Package 'rBiasCorrection'

September 23, 2020

Title Correct Bias in DNA Methylation Analyses

Version 0.2.3

Description Implementation of the algorithms (with minor modifications) to correct bias in quantitative DNA methylation analyses as described by Moskalev et al. (2011) <doi:10.1093/nar/gkr213>.

License GPL-3

URL https://github.com/kapsner/rBiasCorrection

BugReports https://github.com/kapsner/rBiasCorrection/issues

Depends R (>= 2.10)

Imports data.table, future, future.apply, ggplot2, ggpubr, magrittr, nls2, polynom, stats

Suggests knitr, lintr, microbenchmark, rmarkdown, testthat

VignetteBuilder knitr

Date/Publication 2020-09-23 04:50:02 UTC

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

NeedsCompilation no

Author Lorenz A. Kapsner [cre, aut, cph] (<https://orcid.org/0000-0003-1866-860X>), Evgeny A. Moskalev [aut]

Maintainer Lorenz A. Kapsner <lorenz.kapsner@gmail.com>

Repository CRAN

R topics documented:

`aggregated_input` *aggregated_input helper function*

Description

Internal function to present aggregated input data on which calculations are performed. This function does only have an effect, if repeated measurements are used for calibration a/o experimental data.

Usage

```
aggregated_input(datatable, description, vec_cal, type = NULL)
```

Arguments

`datatable`	A data.table object that contains either the experimental data or the calibration data.
`description`	A character string, indicating if `datatable` contains either *"calibration"* data or *"experimental"* data.
`vec_cal`	The vector containing the CpG columns (output of ‘clean_dt‘ with description = "calibration").
`type`	A single integer. Type of data to be corrected: either "1" (one locus in many samples, e.g. pyrosequencing data) or "2" (many loci in one sample, e.g. next-generation sequencing data or microarray data).

Value

A data.table in the long format with aggregated means for each CpG site of each sample and the corresponding standard deviation.

Examples

```
experimental <- rBiasCorrection::example.data_experimental
calibration <- rBiasCorrection::example.data_calibration

vec_cal <- calibration$vec_cal

experimental_aggregated <- aggregated_input(
  datatable = experimental$dat,
  description = "experimental",
  vec_cal = vec_cal,
  type = 1
)
dim(experimental_aggregated)
class(experimental_aggregated)

calibration_aggregated <- aggregated_input(
  datatable = calibration$dat,
  description = "calibration",
  vec_cal = vec_cal
)
dim(calibration_aggregated)
class(calibration_aggregated)
```

---

better_model *better_model helper function*

Description

Internal function to select the better model between hyperbolic regression and cubic regression.

Usage

```
better_model(
  statstable_pre,
  statstable_post_hyperbolic = NULL,
  statstable_post_cubic = NULL,
  selection_method = "SSE"
)
```

Arguments

statstable_pre	A data.table object, containing the output of statisticsList_() of the calculated regression parameters (form the provided calibration data).
statstable_post_hyperbolic	A data.table object, containing the output of statisticsList_() of the calculated regression parameters form the calibration data corrected with hyperbolic regression.
statstable_post_cubic	A data.table object, containing the output of statisticsList_() of the calculated regression parameters form the calibration data corrected with cubic regression.
selection_method	A character string. The method used to select the regression algorithm to correct the respective CpG site. This is by default the sum of squared errors ("SSE"). The second option is "RelError", which selects the regression method based on the theoretical relative error after correction. This metric is calculated by correcting the calibration data with both the hyperbolic regression and the cubic regression and using them again as input data to calculate the 'goodness of fit'-metrics.

Value

The function returns a data.table with 4 columns, the last column being named 'better_model', which indicates in a binary manner, if the hyperbolic model (better_model = 0) or the cubic model (better_model = 1) result in a 'better' 'SSE' or 'RelError' respectively.

