MeTime R Package

January 23, 2023

`add_clusters_wgcna` *Function to add the clusters obtained from wgcna*

Description

A function to add cluster assiginment to the col_data from WGCNA

Usage

```
add_clusters_wgcna(object, which_data, baseline, ...)
```

Arguments

`object`	An S4 object of class metime_analyser
`which_data`	character to define which dataset is to be used
`baseline`	character to define the timepoint to be used as baseline to predict clusters
`...`	other parameters for cutreeDynamic function such minClusterSize, pamDendroRespect etc

Value

metime_analyser object with updated column info about the clustersize

`add_col_stats` *Function to check normality and add data to col data*

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_stats(object, which_data, type, metab_names, all)
```

Arguments

`object`	An object of class metime_analyser
`which_data`	dataset on which the method is to be applied
`type`	type of test, "shapiro" and "kruskal" are available
`metab_names`	column that has the metabolite names in col_data.
`all`	logical to add all kinds of available stats.

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_distribution_vars_to_rows` *Function to add measurements taken at screening time for samples to be added to all timepoints in row data*

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 in row_data lists

Usage

```
add_distribution_vars_to_rows(
  object,
  screening_vars,
  distribution_vars,
  which_data
)
```

Arguments

object	An object of class metime_analyser
screening_vars	Logical to call add_screening_vars() before updating rows
distribution_vars	A character naming the vars of interest
which_data	dataset to which the information is to be added(only 1 can be used at a time)

Value

object of class metime_analyser with phenotype data added to row data

Examples

```
# adding APOEGrp, PTGENDER, and diag group to all data points and prepping the object for viz_distribution_plotter()
object <- add_distribution_vars_to_rows(object=data, screening_vars=c("APOEGrp", "DXGrp_longi", "PTGENDER"),
distribution_vars=c("Age", "BMI", "ADNI_MEM", "ADNI_LAN", "ADNI_EF", "APOEGrp", "DXGrp_longi", "PTGENDER"), which
```

---

add_function_info *Function to add information of function added to the data*

Description

Function to add information about the method applied to the dataset

Usage

```
add_function_info(object, function_name, params)
```

Arguments

object	S4 object of class metime_analyser
function_name	name of the function used
params	other parameters used wrapped as a list

Value

object of class metime_analyser with the information of method applied

add_metabs_as_covariates *Function to add metabolites as covariates for network construction*

Description

Method applied on metime_analyser object to add other metabolite data to a certain dataset

Usage

```
add_metabs_as_covariates(object, which_data, which_metabs)
```

Arguments

object	A S4 object of class metime_analyser
which_data	Dataset to which the metab data is to be added(please note that this a single character)
which_metabs	list of names of metabs and name of the list represents the dataset from which the metabs are to be acquired. eg: which_metabs=list(nmr_data=c("metab1", "metab2"), lipid_data=c(""))

Value

S4 object with metabs added for GGM to another dataset

add_node_features *Function to add features to visnetwork plot from another plotter object*

Description

Function to add node features to see the nodes in the network that affected differently

Usage

```
add_node_features(
  network_plotter_object,
  guide_object,
  which_type,
  metab_colname
)
```

Arguments

network_plotter_object	plotter object with network information
guide_object	guide from which the colors are to be extracted. Both analyser and plotter objects are allowed
which_type	type of the guide plotter object to be used. Current options are c("regression","conservation")
metab_colname	name of the column in guide plotter object that represents the metabolites

Value

network plotter object with new node colors/features

---

add_phenotypes_as_covariates

Function to add covariates to the dataset of interest for GGMs

---

Description

adds Covariates to data matrices in metime_analyser S4 object

Usage

```
add_phenotypes_as_covariates(
  object,
  which_data,
  covariates,
  class.ind,
  phenotype
)
```

Arguments

object	object of class metime_analyser
which_data	Dataset to which the covariates is to be added
covariates	character vector names of covariates.
class.ind	Logical to convert factor variables into class.ind style or not
phenotype	Logical. If True will extract from phenotype dataset else uses row data

Value

S4 object with covariates added to the dataset

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_colinearity *Function to calculate colinearity*

Description

Function to calculate colinearity in a dataset

Usage

