Package 'gwsem'

June 14, 2020

Type Package

Title Genome-Wide Structural Equation Modeling

Version 2.0.5

Description Melds genome-wide association tests with structural equation modeling (SEM) using 'OpenMx'. This package contains low-level C/C++ code to rapidly read genetic data encoded in U.K. Biobank or 'plink' formats. Prebuilt modeling options include one and two factor models. Alternately, analyses may utilize arbitrary, user-provided SEMs. See Verhulst, Maes, & Neale (2017) <doi:10.1007/s10519-017-9842-6> for details. An updated manuscript is in preparation.

License GPL (>= 3)

URL https://github.com/jpritikin/gwsem

BugReports https://github.com/jpritikin/gwsem/issues

Depends R (>= 3.5),
OpenMx (>= 2.15.5)

Imports data.table,
methods,
qqman,
Rcpp,
lifecycle

Suggests testthat (>= 2.1.0),
MASS,
covr,
knitr,
rmarkdown,
curl

LinkingTo BH (>= 1.69.0-1),
Rcpp

VignetteBuilder knitr

RdMacros lifecycle

Encoding UTF-8

Language en-US

LazyData true

NeedsCompilation yes

RoxygenNote 7.1.0

SystemRequirements GNU make

R topics documented:

`gwsem-package` *Genome-wide Structural Equation Modeling*

Description

Psychometricians have long known that little information can be gleened from a single item. Hence, there is a long-standing tradition in education, psychology, and many other fields to use more than one item to measure a latent trait. For example, a math test will always consist of more than one problem (or the single problem with consist of many parts).

Phenotypic data gathered at the same time as genetic data sometimes contains multiple items that measure different aspects of the same latent construct. However, due to the astronomic number of single nucleotide polymorphisms (SNPs) to test, fast analysis methods are generally preferred with much of the prior research employing regression. Regression is fast, but can only predict a single item at time. Hence, associations with rich phenotypic data cannot be properly investigated.

gwsem contains low-level C/C++ code to permit OpenMx to rapidly read genetic data encoded in U.K. Biobank or plink formats. The association between SNPs and a factor model can be explored.

buildItem *Build a model suitable for a single item genome-wide association study*

Description

Maturing

Usage

```
buildItem(
  phenoData,
  depVar,
  covariates = NULL,
  ...,
  fitfun = c("WLS", "ML"),
  minMAF = 0.01,
  gxe = NULL,
  exogenous = NA
)
```

Arguments

phenoData	the file pathway for the phenotypic data (e.g. "myData.txt" or "phenotype/myData.txt"). This data file can include more variables than those included in the analysis, but GW-SEM will only use the items/covariates that are specified. (The dangers of very large dataset is that they can take a long time to load and can take up space in the R environment. This should not affect processing speed for the GWAS analysis, but can create headaches for pre-processing).
depVar	the name of items to predict
covariates	a character list of covariates that the latent variable will be regressed upon. The default vaule is NULL, but this is a silly value as typically analysts will include e.g. age, sex, and ancestry principle components in the analysis.
...	Not used. Forces remaining arguments to be specified by name.
fitfun	The fitfun argument specifies which fit function should be used in evaluating the GWAS model. Users may choose between the relatively rapid "WLS", or the slower but asymptotically optimal "ML". In many cases the the differences between the fit functions is trivial and the faster "WLS" option should be used, but in some situations the differences can be quite meaningful (such as when data are Missing at Random - MAR).
minMAF	The minimum valid minor allele frequency (MAF). Large differences between the variances of two variables in the same model can cause optimization failures that invalidate the model. Further, very small minor allele frequencies are more affected by outliers or influential observations. Accordingly, users can specify the minimum allowable MAF. The default value is MAF > .01. Users may also

wish to filter out small MAF alleles from their genotype files in other software programs, such as PLINK.

`gxe` The observed variable name that will be used to moderate (interact with) the effect of the SNP on the phenotypes. For example, you may want to moderate the SNP regression by sex. In this situation, you would specify gxe = "sex" and add "snp_sex" to the list of covariates.

`exogenous` This argument specifies how you would like to integrate the covariates into the analysis. If exogenous = T, each items will be directly regressed on each covariate. If exogenous = F, the latent factor(s) will be directly regressed on each covariate. Setting exogenous = T does not assume that the items are related to the covariates proportional to their factor loadings (which is probably preferable in most cases).

