Package: metaboprep (via r-universe)

October 2, 2024

Title Metabolomics data preparation and processing pipeline

Version 1.0.1

Description Reads in raw Metabolon and Nightingale xls sheets and aids in data preparation of all metabolomics data sets.

Depends R (>= 3.5.0)

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.1.2

Imports boot, car, dplyr, ggplot2, ggpubr, knitr, magrittr, nFactors, pcaMethods, psych, pwr, RColorBrewer, readxl, tibble

Suggests corrplot, data.table, dendextend, forcats, ggfortify, gridExtra, kableExtra, MASS, purrr, readr, stats, stringr, tidyr

Repository https://mrcieu.r-universe.dev

RemoteUrl https://github.com/remlapmot/metaboprep

RemoteRef suggestions

RemoteSha ee1cdc7579561a904950409991a8e04965307bfa

Contents

`batch_normalization` *median batch normalization*

Description

This function median normalizes multivariable data often processed in batches, such as metabolomic and proteomic data sets.

Usage

```
batch_normalization(
  wdata,
  feature_data_sheet = NULL,
  sample_data_sheet = NULL,
  feature_runmode_col = NULL,
  batch_ids = NULL
)
```

Arguments

wdata	the metabolite data frame samples in row, metabolites in columns
feature_data_sheet	a data frame containing the feature annotation data
sample_data_sheet	a data frame containing the sample annotation data
feature_runmode_col	a string identifying the column name in the feature_data_sheet that identifies the run mode for each feature (metabolites of proteins).
batch_ids	a string vector, with a length equal to the number of samples in the data set that identifies what batch each sample belongs to.

Value

returns the wdata object passed to the function median normalized given the batch information provided.

Examples

```
####################################
## with a vector of batch variables
####################################
## define the data set
d1 = sapply(1:10, function(x){ rnorm(25, 40, 2) })
d2 = sapply(1:10, function(x){ rnorm(25, 35, 2) })
ex_data = rbind(d1,d2)
rownames(ex_data) = paste0("ind", 1:nrow(ex_data))
colnames(ex_data) = paste0("var", 1:ncol(ex_data))
## define the batch
batch = c( rep("A", 25), rep("B", 25)  )
## normalize by batch
norm_wdata = batch_normalization(wdata = ex_data, batch_ids = batch )
```

cramerV *Cramer's V (phi)*

Description

Calculates Cramer's V for a table of nominal variables; confidence intervals by bootstrap. Function taken from the rcompanion Rpackage.

Usage

```
cramerV(
  x,
  y = NULL,
  ci = FALSE,
  conf = 0.95,
  type = "perc",
  R = 1000,
  histogram = FALSE,
  digits = 4,
  bias.correct = FALSE,
  reportIncomplete = FALSE,
  verbose = FALSE,
  ...
)
```

Arguments

x	Either a two-way table or a two-way matrix. Can also be a vector of observations for one dimension of a two-way table.
y	If x is a vector, y is the vector of observations for the second dimension of a two-way table.
ci	If TRUE, returns confidence intervals by bootstrap. May be slow.
conf	The level for the confidence interval.
type	The type of confidence interval to use. Can be any of "norm", "basic", "perc", or "bca". Passed to boot.ci.
R	The number of replications to use for bootstrap.
histogram	If TRUE, produces a histogram of bootstrapped values.
digits	The number of significant digits in the output.
bias.correct	If TRUE, a bias correction is applied.
reportIncomplete	If FALSE (the default), NA will be reported in cases where there are instances of the calculation of the statistic failing during the bootstrap procedure.
verbose	If TRUE, prints additional statistics.
...	Additional arguments passed to chisq.test.

Details

Cramer's V is used as a measure of association between two nominal variables, or as an effect size for a chi-square test of association. For a 2 x 2 table, the absolute value of the phi statistic is the same as Cramer's V.

Because V is always positive, if `type="perc"`, the confidence interval will never cross zero. In this case, the confidence interval range should not be used for statistical inference. However, if `type="norm"`, the confidence interval may cross zero.

When V is close to 0 or very large, or with small counts, the confidence intervals determined by this method may not be reliable, or the procedure may fail.

Value

A single statistic, Cramer's V. Or a small data frame consisting of Cramer's V, and the lower and upper confidence limits.

Author(s)

Salvatore Mangiafico, <mangiafico@njaes.rutgers.edu>

References

http://rcompanion.org/handbook/H_10.html

Examples

```
## Not run:
### Example with table
data(Anderson)
fisher.test(Anderson)
cramerV(Anderson)

## End(Not run)

### Example with two vectors
Species = c(rep("Species1", 16), rep("Species2", 16))
Color   = c(rep(c("blue", "blue", "blue", "green"),4),
            rep(c("green", "green", "green", "blue"),4))
fisher.test(Species, Color)
cramerV(Species, Color)
```

`eval.power.binary` *Estimate power for a binary variable*

Description

This function allows you estimate power for a binary variable given the sample size, effect size, significance threshold.

Usage

```
eval.power.binary(N, effect, alpha)
```

Arguments

N	a numeric vector of total study sample size, cases and controls will both be defined as N/2.
effect	a numeric vector of effect size
alpha	a numeric vector of significance thresholds

Value

a matrix of parameter inputs and an estimate(s) of power are returned as a matrix

Examples

```
eval.power.binary(N = 1000, effect = seq(0.01, 0.3, by = 0.01), alpha = 0.05)
```

---

eval.power.binary.imbalanced

Estimate power for a binary variable in an imbalanced design

Description

This function allows you estimate power for a binary variable given a defined number of case samples, control samples, effect size, and significance threshold.

Usage

```
eval.power.binary.imbalanced(N_case, N_control, effect, alpha)
```

Arguments

N_case	a numeric vector of sample size of cases
N_control	a numeric vector of sample size of controls
effect	a numeric vector of effect size
alpha	a numeric vector of significance thresholds

Value

a matrix of paramater inputs and power estimates are returned as a matrix

Examples

```
eval.power.binary.imbalanced( N_case = 1000,
 N_control = 1000,
 effect = 0.01,
 alpha = 0.05 )

eval.power.binary.imbalanced( N_case = c(1000, 2000),
 N_control = c(1000, 2000),
 effect = 0.01,
 alpha = 0.05 )
```

`eval.power.cont` *estimate power for continuous variable*

Description

This function estimates power for a continuous variable given the sample size, effect size, significance threshold, and the degrees of freedom.