Examples

```
# define list object to save all data
rv <- list()
rv$minmax <- TRUE
rv$selection_method <- "RelError"
rv$sample_locus_name <- "Test"
rv$seed <- 1234

# define logfilename
logfilename <- paste0(tempdir(), "/log.txt")

# import experimental file
exp_type_1 <- rBiasCorrection::example.data_experimental
rv$fileimport_experimental <- exp_type_1$dat

# import calibration file
cal_type_1 <- rBiasCorrection::example.data_calibration
rv$fileimport_calibration <- cal_type_1$dat
rv$vec_cal <- cal_type_1$vec_cal


# perform regression
```

```
regression_results <- regression_utility(
  rv$fileimport_calibration,
  "Testlocus",
  locus_id = NULL,
  rv = rv,
  mode = NULL,
  logfilename,
  minmax = rv$minmax,
  seed = rv$seed
)

# extract regression results
rv$result_list <- regression_results$result_list

# get regression statistics
rv$reg_stats <- statistics_list(
  rv$result_list,
  minmax = rv$minmax
)

# select the better model based on the sum of squared errrors ("SSE")
rv$choices_list <- better_model(
  statstable_pre = rv$reg_stats,
  selection_method = "SSE"
)
```

`biascorrection` *Correct PCR-Bias in Quantitative DNA Methylation Analyses.*

Description

This function implements the algorithms described by Moskalev et. al in their article 'Correction of PCR-bias in quantitative DNA methylation studies by means of cubic polynomial regression', published 2011 in Nucleic acids research, Oxford University Press (https://doi.org/10.1093/nar/gkr213).

Usage

```
biascorrection(
  experimental,
  calibration,
  samplelocusname,
  minmax = FALSE,
  correct_method = "best",
  selection_method = "SSE",
  type = 1,
  csvdir = paste0(tempdir(), "/csvdir/"),
```

```
  plotdir = paste0(tempdir(), "/plotdir/"),
  logfilename = paste0(tempdir(), "/log.txt"),
  plot_height = 5,
  plot_width = 7.5,
  plot_textsize = 16,
  seed = 1234,
  parallel = TRUE
)
```

Arguments

`experimental`	A character string. Path to the file containing the raw methylation values of the samples under investigation.
`calibration`	A character string. In type 1 data (one locus in many samples, e.g. pyrosequencing data): Path to the file containing the raw methylation values of the calibration samples. In type 2 data (many loci in one sample, e.g. next-generationsequencing data or microarray data): Path to the folder that contains at least 4 calibration files (one file per calibration step). Please refer to the FAQ for more detailed information on the specific file requirements (https://raw.githubusercontent.com/kapsner/PCRBiasCorrection/master/FAQ.md).
`samplelocusname`	A character string. In type 1 data: locus name - name of the gene locus under investigation. In type 2 data: sample name - name of the sample under investigation.
`minmax`	A logical, indicating which equations are used for BiasCorrection (default: FALSE). If TRUE, equations are used that include the respective minima and maxima of the provided data.
`correct_method`	A character string. Method used to correct the PCR- bias of the samples under investigation. One of "best" (default), "hyperbolic" or "cubic". If the method is set to "best" (short: "b"), the algorithm will automatically determine the best fitting type of regression for each CpG site based on *selection_method* (by default: sum of squared errors, SSE, https://en.wikipedia.org/wiki/Residual_sum_of_squares). If the method is set to "hyperbolic" (short: "h") or "cubic" (short: "c"), the PCR-bias correction of all samples under investigation will be performed with the hyperbolic or the cubic regression respectively.
`selection_method`	A character string. The method used to select the regression algorithm to correct the respective CpG site. This is by default the sum of squared errors ("SSE"). The second option is "RelError", which selects the regression method based on the theoretical relative error after correction. This metric is calculated by correcting the calibration data with both the hyperbolic regression and the cubic regression and using them again as input data to calculate the 'goodness of fit'-metrics.
`type`	A single integer. Type of data to be corrected: either "1" (one locus in many samples, e.g. pyrosequencing data) or "2" (many loci in one sample, e.g. next-generation sequencing data or microarray data).

