Package 'MetabolomicsBasics'

April 18, 2021

Type Package

Title Basic Functions to Investigate Metabolomics Data Matrices

Version 1.2

Date 2021-04-16

Author Jan Lisec [aut, cre]

Description A set of functions to investigate raw data from (metabol)omics experiments intended to be used on a raw data matrix, i.e. following peak picking and signal deconvolution. Functions can be used to normalize data, detect biomarkers and perform sample classification.

License GPL-3

Depends R(>= 2.10.0)

biocViews

LazyData true

Imports C50, caret, e1071, mixOmics, pcaMethods, plyr, rpart, ropls, rlang

Encoding UTF-8

RoxygenNote 7.1.1

NeedsCompilation no

Maintainer Jan Lisec <jan.lisec@bam.de>

Repository CRAN

Date/Publication 2021-04-18 21:40:49 UTC

R topics documented:

`AdjustSymbols` *AdjustSymbols.*

Description

`AdjustSymbols` will generate plotting character and color vectors based on experimental factors.

Usage

```
AdjustSymbols(cols = NULL, pchs = NULL, colorset = NULL, symbolset = NULL)
```

Arguments

`cols`	Factor (color output) or numeric (greyscale output) vector or NULL (omitted).
`pchs`	Factor vector or NULL (omitted).
`colorset`	Can be selectively specified here. If NULL set automatically, else can be explicitly provided.
`symbolset`	Can be selectively specified here. If NULL upt o 5 nice symbols are selected automatically where background can be colored.

Details

not yet

Value

data.frame with two columns (cols, pchs). Will be used by several plotting functions automatically.

Examples

```
# load data and plot using provided color scheme
utils::data(raw, package = "MetabolomicsBasics")
utils::data(sam, package = "MetabolomicsBasics")
head(sam)
plot(y=raw[,1], x=as.numeric(sam$GT), pch=sam$pchs, bg=sam$cols)

# change colors to greyscale
head(AdjustSymbols(cols=sam$GT, pchs=sam$Origin))
tmp.set <- grDevices::rainbow(length(levels(sam$GT)))
head(AdjustSymbols(cols=sam$GT, pchs=sam$Batch, colorset=tmp.set))
plot(raw[,1]~sam$GT, col=unique_labels(sam=sam, g="GT")[,"cols"])
sam$cols <- AdjustSymbols(cols=as.numeric(sam$GT))
plot(raw[,1]~sam$GT, col=unique_labels(sam=sam, g="GT")[,"cols"])
```

`CheckForOutliers` *CheckForOutliers.*

Description

`CheckForOutliers` will evaluate a numeric vector and check if outliers within groups based on group mean+-n*sd.

Usage

```
CheckForOutliers(
  x = NULL,
  group = NULL,
  n_sd = 3,
  method = c("idx", "logical", "dist")
)
```

Arguments

`x`	Numeric vector.
`group`	Factor vector of length(x).
`n_sd`	Cutoff for outliers in E being mean(group)+-n_sd*sd(group) where group values are calculated without the outlier candidate.
`method`	Different variants of the result value. See details.

Details

The numeric will be split by groups and each value will be evaluated with respect to its distance to the group mean (calculated out of the other values in the group). Distance here means the number of standard deviations the value is off the group mean. With different choices of `method` the output can be switched from the calculated fold-distances to a boolean of length(x) or and Index vector giving the outliers directly (see examples).

Value

Depending on method. See details.