Details

You can pass the result of this function to GWAS to run a GWAS.

Ordinal indicator thresholds are setup by setupThresholds. You can plot the model using omxGraphviz.

Value

A MxModel

WLS Technical Note

When the `depVar` item is/are continuous, covariates are endogenous (the default), and the fit function is `WLS` then the `cumulants` method is used to create observed summary statistics (see mxFitFunctionWLS). In other cases, the `marginals` method is used. The `cumulants` method is more accurate than `marginals`. The difference in accuracy becomes vivid when comparing estimates against the `ML` fit function.

See Also

Other model builder: `buildOneFacRes()`, `buildOneFac()`, `buildTwoFac()`

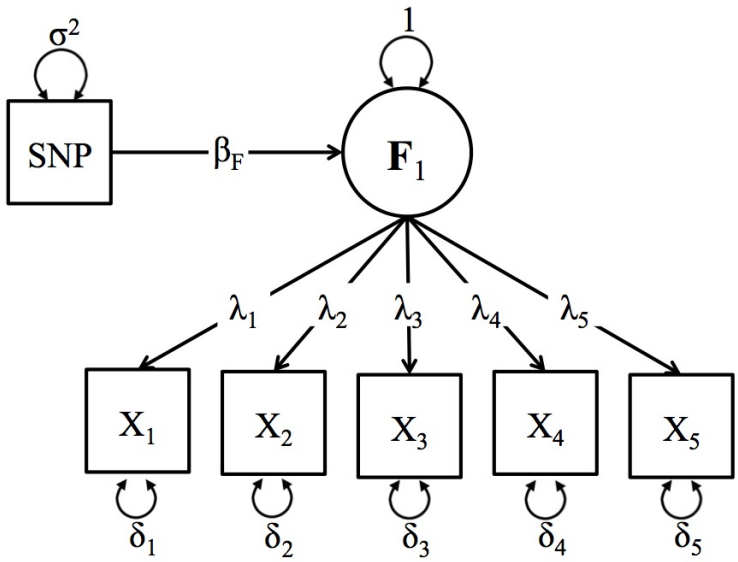
Examples

```
pheno <- data.frame(anxiety=cut(rnorm(500), c(-Inf, -.5, .5, Inf),
                    ordered_result = TRUE))
m1 <- buildItem(pheno, 'anxiety')
```

`buildOneFac` *Build a model suitable for a single factor genome-wide association study*

Description

Maturing The `buildOneFac` function is used to specify a single factor latent variable model where the latent variable is predicted by a genomic variant such as a single nucleotide polymorphism, as well as range of covariates.



Usage

```
buildOneFac(
  phenoData,
  itemNames,
  covariates = NULL,
  ...,
  fitfun = c("WLS", "ML"),
  minMAF = 0.01,
  gxe = NULL,
  exogenous = NA
)
```

Arguments

`phenoData`	the file pathway for the phenotypic data (e.g. "myData.txt" or "phenotype/myData.txt"). This data file can include more variables than those included in the analysis, but GW-SEM will only use the items/covariates that are specified. (The dangers of very large dataset is that they can take a long time to load and can take up space

in the R environment. This should not affect processing speed for the GWAS analysis, but can create headaches for pre-processing).

`itemNames` a character list of the names of the items that load onto the latent variable. These names must match variable names in the phenoData file.

`covariates` a character list of covariates that the latent variable will be regressed upon. The default vaule is NULL, but this is a silly value as typically analysts will include e.g. age, sex, and ancestry principle components in the analysis.

`...` Not used. Forces remaining arguments to be specified by name.

`fitfun` The fitfun argument specifies which fit function should be used in evaluating the GWAS model. Users may choose between the relatively rapid "WLS", or the slower but asymptotically optimal "ML". In many cases the the differences between the fit functions is trivial and the faster "WLS" option should be used, but in some situations the differences can be quite meaningful (such as when data are Missing at Random - MAR).

`minMAF` The minimum valid minor allele frequency (MAF). Large differences between the variances of two variables in the same model can cause optimization failures that invalidate the model. Further, very small minor allele frequencies are more affected by outliers or influential observations. Accordingly, users can specify the minimum allowable MAF. The default value is MAF > .01. Users may also wish to filter out small MAF alleles from their genotype files in other software programs, such as PLINK.

