Package 'sbpiper'

June 26, 2018

Version 1.9.0

Date 2018-06-26

Title Data Analysis Functions for 'SBpipe' Package

Depends R (>= 3.2.0)

Imports colorRamps, data.table, factoextra, FactoMineR, ggplot2 (>= 2.2.0), grDevices, Hmisc, reshape2, scales, stats, stringr, utils

Description Provides an API for analysing repetitive parameter estimations and simulations of mathematical models. Examples of mathematical models are Ordinary Differential equations (ODEs)
or Stochastic Differential Equations (SDEs) models. Among the analyses for parameter estimation 'sbpiper' calculates statistics and generates plots for parameter density, PCA of the best fits, parameter profile likelihood estimations (PLEs), and 2D parameter PLEs. These results can be generated using all or a subset of the best computed parameter sets. Among the analyses for model simulation 'sbpiper' calculates statistics and generates plots for deterministic and stochastic time courses via cartesian and heatmap plots. Plots for the scan of one or two model parameters can also be generated. This package is primarily used by the software 'SBpipe'. Citation: Dalle Pezze P, Le Novère N. SBpipe: a collection of pipelines for automating repetitive simulation and analysis tasks. BMC Systems Biology. 2017;11:46. <doi:10.1186/s12918-017-0423-3>.

License MIT + file LICENSE

URL https://github.com/pdp10/sbpiper

BugReports https://github.com/pdp10/sbpiper/issues

LazyLoad true

Collate 'sbpiper.r' 'sbpiper_ggplot2_themes.r' 'sbpiper_plots.r' 'sbpiper_sim.r' 'sbpiper_ps1.r' 'sbpiper_ps2.r' 'sbpiper_pe.r' 'data.r'

RoxygenNote 6.0.1

Encoding UTF-8

NeedsCompilation no

Author Piero Dalle Pezze [aut, cre, cph] (<https://orcid.org/0000-0003-1695-6763>)

Maintainer Piero Dalle Pezze <piero.dallepezze@gmail.com>

Repository CRAN

Date/Publication 2018-06-26 15:54:28 UTC

R topics documented:

`basic_theme` *A generic basic theme for time courses. It extends ggplot2 theme_classic().*

Description

A generic basic theme for time courses. It extends ggplot2 theme_classic().

Usage

```
basic_theme(base_size = 12, base_family = "")
```

Arguments

base_size	the font size
base_family	the font family

Examples

```
library(ggplot2)
theme_set(basic_theme(36))
```

---

check_exp_dataset *Check that the experimental data set exists.*

Description

Check that the experimental data set exists.

Usage

```
check_exp_dataset(exp_dataset, plot_exp_dataset = FALSE)
```

Arguments

exp_dataset	a full path file containing the experimental data.
plot_exp_dataset	TRUE if the data set file should be plotted.

Value

TRUE if the file exists.

---

combine_param_best_fits_stats *Combine the parameter best fits statistics.*

Description

Combine the parameter best fits statistics.

Usage

```
combine_param_best_fits_stats(plots_dir, fileout_param_estim_best_fits_details)
```

Arguments

plots_dir	the directory to save the generated plots
fileout_param_estim_best_fits_details	the name of the file containing the statistics for the parameter best fits.

`combine_param_ple_stats` *Combine the parameter PLE statistics.*

Description

Combine the parameter PLE statistics.

Usage

```
combine_param_ple_stats(plots_dir, fileout_param_estim_details)
```

Arguments

`plots_dir`	the directory to save the generated plots
`fileout_param_estim_details`	the name of the file containing the detailed statistics for the estimated parameters

`compute_aic` *Compute the Akaike Information Criterion. Assuming additive Gaussian measurement noise of width 1, the term -2ln(L(theta\|y)) ~ SSR ~ Chi^2*

Description

Compute the Akaike Information Criterion. Assuming additive Gaussian measurement noise of width 1, the term -2ln(L(thetaly)) ~ SSR ~ Chi^2

Usage

```
compute_aic(chi2 = 0, params = 0)
```

Arguments

`chi2`	the Chi^2 for the model
`params`	the number of model parameters

Value

the AIC

Examples

```
compute_aic(chi2=10, params=5)
```

compute_aicc *Compute the corrected Akaike Information Criterion. Assuming additive Gaussian measurement noise of width 1, the term -2ln(L(theta|y)) ~ SSR ~ Chi^2*

Description

Compute the corrected Akaike Information Criterion. Assuming additive Gaussian measurement noise of width 1, the term -2ln(L(theta|y)) ~ SSR ~ Chi^2

Usage

```
compute_aicc(chi2 = 0, params = 0, data_points = 0)
```

Arguments

chi2	the Chi^2 for the model
params	the number of model parameters
data_points	the number of data points

Value

the AICc

Examples

```
compute_aicc(chi2=10, params=5, data_points=100)
```

compute_bic *Compute the Bayesian Information Criterion. Assuming additive Gaussian measurement noise of width 1, the term -2ln(L(theta|y)) ~ SSR ~ Chi^2*

Description

Compute the Bayesian Information Criterion. Assuming additive Gaussian measurement noise of width 1, the term -2ln(L(theta|y)) ~ SSR ~ Chi^2

Usage

```
compute_bic(chi2 = 0, params = 1, data_points = 1)
```

Arguments

chi2	the Chi^2 for the model
params	the number of model parameters
data_points	the number of data points

Value

the BIC

Examples

```
compute_bic(chi2=10, params=5, data_points=100)
```

`compute_cl_objval` *Compute the confidence level based on the minimum objective value.*

Description

Compute the confidence level based on the minimum objective value.

Usage

```
compute_cl_objval(min_objval = 0, params = 1, data_points = 2,
  level = 0.05)
```

Arguments

`min_objval`	the minimum objective value
`params`	the number of parameters
`data_points`	the number of data points
`level`	the confidence level threshold (e.g. 0.01, 0.05)

Value

the confidence level based on minimum objective value

Examples

```
compute_cl_objval(min_objval=10, params=5, data_points=100)
```

compute_fratio_threshold

Compute the fratio threshold for the confidence level.

Description

Compute the fratio threshold for the confidence level.

Usage

```
compute_fratio_threshold(params = 1, data_points = 2, level = 0.05)
```

Arguments

params	the number of parameters
data_points	the number of data points
level	the confidence level threshold (e.g. 0.01, 0.05)

Value

the f-ratio threshold

Examples

```
compute_fratio_threshold(params=5, data_points=100)
compute_fratio_threshold(params=5, data_points=100, level=0.01)
```

compute_sampled_ple_stats

Compute the table for the sampled PLE statistics.

Description

Compute the table for the sampled PLE statistics.

Usage

```
compute_sampled_ple_stats(df, min_objval, cl66_objval, cl95_objval, cl99_objval,
  logspace = TRUE)
```

Arguments

df	the complete data frame
min_objval	the minimum objective value
cl66_objval	the 66% confidence level objective value
cl95_objval	the 95% confidence level objective value
cl99_objval	the 99% confidence level objective value
logspace	true if parameters are plotted in logspace (default: TRUE)

Value

the list of parameter values with their confidence intervals

Examples

```
data(insulin_receptor_all_fits)
colnames(insulin_receptor_all_fits)[1] <- "ObjVal"
min_objval <- min(insulin_receptor_all_fits[,1])
# compute the stats for parameter k2.
insulin_receptor_all_fits <- subset(insulin_receptor_all_fits, select=c(1,3))
compute_sampled_ple_stats(df=insulin_receptor_all_fits,
                          min_objval=min_objval,
                          cl66_objval=min_objval+0.01,
                          cl95_objval=min_objval+0.02,
                          cl99_objval=min_objval+0.03,
                          logspace=FALSE)
```

---

gen_stats_table *Generate a table of statistics for each model readout.*

Description

Generate a table of statistics for each model readout.

Usage

```
gen_stats_table(inputfile, outputfile, column_to_read = "X1")
```

Arguments

inputfile	the file to store the simulated repeats
outputfile	the file to store the statistics
column_to_read	the name of the column to process

Examples

```
data(insulin_receptor_1)
data(insulin_receptor_2)
data(insulin_receptor_3)
dir.create(file.path("sim_datasets"))
dir.create(file.path("sim_datasets_sum"))
write.table(insulin_receptor_1,
            file=file.path("sim_datasets","insulin_receptor_1.csv"),
            row.names=FALSE)
write.table(insulin_receptor_2,
            file=file.path("sim_datasets","insulin_receptor_2.csv"),
            row.names=FALSE)
write.table(insulin_receptor_3,
            file=file.path("sim_datasets","insulin_receptor_3.csv"),
            row.names=FALSE)
summarise_data(inputdir="sim_datasets",
               model="insulin_receptor",
               outputfile=file.path("sim_datasets_sum",
                                    "insulin_receptor_IR_beta_pY1146.csv"),
               column_to_read="IR_beta_pY1146")
gen_stats_table(inputfile=file.path("sim_datasets_sum",
                                    "insulin_receptor_IR_beta_pY1146.csv"),
                outputfile="insulin_receptor_IR_beta_pY1146_stats.csv",
                column_to_read="IR_beta_pY1146")
```

---

get_param_names *Get parameter names*

Description

Get parameter names

Usage

```
get_param_names(filename)
```

Arguments

filename	the filename containing the best fits

Value

the parameter names

get_sorted_level_indexes *Return the indexes of the files as sorted by levels.*

Description

Return the indexes of the files as sorted by levels.

