Package 'pepStat'

September 25, 2024

Type Package

Title Statistical analysis of peptide microarrays

Version 1.39.0

Author Raphael Gottardo, Gregory C Imholte, Renan Sauteraud, Mike Jiang

Maintainer Gregory C Imholte <gimholte@uw.edu>

Description Statistical analysis of peptide microarrays

License Artistic-2.0

Depends R (>= 3.0.0), Biobase, IRanges

Imports limma, fields, GenomicRanges, ggplot2, plyr, tools, methods, data.table

Suggests pepDat, Pviz, knitr, shiny

biocViews Microarray, Preprocessing

URL https://github.com/RGLab/pepStat

VignetteBuilder knitr

git_url https://git.bioconductor.org/packages/pepStat

git_branch devel

git_last_commit 3b4c2dd

git_last_commit_date 2024-04-30

Repository Bioconductor 3.20

Date/Publication 2024-09-24

Contents

`baselineCorrect.pSet` *Substract baseline intensities*

Description

Correct intensities by substracting PRE visit sample intensities.

Usage

```
baselineCorrect.pSet(pSet, verbose = FALSE)
```

Arguments

`pSet`	A `peptideSet` with sample PRE and POST visits.
`verbose`	A `logical`. If TRUE, information regarding the pairedness of the data will be displayed.

Details

If samples are not PAIRED (One PRE and POST for each ptid), then the average expression of all PRE visit samples is substracted from each sample.

Value

A `matrix` of the baseline corrected intensities, with as many columns as there are samples POST visit

Author(s)

Raphael Gottardo, Gregory Imholte

`baseline_correct` *Substract baseline intensities*

Description

Correct intensities by substracting PRE visit sample intensities.

Usage

```
baseline_correct(pSet, verbose = FALSE)
```

Arguments

pSet	A peptideSet with sample PRE and POST visits.
verbose	A logical. If TRUE, information regarding the pairedness of the data will be displayed.

Details

The function will try to pair as many sample as possible. The remaining subjects with a POST and no PRE will use the average expression of all baseline samples. Subjects with baseline only will not be represented in the resulting matrix.

Value

A matrix of the baseline corrected intensities, with as many columns as there are samples POST visit

Author(s)

Renan Sauteraud

---

create_db *Create a peptide collection*

Description

Constructor to create peptide collection to be used in summarizePeptides.

Usage

```
create_db(position)
```

Arguments

position	A data.frame or GRanges object. If a data.frame is provided, it should contain 'start' and 'end' or 'width' columns as well as a peptide column. If position is a GRanges object, then it must either have peptide as names or contain a peptide metadata column.

Details

position can have additional columns. These columns will be kept in the resulting peptide collection. This is especially useful to include clades and grouping parameters for the makeCalls function.

If the input contains all the z-scores (z1 to z5), then they will not be re-calculated. If some (but not all) z-scores are missing, a warning message will be sent and the z-scores are re-calculated.

Author(s)

Renan Sauteraud

See Also

GRanges

Examples

```
#construct data.frame object
   AA <- c("A", "C", "D", "E", "F", "G", "H", "I", "K", "L", "M", "N", "P",
   "Q","R", "S", "T", "V", "W", "Y")
   starts <- seq(1, 30, 3)
   ends <- starts + 14
   peptides <- sapply(1:10, function(x) {
     paste0(AA[floor(runif(15, 1, 20))], collapse = "")
   })
   data <- data.frame(start = starts, end = ends, peptide = peptides)
#from data.frame
   new_pep <- create_db(data)
#from GRanges
   new_pep <- create_db(new_pep)
```

---

makeCalls *Make antibody binding positivity calls*

Description

After normalization and data smoothing, this last step makes the call for each peptide of the peptideSet after baseline correcting the peptide intenstities.

