Package 'limpca'

March 3, 2025

Type Package

Title An R package for the linear modeling of high-dimensional designed data based on ASCA/APCA family of methods

Version 1.2.0

Description This package has for objectives to provide a method to make Linear Models for high-dimensional designed data. limpca applies a GLM (General Linear Model) version of ASCA and APCA to analyse multivariate sample profiles generated by an experimental design. ASCA/APCA provide powerful visualization tools for multivariate structures in the space of each effect of the statistical model linked to the experimental design and contrarily to MANOVA, it can deal with mutlivariate datasets having more variables than observations. This method can handle unbalanced design.

License Artistic-2.0

Encoding UTF-8

LazyData FALSE

VignetteBuilder knitr

Imports ggplot2, stringr, plyr, ggrepel, reshape2, grDevices, graphics, doParallel, parallel, dplyr, tibble, tidyr, ggsci, tidyverse, methods, stats, SummarizedExperiment, S4Vectors

Suggests BiocStyle, pander, rmarkdown, car, gridExtra, knitr, testthat (>= 3.0.0)

biocViews StatisticalMethod, PrincipalComponent, Regression, Visualization, ExperimentalDesign, MultipleComparison, GeneExpression, Metabolomics

RoxygenNote 7.3.1

Roxygen list(markdown=TRUE)

BugReports https://github.com/ManonMartin/limpca/issues

URL https://github.com/ManonMartin/limpca, https://manonmartin.github.io/limpca/

Config/testthat/edition 3

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

git_branch RELEASE_3_20

git_last_commit 32706c3

git_last_commit_date 2024-10-29

Repository Bioconductor 3.20

Date/Publication 2025-03-03

Author Bernadette Govaerts [aut, ths],
Sebastien Franceschini [ctb],
Robin van Oirbeek [ctb],
Michel Thiel [aut],
Pascal de Tullio [dtc],
Manon Martin [aut, cre] (<https://orcid.org/0000-0003-4800-0942>),
Nadia Benaiche [ctb]

Maintainer Manon Martin <manon.martin@uclouvain.be>

Contents

`data2LmpDataList` *Converts data to a lmpDataList.*

Description

Creates the lmpDataList from a SummarizedExperiment or by manually defining the design, the outcomes and the model formula. lmpDataList serves as an input for the `lmpModelMatrix` function to start the limpca modeling.

Usage

```
data2LmpDataList(
  se = NULL,
  assay_name = NULL,
  outcomes = NULL,
  design = NULL,
  formula = NULL,
  verbose = TRUE
)
```

Arguments

se	A SummarizedExperiment object.
assay_name	If not NULL (default), a character string naming the assay from the SummarizedExperiment object se. If NULL, the first assay is selected.
outcomes	If not NULL (default), a numerical matrix with *n* observations and *m* response variables. The rownames needs to be non-NULL and match those of the design matrix.
design	If not NULL (default), a data.frame with the experimental design of *n* observations and *q* explanatory variables. The rownames of design has to match the rownames of outcomes.
formula	If not NULL (default), a character string with the formula that will be used to analyze the data. Only the right part of the formula is necessary, eg: "~ A + B", The names of the formula should match the column names of the design
verbose	If TRUE, prints useful information about the outputted list.

Details

Data can be included as a SummarizedExperiment (SE) object or by manually defining one or multiple elements of outcomes, design and formula. If a SE is provided, the outcomes corresponds to a transposed assay of the SE (by default the first one), the design corresponds to the colData of the SE and the formula can be provided as a formula element in the S4Vectors::metadata of SE (metadata(se)$formula).

In the outputted list, the outcomes are structured in a standard statistical fashion, i.e. with observations in rows and the variables (features) in column. If the outcomes argument is not NULL, it has to be formatted that way (see Arguments).

Note that there is a priority to the outcomes, design and formula arguments if they are not NULL (e.g. if both se and outcomes arguments are provided, the resulting outcomes matrix will be from the outcomes argument). outcomes and design elements are mandatory.

Multiple checks are performed to ensure that the data are correctly formatted:

- the rownames of design and outcomes should match
- the names of the model terms in the formula should match column names from the design

Value

A list with the 3 following named elements:

outcomes A *nxm* matrix with the *m* response variables.

design A *nxq* data.frame with the experimental design.

formula A character string with the model formula.

See Also

SummarizedExperiment

Examples

```
data(UCH)

### create manually the dataset

res <- data2LmpDataList(
  outcomes = UCH$outcomes,
  design = UCH$design[, 1, drop = FALSE], formula = "~ Hippurate"
)

### create the dataset from a SummarizedExperiment

library(SummarizedExperiment)

se <- SummarizedExperiment(
  assays = list(
    counts = t(UCH$outcomes),
    counts2 = t(UCH$outcomes * 2)
  ), colData = UCH$design,
  metadata = list(formula = "~ Hippurate + Citrate")
)

res <- data2LmpDataList(se, assay_name = "counts2")

# changing the formula:
res <- data2LmpDataList(se,
  assay_name = "counts2",
  formula = "~ Hippurate + Citrate + Time"
)
```

`limpca` *Linear modeling of high-dimensional designed data based on ASCA/APCA family of methods*

Description

This package has for objectives to provide a method to make Linear Models for high-dimensional designed data. This method handles unbalanced design. More features should be included in the future (e.g. generalized linear models, random effects, ...).

The core functions of the package are:

`data2LmpDataList` Converts data to a lmpDataList, the input argument for `lmpModelMatrix`.

`lmpModelMatrix` Creates the model matrix $\mathbf{X}$ from the design matrix and the model formula.

`lmpEffectMatrices` Estimates the model by OLS based on the outcomes and model matrices provided in the outputs of the `lmpModelMatrix` function and calculates the estimated effect matrices $\hat{\mathbf{M}}_0, \hat{\mathbf{M}}_1, ...\hat{\mathbf{M}}_F$ and residual matrix $\hat{\mathbf{E}}$. It calculates also the type III percentage of variance explained by each effect.

lmpBootstrapTests Tests the significance of one or a combination of the model effects using bootstrap. This function is based on the outputs of the lmpEffectMatrices function.

lmpPcaEffects Performs a PCA on each of the effect matrices from the outputs of lmpEffectMatrices. It has an option to choose the method applied: ASCA, APCA or ASCA-E. Combined effects (i.e. linear combinations of original effect matrices) can also be created and decomposed by PCA.

The functions allowing the visualisation of the Linear Models results are:

lmpScreePlot Provides a barplot of the percentage of variance associated to the PCs of the effect matrices ordered by importance based on the outputs of lmpContributions.

lmpContributions This reports the contribution of each effect to the total variance, but also the contribution of each PC to the total variance per effect. Moreover, these contributions are summarized in a barplot.

lmpScorePlot Draws the score plots of each effect matrix provided in the lmpPcaEffects function output.

lmpLoading1dPlot **or** lmpLoading2dPlot Plots the loadings as a line plot (1D) or in 2D as a scatterplot.

lmpScoreScatterPlotM Plots the scores of all model effects simultaneously in a scatterplot matrix. By default, the first PC only is kept for each model effect.

lmpEffectPlot Plots the ASCA scores by effect levels for a given model effect and for one PC at a time. This graph is especially appealing to interpret interactions or combined effects.