Usage

```
get_sorted_level_indexes(files)
```

Arguments

files	the scanned files.

Value

the index of the levels

histogramplot *Plot a generic histogram*

Description

Plot a generic histogram

Usage

```
histogramplot(dfCol, g = ggplot())
```

Arguments

dfCol	a data frame with exactly one column.
g	the current ggplot to overlap

Value

the plot

Examples

```
a <- as.data.frame(rnorm(100))
histogramplot(dfCol=a)
```

insulin_receptor_1 *A stochastic model simulation*

Description

A stochastic model simulation

Usage

```
insulin_receptor_1
```

Format

A data frame with 2 variables:

Time The time variable

IR_beta_pY1146 The insulin receptor beta phosphorylated at pY1146

insulin_receptor_2 *A stochastic model simulation*

Description

A stochastic model simulation

Usage

```
insulin_receptor_2
```

Format

A data frame with 2 variables:

Time The time variable

IR_beta_pY1146 The insulin receptor beta phosphorylated at pY1146

insulin_receptor_3 *A stochastic model simulation*

Description

A stochastic model simulation

Usage

```
insulin_receptor_3
```

Format

A data frame with 2 variables:

Time The time variable

IR_beta_pY1146 The insulin receptor beta phosphorylated at pY1146

insulin_receptor_all_fits *A parameter estimation data set including all the evaluated fits.*

Description

Estimated parameters for a mini model of the insulin receptor (IRbeta).

Usage

```
insulin_receptor_all_fits
```

Format

A data frame with 4 variables:

ObjectiveValue the objective value for this parameter set

k1 First estimated parameter: kinetic rate constant for IRbeta phosphorylation.

k2 Second estimated parameter: kinetic rate constant for IRbeta refractory status.

k3 Third estimated parameter: kinetic rate constant for IRbeta dephosphorylation.

`insulin_receptor_best_fits`

A parameter estimation data set including only the best evaluated fits.

Description

Estimated parameters for a mini model of the insulin receptor (IRbeta).

Usage

```
insulin_receptor_best_fits
```

Format

A data frame with 4 variables:

Estimation the number of estimated parameter sets

ObjectiveValue the best objective value for this parameter estimation

k1 First estimated parameter: kinetic rate constant for IRbeta phosphorylation.

k2 Second estimated parameter: kinetic rate constant for IRbeta refractory status.

k3 Third estimated parameter: kinetic rate constant for IRbeta dephosphorylation.

`insulin_receptor_exp_dataset`

Experimental data set for the insulin receptor beta phosphorylated at pY1146 as published in Dalle Pezze et al. Science Signaling 2012.

Description

Experimental data set for the insulin receptor beta phosphorylated at pY1146 as published in Dalle Pezze et al. Science Signaling 2012.

Usage

```
insulin_receptor_exp_dataset
```

Format

A data frame with 2 variables:

Time The time variable

IR_beta_pY1146 The insulin receptor beta phosphorylated at pY1146

`insulin_receptor_IR_beta_pY1146` *A stochastic simulation data set for the insulin receptor beta phosphorylated at pY1146.*

Description

Independent stochastic simulation time courses for the phosphorylated insulin receptor upon insulin stimulation. This data set is summarised.

Usage

```
insulin_receptor_IR_beta_pY1146
```

Format

A data frame with 41 variables:

Time The time variable

X1 A stochastic simulation

X2 A stochastic simulation

X3 A stochastic simulation

X4 A stochastic simulation

X5 A stochastic simulation

X6 A stochastic simulation

X7 A stochastic simulation

X8 A stochastic simulation

X9 A stochastic simulation

X10 A stochastic simulation

X11 A stochastic simulation

X12 A stochastic simulation

X13 A stochastic simulation

X14 A stochastic simulation

X15 A stochastic simulation

X16 A stochastic simulation

X17 A stochastic simulation

X18 A stochastic simulation

X19 A stochastic simulation

X20 A stochastic simulation

X21 A stochastic simulation

X22 A stochastic simulation

X23 A stochastic simulation

X24 A stochastic simulation

X25 A stochastic simulation

X26 A stochastic simulation

X27 A stochastic simulation

X28 A stochastic simulation

X29 A stochastic simulation

X30 A stochastic simulation

X31 A stochastic simulation

X32 A stochastic simulation

X33 A stochastic simulation

X34 A stochastic simulation

X35 A stochastic simulation

X36 A stochastic simulation

X37 A stochastic simulation

X38 A stochastic simulation

X39 A stochastic simulation

X40 A stochastic simulation

`insulin_receptor_ps1_l0`

A deterministic simulation of the insulin receptor model upon scanning of 1 model parameter. The initial amount of IR-beta is approx 0.

Description

A deterministic simulation of the insulin receptor model upon scanning of 1 model parameter. The initial amount of IR-beta is approx 0.

Usage

```
insulin_receptor_ps1_l0
```

Format

A data frame with 3 variables:

Time The time variable.

IR_beta The unphosphorylated state of the insulin receptor beta. The scanned variable.

IR_beta_pY1146 The insulin receptor beta phosphorylated at pY1146

`insulin_receptor_ps1_l1`

A deterministic simulation of the insulin receptor model upon scanning of 1 model parameter. The initial amount of IR-beta is approx 1.

Description

A deterministic simulation of the insulin receptor model upon scanning of 1 model parameter. The initial amount of IR-beta is approx 1.

Usage

```
insulin_receptor_ps1_l1
```

Format

A data frame with 3 variables:

Time The time variable.

IR_beta The unphosphorylated state of the insulin receptor beta. The scanned variable.

IR_beta_pY1146 The insulin receptor beta phosphorylated at pY1146

`insulin_receptor_ps1_l11`

A deterministic simulation of the insulin receptor model upon scanning of 1 model parameter. The initial amount of IR-beta is approx 11.

Description

A deterministic simulation of the insulin receptor model upon scanning of 1 model parameter. The initial amount of IR-beta is approx 11.

Usage

```
insulin_receptor_ps1_l11
```

Format

A data frame with 3 variables:

Time The time variable.

IR_beta The unphosphorylated state of the insulin receptor beta. The scanned variable.

IR_beta_pY1146 The insulin receptor beta phosphorylated at pY1146

`insulin_receptor_ps1_l13`

A deterministic simulation of the insulin receptor model upon scanning of 1 model parameter. The initial amount of IR-beta is approx 13.

Description

A deterministic simulation of the insulin receptor model upon scanning of 1 model parameter. The initial amount of IR-beta is approx 13.

Usage

```
insulin_receptor_ps1_l13
```

Format

A data frame with 3 variables:

Time The time variable.

IR_beta The unphosphorylated state of the insulin receptor beta. The scanned variable.

IR_beta_pY1146 The insulin receptor beta phosphorylated at pY1146

`insulin_receptor_ps1_l14`

A deterministic simulation of the insulin receptor model upon scanning of 1 model parameter. The initial amount of IR-beta is approx 14.

Description

A deterministic simulation of the insulin receptor model upon scanning of 1 model parameter. The initial amount of IR-beta is approx 14.

Usage

```
insulin_receptor_ps1_l14
```

Format

A data frame with 3 variables:

Time The time variable.

IR_beta The unphosphorylated state of the insulin receptor beta. The scanned variable.

IR_beta_pY1146 The insulin receptor beta phosphorylated at pY1146

`insulin_receptor_ps1_l16`

A deterministic simulation of the insulin receptor model upon scanning of 1 model parameter. The initial amount of IR-beta is approx 16.

Description

A deterministic simulation of the insulin receptor model upon scanning of 1 model parameter. The initial amount of IR-beta is approx 16.

Usage

```
insulin_receptor_ps1_l16
```

Format

A data frame with 3 variables:

Time The time variable.

IR_beta The unphosphorylated state of the insulin receptor beta. The scanned variable.

IR_beta_pY1146 The insulin receptor beta phosphorylated at pY1146

`insulin_receptor_ps1_l3`

A deterministic simulation of the insulin receptor model upon scanning of 1 model parameter. The initial amount of IR-beta is approx 3.

Description

A deterministic simulation of the insulin receptor model upon scanning of 1 model parameter. The initial amount of IR-beta is approx 3.

Usage

```
insulin_receptor_ps1_l3
```

Format

A data frame with 3 variables:

Time The time variable.

IR_beta The unphosphorylated state of the insulin receptor beta. The scanned variable.

IR_beta_pY1146 The insulin receptor beta phosphorylated at pY1146

`insulin_receptor_ps1_l4`

A deterministic simulation of the insulin receptor model upon scanning of 1 model parameter. The initial amount of IR-beta is approx 4.