Examples

```
set.seed(0)
x <- runif(10)
x[1] <- 2
group <- gl(2,5)
CheckForOutliers(x, group, method="dist")
CheckForOutliers(x, group, method="logical")
CheckForOutliers(x, group, method="idx")
graphics::par(mfrow=c(1,2))
bg <- c(3,2)[1+CheckForOutliers(x, group, method="logical")]
graphics::plot(x=as.numeric(group), y=x, pch=21, cex=3, bg=bg, main="n_sd=3",las=1,xlim=c(0.5,2.5))
bg <- c(3,2)[1+CheckForOutliers(x, group, n_sd=4, method="logical")]
graphics::plot(x=as.numeric(group), y=x, pch=21, cex=3, bg=bg, main="n_sd=4",las=1,xlim=c(0.5,2.5))
graphics::par(mfrow=c(1,1))

# load raw data and sample description
utils::data(raw, package = "MetabolomicsBasics")
utils::data(sam, package = "MetabolomicsBasics")

# no missing data in this matrix
all(is.finite(raw))

# check for outliers (computing n-fold sd distance from group mean)
tmp <- apply(raw, 2, CheckForOutliers, group=sam$GT, method="dist")
# plot a histogram of the observed distances
graphics::hist(tmp, breaks=seq(0,ceiling(max(tmp))), main="n*SD from mean", xlab="n")

# Calculate the amount of values exceeding five-sigma and compare with a standard gaussian
table(tmp>5)
round(100*sum(tmp>5)/length(tmp),2)

gauss <- CheckForOutliers(x=rnorm(prod(dim(raw))), method="dist")
sapply(1:5, function(i) {data.frame("obs"=sum(tmp>i), "gauss"=sum(gauss>i))})

# compare a PCA w/wo outliers
RestrictedPCA(dat=raw, sam=sam, use.sam=sam$GT%in%c("Mo17","B73"), group.col="GT",
        fmod="GT+Batch+Order", P=1, sign.col="GT", legend.x=NULL, text.col="Batch", medsd=TRUE)
raw_filt <- raw
raw_filt[tmp>3] <- NA
RestrictedPCA(dat=raw_filt, sam=sam, use.sam=sam$GT%in%c("Mo17","B73"), group.col="GT",
        fmod="GT+Batch+Order", P=1, sign.col="GT", legend.x=NULL, text.col="Batch", medsd=TRUE)
```

ClassificationCV *ClassificationCV.*

Description

ClassificationCV will perform a classification using SVM's and/or Decision Trees including cross validation on a data set according to a provided grouping vector.

Usage

```
ClassificationCV(
  d = NULL,
  g = NULL,
  n = 1,
  k = 1,
  rand = F,
  method = c("svm", "C50", "rpart", "ropls")[1],
  method.control = list(),
  silent = FALSE
)
```

Arguments

d	Data matrix or data.frame with named rows (samples) and columns (traits).
g	Group-vector, factor.
n	Replicates of classifications.
k	Number of folds per replicate.
rand	Randomize Group-vector (and apply according n and k to this randomization).
method	Currently svm, ropls and decison tree methods C50 and rpart are supported.
method.control	A list of parameters, forwarded to the respective classification function.
silent	Logical. Set TRUE to supress progress bar and warnings.

Details

This function allows to demonstrate the functionality of different classification tools with respect to building classfier for metabolomics data.

Value

A list of classification results which can be analyzed for accuracy, missclassified samples etc.

Examples

```
# check the examples in \code{\link{ClassificationWrapper}} for automatic multifold analysis
```

`ClassificationHistogram` *ClassificationWrapper.*

Description

`ClassificationWrapper` will do classification using SVM's and/or Decision Trees including cross validation.

Usage

```
ClassificationHistogram(out_classific = NULL, breaks = seq(0, 1, 0.05), ...)
```

Arguments

`out_classific`	Output of ClassificationWrapper.
`breaks`	Breaks for histogram.
`...`	Passed on to `par`. Useful to adjust `cex`.

Details

not yet

Value

Classification results as list.

Examples

```
# check the examples in \code{\link{ClassificationWrapper}}
```

`ClassificationWrapper` *ClassificationWrapper.*

Description

`ClassificationWrapper` will do classification using SVM's and/or Decision Trees including cross validation.

