Package: TPP2D (via r-universe)

July 25, 2024

Title Detection of ligand-protein interactions from 2D thermal profiles (DLPTP)

Version 1.21.0

Description Detection of ligand-protein interactions from 2D thermal profiles (DLPTP), Performs an FDR-controlled analysis of 2D-TPP experiments by functional analysis of dose-response curves across temperatures.

License GPL-3

Encoding UTF-8

VignetteBuilder knitr

LazyData false

biocViews Software, Proteomics, DataImport

BugReports https://support.bioconductor.org/

URL http://bioconductor.org/packages/TPP2D

RoxygenNote 7.1.2

Depends R (>= 3.6.0), stats, utils, dplyr, methods

Imports ggplot2, tidyr, foreach, doParallel, openxlsx, stringr, RCurl, parallel, MASS, BiocParallel, limma

Suggests knitr, testthat, rmarkdown, BiocStyle

Repository https://bioc.r-universe.dev

RemoteUrl https://github.com/bioc/TPP2D

RemoteRef HEAD

RemoteSha 7a3f22514d09acaed472480036a6c96c25e7414c

Contents

`annotateDataList` *Annotate imported data list using a config table*

Description

Annotate imported data list using a config table

Usage

```
annotateDataList(dataList, geneNameVar, configLong, intensityStr, fcStr)
```

Arguments

`dataList`	list of datasets from different MS runs corresponding to a 2D-TPP dataset
`geneNameVar`	character string of the column name that describes the gene name of a given protein in the raw data files
`configLong`	long formatted data frame of a corresponding config table
`intensityStr`	character string indicating which columns contain raw intensities measurements
`fcStr`	character string indicating which columns contain the actual fold change values. Those column names containing the suffix `fcStr` will be regarded as containing fold change values.

Value

data frame containing all data annotated by information supplied in the config table

Examples

```
data("config_tab")
data("raw_dat_list")
dataList <- import2dMain(configTable = config_tab,
                         data = raw_dat_list,
                         idVar = "protein_id",
                         fcStr = "rel_fc_",
                         addCol = "gene_name",
                         naStrs = NA,
                         intensityStr = "signal_sum_",
                         nonZeroCols = "qusm",
                         qualColName = "qupm")
configLong <- configWide2Long(configWide = config_tab)
annotateDataList(dataList = dataList,
                 geneNameVar = "gene_name",
                 configLong = configLong,
                 intensityStr = "signal_sum_",
                 fcStr = "rel_fc_")
```

`bootstrapNull` *Bootstrap null distribution of F statistics for FDR estimation*

Description

Bootstrap null distribution of F statistics for FDR estimation

Usage

```
bootstrapNull(
  df,
  maxit = 500,
  independentFiltering = FALSE,
```

```
  fcThres = 1.5,
  minObs = 20,
  optim_fun_h0 = .min_RSS_h0,
  optim_fun_h1 = .min_RSS_h1_slope_pEC50,
  optim_fun_h1_2 = NULL,
  gr_fun_h0 = NULL,
  gr_fun_h1 = NULL,
  gr_fun_h1_2 = NULL,
  ncores = 1,
  B = 20,
  byMsExp = TRUE
)
```

Arguments

df	tidy data_frame retrieved after import of a 2D-TPP dataset, potential filtering and addition of a column "nObs" containing the number of observations per protein
maxit	maximal number of iterations the optimization should be given, default is set to 500
independentFiltering	boolean flag indicating whether independent filtering should be performed based on minimal fold changes per protein profile
fcThres	numeric value of minimal fold change (or inverse fold change) a protein has to show to be kept upon independent filtering
minObs	numeric value of minimal number of observations that should be required per protein
optim_fun_h0	optimization function that should be used for fitting the H0 model
optim_fun_h1	optimization function that should be used for fitting the H1 model
optim_fun_h1_2	optional additional optimization function that will be run with paramters retrieved from optim_fun_h1 and should be used for fitting the H1 model with the trimmed sum model, default is NULL
gr_fun_h0	optional gradient function for optim_fun_h0, default is NULL
gr_fun_h1	optional gradient function for optim_fun_h1, default is NULL
gr_fun_h1_2	optional gradient function for optim_fun_h1_2, default is NULL
ncores	numeric value of numbers of cores that the function should use to parallelize
B	numeric value of rounds of bootstrap, default: 20
byMsExp	boolean flag indicating whether resampling of residuals should be performed separately for data generated by different MS experiments, default TRUE, recommended

Value

data frame containing F statistics of proteins with permuted 2D thermal profiles that are informative on the Null distribution of F statistics

Examples

```
data("simulated_cell_extract_df")
temp_df <- simulated_cell_extract_df %>%
  filter(clustername %in% paste0("protein", 1:3)) %>%
  group_by(representative) %>%
  mutate(nObs = n()) %>%
  ungroup
boot_df <- bootstrapNull(temp_df, B = 2/10)
```

---

bootstrapNullAlternativeModel

Bootstrap null distribution of F statistics for FDR estimation based on resampling alternative model residuals

---

Description

Bootstrap null distribution of F statistics for FDR estimation based on resampling alternative model residuals

Usage

```
bootstrapNullAlternativeModel(
  df,
  params_df,
  maxit = 500,
  independentFiltering = FALSE,
  fcThres = 1.5,
  minObs = 20,
  optim_fun_h0 = TPP2D:::.min_RSS_h0,
  optim_fun_h1 = TPP2D:::.min_RSS_h1_slope_pEC50,
  optim_fun_h1_2 = NULL,
  gr_fun_h0 = NULL,
  gr_fun_h1 = NULL,
  gr_fun_h1_2 = NULL,
  BPPARAM = BiocParallel::SerialParam(progressbar = TRUE),
  B = 20,
  byMsExp = TRUE,
  verbose = FALSE
)
```

