# Package ‘TPP’

April 15, 2017

**Type** Package

**Title** Analyze thermal proteome profiling (TPP) experiments

**Version** 3.0.3

**Author** Dorothee Childs, Nils Kurzawa, Holger Franken, Carola Doce, Mikhail Savitski and Wolfgang Huber

**Maintainer** Dorothee Childs <dorothee.childs@embl.de>

**Depends** R (>= 3.3), dplyr, magrittr, tidyr

**Imports** Biobase, doParallel, foreach, ggplot2, gridExtra, grid, limma, MASS, nls2, openxlsx (>= 2.4.0), parallel, plyr, RColorBrewer, RCurl, reshape2, sme, splines, VennDiagram, VGAM, knitr, rmarkdown, grDevices, stats, utils

**Suggests** BiocStyle, testthat

**Description** Analyze thermal proteome profiling (TPP) experiments with varying temperatures (TR) or compound concentrations (CCR).

**License** Artistic-2.0

**VignetteBuilder** knitr

**biocViews** Proteomics, MassSpectrometry

**RoxygenNote** 5.0.1

**NeedsCompilation** no

## R topics documented:

## analyze2DTPP *Analyze 2D-TPP experiment*

### Description

Performs analysis of a 2D-TPP experiment by invoking routines for data import, data processing, fold change computation, median normalization, TPP-CCR curve fitting, plotting and production of the result table.

### Usage

```
analyze2DTPP(configFile = NULL, data = NULL, resultPath = NULL,
  idVar = "gene_name", fcStr = "rel_fc_", intensityStr = "signal_sum_",
  naStrs = c("NA", "n/d", "NaN", "<NA>"), methods = c("doseRespone",
  "splineFit"), qualColName = "qupm", compFc = TRUE, normalize = TRUE,
  addCol = NULL, nCores = "max", nonZeroCols = "qssm",
  fcTolerance = 0.1, r2Cutoff = 0.8, fcCutoff = 1.5, slopeBounds = c(1,
  50), fractAbund = FALSE, xlsxExport = FALSE, plotAll = FALSE,
  plotAllR2 = FALSE, plotSingle = FALSE, trRef = NULL,
  refFcStr = "norm_rel_fc_", addInfo = FALSE,
  createReport = "html_document")
```

### Arguments

| | |
|---|---|
| `configFile` | dataframe, or character object with the path to a file, that specifies important details of the 2D-TPP experiment. See Section `details` for instructions how to create this object. |
| `data` | single dataframe, containing fold change measurements and additional annotation columns to be imported. Can be used instead of specifying the file path in the `configTable` argument. |
| `resultPath` | location where to store dose-response curve plots and results table. |
| `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 prefix `fcStr` will be regarded as containing fold change values. |
| `intensityStr` | character string indicating which columns contain the actual sumionarea values. Those column names containing the prefix `intensityStr` will be regarded as containing sumionarea values. |
| `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`. |
| `methods` | vector of character strings that indicate which methods should be used for the analysis (default: c("doseResponse"), alternative: c("splineFit") or c("doseResponse", "splineFit")) |
| `qualColName` | character string indicating which column can be used for additional quality criteria when deciding between different non-unique protein identifiers. |
| `compFc` | boolean flag which indicates whether to perform fold change computation regarding reference column from sumionareas (default: TRUE) |

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

`addCol` character vector indicating which additional columns to include from the input data

`nCores` either a numerical value given the desired number of CPUs, or 'max' to automatically assign the maximum possible number (default).

`nonZeroCols` character string indicating a column that will be used for filtering out zero values.

`fcTolerance` tolerance for the fcCutoff parameter. See details.

`r2Cutoff` Quality criterion on dose response curve fit.

`fcCutoff` Cutoff for highest compound concentration fold change.

`slopeBounds` Bounds on the slope parameter for dose response curve fitting.

`fractAbund` boolean variable, if set to TRUE additional information concerning sumionarea fractional abundance and dmso1 vs. dmso2 of adjacent temperatures is added to the output table

`xlsxExport` produce results table in xlsx format and store at the location specified by the `resultPath` argument.

`plotAll` boolean value indicating whether all dose response curves should be generated. Deactivating plotting decreases runtime.

`plotAllR2` boolean value indicating whether all dose response curves which fulfill the demanded criteria (Rsquared, maximum plateau) should be generated. Deactivating plotting decreases runtime.

`plotSingle` boolean value indicating whether all dose response curves which fulfill the demanded criteria (Rsquared, maximum plateau) should be generated. Deactivating plotting decreases runtime.

`trRef` character string containing a valid system path to a previously generated TPP-TR reference object

`refFcStr` character string indicating which columns in the reference data set contain the fold change values

`addInfo` boolean variable, if set to TRUE additional information on counts of stabilization and destabilization of each protein is added to the output table

`createReport` character string indicating whether a markdown report should be created and which format it have (default: "html_document", alternative: "pdf_document" or "none")

## Details

Invokes the following steps:

1. Import data using the `tpp2dImport` function.
2. Remove zero sumionarea values.
3. Compute fold changes from raw data (sumionarea)
4. Perform normalization by fold change medians (optional) using the `tpp2dNormalize` function. To perform normalization, set argument `normalize=TRUE`.

## Value

A data frame in which the fit results are stored row-wise for each protein at the different temperatures.

## References

Becher, I., Werner, T., Doce, C., Zaal, E. A., Berkers, C. R., T"ogel, I., Salzer, E., Bantscheff, M., Savitski, M. M. (2016) Comprehensive thermal and chemoproteomics profiling identifies phenylalanine hydroxylase as a potent off-target of the histone deacetylase inhibitor panobinostat. Nature Chemical Biology (accepted)

## Examples

```
data("panob2D_isobQuant_example")
config_tpp2d <- panobinostat_2DTPP_config
data_tpp2d <- panobinostat_2DTPP_data
tpp2dResults <- analyze2DTPP(configFile = config_tpp2d,
                             data = data_tpp2d,
                             fcStr = NULL,
                             methods=c("doseResponse"),
                             createReport="none",
                             nCores=1)
```

---

# analyzeTPPCCR *Analyze TPP-CCR experiment*

---

## Description

Performs analysis of a TPP-CCR experiment by invoking routines for data import, data processing, normalization, curve fitting, and production of the result table.

## Usage

```
analyzeTPPCCR(configTable, data = NULL, resultPath = NULL,
  idVar = "gene_name", fcStr = "rel_fc_", naStrs = c("NA", "n/d", "NaN",
  "<NA>"), qualColName = "qupm", normalize = TRUE,
  ggplotTheme = tppDefaultTheme(), nCores = "max", nonZeroCols = "qssm",
  r2Cutoff = 0.8, fcCutoff = 1.5, slopeBounds = c(1, 50),
  plotCurves = TRUE, verbose = FALSE, xlsxExport = TRUE,
  fcTolerance = 0.1)
```

## Arguments

| | |
|---|---|
| `configTable` | dataframe, or character object with the path to a file, that specifies important details of the TPP-CCR experiment. See Section `details` for instructions how to create this object. |
| `data` | single dataframe, containing fold change measurements and additional annotation columns to be imported. Can be used instead of specifying the file path in the `configTable` argument. |
| `resultPath` | location where to store dose-response curve plots and results table. |
| `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. |

`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`.

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

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

`ggplotTheme` ggplot theme for dose response curve plots.

`nCores` either a numerical value given the desired number of CPUs, or 'max' to automatically assign the maximum possible number (default).

`nonZeroCols` character string indicating a column that will be used for filtering out zero values.

`r2Cutoff` Quality criterion on dose response curve fit.

`fcCutoff` Cutoff for highest compound concentration fold change.

`slopeBounds` Bounds on the slope parameter for dose response curve fitting.

`plotCurves` boolean value indicating whether dose response curves should be plotted. Deactivating plotting decreases runtime.

`verbose` print name of each fitted or plotted protein to the command line as a means of progress report.

`xlsxExport` produce results table in xlsx format and store at the location specified by the `resultPath` argument.

`fcTolerance` tolerance for the fcCutoff parameter. See details.

## Details

Invokes the following steps:

1. Import data using the `tppccrImport` function.
2. Perform normalization by fold change medians (optional) using the `tppccrNormalize` function. To perform normalization, set argument `normalize=TRUE`.
3. Fit and analyze dose response curves using the `tppccrCurveFit` function.
4. Export results to Excel using the `tppExport` function.

The default settings are tailored towards the output of the python package isobarQuant, but can be customized to your own dataset by the arguments `idVar, fcStr, naStrs, qualColName`.

If `resultPath` is not specified, result files are stored at the path defined in the first entry of `configTable$Path`. If the input data are not specified in `configTable`, no result path will be set. This means that no output files or dose response curve plots are produced and `analyzeTPPCCR` just returns the results as a data frame.

The function `analyzeTPPCCR` reports intermediate results to the command line. To suppress this, use `suppressMessages`.

The dose response curve plots will be stored in a subfolder with name `DoseResponse_Curves` at the location specified by `resultPath`.

Only proteins with fold changes bigger than `[fcCutoff * (1 - fcTolerance)` or smaller than `1/(fcCutoff * (1 - fcTolerance))]` will be used for curve fitting. Additionally, the proteins fulfilling the fcCutoff criterion without tolerance will be marked in the output column `meets_FC_requirement`.

### Value

A data frame in which the fit results are stored row-wise for each protein.

### References

Savitski, M. M., Reinhard, F. B., Franken, H., Werner, T., Savitski, M. F., Eberhard, D., ... & Drewes, G. (2014). Tracking cancer drugs in living cells by thermal profiling of the proteome. Science, 346(6205), 1255784.

### See Also

tppDefaultTheme

### Examples

```
data(hdacCCR_smallExample)
tppccrResults <- analyzeTPPCCR(configTable=hdacCCR_config,
                               data=hdacCCR_data, nCores=1)
```

## `analyzeTPPTR` *Analyze TPP-TR experiment*

### Description

Performs analysis of a TPP-TR experiment by invoking routines for data import, data processing, normalization, curve fitting, and production of the result table.

### Usage

```
analyzeTPPTR(configTable, data = NULL, resultPath = NULL,
  methods = c("meltcurvefit", "splinefit"), idVar = "gene_name",
  fcStr = "rel_fc_", ciStr = NULL, naStrs = c("NA", "n/d", "NaN", "<NA>"),
  qualColName = "qupm", normalize = TRUE,
  normReqs = tpptrDefaultNormReqs(), ggplotTheme = tppDefaultTheme(),
  nCores = "max", startPars = c(Pl = 0, a = 550, b = 10), splineDF = 4,
  maxAttempts = 500, plotCurves = TRUE, fixedReference = NULL,
  pValMethod = "robustZ", pValFilter = list(minR2 = 0.8, maxPlateau = 0.3),
  pValParams = list(binWidth = 300), verbose = FALSE, xlsxExport = TRUE)
```

### Arguments

`configTable` dataframe, or character object with the path to a file, that specifies important details of the TPP-TR experiment. See Section `details` for instructions how to create this object.