`gxe` The observed variable name that will be used to moderate (interact with) the effect of the SNP on the phenotypes. For example, you may want to moderate the SNP regression by sex. In this situation, you would specify gxe = "sex" and add "snp_sex" to the list of covariates.

`exogenous` This argument specifies how you would like to integrate the covariates into the analysis. If exogenous = T, each items will be directly regressed on each covariate. If exogenous = F, the latent factor(s) will be directly regressed on each covariate. Setting exogenous = T does not assume that the items are related to the covariates proportional to their factor loadings (which is probably preferable in most cases).

Details

You can pass the result of this function to GWAS to run a GWAS.

Ordinal indicator thresholds are setup by setupThresholds. You can plot the model using omxGraphviz.

Value

`buildOneFac` returns an MxModel object that can serve as input for the GWAS function.

See Also

Other model builder: `buildItem()`, `buildOneFacRes()`, `buildTwoFac()`

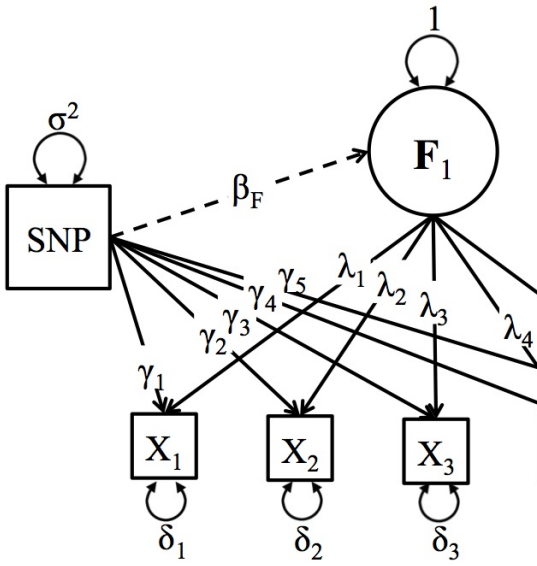
Examples

```
pheno <- list()
for (i in 1:5) pheno[[paste0('i',i)]] <- rnorm(500)
pheno <- as.data.frame(pheno)
buildOneFac(pheno, colnames(pheno))
```

`buildOneFacRes` *Build a model suitable for a single factor residual genome-wide association study*

Description

Maturing The `buildOneFacRes` function is used to specify a single factor latent variable model where a combination of items as well as the latent variable may be predicted by a genomic variant



such as a single nucleotide polymorphism, as well as range of covariates.

Usage

```
buildOneFacRes(
  phenoData,
  itemNames,
  factor = F,
  res = itemNames,
  covariates = NULL,
  ...,
  fitfun = c("WLS", "ML"),
  minMAF = 0.01,
  gxe = NULL,
  exogenous = NA
)
```

Arguments

phenoData	the file pathway for the phenotypic data (e.g. "myData.txt" or "phenotype/myData.txt"). This data file can include more variables than those included in the analysis, but GW-SEM will only use the items/covariates that are specified. (The dangers of very large dataset is that they can take a long time to load and can take up space in the R environment. This should not affect processing speed for the GWAS analysis, but can create headaches for pre-processing).
itemNames	a character list of the names of the items that load onto the latent variable. These names must match variable names in the phenoData file.
factor	A logical expression (FALSE or TRUE) indicating whether to estimate a regression pathway from the SNP to the latent factor (default FALSE).
res	A character vector of phenotypic item names that indicate which specific items the user wishes to regress on the SNP. The default is to regress all of the items on the SNP.
covariates	a character list of covariates that the latent variable will be regressed upon. The default vaule is NULL, but this is a silly value as typically analysts will include e.g. age, sex, and ancestry principle components in the analysis.
...	Not used. Forces remaining arguments to be specified by name.
fitfun	The fitfun argument specifies which fit function should be used in evaluating the GWAS model. Users may choose between the relatively rapid "WLS", or the slower but asymptotically optimal "ML". In many cases the the differences between the fit functions is trivial and the faster "WLS" option should be used, but in some situations the differences can be quite meaningful (such as when data are Missing at Random - MAR).
minMAF	The minimum valid minor allele frequency (MAF). Large differences between the variances of two variables in the same model can cause optimization failures that invalidate the model. Further, very small minor allele frequencies are more affected by outliers or influential observations. Accordingly, users can specify the minimum allowable MAF. The default value is MAF > .01. Users may also wish to filter out small MAF alleles from their genotype files in other software programs, such as PLINK.
gxe	The observed variable name that will be used to moderate (interact with) the effect of the SNP on the phenotypes. For example, you may want to moderate the SNP regression by sex. In this situation, you would specify gxe = "sex" and add "snp_sex" to the list of covariates.
exogenous	This argument specifies how you would like to integrate the covariates into the analysis. If exogenous = T, each items will be directly regressed on each covariate. If exogenous = F, the latent factor(s) will be directly regressed on each covariate. Setting exogenous = T does not assume that the items are related to the covariates proportional to their factor loadings (which is probably preferable in most cases).