csvdir	A character string. Directory to store the resulting tables. (default = paste0(tempdir(), "/plotdir/")). CAUTION: This directory will be newly created on every call of the function - any preexisting files will be deleted without a warning.
plotdir	A character string. Directory to store the resulting plots (default = paste0(tempdir(), "/plotdir/")). CAUTION: This directory will be newly created on every call of the function - any preexisting files will be deleted without a warning.
logfilename	A character string. Path to a file to save the log messages (default = paste0(tempdir(), "/log.txt")).
plot_height	A integer value. The height (unit: inch) of the resulting plots (default: 5).
plot_width	A integer value. The width (unit: inch) of the resulting plots (default: 7.5).
plot_textsize	A integer value. The textsize of the resulting plots (default: 16).
seed	A integer value. The seed used when solving the unknowns in the hyperbolic regression equation and the cubic regression equation. Important for reproducibility (default: 1234).
parallel	A boolean. If TRUE (the default value), initializing ‘future::plan("multiprocess")‘ before running the code.

Value

This function is a wrapper around all of ‘rBiasCorrection‘’s included functions. When executing it, it performs the whole workflow of bias correction and writes resulting csv-files and plots, as well as a log file to the local file system (the respective directories can be specified with the function arguments). The return-value is TRUE, if the correction of PCR measurement biases succeeds. If the correction fails, an error message is returned.

Examples

```
data.table::fwrite(
  rBiasCorrection::example.data_experimental$dat,
  paste0(tempdir(), "/experimental_data.csv")
)
data.table::fwrite(
  rBiasCorrection::example.data_calibration$dat,
  paste0(tempdir(), "/calibration_data.csv")
)
experimental <- paste0(tempdir(), "/experimental_data.csv")
calibration <- paste0(tempdir(), "/calibration_data.csv")

results <- biascorrection(
  experimental = experimental,
  calibration = calibration,
  samplelocusname = "BRAF",
  parallel = FALSE
)
```

`calibration_plot` *calibration_plot helper function*

Description

Internal function to carry out the plotting of the calibrations curves.

Usage

```
calibration_plot(plotlist, coef_hyper, coef_cubic, plot_textsize, minmax)
```

Arguments

`plotlist`	A ggplot object containing a regression plot without regression curves (output of `regression_utility()`).
`coef_hyper`	A list containing the regression parameters of the hyperbolic regression equation.
`coef_cubic`	A list containing the regression parameters of the cubic regression equation.
`plot_textsize`	A integer value. The textsize of the resulting plots (default: 16).
`minmax`	A logical, indicating which equations are used for BiasCorrection (default: FALSE). If TRUE, equations are used that include the respective minima and maxima of the provided data.

Value

The function returns a list containing calibration plots.

`clean_dt` *clean_dt helper function*

Description

Internal function, that checks the formatting of imported files and prepares them to be applicable for the following pcr-bias correction steps.

Usage

```
clean_dt(datatable, description, type, logfilename)
```

Arguments

datatable	A data.table object that contains either the experimental data or the calibration data.
description	A character string, indicating if datatable contains either *"calibration"* data or *"experimental"* data.
type	A single integer. Type of data to be corrected: either "1" (one locus in many samples, e.g. pyrosequencing data) or "2" (many loci in one sample, e.g. next-generation sequencing data or microarray data).
logfilename	A character string. Path to the logfile to save the log messages.

Value

If a valid file is provided, the function returns a cleaned data.table, suited for BiasCorrection.

Examples

```
logfilename <- paste0(tempdir(), "/log.txt")
cleaned_experimental <- clean_dt(
  datatable = rBiasCorrection::example.data_experimental$dat,
  description = "experimental",
  type = 1,
  logfilename = logfilename
)
dim(cleaned_experimental)
class(cleaned_experimental)
```

---

clean_up *clean_up helper function*

Description

Internal function to clean up directories.

Usage

```
clean_up(plotdir, csvdir)
```

Arguments

plotdir	A character string. Path to the folder, where plots are saved.
csvdir	A character string. Path to the folder, where resulting tables are saved.

