Package ‘combi’

January 7, 2024

Type Package

Title Compositional omics model based visual integration

Version 1.14.0

Description This explorative ordination method combines quasi-likelihood estimation, compositional regression models and latent variable models for integrative visualization of several omics datasets. Both unconstrained and constrained integration are available. The results are shown as interpretable, compositional multiplots.

License GPL-2

Encoding UTF-8

Depends R (>= 4.0), DBI

RoxygenNote 7.2.3

Imports ggplot2, nleqslv, phyloseq, tensor, stats, limma, Matrix (>= 1.6.0), BB, reshape2, alabama, cobs, Biobase, vegan, grDevices, graphics, methods, SummarizedExperiment

Suggests knitr, rmarkdown, testthat

VignetteBuilder knitr

biocViews Metagenomics, DimensionReduction, Microbiome, Visualization, Metabolomics

BugReports https://github.com/CenterForStatistics-UGent/combi/issues

git_url https://git.bioconductor.org/packages/combi
git_branch RELEASE_3_18
git_last_commit 79a4a97
git_last_commit_date 2023-10-24

Repository Bioconductor 3.18

Date/Publication 2024-01-07

Author Stijn Hawinkel [cre, aut] (<https://orcid.org/0000-0002-4501-5180>)

Maintainer Stijn Hawinkel <stijn.hawinkel@psb.ugent.be>
R topics documented:

addLink ......................................................... 3
arrayMult ....................................................... 4
buildCentMat .................................................. 5
buildCompMat ............................................... 5
buildConfMat ............................................... 6
buildCovMat ............................................... 7
buildEmptyJac ............................................. 7
buildMarginalOffset ................................. 8
buildMu ....................................................... 9
buildMuMargins ........................................... 9
buildOffsetModel ........................................ 10
checkAlias .................................................. 10
checkMeanVarTrend ......................................... 11
checkMonotonicity ......................................... 11
combi .......................................................... 12
covPlot ........................................................ 14
deriv2LagrangianFeatures ......................... 15
deriv2LagrangianLatentVars ..................... 16
deriv2LagrangianLatentVarsConstr ............ 17
derivLagrangianFeatures ......................... 18
derivLagrangianLatentVars ..................... 19
derivLagrangianLatentVarsConstr ............ 20
estFeatureParameters ............................... 21
estIndepModel ........................................ 23
estLatentVars ........................................... 24
estMeanVarTrend ........................................ 24
estOff ......................................................... 25
extractCoords ............................................. 26
extractData ................................................ 27
extractMat ................................................ 28
filterConfounders ....................................... 28
getInflLatentVar ....................................... 29
gramSchmidtOrth ......................................... 30
indentPlot .................................................. 30
inflPlot ...................................................... 31
influence.combi .......................................... 32
jacConfounders .......................................... 33
jacConfoundersComp .................................. 34
jacFeatures ............................................... 34
jacLatentVars ........................................... 35
jacLatentVarsConstr .................................. 36
plot.combi ................................................. 37
polyHorner .................................................. 39
predictSpline ............................................. 40
prepareJacMat ........................................... 41
prepareJacMatComp ................................. 42
**addLink**

Add a link on a compositional plot

**Description**

Add a link on a compositional plot

**Usage**

```r
addLink(
  DIplot,
  links,
  Views,
  samples,
  variable = NULL,
  Dims = c(1, 2),
  addLabel = FALSE,
  labPos = NULL,
  projColour = "grey",
  latentSize = 0.25
)
```

**Arguments**

- **DIplot**: A list with ggplot object where the links are to be added, and data frames with coordinates (obtained by setting plot(..., returnCoords = TRUE))
- **links**: A matrix containing either feature names (two column matrix) or approximate coordinates (four column matrix)
- **Views**: Indices or names of the views for which the links should be added
arrayMult

Array multiplication

Description

Array multiplication

Usage

arrayMult(centralMat, outerMat, ncols = ncol(outerMat))

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>centralMat</td>
<td>an nxp matrix</td>
</tr>
<tr>
<td>outerMat</td>
<td>an nxd matrix</td>
</tr>
<tr>
<td>ncols</td>
<td>an integer, the number of columns of outerMat</td>
</tr>
</tbody>
</table>

Value

an nxpxd matrix, the stacked matrices of centralMat multiplied to every column of outerMat

dsamples     Sample names or approximate sample coordinates
variable     Name of variable in environmental gradient for which link should be plotted
Dims         vector of length 2 referring to the model dimensions
addLabel     A boolean, should arrow with label be plotted?
labPos       The position of the label, as a numeric vector of length 2
projColour   The colour of the projection, as character string
latentSize   Size of the line from the origin to the latent variable dot

Value

A ggplot object with the links added

Examples

data(Zhang)

## Not run:
#Unconstrained
microMetaboInt = combi(
  list("microbiome" = zhangMicrobio, "metabolomics" = zhangMetabo),
  distributions = c("quasi", "gaussian"), compositional = c(TRUE, FALSE),
  logTransformGaussian = FALSE, verbose = TRUE)

## End(Not run)
load(system.file("extdata", "zhangFits.RData", package = "combi"))
Plot = plot(microMetaboInt, samDf = zhangMetavars, samCol = "ABX",
            returnCoords = TRUE)
addLink(Plot, links = cbind("OTU0565b3","OTUa14fb5"), Views = 1,
         samples = c(1,1))
buildCentMat  

A function to build a centering matrix based on a dataframe

Description

A function to build a centering matrix based on a dataframe

Usage

buildCentMat(object)

Arguments

object  an modelDI object or dataframe

Value

a centering matrix consisting of ones and zeroes, or a list with components

centMat  a centering matrix consisting of ones and zeroes

datFrame  The dataframe with factors with one level removed

buildCompMat  

Build the composition matrix for a certain dimension \( m \) dimensions

Description

Build the composition matrix for a certain dimension \( m \) dimensions

Usage

buildCompMat(
  colMat,
  paramEsts,
  latentVar,
  m,
  norm = TRUE,
  id = seq_len(m),
  subtractMax = TRUE
)

buildConfMat

Arguments

- `colMat`: The nxp independence model composition matrix
- `paramEsts`: The matrix of feature parameter estimates
- `latentVar`: The matrix of latent variables
- `m`: The required dimension
- `norm`: A boolean, should the composition matrix be normalized?
- `id`: The vector of dimensions to consider
- `subtractMax`: A boolean, should the maximum be subtracted from every composition prior to exponentiation? Recommended for numerical stability

Value

A matrix with compositions in the rows

Description

Build confounder design matrices with and without intercepts

Usage

`buildConfMat(confounders)`

Arguments

- `confounders`: A dataframe of confounding variables

Value

A list with components

- `confModelMatTrim`: A confounder matrix without intercept, with all levels of factors present. This will be used to trim out taxa that have zero abundances in any subgroup defined by confounders
- `confModelMat`: A confounder matrix with intercept, and with reference levels for factors absent. This will be used to fit the model to modify the independence model, and may include continuous variables
buildCovMat

A function to build the covariate matrix of the constraints

Description

A function to build the covariate matrix of the constraints

Usage

buildCovMat(datFrame)

Arguments

datFrame
the dataframe with which the covariate matrix is to be built

In this case we will 1) Include dummy's for every level of the categorical variable, and force them to sum to zero. This is needed for plotting and required for reference level independent normalization. 2) Exclude an intercept. The density function f() will provide this already.