Usage

```
ClassificationWrapper(
  d = NULL,
  g = NULL,
  n = 100,
  n_rand = 1,
  k = 5,
  method = c("C50", "svm", "rpart", "ropls"),
  train = NULL,
  method.control = list(),
  silent = FALSE
)
```

Arguments

d	data, matrix or data.frame !! needs row/col-names.
g	Group-vector, factor.
n	replicates of classifications, i.e. number of different split into folds.
n_rand	different number of randomizations, see Details.
k	Fold cross validation.
method	Currently svm, ropls and decison tree methods (C50 and rpart) are supported.
train	Either NULL (random permutations) or an index vector for a training subset out of g.
method.control	A list of parameters, forwarded to the selected methods function.
silent	Logical. Set TRUE to supress progress bar and warnings.

Details

n_rand will influence how permutation testing for robustness is conducted. If n_rand=1 than samples will be permuted exactly one time and subjected to n replications (with respect to fold splitting). If n_rand>1, samples will be permuted this many times but number of replications will be lowered to limit processing time. A good compromise is to balance both, using less replications than for observed data but on several randomizations.

Value

Classification results as list.

Examples

```
utils::data(raw, package = "MetabolomicsBasics")
utils::data(sam, package = "MetabolomicsBasics")
gr <- sam$Origin

# establish a basic rpart model and render a fancy plot including the accuracy
class_res <- ClassificationWrapper(d=raw, g=gr, method=c("rpart","svm"), n=3, k=3)
ClassificationHistogram(class_res)
```

`find_boundaries` *find_boundaries.*

Description

`find_boundaries` will determine peak boundaries within a BPC or mass trace.

Usage

```
find_boundaries(
  int = NULL,
  rt = NULL,
  p = which.max(int),
  k = 3,
  bl = min(int),
  local_min = int[p]
)
```

Arguments

`int`	The measured intensity of the ion mass (obviously ordered according to consecutive RTs).
`rt`	The respective retention times (can be omitted as currently not used).
`p`	The anticipated peak position (as index of int) if several peaks are within the mass trace.
`k`	The smoothing window parameter (provided to runmed).
`bl`	The baseline value. Can be provided explicitly if automatic determination is insufficient.
`local_min`	This is practically the upper end of the baseline. It can be set to avoid boundary detection at local minima (e.g. for peaks suffering ion suppression).

Details

It is yet another peak finder or, more precisely, it is a function to identify two RT values which flank a intensity maximum which is required if one would like to integrate the peak area.

Value

Numeric vector of length=2 specifying the start and end index of the peak.

Examples

```
int <- sin(seq(-0.75*pi,1.75*pi,by=0.1))
plot(int)
abline(v=find_boundaries(int=int))
abline(v=find_boundaries(int=int, p=1))
```

MBoxplot *MBoxplot*

Description

MBoxplot will generate an annotated boxplot. A unifying function for MS-data Boxplots based on \'raw\' and \'sam\'.

Usage

```
MBoxplot(
  pk = pk,
  raw = NULL,
  sam = NULL,
  met = NULL,
  g = NULL,
  flt = NULL,
  an = NULL,
  plot_sample_n = FALSE,
  txt = NULL,
  cex.txt = 0.5,
  plot_rel_axis = NULL,
  ...
)
```

Arguments

pk	Colname of raw to plot if pk is character OR the colnum number if pk is numeric.
raw	Plotting data as samples (rows) x metabolites (cols).
sam	Sample table.
met	Containing at minimum columns for annotation (see parameter an) and nrow(met) should be ncol(raw).
g	Grouping vector if Group not contained in sam.
flt	Filter to exclude certain samples (T/F) vector.
an	Switch to include annotation (from met) in the boxplot providing a character vector of colnames from met.
plot_sample_n	Amend each box with the number of finite values which were a basis for plotting this group.
txt	Character vector with information per sample to be plotted on top of the box as text.
cex.txt	Specify size of annotation text.
plot_rel_axis	Specify one level of g (or sam$Group) which to express the data relative against.
...	Further options parsed to boxplot.

Details

not yet

Value

Nothing. Will produce a plot (or file if specified).

