Oud [aut, cph]

Maintainer Charles Driver <driver@mpib-berlin.mpg.de>

Repository CRAN

Date/Publication 2020-07-07 15:40:02 UTC

R topics documented:

`AnomAuth` *AnomAuth*

Description

A dataset containing panel data assessments of individuals Anomia and Authoritarianism.

Format

data frame with 2722 rows, 14 columns. Column Y1 represents anomia, Y2 Authoritarianism, dTx the time interval for measurement occasion x.

Source

See http://psycnet.apa.org/journals/met/17/2/176/ for details.

ctCI *ctCI Computes confidence intervals on specified parameters / matrices for already fitted ctsem fit object.*

Description

ctCI Computes confidence intervals on specified parameters / matrices for already fitted ctsem fit object.

Usage

```
ctCI(ctfitobj, confidenceintervals, optimizer = "NPSOL", verbose = 0)
```

Arguments

ctfitobj	Already fit ctsem fit object (class: ctsemFit) to estimate confidence intervals for.
confidenceintervals	character vector of matrices and or parameters for which to estimate 95% confidence intervals for.
optimizer	character vector. Defaults to NPSOL (recommended), but other optimizers available within OpenMx (e.g. 'CSOLNP') may be specified.
verbose	Integer between 0 and 3 reflecting amount of output while calculating.

Details

Confidence intervals typically estimate more reliably using the proprietary NPSOL optimizer available within OpenMx only when installing directly from OpenMx website. Use command " source('http://openmx.psyc.virgini " to install OpenMx with NPSOL. If estimating for a multigroup model, specify confidence intervals as normal, e.g. `confidenceintervals = c('DRIFT','diffusion_Y1_Y1')` . The necessary group prefixes are added internally.

Value

ctfitobj, with confidence intervals included.

Examples

```
## Examples set to 'donttest' because they take longer than 5s.

data("ctExample3")
model <- ctModel(n.latent = 1, n.manifest = 3, Tpoints = 100,
 LAMBDA = matrix(c(1, "lambda2", "lambda3"), nrow = 3, ncol = 1),
 MANIFESTMEANS = matrix(c(0, "manifestmean2", "manifestmean3"), nrow = 3,
   ncol = 1))
fit <- ctFit(dat = ctExample3, ctmodelobj = model, objective = "Kalman",
 stationary = c("T0VAR"))

fit <- ctCI(fit, confidenceintervals = 'DRIFT')
```

```
summary(fit)$omxsummary$CI
```

`ctCompareExpected` *ctCompareExpected Compares model implied to observed means and covariances for panel data fit with ctsem.*

Description

ctCompareExpected Compares model implied to observed means and covariances for panel data fit with ctsem.

Usage

```
ctCompareExpected(
  fitobj,
  cov = TRUE,
  outputmatrices = FALSE,
  pause = TRUE,
  varlist = "all",
  ylim = c(-1, 1),
  ...
)
```

Arguments

`fitobj`	Fitted model object from OpenMx or ctsem.
`cov`	Logical. If TRUE, show covariance plots, if FALSE show correlations.
`outputmatrices`	if TRUE, output expected, observed, and residual correlation matrices as well as plots.
`pause`	if TRUE (default) output plots interactively, one at a time. If FALSE, output without stopping.
`varlist`	if "all" include all variables in dataset. Otherwise, specify numeric vector of variables to include.
`ylim`	vector of min and max Y axis limits for plot.
`...`	additional arguments passed to plot.

ctExample1 *ctExample1*

Description

Simulated example dataset for the ctsem package

Format

100 by 17 matrix containing containing ctsem wide format data. 6 measurement occasions of leisure time and happiness and 5 measurement intervals for each of 100 individuals.

ctExample1TIpred *ctExample1TIpred*

Description

Simulated example dataset for the ctsem package

Format

100 by 18 matrix containing containing ctsem wide format data. 6 measurement occasions of leisure time and happiness, 1 measurement of number of friends, and 5 measurement intervals for each of 100 individuals.

ctExample2 *ctExample2*

Description

Simulated example dataset for the ctsem package

Format

100 by 18 matrix containing containing ctsem wide format data. 8 measurement occasions of leisure time and happiness, 7 measurement occasions of a money intervention dummy, and 7 measurement intervals for each of 50 individuals.

ctExample2level *ctExample2level*

Description

Simulated example dataset for the ctsem package

Format

100 by 18 matrix containing ctsem wide format data. 8 measurement occasions of leisure time and happiness, 7 measurement occasions of a money intervention dummy, and 7 measurement intervals for each of 50 individuals.

ctExample3 *ctExample3*

Description

Simulated example dataset for the ctsem package

Format

1 by 399 matrix containing containing ctsem wide format data. 100 observations of variables Y1 and Y2 and 199 measurement intervals, for 1 subject.

ctExample4 *ctExample4*

Description

Simulated example dataset for the ctsem package

Format

20 by 79 matrix containing 20 observations of variables Y1, Y2, Y3, and 19 measurement intervals dTx, for each of 20 individuals.

ctFit *Fit a ctsem object*

Description

This function fits continuous time SEM models specified via ctModel to a dataset containing one or more subjects.

