

MeTime R Package

July 18, 2022

<code>add_col_normality</code>	<i>Function to check normality and add data to col data</i>
--------------------------------	---

Description

A method applied on the s4 object of class "metime_analyser" to check normality of the metabolites and add it to corresponding columns

Usage

```
add_col_normality(object, which_data, type, metab_names)
```

Arguments

<code>object</code>	An object of class metime_analyser
<code>which_data</code>	dataset on which the method is to be applied
<code>type</code>	type of test, currently only shapiro-wilk test is available under the choice "shapiro"
<code>metab_names</code>	column that has the metabolite names in col_data.

Value

S4 object with shapiro wilk test related data in the col_data

Examples

```
object <- add_col_normality(object=data, which_data=c("lipid_data","nmr_data"), type="shapiro", metab_names=c("m
```

add_screening_vars	<i>Function to add measurements taken at screening time for samples to be added to all timepoints</i>
--------------------	---

Description

A method applied on the s4 object of class "metime_analyser" to add all those datapoints that were measured only during screening to all the respective samples at all timepoints

Usage

```
add_screening_vars(object, vars)
```

Arguments

object	An object of class metime_analyser
vars	A character naming the vars of interest

Value

phenotype data which can be replaced into the original object or use it separately with a different object

Examples

```
# adding APOEGrp, PTGENDER to all data points
new_with_apoegrp_sex <- add_screening_vars(object=metime_analyser_object, vars=c("APOEGrp", "PTGENDER"))
```

calc_conservation_metabolite	<i>Function to calculate metabolite conservation index</i>
------------------------------	--

Description

Method applied on the object metime_analyser to calculate the metabotype conservation index

Usage

```
calc_conservation_metabolite(object, which_data, verbose)
```

Arguments

object	An object of class metime_analyser
which_data	Name of the dataset to be used
verbose	Information provided on steps being processed

Value

List of conservation index results

Examples

```
#calculating metabolite_conservation_index  
out <- calc_metabolite_conservation(object=metime_analyser_object, which_data="Name of the dataset")
```

calc_conservation_metabotype

Function to calculate metabotype conservation index

Description

Method applied on the object metime_analyser to calculate the metabotype conservation index

Usage

```
calc_conservation_metabotype(object, which_data, verbose)
```

Arguments

object	An object of class metime_analyser
which_data	Name of the dataset to be used
verbose	Information provided on steps being processed

Value

List of conservation index results

Examples

```
#calculating metabotype_conservation_index  
out <- calc_metabotype_conservation(object=metime_analyser_object, which_data="Name of the dataset")
```

calc_correlation_pairwise

Function to calculate correlation

Description

calculate pairwise correlations This function creates a dataframe for plotting from a dataset.

Usage

```
calc_correlation_pairwise(object, which_data, method)
```

Arguments

object	S4 Object of class metime_analyser
which_data	specify datasets to calculate on. One or more possible
method	default setting: method="pearson", Alternative "spearman" also possible

Value

data.frame with pairwise results

Examples

```
# Example to calculate correlations
dist <- calc_correlation(object=metime_analyser_object, which_data="name of the dataset",
  method="pearson")
```

calc_dimensionality_reduction

Function to calculate dimensionality reduction methods such as tsne, umap and pca.

Description

A method to apply on s4 object of class metime_analyse in order to obtain information after dimensionality reduction on a dataset/s

Usage

```
calc_dimensionality_reduction(object, which_data, type)
```

Arguments

object	An object of class metime_analyser
which_data	a character vector - Names of the dataset from which the samples will be extracted
type	type of the dimensionality reduction method to be applied. Accepted inputs are "UMAP", "tSNE", "PCA"

Value

a list with two dataframes containing the dimensionality reduction information 1) samples - data of the individuals(".\$samples") 2) metabs - data of the metabolites(".\$metabs")

Examples

```
#calculate PCA
pca <- calc_dimensionality_reduction(object=metime_analyser_object, which_data="name/s of the dataset/s", type="PCA")
#calculate UMAP
pca <- calc_dimensionality_reduction(object=metime_analyser_object, which_data="name/s of the dataset/s", type="UMAP")
#calculate tSNE
pca <- calc_dimensionality_reduction(object=metime_analyser_object, which_data="name/s of the dataset/s", type="tSNE")
```

calc_distance_pairwise

Function to calculate dissimilarity using distance measures

Description

calculate pairwise distances This function creates a dataframe for plotting from a dataset.

