Package ‘metabomxtr’

January 30, 2026

Type Package

Title A package to run mixture models for truncated metabolomics data with normal or lognormal distributions

Version 1.44.0

Date 2022-08-17

Author Michael Nodzenski, Anna Reisetter, Denise Scholtens

Maintainer Michael Nodzenski <michael.nodzenski@gmail.com>

Depends methods,Biobase

Imports optimx, Formula, plyr, multtest, BiocParallel, ggplot2

Suggests xtable, reshape2

Description The functions in this package return optimized parameter estimates and log likelihoods for mixture models of truncated data with normal or lognormal distributions.

License GPL-2

biocViews ImmunoOncology, Metabolomics, MassSpectrometry

git_url https://git.bioconductor.org/packages/metabomxtr

git_branch RELEASE_3_22

git_last_commit db88c37

git_last_commit_date 2025-10-29

Repository Bioconductor 3.22

Date/Publication 2026-01-29

Contents

metabomxtr-package . . . 2
addBatchMeans . . . 4
addOutlierInfo . . . 5
allMissingLevels . . . 6
anyMissingLevels . . . 7
euMetabCData . . . 8
euMetabData . . . 8
idMissingLevels . . . 9
metabdata . . . 10
metabplot . . . 12

mixnorm 14
mxtrmod 17
mxtrmodLL 19
mxtrmodLRT 21
mxtrmodstart 22
removeAllMissingCatVar 24
removeMissingLevels 25
runMxtrmod 26
xdesign-methods 28
yvals-methods 28
zdesign-methods 29

Index **30**

`metabomxtr-package` *A package to run mixture models on truncated normal or lognormal data*

Description

The functions in this package return optimized parameter estimates and negative log-likelihoods for mixture models of truncated normal or lognormal data.

Details

Package:	metabomxtr
Type:	Package
Version:	1.11.1
Date:	2017-05-30
License:	GPL-2

The function mxtrmodLL calculates the negative log-likelihood of mixture models. The function mxtrmodstart returns starting parameter estimates to be used when optimizing the mixture model parameters. The function mxtrmod returns optimized mixture model parameter estimates and the negative log-likelihood of the model. The function mxtrmodLRT performs likelihood ratio tests of full vs. reduced mixture models. The function mixnorm performs per-metabolite batch normalization using a mixture model with batch-specific thresholds and run order correction if desired.

Author(s)

Michael Nodzenski, Anna Reisetter, Denise Scholtens

Maintainer: Michael Nodzenski <michael.nodzenski@northwestern.edu>

References

Moulton LH, Halsey NA. A mixture model with detection limits for regression analyses of antibody response to vaccine. Biometrics. 1995 Dec;51(4):1570-8. Nodzenski M, Muehlbauer MJ, Bain JR, Reisetter AC, Lowe WL Jr, Scholtens DM. Metabomxtr: an R package for mixture-model analysis of non-targeted metabolomics data. Bioinformatics. 2014 Nov 15;30(22):3287-8. Reisetter AC, Muehlbauer MJ, Bain JR, Nodzenski M, Stevens RD, Ilkayeva O, Metzger BE, Newgard CB, Lowe WL Jr, Scholtens DM. Mixture model normalization for non-targeted gas chromatography/mass spectrometry metabolomics data. BMC Bioinformatics. 2017 Feb 2;18(1):84.

Examples

```
###Run mixture model analyses

#Create sample data
set.seed(123)
yvar<-rlnorm(200)
these<-sample(1:100,20)
yvar[these]<-NA
logyvar<-log(yvar)
y2var<-rlnorm(200)
those<-sample(1:200,25)
y2var[those]<-NA
logy2var<-log(y2var)
pred1<-sample(0:1,200,replace=TRUE)
pred2<-sample(1:10,200,replace=TRUE)
pred3<-sample(0:1,200,replace=TRUE)
pred3miss<-sample(1:200,50)
pred3[pred3miss]<-NA
testdata<-data.frame(cbind(yvar,y2var,logyvar,logy2var,pred1,pred2,pred3))

#Get the names of the response variables
ynames<-names(testdata)[3:4]

#Run a full mixture model on each response variable
fullMod<-~pred1+pred2+pred3|pred1+pred2+pred3
fullModRes<-mxtrmod(ynames=ynames,mxtrModel=fullMod,data=testdata)
fullModRes

#Run a reduced mixture model on each response variable
redMod<-~pred2|pred2
redModRes<-mxtrmod(ynames=ynames,mxtrModel=redMod,data=testdata,fullModel=fullMod)
redModRes

#Compare models using likelihood ratio test
mxtrmodLRT(fullModRes,redModRes)

###Perform mixture model normalization

#load control data set
data(euMetabCData)

#load experimental data
data(euMetabData)

#specify target metabolites
ynames <- c("betahydroxybutyrate","pyruvic_acid","malonic_acid","aspartic_acid")

#run mixture model normalization
euMetabNorm <- mixnorm(ynames,
                       batch="batch",
                       mxtrModel=~pheno+batch|pheno+batch,
                       batchTvals=c(10.76,11.51,11.36,10.31,11.90),
                       cData=euMetabCData,
                       data=euMetabData)
```

`addBatchMeans` *A function to append batch specific mean metabolite abundances to data frames containing raw values.*

Description

This function appends batch specific mean metabolite abundances to data frames containing raw values.

Usage

```
addBatchMeans(metab.col, batch.col, in.data, sample.type.name, analytical.type.name, color.col=NUL
```

Arguments

`metab.col` A character value indicating the name of the variable in the input data set that corresponds to metabolite abundance.

`batch.col` A character value indicating the name of the variable in the input data sets that corresponds to batch. If not specified, this argument defaults to "Batch".

`in.data` A data frame containing the variables specified in metab.col, batch.col, and color.col.

`sample.type.name` A character value indicating the type of sample: "Quality Control" or "Experimental".

`analytical.type.name` A character value indicating whether the data have been normalized. Either "Before Normalization" or "After Normalization".

`color.col` A character value indicating the name of the variable in the input data set corresponding to a grouping variable (for instance mom vs. baby samples).

Details

This function adds rows reporting batch specific mean metabolite abundances to data frames of raw values. This is not meant to be a stand alone function but rather work as a part of function metabplot.

Value

Returns a dataframe.

Author(s)

Michael Nodzenski

References

Nodzenski M, Muehlbauer MJ, Bain JR, Reisetter AC, Lowe WL Jr, Scholtens DM. Metabomxtr: an R package for mixture-model analysis of non-targeted metabolomics data. Bioinformatics. 2014 Nov 15;30(22):3287-8.

