Package ‘LipidomicsR’

April 22, 2024

Type Package

Title Elegant Tools for Processing and Visualization of Lipidomics Data

Version 0.3.6

Maintainer Hengyu Zhu <hengyu.21@intl.zju.edu.cn>

Description An elegant tool for processing and visualizing lipidomics data generated by mass spectrometry. 'LipidomicsR' simplifies channel and replicate handling while providing thorough lipid species annotation. Its visualization capabilities encompass principal components analysis plots, heatmaps, volcano plots, and radar plots, enabling concise data summarization and quality assessment. Additionally, it can generate bar plots and line plots to visualize the abundance of each lipid species.

License MIT + file LICENSE

Encoding UTF-8

Imports broom, car, ggiraph, rcompanion, tidyr, cowplot, dplyr, fmsb, ggplot2, ggplotify, ggrepel, pheatmap, RColorBrewer, reshape2, scales, stringr, tidyselect, ggforce, ggsci

Depends tidyverse

RoxygenNote 7.3.1

NeedsCompilation no

Author Mingshi Li [aut, com],
Hengyu Zhu [aut, cre]

URL https://github.com/mingshi1/LipidomicsR

BugReports https://github.com/mingshi1/LipidomicsR/issues

Repository CRAN

Date/Publication 2024-04-22 11:22:43 UTC

R topics documented:

`absolute.calculator` *absolute.calculator()*

Description

Internal function of lipidmicR::normalization_calculator()

Usage

```
absolute.calculator(data, absolute.dataset)
```

Arguments

`data`	Row data of lipidomics
`absolute.dataset`	Data frame. Normalization index.

Value

Return a data frame of absolute normalized lipidomic data.

`abundance.lineplot` *Plot Abundance Data*

Description

This function generates line plots for abundance data.

Usage

```
abundance.lineplot(
  data.summary,
  by = data.frame(group = c(), variable.1 = c(), variable.2 = c()),
  .width = 0.5,
  .position_dodge = 0,
  errorbar.width = 0.5,
  .xlab = "group",
  .ylab = "abundance",
  axis.title.size = 10,
  axis.title.x.vjust = 0,
  axis.title.y.vjust = 0,
  axis.text.size = 10,
  axis.line.size = 0.5,
  axis.tick.length = 0.2,
  legend.title = "",
  legend.color = "Set2",
  color.style = "Nature",
  .legend.direction = "vertical",
  .legend.position = "right",
  main.size = 10
)
```

Arguments

`data.summary`	A data frame containing summarized abundance data.
`by`	A data frame specifying additional variables for grouping and plotting (default is an empty data frame).
`.width`	The width of lines in the plot (default is 0.5).
`.position_dodge`	The position adjustment parameter for dodging lines (default is 0.5).
`errorbar.width`	The width of error bars (default is 0.5).
`.xlab`	The label for the x-axis (default is 'group').
`.ylab`	The label for the y-axis (default is 'abundance').
`axis.title.size`	The size of axis title text (default is 10).

`axis.title.x.vjust`	The vertical adjustment parameter for x-axis title (default is 0).
`axis.title.y.vjust`	The vertical adjustment parameter for y-axis title (default is 0).
`axis.text.size`	The size of axis text (default is 10).
`axis.line.size`	The size of axis lines (default is 0.5).
`axis.tick.length`	The length of axis ticks (default is 0.2).
`legend.title`	The title for the legend (default is an empty string).
`legend.color`	The color palette for the legend (default is a set of predefined colors).
`color.style`	The style of colors, which can be "Nature", "Science", "Lancet", "JCO", "D3", "IGV", "Star Trek", "Tron Legacy", "Rick and Morty", and "The Simpsons". Default value is "Nature".
`.legend.direction`	The direction of the legend ('horizontal' or 'vertical', default is 'vertical').
`.legend.position`	The position of the legend ('top', 'bottom', 'left', 'right', or NULL, default is 'right').
`main.size`	The size of plot titles (default is 10).

Value

A list of ggplot objects, each representing a line plot for abundance data of a species. In addition, the list includes a line plot for the sum of abundance of each class of lipid type, unsaturation, and carbon number.

`abundance.plot` *Plot Abundance Data*

Description

This function generates bar plots for abundance data.

Usage

```
abundance.plot(
  data.summary,
  by = data.frame(group = c(), variable.1 = c(), variable.2 = c()),
  .width = 0.5,
  .position_dodge = 0.5,
  errorbar.type = "up",
  errorbar.width = 0.25,
  .xlab = "group",
  .ylab = "abundance",
  axis.title.size = 10,
  axis.title.x.vjust = 0,
  axis.title.y.vjust = 0,
  axis.text.size = 10,
  axis.line.size = 0.5,
  axis.tick.length = 0.2,
  legend.title = "",
 legend.color = c("#A1A9D0", "#F0988C", "#B883D4", "#9E9E9E", "#CFEAF1", "#C4A5DE",
    "#F6CAE5", "#96CCCB"),
  .legend.direction = "vertical",
  .legend.position = "right",
  main.size = 10
)
```

Arguments

`data.summary`	A data frame containing summarized abundance data.
`by`	A data frame specifying additional variables for grouping and plotting (default is an empty data frame).
`.width`	The width of bars in the plot (default is 0.5).
`.position_dodge`	The position adjustment parameter for dodging bars (default is 0.5).
`errorbar.type`	The type of error bars to be plotted ('up', 'down', or 'both', default is 'up').
`errorbar.width`	The width of error bars (default is 0.5).
`.xlab`	The label for the x-axis (default is 'group').
`.ylab`	The label for the y-axis (default is 'abundance').
`axis.title.size`	The size of axis title text (default is 10).
`axis.title.x.vjust`	The vertical adjustment parameter for x-axis title (default is 0).
`axis.title.y.vjust`	The vertical adjustment parameter for y-axis title (default is 0).
`axis.text.size`	The size of axis text (default is 10).
`axis.line.size`	The size of axis lines (default is 0.5).
`axis.tick.length`	The length of axis ticks (default is 0.2).
`legend.title`	The title for the legend (default is an empty string).
`legend.color`	The color palette for the legend (default is a set of predefined colors).
`.legend.direction`	The direction of the legend ('horizontal' or 'vertical', default is 'vertical').
`.legend.position`	The position of the legend ('top', 'bottom', 'left', 'right', or NULL, default is 'right').
`main.size`	The size of plot titles (default is 10).