Usage

```
calc_distance_pairwise(object, which_data, method)
```

Arguments

object	S4 Object of class metime_analyser
which_data	specify datasets to calculate on. One or more possible
method	default setting: method="euclidean", Alternative "maximum", "minimum", "manhattan", "canberra", "minkowski" are also possible

Value

data.frame with pairwise results

Examples

```
# Example to calculate pairwise distances
dist <- calc_pairwise_distance(object=metime_analyser_object, which_data="name of the dataset",
                              method="euclidean")
```

calc_featureselection_boruta

Function to calculate dependent variables

Description

An S4 method to be applied on the metime_analyser object so as to calculate dependent variables

Usage

```
calc_featureselection_boruta(
  object,
  which_x,
  which_y,
  verbose,
  output_loc,
  file_name
)
```

Arguments

object	An object of class metime_analyser
which_x	Name of the dataset to be used for training
which_y	Name of the dataset to be used for testing
verbose	Information provided on steps being processed
output_loc	path to the parent directory where in the out file will be stored
file_name	name of the out file

Value

List of conservation index results

get_append_analyser_object

This function appends an object of class metime_analyser with a new dataset.

Description

function to apply on metime_analyse object to append a new dataset into the existing object

Usage

```
get_append_analyser_object(object, data, col_data, row_data, name)
```

Arguments

object	S4 object of class metime_analyser
data	data.frame containing data
col_data	data.frame containing col_data: id column of col data has to match colnames of data
row_data	data.frame containing row_data: id column of row data has to match rownames of data
name	Name of the new dataset

Value

An object of class metime_analyser

Examples

```
# append data frames into the metime_analyser object
appended_object <- get_append_metab_object(object=metime_analyser_object, data=data, row_data=data, col_data=col_data)
```

get_files_and_names	<i>Function to pack all the data into a single object of class "metime_analyser"</i>
---------------------	--

Description

This function loads all the files from the parent directory. It assumes a certain naming pattern as follows: "datatype_None|collrow_data.rds" Any other naming pattern is not allowed. The function first writes all files into a list and each type of data is packed into its respective class i.e. col_data, row_data or data

Usage

```
get_files_and_names(path, annotations_index)
```

Arguments

path	Path to the parent directory
annotations_index	a list to be filled as follows = list(phenotype="Name or index of the files", medication="Name or index of the files")

Value

An object of class metime_analyser

Examples

```
# Input in the parent directory from which the data files are to be extracted along with annotations_index to specify
get_files_and_names(path=/path/to/parent/directory, annotations_index=list(phenotype="Name of phenotype file", medication="Name of medication file"))
```

```
get_make_analyser_object
```

Function to pack all the data into a single object of class "metime_analyser"

Description

This function creates an object of class metime_analyser from a dataset.

Usage

```
get_make_analyser_object(
  data,
  col_data,
  row_data,
  annotations_index,
  name = NULL
)
```

Arguments

data	data.frame containing data
col_data	data.frame containing col_data: id column of col data has to match colnames of data
row_data	data.frame containing row_data: id column of row data has to match rownames of data
annotations_index	a list to be filled as follows = list(phenotype="Name or index of the file/list", medication="Name or index of the files/list")
name	character. Name you want to assign to the new dataset that is being added on

Value

An object of class metime_analyser

Examples

```
# new_metime_analyser_object <- get_make_metab_object(data=data_frame, col_data=col_data_frame, row_data=row_data,
  annotations_index=list(phenotype="name of phenotype", medication="name of medication"))
```

`get_make_plotter_object`*Function to make a plottable object for viz functions*

Description

function to generate metime_plotter object from plot data and metadata

Usage

```
get_make_plotter_object(data, metadata, calc_type, calc_info, plot_type, style)
```

Arguments

<code>data</code>	dataframe of plotable data obtained from calc object
<code>metadata</code>	dataframe with the metadata for the plot table mentioned above. To obtain these see <code>get_metadata_for_rows()</code> and <code>get_metadata_for_columns()</code>
<code>calc_type</code>	A character to specify type of calculation - will be used for comp_ functions
<code>calc_info</code>	A string to define the information about calculation
<code>plot_type</code>	type of the plot you want to build. eg: "box", "dot" etc. Its a character vector
<code>style</code>	Style of plot, accepted inputs are "ggplot", "circos" and "visNetwork". Is a singular option.