Usage

```
ctFit(
  dat,
  ctmodelobj,
  dataform = "auto",
  objective = "auto",
  stationary = c("T0TRAITEFFECT", "T0TIPREDEFFECT"),
  optimizer = "CSOLNP",
  retryattempts = 5,
  iterationSummary = FALSE,
  carefulFit = TRUE,
  carefulFitWeight = 100,
  showInits = FALSE,
  asymptotes = FALSE,
  meanIntervals = FALSE,
  crossEffectNegStarts = TRUE,
  fit = TRUE,
  nofit = FALSE,
  discreteTime = FALSE,
  verbose = 0,
  useOptimizer = TRUE,
  omxStartValues = NULL,
  transformedParams = TRUE,
  datawide = NA
)
```

Arguments

dat	the data you wish to fit a ctsem model to, in either wide format (one individual per row), or long format (one time point of one individual per row). See details.
ctmodelobj	the ctsem model object you wish to use, specified via the ctModel function.
dataform	either "wide" or "long" depending on which input format you wish to use for the data. See details and or vignette.
objective	'auto' selects either 'Kalman', if fitting to single subject data, or 'mxRAM' for multiple subjects. For single subject data, 'Kalman' uses the mxExpectationStateSpace function from OpenMx to implement the Kalman filter. For more than one subject, 'mxRAM' specifies a wide format SEM with a row of data per subject.

'cov' may be specified, in which case the 'meanIntervals' argument is set to TRUE, and the covariance matrix of the supplied data is calculated and fit instead of the raw data. This is much faster but only a rough approximation, unless there are no individual differences in time interval and no missing data. 'Kalman' may be specified for multiple subjects, however as no trait matrices are used by the Kalman filter one must consider how average level differences between subjects are accounted for. See `ctMultigroupFit` for the possibility to apply the Kalman filter over multiple subjects)

`stationary` Character vector of T0 matrix names in which to constrain any free parameters to stationarity. Defaults to `c('T0TRAITEFFECT','T0TIPREDEFFECT')`, constraining only between person effects to stationarity. Use `NULL` for no constraints, or 'all' to constrain all T0 matrices.

`optimizer` character string, defaults to the open-source 'CSOLNP' optimizer that is distributed in all versions of OpenMx. However, 'NPSOL' may sometimes perform better for these problems, though requires that you have installed OpenMx via the OpenMx web site, by running: `source('http://openmx.psyc.virginia.edu/getOpenMx.R')`

`retryattempts` Number of times to retry the start value randomisation and fit procedure, if non-convergance or uncertain fits occur.

`iterationSummary`
if TRUE, outputs limited fit details after every fit attempt.

`carefulFit` if TRUE, first fits the specified model with a penalised likelihood function to force MANIFESTVAR, DRIFT, TRAITVAR, MANIFESTTRAITVAR parameters to remain close to 0, then fits the specified model normally, using these estimates as starting values. Can help to ensure optimization begins at sensible, non-exteme values, though results in any user specified start values being ignored for the final fit (though they are still used for initial fit).

`carefulFitWeight`
Positive numeric. Sets the weight for the penalisation (or prior) applied by the carefulFit algorithm. Generally unnecessary to adjust, may be helpful to try a selection of values (perhaps between 0 and 1000) when optimization is problematic.

`showInits` if TRUE, prints the list of starting values for free parameters. These are the 'raw' values used by OpenMx, and reflect the log (var / cov matrices) or -log(DRIFT matrices) transformations used in ctsem. These are saved in the fit object under `fitobject$omxStartValues`.

`asymptotes` when TRUE, optimizes over asymptotic parameter matrices instead of continuous time parameter matrices. Can be faster for optimization and in some cases makes reliable convergance easier. Will result in equivalent models when continuous time input matrices (DRIFT, DIFFUSION, CINT) are free, but fixing the values of any such matrices will result in large differences - a value of 0 in a cell of the normal continuous time DIFFUSION matrix does not necessarily result in a value of 0 for the asymptotic DIFFUSION matrix, for instance.

`meanIntervals` Use average time intervals for each column for calculation (both faster and inaccurate to the extent that intervals vary across individuals).

`crossEffectNegStarts`
Logical. If TRUE (default) free DRIFT matrix cross effect parameters have starting values set to small negative values (e.g. -.05), if FALSE, the start values

are 0. The TRUE setting is useful for easy initialisation of higher order models, while the FALSE setting is useful when one has already estimated a model without cross effects, and wishes to begin optimization from those values by using the omxStartValues switch. are re-transformed into regular continuous time parameter matrices, and may be interpreted as normal.