Value

A list of ggplot objects, each representing a bar plot for abundance data of a species.

`abundance.signif` *Analyze Abundance Significance*

Description

This function performs statistical analysis to determine the significance of abundance data.

Usage

```
abundance.signif(
  data,
  group,
  istotal = FALSE,
  by = data.frame(group = c(), variable.1 = c(), variable.2 = c())
)
```

Arguments

`data`	The data frame containing abundance data.
`group`	A vector specifying the group membership for each sample.
`istotal`	Logical. If is true, statistics based on total summary table of lipid type, carbon number and unsaturation rate will be generated.
`by`	A data frame specifying additional variables for the analysis.

Value

A list containing statistical analysis results for each species.

`abundance.summary` *Summarize Abundance Data*

Description

This function summarizes abundance data based on specified groups.

Usage

```
abundance.summary(
  data,
  group,
  istotal = FALSE,
  .summary = function(x) mean(x),
  .error = function(x) sd(x)
)
```

Arguments

`data`	The data frame containing abundance data.
`group`	A vector specifying the group membership for each sample.
`istotal`	Logical. If is true, the total summary table of lipid type, carbon number and unsaturation rate will be generated.
`.summary`	A function to summarize abundance data within each group (default is mean).
`.error`	A function to compute error measures within each group (default is standard deviation).

Value

A data frame summarizing abundance data by species, group, lipid type, carbon number, and unsaturation.

`channel.delete` *Delete repeated channel data generated by LC-MS*

Description

channel.delete select one of the channel with largest mean value out of multiple replicated LC-MS cahnnels, and it returns an edited data frame with no repeated channel and simplified row name.

Usage

```
channel.delete(
  data,
  delete.pattern = c("_\\d", "(\\+)AcO", "_n", "\\(\\d+\\)"),
  group = NA,
  remove = NA
)
```

Arguments

`data`	Primary lipidomic data in .csv format.
`delete.pattern`	Pattern of characters that needs to be removed.
`group`	Vector. The group information, recommended to be generated with groupXpert()
`remove`	Character. The group that needed to be removed.

Value

Edited data with no repeat channels and simplified row name.

Examples

```
x1 <- c(1,1,1,1)
x2 <- c(1.1,1.2,1.3,1.4)
x3 <- c(1,1,1,1)
x4 <- c(2,2,2,2)
rowNames <- c('PC(14:0/14:1)+AcO_1','PC(14:0/14:1)+AcO_2','PC(16:0/16:1)_1','PC(16:0/16:1)_2')
ExampleData <- data.frame(x1,x2,x3,x4, row.names =rowNames)
replicate.delete(ExampleData,delete.pattern= c('_\\d','(\\+)AcO'))
```

---

delRep *delRep()*

Description

A function to delete specific number of replicates, the replicates causing largest standard deviation will be deleted.

Usage

```
delRep(data, group, m, method = "PCA", show.del = FALSE)
```

Arguments

`data`	A dataframe storing concentration of lipids between different samples. The column name should be the sample name and the row name should be the lipid type. The class of column name and row name should be "character". The class of values should be "numeric". The row names are recommended to be in a form like "PL(14:0/20:1)" or "LPL(16:1)".
`group`	Vector. The group information, recommended to be generated with groupXpert()
`m`	The number of replicates you want to delete.
`method`	The method to find the worst replicates, can be "PCA" or "Euclidean". Default value is "PCA".
`show.del`	Whether to show deleted columns. Default value is FALSE.

Value

A new dataframe deleting replicates which cause the largest SD.

Examples

```
WT_1=rnorm(n=5,mean=0.3,sd=0.1)
WT_2=rnorm(n=5,mean=0.3,sd=0.1)
WT_3=rnorm(n=5,mean=0.3,sd=0.1)
WT_4=rnorm(n=5,mean=0.3,sd=0.1)
KO_1=rnorm(n=5,mean=0.3,sd=0.1)
KO_2=rnorm(n=5,mean=0.3,sd=0.1)
```

```
KO_3=rnorm(n=5,mean=0.3,sd=0.1)
KO_4=rnorm(n=5,mean=0.3,sd=0.1)
data=data.frame(WT_1,WT_2,WT_3,WT_4,KO_1,KO_2,KO_3,KO_4)
rownames(data)=c("LPC(16:0)","PC(14:0/16:1)","PC(18:1/18:1)","PE(18:0/20:1)","PS(20:1/20:1)")
m=1
group=colnames(data)
names(group)=rep(c("WT","KO"),each=4)
delRep(data,group,m)
```

FC_P *Calculate Fold Change and p-values for differential expression analysis*

Description

This function calculates the log2 fold change (FC) and p-values for differential expression analysis between two groups of data.