Arguments

df	tidy data frame retrieved after import of a 2D-TPP dataset, potential filtering and addition of a column "nObs" containing the number of observations per protein
params_df	data frame listing all null and alternative model parameters as obtained by 'getModelParamsDf'

maxit	maximal number of iterations the optimization should be given, default is set to 500
independentFiltering	boolean flag indicating whether independent filtering should be performed based on minimal fold changes per protein profile
fcThres	numeric value of minimal fold change (or inverse fold change) a protein has to show to be kept upon independent filtering
minObs	numeric value of minimal number of observations that should be required per protein
optim_fun_h0	optimization function that should be used for fitting the H0 model
optim_fun_h1	optimization function that should be used for fitting the H1 model
optim_fun_h1_2	optional additional optimization function that will be run with paramters retrieved from optim_fun_h1 and should be used for fitting the H1 model with the trimmed sum model, default is NULL
gr_fun_h0	optional gradient function for optim_fun_h0, default is NULL
gr_fun_h1	optional gradient function for optim_fun_h1, default is NULL
gr_fun_h1_2	optional gradient function for optim_fun_h1_2, default is NULL
BPPARAM	BiocParallel parameter for optional parallelization of null distribution generation through bootstrapping, default: BiocParallel::SerialParam()
B	numeric value of rounds of bootstrap, default: 20
byMsExp	boolean flag indicating whether resampling of residuals should be performed separately for data generated by different MS experiments, default TRUE, recommended
verbose	logical indicating whether to print each protein while its profile is boostrapped

Value

data frame containing F statistics of proteins with permuted 2D thermal profiles that are informative on the Null distribution of F statistics

Examples

```
data("simulated_cell_extract_df")
temp_df <- simulated_cell_extract_df %>%
  filter(clustername %in% paste0("protein", 1:3)) %>%
  group_by(representative) %>%
  mutate(nObs = n()) %>%
  ungroup
temp_params_df <- getModelParamsDf(temp_df)
boot_df <- bootstrapNullAlternativeModel(
  temp_df, params_df = temp_params_df, B = 2)
```

`bootstrapNullAlternativeModelFast`

Bootstrap null distribution of F statistics for FDR estimation based on resampling alternative model residuals with only one round of model fitting on resampled data and subsequent resampling of thereby obtained residuals

Description

Bootstrap null distribution of F statistics for FDR estimation based on resampling alternative model residuals with only one round of model fitting on resampled data and subsequent resampling of thereby obtained residuals

Usage

```
bootstrapNullAlternativeModelFast(
  df,
  params_df,
  maxit = 500,
  independentFiltering = FALSE,
  fcThres = 1.5,
  minObs = 20,
  optim_fun_h0 = TPP2D:::.min_RSS_h0,
  optim_fun_h1 = TPP2D:::.min_RSS_h1_slope_pEC50,
  optim_fun_h1_2 = NULL,
  gr_fun_h0 = NULL,
  gr_fun_h1 = NULL,
  gr_fun_h1_2 = NULL,
  BPPARAM = BiocParallel::SerialParam(progressbar = TRUE),
  B = 20,
  byMsExp = TRUE,
  verbose = FALSE
)
```

Arguments

`df`	tidy data frame retrieved after import of a 2D-TPP dataset, potential filtering and addition of a column "nObs" containing the number of observations per protein
`params_df`	data frame listing all null and alternative model parameters as obtained by 'getModelParamsDf'
`maxit`	maximal number of iterations the optimization should be given, default is set to 500
`independentFiltering`	boolean flag indicating whether independent filtering should be performed based on minimal fold changes per protein profile

fcThres	numeric value of minimal fold change (or inverse fold change) a protein has to show to be kept upon independent filtering
minObs	numeric value of minimal number of observations that should be required per protein
optim_fun_h0	optimization function that should be used for fitting the H0 model
optim_fun_h1	optimization function that should be used for fitting the H1 model
optim_fun_h1_2	optional additional optimization function that will be run with paramters retrieved from optim_fun_h1 and should be used for fitting the H1 model with the trimmed sum model, default is NULL
gr_fun_h0	optional gradient function for optim_fun_h0, default is NULL
gr_fun_h1	optional gradient function for optim_fun_h1, default is NULL
gr_fun_h1_2	optional gradient function for optim_fun_h1_2, default is NULL
BPPARAM	BiocParallel parameter for optional parallelization of null distribution generation through bootstrapping, default: BiocParallel::SerialParam()
B	numeric value of rounds of bootstrap, default: 20
byMsExp	boolean flag indicating whether resampling of residuals should be performed separately for data generated by different MS experiments, default TRUE, recommended
verbose	logical indicating whether to print each protein while its profile is boostrapped

Value

data frame containing F statistics of proteins with permuted 2D thermal profiles that are informative on the Null distribution of F statistics

Examples

```
data("simulated_cell_extract_df")
temp_df <- simulated_cell_extract_df %>%
  filter(clustername %in% paste0("protein", 1:3)) %>%
  group_by(representative) %>%
  mutate(nObs = n()) %>%
  ungroup
temp_params_df <- getModelParamsDf(temp_df)
boot_df <- bootstrapNullAlternativeModelFast(
  temp_df, params_df = temp_params_df, B = 20)
```

---

competeModels *Compete H0 and H1 models per protein and obtain F statistic*

Description

Compete H0 and H1 models per protein and obtain F statistic

Usage

```
competeModels(
  df,
  fcThres = 1.5,
  independentFiltering = FALSE,
  minObs = 20,
  optim_fun_h0 = .min_RSS_h0,
  optim_fun_h1 = .min_RSS_h1_slope_pEC50,
  optim_fun_h1_2 = NULL,
  gr_fun_h0 = NULL,
  gr_fun_h1 = NULL,
  gr_fun_h1_2 = NULL,
  maxit = 750
)
```

