

# MeTime R Package

August 18, 2022

---

`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)
```

## 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, "shapiro" and "kruskal" are available
<code>metab_names</code>	column that has the metabolite names in col_data.
<code>all</code>	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
which_data	dataset to which the information is to be added
vars	A character naming the vars of interest

### 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_data="row_data")
```

---

```
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\_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	<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"
...	additional arguments that can be passed on to prcomp(), M3C::tsne() and umap::umap()

**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



---

 calc\_ggm\_genenet\_crosssectional

*An automated function to calculate GGM from genenet crosssectional version*

---

### Description

automated function that can be applied on metime\_analyser object to obtain geneNet network along with threshold used

### Usage

```
calc_ggm_genenet_crosssectional(object, which_data, threshold, timepoint)
```

### Arguments

object	S4 object of class 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"
timepoints	timepoints of interest that are to be used to build networks(as per timepoints in rows)

### Value

Network data with edgelist, partial correlation values and associated p-values and corrected p-values

---

calc\_ggm\_genenet\_longitudnal

*An automated function to calculate GGM from genenet longitudinal version*

---

### Description

automated function 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,
  timepoints,
  all,
  ...
)
```

**Arguments**

object	S4 object of class 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
timepoints	timepoints of interest that are to be used to build networks(as per timepoints in rows)
all	Logical to get all edges without any cutoff.
...	additional arguments for genenet network

**Value**

Network data with edgelist, partial correlation values and associated p-values and corrected p-values

---

calc\_ggm\_multibipartite\_lasso

*An automated function 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)
```

**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)

**Value**

Network data with edges and their respective betas

---

calc_temporal_ggm	<i>An automated function to calculate temporal network with lagged model</i>
-------------------	--

---

**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)
```

**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, similiary 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

**Value**

temporal network data with edgelist and regression values

---

calc_ttest	<i>Function to calculate students t-test</i>
------------	--

---

**Description**

Method for S4 object of class metime\_analyser for performing t-test

**Usage**

```
calc_ttest(object, which_data, timepoints, split_var)
```

**Arguments**

object	S4 object of class metime_analyser
which_data	dataset or datasets to be used for the analysis
timepoints	two timepoints of interest to perform the test on
split_var	split variable for testing such as diagnostic group etc

**Value**

t-test result as a list or a list of 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\_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

---

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_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**

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

**Value**

returns a list with information of the combinations in context

---

```
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\_Nonelcollrow\_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_ggm_genenet	<i>Function to calculate a dynamic GeneNet GGM from a longitudinal data matrix</i>
-----------------	--

---

**Description**

calculates GGM on longitudinal 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 longitudinal 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_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

data	dataframe of plotable data obtained from any calc object
metadata	dataframe 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 to specify type of calculation - will be used for comp_ functions For networks the accepted notations are "genenet_ggm", "multibipartite_ggm", and "temporal_network"
calc_info	A string to define the information about calculation
plot_type	type of the plot you want to build. eg: "box", "dot" etc. Its a character vector
style	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

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
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_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**

<code>object</code>	An object of class <code>metime_analyser</code>
<code>which_data</code>	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**

<code>data</code>	a dataframe with plotting data along with other variables for visualization
<code>colnames</code>	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	<i>Functions for selecting time points</i>
---------------	--

---

**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_merge_metime_analysers	<i>Function to merge one or more metime_analyser objects</i>
----------------------------	--

---

**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_remove_nas	<i>Function to remove NA's from data matrices</i>
----------------	---

---

**Description**

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

**Usage**

```
mod_remove_nas(object)
```

**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	<i>Function to split data according to time</i>
-----------------------	---

---

**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")
```

---

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)
```

**Arguments**

object	An object of class metime_plotter
which_data	Character to specify the dataset

**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\_plot\_from\_plotter  
                          *Function to save interactive plots*

---

**Description**

extracts plot from plotter object and saves it as a widget

**Usage**

```
save_plot_from_plotter(object, out)
```

**Arguments**

object	An object of class metime_plotter
out	Character to specify path of the output file to save the widget in

**Value**

saves the plot and returns nothing

**Examples**