`data` single dataframe, or list of dataframes, containing fold change measurements and additional annotation columns to be imported. Can be used instead of specifying the file path in the `configTable` argument.

`resultPath` location where to store melting curve plots, intermediate results, and the final results table.

`methods` statistical methods for modeling melting behavior and detecting significant differences between experimental conditions. Ich more than one method are specified, results will be computed for each and concatenated in the result table (default: meltcurvefit).

`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.

`ciStr` character string indicating which columns contain confidence intervals for the fold change measurements. If specified, confidence intervals will be plotted around the melting curves.

`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`.

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

`normalize` perform normalization (default: TRUE).

`normReqs` list of filtering criteria for construction of the normalization set.

`ggplotTheme` ggplot theme for melting curve plots.

`nCores` either a numerical value given the desired number of CPUs, or 'max' to automatically assign the maximum possible number (default).

`startPars` start values for the melting curve parameters. Will be passed to function `nls` for curve fitting.

`splineDF` degrees of freedom for natural spline fitting.

`maxAttempts` maximal number of curve fitting attempts if model does not converge.

`plotCurves` boolean value indicating whether melting curves should be plotted. Deactivating plotting decreases runtime.

`fixedReference` name of a fixed reference experiment for normalization. If NULL (default), the experiment with the best R2 when fitting a melting curve through the median fold changes is chosen as the reference.

`pValMethod` Method for p-value computation. Currently restricted to 'robustZ' (see Cox & Mann (2008)).

`pValFilter` optional list of filtering criteria to be applied before p-value computation.

`pValParams` optional list of parameters for p-value computation.

`verbose` print name of each fitted protein to the command lin as a means of progress report.

`xlsxExport` boolean value indicating whether to produce result table in .xlsx format (requires package `openxlsx` and a zip application to be installed).

## Details

Invokes the following steps:

1. Import data using the `tpptrImport` function.

2. Perform normalization (optional) using the `tpptrNormalize` function. To perform normalization, set argument `normalize=TRUE`. The normalization will be filtered according to the criteria specified in the `normReqs` argument (also see the documentation of `tpptrNormalize` and `tpptrDefaultNormReqs` for further information).
3. Fit melting curves using the function `tpptrCurveFit`.
4. Produce result table using the function `tpptrAnalyzeMeltingCurves`.
5. Export results to Excel using the function `tppExport`.

The default settings are tailored towards the output of the python package isobarQuant, but can be customized to your own dataset by the arguments `idVar, fcStr, naStrs, qualColName`.

If `resultPath` is not specified, the location of the first input file specified in `configTable` will be used. If the input data are not specified in `configTable`, no result path will be set. This means that no output files or melting curve plots are produced and `analyzeTPPTR` just returns the results as a data frame.

The function `analyzeTPPTR` reports intermediate results to the command line. To suppress this, use `suppressMessages`.

The `configTable` argument is a dataframe, or the path to a spreadsheet (tab-delimited text-file or xlsx format). Information about each experiment is stored row-wise. It contains the following columns:

- `Path`:location of each datafile. Alternatively, data can be directly handed over by the `data` argument.
- `Experiment`: unique experiment names.
- `Condition`: experimental conditions of each dataset.
- Label columns: each isobaric label names a column that contains the temperatures administered for the label in the individual experiments.

The argument `methods` can be one of the following: More than one method can be specified. For example, parametric testing of melting points and nonparametric spline-based goodness-of-fit tests can be performed sequentially in the same analysis. The results are then written to separate columns of the output table.

If `methods` contains "meltcurvefit", melting curve plots will be stored in a subfolder with name `Melting_Curves` at the location specified by `resultPath`. If `methods` contains "splinefit", plots of the natural spline fits will be stored in a subfolder with name `Spline_Fits` at the location specified by `resultPath`.

The argument `nCores` could be either 'max' (use all available cores) or an upper limit of CPUs to be used.

If `doPlot = TRUE`, melting curve plots are generated separately for each protein and stored in separate pdfs. Each file is named by the unique protein identifier. Filenames are truncated to 255 characters (requirement by most operation systems). Truncated filenames are indicated by the suffix "_truncated[d]", where [d] is a unique number to avoid redundancies. All melting curve plots are stored in a subfolder with name `Melting_Curves` at the location specified by `resultPath`.

If the melting curve fitting procedure does not converge, it will be repeatedly started from perturbed starting parameters (maximum iterations defined by argument `maxAttempts`).

Argument `splineDF` specifies the degrees of freedom for natural spline fitting. As a single numeric value, it is directly passed on to the `splineDF` argument of `splines::ns`. Experience shows that `splineDF = 4` yields good results for TPP data sets with 10 temperature points. It is also possible to provide a numeric vector. In this case, splines are fitted for each entry and the optimal value is chosen per protein using Akaike's Information criterion.

### Value

A data frame in which the fit results are stored row-wise for each protein.

### References

Savitski, M. M., Reinhard, F. B., Franken, H., Werner, T., Savitski, M. F., Eberhard, D., ... & Drewes, G. (2014). Tracking cancer drugs in living cells by thermal profiling of the proteome. Science, 346(6205), 1255784.

### See Also

tppDefaultTheme, tpptrImport, tpptrNormalize, tpptrCurveFit, tpptrAnalyzeMeltingCurves

### Examples

```
data(hdacTR_smallExample)
tpptrResults <- analyzeTPPTR(configTable = hdacTR_config, data = hdacTR_data,
                methods = "splinefit", nCores = 1)
```

## hdacCCR_config *The configuration table to analyze hdacCCR_data.*

### Description

The configuration table to analyze hdacCCR_data.

### Details

`hdacCCR_config` is a data frame that specifies the experiment names, isobaric labels, and the administered drug concentrations at each label.

## hdacCCR_data *TPP-CCR example dataset (replicates 1 and 2)*

### Description

Example subset of a Panobinostat TPP-CCR dataset (replicates 1 and 2)

### Details

A list with two subsets of a dataset obtained by TPP-CCR experiments to investigate drug effects for HDAC inhibitor Panobinostat. It contains 7 HDACs as well as a random selection of 493 further proteins.

You can use this dataset to explore the `TPP` package functionalities without invoking the whole time consuming analysis on the big dataset.

The original dataset is located in the folder `'example_data/CCR_example_data'` in the package's installation directory. You can find it on your system by the `R` command `system.file('example_data', package = 'T`

## `hdacCCR_smallExample` *Example subsets of a Panobinostat TPP-CCR dataset (replicates 1 and 2) and the corresponding configuration table to start the analysis.*

### Description

Example dataset obtained by TPP-CCR experiments for analysis by the TPP-package. It contains all necessary arguments to start the analysis (config table and list of data frames).

## `hdacTR_config` *The configuration table to analyze hdacTR_data.*

### Description

The configuration table to analyze hdacTR_data.

### Details

`hdacTR_config` is a data frame that specifies the experiment name, isobaric labels, and the administered temperatures at each label.

## `hdacTR_data` *TPP-TR example dataset.*

### Description

Example subset of a dataset obtained by TPP-TR experiments to investigate possible targets for HDAC inhibitor Panobinostat.

### Details

`hdacTR_data` is a list of data frames that contain measurements for HDACs as well as a random selection of 500 further proteins.

You can use this dataset to explore the `TPP` package functionalities without invoking the whole time consuming analysis on the whole dataset.

The original dataset is located in the folder `'example_data/TR_example_data'` in the package's installation directory. You can find it on your system by the `R` command `system.file('example_data', package = 'T`

## `hdacTR_resultsTable_smallExample` *Example of a TPP-TR result table.*

### Description

Example of a TPP-TR result table.

### Details

Contains the data object `resultTable`.

## hdacTR_smallExample *Example subset of a Panobinostat TPP-TR dataset and the corresponding configuration table to start the analysis.*

### Description

Example dataset obtained by TPP-TR experiments for analysis by the TPP-package. It contains all necessary arguments to start the analysis (config table and list of data frames).

## panob2D_isobQuant_example *Example subsets of a Panobinostat 2D-TPP dataset and the corresponding configuration table to start the analysis.*

### Description

Example dataset obtained by 2D-TPP experiments for analysis by the TPP-package. It contains all necessary arguments to start the analysis (config table and list of data frames).

## panobinostat_2DTPP_config *The configuration table to analyze panobinostat_2DTPP_data.*

### Description

The configuration table to analyze panobinostat_2DTPP_data.

### Details

`panobinostat_2DTPP_config` is a data frame that specifies the experiment names, isobaric labels, and the administered drug concentrations at each label.

## panobinostat_2DTPP_data *2D-TPP-CCR example dataset*

### Description

Example subset of a Panobinostat 2D-TPP dataset

## Details

A list with two subsets of a dataset obtained by 2D-TPP experiments to investigate drug effects for HDAC inhibitor Panobinostat. The experiment was performed on living HepG2 cells (see Becher et al. (2016). Thermal profiling reveals phenylalanine hydroxylase as an off-target of panobinostat. Nature Chemical Biology, (September)) It contains 7 HDACs as well as a random selection of 493 further proteins.

You can use this dataset to explore the `TPP` package functionalities without invoking the whole time consuming analysis on the big dataset.

The original dataset was not produced by isobarQuant but by a custom-made software instead. For illustrative purposes, the column names have been adapted to the isobarQuant convention (id column = "gene_name", signal columns starting with = "signal_sum_", quantified spectrum matches = "qusm", quantified peptide matches = "qupm"). The original unmodified dataset in plain text format is still located in the folder `'example_data/2D_example_data'` in the package's installation directory. You can find it on your system by the `R` command `system.file('example_data', package = 'TPP')`.

---

`resultTable` *Example of a TPP-TR result table.*

---

## Description

Example of a TPP-TR result table.

## Details

`resultTable` is a data frame that contains the measurements of several TPP-TR experiments, the fitted melting curve parameters, as well as p-values and the results of additional quality checks for each protein. It can be used as input for the function `tppQCPlotsCorrelateExperiments`.

---

`TPP` *Thermal proteome profiling (TPP)*

---

## Description

*TPP* is a toolbox for analyzing thermal proteome profiling (TPP) experiments.

## Usage

```
.onLoad(libname, pkgname)
```

## Arguments

| | |
|---|---|
| `libname` | a character string giving the library directory where the package defining the namespace was found. Passed to .onLoad function. |
| `pkgname` | a character string giving the name of the package. Passed to .onLoad function. |

## Details

In order to start a TPP-TR analysis, use function `analyzeTPPTR`. For a TPP-CCR analysis, use function `analyzeTPPCCR`. See the vignette for detailed instructions.

### Value

No return value defined for this document.

### References

Savitski, M. M., Reinhard, F. B., Franken, H., Werner, T., Savitski, M. F., Eberhard, D., ... & Drewes, G. (2014). Tracking cancer drugs in living cells by thermal profiling of the proteome. Science, 346(6205), 1255784.