Details

Be aware that a latent variable model is not identified if all of the residuals as well as the latent variable are simultaneously predicted by the SNP. Specifically, if users wish to use the SNP to

predict the latent variable, they much choose at least one (and preferably more that one) item to not be predicted by the SNP.

You can pass the result of this function to GWAS to run a GWAS.

Ordinal indicator thresholds are setup by setupThresholds. You can plot the model using omxGraphviz.

Value

`buildOneFacRes` returns an MxModel object that can serve as input for the GWAS function.

See Also

Other model builder: `buildItem()`, `buildOneFac()`, `buildTwoFac()`

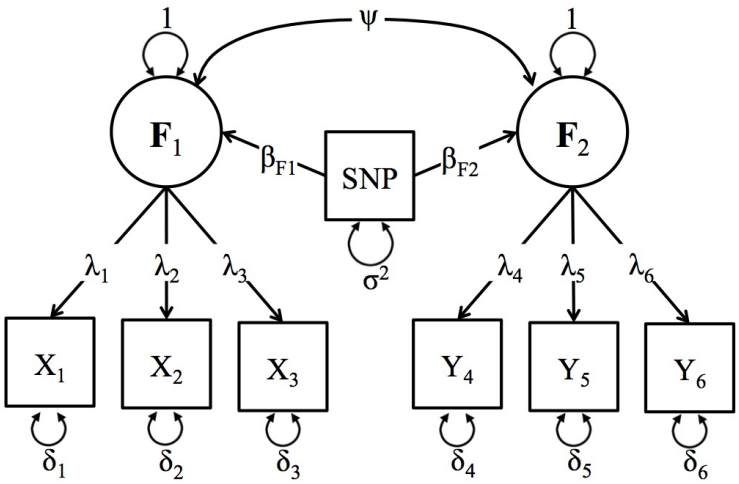
Examples

```
pheno <- list()
for (i in 1:5) pheno[[paste0('i',i)]] <- rnorm(500)
pheno <- as.data.frame(pheno)
buildOneFacRes(pheno, colnames(pheno))
```

`buildTwoFac` *Build a model suitable for a two factor genome-wide association study*

Description

Maturing The buildTwoFac function is used to specify a model with two latent variables where each latent variable is simultaneously predicted by a genomic variant such as a single nucleotide polymorphism, as well as range of covariates. The model allows the latent variables to correlate to accomodate comorbidity between latent traits.



Usage

```
buildTwoFac(
  phenoData,
  F1itemNames,
  F2itemNames,
  covariates = NULL,
  ...,
  fitfun = c("WLS", "ML"),
  minMAF = 0.01,
  gxe = NULL,
  exogenous = NA
)
```

Arguments

phenoData	the file pathway for the phenotypic data (e.g. "myData.txt" or "phenotype/myData.txt"). This data file can include more variables than those included in the analysis, but GW-SEM will only use the items/covariates that are specified. (The dangers of very large dataset is that they can take a long time to load and can take up space in the R environment. This should not affect processing speed for the GWAS analysis, but can create headaches for pre-processing).
F1itemNames	a character list of the names of the items that load onto the first latent variable. These names must match variable names in the phenoData file.
F2itemNames	a character list of the names of the items that load onto the first latent variable. These names must match variable names in the phenoData file.
covariates	a character list of covariates that the latent variable will be regressed upon. The default vaule is NULL, but this is a silly value as typically analysts will include e.g. age, sex, and ancestry principle components in the analysis.
...	Not used. Forces remaining arguments to be specified by name.
fitfun	The fitfun argument specifies which fit function should be used in evaluating the GWAS model. Users may choose between the relatively rapid "WLS", or the slower but asymptotically optimal "ML". In many cases the the differences between the fit functions is trivial and the faster "WLS" option should be used, but in some situations the differences can be quite meaningful (such as when data are Missing at Random - MAR).
minMAF	The minimum valid minor allele frequency (MAF). Large differences between the variances of two variables in the same model can cause optimization failures that invalidate the model. Further, very small minor allele frequencies are more affected by outliers or influential observations. Accordingly, users can specify the minimum allowable MAF. The default value is MAF > .01. Users may also wish to filter out small MAF alleles from their genotype files in other software programs, such as PLINK.
gxe	The observed variable name that will be used to moderate (interact with) the effect of the SNP on the phenotypes. For example, you may want to moderate the SNP regression by sex. In this situation, you would specify gxe = "sex" and add "snp_sex" to the list of covariates.