`fit` if FALSE, output only openmx model without fitting

`nofit` Deprecated. If TRUE, output only openmx model without fitting

`discreteTime` Estimate a discrete time model - ignores timing information, parameter estimates will correspond to those of classical vector autoregression models, OpenMx fit object will be directly output, thus ctsem summary and plot functionality will be unavailable. Time dependent predictor type also becomes irrelevant.

`verbose` Integer between 0 and 3. Sets mxComputeGradientDescent messaging level, defaults to 0.

`useOptimizer` Logical. Defaults to TRUE. Passes argument to `mxRun`, useful for using custom optimizers or fitting to specified parameters.

`omxStartValues` A named vector containing the raw (potentially log transformed) OpenMx starting values for free parameters, as captured by OpenMx function `omxGetParameters(ctmodelobj$mxobj`. These values will take precedence over any starting values already specified using ctModel.

`transformedParams`

Logical indicating whether or not to log transform certain parameters internally to allow unconstrained estimation over entire 'sensible' range for parameters. When TRUE (default) raw OpenMx parameters (only reported if `verbose=TRUE` argument used for summary function) will reflect these transformations and may be harder to interpret, but summary matrices are reported as normal.

`datawide` included for compatibility with scripts written for earlier versions of ctsem. Do not use this argument, instead use the dat argument, and the dataform argument now specifies whether the data is in wide or long format.

Details

For full discussion of how to structure the data and use this function, see the vignette using: `vignette('ctsem')`, or the data examples `data("longexample")` ; `longexample` for long and `data("datastructure")` ; `datastructure` for wide. If using long format, the subject id column must be numeric and grouped by ascending time within subject, and named 'id'. The time column must also be numeric, and representing absolute time (e.g., since beginning of study, *not* time intervals), and called 'time'. Models are specified using the `ctModel` function. For help regarding the summary function, see `summary.ctsemFit`, and for the plot function, `plot.ctsemFit`. Multigroup models may be specified using `ctMultigroupFit`. Confidence intervals for any matrices and or parameters may be estimated using `ctCI`. Difficulties during estimation can sometimes be alleviated using `ctRefineTo` instead of `ctFit` – this uses a multistep fit procedure.

Examples

```
## Examples set to 'donttest' because they take longer than 5s.

mfrowOld<-par()$mfrow
```

```
par(mfrow=c(2, 3))

### example from Driver, Oud, Voelkle (2017),
### simulated happiness and leisure time with unobserved heterogeneity.
data(ctExample1)
traitmodel <- ctModel(n.manifest=2, n.latent=2, Tpoints=6, LAMBDA=diag(2),
  manifestNames=c('LeisureTime', 'Happiness'),
  latentNames=c('LeisureTime', 'Happiness'), TRAITVAR="auto")
traitfit <- ctFit(dat=ctExample1, ctmodelobj=traitmodel)
summary(traitfit)
plot(traitfit, wait=FALSE)


###Example from Voelkle, Oud, Davidov, and Schmidt (2012) - anomia and authoritarianism.
data(AnomAuth)
AnomAuthmodel <- ctModel(LAMBDA = matrix(c(1, 0, 0, 1), nrow = 2, ncol = 2),
Tpoints = 5, n.latent = 2, n.manifest = 2, MANIFESTVAR=diag(0, 2), TRAITVAR = NULL)
AnomAuthfit <- ctFit(AnomAuth, AnomAuthmodel)
summary(AnomAuthfit)


### Single subject time series - using Kalman filter (OpenMx statespace expectation)
data('ctExample3')
model <- ctModel(n.latent = 1, n.manifest = 3, Tpoints = 100,
  LAMBDA = matrix(c(1, 'lambda2', 'lambda3'), nrow = 3, ncol = 1),
  CINT= matrix('cint'),
  MANIFESTMEANS = matrix(c(0, 'manifestmean2', 'manifestmean3'), nrow = 3,
    ncol = 1))
fit <- ctFit(dat = ctExample3, ctmodelobj = model, objective = 'Kalman',
  stationary = c('T0VAR'))


###Oscillating model from Voelkle & Oud (2013).
data("Oscillating")

inits <- c(-39, -.3, 1.01, 10.01, .1, 10.01, 0.05, .9, 0)
names(inits) <- c("crosseffect","autoeffect", "diffusion",
  "T0var11", "T0var21", "T0var22","m1", "m2", 'manifestmean')

oscillatingm <- ctModel(n.latent = 2, n.manifest = 1, Tpoints = 11,
  MANIFESTVAR = matrix(c(0), nrow = 1, ncol = 1),
  LAMBDA = matrix(c(1, 0), nrow = 1, ncol = 2),
  T0MEANS = matrix(c('m1', 'm2'), nrow = 2, ncol = 1),
  T0VAR = matrix(c("T0var11", "T0var21", 0, "T0var22"), nrow = 2, ncol = 2),
  DRIFT = matrix(c(0, "crosseffect", 1, "autoeffect"), nrow = 2, ncol = 2),
  CINT = matrix(0, ncol = 1, nrow = 2),
  MANIFESTMEANS = matrix('manifestmean', nrow = 1, ncol = 1),
  DIFFUSION = matrix(c(0, 0, 0, "diffusion"), nrow = 2, ncol = 2),
  startValues=inits)

oscillatingf <- ctFit(Oscillating, oscillatingm, carefulFit = FALSE)
```

ctGenerateFromFit *Generates data according to the model estimated in a ctsemFit object.*

Description

Generates data according to the model estimated in a ctsemFit object.