Examples

```
data(euMetabCData)
euMetabCData$outlier<-FALSE
cdata.with.batch.means<-addBatchMeans("pyruvic_acid", "batch", euMetabCData, sample.type.name="Quality Contr
```

`addOutlierInfo` *A function to determine whether specific metabolite observations are outliers.*

Description

This function adds a column to a dataframe of metabolite abundance values indicating whether a observations for a specific metabolite are outliers.

Usage

```
addOutlierInfo(metab.col, in.data, outlier.sd.thresh=2)
```

Arguments

`metab.col` A character value indicating the name of the variable in the input data set that corresponds to metabolite abundance.

`in.data` A data frame containing the variables specified in metab.col.

`outlier.sd.thresh` The number of standard deviations from the mean a point must be to be considered an outlier. This argument defaults to 2.

Details

This function adds a column to a dataframe of metabolite abundance values indicating whether a observations for a specific metabolite are outliers.

Value

Returns a dataframe.

Author(s)

Michael Nodzenski

References

Nodzenski M, Muehlbauer MJ, Bain JR, Reisetter AC, Lowe WL Jr, Scholtens DM. Metabomxtr: an R package for mixture-model analysis of non-targeted metabolomics data. Bioinformatics. 2014 Nov 15;30(22):3287-8.

Examples

```
data(euMetabCData)
outlier.data<-addOutlierInfo("pyruvic_acid",  euMetabCData, outlier.sd.thresh=2)
```

`allMissingLevels` *A function to determine whether metabolite levels are present for at most one level of a categorical variable.*

Description

This function determines metabolite data are present for at most one level of a categorical predictor variable, and thus whether that predictor needs to be removed from the mixture model.

Usage

```
allMissingLevels(missing.levels.list, dataset)
```

Arguments

`missing.levels.list`
: A list output by function idMissingLevels indicating which categorical variables have no corresponding metabolite data for at least one level.

`dataset`
: A data frame containing the variables specified in missing.levels.list.

Value

Returns a list indicating whether categorical variables in missing.levels.list have metabolite data present for at most one level.

Author(s)

Michael Nodzenski

Examples

```
#create example analysis data
data(euMetabCData)
example.data<-euMetabCData
example.data[example.data$batch==1, "aspartic_acid"]<-NA

#check to determine if aspartic acid values are entirely missing for
#any level of batch or pheno
missing.levels.check<-lapply( "aspartic_acid", anyMissingLevels, cat.vars=c("pheno", "batch"), dataset=examp
names(missing.levels.check)<-"aspartic_acid"
missing.levels.check

#find the specific missing level
missing.level.ids<-idMissingLevels( "aspartic_acid", missing.levels.check, example.data)

#check to see if those variables have completely missing data
allMissingLevels(missing.level.ids, example.data)
```

`anyMissingLevels` *A function to determine whether any level of a categorical variable has completely missing outcome (metabolite) data*

Description

This function determines whether any level of a categorical predictor variable has completely missing outcome data.

Usage

```
anyMissingLevels( yname,  cat.vars,  dataset)
```

Arguments

`yname`	A character string indicating the outcome variable, i.e., a metabolite.
`cat.vars`	A character vector of categorical variable names.
`dataset`	A data frame containing the yname and cat.vars variables.

Value

Returns a list indicating whether each of the variables specified in argument cat.vars has at least one level with completely missing values for the outcome variable specified in yname.

Author(s)

Michael Nodzenski

Examples

```
#create example analysis data
data(euMetabCData)
example.data<-euMetabCData
example.data[example.data$batch==1, "aspartic_acid"]<-NA

#check to determine if aspartic acid values are entirely missing for
#any level of batch or pheno
anyMissingLevels( "aspartic_acid", c( "batch",  "pheno"), example.data)
```

euMetabCData — *A sample data set of truncated metabolomics data.*

Description

This data set contains a subset of non-targeted GC/MS data for mother / baby pairs of Northern European ancestry who participated in the Hyperglycemia and Adverse Pregnancy Outcome (HAPO) Metabolomics study.

Usage

```
data("euMetabCData")
```

Format

A data frame with 30 observations on the following 6 variables.

`batch` A factor variable for the batch in which the sample was processed with levels `1 2 3 4 5`.

`pheno` A factor variable indicating whether the serum sample was from a mother or baby with levels `BABY MOM`.

`betahydroxybutyrate` A numeric vector of non-normalized GC/MS log2 transformed peak areas for this metabolite.

`pyruvic_acid` A numeric vector of non-normalized GC/MS log2 transformed peak areas for this metabolite.

`malonic_acid` A numeric vector of non-normalized GC/MS log2 transformed peak areas for this metabolite.

`aspartic_acid` A numeric vector of non-normalized GC/MS log2 transformed peak areas for this metabolite.

Details

The 30 rows correspond to 3 mom and 3 baby control samples run at the beginning, middle and end of each batch. All control samples were drawn from an identical pool.

euMetabData — *A sample data set of truncated metabolomics data.*

Description

This data set contains a subset of non-targeted GC/MS data for mother / baby pairs of Northern European ancestry who participated in the Hyperglycemia and Adverse Pregnancy Outcome (HAPO) Metabolomics study.

Usage

```
data("euMetabData")
```

Format

A data frame with 120 observations on the following 6 variables.

`batch` A factor variable for the batch in which the sample was processed with levels `1 2 3 4 5`.

`pheno` A factor variable indicating whether the serum sample was from a mother or baby with levels `BABY MOM`.

`betahydroxybutyrate` A numeric vector of non-normalized GC/MS log2 transformed peak areas for this metabolite.

`pyruvic_acid` A numeric vector of non-normalized GC/MS log2 transformed peak areas for this metabolite.

`malonic_acid` A numeric vector of non-normalized GC/MS log2 transformed peak areas for this metabolite.

`aspartic_acid` A numeric vector of non-normalized GC/MS log2 transformed peak areas for this metabolite.

Details

The 120 rows correspond to mother / baby sample triples of analytical interest. 24 of these samples were run in each batch. Row names ending in "_mf" are for fasting maternal samples. Row names ending in "_m1" are for maternal samples at 1-hour into an oral glucose tolerance test. Row names ending in "_bc" are for samples of baby cord blood collected at birth.

`idMissingLevels` *A function to determine the levels of a categorical variable with completely missing outcome data.*

Description

This function determines the levels of a categorical variable with completely missing outcome data.

Usage

```
idMissingLevels( yname,  missing.levels.list,  dataset)
```

Arguments

`yname` A character string indicating the outcome variable, i.e., a metabolite.

`missing.levels.list` A named list indicating whether categorical predictors in argument dataset have levels where argument yname is completely missing. The names of the list objects should be metabolite names, and should include argument yname.

`dataset` A data frame containing the variables specified in yname and missing.levels.list.