Value

This function silently cleans up the current session and removes both, the ‘plotdir‘- and the ‘csvdir‘-folders. It furthermore resets the ‘future‘-backend to plan = "sequential".

See Also

plan

Examples

```
plotdir <- paste0(tempdir(), "/plots/")
csvdir <- paste0(tempdir(), "/csv/")

clean_up(plotdir, csvdir)
```

---

createbarerrorplots *createbarerrorplots helper function*

Description

Internal function to create relative-error bar plots.

Usage

```
createbarerrorplots(
  statstable_pre,
  statstable_post,
  rv,
  type,
  locus_id = NULL,
  plotdir,
  logfilename,
  mode = NULL,
  plot_height = 5,
  plot_width = 7.5,
  plot_textsize = 16
)
```

Arguments

statstable_pre	A data.table object, containing the output of statisticsList_() of the calculated regression parameters (form the provided calibration data).
statstable_post	A data.table object, containing the output of statisticsList_() of the calculated regression parameters form the corrected calibration data.
rv	A list object. A list that contains additional objects needed for the algorithms.
type	A single integer. Type of data to be corrected: either "1" (one locus in many samples, e.g. pyrosequencing data) or "2" (many loci in one sample, e.g. next-generation sequencing data or microarray data).

locus_id	A character string. Default: NULL. ID of the respective locus (only used in type 2 correction).
plotdir	A character string. Path to the folder, where plots are saved.
logfilename	A character string. Path to a file to save the log messages (default = paste0(tempdir(), "/log.txt")).
mode	A character string. Default: NULL. Used to indicate "corrected" calibration data.
plot_height	A integer value. The height (unit: inch) of the resulting plots (default: 5).
plot_width	A integer value. The width (unit: inch) of the resulting plots (default: 7.5).
plot_textsize	A integer value. The textsize of the resulting plots (default: 16).

Value

This function creates error bar-plots to visualize the relative error before and after bias correction and writes these plots to the local filesystem.

Examples

```
# define list object to save all data
rv <- list()
rv$minmax <- TRUE
rv$selection_method <- "RelError"
rv$sample_locus_name <- "Test"
rv$seed <- 1234

# define plotdir
rv$plotdir <- paste0(tempdir(), "/plots/")
dir.create(rv$plotdir)

# define logfilename
logfilename <- paste0(tempdir(), "/log.txt")

# import experimental file
exp_type_1 <- rBiasCorrection::example.data_experimental
rv$fileimport_experimental <- exp_type_1$dat

# import calibration file
cal_type_1 <- rBiasCorrection::example.data_calibration
rv$fileimport_calibration <- cal_type_1$dat
rv$vec_cal <- cal_type_1$vec_cal


# perform regression
regression_results <- regression_utility(
  rv$fileimport_calibration,
  "Testlocus",
  locus_id = NULL,
  rv = rv,
  mode = NULL,
```

```
  logfilename,
  minmax = rv$minmax,
  seed = rv$seed
)

# extract regression results
rv$result_list <- regression_results$result_list

# get regression statistics
rv$reg_stats <- statistics_list(
  rv$result_list,
  minmax = TRUE
)

# select the better model based on the sum of squared errrors ("SSE")
rv$choices_list <- better_model(
  statstable_pre = rv$reg_stats,
  selection_method = "SSE"
)

# correct calibration data (to show corrected calibration curves)
solved_eq_h <- solving_equations(datatable = rv$fileimport_calibration,
                                 regmethod = rv$choices_list,
                                 type = 1,
                                 rv = rv,
                                 mode = "corrected",
                                 logfilename = logfilename,
                                 minmax = rv$minmax)
rv$fileimport_cal_corrected_h <- solved_eq_h$results
colnames(rv$fileimport_cal_corrected_h) <- colnames(
  rv$fileimport_calibration
)

# calculate new calibration curves from corrected calibration data
regression_results <- regression_utility(
  data = rv$fileimport_cal_corrected_h,
  samplelocusname = rv$sample_locus_name,
  rv = rv,
  mode = "corrected",
  logfilename = logfilename,
  minmax = rv$minmax,
  seed = rv$seed
)
rv$result_list_hyperbolic <- regression_results$result_list


# save regression statistics to reactive value
rv$reg_stats_corrected_h <- statistics_list(
  resultlist = rv$result_list_hyperbolic,
  minmax = rv$minmax
)

createbarerrorplots(
```

```
  statstable_pre = rv$reg_stats,
  statstable_post = rv$reg_stats_corrected_h,
  rv = rv,
  type = 1,
  locus_id = NULL,
  plotdir = rv$plotdir,
  logfilename = logfilename,
  mode = "corrected_h",
  plot_height = 5,
  plot_width = 7.5,
  plot_textsize = 1
)
```