```
calc_colinearity(
  object,
  which_data,
  cols_for_meta,
  show_all,
  name,
  stratifications
)
```

Arguments

object	An S4 object of class metime_analyser
which_data	Dataset to check for colinearity
cols_for_meta	list of character vectors of column names needed in metadata. Id and class name is needed
show_all	logical. True will only filter out colinear data
name	character to define the name of the result
stratifications	List to stratify data into a subset. Usage list(name=value)

Value

plotter object with data for heatmap information

---

calc_conservation_metabolite

Function to calculate metabolite conservation index

Description

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

Usage

```
calc_conservation_metabolite(
  object,
  which_data,
  verbose,
  cols_for_meta,
  stratifications,
  name
)
```

Arguments

object	An object of class metime_analyser
which_data	Name of the dataset to be used
verbose	Information provided on steps being processed
cols_for_meta	A list of a Character vector to define column names that are to be used for plotting purposes
stratifications	list to stratify the data used
name	character vector to define the name of the results generated. length should be equal to which_data

Value

conservation index results that are added to the object

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,
  timepoints,
  verbose,
  cols_for_meta,
  stratifications,
  name
)
```

Arguments

object	An object of class metime_analyser
which_data	Name of the dataset to be used
timepoints	character vector with timepoints of interest
verbose	Information provided on steps being processed
cols_for_meta	Character vector to define column names that are to be used for plotting purposes
stratifications	List to stratify data into a subset. Usage list(name=value)
name	character vector to define the results

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,
  cols_for_meta,
  name,
  stratifications
)
```

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
`cols_for_meta`	list equal to length of which_data defining the columns for metadata
`name`	name of the results should be of length=1
`stratifications`	List to stratify data into a subset. Usage list(name=value)

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,
  cols_for_metabs,
  cols_for_samples,
  stratifications,
  ...
)
```

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"
`cols_for_metabs`	a list of character vectors for getting metadata for columns for plotting purposes
`cols_for_samples`	a character vector to define the columns to extract metadata for plotting purposes
`stratifications`	List to stratify data into a subset. Usage list(name=value)
`...`	additional arguments that can be passed on to prcomp(), M3C::tsne() and umap::umap()

Value

a list with two plotter objects containing the dimensionality reduction information that can be parsed into plotting function 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_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,
  name,
  cols_for_meta,
  stratifications
)
```

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
name	name of the results should be of length=1
cols_for_meta	list equal to length of which_data defining the columns for metadata
stratifications	List to stratify data into a subset. Usage list(name=value)

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 wíll be stored
`file_name`	name of the out file

Value

List of conservation index results

`calc_ggm_genenet_longitudnal` *An automated fucntion to calculate GGM from genenet longitudnal version*

Description

automated funtion that can be applied on metime_analyser object to obtain geneNet network along with threshold used

Usage

```
calc_ggm_genenet_longitudnal(
  object,
  which_data,
  threshold,
  all,
  cols_for_meta,
  covariates,
  stratifications,
  name,
  ...
)
```

Arguments

object	S4 object of cĺass metime_analyser
which_data	a character or a character vector naming the datasets of interest
threshold	type of threshold to be used for extracting significant edge
all	Logical to get all edges without any cutoff.
cols_for_meta	a list of character vectors for getting metadata for columns for plotting purposes
covariates	covariates to be used for this analysis
stratifications	List to stratify data into a subset. Usage list(name=value)
name	character vector for naming the results
...	addtional arguments for genenet network

Value

Network data as a plotter object

---

calc_ggm_genenet *An automated fucntion to calculate GGM from genenet crosssectional version*

Description

automated funtion that can be applied on metime_analyser object to obtain geneNet network along with threshold used

Usage