Usage

```
FC_P(compare1, compare2, p.adj = FALSE, method = "fdr")
```

Arguments

`compare1`	A matrix or data frame representing the first group of data.
`compare2`	A matrix or data frame representing the second group of data.
`p.adj`	Logical. Should p-values be adjusted for multiple testing?
`method`	The method to use for p-value adjustment. Options: 'fdr', 'bonferroni', 'holm', etc.

Value

A data frame containing the log2 fold change and -log10 transformed p-values for each row (e.g., genes, features) in the input data.

Examples

```
compare1 <- matrix(rnorm(100), ncol = 10)
compare2 <- matrix(rnorm(100), ncol = 10)
result <- FC_P(compare1, compare2, p.adj = TRUE)
```

`groupXpert` *groupXpert()*

Description

groupXpert()

Usage

```
groupXpert(data, sep = "_", as.name = TRUE, specify = NULL)
```

Arguments

`data`	Data.frame. Row data to be grouped.
`sep`	Character. The separator used to separate variables from the number of repeats
`as.name`	Logical. If is true, the group name will be used as the name of the return value. If is false, the sample name will be used as the name of the return value.
`specify`	List. Used to set groups manually. Each sublist has a group name and a value based on the column range in which it is located. No duplicate values can appear in the sublist.

Value

A vector named as the colname and uses group name as the value.

`heatmap` *heatmap()*

Description

A function to plot a heatmap based on abundance of lipids. A series of custom functions can be realized such as dividing groups.

Usage

```
heatmap(
  data,
  group,
  cluster_row = TRUE,
  cluster_col = TRUE,
  sel.group = "default",
  constract = 8.5,
  type = "lipid",
  sel.type = "default",
  sel.row = c("default"),
  annotation_legend = TRUE,
  cellwidth = 20,
  cellheight = 15,
  gaps_row = c(0),
  gaps_col = c(0),
  subtype = FALSE,
  labels_row = c("default"),
  labels_col = c("default"),
  title = "",
  show_rownames = TRUE,
  show_colnames = TRUE,
  cellcolor = c("blue", "black", "yellow"),
  legend = TRUE,
  border_color = NA,
  border = FALSE,
  cutree_rows = 1,
  cutree_cols = 1,
  rtitle = "group",
  ctitle = " ",
  fontsize_row = 12,
  fontsize_col = 12,
  fontsize = 8
)
```

Arguments

`data`	A dataframe storing absolute concentration or PL to get normalized data. The column name should be the sample name and the row name should be the lipid type. The class of column name and row name should be "character". The class of values should be "numeric". The row names are recommended to be in a form like "PL(14:0/20:1)" or "LPL(16:1)".
`group`	A vector defining which group the replicates belong to.
`cluster_row`	A boolean variable controlling whether to perfrom clustering to row variables (lipid abundance) or not. The default value is TRUE.
`cluster_col`	A boolean variable controlling whether to perfrom clustering to column variables (lipid abundance) or not. The default value is TRUE.
`sel.group`	A vector containing the group you want to show in the heatmap. The input can be like c("WT","KO"). Default value is "default".
`constract`	The constract of heatmap, default is 8.5, value range from 0 to 10.
`type`	Can accept 3 values: "lipid", "CB", or "sat". Default value is "lipid". If type="lipid“, the heatmap will divide rownames based on lipid types. If pattern="CB“, the heatmap will divide rownames based on carbon number. If pattern="sat“, the heatmap will divide rownames based on the number of double bonds of a lipid type.
`sel.type`	A vector controlling which types to show. If you only want to check data of "PA", "PC“, and "PE". You can set type="lipid", sel.type=c("PA","PC","PE")

`sel.row`	A vector controlling which types to show. If you set it as c("LPC(16:0)","PC(14:0/16:1)", "PC(18:1/18:1)","PE(18:0/20:1)"), only their abundance will be shown.
`annotation_legend`	A boolean controlling whether to show the figure legend. The default value is TRUE.
`cellwidth`	The width of a cell in the heatmap. Default value is 20.
`cellheight`	The height of a cell in the heatmap. Default value is 15.
`gaps_row`	To customize positions of row gaps. Default value is c(0).
`gaps_col`	To customize positions of column gaps. Default value is c(0). Notice: gaps_row and gaps_col are only useful when cluster=FALSE.
`subtype`	A logic value to determine for a lipid like "PC(O-14:0/16:1)", "lipid_type" should be "PC" (subtype=FALSE) or "PC(O)" (subtype=TRUE). Default value is FALSE.
`labels_row`	A vector contains the labels of each row of the heatmap. Default value is row names of dataframe input. It can be input like ("PE(20:1/20:1)","PS(16:0/18:1)","","","","LPA(18:0)")
`labels_col`	A vector contains the labels of each column of the heatmap. Default value is column names of dataframe input.
`title`	The title of heatmap. Default value is "".
`show_rownames`	Whether to show row names or not. Default value is T.
`show_colnames`	Whether to show column names or not. Default value is T.
`cellcolor`	The color range of cells in the heatmap. It should be input in a vector with three color values, such as c("blue","black","yellow").
`legend`	Whether to show legends or not. Default value is FALSE.
`border_color`	Useful when border=T. Default value is NA.
`border`	Whether to show borders or not. Default value is TRUE.
`cutree_rows`	Useful when cluster=T. If cutree_rows=T, the rows of heatmap will be divided according to clustering results. Default value is TRUE.
`cutree_cols`	Useful when cluster=T. If cutree_cols=T, the rows of heatmap will be divided according to clustering results. Default value is TRUE.
`rtitle`	Row title of the heatmap. Default value is "group".
`ctitle`	Column title of the heatmap. Default value is " ".
`fontsize_row`	Fontsize of row labels. Default value is 12.
`fontsize_col`	Fontsize of column labels. Default value is 12.
`fontsize`	Fontsize of all labels. Default value is 8.