Description

A deterministic simulation of the insulin receptor model upon scanning of 1 model parameter. The initial amount of IR-beta is approx 4.

Usage

```
insulin_receptor_ps1_l4
```

Format

A data frame with 3 variables:

Time The time variable.

IR_beta The unphosphorylated state of the insulin receptor beta. The scanned variable.

IR_beta_pY1146 The insulin receptor beta phosphorylated at pY1146

`insulin_receptor_ps1_l6`

A deterministic simulation of the insulin receptor model upon scanning of 1 model parameter. The initial amount of IR-beta is approx 6.

Description

A deterministic simulation of the insulin receptor model upon scanning of 1 model parameter. The initial amount of IR-beta is approx 6.

Usage

```
insulin_receptor_ps1_l6
```

Format

A data frame with 3 variables:

Time The time variable.

IR_beta The unphosphorylated state of the insulin receptor beta. The scanned variable.

IR_beta_pY1146 The insulin receptor beta phosphorylated at pY1146

`insulin_receptor_ps1_l8`

A deterministic simulation of the insulin receptor model upon scanning of 1 model parameter. The initial amount of IR-beta is approx 8.

Description

A deterministic simulation of the insulin receptor model upon scanning of 1 model parameter. The initial amount of IR-beta is approx 8.

Usage

```
insulin_receptor_ps1_l8
```

Format

A data frame with 3 variables:

Time The time variable.

IR_beta The unphosphorylated state of the insulin receptor beta. The scanned variable.

IR_beta_pY1146 The insulin receptor beta phosphorylated at pY1146

`insulin_receptor_ps1_l9`

A deterministic simulation of the insulin receptor model upon scanning of 1 model parameter. The initial amount of IR-beta is approx 9.

Description

A deterministic simulation of the insulin receptor model upon scanning of 1 model parameter. The initial amount of IR-beta is approx 9.

Usage

```
insulin_receptor_ps1_l9
```

Format

A data frame with 3 variables:

Time The time variable.

IR_beta The unphosphorylated state of the insulin receptor beta. The scanned variable.

IR_beta_pY1146 The insulin receptor beta phosphorylated at pY1146

insulin_receptor_ps2_tp2

A deterministic simulation of the insulin receptor model upon scanning of 2 model parameters.

Description

A deterministic simulation of the insulin receptor model upon scanning of 2 model parameters.

Usage

```
insulin_receptor_ps2_tp2
```

Format

A data frame with 4 variables:

Time The time variable. This dataset is at time T=2min upon insulin stimulation

IR_beta_pY1146 The insulin receptor beta phosphorylated at pY1146

IRbetaPercent The percent of available IR_beta amount.

InsulinPercent The percent of available insulin amount.

kurtosis

Calculate the kurtosis of a numeric vector

Description

Calculate the kurtosis of a numeric vector

Usage

```
kurtosis(x, na.rm = FALSE)
```

Arguments

x	the numeric vector
na.rm	TRUE if NA values should be discarded

Value

the kurtosis

Examples

```
kurtosis(x=c(1,2.4,5,NA), na.rm=TRUE)
```

leftCI *Return the left value of the parameter confidence interval. The provided dataset has two columns: ObjVal | ParamValue*

Description

Return the left value of the parameter confidence interval. The provided dataset has two columns: ObjVal | ParamValue

Usage

```
leftCI(smallest.param.value, full_dataset, cl_objval)
```

Arguments

smallest.param.value	the smallest parameter value within the specified confidence level
full_dataset	the full dataset
cl_objval	the objective value at the desired confidence level

Value

the left confidence interval

Examples

```
data(insulin_receptor_all_fits)
colnames(insulin_receptor_all_fits)[1] <- "ObjVal"
min_objval <- min(insulin_receptor_all_fits[,1])
# compute the stats for parameter k2.
insulin_receptor_all_fits <- subset(insulin_receptor_all_fits, select=c(1,3))
leftCI(smallest.param.value=0.466971,
        full_dataset=insulin_receptor_all_fits,
        cl_objval=min_objval+0.01)
leftCI(smallest.param.value=0.467000,
        full_dataset=insulin_receptor_all_fits,
        cl_objval=min_objval+0.01)
```

load_exp_dataset *Load the experimental data set.*

Description

Load the experimental data set.

Usage

```
load_exp_dataset(exp_dataset, plot_exp_dataset = FALSE)
```

Arguments

exp_dataset	a full path file containing the experimental data.
plot_exp_dataset	TRUE if the experimental data should also be plotted

Value

the loaded data set.

---

normalise_vec *Normalise a vector within 0 and 1*

Description

Normalise a vector within 0 and 1

Usage

```
normalise_vec(vec, na.rm = TRUE)
```

Arguments

vec	the vector to normalise
na.rm	TRUE if NA values should be discarded

Value

the normalised vector

Examples

```
v <- c(-4,2,10,25,9,NA)
normalise_vec(vec=v)
```

objval.col *The name of the Objective Value column*

Description

The name of the Objective Value column

Usage

```
objval.col
```

Format

An object of class `character` of length 1.

objval_vs_iters_analysis *Analysis of the Objective values vs Iterations.*

Description

Analysis of the Objective values vs Iterations.

Usage

```
objval_vs_iters_analysis(model, filename, plots_dir)
```

Arguments

`model`	the model name
`filename`	the filename containing the fits sequence
`plots_dir`	the directory to save the generated plots

Examples

```
dir.create(file.path("pe_datasets"))
dir.create(file.path("pe_plots"))
data(insulin_receptor_all_fits)
colnames(insulin_receptor_all_fits)[1] <- "ObjVal"
write.table(insulin_receptor_all_fits,
            file=file.path("pe_datasets", "all_fits.csv"),
            row.names=FALSE)
objval_vs_iters_analysis(model="model",
                         filename=file.path("pe_datasets", "all_fits.csv"),
                         plots_dir="pe_plots")
```

parameter_density_analysis
Parameter density analysis.

Description

Parameter density analysis.

Usage

```
parameter_density_analysis(model, filename, parameter, plots_dir,
  thres = "BestFits", best_fits_percent = 50,
  fileout_param_estim_summary = "", logspace = TRUE,
  scientific_notation = TRUE)
```

Arguments

model	the model name
filename	the filename containing the fits sequence
parameter	the name of the parameter to plot the density
plots_dir	the directory for storing the plots
thres	the threshold used to filter the dataset. Values: "BestFits", "CL66", "CL95", "CL99", "All".
best_fits_percent	the percent of best fits to analyse. Only used if thres="BestFits".
fileout_param_estim_summary	the name of the file containing the summary for the parameter estimation. Only used if thres!="BestFits".
logspace	true if the parameters should be plotted in logspace
scientific_notation	true if the axis labels should be plotted in scientific notation

Examples

```
dir.create(file.path("pe_datasets"))
dir.create(file.path("pe_plots"))
data(insulin_receptor_all_fits)
write.table(insulin_receptor_all_fits,
            file=file.path("pe_datasets", "all_fits.csv"),
            row.names=FALSE)
# generate the global statistics for the parameter estimation
pe_ds_preproc(filename=file.path("pe_datasets", "all_fits.csv"),
              param.names=c('k1', 'k2', 'k3'),
              logspace=TRUE,
              all.fits=TRUE,
              data_point_num=33,
```

```
            fileout_param_estim_summary=file.path("pe_datasets", "param_estim_summary.csv"))
parameter_density_analysis(model="ir_beta",
                           filename=file.path("pe_datasets", "all_fits_log10.csv"),
                           parameter="k1",
                           plots_dir="pe_plots",
                           thres="CL95",
                           fileout_param_estim_summary=file.path("pe_datasets",
                                                                 "param_estim_summary.csv"),
                           logspace=TRUE)

data(insulin_receptor_best_fits)
write.table(insulin_receptor_best_fits,
            file=file.path("pe_datasets", "best_fits.csv"),
            row.names=FALSE)
# generate the global statistics for the parameter estimation
pe_ds_preproc(filename=file.path("pe_datasets", "best_fits.csv"),
              param.names=c('k1', 'k2', 'k3'),
              logspace=TRUE,
              all.fits=FALSE)
parameter_density_analysis(model="ir_beta",
                           filename=file.path("pe_datasets", "best_fits_log10.csv"),
                           parameter="k1",
                           plots_dir="pe_plots",
                           thres="BestFits",
                           best_fits_percent=50,
                           logspace=TRUE)
```

`parameter_pca_analysis` *PCA for the parameters. These plots rely on factoextra fviz functions.*

Description

PCA for the parameters. These plots rely on factoextra fviz functions.

