

# MeTime

July 7, 2022

---

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 "metab\_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 metab_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=metab_analyser_object, vars=c("APOEGrp", "PTGENDER"))
```

`calc_boruta` *Function to calculate dependent variables*

### Description

An S4 method to be applied on the metab\_analyser object so as to calculate dependent variables

### Usage

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

### Arguments

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

### Value

List of conservation index results

`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 metab\_analyse in order to obtain information after dimensionality reduction on a dataset/s

### Usage

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

### Arguments

<code>object</code>	An object of class metab_analyse
<code>which_data</code>	a character vector - Names of the dataset from which the samples will be extracted
<code>type</code>	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=metab_analyser_object, which_data="name/s of the dataset/s", type="PCA")  
#calculate UMAP  
pca <- calc_dimensionality_reduction(object=metab_analyser_object, which_data="name/s of the dataset/s", type="UMAP")  
#calculate tSNE  
pca <- calc_dimensionality_reduction(object=metab_analyser_object, which_data="name/s of the dataset/s", type="tSNE")
```

---

**calc\_metabolite\_conservation**

*Function to calculate metabolite conservation index*

---

**Description**

Method applied on the object metab\_analyser to calculate the metabotype conservation index

**Usage**

```
calc_metabolite_conservation(object, which_data, verbose)
```

**Arguments**

object	An object of class metab_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=metab_analyser_object, which_data="Name of the dataset")
```

**calc\_metabotype\_conservation***Function to calculate metabotype conservation index***Description**

Method applied on the object metab\_analyser to calculate the metabotype conservation index

**Usage**

```
calc_metabotype_conservation(object, which_data, verbose)
```

**Arguments**

object	An object of class metab_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=metab_analyser_object, which_data="Name of the dataset")
```

**get\_append\_metab\_object***This function appends an object for MetabAnalyze with a new dataset.***Description**

function to apply on metab\_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 MetabAnalyze object
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 metab\_analyser

**Examples**

```
# append data frames into the metab_analyser object
appended_object <- get_append_metab_object(object=metab_analyser_object, data=data, row_data=row_data, col_data=col_
```

get\_files\_and\_names     *Function to pack all the data into a single object of class "metab\_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 metab\_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 "metab\_analyser"*

## Description

This function creates an object for MetabAnalyze from a dataset.

## Usage

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

## Arguments

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

## Value

An object of class `metab_analyser`

## Examples

```
# new_metab_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_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 "metab\_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.

`get_palette`

7

### Usage

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

### Arguments

<code>object</code>	S4 object of class <code>metab_analyse</code>
<code>which_data</code>	choose the dataset from which metabolites will be extracted for metadata
<code>metab_groups</code>	choose the column that has metabolite groups
<code>metab_ids</code>	choose the column that has metabolite names
<code>cols_for_vis_samples</code>	character vector representing the name of the columns in phenotype data
<code>screening_vars</code>	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=metab_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", screening_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

<code>n</code>	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 "metab\_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 metab_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=metab_analyser_object, which_data="Name of dataset of interest")
```

---

get_text	<i>Function to Obtain textual information for visualization in interactive plots</i>
----------	--

---

**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"))
```

---

metab_analyser-class	<i>Constructor to generate an object of class metab_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.</i>
----------------------	---

---

**Description**

Constructor to generate an object of class metab\_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 metab\_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 `metab_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 `metab_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 `metab_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_common_sample_extractor`

*Function to get only common samples from the dataframes in list\_of\_data*

## Description

A method applied on object of class `metab_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

<code>object</code>	An object of class <code>metab_anaylser</code>
<code>time_splitter</code>	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=metab_analyser_object)
```

---

mod\_convert\_s4\_to\_s3    *Function to Convert S4 object of class metab\_analyser to an S3 object with same architecture*

---

## Description

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

## Usage

```
mod_convert_s4_to_s3(object)
```

## Arguments

object            An object of class metab\_analyser

## Value

An S3 object of the same data as metab\_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=metab_analyser_object)
```

---

mod\_filter\_tp            *Functions for selecting time points*

---

## Description

a method applied onto class metab\_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 metab\_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 metab\_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 metab_analyser
which_data	Name of the dataset to be used
base	base of log to be used

**Value**

An object of class metab\_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 graphical models. Also converts S4 to S3*

**Description**

function to be applied onto metab\_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 metab_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 metab\_analyser

**Examples**

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

**mod\_split\_acc\_to\_time** *Function to split data according to time***Description**

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

**Usage**

```
mod_split_acc_to_time(object)
```

**Arguments**

object	An object of class metab_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=metab_analyser_object)
```

`mod_zscore`*Function to scale the data***Description**

Functions for scaling

**Usage**

```
mod_zscore(object, which_data)
```

**Arguments**

<code>object</code>	An object of class <code>metab_analyser</code>
<code>which_data</code>	Name of the dataset to be used

**Value**

An object of class `metab_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 metab\_analyse object  
metadata\_list    list obtained after applying get\_metadata\_for\_plotting() on metab\_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 metab_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_dimensionality_reduction, 2
calc_metabolite_conservation, 3
calc_metabotype_conservation, 4
get_append_metab_object, 4
get_files_and_names, 5
get_make_metab_object, 6
get_metadata_for_plotting, 6
get_palette, 7
get_samples_and_timepoints, 8
get_text, 9
metab_analyser-class, 9
mod_common_sample_extractor, 10
mod_convert_s4_to_s3, 11
mod_filter_tp, 11
mod_logtrans, 12
mod_prep_data_for_ggms, 13
mod_split_acc_to_time, 13
mod_zscore, 14
set_parallel_cores, 14
viz_dimensionality_reduction, 15
viz_distribution_plotter, 16
```