Package 'pguIMP'

September 30, 2021

Title 'pguIMP'

Version 0.0.0.3

Maintainer Sebastian Malkusch <malkusch@med.uni-frankfurt.de>

Description Reproducible cleaning of bio-medical laboratory data using methods of visualization,error correction and transformation implemented as interactive R-notebooks.

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.1.1

URL https://github.com/SMLMS/pguIMP

BugReports https://github.com/SMLMS/pguIMP/issues

Imports R6 (>= 2.4.1), DT, DataVisualizations, dbscan, dplyr, e1071, finalfit, ggplot2, ggthemes, Hmisc, magrittr, MASS, RWeka, VIM, bbmle, grid, gridExtra, mice, nortest, outliers, plotly, psych, purrr, rcompanion, readr, readxl, rJava, rlang, rmarkdown, robust, shiny, shinydashboard, shinyjs, shinyWidgets, stats, stringr, tibble, tidyr, tidyselect, tools, writexl

Collate 'dLogLikelihood.R' 'importDataSet.R' 'normalDistribution.R' 'pguCorrValidator.R' 'pguCorrelator.R' 'pguDMwR.R' 'pguFile.R' 'pguData.R' 'pguFilter.R' 'pguLimitsOfQuantification.R' 'pguImporter.R' 'pguStatus.R' 'pguReporter.R' 'pguExporter.R' 'pguValidator.R' 'pguImputation.R' 'pguOutlierDetection.R' 'pguOutliers.R' 'pguMissingsCharacterizer.R' 'pguMissings.R' 'pguNormalizer.R' 'sLogLikelihood.R' 'pguNormDist.R' 'pguModel.R' 'pguTransformator.R' 'pguOptimizer.R' 'pguExplorer.R' 'pguDelegate.R' 'transposeTibble.R' 'pguIMP.R' 'pguRegressor.R'

Suggests knitr, devtools, ellipsis (>= 0.3.2), roxygen2

NeedsCompilation no

Author Sebastian Malkusch [aut, cre] (<https://orcid.org/0000-0001-6766-140X>), Joern Loetsch [aut] (<https://orcid.org/0000-0002-5818-6958>)

Repository CRAN

Date/Publication 2021-09-30 11:50:02 UTC

R topics documented:

`centralValue` *centralValue*

Description

Returns the central value of a variable.

Usage

```
centralValue(x, ws = NULL)
```

Arguments

x	variable
ws	weights

Details

Function that obtains a statistic of centrality of a variable, given a sample of values. If the variable is numeric it returns de median, if it is a factor it returns the mode. In other cases it tries to convert to a factor and then returns the mode. Taken from: https://github.com/ltorgo/DMwR2/

Value

central value

Author(s)

Luis Torgo

Examples

```
centralValue(x = seq(1,10,1))
```

---

dLogLikelihood *dLogLikelihood*

Description

Calculates the log Likelihood of a normally distributed event.

Usage

```
dLogLikelihood(x = "numeric", pars = c(mu = 0, sigma = 1))
```

Arguments

x	The x-value(numeric)
pars	Numeric vector with two entries c(mu, sigma). Where mu is the expectation value and sigma is the standard deviation. (numeric)

Value

The logLikelihood. (numeric)

Author(s)

Sebastian Malkusch, <malkusch@med.uni-frankfurt.de>

Examples

```
y <- pguIMP::dLogLikelihood(x=5, pars = c(mu=0.0, sigma=1.0))
```

`importDataSet` *importDataSet*

Description

Imports a data set to the shiny 'pguIMP' web interface. Extracts import options from a 'pguIMP::file' instance and imports the desired record based on the passed information.

Usage

```
importDataSet(obj = "pgu.file")
```

Arguments

`obj`	Instance of the R6 class pguIMP::pgu.file.

Value

A data frame that contains the imported data (tibble::tibble)

Author(s)

Sebastian Malkusch, <malkusch@med.uni-frankfurt.de>

`knnImputation` *knnImputation*

Description

Imputes missings using kNN.

Usage

```
knnImputation(data, k = 10, scale = TRUE, meth = "weighAvg", distData = NULL)
```

Arguments

`data`	data frame containing missing values
`k`	number of nearest neighbors
`scale`	Indicates if data should be scaled
`meth`	Method for estimating the missing value
`distData`	Distance to the case

Details

Function that fills in all unknowns using the k Nearest Neighbours of each case with unknows. By default it uses the values of the neighbours and obtains an weighted (by the distance to the case) average of their values to fill in the unknows. If meth='median' it uses the median/most frequent value, instead. Taken from https://github.com/ltorgo/DMwR2/

Value

cleaned data

Author(s)

Luis Torgo

Examples

```
centralValue(x = seq(1,10,1))
```

---

nnk *nnk*

Description

Outlier detection using kth Nearest Neighbour Distance method Takes a dataset and finds its outliers using distance-based method

Usage

```
nnk(
  x,
  k = 0.05 * nrow(x),
  cutoff = 0.95,
  Method = "euclidean",
  rnames = FALSE,
  boottimes = 100
)
```

Arguments

x	dataset for which outliers are to be found
k	No. of nearest neighbours to be used, default value is 0.05*nrow(x)
cutoff	Percentile threshold used for distance, default value is 0.95
Method	Distance method, default is Euclidean
rnames	Logical value indicating whether the dataset has rownames, default value is False
boottimes	Number of bootsrap samples to find the cutoff, default is 100 samples

Details

nnk computes kth nearest neighbour distance of an observation and based on the bootstrapped cut-off, labels an observation as outlier. Outlierliness of the labelled 'Outlier' is also reported and it is the bootstrap estimate of probability of the observation being an outlier. For bivariate data, it also shows the scatterplot of the data with labelled outliers.

Value

Outlier Observations: A matrix of outlier observations

Location of Outlier: Vector of Sr. no. of outliers

Outlier probability: Vector of proportion of times an outlier exceeds local bootstrap cutof

Author(s)

Vinay Tiwari, Akanksha Kashikar

References

Hautamaki, V., Karkkainen, I., and Franti, P. 2004. Outlier detection using k-nearest neighbour graph. In Proc. IEEE Int. Conf. on Pattern Recognition (ICPR), Cambridge, UK.

Examples

```
#Create dataset
X=iris[,1:4]
#Outlier detection
nnk(X,k=4)
```

---

normalDistribution *normalDistribution*

Description

Probability density distribution of a normally distributed variable.

Usage

```
normalDistribution(x = "numeric", mu = "numeric", sigma = "numeric")
```

Arguments

x	The x-value (numeric)
mu	The expextation value (numeric)
sigma	The standard deviation (numeric)

Details

Calculates p(x | mu, sigma). Where p is the probability of observing an event x given the expected value mu and the standard deviation sigma.

Value

The probability of observing event x given mu and sigma. (numeric)

Author(s)

Sebastian Malkusch, <malkusch@med.uni-frankfurt.de>

Examples

```
y <- pguIMP::normalDistribution(x=5, mu=0.0, sigma=1.0)
```

---

pgu.correlator *pgu.correlator*

Description

An R6 class that performs pairwise correlation on the pguIMP data set.

Format

R6::R6Class object.

Construction

x <- pguIMP::pgu.correlator$new()

Active bindings

`featureNames` Returns the instance variable featureNames. (character)

`setFeatureNames` Sets the instance variable featureNames. It further initializes the instance variables: intercept, pIntercept, slope, pSlope. (character)

`method` Returns the instance variable method. (character)

`r` Returns the instance variable r. (matrix)

`pPearson` Returns the instance variable pPearson. (matrix)

`tau` Returns the instance variable tau. (matrix)

`pKendall` Returns the instance variable pKendall. (matrix)

`rho` Returns the instance variable rho. (matrix)

`pSpearman` Returns the instance variable pSpearman. (matrix)

`abscissa` Returns the instance variable abscissa. (character)

`setAbscissa` Sets the instance variable abscicca to value.

`ordinate` Returns the instance variable ordinate. (character)

`setOrdinate` Sets the instance variable ordinate to value.

`test` Returns the instance variable test. (stats::cor.test)

Methods

Public methods:

- `pgu.correlator$new()`
- `pgu.correlator$finalize()`
- `pgu.correlator$print()`
- `pgu.correlator$resetCorrelator()`
- `pgu.correlator$resetMatrix()`
- `pgu.correlator$featureIdx()`
- `pgu.correlator$calcCorrelationNumeric()`
- `pgu.correlator$createCorrelationMatrixPearson()`
- `pgu.correlator$createCorrelationMatrixKendall()`
- `pgu.correlator$createCorrelationMatrixSpearman()`
- `pgu.correlator$correlate()`
- `pgu.correlator$printFeature()`
- `pgu.correlator$printRTbl()`
- `pgu.correlator$printPPearsonTbl()`
- `pgu.correlator$printTauTbl()`
- `pgu.correlator$printPKendallTbl()`
- `pgu.correlator$printRhoTbl()`
- `pgu.correlator$printPSpearmanTbl()`
- `pgu.correlator$clone()`

Method `new()`**:** Creates and returns a new `pgu.correlator` object.

Usage:

```
pgu.correlator$new(data = "tbl_df")
```

Arguments:

`data` The data to be modeled. (tibble::tibble)

Returns: A new `pgu.correlator` object. (pguIMP::pgu.correlator)

Method `finalize()`**:** Clears the heap and indicates if instance of `pgu.correlator` is removed from heap.

Usage:

```
pgu.correlator$finalize()
```

Method `print()`**:** Prints instance variables of a `pgu.correlator` object.

Usage:

```
pgu.correlator$print()
```

Returns: string

Method `resetCorrelator()`**:** Performes pair-wise correlation analysis on the attributes of the data frame. Progresse is indicated by the progress object passed to the function.

Usage:

```
pgu.correlator$resetCorrelator(data = "tbl_df", progress = "Progress")
```

Arguments:

`data` Dataframe with at least two numeric attributes. (tibble::tibble)

`progress` Keeps track of the analysis progress. (shiny::Progress)

Method `resetMatrix()`**:** Creates a square matrix which dimension corresponds to the length of the instance variable featureNames. The matrix entries are set to a distict `value`.

Usage:

```
pgu.correlator$resetMatrix(value = "numeric")
```

Arguments:

`value` The value the matrix entries are set to. (numeric)

Returns: A square matrix. (matrix)

Method `featureIdx()`**:** Determines the numerical index of the column of an attribute based on the attribute name.

Usage:

```
pgu.correlator$featureIdx(feature = "character")
```

Arguments:

`feature` The attribute's name. (character)

Returns: The attributes column index. (numeric)

Method `calcCorrelationNumeric()`**:** Creates a correlation test between two attributes of a dataframe. The test is stored as instance variable.

Usage:

```
pgu.correlator$calcCorrelationNumeric(
  abscissa = "numeric",
  ordinate = "numeric",
  method = "character"
)
```

Arguments:

`abscissa` The abscissa values. (numeric)

`ordinate` The ordinate values. (numeric)

`method` The cname of the correlation test. Valid coiced are defined by the instance variable `method`. (chatacter)

Method `createCorrelationMatrixPearson()`**:** Performs the actual correlation test routine after Pearson. Iteratively runs through the attributes known to the class and calculates Pearson's correlation for each valid attribute pair. The test results are stored in the instance variables: r, pPearson. Here, pX represents the p-value of the respective parameter X. Displays the progress if shiny is loaded.

Usage:

```
pgu.correlator$createCorrelationMatrixPearson(
  data = "tbl_df",
  progress = "Progress"
)
```

Arguments:

`data` The data to be analysed. (tibble::tibble)

`progress` If shiny is loaded, the analysis' progress is stored within this instance of the shiny Progress class. (shiny::Progress)

Method `createCorrelationMatrixKendall()`**:** Performs the actual correlation test routine after Kendall. Iteratively runs through the attributes known to the class and calculates Kendall's correlation for each valid attribute pair. The test results are stored in the instance variables: tau, pKendall. Here, pX represents the p-value of the respective parameter X. Displays the progress if shiny is loaded.

Usage:

```
pgu.correlator$createCorrelationMatrixKendall(
  data = "tbl_df",
  progress = "Progress"
)
```

Arguments:

`data` The data to be analysed. (tibble::tibble)

`progress` If shiny is loaded, the analysis' progress is stored within this instance of the shiny Progress class. (shiny::Progress)

Method `createCorrelationMatrixSpearman()`**:** Performs the actual correlation test routine after Spearman. Iteratively runs through the attributes known to the class and calculates Spearman's correlation for each valid attribute pair. The test results are stored in the instance variables: rho, pSpearman. Here, pX represents the p-value of the respective parameter X. Displays the progress if shiny is loaded.

Usage:

```
pgu.correlator$createCorrelationMatrixSpearman(
  data = "tbl_df",
  progress = "Progress"
)
```

Arguments:

`data` The data to be analysed. (tibble::tibble)

`progress` If shiny is loaded, the analysis' progress is stored within this instance of the shiny Progress class. (shiny::Progress)

Method `correlate()`**:** Performs the all three correlation test routines defined within the instance variable `method`. Displays the progress if shiny is loaded.

Usage:

```
pgu.correlator$correlate(data = "tbl_df", progress = "Progress")
```

Arguments:

`data` The data to be analysed. (tibble::tibble)

`progress` If shiny is loaded, the analysis' progress is stored within this instance of the shiny Progress class. (shiny::Progress)

Method `printFeature()`**:** Transforms the results of the correlation procedure for a valid pair of attributes to a dataframe and returns it.

Usage:

```
pgu.correlator$printFeature()
```

Returns: The analyis result as a dataframe. (tibble::tibble)

Method `printRTbl()`**:** Transfroms instance variable `r` to a dataframe and returns it.

Usage:

```
pgu.correlator$printRTbl()
```

Returns: Dataframe of instance variable `r`. (tibble::tibble)

Method `printPPearsonTbl()`**:** Transfroms instance variable `pPearson` to a dataframe and returns it.

Usage:

```
pgu.correlator$printPPearsonTbl()
```

Returns: Dataframe of instance variable `pPearson`. (tibble::tibble)

Method `printTauTbl()`**:** Transfroms instance variable `tau` to a dataframe and returns it.

Usage:

```
pgu.correlator$printTauTbl()
```

Returns: Dataframe of instance variable `tau`. (tibble::tibble)

Method `printPKendallTbl()`**:** Transfroms instance variable `pKendall` to a dataframe and returns it.

Usage:

```
pgu.correlator$printPKendallTbl()
```

Returns: Dataframe of instance variable `pKendall`. (tibble::tibble)

Method `printRhoTbl()`**:** Transfroms instance variable `rho` to a dataframe and returns it.

Usage:

```
pgu.correlator$printRhoTbl()
```

Returns: Dataframe of instance variable `rho`. (tibble::tibble)

Method `printPSpearmanTbl()`**:** Transfroms instance variable `pSpearman` to a dataframe and returns it.

Usage:

```
pgu.correlator$printPSpearmanTbl()
```

Returns: Dataframe of instance variable `pSpearman`. (tibble::tibble)

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
pgu.correlator$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Author(s)

Sebastian Malkusch, <malkusch@med.uni-frankfurt.de>

Examples

```
require(dplyr)
require(tibble)
data(iris)
data_df <- iris %>%
  tibble::as_tibble() %>%
  dplyr::select(-c("Species"))
correlator = pguIMP::pgu.correlator$new(data_df)
```

`pgu.corrValidator` *pgu.corrValidator*

Description

An R6 class that performs pairwise correlation of the features of the original and the imputed data set. The correlation results of both data sets are compared by subtraction.

Format

R6::R6Class object.

Construction

x <- pguIMP::pgu.corrValidator$new()

Active bindings

`featureNames` Returns the instance variable featureNames. (character)

`orgR_mat` Returns the instance variable orgR_mat. (matrix)

`impR_mat` Returns the instance variable impR_mat. (matrix)

`orgP_mat` Returns the instance variable orgP_mat. (matrix)

`impP_mat` Returns the instance variable impP_mat. (matrix)

`corr_df` Returns the instance variable corr_df. (tibble::tibble)

`summary_df` Returns the instance variable summary_df. (tibble::tibble)

Methods

Public methods:

- pgu.corrValidator$new()
- pgu.corrValidator$print()
- pgu.corrValidator$reset()
- pgu.corrValidator$fit()
- pgu.corrValidator$correlationScatterPlot()
- pgu.corrValidator$correlationBarPlot()
- pgu.corrValidator$correlationBoxPlot()
- pgu.corrValidator$correlationCompoundPlot()
- pgu.corrValidator$clone()

Method `new()`**:** Clears the heap and indicates if instance of `pgu.corrValidator` is removed from heap.

Summary of the correlation deviation distribution.

Creates a square matrix which dimension corresponds to the length of the instance variable featureNames. The matrix entries are set to a distinct `value`.

Flattens the results transforms them into a dataframe and stores it into the instance variable corr_df.

Creates and returns a new `pgu.corrValidator` object.

Usage:

```
pgu.corrValidator$new(org_df = "tbl_df", imp_df = "tbl_df")
```

Arguments:

`org_df` The original data to be analyzed. (tibble::tibble)

`imp_df` The imputed version of the org_df data.

Returns: A new `pgu.corrValidator` object. (pguIMP::pgu.corrValidator)

Method `print()`**:** Prints instance variables of a `pgu.corrValidator` object.

Usage:

```
pgu.corrValidator$print()
```

Returns: string

Method `reset()`**:** Resets the object `pgu.corrValidator` based on the instance variable featureNames..

Usage:

```
pgu.corrValidator$reset()
```

Method `fit()`**:** Runs the corraltion analysis.

Usage:

```
pgu.corrValidator$fit(org_df = "tbl_df", imp_df = "tbl_df")
```

Arguments:

`org_df` Adataframe comprising the original data. (tibble::tibble)

`imp_df` Adataframe comprising the imputed data. (tibble::tibble)

Method `correlationScatterPlot()`**:** Plots the correlation analysis results.

Usage:

```
pgu.corrValidator$correlationScatterPlot()
```

Method `correlationBarPlot()`**:** Creates and returns a histogram from the cor_delat values.

Usage:

```
pgu.corrValidator$correlationBarPlot()
```

Returns: Bar plot (ggplot2::ggplot)

Method `correlationBoxPlot()`**:** Plots the correlation analysis results.

Usage:

```
pgu.corrValidator$correlationBoxPlot()
```

Method `correlationCompoundPlot()`**:** Creates and returns a compund graphical analysis of the cor_delta values.

Usage:

```
pgu.corrValidator$correlationCompoundPlot()
```

Returns: Compound plot (gridExtra::grid.arrange)

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
pgu.corrValidator$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Author(s)

Sebastian Malkusch, <malkusch@med.uni-frankfurt.de>

Examples

```
require(dplyr)
require(tibble)
data(iris)
data_df <- iris %>%
  tibble::as_tibble()
comp_df <- data_df %>%
  dplyr::mutate(Sepal.Length = sample(Sepal.Length))
corr_obj = pguIMP::pgu.corrValidator$new()
corr_obj$fit(data_df, comp_df)
print(corr_obj)
```

pgu.data *pgu.data*

Description

Handles the pguIMP dataset.

Format

R6::R6Class object.

Details

Stores the pguIMP dataset as instance variable and keeps track of the attributes of interest. Provides additionally fast access to several statistical information about the data set. This object is used by the shiny based gui and is not for use in individual R-scripts!

Active bindings

`rawData` Returns the instance variable rawData (tibble::tibble)

`setRawData` Sets the instance variable rawData (tibble::tibble)

`attributeNames` Returns the instance variable attributeNames (character)

`numericalAttributeNames` Returns the instance variable numericalAttributeNames (character)

`categoricalAttributeNames` Returns the instance variable categoricalAttributeNames (character)

`classInformation` Returns the instance variable classInformation (tibble::tibble)

`statistics` Returns the instance variable statistics (tibble::tibble)

`reducedStatistics` Returns the instance variable reducedStatistics (tibble::tibble)

`missingsStatistics` Returns the instance variable missingsStatistics (tibble::tibble)

Methods

Public methods:

- `pgu.data$new()`
- `pgu.data$print()`
- `pgu.data$fit()`
- `pgu.data$attribute_index()`
- `pgu.data$numerical_data()`
- `pgu.data$categorical_data()`
- `pgu.data$clone()`

Method `new()`**:** Clears the heap and indicates that instance of pguIMP::pgu.data is removed from heap.

Summarizes information on the instance variable rawData and retruns it in form of a compact data frame.

Summarizes a vector of numericals and returns summary.

Iterativley calls the function summarize_numerical_data on all numerical attributes of the instance variable rawData and returns the result in form of a data frame.

Calls the function calculate_statistics filters the result for the attribute names and arithmetic mean values. and returns the result in form of a data frame.

Calls the class' function dataStatistics filters the result for the attribute names and information about missing values. and returns the result in form of a data frame.

Creates and returns a new pguIMP::pgu.data object.

Usage:

```
pgu.data$new(data_df = "tbl_df")
```

Arguments:

`data_df` The data to be analyzed. (tibble::tibble)

`val` Vector of numericals to be summarized. (numeric)

Returns: A new pguIMP::pgu.data object. (pguIMP::pgu.data)

Method `print()`**:** Prints instance variables of a pguIMP::pgu.data object.

Usage:

```
pgu.data$print()
```

Returns: string

Method `fit()`**:** Extracts information about the instance variable rawData.

Usage:

```
pgu.data$fit()
```

Method `attribute_index()`**:** Returns the index of an attribute within the instance variable attributeNames.

Usage:

```
pgu.data$attribute_index(attribute = "character")
```

Arguments:

`attribute` Attribute's name. (character)

Returns: Index of attribute's name in rawData (numeric)

Method `numerical_data()`**:** Returns the numeric attributes of the instance variable rawData.

Usage:

```
pgu.data$numerical_data()
```

Returns: A data frame (tibble::tibble)

Method `categorical_data()`**:** Returns the categorical attributes of the instance variable rawData.

Usage:

```
pgu.data$categorical_data()
```

Returns: A data frame (tibble::tibble)

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
pgu.data$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Author(s)

Sebastian Malkusch, <malkusch@med.uni-frankfurt.de>

Examples

```
require(dplyr)
require(tibble)
data(iris)
data_df <- iris %>%
  tibble::as_tibble()
data_obj = pguIMP::pgu.data$new(data_df)
```

`pgu.delegate` *pgu.delegate*

Description

Manages the communication between the shiny gui layer and the classes of the pguIMP package

Format

R6::R6Class object.

Details

Comprises all needed classes from the pguIMP package and manages the communication between the gui and the analysis. This object is used by the shiny based gui and is not for use in individual R-scripts!