Value

a list with components

covModelMat The model matrix
datFrame The dataframe used to construct the model matrix

buildEmptyJac

Prepare an empty Jacobian matrix, with useful entries prefilled. In case of distribution "gaussian", it returns the lhs matrix of the linear system for finding the feature parameters

Description

Prepare an empty Jacobian matrix, with useful entries prefilled. In case of distribution "gaussian", it returns the lhs matrix of the linear system for finding the feature parameters

Usage

buildEmptyJac(
  n,
  m,
  lower,
  distribution = "quasi",
  normal = FALSE,
  nLambdas = 1,
  centMat = NULL,
  weights = 1
)

)
Arguments

n the number of parameters
m the dimension
lower the current parameter estimates
distribution A character string, the distributional assumption for the data
normal a boolean, are normalization restrictions in place?
nLambdas The number of centering restrictions
centMat The centering matrix
weights Vector of feature weights

Value

an empty jacobian matrix, or the lhs of the system of estimating equations

Description

Build an offset matrix from an marginal model object

Usage

buildMarginalOffset(indepModel, invLink)

Arguments

indepModel The fitted marginal model, a list
invLink The inverse link function

Value

an offset matrix of the size of the data
buildMu

A function to build the mu matrix

Description

A function to build the mu matrix

Usage

buildMu(offSet, latentVar, paramEsts, distribution, paramMatrix = FALSE)

Arguments

offSet the offset matrix
latentVar, paramEsts, distribution Latent variables, parameter estimates and distribution type
paramMatrix A boolean, are feature parameters provided as matrix

Value

The mean matrix

buildMuMargins Build the marginal mu matrix

Description

Build the marginal mu matrix

Usage

buildMuMargins(x, otherMargin, col)

Arguments

x The marginal parameters begin estimated
otherMargin The parameters of the other margin
col A logical, are the column parameters being estimated?

Value

a matrix of means
buildOffsetModel  

*Build a marginal offset matrix given a model*

**Description**

Build a marginal offset matrix given a model

**Usage**

`buildOffsetModel(modelObj, View, distributions, compositional)`

**Arguments**

- `modelObj`: a modelDI object
- `View`: The view for which to build the offset
- `distributions, compositional`: belong to the view

**Value**

The offset matrix

---

checkAlias  

*Check for alias structures in a dataframe, and throw an error when one is found*

**Description**

Check for alias structures in a dataframe, and throw an error when one is found

**Usage**

`checkAlias(datFrame, covariatesNames)`

**Arguments**

- `datFrame`: the data frame to be checked for alias structure
- `covariatesNames`: The names of the variables to be considered

**Value**

Throws an error when an alias structure is detected, returns invisible otherwise
checkMeanVarTrend

**Description**

Quickly check if the mean variance trend provides a good fit

**Usage**

```r
cHECKMeanVarTrend(data, meanVarFit = "spline", returnTrend = FALSE, ...)
```

**Arguments**

- `data` - Data in any acceptable format (see details ?combi)
- `meanVarFit` - The type of mean variance fit, either "cubic" or "spline"
- `returnTrend` - A boolean, should the estimated trend be returned (TRUE) or only plotted (FALSE)?
- `...` - passed on to the estMeanVarTrend() function

**Value**

A plot object

**Examples**

```r
data(Zhang)
par(mfrow = c(1,2))
lapply(list("microbiome" = zhangMicrobio, "metabolome" = zhangMetabo),
      checkMeanVarTrend)
par(mfrow = c(1,1))
```

checkMonotonicity

**Description**

Check for monotonicity in compositional datasets for given dimensions

**Usage**

```r
cHECKMonotonicity(modelObj, Dim)
```

**Arguments**

- `modelObj` - The combi fit
- `Dim` - The dimensions considered
Value

A boolean matrix indicating monotonicity for every feature

Description

Perform model-based data integration

Usage

combi(
  data,
  M = 2L,
  covariates = NULL,
  distributions,
  compositional,
  maxIt = 300L,
  tol = 0.001,
  verbose = FALSE,
  prevCutoff = 0.95,
  minFraction = 0.1,
  logTransformGaussian = TRUE,
  confounders = NULL,
  compositionalConf = rep(FALSE, length(data)),
  nleq.control = list(maxit = 1000L, cndtol = 1e-16),
  record = TRUE,
  weights = NULL,
  fTol = 1e-05,
  meanVarFit = "spline",
  maxFeats = 2000,
  dispFreq = 10L,
  allowMissingness = FALSE,
  biasReduction = TRUE,
  maxItFeat = 20L,
  initPower = 1
)

Arguments

data A list of data objects with the same number of samples. See details.
M the required dimension of the fit, a non-negative integer
covariates a dataframe of n samples with sample-specific variables.
distributions a character vector describing which distributional assumption should be used. See details.
compositional  A logical vector with the same length as "data", indicating if the datasets should be treated as compositional
maxIt  an integer, the maximum number of iterations
tol  A small scalar, the convergence tolerance
verbose  Logical. Should verbose output be printed to the console?
prevCutOff  a scalar, the prevalence cutoff for the trimming.
minFraction  a scalar, each taxon’s total abundance should equal at least the number of samples n times minFraction, otherwise it is trimmed.
logTransformGaussian  A boolean, should the gaussian data be logtransformed, i.e. are they log-normal?
confounders  A dataframe or a list of dataframes with the same length as data. In the former case the same dataframe is used for conditioning, In the latter case each view has its own conditioning variables (or NULL).
compositionalConf  A logical vector with the same length as "data", indicating if the datasets should be treated as compositional when correcting for confounders. Numerical problems may occur when set to TRUE
nleq.control  A list of arguments to the nleqslv function
record  A boolean, should intermediate estimates be stored? Can be useful to check convergence
weights  A character string, either 'marginal' or 'uniform', indicating how the feature parameters should be weighted in the normalization
fTol  The tolerance for solving the estimating equations
meanVarFit  The type of mean variance fit, see details
maxFeats  The maximal number of features for a Newton-Raphson procedure to be feasible
dispFreq  An integer, the period after which the variances should be reestimated
allowMissingness  A boolean, should NA values be allowed?
biasReduction  A boolean, should bias reduction be applied to allow for confounder correction in groups with all zeroes? Not guaranteed to work
maxItFeat  Integers, the maximum allowed number of iterations in the estimation of the feature parameters
initPower  The power to be applied to the residual matrix used to calculate the starting value. Must be positive; can be tweaked in case of numerical problems (i.e. infinite values returned by nleqslv)