`create_exampleplot` *create_exampleplot helper function*

Description

Internal function to create an example plot.

Usage

```
create_exampleplot(
  data,
  coef_hyper,
  coef_cubic,
  plot_height,
  plot_width,
  plot_textsize,
  filename
)
```

Arguments

`data`	A data.table containing the aggregated calibration data.
`coef_hyper`	A list containing the regression parameters of the hyperbolic regression equation.
`coef_cubic`	A list containing the regression parameters of the cubic regression equation.
`plot_height`	A integer value. The height (unit: inch) of the resulting plots (default: 5).
`plot_width`	A integer value. The width (unit: inch) of the resulting plots (default: 7.5).
`plot_textsize`	A integer value. The textsize of the resulting plots (default: 16).
`filename`	A character. The filename, where to store the resulting example plot.

Value

The function creates an example plot and stores on the local filesystem.

Examples

```
gdat <- rBiasCorrection::example._plot.df_agg

coef_h <- rBiasCorrection::example._plot_coef_h
coef_c <- rBiasCorrection::example._plot_coef_c

create_exampleplot(
  data = gdat,
  coef_hyper = coef_h,
  coef_cubic = coef_c,
  plot_height = 5,
  plot_width = 7.5,
  plot_textsize = 1,
  filename = paste0(tempdir(), "/exampleplot.png")
)
```

`example.data_calibration` *example.data_calibration*

Description

A list containing the calibration data ($dat) and the colnames ($vec_cal) needed by the algorithms to perform the bias correction.

`example.data_experimental` *example.data_experimental*

Description

A list containing the experimental data ($dat) and the colnames ($vec_cal) needed by the algorithms to perform the bias correction.

`example._plot.df_agg` *example._plot.df_agg*

Description

A data.table containing the aggregated calibration data for CpG site 1 to create an example plot.

example._plot_coef_c *example._plot_coef_c*

Description

A list containing exemplary coefficients of the cubic regression equation for CpG site 1 to create an example plot.

example._plot_coef_h *example._plot_coef_h*

Description

A list containing exemplary coefficients of the hyperbolic regression equation for CpG site 1 to create an example plot.

get_timestamp *get_timestamp helper function*

Description

Internal function to get the current timestamp to write it to filenames.

Usage

```
get_timestamp()
```

Value

This function takes no argument and returns a formatted timestamp of the current system time, which can be integrated e.g. into a filename.

See Also

Sys.time

Examples

```
get_timestamp()
```

handle_text_input *handle_text_input helper function*

Description

Internal function to remove punctuation and unneeded stuff from user inputs with regular expressions.

Usage

```
handle_text_input(textinput)
```

Arguments

textinput	A character string with the textinput to perform these predefined regular expressions on.

Value

This function returns a cleaned up character string, limited to a maximum of 15 chars.

Examples

```
textinput <- "This is a dirty! text."
handle_text_input(textinput)
```

on_start *on_start helper function*

Description

Internal function, that initializes plotdir, csvdir and logfilename.

Usage

```
on_start(plotdir, csvdir, logfilename, parallel)
```

Arguments

plotdir	A character string. Path to the folder, where plots are saved.
csvdir	A character string. Path to the folder, where resulting tables are saved.
logfilename	A character string. Path to the logfile to save the log messages.
parallel	A boolean. If TRUE (the default value), initializing ‘future::plan("multiprocess")‘ before running the code.