Active bindings

`status` Returns the instance variable status (pguIMP::pgu.status)

`fileName` Returns the instance variable fileName (pguIMP::pgu.file)

`loqFileName` Returns the instance variable loqFileName (pguIMP::pgu.file)

`rawData` Returns the instance variable rawData (pguIMP::pgu.data)

`filterSet` Returns the instance variable filterSet (pguIMP::pgu.filter)

`filteredData` Returns the instance variable filteredData (pguIMP::pgu.data)

`loq` Returns the instance variable loq (pguIMP::pgu.limitsOfQuantification)

`loqMutatedData` Returns the instance variable loqMutatedData (pguIMP::pgu.data)

`explorer` Returns the instance variable explorer (pguIMP::pgu.explorer)

`optimizer` Returns the instance variable optimizer (pguIMP::pgu.optimizer)

`transformator` Returns the instance variable transformator (pguIMP::pgu.transformator)

`model` Returns the instance variable model (pguIMP::pgu.model)

`transformedData` Returns the instance variable transformedData (pguIMP::pgu.data)

`featureModel` Returns the instance variable featureModel (pguIMP::pgu.normDist)

`normalizer` Returns the instance variable normalizer (pguIMP::pgu.normalizer)

`normalizedData` Returns the instance variable normalizedData (pguIMP::pgu.data)

`missings` Returns the instance variable missings (pguIMP::pgu.missings)

`missingsCharacterizer` Returns the instance variable missingsCharacterizer (pguIMP::pgu.missingsCharacterizer)

`outliers` Returns the instance variable outlierd (pguIMP::pgu.outliers)

`imputer` Returns the instance variable imputer (pguIMP::pgu.imputation)

`imputedData` Returns the instance variable imputedData (pguIMP::pgu.data)

`cleanedData` Returns the instance variable cleanedData (pguIMP::pgu.data)

`validator` Returns the instance variable validator (pguIMP::pgu.validator)

`corrValidator` Returns the instance variable corrValidator (pguIMP::pgu.corrValidator)

`exporter` Returns the instance variable exporter (pguIMP::pgu.exporter)

`reporter` Returns the instance variable reporter (pguIMP::pgu.reporter)

Methods

Public methods:

- `pgu.delegate$new()`
- `pgu.delegate$print()`
- `pgu.delegate$update_import_gui()`
- `pgu.delegate$query_data()`
- `pgu.delegate$import_data()`
- `pgu.delegate$update_import_data_Types_tbl()`
- `pgu.delegate$update_import_data_statistics_tbl()`
- `pgu.delegate$update_import_missings_statistics_tbl()`

- pgu.delegate$update_filter_select_tbl()
- pgu.delegate$update_filter()
- pgu.delegate$update_filter_inverse()
- pgu.delegate$reset_filter()
- pgu.delegate$filter_data()
- pgu.delegate$update_filter_statistics_tbl()
- pgu.delegate$update_filter_missings_tbl()
- pgu.delegate$update_exploration_gui()
- pgu.delegate$update_exploration_abscissa()
- pgu.delegate$update_exploration_ordinate()
- pgu.delegate$update_exploration_graphic()
- pgu.delegate$update_exploration_abscissa_graphic()
- pgu.delegate$update_exploration_ordinate_graphic()
- pgu.delegate$update_exploration_abscissa_table()
- pgu.delegate$update_exploration_ordinate_table()
- pgu.delegate$reset_loq_values()
- pgu.delegate$update_loq_upload_gui()
- pgu.delegate$query_loq()
- pgu.delegate$import_loq()
- pgu.delegate$update_loq_define_gui()
- pgu.delegate$update_loq_define_feature()
- pgu.delegate$update_loq_define_lloq()
- pgu.delegate$update_loq_define_uloq()
- pgu.delegate$update_loq_define_table()
- pgu.delegate$update_loq_define_menu()
- pgu.delegate$set_loq_define_values()
- pgu.delegate$set_loq_define_values_globally()
- pgu.delegate$update_loq_detect_gui()
- pgu.delegate$update_loq_na_handling()
- pgu.delegate$init_detect_loq()
- pgu.delegate$detect_loq()
- pgu.delegate$update_loq_detect_statistics_tbl()
- pgu.delegate$update_loq_detect_outlier_tbl()
- pgu.delegate$update_loq_detect_statistics_graphic()
- pgu.delegate$update_loq_detect_attribute_graphic()
- pgu.delegate$update_loq_detect_attribute_tbl()
- pgu.delegate$update_loq_mutate_gui()
- pgu.delegate$update_lloq_substitute()
- pgu.delegate$update_uloq_substitute()
- pgu.delegate$init_mutate_loq()
- pgu.delegate$mutate_loq()
- pgu.delegate$update_loq_mutate_data_tbl()

- pgu.delegate$update_loq_mutate_statistics_graphic()
- pgu.delegate$update_loq_mutate_attribute_graphic()
- pgu.delegate$init_loq_mutate_attribute_tbl()
- pgu.delegate$update_loq_mutate_attribute_tbl()
- pgu.delegate$optimizeTrafoParameter()
- pgu.delegate$updateDetectedTrafoTypes()
- pgu.delegate$updateDetectedTrafoParameter()
- pgu.delegate$updateTrafoDetectGui()
- pgu.delegate$updateTrafoMutateFeature()
- pgu.delegate$updateTrafoMutateType()
- pgu.delegate$updateTrafoMutateLambda()
- pgu.delegate$updateTrafoMutateMirror()
- pgu.delegate$resetTrafoMutateGui()
- pgu.delegate$updateTrafoMutateGui()
- pgu.delegate$trafoMutateFit()
- pgu.delegate$trafoMutateGlobal()
- pgu.delegate$trafoMutateFeature()
- pgu.delegate$updateTrafoMutateFeatureGraphic()
- pgu.delegate$updateTrafoMutateFeatureParameterTbl()
- pgu.delegate$updateTrafoMutateFeatureQualityTbl()
- pgu.delegate$updateTrafoMutateGlobalParameterTbl()
- pgu.delegate$updateTrafoMutateGlobalModelTbl()
- pgu.delegate$updateTrafoMutateGlobalQualityTbl()
- pgu.delegate$updateTrafoMutateGlobalTestsTbl()
- pgu.delegate$updateTrafoMutateGlobalDataTbl()
- pgu.delegate$updateTrafoNormFeature()
- pgu.delegate$updateTrafoNormMethod()
- pgu.delegate$updateTrafoNormGui()
- pgu.delegate$trafoNormMutate()
- pgu.delegate$updateTrafoNormFeatureGraphic()
- pgu.delegate$resetTrafoNormGui()
- pgu.delegate$updateTrafoNormFeatureStatisticsTbl()
- pgu.delegate$updateTrafoNormStatisticsTbl()
- pgu.delegate$updateTrafoNormParameterTbl()
- pgu.delegate$updateTrafoNormDataTbl()
- pgu.delegate$imputeMissingsAnalyze()
- pgu.delegate$updateImputeMissingsGraphic()
- pgu.delegate$updateImputeMissingsStatisticsTbl()
- pgu.delegate$updateImputeMissingsDistributionTbl()
- pgu.delegate$updateImputeMissingCharacteristicsGraphic()
- pgu.delegate$updateImputeMissingsCharacteristicsTbl()
- pgu.delegate$updateImputeMissingsDetailTbl()

- pgu.delegate$updateImputeOutliersMethod()
- pgu.delegate$updateImputeOutliersFeature()
- pgu.delegate$updateImputeOutliersAlpha()
- pgu.delegate$updateImputeOutliersEpsilon()
- pgu.delegate$updateImputeOutliersMinSamples()
- pgu.delegate$updateImputeOutliersGamma()
- pgu.delegate$updateImputeOutliersNu()
- pgu.delegate$updateImputeOutliersCutoff()
- pgu.delegate$updateImputeOutliersK()
- pgu.delegate$updateImputeOutliersSeed()
- pgu.delegate$updateImputeOutliersGui()
- pgu.delegate$resetImputeOutliersGui()
- pgu.delegate$imputeOutliersDetect()
- pgu.delegate$updateImputeOutliersGraphic()
- pgu.delegate$updateImputeOutliersFeatureGraphic()
- pgu.delegate$updateImputeOutliersFeatureTbl()
- pgu.delegate$updateImputeOutliersStatisticsTbl()
- pgu.delegate$updateImputeOutliersDetailTbl()
- pgu.delegate$updateImputeMutateFeature()
- pgu.delegate$updateImputeMutateMethod()
- pgu.delegate$updateImputeMutateNNeighbors()
- pgu.delegate$updateImputeMutatePredFrac()
- pgu.delegate$updateImputeMutateOutfluxThr()
- pgu.delegate$updateImputeMutateSeed()
- pgu.delegate$updateImputeMutateIterations()
- pgu.delegate$updateImputeMutateGui()
- pgu.delegate$resetImputeMutateGui()
- pgu.delegate$imputeMutateMutate()
- pgu.delegate$updateImputeFluxGraphic()
- pgu.delegate$updateImputeMutateGraphic()
- pgu.delegate$updateImputeMutateStatisticsTbl()
- pgu.delegate$updateImputeMutateDistributionTbl()
- pgu.delegate$updateImputeMutateFeatureDetailGraphic()
- pgu.delegate$updateImputeMutateFeatureDetailTbl()
- pgu.delegate$updateImputeMutateDetailTbl()
- pgu.delegate$updateImputeMutateDataTbl()
- pgu.delegate$validate()
- pgu.delegate$updateAnalysisValidationGui()
- pgu.delegate$updateAnalysisValidationGraphic()
- pgu.delegate$updateAnalysisValidationTestTbl()
- pgu.delegate$updateCentralMomentsOrgTbl()
- pgu.delegate$updateCentralMomentsImpTbl()

- pgu.delegate$updateCentralMomentsDeltaTbl()
- pgu.delegate$updateCorrelationValidationScatterGraphic()
- pgu.delegate$updateCorrelationValidationBoxPlotGraphic()
- pgu.delegate$updateCorrelationValidationDeviationTbl()
- pgu.delegate$updateCorrelationValidationDataTbl()
- pgu.delegate$exportFileName()
- pgu.delegate$exportData()
- pgu.delegate$reportFileName()
- pgu.delegate$writeReport()
- pgu.delegate$hide_outdated_results()
- pgu.delegate$update_help_html()
- pgu.delegate$clone()

Method `new()`**:** Clears the heap and indicates that instance of `pgu.delegate` is removed from heap.

Creates and returns a new `pgu.delegate` object.

Usage:

```
pgu.delegate$new(data = "tbl_df")
```

Arguments:

`data` The data to be analyzed. (tibble::tibble)

Returns: A new `pgu.delegate` object. (pguIMP::pgu.delegate)

Method `print()`**:** Prints instance variables of a `pgu.delegate` object.

Usage:

```
pgu.delegate$print()
```

Returns: string

Method `update_import_gui()`**:** Updates the import gui

Usage:

```
pgu.delegate$update_import_gui(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `query_data()`**:** Manages the data upload to the R server. Updates the instance class status.

Usage:

```
pgu.delegate$query_data(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `import_data()`: Imports uploaded data from the R server into the instance variable rawData. Updates the instance class status.

Usage:

```
pgu.delegate$import_data(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `update_import_data_Types_tbl()`: Updates the tbl.importDataTypes table.

Usage:

```
pgu.delegate$update_import_data_Types_tbl(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `update_import_data_statistics_tbl()`: Updates the tbl.importDataStatistics table.

Usage:

```
pgu.delegate$update_import_data_statistics_tbl(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `update_import_missings_statistics_tbl()`: Updates the tbl.importMissingsStatistics table.

Usage:

```
pgu.delegate$update_import_missings_statistics_tbl(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `update_filter_select_tbl()`: Updates the tbl.filter table.

Usage:

```
pgu.delegate$update_filter_select_tbl(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `update_filter()`**:** Queries the filter parameters selected by the user in the gui and stores them in the instance variable filterSet.

Usage:

```
pgu.delegate$update_filter(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `update_filter_inverse()`**:** Queries the filter parameters selected by the user in the gui inverts them and stores them in the instance variable filterSet.

Usage:

```
pgu.delegate$update_filter_inverse(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `reset_filter()`**:** Generates a filter set that selects the whole data frame. Stores them in the instance variable filterSet. Updates the gui.

Usage:

```
pgu.delegate$reset_filter(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `filter_data()`**:** Filters the data corresponding to the user defined parameters stored in the instance variable filterSet. Results are stored in the instance variables filteredData and filteredMetadata. Updated the instance variable filterSet.

Usage:

```
pgu.delegate$filter_data(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `update_filter_statistics_tbl()`**:** Updates the tbl.filterStatistics table.

Usage:

```
pgu.delegate$update_filter_statistics_tbl(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `update_filter_missings_tbl()`: Updates the tbl.filterMissings table.

Usage:

```
pgu.delegate$update_filter_missings_tbl(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `update_exploration_gui()`: Updates the gui.

Usage:

```
pgu.delegate$update_exploration_gui(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `update_exploration_abscissa()`: Transfers the information oabout the selected abscissa attribute to the explorer class.

Usage:

```
pgu.delegate$update_exploration_abscissa(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `update_exploration_ordinate()`: Transfers the information oabout the selected ordinate attribute to the explorer class.

Usage:

```
pgu.delegate$update_exploration_ordinate(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `update_exploration_graphic()`: Updates the exploration abscissa vs. ordinate scatter plot corresponding to the respective user defined attributes.

Usage:

```
pgu.delegate$update_exploration_graphic(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `update_exploration_abscissa_graphic()`: Updates the abscissa compound plot corresponding to the respective user defined attributes.

Usage:

```
pgu.delegate$update_exploration_abscissa_graphic(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `update_exploration_ordinate_graphic()`: Updates the ordinate compound plot corresponding to the respective user defined attributes.

Usage:

```
pgu.delegate$update_exploration_ordinate_graphic(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `update_exploration_abscissa_table()`: Updates the numerical abscissa analysis table. corresponding to the respective user defined attributes.

Usage:

```
pgu.delegate$update_exploration_abscissa_table(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `update_exploration_ordinate_table()`: Updates the numerical ordinate analysis table. corresponding to the respective user defined attributes.

Usage:

```
pgu.delegate$update_exploration_ordinate_table(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `reset_loq_values()`: Initializes the LOQ object after filtering.

Usage:

```
pgu.delegate$reset_loq_values(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output
`session` Pointer to shiny session

Method `update_loq_upload_gui()`**:** Updates the gui.

Usage:

```
pgu.delegate$update_loq_upload_gui(input, output, session)
```

Arguments:

`input` Pointer to shiny input
`output` Pointer to shiny output
`session` Pointer to shiny session

Method `query_loq()`**:** Manages the loq data upload to the R server.

Usage:

```
pgu.delegate$query_loq(input, output, session)
```

Arguments:

`input` Pointer to shiny input
`output` Pointer to shiny output
`session` Pointer to shiny session

Method `import_loq()`**:** Imports the loq data upload to the R server.

Usage:

```
pgu.delegate$import_loq(input, output, session)
```

Arguments:

`input` Pointer to shiny input
`output` Pointer to shiny output
`session` Pointer to shiny session

Method `update_loq_define_gui()`**:** Updates the gui.

Usage:

```
pgu.delegate$update_loq_define_gui(input, output, session)
```

Arguments:

`input` Pointer to shiny input
`output` Pointer to shiny output
`session` Pointer to shiny session

Method `update_loq_define_feature()`**:** Updates the gui.

Usage:

```
pgu.delegate$update_loq_define_feature(input, output, session)
```

Arguments:

`input` Pointer to shiny input
`output` Pointer to shiny output
`session` Pointer to shiny session

Method `update_loq_define_lloq()`: Updates the gui.

Usage:

```
pgu.delegate$update_loq_define_lloq(input, output, session)
```

Arguments:

`input` Pointer to shiny input
`output` Pointer to shiny output
`session` Pointer to shiny session

Method `update_loq_define_uloq()`: Updates the gui.

Usage:

```
pgu.delegate$update_loq_define_uloq(input, output, session)
```

Arguments:

`input` Pointer to shiny input
`output` Pointer to shiny output
`session` Pointer to shiny session

Method `update_loq_define_table()`: Updates the gui.

Usage:

```
pgu.delegate$update_loq_define_table(input, output, session)
```

Arguments:

`input` Pointer to shiny input
`output` Pointer to shiny output
`session` Pointer to shiny session

Method `update_loq_define_menu()`: Updates the gui.

Usage:

```
pgu.delegate$update_loq_define_menu(input, output, session)
```

Arguments:

`input` Pointer to shiny input
`output` Pointer to shiny output
`session` Pointer to shiny session

Method `set_loq_define_values()`: Updates loq class.

Usage:

```
pgu.delegate$set_loq_define_values(input, output, session)
```

Arguments:

`input` Pointer to shiny input
`output` Pointer to shiny output
`session` Pointer to shiny session

Method `set_loq_define_values_globally()`: Updates loq class.

Usage:

```
pgu.delegate$set_loq_define_values_globally(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session #' @description #' Imports uploaded data from the R server into the instance variable loqData. #' Updates the instance class status. #' @param input #' Pointer to shiny input #' @param output #' Pointer to shiny output #' @param session #' Pointer to shiny session importLoq = function(input, output, session) if (private$.status$query(processName = "dataImported")) tryCatch(private$.loq$setLoq <- private$.importer$importLoq(self$fileName) private$.status$update(processName = "loqImported", value = TRUE) , error = function(e) private$.status$update(processName = "loqImported", value = FALSE) shiny::showNotification(paste(e),type = "error", duration = 10) #error)#tryCatch #if else private$.status$update(processName = "loqImported", value = FALSE) shiny::showNotification(paste("No file uploaded to import. Please upload a valid file first."),type = "error", duration = 10) #else , #function

Method `update_loq_detect_gui()`**:** Updates the gui.

Usage:

```
pgu.delegate$update_loq_detect_gui(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `update_loq_na_handling()`**:** Updates the si.loqHandling shiny widget corresponding to the respective user defined parameter.

Usage:

```
pgu.delegate$update_loq_na_handling(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `init_detect_loq()`**:** Runs the outlier detection routine of the instance variable outliers. Updates the instance class status.

Usage:

```
pgu.delegate$init_detect_loq(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `detect_loq()`**:** Runs the outlier detection routine of the instance variable outliers. Updates the instance class status.

Usage:

```
pgu.delegate$detect_loq(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `update_loq_detect_statistics_tbl()`**:** Updates the numerical loq statistics analysis table

Usage:

```
pgu.delegate$update_loq_detect_statistics_tbl(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `update_loq_detect_outlier_tbl()`**:** Updates the numerical loq table.

Usage:

```
pgu.delegate$update_loq_detect_outlier_tbl(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `update_loq_detect_statistics_graphic()`**:** Updates the loq statistics graphic.

Usage:

```
pgu.delegate$update_loq_detect_statistics_graphic(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `update_loq_detect_attribute_graphic()`**:** Updates the loq feature compound graphic.

Usage:

```
pgu.delegate$update_loq_detect_attribute_graphic(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `update_loq_detect_attribute_tbl()`**:** Updates the numerical loq feature table.

Usage:

```
pgu.delegate$update_loq_detect_attribute_tbl(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `update_loq_mutate_gui()`**:** Updates the gui.

Usage:

```
pgu.delegate$update_loq_mutate_gui(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `update_lloq_substitute()`**:** Updates the si.lloqSubstitute shiny widget.

Usage:

```
pgu.delegate$update_lloq_substitute(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `update_uloq_substitute()`**:** Updates the si.uloqSubstitute shiny widget.

Usage:

```
pgu.delegate$update_uloq_substitute(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `init_mutate_loq()`**:** Calls the mutation routine of the instance variable loq on the instance variable filteredData. The reult is stored in the instance variable loqMutatedData Updates the instance class status.

Usage:

```
pgu.delegate$init_mutate_loq(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `mutate_loq()`**:** Calls the mutation routine of the instance variable loq on the instance variable filteredData. The reult is stored in the instance variable loqMutatedData Updates the instance class status.

Usage:

```
pgu.delegate$mutate_loq(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `update_loq_mutate_data_tbl()`**:** Updates the numerical loq mutate outliers table.

Usage:

```
pgu.delegate$update_loq_mutate_data_tbl(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `update_loq_mutate_statistics_graphic()`**:** Updates the loq mutate statistics graphic.

Usage:

```
pgu.delegate$update_loq_mutate_statistics_graphic(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `update_loq_mutate_attribute_graphic()`**:** Updates the loq mutate feature graphic.

Usage:

```
pgu.delegate$update_loq_mutate_attribute_graphic(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `init_loq_mutate_attribute_tbl()`**:** Updates the numeric loq mutate feature table.

Usage:

```
pgu.delegate$init_loq_mutate_attribute_tbl(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `update_loq_mutate_attribute_tbl()`**:** Updates the numeric loq mutate feature table.

Usage:

```
pgu.delegate$update_loq_mutate_attribute_tbl(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `optimizeTrafoParameter()`**:** Calls the optimize routine of the instance variable optimizer on the instance variable loqMutatedData. Updates the instance class status.

Usage:

```
pgu.delegate$optimizeTrafoParameter(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `updateDetectedTrafoTypes()`**:** Updates the detected trafo types table.

Usage:

```
pgu.delegate$updateDetectedTrafoTypes(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `updateDetectedTrafoParameter()`**:** Updates the detected trafo parameters table.

Usage:

```
pgu.delegate$updateDetectedTrafoParameter(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `updateTrafoDetectGui()`**:** Updates the gui.

Usage:

```
pgu.delegate$updateTrafoDetectGui(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `updateTrafoMutateFeature()`**:** Updates the si.trafoMutateFeature shiny widget.

Usage:

```
pgu.delegate$updateTrafoMutateFeature(input, output, session)
```

Arguments:

`input` Pointer to shiny input
`output` Pointer to shiny output
`session` Pointer to shiny session

Method `updateTrafoMutateType()`: Updates the si.trafoMutateType shiny widget.

Usage:

```
pgu.delegate$updateTrafoMutateType(input, output, session)
```

Arguments:

`input` Pointer to shiny input
`output` Pointer to shiny output
`session` Pointer to shiny session

Method `updateTrafoMutateLambda()`: Updates the ni.trafoMutateLambda shiny widget.

Usage:

```
pgu.delegate$updateTrafoMutateLambda(input, output, session)
```

Arguments:

`input` Pointer to shiny input
`output` Pointer to shiny output
`session` Pointer to shiny session

Method `updateTrafoMutateMirror()`: Updates the cb.trafoMutateMirror shiny widget.

Usage:

```
pgu.delegate$updateTrafoMutateMirror(input, output, session)
```

Arguments:

`input` Pointer to shiny input
`output` Pointer to shiny output
`session` Pointer to shiny session

Method `resetTrafoMutateGui()`: Updates the gui.

Usage:

```
pgu.delegate$resetTrafoMutateGui(input, output, session)
```

Arguments:

`input` Pointer to shiny input
`output` Pointer to shiny output
`session` Pointer to shiny session

Method `updateTrafoMutateGui()`: Updates the gui.

Usage:

```
pgu.delegate$updateTrafoMutateGui(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `trafoMutateFit()`**:** Estimates the optimal transformation parameters. Updates the GUI

Usage:

```
pgu.delegate$trafoMutateFit(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `trafoMutateGlobal()`**:** Calls the transformation routine of the instance variable transformator on the instance variable loqMutatedData. Updates the instance class status.

Usage:

```
pgu.delegate$trafoMutateGlobal(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `trafoMutateFeature()`**:** Calls the transformation routine of the instance variable transformator on a user defined attribute of the instance variable loqMutatedData.

Usage:

```
pgu.delegate$trafoMutateFeature(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `updateTrafoMutateFeatureGraphic()`**:** Updates the trafo mutate feature graphic.

Usage:

```
pgu.delegate$updateTrafoMutateFeatureGraphic(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `updateTrafoMutateFeatureParameterTbl()`**:** Updates the trafo mutate feature patameter table.

Usage:

```
pgu.delegate$updateTrafoMutateFeatureParameterTbl(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `updateTrafoMutateFeatureQualityTbl()`**:** Updates the trafo mutate feature quality table.

Usage:

```
pgu.delegate$updateTrafoMutateFeatureQualityTbl(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `updateTrafoMutateGlobalParameterTbl()`**:** Updates the trafo mutate global parameter table.

Usage:

```
pgu.delegate$updateTrafoMutateGlobalParameterTbl(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `updateTrafoMutateGlobalModelTbl()`**:** Updates the tbl.trafoMutateGlobalModel table.

Usage:

```
pgu.delegate$updateTrafoMutateGlobalModelTbl(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `updateTrafoMutateGlobalQualityTbl()`**:** Updates the tbl.trafoMutateGlobalQuality table.

Usage:

```
pgu.delegate$updateTrafoMutateGlobalQualityTbl(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `updateTrafoMutateGlobalTestsTbl()`**:** Updates the tbl.trafoMutateGlobalTests table.

Usage:

```
pgu.delegate$updateTrafoMutateGlobalTestsTbl(input, output, session)
```

Arguments:

`input` Pointer to shiny input
`output` Pointer to shiny output
`session` Pointer to shiny session

Method `updateTrafoMutateGlobalDataTbl()`**:** Updates the tbl.trafoMutateGlobalData table.

Usage:

```
pgu.delegate$updateTrafoMutateGlobalDataTbl(input, output, session)
```

Arguments:

`input` Pointer to shiny input
`output` Pointer to shiny output
`session` Pointer to shiny session

Method `updateTrafoNormFeature()`**:** Updates the si.trafoNormFeature shiny widget.

Usage:

```
pgu.delegate$updateTrafoNormFeature(input, output, session)
```

Arguments:

`input` Pointer to shiny input
`output` Pointer to shiny output
`session` Pointer to shiny session

Method `updateTrafoNormMethod()`**:** Updates the si.trafoNormMethod shiny widget.

Usage:

```
pgu.delegate$updateTrafoNormMethod(input, output, session)
```

Arguments:

`input` Pointer to shiny input
`output` Pointer to shiny output
`session` Pointer to shiny session

Method `updateTrafoNormGui()`**:** Updates the gui.

Usage:

```
pgu.delegate$updateTrafoNormGui(input, output, session)
```

Arguments:

`input` Pointer to shiny input
`output` Pointer to shiny output
`session` Pointer to shiny session

Method `trafoNormMutate()`**:** Calls the scale routine of the instance variable normalizer on the instance variable transformedData. Updates the instance class status.

