MeTime R Package

July 12, 2022

`add_screening_vars` *Function to add measurements taken at screening time for samples to be added to all timepoints*

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_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_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 wíll be stored
`file_name`	name of the out file

Value

List of conservation index results

`calc_correlation` *Function to calculate correlation*

Description

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

Usage

```
calc_correlation(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="P
#calculate UMAP
pca <- calc_dimensionality_reduction(object=metime_analyser_object, which_data="name/s of the dataset/s", type="U
#calculate tSNE
pca <- calc_dimensionality_reduction(object=metime_analyser_object, which_data="name/s of the dataset/s", type="t
```

calc_metabolite_conservation

Function to calculate metabolite conservation index

Description

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

Usage

```
calc_metabolite_conservation(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_metabotype_conservation

Function to calculate metabotype conservation index

Description

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

Usage

```
calc_metabotype_conservation(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_pairwise_distance

Function to calculate dissimilarity using distance measures

Description

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

Usage

```
calc_pairwise_distance(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")
```

`get_append_metab_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_metab_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
```

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

Description

This function loads all the files from the parent directory. It assumes a certain naming pattern as follows: "datatype_None|col|row_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", m
```

`get_make_metab_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_metab_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_dat
                     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_list, metadata_list, plot_type, aesthetics)
```

Arguments

`data_list`	list of plotable data
`metadata_list`	list of metadata for each plot in data list
`plot_type`	type of the plot you want to build. eg: "box", "dot" etc
`aesthetics`	aesthetics for the plot object

get_metadata_for_plotting

Function to extract metadata of the metabolites and samples for visualization(both dimensionality reduction and GGMs)

Description

A method applied onto s4 object of class "metime_analyser" so as to obtain metadata of the metabolites and samples. Metadata includes their ontology that is the pathway they belong to and also the class or the dataset type. can also add colors for the metabolites for visualization as a separate column. For samples the metadata is basically the columns of interest from the phenotype table that can be used to see sample information in the interactive plot.

Usage

```
get_metadata_for_plotting(
  object,
  which_data,
  metab_groups,
  metab_ids,
  cols_for_vis_samples,
  screening_vars
)
```

Arguments

`object`	S4 object of class metime_analyser
`which_data`	choose the dataset from which metabolites will be extracted for metadata
`metab_groups`	choose the column that has metabolite groups
`metab_ids`	chodse the column that has metabolite names
`cols_for_vis_samples`	character vector representing the name of the columns in phenotype data
`screening_vars`	character vector representing the measurements obtained at baseline to be added to all timepoints for visualization. is set to NULL if nothing is added to it

Value

metadata dataframe with names, groups and class

Examples

```
# metadata_list <- get_metadata_for_plotting(object=metime_analyser_object, which_data="name/s of datasets",
metab_groups="colname/s of the group column in each dataset in order"
metab_ids="colname/s of the metabolite_name column in each dataset in order",
cols_for_vis_samples="colnames of phenotype data for samples", screeing_vars=TRUE/FALSE)
```

`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

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(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` *add aesthetics to plot and so on in this style lol <- aes(x=x, y=y) empty_plot$mappings <- lol creating metime_plotter class that converts calculations and metadata as a plotable object to parse into viz_dot_plotter, viz_heatmap_plotter etc Contains slots - plot_data: Dataframe with plotting data and metadata for visualization - plot_parameters: ggplot() object with predefined aesthetics - aesthetics: list to define aesthetics. Eg: aesthetics=list(x="colname.x", y="colname.y", color="color", shape="shape") - the example above will be predefined in all the methods that creates this object.*

Description

add aesthetics to plot and so on in this style lol <- aes(x=x, y=y) empty_plot$mappings <- lol creating metime_plotter class that converts calculations and metadata as a plotable object to parse into viz_dot_plotter, viz_heatmap_plotter etc Contains slots - plot_data: Dataframe with plotting data and metadata for visualization - plot_parameters: ggplot() object with predefined aesthetics - aesthetics: list to define aesthetics. Eg: aesthetics=list(x="colname.x", y="colname.y", color="color", shape="shape") - the example above will be predefined in all the methods that creates this object.

creating metime_plotter class that converts calculations and metadata as a plotable object to parse into viz_dot_plotter, viz_heatmap_plotter etc Contains slots - plot_data: Dataframe with plotting data and metadata for visualization - plot_parameters: ggplot() object with predefined aesthetics - aesthetics: list to define aesthetics. Eg: aesthetics=list(x="colname.x", y="colname.y", color="color", shape="shape") - the example above will be predefined in all the methods that creates this object.

`mod_common_sample_extractor` *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(modify_split_acc_time()).

Usage

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

Arguments

`object`	An object of class metime_anaylser
`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_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_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_logtrans` *Function to apply log transformation*

Description

Function to log transform data

Usage

```
mod_logtrans(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_prep_data_for_ggms` *Function to prepare and preprocess S4 objects to use it for gaussian gaphical 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 acoording 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_zscore *Function to scale the data*

Description

Functions for scaling

Usage

```
mod_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 parallelly thereby saving time

Usage

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

Arguments

`n_cores`	A number of specified cores.

Value

set a parallel backend

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

Index

add_screening_vars, 1

calc_boruta, 2
calc_correlation, 2
calc_dimensionality_reduction, 3
calc_metabolite_conservation, 4
calc_metabotype_conservation, 4
calc_pairwise_distance, 5

get_append_metab_object, 6
get_files_and_names, 6
get_make_metab_object, 7
get_make_plotter_object, 8
get_metadata_for_plotting, 8
get_palette, 9
get_samples_and_timepoints, 10
get_text, 10

metime_analyser-class, 11
metime_plotter-class, 12
mod_common_sample_extractor, 12
mod_convert_s4_to_s3, 13
mod_filter_tp, 14
mod_logtrans, 14
mod_prep_data_for_ggms, 15
mod_split_acc_to_time, 16
mod_zscore, 16

set_parallel_cores, 17

viz_dimensionality_reduction, 17
viz_distribution_plotter, 18