Value

This function silently creates the directories‘plotdir‘ and ‘csvdir‘ on the local filesystem and initializes the logfile, specified with ‘logfilename‘. Furthermore, if ‘parallel = TRUE‘, the ‘future‘-backend is initialized.

See Also

plan

Examples

```
plotdir <- paste0(tempdir(), "/plots/")
csvdir <- paste0(tempdir(), "/csv/")
logfilename <- paste0(tempdir(), "/log.txt")
parallel <- FALSE

on_start(plotdir, csvdir, logfilename, parallel)
```

---

plotting_utility *plotting_utility helper function*

Description

Internal function to carry out the plotting of the calibrations curves.

Usage

```
plotting_utility(
  data,
  plotlist_reg,
  type,
  samplelocusname,
  locus_id = NULL,
  rv,
  mode = NULL,
  plotdir,
  logfilename,
  minmax,
  plot_height = 5,
  plot_width = 7.5,
  plot_textsize = 1
)
```

Arguments

data	A data.table object that contains the calibration data.
plotlist_reg	A list object contating regression plots without regression curves (output of regression_utility()).
type	A single integer. Type of data to be corrected: either "1" (one locus in many samples, e.g. pyrosequencing data) or "2" (many loci in one sample, e.g. next-generation sequencing data or microarray data).
samplelocusname	A character string. In type 1 data: locus name - name of the gene locus under investigation. In type 2 data: sample name - name of the sample under investigation.
locus_id	A character string. Default: NULL. ID of the respective locus (only used in type 2 correction).
rv	A list object. A list that contains additional objects needed for the algorithms.
mode	A character string. Default: NULL. Used to indicate "corrected" calibration data.
plotdir	A character string. Path to the folder, where plots are saved.
logfilename	A character string. Path to a file to save the log messages (default = paste0(tempdir(), "/log.txt")).
minmax	A logical, indicating which equations are used for BiasCorrection (default: FALSE). If TRUE, equations are used that include the respective minima and maxima of the provided data.
plot_height	A integer value. The height (unit: inch) of the resulting plots (default: 5).
plot_width	A integer value. The width (unit: inch) of the resulting plots (default: 7.5).
plot_textsize	A integer value. The textsize of the resulting plots (default: 16).

Value

This function creates calibration plots and writes them to the local filesystem.

Examples

```
# define list object to save all data
rv <- list()
rv$minmax <- TRUE
rv$selection_method <- "RelError"
rv$sample_locus_name <- "Test"
rv$seed <- 1234

# define logfilename
logfilename <- paste0(tempdir(), "/log.txt")

# define plotdir
rv$plotdir <- paste0(tempdir(), "/plots/")
dir.create(rv$plotdir)
```

```
# import experimental file
exp_type_1 <- rBiasCorrection::example.data_experimental
rv$fileimport_experimental <- exp_type_1$dat

# import calibration file
cal_type_1 <- rBiasCorrection::example.data_calibration
rv$fileimport_calibration <- cal_type_1$dat
rv$vec_cal <- cal_type_1$vec_cal


# perform regression
regression_results <- regression_utility(
  rv$fileimport_calibration,
  "Testlocus",
  locus_id = NULL,
  rv = rv,
  mode = NULL,
  logfilename,
  minmax = rv$minmax,
  seed = rv$seed
)

# extract the plotlist
plotlist_reg <- regression_results$plot_list

plotting_utility(
  data = rv$fileimport_calibration,
  plotlist_reg = plotlist_reg,
  type = 1,
  samplelocusname = rv$sample_locus_name,
  locus_id = NULL,
  rv = rv,
  mode = NULL,
  plotdir = rv$plotdir,
  logfilename = logfilename,
  minmax = rv$minmax,
  plot_height = 5,
  plot_width = 7.5,
  plot_textsize = 1
)
```

`regression_utility` *regression_utility helper function*

Description

Internal function to carry out the regression calculations.