`get_metadata_for_columns`*Get metadata for columns(in most cases for metabolites)*

Description

function to generate a metadata list for building the MeTime plotter object

Usage

```
get_metadata_for_columns(object, which_data, columns, names, index_of_names)
```

Arguments

<code>object</code>	S4 object of class MeTime Analyser
<code>which_data</code>	Names of dataset/s to be used
<code>columns</code>	A list of character vectors for the columns of interest. Length of the list should be same as length of <code>which_data</code>
<code>names</code>	A Character vector with the new names for the columns mentioned above
<code>index_of_names</code>	character vector to define the name of the column in which names of the variables are stored

Value

data.frame with metadata information

get_metadata_for_rows *Get metadata for rows(in most cases for samples)*

Description

function to generate a metadata list for building the MeTime plotter object

Usage

```
get_metadata_for_rows(object, which_data, columns, names)
```

Arguments

object	S4 object of class MeTime Analyser
which_data	Names of dataset/s to be used
columns	A list of character vectors for the columns of interest. Length of the list should be same as length of which_data
names	A Character vector with the new names for the columns mentioned above

Value

data.frame with metadata information for rows

get_palette *Get a palette of "n" distinct colorblind friendly colors*

Description

Function to get a palette of distinct colorblind friendly colors, the distinctiveness is determined by the difference in their hue values.

Usage

```
get_palette(n)
```

Arguments

n	number of colors wanted in the palette
---	--

Value

a color palette vector with colors in the form of hex codes

Examples

```
# colors=get_palette(n=10)
```

```
get_samples_and_timepoints
```

Function to know the number of timepoints and the total number of samples available at that point

Description

A method applied onto s4 object of class "metime_analyser" so as to obtain the number of unique samples available at each timepoint.

Usage

```
get_samples_and_timepoints(object, which_data)
```

Arguments

object	An object of class metime_analyser
which_data	Name of the dataset in context

Value

A data table with timepoints and number of samples at each timepoint

Examples

```
# newdata <- get_samples_and_timepoints(object=metime_analyser_object, which_data="Name of dataset of interest")
```

```
get_text_for_plot
```

Function to Obtain textual information for visualization in interactive plots

Description

a standard function to be applied on data matrices or dataframes with the colnames of interest such that the information from columns is visualized in the interactive plot

Usage

```
get_text_for_plot(data, colnames)
```

Arguments

data	a dataframe with plotting data along with other variables for visualization
colnames	a character vector with the names of the variables that you want to see on the plot

Value

a vector with strings that can be parsed into plot_ly text.

Examples

```
# text = get_text(data=data.frame, colnames=c("names","of","columns", "of", "interest"))
```

metime_analyser-class *Constructor to generate an object of class metime_analyser. contains slots - list_of_data: For the list of all data matrices. - list_of_col_data: list of all the col data files in the same order. - list_of_row_data: list of all the row data files in the same order. - annotations: list with phenotype and medication. Each of which is character that represents the name of the aforementioned dataset types.*

Description

Constructor to generate an object of class metime_analyser. contains slots - list_of_data: For the list of all data matrices. - list_of_col_data: list of all the col data files in the same order. - list_of_row_data: list of all the row data files in the same order. - annotations: list with phenotype and medication. Each of which is character that represents the name of the aforementioned dataset types.

Constructor to generate an object of class metime_analyser. contains slots - list_of_data: For the list of all data matrices. - list_of_col_data: list of all the col data files in the same order. - list_of_row_data: list of all the row data files in the same order. - annotations: list with phenotype and medication. Each of which is character that represents the name of the aforementioned dataset types.

Constructor to generate an object of class metime_analyser. contains slots - list_of_data: For the list of all data matrices. - list_of_col_data: list of all the col data files in the same order. - list_of_row_data: list of all the row data files in the same order. - annotations: list with phenotype and medication. Each of which is character that represents the name of the aforementioned dataset types.