```
save_plot_from_plotter(object)
```

---

save_plotter_object	<i>Function to extract plot data into a csv</i>
---------------------	---

---

**Description**

extracts information from plotter object and saves it as a csv

**Usage**

```
save_plotter_object(object, out)
```

**Arguments**

object	An object of class metime_plotter
out	Character to specify path of the output file or character vector in case of visNetwork

**Value**

saves the data into a csv and returns nothing

**Examples**

```
see examples here
Network : save_plotter_object(object, out=c("edge.csv", "node.csv", "meta.csv"))
Others : save_plotter_object(object, out="outfile.csv")
```

---

set_parallel_cores	<i>register parallel backend</i>
--------------------	----------------------------------

---

**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	<i>Setting new structure definition for the metime_analyser object</i>
-----------	--

---

**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_distribution_plotter	<i>Function for Plotting distributions of phenotypic variables</i>
--------------------------	--

---

**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 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.

**Value**

metime\_plotter object with updated plot

---

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\\_stats](#), [1](#)  
[add\\_distribution\\_vars\\_to\\_rows](#), [2](#)  
[add\\_metabs\\_as\\_covariates](#), [2](#)  
[add\\_phenotypes\\_as\\_covariates](#), [3](#)  
[add\\_screening\\_vars](#), [4](#)  
  
[calc\\_conservation\\_metabolite](#), [4](#)  
[calc\\_conservation\\_metabotype](#), [5](#)  
[calc\\_correlation\\_pairwise](#), [6](#)  
[calc\\_dimensionality\\_reduction](#), [6](#)  
[calc\\_distance\\_pairwise](#), [7](#)  
[calc\\_featuresselection\\_boruta](#), [8](#)  
[calc\\_ggm\\_genenet\\_crosssectional](#), [9](#)  
[calc\\_ggm\\_genenet\\_longitudinal](#), [9](#)  
[calc\\_ggm\\_multibipartite\\_lasso](#), [10](#)  
[calc\\_temporal\\_ggm](#), [11](#)  
[calc\\_ttest](#), [11](#)  
[check\\_col\\_normality](#), [12](#)  
[check\\_ids\\_and\\_classes](#), [12](#)  
[check\\_rownames\\_and\\_colnames](#), [13](#)  
  
[get\\_append\\_analyser\\_object](#), [13](#)  
[get\\_betas\\_for\\_multibipartite\\_lasso](#), [14](#)  
[get\\_files\\_and\\_names](#), [14](#)  
[get\\_ggm\\_genenet](#), [15](#)  
[get\\_make\\_analyser\\_object](#), [16](#)  
[get\\_make\\_plotter\\_object](#), [17](#)  
[get\\_metadata\\_for\\_columns](#), [17](#)  
[get\\_metadata\\_for\\_rows](#), [18](#)  
[get\\_palette](#), [18](#)  
[get\\_samples\\_and\\_timepoints](#), [19](#)  
[get\\_text\\_for\\_plot](#), [19](#)  
  
[metime\\_analyser-class](#), [20](#)  
[metime\\_plotter-class](#), [21](#)  
[mod\\_convert\\_s4\\_to\\_s3](#), [21](#)  
[mod\\_extract\\_common\\_samples](#), [22](#)  
[mod\\_filter\\_tp](#), [23](#)  
[mod\\_merge\\_metime\\_analysers](#), [23](#)  
[mod\\_remove\\_nas](#), [24](#)  
  
[mod\\_split\\_acc\\_to\\_time](#), [24](#)  
[mod\\_trans\\_log](#), [25](#)  
[mod\\_trans\\_zscore](#), [25](#)  
  
[save\\_analyser\\_object](#), [26](#)  
[save\\_plot\\_from\\_plotter](#), [26](#)  
[save\\_plotter\\_object](#), [27](#)  
[set\\_parallel\\_cores](#), [27](#)  
[show,metime\\_analyser-method](#), [28](#)  
[show,metime\\_plotter-method](#), [28](#)  
[structure](#), [30](#)  
[structure,metime\\_plotter-method](#), [29](#)  
  
[viz\\_distribution\\_plotter](#), [30](#)  
[viz\\_plotter\\_circos](#), [31](#)  
[viz\\_plotter\\_ggplot](#), [31](#)  
[viz\\_plotter\\_visNetwork](#), [32](#)