Usage

```
eval.power.cont(N, n_coeff, effect, alpha)
```

Arguments

`N`	Sample size
`n_coeff`	degrees of freedom for numerator
`effect`	effect size
`alpha`	significance level (Type 1 error)

Examples

```
eval.power.cont(N = 1000, n_coeff = 1, effect = 0.0025, alpha = 0.05)
```

feature.describe *summary statistics for features*

Description

This function allows you to 'describe' metabolite features using the describe() function from the psych package, as well as estimate variance, a dispersion index, the coeficent of variation, and shapiro's W-statistic.

Usage

```
feature.describe(wdata)
```

Arguments

wdata	the metabolite data matrix. samples in row, metabolites in columns

Value

a data frame of summary statistics for features (columns) of a matrix

Examples

```
ex_data = sapply(1:20, function(x){ rnorm(250, 40, 5) })
feature.describe(ex_data)
```

feature.missingness *estimate feature missingness*

Description

This function estimates feature missingess, with a step to exclude poor samples identified as those with a sample missingness greater than 50

Usage

```
feature.missingness(wdata)
```

Arguments

wdata	the metabolite data matrix. samples in row, metabolites in columns

Value

a data frame of percent missingness for each feature

Examples

```
ex_data = sapply(1:5, function(x){rnorm(10, 45, 2)})
ex_data[ sample(1:length(ex_data), 15) ] = NA
feature.missingness(wdata = ex_data )
```

`feature.outliers` *outlier sample count for a features*

Description

This function takes a matrix of data (samples in rows, features in columns) and counts the number of outlying samples each feature has.

Usage

```
feature.outliers(wdata, nsd = 5)
```

Arguments

`wdata`	the metabolite data matrix. samples in row, metabolites in columns
`nsd`	the number of standard deviation from the mean outliers are identified at. Default value is 5.

Value

a data frame out sample outlier counts for each feature (column) in the matrix

Examples

```
ex_data = sapply(1:20, function(x){ rnorm(250, 40, 5) })
s = sample(1:length(ex_data), 200)
ex_data[s] = ex_data[s] + 40
## run the function
fout = feature.outliers(ex_data)
```

`feature.sum.stats` *feature summary statistics*

Description

This function estimates feature statistics for samples in a matrix of metabolite features.

Usage

```
feature.sum.stats(
  wdata,
  sammis = NA,
  tree_cut_height = 0.5,
  outlier_udist = 5,
  feature_names_2_exclude = NA
)
```

Arguments

`wdata`	the metabolite data matrix. samples in row, metabolites in columns
`sammis`	a vector of sample missingness estimates, that is ordered to match the samples in the rows of your data matrix.
`tree_cut_height`	tree cut height is the height at which to cut the feature|metabolite dendrogram to identify "independent" features. tree_cut_height is 1-absolute(Spearman's Rho) for intra-cluster correlations.
`outlier_udist`	the interquartile range unit distance from the median to call a sample an outlier at a feature.
`feature_names_2_exclude`	A vector of feature|metabolite names to exclude from the tree building, independent feature identification process.

Value

a list object of length two, with (1) a data frame of summary statistics and (2) a hclust object

Examples

```
## define a covariance matrix
cmat = matrix(1, 4, 4 )
cmat[1,] = c(1, 0.8, 0.6, 0.2)
cmat[2,] = c(0.8, 1, 0.7, 0.5)
cmat[3,] = c(0.6, 0.7, 1, 0.6)
cmat[4,] = c(0.2, 0.5, 0.6,1)
## simulate some correlated data (multivariable random normal)
set.seed(1110)
d1 = MASS::mvrnorm(n = 250, mu = c(5, 45, 25, 15), Sigma = cmat )
```

```
set.seed(1010)
d2 = MASS::mvrnorm(n = 250, mu = c(5, 45, 25, 15), Sigma = cmat )
## simulate some random data
d3 = sapply(1:20, function(x){ rnorm(250, 40, 5) })
## define the data set
ex_data = cbind(d1,d2,d3)
rownames(ex_data) = paste0("ind", 1:nrow(ex_data))
colnames(ex_data) = paste0("var", 1:ncol(ex_data))
## add in some missingness
ex_data[sample(1:length(ex_data), 450)] = NA
## add in some technical error to two samples
m = apply(ex_data, 2, function(x){ mean(x, na.rm = TRUE) })
ex_data[c(1,10), ] = ex_data[1, ] + (m*0.00001)
## run the function
fss = feature.sum.stats(ex_data)
## feature summary table
fss$table[1:5, ]
## plot the dendrogram
plot(fss$tree, hang = -1)
```

`feature.tree.independence` *identify independent features*

Description

This function identifies independent features using Spearman's Rho, and a dendrogram tree cut step. The feature returned as 'independent' within is k-cluster is the feature with the least missingness or chosen at random in case of missingness ties.

Usage

```
feature.tree.independence(wdata)
```

Arguments

`wdata`	the metabolite data matrix. samples in row, metabolites in columns

Value

a data frame of 'k' cluster or group ids, and a 0/1 binary identifying if a feature was identified as and independent ('1') feature or not ('0').