## tpp2dAddAdditionalInfo

*Add additional info to 2D-TPP CCR output data*

### Description

Adds additional info to 2D-TPP CCR output data, like counts on how often a certain protein was stabilized or destabilized

### Usage

```
tpp2dAddAdditionalInfo(data, idVar = "gene_name")
```

### Arguments

| | |
|---|---|
| `data` | output table returned by the `tpp2dCurveFit` function |
| `idVar` | character string indicating which column of the data table contains unique protein ids |

### Value

A data frame to which additional data like how often a protein has been (de-)stabilized has been attached

### Examples

```
load(system.file("example_data/2D_example_data/shortCCRresults.RData", package="TPP"))
shortCCRresults <- tpp2dAddAdditionalInfo(data = shortCCRresults, idVar="representative")
```

## tpp2dCalcFractAbundance

*Calculate fractional abundance and DMSO ratio of successive sumionareas (usage of function is only reasonable when at least two temperatures are multiplexed!)*

### Description

Calculates fractional abundance and DMSO ratio of successive sumionareas and creates respective columns which are added two the data frame which is handed over

### Usage

```
tpp2dCalcFractAbundance(configTable = NULL, data = NULL,
  intensityStr = "signal_sum_", idVar = "gene_name")
```

### Arguments

| | |
|---|---|
| `configTable` | data frame that specifies important details of the TPP-CCR experiment. |
| `data` | data frame of TPP-CCR results (e.g. obtained by `run2DTPPCCR`). |
| `intensityStr` | character string indicating which columns contain the sumionarea values. |
| `idVar` | character string indicating which data column provides the unique identifiers for each protein. |

### Value

Data frame that was handed over with additional columns of fractional abundance and DMSO1 vs DMSO2 ratio

### Examples

```
data("panob2D_isobQuant_example")
cfg <- panobinostat_2DTPP_config
datRaw <- panobinostat_2DTPP_data
data2d <- tpp2dImport(cfg, datRaw, fcStr = NULL)
data2dNew <- tpp2dCalcFractAbundance(cfg, data2d)
```

## tpp2dComputeFoldChanges

*Compute 2D-TPP fold changes*

### Description

Computes fold changes by calculating fold changes of the sumionarea relative to the reference column.

### Usage

```
tpp2dComputeFoldChanges(configTable = NULL, data = NULL,
  intensityStr = "signal_sum_", fcStr = "rel_fc_")
```

### Arguments

| | |
|---|---|
| configTable | data frame that specifies important details of the 2D-TPP experiment |
| data | dataframe that contain the data for the 2D-TPP experiment |
| intensityStr | character string indicating which columns contain the actual sumionarea values. Those column names containing the suffix `intensityStr` will be regarded as containing sumionarea values. |
| fcStr | character string indicating how columns that will contain the actual fold change values will be called. The suffix `fcStr` will be pasted in front of the names of the experiments. |

### Value

A data.frame with additional columns with constitute fold changes calculated with respect to the intensity values of the zero treatment column

### Examples

```
data("panob2D_isobQuant_example")
cfg <- panobinostat_2DTPP_config
datRaw <- panobinostat_2DTPP_data
data2d <- tpp2dImport(cfg, datRaw, fcStr = NULL)
fcData2d <- tpp2dComputeFoldChanges(cfg, data2d)
```

## tpp2dCreateCCRConfigFile

*Create TPP-CCR config files for 2D-TPP experiment*

### Description

Creates a list of config files for the TPP-CCR analysis so that the config files in the list matches the data.list generated by the 2D-TPP workflow.

### Usage

```
tpp2dCreateCCRConfigFile(configTable)
```

### Arguments

| | |
|---|---|
| configTable | data frame that specifies important details of the 2D-TPP experiment |

### Value

A config file of type data.frame which can be used for the `tpp2dCurveFit` function

## Examples

```
data("panob2D_isobQuant_example")
CCRconfig <- tpp2dCreateCCRConfigFile(configTable = panobinostat_2DTPP_config)
```

---

`tpp2dCreateDataFrameList` *Create data frame list for 2D-TPP experiment*

---

## Description

Creates a 2D-TPP data frame-list featuring a dataframe for each temperature analyzed in the experiment.

## Usage

```
tpp2dCreateDataFrameList(configTable = NULL, data = NULL,
  idVar = "gene_name", fcStr = NULL, addCol = NULL,
  intensityStr = "signal_sum_", qualColName = "qupm")
```

## Arguments

| | |
|---|---|
| `configTable` | data frame of experimental conditions. See Section `details` for instructions how to create this object. |
| `data` | can be either a list of data frames featuring a data frame with data for each experiment or can be NULL when filepaths for the respective experiments are indicated in the configTable |
| `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` | additional column names that specify columns in the input data that are to be attached to the data frame throughout the analysis |
| `intensityStr` | character string indicating which columns contain the sumionarea values. |
| `qualColName` | character string indicating which column can be used for additional quality criteria when deciding between different non-unique protein identifiers. |

## Details

Invokes the following steps:

1. either reads in temperature specific experimental data and creates data frame or extracts temperature specific data from pre- existing data frame list

## Value

A list of data frames in which the experimental data are stored row-wise for each protein.

## tpp2dCreateReport *Create Report of 2D-TPP analysis*

### Description

Creates a markdown pdf file that summarizes the 2D-TPP analysis by reporting e.g. R version and package versions used

### Usage

```
tpp2dCreateReport(data = NULL, configFile = NULL, resultPath = NULL,
  documentType = "html_document", configTable = NULL, normalize = TRUE,
  methods = c(""), idVar = "gene_name", fcStr = "rel_fc_",
  fcStrUpdated = "norm_rel_fc_", intensityStr = "signal_sum_",
  addCol = NULL, fcTolerance = NA, r2Cutoff = NA, fcCutoff = NA,
  slopeBounds = c(NA, NA), fTest = FALSE, trRef = "none")
```

### Arguments

| | |
|---|---|
| `data` | output data frame from an 2D-TPP analysis |
| `configFile` | character string containing a valid system path to a file which summarizes the experimental details of the 2D-TPP experiment or respective data frame |
| `resultPath` | character string containing a system path to where the report should be written |
| `documentType` | character string indicating which document type the report should have default: "html_document", alternatives: "pdf_document" |
| `configTable` | data frame summarizing the experimental details of the 2D-TPP experiment |
| `normalize` | boolean flag indicating whether median normalization has been performed |
| `methods` | vector of characters which indicate which methods have been used |
| `idVar` | unique protein identifier prefix |
| `fcStr` | fold change identifier prefix |
| `fcStrUpdated` | character string matching the fold change columns after normalization has been performed |
| `intensityStr` | intensity values prefix |
| `addCol` | vector of strings indicating which additional data columns were imported |
| `fcTolerance` | tolerance for the fcCutoff parameter |
| `r2Cutoff` | Quality criterion on dose response curve fit. |
| `fcCutoff` | Cutoff for highest compound concentration fold change |
| `slopeBounds` | Bounds on the slope parameter for dose response curve fitting |
| `fTest` | boolean variable stating whether an fTest was performed |
| `trRef` | character string containing a valid system path to a previously generated TPP-TR reference object |

### Value

A pdf or html report which summarizes all parameters that were set

---

`tpp2dCreateTPPTRreference` *Create TPP-TR reference for 2D-TPP experiment*

---

## Description

Performs a reference analysis of a TPP-TR experiment and generates boxplots for the distribution of fold changes at the different temperatures if desired.

## Usage

```
tpp2dCreateTPPTRreference(trConfigTable = NULL, resultPath = NULL,
  outputName = NULL, createFCboxplots = FALSE, idVar = "gene_name",
  fcStr = "rel_fc_", qualColName = "qupm", normalize = TRUE)
```

## Arguments

| | |
|---|---|
| `trConfigTable` | config file for a reference TR dataset |
| `resultPath` | character string containing a valid system path to which folder output files will be written |
| `outputName` | character string which will be used as name of the output folder |
| `createFCboxplots` | boolean flag indicating whether quality control boxplots are to be plotted |
| `idVar` | character string indicating which column of the data table contains the unique protein ids |
| `fcStr` | character string indicating which columns contain fold changes |
| `qualColName` | character string indicating which column contain protein identification quality measures |
| `normalize` | boolean argument stating whether the data should be normalized or not |

## Value

A TPP-TR reference object for a certain cell line with different supporting files in a desired output directory. The main object which is of interest for further analysis is the `trRefData.RData` file. This is the file to which a referencing system path has to be indicated when a function as `tpp2dSplineFitAndTest` require to input a TPP-TR reference object. The RData file consists of list carrying four different items:

1. tppCfgTable: the TPP-TR configtable which was used for generating this object
2. sumResTable a list of two elements 1. detail: the exact result data from the TR analysis and 2. summary. a summary of the analyzed TR data comprising the median and standard deviation values of the measurements at the different temperatures (encoded by the isobaric labels)
3. temperatures a table listing the temperatures which were used in the TR experiment in the different replicates
4. lblsByTemp a table matching each temperature to an isobaric label

## tpp2dCurveFit *Run TPP-CCR analysis for 2D-TPP experiment*

### Description

Performs analysis of a TPP-CCR experiment by invoking the routine for TPP-CCR curve fitting for each temperature of the sample.

### Usage

```
tpp2dCurveFit(configFile, data, nCores = 1, naStrs = c("NA", "n/d", "NaN",
  "<NA>"), fcStr = "norm_rel_fc_", idVar = "unique_ID",
  nonZeroCols = "qssm", r2Cutoff = 0.8, fcCutoff = 1.5,
  slopeBounds = c(1, 50), fcTolerance = 0.1)
```

### Arguments

| | |
|---|---|
| `configFile` | list of dataframes, that specifies important details of the 2D-TPP experiment for each temperature. |
| `data` | data frame that contains the data of the 2D-TPP experiment for each temperature. |
| `nCores` | numeric value stating how many cores are to be used for computation |
| `naStrs` | character vector indicating missing values in the data table. When reading data from file, this value will be passed on to the argument |
| `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. |
| `idVar` | character string indicating which data column provides the unique identifiers for each protein. |
| `nonZeroCols` | character string indicating a column that will be used for filtering out zero values. |
| `r2Cutoff` | Quality criterion on dose response curve fit. |
| `fcCutoff` | Cutoff for highest compound concentration fold change. |
| `slopeBounds` | Bounds on the slope parameter for dose response curve fitting. |
| `fcTolerance` | tolerance for the fcCutoff parameter. See details. |

### Value

A data frames in which the fit results are stored row-wise for each protein.