Usage

```
ctGenerateFromFit(
  fit,
  timestep = "asdata",
  n.subjects = 100,
  timerange = "asdata",
  predictorSubjects = "all",
  ...
)
```

Arguments

fit	object of class ctsemFit as returned from ctFit.
timestep	positive numeric value indicating the time interval to use for data generation.
n.subjects	integer. Number of subjects worth of data to generate
timerange	either 'asdata' to calculate range based on data in fit object, or vector of length 2 specifying min and max times for generation.
predictorSubjects	vector of integers, or string 'all', defining which subjects to sample time dependent and independent predictors from.
...	parameters to pass to ctGenerate function, such as wide=FALSE.

Value

matrix of generated data

Examples

```
data(AnomAuth)
AnomAuthmodel <- ctModel(LAMBDA = matrix(c(1, 0, 0, 1), nrow = 2, ncol = 2),
  Tpoints = 5, n.latent = 2, n.manifest = 2, MANIFESTVAR=diag(0, 2))
AnomAuthfit <- ctFit(AnomAuth, AnomAuthmodel)

dwide <- ctGenerateFromFit(AnomAuthfit,timestep=1,n.subjects=5,wide=TRUE)
head(dwide)
```

ctIndplot *ctIndplot*

Description

Convenience function to simply plot individuals trajectories from ctsem wide format data

Usage

```
ctIndplot(
  datawide,
  n.manifest,
  Tpoints,
  n.subjects = "all",
  colourby = "variable",
  vars = "all",
  opacity = 1,
  varnames = NULL,
  xlab = "Time",
  ylab = "Value",
  type = "b",
  start = 0,
  legend = TRUE,
  legendposition = "topright",
  new = TRUE,
  jittersd = 0.05,
  ...
)
```

Arguments

datawide	ctsem wide format data
n.manifest	Number of manifest variables in data structure
Tpoints	Number of discrete time points per case in data structure
n.subjects	Number of subjects to randomly select for plotting, or character vector 'all'.
colourby	set plot colours by "subject" or "variable"
vars	either 'all' or a numeric vector specifying which manifest variables to plot.
opacity	Opacity of plot lines
varnames	vector of variable names for legend (defaults to NULL)
xlab	X axis label.
ylab	Y axis label.
type	character specifying plot type, as per usual base R plot commands. Defaults to 'b', both points and lines.
start	Measurement occasion to start plotting from - defaults to T0.

legend	Logical. Plot a legend?
legendposition	Where to position the legend.
new	logical. If TRUE, creates a new plot, otherwise overlays on current plot.
jittersd	positive numeric indicating standard deviation of noise to add to observed data for plotting purposes.
...	additional plotting parameters.

Examples

```
data(ctExample1)
ctIndplot(ctExample1,n.subjects=1, n.manifest=2,Tpoints=6, colourby='variable')
```

---

ctModelFromFit *Extract a ctsem model structure with parameter values from a ctsem fit object.*

Description

Extract a ctsem model structure with parameter values from a ctsem fit object.

Usage

```
ctModelFromFit(fit)
```

Arguments

fit	object output by ctFit

Value

object of class 'ctsemInit' (as generated by ctModel), which can be used with ctFit and Kalman functions.

Examples

```
data(AnomAuth)
AnomAuthmodel <- ctModel(LAMBDA = matrix(c(1, 0, 0, 1), nrow = 2, ncol = 2),
  Tpoints = 5, n.latent = 2, n.manifest = 2, MANIFESTVAR=diag(0, 2))
AnomAuthfit <- ctFit(AnomAuth, AnomAuthmodel)

fitmodel <- ctModelFromFit(AnomAuthfit)
```

`ctMultigroupFit` *Fits a multiple group continuous time model.*

Description

Fits a single continuous time structural equation models to multiple groups (where each group contains 1 or more subjects), by default, all parameters are free across groups. Can also be used to easily estimate seperate models for each group.