Constructor to generate an object of class metime_analyser. contains slots - list_of_data: For the list of all data matrices. - list_of_col_data: list of all the col data files in the same order. - list_of_row_data: list of all the row data files in the same order. - annotations: list with phenotype and medication. Each of which is character that represents the name of the aforementioned dataset types.

Constructor to generate an object of class metime_analyser. contains slots - list_of_data: For the list of all data matrices. - list_of_col_data: list of all the col data files in the same order. -

list_of_row_data: list of all the row data files in the same order. - annotations: list with phenotype and medication. Each of which is character that represents the name of the aforementioned dataset types.

metime_plotter-class *creating metime_plotter class that converts calculations and meta-data as a plotable object to parse into viz_plotter Contains slots - plot_data: list of Dataframe(s) with plotting data and metadata for visualization. Dataframes is an option only for visNetwork() plots. Need a list of two dataframes: Nodes dataframe and edge dataframe named as .\$node and .\$edge - plot: ggplot(), circos() or visNetwork() object - calc_type: A vector to specify type of calculation - will be used for comp_functions - calc_info: string to define the information about calculation - plot_type: A character vector to define the type of plots that are needed. - style: Character that defines the style of plot i.e. a ggplot(), circos() or visNetwork() plot. Is always a singular input. Cannot have two styles in one object.*

Description

creating metime_plotter class that converts calculations and metadata as a plotable object to parse into viz_plotter Contains slots - plot_data: list of Dataframe(s) with plotting data and metadata for visualization. Dataframes is an option only for visNetwork() plots. Need a list of two dataframes: Nodes dataframe and edge dataframe named as .\$node and .\$edge - plot: ggplot(), circos() or visNetwork() object - calc_type: A vector to specify type of calculation - will be used for comp_functions - calc_info: string to define the information about calculation - plot_type: A character vector to define the type of plots that are needed. - style: Character that defines the style of plot i.e. a ggplot(), circos() or visNetwork() plot. Is always a singular input. Cannot have two styles in one object.

creating metime_plotter class that converts calculations and metadata as a plotable object to parse into viz_plotter Contains slots - plot_data: Dataframe with plotting data and metadata for visualization - plot: ggplot(), circos() or visNetwork() object with predefined aesthetics - calc_type: A vector to specify type of calculation - will be used for comp_functions - calc_info: string to define the information about calculation - plot_type: A character vector to define the type of plots that are needed.

mod_convert_s4_to_s3 *Function to Convert S4 object of class metime_analyser to an S3 object with same architecture*

Description

converter function to be applied onto metime_analyse object to convert into a standard list of S3 type.

Usage

```
mod_convert_s4_to_s3(object)
```

Arguments

object An object of class metime_analyser

Value

An S3 object of the same data as metime_analyser in other words all slots are now converted into nested lists

Examples

```
# convert S4 object to a list
s3_list <- mod_convert_s4_to_s3(object=metime_analyser_object)
```

```
mod_extract_common_samples
```

*Function to get only common samples from the dataframes in
list_of_data*

Description

A method applied on object of class metime_analyse to extract common samples across datasets. Also has an option to split the data according timepoints(mod_split_acc_time()).

Usage

```
mod_extract_common_samples(object, time_splitter = FALSE)
```

Arguments

object An object of class metime_analyser

time_splitter A boolean input: True leads to splitting of the data wrt time, False returns all the dataframes as they are with common rows

Value

list_of_data with common samples across all time points

Examples

```
# extracting common samples across all datasets
new_list_of_data <- mod_common_sample_extractor(object=metime_analyser_object)
```

mod_filter_tp

Functions for selecting time points

Description

a method applied onto class metime_analyser in order to extract timepoints of interest from a dataset

Usage

```
mod_filter_tp(object, timepoints, full, which_data)
```

Arguments

object	An object of class metime_analyser
timepoints	time points to be selected
full	if TRUE subjects are only selected if measured in all selected time points
which_data	Name of the dataset to be used

Value

An object of class metime_analyser with processed data

Examples

```
#example to use this function
object <- mod_filter_tp(object, timepoints=c(0,12,24), full=TRUE, which_data="Name of the dataset")
```

mod_prep_data_for_ggms

Function to prepare and preprocess S4 objects to use it for gaussian graphical models. Also converts S4 to S3

Description

function to be applied onto metime_analyse object to convert into a standard list of S3 type based on the type of GGM analysis to be performed.