### Examples

```
data("panob2D_isobQuant_example")
cfg <- panobinostat_2DTPP_config
datRaw <- panobinostat_2DTPP_data
data2d <- tpp2dImport(cfg, datRaw, fcStr = NULL)
fcData2d <- tpp2dComputeFoldChanges(cfg, data2d)
normData2d <- tpp2dNormalize(cfg, fcData2d)
config_ccr <- tpp2dCreateCCRConfigFile(cfg)
ccr2dResults <- tpp2dCurveFit(config_ccr, normData2d)
```

## tpp2dEvalConfigTable *Evaluation of 2D-TPP Configuration File*

### Description

Evaluates whether the configuration file is handed over as data frame or as file path and loads the file path if necessary

### Usage

```
tpp2dEvalConfigTable(configTable)
```

### Arguments

| | |
|---|---|
| `configTable` | data frame or character object with the path to a file, that specifies important details of the 2D-TPP experiment. See Section `details` for instructions how to create this object |

### Value

A configtable that works with the 2D-TPP workflow

### Examples

```
data("panob2D_isobQuant_example")
configTable <- tpp2dEvalConfigTable(panobinostat_2DTPP_config)

configTable <- tpp2dEvalConfigTable(system.file("example_data/2D_example_data/hdac2D_config.txt",
                                                package = "TPP"))
```

## tpp2dExport *Produce Excel table of 2D-TPP experiment.*

### Description

Produce Excel table of 2D-TPP experiment analysis results.

### Usage

```
tpp2dExport(configTable = NULL, tab = NULL, resultPath = NULL,
  idVar = "gene_name", fcStr = "rel_fc_", intensityStr = "signal_sum_",
  addCol = NULL, normalizedData = FALSE, trRef = NULL)
```

### Arguments

| | |
|---|---|
| configTable | data frame that specifies important details of the 2D-TPP experiment |
| tab | Table with results of the 2D-TPP analysis. |
| resultPath | path for storing results table |
| idVar | character string indicating how the column that contains the unique protein identifiers is called |
| fcStr | character string indicating how columns that contain the actual fold change values are called |
| intensityStr | character string indicating how columns that contain the raw masspec signal (e.g. sumionareas) values are called |
| addCol | additional names of columns which are to be attached to the result table |
| normalizedData | boolean variable indicating whether the data has been normalized |
| trRef | character string containing a valid system path to a TPP-TR reference RData file |

### Value

Creates excel file of the TPP-CCR analysis of the 2D-TPP data.

### Examples

```
data("panob2D_isobQuant_example")
cfg <- panobinostat_2DTPP_config
datRaw <- panobinostat_2DTPP_data
data2d <- tpp2dImport(cfg, datRaw, fcStr = NULL)
fcData2d <- tpp2dComputeFoldChanges(cfg, data2d)
normData2d <- tpp2dNormalize(cfg, fcData2d)
config_ccr <- tpp2dCreateCCRConfigFile(cfg)
ccr2dResults <- tpp2dCurveFit(config_ccr, normData2d)
tpp2dExport(cfg, ccr2dResults, resultPath = getwd())
```

---

## tpp2dExportPlots *Export plots for 2D-TPP experiment.*

---

### Description

Exports plots into plots/ directory in the resultPath

### Usage

```
tpp2dExportPlots(plotList, resultPath, type = "none")
```

### Arguments

| | |
|---|---|
| plotList | list of ggplots returned from one of the plotting functions |
| resultPath | path for storing results |
| type | character string specifying which type of plot is to be exported |

### Details

Creates pdf files of the afore created plots by `plotSplines`, `tpp2dPlotCCRAllCurves`, `tpp2dPlotCCRGoodCurves`, `tpp2dPlotCCRSingleCurves`

### Value

None

## `tpp2dImport` *Import 2D-TPP data*

### Description

Imports data from 2D-TPP experiments by parsing a configTable and reading in corresponding data file or data frames containing raw data (sumionarea values) and creating a big data frame comprising all samples with respective fold changes

### Usage

```
tpp2dImport(configTable = NULL, data = NULL, idVar = "gene_name",
  addCol = NULL, intensityStr = "signal_sum_", qualColName = c("qupm",
  "qssm"), fcStr = "rel_fc_")
```

### Arguments

`configTable` dataframe, or character object with the path to a file, that specifies important details of the 2D-TPP experiment. See Section `details` for instructions how to create this object.

`data` single dataframe, containing raw measurements and if already available fold changes and additional annotation columns to be imported. Can be used instead of specifying the file path in the `configTable` argument.

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

`addCol` additional column names that specify columns in the input data that are to be attached to the data frame throughout the analysis

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

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

`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

A dataframe comprising all experimental data

### Examples

```
data("panob2D_isobQuant_example")
cfg <- panobinostat_2DTPP_config
datRaw <- panobinostat_2DTPP_data
data2d <- tpp2dImport(cfg, datRaw, fcStr = NULL)
```

## tpp2dMerge2dRef *Merge 2D-TPP result data with TPP-TR reference data*

### Description

Merges 2D-TPP result data with TPP-TR reference data to generate a big table including both results

### Usage

```
tpp2dMerge2dRef(data = NULL, trRef = NULL, idVar = "gene_name")
```

### Arguments

| | |
|---|---|
| `data` | dataframe containing the 2D-TPP results |
| `trRef` | character string of a valid system path to a TPP-TR reference RData object |
| `idVar` | character string matching the column containing the unique protein identifiers |

### Value

A data frame with results merged from 2D-TPP and TPP-TR reference

## tpp2dNormalize *Median normalization of protein fold changes of 2D-TPP data*

### Description

Normalizes fold changes retrieved from 2D-TPP experiment by dividing by the median fold change

### Usage

```
tpp2dNormalize(configTable, data, fcStr = "rel_fc_")
```

### Arguments

| | |
|---|---|
| `configTable` | data frame that specifies important details of the 2D-TPP experiment |
| `data` | data frame that contains the data for the 2D-TPP experiment |
| `fcStr` | character string indicating how columns that will contain the actual fold change values will be called. The suffix `fcStr` will be pasted in front of the names of the experiments. |

## Value

A dataframe identical to the input dataframe except that the columns containing the fold change values have been normalized by their median.

## Examples

```
data("panob2D_isobQuant_example")
cfg <- panobinostat_2DTPP_config
datRaw <- panobinostat_2DTPP_data
data2d <- tpp2dImport(cfg, datRaw, fcStr = NULL)
fcData2d <- tpp2dComputeFoldChanges(cfg, data2d)
normData2d <- tpp2dNormalize(cfg, fcData2d)
```

---

# `tpp2dPlotCCRAllCurves` *Plot all 2D-TPP CCR curves*

---

## Description

Generates a list of plots for all proteins with all curves for the different temperatures of for a certain 2D-TPP experiment.

## Usage

```
tpp2dPlotCCRAllCurves(configTable = NULL, data = NULL,
  idVar = "gene_name", fcStr = "rel_fc_", verbose = FALSE)
```

## Arguments

| | |
|---|---|
| `configTable` | data frame that specifies important details of the 2D-TPP experiment |
| `data` | data frame returned by the `tpp2dCurveFit` function |
| `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. |
| `verbose` | boolean variable stating whether a print description of problems/success for plotting of each protein should be printed |

## Value

A list of all dose-response plots that could be fitted

## Examples

```
data("panob2D_isobQuant_example")
cfg <- panobinostat_2DTPP_config
datRaw <- panobinostat_2DTPP_data
data2d <- tpp2dImport(cfg, datRaw, fcStr = NULL)
fcData2d <- tpp2dComputeFoldChanges(cfg, data2d)
normData2d <- tpp2dNormalize(cfg, fcData2d)
```

```
config_ccr <- tpp2dCreateCCRConfigFile(cfg)
ccr2dResults <- tpp2dCurveFit(config_ccr, normData2d)
allCurves <- tpp2dPlotCCRAllCurves(cfg, ccr2dResults,
                                     fcStr = "norm_rel_fc_")
allCurves[["HDAC2"]]
```

---

## tpp2dPlotCCRGoodCurves

*Plot 2D-TPP CCR curves meeting the requirements*

---

### Description

Generates a list of plots for all proteins with all good curves for the different temperatures of for a certain 2D-TPP experiment.

### Usage

```
tpp2dPlotCCRGoodCurves(configTable = NULL, data = NULL,
  idVar = "gene_name", fcStr = "rel_fc_", verbose = FALSE)
```

### Arguments

| | |
|---|---|
| configTable | data frame that specifies important details of the 2D-TPP experiment |
| data | data frame returned by the tpp2dCurveFit function |
| 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. |
| verbose | boolean variable stating whether a print description of problems/success for plotting of each protein should be printed |

### Value

A list of all dose-response plots that could be fitted and fulfilled the requested quality criteria

### Examples

```
data("panob2D_isobQuant_example")
cfg <- panobinostat_2DTPP_config
datRaw <- panobinostat_2DTPP_data
data2d <- tpp2dImport(cfg, datRaw, fcStr = NULL)
fcData2d <- tpp2dComputeFoldChanges(cfg, data2d)
normData2d <- tpp2dNormalize(cfg, fcData2d)
config_ccr <- tpp2dCreateCCRConfigFile(cfg)
ccr2dResults <- tpp2dCurveFit(config_ccr, normData2d)
allCurves <- tpp2dPlotCCRAllCurves(cfg, ccr2dResults,
                                   fcStr = "norm_rel_fc_")
allCurves[["HDAC2"]]
```

## tpp2dPlotCCRSingleCurves

*Plot single 2D-TPP CCR curves*

### Description

Generates a list of plots of all proteins for a certain 2D-TPP experiment

### Usage

```
tpp2dPlotCCRSingleCurves(configTable = NULL, data = NULL,
  idVar = "gene_name", fcStr = "rel_fc_", verbose = FALSE)
```

### Arguments

| | |
|---|---|
| configTable | data frame that specifies important details of the 2D-TPP experiment |
| data | output table returned by the `tpp2dCurveFit` function |
| 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. |
| verbose | boolean variable stating whether a print description of problems/success for plotting of each protein should be printed |

### Value

A list of all single dose-response plots that could be fitted and fulfilled the requested quality criteria

### Examples

```
data("panob2D_isobQuant_example")
cfg <- panobinostat_2DTPP_config
datRaw <- panobinostat_2DTPP_data
data2d <- tpp2dImport(cfg, datRaw, fcStr = NULL)
fcData2d <- tpp2dComputeFoldChanges(cfg, data2d)
normData2d <- tpp2dNormalize(cfg, fcData2d)
config_ccr <- tpp2dCreateCCRConfigFile(cfg)
ccr2dResults <- tpp2dCurveFit(config_ccr, normData2d)
singleCurves <- tpp2dPlotCCRSingleCurves(cfg, ccr2dResults,
                                   fcStr = "norm_rel_fc_")
singleCurves[["HDAC2"]][["54"]]
```

## `tpp2dPlotQChist` *Plot quality control histograms*

### Description

Plots quality control histograms of pEC50 values of reference dataset and indicates the pEC50 values of the 2D-TPP experiment

### Usage

```
tpp2dPlotQChist(configFile = NULL, resultTable = NULL, resultPath = NULL,
  trRef = NULL, fcStr = NULL, idVar = "gene_name", qualColName = "qupm")
```