Value

A heatmap that is color-coded by abundance of lipids.

Examples

```
WT_1=rnorm(n=10,mean=0.4,sd=0.1)
WT_2=rnorm(n=10,mean=0.4,sd=0.1)
WT_3=rnorm(n=10,mean=0.4,sd=0.1)
WT_4=rnorm(n=10,mean=0.4,sd=0.1)
KO_1=rnorm(n=10,mean=0.8,sd=0.1)
KO_2=rnorm(n=10,mean=0.8,sd=0.1)
KO_3=rnorm(n=10,mean=0.8,sd=0.1)
KO_4=rnorm(n=10,mean=0.8,sd=0.1)
WT_treat_1=rnorm(n=10,mean=0.1,sd=0.1)
WT_treat_2=rnorm(n=10,mean=0.1,sd=0.1)
WT_treat_3=rnorm(n=10,mean=0.1,sd=0.1)
WT_treat_4=rnorm(n=10,mean=0.1,sd=0.1)
KO_treat_1=rnorm(n=10,mean=0.6,sd=0.1)
KO_treat_2=rnorm(n=10,mean=0.6,sd=0.1)
KO_treat_3=rnorm(n=10,mean=0.6,sd=0.1)
KO_treat_4=rnorm(n=10,mean=0.6,sd=0.1)
data=data.frame(WT_1,WT_2,WT_3,WT_4,KO_1,KO_2,KO_3,KO_4,
                WT_treat_1,WT_treat_2,WT_treat_3,WT_treat_4,
                KO_treat_1,KO_treat_2,KO_treat_3,KO_treat_4)
rownames(data)=c("LPC(16:0)","PC(14:0/16:1)","PC(18:1/18:1)","PE(18:0/20:1)",
                 "PS(20:1/20:1)","PI(16:0/16:1)","PC(18:0/18:1)","PA(16:0/16:1)",
                 "LPE(18:0)","PE(O-18:1/18:0)")
group=rep(c("WT","KO","WT_treat","KO_treat"),each=4)
heatmap(data,group)
```

`importer` *importer()*

Description

Internal function to import data in lipidomicR, in order to unify data format.

Usage

```
importer(path, header = TRUE, sep = ",")
```

Arguments

`path`	Path of file loaded. The file should be in '.csv' format
`header`	Logical. Whether to use the first row as header.
`sep`	Character. The seperator of the file.

Value

A dataframe, with the first row set as header and the first column set as row name. The data is unified to numeric.

lEa *lEa()*

Description

Internal function to extract label data in lipidomicR

Usage

```
lEa(
  data,
 Inchannel = c("PC(15:0/18:1(d7))+AcO_1", "PE(15:0/18:1(d7))_2", "PS(15:0/18:1(d7))_2",
    "PG(15:0/18:1(d7))_2", "PI(15:0/18:1(d7))_2", "PA(15:0/18:1(d7))_2",
    "LPC18:1(d7)+AcO", "LPE18:1(d7)"),
  sample = 2:5
)
```

Arguments

`data`	Data frame. Row lipidomics data. Should be imported with importer().
`Inchannel`	Vector. Exact channel name for isotope label data.
`sample`	Vector. Column of sample added with isotope label. Default as 2:5.

Value

Data of internal label.

lEr *lEr*

Description

Internal function to extract label data in lipidomicR

Usage

```
lEr(
  data,
  Inlabel = c("PE(17:0/17:0)", "PC(17:0/17:0)", "PS(14:0/14:0)"),
  relative_as_default = TRUE,
  relative.mannual = NULL
)
```

Arguments

`data`	Data frame. Row lipidomics data. Should be imported with importer().
`Inlabel`	Vector. Name of Internal label.
`relative_as_default`	Logical, default as TRUE for automatically searching for internal standard data.
`relative.mannual`	Vector, the exact channel name.

Value

Data of internal label.

`lipidomicsR_env` *Global environment settings of LipidomicsR*

Description

Loading environment for LipidomicsR. Please do not call it directly.

Usage

```
lipidomicsR_env()
```

Value

No return value, called for loading environment.

`nor.absolute` *nor.absolute()*

Description

A function to calculate parameters for absolute normalization.

Usage

```
nor.absolute(data, radio.data = NULL, sample = 2:5)
```

Arguments

`data`	Data frame, row lipidomics data.
`radio.data`	Data frame. Characteristic of the radio label data. The row name must be the exact channel name of the label. Molecular mass should be provided in a column named "Mass". Concentration should be provided in a column named "Concentration(mg/ml)".
`sample`	Vector. Column of sample added with isotope label. Default as 2:5.

Value

Parameters for absolute normalization

`nor.relative` *nor.relative()*

Description

A function to calculate parameters for relative normalization.

Usage

```
nor.relative(
  data,
  Inlabel = c("PE(17:0/17:0)", "PC(17:0/17:0)", "PS(14:0/14:0)"),
  normalize_to = 1:5,
  relative_as_default = TRUE,
  relative.mannual = NULL
)
```

Arguments

`data`	Data frame, row lipidomics data.
`Inlabel`	Vector. Name of Internal label. Default as c('PE(17:0/17:0)','PC(17:0/17:0)','PS(14:0/14:0)')
`normalize_to`	Vector. The column of samples that used as the standard for normalization.
`relative_as_default`	Logical, default as TRUE for automatically searching for internal label data.
`relative.mannual`	Vector, the exact channel name (if you want to define the channel of internal label mannually).