Usage

```
makeCalls(peptideSet, cutoff = 1.2, method = "absolute", freq = TRUE,
  group = NULL, verbose = FALSE)
```

Arguments

peptideSet	A peptideSet object. The peptides, after normalization and possibly data smoothing.
cutoff	A numeric. If FDR, the FDR threshold. Otherwise, a cutoff for the background corrected intensities.
method	A character. The method used to make positivity calls. "absolute" and "FDR" are available. See details below.
freq	A logical. If set to TRUE, return the percentage of slides calling a peptide positive. Otherwise, return a logical indicating binding events.
group	A character. Only used when freq is set to TRUE. A character indicating a variable by which to group slides. If non-null the percentage is calculated by group.
verbose	A logical. If set to TRUE, progress information will be displayed.

Details

This function requires specific variables ptid and visit in pData(peptideSet). The variable `ptid` should indicate subjects, and the variable `visit` should be a factor with levels pre and post.

If slides are paired for subjects, intensities corresponding to post-visit are substracted from pre. If slides are not paired, slides with pre have intensities averaged by peptides, and averaged peptide intensities are subtracted from slides that have entry post. Calls are made on these baseline corrected intensities.

When method = FDR, a left-tail method is used to generate a threshold controlling the False Discovery Rate at level `cutoff`. When method = absolute, Intensities exceeding the threshold are labelled as positive.

When freq = TRUE a group variable may be specified. The argument group indicates the name of a variable in pData(peptideSet) by which positive calls should be grouped. The call frequency for each peptide is calculated within groups.

Value

If freq = TRUE, a `numeric` `matrix` with peptides as rows and groups as columns where the values are the frequency of response in the group. If freq = FALSE, a `logical` `matrix` indicating binding events for each peptide in each subject.

Author(s)

Greg Imholte

Examples

```
## This example curated from the vignette -- please see vignette("pepStat")
## for more information
if (require("pepDat")) {

  ## Get example GPR files + associated mapping file
  dirToParse <- system.file("extdata/gpr_samples", package = "pepDat")
  mapFile <- system.file("extdata/mapping.csv", package = "pepDat")

  ## Make a peptide set
  pSet <- makePeptideSet(files = NULL, path = dirToParse,
                         mapping.file = mapFile, log=TRUE)

  ## Plot array images -- useful for quality control
  plotArrayImage(pSet, array.index = 1)
  plotArrayResiduals(pSet, array.index = 1, smooth = TRUE)

  ## Summarize peptides, using pep_hxb2 as the position database
  data(pep_hxb2)
  psSet <- summarizePeptides(pSet, summary = "mean", position = pep_hxb2)

  ## Normalize the peptide set
  pnSet <- normalizeArray(psSet)

  ## Smooth
  psmSet <- slidingMean(pnSet, width = 9)

  ## Make calls
  calls <- makeCalls(psmSet, freq = TRUE, group = "treatment",
```

```
                    cutoff = .1, method = "FDR", verbose = TRUE)

  ## Produce a summary of the results
  summary <- restab(psmSet, calls)

}
```

`makePeptideSet` *peptideSet constructor*

Description

This function reads GenePix results (.gpr) files and creates a peptideSet object gathering experiment information.

Usage

```
makePeptideSet(files = NULL, path = NULL, mapping.file = NULL,
  use.flags = FALSE, rm.control.list = NULL, empty.control.list = NULL,
  bgCorrect.method = "normexp", log = TRUE, check.row.order = FALSE,
  verbose = FALSE)
```

Arguments

`files`	A `character` vector. If NULL, all files with a .gpr extension in the specified path will be read.
`path`	A `character` string. The directory where the .gpr files to read are located.
`mapping.file`	A `character` string or `data.frame`. A mapping file that gives information for each sample. See details section below for a list of required fields.
`use.flags`	A `logical`. Should spots with flag value -99 or lower be excluded?
`rm.control.list`	A `character` vector. The name of the controls to be excluded from the peptideSet.
`empty.control.list`	A `character` vector. The name of the empty controls. If non NULL, the intensity of these empty spots will be substracted from remaining intensities.
`bgCorrect.method`	A `character` string. The name of the method used for background correction. This is passed to limma's backgroundCorrect method. See details section below and ?backgroundCorrect for more information.
`log`	A `logical`. If TRUE, intensities will be log2 transformed after BG correction.
`check.row.order`	A `logical`. Should slides be reduced to a common set of peptides?
`verbose`	A `logical`. Displays progress and additional information.