Examples

```
x <- data.frame("y"=runif(36), "GT"=gl(3,12), "TP"=factor(rep(rep(1:3,each=4),3)))
x <- cbind(x, AdjustSymbols(cols=x$GT, pchs=x$TP))
MBoxplot(pk="y", raw=x, sam=x, met=data.frame("Peak"="y", "Test"=I("info")),
        g=interaction(x$GT, x$TP), an="Test", plot_n_samples=TRUE, txt=rownames(x))
```

---

met *Metabolite table*

Description

This data frame contains the metabolite definition of 112 metabolites according to the cols of raw.

Usage

```
data(met)
```

Format

An object of class `data.frame` with 112 rows and 2 columns.

Author(s)

Jan Lisec <jan.lisec@charite.de>

References

doi: 10.1111/j.1365313X.2011.04689.x

Examples

```
data(met)
str(met)
```

MetaboliteANOVA *MetaboliteANOVA*

Description

MetaboliteANOVA will perform an ANOVA on columns of a data matrix according to a specified model.

Usage

```
MetaboliteANOVA(
  dat = NULL,
  sam = NULL,
  model = NULL,
  method = "none",
  silent = FALSE
)
```

Arguments

dat	Data matrix (e.g. of metabolite).
sam	Sample table (same number of row as 'dat' and containing all columns specified in 'model'.
model	ANOVA model. May include +, * and : together with column names of sam (cf. Examples).
method	The method to be used in column wise multiple testing adjustment, see p.adjust.
silent	Logical. Shall the function print warnings to the console?

Details

Function is a wrapper for lm including some sanity checks. It will accept a data matrix (traits in columns), sample information (data.frame) and a potential model as input, compute an ANOVA per column and return the respective P-values in a named matrix for further plotting or export.

Value

A named matrix of P-values (rows=metabolites/traits; cols=ANOVA factors).

Examples

```
# load raw data and sample description
utils::data(raw, package = "MetabolomicsBasics")
utils::data(sam, package = "MetabolomicsBasics")
# compute P-values according to specified ANOVA model (simple and complex)
head(m1 <- MetaboliteANOVA(dat=raw, sam=sam, model="GT"))
head(m2 <- MetaboliteANOVA(dat=raw, sam=sam, model="GT+Batch+Order+MP"))
# compare P-values for one factor determined in both models
```

```
hist(log10(m2[,"GT"])-log10(m1[,"GT"]), main="")
```

`msconvert` *msconvert.*

Description

`msconvert` is calling ProteoWizards MSConvert as a command line tool on Windows.

Usage

```
msconvert(
  files = NULL,

   msc_exe = "C:\\Program Files\\ProteoWizard\\ProteoWizard 3.0.11856\\msconvert.exe",
  args = c("--filter \"peakPicking cwt snr=0.01 peakSpace=0.1 msLevel=1\"",
   "--filter \"scanTime [0,3600]\"", "--filter \"metadataFixer\"", "--mzML", "--32",
     "--zlib")
)
```

Arguments

`files`	A character vector of MS data files (wiff, raw, d, ...).
`msc_exe`	The path to the installed msconvert.exe.
`args`	The arguments passed to msconvert on the commandline (see details for documentation).

Details

It is a quick and dodgy function to show how to convert vendor MS data into an open format (mzML). You will have to download/install MSConvert prior to usage, and probably adjust the arguments according to your needs. Arguments are documented here http://proteowizard.sourceforge.net/tools/msconvert.html. If you don't know where the msconvert.exe is installed you can check for the correct path using `list.files(path="C:/",pattern="^msconvert.exe$",recursive = TRUE)`.

Value

Only some infomative output to the console. The specified MS data files will be converted to mzML within the same folder.

PlotMetabolitePCA *PlotMetabolitePCA*

Description

PlotMetabolitePCA will show PC1 and PC2 of a pcaMethods object and generate a flexible plot.