Value

Returns a list of the specific levels of the categorical variables included in missing.levels.list that are missing all values for the metabolite specified in yname.

Author(s)

Michael Nodzenski

Examples

```
#create example analysis data
data(euMetabCData)
example.data<-euMetabCData
example.data[example.data$batch==1, "aspartic_acid"]<-NA

#check to determine if aspartic acid values are entirely missing for
#any level of batch or pheno
missing.levels.check<-lapply( "aspartic_acid", anyMissingLevels, cat.vars=c("pheno", "batch"), dataset=examp
names(missing.levels.check)<-"aspartic_acid"
missing.levels.check

#find the specific missing level
idMissingLevels( "aspartic_acid", missing.levels.check, example.data)
```

---

metabdata — *A sample data set of truncated metabolomics data*

Description

This data set contains log transformed metabolite levels and phenotype data of sample 115 of pregnant women. Metabolite levels are contained in columns 11-59, and missing values are indicated by NA. All metabolites included contain at least 5 missing values. Columns 1:10 represent phenotypic data.

Usage

```
data(metabdata)
```

Format

A data frame with 115 observations on the following 59 variables.

`sample` a numeric vector representing sample number.

`PHENO` a factor representing high and low fasting blood glucose, with levels `MomHighFPG` `MomLowFPG`.

`age_ogtt` a numeric vector representing the woman's age in years when the oral glucose tolerance test was performed.

`age_ogtt_mc` a numeric vector representing mean centered age_ogtt.

`ga_ogtt_wks` a numeric vector representing gestational age in weeks when the oral glucose tolerance test was performed.

`ga_ogtt_wks_mc` a numeric vector representing mean centered ga_ogtt_wks.

`FCg` a factor representing field center where the data were collected with levels `BCD` `E` `F` `P`.

`parity12` a numeric vector with 1 indicative of previous pregnancy and 0 otherwise.

`storageTimesYears` a numeric vector representing the number of years the metabolite sample had been stored prior to assay.

`storageTimesYears_mc` a numeric vector representing mean centered storageTimesYears.

`ketovaline` a numeric vector representing log2 transformed metabolite abundance.

`alpha.ketoglutaric.acid` a numeric vector representing log2 transformed metabolite abundance.

`ketoleucine` a numeric vector representing log2 transformed metabolite abundance.

`acetoacetate` a numeric vector representing log2 transformed metabolite abundance.

`aldohexose` a numeric vector representing log2 transformed metabolite abundance.

`beta.alanine` a numeric vector representing log2 transformed metabolite abundance.

`methylmalonic.acid` a numeric vector representing log2 transformed metabolite abundance.

`creatinine` a numeric vector representing log2 transformed metabolite abundance.

`hexuronic.acid` a numeric vector representing log2 transformed metabolite abundance.

`ethanolamine` a numeric vector representing log2 transformed metabolite abundance.

`glutamine` a numeric vector representing log2 transformed metabolite abundance.

`glycolic.acid` a numeric vector representing log2 transformed metabolite abundance.

`isoleucine` a numeric vector representing log2 transformed metabolite abundance.

`malonic.acid` a numeric vector representing log2 transformed metabolite abundance.

`ribose` a numeric vector representing log2 transformed metabolite abundance.

`phenylalanine` a numeric vector representing log2 transformed metabolite abundance.

`pyruvic.acid` a numeric vector representing log2 transformed metabolite abundance.

`hexitol.1` a numeric vector representing log2 transformed metabolite abundance.

`lysine` a numeric vector representing log2 transformed metabolite abundance.

`disaccharide.1` a numeric vector representing log2 transformed metabolite abundance.

`tyrosine` a numeric vector representing log2 transformed metabolite abundance.

`leucine` a numeric vector representing log2 transformed metabolite abundance.

`hexitol.2` a numeric vector representing log2 transformed metabolite abundance.

`disaccharide.2` a numeric vector representing log2 transformed metabolite abundance.

`ornithine` a numeric vector representing log2 transformed metabolite abundance.

`disaccharide.3` a numeric vector representing log2 transformed metabolite abundance.

`beta.tocopherol` a numeric vector representing log2 transformed metabolite abundance.

`hexitol.3` a numeric vector representing log2 transformed metabolite abundance.

`benzene.1.2.4.triol` a numeric vector representing log2 transformed metabolite abundance.

`heptadecane` a numeric vector representing log2 transformed metabolite abundance.

`nonadecane` a numeric vector representing log2 transformed metabolite abundance.

`tetradecanedioic.acid` a numeric vector representing log2 transformed metabolite abundance.

`pentadecanoic.acid` a numeric vector representing log2 transformed metabolite abundance.

`undecane` a numeric vector representing log2 transformed metabolite abundance.

`methyl.heptadecanoate` a numeric vector representing log2 transformed metabolite abundance.

`hydrocarbon` a numeric vector representing log2 transformed metabolite abundance.

`deoxyhexose` a numeric vector representing log2 transformed metabolite abundance.

`glucose` a numeric vector representing log2 transformed metabolite abundance.

`pentose.sugar` a numeric vector representing log2 transformed metabolite abundance.

`hexitol.4` a numeric vector representing log2 transformed metabolite abundance.

`beta.sitosterol` a numeric vector representing log2 transformed metabolite abundance.

`X1.5.anhydroglucitol` a numeric vector representing log2 transformed metabolite abundance.

`threonine` a numeric vector representing log2 transformed metabolite abundance.

`proline` a numeric vector representing log2 transformed metabolite abundance.

`campesterol` a numeric vector representing log2 transformed metabolite abundance.

`X6.deoxy.glucose` a numeric vector representing log2 transformed metabolite abundance.

`erythronic.acid` a numeric vector representing log2 transformed metabolite abundance.

`methyl.myristate` a numeric vector representing log2 transformed metabolite abundance.

`methyl.eicosanoate` a numeric vector representing log2 transformed metabolite abundance.

Source

Scholtens DM, Muehlbauer MJ, Daya NR, Stevens RD, Dyer AR, Lowe LP, Metzger BE, Newgard CB, Bain JR, Lowe WL Jr; HAPO Study Cooperative Research Group. Metabolomics reveals broad-scale metabolic perturbations in hyperglycemic mothers during pregnancy. Diabetes Care. 2014 Jan; 37(1):158-66.

`metabplot` *A function to plot metabolite abundance before and after normalization.*

Description

This function plots metabolite abundance before and after normalization.

