Package: chemhelper (via r-universe)

September 13, 2024

Version 0.0.0.9000

Title Helper Functions For Dealing With GCMS and LCMS data from IonAnalytics

Description Provides helper functions for parsing data exported from IonAnalytics, calculating retention indecies, and other miscelanous helper functions to assist in data wrangling.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

ByteCompile true

Roxygen list(markdown = TRUE)

RoxygenNote 6.1.1

Imports readr, dplyr, stringr, ropls, tibble, purrr, webchem, broom, MASS, latex2exp, ggplot2

Suggests testthat, covr

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

RemoteUrl https://github.com/Aariq/chemhelper

RemoteRef HEAD

RemoteSha e51054c1856e3e446419d58c0351e391299e94e3

Contents

`calc_RI` *Calculate Van Den Dool and Kratz Retention Indicies*

Description

This function calculates retention indices using the Van Den Dool and Kratz equation

Usage

```
calc_RI(rts, alkanesRT, C_num)
```

Arguments

`rts`	A vector of retention times to be converted to retention indices
`alkanesRT`	A vector of retention times of standard alkanes, in descending order
`C_num`	A vector of the numbers of carbons for each of the alkanes

Value

A vector of retention indices

See Also

`calc_RT`

Examples

```
alkanes <- data.frame(RT = c(1.88, 2.23, 5.51, 8.05, 10.99,
                             14.10, 17.20, 20.20, 22.90, 25.60,
                             28.10, 30.50, 32.81, 35.22, 37.30),
                      C_num = 6:20)
calc_RI(11.237, alkanes$RT, alkanes$C_num)
```

calc_RT *Back-calculate Retention Times*

Description

This function back-calculates expected retention times given a Van Den Dool and Kratz retention index

Usage

```
calc_RT(ris, alkanesRT, C_num)
```

Arguments

`ris`	A vector of retention indices used to estimate retention times
`alkanesRT`	A vector of retention times of standard alkanes, in descending order
`C_num`	A vector of the numbers of carbons for each of the alkanes

Value

A vector of expected retention times

See Also

`calc_RI`

Examples

```
alkanes <- data.frame(RT = c(1.88, 2.23, 5.51, 8.05, 10.99,
                             14.10, 17.20, 20.20, 22.90, 25.60,
                             28.10, 30.50, 32.81, 35.22, 37.30),
                      C_num = 6:20)
calc_RT(1007.942, alkanes$RT, alkanes$C_num)
```

chem_scale *Scaling Functions for Metabolomics*

Description

Provides additional scaling functions besides autoscaling. Reviewed in van den Berg et al. 2006.

Usage

```
chem_scale(x, center = TRUE, scale = c("auto", "pareto", "range",
  "vast", "level", "none"))
```

Arguments

`x`	a vector
`center`	logical. Do you want to apply centering?
`scale`	choice of scaling functions. Defaults to autoscaling (dividing by standard deviation). See details for more.

Details

Currently the choices for `scale =` allow for all of the scaling methods reviewed in Berg et al. 2006. *Centering, scaling, and transformations: improving the biological information content of metabolomics data.* BMC Genomics 7:142. Autoscaling divides each number by the column standard deviation. Pareto scaling divides each number by the square root of the column standard deviation. Compared to autoscaling, this stays closer to the original measurments, but is highly sensitive to large fold changes. Range scaling divides the numbers by the column range, which may be useful in cases when scaling relative to a biologically possible range is desired, however this method is highly sensitive to outliers. Vast scaling multiplies the autoscaled results by the ratio of the column mean or some group mean to the column/group standard deviation. With this method, one could take knowledge of groups into account, although this isn't currently implemented in this function. Level scaling simply divides by the column mean, transforming values into relative responses.

Value

Scaled vector with attributes showing the scaling and centering parameters

Examples

```
x = c(0, 0.1, 0.2, 10)
y = c(1000, 1232, 2022, 4000)

chem_scale(x, center = TRUE, scale = "auto")
chem_scale(y, center = TRUE, scale = "pareto")
```

`get_loadings` *Get axis loadings from models created by* `ropls::opls()`

Description

Provides a wrapper for `getLoadingMN` from the ropls package that returns a tibble rather than a matrix

Usage

```
get_loadings(.model)
```

Arguments

`.model`	a pls object created by `opls`

Value

a tibble

Examples

```
## Not run:
pls.model <- opls(X, Y)
get_loadings(pls.model)

## End(Not run)
```

---

get_modelinfo — *Retrieve model parameters from models created by `ropls::opls()` For PCA, returns percent variance explained by each axis. For (o)PLS(-DA), returns variance explained by axes and cross-validation statistics.*

Description

Retrieve model parameters from models created by `ropls::opls()` For PCA, returns percent variance explained by each axis. For (o)PLS(-DA), returns variance explained by axes and cross-validation statistics.

Usage

```
get_modelinfo(model)
```

Arguments

`model`	a model object created by `opls()`

Value

a list of two dataframes, `axis_stats` and `validation`

Examples

```
## Not run:
pls.model <- opls(X, Y)
get_modelinfo(pls.model)

## End(Not run)
```

get_plotdata *Extract data for plotting (O)PLS(-DA) data with ggplot2*

Description

Extracts relevant data from an "opls" object for making annotated score plots with ggplot2 or other plotting packages.