Usage:

```
pgu.delegate$trafoNormMutate(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `updateTrafoNormFeatureGraphic()`: Updates the impute norm feature compound graphic.

Usage:

```
pgu.delegate$updateTrafoNormFeatureGraphic(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `resetTrafoNormGui()`: Updates the gui.

Usage:

```
pgu.delegate$resetTrafoNormGui(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `updateTrafoNormFeatureStatisticsTbl()`: Updates the numerical impute norm analysis table for a user defined feature. corresponding to the respective user defined attributes.

Usage:

```
pgu.delegate$updateTrafoNormFeatureStatisticsTbl(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `updateTrafoNormStatisticsTbl()`: Updates the numerical impute norm analysis table. corresponding to the respective user defined attributes.

Usage:

```
pgu.delegate$updateTrafoNormStatisticsTbl(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `updateTrafoNormParameterTbl()`: Updates the impute norm parameter table. corresponding to the respective user defined attributes.

Usage:

```
pgu.delegate$updateTrafoNormParameterTbl(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `updateTrafoNormDataTbl()`**:** Updates the impute norm scaled data table. corresponding to the respective user defined attributes.

Usage:

```
pgu.delegate$updateTrafoNormDataTbl(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `imputeMissingsAnalyze()`**:** Calls the missing detection routine of the instance variable imputer on the instance variable normalizedData. Updates the instance class status.

Usage:

```
pgu.delegate$imputeMissingsAnalyze(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `updateImputeMissingsGraphic()`**:** Updates the plt.imputeMissingsSummary graphic.

Usage:

```
pgu.delegate$updateImputeMissingsGraphic(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `updateImputeMissingsStatisticsTbl()`**:** Updates the tbl.imputeMissingsStatistics table.

Usage:

```
pgu.delegate$updateImputeMissingsStatisticsTbl(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `updateImputeMissingsDistributionTbl()`**:** Updates the tbl.imputeMissingsDistribution table.

Usage:

```
pgu.delegate$updateImputeMissingsDistributionTbl(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `updateImputeMissingCharacteristicsGraphic()`**:** Updates the plt.imputeMissingsPairs graphic.

Usage:

```
pgu.delegate$updateImputeMissingCharacteristicsGraphic(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `updateImputeMissingsCharacteristicsTbl()`**:** Updates the tbl.imputeMissingsCharacteristics table.

Usage:

```
pgu.delegate$updateImputeMissingsCharacteristicsTbl(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `updateImputeMissingsDetailTbl()`**:** Updates the tbl.imputeDetectDetail table.

Usage:

```
pgu.delegate$updateImputeMissingsDetailTbl(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

```
                            #' @description
                            #' Updates the tbl.imputeDetectData table.
                            #' @param input
                            #' Pointer to shiny input
                            #' @param output
                            #' Pointer to shiny output
                            #' @param session
                            #' Pointer to shiny session
                        updateImputeMissingsDataTbl = function(input, output, session){
                               if(self$status$query(processName = "naDetected")){
                            output$tbl.imputeMissingsData <- DT::renderDataTable(
                                      self$filteredMetadata$rawData %>%
```

```
                dplyr::right_join(self$normalizedData$rawData, by = "Sample Name") %>%
                 format.data.frame(scientific = TRUE, digits = 4) %>%
                          DT::datatable(
                            extensions = "Buttons",
                            options = list(
                              scrollX = TRUE,
                              scrollY = '350px',
                              paging = FALSE,
                              dom = "Blfrtip",
                              buttons = list(list(
                                extend = 'csv',
                   filename = self$fileName$predict("imputationSiteDetectionData") %>%
                                   tools::file_path_sans_ext(),
                                text = "Download"
                              ))#buttons
                            )#options
                          )#DT::datatable
                        )#output
                       }#if
                       else{
              output$tbl.imputeMissingsData <- DT::renderDataTable(NULL)
                       }#else
                     }, #function
```

Method `updateImputeOutliersMethod()`**:** Updates the si.imputeOutliersMethod shiny widget.

Usage:

```
pgu.delegate$updateImputeOutliersMethod(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `updateImputeOutliersFeature()`**:** Updates the si.imputeOutliersFeature shiny widget.

Usage:

```
pgu.delegate$updateImputeOutliersFeature(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `updateImputeOutliersAlpha()`**:** Updates the ni.imputeOutliersAlpha shiny widget.

Usage:

```
pgu.delegate$updateImputeOutliersAlpha(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `updateImputeOutliersEpsilon()`: Updates the ni.imputeOutliersEpsilon shiny widget.

Usage:

```
pgu.delegate$updateImputeOutliersEpsilon(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `updateImputeOutliersMinSamples()`: Updates the ni.imputeOutliersMinSamples shiny widget.

Usage:

```
pgu.delegate$updateImputeOutliersMinSamples(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `updateImputeOutliersGamma()`: Updates the ni.imputeOutliersGamma shiny widget.

Usage:

```
pgu.delegate$updateImputeOutliersGamma(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `updateImputeOutliersNu()`: Updates the ni.imputeOutliersNu shiny widget.

Usage:

```
pgu.delegate$updateImputeOutliersNu(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `updateImputeOutliersCutoff()`: Updates the ni.imputeOutliersCutoff shiny widget.

Usage:

```
pgu.delegate$updateImputeOutliersCutoff(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output
`session` Pointer to shiny session

Method `updateImputeOutliersK()`**:** Updates the ni.imputeOutliersK shiny widget.

Usage:

```
pgu.delegate$updateImputeOutliersK(input, output, session)
```

Arguments:

`input` Pointer to shiny input
`output` Pointer to shiny output
`session` Pointer to shiny session

Method `updateImputeOutliersSeed()`**:** Updates the ni.imputeOutliersSeed shiny widget.

Usage:

```
pgu.delegate$updateImputeOutliersSeed(input, output, session)
```

Arguments:

`input` Pointer to shiny input
`output` Pointer to shiny output
`session` Pointer to shiny session

Method `updateImputeOutliersGui()`**:** Updates the gui.

Usage:

```
pgu.delegate$updateImputeOutliersGui(input, output, session)
```

Arguments:

`input` Pointer to shiny input
`output` Pointer to shiny output
`session` Pointer to shiny session

Method `resetImputeOutliersGui()`**:** Updates the gui.

Usage:

```
pgu.delegate$resetImputeOutliersGui(input, output, session)
```

Arguments:

`input` Pointer to shiny input
`output` Pointer to shiny output
`session` Pointer to shiny session

Method `imputeOutliersDetect()`**:** Calls the detectOutliers routine of the instance variable outliers on the instance variable normalizedData. Updates the instance class status.

Usage:

```
pgu.delegate$imputeOutliersDetect(input, output, session)
```

Arguments:

`input` Pointer to shiny input
`output` Pointer to shiny output

`session` Pointer to shiny session

Method `updateImputeOutliersGraphic()`: Updates the plt.outliersImputeSummary graphic.

Usage:

```
pgu.delegate$updateImputeOutliersGraphic(input, output, session)
```

Arguments:

`input` Pointer to shiny input
`output` Pointer to shiny output
`session` Pointer to shiny session

Method `updateImputeOutliersFeatureGraphic()`: Updates the plt.outliersImputeFeature graphic.

Usage:

```
pgu.delegate$updateImputeOutliersFeatureGraphic(input, output, session)
```

Arguments:

`input` Pointer to shiny input
`output` Pointer to shiny output
`session` Pointer to shiny session

Method `updateImputeOutliersFeatureTbl()`: Updates the numeric outlier feature table.

Usage:

```
pgu.delegate$updateImputeOutliersFeatureTbl(input, output, session)
```

Arguments:

`input` Pointer to shiny input
`output` Pointer to shiny output
`session` Pointer to shiny session

Method `updateImputeOutliersStatisticsTbl()`: Updates the numerical loq statistics analysis table

Usage:

```
pgu.delegate$updateImputeOutliersStatisticsTbl(input, output, session)
```

Arguments:

`input` Pointer to shiny input
`output` Pointer to shiny output
`session` Pointer to shiny session

Method `updateImputeOutliersDetailTbl()`: Updates the numerical outlier table.

Usage:

```
pgu.delegate$updateImputeOutliersDetailTbl(input, output, session)
```

Arguments:

`input` Pointer to shiny input
`output` Pointer to shiny output
`session` Pointer to shiny session

Method `updateImputeMutateFeature()`: Updates the si.imputeMutateFeature shiny widget.

Usage:

```
pgu.delegate$updateImputeMutateFeature(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `updateImputeMutateMethod()`: Updates the si.imputeMutateMethod shiny widget.

Usage:

```
pgu.delegate$updateImputeMutateMethod(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `updateImputeMutateNNeighbors()`: Updates the ni.imputeMutateNumberOfNeighbors shiny widget.

Usage:

```
pgu.delegate$updateImputeMutateNNeighbors(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `updateImputeMutatePredFrac()`: Updates the ni.imputeMutatePredFrac shiny widget.

Usage:

```
pgu.delegate$updateImputeMutatePredFrac(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `updateImputeMutateOutfluxThr()`: Updates the ni.imputeMutateOutfluxThr shiny widget.

Usage:

```
pgu.delegate$updateImputeMutateOutfluxThr(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `updateImputeMutateSeed()`: Updates the ni.imputeMutateSeed shiny widget.

Usage:

```
pgu.delegate$updateImputeMutateSeed(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `updateImputeMutateIterations()`: Updates the ni.imputeMutateIterations shiny widget.

Usage:

```
pgu.delegate$updateImputeMutateIterations(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `updateImputeMutateGui()`: Updates the gui.

Usage:

```
pgu.delegate$updateImputeMutateGui(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `resetImputeMutateGui()`: Resets the gui.

Usage:

```
pgu.delegate$resetImputeMutateGui(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `imputeMutateMutate()`: Calls the mutate imputation site routine of the instance variable imputer on the instance variable transformedData. Updates the instance class status.

Usage:

```
pgu.delegate$imputeMutateMutate(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `updateImputeFluxGraphic()`: Updates the plt.imputeMutateFlux graphic.

Usage:

```
pgu.delegate$updateImputeFluxGraphic(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `updateImputeMutateGraphic()`**:** Updates the plt.imputeMutateSummary graphic.

Usage:

```
pgu.delegate$updateImputeMutateGraphic(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `updateImputeMutateStatisticsTbl()`**:** Updates the tbl.imputeMutateStatistics table.

Usage:

```
pgu.delegate$updateImputeMutateStatisticsTbl(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `updateImputeMutateDistributionTbl()`**:** Updates the tbl.imputeMutateDistribution table.

Usage:

```
pgu.delegate$updateImputeMutateDistributionTbl(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `updateImputeMutateFeatureDetailGraphic()`**:** Updates the plt.imputeMutateFeatureDetail graphic.

Usage:

```
pgu.delegate$updateImputeMutateFeatureDetailGraphic(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `updateImputeMutateFeatureDetailTbl()`**:** Updates the tbl.imputeMutateFeatureDetail table.

Usage:

```
pgu.delegate$updateImputeMutateFeatureDetailTbl(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `updateImputeMutateDetailTbl()`**:** Updates the tbl.imputeMutateDetail table.

Usage:

```
pgu.delegate$updateImputeMutateDetailTbl(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `updateImputeMutateDataTbl()`**:** Updates the tbl.imputeMutateData table.

Usage:

```
pgu.delegate$updateImputeMutateDataTbl(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `validate()`**:** Calls the validate routine of the instance variable validator on the instance variables rawData and clenaedData. Updates the instance class status.

Usage:

```
pgu.delegate$validate(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `updateAnalysisValidationGui()`**:** Updates the si.analysisValidationFeature shiny widget.

Usage:

```
pgu.delegate$updateAnalysisValidationGui(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `updateAnalysisValidationGraphic()`**:** Updates the plt.analysisValidationFeature shiny widget.

Usage:

```
pgu.delegate$updateAnalysisValidationGraphic(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `updateAnalysisValidationTestTbl()`: Updtates the tbl.analysisValidationTest table.

Usage:

```
pgu.delegate$updateAnalysisValidationTestTbl(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `updateCentralMomentsOrgTbl()`: Updtates the tbl.centralMomentsOrg table.

Usage:

```
pgu.delegate$updateCentralMomentsOrgTbl(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `updateCentralMomentsImpTbl()`: Updtates the tbl.centralMomentsImp table.

Usage:

```
pgu.delegate$updateCentralMomentsImpTbl(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `updateCentralMomentsDeltaTbl()`: Updtates the tbl.centralMomentsDelta table.

Usage:

```
pgu.delegate$updateCentralMomentsDeltaTbl(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `updateCorrelationValidationScatterGraphic()`: Updtates the plt.correlationValidationScatter graphic.

Usage:

```
pgu.delegate$updateCorrelationValidationScatterGraphic(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `updateCorrelationValidationBoxPlotGraphic()`**:** Updtates the plt.correlationValidationBoxPlot graphic.

Usage:

```
pgu.delegate$updateCorrelationValidationBoxPlotGraphic(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `updateCorrelationValidationDeviationTbl()`**:** Updtates the tbl.correlationValidationDeviation table.

Usage:

```
pgu.delegate$updateCorrelationValidationDeviationTbl(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `updateCorrelationValidationDataTbl()`**:** Updtates the tbl.correlationValidationData table.

Usage:

```
pgu.delegate$updateCorrelationValidationDataTbl(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Method `exportFileName()`**:** Creates and returns an export filename.

Usage:

```
pgu.delegate$exportFileName(input, output, session)
```

Arguments:

`input` Pointer to shiny input

`output` Pointer to shiny output

`session` Pointer to shiny session

Returns: export filename (character)

Method `exportData()`**:** Exports the pguIMP analysis results

Usage:
```
pgu.delegate$exportData(input, file)
```
Arguments:
`input` Pointer to shiny input
`file` export filename (character)

Method `reportFileName()`**:** Creates and returns a report filename.
Usage:
```
pgu.delegate$reportFileName(input, output, session)
```
Arguments:
`input` Pointer to shiny input
`output` Pointer to shiny output
`session` Pointer to shiny session
Returns: export filename (character)

Method `writeReport()`**:** Exports a report on the pguIMP analysis in pdf format.
Usage:
```
pgu.delegate$writeReport(input, file)
```
Arguments:
`input` Pointer to shiny input
`file` export filename (character)

Method `hide_outdated_results()`**:** Updates the gui if analysis parameters change.
Usage:
```
pgu.delegate$hide_outdated_results(input, output, session)
```
Arguments:
`input` Pointer to shiny input
`output` Pointer to shiny output
`session` Pointer to shiny session

Method `update_help_html()`**:** Updates the gui if analysis parameters change.
Usage:
```
pgu.delegate$update_help_html(input, output, session)
```
Arguments:
`input` Pointer to shiny input
`output` Pointer to shiny output
`session` Pointer to shiny session

Method `clone()`**:** The objects of this class are cloneable with this method.
Usage:
```
pgu.delegate$clone(deep = FALSE)
```
Arguments:
`deep` Whether to make a deep clone.

Author(s)

Sebastian Malkusch, <malkusch@med.uni-frankfurt.de>

pgu.explorer *pgu.explorer*

Description

Visual exploration of the pguIMP dataset.

Format

R6::R6Class object.

Details

Pariwise anlysis of attributes from the pguIMP dataset. This object is used by the shiny based gui and is not for use in individual R-scripts!

Active bindings

`rawData` Returns the instance variable rawData (tibble::tibble)

`setRawData` Sets the instance variable rawData (tibble::tibble)

`abscissa` Returns the instance variable abscissa (character)

`setAbscissa` Sets the instance variable abscissa (character)

`ordinate` Returns the instance variable ordinate (character)

`setOrdinate` Sets the instance variable ordinate (character)

`abscissaStatistics` Returns the instance variable abscissaStatistics (character)

`ordinateStatistics` Returns the instance variable ordinateStatistics (character)

Methods

Public methods:

- `pgu.explorer$new()`
- `pgu.explorer$print()`
- `pgu.explorer$reset()`
- `pgu.explorer$fit()`
- `pgu.explorer$scatterPlot()`
- `pgu.explorer$abscissaBarPlot()`
- `pgu.explorer$abscissaBoxPlot()`
- `pgu.explorer$abscissaPlot()`
- `pgu.explorer$ordinateBarPlot()`
- `pgu.explorer$ordinateBoxPlot()`
- `pgu.explorer$ordinatePlot()`
- `pgu.explorer$clone()`

Method `new()`**:** Tests if the abscissa attribute is of type numeric.

Tests if the ordinate attribute is of type numeric.

Summarizes the numeric values of a vector.

Calculates the statistics of the abscissa values. Stores the result in the instance variable abscissaStatistics.

Calculates the statistics of the ordinate values. Stores the result in the instance variable ordinateStatistics.

Clears the heap and indicates that instance of `pgu.explorer` is removed from heap.

Creates and returns a new `pgu.explorer` object.

Usage:

```
pgu.explorer$new(data_df = "tbl_df")
```

Arguments:

`data_df` The data to be analyzed. (tibble::tibble)

Returns: A new `pgu.explorer` object. (pguIMP::pgu.optimizer)

Method `print()`**:** Prints instance variables of a `pgu.explorer` object.

Usage:

```
pgu.explorer$print()
```

Returns: string

Method `reset()`**:** Resets the instance of the pgu.explorer class

Usage:

```
pgu.explorer$reset(data_df = "tbl_df", abs = "character", ord = "character")
```

Arguments:

`data_df` The data to be analyzed. (tibble::tibble)

`abs` The abscissa attribute (character)

`ord` The ordinate attribute (character)

Method `fit()`**:** Calculates the abscissa and ordinate statistics

Usage:

```
pgu.explorer$fit()
```

Method `scatterPlot()`**:** Creates and returns a scatter plot abscissa and ordinate value pairs.

Usage:

```
pgu.explorer$scatterPlot()
```

Returns: Scatter plot (ggplot2::ggplot)

Method `abscissaBarPlot()`**:** Creates and returns a histogram from the abscissa values.

Usage:

```
pgu.explorer$abscissaBarPlot()
```

Returns: Bar plot (ggplot2::ggplot)

Method `abscissaBoxPlot()`**:** Creates and returns a box plot from the abscissa values.

Usage:

```
pgu.explorer$abscissaBoxPlot()
```

Returns: Box plot (ggplot2::ggplot)

Method `abscissaPlot()`**:** Creates and returns a compund graphical analysis of the abscissa values.

Usage:

```
pgu.explorer$abscissaPlot()
```

Returns: Compound plot (gridExtra::grid.arrange)

Method `ordinateBarPlot()`**:** Creates and returns a histogram from the ordinate values.

Usage:

```
pgu.explorer$ordinateBarPlot()
```

Returns: Bar plot (ggplot2::ggplot)

Method `ordinateBoxPlot()`**:** Creates and returns a box plot from the ordinate values.

Usage:

```
pgu.explorer$ordinateBoxPlot()
```

Returns: Box plot (ggplot2::ggplot)

Method `ordinatePlot()`**:** Creates and returns a compund graphical analysis of the ordinate values.

Usage:

```
pgu.explorer$ordinatePlot()
```

Returns: Compound plot (gridExtra::grid.arrange)

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
pgu.explorer$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Author(s)

Sebastian Malkusch, <malkusch@med.uni-frankfurt.de>

pgu.exporter *pgu.exporter*

Description

A class that writes the results of the pguIMP analysis to an Excel file.

Format

R6::R6Class object.

Details

Creates a download file name and saves a list of tibbles to an Excel file. Each tibble is written to a separate sheet. This object is used by the shiny based gui and is not for use in individual R-scripts!

Active bindings

`fileName` Returns the fileName. (character)

`setFileName` Set the fileName. (character)

`suffix` Returns the file suffix. (character)

Methods

Public methods:

- `pgu.exporter$new()`
- `pgu.exporter$finalize()`
- `pgu.exporter$print()`
- `pgu.exporter$extractSuffix()`
- `pgu.exporter$writeDataToExcel()`
- `pgu.exporter$clone()`

Method `new()`**:** Creates and returns a new `pgu.exporter` object.

Usage:

```
pgu.exporter$new()
```

Returns: A new `pgu.exporter` object. (pguIMP::pgu.exporter)

Method `finalize()`**:** Clears the heap and indicates if instance of `pgu.exporter` is removed from heap.

Usage:

```
pgu.exporter$finalize()
```

Method `print()`**:** Prints instance variables of a `pgu.exporter` object.

Usage:

```
pgu.exporter$print()
```

Returns: string

Method `extractSuffix()`**:** extracts the suffix from the fileName

Usage:

```
pgu.exporter$extractSuffix()
```

Method `writeDataToExcel()`**:** writes tibble to an excel file of the name fileName.

Usage:

```
pgu.exporter$writeDataToExcel(obj = "list")
```

Arguments:

`obj` A tibble or list of tibble. If obj is a list, each member will be written to a seperate sheet.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
pgu.exporter$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Author(s)

Sebastian Malkusch, <malkusch@med.uni-frankfurt.de>

`pgu.file` *pgu.fie*

Description

Handles file names for the pguIMP shiny web interface.

Format

R6::R6Class object.

Details

The class stores filenames and upload specifications for the pguIMP shiny web interface in its instance variables. This object is used by the shiny based gui and is not for use in individual R-scripts!

Active bindings

`uploadFileName` Returns the instance variable uploadFileName (character)

`fileName` Returns the instance variable fileName (character)

`baseName` Returns the instance variable baseName (character)

`folderName` Returns the instance variable folderName (character)

`suffix` Returns the instance variable suffix (character)

`exportSuffix` Returns the instance variable exportSuffix (character)

`timeString` Returns the instance variable timeString (character)

`sheetIndex` Returns the instance variable sheetIndex (numeric)

`separator` Returns the instance variable separator (character)

`skipRows` Returns the instance variable skipRows (numeric)

`columnNames` Returns the instance variable columnNames (logical)

`naChar` Returns the instance variable naChar (character)

Methods

Public methods:

- `pgu.file$new()`
- `pgu.file$print()`
- `pgu.file$reset()`
- `pgu.file$fit()`
- `pgu.file$predict()`
- `pgu.file$fit_predict()`
- `pgu.file$clone()`

Method `new()`**:** Clears the heap and indicates that instance of `pgu.file` is removed from heap.
Splits fileName and writes the results in the class' instance variables folderName, baseName, suffix.
Stores the current system time into the instance variable timeString.
Creates and returns a new object of type pgu.file.

Usage:

```
pgu.file$new(
  uploadFileName = "",
  fileName = "",
  sheetIndex = 1,
  separator = ",",
  skipRows = 0,
  columnNames = TRUE,
  naChar = "NA"
)
```

Arguments:

`uploadFileName` Name of uploaded file. (string)

`fileName` Actual file name. (string)
`sheetIndex` Index excel sheet to import. (integer)
`separator` Character for column separation. (character)
`skipRows` Number of rows to skip. (integer)
`columnNames` Indicates if the data source file has a columnNames. (logical)
`naChar` Character for missing values. (string)

Returns: A new pgu.file object. (pguIMP::pgu.file)

Method `print()`**:** Prints the instance variables of the object.

Usage:

```
pgu.file$print()
```

Returns: string

Method `reset()`**:** Resets the instance variables of the object.

Usage:

```
pgu.file$reset(
  uploadFileName = "",
  fileName = "",
  sheetIndex = 1,
  separator = ",",
  skipRows = 0,
  columnNames = TRUE,
  naChar = "NA"
)
```

Arguments:

`uploadFileName` Name of uploaded file. (string)
`fileName` Actual file name. (string)
`sheetIndex` Index excel sheet to import. (integer)
`separator` Character for column separation. (character)
`skipRows` Number of rows to skip. (integer)
`columnNames` Indicates if the data source file has a columnNames. (logical)
`naChar` Character for missing values. (string)

Method `fit()`**:** Extracts information about upload specifications from the instance variables.

Usage:

```
pgu.file$fit()
```

Method `predict()`**:** Predicts an export file name.

Usage:

```
pgu.file$predict(affix = "analysis")
```

Arguments:

`affix` User dedined file name affix. (string)

Returns: A file name. (string)

Method `fit_predict()`**:** Extracts information about upload specifications from the instance variables and predicts an export file name.

Usage:

```
pgu.file$fit_predict(affix = "analysis")
```

Arguments:

`affix` User dedined file name affix. (string)

Returns: A file name. (string)

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
pgu.file$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Author(s)

Sebastian Malkusch, <malkusch@med.uni-frankfurt.de>

`pgu.filter` *pgu.filter*

Description

Filter the pguIMP dataset.

Format

R6::R6Class object.

Details

The filtering is done by column and row indices. This object is used by the shiny based gui and is not for use in individual R-scripts!

Active bindings

`colIdx` Returns the instance variable colIdx (numeric)

`setColIdx` Sets the instance variable colIdx (numeric)

`rowIdx` Returns the instance variable rowIdx (numeric)

`setRowIdx` Sets the instance variable rowIdx (numeric)

Methods

Public methods:

- pgu.filter$new()
- pgu.filter$print()
- pgu.filter$reset()
- pgu.filter$predict()
- pgu.filter$clone()

Method `new()`**:** Resets the filter parameter colIdx to the full dataframe.
Resets the filter parameter rowIdx to the full dataframe.
Clears the heap and indicates that instance of pguIMP::pgu.filter is removed from heap.
Creates and returns a new pguIMP::pgu.filter object.

Usage:

```
pgu.filter$new(data_df = "tbl_df")
```

Arguments:

`data_df` The data to be analyzed. (tibble::tibble)

Returns: A new pguIMP::pgu.filter object. (pguIMP::pgu.filter)

Method `print()`**:** Prints instance variables of a pguIMP::pgu.filter object.

Usage:

```
pgu.filter$print()
```

Returns: string

Method `reset()`**:** Resets the filter parameter colIdx and rowIdx to the full dataframe.

Usage:

```
pgu.filter$reset(data_df = "tbl_df")
```

Arguments:

`data_df` The data to be analyzed. (tibble::tibble)

Method `predict()`**:** Filters and returns the given data frame.

Usage:

```
pgu.filter$predict(data_df = "tbl_df")
```

Arguments:

`data_df` The data to be analyzed. (tibble::tibble)

Returns: The filtered data frame (tibble::tibble)

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
pgu.filter$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Author(s)

Sebastian Malkusch, <malkusch@med.uni-frankfurt.de>

pgu.importer *pgu.importer*

Description

Handles the data import

Format

R6::R6Class object.

Details

Menages the import of the pguIMP dataset This object is used by the shiny based gui and is not for use in individual R-scripts!

Active bindings

`suffixes` Returns the instance variable suffixes (character)

Methods

Public methods:

- `pgu.importer$new()`
- `pgu.importer$finalize()`
- `pgu.importer$print()`
- `pgu.importer$suffixIsKnown()`
- `pgu.importer$importData()`
- `pgu.importer$importLoq()`
- `pgu.importer$importMetadata()`
- `pgu.importer$clone()`

Method `new()`**:** Creates and returns a new `pgu.importer` object.

Usage:

```
pgu.importer$new()
```

Returns: A new `pgu.importer` object. (pguIMP::pgu.importer)

Method `finalize()`**:** Clears the heap and indicates that instance of `pgu.importer` is removed from heap.

Usage:

```
pgu.importer$finalize()
```

Method `print()`**:** Prints instance variables of a `pgu.importer` object.

Usage:

```
pgu.importer$print()
```

Returns: string

Method `suffixIsKnown()`**:** Takes an instance of pgu.file and tests if the suffix is valid.

Usage:

```
pgu.importer$suffixIsKnown(obj = "pgu.file")
```

Arguments:

`obj` instance of pgu.file. (pguIMP::pgu.file)

Returns: test result (logical)

Method `importData()`**:** Takes an instance of pgu.file imports a dataset.

Usage:

```
pgu.importer$importData(obj = "pgu.file")
```

Arguments:

`obj` instance of pgu.file. (pguIMP::pgu.file)

Returns: data frame (tibble::tibble)

Method `importLoq()`**:** Takes an instance of pgu.file imports a loq dataset.

Usage:

```
pgu.importer$importLoq(obj = "pgu.file")
```

Arguments:

`obj` instance of pgu.file. (pguIMP::pgu.file)

Returns: data frame (tibble::tibble)

Method `importMetadata()`**:** Takes an instance of pgu.file imports a metadata dataset.

Usage:

```
pgu.importer$importMetadata(obj = "pgu.file")
```

Arguments:

`obj` instance of pgu.file. (pguIMP::pgu.file)

Returns: data frame (tibble::tibble)

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
pgu.importer$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Author(s)

Sebastian Malkusch, <malkusch@med.uni-frankfurt.de>

pgu.imputation *pgu.imputation*

Description

Analyses and substitutes imputation sites in a data set.

Format

R6::R6Class object.

Details

Analyses imputation sites in a data set. Replaces imputation sites by missing values and substitutes NAs by classical and ML-powered substitution algorithms. This object is used by the shiny based gui and is not for use in individual R-scripts!

Active bindings

`imputationStatistics` Returns the instance variable imputationStatistics. (tibble::tibble)

`imputationSites` Returns the instance variable imputationSites. (tibble::tibble)

`one_hot_df` Returns the positions of missings in one_hot encoding (tibble::tibble)

`imputationSiteDistribution` Returns the instance variable imputationSiteDistribution. (matrix)

`imputationAgentAlphabet` Returns the instance variable imputationagentAlphabet. (character)

`imputationAgent` Returns the instance variable imputationAgent. (character)

`setImputationAgent` Sets the instance variable imputationAgent. (character)

`nNeighbors` Returns the instance variable nNeighbors. (integer)

`setNNeighbors` Sets the instance variable nNeighbors. (integer)

`flux_df` Returns the instance variable flux_df (tibble::tibble)

`outflux_thr` Returns the instance variable outflux_thr. (numeric)

`setOutflux_thr` Sets the instance variable outflux_thr. (numeric)

`pred_frac` Returns the instance variable pred_frac. (numeric)

`setPred_frac` Sets the instance variable pred_frac. (numeric)

`pred_mat` Returns the instance variable pred_mat. (matrix)

`exclude_vec` Returns the instance variable exclude_vec (character)

`seed` Returns the instance variable seed. (numeric)

`setSeed` Sets the instance variable seed. (numeric)

`iterations` Returns the instance variable iterations. (numeric)

`setIterations` Sets the instance variable iterations. (numeric)

`amv` Returns the instance variable amv. (numeric)

`success` Returns the instance variable success. (logical)

Methods

Public methods:

- pgu.imputation$new()
- pgu.imputation$finalize()
- pgu.imputation$print()
- pgu.imputation$gatherImputationSites()
- pgu.imputation$gatherImputationSiteStatistics()
- pgu.imputation$gatherImputationSiteDistribution()
- pgu.imputation$insertImputationSites()
- pgu.imputation$one_hot()
- pgu.imputation$analyzeImputationSites()
- pgu.imputation$imputationSiteIdxByFeature()
- pgu.imputation$nanFeatureList()
- pgu.imputation$average_number_of_predictors()
- pgu.imputation$detectPredictors()
- pgu.imputation$handleImputationSites()
- pgu.imputation$imputeByMedian()
- pgu.imputation$imputeByMean()
- pgu.imputation$imputeByExpectationValue()
- pgu.imputation$imputeByMC()
- pgu.imputation$imputeByKnn()
- pgu.imputation$imputeByMice()
- pgu.imputation$imputeByM5P()
- pgu.imputation$imputationSiteHeatMap()
- pgu.imputation$featureBarPlot()
- pgu.imputation$featureBoxPlotWithSubset()
- pgu.imputation$featurePlot()
- pgu.imputation$fluxPlot()
- pgu.imputation$clone()

Method `new()`**:** Creates and returns a new `pgu.imputation` object.

Usage:

```
pgu.imputation$new(
  seed = 42,
  iterations = 4,
  imputationAgent = "none",
  nNeighbors = 3,
  pred_frac = 1,
  outflux_thr = 0.5
)
```

Arguments:

`seed` Initially sets the instance variable seed. Default is 42. (integer)

`iterations` Initially sets the instance variable iterations. Default is 4. (integer)

`imputationAgent` Initially sets the instance variable imputationAgent. Default is "none". Options are: ""none", "median", "mean", "expValue", "monteCarlo", "knn", "pmm", "cart", "randomForest", "M5P". (string)

`nNeighbors` Initially sets the instance variable nNeighbors. (integer)

`pred_frac` Initially sets the instance variable pred_frac. (numeric)

`outflux_thr` Initially sets the instance fariable outflux_thr

Returns: A new `pgu.imputation` object. (pguIMP::pgu.imputation)

Method `finalize()`**:** Clears the heap and indicates that instance of `pgu.imputation` is removed from heap.

Usage:

```
pgu.imputation$finalize()
```

Method `print()`**:** Prints instance variables of a `pgu.imputation` object.

Usage:

```
pgu.imputation$print()
```

Returns: string

Method `gatherImputationSites()`**:** Gathers imputation sites from pguIMP's missings and outliers class.

Usage:

```
pgu.imputation$gatherImputationSites(
  missings_df = "tbl_df",
  outliers_df = "tbl_df"
)
```

Arguments:

`missings_df` Dataframe comprising information about the imputation sites of pguIMP's missings class. (tibble::tibble)

`outliers_df` Dataframe comprising information about the imputation sites of pguIMP's outliers class. (tibble::tibble)

Method `gatherImputationSiteStatistics()`**:** Gathers statistical information about imputation sites The information is stored within the classes instance variable `imputationStatistics`

Usage:

```
pgu.imputation$gatherImputationSiteStatistics(data_df = "tbl_df")
```

Arguments:

`data_df` The data frame to be analyzed. (tibble::tibble)

Method `gatherImputationSiteDistribution()`**:** Gathers the distribution of imputation sites within the data frame. The information is stored within the classes instance variable imputationSiteDistribution.

Usage:

```
pgu.imputation$gatherImputationSiteDistribution(data_df = "tbl_df")
```

Arguments:

`data_df` The data frame to be analyzed. (tibble::tibble)

Returns: A data frame (tibble::tibble)

Method `insertImputationSites()`**:** Takes a dataframe, replaces the imputation sites indicated by the instance variable `imputationsites` by NA, and returns the mutated dataframe.

Usage:

```
pgu.imputation$insertImputationSites(data_df = "tbl_df")
```

Arguments:

`data_df` The data frame to be analyzed. (tibble::tibble)

Returns: A mutated version of data_df. (tibble::tibble)

Method `one_hot()`**:** Gathers statistical information about missing values in one hot format. The result is stored in the instance variable one_hot_df.

Usage:

```
pgu.imputation$one_hot(data_df = "tbl_df")
```

Arguments:

`data_df` The data frame to be analyzed. (tibble::tibble)

Method `analyzeImputationSites()`**:** Takes a dataframe and analyses the imputation sites.

Usage:

```
pgu.imputation$analyzeImputationSites(data_df = "tbl_df")
```

Arguments:

`data_df` The data frame to be analyzed. (tibble::tibble)

Method `imputationSiteIdxByFeature()`**:** Returns the position of an attribute's imputation sites within a data frame.

Usage:

```
pgu.imputation$imputationSiteIdxByFeature(featureName = "character")
```

Arguments:

`featureName` The attribute's name. (character)

Returns: The postion of the imputation sites. (numeric)

Method `nanFeatureList()`**:** Characterizes each row of the data frame as either `complete` or indicates which attribute are missing within the row. If multiple attributes' row entries are missing, the row is characterized by `multiple`.

Usage:

```
pgu.imputation$nanFeatureList(data_df = "tbl_df")
```

Arguments:

`data_df` The data frame to be analyzed. (tibble::tibble)

Returns: Vector of row characteristics. (character)

Method `average_number_of_predictors()`**:** Calculates the average number of predictors for a given dataframe and minpuc and mincor variables using the mice::quickpred routine.

Usage:

```
pgu.imputation$average_number_of_predictors(
  data_df = "tbl_df",
  minpuc = 0,
  mincor = 0.1
)
```

Arguments:

`data_df` The dataframe to be analyzed (tibble::tibble)

`minpuc` Specifies the minimum threshold for the proportion of usable cases. (numeric)

`mincor` Specifies the minimum threshold against which the absolute correlation in the dataframe is compared. (numeric)

Returns: Average_number_of_predictors. (numeric)

Method `detectPredictors()`**:** Identifies possible predictors for each feature. Analysis results are written to the instance variable pred_mat. Intermediate analysis results are an influx/outflux dataframe that is written to the instance variable flux_df and detect predictors and a list of features that is excluded from the search for possible predictors that is written to the instance variable exclude_vec.

Usage:

```
pgu.imputation$detectPredictors(data_df = "tbl_df")
```

Arguments:

`data_df` The dataframe to be analyzed. (tibble::tibble)

Method `handleImputationSites()`**:** Chooses a cleaning method based upon the instance variable `imputationAgent` and handles the imputation sites in the dataframe. Returns a cleaned data set. Display the progress if shiny is loaded.

Usage:

```
pgu.imputation$handleImputationSites(data_df = "tbl_df", progress = "Progress")
```

Arguments:

`data_df` The data frame to be analyzed. (tibble::tibble)

`progress` If shiny is loaded, the analysis' progress is stored within this instance of the shiny Progress class. (shiny::Progress)

Returns: Cleaned dataframe. (tibble:tibble)

Method `imputeByMedian()`**:** Substitutes imputation sites by the median of the respective attribute. Returns the cleaned dataframe. Display the progress if shiny is loaded.

Usage:

```
pgu.imputation$imputeByMedian(data_df = "tbl_df", progress = "Progress")
```

Arguments:

`data_df` The data frame to be analyzed. (tibble::tibble)

`progress` If shiny is loaded, the analysis' progress is stored in this instance of the shiny Progress class. (shiny::Progress)

Returns: Cleaned dataframe. (tibble:tibble)

Method `imputeByMean()`**:** Substitutes imputation sites by the aritmertic mean of the respective attribute. Returns the cleaned dataframe. Display the progress if shiny is loaded.

Usage:

```
pgu.imputation$imputeByMean(data_df = "tbl_df", progress = "Progress")
```

Arguments:

`data_df` The data frame to be analyzed. (tibble::tibble)

`progress` If shiny is loaded, the analysis' progress is stored in this instance of the shiny Progress class. (shiny::Progress)

Returns: Cleaned dataframe. (tibble:tibble)

Method `imputeByExpectationValue()`**:** Substitutes imputation sites by the expectation value of the respective attribute. Returns the cleaned dataframe. Display the progress if shiny is loaded.

Usage:

```
pgu.imputation$imputeByExpectationValue(
  data_df = "tbl_df",
  progress = "Progress"
)
```

Arguments:

`data_df` The data frame to be analyzed. (tibble::tibble)

`progress` If shiny is loaded, the analysis' progress is stored in this instance of the shiny Progress class. (shiny::Progress)

Returns: Cleaned dataframe. (tibble:tibble)

Method `imputeByMC()`**:** Substitutes imputation sites by values generated by a monte carlo simulation. The procedure runs several times as defined by the instance variable `iterations`. The run with the best result is identified and used for substitution. Returns the cleaned dataframe. Display the progress if shiny is loaded.

Usage:

```
pgu.imputation$imputeByMC(data_df = "tbl_df", progress = "Progress")
```

Arguments:

`data_df` The data frame to be analyzed. (tibble::tibble)

`progress` If shiny is loaded, the analysis' progress is stored in this instance of the shiny Progress class. (shiny::Progress)

Returns: Cleaned dataframe. (tibble:tibble)

Method `imputeByKnn()`**:** Substitutes imputation sites by predictions of a KNN analysis of the whole dataframe. Returns the cleaned dataframe. Display the progress if shiny is loaded.

Usage:

```
pgu.imputation$imputeByKnn(data_df = "tbl_df", progress = "Progress")
```

Arguments:

`data_df` The data frame to be analyzed. (tibble::tibble)

`progress` If shiny is loaded, the analysis' progress is stored in this instance of the shiny Progress class. (shiny::Progress)

Returns: Cleaned dataframe. (tibble:tibble)

Method `imputeByMice()`**:** Substitutes imputation sites by values generated by a different methods of the mice package. The procedure runs several times as defined by the instance variable `iterations`. The run with the best result is identified and used for substitution. Returns the cleaned dataframe. Display the progress if shiny is loaded.

Usage:

```
pgu.imputation$imputeByMice(data_df, progress = "Progress")
```

Arguments:

`data_df` The data frame to be analyzed. (tibble::tibble)

`progress` If shiny is loaded, the analysis' progress is stored in this instance of the shiny Progress class. (shiny::Progress)

Returns: Cleaned dataframe. (tibble:tibble)

Method `imputeByM5P()`**:** Substitutes imputation sites by predictions of a M5P tree trained on the whole dataframe. Returns the cleaned dataframe. Display the progress if shiny is loaded.

Usage:

```
pgu.imputation$imputeByM5P(data_df = "tbl_df", progress = "Progress")
```

Arguments:

`data_df` The data frame to be analyzed. (tibble::tibble)

`progress` If shiny is loaded, the analysis' progress is stored in this instance of the shiny Progress class. (shiny::Progress)

Returns: Cleaned dataframe. (tibble:tibble)

Method `imputationSiteHeatMap()`**:** Displays the distribution of missing values in form of a heatmap.

Usage:

```
pgu.imputation$imputationSiteHeatMap()
```

Returns: A heatmap plot. (ggplot2::ggplot)

Method `featureBarPlot()`**:** Displays the distribution of an attribute values as histogram.

Usage:

```
pgu.imputation$featureBarPlot(data_df = "tbl_df", feature = "character")
```

Arguments:

`data_df` dataframe to be analyzed. (tibble::tibble)

`feature` attribute to be shown. (character)

Returns: A histogram. (ggplot2::ggplot)

Method `featureBoxPlotWithSubset()`**:** Displays the distribution of an attribute's values as box plot.

Usage:

```
pgu.imputation$featureBoxPlotWithSubset(
  data_df = "tbl_df",
  feature = "character"
)
```

Arguments:

`data_df` dataframe to be analyzed. (tibble::tibble)

`feature` attribute to be shown. (character)

Returns: A box plot. (ggplot2::ggplot)

Method `featurePlot()`**:** Displays the distribution of an attribute's values as a composition of a box plot and a histogram.

Usage:

```
pgu.imputation$featurePlot(data_df = "tbl_df", feature = "character")
```

Arguments:

`data_df` dataframe to be analyzed. (tibble::tibble)

`feature` attribute to be shown. (character)

Returns: A composite plot. (ggplot2::ggplot)

Method `fluxPlot()`**:** Displays an influx/outflux plot

Usage:

```
pgu.imputation$fluxPlot()
```

Returns: A composite plot. (ggplot2::ggplot)

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
pgu.imputation$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Author(s)

Sebastian Malkusch, <malkusch@med.uni-frankfurt.de>

`pgu.limitsOfQuantification` *pgu.limitsOfQuantification*

Description

Handles values in the pguIMP dataset that exceed the limits of quantification. This object is used by the shiny based gui and is not for use in individual R-scripts!

Format

R6::R6Class object.

Details

more information

Active bindings

`loq` Returns the instance variable loq (tibble::tibble)

`setLoq` Sets the instance variable loq. (tibble::tibble)

`outliers` Returns instance variable outliers (tibble::tibble)

`lloqSubstituteAlphabet` Returns the instance variable lloqSubstititeAlphabet (character)

`lloqSubstituteAgent` Returns the instance variable lloqSubstituteAgent (character)

`setLloqSubstituteAgent` Sets the instance variable lloqSubstituteAgent. (character)

`uloqSubstituteAlphabet` Returns the instance variable uloqSubstititeAlphabet (character)

`uloqSubstituteAgent` Returns the instance variable uloqSubstituteAgent (character)

`setUloqSubstituteAgent` Sets the instance variable uloqSubstituteAgent. (character)

`naHandlingAlphabet` Returns the instance variable naHandlingAlphabet (character)

`naHandlingAgent` Returns the instance variable naHandlingAgentt (character)

`setNaHandlingAgent` Sets the instance variable naHandlingAgentt (character)

`loqStatistics` Returns the instance variable loqStatistics

Methods

Public methods:

- `pgu.limitsOfQuantification$new()`
- `pgu.limitsOfQuantification$print()`
- `pgu.limitsOfQuantification$reset()`
- `pgu.limitsOfQuantification$fit()`
- `pgu.limitsOfQuantification$predict()`
- `pgu.limitsOfQuantification$attribute_lloq()`
- `pgu.limitsOfQuantification$attribute_uloq()`
- `pgu.limitsOfQuantification$set_attribute_lloq()`
- `pgu.limitsOfQuantification$set_attribute_uloq()`
- `pgu.limitsOfQuantification$attribute_outliers()`
- `pgu.limitsOfQuantification$plot_loq_distribution()`
- `pgu.limitsOfQuantification$attribute_bar_plot()`
- `pgu.limitsOfQuantification$attribute_box_plot_with_subset()`
- `pgu.limitsOfQuantification$attribute_plot()`
- `pgu.limitsOfQuantification$clone()`

Method `new()`**:** Mutates outlier candidates characterized as below LLOQ based on user defined actions.

Mutates outlier candidates characterized as above ULOQ based on user defined actions.

Searches for outliers in the given data frame. If an outlier was found, it is appended to the instance variable outliers. Indicates if an outlier was found.

Extends the instance variable outliers by one entry.

Tests if the provided attributes are known to the class.

Resets the class' instance variable outliers

Calculates statistics of outlier appearance. Stores it into the instance variable loqStatistics

Resets the class' instance variable loqStatistics

Resets the class by a data frame comprising information about LOQs.

Resets the class by a vector of attribute names. The Attributes' LOQs are initially assigned to na.

Clears the heap and indicates that instance of `pgu.limitaOfQuantification` is removed from heap.

Creates and returns a new `pgu.limitsOfQuantification` object.

Usage:

```
pgu.limitsOfQuantification$new(attribute_names = "character")
```

Arguments:

`attribute_names` Vector of attribute names with to be analyzed by the loq object. (character)

Returns: A new `pgu.limitsOfQuantification` object. (pguIMP::pgu.limitsOfQuantification)

Method `print()`**:** Prints instance variables of a `pgu.limitsOfQuantification` object.

Usage:

```
pgu.limitsOfQuantification$print()
```

Returns: string

Method `reset()`**:** Resets the pguIMP::pgu.limitsOfQuantification object on the given parameters attribute_names and data_df

Usage:

```
pgu.limitsOfQuantification$reset(
  attribute_names = "character",
  data_df = "tbl_df"
)
```

Arguments:

`attribute_names` Vector of attribute names with to be analyzed by the loq object. (character)

`data_df` Dataframe comprising loq information. Feature names need to be 'attribute', 'LLOQ' and 'ULOQ'. (tibble::tibble)

Method `fit()`**:** Analyses the data dets for instances outside of the LOQ defined value interval.

Usage:

```
pgu.limitsOfQuantification$fit(data_df = "tbl_df")
```

Arguments:

`data_df` Dataframe to be analyzed

Method `predict()`**:** Mutates all outlier candidates based on user defined actions.

Usage:

```
pgu.limitsOfQuantification$predict(data_df = "tbl_df")
```

Arguments:

`data_df` The data to be analyzed. (tibble::tibble)

Returns: The revised data frame (tibble::tibble)

Method `attribute_lloq()`**:** Returns the attribute's specific lloq.

Usage:

```
pgu.limitsOfQuantification$attribute_lloq(attribute = "character")
```

Arguments:

`attribute` The attribute to be analyzed (character)

Returns: The attribute's lloq (numeric)

Method `attribute_uloq()`**:** Returns the attribute's specific uloq.

Usage:

```
pgu.limitsOfQuantification$attribute_uloq(attribute = "character")
```

Arguments:

`attribute` The attribute to be analyzed (character)

Returns: The attribute's uloq (numeric)

Method `set_attribute_lloq()`**:** sets the attribute's specific lloq to value.

Usage:

```
pgu.limitsOfQuantification$set_attribute_lloq(
  attribute = "character",
  value = NA
)
```

Arguments:

`attribute` The attribute to be updated (character)

`value` The value parsed to the attributes lloq (numeric)

Method `set_attribute_uloq()`**:** sets the attribute's specific uloq to value.

Usage:

```
pgu.limitsOfQuantification$set_attribute_uloq(
  attribute = "character",
  value = NA
)
```

Arguments:

`attribute` The attribute to be updated (character)

`value` The value parsed to the attributes lloq (numeric)

Method `attribute_outliers()`**:** Returns the detected outliers of a given attribute.

Usage:

```
pgu.limitsOfQuantification$attribute_outliers(attribute = "character")
```

Arguments:

`attribute` The attribute to be analyzed (character)

Returns: The attribute's outliers (tibble::tibble)

```
                    ####################
                    # data information #
                    ####################
                    #' @description
                #' Gathers and returns class information
                    dataInformation = function(){
                      self$loq %>%
                        dplyr::summarise_all(class) %>%
                      tidyr::gather(variable, class) %>%
                        return()
                    }, #function

                    ####################
                    # output functions #
                    ####################
                    #' @description
#' Merges dfData and dfMetadata and returns a fromatted data table.
                    #' @param dfData
                    #' The data to be analyzed.
                    #' (tibble::tibble)
                    #' @param dfMetadata
                  #' The data frame containing metadata.
                    #' (tibble::tibble)
                    #' @return
                    #' A formatted data table
                    #' (DT::datatable)
loqDataTable = function(dfData = "tbl_df", dfMetadata = "tbl_df"){
 options(htmlwidgets.TOJSON_ARGS = list(na = 'string'))
                    t <- NULL
                    featureNames <- colnames(dfData)
                    tryCatch(
  dfMerge <- dplyr::bind_cols(dfMetadata, dfData),
                      error = function(e){
                        print("error")
                        print(e)
                        dfMerge <- dfData
                      }#error
                    )#tryCatch
                  if(self$checkValidity(featureNames)){
                      t <- dfMerge %>%
        dplyr::mutate_if(is.numeric, round, 3) %>%
      DT::datatable(options = list(scrollX = TRUE,
                                   scrollY = '350px',
                                     paging = FALSE))
                    for (featureName in featureNames){
```

```
                featureOutlier <- self$outliers %>%
              dplyr::filter(feature == featureName) %>%
                 dplyr::mutate_if(is.numeric, round, 3)
                        if (nrow(featureOutlier) > 0){
                          t <- DT::formatStyle(t,
                                               featureName,
                        backgroundColor = DT::styleEqual(dfMerge %>%
                                                           dplyr::select(!!featureName
                                                           dplyr::slice(featureOutlier
                                                           unlist() %>%
                                                           round(digits = 3),
                                                         featureOutlier[["color"]]))
                          }#if
                        }#for
                      }#if
                      return(t)
                    }, #function

                    #' @description
#' Returns a formatted data table with comrising the information of a user
                    #' @param obj
                    #' The data to be analyzed.
                    #' (tibble::tibble)
                    #' @param feature
                    #' The attribute to be analyzed
                    #' (character)
                    #' @return
                    #' A formatted data table
                    #' (DT::datatable)
loqFeatureTable = function(obj = "tbl_df", feature = "character"){
 options(htmlwidgets.TOJSON_ARGS = list(na = 'string'))
                     t <- NULL
                     if(self$checkValidity(feature)){
                    featureOutlier <- self$outliers %>%
                  dplyr::filter(feature == !!feature) %>%
                   dplyr::mutate_if(is.numeric, round, 3)

                        dfFeature <- obj %>%
                   dplyr::mutate_if(is.numeric, round, 3)

                        print(dfFeature)

                        t <- dfFeature %>%
          DT::datatable(options = list(scrollX = TRUE,
                                       scrollY = '350px',
                                         paging = FALSE))
                        if (nrow(featureOutlier) > 0){
                          t <- DT::formatStyle(
```

```
                                t,
                                feature,
                backgroundColor = DT::styleEqual(dfFeature %>%
                                        dplyr::select(!!feature) %>%
                                        dplyr::slice(featureOutlier[["measurement
                                                 unlist() %>%
                                             round(digits = 3),
                                        featureOutlier[["color"]]))
                              }#if
                            }#if
                            return(t)
                          }, #function
```

Method `plot_loq_distribution()`**:** Creates a plot of the instance variable loqStatistics.

Usage:

```
pgu.limitsOfQuantification$plot_loq_distribution()
```

Returns: A plot. (ggplot2::ggplot)

Method `attribute_bar_plot()`**:** Creates a bar plot of a user defined attribute's value distribution. LOQs are indicated as dotted lines

Usage:

```
pgu.limitsOfQuantification$attribute_bar_plot(
  data_df = "tbl_df",
  attribute = "character"
)
```

Arguments:

`data_df` The data to be analyzed. (tibble::tibble)

`attribute` The attribute to be analyzed (character)

Returns: A bar plot. (ggplot2::ggplot)

Method `attribute_box_plot_with_subset()`**:** Creates a box plot of a user defined attribute's value distribution. LOQs are indicated as dotted lines

Usage:

```
pgu.limitsOfQuantification$attribute_box_plot_with_subset(
  data_df = "tbl_df",
  attribute = "character"
)
```

Arguments:

`data_df` The data to be analyzed. (tibble::tibble)

`attribute` The attribute to be analyzed (character)

Returns: A box plot. (ggplot2::ggplot)

Method `attribute_plot()`**:** Creates and returns a composite graphical analysis of the outlier analysis of a user defined attribute.

Usage:

```
pgu.limitsOfQuantification$attribute_plot(
  data_df = "tbl_df",
  attribute = "character"
)
```

Arguments:

`data_df` The data to be analyzed. (tibble::tibble)

`attribute` Attribute's name. (character)

Returns: Composite result plot. (gridExtra::grid.arrange)

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
pgu.limitsOfQuantification$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Author(s)

Sebastian Malkusch, <malkusch@med.uni-frankfurt.de>

`pgu.missings` *pgu.missings*

Description

Detects and substitutes missing values from data set.

Format

R6::R6Class object.

Details

Detects missing values in the transformed and normalized data set. This object is used by the shiny based gui and is not for use in individual R-scripts!

Active bindings

`imputationParameter` Returns the instance variable outliersParameter. (tibble::tibble)

`imputationSites` Returns the instance variable imputationSites. (tibble::tibble)

`one_hot_df` Returns the positions of missings in one_hot encoding (tibble::tibble)

`amv` Returns the instance variable amv. (numeric)

Methods

Public methods:

- pgu.missings$new()
- pgu.missings$finalize()
- pgu.missings$print()
- pgu.missings$resetImputationParameter()
- pgu.missings$featureIdx()
- pgu.missings$filterFeatures()
- pgu.missings$gatherMeasurements()
- pgu.missings$gatherMissings()
- pgu.missings$gatherExistings()
- pgu.missings$gatherFractionOfMissings()
- pgu.missings$gatherImputationStatistics()
- pgu.missings$one_hot()
- pgu.missings$detectImputationSites()
- pgu.missings$imputationSiteDistribution()
- pgu.missings$imputationSiteHeatMap()
- pgu.missings$clone()

Method `new()`**:** Creates and returns a new `pgu.missings` object.

Usage:

```
pgu.missings$new(data_df = "tbl_df")
```

Arguments:

`data_df` The data to be cleaned. (tibble::tibble)

Returns: A new `pgu.missings` object. (pguIMP::pgu.missings)

Method `finalize()`**:** Clears the heap and indicates that instance of `pgu.missings` is removed from heap.

Usage:

```
pgu.missings$finalize()
```

Method `print()`**:** Prints instance variables of a `pgu.missings` object.

Usage:

```
pgu.missings$print()
```

Returns: string

Method `resetImputationParameter()`**:** Resets instance variables and identifies missings in the normalized data set.

Usage:

```
pgu.missings$resetImputationParameter(data_df = "tbl_df")
```

Arguments:

`data_df` Dataframe to be analyzed. (tibble::tibble)

Method `featureIdx()`**:** Returns the position of an attribute within a data frame.

Usage:

```
pgu.missings$featureIdx(feature = "character")
```

Arguments:

`feature` The attribute's name. (character)

Returns: The postion of the attribute. (numeric)

Method `filterFeatures()`**:** Selects features cotaining missing values from a dataset.

Usage:

```
pgu.missings$filterFeatures(data_df = "tbl_df")
```

Arguments:

`data_df` Dataframe to be analyzed. (tibble::tibble)

Returns: The filtered data frame. (tibble::tibble)

Method `gatherMeasurements()`**:** Calculates the number of values of a vector.

Usage:

```
pgu.missings$gatherMeasurements(value = "numeric")
```

Arguments:

`value` A vector comprising numeric data. (numeric)

Returns: The lenght of the vector. (numeric)

Method `gatherMissings()`**:** Calculates the number of missing values of a vector.

Usage:

```
pgu.missings$gatherMissings(value = "numeric")
```

Arguments:

`value` A vector comprising numeric data. (numeric)

Returns: The number of missing in the vector. (numeric)

Method `gatherExistings()`**:** Calculates the number of existing values of a vector.

Usage:

```
pgu.missings$gatherExistings(value = "numeric")
```

Arguments:

`value` A vector comprising numeric data. (numeric)

Returns: The number of existing values in the vector. (numeric)

Method `gatherFractionOfMissings()`**:** Calculates the fraction of missing values of a vector.

Usage:

```
pgu.missings$gatherFractionOfMissings(value = "numeric")
```

Arguments:

`value` A vector comprising numeric data. (numeric)

Returns: The fraction of missing values in the vector. (numeric)

Method `gatherImputationStatistics()`**:** Gathers statistical information about missing values that are provided by the classes public `gather` functions. The information is stored within the classes instance variable `imputationParameter`

Usage:

```
pgu.missings$gatherImputationStatistics(data_df = "tbl_df")
```

Arguments:

`data_df` The data frame to be analyzed. (tibble::tibble)

Method `one_hot()`**:** Gathers statistical information about missing values in one hot format. The result is stored in the instance variable one_hot_df.

Usage:

```
pgu.missings$one_hot(data_df = "tbl_df")
```

Arguments:

`data_df` The data frame to be analyzed. (tibble::tibble)

Method `detectImputationSites()`**:** Detects missing values within the data frame and writes the to the instance variable `imputationsites`.

Usage:

```
pgu.missings$detectImputationSites(data_df = "tbl_df")
```

Arguments:

`data_df` The data frame to be analyzed. (tibble::tibble)

Method `imputationSiteDistribution()`**:** Numeric representation of the distribution of missing values within the data frame.

Usage:

```
pgu.missings$imputationSiteDistribution(data_df = "tbl_df")
```

Arguments:

`data_df` The data frame to be analyzed. (tibble::tibble)

Returns: A data frame (tibble::tibble) #' @description #' Merges the numeric attributes of the pguIMP data with its metadata. #' @param data_df #' The data frame to be analyzed. #' (tibble::tibble) #' @param metadata_df #' The corresponding metadata. #' (tibble::tibble) #' @return #' A data frame #' (tibble::tibble) mergeImputationSiteData = function(data_df = "tbl_df", metadata_df = "tbl_df") dfMerge <- data_df if(nrow(data_df) == nrow(metadata_df)) dfMerge <- dplyr::bind_cols(metadata_df, data_df) #if dfMerge %>% dplyr::filter_all(dplyr::any_vars(is.na(.))) %>% return() , #function

Method `imputationSiteHeatMap()`**:** Displays the distribution of missing values in form of a heatmap.

Usage:

```
pgu.missings$imputationSiteHeatMap()
```

Returns: A heatmap plot. (ggplot2::ggplot)

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
pgu.missings$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Author(s)

Sebastian Malkusch, <malkusch@med.uni-frankfurt.de>

`pgu.missingsCharacterizer` *pgu.missingsCharacterizer*

Description

A class that characterizes the origin of missing values.

Format

R6::R6Class object.

Details

A class that characterizes the origin of missing values. This object is used by the shiny based gui and is not for use in individual R-scripts!

Active bindings

`featureAlphabet` Returns the instance variable featureAlphabet. (character)

`featureAgent` Returns the instance variable featureAgent. (character)

`setFeatureAgent` Sets the instance variable featureAgent. (character)

`missingsCharacteristics_df` Returns the instance variable missingsCharacteristics_df. (tibble::tibble)

Methods

Public methods:

- `pgu.missingsCharacterizer$new()`
- `pgu.missingsCharacterizer$finalize()`
- `pgu.missingsCharacterizer$print()`
- `pgu.missingsCharacterizer$reset()`
- `pgu.missingsCharacterizer$analyze()`
- `pgu.missingsCharacterizer$plot_pair_dist()`
- `pgu.missingsCharacterizer$clone()`

Method `new()`**:** Creates and returns a new `pgu.missingsCharacterizer` object.

Usage:

```
pgu.missingsCharacterizer$new(data_df = "tbl_df")
```

Arguments:

`data_df` The data to be analyzed. (tibble::tibble)

Returns: A new `pgu.missingsCharacterizer` object. (pguIMP::pgu.missingsCharacterizer)

Method `finalize()`**:** Clears the heap and indicates if instance of `pgu.missingsCharacterizer` is removed from heap.

Usage:

```
pgu.missingsCharacterizer$finalize()
```

Method `print()`**:** Prints instance variables of a `pgu.missingsCharacterizer` object.

Usage:

```
pgu.missingsCharacterizer$print()
```

Returns: string

Method `reset()`**:** Takes a dataframe that will be analyzed using the analyze function and resets the instance variables.

Usage:

```
pgu.missingsCharacterizer$reset(data_df = "tbl_df")
```

Arguments:

`data_df` The data to be analyzed. (tibble::tibble)

Method `analyze()`**:** resets the instance variables and analyzes a dataframe.

Usage:

```
pgu.missingsCharacterizer$analyze(data_df = "tbl_df", progress = "Progress")
```

Arguments:

`data_df` The data to be analyzed. (tibble::tibble)

`progress` If shiny is loaded, the analysis' progress is stored within this instance of the shiny Progress class. (shiny::Progress)

Method `plot_pair_dist()`**:** Plots the analysis result.

Usage:

```
pgu.missingsCharacterizer$plot_pair_dist(data_df = "tbl_df")
```

Arguments:

`data_df` The data to be analyzed. (tibble::tibble)

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
pgu.missingsCharacterizer$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Author(s)

Sebastian Malkusch, <malkusch@med.uni-frankfurt.de>

pgu.model *pgu.model*

Description

Comprises a list of models for data manipulation.

Format

R6::R6Class object.

Details

Comprises a list of pgu.normDist objects and model parameters. These can be used to scale data. This object is used by the shiny based gui and is not for use in individual R-scripts!

Active bindings

`modelList` Returns a vector of pgu-normDist objects. (pgu.normDist)

`modelParameter` Returns a dataframe comrising model parameters. (tibble::tibble)

Methods

Public methods:

- `pgu.model$new()`
- `pgu.model$finalize()`
- `pgu.model$print()`
- `pgu.model$resetModelParameter()`
- `pgu.model$resetModelList()`
- `pgu.model$resetModel()`
- `pgu.model$setNormDist()`
- `pgu.model$featureIdx()`
- `pgu.model$fitFeature()`
- `pgu.model$fitData()`
- `pgu.model$logFitResultsFeature()`
- `pgu.model$logFailedFitResultsFeature()`
- `pgu.model$scaleNumeric()`
- `pgu.model$scaleData()`
- `pgu.model$rescaleNumeric()`
- `pgu.model$rescaleData()`
- `pgu.model$modelParameterData()`
- `pgu.model$modelParameterFeature()`
- `pgu.model$modelQualityData()`
- `pgu.model$modelQualityFeature()`

- pgu.model$fitResultData()
- pgu.model$fitResultFeature()
- pgu.model$testResultData()
- pgu.model$testResultFeature()
- pgu.model$plotModel()
- pgu.model$clone()

Method `new()`**:** Creates and returns a new `pgu.model` object.

Usage:

```
pgu.model$new(data = "tbl_df")
```

Arguments:

`data` The data to be analyzed. (tibble::tibble)

Returns: A new `pgu.model` object. (pguIMP::pgu.model)

Method `finalize()`**:** Clears the heap and indicates that instance of `pgu.model` is removed from heap.

Usage:

```
pgu.model$finalize()
```

Method `print()`**:** Prints instance variables of a `pgu.model` object.

Usage:

```
pgu.model$print()
```

Returns: string

Method `resetModelParameter()`**:** Resets instance variable `modelParameter`

Usage:

```
pgu.model$resetModelParameter(data = "tbl_df")
```

Arguments:

`data` Dataframe to be analyzed. (tibble::tibble)

Method `resetModelList()`**:** Resets instance variable `modelList`

Usage:

```
pgu.model$resetModelList(data = "tbl_df")
```

Arguments:

`data` Dataframe to be analyzed. (tibble::tibble)

Method `resetModel()`**:** Resets instance variable `modelList`. Resets instance variable `modelParameter`. Displays progress if shiny is loaded.

Usage:

```
pgu.model$resetModel(data = "tbl_df", progress = "Progress")
```

Arguments:

`data` Dataframe to be analyzed. (tibble::tibble)

`progress` If shiny is loaded, the analysis' progress is stored in this instance of the shiny Progress class. (shiny::Progress)

Method `setNormDist()`**:** Stores the information of a pgu.norDist object in an entry of the instance variable `modelList`

Usage:

```
pgu.model$setNormDist(data = "pgu.normDist", feature = "character")
```

Arguments:

`data` Instance of pgu.normDist (pguIMP::pgu.normDist)

`feature` Attribute corresponding to the pgu.normDist object data. (character)

Method `featureIdx()`**:** Returns the index of a pgu.normDist object wihtin the instance variable `modelParameter`.

Usage:

```
pgu.model$featureIdx(feature = "character")
```

Arguments:

`feature` Attribute's name. (character)

Returns: Index of attribute entry in dataframe (numeric)

Method `fitFeature()`**:** Runs the fit function of a pgu.normDist object at a user denied position within the instance variable modelList.

Usage:

```
pgu.model$fitFeature(feature = "character")
```

Arguments:

`feature` Attribute's name. (character)

Method `fitData()`**:** Loops through all attributes and calls the object's ftiFeature function. Displays progress if shiny is loaded.

Usage:

```
pgu.model$fitData(progress = "Progress")
```

Arguments:

`progress` If shiny is loaded, the analysis' progress is stored in this instance of the shiny Progress class. (shiny::Progress)

Method `logFitResultsFeature()`**:** Stores results from fitting procedure of a user defined attribute into the corrsponding attribute of instance variable `modelParameter`.

Usage:

```
pgu.model$logFitResultsFeature(feature = "character")
```

Arguments:

`feature` Attribute's name. (character)

Method `logFailedFitResultsFeature()`**:** Stores results from fitting procedure of a user defined attribute into the corrsponding attribute of instance variable `modelParameter` in case of a failed fitting routine.

Usage:

```
pgu.model$logFailedFitResultsFeature(feature = "character")
```

Arguments:

`feature` Attribute's name. (character)

Method `scaleNumeric()`**:** Scales numeric data based upon the model of a user defined attribute.

Usage:

```
pgu.model$scaleNumeric(value = "numeric", feature = "character")
```

Arguments:

`value` Numeric vector (numeric)

`feature` Attribute's name. (character)

Returns: scaled version of the given vector (numeric)

Method `scaleData()`**:** Scales a dataframe based upon a list of models stored in the instance variable modelList..

Usage:

```
pgu.model$scaleData(data = "tbl_df")
```

Arguments:

`data` Dataframe to be analyzed. (tibble::tibble)

Returns: scaled version of the given dataframe (tibble::tibble)

Method `rescaleNumeric()`**:** Re-scales numeric data based upon the model of a user defined attribute.

Usage:

```
pgu.model$rescaleNumeric(value = "numeric", feature = "character")
```

Arguments:

`value` Numeric vector (numeric)

`feature` Attribute's name. (character)

Returns: Re-scaled version of the given vector (numeric)

Method `rescaleData()`**:** Re-scales a dataframe based upon a list of models stored in the instance variable modelList..

Usage:

```
pgu.model$rescaleData(data = "tbl_df")
```

Arguments:

`data` Dataframe to be analyzed. (tibble::tibble)

Returns: Re-scaled version of the given dataframe (tibble::tibble)

Method `modelParameterData()`**:** Returns the model parameter (expectation value, standard deviation).

Usage:

```
pgu.model$modelParameterData()
```

Returns: Dataframe comprising model parameter. (tibble::tibble)

Method `modelParameterFeature()`**:** Returns the model parameter (expectation value, standard deviation) for a user deined attribute.

Usage:

```
pgu.model$modelParameterFeature(feature = "character")
```

Arguments:

`feature` Attribute's name. (character)

Returns: Dataframe comprising model parameter. (tibble::tibble)

Method `modelQualityData()`**:** Returns the model parameters connected to model quality.

Usage:

```
pgu.model$modelQualityData()
```

Returns: Dataframe comprising model parameter. (tibble::tibble)

Method `modelQualityFeature()`**:** Returns the model parameters connected to model quality for a user deined attribute.

Usage:

```
pgu.model$modelQualityFeature(feature = "character")
```

Arguments:

`feature` Attribute's name. (character)

Returns: Dataframe comprising model parameter. (tibble::tibble)

Method `fitResultData()`**:** Returns the model fit results.

Usage:

```
pgu.model$fitResultData()
```

Returns: Dataframe comprising model fit results. (tibble::tibble)

Method `fitResultFeature()`**:** Returns the model fit results for a user deined attribute.

Usage:

```
pgu.model$fitResultFeature(feature = "character")
```

Arguments:

`feature` Attribute's name. (character)

Returns: Dataframe comprising model fit results. (tibble::tibble)

Method `testResultData()`**:** Returns the hypothesis test results.

Usage:

```
pgu.model$testResultData()
```

Returns: Dataframe comprising the hypothesis test results. (tibble::tibble)

Method `testResultFeature()`**:** Returns the hypothesis test results. for a user deined attribute.

Usage:

```
pgu.model$testResultFeature(feature = "character")
```

Arguments:

`feature` Attribute's name. (character)

Returns: Dataframe comprising the hypothesis test results. (tibble::tibble)

Method `plotModel()`**:** Creates and returns a composite graphical analysis of the modeling procedure of a user defined attribute.

Usage:

```
pgu.model$plotModel(feature = "character")
```

Arguments:

`feature` Attribute's name. (character)

Returns: Composite result plot. (ggplot2::ggplot)

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
pgu.model$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Author(s)

Sebastian Malkusch, <malkusch@med.uni-frankfurt.de>

`pgu.normalizer` *pgu.normalizer*

Description

Normalization of data. Part of pguIMP.

Format

R6::R6Class object.

Details

Performs a data normalization in order to achieve a standardized version of the dataframe. This object is used by the shiny based gui and is not for use in individual R-scripts!

Active bindings

`normAgentAlphabet` Returns the instance variable normAgentAlphabt.

`normAgent` Returns the instance variable normAgent. (character)

`setNormAgent` Sets the instance variable normAgent. (character)

`features` Returns instance variable features. (character)

`normParameter` Returns the instance variable normParameter.

Methods

Public methods:

- pgu.normalizer$new()
- pgu.normalizer$finalize()
- pgu.normalizer$print()
- pgu.normalizer$detectNormParameter()
- pgu.normalizer$scale_data()
- pgu.normalizer$scale_minMax()
- pgu.normalizer$scale_minMax_numeric()
- pgu.normalizer$scale_mean()
- pgu.normalizer$scale_mean_numeric()
- pgu.normalizer$scale_zScore()
- pgu.normalizer$scale_zScore_numeric()
- pgu.normalizer$rescale_data()
- pgu.normalizer$rescale_minMax()
- pgu.normalizer$rescale_minMax_numeric()
- pgu.normalizer$rescale_mean()
- pgu.normalizer$rescale_mean_numeric()
- pgu.normalizer$rescale_zScore()
- pgu.normalizer$rescale_zScore_numeric()
- pgu.normalizer$featureBarPlot()
- pgu.normalizer$featureBoxPlotWithSubset()
- pgu.normalizer$featurePlot()
- pgu.normalizer$clone()

Method `new()`**:** Creates and returns a new `pgu.normalizer` object.

Usage:

```
pgu.normalizer$new(data_df = "tbl_df")
```

Arguments:

`data_df` The data to be analyzed. (tibble::tibble)

Returns: A new `pgu.normalizer` object. (pguIMP::pgu.normalizer)

Method `finalize()`**:** Clears the heap and indicates that instance of `pgu.normalizer` is removed from heap.

Usage:

```
pgu.normalizer$finalize()
```

Method `print()`**:** Prints instance variables of a `pgu.normalizer` object.

Usage:

```
pgu.normalizer$print()
```

Returns: string

Method `detectNormParameter()`**:** Resets instance variable `normParameter`

Usage:

```
pgu.normalizer$detectNormParameter(data_df = "tbl_df")
```

Arguments:

`data_df` Dataframe to be analyzed. (tibble::tibble)

Method `scale_data()`**:** Scales a tibble using the method defined by the instance variable normAgent

Usage:

```
pgu.normalizer$scale_data(data_df = "tbl_df")
```

Arguments:

`data_df` Dataframe to be scaled (tible::tibble)

Returns: A normalized version of the dataframe. (tibble::tibble)

Method `scale_minMax()`**:** Scales a tibble using min-max normalization

Usage:

```
pgu.normalizer$scale_minMax(data_df = "tbl_df")
```

Arguments:

`data_df` Dataframe to be scaled (tibble::tibble)

Returns: A min-max normalized version of the dataframe

Method `scale_minMax_numeric()`**:** Scales a numeric object using min-max normalization

Usage:

```
pgu.normalizer$scale_minMax_numeric(values = "numeric", feature = "character")
```

Arguments:

`values` Values to be scaled. Either a number or a vector (numeric)
`feature` Character to idtentify the proper normalization parameters. (character)

Returns: A min-max normalized version of the numeric object

Method `scale_mean()`**:** Scales a tibble using mean normalization

Usage:

```
pgu.normalizer$scale_mean(data_df = "tbl_df")
```

Arguments:

`data_df` Dataframe to be scaled. (tibble::tibble)

Returns: A mean normalized version of the dataframe

Method `scale_mean_numeric()`**:** Scales a numeric object using mean normalization

Usage:

```
pgu.normalizer$scale_mean_numeric(values = "numeric", feature = character)
```

Arguments:

`values` Values to be scaled. Either a number or a vector (numeric)
`feature` Character to idtentify the proper normalization parameters. (character)

Returns: A mean normalized version of the numeric object

Method `scale_zScore()`**:** Scales a tibble using z-score normalization

Usage:

```
pgu.normalizer$scale_zScore(data_df = "tbl_df")
```

Arguments:

`data_df` Dataframe to be scaled (tibble::tibble)

Returns: A z-score normalized version of the dataframe

Method `scale_zScore_numeric()`**:** Scales a numeric object using z-score normalization

Usage:

```
pgu.normalizer$scale_zScore_numeric(values = "numeric", feature = character)
```

Arguments:

`values` Values to be scaled. Either a number or a vector (numeric)

`feature` Character to idtentify the proper normalization parameters. (character)

Returns: A z-score normalized version of the numeric object

Method `rescale_data()`**:** Rescales a tibble using the method defined by the instance variable normAgent

Usage:

```
pgu.normalizer$rescale_data(data_df = "tbl_df")
```

Arguments:

`data_df` Normalized dataframe to be rescaled (tible::tibble)

Returns: A rescaled version of the normalized dataframe. (tibble::tibble)

Method `rescale_minMax()`**:** Rescales a tibble using min-max normalization

Usage:

```
pgu.normalizer$rescale_minMax(data_df = "tbl_df")
```

Arguments:

`data_df` Normalized dataframe to be rescaled (tibble::tibble)

Returns: A rescaled version of a min-max normalized dataframe

Method `rescale_minMax_numeric()`**:** Rescales a numeric object using min-max normalization

Usage:

```
pgu.normalizer$rescale_minMax_numeric(values = "numeric", feature = character)
```

Arguments:

`values` Normalized values to be rescaled. Either a number or a vector (numeric)

`feature` Character to idtentify the proper normalization parameters. (character)

Returns: Rescaled version of min-max normalized numeric object

Method `rescale_mean()`**:** Rescales a tibble using mean normalization

Usage:

```
pgu.normalizer$rescale_mean(data_df = "tbl_df")
```

Arguments:

`data_df` Normalized dataframe to be rescaled (tibble::tibble)

Returns: A rescaled version of a mean normalized dataframe

Method `rescale_mean_numeric()`**:** Rescales a numeric object using mean normalization

Usage:

```
pgu.normalizer$rescale_mean_numeric(values = "numeric", feature = character)
```

Arguments:

`values` Normalized values to be rescaled. Either a number or a vector (numeric)
`feature` Character to idtentify the proper normalization parameters. (character)

Returns: Rescaled version of mean normalized numeric object

Method `rescale_zScore()`**:** Rescales a tibble using z-score normalization

Usage:

```
pgu.normalizer$rescale_zScore(data_df = "tbl_df")
```

Arguments:

`data_df` Normalized dataframe to be rescaled (tibble::tibble)

Returns: A rescaled version of a z-score normalized dataframe

Method `rescale_zScore_numeric()`**:** Rescales a numeric object using z-score normalization

Usage:

```
pgu.normalizer$rescale_zScore_numeric(values = "numeric", feature = character)
```

Arguments:

`values` Normalized values to be rescaled. Either a number or a vector (numeric)
`feature` Character to idtentify the proper normalization parameters. (character)

Returns: Rescaled version of z-score normalized numeric object

Method `featureBarPlot()`**:** Displays the distribution of an attribute values as histogram.

Usage:

```
pgu.normalizer$featureBarPlot(data_df = "tbl_df", feature = "character")
```

Arguments:

`data_df` dataframe to be analyzed. (tibble::tibble)
`feature` attribute to be shown. (character)

Returns: A histogram. (ggplot2::ggplot)

Method `featureBoxPlotWithSubset()`**:** Displays the distribution of an attribute's values as box plot.

Usage:

```
pgu.normalizer$featureBoxPlotWithSubset(
  data_df = "tbl_df",
  feature = "character"
)
```

Arguments:

`data_df` dataframe to be analyzed. (tibble::tibble)

`feature` attribute to be shown. (character)

Returns: A box plot. (ggplot2::ggplot)

Method `featurePlot()`**:** Displays the distribution of an attribute's values as a composition of a box plot and a histogram.

Usage:

```
pgu.normalizer$featurePlot(data_df = "tbl_df", feature = "character")
```

Arguments:

`data_df` dataframe to be analyzed. (tibble::tibble)

`feature` attribute to be shown. (character)

Returns: A composite plot. (ggplot2::ggplot)

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
pgu.normalizer$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Author(s)

Sebastian Malkusch, <malkusch@med.uni-frankfurt.de>

`pgu.normDist` *pgu.normDist*

Description

Compares the distribution of a single attribute's values to normal distribution by using several statistic tests.

Format

R6::R6Class object.

Details

The distribution of a single value is tested for normality by Shapiro-Wilk test, Kolmogorov-Smirnov test, Anderson-Darling test. The expectation value and standard deviation of a normal distribution representing the data are determined by maximizing the log Likelihood with respect to the expectation value and standard deviation. This object is used by the shiny based gui and is not for use in individual R-scripts!

Active bindings

`featureName` Returns the instance variable featureName. (character)

`rawData` Returns the instance variable rawData. (tibble::tibble)

`setRawData` Sets the instance variable rawData. (tibble::tibble)

`histogram` Returns the instance variable histogram. (tibble::tibble)

`expMu` Returns the instance variable expMu. (numeric)

`expSigma` Returns the instance variable expSigma. (numeric)

`dataPoints` Returns the instance variable dataPoints. (numeric)

`logLikelihood` Returns the instance variable logLikelihood. (numeric)

`degOfFreedom` Returns the instance variable degOfFreedom. (numeric)

`n` Returns the instance variable n. (integer)

`bic` Returns the instance variable bic. (numeric)

`aic` Returns the instance variable aic. (numeric)

`aicc` Returns the instance variable aicc. (numeric)

`rmse` Returns the instance variable rmse. (numeric)

`fitSuccess` Returns the instance variable fitSuccess. (logical)

`testNames` Returns the instance variable testNames. (character)

`testParameterNames` Returns the instance variable testParameterNames. (character)

`alpha` Returns the instance variable alpha. (numeric)

`w.shapiro` Returns the instance variable w.shapiro. (numeric)

`p.shapiro` Returns the instance variable p.shapiro. (numeric)

`d.kolmogorow` Returns the instance variable d.kolmogorow. (numeric)

`p.kolmogorow` Returns the instance variable p.kolmogorow. (numeric)

`a.anderson` Returns the instance variable a.anderson. (numeric)

`p.anderson` Returns the instance variable p.anderson. (numeric)

Methods

Public methods:

- `pgu.normDist$new()`
- `pgu.normDist$finalize()`
- `pgu.normDist$print()`
- `pgu.normDist$resetNormDist()`
- `pgu.normDist$resetFail()`
- `pgu.normDist$optimize()`
- `pgu.normDist$createHistogram()`
- `pgu.normDist$normalQQData()`
- `pgu.normDist$test.shapiro()`
- `pgu.normDist$test.kolmogorow()`

- pgu.normDist$test.anderson()
- pgu.normDist$fitResult()
- pgu.normDist$testResult()
- pgu.normDist$testResultCompendium()
- pgu.normDist$plotHistogram()
- pgu.normDist$plotResiduals()
- pgu.normDist$plotResidualDist()
- pgu.normDist$plotRawResidualDist()
- pgu.normDist$plotRawDataDist()
- pgu.normDist$normalQQPlot()
- pgu.normDist$fit()
- pgu.normDist$clone()

Method `new()`**:** Creates and returns a new `pgu.normDist` object.

Usage:

```
pgu.normDist$new(data = "tbl_df")
```

Arguments:

`data` The data to be analyzed. (tibble::tibble)

Returns: A new `pgu.normDist` object. (pguIMP::pgu.normDist)

Method `finalize()`**:** Clears the heap and indicates that instance of `pgu.normDist` is removed from heap.

Usage:

```
pgu.normDist$finalize()
```

Method `print()`**:** Prints instance variables of a `pgu.normDist` object.

Usage:

```
pgu.normDist$print()
```

Returns: string

Method `resetNormDist()`**:** Resets instance variables

Usage:

```
pgu.normDist$resetNormDist(data = "tbl_df")
```

Arguments:

`data` Dataframe to be analyzed. (tibble::tibble)

Method `resetFail()`**:** Resets instance variables in case of a failed analysis.

Usage:

```
pgu.normDist$resetFail()
```

Method `optimize()`**:** Optimizes the logLikelihood between the data and a normal distribution with respect to the expectation value and standard deviation. The quality of the best model ist calculated subsequently.

Usage:

```
pgu.normDist$optimize()
```

Method `createHistogram()`**:** Creates a histogram from the instance variable `rawData`. The histogram is stored in the instance variable `histogram`.

Usage:

```
pgu.normDist$createHistogram()
```

Method `normalQQData()`**:** Performes a qq-analysis of the instance variable `rawData` The qq-analysis is stored in the attributes `sample_quantile` and `theoretical_quantile` of the instance variable `rawData`.

Usage:

```
pgu.normDist$normalQQData()
```

Method `test.shapiro()`**:** Performes Shapiro-Wilk's test for normality on the instance variable `rawData`. The test result is stored in the instance variable `w.shapiro`. The p-value of the test is stored in the instance variable `p.shapiro`

Usage:

```
pgu.normDist$test.shapiro()
```

Method `test.kolmogorow()`**:** Performes Kolmogorow-Smirnow's test for normality on the instance variable `rawData`. The test result is stored in the instance variable `d.kolmogorow`. The p-value of the test is stored in the instance variable `p.kolmogorow`

Usage:

```
pgu.normDist$test.kolmogorow()
```

Method `test.anderson()`**:** Performes Anderson-Darling's test for normality on the instance variable `rawData`. The test result is stored in the instance variable `a.anderson`. The p-value of the test is stored in the instance variable `p.anderson`

Usage:

```
pgu.normDist$test.anderson()
```

Method `fitResult()`**:** Returns the result of the classes optimize function in form of a formated string.

Usage:

```
pgu.normDist$fitResult()
```

Returns: String of the results of the fitting routine (character)

Method `testResult()`**:** Returns the result of the classes test functions in form of a formated string.

Usage:

```
pgu.normDist$testResult(testName = "Shapiro-Wilk")
```

Arguments:

`testName` Defines the test which result shall be returned. Can be of type:`Shapiro-Wilk`, `Kolmogorow-Smirnow` or `Anderson-Darling`. (character)

Returns: String of the results of the testing routine (character)

Method `testResultCompendium()`**:** Returns the result of the classes test functions `Shapiro-Wilk`, `Kolmogorow-Smirnow` and `Anderson-Darling` in form of a formated string. (character)

Usage:

```
pgu.normDist$testResultCompendium()
```

Returns: String of the results of the testing routine (character)

Method `plotHistogram()`**:** Displays the instance variable `histogram` in form of a bar plot and overlays the corresponding normal distribution.

Usage:

```
pgu.normDist$plotHistogram()
```

Returns: A bar plot. (ggplot2::ggplot)

Method `plotResiduals()`**:** Displays the residuals between the instance variable `histogram` and the corresponding normal distribution.

Usage:

```
pgu.normDist$plotResiduals()
```

Returns: A scatter plot. (ggplot2::ggplot)

Method `plotResidualDist()`**:** Displays the distribution of the residuals between the distribution of the instance variable `histogram` in form of a histogram.

Usage:

```
pgu.normDist$plotResidualDist()
```

Returns: A bar plot. (ggplot2::ggplot)

Method `plotRawResidualDist()`**:** Displays the distribution of the residuals between the distribution of the instance variable `rawData` in form of a histogram.

Usage:

```
pgu.normDist$plotRawResidualDist()
```

Returns: A bar plot. (ggplot2::ggplot)

Method `plotRawDataDist()`**:** Displays the distribution of the instance variable `rawData` in form of a histogram.

Usage:

```
pgu.normDist$plotRawDataDist()
```

Returns: A bar plot. (ggplot2::ggplot)

Method `normalQQPlot()`**:** Displays a qqplot of the instance variable `rawData`.

Usage:

```
pgu.normDist$normalQQPlot()
```

Returns: A qq-plot. (ggplot2::ggplot)

Method `fit()`**:** Runs the optimization process and performs all implemented quality controls. Additionally performs hypothesis tests for nromality.

Usage:

```
pgu.normDist$fit()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
pgu.normDist$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Author(s)

Sebastian Malkusch, <malkusch@med.uni-frankfurt.de>

`pgu.optimizer` *pgu.optimizer*

Description

Finds the transformation models that result in distributions that come closest to a normal distribution.

Format

R6::R6Class object.

Details

Analysis is performed individually on each attribute. This object is used by the shiny based gui and is not for use in individual R-scripts!

Active bindings

`features` Returns the instance variable features. (character)

`trafoAlphabet` Returns the instance variable trafoAlphabet. (character)

`setTrafoAlphabet` Sets the instance variable trafoAlphabet to data. (character)

`mirror` Returns the instance variable mirror (logical)

`setMirror` Sets the instance variable mirror to data (logical)

`optParameter` Returns the instance variable optParameter (tibble::tibble)

`optTypes` Returns the instance variable optTypes (tibble::tibble)

Methods

Public methods:

- pgu.optimizer$new()
- pgu.optimizer$finalize()
- pgu.optimizer$print()
- pgu.optimizer$resetFeatures()
- pgu.optimizer$resetOptParameter()
- pgu.optimizer$resetOptTypes()
- pgu.optimizer$resetOptimizer()
- pgu.optimizer$featureIdx()
- pgu.optimizer$modelParameterIsBigger()
- pgu.optimizer$modelParameterIsSmaller()
- pgu.optimizer$updateTrafoType()
- pgu.optimizer$updateMirrorLogic()
- pgu.optimizer$updateOptParameter()
- pgu.optimizer$optimize()
- pgu.optimizer$trafoAlpahbetTblDf()
- pgu.optimizer$clone()

Method `new()`**:** Creates and returns a new `pgu.optimizer` object.

Usage:

```
pgu.optimizer$new(data = "tbl_df")
```

Arguments:

`data` The data to be analyzed. (tibble::tibble)

Returns: A new `pgu.optimizer` object. (pguIMP::pgu.optimizer)

Method `finalize()`**:** Clears the heap and indicates that instance of `pgu.optimizer` is removed from heap.

Usage:

```
pgu.optimizer$finalize()
```

Method `print()`**:** Prints instance variables of a `pgu.optimizer` object.

Usage:

```
pgu.optimizer$print()
```

Returns: string

Method `resetFeatures()`**:** Extract the attribute names from the given data frame and stores them in the class' instance variable features,

Usage:

```
pgu.optimizer$resetFeatures(data = "tbl_df")
```

Arguments:

`data` The data to be analyzed. (tibble::tibble)

Method `resetOptParameter()`**:** Initializes the instance variable optParameter.

Usage:

```
pgu.optimizer$resetOptParameter()
```

Method `resetOptTypes()`**:** Initializes the instance variable optTypes.

Usage:

```
pgu.optimizer$resetOptTypes()
```

Method `resetOptimizer()`**:** Initializes the optimizer instance variables. Here, initialization defines a consecutive sequence of the class' functions: resetFeatures, setTrafoAlphabet, setMirror, resetOptParameter and resetOptTypes.

Usage:

```
pgu.optimizer$resetOptimizer(data = "tbl_df")
```

Arguments:

`data` The data to be analyzed. (tibble::tibble)

Method `featureIdx()`**:** Determines the numerical index of the column of an attribute based on the attribute name.

Usage:

```
pgu.optimizer$featureIdx(feature = "character")
```

Arguments:

`feature` The attribute's name. (character)

Returns: The attributes column index. (numeric)

Method `modelParameterIsBigger()`**:** Compares a model parameter to a reference parameter and tests, if the model parameter is bigger.

Usage:

```
pgu.optimizer$modelParameterIsBigger(
  modelParameter = "numeric",
  referenceParameter = "numeric"
)
```

Arguments:

`modelParameter` The model parameter (numeric)

`referenceParameter` The reference parameter (numeric)

Returns: Test Result (logical)

Method `modelParameterIsSmaller()`**:** Compares a model parameter to a reference parameter and tests, if the model parameter is smaller.

Usage:

```
pgu.optimizer$modelParameterIsSmaller(
  modelParameter = "numeric",
  referenceParameter = "numeric"
)
```

Arguments:

`modelParameter` The model parameter (numeric)

`referenceParameter` The reference parameter (numeric)

Returns: Test Result (logical)

Method `updateTrafoType()`**:** Takes an instance of the pgu.transfromator class and sets the transformation type to a user defined value.

Usage:

```
pgu.optimizer$updateTrafoType(
  transformator = "pgu.transformator",
  type = "character"
)
```

Arguments:

`transformator` An instance of the pgu.transformator class (pguIMP::pgu.transformator)

`type` A transfromation type (character)

Returns: An updated instance of the pgu.transformator class (pguIMP::pgu.transformator)

Method `updateMirrorLogic()`**:** Takes an instance of the pgu.transfromator class and sets the mirrorLogic parameter to a user defined value.

Usage:

```
pgu.optimizer$updateMirrorLogic(
  transformator = "pgu.transformator",
  logic = "logical"
)
```

Arguments:

`transformator` An instance of the pgu.transformator class (pguIMP::pgu.transformator)

`logic` The mirrorLogic parameter (logic)

Returns: An updated instance of the pgu.transformator class (pguIMP::pgu.transformator)

Method `updateOptParameter()`**:** Takes an instance of the pgu.model class and analyzes it. Keeps track of the optimal model parameters during optimization and stores them in the instance variables optTypes and optParameter.

Usage:

```
pgu.optimizer$updateOptParameter(
  model = "pgu.model",
  type = "character",
  logic = "character"
)
```

Arguments:

`model` An instance of the pgu.model class (pguIMP::pgu.model)

`type` A transfromation type (character)

`logic` The mirrorLogic parameter (logic)

Method `optimize()`**:** Permutates all possible variations of data transfromations and iterates through them. Analysis the optimal transformation parameters for each attribute in the data frame and stores them in the instance variables optParameter, optTypes.

Usage:

```
pgu.optimizer$optimize(data = "tbl_df", progress = "Progress")
```

Arguments:

`data` The data frame to be analyzed. (tibble::tibble)

`progress` If shiny is loaded, the analysis' progress is stored within this instance of the shiny Progress class. (shiny::Progress)

Method `trafoAlpahbetTblDf()`**:** Returns information on the optimization progress

Usage:

```
pgu.optimizer$trafoAlpahbetTblDf()
```

Returns: The data frame comprizing analysis information. (tibble::tibble)

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
pgu.optimizer$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Author(s)

Sebastian Malkusch, <malkusch@med.uni-frankfurt.de>

`pgu.outliers` *pgu.outliers*

Description

Detects and replaces possible outliers from data set.

Format

R6::R6Class object.

Details

Performes Grubb's test for outliers to detect outliers in the normalized and Z-score transfromed data set. Replace missing values with substitutes by classical and AI-powerd substitution algorithms. For this purpose outliers are handled as imputation sites.

Active bindings

`outliersParameter` Returns the instance variable outliersParameter. (tibble::tibble)

`outliers` Returns the instance variable outliers. (tibble::tibble)

`one_hot_df` Returns the positions of missings in one_hot encoding (tibble::tibble)

`outliersStatistics` Returns the instance variable outliersStatistics. (tibble::tibble)

`outliersAgentAlphabet` Returns the instance variable of outliersAgentAlphabet (character)

`outliersAgent` Returns the instance variable outliersAgent. (character)

`setOutliersAgent` Sets the instance variable outliersAgent. (character)

`featureData` Returns the instance variable featureData. (numeric)

`alpha` Returns the instance variable alpha. (numeric)

`setAlpha` Set the instance variable alpha. (numeric)

`epsilon` Returns the instance variable epsilon. (numeric)

`setEpsilon` Set the instance variable epsilon. (numeric)

`minSamples` Returns the instance variable minSamples. (integer)

`setMinSamples` Set the instance variable minSamples. (integer)

`gamma` Returns the instance variable gamma. (numeric)

`setGamma` Set the instance variable gamma. (numeric)

`nu` Returns the instance variable nu. (numeric)

`setNu` Set the instance variable nu. (numeric)

`k` Returns the instance variable k (integer)

`setK` Sets the instance variable k. (integer)

`cutoff` Returns the instance variable cutoff. (numeric)

`setCutoff` Sets the instance variable cutoff. (numeric)

`seed` Returns the instance variable seed. (integer)

`setSeed` Set the instance variable seed. (integer)

Methods

Public methods:

- `pgu.outliers$new()`
- `pgu.outliers$finalize()`
- `pgu.outliers$print()`
- `pgu.outliers$resetOutliers()`
- `pgu.outliers$filterFeatures()`
- `pgu.outliers$checkFeatureValidity()`
- `pgu.outliers$detectOutliersParameter()`
- `pgu.outliers$outliersFeatureList()`
- `pgu.outliers$featureOutlier()`
- `pgu.outliers$one_hot()`

- pgu.outliers$detectOutliers()
- pgu.outliers$detectByGrubbs()
- pgu.outliers$grubbs_numeric()
- pgu.outliers$detectByDbscan()
- pgu.outliers$dbscan_numeric()
- pgu.outliers$detectBySvm()
- pgu.outliers$svm_numeric()
- pgu.outliers$detectByKnn()
- pgu.outliers$knn_numeric()
- pgu.outliers$setImputationSites()
- pgu.outliers$calcOutliersStatistics()
- pgu.outliers$outlierTable()
- pgu.outliers$plotOutliersDistribution()
- pgu.outliers$featureBarPlot()
- pgu.outliers$featureBoxPlotWithSubset()
- pgu.outliers$featurePlot()
- pgu.outliers$clone()

Method `new()`**:** Creates and returns a new `pgu.outliers` object.

Usage:

```
pgu.outliers$new(
  data_df = "tbl_df",
  alpha = 0.05,
  epsilon = 0.1,
  minSamples = 4,
  gamma = 0.05,
  nu = 0.1,
  k = 4,
  cutoff = 0.99,
  seed = 42
)
```

Arguments:

`data_df` The data to be cleaned. (tibble::tibble)

`alpha` Initial definition of the instance variable alpha. (numeric)

`epsilon` Initial definition of the instance variable epsilon. (numeric)

`minSamples` Initial definition of the instance variable minSamples. (integer)

`gamma` Initial definition of the instance variable gamma. (numeric)

`nu` Initial definition of the instance variable nu. (numeric)

`k` Initial definition of the instance variable k. (integer)

`cutoff` Initial definition of the instance variable cutoff. (numeric)

`seed` Initial definition of the instance variable seed. (integer)

Returns: A new `pgu.outliers` object. (pguIMP::pgu.outliers)

Method `finalize()`**:** Clears the heap and indicates that instance of `pgu.outliers` is removed from heap.

Usage:

```
pgu.outliers$finalize()
```

Method `print()`**:** Prints instance variables of a `pgu.outliers` object.