Usage

```
metabplot(metab.name, batch="Batch", raw.obs.data=NULL, raw.cont.data=NULL, norm.obs.data=NULL, no
```

Arguments

`metab.name`	A character value indicating the name of the variable in the input data sets that corresponds to metabolite abundance. If more than one data set is input, the metab.name variable must be the same across all inputs.
`batch`	A character value indicating the name of the variable in the input data sets that corresponds to batch. If not specified, this argument defaults to "Batch".
`raw.obs.data`	A data frame of raw (non-normalized) experimental sample metabolite abundances and predictor variables. If input, this must include the variables specified in metab.name, batch, and color.var.
`raw.cont.data`	A data frame of raw (non-normalized) quality control sample metabolite abundances and predictor variables. If input, this must include the variables specified in metab.name, batch, and color.var.

`norm.obs.data` A data frame or matrix of normalized experimental sample metabolite abundances output by function mixnorm. If raw.obs.data is specified, the variables specified in batch and color.var do not need to be included in this object, and it is assumed that the row order in norm.obs.data is identical to raw.obs.data. If raw.obs.data is not specified, then this object must contain the variables specified in batch and color.var.

`norm.cont.data` A data frame or matrix of normalized quality control sample metabolite abundances output by function mixnorm. If raw.cont.data is specified, the variables specified in batch and color.var do not need to be included in this object, and it is assumed that the row order in norm.cont.data is identical to raw.obs.data. If raw.cont.data is not specified, then this object must contain the variables specified in batch and color.var.

`color.var` A character value indicating the name of the variable that will be used to define the color and grouping of data points in output plots.

`cont.outlier.sd.thresh`
The number of standard deviations from the mean a point must be to be considered an outlier for the raw quality control data. If plotting data output by function mixnorm, this should match the value for argument qc.sd.outliers in mixnorm. This argument defaults to 2, which is the same as the default for qc.sd.outliers. These outlying points will be indicated in the plots as "Excluded Outlier", and represent data points that were not included in estimating batch and other technical effects during normalization.

`norm.outlier.sd.thresh`
The number of standard deviations from the mean a point must be to be considered an outlier for the normalized experimental data. This argument defaults to 4. These outlying points will be indicated in the plots as "Potential Outlier", and represent data points that remain outliers after normalization that the user may wish to remove before downstream analysis.

Details

This function aims to show a 4 panelled plot of quality control and experimental sample metabolite abundance data both before and after normalization wit h function mixnorm. However, it will produce a plot as long as at least one of raw.obs.data, raw.cont.data, norm.obs.data, or norm.cont.data is specified. The user may include as few as one or as many as all 4 of these arguments.

Value

Returns a graph of metabolite abudance.

Author(s)

Michael Nodzenski

References

Nodzenski M, Muehlbauer MJ, Bain JR, Reisetter AC, Lowe WL Jr, Scholtens DM. Metabomxtr: an R package for mixture-model analysis of non-targeted metabolomics data. Bioinformatics. 2014 Nov 15;30(22):3287-8.

Examples

```
data(euMetabCData)
data(euMetabData)

ynames <- c("betahydroxybutyrate","pyruvic_acid","malonic_acid","aspartic_acid")

#in this example, batch minima specified in batchTvals were calculated from the full data set for this experimen
euMetabNorm <- mixnorm(ynames,
                       batch="batch",
                       mxtrModel=~pheno+batch|pheno+batch,
                       batchTvals=c(10.76,11.51,11.36,10.31,11.90),
                       cData=euMetabCData,
                       data=euMetabData,
                       qc.sd.outliers=2)

 #plot results
 metab.name.list<-names(euMetabNorm$obsNorm)
 plot.list<-lapply(metab.name.list, metabplot, batch="batch", raw.obs.data=euMetabData, raw.cont.data=euMeta
    norm.cont.data=euMetabNorm$ctlNorm, color.var="pheno" )
plot.list[[1]]
```

`mixnorm` *A function to perform per-metabolite batch normalization using a mixture model with batch-specific thresholds and run order correction if desired.*

Description

This function performs per-metabolite batch normalization using a mixture model with batch-specific thresholds and run order correction if desired.

Usage

```
mixnorm(ynames, batch = "Batch", mxtrModel=NULL, cData, data, batchTvals = NULL, removeCorrection=N
```

Arguments

`ynames`	A character vector of the mixture model outcome names, e.g. metabolites. If the input data object is a matrix or data frame, these should be column names. If the input data object is an expression set, these should be row names. Response variables should have a normal or lognormal distribution. If lognormal,log transformed variables should be input. Missing values should be denoted by NA.
`batch`	A character value indicating the name of the variable in cData and data that indicates batch. If not specified, this argument defaults to "Batch"'.
`mxtrModel`	A formula of the form ~x1+x2...|z1+z2..., where x's are the names of covariates included in the discrete portion of the model and z's are names of covariates included in the continuous portion. The covariate names must be the same for cData and data. The default model includes a variable specified in argument 'batch' for both discrete and continuous model components.If manually specified, mixture models must include at minimum batch in both the continuous and discrete portions. Models with covariates containing missing values will not run. See documentation for mxtrmod for additional details.

`cData` The input data object of control data to estimate normalization parameters. Matrices, data frames, and expression sets are acceptable classes. If a data frame or matrix, rows are subjects and columns are metabolites or outcomes.

`data` The input data object for observed values to be normalized (i.e. not controls). Matrices, data frames, and expression sets are acceptable classes. If a data frame or matrix, rows are subjects and columns are metabolites or outcomes.

`batchTvals` A vector of thresholds below which continuous variables are not observable. If specifying this argument, it must be the length of the unique levels of batch, with the thresholds in the same order that batch levels appear in cData. For instance, if there are 5 batches in cData, and they appear in order from 1 to 5, then argument batchTvals would need to be a numeric vector of length 5 with batch specific thresholds specified in that order. The default is the minimum across all response variables (metabolites) for each batch. ***Note that this default behavior may yield different normalization results for the same metabolite depending on the other metabolites entered as part of argument ynames. This should not be a concern if using all metabolite names as part of the ynames argument***. If running normalization on a subset of the full list of metabolites, to get the same results as the full set, the user should manually enter the minimum observed abundance across metabolites for each batch.

`removeCorrection` A character vector of variable names from mxtrModel whose effects should be estimated, but not subtracted from the non-normalized data. This parameter may be useful when data sets contain control samples of different types, for instance mothers and babies. In those instances, sample type may be an important covariate with respect to accurately estimating batch effects, necessitating inclusion in the mixture model, but it may not be of interest to actually subtract the estimated sample effect from the non-normalized data. If not specified, all estimated effects from the mixture model will be subtracted from the non-normalized data.

`nNA` The minimum number of unobserved values needed to be present for the discrete portion of the model likelihood to be calculated. Models for variables with fewer than nNA missing values will include only the continuous portion. The default value is 5.

`minProp` The minimum proportion of non-missing data in the response variable necessary to run the model. The default value is 0.2. Models will not be run if more than 80% of response variable values are missing.