Other useful functions to visualise and explore by PCA the multivariate data are:

plotDesign Provides a graphical representation of the experimental design. It allows to visualize factor levels and check the design balance.

plotScatter Produces a plot describing the relationship between two columns of the outcomes matrix $\mathbf{Y}$. It allows to choose colors and symbols for the levels of the design factors. Ellipses, polygons or segments can be added to group different sets of points on the graph.

plotScatterM Produces a scatter plot matrix between the selected columns of the outcomes matrix $\mathbf{Y}$ choosing specific colors and symbols for up to four factors from the design on the upper and lower diagonals.

plotMeans Draws, for a given response variable, a plot of the response means by levels of up to three categorical factors from the design. When the design is balanced, it allows to visualize main effects or interactions for the response of interest. For unbalanced designs, this plot must be used with caution.

plotLine Generates the response profile of one or more observations i.e. plots of one or more rows of the outcomes matrix on the y-axis against the m response variables on the x-axis. Depending on the response type (spectra, gene expression...), point, line or segment plots can be used.

pcaBySvd Operates a principal component analysis on the $\mathbf{Y}$ outcome/response matrix by a singular value decomposition. Outputs are can be visulised with the functions pcaScorePlot, pcaLoading1dPlot, pcaLoading2dPlot and pcaScreePlot.

pcaScorePlot Produces score plots from the pcaBySvd output.

pcaLoading1dPlot **or** pcaLoading2dPlot Plots the PCA loadings as a line plot (1D) or in 2D as a scatterplot.

pcaScreePlot Returns a bar plot of the percentage of variance explained by each Principal Component (PC) calculated by pcaBySvd.

Details

Package:	limpca
Type:	Package
License:	GPL-2

See the package vignettes (`vignette(package = "limpca")`) for detailed case studies.

References

Thiel, M., Benaiche, N., Martin, M., Franceschini, S., Van Oirbeek, R., Govaerts, B. (2023). limpca: An R package for the linear modeling of high-dimensional designed data based on ASCA/APCA family of methods. *Journal of Chemometrics*. e3482. https://doi.org/10.1002/cem.3482

Martin, M. (2020). *Uncovering informative content in metabolomics data: from pre-processing of 1H NMR spectra to biomarkers discovery in multifactorial designs.* Prom.: Govaerts, B. PhD thesis. Institut de statistique, biostatistique et sciences actuarielles, UCLouvain, Belgium. http://hdl.handle.net/2078.1/227671

Thiel M., Feraud B. and Govaerts B. (2017). ASCA+ and APCA+: Extensions of ASCA and APCA in the analysis of unbalanced multifactorial designs. *Journal of Chemometrics*. 31:e2895. https://doi.org/10.1002/cem.2895

`lmpBootstrapTests` *Tests the significance of model effects by bootstrap.*

Description

Tests the significance of the effects from the model using bootstrap. This function is based on the outputs of `lmpEffectMatrices`. Tests on combined effects are also provided.

Usage

```
lmpBootstrapTests(
  resLmpEffectMatrices,
  nboot = 100,
  nCores = 2,
  verbose = FALSE
)
```

Arguments

`resLmpEffectMatrices`	A list of 12 from `lmpEffectMatrices`.
`nboot`	An integer with the number of bootstrap sample to be drawn.
`nCores`	The number of cores to use for parallel execution.
`verbose`	If `TRUE`, will display a message with the duration of execution.

Value

A list with the following elements:

`f.obs` A vector of size F (number of effects in the model) with the F statistics for each model term calculated on the initial data.

`f.boot` b × F matrix with the F statistics calculated on the bootstrap samples.

`p.values` A vector of size F with the p-value for each model effect.

`resultsTable` A 2 × F matrix with the p-value and the percentage of variance for each model effect.

References

Thiel M.,Feraud B. and Govaerts B. (2017) *ASCA+ and APCA+: Extensions of ASCA and APCA in the analysis of unbalanced multifactorial designs*, Journal of Chemometrics

Thiel, M., Benaiche, N., Martin, M., Franceschini, S., Van Oirbeek, R., & Govaerts, B. (2023) *limpca: an R package for the linear modeling of high dimensional designed data based on ASCA/APCA family of methods*, Journal of Chemometrics

Examples

```
data("UCH")
resLmpModelMatrix <- lmpModelMatrix(UCH)
resLmpEffectMatrices <- lmpEffectMatrices(resLmpModelMatrix = resLmpModelMatrix)

res <- lmpBootstrapTests(
  resLmpEffectMatrices = resLmpEffectMatrices,
  nboot = 10, nCores = 2, verbose = TRUE
)
```

---

lmpContributions *Summary of the contributions of each effect*

Description

Reports the contribution of each effect to the total variance, but also the contribution of each PC to the total variance per effect. These contributions are also summarized in a barplot.

Usage

```
lmpContributions(resLmpPcaEffects, nPC = 5)
```

Arguments

`resLmpPcaEffects`	A list corresponding to the output value of `lmpPcaEffects`.
`nPC`	The number of Principal Components to display.

Value

A list of:

`totalContribTable` Table of the percentage of contribution of each effect to the total variance.

`effectTable` Table of the percentage of variance explained by each principal component in each model effect decomposition.

`contribTable` Table of the percentage of variance explained by each principal component of each effect reported to the percentage contribution of the given effect to the total variance.

`combinedEffectTable` Equivalent of the *EffectTable* for combined effects.

`plotTotal` Plot of the ordered contributions of *TotalContribTable*.

`plotContrib` Plot of the ordered contributions of *ContribTable*.

Examples

```
data("UCH")
resLmpModelMatrix <- lmpModelMatrix(UCH)
resLmpEffectMatrices <- lmpEffectMatrices(resLmpModelMatrix)
resLmpPcaEffects <- lmpPcaEffects(resLmpEffectMatrices, method = "ASCA-E")

lmpContributions(resLmpPcaEffects)
```

`lmpEffectMatrices` *Computes the effect matrices*

Description

Estimates the model by OLS based on the outcomes and model matrices provided in the outputs of lmpModelMatrix function and calculates the estimated effect matrices $\hat{\mathbf{M}}_0, \hat{\mathbf{M}}_1, ...\hat{\mathbf{M}}_F$, ... and residual matrix $\hat{\mathbf{E}}$. It calculates also the type III percentage of variance explained by each effect.

Usage

```
lmpEffectMatrices(resLmpModelMatrix, SS = TRUE, contrastList = NA)
```

Arguments

`resLmpModelMatrix`	A list of 5 elements from `lmpModelMatrix`.
`SS`	Logical. If `FALSE`, won't compute the percentage of variance for each effect.
`contrastList`	A list of contrasts for each parameter. If `NA`, the function creates automatically the list by default.

Value

A list with the following elements:

`lmpDataList` The initial object: a list with outcomes, design and formula.

`modelMatrix` A *nxp* model matrix specifically encoded for the ASCA-GLM method.

`modelMatrixByEffect` A list of *F+1* model matrices for each effect.

`effectsNamesUnique` A character vector with the *F+1* names of the model effects, each repeated once.

`effectsNamesAll` A character vector with the *p* names of the model effects ordered and repeated as the column names of the model matrix.

`effectMatrices` A list of *F+1* effect matrices for each model effect.

`predictedvalues` The *nxm* matrix of predicted outcome values.

`residuals` The *nxm* matrix of model residuals.

`parameters` The *pxm* matrix of the estimated parameters.