Details

Data can be provided as raw matrices with features in the columns, or as phyloseq, SummarizedExperiment or ExpressionSet objects. Estimation of independence model and view wise parameters can be parametrized. See ?BiocParallel::bplapply and ?BiocParallel::register. meanVarFit = "spline" yields a cubic spline fit for the abundance-variance trend, "cubic" gives a third degree polynomial. Both converge to the diagonal line with slope 1 for small means. Distribution can be either "quasi" for quasi likelihood or "gaussian" for Gaussian data.
Value

An object of the "combi" class, containing all information on the data integration and fitting procedure.

Examples

data(Zhang)

# The method works on several datasets at once, and simply is not very fast.
# Hence the "Not run" statement
## Not run:
## Unconstrained
microMetaboInt = combi(
  list("microbiome" = zhangMicrobio, "metabolomics" = zhangMetabo),
  distributions = c("quasi", "gaussian"), compositional = c(TRUE, FALSE),
  logTransformGaussian = FALSE, verbose = TRUE)
## Constrained
microMetaboIntConstr = combi(
  list("microbiome" = zhangMicrobio, "metabolomics" = zhangMetabo),
  distributions = c("quasi", "gaussian"), compositional = c(TRUE, FALSE),
  logTransformGaussian = FALSE, covariates = zhangMetavars, verbose = TRUE)
## End(Not run)

convPlot

Plot the convergence of the different parameter estimates in a line plot

Description

Plot the convergence of the different parameter estimates in a line plot

Usage

convPlot(
  model,
  latent = is.null(View),
  nVars = Inf,
  Dim = 1L,
  View = NULL,
  size = 0.125
)

Arguments

model A fitted modelDI object
latent A boolean, should latent variable trajectory be plotted
nVars An integer, the number of variables to plot. By default all are plotted
Dim An integer, the dimension to be plotted
deriv2LagrangianFeatures

View
An integer or character string, indicating the view to be plotted (if latent = FALSE)
size
The line size (see ?geom_path)

Value
A ggplot object containing the convergence plot

Examples
## Not run:
data(Zhang)
#Unconstrained
microMetaboInt = combi(
list("microbiome" = zhangMicrobio, "metabolomics" = zhangMetabo),
distributions = c("quasi", "gaussian"), compositional = c(TRUE, FALSE),
logTransformGaussian = FALSE, verbose = TRUE)
## End(Not run)
load(system.file("extdata", "zhangFits.RData", package = "combi"))
convPlot(microMetaboInt)
convPlot(microMetaboInt, Dim = 2)
convPlot(microMetaboInt, View = "microbiome")

deriv2LagrangianFeatures

Description
The score function to estimate the latent variables

Usage
deriv2LagrangianFeatures(
  x,
  data,
  distribution,
  offSet,
  latentVars,
  numVar,
  paramEstsLower,
  mm,
  Jac,
  meanVarTrend,
  weights,
  compositional,
  indepModel,
  ...
)
Arguments

\text{deriv2LagrangianLatentVars} \text{ arguments and their descriptions}

\text{Arguments}

- \text{x} \quad \text{parameter estimates}
- \text{data} \quad \text{A list of data matrices}
- \text{distribution}, \text{compositional}, \text{meanVarTrend}, \text{offSet}, \text{numVar} \quad \text{Characteristics of the view}
- \text{latentVars} \quad \text{A vector of latent variables}
- \text{paramEstsLower} \quad \text{lower dimension estimates}
- \text{mm} \quad \text{the current dimension}
- \text{Jac} \quad \text{a prefab jacobian}
- \text{weights} \quad \text{The normalization weights}
- \text{indepModel} \quad \text{the independence model}
- \text{...} \quad \text{Additional arguments passed on to the score and jacobian functions}

\text{Value}

\text{A vector of length n, the evaluation of the score functions of the latent variables}

\text{Usage}

\text{deriv2LagrangianLatentVars}(
\quad \text{x},
\quad \text{data},
\quad \text{distributions},
\quad \text{offsets},
\quad \text{paramEsts},
\quad \text{paramMats},
\quad \text{numVar},
\quad \text{latentVarsLower},
\quad \text{n},
\quad \text{m},
\quad \text{Jac},
\quad \text{numSets},
\quad \text{meanVarTrends},
\quad \text{links},
\quad \text{varPosts},
\quad \text{indepModels},
\quad \text{compositional},
\quad \text{...}
\)
Arguments

x The current estimates of the latent variables distributions, links, compositional, data, meanVarTrends, offsets, numVars, numSets, paramMats, paramEsts latentVarsLower Characteristics of the views The parameter estimates of the lower dimensions n, m integers, number of samples and dimensions Jac an empty jacobian matrix ... arguments to the jacobian function, currently ignored

Value

A vector of length n, the evaluation of the score functions of the latent variables

Description

The score function to estimate the latent variables

Usage

deriv2LagrangianLatentVarsConstr( x, data, distributions, offsets, paramEsts, paramMats, numVars, latentVarsLower, nn, m, Jac, numSets, meanVarTrends, links, numCov, covMat, nLambda1s, varPosts, compositional, indepModels, ... )
Arguments

- **x**: The current estimates of the latent variables distributions, data, links, compositional, meanVarTrends, offsets, numVars, paramMats, paramEsts
- **latentVarsLower**: Characteristics of the view
  - The parameter estimates of the lower dimensions
- **nn**: number of samples
- **m, numSets, varPosts, indepModels**: other arguments
- **Jac**: an empty jacobian matrix
- **numCov**: The number of covariates
- **covMat**: the covariates matrix
- **nLambda1s**: The number of centering restrictions
- **...**: arguments to the jacobian function, currently ignored

Value

A vector of length nn, the evaluation of the score functions of the latent variables

---

**derivLagrangianFeatures**

*The score function to estimate the feature parameters*

---

Description

The score function to estimate the feature parameters

Usage

def derivLagrangianFeatures(
    x, data, distribution, offSet, latentVars, numVar, paramEstsLower, mm, indepModel, meanVarTrend, weights, compositional,
    ...
)
)
**derivLagrangianLatentVars**