```
calc_ggm_genenet(
  object,
  which_data,
  threshold,
  all,
  cols_for_meta,
  covariates,
  stratifications,
  name,
  ...
)
```

Arguments

object	S4 object of cĺass metime_analyser
which_data	a character or a character vector naming the datasets of interest
threshold	type of threshold to be used for extracting significant edges. allowed inputs are "li", "FDR", "bonferroni"
all	Logical to extract all edges without any pval correction
cols_for_meta	list of character vector for extracting metadata of metabolites for plotting
covariates	covariates to be used for this analysis
stratifications	List to stratify data into a subset. Usage list(name=value)
name	Name of the result
...	additional arguments for GeneNet

Value

Network data as a plotter object

---

calc_ggm_multibipartite_lasso

An automated fucntion to calculate GGM from multibipartite lasso approach

---

Description

automated funtion that can be applied on s4 object of class metime_analyser to calculate a network using multibipartite lasso

Usage

```
calc_ggm_multibipartite_lasso(
  object,
  which_data,
  alpha,
  nfolds,
  timepoints,
  cols_for_meta
)
```

Arguments

object	S4 object of class metime_analyser
which_data	a character or a character vector naming the datasets of interest
alpha	tuning parameter for lasso + ridge regression in glmnet
nfolds	nfolds for cv.glmnet
timepoints	timepoints of interest that are to be used to build networks(as per timepoints in rows)
cols_for_meta	a list of character vectors of column names to be used for visualization of the networks.

Value

list of plotter objects that can be used for plotting.

---

calc_lm_matrixeqtl *Function to perform matrixEQTL style regression longitudinally*

Description

Function to perform matrixEQTL style regression longitudinally

Usage

```
calc_lm_matrixeqtl(
  object,
  which_data,
  covariates,
  feature_selection,
  regression_phenotype,
  name,
  stratifications,
  cols_for_meta
)
```

Arguments

object	An S4 object of class metime_analyser
which_data	character to define dataset to be used
covariates	covariates to be used
feature_selection	results from feature_selection
regression_phenotype	character to define which phenotype
name	character vector for naming the results
stratifications	List to stratify data into a subset. Usage list(name=value)
cols_for_meta	colnames to be added for meta information. list of character vectors of length which_data

Value

plotter object with a forest plot

---

calc_parafac *Function to perform PARAFAC analysis*

Description

Method to be applied on S4 object of class metime_analyser to perform PARAFAC analysis

Usage

```
calc_parafac(object, which_data, timepoints, nfac = 3, ...)
```

Arguments

object	S4 object of class metime_analyser
which_data	character vector for dataset to be used
timepoints	character vector to define timepoints of interest
nfac	parameter nfac for parafac(). Numeric value to define the number of factors. Default is set to 3
...	Additional arguments to be used for the function parafac()

Value

An object of class PARAFAC. See multiway library for more information

`calc_temporal_ggm` *An automated function to caluclate temporal network with lagged model*

Description

calculates temporal networks for each dataset with a lagged model as used in graphical VAR

Usage

```
calc_temporal_ggm(
  object,
  which_data,
  lag,
  timepoints,
  alpha,
  nfolds,
  cols_for_meta,
  cores
)
```

Arguments

`object`	S4 object of class metab_analyser
`which_data`	dataset or datasets to be used
`lag`	which lagged model to use. 1 means one-lagged model, similary 2,3,..etc
`timepoints`	timepoints of interest that are to be used to build networks(in the order of measurement)
`alpha`	parameter for regression coefficient
`nfolds`	nfolds parameter for glmnet style of regression
`cols_for_meta`	a list of character vectors of column names to be used for visualization of the networks.
`cores`	Number of cores to be used for the process

Value

temporal network data with edgelist and regression values

calc_trajectories_by_mean

Function to get mean trajectories of metabolites and phenotypic traits

Description

function to extract mean trajectories

Usage

```
calc_trajectories_by_mean(object, which_data, columns)
```

Arguments

object	An S4 object of class metime_analyser
which_data	Dataset of interest
columns	Other data that you want to see along with metabolites(column names from row-data)

Value

plot_data table that can be used to make the plotter object without metadata

calc_ttest_metabolites

Function to calculate students t-test between metabolites at different timepoints

Description

Method for S4 object of class metime_analyser for performing t-test

Usage