Usage

```
parameter_pca_analysis(model, filename, plots_dir, best_fits_percent = 50,
  label.ind = "all", select.ind = NULL, repel.ind = TRUE,
  label.var = "all", select.var = NULL, repel.var = TRUE)
```

Arguments

`model`	the model name
`filename`	the filename containing the fits sequence
`plots_dir`	the directory to save the generated plots
`best_fits_percent`	the percent of best fits to analyse.

label.ind	parameter ‘label‘ passed to factoextra::fviz_pca_ind(). Labels shown if <= 75 and select.ind is NULL.
select.ind	parameter ‘select.ind‘ passed to factoextra::fviz_pca_ind().
repel.ind	parameter ‘repel‘ passed to factoextra::fviz_pca_ind()
label.var	parameter ‘label‘ passed to factoextra::fviz_pca_var().
select.var	parameter ‘select.var‘ passed to factoextra::fviz_pca_var().
repel.var	parameter ‘repel‘ passed to factoextra::fviz_pca_var() dir.create(file.path("pe_datasets")) dir.create(file.path("pe_plots")) data(insulin_receptor_best_fits) write.table(insulin_receptor_best_fits, file=file.path("pe_datasets", "best_fits.csv"), row.names=FALSE) # generate the global statistics for the parameter estimation pe_ds_preproc(filename=file.path("pe_datasets", "best_fits.csv"), param.names=c('k1', 'k2', 'k3'), logspace=TRUE, all.fits=FALSE) parameter_pca_analysis(model="ir_beta", filename=file.path("pe_datasets", "best_fits_log10.csv"), plots_dir="pe_plots", best_fits_percent=50)

---

pca_theme *A generic basic theme for pca. It extends ggplot2 theme_classic().*

Description

A generic basic theme for pca. It extends ggplot2 theme_classic().

Usage

```
pca_theme(base_size = 12, base_family = "")
```

Arguments

base_size	the font size
base_family	the font family

Examples

```
library(ggplot2)
theme_set(pca_theme(36))
```

pe_ds_preproc *Parameter estimation pre-processing. It renames the data set columns, and applies a log10 transformation if logspace is TRUE. If all.fits is true, it also computes the confidence levels.*

Description

Parameter estimation pre-processing. It renames the data set columns, and applies a log10 transformation if logspace is TRUE. If all.fits is true, it also computes the confidence levels.

Usage

```
pe_ds_preproc(filename, param.names = c(), logspace = TRUE,
  all.fits = FALSE, data_point_num = 0,
  fileout_param_estim_summary = "param_estim_summary.csv")
```

Arguments

filename	the dataset filename containing the fits sequence
param.names	the list of estimated parameter names
logspace	true if the data set shoud be log10-transformed.
all.fits	true if filename contains all fits, false otherwise
data_point_num	the number of data points used for parameterise the model. Ignored if all.fits is false
fileout_param_estim_summary	the name of the file containing the summary for the parameter estimation. Ignored if all.fits is false

Examples

```
dir.create(file.path("pe_datasets"))
data(insulin_receptor_all_fits)
write.table(insulin_receptor_all_fits,
            file=file.path("pe_datasets", "all_fits.csv"),
            row.names=FALSE)
pe_ds_preproc(filename=file.path("pe_datasets", "all_fits.csv"),
              param.names=c('k1', 'k2', 'k3'),
              logspace=TRUE,
              all.fits=TRUE,
              data_point_num=33,
            fileout_param_estim_summary=file.path("pe_datasets", "param_estim_summary.csv"))
data(insulin_receptor_best_fits)
write.table(insulin_receptor_best_fits,
            file=file.path("pe_datasets", "best_fits.csv"),
            row.names=FALSE)
pe_ds_preproc(filename=file.path("pe_datasets", "best_fits.csv"),
              param.names=c('k1', 'k2', 'k3'),
```

```
                logspace=TRUE,
                all.fits=FALSE)
```

`plot_combined_tc` *Plot repeated time courses in the same plot with mean, 1 standard deviation, and 95% confidence intervals.*

Description

Plot repeated time courses in the same plot with mean, 1 standard deviation, and 95% confidence intervals.

Usage

```
plot_combined_tc(df, g = ggplot(), title = "", xaxis_label = "",
  yaxis_label = "", bar_type = "mean", alpha = 1, yaxis.min = NULL,
  yaxis.max = NULL)
```

Arguments

`df`	a data frame
`g`	the current ggplot to overlap
`title`	the title
`xaxis_label`	the xaxis label
`yaxis_label`	the yaxis label
`bar_type`	the type of bar ("mean", "mean_sd", "mean_sd_ci95")
`alpha`	the amount of alpha transparency
`yaxis.min`	the lower limit for the y axis
`yaxis.max`	the upper limit for the y axis

Value

the plot

Examples

```
data(insulin_receptor_1)
data(insulin_receptor_2)
data(insulin_receptor_3)
df <- data.frame(Time=insulin_receptor_1[,1],
                 X1=insulin_receptor_1[,2],
                 X2=insulin_receptor_2[,2],
                 X3=insulin_receptor_3[,2])
plot_combined_tc(df=df,
                 xaxis_label="Time [m]", yaxis_label="Level [a.u.]",
                 bar_type="mean", alpha=1, yaxis.min=NULL, yaxis.max=NULL)
```

```
plot_combined_tc(df=df,
                 xaxis_label="Time [m]", yaxis_label="Level [a.u.]",
                 bar_type="mean_sd", alpha=1, yaxis.min=NULL, yaxis.max=NULL)
plot_combined_tc(df=df,
                 xaxis_label="Time [m]", yaxis_label="Level [a.u.]",
                 bar_type="mean_sd_ci95", alpha=0.3, yaxis.min=NULL, yaxis.max=NULL)
```

`plot_comb_sims` *Plot the simulation time courses using a heatmap representation.*

Description

Plot the simulation time courses using a heatmap representation.

Usage

```
plot_comb_sims(inputdir, outputdir, model, exp_dataset,
  plot_exp_dataset = FALSE, exp_dataset_alpha = 1, xaxis_label = "",
  yaxis_label = "", column_to_read = "X1", yaxis.min = NULL,
  yaxis.max = NULL)
```

Arguments

`inputdir`	the input directory containing the time course files
`outputdir`	the output directory
`model`	the model name
`exp_dataset`	a full path file containing the experimental data.
`plot_exp_dataset`	TRUE if the experimental data should also be plotted
`exp_dataset_alpha`	the alpha level for the data set
`xaxis_label`	the label for the x axis (e.g. Time (min))
`yaxis_label`	the label for the y axis (e.g. Level (a.u.))
`column_to_read`	the name of the column to process
`yaxis.min`	the lower limit for the y axis
`yaxis.max`	the upper limit for the y axis

Examples

```
data(insulin_receptor_IR_beta_pY1146)
data(insulin_receptor_exp_dataset)
dir.create(file.path("sim_datasets_sum"))
write.table(insulin_receptor_IR_beta_pY1146,
            file=file.path("sim_datasets_sum","insulin_receptor_IR_beta_pY1146.csv"),
            row.names=FALSE)
```

```
write.table(insulin_receptor_exp_dataset,
            file="insulin_receptor_exp_dataset.csv",
            row.names=FALSE)
plot_comb_sims(inputdir="sim_datasets_sum",
               outputdir="sim_plots",
               model="insulin_receptor",
               exp_dataset="insulin_receptor_exp_dataset.csv",
               plot_exp_dataset=TRUE,
               exp_dataset_alpha=1.0,
               xaxis_label="Time [m]",
               yaxis_label="Level [a.u.]",
               column_to_read='IR_beta_pY1146',
               yaxis.min=NULL,
               yaxis.max=NULL)
```

`plot_double_param_scan_data` *Plot model double parameter scan time courses.*

Description

Plot model double parameter scan time courses.

Usage

```
plot_double_param_scan_data(model, scanned_par1, scanned_par2, inputdir,
  outputdir, run)
```

Arguments

`model`	the model name without extension
`scanned_par1`	the 1st scanned parameter
`scanned_par2`	the 2nd scanned parameter
`inputdir`	the input directory
`outputdir`	the output directory
`run`	the simulation run

Examples

```
data(insulin_receptor_ps2_tp2)
dir.create(file.path("ps2_datasets"))
write.table(insulin_receptor_ps2_tp2,
            file=file.path("ps2_datasets",
                     "insulin_receptor_InsulinPercent__IRbetaPercent__rep_1__tp_2.csv"),
            row.names=FALSE)
plot_double_param_scan_data(model="insulin_receptor_InsulinPercent__IRbetaPercent",
                            scanned_par1="InsulinPercent",
                            scanned_par2="IRbetaPercent",
```

```
                    inputdir="ps2_datasets",
                    outputdir="ps2_plots",
                    run=1)
```

`plot_fits` *Plot the number of iterations vs objective values in log10 scale.*

Description

Plot the number of iterations vs objective values in log10 scale.

Usage

```
plot_fits(objval.vec, g = ggplot())
```

Arguments

`objval.vec`	the array of objective function values.
`g`	the current ggplot to overlap

Value

the plot

Examples

```
v <- 10*(rnorm(10000))^4 + 10
plot_fits(objval.vec=v) + basic_theme(36)
```

`plot_heatmap_tc` *Plot time courses organised as data frame columns with a heatmap.*

Description

Plot time courses organised as data frame columns with a heatmap.