Details

GenePix results files (.gpr) are read when found in either the files or path arguments. By default, the foreground and background median intensities of the "red" channels, "F635 Median" and "B635 Median", are read. The background correction specified in bgCorrect.method is passed to the backgroundCorrect method in the limma package.

The mapping.file can be either a filename or a data.frame. In any case, it should contain at least three columns labeled "filename", "ptid" and "visit". The filenames given here should match those read from the path or files argument, or be a subset of it. For each ptid (patient ID), the visit column should have at least one "pre" and one "post" sample. Any additional column will be kept into the resulting `peptideSet` and can be used later on for groupping.

If check.row.order = TRUE, the final set of probes is taken to be those with IDs found in all arrays that were read.

Row, Column and Block spatial array position for each probe are stored in the `featureRanges` slot of the returned object.

Value

A `peptideSet` object that contain the intensities, peptide sequences and annotations available in the mapping file.

Author(s)

Raphael Gottardo, Gregory Imholte

See Also

`peptideSet`, `read.maimages`, `backgroundCorrect`

Examples

```
# Read gpr files
library(pepDat)
mapFile <- system.file("extdata/mapping.csv", package = "pepDat")
dirToParse <- system.file("extdata/gpr_samples", package = "pepDat")
pSet <- makePeptideSet(files = NULL, path = dirToParse,
                       mapping.file = mapFile, log=TRUE)

# Specify controls to be removed and empty controls
# to be used for background correction.
pSet <- makePeptideSet(files = NULL, path = dirToParse,
                       mapping.file = mapFile, log = TRUE,
                       rm.control.list = c("JPT-control", "Ig", "Cy3"),
                       empty.control.list= c("empty", "blank control"))
```

`normalizeArray` *Normalize tiling array data using sequence information*

Description

This function is used to normalize the peptide microarray data using sequence information.

Usage

```
normalizeArray(peptideSet, method = "ZpepQuad", robust = TRUE,
  centered = TRUE)
```

Arguments

peptideSet	A peptideSet. The expression data for the peptides as well as annotations and ranges.
method	A character. The normalization method to be used. Can be "Zpep" or "ZpepQuad".
robust	A logical. If TRUE, reweigthed least-squares estimates are computed.
centered	A logical. If TRUE, recenter the data.

Details

The available methods are "Zpep" and "ZpepQuad". These methods fit a linear model using either linear or linear and quadratic terms (respectively), regressing intensity on the peptides' five Z-scale scores. A peptide Z-scale score is obtained by summing over the Z-scale values in Sandburg et al (1998) of the amino acids the peptide comprises.

Peptide Z-scale scores may be provided in the featureRange slot of peptideSet. This slot is a GRanges object x, and the function will seek five columns labelled z1 through z5 in values(x). If these are not found, the function attempts to calculate Z-scales from sequence information found in peptide(peptideSet)

If robust = TRUE the linear model is fit with t_4 distributed errors. The method returns the residuals of each peptide intensity in the fitted linear model. If centered = TRUE the fitted intercept term is added back to the residuals of the fit.

Value

A peptideSet object with updated normalized intensity values.

Author(s)

Raphael Gottardo, Gregory Imholte

References

Sandberg, M., Eriksson, L., Jonsson, J., Sjostrom, M., and Wold, S. (1998). New chemical descriptors relevant for the design of biologically active peptides. A multivariate characterization of 87 amino acids. Journal of Medicinal Chemistry 41, 2481-2491.