`type3SS` A vector with the type III sum of squares for each model effect *(If SS = TRUE)*.

`variationPercentages` A vector with the percentage of variance for each model effect *(If SS = TRUE)*.

`varPercentagesPlot` A ggplot bar plot of the contributions of each model effect to the total variance *(If SS = TRUE)*.

References

Thiel M.,Feraud B. and Govaerts B. (2017) *ASCA+ and APCA+: Extensions of ASCA and APCA in the analysis of unbalanced multifactorial designs*, Journal of Chemometrics

Examples

```
data("UCH")
resLmpModelMatrix <- lmpModelMatrix(UCH)
reslmpEffectMatrices <- lmpEffectMatrices(resLmpModelMatrix)
reslmpEffectMatrices$varPercentagesPlot
```

`lmpEffectPlot` *Effect plot*

Description

Plots the ASCA scores by effect levels for a given model effect and for one PC at a time. This graph is especially appealing to interpret interactions or combined effects. It is a wrapper of plotMeans.

Usage

```
lmpEffectPlot(
  resASCA,
  effectName,
  axes = 1,
  x,
  z = NULL,
  w = NULL,
  hline = 0,
  ...
)
```

Arguments

resASCA	A list corresponding to the ASCA output value of lmpPcaEffects.
effectName	Name of the effect to be used to plot the scores.
axes	A numerical vector with the Principal Components axes to be drawn.
x	A character string giving the design factor whose levels will form the x axis.
z	A character string giving the design factor whose levels will form the traces.
w	A character string giving the design factor whose levels will be used for the facet.
hline	If not NULL, draws one or several horizontal line(s) at values given in hline.
...	Additional arguments to be passed to plotMeans.

Details

lmpEffectPlot is a wrapper of plotMeans.

Value

An effect plot (ggplot).

Examples

```
data("UCH")
resLmpModelMatrix <- lmpModelMatrix(UCH)
resLmpEffectMatrices <- lmpEffectMatrices(resLmpModelMatrix)
resASCA <- lmpPcaEffects(
  resLmpEffectMatrices = resLmpEffectMatrices,
  method = "ASCA", combineEffects = list(c("Hippurate", "Time", "Hippurate:Time"))
)

# Effect plot for an interaction effect
lmpEffectPlot(resASCA, effectName = "Hippurate:Time", x = "Hippurate", z = "Time")
# Effect plot for a combined effect
lmpEffectPlot(resASCA, effectName = "Hippurate+Time+Hippurate:Time", x = "Hippurate", z = "Time")
```

lmpLoading1dPlot *Loadings represented on a line plot.*

Description

Plots the loading vectors for each effect matrix from the lmpPcaEffects outputs with line plots. This is a wrapper of plotLine.

Usage

```
lmpLoading1dPlot(resLmpPcaEffects, effectNames = NULL, axes = c(1, 2), ...)
```

Arguments

resLmpPcaEffects	A list corresponding to the output value of lmpPcaEffects.
effectNames	Names of the effects to be plotted. if NULL, all the effects are plotted.
axes	A numerical vector with the Principal Components axes to be drawn.
...	Additional arguments to be passed to plotLine such as xaxis_type, type or ang_x_axis.

Details

lmpLoading1dPlot is a wrapper of plotLine. See ?plotLine for more information on the additional arguments.

Value

A list of ggplot objects representing the loading plots.

Examples

```
# Example of "spectral" type loadings (line and numerical x-axis)
data("UCH")
resLmpModelMatrix <- lmpModelMatrix(UCH)
resLmpEffectMatrices <- lmpEffectMatrices(resLmpModelMatrix)
resASCA <- lmpPcaEffects(resLmpEffectMatrices,
  combineEffects = list(c("Time", "Hippurate:Time"))
)
lmpLoading1dPlot(resASCA)
lmpLoading1dPlot(resASCA, effectNames = c("Hippurate", "Citrate"))

# Example of "segment" and discrete type loadings (segments and character x-axis)
data("trout")
resLmpModelMatrix <- lmpModelMatrix(trout)
resLmpEffectMatrices <- lmpEffectMatrices(resLmpModelMatrix)
resASCA <- lmpPcaEffects(resLmpEffectMatrices)
lmpLoading1dPlot(resASCA,
  effectNames = "Day",
  xaxis_type = "character", type = "s", ang_x_axis = 90
)
```

`lmpLoading2dPlot` *Loading plots on a 2D scatter plot*

Description

Draws a 2D loading plot of each effect matrix provided in `lmpPcaEffects` outputs. As a wrapper of the `plotScatter` function, it allows the visualization of effect loading matrices for two components at a time with all options available in `plotScatter`.

Usage

```
lmpLoading2dPlot(
  resLmpPcaEffects,
  effectNames = NULL,
  axes = c(1, 2),
  addRownames = FALSE,
  pl_n = 10,
  metadata = NULL,
  drawOrigin = TRUE,
  ...
)
```

Arguments

`resLmpPcaEffects`	A list corresponding to the output value of `lmpPcaEffects`.
`effectNames`	Names of the effects to be plotted. If `NULL`, all the effects are plotted.
`axes`	A numerical vector with the 2 Principal Components axes to be drawn.
`addRownames`	Boolean indicating if the labels should be plotted. By default, uses the column names of the outcome matrix but it can be manually specified with the `points_labs` argument from `plotScatter`.
`pl_n`	The number of labels that should be plotted, based on the distance measure d (*see* Details).
`metadata`	A nxk "free encoded" data.frame corresponding to `design` in `plotScatter`.
`drawOrigin`	if `TRUE`, draws horizontal and vertical intercepts at (0,0) based on the `plotScatter` function.
`...`	Additional arguments to be passed to `plotScatter`.

Details

`lmpLoading2dPlot` is a wrapper of `plotScatter`. See `?plotScatter` for more information on the additional arguments.

The distance measure d that is used to rank the variables is based on the following formula:

$$d = \sqrt{(P_{ab}^2 * \lambda_{ab}^2)}$$

where a and b are two selected Principal Components, P_{ab} represents their loadings and λ_{ab} their singular values.

Value

A list of loading plots (ggplot).

Examples

```
data("UCH")
resLmpModelMatrix <- lmpModelMatrix(UCH)
resLmpEffectMatrices <- lmpEffectMatrices(resLmpModelMatrix)
resASCA <- lmpPcaEffects(resLmpEffectMatrices)

lmpLoading2dPlot(resASCA, effectNames = "Hippurate")

# adding color, shape and labels to points
id_hip <- c(seq(126, 156), seq(362, 375))
peaks <- rep("other", ncol(UCH$outcomes))
peaks[id_hip] <- "hip"
metadata <- data.frame(peaks)

lmpLoading2dPlot(resASCA,
  effectNames = "Hippurate",
  metadata = metadata, addRownames = TRUE, color = "peaks",
  shape = "peaks"
)

# changing max.overlaps of ggrepel
options(ggrepel.max.overlaps = 30)
lmpLoading2dPlot(resASCA,
  effectNames = "Hippurate",
  metadata = metadata, addRownames = TRUE, color = "peaks",
  shape = "peaks", pl_n = 20
)
```

---

lmpModelMatrix *Creates the model matrix* X

Description

Creates the model matrix X from the design matrix and the model formula.

Usage

```
lmpModelMatrix(lmpDataList)
```

Arguments

lmpDataList	A list containing the outcomes, the experimental design and the formula.