Examples

```
cmat = matrix(1, 4, 4 )
cmat[1,] = c(1, 0.9, 0.9, 0.8)
cmat[2,] = c(0.9, 1, 0.7, 0.6)
```

```
cmat[3,] = c(0.9, 0.7, 1, 0.8)
cmat[4,] = c(0.8, 0.6, 0.8,1)
## simulate some correlated data (multivariable random normal)
set.seed(1110)
ex_data = MASS::mvrnorm(n = 250, mu = c(5, 45, 25, 15), Sigma = cmat )
## define the data set
rownames(ex_data) = paste0("ind", 1:nrow(ex_data))
colnames(ex_data) = paste0("var", 1:ncol(ex_data))
## run the function
df = feature.tree.independence(ex_data)
```

`feature_plots` *feature plots to file*

Description

This function to plots a scatter plot, a histogram, and a few summary statistics of each feature in a data frame to a pdf file

Usage

```
feature_plots(wdata, outdir = NULL, nsd = 5)
```

Arguments

`wdata`	a data frame of feature (ex: metabolite or protein) abundance levels
`outdir`	output directory path
`nsd`	number of SD from the mean to plot an outlier line on the scatter plot and histogram

Value

a ggplot2 object

Examples

```
ex_data = sapply(1:20, function(x){ rnorm(250, 40, 5) })
colnames(ex_data) = paste0("var", 1:ncol(ex_data))
feature_plots(ex_data)
```

find.cont.effect.sizes.2.sim

identify continuos trait effect sizes

Description

This function estimates an appropriate distribution of effect sizes to simulate in a continuous trait power analysis.

Usage

```
find.cont.effect.sizes.2.sim(mydata)
```

Arguments

mydata	Your metabolite data matrix, with samples in rows

Value

a vector of effect sizes

Examples

```
ex_data = sapply(1:10, function(x){ rnorm(250, 40, 5) })
find.cont.effect.sizes.2.sim(ex_data)
```

find.PA.effect.sizes.2.sim

identify effect sizes

Description

This function estimates an appropriate distribution of effect sizes to simulate in a power analysis.

Usage

```
find.PA.effect.sizes.2.sim(mydata)
```

Arguments

mydata	Your metabolite data matrix, with samples in rows

Value

a vector of effect sizes

Examples

```
ex_data = sapply(1:10, function(x){ rnorm(250, 40, 5) })
find.PA.effect.sizes.2.sim(ex_data)
```

`generate_report` *generate metaboprep summary html report*

Description

This function generates the html report.

Usage

```
generate_report(
  full_path_2_Rdatafile = "ReportData.Rdata",
  dir_4_report = "./",
 path_2_Rmd_template = file.path(system.file("rmarkdown", package = "metaboprep"),
    "metaboprep_Report_v0.Rmd")
)
```

Arguments

`full_path_2_Rdatafile`	full path to the Rdatafile
`dir_4_report`	directory to place the report
`path_2_Rmd_template`	full path to the html report template

Value

Writes a html report to file

an html file written to file

`greedy.pairwise.n.filter` *greedy selection*

Description

This function identifies features that have less than a minimum number of complete pairwise observations and removes one of them, in a greedy fashion. The need for this function is in instances where missingness is extreme between two features the number of paired observation between them may be to to be informative. Thus one, but not both should be removed from the analysis to avoid analytical error based on sample sizes.

Usage

```
greedy.pairwise.n.filter(wdata, minN = 50)
```

Arguments

wdata	the metabolite data matrix. samples in row, metabolites in columns
minN	the minimum sample size (n) for pairwise comparisons

Value

a vector of feature names

Examples

```
set.seed(123)
ex_data = sapply(1:10, function(x){ rnorm(250, 40, 5) })
## define the data set
rownames(ex_data) = paste0("ind", 1:nrow(ex_data))
colnames(ex_data) = paste0("var", 1:ncol(ex_data))
## add in some missingness
ex_data[ sample(1:250, 200) ,1] = NA
ex_data[ sample(1:250, 200) ,2] = NA
ex_data[ sample(1:250, 200) ,3] = NA
## Estimate missingness and generate plots
greedy.pairwise.n.filter(ex_data)
```

---

id.outliers *identify outliers*

Description

given a vector of data, identify those positions that are 'nsd' standard deviation units from the mean

Usage

```
id.outliers(x, nsd = 5)
```

Arguments

x	a vector of numerical values
nsd	the number of standard deviation from the mean outliers are identified at. Default value is 5.

Value

a vector indexing which samples are outliers

Examples

```
ex_data = rnbinom(500, mu = 40, size = 5)
id.outliers(ex_data, nsd = 2)
```

`make.cor.matrix` *correlation matrix*

Description

This function estimates a correlation matrix returning wither the correlation estimates or their p-values

Usage

```
make.cor.matrix(wdata, cor_method = "kendall", minN = 50, var2return = "cor")
```

Arguments

`wdata`	the metabolite data matrix. samples in row, metabolites in columns
`cor_method`	defaulted to "kendall" this is the correlation method to use in the function cor.test()
`minN`	sefaulted to 50, this is the minimum number of observations that must be available in pairs to perform analysis
`var2return`	sefaulted to "cor", other option is "pvalue" is a the flag indicating which estimate to return from the function.

Value

a matrix of correlation estimates or p-values

Examples

```
cmat = matrix(1, 4, 4 )
cmat[1,] = c(1, 0.8, 0.6, 0.2)
cmat[2,] = c(0.8, 1, 0.7, 0.5)
cmat[3,] = c(0.6, 0.7, 1, 0.6)
cmat[4,] = c(0.2, 0.5, 0.6,1)
## simulate some correlated data (multivariable random normal)
set.seed(1110)
ex_data = MASS::mvrnorm(n = 250, mu = c(5, 45, 25, 15), Sigma = cmat )
## return correlation estimates
cor_mat = make.cor.matrix(ex_data, var2return = "cor")
## return p-values
cor_mat = make.cor.matrix(ex_data, var2return = "pvalue")
```

make.tree *generate a hclust dendrogram*

Description

This estimates a dendrogram of feautres based on correlation coefficeint of your choice, and a clustering method of choice

Usage

```
make.tree(wdata, cor_method = "spearman", hclust_method = "complete")
```

Arguments

wdata	the metabolite data matrix. samples in row, metabolites in columns
cor_method	the correlation method used in the function cor(). Default is "spearman".
hclust_method	the dendrogram clustering method used in the construction of the tree. Default is "complete".