**Arguments**

- `x`: current parameter estimates
- `data`: A list of data matrices
distribution, compositional, meanVarTrend, offSet, numVar, indepModel, paramEstsLower
- `latentVars`: A vector of latent variables
- `mm`: the current dimension
- `weights`: The normalization weights
- `...`: arguments to the jacobian function, currently ignored

**Value**

A vector with the evaluation of the score functions of the feature parameters

**Description**

The score function to estimate the latent variables

**Usage**

```
derivLagrangianLatentVars(
  x,
  data,
  distributions,
  offsets,
  paramEsts,
  paramMats,
  numVars,
  n,
  m,
  numSets,
  meanVarTrends,
  links,
  varPosts,
  latentVarsLower,
  compositional,
  indepModels,
  ...
)
```
Arguments

- **x**: The current estimates of the latent variables
- **n**: The number of samples
- **m**: The dimensions
- **numSets**: The number of views
- **latentVarsLower**: The parameter estimates of the lower dimensions

...additional arguments to the jacobian function, currently ignored

Value

A vector of length `n`, the evaluation of the score functions of the latent variables

---

**derivLagrangianLatentVarsConstr**

*The score function to estimate the latent variables*

---

Description

The score function to estimate the latent variables

Usage

```r
derivLagrangianLatentVarsConstr(
  x,  
  data,  
  distributions,  
  offsets,  
  paramEsts,  
  numVars,  
  latentVarsLower,  
  n,  
  m,  
  numSets,  
  meanVarTrends,  
  links,  
  covMat,  
  numCov,  
  centMat,  
  nLambdas,  
  varPosts,  
  compositional,  
  indepModels,  
  ...  
)```
**estFeatureParameters**

```r
estFeatureParameters(
  paramMats, 
  ... 
)
```

### Arguments

- `x` The current estimates of the latent variables
- `latentVarsLower` The parameter estimates of the lower dimensions
- `n` The number of samples
- `m` The dimensions
- `numSets` The number of views
- `covMat` The covariance matrix
- `numCov` The number of covariates
- `centMat` A centering matrix
- `nLambda1s` The number of dummy variables
- `compositional, links, indepModels, meanVarTrends, numVars, distributions, data, offsets, varPosts, paramMats, paramEsts` Lists of information on all the views
- `...` arguments to the jacobian function, currently ignored

### Value

A vector of length n, the evaluation of the score functions of the latent variables

---

**estFeatureParameters** *Estimate the feature parameters*

### Description

Estimate the feature parameters

### Usage

```r
estFeatureParameters(
  paramEsts, 
  lambdasParams, 
  seqSets, 
  data, 
  distributions, 
  offsets, 
  nCores, 
  m, 
  JacFeatures, 
  meanVarTrends, 
  latentVars, 
```
estFeatureParameters

numVars,
control,
weights,
compositional,
indepModels,
fTol,
allowMissingness,
maxItFeat,
...
)

Arguments

paramEsts Current list of parameter estimates for the different views
lambdasParams The lagrange multipliers
seqSets A vector with view indices
data A list of data matrices
distributions A character vector describing the distributions
offsets A list of offset matrices
nCores The number of cores to use in multithreading
m The dimension
JacFeatures An empty Jacobian matrix
meanVarTrends The mean-variance trends of the different views
latentVars A vector of latent variables
numVars The number of variables
control A list of control arguments for the nleqslv function
weights The normalization weights
compositional A list of booleans indicating compositionality
indepModels A list of independence model
fTol A convergence tolerance
allowMissingness A boolean indicating whether missing values are allowed
maxItFeat An integer, the maximum number of iterations
... Additional arguments passed on to the score and jacobian functions

Value

A vector with estimates of the feature parameters
estIndepModel

Estimate the independence model belonging to one view

Description

Estimate the independence model belonging to one view

Usage

estIndepModel(
  data,
  distribution,
  compositional,
  maxIt,
  tol,
  link,
  invLink,
  meanVarFit,
  newtonRaphson,
  dispFreq,
  ...
)

Arguments

data  a list of data matrices with the same number of samples n in the rows. Also phyloseq objects are acceptable
distribution  a character string describing which distributional assumption should be used.
compositional  A logical indicating if the dataset should be treated as compositional
maxIt  an integer, the maximum number of iterations
tol  A small scalar, the convergence tolerance
link, invLink  link and inverse link function
meanVarFit  mean variance model
newtonRaphson  a boolean, should newton-raphson be used
dispFreq  An integer, frequency of dispersion estimation
...  passed on to the estOff() function

Value

A list with elements

rowOff  The row offsets
colOff  The column offsets
converged  A logical flag, indicating whether the fit converged
iter  An integer, the number of iterations
estLatentVars  Estimate the latent variables

Description
Estimate the latent variables

Usage
estLatentVars(latentVars, lambdasLatent, constrained, fTol, ...)

Arguments
- latentVars: A vector of latent variables
- lambdasLatent: A vector of Lagrange multipliers
- constrained: A boolean, is the ordination constrained?
- fTol: The convergence tolerance
- ...: additional arguments passed on to score and jacobian functions

Value
A vector of length n, the estimates of the latent variables

estMeanVarTrend  Estimate a column-wise mean-variance trend

Description
Estimate a column-wise mean-variance trend

Usage
estMeanVarTrend(
    data,
    meanMat,
    baseAbundances,
    libSizes,
    plot = FALSE,
    meanVarFit,
    degree = 2L,
    constraint = "none",
    ...
)
Arguments

- **data**: the data matrix with n rows
- **meanMat**: the estimated mean matrix
- **baseAbundances**: The baseline abundances
- **libSizes**: Library sizes
- **plot**: A boolean, should the trend be plotted?
- **meanVarFit**: A character string describing the type of trend to be fitted: either "spline" or "cubic"
- **degree**: The degree of the spline
- **constraint**: Constraint to the spline
- **...**: additional arguments passed on to the plot() function

Value

A list with components

- **meanVarTrend**: An smoothed trend function, that can map a mean on a variance
- **meanVarTrendDeriv**: A derivative function of this

---

**estOff**  
*Estimate the row/column parameters of the independence model*

Description

Estimate the row/column parameters of the independence model

Usage

```r
estOff(
  data,
  distribution,
  rowOff,
  colOff,
  meanVarTrend,
  col,
  newtonRaphson,
  libSizes,
  ...
)
```
Arguments

data a list of data matrices with the same number of samples n in the rows. Also phyloseq objects are acceptable
distribution a character string describing which distributional assumption should be used.
rowOff, colOff current row and column offset estimates
meanVarTrend The estimated mean-variance trend
col A logical, should column offsets be estimated
newtonRaphson A boolean, should Newton-Raphson be used to solve the estimating equations
libSizes The library sizes, used to evaluate the mean-variance trend
... passed onto nleqslv

Value

The estimated marginal parameters

extractCoords Extract coordinates from fitted object

Description

Extract coordinates from fitted object

Usage

extractCoords(modelObj, Dim)

Arguments

modelObj The fitted model
Dim the required dimensions

Value

A list with components (matrices with two columns)

latentData The latent variables
featureData The feature parameters
varData The variables
Examples