Arguments

df	tidy data frame retrieved after import of a 2D-TPP dataset, potential filtering and addition of a column "nObs" containing the number of observations per protein
fcThres	numeric value of minimal fold change (or inverse fold change) a protein has to show to be kept upon independent filtering
independentFiltering	boolean flag indicating whether independent filtering should be performed based on minimal fold changes per protein profile
minObs	numeric value of minimal number of observations that should be required per protein
optim_fun_h0	optimization function that should be used for fitting the H0 model
optim_fun_h1	optimization function that should be used for fitting the H1 model
optim_fun_h1_2	optional additional optimization function that will be run with paramters retrieved from optim_fun_h1 and should be used for fitting the H1 model with the trimmed sum model, default is NULL
gr_fun_h0	optional gradient function for optim_fun_h0, default is NULL
gr_fun_h1	optional gradient function for optim_fun_h1, default is NULL
gr_fun_h1_2	optional gradient function for optim_fun_h1_2, default is NULL
maxit	maximal number of iterations the optimization should be given, default is set to 500

Value

data frame summarising the fit characteristics of H0 and H1 models and therof resulting computed F statistics per protein

Examples

```
data("simulated_cell_extract_df")
temp_df <- simulated_cell_extract_df %>%
```

```
  filter(clustername %in% paste0("protein", 1:10)) %>%
  group_by(representative) %>%
  mutate(nObs = n()) %>%
  ungroup
competeModels(temp_df)
```

`computeFstat` *Compute F statistic from H1 and H0 model characteristics*

Description

Compute F statistic from H1 and H0 model characteristics

Usage

```
computeFstat(h0_df, h1_df)
```

Arguments

`h0_df`	data frame with H0 model characteristics for each protein
`h1_df`	data frame with H1 model characteristics for each protein

Value

data frame with H0 and H1 model characteristics for each protein and respectively computed F statistics

Examples

```
data("simulated_cell_extract_df")
temp_df <- simulated_cell_extract_df %>%
  filter(clustername %in% paste0("protein", 1:20)) %>%
  group_by(representative) %>%
  mutate(nObs = n()) %>%
  ungroup

h0_df <- fitH0Model(temp_df)
h1_df <- fitH1Model(temp_df)

computeFstat(h0_df, h1_df)
```

computeFStatFromParams

Compute F statistics from paramter data frame

Description

Compute F statistics from paramter data frame

Usage

```
computeFStatFromParams(params_df)
```

Arguments

params_df	data frame listing all null and alternative model parameters as obtained by 'getModelParamsDf'

Value

data frame of all proteins and computed F statistics and parameters that were used for the computation

Examples

```
data("simulated_cell_extract_df")
params_df <- getModelParamsDf(simulated_cell_extract_df)
computeFStatFromParams(params_df)
```

configWide2Long

Tranform configuration table from wide to long

Description

Tranform configuration table from wide to long

Usage

```
configWide2Long(configWide)
```

Arguments

configWide	data frame containing a config table

Value

data frame containing config table in long format

Examples

```
data("config_tab")
configWide2Long(configWide = config_tab)
```

---

config_tab — *Example config table for a import of a simulated 2D-TPP cell extract dataset*

Description

Config table fot import of simulated example dataset obtained by 2D-TPP experiments for analysis by the TPP2D-package. It's a data frame with the columns "Compound" describing the compound used for the assay, "Experiment" listing MS experiment ids of the separate runs (typically comprising two multiplexed adjacent temperature), "Temperature": the temperature used for a given sub-experimet, the respective TMT labels "126"-"131L", RefCol referring to the label used as a reference label for computing relative fold changes (usually the label used for the control treatment). Please note that when the data is not supplied as a list of already imported data frames the config table for the import function should be a path to an txt, csv or xlsx file containing an additional column "Path" listing for each row the respective path to a searched protein output file.

Usage

```
data("config_tab")
```

Format

"Compound" describing the compound used for the assay, "Experiment" listing MS experiment ids of the separate runs (typically comprising two multiplexed adjacent temperature), "Temperature": the temperature used for a given sub-experimet, the respective TMT labels "126"-"131L", RefCol referring to the label used as a reference label for computing relative fold changes (usually the label used for the control treatment).

---

filterOutContaminants — *Filter out contaminants*

Description

Filter out contaminants

Usage

```
filterOutContaminants(dataLong)
```

Arguments

dataLong	long format data frame of imported dataset

Value

data frame containing full dataset filtered to contain no contaminants

Examples

```
data("simulated_cell_extract_df")
filterOutContaminants(simulated_cell_extract_df)
```

---

findHits *Find hits according to FDR threshold*

Description

Find hits according to FDR threshold

Usage

```
findHits(fdr_df, alpha)
```

Arguments

fdr_df	data frame obtained from computeFdr
alpha	significance threshold, default is set to 0.1

Value

data frame of significant hits at FDR <= alpha

Examples

```
data("simulated_cell_extract_df")
temp_df <- simulated_cell_extract_df %>%
  filter(clustername %in% paste0("protein", 1:5)) %>%
  group_by(representative) %>%
  mutate(nObs = n()) %>%
  ungroup
example_out <- fitAndEvalDataset(temp_df)
example_null <- bootstrapNull(temp_df, B = 1)
fdr_df <- getFDR(example_out, example_null)
findHits(fdr_df, 0.1)
```

`fitAndEvalDataset` *Fit H0 and H1 model to 2D thermal profiles of proteins and compute F statistic*