Usage

```
mod_prep_data_for_ggms(object, which_type, mlp_or_temp)
```

Arguments

object	An object of class metime_analyser
which_type	two choices either: 1) single - converts S4 to S3 and returns the nested list 2) multi - extracts common samples across the dataframes and returns an S3 nested list
mlp_or_temp	boolean. If true preps data for multibipartite lasso or temporal networks

Value

An S3 object(nested list) with the same architecture as that of class metime_analyser

Examples

```
# prepping data for genenet ggm for single dataset
object <- mod_prep_data_for_ggms(object, which_type="single", mlp_or_temp=FALSE)
```

mod_split_acc_to_time *Function to split data according to time*

Description

Function to split the list of dataframes into a nested list with each dataframe being split into into dataframes of different timepoints

Usage

```
mod_split_acc_to_time(object)
```

Arguments

object	An object of class metime_analyser
--------	------------------------------------

Value

list_of_data with each dataframe being broken into a list of dataframes with respect to the timepoint they belong to

Examples

```
#splitting data according to time
new_data <- mod_split_acc_to_time(object=metime_analyser_object)
```

mod_trans_log	<i>Function to apply log transformation</i>
---------------	---

Description

Function to log transform data

Usage

```
mod_trans_log(object, which_data, base)
```

Arguments

object	An object of class metime_analyser
which_data	Name of the dataset to be used
base	base of log to be used

Value

An object of class metime_analyser with processed data

Examples

```
# example to apply log transformation
object <- mod_logtrans(object, which_data="name of the dataset", base=2)
```

mod_trans_zscore	<i>Function to scale the data</i>
------------------	-----------------------------------

Description

Functions for scaling

Usage

```
mod_trans_zscore(object, which_data)
```

Arguments

object	An object of class metime_analyser
which_data	Name of the dataset to be used

Value

An object of class metime_analyser with processed data

Examples

```
# example to apply scaling
object <- mod_zscore(object, which_data="name of the dataset")
```

```
set_parallel_cores      register parallel backend
```

Description

function to run in order to perform the analysis parallely thereby saving time

Usage

```
set_parallel_cores(n_cores = NULL)
```

Arguments

n_cores A number of specified cores.

Value

set a parallel backend

```
show,metime_analyser-method
Setting new print definition for the metime_analyser object
```

Description

function to see the structure of metime_analyser object

Usage

```
## S4 method for signature 'metime_analyser'
show(object)
```

Arguments

object S4 object of class metime_analyser

Value

structure of the S4 object

Examples

```
structure(object)
```

`show,metime_plotter-method`*Setting new print definition for the metime_plotter object*

Description

function to see the structure of metime_plotter object

Usage

```
## S4 method for signature 'metime_plotter'  
show(object)
```

Arguments

object S4 object of class metime_plotter

Value

structure of the S4 object

Examples

```
structure(object)
```

`structure,metime_plotter-method`*Setting new structure definition for the metime_plotter object*

Description

function to see the structure of metime_plotter object

Usage

```
## S4 method for signature 'metime_plotter'  
structure(object)
```

Arguments

object S4 object of class metime_plotter

Value

structure of the S4 object

Examples

```
structure(object)
```

```
structure
```

Setting new structure definition for the metime_analyser object

Description

function to see the structure of metime_analyser object

Usage

```
structure(object)
```

Arguments

object S4 object of class metime_analyser

Value

structure of the S4 object

Examples

```
structure(object)
```

```
viz_dimensionality_reduction
```

Function to dot plot any kind of dot_plotter including for dimensionality reduction

Description

General function to be implemented on data_list that is obtained after applying a dimensionality reduction method

Usage

```
viz_dimensionality_reduction(  
  data_list,  
  metadata_list,  
  axes_labels,  
  title_metabs,  
  title_samples  
)
```

Arguments

data_list	list obtained after applying calc_dimensionality_reduction() on metime_analyse object
metadata_list	list obtained after applying get_metadata_for_plotting() on metime_analyse object
axes_labels	character vector to specify the labels of the axes in the order x and y.
title_metabs	character to specify the title of the plot of metabolites