```
calc_ttest_metabolites(object, which_data, timepoints, split_var, type, paired)
```

Arguments

object	S4 object of class metime_analyser
which_data	dataset or datasets to be used for the analysis
timepoints	timepoints of interest to perform the test on
split_var	split variable for testing such as diagnostic group etc
type	type of ttest to be used either "two.sided", "less", or "greater"
paired	Logical to perform paired t.test or not

Value

plotter object with t-test results

`calc_ttest_samples` *Function to calculate students t-test between samples at different time-points*

Description

Method for S4 object of class metime_analyser for performing t-test

Usage

```
calc_ttest_samples(object, which_data, timepoints, type, paired = TRUE)
```

Arguments

`object`	S4 object of class metime_analyser
`which_data`	dataset or datasets to be used for the analysis
`timepoints`	timepoints of interest to perform the test on
`type`	type of ttest to be used either "two.sided", "less", or "greater"
`paired`	Logical to perform paired t.test or not

Value

plotter object with t-test results

`check_col_normality` *Function to check for col_normality data whether it is added or not.*

Description

function to check whether col_normality data is added to the object or not

Usage

```
check_col_normality(object, which_data)
```

Arguments

`object`	S4 object of class of metime_analyser
`which_data`	dataset/s to check

Value

NULL if it passes all the sanity checks

check_ids_and_classes *Function to check the ids in the data and data format*

Description

sanity check to check for ids and order of the data

Usage

```
check_ids_and_classes(object)
```

Arguments

object	S4 object of class of metime_analyser

Value

NULL if it passes all the sanity checks

check_results *Function to check the format of results if they exist*

Description

sanity check to check for results of the analysis

Usage

```
check_results(object)
```

Arguments

object	S4 object of class of metime_analyser

Value

NULL if it passes all the sanity checks

check_rownames_and_colnames

Function to check the format of rownames and colnames and if they are same or not

Description

sanity check to check for rownames of the data

Usage

```
check_rownames_and_colnames(object)
```

Arguments

object	S4 object of class of metime_analyser

Value

NULL if it passes all the sanity checks

check_scaling_and_transformation

Function to check if the data is already scaled or log transformed

Description

Function to be applied on metime_analyser to check for log transformation and scaling

Usage

```
check_scaling_and_transformation(object, which_data)
```

Arguments

object	An S4 object of metime_anlyser class
which_data	the dataset/s to be checked

Value

NULL but checks if the data is scaled or not

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
```

get_betas_for_multibipartite_lasso *Function to perform multibipartite style regression on a list of matrices*

Description

Performs multibipartite lasso in cv.glmnet style on a list of matrices that have metabolite information from different platforms

Usage

```
get_betas_for_multibipartite_lasso(list_of_mats, alpha, nfolds)
```

Arguments

list_of_mats	a list with matrices and samples ordered similarly
alpha	alpha for cv.glmnet regression. Defines style of penalty.
nfolds	nfolds for cv.glmnet

Value

returns a list with information of the combinations in context

---

get_class_info_from_edges *Function to get information on how many class edges are present*

Description

Function to check how the different edges in a GGM are associated to their respective classes(it could be super-pathway or sub-pathway)

Usage

```
get_class_info_from_edges(calc_networks, metadata, phenotypes)
```

Arguments

calc_networks	list of calculated networks
metadata	metadata of the edges present
phenotypes	character vector to define phenotypes that were used for correcting the data

Value

table with information on different type of edges present

---

get_coldata *Function to extract col data of a dataset*

Description

Function to get coldata

Usage

```
get_coldata(object, which_data)
```

Arguments

object	An object of class S4
which_data	Dataset of interest

Value

col data of the dataset of interest

---

get_environment *Function to get the R environment*

Description

function to print the R environment

Usage

```
get_environment()
```

Value

null

---

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 list(phenotype="Name or index of the files", medication="Name or index of the files")

Value

An object of class metime_analyser

Examples

```
get_files_and_names(path="/path/to/parent/directory",
annotations_index=list(phenotype="Name of phenotype file",
medication="name of phenotype file"))
```