Description

Fit H0 and H1 model to 2D thermal profiles of proteins and compute F statistic

Usage

```
fitAndEvalDataset(
  df,
  maxit = 500,
  optim_fun_h0 = .min_RSS_h0,
  optim_fun_h1 = .min_RSS_h1_slope_pEC50,
  optim_fun_h1_2 = NULL,
  gr_fun_h0 = NULL,
  gr_fun_h1 = NULL,
  gr_fun_h1_2 = NULL,
  ec50_lower_limit = NULL,
  ec50_upper_limit = NULL,
  slopEC50 = TRUE
)
```

Arguments

`df`	tidy data_frame retrieved after import of a 2D-TPP dataset, potential filtering and addition of a column "nObs" containing the number of observations per protein
`maxit`	maximal number of iterations the optimization should be given, default is set to 500
`optim_fun_h0`	optimization function that should be used for fitting the H0 model
`optim_fun_h1`	optimization function that should be used for fitting the H1 model
`optim_fun_h1_2`	optional additional optimization function that will be run with paramters retrieved from optim_fun_h1 and should be used for fitting the H1 model with the trimmed sum model, default is NULL
`gr_fun_h0`	optional gradient function for optim_fun_h0, default is NULL
`gr_fun_h1`	optional gradient function for optim_fun_h1, default is NULL
`gr_fun_h1_2`	optional gradient function for optim_fun_h1_2, default is NULL
`ec50_lower_limit`	lower limit of ec50 parameter
`ec50_upper_limit`	lower limit of ec50 parameter
`slopEC50`	logical flag indicating whether the h1 model is fitted with a linear model describing the shift od the pEC50 over temperatures

Value

data frame with H0 and H1 model characteristics for each protein and respectively computed F statistics

Examples

```
data("simulated_cell_extract_df")
temp_df <- simulated_cell_extract_df %>%
  group_by(representative) %>%
  mutate(nObs = n()) %>%
  ungroup
fitAndEvalDataset(temp_df)
```

---

fitH0Model — *Fit H0 model and evaluate fit statistics*

Description

Fit H0 model and evaluate fit statistics

Usage

```
fitH0Model(df, maxit = 500, optim_fun = .min_RSS_h0, gr_fun = NULL)
```

Arguments

df	tidy data_frame retrieved after import of a 2D-TPP dataset, potential filtering and addition of a column "nObs" containing the number of observations per protein
maxit	maximal number of iterations the optimization should be given, default is set to 500
optim_fun	optimization function that should be used for fitting the H0 model
gr_fun	optional gradient function for optim_fun, default is NULL

Value

data frame with H0 model characteristics for each protein

Examples

```
data("simulated_cell_extract_df")
temp_df <- simulated_cell_extract_df %>%
  filter(clustername %in% paste0("protein", 1:5)) %>%
  group_by(representative) %>%
  mutate(nObs = n()) %>%
  ungroup
```

```
fitH0Model(temp_df)
```

`fitH1Model` *Fit H1 model and evaluate fit statistics*

Description

Fit H1 model and evaluate fit statistics

Usage

```
fitH1Model(
  df,
  maxit = 500,
  optim_fun = .min_RSS_h1_slope_pEC50,
  optim_fun_2 = NULL,
  gr_fun = NULL,
  gr_fun_2 = NULL,
  ec50_lower_limit = NULL,
  ec50_upper_limit = NULL,
  slopEC50 = TRUE
)
```

Arguments

`df`	tidy data_frame retrieved after import of a 2D-TPP dataset, potential filtering and addition of a column "nObs" containing the number of observations per protein
`maxit`	maximal number of iterations the optimization should be given, default is set to 500
`optim_fun`	optimization function that should be used for fitting the H0 model
`optim_fun_2`	optional secound optimization function for fitting the H1 model that should be used based on the fitted parameters of the optimizationfor based on optim_fun
`gr_fun`	optional gradient function for optim_fun, default is NULL
`gr_fun_2`	optional gradient function for optim_fun_2, default is NULL
`ec50_lower_limit`	lower limit of ec50 parameter
`ec50_upper_limit`	lower limit of ec50 parameter
`slopEC50`	logical flag indicating whether the h1 model is fitted with a linear model describing the shift od the pEC50 over temperatures

Value

data frame with H1 model characteristics for each protein

Examples

```
data("simulated_cell_extract_df")
temp_df <- simulated_cell_extract_df %>%
  filter(clustername %in% paste0("protein", 1:5)) %>%
  group_by(representative) %>%
  mutate(nObs = n()) %>%
  ungroup

fitH1Model(temp_df)
```

---

getFDR — *Get FDR for given F statistics based on true and null dataset*

Description

Get FDR for given F statistics based on true and null dataset

Usage

```
getFDR(df_out, df_null, squeezeDenominator = TRUE)
```

Arguments

df_out	data frame containing results from analysis by fitAndEvalDataset
df_null	data frame containing results from analysis by bootstrapNull
squeezeDenominator	logical indicating whether F statistic denominator should be shrinked using limma::squeezeVar

Value

data frame annotating each protein with a FDR based on it's F statistic and number of observations

Examples

```
data("simulated_cell_extract_df")
temp_df <- simulated_cell_extract_df %>%
  filter(clustername %in% paste0("protein", 1:5)) %>%
  group_by(representative) %>%
  mutate(nObs = n()) %>%
  ungroup
example_out <- fitAndEvalDataset(temp_df)
example_null <- bootstrapNull(temp_df, B = 1)
getFDR(example_out, example_null)
```

getModelParamsDf *Get H0 and H1 model parameters*

Description

Get H0 and H1 model parameters

Usage

```
getModelParamsDf(
  df,
  minObs = 20,
  optim_fun_h0 = .min_RSS_h0,
  optim_fun_h1 = .min_RSS_h1_slope_pEC50,
  optim_fun_h1_2 = NULL,
  gr_fun_h0 = NULL,
  gr_fun_h1 = NULL,
  gr_fun_h1_2 = NULL,
  slopEC50 = TRUE,
  maxit = 500,
  qualColName = "qupm"
)
```