Value

a list with both the plots of samples and metabolites. Can be accessed by using ".\$samples" and ".\$metabs"

viz_distribution_plotter

Function for Plotting distributions of phenotypic variables

Description

A method to be applied onto s4 object so as to obtain distributions of various phenotypic variables

Usage

```
viz_distribution_plotter(object, colname, which_data, strats)
```

Arguments

object	An object of class metime_analyser
colname	Name of the variable whose distribution is of interest
which_data	Name of the dataset from which the samples will be extracted

Value

a list with either 1) density plot, mean table acc to timepoint and variable type or 2) bar plot, line plot, and variable type

Examples

```
# extracting distribuion of Age from dataset1
plot <- viz_distribution_plotter(object, colname="Age", which_data="dataset1", strats="additional columns for fac
```

viz_plotter_circos	<i>Setting up standard wrapper for all circos plots for a metime_plotter object.</i>
--------------------	--

Description

plot function for metime_plotter object with different inputs to specialize plots. Used for all calc outputs.

Usage

```
viz_plotter_circos(object, aesthetics, outfile)
```

Arguments

object	S4 object of class metime_plotter
aesthetics	list for aesthetics. eg: list(list(x="colname",y="colname",color="colname", shape="colname"), list(...)) for "dot" plot and "heatmap" plot, for heatmap: list(x="colname", y="colname", fill="colname"). Additionally two other character vectors are allowed namely .vis and .strats for text and for facet wrapping.

viz_plotter_ggplot	<i>Setting up standard wrapper for all ggplot plots for a metime_plotter object.</i>
--------------------	--

Description

plot function for metime_plotter object with different inputs to specialize plots. Used for all calc outputs.

Usage

```
viz_plotter_ggplot(object, aesthetics)
```

Arguments

object	S4 object of class metime_plotter
aesthetics	list for aesthetics. eg: list(list(x="colname",y="colname",color="colname", shape="colname"), list(...)) for "dot" plot and "heatmap" plot, for heatmap: list(x="colname", y="colname", fill="colname"). Additionally two other character vectors are allowed namely .vis and .strats for text and for facet wrapping.

`viz_plotter_visNetwork`

Setting up standard wrapper for network plots from visNetwork for a metime_plotter object.

Description

plot function for metime_plotter object with different inputs to specialize plots. Used for all calc outputs.

Usage

```
viz_plotter_visNetwork(object, title)
```

Arguments

object	S4 object of class metime_plotter
title	character/string that is the title of the graph output

Index

`add_col_normality`, [1](#)
`add_screening_vars`, [2](#)

`calc_conservation_metabolite`, [2](#)
`calc_conservation_metabotype`, [3](#)
`calc_correlation_pairwise`, [4](#)
`calc_dimensionality_reduction`, [4](#)
`calc_distance_pairwise`, [5](#)
`calc_featuresselection_boruta`, [6](#)

`get_append_analyser_object`, [6](#)
`get_files_and_names`, [7](#)
`get_make_analyser_object`, [8](#)
`get_make_plotter_object`, [9](#)
`get_metadata_for_columns`, [9](#)
`get_metadata_for_rows`, [10](#)
`get_palette`, [10](#)
`get_samples_and_timepoints`, [11](#)
`get_text_for_plot`, [11](#)

`metime_analyser-class`, [12](#)
`metime_plotter-class`, [13](#)
`mod_convert_s4_to_s3`, [13](#)
`mod_extract_common_samples`, [14](#)
`mod_filter_tp`, [15](#)
`mod_prep_data_for_ggms`, [15](#)
`mod_split_acc_to_time`, [16](#)
`mod_trans_log`, [17](#)
`mod_trans_zscore`, [17](#)

`set_parallel_cores`, [18](#)
`show,metime_analyser-method`, [18](#)
`show,metime_plotter-method`, [19](#)
`structure`, [20](#)
`structure,metime_plotter-method`, [19](#)

`viz_dimensionality_reduction`, [20](#)
`viz_distribution_plotter`, [21](#)
`viz_plotter_circos`, [22](#)
`viz_plotter_ggplot`, [22](#)
`viz_plotter_visNetwork`, [23](#)