Details

In typical ASCA-GLM (ASCA+) analysis, the effects of the GLM model must first be used to transform the design matrix to a model matrix where the design factors encoded usign *sum coding* commonly used in industrial experimental design. Suppose the design matrix is *nxk* with n observations and *k* factors. After the transformation, the model matrix will be of size *nxp*. For a fator with

a levels, the *sum coding* creates *a-1* columns in the model matrix with 0 and 1 for the *a-1* first levels and -1 for the last one. *p* is the total number parameter for each response (outcome) in the ASCA model. More information is available in the article (*Thiel et al*, 2017) Note that at the moment, only factors can be used as explanatory variables.

Value

A list with the 5 following named elements :

`lmpDataList` The initial object: a list with outcomes, design and formula, as outputted by `data2LmpDataList`.

`modelMatrix` A *nxK* model matrix specifically encoded for the ASCA-GLM method.

`modelMatrixByEffect` A list of *p* model matrices for each model effect.

`effectsNamesUnique` A character vector with the *p* names of the model effects, each repeated once.

`effectsNamesAll` A character vector with the *K* names of the model effects ordered and repeated as the column names of the model matrix.

References

Thiel M.,Feraud B. and Govaerts B. (2017) *ASCA+ and APCA+: Extensions of ASCA and APCA in the analysis of unbalanced multifactorial designs*, Journal of Chemometrics

See Also

`model.matrix`

Examples

```
data("UCH")
resLmpModelMatrix <- lmpModelMatrix(UCH)

head(resLmpModelMatrix$modelMatrix)
```

---

lmpPcaEffects *PCA on the effect matrices*

Description

Performs a PCA on each of the effect matrices from the outputs of `lmpEffectMatrices`. It has an option to choose the method applied: ASCA, APCA or ASCA-E. Combined effects (i.e. linear combinations of original effect matrices) can also be created and decomposed by PCA.

Usage

```
lmpPcaEffects(
  resLmpEffectMatrices,
  method = c("ASCA", "APCA", "ASCA-E"),
  combineEffects = NULL,
  verbose = FALSE
)
```

Arguments

resLmpEffectMatrices	A resLmpEffectMatrices list resulting of lmpEffectMatrices.
method	The method used to compute the PCA. One of c("ASCA","APCA","ASCA-E").
combineEffects	If not NULL, a list of vectors containing the names of the effects to be combined.
verbose	If TRUE, will display a message with the duration of execution.

Details

The function allows 3 different methods :

ASCA PCA is applied directly on each pure effect matrix $\hat{\mathbf{M}}_f, f = 1...F$.

ASCA-E PCA is applied on each pure effect matrix but then the augmented effect matrix is projected in the space of the ASCA components.

APCA PCA is applied on each augmented effect matrix : $\hat{\mathbf{M}}_f + \hat{\mathbf{E}}$.

Value

A list with first,the PCA results from pcaBySvd for each effect matrix. Those results contain :

scores Scores from the PCA for each principal component.

loadings Loadings from the PCA for each principal component.

eigval Eigenvalues of each principal component.

singvar Singular values of each principal component.

var Explained variances of each principal component.

cumvar Cumulated explained variances of each principal component.

original.dataset Original dataset.

There are also others outputs :

lmpDataList The initial object: a list of outcomes, design and formula.

effectsNamesUnique A character vector with the *F+1* names of the model terms, each repeated once.

method The dimension reduction method used: c("ASCA","APCA","ASCA-E").

type3SS A vector with the type III SS for each model term.

variationPercentages A vector with the percentage of variance explained by each model term.

References

Thiel M.,Feraud B. and Govaerts B. (2017) ASCA+ and APCA+: Extensions of ASCA and APCA in the analysis of unbalanced multifactorial designs. *Journal of Chemometrics*. 31:e2895. https://doi.org/10.1002/cem.2895

Examples

```
data("UCH")
resLmpModelMatrix <- lmpModelMatrix(UCH)
resLmpEffectMatrices <- lmpEffectMatrices(resLmpModelMatrix)
resLmpPcaEffects <- lmpPcaEffects(resLmpEffectMatrices, method = "ASCA-E")
```

`lmpScorePlot` *Score plots of effect matrices*

Description

Draws the score plots of each (augmented) effect matrix provided in `lmpPcaEffects`. As a wrapper of the `plotScatter` function, it allows to visualize the scores of the effect matrices for two components at a time with all the available options in `plotScatter`.

Usage

```
lmpScorePlot(resLmpPcaEffects, effectNames = NULL, axes = c(1, 2), ...)
```

Arguments

`resLmpPcaEffects`	A list corresponding to the output value of `lmpPcaEffects`.
`effectNames`	Names of the effects to be plotted. If `NULL`, all the effects are plotted.
`axes`	A numerical vector with the 2 Principal Components axes to be drawn.
`...`	Additional arguments to be passed to `plotScatter`.

Details

`lmpScorePlot` is a wrapper of `plotScatter`.

Value

A list of score plots (ggplot).

Examples

```
data("UCH")
resLmpModelMatrix <- lmpModelMatrix(UCH)
resLmpEffectMatrices <- lmpEffectMatrices(resLmpModelMatrix)

# PCA decomposition of effect matrices (ASCA)
resASCA <- lmpPcaEffects(resLmpEffectMatrices)
# Score plot of Hippurate effect matrix
lmpScorePlot(resASCA,
  effectNames = "Hippurate",
  color = "Hippurate", shape = "Hippurate"
)

# PCA decomposition of augmented effect matrices (APCA)
resASCA <- lmpPcaEffects(resLmpEffectMatrices, method = "APCA")
# Score plot of Hippurate augmented effect matrix
lmpScorePlot(resASCA,
  effectNames = "Hippurate",
  color = "Hippurate", shape = "Hippurate", drawShapes = "ellipse"
)
```

`lmpScoreScatterPlotM` *Scatterplot matrix of effect matrices scores*

Description

Plots the scores of all model effects simultaneously in a scatterplot matrix. By default, the first PC only is kept for each model effect and, as a wrapper of `plotScatterM`, the choice of symbols and colors to distinguish factor levels allows an enriched visualization of the factors' effect on the responses.

Usage

```
lmpScoreScatterPlotM(
  resLmpPcaEffects,
  effectNames = NULL,
  PCdim = NULL,
  modelAbbrev = FALSE,
  ...
)
```

Arguments

`resLmpPcaEffects`	A list corresponding to the output value of `lmpPcaEffects`.
`effectNames`	A character vector with the name of the effects to plot.
`PCdim`	A numeric vector with the same length than effectNames and indicating the number of component to plot.
`modelAbbrev`	A logical whether to abbreviate the interaction terms or not.
`...`	Additional arguments to be passed to `plotScatterM`.

Details

`lmpScoreScatterPlotM` is a wrapper of `plotScatterM`.