`method` The method used to optimize the parameter estimates of the mixture model. "BFGS" is the default method. Other options are documented in the manual for the function 'optimx' in package optimx.

`qc.sd.outliers` The maximum number of standard deviations from the mean that should be considered non-outlying metabolite abundance values for the control data. Metabolite abundances greater than this will be removed from modeling batch effects to avoid having outliers unduly influence normalization parameters. This defaults to 2 standard deviations from the mean. To not exclude outliers, enter Inf for this argument.

Details

This function adapts the mxtrmod function in a normalization context in which aliquots from one or more control samples are run with each batch in a series of non-targeted metabolomics assays. The function accepts a data frame of log2 peak areas from control samples and a separate data frame of log2 peak areas from samples of analytical interest.

Value

Returns a list with the following components:

`normParamsZ` A data frame of the per-metabolite parameter estimates from the mixture model that are subtracted from the observed values to created the normalized data set. If a parameter estimated is NA, that parameter level was used as the reference. This occurs when the outcome (metabolite) values are completely missing for a particular level of a categorical variable, and the specific missing level can be found in the conv element of the function outcome.

`ctlNorm` A data frame of normalized values for the control samples. Observations that were NA in the input data will remain NA, but observations considered outliers based on argument qc.sd.outliers will not be normalized and instead coded as Inf.

`obsNorm` A data frame of normalized values for the samples of analytical interest. Observations that were NA in the input data will remain NA, but observations that could not be normalized due to too much missing quality control data, or too many outlying values in the quality control data, will not be normalized and instead coded Inf. This may occur if there is not enough data to estimated a specific batch effect (i.e., the effect of one out of 5 batches could not be estimated) or if there is not enough data to estimate the effect of a categorical predictor (i.e., if there is only QC data present for maternal samples, we can't estimate the effect relative to baby samples).

`conv` A data frame indicating whether models converged (indicated by a 0). This also indicates whether any categorical predictor levels were not modeled, and whether any categorical predictors were omitted from the model entirely. Specific levels of categorical predictors will be excluded from models when quality control metabolite data are entirely missing for that level. These variables and corresponding levels are reported in column "predictors_missing_levels". Categorical variables will be completely removed from models when quality control metabolite values are missing entirely or present for only one level of a categorical variable. These omitted variables are reported in column "excluded_predictors". Both columns indicate variables, or specific levels of variables, whose effects could not be accounted for in normalization. Normalized values will therefore not be output for metabolites with these warnings. For example, if metabolites were assayed in 20 total batches, and quality control data were completely missing for batch 2, the effect of batch 2 cannot be estimated but the effects of the other 19 batches can and will be estimated. In the normalized data, non-missing metabolite values in batch 2 will be re-coded as Inf, but the correction will be made for the remaining batches. If columns predictors_missing_levels and excluded_predictors are not present, then no exclusions were made.

Author(s)

Denise Scholtens, Michael Nodzenski, Anna Reisetter

References

Moulton LH, Halsey NA. A mixture model with detection limits for regression analyses of antibody response to vaccine. Biometrics. 1995 Dec;51(4):1570-8. Nodzenski M, Muehlbauer MJ, Bain JR, Reisetter AC, Lowe WL Jr, Scholtens DM. Metabomxtr: an R package for mixture-model analysis of non-targeted metabolomics data. Bioinformatics. 2014 Nov 15;30(22):3287-8. Reisetter AC,

Muehlbauer MJ, Bain JR, Nodzenski M, Stevens RD, Ilkayeva O, Metzger BE, Newgard CB, Lowe WL Jr, Scholtens DM. Mixture model normalization for non-targeted gas chromatography/mass spectrometry metabolomics data. BMC Bioinformatics. 2017 Feb 2;18(1):84.

Examples

```
data(euMetabCData)
data(euMetabData)

ynames <- c("betahydroxybutyrate","pyruvic_acid","malonic_acid","aspartic_acid")

#in this example, batch minima specified in batchTvals were calculated from the full data set for this experimen
euMetabNorm <- mixnorm(ynames,
                       batch="batch",
                       mxtrModel=~pheno+batch|pheno+batch,
                       batchTvals=c(10.76,11.51,11.36,10.31,11.90),
                       cData=euMetabCData,
                       data=euMetabData)
```

`mxtrmod` *A function to return optimized parameter estimates and the negative log-likelihood of mixture models for truncated normal or lognormal data*

Description

This function returns optimized parameter estimates and the negative log-likelihood of mixture models for truncated normal or lognormal data, via call to function runMxtrmod, after properly accounting for factor variable predictors with entirely missing outcome data.

Usage

```
mxtrmod(ynames, mxtrModel, Tvals=NULL, nNA=5, minProp=0.2, method="BFGS", data, fullModel=NULL, rem
```

Arguments

`ynames`	A character vector of the mixture model outcome names, e.g. metabolites. If the input data object is a matrix or data frame, these should be column names. If the input data object is an expression set, these should be row names. Response variables should have normal or lognormal distributions. If lognormal, log transformed variables should be input. Missing values should be denoted by NA.
`mxtrModel`	A formula of the form ~x1+x2...|z1+z2..., where x's are the names of covariates included in the discrete portion of the model and z's are names of covariates included in the continuous portion. For intercept only models, enter 1 instead of covariate names on the appropriate side of the |.
`Tvals`	A vector of thresholds below which continuous variables are not observable. By default, this parameter will be set to the minimum of the response variable.
`nNA`	The minimum number of unobserved values needed to be present for the discrete portion of the model likelihood to be calculated. Models for variables with fewer than nNA missing values will include only the continuous portion. The default value is 5.

`minProp` The minimum proportion of non-missing data in the response variable necessary to run the model. The default value is 0.2. Models will not be run if more than 80% of response variable values are missing.

`method` The method used to optimize the parameter estimates of the mixture model. "BFGS" is the default method. Other options are documented in the manual for the function 'optimx' in package optimx.

`data` The input data object. Matrices, data frames, and expression sets are all acceptable classes. If a data frame or matrix, rows are subjects and columns are metabolites or outcomes.

`fullModel` A formula of the form ~x1+x2...|z1+z2..., where x's are the names of covariates included in the discrete portion of the full model and z's are names of covariates included in the continuous portion. Input if the mxtrModel parameter represents a reduced model.

`remove.outlier.sd` The maximum number of standard deviations from the mean that should be considered non-outlying metabolite abundance values. Metabolite abundances greater than this will be removed from modeling to avoid having outliers unduly influence model parameters. This defaults to NULL, meaning all data will be used in modeling.