Usage

```
plot_heatmap_tc(df, g = ggplot(), scaled = TRUE, title = "",
  xaxis_label = "", yaxis_label = "")
```

Arguments

df	a data frame, with Time as first column
g	the current ggplot to overlap
scaled	TRUE if the time course values should be scaled within 0 and 1.
title	the title of the plot
xaxis_label	the xaxis label of the plot
yaxis_label	the yaxis label of the plot

Value

the plot

Examples

```
data(insulin_receptor_1)
data(insulin_receptor_2)
data(insulin_receptor_3)
df <- data.frame(Time=insulin_receptor_1[,1],
                 X1=insulin_receptor_1[,2],
                 X2=insulin_receptor_2[,2],
                 X3=insulin_receptor_3[,2])
plot_heatmap_tc(df=df, scaled=FALSE, xaxis_label="Time [m]", yaxis_label="repeats")
plot_heatmap_tc(df=df, scaled=TRUE, xaxis_label="Time [m]", yaxis_label="repeats")
```

---

plot_objval_vs_iters *Plot the Objective values vs Iterations*

Description

Plot the Objective values vs Iterations

Usage

```
plot_objval_vs_iters(objval.vec, model, plots_dir)
```

Arguments

objval.vec	the vector containing the objective values
model	the model name
plots_dir	the directory to save the generated plots

Examples

```
dir.create(file.path("pe_plots"))
v <- 10*(rnorm(10000))^4 + 10
plot_objval_vs_iters(objval.vec=v, model="model", plots_dir="pe_plots")
```

plot_parameter_density *Plot parameter density.*

Description

Plot parameter density.

Usage

```
plot_parameter_density(df, parameter, fileout, title = "", logspace = TRUE,
  scientific_notation = TRUE)
```

Arguments

df	the data set containing the parameter estimates to plot.
parameter	the name of the parameter to plot the density
fileout	the output file
title	the plot title (default: "")
logspace	true if the parameters should be plotted in logspace (default: TRUE)
scientific_notation	true if the axis labels should be plotted in scientific notation (default: TRUE)

Examples

```
dir.create(file.path("pe_plots"))
data(insulin_receptor_all_fits)
colnames(insulin_receptor_all_fits)[1] <- "ObjVal"
insulin_receptor_all_fits[,2:4] <- log10(insulin_receptor_all_fits[,2:4])
fileout <- file.path("pe_plots", "dens_k1.pdf")
plot_parameter_density(df=insulin_receptor_all_fits,
                       parameter="k1",
                       fileout=fileout)
```

plot_raw_dataset *Add experimental data points to a plot. The length of the experimental time course to plot is limited by the length of the simulated time course (=max_sim_tp).*

Description

Add experimental data points to a plot. The length of the experimental time course to plot is limited by the length of the simulated time course (=max_sim_tp).

Usage

```
plot_raw_dataset(df_exp_dataset, g = ggplot(), readout = "time",
  max_sim_tp = 0, alpha = 1, yaxis.min = NULL, yaxis.max = NULL)
```

Arguments

df_exp_dataset	the experimental data set
g	the current ggplot to overlap
readout	the name of the readout
max_sim_tp	the maximum simulated time point
alpha	the amount of alpha transparency
yaxis.min	the lower limit for the y axis
yaxis.max	the upper limit for the y axis

Value

the plot

Examples

```
data(insulin_receptor_exp_dataset)
plot_raw_dataset(insulin_receptor_exp_dataset, readout="IR_beta_pY1146",
                 max_sim_tp=30, alpha=1, yaxis.min=NULL, yaxis.max=NULL)
```

---

plot_repeated_tc *Plot repeated time courses in the same plot separately. First column is Time.*

Description

Plot repeated time courses in the same plot separately. First column is Time.

Usage

```
plot_repeated_tc(df, g = ggplot(), title = "", xaxis_label = "",
  yaxis_label = "", alpha = 1, yaxis.min = NULL, yaxis.max = NULL)
```

Arguments

df	a data frame
g	the current ggplot to overlap
title	the title of the plot
xaxis_label	the xaxis label of the plot
yaxis_label	the yaxis label of the plot
alpha	the amount of alpha transparency
yaxis.min	the lower limit for the y axis
yaxis.max	the upper limit for the y axis

Value

the plot

Examples

```
data(insulin_receptor_1)
data(insulin_receptor_2)
data(insulin_receptor_3)
df <- data.frame(Time=insulin_receptor_1[,1],
                 X1=insulin_receptor_1[,2],
                 X2=insulin_receptor_2[,2],
                 X3=insulin_receptor_3[,2])
plot_repeated_tc(df=df,
                 xaxis_label="Time [m]", yaxis_label="Level [a.u.]",
                 alpha=1, yaxis.min=NULL, yaxis.max=NULL)
```

---

plot_sampled_2d_ple *Plot 2D profile likelihood estimations.*

Description

Plot 2D profile likelihood estimations.

Usage

```
plot_sampled_2d_ple(df, parameter1, parameter2, fileout, title = "",
  logspace = TRUE, scientific_notation = TRUE)
```

Arguments

df	the data set containing the parameter estimates to plot.
parameter1	the name of the first parameter
parameter2	the name of the second parameter
fileout	the output file
title	the plot title (default: "")
logspace	true if the parameters should be plotted in logspace (default: TRUE)
scientific_notation	true if the axis labels should be plotted in scientific notation (default: TRUE)

Examples

```
dir.create(file.path("pe_plots"))
data(insulin_receptor_all_fits)
colnames(insulin_receptor_all_fits)[1] <- "ObjVal"
insulin_receptor_all_fits[,2:4] <- log10(insulin_receptor_all_fits[,2:4])
fileout <- file.path("pe_plots", "2d_ple_k1_k2.pdf")
plot_sampled_2d_ple(df=insulin_receptor_all_fits,
```

```
                    parameter1="k1",
                    parameter2="k2",
                    fileout=fileout)
```

`plot_sampled_ple` *Plot the sampled profile likelihood estimations (PLE). The table is made of two columns: ObjVal | Parameter*

Description

Plot the sampled profile likelihood estimations (PLE). The table is made of two columns: ObjVal | Parameter

Usage

```
plot_sampled_ple(df99, cl66_objval, cl95_objval, cl99_objval, plots_dir, model,
  logspace = TRUE, scientific_notation = TRUE)
```

Arguments

`df99`	the data set including the fits within 99% confidence level
`cl66_objval`	the objective value at 66% confidence level
`cl95_objval`	the objective value at 95% confidence level
`cl99_objval`	the objective value at 99% confidence level
`plots_dir`	the directory to save the generated plots
`model`	the model name
`logspace`	true if parameters should be plotted in logspace
`scientific_notation`	true if the axis labels should be plotted in scientific notation

Examples

```
dir.create(file.path("pe_datasets"))
dir.create(file.path("pe_plots"))
data(insulin_receptor_all_fits)
write.table(insulin_receptor_all_fits,
            file=file.path("pe_datasets", "all_fits.csv"),
            row.names=FALSE)
# generate the global statistics for the parameter estimation
pe_ds_preproc(filename=file.path("pe_datasets", "all_fits.csv"),
              param.names=c('k1', 'k2', 'k3'),
              logspace=TRUE,
              all.fits=TRUE,
              data_point_num=33,
            fileout_param_estim_summary=file.path("pe_datasets", "param_estim_summary.csv"))
# load the fits for this parameter
df <- as.data.frame(data.table::fread(file.path("pe_datasets", "all_fits_log10.csv"),
```

```
                                        select=c("ObjVal", "k2")))
# load the global statistics for the parameter estimation
dt.stats <- data.table::fread(file.path("pe_datasets", "param_estim_summary.csv"),
                          select=c("MinObjVal", "CL66ObjVal", "CL95ObjVal", "CL99ObjVal"))
df99 <- df[df[ ,"ObjVal"] <= dt.stats$CL99ObjVal, ]
# compute the stats for parameter k2.
plot_sampled_ple(df99=df99,
                 cl66_objval=dt.stats$CL66ObjVal,
                 cl95_objval=dt.stats$CL95ObjVal,
                 cl99_objval=dt.stats$CL99ObjVal,
                 plots_dir="pe_plots",
                 model="ir_beta",
                 logspace=TRUE)
```

`plot_sep_sims` *Plot the simulations time course separately.*

Description

Plot the simulations time course separately.

