

# Package ‘MetAlyzer’

October 12, 2022

**Type** Package

**Title** Read and Analyze 'MetIDQ&trade;' Software Output Files

**Version** 0.1.0

**Depends** R (>= 4.0.0)

**Imports** openxlsx, dplyr, tidyr, tibble, agricolae, methods, rlang

**Description** The 'MetAlyzer' S4 object provides methods to read and reformat metabolomics data for convenient data handling, statistics and downstream analysis. The resulting format corresponds to input data of the Shiny app 'MetaboExtract' (<<https://www.metaboextract.shiny.dkfz.de/MetaboExtract/>>).

**License** GPL-3

**Encoding** UTF-8

**Language** en-US

**RoxygenNote** 7.1.2

**Suggests** rmarkdown, knitr, ggplot2

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Nils Mechtel [aut, cre] (<<https://orcid.org/0000-0002-1278-7125>>),  
Carolin Andresen [aut] (<<https://orcid.org/0000-0002-8960-7719>>),  
Daniel Huebschmann [aut] (<<https://orcid.org/0000-0002-6041-7049>>)

**Maintainer** Nils Mechtel <[nils.mech@gmail.com](mailto:nils.mech@gmail.com)>

**Repository** CRAN

**Date/Publication** 2022-02-01 09:10:08 UTC

## R topics documented:

createPlottingData . . . . .	2
filterMetabolites . . . . .	3
filterMetaData . . . . .	4
imputePlottingData . . . . .	5
metabolites . . . . .	6

metaData . . . . .	7
MetAlyzer-class . . . . .	8
MetAlyzerDataset . . . . .	9
performANOVA . . . . .	9
plottingData . . . . .	10
quantStatus . . . . .	11
rawData . . . . .	12
renameMetaData . . . . .	12
resetMetabolites . . . . .	13
resetMetaData . . . . .	14
setPlottingData . . . . .	15
show, MetAlyzer-method . . . . .	15
summariseQuantData . . . . .	16
transformPlottingData . . . . .	17
updateMetaData . . . . .	17

<b>Index</b>	<b>19</b>
--------------	-----------

---

createPlottingData	<i>Create plotting data</i>
--------------------	-----------------------------

---

## Description

This method reshapes `raw_data`, `quant_status` and `meta_data` and combines them together with basic statistics in a tibble data frame for plotting with `ggplot2`. `plotting_data` is grouped by metabolites as well as the selection of additional variables. Statistics are then calculated for each group.

## Usage

```
createPlottingData(
  object,
  ...,
  ungrouped = NULL,
  ts = c(0.1, 0.2, 0.3),
  valid_vec = c("Valid", "LOQ"),
  t = 0.5
)

## S4 method for signature 'MetAlyzer'
createPlottingData(
  object,
  ...,
  ungrouped = NULL,
  ts = c(0.1, 0.2, 0.3),
  valid_vec = c("Valid", "LOQ"),
  t = 0.5
)
```

**Arguments**

object	MetAlyzer object
...	A selection of columns from meta_data to add to reshaped data frame
ungrouped	A column from meta_data to add to reshaped data frame that will not be used as grouping variables
ts	A numeric vector of thresholds between 0 and 1 for CV categorization
valid_vec	A character vector containing each quantification status that is considered to be a valid measurement
t	A numeric threshold to determine valid measurements

**Value**

An updated MetAlyzer object

**Methods (by class)**

- MetAlyzer: Create plotting data

**Examples**

```
fpath <- system.file("extdata", "example_data.xlsx", package = "MetAlyzer")
obj <- MetAlyzerDataset(file_path = fpath)

obj <- createPlottingData(obj, Tissue, Group,
  ungrouped = NULL,
  ts = c(0.1, 0.2, 0.3),
  valid_vec = c("Valid", "LOQ"), t = 0.5)
```

---

filterMetabolites      *Filter metabolites*

---

**Description**

This method filters out certain classes or metabolites of the metabolites vector.