Value

Returns a data frame containing optimized estimates for all parameters in the mixture model, the negative log likelihood of the model, the optimization method used, the total number of observations used, whether the algorithm converged, whether any categorical predictor levels were not modeled, and whether any categorical predictors were omitted from the model entirely. Specific levels of categorical predictors will be excluded from models when metabolite data are entirely missing for that level. These variables and corresponding levels are reported in column "predictors_missing_levels". Categorical variables will be completely removed from models when metabolite values are missing entirely or present for only one level of the categorical variable. These omitted variables are reported in column "excluded_predictors". If those columns are not present in the output, no predictor exclusions occured.

Note

This function may generate warning messages about production of NaNs, but the function is still operating normally.

Author(s)

Michael Nodzenski, Anna Reisetter, Denise Scholtens

References

Moulton LH, Halsey NA. A mixture model with detection limits for regression analyses of antibody response to vaccine. Biometrics. 1995 Dec;51(4):1570-8. Nodzenski M, Muehlbauer MJ, Bain JR, Reisetter AC, Lowe WL Jr, Scholtens DM. Metabomxtr: an R package for mixture-model analysis of non-targeted metabolomics data. Bioinformatics. 2014 Nov 15;30(22):3287-8.

Examples

```
#Create sample data frame
set.seed(123)
```

```
yvar<-rlnorm(200)
these<-sample(1:100,20)
yvar[these]<-NA
logyvar<-log(yvar)
y2var<-rlnorm(200)
those<-sample(1:200,25)
y2var[those]<-NA
logy2var<-log(y2var)
pred1<-sample(0:1,200,replace=TRUE)
pred2<-sample(1:10,200,replace=TRUE)
pred3<-sample(0:1,200,replace=TRUE)
pred3miss<-sample(1:200,50)
pred3[pred3miss]<-NA
testdata<-data.frame(cbind(yvar,y2var,logyvar,logy2var,pred1,pred2,pred3))

#Get the names of the response variables
ynames<-names(testdata)[3:4]

#Run a mixture model on each response variable
mod<-~pred1+pred2+pred3|pred1+pred2+pred3
mxtrmod(ynames=ynames,mxtrModel=mod,data=testdata)

#Create example expression set
#Specify the response variables
exprsobs<-t(testdata[,3:4])

#Specify the phenotype data
exprspheno<-testdata[,5:7]

#make phenotype data an annotated data frame
phenoData <- new("AnnotatedDataFrame",data=exprspheno)

#combine into example expression set
testexpr<-ExpressionSet(assayData=exprsobs,phenoData=phenoData)

#Get the names of the response variables
ynames<-rownames(exprs(testexpr))

#Run the mixture model on each response variable
mxtrmod(ynames=ynames,mxtrModel=mod,data=testexpr)

#Load the data set from the package
data(metabdata)

#Select the response variables
ynames<-names(metabdata)[11:17]

#Run the mixture models
mod2<-~PHENO|PHENO+age_ogtt_mc+parity12+ga_ogtt_wks_mc
mxtrmod(ynames,mxtrModel=mod2,data=metabdata)
```

`mxtrmodLL` *A function to return the negative log-likelihood of mixture models of truncated, normal or lognormal data*

Description

This function returns the negative log-likelihood of the specified mixture model.

Usage

```
mxtrmodLL(params, obsY, xVars, zVars, Tvals, includeDiscrete)
```

Arguments

params	A vector of parameter estimates for all paramters in the mixture model.
obsY	A vector containing the response variable, which must be normally or log normally distributed. If lognormal, log transformed Y's should be input as the response variable. Missing Y values should be indicated by NA.
xVars	The design matrix for the covariates included in the discrete portion of the model.
zVars	The design matrix for the covariates inlcuded in the continuous portion of the model.
Tvals	A vector of thresholds below which continuous variables are not observable.
includeDiscrete	A logical indicator for whether or not to include the discrete portion of the model.

Value

Returns the negative log-likelihood of the specified mixture model.

Author(s)

Michael Nodzenski, Anna Reisetter, Denise Scholtens

References

Moulton LH, Halsey NA. A mixture model with detection limits for regression analyses of antibody response to vaccine. Biometrics. 1995 Dec;51(4):1570-8.

Examples

```
#Create sample data
set.seed(123)
yvar<-rlnorm(200)
these<-sample(1:100,20)
yvar[these]<-NA
logyvar<-log(yvar)
y2var<-rlnorm(200)
those<-sample(1:200,25)
y2var[those]<-NA
logy2var<-log(y2var)
pred1<-sample(0:1,200,replace=TRUE)
pred2<-sample(1:10,200,replace=TRUE)
testdata<-data.frame(cbind(yvar,y2var,logyvar,logy2var,pred1,pred2))

#Create a vector of starting values for the function
startvals<-c(2,0,0,1.5,0,0,2)
```

```
#Create a vector of response variables
obsY<-testdata$logyvar

#Create the design matrix for the discrete portion of the model
xVars<-model.matrix(~pred1+pred2,data=testdata)

#Create the design matrix for the continuous portion of the model
zVars<-model.matrix(~pred1+pred2,data=testdata)

#Create the Tvals vector
Tvals<-rep(min(obsY,na.rm=TRUE),length(obsY))

#Determine if the discrete portion should be inlcuded in the model
includeDiscrete<-sum(is.na(obsY))>5

#Calculate the negative log-likelihood
mxtrmodLL(params=startvals,obsY=obsY,xVars=xVars,zVars=zVars,Tvals=Tvals,
          includeDiscrete=includeDiscrete)
```

`mxtrmodLRT` *A function to run likelihood ratio tests on full vs. reduced mixture models*

Description

This function runs likelihood ratio tests on full vs. reduced mixture models. Input arguments are data frame outputs from the mxtrmod function.

Usage

```
mxtrmodLRT(fullmod, redmod, adj = NULL)
```

Arguments

`fullmod`	The output data frame from the mxtrmod function on the full mixture model.
`redmod`	The output data frame from the mxtrmod function on the reduced mixture model.
`adj`	The adjustment method for multiple comparisons. The default is set to NULL. Options for adjustment methods are described in the documentation for the function mt.rawp2adjp in the multtest package.

Value

A data frame containing the response variables (i.e. metabolites), negative log likelihoods of full and reduced models, chi square statistics, degrees of freedom, p-values, and, if requested, adjusted p-values.

Author(s)

Michael Nodzenski, Anna Reisetter, Denise Scholtens

References

Moulton LH, Halsey NA. A mixture model with detection limits for regression analyses of antibody response to vaccine. Biometrics. 1995 Dec;51(4):1570-8.