Usage

```
regression_utility(
  data,
  samplelocusname,
  locus_id = NULL,
  rv,
  mode = NULL,
  logfilename,
  minmax,
  seed = 1234
)
```

Arguments

data	A data.table object that contains the calibration data.
samplelocusname	A character string. In type 1 data: locus name - name of the gene locus under investigation. In type 2 data: sample name - name of the sample under investigation.
locus_id	A character string. Default: NULL. ID of the respective locus (only used in type 2 correction).
rv	A list object. A list that contains additional objects needed for the algorithms.
mode	A character string. Default: NULL. Used to indicate "corrected" calibration data.
logfilename	A character string. Path to a file to save the log messages (default = paste0(tempdir(), "/log.txt")).
minmax	A logical, indicating which equations are used for BiasCorrection (default: FALSE). If TRUE, equations are used that include the respective minima and maxima of the provided data.
seed	A integer value. The seed used when solving the unknowns in the hyperbolic regression equation and the cubic regression equation. Important for reproducibility (default: 1234).

Value

The function performs the regression calculations and returns the results in a list.

Examples

```
# define list object to save all data
rv <- list()
rv$minmax <- TRUE
rv$selection_method <- "RelError"
rv$sample_locus_name <- "Test"
rv$seed <- 1234

# define logfilename
```

```
logfilename <- paste0(tempdir(), "/log.txt")

# import experimental file
exp_type_1 <- rBiasCorrection::example.data_experimental
rv$fileimport_experimental <- exp_type_1$dat

# import calibration file
cal_type_1 <- rBiasCorrection::example.data_calibration
rv$fileimport_calibration <- cal_type_1$dat
rv$vec_cal <- cal_type_1$vec_cal

# perform regression
regression_results <- regression_utility(
  rv$fileimport_calibration,
  "Testlocus",
  locus_id = NULL,
  rv = rv,
  mode = NULL,
  logfilename,
  minmax = rv$minmax,
  seed = rv$seed
)
length(regression_results)
class(regression_results)
```

`solving_equations` *solving_equations helper function*

Description

Internal function to solve the hyperbolic and cubic regression.

Usage

```
solving_equations(
  datatable,
  regmethod,
  type,
  rv,
  mode = NULL,
  logfilename,
  minmax
)
```

Arguments

`datatable`	A data.table object that contains either the experimental data or the calibration data.

regmethod	A data.table object, with 2 columns, containing the names of the samples to correct (columns 1) and a binary variable *better_model* that indicates, if the data should be corrected with the hyperbolic regression parameters (better_model = 0) or with the cubic regression parameters (better_model = 1).
type	A single integer. Type of data to be corrected: either "1" (one locus in many samples, e.g. pyrosequencing data) or "2" (many loci in one sample, e.g. next-generation sequencing data or microarray data).
rv	A list object. A list that contains additional objects needed for the algorithms.
mode	A character string. Default: NULL. Used to indicate "corrected" calibration data.
logfilename	A character string. Path to the logfile to save the log messages.
minmax	A logical, indicating which equations are used for BiasCorrection (default: FALSE). If TRUE, equations are used that include the respective minima and maxima of the provided data.

Value

This function solves the equations of the hyperbolic and the cubic regression and returns the respectively interpolated values of the provided ‘datatable‘.