`get_ggm_genenet` *Function to calculate a dynamic GeneNet GGM from a longitudnal data matrix*

Description

calculates GGM on longitudnal data matrix and returns a dataframe with edges, partial correlation and associated p-values

Usage

```
get_ggm_genenet(data, threshold = c("bonferroni", "FDR", "li"), all, ...)
```

Arguments

`data`	data matrix in a longitudnal format
`threshold`	type of multiple hypothesis correction. Available are Bonferoni("bonferroni"), Benjamini-Hochberg("FDR") and independent tests method("li", also see Li et al ....)
`all`	Logical to get all edges without any cutoff.
`...`	additional arguments for ggm.estimate.pcor()

Value

a dataframe with edges, partial correlation and associated p-values

get_li_thresh *Function to calculate multiple tests using method described by li*

Description

li test to check for colinearlity and use it for feature selection

Usage

```
get_li_thresh(object, which_data, verbose)
```

Arguments

object	an S4 object of class metime_analyser
which_data	dataset to be used for testing
verbose	Logical to print out the number of independent tests

Value

li threshold value

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 = list(),
  name = NULL,
  results = list()
)
```

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
results	list set to empty but can add any existing results

Value

An object of class metime_analyser

---

get_make_results *Function to make results list for metime_analyser object*

Description

function to generate results for metime_analyser object

Usage

```
get_make_results(object, data, metadata, calc_type, calc_info, name)
```

Arguments

object	An S4 object of class metime_analyser
data	list of dataframes of plotable data obtained from any calc function
metadata	list of dataframes with the metadata for the plot table mentioned above. To obtain these see get_metadata_for_rows() and get_metadata_for_columns()
calc_type	A character vector to specify type of calculation - will be used for comp_ functions For networks the accepted notations are "genenet_ggm", "multibipartite_ggm", and "temporal_network" sjould be the same length as the list of data provided
calc_info	A string to define the information about calculation, should be the same length as the list data provided
name	Name of the result

Value

object with results of the calculation updated

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

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 id should always be first in order
index_of_names	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)
```

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

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_parameters_of_results` *Get parameters of the results*

Description

Function to get parameters and functions applied to obtain results

Usage

```
get_parameters_of_results(object, results_index)
```

Arguments

`object`	An S4 object of class metime_analyser
`results_index`	name or index to get to the results of interest

Value

a dataframes with functions(rownames) and parameters(colnames)

get_rowdata *Function to extract row data of a dataset*

Description

Function to get rowdata

Usage

```
get_rowdata(object, which_data)
```

Arguments

object	An object of class S4
which_data	Dataset of interest

Value

row data of the dataset of interest

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.

mod_code_metab_names *Function to convert metabolite names to IDs*

Description

Function to convert metabolite names to IDs

Usage

```
mod_code_metab_names(object, which_data)
```

Arguments

object	An S4 object of class metime_analyser
which_data	character vector to define the datasets to use

Value

A list with S4 object and list of mapping tables, the object can be used for GGMs

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_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_filter_timepoints` *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_timepoints(object, timepoints, complete, which_data)
```

Arguments

`object`	An object of class metime_analyser
`timepoints`	time points to be selected.
`complete`	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("t0","t12","t24"), full=TRUE, which_data="Name of the dataset")
```

---

mod_merge_data *Function to merge two sets of data for any analysis*

Description

A function to merge two or more datasets and use it for analysis

Usage

```
mod_merge_data(object, which_data, name)
```

Arguments

object	An S4 object of class metime_analyser
which_data	Datasets to be merged. Only two or more are allowed
name	character vector to define the name of the new dataset

Value

A new S4 object of class metime_analyser with the new merged dataset appended to it

---

mod_merge_metime_analysers *Function to merge one or more metime_analyser objects*

Description

function to merge multiple metime_analyser objects

Usage

```
mod_merge_metime_analysers(list_of_objects, annotations_index)
```

Arguments

list_of_objects	list of metime analyser objects that are to be merged
annotations_index	new list with annotations_index. Can also set to be NULL.