Value

Parameters for relative normalization

noridx *noridx()*

Description

An integrated function that call nor.relative() and nor.absolute(), for simplifying.

Usage

```
noridx(data, radio.data = NULL,
normalization.mode='both',
sample = 1:5, normalize_to = 2:5,
Inlabel=c('PE(17:0/17:0)','PC(17:0/17:0)','PS(14:0/14:0)'),
relative_as_default = TRUE,
relative.mannual = NULL)
```

Arguments

`data`	Data frame, row lipidomics data.
`radio.data`	Data frame. Characteristic of the radio label data. The row name must be the exact channel name of the label. Molecular mass should be provided in a column named "Mass". Concentration should be provided in a column named "Concentration(mg/ml)".
`normalization.mode`	Character. "absolute" tp output absolute normalization index 'relative' to output relative normalization index. "both" to output both of them. Default as "both".
`sample`	Vector. Column of sample added with isotope label. Default as 2:5.
`normalize_to`	Vector. The column of samples that used as the standard for normalization.
`Inlabel`	Inlabel Vector. Name of Internal label. Default as c('PE(17:0/17:0)','PC(17:0/17:0)','PS(14:0/14:0)')
`relative_as_default`	Logical, default as TRUE for automatically searching for internal label data.
`relative.mannual`	Vector, the exact channel name (if you want to define the channel of internal label mannually).

Value

1. normalization.mode='both'. A list of data frames of normalization indexes of the two modes. 2. normalization.mode='absolute' or 'relative'. A data frame of the respective normlaization index.

---

normalization_calculator *normalization_calculator()*

Description

A function uses normalization parameters to calculate normalized lipidomic data.

Usage

```
normalization_calculator(
  data,
  normalization.mode,
  normalization.index,
  delete.pattern = c("_\\d", "(\\+)AcO", "_n", "\\(\\d+\\)"),
  group,
  to
)
```

Arguments

`data`	Data frame. Row lipidomic data.
`normalization.mode`	Vector. Options can be 'absolute', 'relative', 'TSA', 'toGroup'.
`normalization.index`	Data frame. normalization parameters, Suggested to be calculated by noridx.ouput()
`delete.pattern`	Pattern of characters that needs to be removed.
`group`	Vector. The group information, recommended to be generated with groupXpert()
`to`	Character. The group to be normalized to.

Value

A data frame of normalized data if either 'relative' or 'absolute' mode is used. A list if both of them are used.

---

percent.calculator *percent.calculator()*

Description

A function to calculate the proportion of each lipid content.

Usage

```
percent.calculator(
  data,
  delete.pattern = c("_\\d", "(\\+)AcO", "_n", "\\(\\d+\\)")
)
```

Arguments

`data`	Data frame. The row lipidomic data.
`delete.pattern`	Pattern of characters that needs to be removed.

Value

Return a data frame of normalized lipidomic data in the percentage of lipid content.

`plotRadar` *plotRadar()*

Description

A function to produce radar plot based on lipid types, carbon number, and number of double bonds.

Usage

```
plotRadar(
  data,
  pattern,
  group,
  max = 0.6,
  min = 0,
  method = "median",
  axislabcol = "grey",
  plwd = 2,
  plty = 1,
  cglcol = 1,
  seg = 4,
  cglwd = 1,
  cglty = 3,
  vlcex = 1,
  axistype = 1,
  t.size = 15,
  t.vjust = 0,
  t.color = "black",
  l.postion = "topright",
  l.bty = "n",
  lt.col = "grey25",
  lt.cex = 2,
  l.cex = 1
)
```

Arguments

`data`	A dataframe storing absolute concentration or PL to get normalized data. The column name should be the sample name and the row name should be the lipid type. The class of column name and row name should be "character". The class of values should be "numeric". The row names are recommended to be in a form like "PL(14:0/20:1)" or "LPL(16:1)".
`pattern`	Can accept 4 values: "lipid", "CB", "sat", or "all" If pattern="lipid“, a new radar diagram based on lipid type will be saved, which was named as "lipid_RadarChart.pdf" If pattern="CB“, a new radar diagram based on carbon number will be saved, which was named as "carbon_number_RadarChart.pdf" If pattern="sat“, a new radar diagram based on the number of double bonds will be saved, which was named as "unsaturation_RadarChart.pdf" If pattern="all“, all three diagrams will be saved.
`group`	A vector defining which group the replicates belong to. Notice: the number of groups should be less than 17.
`max`	The maximal absolute concentration or PL% values of each class. The default value is 0.6.
`min`	The minimal absolute concentration or PL% values of each class. The default value is 0.
`method`	The method to select the representative value from a group, which can be "median" or "mean". If it equals "median", the median of the group samples will be chosen. Otherwise, the mean will be chosen to plot.
`axislabcol`	The color of axis, default value is "grey".
`plwd`	Defines the width of the data series line. Default value is 2.
`plty`	Specifies the style of the data series line, which can be 1-6. Default value is 1.
`cglcol`	Specifies the color of the gridlines. Default value is 1.
`seg`	Defines the number of gridlines. Default value is 4, which means 5 gridlines: "0%", "25%", "50%", "75%", and "100%".
`cglwd`	Specifies the width of the gridlines. Default value is 1.
`cglty`	Specifies the grid line style, which can be 1-6. Default value is 3.
`vlcex`	Specifies the size of the group label font. Default value is 1.
`axistype`	Specifies the style of the axis, which can be 0-5. Default value is 1.
`t.size`	The size of picture title. Default value is 15.
`t.vjust`	The vertical position of picture title, which can be negative or positivew values. Default value is 0.
`t.color`	The color of picture title. Default value is "black".