Usage

```
plot_sep_sims(inputdir, outputdir, model, exp_dataset,
  plot_exp_dataset = FALSE, exp_dataset_alpha = 1, xaxis_label = "",
  yaxis_label = "", column_to_read = "X1", yaxis.min = NULL,
  yaxis.max = NULL)
```

Arguments

`inputdir`	the input directory containing the time course files
`outputdir`	the output directory
`model`	the model name
`exp_dataset`	a full path file containing the experimental data.
`plot_exp_dataset`	TRUE if the experimental data should also be plotted
`exp_dataset_alpha`	the alpha level for the data set
`xaxis_label`	the label for the x axis (e.g. Time (min))
`yaxis_label`	the label for the y axis (e.g. Level (a.u.))
`column_to_read`	the name of the column to process
`yaxis.min`	the lower limit for the y axis
`yaxis.max`	the upper limit for the y axis

Examples

```
data(insulin_receptor_IR_beta_pY1146)
data(insulin_receptor_exp_dataset)
dir.create(file.path("sim_datasets_sum"))
write.table(insulin_receptor_IR_beta_pY1146,
            file=file.path("sim_datasets_sum","insulin_receptor_IR_beta_pY1146.csv"),
            row.names=FALSE)
write.table(insulin_receptor_exp_dataset,
            file="insulin_receptor_exp_dataset.csv",
            row.names=FALSE)
plot_sep_sims(inputdir="sim_datasets_sum",
              outputdir="sim_plots",
              model="insulin_receptor",
              exp_dataset="insulin_receptor_exp_dataset.csv",
              plot_exp_dataset=TRUE,
              exp_dataset_alpha=1.0,
              xaxis_label="Time [m]",
              yaxis_label="Level [a.u.]",
              column_to_read='IR_beta_pY1146',
              yaxis.min=NULL,
              yaxis.max=NULL)
```

---

plot_single_param_scan_data

Plot model single parameter scan time courses

Description

Plot model single parameter scan time courses

Usage

```
plot_single_param_scan_data(model, inhibition_only, inputdir, outputdir, run,
  percent_levels = TRUE, min_level = 0, max_level = 100,
  levels_number = 10, xaxis_label = "", yaxis_label = "")
```

Arguments

model	The model name
inhibition_only	true if the scanning only decreases the variable amount (inhibition only)
inputdir	the input directory containing the simulated data
outputdir	the output directory that will contain the simulated plots
run	the simulation number
percent_levels	true if scanning levels are in percent (default TRUE)
min_level	the minimum level (default: 0)

max_level	the maximum level (default: 100)
levels_number	the number of levels (default: 10)
xaxis_label	the label for the x axis (e.g. Time (min))
yaxis_label	the label for the y axis (e.g. Level (a.u.))

Examples

```
data(insulin_receptor_ps1_l0)
data(insulin_receptor_ps1_l1)
data(insulin_receptor_ps1_l3)
data(insulin_receptor_ps1_l4)
data(insulin_receptor_ps1_l6)
data(insulin_receptor_ps1_l8)
data(insulin_receptor_ps1_l9)
data(insulin_receptor_ps1_l11)
data(insulin_receptor_ps1_l13)
data(insulin_receptor_ps1_l14)
data(insulin_receptor_ps1_l16)
dir.create(file.path("ps1_datasets"))
write.table(insulin_receptor_ps1_l0,
            file=file.path("ps1_datasets",
                           "insulin_receptor_scan_IR_beta__rep_1__level_0.csv"),
            row.names=FALSE)
write.table(insulin_receptor_ps1_l1,
            file=file.path("ps1_datasets",
                           "insulin_receptor_scan_IR_beta__rep_1__level_1.csv"),
            row.names=FALSE)
write.table(insulin_receptor_ps1_l3,
            file=file.path("ps1_datasets",
                           "insulin_receptor_scan_IR_beta__rep_1__level_3.csv"),
            row.names=FALSE)
write.table(insulin_receptor_ps1_l4,
            file=file.path("ps1_datasets",
                           "insulin_receptor_scan_IR_beta__rep_1__level_4.csv"),
            row.names=FALSE)
write.table(insulin_receptor_ps1_l6,
            file=file.path("ps1_datasets",
                           "insulin_receptor_scan_IR_beta__rep_1__level_6.csv"),
            row.names=FALSE)
write.table(insulin_receptor_ps1_l8,
            file=file.path("ps1_datasets",
                           "insulin_receptor_scan_IR_beta__rep_1__level_8.csv"),
            row.names=FALSE)
write.table(insulin_receptor_ps1_l9,
            file=file.path("ps1_datasets",
                           "insulin_receptor_scan_IR_beta__rep_1__level_9.csv"),
            row.names=FALSE)
write.table(insulin_receptor_ps1_l11,
            file=file.path("ps1_datasets",
                           "insulin_receptor_scan_IR_beta__rep_1__level_11.csv"),
            row.names=FALSE)
write.table(insulin_receptor_ps1_l13,
```

```
            file=file.path("ps1_datasets",
                           "insulin_receptor_scan_IR_beta__rep_1__level_13.csv"),
            row.names=FALSE)
write.table(insulin_receptor_ps1_l14,
            file=file.path("ps1_datasets",
                           "insulin_receptor_scan_IR_beta__rep_1__level_14.csv"),
            row.names=FALSE)
write.table(insulin_receptor_ps1_l16,
            file=file.path("ps1_datasets",
                           "insulin_receptor_scan_IR_beta__rep_1__level_16.csv"),
            row.names=FALSE)
plot_single_param_scan_data(
       model="insulin_receptor_scan_IR_beta",
       inhibition_only=FALSE,
       inputdir="ps1_datasets",
       outputdir="ps1_plots",
       run=1,
       percent_levels=TRUE,
       min_level=0,
       max_level=250,
       levels_number=10,
       xaxis_label="Time [m]",
       yaxis_label="Level [a.u.]")
```

`plot_single_param_scan_data_homogen` *Plot model single parameter scan time courses using homogeneous lines.*

Description

Plot model single parameter scan time courses using homogeneous lines.

Usage

```
plot_single_param_scan_data_homogen(model, inputdir, outputdir, run,
  xaxis_label = "", yaxis_label = "")
```

Arguments

`model`	The model name
`inputdir`	the input directory containing the simulated data
`outputdir`	the output directory that will contain the simulated plots
`run`	the simulation number
`xaxis_label`	the label for the x axis (e.g. Time (min))
`yaxis_label`	the label for the y axis (e.g. Level (a.u.))

`replace_colnames` *Rename data frame columns. 'ObjectiveValue' is renamed as 'ObjVal'. Substrings 'Values.' and '..InitialValue' are removed.*

Description

Rename data frame columns. 'ObjectiveValue' is renamed as 'ObjVal'. Substrings 'Values.' and '..InitialValue' are removed.

Usage

```
replace_colnames(df.cols)
```

Arguments

`df.cols`	The columns of a data frame.

Value

the renamed columns

`rightCI` *Return the right value of the parameter confidence interval. The provided dataset has two columns: ObjVal | ParamValue*

Description

Return the right value of the parameter confidence interval. The provided dataset has two columns: ObjVal | ParamValue

Usage

```
rightCI(largest.param.value, full_dataset, cl_objval)
```

Arguments

`largest.param.value`	the largest parameter value within the specified confidence level
`full_dataset`	the full dataset
`cl_objval`	the objective value at the desired confidence level

Value

the right confidence interval

Examples

```
data(insulin_receptor_all_fits)
colnames(insulin_receptor_all_fits)[1] <- "ObjVal"
min_objval <- min(insulin_receptor_all_fits[,1])
# compute the stats for parameter k2.
insulin_receptor_all_fits <- subset(insulin_receptor_all_fits, select=c(1,3))
rightCI(largest.param.value=0.477115,
        full_dataset=insulin_receptor_all_fits,
        cl_objval=min_objval+0.01)
rightCI(largest.param.value=0.467000,
        full_dataset=insulin_receptor_all_fits,
        cl_objval=min_objval+0.01)
```

---

sampled_2d_ple_analysis

2D profile likelihood estimation analysis.

Description

2D profile likelihood estimation analysis.

Usage

```
sampled_2d_ple_analysis(model, filename, parameter1, parameter2, plots_dir,
  thres = "BestFits", best_fits_percent = 50,
  fileout_param_estim_summary = "", logspace = TRUE,
  scientific_notation = TRUE)
```

Arguments

model	the model name
filename	the filename containing the fits sequence
parameter1	the name of the first parameter
parameter2	the name of the second parameter
plots_dir	the directory for storing the plots
thres	the threshold used to filter the dataset. Values: "BestFits", "CL66", "CL95", "CL99", "All".
best_fits_percent	the percent of best fits to analyse. Only used if thres="BestFits".
fileout_param_estim_summary	the name of the file containing the summary for the parameter estimation. Only used if thres!="BestFits".
logspace	true if the parameters should be plotted in logspace
scientific_notation	true if the axis labels should be plotted in scientific notation

Examples

```
dir.create(file.path("pe_datasets"))
dir.create(file.path("pe_plots"))
data(insulin_receptor_all_fits)
write.table(insulin_receptor_all_fits,
            file=file.path("pe_datasets", "all_fits.csv"),
            row.names=FALSE)
# generate the global statistics for the parameter estimation
pe_ds_preproc(filename=file.path("pe_datasets", "all_fits.csv"),
              param.names=c('k1', 'k2', 'k3'),
              logspace=TRUE,
              all.fits=TRUE,
              data_point_num=33,
            fileout_param_estim_summary=file.path("pe_datasets", "param_estim_summary.csv"))
sampled_2d_ple_analysis(model="ir_beta",
                        filename=file.path("pe_datasets", "all_fits_log10.csv"),
                        parameter1="k1",
                        parameter2="k2",
                        plots_dir="pe_plots",
                        thres="CL95",
                        fileout_param_estim_summary=file.path("pe_datasets",
                                                              "param_estim_summary.csv"),
                        logspace=TRUE)

data(insulin_receptor_best_fits)
write.table(insulin_receptor_best_fits,
            file=file.path("pe_datasets", "best_fits.csv"),
            row.names=FALSE)
# generate the global statistics for the parameter estimation
pe_ds_preproc(filename=file.path("pe_datasets", "best_fits.csv"),
              param.names=c('k1', 'k2', 'k3'),
              logspace=TRUE,
              all.fits=FALSE)
sampled_2d_ple_analysis(model="ir_beta",
                        filename=file.path("pe_datasets", "best_fits_log10.csv"),
                        parameter1="k1",
                        parameter2="k2",
                        plots_dir="pe_plots",
                        thres="BestFits",
                        best_fits_percent=50,
                        logspace=TRUE)
```

`sampled_ple_analysis` *Run the profile likelihood estimation analysis.*

Description

Run the profile likelihood estimation analysis.