```r
data(Zhang)
## Not run:
#Unconstrained
microMetaboInt = combi(
  list("microbiome" = zhangMicrobio, "metabolomics" = zhangMetabo),
  distributions = c("quasi", "gaussian"), compositional = c(TRUE, FALSE),
  logTransformGaussian = FALSE, verbose = TRUE)
## End(Not run)

#Load the fits
load(system.file("extdata", "zhangFits.RData", package = "combi"))
extractCoords(microMetaboInt, Dim = c(1, 2))
```

extractData

```
Helper function to extract data matrix from phyloseq, expressionset objects etc. Also filters out all zero rows
```

Description

Helper function to extract data matrix from phyloseq, expressionset objects etc. Also filters out all zero rows.

Usage

```
extractData(data, logTransformGaussian = TRUE)
```

Arguments

- **data**: The list of data objects, either matrix, phyloseq or ExpressionSet objects
- **logTransformGaussian**: A boolean, should array data be logtransformed

Value

The raw data matrices, samples in the rows

Examples

```
data(Zhang)
matrixList = extractData(list("microbiome" = zhangMicrobio,
  "metabolome" = zhangMetabo))
```
extractMat

A function to extract a data matrix from a number of objects

Description

A function to extract a data matrix from a number of objects

Usage

extractMat(Y, ...)

## S4 method for signature 'ExpressionSet'
extRACTMat(Y, logTransformGaussian, ...)

## S4 method for signature 'SummarizedExperiment'
extRACTMat(Y, ...)

## S4 method for signature 'matrix'
extRACTMat(Y, ...)

Arguments

Y
  a phyloseq or eSet object, or another object, or a raw data matrix

... additional arguments for the extractor function

logTransformGaussian
  A boolean, should array data be logtransformed

Value

A data matrix with samples in the rows and features in the columns

filterConfounders

Filter out the effect of known confounders

Description

Filter out the effect of known confounders
Usage

filterConfounders(
    confMat,  
data,    
distribution,    
link,    
invLink,    
compositional,    
control,    
meanVarTrend,    
offSet,    
numVar,    
marginModel,    
biasReduction,    
allowMissingness
)

Arguments

confMat  A confounder design matrix
data  data matrix
distribution, link, invLink, compositional, meanVarTrend, offSet, numVar, marginModel  Characteristics of the view
control  A list of control elements to the nleqslv function
biasReduction  A boolean, should bias reduction be applied
allowMissingness  A boolean, are missing values allowed?

Value

Parameter estimates accounting for the effects of the confounders

getInflLatentVar  Extract the influence on the estimation of the latent variable

description

Extract the influence on the estimation of the latent variable

Usage

getInflLatentVar(score, InvJac, i)
Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>score</td>
<td>The score matrix</td>
</tr>
<tr>
<td>InvJac</td>
<td>The inverse Jacobian</td>
</tr>
<tr>
<td>i</td>
<td>the sample index</td>
</tr>
</tbody>
</table>

Value

The influence of all observations on the i-th latent variable

**gramSchmidtOrth**

*Gram schimdt orhtogonalize a with respect to b, and normalize*

Description

Gram schimdt orhtogonalize a with respect to b, and normalize

Usage

```r
gramSchmidtOrth(a, b, weights = 1, norm = TRUE)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>the vector to be orthogonalized</td>
</tr>
<tr>
<td>b</td>
<td>the vector to be orthogonalized to</td>
</tr>
<tr>
<td>weights</td>
<td>weights vector</td>
</tr>
<tr>
<td>norm</td>
<td>a boolean, should the result be normalized?</td>
</tr>
</tbody>
</table>

Value

The orthogonalized vector

**indentPlot**

*Functions to indent the plot to include the entire labels*

Description

Functions to indent the plot to include the entire labels

Usage

```r
indentPlot(plt, xInd = 0, yInd = 0)
```
Arguments

plt a ggplot object

xInd a scalar or a vector of length 2, specifying the indentation left and right of the plot to allow for the labels to be printed entirely

yInd a a scalar or a vector of length 2, specifying the indentation top and bottom of the plot to allow for the labels to be printed entirely

Value

a ggplot object, squared

Description

A ggplot line plot showing the influences

Usage

inflPlot(
  modelObj,
  plotType = ifelse(length(modelObj$data) <= 2, "pointplot", "boxplot"),
  pointFun = "sum",
  lineSize = 0.07,
  Dim = 1,
  samples = seq_len(nrow(if (is.null(modelObj$covariates)) modelObj$latentVars else modelObj$alphas)),
  ...
)

Arguments

modelObj The fitted data integration
plotType The type of plot requested, see details
pointFun The function to calculate the summary measure to be plotted
lineSize The line size
Dim The dimension required
samples index vector of which samples to be plotted
... additional arguments passed on to the influence() function
Details

The options for plotType are: "pointPlot": Dot plot of total influence per view and sample, "boxplot": plot boxplot of influence of all observations per view and sample, "boxplotSingle": boxplot of log absolute total influence per view, "lineplot": line plot of total influence per view and sample. In the pointplot, dots crosses represent parameter estimates.

Value

A ggplot object

Examples

data(Zhang)
#Unconstrained
microMetaboInt = combi(
  list("microbiome" = zhangMicrobio, "metabolomics" = zhangMetabo),
  distributions = c("quasi", "gaussian"), compositional = c(TRUE, FALSE),
  logTransformGaussian = FALSE, verbose = TRUE)
#Constrained
microMetaboIntConstr = combi(
  list("microbiome" = zhangMicrobio, "metabolomics" = zhangMetabo),
  distributions = c("quasi", "gaussian"), compositional = c(TRUE, FALSE),
  logTransformGaussian = FALSE, covariates = zhangMetavars, verbose = TRUE)
load(system.file("extdata", "zhangFits.RData", package = "combi"))
inflPlot(microMetaboInt)
#Constrained
inflPlot(microMetaboIntConstr)
Details
Especially the influence of the different views on the latent variable or gradient estimation may be of interest. The influence values are not all calculated. Rather, the score values and inverse jacobian are returned so they can easily be calculated.