### Arguments

| | |
|---|---|
| `configFile` | data frame or system path to table that specifies important details of the 2D-TPP experiment |
| `resultTable` | data.frame containing the results of a CCR analysis of 2D-TPP data |
| `resultPath` | character string containing a valid system path to which the the qc plots will be written |
| `trRef` | character string with a link to a TPP-TR reference object RData file |
| `fcStr` | character string indicating how columns that will contain the actual fold change values are called. |
| `idVar` | character string indicating name of the columns containing the unique protein identifiers |
| `qualColName` | character string indicating which column contain protein identification quality measures |

### Value

A pdf with various quality control plots for a specified 2D-TPP data set

## `tpp2dPlotQCpEC50` *Plot quality control pEC50 plots*

### Description

Plots quality control plots which indicate at which temperatures the pEC50 values of the treatment curves lie in comparison to those of the reference data

### Usage

```
tpp2dPlotQCpEC50(resultTable = NULL, resultPath = NULL, trRef = NULL,
  idVar = "gene_name")
```

### Arguments

| | |
|---|---|
| `resultTable` | data.frame containing the results of a CCR analysis of 2D-TPP data |
| `resultPath` | character string containing a valid system path to which the the qc plots will be written |
| `trRef` | character string with a link to a TPP-TR reference object RData file |
| `idVar` | character string indicating how the column that contains the unique protein identifiers is called |

### Value

A folder with plots for each identified protein that compare melting points in the reference data set with the 2D-TPP data set

## `tpp2dRemoveNAs` *Remove NAs in 2D-TPP data*

### Description

Removes NAs in the fold change columns of the 2D-TPP data

### Usage

```
tpp2dRemoveNAs(data.list)
```

### Arguments

| | |
|---|---|
| `data.list` | list of data frames that contain the data for the 2D-TPP experiment after computation of fold changes |

### Value

A dataframe without any NA values

## `tpp2dRemoveZeroSias` *Remove rows with zero sumionarea values*

### Description

Removes zero sumionarea values in a specified data.list so that no errors are generated in the following fold change computation step. A corresponding data.list with NAs instead of zeros is returned.

### Usage

```
tpp2dRemoveZeroSias(configTable, data.list, intensityStr = "signal_sum_")
```

### Arguments

| | |
|---|---|
| configTable | data frame that specifies important details of the 2D-TPP experiment. |
| data.list | list of data frames of corresponding experiment data |
| intensityStr | character string indicating which columns contain the sumionarea values. Those column names containing the suffix `intensityStr` will be regarded as containing sumionare values. |

### Value

A list of data frames with NAs instead of zeros.

## tpp2dReplaceColNames *Replace column names for 2D-TPP data*

### Description

Replaces column names for 2D-TPP data so that the TPP-CCR main function can deal with the data

### Usage

```
tpp2dReplaceColNames(configTable, data.list, intensityStr, fcStr)
```

### Arguments

| | |
|---|---|
| configTable | data frame that specifies important details of the 2D-TPP experiment |
| data.list | list of data frames that contain the data for the 2D-TPP experiment |
| intensityStr | character string indicating which columns contain the actual sumionarea values. Those column names containing the suffix `intensityStr` will be regarded as containing sumionarea values. |
| fcStr | character string indicating which columns contain the fold changes |

### Value

A list of dataframe with colnames which match concentrations instead of isobaric labels

## tpp2dSplineFitAndTest *Fit splines and perform f-Test*

### Description

Fit splines through TR reference dataset and extrapolates relative 2D-TPP datapoints, then compares spline fits of different treatments with non-treatment with an f-test

### Usage

```
tpp2dSplineFitAndTest(data_2D, trRefDataPath, idVar = "gene_name",
  fcStr = "norm_rel_fc_", refFcStr = "norm_rel_fc_", resultPath = NULL,
  ggplotTheme = tppDefaultTheme(), doPlot = TRUE, verbose = FALSE)
```

## Arguments

| | |
|---|---|
| `data_2D` | result data.frame from a 2D-TPP CCR analysis |
| `trRefDataPath` | character string with a link to a TPP-TR reference object RData file |
| `idVar` | character string indicating name of the columns containing the unique protein identifiers |
| `fcStr` | character string indicating how columns that will contain the actual fold change values will be called. The suffix `fcStr` will be pasted in front of the names of the experiments. |
| `refFcStr` | same as argument fcStr, but for the reference data. |
| `resultPath` | location where to store dose-response curve plots and results table. |
| `ggplotTheme` | ggplot theme for protein-wise plots. |
| `doPlot` | boolean value indicating whether protein-wise plots should be produced Deactivating plotting decreases runtime. |
| `verbose` | print description of problems for each protein for which splines fits could not be performed |

## Value

None

---

# `tpp2dSplinePlot` *Fit splines and generate ggplot visualizations*

---

## Description

Fit splines through TR reference dataset and extrapolates relative 2D-TPP datapoints, then compares spline fits of different treatments with non-treatment with an f-test

## Usage

```
tpp2dSplinePlot(data_2D = NULL, trData = NULL, fcStr = NULL,
  idVar = NULL, methods = c("doseResponse", "splineFit"),
  refFcStr = "norm_rel_fc_", verbose = FALSE)
```

## Arguments

| | |
|---|---|
| `data_2D` | result data.frame from a 2D-TPP CCR analysis |
| `trData` | character string with a link to a TPP-TR reference object RData file |
| `fcStr` | character string indicating how columns that will contain the actual fold change values will be called. The suffix `fcStr` will be pasted in front of the names of the experiments. |
| `idVar` | character string indicating name of the columns containing the unique protein identifiers |
| `methods` | vector of character strings that indicate which methods has been used for the previous analysis (default: c("doseResponse"), alternative: c("splineFit") or c("doseResponse", "splineFit")) |
| `refFcStr` | character string indicating how columns that will contain the fold change values in the reference data set |
| `verbose` | print description of problems for each protein for which splines fits could not be performed |

### Value

A list of ggplots which can be accessed via the unique protein ids in the idVar column

### Examples

```
load(system.file("example_data/2D_example_data/shortData2d.RData", package="TPP"))
trRef <- system.file("example_data/2D_example_data/referenceNormData.RData", package="TPP")
```

## tpp2dTRReferenceObject *TPP-TR reference object*

### Description

Definition of a TPP-TR reference object

### Usage

```
tpp2dTRReferenceObject(tppRefData = NULL, tppRefDataPath = NULL,
  fcStr = "norm_rel_fc_", qualColName = "qupm")
```

### Arguments

`tppRefData` TPP-TR reference object that can be directly passed to the function

`tppRefDataPath` character string containing a system path to a RData file containing an TPP-TR reference object

`fcStr` character string indicating which columns contain the fold changes

`qualColName` character string indicating which column contain protein identification quality measures

### Value

A TPP-TR reference object

## tppccrCurveFit *Fit dose response curves*

### Description

`tppccrCurveFit` fits logistic dose response curves to fold change measurements of a TPP-CCR experiment.

### Usage

```
tppccrCurveFit(data = NULL, fcTable = NULL, cpdEffects = NULL,
  slopeBounds = c(1, 50), nCores = "max", verbose = FALSE)
```

## Arguments

`data` list of expressionSet objects containing protein fold changes for dose response curve fitting.

`fcTable` optional long table with fold changes for each experiment. Can be provided instead of the input argument `data`.

`cpdEffects` optional long table of compound effects per protein and experiment. Can be provided instead of the input argument `data`.

`slopeBounds` bounds on the slope parameter for dose response curve fitting.

`nCores` either a numerical value given the desired number of CPUs, or 'max' to automatically assign the maximum possible number (default).

`verbose` print name of each fitted protein to the command line as a means of progress report.

## Details

`data` is a list of expressionSet objects created by `tppccrImport`. If desired, it can be already preprocessed by `tppccrNormalize` or `tppccrTransform`. It contains the isobaric labels and administered drug concentrations in the `phenoData` and user-defined protein properties in the `featureData`. Protein IDs are stored in the `featureNames`.

Measurements and compound effects for curve fitting can be provided by the arguments `fcTable` and `cpdEffects`, instead of being stored in expressionSets in `data`.

If specified, `fcTable` needs to be a long table with column names "id" (the protein names), "concentration" (the fold changes), "labelName" (the isobaric label to each measurement), and "experiment" (e.g. "Vehicle_1" or "Panobinostat_1").

If specified, `cpdEffects` needs to be a long table with column names "id" (the protein names), "cpdEff" (character vector of compound effects, may contain NAs), and "experiment" (e.g. "Vehicle_1" or "Panobinostat_1").

## Value

A list of expressionSet objects storing fold changes, the fitted curve parameters, as well as row and column metadata. In each expressionSet `S`, the fold changes can be accessed by `exprs(S)`. Protein expNames can be accessed by `featureNames(S)`. Isobaric labels and the corresponding concentrations are returned by `S$label` and `S$concentration`. The fitted curve parameters are stored in codefeatureData(S).

## See Also

`tppccrImport`, `tppccrNormalize`, `tppccrTransform`

## Examples

```
data(hdacCCR_smallExample)
tppccrData <- tppccrImport(configTable=hdacCCR_config,
                           data=hdacCCR_data)
tppccrNorm <- tppccrNormalize(data=tppccrData)
tppccrTransformed <- tppccrTransform(data=tppccrNorm)
tppccrFitted <- tppccrCurveFit(data=tppccrTransformed, nCores=1)
```

## tppccrImport *Import TPP-CCR dataset for analysis by the TPP package.*

### Description

tppccrImport imports a table of protein fold changes and stores them in an ExpressionSet for use in the TPP package.

### Usage

```
tppccrImport(configTable, data = NULL, idVar = "gene_name",
  fcStr = "rel_fc_", naStrs = c("NA", "n/d", "NaN", "<NA>"),
  qualColName = "qupm", nonZeroCols = "qssm")
```

### Arguments

| | |
|---|---|
| configTable | either a dataframe or the path to a spreadsheet. In both cases it specifies necessary information of the TPP-CCR experiment. |
| data | dataframe containing fold change measurements and additional annotation columns to be imported. Can be used instead of specifying the file path in configTable. |
| 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. |
| 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. |
| qualColName | character string indicating which column can be used for additional quality criteria when deciding between different non-unique protein identifiers. |
| nonZeroCols | character string indicating a column that will be used for filtering out zero values. |

### Details

The imported dataset has to contain measurements obtained by a TPP-CCR experiment. Fold changes need to be pre-computed using the lowest concentration as reference.

The dataset can be specified by filename in the configTable argument, or given directly in the data argument

The default settings are adjusted to analyze data of the python package isobarQuant. You can also customize them for your own dataset.

The configTable argument is a dataframe, or the path to a spreadsheet (tab-delimited text-file without quoted strings, or xlsx format). Information about each experiment is stored row-wise. It contains the following columns:

- Path: location of the datafile. Alternatively, data can be directly handed over by the data argument.
- Experiment: unique experiment name.

- Label columns: each isobaric label names a column that contains the concentration administered for the label in the individual experiments.

During data import, proteins with NAs in the data column specified by `idVar` receive unique generic IDs so that they can be processed by the package.