`l.postion`	The position of legend, which can be "bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right" or "center". Default value is "topright".
`l.bty`	Whether the legend box is drawn, "o" means drawn, and the default value is "n" not drawn.
`lt.col`	The color of legend text. Default value is "grey25".
`lt.cex`	The fontsize of legend text. Default value is 2.
`l.cex`	The size of legend. Default value is 1.

Value

No return value, called for side effects, which is a radar diagram based on lipid type, carbon number, or unsaturation among different groups.

Examples

```
WT_1=rnorm(n=10,mean=0.4,sd=0.1)
WT_2=rnorm(n=10,mean=0.4,sd=0.1)
WT_3=rnorm(n=10,mean=0.4,sd=0.1)
WT_4=rnorm(n=10,mean=0.4,sd=0.1)
KO_1=rnorm(n=10,mean=0.8,sd=0.1)
KO_2=rnorm(n=10,mean=0.8,sd=0.1)
KO_3=rnorm(n=10,mean=0.8,sd=0.1)
KO_4=rnorm(n=10,mean=0.8,sd=0.1)
WT_treat_1=rnorm(n=10,mean=0.1,sd=0.1)
WT_treat_2=rnorm(n=10,mean=0.1,sd=0.1)
WT_treat_3=rnorm(n=10,mean=0.1,sd=0.1)
WT_treat_4=rnorm(n=10,mean=0.1,sd=0.1)
KO_treat_1=rnorm(n=10,mean=0.6,sd=0.1)
KO_treat_2=rnorm(n=10,mean=0.6,sd=0.1)
KO_treat_3=rnorm(n=10,mean=0.6,sd=0.1)
KO_treat_4=rnorm(n=10,mean=0.6,sd=0.1)
data=data.frame(WT_1,WT_2,WT_3,WT_4,KO_1,KO_2,KO_3,KO_4,
                WT_treat_1,WT_treat_2,WT_treat_3,WT_treat_4,
                KO_treat_1,KO_treat_2,KO_treat_3,KO_treat_4)
rownames(data)=c("LPC(16:0)","PC(14:0/16:1)","PC(18:1/18:1)","PE(18:0/20:1)",
                 "PS(20:1/20:1)","PI(16:0/16:1)","PC(18:0/18:1)","PA(16:0/16:1)",
                 "LPE(18:0)","PE(O-18:1/18:0)")
group=rep(c("WT","KO","WT_treat","KO_treat"),each=4)
plotRadar(data,"all",group) # This is the most simplified version
```

`QCplot` *QCplot()*

Description

A function to exhibit the data quality between different samples, including correlation heatmap, PCA, and quality boxplot.

Usage

```
QCplot(
  data,
  ptype,
  group,
  qcdt,
  box.sample.name = c("default"),
  box.x = "sample",
  box.y = "abundance",
  box.title = "",
  errorbar.show = TRUE,
  group.col = c("default"),
  outlie.col = NA,
  outlie.shape = NA,
  heat.sample.name = c("default"),
  heat.start.col = "white",
  heat.end.col = "#3E8BCA",
  heat.title = "Correlation Heatmap",
  group.show = TRUE,
  range.show = TRUE,
  range.alpha = 0.25,
  shape = TRUE,
  pca.title = "PCA Scores Plot",
  point.size = 3.5,
  point.t.size = 1.5,
  point.t.color = "grey25",
  point.t.overlap = 200,
  marked = c("A", "B", "C"),
  combine = TRUE,
  title.hjust = 0.5,
  title.vjust = 0,
  title.size = 15,
  a.title.size = 13,
  a.text.size = 8,
  a.text.angle = 45,
  a.text.vjust = 1,
  a.text.hjust = 1,
  l.text.size = 11,
  l.title.size = 13,
  margin = c(1, 1, 1, 1),
  unit = "in",
  cellsize = 8,
  interactive = TRUE
)
```

Arguments

`data`	A dataframe storing concentration of lipids between different samples. The column name should be the sample name and the row name should be the lipid type. The class of column name and row name should be "character". The class of values should be "numeric". The row names are recommended to be in a form like "PL(14:0/20:1)" or "LPL(16:1)". It is highly recommended to input data after normalization.
`ptype`	A vector to define picture types output. The input can include "heatmap", "PCA" and "boxplot".
`group`	A vector defining which group the replicates belong to.
`qcdt`	A dataframe containing internal labels and their abundance, which is used to draw quality boxplot. If you don't want to draw the boxplot, the paramenter can be ignored.
`box.sample.name`	A vector containing the sample names, the length of "box.sample.name" should equal the number of samples. This parameter can only change the sample name of boxplot. The default values are the column names of the data input.
`box.x`	The name of the x axis of the boxplot. The default value is "sample".
`box.y`	The name of the x axis of the boxplot. The default value is "abundance".
`box.title`	The picture title of the boxplot. The default value is "".
`errorbar.show`	Whether show the errorbars of the boxplot or not. The default value is "".
`group.col`	A vector containing the colors for each group. The length of "group.col" should equal the number of the groups. If not input, the color will be default values.
`outlie.col`	The color of outliers. The default value is NA (not show outliers).
`outlie.shape`	The shape of outliers, which can be 1 - 25. The default value is NA (not show outliers).
`heat.sample.name`	A vector containing the sample names, the length of "heat.sample.name" should equal the number of samples. This parameter can only change the sample name of heatmap. The default values are the column names of the data input.
`heat.start.col`	The lightest color of the heatmap. The default value is "white".
`heat.end.col`	The deepest color of the heatmap. The default value is "#3E8BCA".
`heat.title`	The picture title of the heatmap. The default value is "Correlation Heatmap".
`group.show`	Whether to show the classification of different groups of the heatmap. The default value is TRUE.
`range.show`	Whether to show the range of PCA plot. The default value is TRUE.
`range.alpha`	The transparency of the range in the PCA, only useful when range.show = TRUE. The default value is 0.25.
`shape`	Whether to classify different groups by shape. The default value is TRUE.
`pca.title`	The picture title of the PCA plot. The default value is "PCA Scores Plot".
`point.size`	The size of points in the PCA plot. The default value is 3.5.