**Usage**

```
filterMetabolites(
  object,
  class_name = "Metabolism Indicators",
  metabo_vec = NULL
)

## S4 method for signature 'MetAlyzer'
```

```
filterMetabolites(
  object,
  class_name = "Metabolism Indicators",
  metabo_vec = NULL
)
```

### Arguments

object	MetAlyzer object
class_name	A character value defining the class to be removed
metabo_vec	A character vector defining metabolites to be removed

### Details

Note: If both "metabo\_vec" and "class\_name" arguments are used "metabo\_vec" overwrites the "class\_name" argument!

### Value

An updated MetAlyzer object

### Methods (by class)

- MetAlyzer: Filter metabolites

### Examples

```
fpath <- system.file("extdata", "example_data.xlsx", package = "MetAlyzer")
obj <- MetAlyzerDataset(file_path = fpath)

obj <- filterMetabolites(obj, class_name = "Metabolism Indicators")
# or
obj <- filterMetabolites(obj, metabo_vec = c("C0", "C2", "C3"))
```

---

filterMetaData	<i>Filter meta data</i>
----------------	-------------------------

---

### Description

This function updates the "Filter" column in meta\_data to filter out samples.

### Usage

```
filterMetaData(object, column, keep = NULL, remove = NULL)

## S4 method for signature 'MetAlyzer'
filterMetaData(object, column, keep = NULL, remove = NULL)
```

**Arguments**

object	MetAlyzer object
column	A column of meta_data for filtering
keep	A vector defining which entries to keep from meta_data
remove	A vector defining which entries to remove meta_data

**Details**

If both "keep" and "remove" arguments are used "keep" overwrites the "remove" argument.

**Value**

An updated MetAlyzer object

**Methods (by class)**

- MetAlyzer: Filter meta data

**Examples**

```
fpath <- system.file("extdata", "example_data.xlsx", package = "MetAlyzer")
obj <- MetAlyzerDataset(file_path = fpath)

obj <- filterMetaData(obj, column = Group, keep = 1:6)
# or
obj <- filterMetaData(obj, column = Group, remove = 7)
```

---

imputePlottingData     *Impute plotting data*

---

**Description**

This method imputes zero concentration values (Concentration) with the minimal positive value multiplied by *i*. If all values are zero or NA, they are set to NA. The imputed values are added to plotting\_data in an extra column imp\_Conc.

**Usage**

```
imputePlottingData(object, ..., i = 0.2, imputeNA = FALSE)

## S4 method for signature 'MetAlyzer'
imputePlottingData(object, ..., i = 0.2, imputeNA = FALSE)
```

**Arguments**

object	MetAlyzer object
...	Variables to group by
i	A numeric value below 1
imputeNA	Logical value whether to impute NA values

**Value**

An updated MetAlyzer object

**Methods (by class)**

- MetAlyzer: Impute plotting data

**Examples**

```
# To see an example, please check out the vignette.
```

---

metabolites

*Get metabolites*

---

**Description**

This method returns the filtered metabolites vector.

**Usage**

```
metabolites(object)
```

```
## S4 method for signature 'MetAlyzer'  
metabolites(object)
```

**Arguments**

object	MetAlyzer object
--------	------------------

**Value**

The metabolites vector

**Methods (by class)**

- MetAlyzer: Get metabolites

## Examples

```
fpath <- system.file("extdata", "example_data.xlsx", package = "MetAlyzer")
obj <- MetAlyzerDataset(file_path = fpath)

metabolites(obj)
```

---

metaData

*Get meta data*

---

## Description

This method returns the meta\_data data frame with filtered samples (rows) and all columns.

## Usage

```
metaData(object)

## S4 method for signature 'MetAlyzer'
metaData(object)
```

## Arguments

object            MetAlyzer object

## Value

The meta\_data data frame

## Methods (by class)

- MetAlyzer: Get meta data

## Examples

```
fpath <- system.file("extdata", "example_data.xlsx", package = "MetAlyzer")
obj <- MetAlyzerDataset(file_path = fpath)

meta_data <- metaData(obj)
head(meta_data)
```