See Also

summarizePeptides, makeCalls

Examples

```
## This example curated from the vignette -- please see vignette("pepStat")
## for more information
if (require("pepDat")) {

  ## Get example GPR files + associated mapping file
  dirToParse <- system.file("extdata/gpr_samples", package = "pepDat")
```

```
  mapFile <- system.file("extdata/mapping.csv", package = "pepDat")

  ## Make a peptide set
  pSet <- makePeptideSet(files = NULL, path = dirToParse,
                         mapping.file = mapFile, log=TRUE)

  ## Plot array images -- useful for quality control
  plotArrayImage(pSet, array.index = 1)
  plotArrayResiduals(pSet, array.index = 1, smooth = TRUE)

  ## Summarize peptides, using pep_hxb2 as the position database
  data(pep_hxb2)
  psSet <- summarizePeptides(pSet, summary = "mean", position = pep_hxb2)

  ## Normalize the peptide set
  pnSet <- normalizeArray(psSet)

  ## Smooth
  psmSet <- slidingMean(pnSet, width = 9)

  ## Make calls
  calls <- makeCalls(psmSet, freq = TRUE, group = "treatment",
                     cutoff = .1, method = "FDR", verbose = TRUE)

  ## Produce a summary of the results
  summary <- restab(psmSet, calls)

}
```

`peptideSet` *peptideSet class*

Description

This class gathers all information from gpr files, annotation data and sequence data

Details

See `?`peptideSet-methods`` for a list of accessors and method associated with the class.

Slots

featureRange A `GRangesobject`. The ranges and sequences of the peptides and their associated annotation.

phenoData An `AnnotatedDataFrame`. Annotation for the samples.

assayData

featureData

annotation

protocolData Slots inherited from `ExpressionSet`.

Author(s)

Greg Imholte

See Also

ExpressionSet, peptideSet-methods

---

peptideSet-methods *peptideSet methods*

Description

Methods for handling peptideSet objects

Accessors

`nrow(x)`: The number of peptides in x.

`ncol(x)`: The number of samples in x.

`start(x)`: Get the starts of the peptides.

`end(x)`: Get the ends of the peptides.

`width(x)`: Get the widths of the peptides.

`position(x)`: Get the coordinates of the central amino-acid of each peptide, given by: `round((start(x) + end(x))/2)`.

`ranges(x)`: Returns a `GRanges` object that contains the annotations for the peptides.

`ranges(x)<- value` Set annotations for the peptides.

`values(x)`: Returns a `SplitDataFrameList`. Accessor for the values of the featureRange slot.

`clade(x)`: If available, returns the clade information for each peptide as a `matrix`.

`peptide(x)`: Get the sequence of the peptides.

`peptide(x) <- value` Set the sequence of the peptides.

`featureID(x)`: Get the ID of the peptides.

`pepZscore(x)`: If available, returns a `matrix` of the zScores for each peptide.

`pepZscore(x) <- value` Set the zScores for each peptide

Display

`show(object)`: Display a peptideSet object.

`summary(object)`: Summarize a peptideSet object.

Subset

`x[i, j]`: Subset x by peptides (i), or samples (j).

`subset(x, subset, drop=FALSE)`: Subset x given an expression 'subset'.

plotArrayImage *Plot microarray images*

Description

Plot a color image of the intensities on a slide. These plots are helpful to diagnose spatial abnormalities.

Usage

```
plotArrayImage(peptideSet, array.index = NULL, low = "white",
  high = "steelblue", ask = dev.interactive(orNone = TRUE) & 1 <
  length(array.index))

plotArrayResiduals(peptideSet, array.index = NULL, smooth = FALSE,
  low = "blue", high = "red", ask = dev.interactive(orNone = TRUE) & 1 <
  length(array.index))
```

Arguments

peptideSet	A peptideSet object. The object must contain all the original probes. See details below.
array.index	A vector subsetting exprs(peptideSet), indicating which slides to plot
smooth	A logical, a 2D spatial smoother is applied to residuals, the fitted residuals are plotted.
low	A character string. The color of the lowest slide intensity. passed to scale_fill_gradient2. the fitted residuals are plotted.
high	A character string. The color of the highest slide intensity. passed to scale_fill_gradient2.
ask	A logical. If TRUE, the user is asked before each plot. See par(ask=.).