Usage

```
ctMultigroupFit(
  dat,
  groupings,
  ctmodelobj,
  dataform = "wide",
  fixedmodel = NA,
  freemodel = NA,
  carefulFit = TRUE,
  omxStartValues = NULL,
  retryattempts = 5,
  showInits = FALSE,
  ...
)
```

Arguments

`dat`	Wide format data, as used in `ctFit`. See `ctLongToWide` to easily convert long format data.
`groupings`	For wide format: Vector of character labels designating group membership for each row of dat. For long format: Named list of groups, with each list element containing a vector of subject id's for the group. In both cases, group names will be prefixed on relevant parameter estimates in the summary.
`ctmodelobj`	Continuous time model to fit, specified via `ctModel` function.
`dataform`	either "wide" or "long" depending on which input format you wish to use for the data. See details of `ctFit` and or vignette.
`fixedmodel`	Modified version of ctmodelobj, wherein any parameters you wish to keep fixed over groups should be given the value 'groupfixed'. If specified, all other parameters will be free across groups.
`freemodel`	Modified version of ctmodelobj, wherein any parameters you wish to free across groups should be given the label 'groupfree'. If specified, all other parameters will be fixed across groups. If left NULL, the default, all parameters are free across groups.

carefulFit	if TRUE, first fits the specified model with a penalised likelihood function to discourage parameters from boundary conditions, then fits the specified model normally, using these estimates as starting values. Can help / speed optimization, though results in user specified inits being ignored for the final fit.
omxStartValues	A named vector containing the raw (potentially log transformed) OpenMx starting values for free parameters, as captured by OpenMx function omxGetParameters(ctmodelobj$mxobj These values will take precedence over any starting values already specified using ctModel.
retryattempts	Number of fit retries to make.
showInits	Displays start values prior to optimization
...	additional arguments to pass to ctFit.

Details

Additional ctFit parameters may be specified as required. Confidence intervals for any matrices and or parameters may be estimated afer fitting using ctCI.

Value

Returns an OpenMx fit object.

See Also

ctFit and ctModel

Examples

```
#Two group model, all parameters except LAMBDA[3,1] constrained across groups.
data(ctExample4)
basemodel<-ctModel(n.latent=1, n.manifest=3, Tpoints=20,
                   LAMBDA=matrix(c(1, 'lambda2', 'lambda3'), nrow=3, ncol=1),
                   MANIFESTMEANS=matrix(c(0, 'manifestmean2', 'manifestmean3'),
                   nrow=3, ncol=1), TRAITVAR = 'auto')

freemodel<-basemodel
freemodel$LAMBDA[3,1]<-'groupfree'
groups<-paste0('g',rep(1:2, each=10),'_')

multif<-ctMultigroupFit(dat=ctExample4, groupings=groups,
                        ctmodelobj=basemodel, freemodel=freemodel)
summary(multif,group=1)



#fixed model approach
fixedmodel<-basemodel
fixedmodel$LAMBDA[2,1]<-'groupfixed'
groups<-paste0('g',rep(1:2, each=10),'_')
```

```
multif<-ctMultigroupFit(dat=ctExample4, groupings=groups,
                        ctmodelobj=basemodel, fixedmodel=fixedmodel)
summary(multif,group=2)
```

---

ctPlot *ctPlot*

Description

Plots mean trajectories, autoregression, and crossregression plots, for ctsemFit objects. More customizeable than basic plot.ctsemFit function.

Usage

```
ctPlot(
  x,
  plotType,
  xlim,
  resolution = 50,
  impulseIndex = NULL,
  subject = 1,
  typeVector = "auto",
  colVector = "auto",
  ltyVector = "auto",
  ...
)
```

Arguments

x	ctsemFit object as generated by ctFit.
plotType	string. "mean" for expectation independent of any data, "AR" for autoregressions, "CR" for cross regressions, "standardiseCR" for standardised cross regressions (standardised based on estimated within subject variance), "withinVar" for within variance and covariance, "randomImpulse" for expected change in processes given a random fluctuation of +1 for each process (so a mixture of DIFFUSION and DRIFT characteristics), "experimentalImpulse" for expected change in processes given an exogenous input of +1 for each process, provides alternate characterisation of autoregressive and cross regressive plots.
xlim	vector. As per usual for plot(), but xlim may not be negative.
resolution	Numeric. Plot points between each unit of time. Default of 'auto' adapts to xlim and results in 500 points in total.
impulseIndex	Numeric. Only required for impulse plot types, specifies which column of the DRIFT matrix the impulse relates to.

subject	numeric. Specifies the subject (row of data from the mxobj) to plot for factorScores type plot.
typeVector	Vector of plot types to use for plotting.
colVector	vector of colours to use for plotting.
ltyVector	Vector of line types to use for plotting.
...	Other options passed to plot(). ylim is required.

Value

Character vector of labels from the DRIFT matrix in order plotted - useful for legends. Side-effect: plots graphs.