exogenous — This argument specifies how you would like to integrate the covariates into the analysis. If exogenous = T, each items will be directly regressed on each covariate. If exogenous = F, the latent factor(s) will be directly regressed on each covariate. Setting exogenous = T does not assume that the items are related to the covariates proportional to their factor loadings (which is probably preferable in most cases).

Details

You can pass the result of this function to GWAS to run a GWAS.

Ordinal indicator thresholds are setup by setupThresholds. You can plot the model using omxGraphviz.

Value

buildTwoFac returns an MxModel object that can serve as input for the GWAS function.

See Also

Other model builder: buildItem(), buildOneFacRes(), buildOneFac()

Examples

```
pheno <- list()
for (i in 1:10) pheno[[paste0('i',i)]] <- rnorm(500)
pheno <- as.data.frame(pheno)
buildTwoFac(pheno, paste0('i',1:6), paste0('i',5:10))
```

GWAS — *Run a genome-wide association study (GWAS) using the provided model*

Description

Maturing The GWAS function is used to run a genome-wide association study based on the specified model. This function is design to take the output from buildOneFac, buildOneFacRes, and buildTwoFac as input, but can also take a similar user specified model. Users should be confident that the models they are running are statistically identified. It is advisable that the users empirically gauge time requirements by running a limited number of SNPs (e.g. 10) to ensure that all SNPs can be fit in a reasonable amount of time.

Usage

```
GWAS(model, snpData, out = "out.log", ..., SNP = NULL, startFrom = 1L)
```

Arguments

model	an MxModel model, specified using RAM or LISREL notation. The model argument is designed to take the output from e.g. buildOneFac (or the other prebuilt GW-SEM functions), but advanced users can specify their own arbitrary OpenMx Model or use Onyx to draw their path diagrams.
snpData	a pathway to a file containing GWAS data. The data can be in a variety of forms, such as standard PLINK format (bed/bim/fam), PLINK2 format (pgen/pvar/psam), Oxford format (bgen/sample), or CSV format (csv format in much slower due to the lack of compression for non-binary files).
out	a file name or pathway where the output from the analysis will be saved. The default pathway is "out.log", which will save the file in the working directory. Users should take caution when specifying the output file name so that the output from different analyses/chromosomes do not overwrite existing files.
...	Not used. Forces remaining arguments to be specified by name.
SNP	a numerical range that specifies the number of SNPs to be evaluated from the snpData file. This argument can be used to evaluate a subset of snps for model testing. e.g. 1:10 will run the first 10 snps to make sure that the model is functioning the way the users intends, that the files exist pathways are correct. This option is also very useful to specify a range of snps to be evaluated that is smaller than the complete file. For example, users may wish to run several discrete batches of analyses for chromosome 1, by running 1:10000, 100001:200000, etc. This prevents users from constructing numerous snap files for each chromosome. The default value of the SNP argument is NULL, which will run all snps in the file.
startFrom	a numerical value indicating which SNP is the first SNP to be analyzed. The function will then run every SNP from the specified SNP to the end of the GWAS data file. This is very useful if the analysis stops for some reason (i.e. the cluster is restarted for maintenance) and you can start from the last SNP that you analyzed. Note, you will want to label the output file (specified in out) with a new file name so that you don't overwrite the existing results.

Details

Adds a compute plan returned by prepareComputePlan to the provided model and runs it. Once analyses are complete, load your aggregated results with loadResults.