---

MetAlyzer-class      *A S4 class to read and analyze 'MetIDQ' output*

---

## Description

A S4 class to read and analyze 'MetIDQ' output

## Slots

`file_path` A length-one character vector giving the file path

`sheet` A length-one numeric vector giving the sheet index

`metabolites` A character vector with all 630 measured metabolites and optional 234 additional metabolism indicators

`raw_data` A data frame containing all raw measurements; dimension: # samples x # metabolites

`quant_status` A data frame containing the quantification status of all measurements; dimension: # samples x # metabolites

`meta_data` A data frame containing any meta data; dimension: # samples x # meta variables

`plotting_data` A tibble data frame containing reshaped information of `raw_data`, `quant_status` and `meta_data` for plotting with `ggplot2`

`.full_sheet` A matrix containing the un-sliced Excel sheet

`.data_ranges` A length-six numeric list with rows and columns information for slicing

`.orig_metabolites` A character vector storing the unfiltered metabolites vector

## Examples

```
fpath <- system.file("extdata", "example_data.xlsx", package = "MetAlyzer")
obj <- MetAlyzerDataset(file_path = fpath)
obj <- filterMetabolites(obj)
show(obj)
summariseQuantData(obj)

obj <- renameMetaData(obj, Method = Group)
obj <- filterMetaData(obj, column = Method, keep = 1:6)

obj <- createPlottingData(obj, Method, Tissue)
obj <- imputePlottingData(obj, Method, Metabolite)
obj <- transformPlottingData(obj)
obj <- performANOVA(obj, categorical = Method)
```



---

MetAlyzerDataset	<i>Open file and read data</i>
------------------	--------------------------------

---

### Description

This function creates a 'MetAlyzer' object, opens the given 'MetIDQ' output Excel sheet and extracts metabolites, raw data, quantification status and meta data.

### Usage

```
MetAlyzerDataset(file_path, sheet = 1)
```

### Arguments

file_path	file path
sheet	sheet index

### Value

An MetAlyzer object

### Examples

```
fpath <- system.file("extdata", "example_data.xlsx", package = "MetAlyzer")  
obj <- MetAlyzerDataset(file_path = fpath)
```

---

performANOVA	<i>ANOVA</i>
--------------	--------------

---

### Description

This method performs a one-way ANOVA on the grouped plotting\_data (the categorical variable is removed from grouping first). For this, the column valid\_replicates must have at least one entry that is TRUE in each group. Otherwise, a vector of NA is returned. A Tukey post-hoc test is then used to determine group names, starting with "A" followed by further letters. These group names are added to plotting\_data in the column ANOVA\_group. Thereby, metabolites can be identified which are significantly higher in one or more of the categorical variable compared to all other for each metabolite.

### Usage

```
performANOVA(object, categorical)  
  
## S4 method for signature 'MetAlyzer'  
performANOVA(object, categorical)
```

**Arguments**

object	MetAlyzer object
categorical	A column defining the categorical variable

**Value**

An updated MetAlyzer object

**Methods (by class)**

- MetAlyzer: ANOVA

**Examples**

```
# To see an example, please check out the vignette.
```

---

plottingData	<i>Get plotting data</i>
--------------	--------------------------

---

**Description**

This method returns the plotting\_data tibble data frame.

**Usage**

```
plottingData(object)  
  
## S4 method for signature 'MetAlyzer'  
plottingData(object)
```

**Arguments**

object	MetAlyzer object
--------	------------------

**Value**

The plotting\_data data frame

**Methods (by class)**

- MetAlyzer: Get plotting data

## Examples

```
fpath <- system.file("extdata", "example_data.xlsx", package = "MetAlyzer")
obj <- MetAlyzerDataset(file_path = fpath)
obj <- createPlottingData(obj, Tissue, Group)

plottingData(obj)
```

---

quantStatus	<i>Get quantification status</i>
-------------	----------------------------------

---

## Description

This method returns the `quant_status` data frame with filtered samples (rows) and metabolites (columns).