Examples

```
## Examples set to 'donttest' because they take longer than 5s.

### example from Driver, Oud, Voelkle (2016),
### simulated happiness and leisure time with unobserved heterogeneity.

data(ctExample1)
traitmodel <- ctModel(n.manifest=2, n.latent=2, Tpoints=6, LAMBDA=diag(2),
  manifestNames=c('LeisureTime', 'Happiness'),
  latentNames=c('LeisureTime', 'Happiness'), TRAITVAR="auto")
traitfit <- ctFit(dat=ctExample1, ctmodelobj=traitmodel)
ctPlot(traitfit, plotType='CR', xlim=c(0,5),ylim=c(-1,1))
```

---

ctPostPredict *Posterior predictive type check for ctsemFit.*

Description

Samples data according to the ctsemFit object, computes quantiles over time based on model fit, plots these against original data.

Usage

```
ctPostPredict(
  fit,
  timestep = 0.1,
  n.subjects = 100,
  probs = c(0.025, 0.5, 0.975),
  plot = TRUE,
  ctPlotArrayArgs = list(grid = FALSE, legend = FALSE),
 indPlotArgs = list(colourby = "subject", lwd = 2, new = FALSE, type = "p", opacity =
    0.3),
  mfrow = "auto"
)
```

Arguments

fit	object of class ctsemFit as returned from ctFit
timestep	positive value denoting the time interval to use for sampling.
n.subjects	Number of subjects worth of data to sample.
probs	Vector of values between 0 and 1 denoting quantiles to generate. For plotting, vector should be of length 3 and values should be rising.
plot	Whether to plot or return the generated data.
ctPlotArrayArgs	additional arguments to pass to ctPlotArray function, for plotting generated distributions.
indPlotArgs	list of parameters to pass to ctIndplot, for plotting original data. Only used if plot=TRUE.
mfrow	2 dimensional integer vector defining number of rows and columns of plots, as per the mfrow argument to par. 'auto' determines automatically, to a maximum of 4 by 4, while NULL uses the current system setting.

Value

Either nothing (if plot=TRUE) or an array containing generated data over quantiles.

Examples

```
data("AnomAuth")
AnomAuthmodel <- ctModel(LAMBDA = matrix(c(1, 0, 0, 1), nrow = 2, ncol = 2),
  Tpoints = 5, n.latent = 2, n.manifest = 2, MANIFESTVAR=diag(0, 2), TRAITVAR = 'auto')
AnomAuthFit <- ctFit(AnomAuth, AnomAuthmodel)
ctPostPredict(AnomAuthFit,timestep=.5,n.subjects=100)
```

---

ctRefineTo *ctRefineTo*

Description

Fits a ctsem m in a stepwise fashion to help with difficult optimization.

Usage

```
ctRefineTo(datawide, ctmodelobj, modfunc = NULL, ...)
```

Arguments

datawide	Data in ctsem wide format
ctmodelobj	A continuous time m specified via the ctModel function.
modfunc	function to run prior to each optimization step, that takes ctsem fit object, modifies it as desired, and returns the fit object.
...	additional parameters to pass to ctFit.

Details

This function fits a sequence of ctsem models increasing in complexity, starting with a m involving fixed and relatively strong auto effects, no cross effects, no predictors, and no off-diagonal covariances. For many models this can improve the speed and robustness of fitting

Value

Returns a fitted ctsem object in the same manner as ctFit.

---

ctsemOMX *ctsemOMX*

Description

ctsem is an R package for continuous time structural equation modelling of panel (N > 1) and time series (N = 1) data, using either a frequentist or Bayesian approach, or middle ground forms like maximum a posteriori. This ctsemOMX addition includes the original OpenMx based functions which have been split off from the main package.

Details

The general workflow begins by specifying a model using the ctModel function, in which the type of model is also specified. Then the model is fit to data using either ctFit if the original 'omx' (OpenMx, SEM, max likelihood) model is specified. The omx forms are no longer in development and for most purposes, the newer stan based forms (contained in the base ctsem package) are more robust and flexible. For citation info, please run `citation('ctsem')` .

References

https://www.jstatsoft.org/article/view/v077i05

Driver, C. C., & Voelkle, M. C. (2018). Hierarchical Bayesian continuous time dynamic modeling. Psychological Methods. Advance online publication.http://dx.doi.org/10.1037/met0000168

`datastructure` *datastructure*

Description

Simulated example dataset for the ctsem package

Format

2 by 15 matrix containing containing ctsem wide format data. 3 measurement occasions of manifest variables Y1 and Y2, 2 measurement occasions of time dependent predictor TD1, 2 measurement intervals dTx, and 2 time independent predictors TI1 and TI2, for 2 individuals.

`longexample` *longexample*

Description

Simulated example dataset for the ctsem package

Format

7 by 8 matrix containing ctsem long format data, for two subjects, with three manifest variables Y1, Y2, Y3, one time dependent predictor TD1, two time independent predictors TI1 and TI2, and absolute timing information Time.

`Oscillating` *Oscillating*

Description

Simulated example dataset for the ctsem package.