Usage

```
sampled_ple_analysis(model, filename, parameter, plots_dir,
  fileout_param_estim_summary, logspace = TRUE, scientific_notation = TRUE)
```

Arguments

model	the model name
filename	the filename containing the fits sequence
parameter	the parameter to compute the PLE analysis
plots_dir	the directory to save the generated plots
fileout_param_estim_summary	the name of the file containing the summary for the parameter estimation
logspace	true if parameters should be plotted in logspace. (default: TRUE)
scientific_notation	true if the axis labels should be plotted in scientific notation (default: TRUE)

Examples

```
dir.create(file.path("pe_datasets"))
dir.create(file.path("pe_plots"))
data(insulin_receptor_all_fits)
write.table(insulin_receptor_all_fits,
            file=file.path("pe_datasets", "all_fits.csv"),
            row.names=FALSE)
# generate the global statistics for the parameter estimation
pe_ds_preproc(filename=file.path("pe_datasets", "all_fits.csv"),
              param.names=c('k1', 'k2', 'k3'),
              logspace=TRUE,
              all.fits=TRUE,
              data_point_num=33,
           fileout_param_estim_summary=file.path("pe_datasets", "param_estim_summary.csv"))
sampled_ple_analysis(model="ir_beta",
                     filename=file.path("pe_datasets", "all_fits_log10.csv"),
                     parameter="k1",
                     plots_dir="pe_plots",
                     fileout_param_estim_summary=file.path("pe_datasets",
                                                           "param_estim_summary.csv"),
                     logspace=TRUE)
```

---

sbpiper_pe *Main R function for SBpipe pipeline: parameter_estimation().*

Description

Main R function for SBpipe pipeline: parameter_estimation().

Usage

```
sbpiper_pe(model, finalfits_filenamein, allfits_filenamein, plots_dir,
  data_point_num,
  fileout_param_estim_best_fits_details = "param_estim_best_fits_details.csv",
  fileout_param_estim_details = "param_estim_details.csv",
  fileout_param_estim_summary = "param_estim_summary.csv",
  best_fits_percent = 50, plot_2d_66cl_corr = TRUE,
  plot_2d_95cl_corr = TRUE, plot_2d_99cl_corr = TRUE, logspace = TRUE,
  scientific_notation = TRUE)
```

Arguments

model	the name of the model
finalfits_filenamein	the dataset containing the best parameter fits
allfits_filenamein	the dataset containing all the parameter fits
plots_dir	the directory to save the generated plots.
data_point_num	the number of data points used for parameterise the model.
fileout_param_estim_best_fits_details	the name of the file for the statistics of the parameters best fits.
fileout_param_estim_details	the name of the file containing the detailed statistics for the estimated parameters.
fileout_param_estim_summary	the name of the file containing the summary for the parameter estimation.
best_fits_percent	the percent of best fits to analyse.
plot_2d_66cl_corr	true if the 2D parameter correlation plots for 66% confidence intervals should be plotted.
plot_2d_95cl_corr	true if the 2D parameter correlation plots for 95% confidence intervals should be plotted.
plot_2d_99cl_corr	true if the 2D parameter correlation plots for 99% confidence intervals should be plotted.
logspace	true if parameters should be plotted in logspace.
scientific_notation	true if axis labels should be plotted in scientific notation.

Examples

```
dir.create(file.path("pe_datasets"))
dir.create(file.path("pe_plots"))
```

```
data(insulin_receptor_best_fits)
write.table(insulin_receptor_best_fits,
            file=file.path("pe_datasets", "best_fits.csv"),
            row.names=FALSE)
data(insulin_receptor_all_fits)
write.table(insulin_receptor_all_fits,
            file=file.path("pe_datasets", "all_fits.csv"),
            row.names=FALSE)
sbpiper_pe(model="ir_beta",
           finalfits_filenamein=file.path("pe_datasets", "best_fits.csv"),
           allfits_filenamein=file.path("pe_datasets", "all_fits.csv"),
           plots_dir="pe_plots",
           data_point_num=33,
           fileout_param_estim_best_fits_details=file.path("pe_datasets",
                                                 "param_estim_best_fits_details.csv"),
           fileout_param_estim_details=file.path("pe_datasets",
                                                 "param_estim_details.csv"),
           fileout_param_estim_summary=file.path("pe_datasets",
                                                 "param_estim_summary.csv"),
           best_fits_percent=50,
           plot_2d_66cl_corr=TRUE,
           plot_2d_95cl_corr=TRUE,
           plot_2d_99cl_corr=TRUE,
           logspace=TRUE,
           scientific_notation=TRUE)
```

`sbpiper_ps1` *Main R function for SBpipe pipeline: parameter_scan1().*

Description

Main R function for SBpipe pipeline: parameter_scan1().

Usage

```
sbpiper_ps1(model, inhibition_only, inputdir, outputdir, run, percent_levels,
  min_level, max_level, levels_number, homogeneous_lines, xaxis_label,
  yaxis_label)
```

Arguments

`model`	the model name
`inhibition_only`	true if the the variable amount can only decrease
`inputdir`	the input directory containing the simulated data
`outputdir`	the output directory that will contain the simulated plots
`run`	the simulation number

percent_levels	true if scanning levels are in percent
min_level	the minimum level
max_level	the maximum level
levels_number	the number of levels
homogeneous_lines	true if lines should be plotted homogeneously
xaxis_label	the label for the x axis (e.g. Time (min))
yaxis_label	the label for the y axis (e.g. Level (a.u.))

Examples

```
data(insulin_receptor_ps1_l0)
data(insulin_receptor_ps1_l1)
data(insulin_receptor_ps1_l3)
data(insulin_receptor_ps1_l4)
data(insulin_receptor_ps1_l6)
data(insulin_receptor_ps1_l8)
data(insulin_receptor_ps1_l9)
data(insulin_receptor_ps1_l11)
data(insulin_receptor_ps1_l13)
data(insulin_receptor_ps1_l14)
data(insulin_receptor_ps1_l16)
dir.create(file.path("ps1_datasets"))
write.table(insulin_receptor_ps1_l0,
            file=file.path("ps1_datasets",
                           "insulin_receptor_scan_IR_beta__rep_1__level_0.csv"),
            row.names=FALSE)
write.table(insulin_receptor_ps1_l1,
            file=file.path("ps1_datasets",
                           "insulin_receptor_scan_IR_beta__rep_1__level_1.csv"),
            row.names=FALSE)
write.table(insulin_receptor_ps1_l3,
            file=file.path("ps1_datasets",
                           "insulin_receptor_scan_IR_beta__rep_1__level_3.csv"),
            row.names=FALSE)
write.table(insulin_receptor_ps1_l4,
            file=file.path("ps1_datasets",
                           "insulin_receptor_scan_IR_beta__rep_1__level_4.csv"),
            row.names=FALSE)
write.table(insulin_receptor_ps1_l6,
            file=file.path("ps1_datasets",
                           "insulin_receptor_scan_IR_beta__rep_1__level_6.csv"),
            row.names=FALSE)
write.table(insulin_receptor_ps1_l8,
            file=file.path("ps1_datasets",
                           "insulin_receptor_scan_IR_beta__rep_1__level_8.csv"),
            row.names=FALSE)
write.table(insulin_receptor_ps1_l9,
            file=file.path("ps1_datasets",
```

```
                            "insulin_receptor_scan_IR_beta__rep_1__level_9.csv"),
            row.names=FALSE)
write.table(insulin_receptor_ps1_l11,
            file=file.path("ps1_datasets",
                            "insulin_receptor_scan_IR_beta__rep_1__level_11.csv"),
            row.names=FALSE)
write.table(insulin_receptor_ps1_l13,
            file=file.path("ps1_datasets",
                            "insulin_receptor_scan_IR_beta__rep_1__level_13.csv"),
            row.names=FALSE)
write.table(insulin_receptor_ps1_l14,
            file=file.path("ps1_datasets",
                            "insulin_receptor_scan_IR_beta__rep_1__level_14.csv"),
            row.names=FALSE)
write.table(insulin_receptor_ps1_l16,
            file=file.path("ps1_datasets",
                            "insulin_receptor_scan_IR_beta__rep_1__level_16.csv"),
            row.names=FALSE)
sbpiper_ps1(
       model="insulin_receptor_scan_IR_beta",
       inhibition_only=FALSE,
       inputdir="ps1_datasets",
       outputdir="ps1_plots",
       run=1,
       percent_levels=TRUE,
       min_level=0,
       max_level=250,
       levels_number=10,
       homogeneous_lines=FALSE,
       xaxis_label="Time [m]",
       yaxis_label="Level [a.u.]")
```

`sbpiper_ps2` *Main R function for SBpipe pipeline: parameter_scan2().*

Description

Main R function for SBpipe pipeline: parameter_scan2().