Usage

```
get_plotdata(model)
```

Arguments

`model`	An object created by `opls`

Value

A list containing dataframes for scores, loadings, axis statistics (

Examples

```
## Not run:
library(ropls)
data(sacurine)
sacurine.oplsda <- opls(sacurine$dataMatrix, sacurine$sampleMetadata[, "gender"],
                        predI = 1,
                        orthoI = NA)
df <- get_plotdata(sacurine.oplsda)

## End(Not run)
```

get_scores *Get axis scores from models created by `ropls::opls()` Returns a dataframe of PC axis scores for PCA, predictive axis scores for PLS and PLS-DA, and predictive and orthogonal axis scores for OPLS and OPLS-DA models.*

Description

Get axis scores from models created by `ropls::opls()` Returns a dataframe of PC axis scores for PCA, predictive axis scores for PLS and PLS-DA, and predictive and orthogonal axis scores for OPLS and OPLS-DA models.

Usage

```
get_scores(model)
```

Arguments

`model`	a model object created by `opls()`

Value

a dataframe

Examples

```
## Not run:
pls.model <- opls(X, Y)
get_scores(pls.model)

## End(Not run)
```

get_VIP — *Get VIP scores from PLS and OPLS models created by* `ropls::opls()`

Description

Provides a wrapper for `getVipVn` from the ropls package that returns a tibble rather than a named numeric vector.

Usage

```
get_VIP(.model)
```

Arguments

`.model`	a pls object created by `opls`

Value

a tibble

Examples

```
## Not run:
pls.model <- opls(X, Y)
get_VIP(pls.model)

## End(Not run)
```

`parse_IA` *Parse IonAnalytics CSV files*

Description

Parse IonAnalytics CSV files

Usage

```
parse_IA(file)
```

Arguments

`file`	raw text

Value

a string.

`plot_opls` *Plot OPLS regression models produced by* `ropls::opls()`

Description

Plot OPLS regression models produced by `ropls::opls()`

Usage

```
plot_opls(ropls_pls, annotate = c("caption", "subtitle"))
```

Arguments

`ropls_pls`	an OPLS model with a continuous Y variable produced by `ropls::opls()`
`annotate`	location on the plot to print model statistics

Value

a ggplot object

Examples

```
## Not run:
plot_opls(opls)

## End(Not run)
```

plot_oplsda *Plot OPLS-DA models produced by* `ropls::opls()`

Description

Plot OPLS-DA models produced by `ropls::opls()`

Usage

```
plot_oplsda(ropls_pls, annotate = c("caption", "subtitle"))
```

Arguments

`ropls_pls`	an OPLS-DA model with a discrete Y variable produced by `ropls::opls()`
`annotate`	location on the plot to print model statistics

Value

a ggplot object

Examples

```
## Not run:
plot_oplsda(oplsda)

## End(Not run)
```

plot_pca *Plot PCA models created by* `ropls::opls()`

Description

Plot PCA models created by `ropls::opls()`

Usage

```
plot_pca(ropls_pca, group_var = NULL, annotate = c("caption",
  "subtitle", "none"))
```

Arguments

`ropls_pca`	a PCA model produced by `ropls::opls()`
`group_var`	a discrete variable used to plot groups
`annotate`	location on the plot to print model statistics

Value

a ggplot object

Examples

```
## Not run:
plot_pca(pca, data$treatment)

## End(Not run)
```

`plot_pls` *Plot PLS regression models produced by* `ropls::opls()`

Description

Plot PLS regression models produced by `ropls::opls()`

Usage

```
plot_pls(ropls_pls, annotate = c("caption", "subtitle"))
```

Arguments

`ropls_pls`	a PLS model with a continuous Y variable produced by `ropls::opls()`
`annotate`	location on the plot to print model statistics

Value

a ggplot object

Examples

```
## Not run:
plot_pls(pls)

## End(Not run)
```

`plot_plsda` *Plot PLS-DA models produced by* `ropls::opls()`

Description

Plot PLS-DA models produced by `ropls::opls()`

Usage

```
plot_plsda(ropls_plsda, annotate = c("caption", "subtitle"))
```

Arguments

`ropls_plsda`	a PLS-DA model with a discrete Y variable produced by `ropls::opls()`
`annotate`	location on the plot to print model statistics

Value

a ggplot object

Examples

```
## Not run:
plot_plsda(plsda)

## End(Not run)
```

`read_IA` *Read IonAnalytics CSV files*

Description

Reads csv files exported from IonAnalytics methods or integration reports. These csv files are poorly formatted and include line breaks within headers so `read_csv()` doesn't work

Usage

```
read_IA(file)
```

Arguments

`file`	a path to a csv file exported by IonAnalytics

Value

A dataframe

Examples

```
## Not run:
read_IA("report.csv")

## End(Not run)
```

VDDK_RI *Calculate a single Van Den Dool and Kratz Retention Index*

Description

Calculate a single Van Den Dool and Kratz Retention Index

Usage

```
VDDK_RI(rt, alkanesRT, C_num)
```

Arguments

rt	The retention time of the compound
alkanesRT	A vector of retention times of alkanes, in descending order
C_num	A vector of the numbers of carbons for each of the alkanes

Value

A retention index

VDDK_RT *Calculate a single retention time given a Van Den Dool and Kratz RI*

Description

Calculate a single retention time given a Van Den Dool and Kratz RI

Usage

```
VDDK_RT(ri, alkanesRT, C_num)
```

Arguments

ri	The retention index of the compound
alkanesRT	A vector of retention times of alkanes, in descending order
C_num	A vector of the numbers of carbons for each of the alkanes

Value

a retention time

Index