Arguments

df	tidy data_frame retrieved after import of a 2D-TPP dataset, potential filtering and addition of a column "nObs" containing the number of observations per protein
minObs	numeric value of minimal number of observations that should be required per protein
optim_fun_h0	optimization function that should be used for fitting the H0 model
optim_fun_h1	optimization function that should be used for fitting the H1 model
optim_fun_h1_2	optional additional optimization function that will be run with paramters retrieved from optim_fun_h1 and should be used for fitting the H1 model with the trimmed sum model, default is NULL
gr_fun_h0	optional gradient function for optim_fun_h0, default is NULL
gr_fun_h1	optional gradient function for optim_fun_h1, default is NULL
gr_fun_h1_2	optional gradient function for optim_fun_h1_2, default is NULL
slopEC50	logical flag indicating whether the h1 model is fitted with a linear model describing the shift od the pEC50 over temperatures
maxit	maximal number of iterations the optimization should be given, default is set to 500
qualColName	name of column indicating quantification quality e.g. number of unique peptides used for quantification, default: "qupm"

Value

a data.frame with fitted null and alternative model parameters

Examples

```
data("simulated_cell_extract_df")
getModelParamsDf(simulated_cell_extract_df)
```

---

getPEC504Temperature — *Get pEC50 for a protein of interest at a specific temperatures (optimally the melting point of the protein)*

Description

Get pEC50 for a protein of interest at a specific temperatures (optimally the melting point of the protein)

Usage

```
getPEC504Temperature(fstat_df, protein, temperaturePEC50 = 60)
```

Arguments

`fstat_df`	data frame as obtained after calling `getModelParamsDf`, containing fitted null and alternative model parameters for each protein
`protein`	character string referring to the protein of interest
`temperaturePEC50`	temperature (numeric) at which pEC50 should be inferred

Value

numeric value specifying the pEC50 for the indicated protein and temperature

Examples

```
data("simulated_cell_extract_df")

model_params_df <- getModelParamsDf(
   df = filter(simulated_cell_extract_df,
           clustername == "tp1"))

getPEC504Temperature(
    fstat_df = model_params_df,
    protein = "tp1",
    temperaturePEC50 = 60)
```

getPvalues — *Compute p-values for given F statistics based on true and null dataset*

Description

Compute p-values for given F statistics based on true and null dataset

Usage

```
getPvalues(df_out, df_null, pseudo_count = 1, squeezeDenominator = FALSE)
```

Arguments

df_out	data frame containing results from analysis by fitAndEvalDataset
df_null	data frame containing results from analysis by bootstrapNull
pseudo_count	numeric larger or equal to 0 added to both counts of protein with an F-statistic higher than a threshold theta of the true and bootstrapped datasets
squeezeDenominator	logical indicating whether F statistic denominator should be shrinked using limma::squeezeVar

Value

data frame annotating each protein with a FDR based on it's F statistic and number of observations

Examples

```
data("simulated_cell_extract_df")
temp_df <- simulated_cell_extract_df %>%
  filter(clustername %in% paste0("protein", 1:3)) %>%
  group_by(representative) %>%
  mutate(nObs = n()) %>%
  ungroup
example_out <- fitAndEvalDataset(temp_df)
example_null <- bootstrapNull(temp_df, B = 2)
getPvalues(
  example_out,
  example_null)
```

gg_qq *Plot qq-plot of true data and bootstrapped null with ggplot*

Description

Plot qq-plot of true data and bootstrapped null with ggplot

Usage

```
gg_qq(
  x,
  y,
  xlab = "F-statistics from sampled Null distr.",
  ylab = "observed F-statistics",
  alpha = 0.25,
  gg_theme = theme_classic(),
  offset = 1,
  plot_diagonal = TRUE
)
```

Arguments

x	vector containing values of values of first distribution to compare
y	vector containing values of values of secound distribution to compare
xlab	x-axis label
ylab	y-axis label
alpha	transparency paramenter between 0 and 1
gg_theme	ggplot theme, default is theme_classic()
offset	offset for x and y axis on top of maximal values
plot_diagonal	logical parameter indicating whether an identity line should be plotted

Value

A ggplot displaying the qq-plot of a true and a a bootstrapped null distribution

Examples

```
data("simulated_cell_extract_df")
recomputeSignalFromRatios(simulated_cell_extract_df)
```

import2dDataset *Import 2D-TPP dataset using a config table*

Description

Import 2D-TPP dataset using a config table

Usage

```
import2dDataset(
  configTable,
  data,
  idVar = "representative",
  intensityStr = "sumionarea_protein_",
  fcStr = "rel_fc_protein_",
  nonZeroCols = "qssm",
  geneNameVar = "clustername",
  addCol = NULL,
  qualColName = "qupm",
  naStrs = c("NA", "n/d", "NaN"),
  concFactor = 1e+06,
  medianNormalizeFC = TRUE,
  filterContaminants = TRUE
)
```

Arguments

configTable	character string of a file path to a config table
data	possible list of datasets from different MS runs corresponding to a 2D-TPP dataset, circumvents loading datasets referencend in config table, default is NULL
idVar	character string indicating which data column provides the unique identifiers for each protein.
intensityStr	character string indicating which columns contain raw intensities measurements
fcStr	character string indicating which columns contain the actual fold change values. Those column names containing the suffix `fcStr` will be regarded as containing fold change values.
nonZeroCols	column like default qssm that should be imported and requested to be non-zero in analyzed data
geneNameVar	character string of the column name that describes the gene name of a given protein in the raw data files
addCol	character string indicating additional column to import
qualColName	character string indicating which column can be used for additional quality criteria when deciding between different non-unique protein identifiers.

naStrs — character vector indicating missing values in the data table. When reading data from file, this value will be passed on to the argument `na.strings` in function `read.delim`.

concFactor — numeric value that indicates how concentrations need to be adjusted to yield total unit e.g. default mmol - 1e6

medianNormalizeFC — perform median normalization (default: TRUE).

filterContaminants — boolean variable indicating whether data should be filtered to exclude contaminants (default: TRUE).