Usage:

```
pgu.outliers$print()
```

Returns: string

Method `resetOutliers()`**:** Resets instance variables and performes Grubb's test for outliers to detect outliers in the normalized and Z-score transfromed data set. Progresse is indicated by the progress object passed to the function.

Usage:

```
pgu.outliers$resetOutliers(data_df = "tbl_df")
```

Arguments:

`data_df` Dataframe to be analyzed. (tibble::tibble)

Method `filterFeatures()`**:** Filters attributes from the given dataframe that are known to the class.

Usage:

```
pgu.outliers$filterFeatures(data_df = "tbl_df")
```

Arguments:

`data_df` Dataframe to be filtered. (tibble::tibble)

Returns: A filterd dataframe. (tibble::tibble)

Method `checkFeatureValidity()`**:** Checks if the feature consists of a sufficient number of instances.

Usage:

```
pgu.outliers$checkFeatureValidity(data_df = "tbl_df", feature = "character")
```

Arguments:

`data_df` Dataframe to be analyzed (tibble::tibble)

`feature` The attribute to be analyzed. (character)

Method `detectOutliersParameter()`**:** determines the outliers parameter by analyzing the tibble data_df and the instance variable outliers. Results are stored to instance variable outliersParameter.

Usage:

```
pgu.outliers$detectOutliersParameter(data_df = "tbl_df")
```

Arguments:

`data_df` Dataframe to be analyzed. (tibble::tibble)

Method `outliersFeatureList()`**:** Characterizes each row of the data frame as either `complete` or indicates which attribute has been identified as an outlier within the row. If multiple attributes' row entries were identified as outliers, the row is characterized by `multiple`.

Usage:

```
pgu.outliers$outliersFeatureList(data_df = "tbl_df")
```

Arguments:

`data_df` The data frame to be analyzed. (tibble::tibble)

Returns: Vector of row characteristics. (character)

Method `featureOutlier()`**:** Returns the detected outliers of a given attribute.

Usage:

```
pgu.outliers$featureOutlier(feature = "character")
```

Arguments:

`feature` The attribute to be analyzed (character)

Returns: The attribute's outliers (tibble::tibble)

Method `one_hot()`**:** Gathers statistical information about missing values in one hot format. The result is stored in the instance variable one_hot_df.

Usage:

```
pgu.outliers$one_hot(data_df = "tbl_df")
```

Arguments:

`data_df` The data frame to be analyzed. (tibble::tibble)

Method `detectOutliers()`**:** Chooses a method for identification of anomalies based upon the instance variable `outliersAgent` Detects anomalies in a data frame by one-dimensional analysis of each feature.

Usage:

```
pgu.outliers$detectOutliers(data_df = "tbl_df", progress = "Progress")
```

Arguments:

`data_df` Data frame to be analyzed. (tibble::tibble)

`progress` If shiny is loaded, the analysis' progress is stored in this instance of the shiny Progress class. (shiny::Progress)

Method `detectByGrubbs()`**:** Identifies anomalies in the data frame based on Grubb's test. Iterates over the whole data frame. Calls the object's public function `grubbs_numeric` until no more anomalies are identified. The threshold for anomaly detection is defined in the instance variable `alpha`. Display the progress if shiny is loaded.

Usage:

```
pgu.outliers$detectByGrubbs(data_df = "tbl_df", progress = "Progress")
```

Arguments:

`data_df` Data frame to be analyzed. (tibble::tibble)

`progress` If shiny is loaded, the analysis' progress is stored in this instance of the shiny Progress class. (shiny::Progress)

Method `grubbs_numeric()`**:** Performs Grubb's test for anomalies to detect a single outlier in the provided attributes data. If an outlier is found, it is added to the instance variable `outliers`. The threshold for anomaly detection is difined in the instance variable `alpha`. The function indicates a find by a positive feedback.

Usage:

```
pgu.outliers$grubbs_numeric(data_df = "tbl_df", feature = "character")
```

Arguments:

`data_df` The data frame to be analyzed. (tibble::tibble)

`feature` The attribute within the data frame to be analyzed.

Returns: Feedback if an outlier was found. (logical)

Method `detectByDbscan()`**:** Identifies anomalies in the data frame based on DBSCAN. Iterates over the whole data frame. Calls the object's public function `dbscan_numeric` until all features are analyzed. The cluster hyper parameter are defined in the instance variables `epsilon` and `minSamples`. The results of the `dbscan_numeric` routine are added to the instance variable `outliers`. Display the progress if shiny is loaded.

Usage:

```
pgu.outliers$detectByDbscan(data_df = "tbl_df", progress = "Progress")
```

Arguments:

`data_df` Data frame to be analyzed. (tibble::tibble)

`progress` If shiny is loaded, the analysis' progress is stored in this instance of the shiny Progress class. (shiny::Progress)

Method `dbscan_numeric()`**:** Identifies anomalies in a single feature of a data frame based on DBSCAN. The cluster hyperparameter are defined in the instance variables `epsilon` and `minSamples`. Display the progress if shiny is loaded.

Usage:

```
pgu.outliers$dbscan_numeric(data_df = "tbl_df", feature = "character")
```

Arguments:

`data_df` Data frame to be analyzed. (tibble::tibble)

`feature` Feature to be analyzed (character)

`progress` If shiny is loaded, the analysis' progress is stored in this instance of the shiny Progress class. (shiny::Progress)

Returns: A data frame comprising the information about detected anomalies of the feature. (tibble::tibble)

Method `detectBySvm()`**:** Identifies anomalies in the data frame based on one class SVM. Iterates over the whole data frame. Calls the object's public function `svm_numeric` until all features are analyzed. The cluster hyper parameter are defined in the instance variables `gamma` and `nu`. The results of the `svm_numeric` routine are added to the instance variable `outliers`. Display the progress if shiny is loaded.

Usage:

```
pgu.outliers$detectBySvm(data_df = "tbl_df", progress = "Process")
```

Arguments:

`data_df` Data frame to be analyzed. (tibble::tibble)

`progress` If shiny is loaded, the analysis' progress is stored in this instance of the shiny Progress class. (shiny::Progress)

Method `svm_numeric()`**:** Identifies anomalies in a single feature of a data frame based on one class SVM. The cluster hyperparameter are defined in the instance variables `gamma` and `nu`. Display the progress if shiny is loaded.

Usage:

```
pgu.outliers$svm_numeric(data_df = "tbl_df", feature = "character")
```

Arguments:

`data_df` Data frame to be analyzed. (tibble::tibble)

`feature` Feature to be analyzed (character)

`progress` If shiny is loaded, the analysis' progress is stored in this instance of the shiny Progress class. (shiny::Progress)

Returns: A data frame comprising the information about detected anomalies of the feature. (tibble::tibble)

Method `detectByKnn()`**:** Identifies anomalies in the data frame based on knnO. Iterates over the whole data frame. Calls the object's public function `svm_numeric` until all features are analyzed. The cluster hyper parameter are defined in the instance variables `alpha` and `minSamples`. The results of the `knn_numeric` routine are added to the instance variable `outliers`. Display the progress if shiny is loaded.

Usage:

```
pgu.outliers$detectByKnn(data_df = "tbl_df", progress = "Process")
```

Arguments:

`data_df` Data frame to be analyzed. (tibble::tibble)

`progress` If shiny is loaded, the analysis' progress is stored in this instance of the shiny Progress class. (shiny::Progress)

Method `knn_numeric()`**:** Identifies anomalies in a single feature of a data frame based on knnO. The cluster hyperparameter are defined in the instance variables `alpha` and `minSmaples`. Display the progress if shiny is loaded.

Usage:

```
pgu.outliers$knn_numeric(data_df = "tbl_df", feature = "character")
```

Arguments:

`data_df` Data frame to be analyzed. (tibble::tibble)

`feature` Feature to be analyzed (character)

`progress` If shiny is loaded, the analysis' progress is stored in this instance of the shiny Progress class. (shiny::Progress)

Returns: A data frame comprising the information about detected anomalies of the feature. (tibble::tibble)

Method `setImputationSites()`**:** Replaces the detected anomalies of a user provided data frame with `NA` for further imputation routines.

Usage:

```
pgu.outliers$setImputationSites(data_df = "tbl_df")
```

Arguments:

`data_df` Data frame to be mutated. (tibble::tibble)

Returns: A data frame with anomalies replaced by `NA`. (tibble::tibble)

Method `calcOutliersStatistics()`**:** Calculates the statistics on the previously performed outlier detection analysis and stores the results in the instance variable `outliersStatistcs`.

Usage:

```
pgu.outliers$calcOutliersStatistics(data_df = "tbl_df")
```

Arguments:

`data_df` The data frame to be analyzed. (tibble::tibble)

Method `outlierTable()`**:** Creates a datatable with substituted outliers highlightes by colored background.

Usage:

```
pgu.outliers$outlierTable(data_df = "tbl_df")
```

Arguments:

`data_df` The data frame to be analyzed. (tibble::tibble)

Returns: A colored datatable (DT::datatable)

Method `plotOutliersDistribution()`**:** Displays the occurrence of outlier candidates per attribute as bar plot.

Usage:

```
pgu.outliers$plotOutliersDistribution()
```

Returns: A bar plot. (ggplot2::ggplot)

Method `featureBarPlot()`**:** Displays the distribution of an attribute's values as histogram.

Usage:

```
pgu.outliers$featureBarPlot(data_df = "tbl_df", feature = "character")
```

Arguments:

`data_df` dataframe to be analyzed. (tibble::tibble)

`feature` attribute to be shown. (character)

Returns: A histogram. (ggplot2::ggplot)

Method `featureBoxPlotWithSubset()`**:** Displays the distribution of an attribute's vlues as box plot.

Usage:

```
pgu.outliers$featureBoxPlotWithSubset(
  data_df = "tbl_df",
  feature = "character"
)
```

Arguments:

`data_df` dataframe to be analyzed. (tibble::tibble)

`feature` attribute to be shown. (character)

Returns: A box plot. (ggplot2::ggplot)

Method `featurePlot()`**:** Displays the distribution of an attribute's values as a composition of a box plot and a histogram.

Usage:

```
pgu.outliers$featurePlot(data_df = "tbl_df", feature = "character")
```

Arguments:

`data_df` dataframe to be analyzed. (tibble::tibble)

`feature` attribute to be shown. (character)

Returns: A composite plot. (ggplot2::ggplot)

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
pgu.outliers$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Author(s)

Sebastian Malkusch, <malkusch@med.uni-frankfurt.de>

`pgu.regressor` *pgu.regressor*

Description

A class that performs pairwise robust regression on the pguIMP data set.

Format

R6::R6Class object.

Details

A class that performs pairwise robust regression on the pguIMP data set. This object is used by the shiny based gui and is not for use in individual R-scripts!

Active bindings

`featureNames` Returns the instance variable featureNames. (character)

`setFeatureNames` Sets the instance variable featureNames. It further initializes the instance variables: intercept, pIntercept, slope, pSlope. (character)

`intercept` Returns the instance variable intercept. (matrix)

`pIntercept` Returns instance variable pIntercept. (matrix)

`slope` Returns the instance variable slope. (matrix)

`pSlope` Returns the instance variable pSlope. (matrix)

`model` Returns the instance variable model. (robust::lmRob)

Methods

Public methods:

- pgu.regressor$new()
- pgu.regressor$finalize()
- pgu.regressor$print()
- pgu.regressor$resetRegressor()
- pgu.regressor$resetDiagonal()
- pgu.regressor$resetMatrix()
- pgu.regressor$featureIdx()
- pgu.regressor$featureIsValid()
- pgu.regressor$featurePairIsValid()
- pgu.regressor$createModel()
- pgu.regressor$createRegressionMatrix()
- pgu.regressor$printModel()
- pgu.regressor$printInterceptTbl()
- pgu.regressor$printPInterceptTbl()
- pgu.regressor$printSlopeTbl()
- pgu.regressor$printPSlopeTbl()
- pgu.regressor$plotRegression()
- pgu.regressor$plotResidualDist()
- pgu.regressor$plotResidualBox()
- pgu.regressor$plotModel()
- pgu.regressor$clone()

Method `new()`**:** Creates and returns a new `pgu.regressor` object.

Usage:

```
pgu.regressor$new(data = "tbl_df")
```

Arguments:

`data` The data to be modeled. (tibble::tibble)

Returns: A new `pgu.regressor` object. (pguIMP::pgu.regressor)

Method `finalize()`**:** Clears the heap and indicates if instance of `pgu.regressor` is removed from heap.

Usage:

```
pgu.regressor$finalize()
```

Method `print()`**:** Prints instance variables of a `pgu.regressor` object.

Usage:

```
pgu.regressor$print()
```

Returns: string

Method `resetRegressor()`**:** Performes pair-wise robust linear regression on the attributes of the data tibble. Progresse is indicated by the progress object passed to the function.

Usage:

```
pgu.regressor$resetRegressor(data = "tbl_df", progress = "Progress")
```

Arguments:

`data` Dataframe with at least two numeric attributes. (tibble::tibble)

`progress` Keeps track of the analysis progress. (shiny::Progress)

Method `resetDiagonal()`**:** Sets the diagonal of a square matrix to NA.

Usage:

```
pgu.regressor$resetDiagonal(data = "matrix")
```

Arguments:

`data` The matrix whose diagonal is to be reset. (matrix)

Returns: A matrix with its diagonal reset to NA. (matrix)

Method `resetMatrix()`**:** Creates a square matrix which dimension corresponds to the length of the instance variable featureNames. The matrix entries are set to a distict `value`. The diagonal is set to NA.

Usage:

```
pgu.regressor$resetMatrix(value = "numeric")
```

Arguments:

`value` The value the matrix entries are set to. (numeric)

Returns: A square matrix. (matrix)

Method `featureIdx()`**:** Determines the numerical index of the column of an attribute based on the attribute name.

Usage:

```
pgu.regressor$featureIdx(feature = "character")
```

Arguments:

`feature` The attribute's name. (character)

Returns: The attributes column index. (numeric)

Method `featureIsValid()`**:** Checks if the feature is known to the class.

Usage:

```
pgu.regressor$featureIsValid(feature = "character")
```

Arguments:

`feature` An attribute's name that is to be checked. (character)

Returns: The test result. (logical)

Method `featurePairIsValid()`**:** Checks a if a pair of attributes different and known to the class.

Usage:

```
pgu.regressor$featurePairIsValid(
  abscissa = "character",
  ordinate = "character"
)
```

Arguments:

`abscissa` An attribute's name that is to be checked. (character)

`ordinate` An attribute's name that is to be checked. (character)

Returns: The test result. (logical)

Method `createModel()`**:** Creates a robust model of linear regression between two attributes of a dataframe. The model is stored as instance variable.

Usage:

```
pgu.regressor$createModel(
  data = "tbl_df",
  abscissa = "character",
  ordinate = "character"
)
```

Arguments:

`data` The data to be modeled. (tibble::tibble)

`abscissa` An attribute's name that equals a column name in the data. (character)

`ordinate` An attribute's name that equals a column name in the data. (character)

Method `createRegressionMatrix()`**:** Performs the actual robust linear regression routine. Iteratively runs through the attributes known to the class and creates a robust linear regression model for each valid attribute pair. The model results are stored in the instance variables: intercept, pIntercept, slope, pSlope. Here, pX represents the p-value of the respective parameter X. Displays the progress if shiny is loaded.

Usage:

```
pgu.regressor$createRegressionMatrix(data = "tbl_df", progress = "Progress")
```

Arguments:

`data` The data to be modeled. (tibble::tibble)

`progress` If shiny is loaded, the analysis' progress is stored within this instance of the shiny Progress class. (shiny::Progress)

Method `printModel()`**:** Transforms the results of the modeling procedure for a valid pair of attributes to a dataframe and returns it.

Usage:

```
pgu.regressor$printModel(abscissa = "character", ordinate = "character")
```

Arguments:

`abscissa` The name of the attribute which is assigned to the abscissa. (character)

`ordinate` The name of the attribute which is assigned to the ordinate. (character)

Returns: The analyis result as a dataframe. (tibble::tibble)

Method `printInterceptTbl()`**:** Transfroms instance variable intercept to a dataframe and returns it.

Usage:

```
pgu.regressor$printInterceptTbl()
```

Returns: Dataframe of instance variable intercept. (tibble::tibble)

Method `printPInterceptTbl()`**:** Transfroms instance variable pIntercept to a dataframe and returns it.

Usage:

```
pgu.regressor$printPInterceptTbl()
```

Returns: Dataframe of instance variable pIntercept. (tibble::tibble)

Method `printSlopeTbl()`**:** Transfroms instance variable slope to a dataframe and returns it.

Usage:

```
pgu.regressor$printSlopeTbl()
```

Returns: Dataframe of instance variable slope. (tibble::tibble)

Method `printPSlopeTbl()`**:** Transfroms instance variable pSlope to a dataframe and returns it.

Usage:

```
pgu.regressor$printPSlopeTbl()
```

Returns: Dataframe of instance variable pSlope. (tibble::tibble)

Method `plotRegression()`**:** Creates a scatter plot of the model stored within the instance variable of the class.

Usage:

```
pgu.regressor$plotRegression()
```

Returns: A scatter plot. (ggplot2::ggplot)

Method `plotResidualDist()`**:** Creates a histogram of the residual distribution of the model stored within the instance variable of the class.

Usage:

```
pgu.regressor$plotResidualDist()
```

Returns: A histogram plot. (ggplot2::ggplot)

Method `plotResidualBox()`**:** Creates a box plot of the residual distribution of the model stored within the instance variable of the class.

Usage:

```
pgu.regressor$plotResidualBox()
```

Returns: A box plot. (ggplot2::ggplot)

Method `plotModel()`**:** Creates a model of robust linear regression. Executes all graphical exploration functions of the class and creates a composite graph based on their results.

Usage:

```
pgu.regressor$plotModel(
  data = "tbl_df",
  abscissa = "character",
  ordinate = "character"
)
```

Arguments:

`data` The data to be modeled. (tibble::tibble)

`abscissa` The name of the attribute which is assigned to the abscissa. (character)

`ordinate` The name of the attribute which is assigned to the ordinate. (character)

Returns: A composite graph. (gridExtra::grid.arrange)

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
pgu.regressor$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Author(s)

Sebastian Malkusch, <malkusch@med.uni-frankfurt.de>

`pgu.reporter` *pgu.reporter*

Description

Creates a human readable report file of the pguIMP analysis in odf format via rmarkdown and latex. This object is used by the shiny based gui and is not for use in individual R-scripts!

Format

R6::R6Class object.

Details

I run at the end of the analysis.

Active bindings

`filename` Returns the instance variable filename (character)

`setFilename` Sets the instance variable filename to name. (character)

Methods

Public methods:

- `pgu.reporter$new()`
- `pgu.reporter$finalize()`
- `pgu.reporter$print()`
- `pgu.reporter$write_report()`
- `pgu.reporter$clone()`

Method `new()`**:** Creates and returns a new `pgu.reporter` object.

Usage:

```
pgu.reporter$new(name = "character")
```

Arguments:

`name` Filename of the report pdf. (character)

Returns: A new `pgu.reporter` object. (pguIMP::pgu.report)

Method `finalize()`**:** Clears the heap and indicates that instance of `pgu.reporter` is removed from heap.

Usage:

```
pgu.reporter$finalize()
```

Method `print()`**:** Prints instance variables of a `pgu.reporter` object.

Usage:

```
pgu.reporter$print()
```

Returns: string

Method `write_report()`**:** Writes a report of the pguIMP analysis to a pdf file.

Usage:

```
pgu.reporter$write_report(obj)
```

Arguments:

`obj` A list of class objects that are passed to the rmarkdown script.

Returns: t.b.a.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
pgu.reporter$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Author(s)

Sebastian Malkusch, <malkusch@med.uni-frankfurt.de>

pgu.status *pgu.status*

Description

A class that keeps track of the pguIMP analysis process.

Format

R6::R6Class object.

Details

pguIMP uses a defined linear analysis path The current status therefore provides information about which analyses have already been performed and which still have to be performed. This way pguIMG knows any time during analysis, if all necessary information for the next desired analysis step are available. This object is used by the shiny based gui and is not for use in individual R-scripts!

Active bindings

`processAlphabet` Reurns the process alphabet of the pguIMP analysis routine. (character)

`processStatus` Returns the process status of the pguIMP routine. (logical)

Methods

Public methods:

- `pgu.status$new()`
- `pgu.status$finalize()`
- `pgu.status$print()`
- `pgu.status$reset()`
- `pgu.status$update()`
- `pgu.status$query()`
- `pgu.status$clone()`

Method `new()`**:** Creates and returns a new ‘pgu.status“ object.

Usage:

```
pgu.status$new()
```

Returns: A new `pgu.status` object. (pguIMP::pgu.status)

Method `finalize()`**:** Clears the heap and indicates if instance of `pgu.status` is removed from heap.

Usage:

```
pgu.status$finalize()
```

Method `print()`**:** Prints instance variables of a `pgu.status` object.

Usage:

```
pgu.status$print()
```

Returns: string

Method `reset()`**:** resets the intance variables `processAlphabet` and `processStatus` to their initial values (FALSE).

Usage:

```
pgu.status$reset()
```

Method `update()`**:** updates the process status

Usage:

```
pgu.status$update(processName = "character", value = "logical")
```

Arguments:

`processName` The name of the `pguIMP` process that has been performed. (character)

`value` Indicates if the process ended with success. (logical)

Method `query()`**:** Queries if a process has already been performed successfully.

Usage:

```
pgu.status$query(processName = "character")
```

Arguments:

`processName` Name of the process to be checked. (character)

Returns: Status of the queried process (logical)

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
pgu.status$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Author(s)

Sebastian Malkusch, <malkusch@med.uni-frankfurt.de>

pgu.transformator *pgu.transformator*

Description

Transforms the data of pguIMP.

Format

R6::R6Class object.

Details

Performs a data transformation in order to achieve a normally distributed version of the dataframe. This object is used by the shiny based gui and is not for use in individual R-scripts!

Active bindings

`trafoAlphabet` Returns the instance variable trafoAlphabte.

`trafoParameter` Returns the instance variable trafoParameter.

Methods

Public methods:

- `pgu.transformator$new()`
- `pgu.transformator$finalize()`
- `pgu.transformator$print()`
- `pgu.transformator$resetTrafoParameter()`
- `pgu.transformator$trafoType()`
- `pgu.transformator$setTrafoType()`
- `pgu.transformator$addConstant()`
- `pgu.transformator$mirrorLogic()`
- `pgu.transformator$setMirrorLogic()`
- `pgu.transformator$lambdaLOP()`
- `pgu.transformator$setLambdaLOP()`
- `pgu.transformator$lambdaBC()`
- `pgu.transformator$lambdaAS()`
- `pgu.transformator$featureIdx()`
- `pgu.transformator$addConstGenerator()`
- `pgu.transformator$mirrorNumeric()`
- `pgu.transformator$mirrorData()`
- `pgu.transformator$calculateAddConst()`
- `pgu.transformator$translateNumeric()`
- `pgu.transformator$translateData()`

- pgu.transformator$backTranslateNumeric()
- pgu.transformator$backTranslateData()
- pgu.transformator$lambdaEstimator()
- pgu.transformator$estimateLambda_temp()
- pgu.transformator$estimateLambda()
- pgu.transformator$normalizeArcSine()
- pgu.transformator$optimizeTukeyLadderOfPowers()
- pgu.transformator$optimizeBoxCox()
- pgu.transformator$transformNumeric()
- pgu.transformator$transformData()
- pgu.transformator$transformLogModulus()
- pgu.transformator$transformSquareRoot()
- pgu.transformator$transformCubeRoot()
- pgu.transformator$transformArcsine()
- pgu.transformator$transformInverse()
- pgu.transformator$transformTukeyLadderOfPowers()
- pgu.transformator$transformBoxCox()
- pgu.transformator$inverseTransformNumeric()
- pgu.transformator$inverseTransformData()
- pgu.transformator$inverseTransformLogModulus()
- pgu.transformator$inverseTransformSquareRoot()
- pgu.transformator$inverseTransformCubeRoot()
- pgu.transformator$inverseTransformArcsine()
- pgu.transformator$inverseTransformInverse()
- pgu.transformator$inverseTransformTukeyLadderOfPowers()
- pgu.transformator$inverseTransformBoxCox()
- pgu.transformator$fit()
- pgu.transformator$mutateData()
- pgu.transformator$reverseMutateData()
- pgu.transformator$clone()

Method `new()`**:** Creates and returns a new `pgu.transformator` object.