Value
A list with components

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>score</td>
<td>The evaluation of the score function</td>
</tr>
<tr>
<td>InvJac</td>
<td>The inverted jacobian matrix</td>
</tr>
</tbody>
</table>

jacConfounders

Description
Jacobian when estimating confounder variables

Usage
jacConfounders(
  confMat,  
data,  
distribution,  
x,  
meanVarTrend,  
offSet,  
CompMat,  
libSizes,  
allowMissingness  
)

Arguments

data, confMat, meanVarTrend
  Characteristic of the views

distribution, offSet
  Distribution and offset of the view

x
  The parameter estimates

libSizes, CompMat
  Library sizes and relative abundance

allowMissingness
  A boolean, should missing values be allowed

Value
the jacobian matrix
jacConfoundersComp  Jacobian for conditioning under compositionality

Description

Jacobian for conditioning under compositionality

Usage

jacConfoundersComp(
  x,
  confMat, data, meanVarTrend,
  marginModel, allowMissingness, biasReduction,
  subtractMax = TRUE
)

Arguments

  x  the parameter estimates
confMat, data, meanVarTrend
    arguments belonging to views
marginModel, biasReduction, subtractMax
    The marginal mode, and booleans indicating bias reduction and maximum subtraction
allowMissingness
    a boolean, should missing values be allowed

Value

  the jacobian matrix

jacFeatures  Evaluate the jacobian for estimating the feature parameters for one view

Description

Evaluate the jacobian for estimating the feature parameters for one view
**Usage**

```
jacFeatures(
    latentVars,
    data,
    distribution,
    paramEsts,
    meanVarTrend,
    offSet,
    compositional,
    indepModel,
    m,
    paramEstsLower,
    allowMissingness,
    ...
)
```

**Arguments**

- `latentVars` A vector of latent variables
- `data` A list of data matrices
- `distribution`, `compositional`, `meanVarTrend`, `offSet`, `paramEsts`, `paramEstsLower`, `indepModel` Characteristics of each view
- `m` dimension
- `allowMissingness` a boolean, are missing values allowed?
- `...` Additional arguments passed on to the score and jacobian functions

**Value**

The jacobian matrix

---

** jacLatentVars **

*Evaluate the jacobian for estimating the latent variable for one view*

**Description**

Evaluate the jacobian for estimating the latent variable for one view

**Usage**

```
jacLatentVars(
    latentVar,
    data,
    distribution,
    paramEsts,
    paramMats,
```
jacLatentVarsConstr

Arguments

latentVar     the latent variable estimates
distribution, data, varPosts, compositional, meanVarTrend, offSet, paramEsts, paramMats, indepModel
Characteristics of each view

n             the number of samples
mm            the current dimension
latentVarsLower the lower dimensional latent variables
allowMissingness a boolean, should missing values be allowed
...          additional arguments passed on to score and jacobian functions

Value

The diagonal of the jacobian matrix

jacLatentVarsConstr      Evaluate the jacobian for estimating the latent variable for one view for constrained ordination

Description

Evaluate the jacobian for estimating the latent variable for one view for constrained ordination

Usage

jacLatentVarsConstr(
  latentVar,
  data,
  distribution,
  paramEsts,
  offSet,
  meanVarTrend,
plot.combi

numCov,
covMat,
varPosts,
compositional,
mm,
indepModel,
lateralVarsLower,
...
)

Arguments

latentVar  current latent variable estimates
distribution, compositional, meanVarTrend, offSet, paramEsts, indepModel, varPosts, data

Characteristics of each view

numCov  the number of covariates
covMat  the covariates matrix
mm  the dimension

latentVarsLower  latent variable estimates of lower dimensions
...

additional arguments passed on to score and jacobian functions

Value

The jacobian matrix

plot.combi  Make multiplots of the data integration object

Description

Make multiplots of the data integration object

Usage

## S3 method for class 'combi'
plot(
x, 
..., 
Dim = c(1, 2),
samDf = NULL, 
samShape = NULL, 
samCol = NULL, 
featurePlot = "threshold", 
featNum = 15L, 
samColValues = NULL,
manExpFactorTaxa = 0.975,
featSize = switch(featurePlot, threshold = 2.5, points = samSize * 0.7, density = 0.35),
crossSize = 4,
manExpFactorVar = 0.975,
varNum = nrow(x$alphas),
varSize = 2.5,
samSize = 1.75,
featCols = c("darkblue", "darkgreen", "grey10", "turquoise4", "blue", "green", "grey",
        "cornflowerblue", "lightgreen", "grey75"),
strokeSize = 0.05,
warnMonotonicity = FALSE,
returnCoords = FALSE,
squarePlot = TRUE,
featAlpha = 0.5,
featShape = 8,
xInd = 0,
yInd = 0,
checkOverlap = FALSE,
shapeValues = (21:(21 + length(unique(samDf[[samShape]])))))

Arguments

x                 the fit
...               additional arguments, currently ignored
Dim               the dimensions to be plotted
samDf             a dataframe of sample variables
samShape          A variable name from samDf used to shape the samples
samCol            A variable name from samDf used to colour the samples
featurePlot       A character string, either "threshold", "points" or "density". See details
featNum, varNum   The number of features and variables to plot
samColValues      Colours for the samples
manExpFactorTaxa, manExpFactorVar
        Expansion factors for taxa and variables, normally calculated natively
featSize, crossSize, varSize, samSize, strokeLine
        Size parameters for the features (text, dots or density contour lines), central
cross, variable labels, sample dots, sample strokes and feature contour lines
featCols          Colours for the features
warnMonotonicity  A boolean, should a warning be thrown when the feature proportions of compo-
sitional views do not all vary monotonically with all latent variables?
returnCoords      A boolean, should coordinates be returned, e.g. for use in third party software
squarePlot        A boolean, should the axes be square? Strongly recommended
featAlpha         Controls the transparency of the features
polyHorner

featShape      Shape of feature dots when featurePlot = "points"
xInd, yInd     x and y indentations
checkOverlap   A boolean, should overlapping labels be omitted?
shapeValues    the shapes, as numeric values

Details

It is usually impossible to plot all features with their labels. Therefore, the default option of the 'featurePlot' parameter is "threshold", whereby only the 'featNum' features furthest away from the origin are shown. Alternatively, the "points" or "density" options are available to plot all features as a point or density cloud, but without labels.