Usage

```
PlotMetabolitePCA(
  pca_res = NULL,
  sam = NULL,
  g = NULL,
  medsd = FALSE,
  text.col = "ID",
  legend.x = "bottomleft",
  comm = NULL
)
```

Arguments

pca_res	A pcaRes object from the pcaMethods package.
sam	Sample table including columns 'cols', 'pchs' (for data point color and shape) and 'ID' (to label data points) 'Group' (to split cols for legend) 'MP' (to adjust point size).
g	Can be a factor vector of length=nrow(sam) and will influence legend and medsd.
medsd	Calculate mean and sd for groups and overlay PCA plot with this information.
text.col	Datapoints may be overlaid by textual information, e.g. sample ID and 'text.col' specifies the columnn name of sam to use for this purpose.
legend.x	Position of a legend or NULL to omit it.
comm	Will print commentary text to the bottom right of the plot (can be a character vector).

Details

not yet

Value

A vector fo similar length as input but with various name components removed.

Examples

```
# load raw data and sample description
utils::data(raw, package = "MetabolomicsBasics")
utils::data(sam, package = "MetabolomicsBasics")

# calculate pca Result using pcaMethods and plot
pca_res <- pcaMethods::pca(raw, method="rnipals", scale=c("none", "pareto", "uv")[2])
PlotMetabolitePCA(pca_res=pca_res, sam=sam, g=sam$GT)
# plot without legend and Group means instead
PlotMetabolitePCA(pca_res=pca_res, sam=sam, g=sam$GT, legend.x=NULL, text.col=NULL,
                  medsd=TRUE, comm=LETTERS[1:4])

sam$Group <- interaction(sam$Origin, sam$Class, sep="_")
sam[,c("cols","pchs")] <- AdjustSymbols(cols=sam$Group, pchs=sam$Group)
PlotMetabolitePCA(pca_res=pca_res, sam=sam, g=sam$Group)
```

PlotPValueHist *PlotPValueHist.*

Description

`PlotPValueHist` will take a named matrix of P-values (i.e. numeric between 0..1) and plot histograms for each column. In the easiest case this matrix is generated by MetaboliteANOVA.

Usage

```
PlotPValueHist(
  out = NULL,
  method = "BH",
  xl = "ANOVA P-values",
  yl = "Number of metabolites",
  frac.col = NULL,
  ...
)
```

Arguments

`out`	matrix/data.frame; P-value table from 'MetaboliteANOVA.R' with factors in named columns and trait P-values in rows.
`method`	Multiple testing correction method applied, piped to `p.adjust()`.
`xl`	xlab.
`yl`	ylab.
`frac.col`	Render histogram bars in stacked colors according to provided color vector (should be a vector of valid color names of length=nrow(out)).
`...`	Passed on to `par`. Useful to adjust `cex`.

Details

not yet

Value

NULL. Will generate a P-value histogram plot.

Examples

```
# load raw data and sample description
utils::data(raw, package = "MetabolomicsBasics")
utils::data(sam, package = "MetabolomicsBasics")

# compute P-values according to specified ANOVA model (simple and complex)
head(pvals <- MetaboliteANOVA(dat=raw, sam=sam, model="GT+Batch+Order"))
PlotPValueHist(out=pvals)

# adjust multiple testing correction method and y lable
PlotPValueHist(out=pvals, method="none", yl="Number of Genes")

# color bars (by chance or according to a metabolite group)
PlotPValueHist(out=pvals, method="bonferroni", frac.col=rep(2:3,length.out=nrow(pvals)))
utils::data(met, package = "MetabolomicsBasics")
met$Name[grep("ine$",met$Name)]
PlotPValueHist(out=pvals, method="bonferroni", frac.col=2+1:nrow(pvals) %in% grep("ine$",met$Name))
```

---

raw *Metabolomics data set*

Description

This data set contains log10-transformed raw data of a maize root metabolomics study for in total 112 metabolites in 120 samples.

Usage

```
data(raw)
```

Format

An object of class `matrix` (inherits from `array`) with 120 rows and 112 columns.

Author(s)

Jan Lisec <jan.lisec@charite.de>

References

doi: 10.1111/j.1365313X.2011.04689.x

Examples

```
data(raw)
dim(raw)
```

RemoveFactorsByANOVA *RemoveFactorsByANOVA.*

Description

RemoveFactorsByANOVA will remove variance from data using an ANOVA model.