Format

200 by 21 matrix containing containing ctsem wide format data. 11 measurement occasions and 10 measurement intervals for each of 200 individuals

Source

See http://onlinelibrary.wiley.com/doi/10.1111/j.2044-8317.2012.02043.x/abstract

plot.ctKalman *Plots Kalman filter output from ctKalman.*

Description

Plots Kalman filter output from ctKalman.

Usage

```
## S3 method for class 'ctKalman'
plot(
  x,
  subjects = 1,
  kalmanvec = c("y", "yprior"),
  errorvec = "auto",
  errormultiply = 1.96,
  ltyvec = "auto",
  colvec = "auto",
  lwdvec = "auto",
  subsetindices = NULL,
  pchvec = "auto",
  typevec = "auto",
  grid = FALSE,
  add = FALSE,
  plotcontrol = list(ylab = "Value", xlab = "Time", xaxs = "i", lwd = 2, mgp = c(2,
    0.8, 0)),
  polygoncontrol = list(steps = 20),
  polygonalpha = 0.1,
  legend = TRUE,
  legendcontrol = list(x = "topright", bg = "white", cex = 0.7),
  ...
)
```

Arguments

x	Output from ctKalman. In general it is easier to call ctKalman directly with the plot=TRUE argument, which calls this function.
subjects	vector of integers denoting which subjects (from 1 to N) to plot predictions for.
kalmanvec	string vector of names of any elements of the output you wish to plot, the defaults of 'y' and 'yprior' plot the original data, 'y', and the prior from the Kalman filter for y. Replacing 'y' by 'eta' will plot latent variables instead (though 'eta' alone does not exist) and replacing 'prior' with 'upd' or 'smooth' respectively plotting updated (conditional on all data up to current time point) or smoothed (conditional on all data) estimates.
errorvec	vector of names of covariance elements to use for uncertainty indication around the kalmanvec items. 'auto' uses the latent covariance when plotting latent

	states, and total covariance when plotting expectations of observed states. Use NA to skip uncertainty plotting.
errormultiply	Numeric denoting the multiplication factor of the std deviation of errorvec objects. Defaults to 1.96, for 95% intervals.
ltyvec	vector of line types, varying over dimensions of the kalmanvec object.
colvec	color vector, varying either over subject if multiple subjects, or otherwise over the dimensions of the kalmanvec object.
lwdvec	vector of line widths, varying over the kalmanvec objects.
subsetindices	Either NULL, or vector of integers to use for subsetting the (columns) of kalmanvec objects.
pchvec	vector of symbol types, varying over the dimensions of the kalmanvec object.
typevec	vector of plot types, varying over the kalmanvec objects. 'auto' plots lines for any 'prior', 'upd', or 'smooth' objects, and points otherwise.
grid	Logical. Plot a grid?
add	Logical. Create a new plot or update existing plot?
plotcontrol	List of graphical arguments (see par), though lty,col,lwd,x,y, will all be ignored.
polygoncontrol	List of arguments to the ctPoly function for filling the uncertainty region.
polygonalpha	Numeric for the opacity of the uncertainty region.
legend	Logical, whether to include a legend if plotting.
legendcontrol	List of arguments to the legend function.
...	not used.

Value

Generates plots.

Examples

```
data(AnomAuth)
AnomAuthmodel <- ctModel(LAMBDA = matrix(c(1, 0, 0, 1), nrow = 2, ncol = 2),
  Tpoints = 5, n.latent = 2, n.manifest = 2)
AnomAuthfit <- ctFit(AnomAuth, AnomAuthmodel)
ctKalman(AnomAuthfit,subjects=1,plot=TRUE)
```

plot.ctsemFit *Plotting function for object class ctsemFit*

Description

Ouputs mean trajectories, autoregression, and crossregression plots. For more customization possibilities, see ctPlot.

Usage

```
## S3 method for class 'ctsemFit'
plot(
  x,
  resolution = 50,
  wait = TRUE,
  max.time = "auto",
  mean = TRUE,
  withinVariance = TRUE,
  AR = TRUE,
  CR = TRUE,
  standardiseCR = FALSE,
  randomImpulse = FALSE,
  experimentalImpulse = FALSE,
  xlab = "Time",
  meansylim = "auto",
  ARylim = "auto",
  CRylim = "auto",
  ylab = "Value",
  ...
)
```

Arguments

x	ctsemFit object as generated by ctFit.
resolution	Numeric. Plot points between each unit of time. Default of 'auto' adapts to max.time and results in 500 in total.
wait	If true, user is prompted to continue before plotting next graph. If false, graphs are plotted one after another without waiting.
max.time	Time scale on which to plot parameters. If auto, parameters are plotted for full range of observed variables.
mean	if TRUE, plot of means from 0 to max.time included in output.
withinVariance	if TRUE, plot within subject variance / covariance.
AR	if TRUE, plot of autoregressive values from 0 to max.time included in output.
CR	if TRUE, plot of cross regressive values from 0 to max.time included in output.

standardiseCR	if TRUE , cross regression values are standardised based on estimated within subject variance.
randomImpulse	if TRUE (default), plots expected change in processes given a random fluctuation of +1 for each process – plot is then a mixture of DIFFUSION and DRIFT characteristics.
experimentalImpulse	if TRUE (default), plots expected change in processes given an exogenous input of +1 for each process – alternate characterisation of autoregressive and cross regressive plots.
xlab	X axis label.
meansylim	Vector of min and max limits for mean trajectory plot. 'auto' calculates automatically.
ARylim	Vector of min and max limits for autoregression plot. 'auto' is c(0,1), and expands if necessary.
CRylim	Vector of min and max limits for cross regression plot. 'auto' is c(-1,1), and expands if necessary.
ylab	Y axis label.
...	Other options passed to plot().