`point.t.size`	The size of texts labeled on points. The default value is 1.5.
`point.t.color`	The color of texts labeled on points. The default value is "grey25".
`point.t.overlap`	To let the texts of points be shown without overlapping. The default value is 200. If you don't want to show texts labeled on points, please set "point.t.size=0" and "point.t.overlap=0".
`marked`	Only useful when combine = T, it decides the labels on the top left of the picture. The default value is c("A", "B","C"). If you don't want to show, use "marked=c("", "","") ".
`combine`	Whether to combine the three plots when ptype = "all". The default value is T. If combine = FALSE, the three plots will be returned separately.
`title.hjust`	Define the horizontal position of the picture title. The default value is 0.5.
`title.vjust`	Define the vertical position of the picture title. The default value is 0.
`title.size`	Define the size of the picture title. The default value is 15.
`a.title.size`	Define the size of the axis title. The default value is 13.
`a.text.size`	Define the size of the axis text. The default value is 8.
`a.text.angle`	Define the angle of the X axis text of the boxplot, which can be 0 - 360. The default value is 45.
`a.text.vjust`	Define the vertical position of the axis text. The default value is 1.
`a.text.hjust`	Define the horizontal position of the axis text. The default value is 1.
`l.text.size`	Define the size of the legend. The default value is 11.
`l.title.size`	Define the size of the legend. The default value is 13.
`margin`	Define the margin surrounding the plot area of each plot. It should be a vector whose length = 4. The default value is c(0.4,0.4,0.4,0.4).
`unit`	The unit of the "margin" parameter, which can be "mm", "cm", "in", "pt", and "pc". The default value is "in".
`cellsize`	The size of a cell in the heatmap. Default value is 8.
`interactive`	Whether to get an interactive PCA plot or not. Default value is TRUE.

Value

A correlation heatmap, PCA plot, quality boxplot, or a merged pictures containing the above three.

`relative_calculator` *relative.calculator()*

Description

Internal function of lipidmicR::normalization_calculator()

Usage

```
relative_calculator(data, relative.dataset)
```

Arguments

`data`	Row data of lipidomics
`relative.dataset`	Data frame. Normalization index.

Value

Return a data frame of relative normalized lipidomic data.

`replicate.delete` *Delete repeated channel data generated by LC-MS*

Description

replicate.delete select one of the channel with largest mean value out of multiple replicated LC-MS cahnnels, and it returns an edited data frame with no repeated channel and simplified row name.

Usage

```
replicate.delete(data, delete.pattern = c("_\\d", "(\\+)AcO"))
```

Arguments

`data`	Primary lipidomic data in .csv format.
`delete.pattern`	Pattern of characters that needs to be removed.

Value

Edited data with no repeat channels and simplified row name.

Examples

```
x1 <- c(1,1,1,1)
x2 <- c(1.1,1.2,1.3,1.4)
x3 <- c(1,1,1,1)
x4 <- c(2,2,2,2)
rowNames <- c('PC(14:0/14:1)+AcO_1','PC(14:0/14:1)+AcO_2','PC(16:0/16:1)_1','PC(16:0/16:1)_2')
ExampleData <- data.frame(x1,x2,x3,x4, row.names =rowNames)
replicate.delete(ExampleData,delete.pattern= c('_\\d','(\\+)AcO'))
```

---

rsd_calculator *Calculate Relative Standard Deviation (RSD)*

Description

This function calculates the relative standard deviation (RSD) based on the specified data range.

Usage

```
rsd_calculator(
  data,
  start,
  end,
  threshold = 0.2,
  show.del = FALSE,
  del.zero = TRUE
)
```

Arguments

`data`	The data frame containing abundance data.
`start`	The starting column index of the QC data range.
`end`	The ending column index of the QC data range.
`threshold`	The threshold value for RSD. Default is 0.2.
`show.del`	Logical value indicating whether to show the deleted data. Default is FALSE.
`del.zero`	Logical value indicating whether to delete rows with all QC being zero . Default is TRUE.

Value

A data frame containing the calculated RSD values and the corresponding data.

Examples

```
qc_1=rnorm(n=5,mean=0.3,sd=0.2)
qc_2=rnorm(n=5,mean=0.3,sd=0.2)
qc_3=rnorm(n=5,mean=0.3,sd=0.2)
qc_4=rnorm(n=5,mean=0.3,sd=0.2)
qc_5=rnorm(n=5,mean=0.3,sd=0.2)
WT_1=rnorm(n=5,mean=0.3,sd=0.1)
WT_2=rnorm(n=5,mean=0.3,sd=0.1)
WT_3=rnorm(n=5,mean=0.3,sd=0.1)
KO_1=rnorm(n=5,mean=0.3,sd=0.1)
KO_2=rnorm(n=5,mean=0.3,sd=0.1)
KO_3=rnorm(n=5,mean=0.3,sd=0.1)
data=data.frame(qc_1,qc_2,qc_3,qc_4,qc_5,WT_1,WT_2,WT_3,KO_1,KO_2,KO_3)
rownames(data)=c("LPC(16:0)","PC(14:0/16:1)","PC(18:1/18:1)","PE(18:0/20:1)","PS(20:1/20:1)")
rsd_calculator(data,1,5,show.del = TRUE)
```

`sepclass` *sepclass()*

Description

A function to identify lipid type and calculate the number of carbons and unsaturation rate of lipids.