## Usage

```
quantStatus(object)

## S4 method for signature 'MetAlyzer'
quantStatus(object)
```

## Arguments

object            MetAlyzer object

## Value

The `quant_status` data frame

## Methods (by class)

- MetAlyzer: Get quantification status

## Examples

```
fpath <- system.file("extdata", "example_data.xlsx", package = "MetAlyzer")
obj <- MetAlyzerDataset(file_path = fpath)

quant_status <- quantStatus(obj)
head(quant_status, c(5, 5))
```

rawData *Get raw data*

---

### Description

This method returns the raw\_data data frame with filtered samples (rows) and metabolites (columns).

### Usage

```
rawData(object)

## S4 method for signature 'MetAlyzer'
rawData(object)
```

### Arguments

object MetAlyzer object

### Value

The raw\_data data frame

### Methods (by class)

- MetAlyzer: Get raw data

### Examples

```
fpath <- system.file("extdata", "example_data.xlsx", package = "MetAlyzer")
obj <- MetAlyzerDataset(file_path = fpath)

raw_data <- rawData(obj)
head(raw_data, c(5, 5))
```

---

renameMetaData *Rename meta data*

---

### Description

This method renames a column of meta\_data using rename 'dplyr'.

### Usage

```
renameMetaData(object, ...)

## S4 method for signature 'MetAlyzer'
renameMetaData(object, ...)
```

**Arguments**

object            MetAlyzer object  
...                Use new\_name = old\_name to rename selected variables

**Value**

An updated MetAlyzer object

**Methods (by class)**

- MetAlyzer: Rename meta data

**Examples**

```
fpath <- system.file("extdata", "example_data.xlsx", package = "MetAlyzer")  
obj <- MetAlyzerDataset(file_path = fpath)  
  
obj <- renameMetaData(obj, Method = Group)
```

---

resetMetabolites            *Reset metabolites*

---

**Description**

This method resets the filtering of metabolites.

**Usage**

```
resetMetabolites(object)  
  
## S4 method for signature 'MetAlyzer'  
resetMetabolites(object)
```

**Arguments**

object            MetAlyzer object

**Value**

An updated MetAlyzer object

**Methods (by class)**

- MetAlyzer: Reset metabolites

## Examples

```
fpath <- system.file("extdata", "example_data.xlsx", package = "MetAlyzer")
obj <- MetAlyzerDataset(file_path = fpath)

obj <- resetMetabolites(obj)
```

---

resetMetaData	<i>Reset meta data</i>
---------------	------------------------

---

## Description

This method resets the filter of meta\_data.

## Usage

```
resetMetaData(object)

## S4 method for signature 'MetAlyzer'
resetMetaData(object)
```

## Arguments

object            MetAlyzer object

## Value

An updated MetAlyzer object

## Methods (by class)

- MetAlyzer: Reset meta data

## Examples

```
fpath <- system.file("extdata", "example_data.xlsx", package = "MetAlyzer")
obj <- MetAlyzerDataset(file_path = fpath)

obj <- resetMetaData(obj)
```

---

setPlottingData	<i>Update plotting data</i>
-----------------	-----------------------------

---

**Description**

This method replaces plotting\_data with an updated version.

**Usage**

```
setPlottingData(object, plotting_data)
```

```
## S4 method for signature 'MetAlyzer'  
setPlottingData(object, plotting_data)
```

**Arguments**

object            MetAlyzer object  
plotting\_data    An updated plotting\_data tibble data frame

**Value**

An updated MetAlyzer object

**Methods (by class)**

- MetAlyzer: Update plotting data

**Examples**

```
# To see an example, please check out the vignette.
```

---

```
show, MetAlyzer-method    Show a 'MetAlyzer' object
```

---

**Description**

This method shows a summary of 'MetAlyzer' slot values.

**Usage**

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

**Arguments**

object            MetAlyzer object

**Value**

A summary of the MetAlyzer object

**Examples**

```
fpath <- system.file("extdata", "example_data.xlsx", package = "MetAlyzer")
obj <- MetAlyzerDataset(file_path = fpath)
```

```
show(obj)
# or
obj
```

---

summariseQuantData	<i>Summarize quantification status</i>
--------------------	--

---

**Description**

This method lists the number of each quantification status and its percentage.