### Value

ExpressionSet object storing the measured fold changes, as well as row and column metadata. In each ExpressionSet `S`, the fold changes can be accessed by `exprs(S)`. Protein expNames can be accessed by `featureNames(S)`. Isobaric labels and the corresponding concentrations are returned by `S$label` and `S$concentration`.

### See Also

`tpptrImport`, `tppccrCurveFit`

### Examples

```
data(hdacCCR_smallExample)
tppccrData <- tppccrImport(configTable=hdacCCR_config,
data = hdacCCR_data)
```

## tppccrNormalize *Normalize data from TPP-CCR experiments*

### Description

Normalize each fold change column by its median.

### Usage

```
tppccrNormalize(data)
```

### Arguments

| | |
|---|---|
| `data` | list of expressionSets with measurements to be normalized |

### Value

List of expressionSet objects storing the normalized fold changes, as well as row and column metadata. In each expressionSet `S`, the fold changes can be accessed by `exprs(S)`. Protein names can be accessed by `featureNames(S)`. Isobaric labels and the corresponding concentrations are returned by `S$label` and `S$concentration`.

### Examples

```
data(hdacCCR_smallExample)
tppccrData <- tppccrImport(configTable=hdacCCR_config, data = hdacCCR_data)
tppccrNorm <- tppccrNormalize(data=tppccrData)
head(Biobase::exprs(tppccrNorm[[1]]))
```

## tppccrNormalizeToReference

*Normalize fold changes of TPP-CCR experiment to a reference column*

### Description

Normalize fold changes of TPP-CCR experiment to a reference column (usually that with the lowest concentration) to ensure that the transformation by tppccrTransform yields values between 0 and 1.

### Usage

```
tppccrNormalizeToReference(data, refCol = NULL)
```

### Arguments

| | |
|---|---|
| `data` | expressionSet object containing the data to be normalized |
| `refCol` | column number to use as a reference. Will contain only 1s after the normalization. |

### Value

List of expressionSet objects storing the normalized fold changes, as well as row and column metadata. In each expressionSet `S`, the fold changes can be accessed by `exprs(S)`. Protein expNames can be accessed by `featureNames(S)`. Isobaric labels and the corresponding concentrations are returned by `S$label` and `S$concentration`.

### Examples

```
data(hdacCCR_smallExample)
tppccrData <- tppccrImport(configTable=hdacCCR_config, data = hdacCCR_data)
tppccrNorm <- tppccrNormalize(data=tppccrData)
# Normalize to lowest concentration (in the first column):
tppccrNormToRef <- tppccrNormalizeToReference(data=tppccrNorm, refCol=1)
# Obtain results per replicate:
refTransf_replicate1 <- tppccrNormToRef$Panobinostat_1
head(Biobase::exprs(refTransf_replicate1))
# Perform transformation:
tppccrTransformed <- tppccrTransform(data=tppccrNormToRef)
# Obtain transformed measurements per replicate:
transf_replicate1 <- tppccrTransformed$Panobinostat_1
transf_replicate2 <- tppccrTransformed$Panobinostat_2
# Inspect transformed data in replicate 1:
effects_replicate1 <- Biobase::featureData(transf_replicate1)$compound_effect
newData_repl1 <- data.frame(Biobase::exprs(transf_replicate1),
                              Type=effects_replicate1)[!is.na(effects_replicate1),]
```

## tppccrPlotCurves *Plot dose response curves*

### Description

tppccrPlotCurves plots the logistic dose response curves, as well as the underlying fold change measurements for each TPP-CCR experiment in a study.

### Usage

```
tppccrPlotCurves(data = NULL, fcTable = NULL, curvePars = NULL,
  resultPath = NULL, ggplotTheme = tppDefaultTheme(), nCores = "max",
  verbose = FALSE)
```

### Arguments

| | |
|---|---|
| data | list of expressionSet objects containing protein fold changes, as well as fitted curve parameters. |
| fcTable | optional long table with fold changes for each experiment. Can be provided instead of the input argument data. |
| curvePars | optional long table of curve parameters per protein and experiment. Can be provided instead of the input argument data. |
| resultPath | location where to store dose-response curve plots. |
| ggplotTheme | ggplot theme for dose response curve plots. |
| nCores | either a numerical value given the desired number of CPUs, or 'max' to automatically assign the maximum possible number (default). |
| verbose | print name of each plotted protein to the command line as a means of progress report. |

### Details

data is a list of expressionSet objects created by tppccrCurveFit. It contains the isobaric labels and administered drug concentrations in the phenoData and user-defined protein properties (including dose response curve parameters) in the featureData. Protein IDs are stored in the featureNames.

Measurements and compound effects for curve fitting can be provided by the arguments fcTable and cpdEffects, instead of being stored in expressionSets in data.

If specified, fcTable needs to be a long table with column names "id" (the protein names), "concentration" (the fold changes), "labelName" (the isobaric label to each measurement), and "experiment" (e.g. "Vehicle_1" or "Panobinostat_1").

If specified, curvePars needs to be a long table with column names "id" (the protein names), "param" (curve parameter per protein and experiment, see TPP:::drCurveParamNames(names=TRUE, info=FALSE) for possibilities), and "experiment" (e.g. "Vehicle_1" or "Panobinostat_1").

The dose response curve plots will be stored in a subfolder with name DoseResponse_Curves at the location specified by resultPath.

## Value

A list of expressionSet objects storing fold changes, as well as row and column metadata. In each expressionSet S, the fold changes can be accessed by exprs(S). Protein expNames can be accessed by featureNames(S). Isobaric labels and the corresponding concentrations are returned by S$label and S$concentration. Paths to the produced plots are stored in codefeatureData(S)$plot.

## See Also

tppccrCurveFit,tppDefaultTheme

## Examples

```
data(hdacCCR_smallExample)
tppccrData <- tppccrImport(configTable=hdacCCR_config,
                           data=hdacCCR_data)
tppccrNorm <- tppccrNormalize(data=tppccrData)
tppccrTransformed <- tppccrTransform(data=tppccrNorm)
tppccrFitted <- tppccrCurveFit(data=tppccrTransformed, nCores=1)
hdacSubset <- sapply(tppccrFitted, function(d)d[grepl("HDAC", rownames(d)),])
tppccrPlotted <- tppccrPlotCurves(hdacSubset, resultPath=getwd(), nCores = 1)
```

---

# tppccrResultTable *Summarize results of a TPP-CCR study*

---

## Description

tppccrResultTable summarizes the outcomes of a TPP-CCR study in a results table and includes quality information about the estimated dose response curves.

## Usage

```
tppccrResultTable(data, r2Cutoff = 0.8)
```

## Arguments

| | |
|---|---|
| data | list of expressionSet objects containing protein fold changes, as well as fitted curve parameters. |
| r2Cutoff | quality criterion on dose response curve fit.<br>@details data is a list of expressionSet objects created by tppccrCurveFit or tppccrPlotCurves. It contains the isobaric labels and administered drug concentrations in the phenoData and user-defined protein properties (including dose response curve parameters) in the featureData. Protein IDs are stored in the featureNames.<br>If data is the output of tppccrPlotCurves, plot locations are given in the plot column of the featureData. |

## Value

A data frame in which the results are stored row-wise for each protein, together with the original annotation from the input files.

## See Also

tppccrCurveFit,tppccrPlotCurves

## Examples

```
data(hdacCCR_smallExample)
tppccrData <- tppccrImport(configTable=hdacCCR_config,
                           data=hdacCCR_data)
tppccrNorm <- tppccrNormalize(data=tppccrData)
tppccrTransformed <- tppccrTransform(data=tppccrNorm)
tppccrFitted <- tppccrCurveFit(data=tppccrTransformed, nCores=1)
tppccrResults <- tppccrResultTable(data=tppccrFitted)
subset(tppccrResults, passed_filter_Panobinostat_1 & passed_filter_Panobinostat_2)
```

---

# tppccrTransform *Transform fold changes of TPP-CCR experiment*

---

## Description

Transform fold changes of TPP-CCR experiment to prepare them for dose response curve fitting.

## Usage

```
tppccrTransform(data, fcCutoff = 1.5, fcTolerance = 0.1)
```

## Arguments

| | |
|---|---|
| data | expressionSet object containing the data to be transformed. |
| fcCutoff | cutoff for highest compound concentration fold change. |
| fcTolerance | tolerance for the fcCutoff parameter. See details. |

## Details

Only proteins with fold changes bigger than [fcCutoff * (1 - fcTolerance) or smaller than 1/(fcCutoff * (1 - fcTolerance))] will be used for curve fitting. Additionally, the proteins fulfilling the fcCutoff criterion without tolerance will be marked in the output column meets_FC_requirement.

## Value

List of expressionSet objects storing the transformed fold changes, as well as row and column meta-data. In each expressionSet S, the fold changes can be accessed by exprs(S). Protein expNames can be accessed by featureNames(S). Isobaric labels and the corresponding concentrations are returned by S$label and S$concentration.

## Examples

```
data(hdacCCR_smallExample)
tppccrData <- tppccrImport(configTable=hdacCCR_config, data = hdacCCR_data)
tppccrNorm <- tppccrNormalize(data=tppccrData)
# Perform transformation:
tppccrTransformed <- tppccrTransform(data=tppccrNorm)
# Obtain transformed measurements per replicate:
transf_replicate1 <- tppccrTransformed$Panobinostat_1
transf_replicate2 <- tppccrTransformed$Panobinostat_2
# Inspect transformed data in replicate 1:
effects_replicate1 <- Biobase::featureData(transf_replicate1)$compound_effect
newData_repl1 <- data.frame(Biobase::exprs(transf_replicate1),
                              Type=effects_replicate1)[!is.na(effects_replicate1),]
```

# `tppDefaultTheme` *Default ggplot theme for melting curve plots.*

## Description

Default theme to be passed to the gplots produced by the TPP package.

## Usage

```
tppDefaultTheme()
```

## Details

Internally, the theme is used as an argument for the function `ggplot2::theme_set` in order specify the appearance of the melting curve plots.

The specified plot properties include bold font and increased font size for axis labels and title, as well as a 90 degree angle for y axis labels.

## Value

ggplot theme with default settings for melting plot appearance.

## Examples

```
# Import data:
data(hdacTR_smallExample)
tpptrData <- tpptrImport(configTable=hdacTR_config, data=hdacTR_data)
# Obtain template with default settings:
normRequirements <- tpptrDefaultNormReqs()
print(normRequirements)
# Relax filter on the 10th fold change column for
# normalization set production:
normRequirements$fcRequirements[3,3] <- 0.25
# Perform normalization:
tpptrNorm <- tpptrNormalize(data=tpptrData, normReqs=)
```

## tppExport *Produce Excel table of TPP-TR or TPP-CCR experiment.*

### Description

Produce Excel table of TPP-TR or TPP-CCR experiment out of the data frame returned by tpptrAnalyzeMeltingCurves

### Usage

```
tppExport(tab, file, expNames = NULL, expColors = NULL)
```

### Arguments

tab Table with results of the TPP analysis.

file path for storing results table

expNames character vector of experiment names of the same length as expColors.

expColors character vector of background colors to group the result columns belonging to different experiments.