Usage

```
sepclass(data, pattern, subtype = FALSE)
```

Arguments

`data`	A dataframe storing concentration of lipids between different samples. The column name should be the sample name and the row name should be the lipid type. The class of column name and row name should be "character". The class of values should be "numeric". The row names are recommended to be in a form like "PL(14:0/20:1)" or "LPL(16:1)".
`pattern`	Can accept 4 values: "lipid", "CB", "sat", or "all" If pattern="lipid“, a new column named "lipid_type" will be added, which stores type of lipid like "PE", "LPC", and "TAG". If pattern="CB“, a new column named "carbon_number" will be added, which stores the number of carbons. For example, the carbon_number of "PC(14:0/16:1)" is 14+16=30. If pattern="sat“, a new column named "unsaturation" will be added, which stores the number of double bonds. For example, the unsaturation of "PC(14:1/16:1)" is 1+1=2. If pattern="all“, all three columns will be added.
`subtype`	A logic value to determine for a lipid like "PC(O-14:0/16:1)", "lipid_type" should be "PC" (subtype=FALSE) or "PC(O)" (subtype=TRUE). Default value is FALSE.

Value

A dataframe with new columns containing the class of lipid type, carbon number, or unsaturation based on the original data input.

Examples

```
WT_1=rnorm(n=5,mean=0.3,sd=0.1)
WT_2=rnorm(n=5,mean=0.3,sd=0.1)
WT_3=rnorm(n=5,mean=0.3,sd=0.1)
KO_1=rnorm(n=5,mean=0.3,sd=0.1)
KO_2=rnorm(n=5,mean=0.3,sd=0.1)
KO_3=rnorm(n=5,mean=0.3,sd=0.1)
data=data.frame(WT_1,WT_2,WT_3,KO_1,KO_2,KO_3)
rownames(data)=c("LPC(16:0)","PC(O-14:0/16:1)","TAG56:2-FA20:1","PE(P-18:0/20:1)","PS(20:1/20:1)")
pattern="all" ## or "lipid", "CB", "sat"
sepclass(data,pattern)
```

---

toGroup.calculator *toGroup.calculator()*

Description

toGroup.calculator()

Usage

```
toGroup.calculator(
  data,
  group,
  to,
  delete.pattern = c("_\\d", "(\\+)AcO", "_n", "\\(\\d+\\)")
)
```

Arguments

data	Data frame. The row lipidomic data.
group	Vector. The group information, recommended to be generated with groupX-pert()
to	Character. The group to be normalized to.
delete.pattern	Pattern of characters that needs to be removed.

Value

A dataframe normalized to specified group.

`total.abundance` *Calculate Total Abundance Data*

Description

This function calculates total abundance data based on specified groups.

Usage

```
total.abundance(data)
```

Arguments

`data`	The data frame containing abundance data.

Value

A data frame containing total abundance of different lipid type, carbon number, and unsaturation.

`volcano` *Generate Volcano Plot*

Description

This function generates a volcano plot for differential expression analysis results.

Usage

```
volcano(
  data,
  x.scale,
  y.scale,
  interact = FALSE,
  FC.threshold = 2,
  P.threshold = 0.05,
  change.label = c("Up", "Down", "Notsig"),
  point.size = 2,
  point.color = c("lightsalmon2", "cadetblue", "grey"),
  x_scale_mannual = FALSE,
  y_scale_mannual = FALSE,
  linetype = 4,
  line.alpha = 0.4,
  line.color = "grey34",
  line.size = 1,
  annotation.label = NULL,
  annotation.color = "#C82423",
  text.size = 2.5,
  max.overlap = 40,
  title = NULL
)
```

Arguments

`data`	The data frame containing the results of differential expression analysis.
`x.scale`	The manual limits for the x-axis (default is NULL).
`y.scale`	The manual limits for the y-axis (default is NULL).
`interact`	Logical value indicating whteher to generate interactive volcano plot.
`FC.threshold`	The fold change threshold for determining significant changes (default is 2).
`P.threshold`	The significance threshold for p-values (default is 0.05).
`change.label`	The labels for differentially expressed genes (default is c('Up', 'Down', 'Not-sig')).
`point.size`	The size of data points in the plot (default is 2).
`point.color`	The colors for differentially expressed genes (default is c('lightsalmon2', 'cadetblue', 'grey')).
`x_scale_mannual`	Logical value indicating whether to manually specify x-axis limits (default is FALSE).
`y_scale_mannual`	Logical value indicating whether to manually specify y-axis limits (default is FALSE).
`linetype`	The line type for significance thresholds (default is 4).
`line.alpha`	The transparency level for significance thresholds (default is 0.4).
`line.color`	The color for significance thresholds (default is 'grey34').
`line.size`	The size of significance thresholds (default is 1).
`annotation.label`	The name of species that need to be annotated (default is NULL).
`annotation.color`	The color of the annotation points (default is '#C82423')
`text.size`	The size of gene labels (default is 1.5).
`max.overlap`	The maximum number of overlapping labels allowed (default is 40).
`title`	The title for the plot (default is NULL).

Value

A list containing the plot object, data frame with plotted points, and omitted data points. If interact = TRUE, the html object of the interactive plot will be also returned.

Index