Value

Nothing. Side-effect: plots graphs.

Examples

```
## Examples set to 'donttest' because they take longer than 5s.

### example from Driver, Oud, Voelkle (2015),
### simulated happiness and leisure time with unobserved heterogeneity.

data(ctExample1)
traitmodel <- ctModel(n.manifest=2, n.latent=2, Tpoints=6, LAMBDA=diag(2),
  manifestNames=c('LeisureTime', 'Happiness'),
  latentNames=c('LeisureTime', 'Happiness'), TRAITVAR="auto")
traitfit <- ctFit(dat=ctExample1, ctmodelobj=traitmodel)
plot(traitfit, wait=FALSE)
```

---

plot.ctsemMultigroupFit

Plot function for ctsemMultigroupFit object

Description

Plots ctMultigroupFit objects.

Usage

```
## S3 method for class 'ctsemMultigroupFit'
plot(x, group = "show chooser", ...)
```

Arguments

x	ctsemMultigroupFit object as generated by ctMultigroupFit
group	character string of subgroup to plot. Default of 'show chooser' displays list and lets you select.
...	additional parameters to pass to plot.ctsemFit function.

Value

Nothing. Side-effect: plots graphs.

---

summary.ctsemFit *Summary function for ctsemFit object*

Description

Provides summary details for ctsemFit objects.

Usage

```
## S3 method for class 'ctsemFit'
summary(object, ridging = FALSE, timeInterval = 1, verbose = FALSE, ...)
```

Arguments

object	ctsemFit object as generated by ctFit.
ridging	if TRUE, adds a small amount of variance to diagonals when calculating standardised (correlation) matrices, should only be used if standardised matrices return NAN.
timeInterval	positive numeric value specifying time interval to use for discrete parameter matrices, defaults to 1.
verbose	Logical. If TRUE, displays the raw, internally transformed (when fitting with default arguments) OpenMx parameters and corresponding standard errors, as well as additional summary matrices. Parameter transforms are described in the vignette, vignette('ctsem'). Additional summary matrices include: 'discrete' matrices – matrices representing the effect for the given time interval (default of 1); 'asymptotic' matrices – represents the effect as time interval approaches infinity (therefore asymCINT describes mean level of processes at the asymptote, asymDIFFUSION describes total within- subject variance at the asymptote, etc); 'standardised' matrices – transforms covariance matrices to correlation matrices, and transforms discreteDRIFT based on DIFFUSION, to give effect sizes.
...	additional parameters to pass.

Details

Important: Although ctModel takes cholesky decomposed variance-covariance matrices as input, the summary function displays the full variance-covariance matrices. These can be cholesky decomposed for comparison purposes using `t(chol(summary(ctfitobject)$covariancematrix))`. Standard errors are displayed in the $ctparameters section, however if ctFit was used with transformedParams=TRUE (the default, and recommended) covariance matrix standard errors will have been approximated using the delta method. For inferential purposes, maximum likelihood confidence intervals may be estimated using the ctCI function.

Value

Summary of ctsemFit object

Examples

```
## Examples set to 'donttest' because they take longer than 5s.


### example from Driver, Oud, Voelkle (2015),
### simulated happiness and leisure time with unobserved heterogeneity.
data(ctExample1)
traitmodel <- ctModel(n.manifest=2, n.latent=2, Tpoints=6, LAMBDA=diag(2),
  manifestNames=c('LeisureTime', 'Happiness'),
  latentNames=c('LeisureTime', 'Happiness'), TRAITVAR="auto")
traitfit <- ctFit(dat=ctExample1, ctmodelobj=traitmodel)
summary(traitfit,timeInterval=1)
```

---

summary.ctsemMultigroupFit

Summary function for ctsemMultigroupFit object

Description

Provides summary details for objects fitted with ctMultigroupFit.

Usage

```
## S3 method for class 'ctsemMultigroupFit'
summary(object, group = "show chooser", ...)
```

Arguments

object	ctsemMultigroupFit object as generated by ctMultigroupFit
group	character string of subgroup to display summary parameters for. Default of 'show chooser' displays list and lets you select.
...	additional parameters to pass to summary.ctsemFit.

Value

Summary of ctsemMultigroupFit object

Index