Value

an hclust object

Examples

```
cmat = matrix(1, 4, 4 )
cmat[1,] = c(1, 0.8, 0.6, 0.2)
cmat[2,] = c(0.8, 1, 0.7, 0.5)
cmat[3,] = c(0.6, 0.7, 1, 0.6)
cmat[4,] = c(0.2, 0.5, 0.6,1)
## simulate some correlated data (multivariable random normal)
set.seed(1110)
ex_data = MASS::mvrnorm(n = 250, mu = c(5, 45, 25, 15), Sigma = cmat )
## estiamte the dendrogram
tree = make.tree(ex_data)
## plot the dendrogram
plot(tree, hang = -1)
```

median_impute *median impute missing data*

Description

This function imputes features (columns) of a metabolome matrix to median estimates. Useful for PCA.

Usage

```
median_impute(wdata)
```

Arguments

wdata	the metabolite data matrix. samples in row, metabolites in columns

Value

the matrix passed to the function but with NA's imputed to each columns median value.

Examples

```
ex_data = sapply(1:100, function(x){ rnorm(250, 40, 5) })
## define the data set
rownames(ex_data) = paste0("ind", 1:nrow(ex_data))
colnames(ex_data) = paste0("var", 1:ncol(ex_data))
## add in some missingness
ex_data[sample(1:length(ex_data), 500)] = NA
## Estimate missingness and generate plots
imp_data = median_impute(ex_data)
```

met2batch *batch effect on numeric matrix*

Description

This function estimates the effects of a categorical (batch) variable on a matrix of features in univariate linear models.

Usage

```
met2batch(wdata, batch)
```

Arguments

wdata	the numeric data matrix with samples in row, features in columns
batch	a single vector containing a vector based batch variable

Value

a list object of length two with (1) a data frame of summary statistics on (a) the number of tested features ,(b) the mean batch effect across all features ,(c) the mean batch effect across all associated (BH FDR<0.05) features ,(d) the number of associated (BH FDR<0.05) features.

Examples

```
d1 = sapply(1:10, function(x){ rnorm(50, 30, 5) })
d2 = sapply(1:10, function(x){ rnorm(50, 40, 5) })
ex_data = rbind(d1, d2)
d3 = sapply(1:10, function(x){ rnorm(100, 40, 5) })
ex_data = cbind(ex_data, d3)
lot = c( rep("A",50), rep("B",50) )
ex = met2batch(wdata = ex_data, batch = lot)
```

---

missingness.sum *missingness summary plots*

Description

This function sumarizes sample and feature missingness in tables and in a summary plot.

Usage

```
missingness.sum(mydata)
```

Arguments

mydata	metabolite data matrix, with samples in rows and metabolite features in columns.

Value

a list object of length four with (1) a vector of sample missingess,(2) a vector of feature missingness ,(3) a table summarizing missingness ,(4) a list of ggplot2 plots for sample and feature histogram distribution and summary tables

Examples

```
ex_data = sapply(1:100, function(x){ rnorm(250, 40, 5) })
## define the data set
rownames(ex_data) = paste0("ind", 1:nrow(ex_data))
colnames(ex_data) = paste0("var", 1:ncol(ex_data))
## add in some missingness
ex_data[sample(1:length(ex_data), 500)] = NA
## Estimate missingness and generate plots
ms = missingness.sum(ex_data)
## plots
ggpubr::ggarrange(plotlist = ms$plotsout, ncol = 2, nrow = 2)
```

`multivariate.anova` *multivariate analysis*

Description

This function performs a multivariate analysis over a dependentlresponse and numerous independentlexplanatory variable

Usage

```
multivariate.anova(dep, indep_df)
```

Arguments

`dep`	a vector of the dependent variable
`indep_df`	a data frame of the independent variable

Value

ggplot2 table figure of

Examples

```
cmat = matrix(1, 3, 3 )
cmat[1,] = c(1, 0.5, 0.3)
cmat[2,] = c(0.5, 1, 0.25)
cmat[3,] = c(0.3, 0.25, 1)
## simulate some correlated data (multivariable random normal)
set.seed(1110)
ex_data = MASS::mvrnorm(n = 250, mu = c(5, 45, 25), Sigma = cmat )
colnames(ex_data) = c("outcome","age","bmi")
multivariate.anova(dep = ex_data[,1], indep_df = ex_data[, 2:3])
```

ng_anno *Nightingale Health metabolomics annotation data set*

Description

A dataset containing annotation data on Nightingale Health NMR metabolites (lipids), compiled from public resources with additional annotation made here. Some metabolites are repeated in the ng_anno data frame as we have observed that Nightingale Health has changed the spellings a few times. Our code and this data frame attempts to capture all possible spellings that we have observed. We note that as Nightingale Health continues to add new metabolites, updates names, modifies names, or changes spellings the automation of the annotation may fail for some features|metabolites.

Usage

```
ng_anno
```

Format

A data frame with 291 rows and 7 variables:

metabolite metabolite id

raw.label metabolite name and units

class metabolite annotation class

subclass metabolite annotation subclass

label metabolite name and units

label.no.units metabolite name without units

derived_features a binary yes|no indicating if the metabolite variable is a variable derived of two or more other features in the data set ...

Source

http://nightingalehealth.com/

outlier.matrix *identify outlier sample indexes in a matrix*

Description

Given a matrix of data this function returns a matrix of 0|1, of the same structure with 1 values indicating outliers. It is an expansion of the function id.outliers(), applied to columns of a matrix.

Usage

```
outlier.matrix(data, nsd = 5, meansd = FALSE)
```

Arguments

data	a matrix of numerical values, samples in row, features in columns
nsd	the unit distance in SD or IQR from the mean or median estimate, respectively outliers are identified at. Default value is 5.
meansd	set to TRUE if you would like to estimate outliers using a mean and SD method; set to FALSE if you would like to estimate medians and inter quartile ranges. The default is FALSE.

Value

a matrix of 0 (not a sample outlier) and 1 (outlier)

Examples

```
ex_data = sapply(1:25, function(x){ rnorm(250, 40, 5) })
## define the data set
rownames(ex_data) = paste0("ind", 1:nrow(ex_data))
colnames(ex_data) = paste0("var", 1:ncol(ex_data))
## add in some technical error to two samples
m = apply(ex_data, 2, function(x){ mean(x, na.rm = TRUE) })
ex_data[c(1,50), ] = ex_data[1, ] + (m*4)
Omat = outlier.matrix(ex_data)
## how many outliers identified
sum(Omat)
```

---

outlier.summary *feature summary plots*

Description

This function plots the distribution of and identifiy outliers for every metabolite (column) in a data frame.