Value

A merged metime_analyser object

`mod_merge_results` *Function to combine plotter objects*

Description

Function to combine plotter objects based on similar calc type

Usage

```
mod_merge_results(object, results_indices, groups)
```

Arguments

`object`	An S4 object of class metime_analyser
`results_indices`	one or more indices of results to merge data
`groups`	character vector to define the groups that are involved

Value

object with add merged letters

`mod_remove_duplicates` *Function to remove duplicates*

Description

Function to remove duplicates from the analyser object

Usage

```
mod_remove_duplicates(object)
```

Arguments

`object`	An S4 object of class metime_analyser

Value

object after removing duplicated data

mod_remove_nas *Function to remove NA's from data matrices*

Description

A method applied on S4 object to remove NA's and change data accordingly

Usage

```
mod_remove_nas(object, which_data)
```

Arguments

object	S4 object of class metime_analyser
which_data	dataset/s for which the method is to be applied

Value

S4 object with NA's removed and data manipulated accordingly

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_stratify_analyser *Function to stratify data in the metime analyser object*

Description

Function to stratify the data of interest into different objects that can be used to perform calculations according the said stratification variable

Usage

```
mod_stratify_analyser(object, which_data, variable)
```

Arguments

object	S4 object of class metime_analyser
which_data	Dataset/datasets to be used for stratification
variable	Phenotype based on which the stratification would be performed

Value

list of metime_analyser objects which are stratified based on the variable chosen

mod_trans_eigendata *Function to add the clusters obtained from wgcna*

Description

A function to add cluster assiginment to the col_data from WGCNA

Usage

```
mod_trans_eigendata(object, which_data, append, clusters, ...)
```

Arguments

object	An S4 object of class metime_analyser
which_data	character to define which dataset is to be used
append	logical if set to true adds the new data to the object used else creates new object
clusters	logical if set to true will add already existing cluster info otherwise creates new
...	arguments for add_clusters_wgcna

Value

metime_analyser object with new dataset with eigendata of the metabolites

mod_trans_log *Function to apply log transformation*

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 *Function to scale the data*

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")
```

`plot` *Setting a plotting method for the metime_analyser class*

Description

Function to plot results of a certain calculation

Usage

```
plot(object, results_index, interactive, ...)
```

Arguments

`object`	An S4 object of class metime_analyser
`results_index`	Index/name of the results to be plotted
`interactive`	logical. Set TRUE for interactive plot
`...`	other parameters to pass color, fill, strat, viz(character vector with colnames for interactive)

Value

plots for a certain set of results

`save_analyser_object` *Function to extract analyser object data into a csv*

Description

extracts information from analyser object and saves it as a csv

Usage

```
save_analyser_object(object, which_data, type)
```

Arguments

`object`	An object of class metime_plotter
`which_data`	Character to specify the dataset
`type`	which type of output file. Can be "csv", "tsv" and "xlsx"

Value

saves the data in the working directory as a csv and returns nothing

Examples

```
see examples here
save_analyser_object(object, which_data="dataset")
```

`save_results` *Function to extract results into different types of files*

Description

extracts results from analyser object and saves it

Usage

```
save_results(object, results_index, type)
```

Arguments

`object`	An object of class metime_analyser
`results_index`	character or numeric to define the results of interest
`type`	character to define outfile type that is "csv", "xlsx" or "tsv"

Value

saves the data into a csv and returns nothing

Examples

```
see examples here and maintain the order
If type is xlsx one out file is enough and the network data will be stored in different sheets
Network : save_results(object, results_index, out=c("edge", "node", "meta"), type="csv")
Others : save_results(object, results_index, out=c("outfile1", ...), type="tsv")
```

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)
```

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)
```

validity *Validity function to check if the object is valid or not*

Description

Function to check the validity of metime_analyser

Usage

```
validity(object)
```

Arguments

object	An S4 object of class metime_analyser

Value

logical suggesting if the object is intact or not

Examples

```
validObject(object)
```

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, phenotype)
```

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
strats	Character vector with colnames that are to be used for stratification
phenotype	Logical. If true data will be collected from phenotype_data matrix else from row data

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