Details

The most coherent results are achieved when the peptideSet object is read with makePeptideSet with empty.control.list = NULL and rm.control.list = NULL

Author(s)

Gregory Imholte

Examples

```
## This example curated from the vignette -- please see vignette("pepStat")
## for more information
if (require("pepDat")) {

  ## Get example GPR files + associated mapping file
  dirToParse <- system.file("extdata/gpr_samples", package = "pepDat")
  mapFile <- system.file("extdata/mapping.csv", package = "pepDat")

  ## Make a peptide set
  pSet <- makePeptideSet(files = NULL, path = dirToParse,
```

```
                        mapping.file = mapFile, log=TRUE)

  ## Plot array images -- useful for quality control
  plotArrayImage(pSet, array.index = 1)
  plotArrayResiduals(pSet, array.index = 1, smooth = TRUE)

  ## Summarize peptides, using pep_hxb2 as the position database
  data(pep_hxb2)
  psSet <- summarizePeptides(pSet, summary = "mean", position = pep_hxb2)

  ## Normalize the peptide set
  pnSet <- normalizeArray(psSet)

  ## Smooth
  psmSet <- slidingMean(pnSet, width = 9)

  ## Make calls
  calls <- makeCalls(psmSet, freq = TRUE, group = "treatment",
                     cutoff = .1, method = "FDR", verbose = TRUE)

  ## Produce a summary of the results
  summary <- restab(psmSet, calls)

}
```

`restab` *Result table*

Description

Tabulate the results of a peptide microarray analysis.

Usage

```
restab(peptideSet, calls)
```

Arguments

`peptideSet`	A `peptideSet` object.
`calls`	A `matrix`, as returned by the `makeCalls` function.

Details

The peptideSet should be the one used in the function call to `makeCalls` that generated the calls used. They should have identical peptides.

Value

A `data.frame` with the peptides and some information from the `peptideSet` as well as the frequency of binding for each group of the calls.

Examples

```
## This example curated from the vignette -- please see vignette("pepStat")
## for more information
if (require("pepDat")) {

  ## Get example GPR files + associated mapping file
  dirToParse <- system.file("extdata/gpr_samples", package = "pepDat")
  mapFile <- system.file("extdata/mapping.csv", package = "pepDat")

  ## Make a peptide set
  pSet <- makePeptideSet(files = NULL, path = dirToParse,
                         mapping.file = mapFile, log=TRUE)

  ## Plot array images -- useful for quality control
  plotArrayImage(pSet, array.index = 1)
  plotArrayResiduals(pSet, array.index = 1, smooth = TRUE)

  ## Summarize peptides, using pep_hxb2 as the position database
  data(pep_hxb2)
  psSet <- summarizePeptides(pSet, summary = "mean", position = pep_hxb2)

  ## Normalize the peptide set
  pnSet <- normalizeArray(psSet)

  ## Smooth
  psmSet <- slidingMean(pnSet, width = 9)

  ## Make calls
  calls <- makeCalls(psmSet, freq = TRUE, group = "treatment",
                     cutoff = .1, method = "FDR", verbose = TRUE)

  ## Produce a summary of the results
  summary <- restab(psmSet, calls)

}
```

`shinyPepStat` *Launch the pepStat Shiny Application*

Description

Launches the `pepStat` Shiny application, providing an interactive interface for constructing peptide sets, normalizing intensities, generating calls. Quality control is also facilitated through interactive plotting features.

Usage

```
shinyPepStat()
```

Examples

```
if (interactive()) {
  shinyPepStat()
}
```

slidingMean *Data smoothing for peptide microarray.*

Description

This function applies a sliding mean window to intensities to reduce noise generated by experimental variation, as well as take advantage of the overlapping nature of array peptides to share signal.

Usage

```
slidingMean(peptideSet, width = 9, verbose = FALSE, split.by.clade = TRUE)
```

Arguments

peptideSet	A peptideSet. The expression data for the peptides as well as annotations and ranges. The range information is required to run this function.
width	A numeric. The width of the sliding window.
verbose	A logical. If set to TRUE, progress information will be displayed.
split.by.clade	A logical. If TRUE, the peptides will be smoothed by clade (See details section below for more information).

Details

Peptide membership in the sliding mean window is determined by its position and the width argument. Two peptides are in the same window if the difference in their positions is less than or equal to width/2. A peptide's position is taken to be position(peptideSet).