Usage

```
sbpiper_ps2(model, scanned_par1, scanned_par2, inputdir, outputdir, run)
```

Arguments

`model`	the model name
`scanned_par1`	the 1st scanned parameter
`scanned_par2`	the 2nd scanned parameter

inputdir	the input directory
outputdir	the output directory
run	the simulation run

Examples

```
data(insulin_receptor_ps2_tp2)
dir.create(file.path("ps2_datasets"))
write.table(insulin_receptor_ps2_tp2,
            file=file.path("ps2_datasets",
                     "insulin_receptor_InsulinPercent__IRbetaPercent__rep_1__tp_2.csv"),
            row.names=FALSE)
sbpiper_ps2(model="insulin_receptor_InsulinPercent__IRbetaPercent",
           scanned_par1="InsulinPercent",
           scanned_par2="IRbetaPercent",
           inputdir="ps2_datasets",
           outputdir="ps2_plots",
           run=1)
```

---

sbpiper_sim *Main R function for SBpipe pipeline: simulate().*

Description

Main R function for SBpipe pipeline: simulate().

Usage

```
sbpiper_sim(model, inputdir, outputdir, outputfile_stats, outputfile_repeats,
  exp_dataset, plot_exp_dataset, exp_dataset_alpha, xaxis_label, yaxis_label,
  column_to_read)
```

Arguments

model	the model name
inputdir	the input directory
outputdir	the output directory
outputfile_stats	the output file containing the statistics
outputfile_repeats	the output file storing the model simulation repeats
exp_dataset	the file containing the experimental data.
plot_exp_dataset	TRUE if the experimental data should also be plotted

exp_dataset_alpha
: the alpha level for the data set

xaxis_label
: the label for the x axis (e.g. Time (min))

yaxis_label
: the label for the y axis (e.g. Level (a.u.))

column_to_read
: the name of the column to process

Examples

```
data(insulin_receptor_1)
data(insulin_receptor_2)
data(insulin_receptor_3)
data(insulin_receptor_exp_dataset)
dir.create(file.path("sim_datasets"))
dir.create(file.path("sim_datasets_sum"))
write.table(insulin_receptor_1,
            file=file.path("sim_datasets", "insulin_receptor_1.csv"),
            row.names=FALSE)
write.table(insulin_receptor_2,
            file=file.path("sim_datasets", "insulin_receptor_2.csv"),
            row.names=FALSE)
write.table(insulin_receptor_3,
            file=file.path("sim_datasets", "insulin_receptor_3.csv"),
            row.names=FALSE)
write.table(insulin_receptor_exp_dataset,
            file="insulin_receptor_exp_dataset.csv",
            row.names=FALSE)
sbpiper_sim(model="insulin_receptor",
           inputdir="sim_datasets",
           outputdir="sim_plots",
           outputfile_stats="insulin_receptor_IR_beta_pY1146_stats.csv",
           outputfile_repeats=file.path("sim_datasets_sum",
                                        "insulin_receptor_IR_beta_pY1146.csv"),
           exp_dataset="insulin_receptor_exp_dataset.csv",
           plot_exp_dataset=TRUE,
           exp_dataset_alpha=1.0,
           xaxis_label=NULL,
           yaxis_label=NULL,
           column_to_read="IR_beta_pY1146")
```

---

scatterplot *Plot a generic scatter plot*

Description

Plot a generic scatter plot

Usage

```
scatterplot(df, g = ggplot(), colNameX = "x", colNameY = "y",
  dot_size = 0.5)
```

Arguments

df	a data frame
g	the current ggplot to overlap
colNameX	the name of the column for the X axis
colNameY	the name of the column for the Y axis
dot_size	the size of the dots in the scatterplot

Value

the plot

Examples

```
df <- data.frame(a=rnorm(10000), b=rnorm(10000))
scatterplot(df, colNameX="a", colNameY="b")
```

---

scatterplot_log10 *Plot a generic scatter plot in log10 scale*

Description

Plot a generic scatter plot in log10 scale

Usage

```
scatterplot_log10(df, g = ggplot(), colNameX = "x", colNameY = "y",
  dot_size = 0.5)
```

Arguments

df	a data frame to trasform to log10 scale
g	the current ggplot to overlap
colNameX	the name of the column for the X axis
colNameY	the name of the column for the Y axis
dot_size	the size of the dots in the scatterplot

Value

the plot

Examples

```
df <- data.frame(a=exp(rnorm(10000)), b=exp(rnorm(10000)))
scatterplot_log10(df, colNameX="a", colNameY="b")
```

`scatterplot_ple` *Plot a profile likelihood estimation (PLE) scatter plot*

Description

Plot a profile likelihood estimation (PLE) scatter plot

Usage

```
scatterplot_ple(df, g = ggplot(), colNameX = "x", colNameY = "y",
  conf_level_66 = 0, conf_level_95 = 0, conf_level_99 = 0,
  dot_size = 0.1)
```

Arguments

`df`	a data frame
`g`	the current ggplot to overlap
`colNameX`	the name of the column for the X axis
`colNameY`	the name of the column for the Y axis
`conf_level_66`	the 66% confidence level to plot
`conf_level_95`	the 95% confidence level to plot
`conf_level_99`	the 99% confidence level to plot
`dot_size`	the size of the dots in the scatterplot

Value

the plot

Examples

```
a <- rnorm(10000)
b <- a^2+10
df<-data.frame(a, b)
scatterplot_ple(df, colNameX="a", colNameY="b", conf_level_66=0)
scatterplot_ple(df, colNameX="a", colNameY="b",
                conf_level_66=13, conf_level_95=16.5, conf_level_99=20)
```

scatterplot_w_colour *Plot a scatter plot using a coloured palette*

Description

Plot a scatter plot using a coloured palette

Usage

```
scatterplot_w_colour(df, g = ggplot(), colNameX = "x", colNameY = "y",
  colNameColor = "colour", dot_size = 1,
  colours = colorRamps::matlab.like(256), limits = NULL)
```

Arguments

df	a data frame
g	the current ggplot to overlap
colNameX	the name of the column for the X axis
colNameY	the name of the column for the Y axis
colNameColor	the name of the column whose values are used as 3rd dimension
dot_size	the size of the dots in the scatterplot
colours	the palette to use
limits	the limits for the palette (NULL if no limit is used)

Value

the plot

Examples

```
df <- data.frame(a=rnorm(10000), b=rnorm(10000), c=rev(seq(10000)))
scatterplot_w_colour(df, colNameX="a", colNameY="b", colNameColor="c")
```

skewness *Calculate the skewness of a numeric vector*

Description

Calculate the skewness of a numeric vector

Usage

```
skewness(x, na.rm = FALSE)
```

Arguments

x	the numeric vector
na.rm	TRUE if NA values should be discarded

Value

the skewness

Examples

```
skewness(x=c(1,2.4,5,NA), na.rm=TRUE)
```

---

summarise_data *Summarise the model simulation repeats in a single file.*

Description

Summarise the model simulation repeats in a single file.

Usage

```
summarise_data(inputdir, model, outputfile, column_to_read = "X1")
```

Arguments

inputdir	the input directory containing the time course files
model	the model name
outputfile	the file to store the simulated repeats
column_to_read	the name of the column to process

Examples

```
data(insulin_receptor_1)
data(insulin_receptor_2)
data(insulin_receptor_3)
dir.create(file.path("sim_datasets"))
dir.create(file.path("sim_datasets_sum"))
write.table(insulin_receptor_1,
            file=file.path("sim_datasets","insulin_receptor_1.csv"),
            row.names=FALSE)
write.table(insulin_receptor_2,
            file=file.path("sim_datasets","insulin_receptor_2.csv"),
            row.names=FALSE)
write.table(insulin_receptor_3,
            file=file.path("sim_datasets","insulin_receptor_3.csv"),
            row.names=FALSE)
summarise_data(inputdir="sim_datasets",
```

```
                model="insulin_receptor",
                outputfile=file.path("sim_datasets_sum",
                                     "insulin_receptor_IR_beta_pY1146.csv"),
                column_to_read="IR_beta_pY1146")
```

---

tc_theme *A theme for time courses. It extends ggplot2 theme_classic().*

Description

A theme for time courses. It extends ggplot2 theme_classic().

Usage

```
tc_theme(base_size = 12, base_family = "")
```

Arguments

base_size	the font size
base_family	the font family

Examples

```
library(ggplot2)
theme_set(tc_theme(36))
```

Index