Usage

```
RemoveFactorsByANOVA(
  y = NULL,
  sam = NULL,
  fmod = NULL,
  kmod = NULL,
  output = c("y_norm", "y_lm", "anova_y", "anova_y_norm", "boxplot")[1],
  remove_outliers = 0
)
```

Arguments

y	Data vector (or data matrix) to normalize (numeric + in same order as sam).
sam	data.frame containing the factors or numerical vars for ANOVA model.
fmod	Full model describing the experimental setting (provided as character string).
kmod	Reduced model describing all the biological factors to keep (provided as character string).
output	Should be y_norm in general but can be switched for testing.
remove_outliers	Should be a numeric integer x (with $x=0$: no effect; $x>=1$ remove all values which have error e with $e > abs(mean + x * sd)$).

Details

not yet

Value

Depends on output. Usually the normalized data vector (or matrix).

Examples

```
# set up sample information
sam <- data.frame("GT"=gl(4,10),
                  "TR"=rep(gl(2,5),4),
                  "Batch"=sample(gl(2,20)),
                  "Order"=sample(seq(-1,1,length.out=40)))
# set up artificial measurement data
set.seed(1)
m1=c(5,6,2,9)[sam$GT]+c(-2,2)[sam$TR]+c(-3,3)[sam$Batch]+3*sam$Order+rnorm(nrow(sam), sd=0.5)
m2=c(5,-6,2,4)[sam$GT]+c(-2,2)[sam$TR]-5*sam$Order+rnorm(nrow(sam), sd=0.8)
dat <- data.frame(m1,m2)

# apply function to remove variance
# full model incorporating all relevant factors defined in sample table
fmod="GT*TR+Batch+Order"
# reduced model: factors to be kept from full model; everything else will be removed from the data
kmod="GT*TR"
RemoveFactorsByANOVA(y=dat[,"m1"], sam=sam, fmod=fmod, kmod=kmod, output="anova_y")
RemoveFactorsByANOVA(y=dat[,"m1"], sam=sam, fmod=fmod, kmod=kmod, output="anova_y_norm")
```

`ReplaceMissingValues` *ReplaceMissingValues.*

Description

`ReplaceMissingValues` will replace missing values within a numeric matrix based on a principal component analysis.

Usage

```
ReplaceMissingValues(x, ncomp = 10, silent = FALSE)
```

Arguments

`x`	Numeric matrix.
`ncomp`	Number of components to be used.
`silent`	FALSE, suppress messages setting silent=TRUE.

Details

The nipals algorithm is used to basically perform a PCA on the sparse matrix. Missing values are imputed based on the major components observed. Please check also the 'impute.nipals' function from mixOmics – it should basically give the same functionality since the 04/2021 update.

Value

Matrix without missing values.

Examples

```
# load raw data and sample description
utils::data(raw, package = "MetabolomicsBasics")
utils::data(sam, package = "MetabolomicsBasics")

idx <- apply(raw, 2, CheckForOutliers, group=sam$GT, n_sd=5, method="logical")
sum(idx) # 215 values would be classified as outlier using a five-sigma band
old_vals <- raw[idx] # keep outlier values for comparison
raw_filt <- raw
raw_filt[idx] <- NA
raw_means <- apply(raw, 2, function(x) {
 sapply(split(x, sam$GT), mean, na.rm=TRUE)[as.numeric(sam$GT)]
})[idx]
raw_repl <- ReplaceMissingValues(x=raw_filt)
new_vals <- raw_repl[idx]
par(mfrow=c(2,1))
breaks <- seq(-0.7,1.3,0.05)
hist(raw_means-old_vals, breaks=breaks, main="", xlab="Outliers", las=1)
hist(raw_means-new_vals, breaks=breaks, main="", xlab="Replaced values", las=1)
```

`RestrictedPCA` *RestrictedPCA.*

Description

`RestrictedPCA` Combines an ANOVA based on 'fmod' and restricts a PCA using the ANOVA result as a filter.