Usage

```
outlier.summary(dtst, pdf_filename = "./feature_distributions.pdf", nsd = 5)
```

Arguments

dtst	numeric data frame
pdf_filename	name of the pdf out file
nsd	number of SD to consider as outliers, 5 is default

Value

print summary figures for each column of data in the data frame to a pdf file.

Examples

```
## define a covariance matrix
cmat = matrix(1, 4, 4 )
cmat[1,] = c(1, 0.8, 0.6, 0.2)
cmat[2,] = c(0.8, 1, 0.7, 0.5)
cmat[3,] = c(0.6, 0.7, 1, 0.6)
cmat[4,] = c(0.2, 0.5, 0.6,1)
## simulate some correlated data (multivariable random normal)
set.seed(1110)
d1 = MASS::mvrnorm(n = 250, mu = c(5, 45, 25, 15), Sigma = cmat )
set.seed(1010)
d2 = MASS::mvrnorm(n = 250, mu = c(5, 45, 25, 15), Sigma = cmat )
## simulate some random data
d3 = sapply(1:20, function(x){ rnorm(250, 40, 5) })
## define the data set
ex_data = cbind(d1,d2,d3)
rownames(ex_data) = paste0("ind", 1:nrow(ex_data))
colnames(ex_data) = paste0("var", 1:ncol(ex_data))
## run the function
outlier.summary(ex_data)
```

---

outliers *identify outliers*

Description

This function identifies outliers from a vector of data at SD units from the mean.

Usage

```
outliers(x, nsd = 3)
```

Arguments

x	a numerical vector of data
nsd	the number of SD units from the mean to be used as an outlier cutoff.

Value

a list object of length three. (1) a vector of sample indexes indicating the outliers, (2) the lower outlier cuttoff value, (3) the upper outlier cuttoff value.

Examples

```
ex_data = rnbinom(500, mu = 40, size = 5)
outliers(ex_data)
```

`pc.and.outliers` *principal component analysis*

Description

This function performs two principal component analysis. In the first, missing data is imputed to the median. In the second a probablistic PCA is run to account for the missingness. Subsequent to the derivation of the PC, the median imputed PC data is used to identify the number of informative or "significant" PC by (1) an acceleration analysis, and (2) a parrallel analysis. Finally the number of sample outliers are determined at 3, 4, and 5 standard deviations from the mean on the top PCs as determined by the acceleration factor analysis.

Usage

```
pc.and.outliers(metabolitedata, indfeature_names, outliers = TRUE)
```

Arguments

`metabolitedata`	the metabolite data matrix. samples in row, metabolites in columns
`indfeature_names`	a vector of independent feature names | column names.
`outliers`	defaulted to TRUE, a TRUE|FALSE binary flagging if you would like outliers identified.

Value

a list object of length five, with (1) a data frame of PC loadings, (2) a vector of variance explained estimates for each PC, (3) an estimate of the number of informative or top PCs determined by the acceleration factor analysis, (4) an estimate of the number of informative or top PCs determined by parrallel analysis, (5) a data frame of the probablisitic PC loadings

Examples

```
## define a covariance matrix
cmat = matrix(1, 4, 4 )
cmat[1,] = c(1, 0.8, 0.6, 0.2)
cmat[2,] = c(0.8, 1, 0.7, 0.5)
cmat[3,] = c(0.6, 0.7, 1, 0.6)
cmat[4,] = c(0.2, 0.5, 0.6,1)
## simulate some correlated data (multivariable random normal)
set.seed(1110)
d1 = MASS::mvrnorm(n = 250, mu = c(5, 45, 25, 15), Sigma = cmat )
set.seed(1010)
d2 = MASS::mvrnorm(n = 250, mu = c(5, 45, 25, 15), Sigma = cmat )
## simulate some random data
d3 = sapply(1:20, function(x){ rnorm(250, 40, 5) })
## define the data set
ex_data = cbind(d1,d2,d3)
```

```
rownames(ex_data) = paste0("ind", 1:nrow(ex_data))
colnames(ex_data) = paste0("var", 1:ncol(ex_data))
## add in some missingness
ex_data[sample(1:length(ex_data), 450)] = NA
## add in some technical error to two samples
m = apply(ex_data, 2, function(x){ mean(x, na.rm = TRUE) })
ex_data[c(1,10), ] = ex_data[1, ] + (m*0.00001)
## run the PCA
ex_out = pc.and.outliers(ex_data, indfeature_names = sample(colnames(ex_data), 15) )
```

`pca.factor.analysis` *PCA factor analysis and annotation enrichment*

Description

This function performs (1) a factor analysis on numeric data and PC loadings derived from said data, then subsequently (3) performs a hypergeometrix enrichment test to ask if a certain class of variables are enriched on a particular PC.

Usage

```
pca.factor.analysis(
  metabolitedata,
  pcloadings,
  sigthreshold = 0.3,
  feature_anno = feature_data$SUPER_PATHWAY
)
```

Arguments

`metabolitedata`	a matrix or data frame of metabolite data
`pcloadings`	a matrix or data frame of pc loadings you wish to test
`sigthreshold`	Spearman's rho correlation threshold to declare an association between the numeric variable and a PC loading
`feature_anno`	a vector of variable annotations to perform the hypergeomtric enrichment on

Value

a list object of length 2: (1) a list object of enrichment_tables, and (2) the Spearman's correlation matrix between features and the PC loadings