Usage:

```
pgu.transformator$new(data_df = "tbl_df")
```

Arguments:

`data_df` The data to be analyzed. (tibble::tibble)

Returns: A new `pgu.transformator` object. (pguIMP::pgu.transformator)

Method `finalize()`**:** Clears the heap and indicates that instance of `pgu.transformator` is removed from heap.

Usage:

```
pgu.transformator$finalize()
```

Method `print()`**:** Prints instance variables of a `pgu.transformator` object.

Usage:

```
pgu.transformator$print()
```

Returns: string

Method `resetTrafoParameter()`**:** Resets instance variable `trafoParameter`

Usage:

```
pgu.transformator$resetTrafoParameter(data = "tbl_df")
```

Arguments:

`data` Dataframe to be analyzed. (tibble::tibble)

Method `trafoType()`**:** Returns entry of `trafoType` for user defined attribute.

Usage:

```
pgu.transformator$trafoType(feature = "character")
```

Arguments:

`feature` Attribute's name. (character)

Returns: Value of entry. (character)

Method `setTrafoType()`**:** Sets entry of `trafoType` for user defined attribute.

Usage:

```
pgu.transformator$setTrafoType(feature = "character", type = "character")
```

Arguments:

`feature` Attribute's name. (character)

`type` Trafo type parameter. Valid choices are: "none", "exponential", "log2", "logNorm", "log10", "arcsine", "tukeyLOP", "boxCox". (character)

Method `addConstant()`**:** Returns entry of `addConst` for user defined attribute.

Usage:

```
pgu.transformator$addConstant(feature = "character")
```

Arguments:

`feature` Attribute's name. (character)

Returns: Value of entry. (numeric)

Method `mirrorLogic()`**:** Returns entry of `mirrorLogic` for user defined attribute.

Usage:

```
pgu.transformator$mirrorLogic(feature = "character")
```

Arguments:

`feature` Attribute's name. (character)

Returns: Value of entry. (logical)

Method `setMirrorLogic()`**:** Sets entry of `mirrorLogic` for user defined attribute.

Usage:

```
pgu.transformator$setMirrorLogic(feature = "character", logic = "logical")
```

Arguments:

`feature` Attribute's name. (character)

`logic` Specifies whether the data should be mirrored at the coordinate origin. (logical)

Method `lambdaLOP()`**:** Returns entry of `lambda_lop` for user defined attribute. Lambda is a specific optimization parameter that is derived from the Tukey-LOP transfromation procedure.

Usage:

```
pgu.transformator$lambdaLOP(feature = "character")
```

Arguments:

`feature` Attribute's name. (character)

Returns: Value of entry. (numeric)

Method `setLambdaLOP()`**:** Sets entry of `lambda_lop` for user defined attribute.

Usage:

```
pgu.transformator$setLambdaLOP(feature = "character", lambda = "numeric")
```

Arguments:

`feature` Attribute's name. (character)

`lambda` Sets the feature specific exponential value. (numeric)

Method `lambdaBC()`**:** Returns entry of `lambda_bc` for user defined attribute. Lambda is a specific optimization parameter that is derived from the Box-Cox transfromation procedure.

Usage:

```
pgu.transformator$lambdaBC(feature = "character")
```

Arguments:

`feature` Attribute's name. (character)

Returns: Value of entry. (numeric)

Method `lambdaAS()`**:** Returns entry of `lambda_as` for user defined attribute. Lambda is a specific optimization parameter that is derived from the arcsine transfromation procedure.

Usage:

```
pgu.transformator$lambdaAS(feature = "character")
```

Arguments:

`feature` Attribute's name. (character)

Returns: Value of entry. (numeric)

Method `featureIdx()`**:** Returns the index of a pgu.normDist object wihtin the instance variable `trafoParameter`.

Usage:

```
pgu.transformator$featureIdx(feature = "character")
```

Arguments:

`feature` Attribute's name. (character)

Returns: Index of attribute entry in dataframe (numeric)

Method `addConstGenerator()`**:** Calculates and returns the addConst. A constant that prevents the occurrence of negative values as well as zero, if added to an attribute.

Usage:

```
pgu.transformator$addConstGenerator(value = "numeric")
```

Arguments:

`value` The smallest of the attribute's values. (numeric)

Returns: The addConst for the attribute (numeric)

Method `mirrorNumeric()`**:** Mirrors the assigned values at the coordinate origin.

Usage:

```
pgu.transformator$mirrorNumeric(value = "numeric")
```

Arguments:

`value` Value or vector of values. (numeric)

Returns: Value or vector of values. (numeric)

Method `mirrorData()`**:** Calls the class' mirrorNumeric function on all numeric attributes of a data frame.

Usage:

```
pgu.transformator$mirrorData(data = "tbl_df")
```

Arguments:

`data` A data frame. (tibble:tibble)

Returns: A data frame (tibble::tibble)

Method `calculateAddConst()`**:** Calculates the addConst value for each attribute of the assigned data frame, by calling the class' addConstGenerator function. The results are stored in addConst attribute of the trafoParameter instance variable.

Usage:

```
pgu.transformator$calculateAddConst(data = "tbl_df")
```

Arguments:

`data` A data frame. (tibble:tibble)

Method `translateNumeric()`**:** Translates the assigned values by a constant.

Usage:

```
pgu.transformator$translateNumeric(value = "numeric", const = "numeric")
```

Arguments:

`value` A numeric or a vector of numerics to be translated. (numeric)

`const` A constant value. (numeric)

Returns: A numeric or a vector of numerics. (numeric)

Method `translateData()`**:** Translates each attribute of the assigned data frame, by calling the class' translateNumeric function. The respective addConst values of the individual attributes of the data frame serve as const variables.

Usage:

```
pgu.transformator$translateData(data = "tbl_df")
```

Arguments:

`data` A data frame. (tibble:tibble)

Returns: A data frame. (tibble:tibble)

Method `backTranslateNumeric()`**:** Back-translates the assigned values by a constant.

Usage:

```
pgu.transformator$backTranslateNumeric(value = "numeric", const = "numeric")
```

Arguments:

`value` A numeric or a vector of numerics to be back-translated. (numeric)

`const` A constant value. (numeric)

Returns: A numeric or a vector of numerics. (numeric)

Method `backTranslateData()`**:** Back-translates each attribute of the assigned data frame, by calling the class' backTranslateNumeric function. The respective addConst values of the individual attributes of the data frame serve as const variables.

Usage:

```
pgu.transformator$backTranslateData(data = "tbl_df")
```

Arguments:

`data` A data frame. (tibble:tibble)

Returns: A data frame. (tibble:tibble)

Method `lambdaEstimator()`**:** Estimates the lambda factor for the given values, that are assigned to a user defined attribute..

Usage:

```
pgu.transformator$lambdaEstimator(value = "numeric", feature = "character")
```

Arguments:

`value` A numeric or a vector of numerics to be analyzed. (numeric)

`feature` The attribute which the given values are assigned to. (character)

Returns: The specific lambda factor. (numeric)

Method `estimateLambda_temp()`**:** Estimates the lambda factor for each attribute of the assigned data frame, by calling the class' lambdaEstimator function. The respective lambda values of the individual attributes of the data frame are stored in the lambda attribute of the instance variable trafoParameter.

Usage:

```
pgu.transformator$estimateLambda_temp(data = "tbl_df")
```

Arguments:

`data` A data frame. (tibble:tibble)

Method `estimateLambda()`**:** Estimates the arcsine transformation lambda factor for each attribute of the assigned data frame. The respective lambda values of the individual attributes of the data frame are stored in the lambda attribute of the instance variable trafoParameter.

Usage:

```
pgu.transformator$estimateLambda(data = "tbl_df")
```

Arguments:

`data` A data frame. (tibble:tibble)

Method `normalizeArcSine()`**:** Estimates the lambda factor for an arcsine transformation for the given values,

Usage:

```
pgu.transformator$normalizeArcSine(value = "numeric")
```

Arguments:

`value` A numeric or a vector of numerics to be analyzed. (numeric)

Returns: The specific lambda factor. (numeric)

Method `optimizeTukeyLadderOfPowers()`**:** Estimates the lambda factor for a tukeyLOP transformation for the given values,

Usage:

```
pgu.transformator$optimizeTukeyLadderOfPowers(value = "numeric")
```

Arguments:

`value` A numeric or a vector of numerics to be analyzed. (numeric)

Returns: The specific lambda factor. (numeric)

Method `optimizeBoxCox()`**:** Estimates the lambda factor for a boxcox transformation for the given values,

Usage:

```
pgu.transformator$optimizeBoxCox(value = "numeric")
```

Arguments:

`value` A numeric or a vector of numerics to be analyzed. (numeric)

Returns: The specific lambda factor. (numeric)

Method `transformNumeric()`**:** Transforms the given numeric values, that are assigned to a user defined attribute.

Usage:

```
pgu.transformator$transformNumeric(value = "numeric", feature = "character")
```

Arguments:

`value` A numeric or a vector of numerics to be tranformed. (numeric)

`feature` The attribute which the given values are assigned to. (character)

Returns: A transfromed version of the given numeric or vector of numerics. (numeric)

Method `transformData()`**:** Transforms each attribute of the assigned data frame, by calling the class' tranformNumeric function. The respective lambda values of the individual attributes of the data frame are read from the lambda attribute of the instance variable trafoParameter.

Usage:

```
pgu.transformator$transformData(data = "tbl_df")
```

Arguments:

`data` A data frame. (tibble:tibble)

Method `transformLogModulus()`**:** Performes a log transformation for the given values, based on a user defined base value.

Usage:

```
pgu.transformator$transformLogModulus(value = "numeric", base = "numeric")
```

Arguments:

`value` A numeric or a vector of numerics to be analyzed. (numeric)

`base` Logarithmic base. (numeric)

Returns: The transformed values. (numeric)

Method `transformSquareRoot()`**:** Performes a square root transformation for the given values.

Usage:

```
pgu.transformator$transformSquareRoot(value = "numeric")
```

Arguments:

`value` A numeric or a vector of numerics to be analyzed. (numeric)

Returns: The transformed values. (numeric)

Method `transformCubeRoot()`**:** Performes a cube root transformation for the given values.

Usage:

```
pgu.transformator$transformCubeRoot(value = "numeric")
```

Arguments:

`value` A numeric or a vector of numerics to be analyzed. (numeric)

Returns: The transformed values. (numeric)

Method `transformArcsine()`**:** Performes an arcsine transformation for the given values.

Usage:

```
pgu.transformator$transformArcsine(value = "numeric", lambda = "numeric")
```

Arguments:

`value` A numeric or a vector of numerics to be analyzed. (numeric)

`lambda` Normalization factor. (numeric)

Returns: The transformed values. (numeric)

Method `transformInverse()`**:** Performes an inverse transformation for the given values.

Usage:

```
pgu.transformator$transformInverse(value = "numeric")
```

Arguments:

`value` A numeric or a vector of numerics to be analyzed. (numeric)

Returns: The transformed values. (numeric)

Method `transformTukeyLadderOfPowers()`**:** Performes a tukeyLOP transformation for the given values.

Usage:

```
pgu.transformator$transformTukeyLadderOfPowers(
  value = "numeric",
  lambda = "numeric"
)
```

Arguments:

`value` A numeric or a vector of numerics to be analyzed. (numeric)

`lambda` Lambda factor. (numeric)

Returns: The transformed values. (numeric)

Method `transformBoxCox()`**:** Performes a boxcox transformation for the given values.

Usage:

```
pgu.transformator$transformBoxCox(value = "numeric", lambda = "numeric")
```

Arguments:

`value` A numeric or a vector of numerics to be analyzed. (numeric)

`lambda` Lambda factor. (numeric)

Returns: The transformed values. (numeric)

Method `inverseTransformNumeric()`**:** Inverse transforms the given numeric values, that are assigned to a user defined attribute.

Usage:

```
pgu.transformator$inverseTransformNumeric(
  value = "numeric",
  feature = "character"
)
```

Arguments:

`value` A numeric or a vector of numerics to be tranformed. (numeric)

`feature` The attribute which the given values are assigned to. (character)

Returns: An inverse transfromed version of the given numeric or vector of numerics. (numeric)

Method `inverseTransformData()`**:** Inverse transforms each attribute of the assigned data frame, by calling the class' tranformNumeric function. The respective lambda values of the individual attributes of the data frame are read from the lambda attribute of the instance variable trafoParameter.

Usage:

```
pgu.transformator$inverseTransformData(data = "tbl_df")
```

Arguments:

`data` A data frame. (tibble:tibble)

Method `inverseTransformLogModulus()`**:** Performes an inverse log transformation for the given values, based on a user defined base value.

Usage:

```
pgu.transformator$inverseTransformLogModulus(
  value = "numeric",
  base = "numeric"
)
```

Arguments:

`value` A numeric or a vector of numerics to be analyzed. (numeric)

`base` Logarithmic base. (numeric)

Returns: The transformed values. (numeric)

Method `inverseTransformSquareRoot()`**:** Performes an inverse square root transformation for the given values.

Usage:

```
pgu.transformator$inverseTransformSquareRoot(value = "numeric")
```

Arguments:

`value` A numeric or a vector of numerics to be analyzed. (numeric)

Returns: The transformed values. (numeric)

Method `inverseTransformCubeRoot()`**:** Performes an inverse cube root transformation for the given values.

Usage:

```
pgu.transformator$inverseTransformCubeRoot(value = "numeric")
```

Arguments:

`value` A numeric or a vector of numerics to be analyzed. (numeric)

Returns: The transformed values. (numeric)

Method `inverseTransformArcsine()`**:** Performes an inverse arcsine transformation for the given values.

Usage:

```
pgu.transformator$inverseTransformArcsine(
  value = "numeric",
  lambda = "numeric"
)
```

Arguments:

`value` A numeric or a vector of numerics to be analyzed. (numeric)

`lambda` Normalization factor. (numeric)

Returns: The transformed values. (numeric)

Method `inverseTransformInverse()`**:** Performes an inverse inverse-transformation for the given values.

Usage:

```
pgu.transformator$inverseTransformInverse(value = "numeric")
```

Arguments:

`value` A numeric or a vector of numerics to be analyzed. (numeric)

Returns: The transformed values. (numeric)

Method `inverseTransformTukeyLadderOfPowers()`**:** Performes an inverse tukeyLOP transformation for the given values.

Usage:

```
pgu.transformator$inverseTransformTukeyLadderOfPowers(
  value = "numeric",
  lambda = "numeric"
)
```

Arguments:

`value` A numeric or a vector of numerics to be analyzed. (numeric)

`lambda` Lambda factor. (numeric)

Returns: The transformed values. (numeric)

Method `inverseTransformBoxCox()`**:** Performes an inverse boxcox transformation for the given values.

Usage:

```
pgu.transformator$inverseTransformBoxCox(value = "numeric", lambda = "numeric")
```

Arguments:

`value` A numeric or a vector of numerics to be analyzed. (numeric)

`lambda` Lambda factor. (numeric)

Returns: The transformed values. (numeric)

Method `fit()`**:** Estimate all transformation parameters(lambda, addConst,...) for each attribute of a given data frame. The function calls the class' functions calculateAddConst and estimateLambda. The results are stored in the respective attributes of the instance variable trafoParameter.

Usage:

```
pgu.transformator$fit(data = "tbl_df")
```

Arguments:

`data` A data frame. (tibble:tibble)

Method `mutateData()`**:** Mutates the values of each attribute of a given data frame. Here, mutation is defined as the cesecutive sequence of the class' functions mirrorData, tranlsateData and transfromData.

Usage:

```
pgu.transformator$mutateData(data = "tbl_df")
```

Arguments:

`data` A data frame. (tibble:tibble)

Returns: A mutated data frame. (tibble::tibble)

Method `reverseMutateData()`**:** Re-mutates the values of each attribute of a given data frame. Here, re-mutation is defined as the cesecutive sequence of the class' functions inverseTransformData, backTranslateData, mirrorData

Usage:

```
pgu.transformator$reverseMutateData(data = "tbl_df")
```

Arguments:

`data` A data frame. (tibble:tibble)

Returns: A mutated data frame. (tibble::tibble)

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
pgu.transformator$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Author(s)

Sebastian Malkusch, <malkusch@med.uni-frankfurt.de>

`pgu.validator` *pgu.validator*

Description

Validates that the distribution is not significantly altered by the imputation process. This object is used by the shiny based gui and is not for use in individual R-scripts!

Format

R6::R6Class object.

Details

Takes two distributions (before and after imputation). Performs a Wilcoxon-Mann-Whitney U test. Performs a Kolmogorow-Smirnow test.

Active bindings

`testStatistics_df` Returns the instance variable `testStatistics_df`. (tibble::tibble)

`centralMoments_org` Returns the instance variable `centralMoments_org` (tibble::tibble)

`centralMoments_imp` Returns the instance variable `centralMoments_imp` (tibble::tibble)

`centralMoments_delta` Returns the instance variable `centralMoments_delta` (tibble::tibble)

`features` Returns the instance variable `features` (character)

`seed` Returns the instance variable seed (integer)

`setSeed` Sets the instance variable seed. (numeric)

Methods

Public methods:

- pgu.validator$new()
- pgu.validator$finalize()
- pgu.validator$print()
- pgu.validator$resetValidator()
- pgu.validator$kolmogorowTestFeature()
- pgu.validator$wilcoxonTestFeature()
- pgu.validator$centralMomentsFeature()
- pgu.validator$validate()
- pgu.validator$featurePdf()
- pgu.validator$featureCdf()
- pgu.validator$featureVs()
- pgu.validator$featureBoxPlot()
- pgu.validator$featurePlot()
- pgu.validator$clone()

Method `new()`**:** Creates and returns a new `pgu.validator` object.

Usage:

```
pgu.validator$new(seed = 42)
```

Arguments:

`seed` Set the instance variable `seed`. (integer)

Returns: A new `pgu.validator` object. (pguIMP::pgu.validator)

Method `finalize()`**:** Clears the heap and indicates that instance of `pgu.validator` is removed from heap.

Usage:

```
pgu.validator$finalize()
```

Method `print()`**:** Prints instance variables of a `pgu.validator` object.

Usage:

```
pgu.validator$print()
```

Returns: string

Method `resetValidator()`**:** Resets instance variables

Usage:

```
pgu.validator$resetValidator()
```

Method `kolmogorowTestFeature()`**:** Performs a comparison between the original and the imputated distribution of a given feature using a two-sided Kolmorogow-Smirnow test with simulated p-vaue distribution.

Usage:

```
pgu.validator$kolmogorowTestFeature(
  org = "numeric",
  imp = "numeric",
  feature = "character"
)
```

Arguments:

`org` Original data to be analzed. (numeric)

`imp` Imputed data to be analyzed. (numeric)

`feature` Feature name of the analyzed distributions. (character)

Returns: One row dataframe comprising the test results. (tibble::tibble)

Method `wilcoxonTestFeature()`**:** Performs a comparison between the original and the imputated distribution of a given feature using a two-sided Wilcoxon/Mann-Whitney test.

Usage:

```
pgu.validator$wilcoxonTestFeature(
  org = "numeric",
  imp = "numeric",
  feature = "character"
)
```

Arguments:

`org` Original data to be analzed. (numeric)

`imp` Imputed data to be analyzed. (numeric)

`feature` Feature name of the analyzed distributions. (character)

Returns: One row dataframe comprising the test results. (tibble::tibble)

Method `centralMomentsFeature()`**:** Estimates estimates the central moments of the given distribution.

Usage:

```
pgu.validator$centralMomentsFeature(values = "numeric", feature = "character")
```

Arguments:

`values` Data to be analzed. (numeric)

`feature` Feature name of the analyzed distributions. (character)

Returns: One row dataframe comprising the statistics. (tibble::tibble)

Method `validate()`**:** Validates the feature distributions of the original and the imputated dataframe“ using a two-sided Kolmorogow-Smirnow test and a two-sided Wilcoxon/Mann-Whitney test. The result is stored in the instance variables `testStatistics_df`and ‘distributionStatistics_df‘. Displays the progress if shiny is loaded.

Usage:

```
pgu.validator$validate(
  org_df = "tbl_df",
  imp_df = "tbl_df",
  progress = "Progress"
)
```

Arguments:

`org_df` Original dataframe to be analzed. (tibble::tibble)

`imp_df` Imputed dataframe to be analyzed. (tibble::tibble)

`progress` If shiny is loaded, the analysis' progress is stored in this instance of the shiny Progress class. (shiny::Progress)

Method `featurePdf()`**:** Receives a dataframe and plots the pareto density of the features 'org_pdf' and 'imp_pdf'. Returns the plot

Usage:

```
pgu.validator$featurePdf(data_df = "tbl_df")
```

Arguments:

`data_df` dataframe to be plotted (tibble::tibble)

Returns: A ggplot2 object (ggplot2::ggplot)

Method `featureCdf()`**:** Receives a dataframe and plost the feature 'x' against the features 'org_cdf' and 'imp_cdf'. Returns the plot

Usage:

```
pgu.validator$featureCdf(data_df = "tbl_df")
```

Arguments:

`data_df` dataframe to be plotted (tibble::tibble)

Returns: A ggplot2 object (ggplot2::ggplot)

Method `featureVs()`**:** Receives two numeric vectors 'org' and 'imp'. Plots the qq-plot of both vectors. Returns the plot

Usage:

```
pgu.validator$featureVs(org = "numeric", imp = "numeric")
```

Arguments:

`org` Numric vector comprising the original data. (numeric)

`imp` Numeric vector comprising the imputed data. (numeric)

Returns: A ggplot2 object (ggplot2::ggplot)

Method `featureBoxPlot()`**:** Receives a dataframe and information about the lloq and uloq and retuns a boxplot

Usage:

```
pgu.validator$featureBoxPlot(
  data_df = "tbl_df",
  lloq = "numeric",
  uloq = "numeric",
  feature = "character"
)
```

Arguments:

`data_df` Dataframe to be analyzed (tibble::tibble)

`lloq` lower limit of quantification (numeric)

uloq upper limit of quantification (numeric)

feature Feature name (character)

Returns: A ggplot2 object (ggplot2::ggplot)

Method `featurePlot()`**:** Receives two numeric dataframes 'org_df' and 'imp_df', and a feature name. Creates a compund plot of the validation results for the given feature.. Returns the plot

Usage:

```
pgu.validator$featurePlot(
  org_df = "tbl_df",
  imp_df = "tbl_df",
  lloq = "numeric",
  uloq = "numeric",
  impIdx_df = "tbl_df",
  feature = "character"
)
```

Arguments:

org_df Dataframe comprising the original data. (tibble::tibble)

imp_df Dataframe comprising the imputed data. (tibble::tibble)

lloq lower limit of quantification (numeric)

uloq upper limit of quantification (numeric)

impIdx_df dataframe comprising information about imputation sites (tibble::tibble)

feature Feature name. (character)

Returns: A ggplot2 object (ggplot2::ggplot)

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
pgu.validator$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

Author(s)

Sebastian Malkusch, <malkusch@med.uni-frankfurt.de>

---

pguIMP *pguIMP*

Description

Reproducible cleaning of biomedical laboratory data using methods of visualization, error correction and transformation implemented as interactive R-notebooks.

Usage

```
pguIMP()
```

Details

A graphical data preprocessing toolbox, named "pguIMP", that includes a fixed sequence of preprocessing steps to enable error-free data preprocessing interactively. By implementing contemporary data processing methods including machine learning-based imputation procedures, the creation of corrected and cleaned bioanalytical datasets is ensured, which preserve data structures such as clusters better than resulting with classical methods.

Value

shiny application object

Author(s)

Sebastian Malkusch, <malkusch@med.uni-frankfurt.de>

`sLogLikelihood` *dLogLikelihood*

Description

Calculates bbmle snmor function.

Usage

```
sLogLikelihood(mu = "numeric", sigma = "numeric")
```

Arguments

`mu`	The expextation value. (numeric)
`sigma`	The standard deviation. (numeric)

Value

the bbmle::snorm results.

Author(s)

Sebastian Malkusch, <malkusch@med.uni-frankfurt.de>

Examples

```
y <- sLogLikelihood (mu=0.0, sigma=1.0)
```

transposeTibble *transposeTibble*

Description

Transposes a tibble This object is used by the shiny based gui and is not for use in individual R-scripts!

Usage

```
transposeTibble(obj = "tbl_df")
```

Arguments

obj	The data frame to be transposed. (numeric)

Value

The transposed data frame. (tibble:tibble)

Author(s)

Sebastian Malkusch, <malkusch@med.uni-frankfurt.de>

Index