Usage

```
RestrictedPCA(
  dat = NULL,
  sam = NULL,
  use.sam = NULL,
  group.col = NULL,
  text.col = NULL,
  fmod = NULL,
  sign.col = NULL,
  p.adjust.method = "none",
  P = 0.01,
  pcaMethods.scale = "pareto",
  n.metab.min = 20,
  ...
)
```

Arguments

dat	Metabolite matrix (samples x metabolites).
sam	Sample definition dataframe.
use.sam	Numeric index vector (or logical) to select specific samples to be included in the analysis or NULL to include all.
group.col	Column used for legend creation (column name from sam).
text.col	Column used for text annotation of data points (column name from sam).
fmod	ANOVA model to calculate before PCA.
sign.col	Which column(s) of the ANOVA result shall be used for P-value filtering (specify column names or leave on NULL to filter on all).
p.adjust.method	Method use to adjust P-values (e.g. none, BH or bonferroni).
P	P-value threshold used as a cutoff after P-value adjustment.
pcaMethods.scale	pcaMethods scale parameter (usually pareto for metabolite data).
n.metab.min	Minimum number of metabolites kept for PCA calculation (even if they exceed P).
...	Handed through to PlotMetabolitePCA.

Details

fmod should be something like 'GT*TR+Batch' to perform an ANOVA with these factors defined as columns in sam.

Value

Will generate a PCA plot (generated by PlotMetabolitePCA internally) restricted based on an ANOVA result based on MetaboliteANOVA.

Examples

```
# load raw data and sample description
utils::data(raw, package = "MetabolomicsBasics")
utils::data(sam, package = "MetabolomicsBasics")
# standard behavior
RestrictedPCA(dat=raw, sam=sam, group.col="GT")
## Not run:
# apply multiple testing using a strict P-value cutoff,
# dont show a legend but plot group mean values and sd's as overlay
RestrictedPCA(dat=raw, sam=sam, group.col="GT", p.adjust.method = "BH", P=10^-10,
            fmod="GT+Batch+Order", sign.col="GT", medsd=T, legend.x=NULL)
# limit to a subset of samples, switching the ANOVA selection of by setting P=1
# and adding text (from \code{sam}) to each data point
RestrictedPCA(dat=raw, sam=sam, use.sam=which(sam$GT%in%c("Mo17","B73")), group.col="GT",
            fmod="GT+Batch+Order", P=1, sign.col="GT", legend.x=NULL, text.col="Batch")

## End(Not run)
```

sam *Sample table*

Description

This data frame contains the sample definition of 120 samples according to the rows of raw.

Usage

```
data(sam)
```

Format

An object of class `data.frame` with 120 rows and 10 columns.

Author(s)

Jan Lisec <jan.lisec@charite.de>

References

doi: 10.1111/j.1365313X.2011.04689.x

Examples

```
data(sam)
str(sam)
```

spectra_format_converter *spectra_format_converter.*

Description

`spectra_format_converter` will generate a matrix with mz and int out of a text representation of a spectrum.

Usage

```
spectra_format_converter(txt = NULL, m_prec = 3, i_prec = 0)
```

Arguments

`txt`	Sample table.
`m_prec`	Mass precision of output spectrum.
`i_prec`	Intensity precision of output spectrum.

Details

none.

Value

Matrix with mz and int columns.

Examples

```
spectra_format_converter(txt="57.1:100 58.0001:10")
spectra_format_converter(txt="58.0001:10 57.1:100", m_prec=4)
```

---

unique_labels *unique_labels.*

Description

`unique_labels` will generate a dataframe with color and plotting character specification out of a sample table definition.

Usage

```
unique_labels(sam = NULL, g = NULL)
```

Arguments

`sam`	Sample table.
`g`	Either column name from sam containing factor column or factor of same length as sam.

Details

If a color/symbol specification exists for a sample set containing replicate groups this function will help in retrieving this information per group which is useful in boxplot or legend functions (cf. examples).

Value

Dataframe with group levels names and their color and plotting character specification.

Examples

```
utils::data(raw, package = "MetabolomicsBasics")
utils::data(sam, package = "MetabolomicsBasics")
unique_labels(sam=sam, g="GT")
```

Index