Value

A ggplot object containing the plot

Examples

data(Zhang)
## Not run:
#Unconstrained
microMetaboInt = combi(
  list("microbiome" = zhangMicrobio, "metabolomics" = zhangMetabo),
  distributions = c("quasi", "gaussian"), compositional = c(TRUE, FALSE),
  logTransformGaussian = FALSE, verbose = TRUE)
#Constrained
microMetaboIntConstr = combi(
  list("microbiome" = zhangMicrobio, "metabolomics" = zhangMetabo),
  distributions = c("quasi", "gaussian"), compositional = c(TRUE, FALSE),
  logTransformGaussian = FALSE, covariates = zhangMetavars, verbose = TRUE)
## End(Not run)

#Load the fits
load(system.file("extdata", "zhangFits.RData", package = "combi"))
plot(microMetaboInt)
plot(microMetaboInt, samDf = zhangMetavars, samCol = "ABX")
#Plot all features as points or density
plot(microMetaboInt, samDf = zhangMetavars, samCol = "ABX", featurePlot = "points")
plot(microMetaboInt, samDf = zhangMetavars, samCol = "ABX", featurePlot = "density")
#Constrained
plot(microMetaboIntConstr, samDf = zhangMetavars, samCol = "ABX")

---

data(Zhang)

$polyHorner$ Horner’s method to evaluate a polynomial, copied from the polynom package. the most efficient way
Description
Horner's method to evaluate a polynomial, copied from the polynom package. the most efficient way

Usage
polyHorner(coefs, x)

Arguments
coefs the polynomial coefficients
x the input values for the polynomial function

Value
the evaluated polynomial

predictSpline A custom spline prediction function, extending linearly with a slope such that prediction never drops below first bisectant

Description
A custom spline prediction function, extending linearly with a slope such that prediction never drops below first bisectant

Usage
predictSpline(
    fit, newdata, linX, coefsQuad, deriv = 0L, meanVarFit, minFit, new.knots, degree
)

Arguments
fit The existing spline fit
newdata points in which the spline needs to be evaluated
linX The x at which the fit becomes linear and intersects the diagonal line
coefsQuad parameters of a quadratic fit
**prepareJacMat**

| deriv | An integer. Which derivative is required? |
| meanVarFit | A character string, indicating which type of mean variance fit is being used |
| minFit | The lower bound of the cubic fit |
| new.knots | The knots at which the spline is to be evaluated |
| degree | The degree of the polynomial fit |

**Value**

The evaluation of the spline, i.e. the predicted variance

---

**Description**

prepare the jacobian matrix

**Usage**

```r
prepareJacMat(mu, data, meanVarTrend, CompMat, libSizes)
```

**Arguments**

- **mu**  
  the mean matrix
- **data**  
  the count matrix
- **meanVarTrend**  
  The mean variance trend
- **CompMat**  
  The composition matrix
- **libSizes**  
  The library sizes

**Value**

the matrix which can be summed over
prepareJacMatComp

prepare the jacobian for the latent variables compositional

Description

prepare the jacobian for the latent variables compositional

Usage

prepareJacMatComp(mu, paramEsts, CompMat0, meanVarTrend, data, libSizes)

Arguments

mu the mean matrix
paramEsts Current parameter estimates
CompMat0 The composition matrix
meanVarTrend The mean variance trend
data the count matrix
libSizes The library sizesv

Value

The empty jacobian matrix with entries maximally filled out

prepareScoreMat

Prepare a helper matrix for score function evaluation under quasi-likelihood

Description

Prepare a helper matrix for score function evaluation under quasi-likelihood

Usage

prepareScoreMat(data, mu, meanVarTrend, CompMat, libSizes)

Arguments

data the count matrix
mu the mean matrix
meanVarTrend The mean variance trend
CompMat The composition matrix
libSizes The library sizesv
Value

The helper matrix

Description

Print an overview of a fitted combi x

Usage

```r
## S3 method for class 'combi'
print(x, ...)
```

Arguments

- `x` a fitted combi x
- `...` Further arguments, currently ignored

Value

An overview of the number of dimensions, views and parameters, type of ordination and importance parameters

Examples

data(Zhang)
```r
## Not run:
# Unconstrained
microMetaboInt = combi(
  list("microbiome" = zhangMicrobio, "metabolomics" = zhangMetabo),
  distributions = c("quasi", "gaussian"), compositional = c(TRUE, FALSE),
  logTransformGaussian = FALSE, verbose = TRUE)
# Constrained
microMetaboIntConstr = combi(
  list("microbiome" = zhangMicrobio, "metabolomics" = zhangMetabo),
  distributions = c("quasi", "gaussian"), compositional = c(TRUE, FALSE),
  logTransformGaussian = FALSE, covariates = zhangMetavars, verbose = TRUE)
## End(Not run)
```
```r
# Load the fits
load(system.file("extdata", "zhangFits.RData", package = "combi"))
print(microMetaboInt)
print(microMetaboIntConstr)
# Or simply
microMetaboInt
```
quasiJacIndep

The jacobian for column offset estimation

Description
The jacobian for column offset estimation

Usage
quasiJacIndep(x, data, otherMargin, meanVarTrend, col, libSizes, ...)

Arguments
- **x**: the initial guess for the current margin
- **data**: the data matrix
- **otherMargin**: The other margin
- **meanVarTrend**: the function describing the mean-variance trend
- **col**: A logical, is the column being estimated?
- **libSizes**: The library sizes
- **...**: passed on to prepareJacMat

Value
the jacobian matrix

quasiScoreIndep

Quasi score equations for column offset parameters of sequence count data

Description
Quasi score equations for column offset parameters of sequence count data

Usage
quasiScoreIndep(x, data, otherMargin, meanVarTrend, col, libSizes, ...)

Arguments
- **x**: the initial guess for the current margin
- **data**: the data matrix
- **otherMargin**: The other margin
- **meanVarTrend**: the function describing the mean-variance trend
- **col**: A logical, is the column being estimated?
- **libSizes**: The library sizes
- **...**: passed on to prepareJacMat
**Value**

the evaluated estimating equation

---

**Description**

A function to efficiently row multiply a matrix and a vector

**Usage**

```r
rowMultiply(matrix, vector)
```

**Arguments**

- `matrix`: a numeric matrix of dimension a-by-b
- `vector`: a numeric vector of length b
  
  `t(t(matrix) * vector)` but then faster

**Details**

Memory intensive but that does not matter with given matrix sizes

**Value**

a matrix, row multiplied by the vector

---

**scaleCoords**

*A helper function to rescale coordinates*

---

**Description**

A helper function to rescale coordinates

**Usage**

```r
scaleCoords(featCoords, latentData, manExpFactorTaxa, featNum = NULL)
```

**Arguments**

- `featCoords`: the feature coordinates to be rescaled
- `latentData`: latent variables
- `manExpFactorTaxa`: an expansion factor
- `featNum`: the number of features to retain
Value

The rescaled feature coordinates

---

`scoreConfounders`  Score functions for confounder variables

Description

Score functions for confounder variables

Usage

```r
scoreConfounders(
  x,
  data, distribution, offset, confMat, meanVarTrend, allowMissingness, libSizes, CompMat
)
```