Examples

```
## define a covariance matrix
cmat = matrix(1, 4, 4 )
cmat[1,] = c(1, 0.8, 0.6, 0.2)
cmat[2,] = c(0.8, 1, 0.7, 0.5)
cmat[3,] = c(0.6, 0.7, 1, 0.6)
cmat[4,] = c(0.2, 0.5, 0.6,1)
## simulate some correlated data (multivariable random normal)
set.seed(1110)
d1 = MASS::mvrnorm(n = 250, mu = c(5, 45, 25, 15), Sigma = cmat )
set.seed(1010)
d2 = MASS::mvrnorm(n = 250, mu = c(5, 45, 25, 15), Sigma = cmat )
## simulate some random data
d3 = sapply(1:20, function(x){ rnorm(250, 40, 5) })
ex_data = cbind(d1,d2,d3)
rownames(ex_data) = paste0("ind", 1:nrow(ex_data))
colnames(ex_data) = paste0("var", 1:ncol(ex_data))
## annotation
met_anno = c( rep("A", 10), rep("B", 10), rep("C", 8) )
## PCA
pca = prcomp(ex_data, center = TRUE, scale = TRUE)
## run pca.factor.analysis()
ex_out = pca.factor.analysis(metabolitedata = ex_data,
                             pcloadings = pca$x[,1:5],
                             sigthreshold = 0.3,
                             feature_anno = met_anno )
```

`pcapairs_bymoose` *pca pairs plot*

Description

This function generates an upper triangle PCA plot for all pairs of PC loadings provided.

Usage

```
pcapairs_bymoose(myloadings, varexp, pcol = "dodgerblue")
```

Arguments

`myloadings`	a matrix or data frame of PC loadings (only those you would like to plot).
`varexp`	a vector of the the variance explained by each PC
`pcol`	plot colors for the dots background

Value

a base R plot

Examples

```
ex_data = sapply(1:5, function(x){rnorm( 50, 0, 2) })
## add in some extreme values to a sample
ex_data[1,] = ex_data[1,] + sample( c(8, -8), ncol(ex_data), replace = TRUE )
ex_data[2,] = ex_data[2,] + sample( c(3, -3), ncol(ex_data), replace = TRUE )
## plot
pcapairs_bymoose(myloadings = ex_data,
    varexp = rep(1/ncol(ex_data), ncol(ex_data)),
    pcol = "tomato" )
```

---

perform.metabolite.qc *perform metabolomics quality control*

Description

This function is a wrapper function that performs the key quality controls steps on a metabolomics data set

Usage

```
perform.metabolite.qc(
  wdata,
  fmis = 0.2,
  smis = 0.2,
  tpa_out_SD = 5,
  outlier_udist = 5,
  outlier_treatment = "leave_be",
  winsorize_quantile = 1,
  tree_cut_height = 0.5,
  PC_out_SD = 5,
  feature_colnames_2_exclude = NA,
  derived_colnames_2_exclude = NA
)
```

Arguments

wdata	the metabolite data matrix. samples in row, metabolites in columns
fmis	defaulted at 0.2, this defines the feature missingness cutoff
smis	defaulted at 0.2, this defines the sample missingness cutoff
tpa_out_SD	defaulted at 5, this defines the number of standard deviation from the mean in which samples will be excluded for total peak area. Pass NA to this paramater to exclude this step.
outlier_udist	defaulted at 5, this defines the number of interquartile range units from the median to define a value as an outlier

`outlier_treatment`
: defaulted to "leave_be". Choices are "leave_be", "winsorize", or "turn_NA", which defines how to treat outlier values prior to estimating principal components. "leave_be" will do nothing to ourlier values. "turn_NA" will turn outliers into NA and thus be median imputed for the purpose of the PCA. "winsorize" will turn NA values into the "winsorize_quantile" of all remaining (non-outlying) values at a feature.

`winsorize_quantile`
: the quantile (0-1) to winzorise outlier values to, if "outlier_treatment" parameter set to "winsorize". Defaulted to 1, or the maximum value of all remaining (non-outlying) values at a feature.

`tree_cut_height`
: The height at which to cut the feature|metabolite dendrogram to identify "independent" features. tree_cut_height is 1-absolute(Spearman's Rho) for intra-cluster correlations.

`PC_out_SD`
: defaulted at '5', this defines the number of standard deviation from the mean in which samples will be excluded for principle components. NA is NOT an excepted paramater.

`feature_colnames_2_exclude`
: names of columns to be excluded from all analysis, such as for Xenobiotics. Pass NA to this parameter to exclude this step.

`derived_colnames_2_exclude`
: names of columns to exclude from all sample QC steps, including independent feature identification, which is used to generate the sample PCA.

Value

a list object of: (1) "wdata" qc'd data matrix, (2) "featuresumstats" a list with a (1:"table") data frame of feature summary statistics and a (2:"tree") hclust object (3) "pca" a list with a (1:"pcs") data frame of the top 10 PCs and a binary for outliers on the top 2 PCs at 3,4, and 5 SD from the mean, a (2:"varexp") vector of the variance explained for each PC, an estimate of the number of 'significant' PCs as determined by (3:"accelerationfactor") an acceleration factor and (4:"nsig_parrallel") a parrallel analysis, and finally (5:"prob_pca") the top 10 PCs derived by a probabilistic PC analysis. (4) "exclusion_data" a matrix of exclusion summary statistics