Value

A matrix of graphs

Examples

```
data("UCH")
resLmpModelMatrix <- lmpModelMatrix(UCH)
ResLmpEffectMatrices <- lmpEffectMatrices(resLmpModelMatrix)
resLmpPcaEffects <- lmpPcaEffects(ResLmpEffectMatrices, method = "ASCA-E")

lmpScoreScatterPlotM(resLmpPcaEffects,
  varname.colorup = "Citrate",
  varname.pchup = "Hippurate",
  varname.pchdown = "Day",
  varname.colordown = "Time"
)
```

```
# advanced setting
lmpScoreScatterPlotM(resLmpPcaEffects,
  modelAbbrev = FALSE,
  effectNames = c("Citrate", "Hippurate", "Hippurate:Citrate"),
  PCdim = c(2, 2, 2),
  varname.colorup = "Citrate",
  vec.colorup = c("red", "blue", "green"),
  varname.pchup = "Hippurate",
  vec.pchup = c(1, 2, 3),
  varname.pchdown = "Day",
  vec.pchdown = c(4, 5),
  varname.colordown = "Time",
  vec.colordown = c("brown", "grey")
)
```

`lmpScreePlot` *Scree Plot*

Description

Provides a barplot of the percentage of variance associated to the PCs of the effect matrices ordered by importance based on the outputs of `lmpContributions`.

Usage

```
lmpScreePlot(
  resLmpContributions,
  effectNames = NULL,
  nPC = 5,
  theme = theme_bw()
)
```

Arguments

`resLmpContributions`	A resLmpContributions list from the function `lmpContributions`.
`effectNames`	Names of the effects to be plotted. if `NULL`, all the effects are plotted.
`nPC`	An integer with the number of components to plot.
`theme`	`ggplot` theme

Value

A scree plot (ggplot).

Examples

```
data("UCH")
resLmpModelMatrix <- lmpModelMatrix(UCH)
resLmpEffectMatrices <- lmpEffectMatrices(resLmpModelMatrix)
resASCAE <- lmpPcaEffects(resLmpEffectMatrices, method = "ASCA-E")
```

```
resLmpContributions <- lmpContributions(resASCAE)
lmpScreePlot(resLmpContributions, effectNames = "Hippurate:Citrate", nPC = 4)
```

---

pcaBySvd *Singular Value Decomposition for PCA analysis*

Description

Operates a Principal Component Analysis on the `Y` outcome/response matrix by a singular Value Decomposition (the pre-processing involves the mean-centering of `Y`). Outputs are represented with the functions `pcaScorePlot`, `pcaLoading1dPlot`, `pcaLoading2dPlot` and `pcaScreePlot`.

Usage

```
pcaBySvd(Y = NULL, lmpDataList = NULL, nPC = min(dim(Y)))
```

Arguments

`Y`	The $n \times m$ matrix with n observations and m (response) variables.
`lmpDataList`	A list with outcomes, design and formula, as outputted by `data2LmpDataList`.
`nPC`	Number of Principal Components to extract.

Value

A list containing the following elements:

`scores` Scores

`loadings` Loadings

`eigval` Eigenvalues

`singvar` Singular values

`var` Explained variances

`cumvar` Cumulated explained variances

`original.dataset` Original dataset

`design` Design of the study

Examples

```
data("UCH")

PCA.res1 <- pcaBySvd(Y = UCH$outcomes)

PCA.res2 <- pcaBySvd(lmpDataList = UCH)

identical(PCA.res1, PCA.res2)
```

pcaLoading1dPlot *Loadings represented on a line plot.*

Description

Plots the loading vectors from pcaBySvd output with different available line types.

Usage

```
pcaLoading1dPlot(resPcaBySvd, axes = c(1, 2), title = "PCA loading plot", ...)
```

Arguments

resPcaBySvd	A list corresponding to the output value of pcaBySvd.
axes	A numerical vector of length 2 with the Principal Components axes to be drawn.
title	Plot title.
...	Additional arguments to be passed to plotLine.

Details

pcaLoading1dPlot is a wrapper of plotLine. See ?plotLine for more information on the additional arguments.

Value

A ggplot2 object with the PCA loading plot.

Examples

```
data("UCH")
ResPCA <- pcaBySvd(UCH$outcomes)

pcaLoading1dPlot(
  resPcaBySvd = ResPCA, axes = c(1, 2),
  title = "PCA loading plot UCH", xlab = "ppm", ylab = "Values"
)
```

pcaLoading2dPlot *Loading plots on a 2D scatter plot*

Description

Produces 2D loading plots from pcaBySvd with the same graphical options as plotScatter as this is a wrapper of this function.

Usage

```
pcaLoading2dPlot(
  resPcaBySvd,
  axes = c(1, 2),
  title = "PCA loading plot",
  addRownames = FALSE,
  pl_n = 10,
  metadata = NULL,
  drawOrigin = TRUE,
  ...
)
```

Arguments

resPcaBySvd	A list corresponding to the output value of pcaBySvd.
axes	A numerical vector of length 2 with the Principal Components axes to be drawn.
title	Plot title.
addRownames	Boolean indicating if the labels should be plotted. By default, uses the row names of the loadings matrix but it can be manually specified with the points_labs argument from plotScatter.
pl_n	The number of labels that should be plotted, based on a distance measure (see Details).
metadata	A $n \times k$ "freely encoded" data.frame corresponding to the design argument in plotScatter.
drawOrigin	if TRUE, draws horizontal and vertical intercepts at (0,0) based on the plotScatter function.
...	Additional arguments to be passed to plotScatter.

Details

pcaLoading2dPlot is a wrapper of plotScatter. See ?plotScatter for more information on the additional arguments.

The distance measure d that is used to rank the variables is based on the following formula:

$$d = \sqrt{(P_{ab}^2 * \lambda_{ab}^2)}$$

where a and b are two selected Principal Components, P_{ab} represents their loadings and λ_{ab} their singular values.

Value

A ggplot2 object with the PCA loading plot.

Examples

```
data("UCH")
ResPCA <- pcaBySvd(UCH$outcomes)

pcaLoading2dPlot(
  resPcaBySvd = ResPCA, axes = c(1, 2),
  title = "PCA loading plot UCH"
```

```
)

# adding color,  shape and labels to points
id_cit <- seq(446, 459)
id_hip <- c(seq(126, 156), seq(362, 375))
peaks <- rep("other", ncol(UCH$outcomes))
peaks[id_hip] <- "hip"
peaks[id_cit] <- "cit"
metadata <- data.frame(peaks)

pcaLoading2dPlot(
  resPcaBySvd = ResPCA, axes = c(1, 2),
  title = "PCA loading plot UCH", metadata = metadata,
  color = "peaks", shape = "peaks", addRownames = TRUE
)

# changing max.overlaps of ggrepel
options(ggrepel.max.overlaps = 30)
pcaLoading2dPlot(
  resPcaBySvd = ResPCA, axes = c(1, 2),
  title = "PCA loading plot UCH", metadata = metadata,
  color = "peaks", shape = "peaks", addRownames = TRUE,
  pl_n = 35
)
```

`pcaScorePlot` *Score plots*

Description

Produces score plots from `pcaBySvd` output with the same graphical options as `plotScatter` as this is a wrapper of this function..

Usage

```
pcaScorePlot(
  resPcaBySvd,
  design = NULL,
  axes = c(1, 2),
  title = "PCA score plot",
  points_labs_rn = FALSE,
  ...
)
```

Arguments

`resPcaBySvd`	A list corresponding to the output value of `pcaBySvd`.
`design`	A $n \times k$ "freely encoded" experimental design data.frame.
`axes`	A numerical vector of length 2 with the Principal Components axes to be drawn.
`title`	Plot title.
`points_labs_rn`	Boolean indicating if the rownames of the scores matrix should be plotted.
`...`	Additional arguments to be passed to `plotScatter`.