Value

tidy data frame representing a 2D-TPP dataset

Examples

```
data("config_tab")
data("raw_dat_list")
import_df <- import2dDataset(configTable = config_tab,
                             data = raw_dat_list,
                             idVar = "protein_id",
                             intensityStr = "signal_sum_",
                             fcStr = "rel_fc_",
                             nonZeroCols = "qusm",
                             geneNameVar = "gene_name",
                             addCol = NULL,
                             qualColName = "qupm",
                             naStrs = c("NA", "n/d", "NaN"),
                             concFactor = 1e6,
                             medianNormalizeFC = TRUE,
                             filterContaminants = TRUE)
```

---

import2dMain — *Import 2D-TPP dataset main function*

Description

Import 2D-TPP dataset main function

Usage

```
import2dMain(
  configTable,
  data,
  idVar,
  fcStr,
```

```
  addCol,
  naStrs,
  intensityStr,
  qualColName,
  nonZeroCols
)
```

Arguments

configTable	character string of a file path to a config table
data	possible list of datasets from different MS runs corresponding to a 2D-TPP dataset, circumvents loading datasets referencend in config table, default is NULL
idVar	character string indicating which data column provides the unique identifiers for each protein.
fcStr	character string indicating which columns contain the actual fold change values. Those column names containing the suffix `fcStr` will be regarded as containing fold change values.
addCol	character string indicating additional column to import
naStrs	character vector indicating missing values in the data table. When reading data from file, this value will be passed on to the argument `na.strings` in function `read.delim`.
intensityStr	character string indicating which columns contain raw intensities measurements
qualColName	character string indicating which column can be used for additional quality criteria when deciding between different non-unique protein identifiers.
nonZeroCols	column like default qssm that should be imported and requested to be non-zero in analyzed data

Value

list of data frames containing different datasets

Examples

```
data("config_tab")
data("raw_dat_list")
dataList <- import2dMain(configTable = config_tab,
                         data = raw_dat_list,
                         idVar = "protein_id",
                         fcStr = "rel_fc_",
                         addCol = "gene_name",
                         naStrs = NA,
                         intensityStr = "signal_sum_",
                         nonZeroCols = "qusm",
                         qualColName = "qupm")
```

plot2dTppFcHeatmap — *Plot heatmap of 2D thermal profile fold changes of a protein of choice*

Description

Plot heatmap of 2D thermal profile fold changes of a protein of choice

Usage

```
plot2dTppFcHeatmap(df, name, drug_name = "", midpoint = 1)
```

Arguments

df	tidy data frame of a 2D-TPP dataset
name	gene name (clustername) of protein that should be visualized
drug_name	character string of profiled drug name
midpoint	midpoint of fold changes for color scaling, default: 1

Value

A ggplot displaying the thermal profile as a heatmap of fold changes of a protein of choice in a dataset of choice

Examples

```
data("simulated_cell_extract_df")
plot2dTppFcHeatmap(simulated_cell_extract_df,
 "tp2", drug_name = "drug1")
```

plot2dTppFit — *Plot H0 or H1 fit of 2D thermal profile intensities of a protein of choice*

Description

Plot H0 or H1 fit of 2D thermal profile intensities of a protein of choice

Usage

```
plot2dTppFit(
  df,
  name,
  model_type = "H0",
  optim_fun = .min_RSS_h0,
  optim_fun_2 = NULL,
  maxit = 500,
  xlab = "-log10(conc.)",
  ylab = "log2(summed intensities)",
  dot_size = 1,
  line_type = "solid",
  fit_color = "gray30"
)
```

Arguments

df	tidy data frame of a 2D-TPP dataset
name	gene name (clustername) of protein that should be visualized
model_type	character string indicating whether the "H0" or the "H1" model should be fitted
optim_fun	optimization function that should be used for fitting either the H0 or H1 model
optim_fun_2	optional additional optimization function that will be run with paramters retrieved from optim_fun and should be used for fitting the H1 model with the trimmed sum model, default is NULL
maxit	maximal number of iterations the optimization should be given, default is set to 500
xlab	character string of x-axis label of plot
ylab	character string of y-axis label of plot
dot_size	numeric indicating the size of the data points to plot
line_type	character string defining the line type of the fitted curve, default "dashed"
fit_color	character string defining the color of the fitted curve

Value

A ggplot displaying the thermal profile of a protein of choice in a datset of choice

Examples

```
data("simulated_cell_extract_df")
plot2dTppProfile(simulated_cell_extract_df, "protein1")
```

plot2dTppProfile *Plot 2D thermal profile intensities of a protein of choice*

Description

Plot 2D thermal profile intensities of a protein of choice

Usage

```
plot2dTppProfile(df, name)
```

Arguments

df	tidy data frame of a 2D-TPP dataset
name	gene name (clustername) of protein that should be visualized

Value

A ggplot displaying the thermal profile of a protein of choice in a datset of choice