### Value

No value returned.

### Examples

```
data(hdacTR_resultsTable_smallExample)
tppExport(resultTable, "tpptr_example_results.xlsx")
```

## tppQCPlotsCorrelateExperiments *Visually compare fold changes of different TPP experiments.*

### Description

Plot pairwise relationships between the proteins in different TPP experiments.

### Usage

```
tppQCPlotsCorrelateExperiments(tppData, annotStr = "", path = NULL,
  ggplotTheme = tppDefaultTheme())
```

### Arguments

tppData List of expressionSets with data to be plotted.

annotStr String with additional information to be added to the plot.

path Location where to store resulting plot.

ggplotTheme ggplot theme for the created plots.

### Value

List of plots for each experiment.

### See Also

tppDefaultTheme

### Examples

```
data(hdacTR_smallExample)
tpptrData <- tpptrImport(configTable=hdacTR_config, data=hdacTR_data)
# Quality control (QC) plots BEFORE normalization:
tppQCPlotsCorrelateExperiments(tppData=tpptrData,
annotStr="Non-normalized Fold Changes")
# Quality control (QC) plots AFTER normalization:
tpptrNorm <- tpptrNormalize(data=tpptrData, normReqs=tpptrDefaultNormReqs())
tpptrDataNormalized <- tpptrNorm$normData
tppQCPlotsCorrelateExperiments(tppData=tpptrDataNormalized,
annotStr="Normalized Fold Changes")
```

---

## tpptrAnalyzeMeltingCurves

*Analyze fitted curve parameters to detect significant shifts in melting points.*

---

### Description

Compute p-values for the pairwise comparisons of melting curve shifts between different conditions.

### Usage

```
tpptrAnalyzeMeltingCurves(data, pValMethod = "robustZ",
  pValFilter = list(minR2 = 0.8, maxPlateau = 0.3),
  pValParams = list(binWidth = 300))
```

### Arguments

| | |
|---|---|
| data | list of ExpressionSets containing fold changes and metadata. Their featureData fields contain the fitted melting curve parameters. |
| pValMethod | Method for p-value computation. Currently restricted to 'robustZ' (see Cox & Mann (2008)). |
| pValFilter | optional list of filtering criteria to be applied before p-value computation. |
| pValParams | optional list of parameters for p-value computation. |

### Details

The pValParams argument is a list that can contain optional parameters for the chosen p-value computation pValMethod. The following options are available:

1. pValMethod = "robustZ":
   pValParams=list(binWidth=[your_binWidth]).

## Value

A data frame in which the fit results are stored row-wise for each protein.

## References

Cox, J., & Mann, M. (2008). MaxQuant enables high peptide identification rates, individualized ppb-range mass accuracies and proteome-wide protein quantification. Nature biotechnology, 26(12), 1367-1372.

## Examples

```
data(hdacTR_smallExample)
tpptrData <- tpptrImport(hdacTR_config, hdacTR_data)
tpptrNorm <- tpptrNormalize(data=tpptrData,
                            normReqs=tpptrDefaultNormReqs())
normalizedData <- tpptrNorm$normData
## Not run:
# Fit melting curves to each protein
# (can take some time depending on device used):
 fittedData <- tpptrCurveFit(normalizedData, nCores=1)
 resultTable <- tpptrAnalyzeMeltingCurves(fittedData)
 subset(resultTable, fulfills_all_4_requirements)$Protein_ID

## End(Not run)
```

---

# `tpptrCurveFit` *Fit melting curves to all proteins in a dataset.*

---

## Description

Fit melting curves to all proteins in a dataset.

## Usage

```
tpptrCurveFit(data, dataCI = NULL, resultPath = NULL,
  ggplotTheme = tppDefaultTheme(), doPlot = TRUE, startPars = c(Pl = 0, a
  = 550, b = 10), maxAttempts = 500, nCores = "max", verbose = FALSE)
```

## Arguments

| | |
|---|---|
| `data` | list of `ExpressionSets` with protein fold changes for curve fitting. |
| `dataCI` | list of `ExpressionSets` with protein fold change confidence intervals for curve fitting. Default to NULL. |
| `resultPath` | location where to store the melting curve plots. |
| `ggplotTheme` | ggplot theme for melting curve plots. |
| `doPlot` | boolean value indicating whether melting curves should be plotted, or whether just the curve parameters should be returned. |
| `startPars` | start values for the melting curve parameters. Will be passed to function `nls` for curve fitting. |

| | |
|---|---|
| maxAttempts | maximal number of curve fitting attempts if model does not converge. |
| nCores | either a numerical value given the desired number of CPUs, or 'max' to automatically assign the maximum possible number (default). |
| verbose | plot name of each fitted protein to the command lin as a means of progress report. |

## Details

If the melting curve fitting procedure does not converge, it will be repeatedly started from perturbed starting parameters (maximum iterations defined by argument `maxAttempts`)

If `doPlot = TRUE`, melting curves are be plotted in individual files per protein. Each file is named by its unique identifier. Filenames are truncated to 255 characters (requirement by most operation systems). Truncated filenames are indicated by the suffix "_truncated[d]", where [d] is a unique number to avoid redundancies.

The melting curve plots will be stored in a subfolder with name `Melting_Curves` at the location specified by `resultPath`.

## Value

A list of ExpressionSets storing the data together with the melting curve parameters for each experiment. Each ExpressionSet contains the measured fold changes, as well as row and column metadata. In each ExpressionSet `S`, the fold changes can be accessed by `exprs(S)`. Protein expNames can be accessed by `featureNames(S)`. Isobaric labels and the corresponding temperatures are returned by `S$label` and `S$temperature`.

## See Also

tppDefaultTheme

## Examples

```
data(hdacTR_smallExample)
tpptrData <- tpptrImport(configTable=hdacTR_config, data=hdacTR_data)
tpptrNorm <- tpptrNormalize(data=tpptrData, normReqs=tpptrDefaultNormReqs())
normalizedData <- tpptrNorm$normData
hdacSubsets <- lapply(normalizedData,
                      function(d) d[grepl("HDAC", Biobase::featureNames(d))])
tpptrFittedHDACs <- tpptrCurveFit(hdacSubsets, nCores=1)
# Show estimated parameters for vehicle and treatment experiments:
Biobase::pData(Biobase::featureData(tpptrFittedHDACs[["Vehicle_1"]]))
Biobase::pData(Biobase::featureData(tpptrFittedHDACs[["Panobinostat_1"]]))
```

# tpptrDefaultNormReqs *Default filter criteria for fold change normalization*

## Description

Filter criteria as described in the publication.

### Usage

```
tpptrDefaultNormReqs()
```

### Value

List with two entries: ’fcRequirements’ describes filtering requirements on fold change columns, ’otherRequirements’ contains criteria on additional metadata columns.

### Examples

```
data(hdacTR_smallExample)
tpptrData <- tpptrImport(configTable=hdacTR_config, data=hdacTR_data)
tpptrNorm <- tpptrNormalize(data=tpptrData, normReqs=tpptrDefaultNormReqs())
```

---

## `tpptrFitSplines` *Perform spline fitting*

---

### Description

Fit natural splines to all proteins in a dataset.

### Usage

```
tpptrFitSplines(data, factorsH1, factorsH0 = c(), splineDF = 4,
  computeAUC = FALSE, returnModels = TRUE)
```

### Arguments

| | |
|---|---|
| `data` | the data to be fitted |
| `factorsH1` | which factors should be included in the alternative model? |
| `factorsH0` | which factors should be included in the null model? |
| `splineDF` | degrees of freedom for natural spline fitting. |
| `computeAUC` | should areas under the spline curves be computed? Activation increases runtime requirements. |
| `returnModels` | should the linear models be returned in a column of the result table? Activation increases memory requirements. |

Argument `splineDF` specifies the degrees of freedom for natural spline fitting. As a single numeric value, it is directly passed on to the `splineDF` argument of `splines::ns`. Experience shows that `splineDF = 4` yields good results for TPP data sets with 10 temperature points. It is also possible to provide a numeric vector. In this case, splines are fitted for each entry and the optimal value is chosen per protein using Akaike’s Information criterion.

### Value

A table containing the fitted models per protein

### See Also

`ns`, `AICc`

## Examples

```
data(hdacTR_smallExample)
tpptrData <- tpptrImport(configTable = hdacTR_config, data = hdacTR_data)
normResults <- tpptrNormalize(data = tpptrData,
                normReqs = tpptrDefaultNormReqs())
normData_eSets <- normResults$normData
normData_longTable <- tpptrTidyUpESets(normData_eSets)$proteinMeasurements
hdacSubset <- subset(normData_longTable, grepl("HDAC", uniqueID))
hdacSplineFits <- tpptrFitSplines(data = hdacSubset,
                                  factorsH1 = c("condition"))
```

# tpptrFTest *Analyze spline fits to detect differential behavior over time*

## Description

Analyze fitted natural spline models and look for differential behavior between conditions by a moderated F-test.

## Usage

```
tpptrFTest(fittedModels, doPlot = FALSE, resultPath = NULL)
```

## Arguments

| | |
|---|---|
| fittedModels | a table of fitted spline models (produced by tpptrFitSplines). |
| doPlot | boolean value indicating whether QC plots should be produced. Currently, QC plots comprise distributions of the F statistics, and the p-values before/ after Benjamini Hochberg adjustment. |
| resultPath | location where to store QC plots, if doPlot = TRUE. |

## Details

If doPlot is TRUE, but no resultPath is specified, the plots will be prompted to the active device.

The moderated F-statistic is calculated by the following equation: ...

## Value

A long table containing the hypothesis test results per protein.

## See Also

ns, squeezeVar

## Examples

```
data(hdacTR_smallExample)
tpptrData <- tpptrImport(configTable = hdacTR_config, data = hdacTR_data)
normResults <- tpptrNormalize(data = tpptrData, normReqs = tpptrDefaultNormReqs())
normData_eSets <- normResults$normData
longTables <- normData_eSets %>% tpptrTidyUpESets
fitData <- longTables %>% extract2("proteinMeasurements")
fits <- tpptrFitSplines(data = fitData, factorsH1 = "condition")
testResults <- tpptrFTest(fittedModels = fits)
```

---

# tpptrImport *Import TPP-TR datasets for analysis by the TPP package.*

---

## Description

`tpptrImport` imports several tables of protein fold changes and stores them in a list of ExpressionSets for use in the TPP package.

## Usage

```
tpptrImport(configTable, data = NULL, idVar = "gene_name",
  fcStr = "rel_fc_", naStrs = c("NA", "n/d", "NaN"), qualColName = "qupm",
  outputFormat = "eSetList")
```

## Arguments

`configTable` either a dataframe or the path to a spreadsheet. In both cases it specifies necessary information of the TPP-CCR experiment.

`data` single dataframe, or list of dataframes, containing fold change measurements and additional annotation columns to be imported. Can be used instead of specifying the file path in `configTable`.