Value

The results for each SNP are recorded in the specified log file (out). In addition, data and estimates for the last SNP run are returned as an MxModel object (similar to the return value of mxRun). In this way, the last SNP processed is available for close inspection.

Examples

```
dir <- system.file("extdata", package = "gwsem")
pheno <- data.frame(anxiety=rnorm(500))
m1 <- buildItem(pheno, 'anxiety')
```

```
GWAS(m1, file.path(dir,"example.pgen"),
     file.path(tempdir(),"out.log"))
```

`isSuspicious` *Determine which results are suspicious*

Description

Maturing The GWAS function writes all results, both valid and invalid, to a log file. This function uses heuristics to try to classify rows as suspicious or unsuspicious. The loadResults function returns unsuspicious rows while loadSuspicious returns suspicious rows.

Usage

```
isSuspicious(got, pars)
```

Arguments

`got`	output passed from loadResults
`pars`	names of the parameters available in `got`

Details

Is it not recommended to call `isSuspicious` directly because it is already called by loadResults and loadSuspicious.

OpenMx reports exceptions in the 'catch1' column. Any error message in the 'catch1' column is suspicious. Any optimizer status code besides 'OK' is suspicious. It is suspicious if the focal parameter or its standard error is `NA`. If 'signAdj' was requested and it is `NA` then suspicion is also aroused. A mean and standard deviation are computed for each available parameter. Any estimate with an absolute Z score larger than 10 is suspicious.

Value

a vector of logicals for each row of `got` indicating suspicion (if TRUE)

See Also

Other reporting: `loadResults()`, `loadSuspicious()`, `plot.gwsemResult()`

Examples

```
tdir <- tempdir()
dir <- system.file("extdata", package = "gwsem")
pheno <- data.frame(anxiety=rnorm(500))
m1 <- buildItem(pheno, 'anxiety')
GWAS(m1, file.path(dir,"example.pgen"),
    file.path(tdir,"out.log"))
loadResults(file.path(tdir,"out.log"), "snp2anxiety")  # called in here
```

`loadResults` *Load GWAS results into a single data.frame*

Description

Maturing The `signAdj` column is important and not optional for latent factor models. Loadings to factor indicators can take any sign. If your focus is the regression from the SNP to the factor then this regression estimate will need to be multiplied by the sign of one of the factor loadings. Pick a loading associated with a strong indicator of the factor.

Usage

```
loadResults(
  path,
  focus,
  ...,
  extraColumns = c(),
  .retainSE = FALSE,
  signAdj = NULL,
  moderatorLevel = NULL
)
```

Arguments

`path`	vector of paths to result files created by GWAS
`focus`	parameter name on which to calculate a Z score and p-value
`...`	Not used. Forces remaining arguments to be specified by name.
`extraColumns`	character vector of additional columns to load
`.retainSE`	logical. Keep a column for the SE of the focus parameter
`signAdj`	name of column. Value of focus parameter is multiplied by the sign of the named column
`moderatorLevel`	see details

Details

A1 is the reference allele and A2 is the alternate allele.

Two columns are added, `Z` and `P`. `Z` is the focal parameter divded by its standard error. `P` is the unadjusted two-sided normal CDF corresponding to the absolute `Z` score.

Suspicious rows are excluded.

See Also

Other reporting: `isSuspicious()`, `loadSuspicious()`, `plot.gwsemResult()`

Examples

```
tdir <- tempdir()
dir <- system.file("extdata", package = "gwsem")
pheno <- data.frame(anxiety=rnorm(500))
m1 <- buildItem(pheno, 'anxiety')
GWAS(m1, file.path(dir,"example.pgen"),
    file.path(tdir,"out.log"))
loadResults(file.path(tdir,"out.log"), "snp2anxiety")
```

`loadSuspicious` *Load suspicious GWAS results into a single data.frame*

Description

Maturing These rows are excluded by loadResults.