**Usage**

```
summariseQuantData(object)

## S4 method for signature 'MetAlyzer'
summariseQuantData(object)
```

**Arguments**

object            MetAlyzer object

**Value**

A summary of the quantification status

**Methods (by class)**

- MetAlyzer: Summarize quantification status

**Examples**

```
fpath <- system.file("extdata", "example_data.xlsx", package = "MetAlyzer")
obj <- MetAlyzerDataset(file_path = fpath)
```

```
summariseQuantData(obj)
```



---

transformPlottingData *Transform plotting data*

---

**Description**

This method performs a transformation of imputed concentration values (imp\_Conc) with a given function. NA values are skipped. The transformed values are added to plotting\_data in an extra column transf\_Conc.

**Usage**

```
transformPlottingData(object, func = log2)

## S4 method for signature 'MetAlyzer'
transformPlottingData(object, func = log2)
```

**Arguments**

object	MetAlyzer object
func	A function to transform concentration values with number of samples

**Value**

An updated MetAlyzer object

**Methods (by class)**

- MetAlyzer: Transform plotting data

**Examples**

```
# To see an example, please check out the vignette.
```

---

updateMetaData *Update meta data*

---

**Description**

This method adds another column to filtered meta\_data.

**Usage**

```
updateMetaData(object, name, new_colum)

## S4 method for signature 'MetAlyzer'
updateMetaData(object, name, new_colum)
```

**Arguments**

<code>object</code>	MetAlyzer object
<code>name</code>	The new column name
<code>new_colum</code>	A vector for the new column (length has to be same as the number of filtered samples)

**Value**

An updated MetAlyzer object

**Methods (by class)**

- MetAlyzer: Update meta data

**Examples**

```
fpath <- system.file("extdata", "example_data.xlsx", package = "MetAlyzer")
obj <- MetAlyzerDataset(file_path = fpath)

obj <- updateMetaData(obj, name = Date, new_colum = Sys.Date())
obj <- updateMetaData(obj, name = Analyzed, new_colum = TRUE)
```

# Index

createPlottingData, [2](#)  
createPlottingData, MetAlyzer-method  
    (createPlottingData), [2](#)

filterMetabolites, [3](#)  
filterMetabolites, MetAlyzer-method  
    (filterMetabolites), [3](#)  
filterMetaData, [4](#)  
filterMetaData, MetAlyzer-method  
    (filterMetaData), [4](#)

imputePlottingData, [5](#)  
imputePlottingData, MetAlyzer-method  
    (imputePlottingData), [5](#)

metabolites, [6](#)  
metabolites, MetAlyzer-method  
    (metabolites), [6](#)  
metaData, [7](#)  
metaData, MetAlyzer-method (metaData), [7](#)  
MetAlyzer (MetAlyzer-class), [8](#)  
MetAlyzer-class, [8](#)  
MetAlyzerDataset, [9](#)

performANOVA, [9](#)  
performANOVA, MetAlyzer-method  
    (performANOVA), [9](#)  
plottingData, [10](#)  
plottingData, MetAlyzer-method  
    (plottingData), [10](#)

quantStatus, [11](#)  
quantStatus, MetAlyzer-method  
    (quantStatus), [11](#)

rawData, [12](#)  
rawData, MetAlyzer-method (rawData), [12](#)  
renameMetaData, [12](#)  
renameMetaData, MetAlyzer-method  
    (renameMetaData), [12](#)  
resetMetabolites, [13](#)  
resetMetabolites, MetAlyzer-method  
    (resetMetabolites), [13](#)  
resetMetaData, [14](#)  
resetMetaData, MetAlyzer-method  
    (resetMetaData), [14](#)  
setPlottingData, [15](#)  
setPlottingData, MetAlyzer-method  
    (setPlottingData), [15](#)  
show, MetAlyzer-method, [15](#)  
summariseQuantData, [16](#)  
summariseQuantData, MetAlyzer-method  
    (summariseQuantData), [16](#)  
transformPlottingData, [17](#)  
transformPlottingData, MetAlyzer-method  
    (transformPlottingData), [17](#)  
updateMetaData, [17](#)  
updateMetaData, MetAlyzer-method  
    (updateMetaData), [17](#)