Examples

```
#Create sample data
set.seed(123)
yvar<-rlnorm(200)
these<-sample(1:100,20)
yvar[these]<-NA
logyvar<-log(yvar)
y2var<-rlnorm(200)
those<-sample(1:200,25)
y2var[those]<-NA
logy2var<-log(y2var)
pred1<-sample(0:1,200,replace=TRUE)
pred2<-sample(1:10,200,replace=TRUE)
pred3<-sample(0:1,200,replace=TRUE)
pred3miss<-sample(1:200,50)
pred3[pred3miss]<-NA
testdata<-data.frame(cbind(yvar,y2var,logyvar,logy2var,pred1,pred2,pred3))

#Get the names of the response variables
ynames<-names(testdata)[3:4]

#Run a full mixture model on each response variable
fullMod<-~pred1+pred2+pred3|pred1+pred2+pred3
fullModRes<-mxtrmod(ynames=ynames,mxtrModel=fullMod,data=testdata)
fullModRes

#Run a reduced mixture model on each response variable
redMod<-~pred2|pred2
redModRes<-mxtrmod(ynames=ynames,mxtrModel=redMod,data=testdata,fullModel=fullMod)
redModRes

#Compare models using likelihood ratio test
mxtrmodLRT(fullModRes,redModRes)
```

`mxtrmodstart` *A function to generate starting parameter estimates for the optimization of mixture model parameters*

Description

This function returns starting parameter estimates for the optimization of the mixture model parameters. The intercept of the continuous portion is set to the mean of the observed responses and the intercept of the discrete portion is set to the log odds of having observed a response. All other parameter starting values are set to zero.

Usage

```
mxtrmodstart(obsY, xVars, zVars, includeDiscrete)
```

Arguments

`obsY` A vector containing the response variable, which must be normally or log normally distributed. If lognormal, log transformed Y's should be input as the response variable. Missing Y values should be indicated by NA.

`xVars` The design matrix for the covariates included in the discrete portion of the model.

`zVars` The design matrix for the covariates inlcuded in the continuous portion of the model.

`includeDiscrete` A logical indicator for whether or not to include the discrete portion of the model.

Value

A vector containing the starting values for each parameter in the mixture model function, to be used as starting points when optimizing the parameter estimates.

Author(s)

Michael Nodzenski, Anna Reisetter, Denise Scholtens

References

Moulton LH, Halsey NA. A mixture model with detection limits for regression analyses of antibody response to vaccine. Biometrics. 1995 Dec;51(4):1570-8.

Examples

```
#Create sample data
set.seed(123)
yvar<-rlnorm(200)
these<-sample(1:100,20)
yvar[these]<-NA
logyvar<-log(yvar)
y2var<-rlnorm(200)
those<-sample(1:200,25)
y2var[those]<-NA
logy2var<-log(y2var)
pred1<-sample(0:1,200,replace=TRUE)
pred2<-sample(1:10,200,replace=TRUE)
testdata<-data.frame(cbind(yvar,y2var,logyvar,logy2var,pred1,pred2))

#Create a vector of response variables
obsY<-testdata$logyvar

#Create the design matrix for the discrete portion of the model
xVars<-model.matrix(~pred1+pred2,data=testdata)

#Create the design matrix for the continuous portion of the model
zVars<-model.matrix(~pred1+pred2,data=testdata)

#Determine if the discrete portion should be inlcuded in the model
includeDiscrete<-sum(is.na(obsY))>5
```

```
#Calculate starting values
mxtrmodstart(obsY=obsY,xVars=xVars,zVars=zVars,includeDiscrete=includeDiscrete)
```

`removeAllMissingCatVar` *A function to remove categorical variables with insufficient data to estimate effects from mixture models*

Description

This function removes categorical variables from mixture models. This is needed when metabolite values are entirely missing or only present for one level of the categorical variable, making it impossible to estimate effects for the categorical variable.

Usage

```
removeAllMissingCatVar(cat.varname,  mxtrModel)
```

Arguments

`cat.varname`	A character string indicating a categorical predictor variable in the mixture model.
`mxtrModel`	A mixture model formula of class formula.

Value

Returns a mixture model formula of class formula with the variable indicated in cat.varname omitted.

Author(s)

Michael Nodzenski

Examples

```
#specifiy mixture model
mix.model<- ~ var1 + var2 | var1 + var2

#remove var1
removeAllMissingCatVar("var1", mix.model)
```

`removeMissingLevels` *A function to remove levels of categorical variables with completely missing outcome data from data frames or matrices.*

Description

This function removes levels of categorical variables with completely missing outcome data from data frames or matrices.

Usage

```
removeMissingLevels(missing.levels.list,  dataset)
```

Arguments

`missing.levels.list`
: A list, output by function idMissingLevels, containing the specific levels of categorical variables that are missing all outcome data.

`dataset`
: A data frame containing the variables specified in yname and missing.levels.list.

Value

Returns a data frame with the levels indicated in missing.levels.list removed.

Author(s)

Michael Nodzenski

Examples

```
#create example analysis data
data(euMetabCData)
example.data<-euMetabCData
example.data[example.data$batch==1, "aspartic_acid"]<-NA

#check to determine if aspartic acid values are entirely missing for
#any level of batch or pheno
missing.levels.check<-lapply( "aspartic_acid", anyMissingLevels, cat.vars=c("pheno", "batch"), dataset=examp
names(missing.levels.check)<-"aspartic_acid"
missing.levels.check

#find the specific missing level
missing.levels<-idMissingLevels( "aspartic_acid", missing.levels.check, example.data)

#remove the missing level
cleaned.data<-removeMissingLevels(missing.levels, example.data)
```

`runMxtrmod` *A function to return optimized parameter estimates and the negative log-likelihood of mixture models for truncated normal or lognormal data*

Description

This function returns optimized parameter estimates and the negative log-likelihood of mixture models for truncated normal or lognormal data.The function does not take into account factor variable predictors with entirely missing outcome data, and therefore should not be used outside the mxtrmod function.

Usage

```
runMxtrmod(ynames,mxtrModel,Tvals=NULL,nNA=5,minProp=0.2,method="BFGS",data,fullModel=NULL)
```

Arguments

`ynames`	A character vector of the mixture model outcome names, e.g. metabolites. If the input data object is a matrix or data frame, these should be column names. If the input data object is an expression set, these should be row names. Response variables should have normal or lognormal distributions. If lognormal, log transformed variables should be input. Missing values should be denoted by NA.
`mxtrModel`	A formula of the form ~x1+x2...|z1+z2..., where x's are the names of covariates included in the discrete portion of the model and z's are names of covariates included in the continuous portion. For intercept only models, enter 1 instead of covariate names on the appropriate side of the |.
`Tvals`	A vector of thresholds below which continuous variables are not observable. By default, this parameter will be set to the minimum of the response variable.
`nNA`	The minimum number of unobserved values needed to be present for the discrete portion of the model likelihood to be calculated. Models for variables with fewer than nNA missing values will include only the continuous portion. The default value is 5.
`minProp`	The minimum proportion of non-missing data in the response variable necessary to run the model. The default value is 0.2. Models will not be run if more than 80% of response variable values are missing.
`method`	The method used to optimize the parameter estimates of the mixture model. "BFGS" is the default method. Other options are documented in the manual for the function 'optimx' in package optimx.
`data`	The input data object. Matrices, data frames, and expression sets are all acceptable classes. If a data frame or matrix, rows are subjects and columns are metabolites or outcomes.
`fullModel`	A formula of the form ~x1+x2...|z1+z2..., where x's are the names of covariates included in the discrete portion of the full model and z's are names of covariates included in the continuous portion. Input if the mxtrModel parameter represents a reduced model.