Examples

```
data("simulated_cell_extract_df")
plot2dTppProfile(simulated_cell_extract_df, "protein1")
```

plot2dTppRelProfile *Plot 2D thermal profile ratios of a protein of choice*

Description

Plot 2D thermal profile ratios of a protein of choice

Usage

```
plot2dTppRelProfile(df, name)
```

Arguments

df	tidy data frame of a 2D-TPP dataset
name	gene name (clustername) of protein that should be visualized

Value

A ggplot displaying the thermal profile ratios of a protein of choice in a datset of choice

Examples

```
data("simulated_cell_extract_df")
plot2dTppRelProfile(simulated_cell_extract_df, "protein1")
```

`plot2dTppVolcano` *Plot Volcano plot of TPP2D results*

Description

Plot Volcano plot of TPP2D results

Usage

```
plot2dTppVolcano(
  fdr_df,
  hits_df,
  alpha = 0.5,
  title_string = "",
  x_lim = NULL,
  y_lim = NULL,
  facet_by_obs = FALSE
)
```

Arguments

`fdr_df`	data frame obtained from ‘getFDR‘
`hits_df`	hits_df data frame obtained from ‘findHits‘
`alpha`	transparency level of plotted points
`title_string`	character argument handed over to ggtitle
`x_lim`	vector with two numerics indicating the x axis limits
`y_lim`	vector with two numerics indicating the y axis limits
`facet_by_obs`	logical indicating whether plot should be facetted by number of observations, default: FALSE

Value

a ggplot displaying a volcano plot of the results obtained after a TPP2D analysis

Examples

```
data("simulated_cell_extract_df")
temp_df <- simulated_cell_extract_df %>%
  filter(clustername %in% paste0("protein", 1:5)) %>%
  group_by(representative) %>%
  mutate(nObs = n()) %>%
  ungroup
example_params <- getModelParamsDf(temp_df)
example_fstat <- computeFStatFromParams(example_params)
example_null <- bootstrapNullAlternativeModel(
   df = temp_df, params_df = example_params,
   B = 2)
fdr_df <- getFDR(example_fstat, example_null)
hits_df <- findHits(fdr_df, 0.1)
plot2dTppVolcano(fdr_df = fdr_df, hits_df = hits_df)
```

---

raw_dat_list

Example raw data for a subset of a simulated 2D-TPP cell extract dataset

---

Description

Simulated example dataset obtained by 2D-TPP experiments for analysis by the TPP2D-package. It contains a list of data frames resembling raw data files returned from a MS database search with 200 simulated protein profiles (protein1-200) and 3 spiked-in true positives (TP1-3).

Usage

```
data("raw_dat_list")
```

Format

list of data frames with columns representative (protein id), clustername (gene name), temperature, log_conc, raw_value, rel_value, value and log2_value

---

recomputeSignalFromRatios

Recompute robust signal intensities based on bootstrapped TMT channel ratios

---

Description

Recompute robust signal intensities based on bootstrapped TMT channel ratios

Usage

```
recomputeSignalFromRatios(df)
```

Arguments

df	tidy data_frame retrieved after import of a 2D-TPP dataset

Value

A data_frame with recomputed signal intensities (columname: value) and log2 transformed signal intensities (columnanme: log2_value) that more reliably reflect relative ratios between the TMT channels

Examples

```
data("simulated_cell_extract_df")
recomputeSignalFromRatios(simulated_cell_extract_df)
```

---

renameColumns *Rename columns of imported data frame*

Description

Rename columns of imported data frame

Usage

```
renameColumns(dataLong, idVar, geneNameVar)
```

Arguments

dataLong	long format data frame of imported dataset
idVar	character string indicating which data column provides the unique identifiers for each protein.
geneNameVar	character string of the column name that describes the gene name of a given protein in the raw data files

Value

data frame containing imported data with renamed columns

Examples

```
data("config_tab")
data("raw_dat_list")

dataList <- import2dMain(configTable = config_tab,
                         data = raw_dat_list,
                         idVar = "protein_id",
                         fcStr = "rel_fc_",
                         addCol = "gene_name",
```

```
                          naStrs = NA,
                          intensityStr = "signal_sum_",
                          nonZeroCols = "qusm",
                          qualColName = "qupm")
configLong <- configWide2Long(configWide = config_tab)
annoDat <- annotateDataList(dataList = dataList,
                            geneNameVar = "gene_name",
                            configLong = configLong,
                            intensityStr = "signal_sum_",
                            fcStr = "rel_fc_")
renameColumns(annoDat,
              idVar = "protein_id",
              geneNameVar = "gene_name")
```

`resolveAmbiguousProteinNames`
Resolve ambiguous protein names

Description

Resolve ambiguous protein names

Usage

```
resolveAmbiguousProteinNames(df, includeIsoforms = FALSE)
```

Arguments

`df`	tidy data_frame retrieved after import of a 2D-TPP dataset
`includeIsoforms`	logical indicating whether protein isoform should be kept for analysis

Value

data frame with resolved protein name ambiguity

Examples

```
tst_df <- bind_rows(tibble(representative = rep(1:3, each = 3),
                           clustername = rep(letters[1:3], each = 3)),
                    tibble(representative = rep(c(4, 5), c(3, 2)),
                           clustername = rep(c("a", "b"), c(3, 2))))

resolveAmbiguousProteinNames(tst_df)
```

runTPP2D *Run complete TPP2D analysis*

Description

Run complete TPP2D analysis

Usage

```
runTPP2D(
  df = NULL,
  configTable = NULL,
  data = NULL,
  idVar = "protein_id",
  intensityStr = "signal_sum_",
  fcStr = "rel_fc_",
  nonZeroCols = "qusm",
  geneNameVar = "gene_name",
  addCol = NULL,
  qualColName = "qupm",
  naStrs = c("NA", "n/d", "NaN"),
  concFactor = 1e+06,
  medianNormalizeFC = TRUE,
  filterContaminants = TRUE,
  recomputeSignalRatios = FALSE,
  minObs = 20,
  independentFiltering = FALSE,
  fcThres = 1.5,
  optim_fun_h0 = .min_RSS_h0,
  optim_fun_h1 = .min_RSS_h1_slope_pEC50,
  optim_fun_h1_2 = NULL,
  gr_fun_h0 = NULL,
  gr_fun_h1 = NULL,
  gr_fun_h1_2 = NULL,
  slopEC50 = TRUE,
  maxit = 750,
  BPPARAM = BiocParallel::SerialParam(progressbar = TRUE),
  B = 20,
  byMsExp = TRUE,
  alpha = 0.1
)
```