Details

pcaScorePlot is a wrapper of plotScatter. See ?plotScatter for more information on the additional arguments.

Value

A ggplot2 PCA score plot.

Examples

```
data("UCH")

# design is explicitly defined
ResPCA <- pcaBySvd(Y = UCH$outcomes)

pcaScorePlot(
  resPcaBySvd = ResPCA, axes = c(1, 2),
  title = "PCA score plot UCH", design = UCH$design,
  color = "Hippurate", shape = "Citrate"
)

# design is recovered from lmpDataList through pcaBySvd()
ResPCA <- pcaBySvd(lmpDataList = UCH)

pcaScorePlot(
  resPcaBySvd = ResPCA, axes = c(1, 2),
  title = "PCA score plot UCH",
  color = "Hippurate", shape = "Citrate"
)
```

---

pcaScreePlot *Scree Plot*

Description

Returns a bar plot of the percentage of variance explained by each Principal Component calculated by pcaBySvd.

Usage

```
pcaScreePlot(
  resPcaBySvd,
  nPC = 5,
  title = "PCA scree plot",
  theme = theme_bw()
)
```

Arguments

resPcaBySvd	A list corresponding to the output value of pcaBySvd.
nPC	An integer with the number of Principal Components to plot.
title	Plot title.
theme	ggplot2 theme, see ?ggtheme for more info.

Value

A ggplot2 PCA scree plot

Examples

```
data("UCH")
resPCA <- pcaBySvd(UCH$outcomes)
pcaScreePlot(resPCA, nPC = 4)
```

---

plotDesign *Plot of the design matrix*

Description

Provides a graphical representation of the experimental design. It allows to visualize factors levels and check the design balance.

Usage

```
plotDesign(
  design = NULL,
  lmpDataList = NULL,
  x = NULL,
  y = NULL,
  rows = NULL,
  cols = NULL,
  title = "Plot of the design",
  theme = theme_bw()
)
```

Arguments

design	A data.frame representing the $n \times k$ "freely encoded" experimental design. Can be NULL if lmpDataList is defined.
lmpDataList	If not NULL, a list with outcomes, design and formula, as outputted by data2LmpDataList.
x	By default, the first column of design ; otherwise if not NULL, a character string giving the column name of design to be used for the x-axis. The column needs to be a factor.
y	By default, the second column of design if present ; otherwise if not NULL, a character string giving the column name of design to be used for the y-axis.
rows	By default, the fourth column of design if present ; otherwise if not NULL, a character vector with one or several column name(s) of design to be used for faceting along the rows. The column needs to be a factor.
cols	By default, the third column of design if present ; otherwise if not NULL, a character vector with one or several column name(s) of design to be used for faceting along the columns. The column needs to be a factor.
title	Plot title.
theme	The ggplot2 theme, see ?ggtheme for more info.

Details

Either design or lmpDataList need to be defined. If both are given, the priority goes to design. The default behavior (parameters x, y, cols and rows are NULL) uses the first four columns of df. If at least one of these arguments is not NULL, the function will only use the non NULL parameters to be displayed.

Value

A ggplot2 plot of the design matrix.

Examples

```
### trout data
data(trout)

plotDesign(design = trout$design, x = "Day", y = "Treatment")

# equivalent to:
plotDesign(lmpDataList = trout, x = "Day", y = "Treatment")

### mtcars
data(mtcars)
library(tidyverse)
df <- mtcars %>%
  dplyr::select(cyl, vs, am, gear, carb) %>%
  as.data.frame() %>%
  dplyr::mutate(across(everything(), as.factor))

# Default behavior: display the 4 first factors in the design
plotDesign(design = df)

# 2 factors
plotDesign(
  design = df, x = "cyl", y = "vs",
  cols = NULL, rows = NULL
)
# 3 factors
plotDesign(
  design = df, x = "cyl", y = "vs",
  cols = NULL, rows = c("am")
)
# 4 factors
plotDesign(
  design = df, x = "cyl", y = "vs",
  cols = c("gear"), rows = c("am")
)
# 5 factors
plotDesign(
  design = df, x = "cyl", y = "vs",
  cols = c("gear"), rows = c("am", "carb")
)

plotDesign(
  design = df, x = "cyl", y = "vs",
  cols = c("vs"), rows = c("am", "carb")
)
```

```
### UCH
data("UCH")
plotDesign(design = UCH$design, x = "Hippurate", y = "Citrate", rows = "Day")
```

`plotLine` *Line plot*

Description

Generates the response profile of one or more observations i.e. plots of one or more rows of the outcomes matrix on the y-axis against the m response variables on the x-axis. Depending on the response type (spectra, gene expression...), point, line or segment plots can be used.

Usage

```
plotLine(
  Y = NULL,
  lmpDataList = NULL,
  rows = 1,
  type = c("l", "p", "s"),
  title = "Line plot",
  xlab = NULL,
  ylab = NULL,
  xaxis_type = c("numeric", "character"),
  stacked = FALSE,
  ncol = 1,
  nrow = NULL,
  facet_label = NULL,
  hline = 0,
  size = 0.5,
  color = NULL,
  shape = 1,
  theme = theme_bw(),
  ang_x_axis = NULL
)
```

Arguments

`Y`	A numerical matrix containing the rows to be drawn. Can be `NULL` if `lmpDataList` is defined.
`lmpDataList`	If not `NULL`, a list with outcomes, design and formula, as outputted by `data2LmpDataList`.
`rows`	A vector with either the row name(s) of the Y matrix to plot (character) or the row index position(s) (integer). Default to 1.
`type`	Type of graph to be drawn: `"p"` for point, `"l"` for line (default) or `"s"` for segment.
`title`	Plot title.
`xlab`	If not `NULL`, label for the x-axis.
`ylab`	If not `NULL`, label for the y-axis.

xaxis_type	The data type of the x-axis: either "numeric" (default) or "character".
stacked	Logical. If TRUE, will draw stacked plots, otherwise will draw separate plots.
ncol	If stacked is FALSE, the number of columns to represent the separate plots. Default to 1.
nrow	If stacked is FALSE, the number of rows to represent the separate plots.
facet_label	If stacked is FALSE, the labels of the separate plots.
hline	If not NULL, draws (a) horizontal line(s), by default at y intercept = 0.
size	Argument of length 1 giving the points size (if type == "p") or the line size (if type == "l" or "s").
color	If not NULL, argument of length 1 with possible values: "rows", a color name (character) or a numeric value representing a color.
shape	The points shape (default = 1) if type == "p".
theme	The ggplot2 theme (default: theme_bw()), see ?ggtheme for more info.
ang_x_axis	If not NULL, rotation angle to rotate the x-axis text (based on the argument axis.text.x from ggplot2::theme())

Details

Either Y or lmpDataList need to be defined. If both are given, the priority goes to Y.

Value

A ggplot2 line plot.

Examples

```
data("UCH")
plotLine(Y = UCH$outcomes)

plotLine(lmpDataList = UCH)

# separate plots
plotLine(Y = UCH$outcomes, rows = seq(1, 8), hline = NULL)
plotLine(Y = UCH$outcomes, rows = seq(1, 8), color = 2)
plotLine(Y = UCH$outcomes, rows = seq(1, 8), ncol = 2)
plotLine(
  Y = UCH$outcomes, type = "p",
  rows = seq(1, 8), ncol = 2
)

# stacked plots
library(ggplot2)
plotLine(
  Y = UCH$outcomes, rows = seq(1, 1),
  stacked = TRUE, color = "rows"
) +
  scale_color_brewer(palette = "Set1")
```

plotMeans *Means plot*

Description

For a given response variable, draws a plot of the response means by levels of up to three categorical factors from the design. When the design is balanced, it allows to visualize main effects or interactions for the response of interest. For unbalanced designs, this plot must be used with caution.