Value

Returns a data frame containing optimized estimates for all parameters in the mixture model, the negative log likelihood of the model, the optimization method used, whether the algorithm converged, and the total number of observations used.

Note

This function may generate warning messages about production of NaNs, but the function is still operating normally.

Author(s)

Michael Nodzenski, Anna Reisetter, Denise Scholtens

References

Moulton LH, Halsey NA. A mixture model with detection limits for regression analyses of antibody response to vaccine. Biometrics. 1995 Dec;51(4):1570-8.

Examples

```
#Create sample data frame
set.seed(123)
yvar<-rlnorm(200)
these<-sample(1:100,20)
yvar[these]<-NA
logyvar<-log(yvar)
y2var<-rlnorm(200)
those<-sample(1:200,25)
y2var[those]<-NA
logy2var<-log(y2var)
pred1<-sample(0:1,200,replace=TRUE)
pred2<-sample(1:10,200,replace=TRUE)
pred3<-sample(0:1,200,replace=TRUE)
pred3miss<-sample(1:200,50)
pred3[pred3miss]<-NA
testdata<-data.frame(cbind(yvar,y2var,logyvar,logy2var,pred1,pred2,pred3))

#Get the names of the response variables
ynames<-names(testdata)[3]

#Run a mixture model on each response variable
mod<-~pred1+pred2+pred3|pred1+pred2+pred3
runMxtrmod(ynames=ynames,mxtrModel=mod,data=testdata)

#Create example expression set
#Specify the response variables
exprsobs<-t(testdata[,3:4])

#Specify the phenotype data
exprspheno<-testdata[,5:7]

#make phenotype data an annotated data frame
phenoData <- new("AnnotatedDataFrame",data=exprspheno)

#combine into example expression set
```

```
testexpr<-ExpressionSet(assayData=exprsobs,phenoData=phenoData)

#Get the names of the response variable
ynames<-rownames(exprs(testexpr))[1]

#Run the mixture model on the response variable
runMxtrmod(ynames=ynames,mxtrModel=mod,data=testexpr)

#Load the data set from the package
data(metabdata)

#Select the response variable
ynames<-names(metabdata)[11]

#Run the mixture models
mod2<-~PHENO|PHENO+age_ogtt_mc+parity12+ga_ogtt_wks_mc
runMxtrmod(ynames,mxtrModel=mod2,data=metabdata)
```

---

xdesign-methods ~~ *Methods for Function* xdesign ~~

Description

~~ Methods for function xdesign ~~

Methods

`signature(x="data.frame",m="ANY")` The columns of data frame 'x' specified in the input Formula object 'm' are converted to the design matrix for the discrete portion of the mixture model.

`signature(x="ExpressionSet",m="ANY")` The columns of the phenoData section of expression set 'x' specified in the input Formula object 'm' are converted to the design matrix for the discrete portion of the mixture model.

`signature(x="matrix",m="ANY")` The columns of matrix 'x' specified in the input Formula object 'm' are converted to the design matrix for the discrete portion of the mixture model.

---

yvals-methods ~~ *Methods for Function* yvals ~~

Description

~~ Methods for function yvals ~~

Methods

`signature(y="data.frame",n="character")` The columns of data frame 'y' with names 'n' are converted to a data frame, whose columns are to be used as the response variables in the specified mixture model.

`signature(y="ExpressionSet",n="character")` The rows of the assayData section of expression set 'y' with names 'n' are converted to a data frame, whose columns are to be used as the response variables in the specified mixture model.

`signature(y="matrix",n="character")` The columns of matrix 'y' with names 'n' are converted to a data frame, whose columns are to be used as the response variables in the specified mixture model.

`zdesign-methods` *~~ Methods for Function* `zdesign` *~~*

Description

~~ Methods for function `zdesign` ~~

Methods

`signature(x="data.frame",m="ANY")` The columns of data frame 'x' specified in the input Formula object 'm' are converted to the design matrix for the continuous portion of the mixture model.

`signature(x="ExpressionSet",m="ANY")` The columns of the phenoData section of expression set 'x' specified in the input Formula object 'm' are converted to the design matrix for the continuous portion of the mixture model.

`signature(x="matrix",m="ANY")` The columns of matrix 'x' specified in the input Formula object 'm' are converted to the design matrix for the continuous portion of the mixture model.

Index

∗ **datasets**
euMetabCData, 8
euMetabData, 8
metabdata, 10
∗ **methods**
xdesign-methods, 28
yvals-methods, 28
zdesign-methods, 29
∗ **package**
metabomxtr-package, 2

addBatchMeans, 4
addOutlierInfo, 5
allMissingLevels, 6
anyMissingLevels, 7

euMetabCData, 8
euMetabData, 8

idMissingLevels, 9

metabdata, 10
metabomxtr *(metabomxtr-package)*, 2
metabomxtr-package, 2
metabplot, 12
mixnorm, 14
mxtrmod, 17
mxtrmodLL, 19
mxtrmodLRT, 21
mxtrmodstart, 22

removeAllMissingCatVar, 24
removeMissingLevels, 25
runMxtrmod, 26

xdesign *(xdesign-methods)*, 28
xdesign,data.frame-method *(xdesign-methods)*, 28
xdesign,ExpressionSet-method *(xdesign-methods)*, 28
xdesign,matrix-method *(xdesign-methods)*, 28
xdesign-methods, 28

yvals *(yvals-methods)*, 28
yvals,data.frame,character-method *(yvals-methods)*, 28
yvals,ExpressionSet,character-method *(yvals-methods)*, 28
yvals,matrix,character-method *(yvals-methods)*, 28
yvals-methods, 28

zdesign *(zdesign-methods)*, 29
zdesign,data.frame-method *(zdesign-methods)*, 29
zdesign,ExpressionSet-method *(zdesign-methods)*, 29
zdesign,matrix-method *(zdesign-methods)*, 29
zdesign-methods, 29