A peptide's intensity is replaced by the mean of all peptide intensities within the peptide's sliding mean window.

When split.by.clade = TRUE, peptides are smoothed within clades defined by the clade column of the GRanges object occupying the featureRange slot of peptideSet. If set to FALSE, a peptide at a given position will borrow information from the neighboring peptides as well as the ones from other clades around this position.

Value

A peptideSet object with smoothed intensities.

Author(s)

Gregory Imholte

See Also

summarizePeptides, normalizeArray

Examples

```
## This example curated from the vignette -- please see vignette("pepStat")
## for more information
if (require("pepDat")) {

  ## Get example GPR files + associated mapping file
  dirToParse <- system.file("extdata/gpr_samples", package = "pepDat")
  mapFile <- system.file("extdata/mapping.csv", package = "pepDat")

  ## Make a peptide set
  pSet <- makePeptideSet(files = NULL, path = dirToParse,
                         mapping.file = mapFile, log=TRUE)

  ## Plot array images -- useful for quality control
  plotArrayImage(pSet, array.index = 1)
  plotArrayResiduals(pSet, array.index = 1, smooth = TRUE)

  ## Summarize peptides, using pep_hxb2 as the position database
  data(pep_hxb2)
  psSet <- summarizePeptides(pSet, summary = "mean", position = pep_hxb2)

  ## Normalize the peptide set
  pnSet <- normalizeArray(psSet)

  ## Smooth
  psmSet <- slidingMean(pnSet, width = 9)

  ## Make calls
  calls <- makeCalls(psmSet, freq = TRUE, group = "treatment",
                     cutoff = .1, method = "FDR", verbose = TRUE)

  ## Produce a summary of the results
  summary <- restab(psmSet, calls)

}
```

`summarizePeptides` *Add information to a peptideSet and summarize peptides*

Description

This function merges the replicates and adds information from a peptide collection to a peptideSet. This collection can include coordinates, alignment information, Z-scales, and other peptide information.

Usage

```
summarizePeptides(peptideSet, summary = "median", position = NULL)
```

Arguments

`peptideSet`	A `peptideSet`, as created by `makePeptideSet`
`summary`	A `character` string. The method used for merging replicates. Available are: "mean" and "median".

position — A data.frame or GRanges object. A peptide collection such as the ones available in pepDat. See details below and vignettes for more information.

Details

The object in the position argument will be passed to create_db, it can either be a GRanges object with a peptide as a metadata column, or a data.frame that can be used to create such GRanges.

Some peptide collections can be found in the pepDat package.

Value

An object of class peptideSet with added columns and updated ranges.

Author(s)

Raphael Gottardo, Greory Imholte

See Also

makePeptideSet, create_db, create_db

Examples

```
## This example curated from the vignette -- please see vignette("pepStat")
## for more information
if (require("pepDat")) {

  ## Get example GPR files + associated mapping file
  dirToParse <- system.file("extdata/gpr_samples", package = "pepDat")
  mapFile <- system.file("extdata/mapping.csv", package = "pepDat")

  ## Make a peptide set
  pSet <- makePeptideSet(files = NULL, path = dirToParse,
                         mapping.file = mapFile, log=TRUE)

  ## Plot array images -- useful for quality control
  plotArrayImage(pSet, array.index = 1)
  plotArrayResiduals(pSet, array.index = 1, smooth = TRUE)

  ## Summarize peptides, using pep_hxb2 as the position database
  data(pep_hxb2)
  psSet <- summarizePeptides(pSet, summary = "mean", position = pep_hxb2)

  ## Normalize the peptide set
  pnSet <- normalizeArray(psSet)

  ## Smooth
  psmSet <- slidingMean(pnSet, width = 9)

  ## Make calls
  calls <- makeCalls(psmSet, freq = TRUE, group = "treatment",
                     cutoff = .1, method = "FDR", verbose = TRUE)

  ## Produce a summary of the results
  summary <- restab(psmSet, calls)
```

```
}
```

Index