Usage

```
plotMeans(
  Y = NULL,
  design = NULL,
  lmpDataList = NULL,
  cols = NULL,
  x,
  z = NULL,
  w = NULL,
  title = NULL,
  xlab = NULL,
  ylab = NULL,
  color = NULL,
  shape = NULL,
  linetype = NULL,
  size = 2,
  hline = NULL,
  theme = theme_bw()
)
```

Arguments

Y	A numerical matrix containing the columns to be drawn. Can be NULL if lmpDataList is defined.
design	A $n \times k$ "freely encoded" experimental design data.frame. Can be NULL if lmpDataList is defined.
lmpDataList	If not NULL, a list with outcomes, design and formula, as outputted by data2LmpDataList.
cols	A vector with either the column name(s) of the Y matrix to plot (character) or the column index position(s) (integer).
x	A character string giving the design factor whose levels will form the x-axis.
z	A character string giving the design factor whose levels will form the traces.
w	A character string giving the design factor whose levels will be used for the facet.
title	Plot title.
xlab	If not NULL, the label for the x-axis.
ylab	If not NULL, the label for the y-axis.
color	If not NULL, the color of the points and the line.

shape	If not NULL, the points shape.
linetype	If not NULL, the line type.
size	Points size.
hline	If not NULL, draws (a) horizontal line(s).
theme	The ggplot2 theme, see ?ggtheme for more info.

Details

Either Y or lmpDataList need to be defined. If both are given, the priority goes to Y. The same rule applies for design or lmpDataList.

Value

A list of ggplot2 means plot(s).

Examples

```
data("UCH")
# 1 factor
plotMeans(
  Y = UCH$outcomes, design = UCH$design, cols = "4.0628702",
  x = "Hippurate", color = "blue"
)

# equivalent to:
plotMeans(
  lmpDataList = UCH, cols = "4.0628702",
  x = "Hippurate", color = "blue"
)

# 2 factors
plotMeans(
  Y = UCH$outcomes, design = UCH$design, cols = c(364, 365),
  x = "Hippurate", z = "Time", shape = c(15, 1)
)

# 3 factors
plotMeans(
  Y = UCH$outcomes, design = UCH$design, cols = c(364, 365),
  x = "Hippurate", z = "Time", w = "Citrate", linetype = c(3, 3)
)
```

---

plotScatter *Scatter plot*

Description

Produces a plot describing the relationship between two columns of the outcomes matrix Y. Colors and symbols can be chosen for the levels of the design factors. Ellipses, polygons or segments can be added to group different sets of points on the graph.

Usage

```
plotScatter(
  Y = NULL,
  design = NULL,
  lmpDataList = NULL,
  xy,
  color = NULL,
  shape = NULL,
  points_labs = NULL,
  title = "Scatter plot",
  xlab = NULL,
  ylab = NULL,
  size = 2,
  size_lab = 3,
  drawShapes = c("none", "ellipse", "polygon", "segment"),
  typeEl = c("norm", "t", "euclid"),
  levelEl = 0.9,
  alphaPoly = 0.4,
  theme = theme_bw(),
  drawOrigin = FALSE
)
```

Arguments

Y	A $n \times m$ matrix with n observations and m variables. Can be NULL if lmpDataList is defined.
design	A $n \times k$ "freely encoded" experimental design data.frame. Can be NULL if lmpDataList is defined.
lmpDataList	If not NULL, a list with outcomes, design and formula, as outputted by data2LmpDataList.
xy	x- and y-axis values: a vector of length 2 with either the column name(s) of the Y matrix to plot (character) or the index position(s) (integer).
color	If not NULL, a character string giving the column name of design to be used as color. Currently treated as a discrete variable.
shape	If not NULL, a character string giving the column name of design to be used as shape. Currently treated as a discrete variable.
points_labs	If not NULL, a character vector with point labels.
title	Plot title.
xlab	If not NULL, label for the x-axis.
ylab	If not NULL, label for the y-axis.
size	The points size, by default 2.
size_lab	The size of points labels, by default 3.
drawShapes	Multiple shapes can be drawn based on the color: "none" for no shape (default), "ellipse" (ellipses with ggplot2::stat_ellipse()), "polygon" (polygons with ggplot2::geom_polygon()) or "segment" (segment from the centroids with ggplot2::geom_segment()).
typeEl	The type of ellipse, either "norm" (multivariate normal distribution, the default), "t" (multivariate t-distribution) or "euclid" (draws a circle with the radius equal to level, representing the euclidean distance from the center).

levelEl	The confidence level at which to draw an ellipse, by default 0.9.
alphaPoly	The degree of transparency for polygons, by default 0.4.
theme	The ggplot2 theme (default: theme_bw()), see ?ggtheme for more info.
drawOrigin	If TRUE, draws horizontal and vertical intercepts at (0,0).

Details

Either Y or lmpDataList need to be defined. If both are given, the priority goes to Y. The same rule applies for design or lmpDataList.

Value

A ggplot2 scatter plot.

Examples

```
data("UCH")

# Without the design info
plotScatter(Y = UCH$outcomes, xy = c(453, 369))

# equivalent to:
plotScatter(lmpDataList = UCH, xy = c(453, 369))

# With color and shape
plotScatter(
  lmpDataList = UCH,
  xy = c(453, 369), color = "Hippurate",
  shape = "Citrate"
)

# equivalent to:
plotScatter(
  Y = UCH$outcomes, design = UCH$design,
  xy = c(453, 369), color = "Hippurate",
  shape = "Citrate"
)

# With color and shapes
plotScatter(
  Y = UCH$outcomes, design = UCH$design,
  xy = c(453, 369), color = "Hippurate",
  drawShapes = "ellipse"
)

plotScatter(
  Y = UCH$outcomes, design = UCH$design,
  xy = c(453, 369), color = "Hippurate",
  drawShapes = "polygon"
)

plotScatter(
  Y = UCH$outcomes, design = UCH$design,
  xy = c(453, 369), color = "Hippurate",
  drawShapes = "segment"
)
```

```
# With customized shapes
library(ggplot2)
plotScatter(
  Y = UCH$outcomes, design = UCH$design,
  xy = c(453, 369), shape = "Hippurate", size = 3
) +
  scale_discrete_identity(
    aesthetics = "shape",
    guide = "legend"
  )

plotScatter(
  Y = UCH$outcomes, design = UCH$design,
  xy = c(453, 369), shape = "Hippurate"
) +
  scale_shape_discrete(solid = FALSE)

plotScatter(
  Y = UCH$outcomes, design = UCH$design,
  xy = c(453, 369), shape = "Hippurate"
) +
  scale_shape_manual(values = c(15, 16, 17))

# With labels
plotScatter(
  Y = UCH$outcomes, design = UCH$design,
  xy = c(453, 369), points_labs = rownames(UCH$design)
)
```

`plotScatterM` *Scatter plot matrix*

Description

Produces a scatter plot matrix between the selected columns of the outcomes matrix Y choosing specific colors and symbols for up to four factors from the design on the upper and lower diagonals.