Examples

```
# define list object to save all data
rv <- list()
rv$minmax <- TRUE
rv$selection_method <- "RelError"
rv$sample_locus_name <- "Test"
rv$seed <- 1234

# define logfilename
logfilename <- paste0(tempdir(), "/log.txt")

# import experimental file
exp_type_1 <- rBiasCorrection::example.data_experimental
rv$fileimport_experimental <- exp_type_1$dat

# import calibration file
cal_type_1 <- rBiasCorrection::example.data_calibration
rv$fileimport_calibration <- cal_type_1$dat
rv$vec_cal <- cal_type_1$vec_cal


# perform regression
regression_results <- regression_utility(
  rv$fileimport_calibration,
  "Testlocus",
  locus_id = NULL,
  rv = rv,
  mode = NULL,
```

```
  logfilename,
  minmax = rv$minmax,
  seed = rv$seed
)

# extract regression results
rv$result_list <- regression_results$result_list

# get regression statistics
rv$reg_stats <- statistics_list(
  rv$result_list,
  minmax = TRUE
)

# select the better model based on the sum of squared errrors ("SSE")
rv$choices_list <- better_model(
  statstable_pre = rv$reg_stats,
  selection_method = "SSE"
)

# correct calibration data (to show corrected calibration curves)
solved_eq_h <- solving_equations(datatable = rv$fileimport_calibration,
                                 regmethod = rv$choices_list,
                                 type = 1,
                                 rv = rv,
                                 mode = "corrected",
                                 logfilename = logfilename,
                                 minmax = rv$minmax)
rv$fileimport_cal_corrected_h <- solved_eq_h$results
colnames(rv$fileimport_cal_corrected_h) <- colnames(
  rv$fileimport_calibration
)
```

`statistics_list` *statistics_list helper function*

Description

Internal function that converts the results_list (output of `regression_utility()`) into a data.table object.

Usage

```
statistics_list(resultlist, minmax = FALSE)
```

Arguments

resultlist	A list object. The results_list output of regression_utility().
minmax	A logical, indicating which equations are used for BiasCorrection (default: FALSE). If TRUE, equations are used that include the respective minima and maxima of the provided data.

Value

The function takes the ‘resultslist‘ and converts it to a statistics list, which is basically a ‘data.table‘ with the results of the hyperbolic and the cubic regression.

Examples

```
# define list object to save all data
rv <- list()
rv$minmax <- TRUE
rv$selection_method <- "RelError"
rv$sample_locus_name <- "Test"
rv$seed <- 1234

# define logfilename
logfilename <- paste0(tempdir(), "/log.txt")

# import experimental file
exp_type_1 <- rBiasCorrection::example.data_experimental
rv$fileimport_experimental <- exp_type_1$dat

# import calibration file
cal_type_1 <- rBiasCorrection::example.data_calibration
rv$fileimport_calibration <- cal_type_1$dat
rv$vec_cal <- cal_type_1$vec_cal

# perform regression
regression_results <- regression_utility(
  rv$fileimport_calibration,
  "Testlocus",
  locus_id = NULL,
  rv = rv,
  mode = NULL,
  logfilename,
  minmax = rv$minmax,
  seed = rv$seed
)

# extract regression results
rv$result_list <- regression_results$result_list

# get regression statistics
rv$reg_stats <- statistics_list(
  rv$result_list,
  minmax = rv$minmax
```

```
)
```

`substitutions_create` *substitutions_create helper function*

Description

Internal function to initialize a data.table object to store the substitutions.

Usage

```
substitutions_create()
```

Value

This function takes no argument and initializes an empty ‘data.table‘ to hold the substituted values, if substitutions occur during BiasCorrection.

Examples

```
substitutions <- substitutions_create()
class(substitutions)
```

`write_csv` *write_csv helper function*

Description

Internal function to store the created tables in csv files.

Usage

```
write_csv(table, filename)
```

Arguments

`table`	A data.table object to store on the local file system
`filename`	The file name (including the path) to store `table`.

Value

This function silently writes a ‘data.table‘ object to a CSV file.

See Also

fwrite

Examples

```
table <- data.table::data.table(
  a = stats::runif(1000),
  b = stats::runif(1000)
)

write_csv(table, paste0(tempdir(), "/example.csv"))
```

`write_log` *write_log helper function*

Description

Internal function to write log-messages to the file specified in logfilename.

Usage

```
write_log(message, logfilename)
```

Arguments

`message`	A character string containing the log message.
`logfilename`	A character string. Path to the logfile to save the log messages.

Value

The function prints the logging message to the console and writes it to the local logfile, specified with ‘logfilename‘.

Examples

```
message <- "This is a logmessage"
logfilename <- paste0(tempdir(), "/log.txt")

write_log(message, logfilename)
```

Index