Examples

```
## define a covariance matrix
cmat = matrix(1, 4, 4 )
cmat[1,] = c(1, 0.8, 0.6, 0.2)
cmat[2,] = c(0.8, 1, 0.7, 0.5)
cmat[3,] = c(0.6, 0.7, 1, 0.6)
cmat[4,] = c(0.2, 0.5, 0.6,1)
## simulate some correlated data (multivariable random normal)
set.seed(1110)
d1 = MASS::mvrnorm(n = 250, mu = c(5, 45, 25, 15), Sigma = cmat )
set.seed(1010)
d2 = MASS::mvrnorm(n = 250, mu = c(5, 45, 25, 15), Sigma = cmat )
## simulate some random data
```

```
d3 = sapply(1:20, function(x){ rnorm(250, 40, 5) })
## define the data set
ex_data = cbind(d1,d2,d3)
rownames(ex_data) = paste0("ind", 1:nrow(ex_data))
colnames(ex_data) = paste0("var", 1:ncol(ex_data))
## add in some missingness
ex_data[sample(1:length(ex_data), 450)] = NA
## add in some technical error to two samples
m = apply(ex_data, 2, function(x){ mean(x, na.rm = TRUE) })
ex_data[c(1,10), ] = ex_data[1, ] + (m*0.00001)

## run the quality control
example_qc = perform.metabolite.qc(ex_data)

## the filtered data is found in
dim(example_qc$wdata)
example_qc$wdata[1:5, 1:5]
## a data frame of summary statistics
head( example_qc$featuresumstats$table )
## a hclust dendrogram can be plotted
plot( example_qc$featuresumstats$tree , hang = -1)
abline(h = 0.5, col = "red", lwd = 1.5)
## (median imputed) PCs for all samples can be plotted
pcol = c("blue","red")[ as.factor( example_qc$pca$pcs[, "PC1_5_SD_outlier"] )]
plot(example_qc$pca$pcs[,"PC1"], example_qc$pca$pcs[,"PC2"],
     pch = 21, cex = 1.5, bg = pcol,
    xlab = paste0( "PC1: Var Exp = " , round(example_qc$pca$varexp[1], d = 4)*100, "%" ) ,
    ylab = paste0( "PC2: Var Exp = " , round(example_qc$pca$varexp[2], d = 4)*100, "%" ) )
## A Scree plot can be generated by
plot( x = 1:length(example_qc$pca$varexp),
      y = example_qc$pca$varexp,
      type = "b", pch = 21, cex = 2, bg = "blue",
      xlab = "PC", ylab = "Variance Explained", main = "Scree Plot")
abline(v = example_qc$pca$accelerationfactor, col = "red", lwd = 2)
abline(v = example_qc$pca$nsig_parrallel, col = "green", lwd = 2)
## Probablistic PCs for all samples can be plotted
plot(example_qc$pca$prob_pca[,1], example_qc$pca$prob_pca[,2],
     pch = 21, cex = 1.5, bg = pcol,
     xlab = "PC1",
     ylab = "PC2" )
## A summary of the exclusion statistics
example_qc$exclusion_data
```

`read.in.metabolon` *read in Metabolon (v1) metabolomics data*

Description

This function reads in a Metabolon (v1 format) raw data excel file, writes the (1) metabolite, (2) sample annotation, and (3) feature annotation data to flat text files. It also returns a list object of the

same data.

Usage

```
read.in.metabolon(file2process, data_dir, projectname)
```

Arguments

file2process	the name of the xls file to process
data_dir	the full path to the directory holding your Metabolon excel file
projectname	a name for your project

Value

a list object of (1) metabolite, (2) sample annotation, and (3) feature annotation data

Examples

```
# read.in.metabolon(file2process = "Metabolon_data_release.xls",
#  data_dir = "/File/sits/here/",
#  projectname = "My Amazing Project")
```

---

read.in.nightingale *read in Nightingale Health metabolomics data*

Description

This function reads in a Nightingale raw data excel file, writes the (1) metabolite, (2) sample annotation, and (3) feature annotation data to flat text files. It also returns a list object of the same data.

Usage

```
read.in.nightingale(file2process, data_dir, projectname)
```

Arguments

file2process	the name of the xls file to process
data_dir	the full path to the directory holding your Nightingale excel file
projectname	a name for your project

Value

a list object of (1) metabolite, (2) sample annotation, and (3) feature annotation data

Examples

```
# read.in.nightingale(file2process = "NH_data_release.xls",
#  data_dir = "/File/sits/here/",
#  projectname = "My Amazing Project")
```

`rntransform` *rank normal tranformation*

Description

This function rank normal transforms a vector of data. The procedure is built off of that provided in the GenABEL pacakge.

Usage

```
rntransform(y, split_ties = TRUE)
```

Arguments

`y`	a numeric vector which will be rank normal transformed
`split_ties`	a binary string of TRUE (default) or FALSE indicating if tied values, of the same rank, should be randomly split giving them unique ranks.

Value

returns a numeric vector, with the same length as y, of rank normal transformed values

Examples

```
## simulate a negative binomial distribution of values
nb_data = rnbinom(500, mu = 40, size = 100)
## rank normal transform those values
rnt_data = rntransform( nb_data , split_ties = TRUE )
```

run.cont.power.make.plot

continuous trait power analysis plot

Description

This function (1) identifies an informative distribution of effect and power estimates given your datas total sample size and (2) returns a summary plot.

Usage

```
run.cont.power.make.plot(mydata)
```

Arguments

mydata	Your metabolite data matrix, with samples in rows

Value

a ggplot2 object

Examples

```
ex_data = matrix(NA, 1000, 2)
run.cont.power.make.plot( ex_data )
```

run.pa.imbalanced.power.make.plot

binary trait imbalanced design power analysis plot

Description

This function (1) estimates an informative distribution of effect and power estimates given your datas total sample size, over a distribution of imbalanced sample sizes and (2) returns a summary plot.

Usage

```
run.pa.imbalanced.power.make.plot(mydata)
```

Arguments

mydata	a numeric data matrix with samples in rows and features in columns

Value

a ggplot2 object

Examples

```
ex_data = matrix(NA, 1000, 2)
run.pa.imbalanced.power.make.plot( ex_data )
```

`sam.missingness.exclusion` *sample exlusions on missingness and total peak area*

Description

This function provides missingnes and tpa estimates along with exlcusion at 3, 4, and 5 SD from the mean.

Usage

```
sam.missingness.exclusion(mydata, sdata, fdata)
```

Arguments

`mydata`	metabolite data
`sdata`	sample data
`fdata`	feature data

Value

a data frame of missingness and TPA exclusions

Examples

```
# sam.missingness.exclusion()
```

`sample.missingness` *estimate sample missingness*

Description

This function estimates sample missingness in a matrix of data and provides an option to exclude certain columns or features from the analysis, such as xenobiotics (with high missingness rates) in metabolomics data sets.

Usage

```
sample.missingness(wdata, excludethesefeatures = NA)
```

Arguments

`wdata`	a numeric matrix with samples in row and features in columns
`excludethesefeatures`	a vector of feature names (i.e. column names) to exclude from missingness estimates

Value

A data frame of missingness estimates for each sample. If a vector of feature names was also passed to the function a second column of missingness estimates will also be returned providing missingness estimates for each sample to the exclusion of those features provided.