Usage

```
loadSuspicious(
  path,
  focus,
  ...,
  extraColumns = c(),
  signAdj = NULL,
  moderatorLevel = NULL
)
```

Arguments

`path`	vector of paths to result files created by GWAS
`focus`	parameter name on which to calculate a Z score and p-value
`...`	Not used. Forces remaining arguments to be specified by name.
`extraColumns`	character vector of additional columns to load
`signAdj`	name of column. Value of focus parameter is multiplied by the sign of the named column
`moderatorLevel`	see details

See Also

Other reporting: `isSuspicious()`, `loadResults()`, `plot.gwsemResult()`

Examples

```
tdir <- tempdir()
dir <- system.file("extdata", package = "gwsem")
pheno <- data.frame(anxiety=rnorm(500))
m1 <- buildItem(pheno, 'anxiety')
GWAS(m1, file.path(dir,"example.pgen"),
    file.path(tdir,"out.log"))
loadSuspicious(file.path(tdir,"out.log"), "snp2anxiety")
```

`plot.gwsemResult` *Creates a Manhattan plot*

Description

Uses the qqman package to create a Manhattan plot.

Usage

```
## S3 method for class 'gwsemResult'
plot(x, y, ...)
```

Arguments

`x`	the result of loadResults
`y`	an extra argument that should not be used
`...`	arguments forwarded to manhattan

Value

A Manhattan plot.

See Also

Other reporting: `isSuspicious()`, `loadResults()`, `loadSuspicious()`

Examples

```
tdir <- tempdir()
dir <- system.file("extdata", package = "gwsem")
pheno <- data.frame(anxiety=rnorm(500))
m1 <- buildItem(pheno, 'anxiety')
GWAS(m1, file.path(dir,"example.pgen"),
    file.path(tdir,"out.log"))
got <- loadResults(file.path(tdir,"out.log"), "snp_to_anxiety")
plot(got)
```

`prepareComputePlan` *Return a suitable compute plan for a genome-wide association study*

Description

Maturing Instead of using OpenMx's default model processing sequence (i.e., omxDefaultComputePlan), it is more efficient and convienient to assemble a compute plan tailored for a genome-wide association study. This function returns a compute plan that loads SNP data into model `modelName`, fits the model, outputs the results to `out`, and repeats this procedure for all SNPs.

Usage

```
prepareComputePlan(
  model,
  snpData,
  out = "out.log",
  ...,
  SNP = NULL,
  startFrom = 1L
)
```

Arguments

`model`	an MxModel model, specified using RAM or LISREL notation. The model argument is designed to take the output from e.g. buildOneFac (or the other prebuilt GW-SEM functions), but advanced users can specify their own arbitrary OpenMx Model or use Onyx to draw their path diagrams.
`snpData`	a pathway to a file containing GWAS data. The data can be in a variety of forms, such as standard PLINK format (bed/bim/fam), PLINK2 format (pgen/pvar/psam), Oxford format (bgen/sample), or CSV format (csv format in much slower due to the lack of compression for non-binary files).
`out`	a file name or pathway where the output from the analysis will be saved. The default pathway is "out.log", which will save the file in the working directory. Users should take caution when specifying the output file name so that the output from different analyses/chromosomes do not overwrite existing files.
`...`	Not used. Forces remaining arguments to be specified by name.
`SNP`	a numerical range that specifies the number of SNPs to be evaluated from the snpData file. This argument can be used to evaluate a subset of snps for model testing. e.g. 1:10 will run the first 10 snps to make sure that the model is functioning the way the users intends, that the files exist pathways are correct. This option is also very useful to specify a range of snps to be evaluated that is smaller than the complete file. For example, users may wish to run several discrete batches of analyses for chromosome 1, by running 1:10000, 100001:200000, etc. This prevents users from constructing numerous snap files for each chromosome. The default value of the SNP argument is NULL, which will run all snps in the file.

`startFrom` a numerical value indicating which SNP is the first SNP to be analyzed. The function will then run every SNP from the specified SNP to the end of the GWAS data file. This is very useful if the analysis stops for some reason (i.e. the cluster is restarted for maintenance) and you can start from the last SNP that you analyzed. Note, you will want to label the output file (specified in out) with a new file name so that you don't overwrite the existing results.

Details

You can request a specific list of SNPs using the `SNP` argument. The numbers provided in `SNP` refer to offsets in the `snpData` file. For example, `SNP=c(100,200)` will process the 100th and 200th SNP. The first SNP in the `snpData` file is at offset 1. When `SNP` is omitted then all available SNPs are processed.

The suffix of `snpData` filename is interpreted to signal the format of how the SNP data is stored on disk. Suffixes 'pgen', 'bed', and 'bgen' are supported. Per-SNP descriptions are found in different places depending on the suffix. For 'bgen', both the SNP data and description are built into the same file. In the case of 'pgen', an associated file with suffix 'pvar' is expected to exist (see the spec for details). In the case of 'bed', an associated 'bim' file is expected to exist (see the spec for details). The chromosome, base-pair coordinate, and variant ID are added to each line of `out`.