Arguments

df	tidy data_frame retrieved after import of a 2D-TPP dataset, potential filtering and addition of a column "nObs" containing the number of observations per protein

`configTable` character string of a file path to a config table

`data` possible list of datasets from different MS runs corresponding to a 2D-TPP dataset, circumvents loading datasets referencend in config table, default is NULL

`idVar` character string indicating which data column provides the unique identifiers for each protein.

`intensityStr` character string indicating which columns contain raw intensities measurements

`fcStr` character string indicating which columns contain the actual fold change values. Those column names containing the suffix `fcStr` will be regarded as containing fold change values.

`nonZeroCols` column like default qssm that should be imported and requested to be non-zero in analyzed data

`geneNameVar` character string of the column name that describes the gene name of a given protein in the raw data files

`addCol` character string indicating additional column to import

`qualColName` character string indicating which column can be used for additional quality criteria when deciding between different non-unique protein identifiers.

`naStrs` character vector indicating missing values in the data table. When reading data from file, this value will be passed on to the argument `na.strings` in function `read.delim`.

`concFactor` numeric value that indicates how concentrations need to be adjusted to yield total unit e.g. default mmol - 1e6

`medianNormalizeFC`
perform median normalization (default: TRUE).

`filterContaminants`
logical variable indicating whether data should be filtered to exclude contaminants (default: TRUE).

`recomputeSignalRatios`
logical variable indicaiting whether signals should be recomputed from relative fold changes, recommended if Isobarquant was used for protein quantification

`minObs` number of minimal observations per protein to include it in the analysis

`independentFiltering`
logical variable indicating whether independent filtering should be performed based on minimal fold changes per protein profile

`fcThres` numeric value of minimal fold change (or inverse fold change) a protein has to show to be kept upon independent filtering

`optim_fun_h0` optimization function that should be used for fitting the H0 model

`optim_fun_h1` optimization function that should be used for fitting the H1 model

`optim_fun_h1_2` optional additional optimization function that will be run with paramters retrieved from optim_fun_h1 and should be used for fitting the H1 model with the trimmed sum model, default is NULL

`gr_fun_h0` optional gradient function for optim_fun_h0, default is NULL

`gr_fun_h1` optional gradient function for optim_fun_h1, default is NULL

gr_fun_h1_2	optional gradient function for optim_fun_h1_2, default is NULL
slopEC50	logical flag indicating whether the h1 model is fitted with a linear model describing the shift od the pEC50 over temperatures
maxit	maximal number of iterations the optimization should be given, default is set to 500
BPPARAM	= BiocParallel::SerialParam(progressbar = TRUE),
B	numeric value indicating number of rounds of bootstraps that should be performed to estimate the null distribution
byMsExp	logical indicating whether bootstrapping should be performed within MS experiments
alpha	FDR level that should be controlled

Value

a tpp2dExperiment object

Examples

```
data("simulated_cell_extract_df")
runTPP2D(df = simulated_cell_extract_df %>%
   filter(representative %in% 1:3),
   B = 1)
```

---

simulated_cell_extract_df

Example subset of a simulated 2D-TPP cell extract dataset

Description

Simulated example dataset obtained by 2D-TPP experiments for analysis by the TPP2D-package. It contains a tidy data frame after import and recomputing of robust signal intensities with 200 simulated protein profiles (protein1-200) and 3 spiked-in true positives (TP1-3)

Usage

```
data("simulated_cell_extract_df")
```

Format

data frame with columns representative (protein id), clustername (gene name), temperature, log_conc, raw_value, rel_value, value and log2_value

tpp2dExperiment-class *S4 TPP2D Experiment Class*

Description

S4 TPP2D Experiment Class

Value

an object of class tpp2dExperiment

Slots

`configTable` data.frame.

`idVar` character.

`intensityStr` character.

`fcStr` character.

`nonZeroCols` character.

`geneNameVar` character.

`qualColName` character.

`naStrs` character.

`concFactor` numeric.

`medianNormalizeFC` logical.

`filterContaminants` logical.

`minObs` numeric.

`independentFiltering` logical.

`fcThres` numeric.

`optim_fun_h0` function.

`optim_fun_h1` function.

`slopEC50` logical.

`maxit` numeric.

`BPPARAM` character.

`B` numeric

`byMsExp` logical.

`alpha` numeric.

`tidyDataTable` data.frame.

`modelParamsDf` data.frame

`resultTable` data.frame

`bootstrapNullDf` data.frame

`hitTable` data.frame

Examples

```
tpp2dObj <- new("tpp2dExperiment")
```

TPP_importCheckConfigTable

Import and chech configuration table

Description

Import and chech configuration table

Usage

```
TPP_importCheckConfigTable(infoTable, type = "2D")
```

Arguments

infoTable	character string of a file path to a config table (excel,txt or csv file) or data frame containing a config table
type	charater string indicating dataset type default is 2D

Value

data frame with config table

Examples

```
data("config_tab")
TPP_importCheckConfigTable(config_tab, type = "2D")
```

Index