`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.

`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`.

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

`outputFormat` output format. Either "eSetList" to obtain output in the same way as previously (will be deprecated soon), or "tidy" to obtain a

## Details

The imported datasets have to contain measurements obtained by TPP-TR experiments. Fold changes need to be pre-computed using the lowest temperature as reference.

An arbitrary number of datasets can be specified by filename in the `Path`-column of the `configTable` argument, or given directly as a list of dataframes in the `data` argument. They can differ, for example, by biological replicate or by experimental condition (for example, treatment versus vehicle). Their names are defined uniquely by the `Experiment` column in `configTable`. Experimental conditions can be specified by an optional column in `configTable`.

The default settings are adjusted to analyze data of the python package `isobarQuant`. You can also customize them for your own dataset.

The `configTable` argument is a dataframe, or the path to a spreadsheet (tab-delimited text-file without quoted strings, or xlsx format). Information about each experiment is stored row-wise. It contains the following columns:

- `Path`:location of each datafile. Alternatively, data can be directly handed over by the `data` argument.
- `Experiment`: unique experiment names.
- `Condition`: experimental conditions of each dataset.
- Label columns: each isobaric label names a column that contains the temperatures administered for the label in the individual experiments.

Proteins with NAs in the data column specified by `idVar` receive unique generic IDs so that they can be processed by the package.

## Value

A list of ExpressionSets storing the imported data for experiment. Each ExpressionSet contains the measured fold changes, as well as row and column metadata. In each ExpressionSet `S`, the fold changes can be accessed by `exprs(S)`. Protein expNames can be accessed by `featureNames(S)`. Isobaric labels and the corresponding temperatures are returned by `S$label` and `S$temperature`

## See Also

`tppccrImport`

## Examples

```
data(hdacTR_smallExample)
tpptrData <- tpptrImport(hdacTR_config, hdacTR_data)
```

# `tpptrNormalize` *Normalize protein fold changes*

## Description

Normalizes fold changes determined by TPP-TR experiments over different experimental groups.

### Usage

```
tpptrNormalize(data, normReqs = tpptrDefaultNormReqs(),
  qcPlotTheme = tppDefaultTheme(), qcPlotPath = NULL, startPars = c(Pl =
  0, a = 550, b = 10), maxAttempts = 1, fixedReference = NULL)
```

### Arguments

| | |
|---|---|
| data | List of ExpressionSets with protein fold changes to be normalized. |
| normReqs | List of filtering criteria for construction of the normalization set. |
| qcPlotTheme | ggplot theme for the created plots |
| qcPlotPath | location where plots of the curves fitted to the normalization set medians should be stored. |
| startPars | start values for the melting curve parameters. Will be passed to function nls for curve fitting. |
| maxAttempts | maximal number of curve attempts to fit melting curve to fold change medians when computing normalization factors. |
| fixedReference | name of a fixed reference experiment for normalization. If NULL (default), the experiment with the best R2 when fitting a melting curve through the median fold changes is chosen as the reference. |

### Details

Performs normalization of all fold changes in a given list of ExpressionSets. The normalization procedure is described in detail in Savitski et al. (2014). Whether normalization needs to be performed and what method is best suited depends on the experiment. Here we provide a reasonable solution for the data at hand.

We distinguish between filtering conditions on fold changes and on additional annotation columns. Correspondingly, normReqs contains two fields, fcFilters and otherFilters. Each entry contains a data frame with three columns specifying the column to be filtered, as well as upper and lower bounds. An example is given by tpptrDefaultNormReqs.

### Value

A list of ExpressionSets storing the normalized data for each experiment. Each ExpressionSet contains the measured fold changes, as well as row and column metadata. In each ExpressionSet S, the fold changes can be accessed by exprs(S). Protein expNames can be accessed by featureNames(S). Isobaric labels and the corresponding temperatures are returned by S$label and S$temperature

### References

Savitski, M. M. and Reinhard, F. BM. and Franken, H. and Werner, T. and Savitski, M. F. and Eberhard, D. and Molina, D. M. and Jafari, R. and Dovega, R. B. and Klaeger, S. and others (2014) Tracking cancer drugs in living cells by thermal profiling of the proteome. Science 346(6205), p. 1255784.

### See Also

tpptrImport

## Examples

```
data(hdacTR_smallExample)
tpptrData <- tpptrImport(hdacTR_config, hdacTR_data)
tpptrNorm <- tpptrNormalize(data=tpptrData, normReqs=tpptrDefaultNormReqs())
names(tpptrNorm)
```

---

# tpptrPlotSplines *Plot spline fits per protein*

---

## Description

Plot spline fits per protein

## Usage

```
tpptrPlotSplines(data, factorsH1, factorsH0 = c(), fittedModels, testResults,
  resultPath, ggplotTheme = tppDefaultTheme(), plotAlphabetical = FALSE,
  plotRanked = TRUE, maxRank = 500)
```

## Arguments

| | |
|---|---|
| data | the data to be plotted. |
| factorsH1 | which factors were included in the alternative model? (necessary for correct prediction) |
| factorsH0 | which factors were included in the null model? (necessary for correct prediction) |
| fittedModels | long table of fitted models. Output of tpptrFitSplines. |
| testResults | long table of p-values per protein. Output of tpptrFTest. |
| resultPath | location where to store the spline plots per protein. |
| ggplotTheme | ggplot theme for melting curve plots. |
| plotAlphabetical | Generate a summary pdf with 20 plots per page in alphabetical order? |
| plotRanked | Generate a summary pdf with 20 plots per page, ordered by p-value? |
| maxRank | if plotRanked = TRUE, how many of the top hits should be plotted (default: 500)? |

## Details

Plots of the natural spline fits will be stored in a subfolder with name Spline_Fits at the location specified by resultPath.

## Value

None

## See Also

ns, AICc, tppDefaultTheme,tpptrFitSplines, tpptrFTest

### Examples

```
data(hdacTR_smallExample)
tpptrData <- tpptrImport(configTable = hdacTR_config, data = hdacTR_data)
normResults <- tpptrNormalize(data = tpptrData, normReqs = tpptrDefaultNormReqs())
normData_eSets <- normResults$normData
normData_longTable <- tpptrTidyUpESets(normData_eSets)$proteinMeasurements
splineFits <- tpptrFitSplines(data = normData_longTable,
                              factorsH1 = "condition", returnModels = TRUE)
testResults <- tpptrFTest(fittedModels = splineFits, doPlot = FALSE)
tpptrPlotSplines(data = normData_longTable, fittedModels = splineFits,
                 factorsH1 = "condition", factorsH0 = c(),
                 testResults = testResults, resultPath = getwd(),
                 plotRanked = TRUE, maxRank = 20)
```

---

## tpptrSplineFitAndTest *Perform spline fitting and analyze by moderated F-test*

---

### Description

A wrapper function around the functions tpptrFitSplines, tpptrFTest, tpptrPlotSplines, which fits natural splines to all proteins in a dataset and detect differential behavior between conditions by a moderated F-test. The results are formatted as a wide table with one row per protein. This table contains all the original data, the test results, and (optionally) additional annotation columns for each protein.

### Usage

```
tpptrSplineFitAndTest(data, factorsH1, factorsH0 = c(), resultPath = NULL,
  ggplotTheme = tppDefaultTheme(), doPlot = TRUE, nCores = "max",
  splineDF = 4, additionalCols = NULL, verbose = FALSE)
```

### Arguments

| | |
|---|---|
| data | the data to be fitted. |
| factorsH1 | which factors should be included in the alternative model? |
| factorsH0 | which factors should be included in the null model? |
| resultPath | location where to store the spline plots per protein. |
| ggplotTheme | ggplot theme for melting curve plots. |
| doPlot | boolean value indicating whether melting curves should be plotted, or whether just the curve parameters should be returned. |
| nCores | either a numerical value given the desired number of CPUs, or 'max' to automatically assign the maximum possible number (default). |
| splineDF | degrees of freedom for natural spline fitting. |
| additionalCols | additional annotation per protein to append to the result table. |
| verbose | plot name of each fitted protein to the command lin as a means of progress report. |

## Details

Plots of the natural spline fits will be stored in a subfolder with name `Spline_Fits` at the location specified by `resultPath`.

Argument `splineDF` specifies the degrees of freedom for natural spline fitting. As a single numeric value, it is directly passed on to the `splineDF` argument of `splines::ns`. Experience shows that `splineDF = 4` yields good results for TPP data sets with 10 temperature points. It is also possible to provide a numeric vector. In this case, splines are fitted for each entry and the optimal value is chosen per protein using Akaike's Information criterion.

## Value

A list of two data frames: 1. A long table containing the spline predictions per protein and TMT-label 2. A long table containing the hypothesis test results per protein.

## See Also

`ns`, `AICc`, `tppDefaultTheme`

## Examples

```
data(hdacTR_smallExample)
tpptrData <- tpptrImport(configTable = hdacTR_config, data = hdacTR_data)
normResults <- tpptrNormalize(data = tpptrData,
                              normReqs = tpptrDefaultNormReqs())
normData_eSets <- normResults$normData
longTables <- normData_eSets %>% tpptrTidyUpESets
fitData <- longTables %>% extract2("proteinMeasurements")
proteinInfos <- longTables %>% extract2("proteinAnnotation")
hdacSplineFits <- tpptrSplineFitAndTest(data = fitData,
                                        factorsH1 = "condition",
                                        additionalCols = proteinInfos,
                                        nCores = 1)
# Show estimated splines for HDAC1:
filter(hdacSplineFits, Protein_ID == "HDAC1")
# Quality control: test for replicate-specific effects:
 testResults <- tpptrSplineFitAndTest(data = fitData,
                                     factorsH1 = "replicate")
# -> Which proteins showed significant replicate effects?
testResults %>% filter(p_adj_NPARC <= 0.01) %>% select(Protein_ID, p_adj_NPARC)
```

# `tpptrTidyUpESets` *Tidy up expressionSets*

## Description

Convert list of expressionSets (intermediate output of several TR and CCR functions) to list of tidy tables.

## Usage

```
tpptrTidyUpESets(tppESetList)
```

## Arguments

`tppESetList` A list of expressionSets, which currently still is the format for intermediate results of most TR and CCR functions.

## Details

expressionSet lists are for example produced by tpptrImport, tpptrNormalize, tpptrCurveFit, tppccrImport, tppccrNormalize, tppccrCurveFit.

## Value

A list with two entries: `proteinMeasurements` contains the fold changes per protein, across all experiments. `proteinAnnotation` contains additional annotation per protein and experiment. For example, the peptide counts per identified protein would go here.

## Examples

```
data(hdacTR_smallExample)
tpptrData <- tpptrImport(configTable = hdacTR_config, data = hdacTR_data)
longTables <- tpptrData %>% tpptrTidyUpESets
concentrations <- longTables %>% extract2("proteinMeasurements")
additionalInfos <- longTables %>% extract2("proteinAnnotation")
summary(concentrations)
```

# Index