The code to implement method='pgen' is based on plink 2.0 alpha. plink's 'bed' file format is supported in addition to 'pgen'. Data are coerced appropriately depending on the type of the destination column. For a numeric column, data are recorded as the values NA, 0, 1, or 2. An ordinal column must have exactly 3 levels.

For `method='bgen'`, the file `path+".bgi"` must also exist. If not available, generate this index file with the bgenix tool.

For 'bgen' and 'pgen' formats, the numeric column can be populated with a dosage (sum of probabilities multiplied by genotypes) if these data are available.

A compute plan does not do anything by itself. You'll need to combine the compute plan with a model (such as returned by buildOneFac) to perform a GWAS.

Value

The given model with an appropriate compute plan.

See Also

GWAS

Examples

```
pheno <- data.frame(anxiety=cut(rnorm(500), c(-Inf, -.5, .5, Inf),
ordered_result = TRUE))
m1 <- buildItem(pheno, 'anxiety')
dir <- system.file("extdata", package = "gwsem")
m1 <- prepareComputePlan(m1, file.path(dir,"example.pgen"))
m1$compute
```

setupExogenousCovariates *Set up exogenous model covariates*

Description

Experimental In GWAS, including a number of the first principle components as covariates helps reduce false positives caused by population stratification. This function adds paths from covariates to manifest indicators (`itemNames`). Covariates are always treated as continuous variables (not ordinal).

Usage

```
setupExogenousCovariates(model, covariates, itemNames)
```

Arguments

`model`	an MxModel model, specified using RAM or LISREL notation. The model argument is designed to take the output from e.g. buildOneFac (or the other prebuilt GW-SEM functions), but advanced users can specify their own arbitrary OpenMx Model or use Onyx to draw their path diagrams.
`covariates`	a character vector naming covariates available in the model data
`itemNames`	a character vector of item names

Details

This is not the only way to adjust a model for covariates. For example, in a single factor model (e.g., buildOneFac), it would be more appropriate to adjust the latent factor instead of the manifest indicators. This is how endogenous covariates work. However, exogenous covariate adjustments to latent variables are only possible with a maximum likelihood fit function (mxFitFunctionML). For mxFitFunctionWLS, only manifest indicators can be adjusted for exogenous covariates. This function always adjusts manifest indicators regardless of the fit function.

You generally do not need to call this function directly because it is already called by buildOneFac and similar. This function is provided for advanced users who wish to write their own model building functions.

Value

The given MxModel with paths added from covariates to manifest indicators.

Examples

```
m1 <- mxModel("test", type="RAM",
             latentVars = "sex", manifestVars = "anxiety",
             mxData(data.frame(sex=rbinom(10,1,.5)), 'raw'))
m1 <- setupExogenousCovariates(m1, 'sex', 'anxiety')
```

`setupThresholds` *Set up thresholds for ordinal indicators*

Description

Experimental Ordinal indicator thresholds are freely estimated with fixed means and variance. This function adds thresholds to the given `model`. If no indicators are ordinal, the given `model` is returned without changes.

Usage

```
setupThresholds(model)
```

Arguments

`model`	an MxModel model, specified using RAM or LISREL notation. The model argument is designed to take the output from e.g. buildOneFac (or the other prebuilt GW-SEM functions), but advanced users can specify their own arbitrary OpenMx Model or use Onyx to draw their path diagrams.

Details

Thresholds are added using mxThreshold. Starting values for thresholds use the defaults provided by this function which assumes a mean of zero and variance of the square root of two. This variance is appropriate for buildOneFac where the implied model variance of ordinal indicators is one plus the square of the factor loading, and the loading's starting value is 1.0.

You generally do not need to call this function directly because it is already called by buildOneFac and similar. This function is provided for advanced users who wish to write their own model building functions.

Value

The given MxModel with appropriate thresholds added.

Examples

```
pheno <- data.frame(anxiety=cut(rnorm(500), c(-Inf, -.5, .5, Inf),
                    ordered_result = TRUE))
m1 <- buildItem(pheno, 'anxiety')
m1 <- setupThresholds(m1)
m1$Thresholds
```

Index