Arguments

- `x`  the parameter estimates
- `data, distribution, offset, confMat, meanVarTrend`  Characteristics of the views
- `allowMissingness`  a boolean, should missing values be allowed
- `libSizes, CompMat`  Library sizes and relative abundance

Value

The evaluation of the estimating equations
Description

Score equations for conditioning under compositionality

Usage

scoreConfoundersComp(
  x,
  confMat,
  data,
  meanVarTrend,
  marginModel,
  biasReduction,
  allowMissingness,
  subtractMax = TRUE
)

Arguments

x Confounder parameter estimates
confMat confounder matrix
data data
meanVarTrend mean variance trend
marginModel marginal models
biasReduction A boolean, should a bias reduced estimation be applied?
allowMissingness A boolean, are missing values allowed
subtractMax A boolean, should the maximum be subtracted before softmax transformation? Recommended for numerical stability

Value

The evaluation of the estimating equations
Evaluate the score functions for the estimation of the feature parameters for a single dataset

**Description**

Evaluate the score functions for the estimation of the feature parameters for a single dataset

**Usage**

```r
scoreFeatureParams(
  x,
  data, distribution, offset, latentVar, meanVarTrend, mm, indepModel, compositional, paramEstsLower, allowMissingness,
  ...
)
```

**Arguments**

- `x` the parameter estimates
- `data`, `distribution`, `offset`, `meanVarTrend`, `indepModel`, `compositional`, `paramEstsLower` Characteristics of the views
- `latentVar` the latent variables
- `mm` the dimension
- `allowMissingness` a boolean, should missing values be allowed
- `...` Additional arguments passed on to the score and jacobian functions

**Value**

A vector with evaluated score function
scoreLatentVars

Evaluate the score functions for the estimation of the latent variables for a single dataset

Description
Evaluate the score functions for the estimation of the latent variables for a single dataset

Usage
```
scoreLatentVars(
  data, distribution, paramEsts, paramMats, offSet, latentVar, meanVarTrend, constrained = FALSE, covMat = NULL, varPosts, compositional, indepModel, mm, latentVarsLower, allowMissingness, ...
)
```

Arguments
data, distribution, offSet, meanVarTrend, indepModel, varPosts, paramEsts, paramMats, compositional
  Characteristics of the views
latentVar the latent variable estimates
constrained a boolean, is this a constrained analysis
covMat a matrix of constraining covariates
mm the current dimension
latentVarsLower the lower dimensional latent variables
allowMissingness a boolean, should missing values be allowed
...
  additional arguments passed on to score and jacobian functions

Value
A vector of length n, with evaluated score function
seqM

A small auxiliary function for the indices of the Lagrange multipliers

Description

A small auxiliary function for the indices of the Lagrange multipliers

Usage

seqM(y, normal = TRUE, nLambda1s = 1)

Arguments

y
an integer, the current dimension

normal
a logical, is there a normalization restriction?

nLambda1s
the number of centering restrictions

Value

a vector containing the ranks of the current Lagrangian multipliers

trimOnConfounders

Trim based on confounders to avoid taxa with only zero counts

Description

Trim based on confounders to avoid taxa with only zero counts

Usage

trimOnConfounders(confounders, data, prevCutOff, minFraction, n)

Arguments

confounders
a nx t confounder matrix

data
the data matrix

prevCutOff
a scalar between 0 and 1, the prevalence cut off

minFraction
a scalar between 0 and 1, each taxon's total abundance should equal at least the
number of samples n times minFraction, otherwise it is trimmed

n
the number of samples
Should be called prior to fitting the independence model

Value

A trimmed data matrix nxp'
zhangMetabo

Metabolomes of mice that underwent Pulsed Antibiotic Treatment (PAT) and controls

Description
Metabolome of mice that underwent Pulsed Antibiotic Treatment (PAT) and controls

Usage
data(Zhang)

Format
SummarizedExperiment with metabolome data

zhangMetabo The metabolome data as a SummarizedExperiment object

Source
https://www.ibd MDB.org/

zhangMetavars Baseline sample variables of PAT and control mice

Description
Baseline covariates of PAT mice and healthy controls

Usage
data(Zhang)

Format
A dataframe with baseline sample variables

zhangMetavars The metadata on the mice

Source
https://www.ibd MDB.org/
zhangMicrobio

Microbiomes of mice that underwent Pulsed Antibiotic Treatment (PAT) and controls.

Description

Microbiome of mice that underwent Pulsed Antibiotic Treatment (PAT) and controls. The data were extracted from the source https://www.ibdmdl.org/, and then only the samples matching between microbiome and metabolome were retained.

Usage

data(Zhang)

Format

A phyloseq object containing microbiome data

zhangMicrobio The microbiome dataset pruned for matches with the metabolome object

Source

https://www.ibdmdl.org/
Index

* **datasets**
  - zhangMetabo, 51
  - zhangMetavars, 51
  - zhangMicrobio, 52

addLink, 3
arrayMult, 4
buildCentMat, 5
buildCompMat, 5
buildConfMat, 6
buildCovMat, 7
buildEmptyJac, 7
buildMarginalOffset, 8
buildMu, 9
buildMuMargins, 9
buildOffsetModel, 10
checkAlias, 10
checkMeanVarTrend, 11
checkMonotonicity, 11
combi, 12
convPlot, 14
deriv2LagrangianFeatures, 15
deriv2LagrangianLatentVars, 16
deriv2LagrangianLatentVarsConstr, 17
derivLagrangianFeatures, 18
derivLagrangianLatentVars, 19
derivLagrangianLatentVarsConstr, 20
estFeatureParameters, 21
estIndepModel, 23
estLatentVars, 24
estMeanVarTrend, 24
estOff, 25
extractCoords, 26
extractData, 27
extractMat, 28
extractMat, ExpressionSet-method (extractMat), 28
extractMat, matrix-method (extractMat), 28
extractMat, SummarizedExperiment-method (extractMat), 28
filterConfounders, 28
getInflLatentVar, 29
gramSchmidtOrth, 30
indentPlot, 30
inflPlot, 31
influence.combi, 32
jacConfounders, 33
jacConfoundersComp, 34
jacFeatures, 34
jacLatentVars, 35
jacLatentVarsConstr, 36
plot.combi, 37
polyHorner, 39
predictSpline, 40
prepareJacMat, 41
prepareJacMatComp, 42
prepareScoreMat, 42
print.combi, 43
quasiJacIndep, 44
quasiScoreIndep, 44
rowMultiply, 45
scaleCoords, 45
scoreConfounders, 46
scoreConfoundersComp, 47
scoreFeatureParams, 48
scoreLatentVars, 49
seqM, 50
trimOnConfounders, 50
zhangMetabo, 51
zhangMetavars, 51
zhangMicrobio, 52