Examples

```
## simulate some data
set.seed(1110)
ex_data = sapply(1:5, function(x){ rnorm(10, 40, 5) })
rownames(ex_data) = paste0("ind", 1:nrow(ex_data))
colnames(ex_data) = paste0("var", 1:ncol(ex_data))
## add some missingness to the data
ex_data[ sample(1:50, 10) ] = NA
## estimate missingness
mis_est = sample.missingness(ex_data)
mis_est_v2 = sample.missingness(ex_data, excludethesefeatures = "var5")
```

sample.outliers *outlier features count for samples*

Description

This function takes a matrix of numeric data and counts the number of outlying features each sample has.

Usage

```
sample.outliers(wdata, nsd = 5)
```

Arguments

wdata	a metabolite data matrix with samples in row, metabolites in columns
nsd	the number of standard deviation from the mean outliers are identified at. The default value is 5.

Value

a data frame of outiler counts for each sample

Examples

```
d = sapply(1:5, function(x){ rnorm(50, 50, 15) })
sample.outliers(d, nsd = 2)
```

sample.sum.stats *summary statistics for samples*

Description

This function estimates summary statistics for samples in a matrix of numeric features. This includes missingness, total peak area, and a count of the number of outlying features for a sample.

Usage

```
sample.sum.stats(wdata, feature_names_2_exclude = NA, outlier_udist = 5)
```

Arguments

wdata	the metabolite data matrix. samples in row, metabolites in columns
feature_names_2_exclude	a vector of feature|column names to exclude from missingness estimates
outlier_udist	the interquartile range unit distance from the median to call a sample an outlier at a feature.

Value

a data frame of summary statistics

Examples

```
## simulate some data
set.seed(1110)
ex_data = sapply(1:5, function(x){ rnorm(10, 40, 5) })
rownames(ex_data) = paste0("ind", 1:nrow(ex_data))
colnames(ex_data) = paste0("var", 1:ncol(ex_data))
## add some missingness to the data
ex_data[ sample(1:50, 10) ] = NA
## run estimate sample summary statistics
sample.sum.stats(ex_data)
```

`total.peak.area` *estimates total peak abundance*

Description

This function estimates total peak abundance|area for numeric data in a matrix, for (1) all features and (2) all features with complete data.

Usage

```
total.peak.area(wdata, feature_names_2_exclude = NA, ztransform = TRUE)
```

Arguments

`wdata`	the metabolite data matrix. samples in row, metabolites in columns
`feature_names_2_exclude`	A vector of feature|metabolite names to exclude from the tree building, independent feature identification process.
`ztransform`	should the feature data be z-transformed and absolute value minimum, mean shifted prior to summing the feature values. TRUE or FALSE.

Value

a data frame of estimates for (1) total peak abundance and (2) total peak abundance at complete features for each samples

Examples

```
set.seed(1110)
ex_data = sapply(1:5, function(x){ rnorm(10, 40, 5) })
rownames(ex_data) = paste0("ind", 1:nrow(ex_data))
ex_data[ sample(1:50, 4) ] = NA
tpa_est = total.peak.area(ex_data)
```

---

tree_and_independent_features

identify independent features in a numeric matrix

Description

This function identifies independent features using Spearman's rho correlation distances, and a dendrogram tree cut step.

Usage

```
tree_and_independent_features(
  wdata,
  minimum_samplesize = 50,
  tree_cut_height = 0.5,
  feature_names_2_exclude = NA
)
```

Arguments

wdata	the metabolite data matrix. samples in row, metabolites in columns
minimum_samplesize	the metabolite data matrix. samples in row, metabolites in columns
tree_cut_height	the tree cut height. A value of 0.2 (1-Spearman's rho) is equivalent to saying that features with a rho >= 0.8 are NOT independent.
feature_names_2_exclude	A vector of feature|metabolite names to exclude from this analysis. This might be features heavily present|absent like Xenobiotics or variables derived from two or more variable already in the dataset.

Value

a list object of (1) an hclust object, (2) independent features, (3) a data frame of feature ids, k-cluster identifiers, and a binary identifier of independent features

Examples

```
## define a covariance matrix
cmat = matrix(1, 4, 4 )
cmat[1,] = c(1, 0.7, 0.4, 0.2)
cmat[2,] = c(0.7, 1, 0.2, 0.05)
cmat[3,] = c(0.4, 0.2, 1, 0.375)
cmat[4,] = c(0.2, 0.05, 0.375,1)

## simulate the data (multivariable random normal)
set.seed(1110)
ex_data = MASS::mvrnorm(n = 500, mu = c(5, 45, 25, 15), Sigma = cmat )
rownames(ex_data) = paste0("ind", 1:nrow(ex_data))
colnames(ex_data) = paste0("var", 1:ncol(ex_data))

## run function to identify independent variables at a tree cut height
## of 0.5 which is equivalent to clustering variables with a Spearman's
## rho > 0.5 or (1 - tree_cut_height)
ind = tree_and_independent_features(ex_data, tree_cut_height = 0.5)
```

`variable.by.factor` *ggplot2 violin plot*

Description

This function performs univariate linear analysis of a dependent and an independent variable and generates a viloin or box plot to illustrate the associated structure.

Usage

```
variable.by.factor(
  dep,
  indep,
  dep_name = "dependent",
  indep_name = "independent",
  orderfactor = TRUE,
  violin = TRUE
)
```

Arguments

`dep`	a vector of the dependent variable
`indep`	a vector of the independent variable
`dep_name`	name of the dependent variable
`indep_name`	name of the independent variable
`orderfactor`	order factors alphebetically
`violin`	box plot or violin plot. violin = TRUE is default

Value

a ggplot2 object

Examples

```
x = c( rnorm(20, 10, 2), rnorm(20, 20, 2) )
y = as.factor( c( rep("A", 20), rep("B", 20)  ) )
variable.by.factor(dep = x , indep = y, dep_name = "expression", indep_name = "species" )
```

Index