Usage

```
plotScatterM(
  Y = NULL,
  design = NULL,
  lmpDataList = NULL,
  cols,
  labelVector = NULL,
  title = "Scatterplot matrix",
  varname.colorup = NULL,
  varname.colordown = NULL,
  varname.pchup = NULL,
  varname.pchdown = NULL,
  vec.colorup = NULL,
```

```
  vec.colordown = NULL,
  vec.pchup = NULL,
  vec.pchdown = NULL
)
```

Arguments

Y	$n \times m$ matrix with n observations and m variables. Can be NULL if lmpDataList is defined.
design	A $n \times k$ "freely encoded" experimental design data.frame. Can be NULL if lmpDataList is defined.
lmpDataList	If not NULL, a list with outcomes, design and formula, as outputted by data2LmpDataList.
cols	A vector with either the column names of the Y matrix to plot (character) or the column index positions.
labelVector	Labels to display on the diagonal. If NULL, the column names are deduced from cols.
title	Title of the graph.
varname.colorup	A character string with the name of the variable used to color the upper triangle.
varname.colordown	A character string with the name of the variable used to color the lower triangle.
varname.pchup	A character string with the name of the variable used to mark points for the upper triangle.
varname.pchdown	A character string with the name of the variable used to mark points for the lower triangle.
vec.colorup	A color vector (character or numeric) with a length equal to the number of levels of varname.colorup.
vec.colordown	A color vector (character or numeric) with a length equal to the number of levels of varname.colordown.
vec.pchup	A symbol vector (character or numeric) with a length equal to the number of levels of varname.pchup.
vec.pchdown	A symbol vector (character or numeric) with a length equal to the number of levels of varname.pchdown.

Details

Either Y or lmpDataList need to be defined. If both are given, the priority goes to Y. The same rule applies for design or lmpDataList.

Value

A matrix of scatter plots.

Examples

```
data("UCH")

# basic usage
plotScatterM(
```

```
  Y = UCH$outcomes, design = UCH$design, cols = c(1:4)
)

# equivalent to:
plotScatterM(
  lmpDataList = UCH, cols = c(1:4)
)

# with optionnal arguments
plotScatterM(
  Y = UCH$outcomes, design = UCH$design, cols = c(1:4),
  varname.colorup = "Hippurate", varname.colordown = "Citrate",
  varname.pchup = "Time", varname.pchdown = "Day",
  vec.colorup = c(2, 4, 1),
  vec.colordown = c("orange", "purple", "green"),
  vec.pchup = c(1, 2), vec.pchdown = c("a", "b")
)
```

`trout` *trout: the Rainbow trouts transcriptomic dataset*

Description

This dataset comes from the study of the modulation of immunity in rainbow trout (Oncorhynchus mykiss) by exposure to cadmium (Cd) combined with polyunsaturated fatty acids (PUFAs) enriched diets [Cornet et al., 2018].

The responses were quantified by measuring the modification of the expression of 15 immune-related genes (m = 15) by RT-qPCR (reverse transcription quantita- tive polymerase chain reaction). The experiment was carried out on 72 trouts and 3 factors were considered in the experimental design:

`Day` Measurements on trouts were collected on days 28, 70 and 72

`Treatment` Four polyunsaturated fatty acid diets: alpha-linolenic acid (ALA), linoleic acid (LA), eicosapentaenoic acid (EPA) and docosahexaenoic acid (DHA)

`Exposure` Trouts were exposed (level = 2) or not (level = 0) to high cadmium concentrations.

This gives a $3 \times 4 \times 2$ factorial design. Each of the 24 trials corresponds to a different aquarium. Three fish were analysed (3 replicates) for each condition, giving a total of 72 observations.

In the limpca vignette, some outliers are first detected and removed from the dataset. The data of each aquarium are then transformed in log10 and mean aggregated in order to avoid the use of an aquarium random factor in the statictical model. Data are then centered and scaled by column. The ASCA+/APCA+ analysis is then applyed on the transformed data.

Usage

```
data("trout")
```

Format

A list of 3 : the experimental design, the outcomes for every observation and the formula considered to analyze the data. Caution ! Here, the data must first be aggregated before being analyzed with ASCA+ (see details in related vignette)

`outcomes` A dataset with 72 observations and 15 response variables

`formula` The suggested formula to analyze the data

`design` The experimental design of 72 observations and 4 explanatory variables

Details

The data must first be aggregated before being analyzed with limpca. This will remove the hierarchy in the design and allow to apply a classical fixed effect general linear model to the data. See more details in the trout vignette (`print(vignette(topic = "Trout", package = "limpca"))`).

Source

Cornet, V., Ouaach, A., Mandiki, S., Flamion, E., Ferain, A., Van Larebeke, M., Lemaire, B., Reyes Lopez F., Tort, L., Larondelle, Y. and Kestemont, P. (2018). Environmentally-realistic concentration of cadmium combined with polyunsaturated fatty acids enriched diets modulated non-specific immunity in rainbow trout. *Aquatic Toxicology*, **196**, 104–116. https://doi.org/10.1016/j.aquatox.2018.01.012

Benaiche, N. (2022). *Stabilisation of the R package LMWiRe – Linear Models for Wide Responses.* Prom. : Govaerts, B. Master thesis. Institut de statistique, biostatistique et sciences actuarielles, UCLouvain, Belgium. http://hdl.handle.net/2078.1/thesis:33996

Examples

```
data("trout")
```

UCH *UCH: the Urine Citrate-Hippurate metabolomic dataset*

Description

This dataset comes from a 1H NMR analysis of urine of female rats with hippuric and citric acid were added to the samples in different known concentrations.

Usage

```
data("UCH")
```

Format

A list of length 3: the experimental design (‘design‘), the outcomes for every observations (‘outcomes‘) and the formula considered to analyze the data (‘formula‘).

`design` A data.frame with the experimental design of 34 observations and 5 explanatory variables: `Hippurate`: concentration of hippuric acid; `Citrate`: concentration of citric acid; `Dilution`: dilution, here all the samples are diluted with a dilution rate of 50 %; `Day`: for each medium, the preparation of the mixtures were performed in two series; `Time`: each mixture or experimental condition was repeated twice.

`outcomes` A numerical matrix with 34 observations and 600 response variables

`formula` A character string with the suggested formula to analyze the data

Details

The UCH vignette can be accessed with: (`print(vignette(topic = "UCH", package = "limpca"))`).

The database has been experimentally created in order to control the spectral locations of the biomarkers to find (i.e. `Hippurate` and `Citrate`). This property allows us to evaluate the performances of the data analysis of various statistical methods. This urine experimental database is also designed in order to explore the influence on spectra of intra-sample 1H NMR replications (`Time`), and inter-day 1H NMR measurements (`Day`).

The model formula is:

`outcomes = Hippurate + Citrate + Time + Hippurate:Citrate + Time:Hippurate + Time:Citrate + Hippurate:Citrate:Time`

Source

Martin, M. (2020). *Uncovering informative content in metabolomics data: from pre-processing of 1H NMR spectra to biomarkers discovery in multifactorial designs.* Prom.: Govaerts, B. PhD thesis. Institut de statistique, biostatistique et sciences actuarielles, UCLouvain, Belgium. http://hdl.handle.net/2078.1/227671

Examples

```
data("UCH")
str(UCH)
```

Index