

Package ‘oppti’

October 14, 2021

Type Package

Title Outlier Protein and Phosphosite Target Identifier

Version 1.6.0

Date 2019-07-03

Author Abdulkadir Elmas

Maintainer Abdulkadir Elmas <abdulkadir.elmas@mssm.edu>

Description

The aim of oppti is to analyze protein (and phosphosite) expressions to find outlying markers for each sample in the given cohort(s) for the discovery of personalized actionable targets.

License MIT

Encoding UTF-8

LazyData true

Depends R (>= 3.6)

Imports limma, stats, reshape, ggplot2, grDevices, RColorBrewer, pheatmap, knitr, methods, devtools

Suggests

VignetteBuilder knitr

URL <https://github.com/Huang-lab/oppti>

BugReports <https://github.com/Huang-lab/oppti/issues>

biocViews Proteomics, Regression, DifferentialExpression, BiomedicalInformatics, GeneTarget, GeneExpression, Network

RoxygenNote 6.1.1

git_url <https://git.bioconductor.org/packages/oppti>

git_branch RELEASE_3_13

git_last_commit db1f27a

git_last_commit_date 2021-05-19

Date/Publication 2021-10-14

R topics documented:

artImpute	2
clusterData	3
dropMarkers	5
dysReg	6
markOut	6
oppti	8
outScores	9
plotDen	10
rankPerOut	10
statTest	11

Index	13
--------------	-----------

artImpute	<i>Artificially miss and impute each data entry individually by ignoring outlying values</i>
-----------	--

Description

Infers the normal-state expression of a marker based on its co-expression network, i.e., the weighted average of the marker's nearest neighbors in the data. The returned imputed data will later be used to elucidate dysregulated (protruding) events.

Usage

```
artImpute(dat, ku = 6, marker.proc.list = NULL, miss.pstat = 0.4,
          verbose = FALSE)
```

Arguments

dat	an object of log ₂ -normalized protein (or gene) expressions, containing markers in rows and samples in columns.
ku	an integer in [1,num.markers], upper bound on the number of nearest neighbors of a marker.
marker.proc.list	character array, the row names of the data to be processed/imputed.
miss.pstat	the score threshold for ignoring potential outliers during imputation. miss.pstat = 1 ignores values outside of the density box (i.e., 1st-3rd quartiles). The algorithm ignores values lying at least (1/miss.pstat)-1 times IQR away from the box; e.g., use miss.pstat=1 to ignore all values lying outside of the box; use miss.pstat=0.4 to ignore values lying at least 1.5 x IQR away from the box; use miss.pstat=0 to employ all data during imputation.
verbose	logical, to show progress of the algorithm.

Value

the imputed data that putatively represents the expressions of the markers in the (matched) normal states.

Examples

```
dat = setNames(as.data.frame(matrix(1:(5*10),5,10),
row.names = paste('marker',1:5,sep='')), paste('sample',1:10,sep=''))
imputed = artImpute(dat, ku = 2)
```

clusterData

Hierarchical cluster analysis

Description

Displays the hierarchically clustered data by the "pheatmap" package. The numbers of clusters along the markers/samples can be set by the user, then the cluster structures are estimated by pairwise analysis.

Usage

```
clusterData(data, annotation_row = NULL, annotation_col = NULL,
annotation_colors = NULL, main = NA, legend = TRUE,
clustering_distance_rows = "euclidean",
clustering_distance_cols = "euclidean", display_numbers = FALSE,
number_format = "%.0f", num_clusters_row = NULL,
num_clusters_col = NULL, cluster_rows = TRUE, cluster_cols = TRUE,
border_color = "gray60", annotate_new_clusters_col = FALSE,
zero_white = FALSE, color_palette = NULL, show_rownames = FALSE,
show_colnames = FALSE, min_data = min(data, na.rm = TRUE),
max_data = max(data, na.rm = TRUE),
treeheight_row = ifelse(methods::is(cluster_rows, "hclust") ||
cluster_rows, 50, 0), treeheight_col = ifelse(methods::is(cluster_cols,
"hclust") || cluster_cols, 50, 0))
```

Arguments

data	an object of log ₂ -normalized protein (or gene) expressions, containing markers in rows and samples in columns.
annotation_row	data frame that specifies the annotations shown on left side of the heat map. Each row defines the features for a specific row. The rows in the data and in the annotation are matched using corresponding row names. Note that color schemes takes into account if variable is continuous or discrete.
annotation_col	similar to annotation_row, but for columns.
annotation_colors	list for specifying annotation_row and annotation_col track colors manually. It is possible to define the colors for only some of the features.

<code>main</code>	character string, an overall title for the plot.
<code>legend</code>	logical, to determine if legend should be drawn or not.
<code>clustering_distance_rows</code>	distance measure used in clustering rows. Possible values are "correlation" for Pearson correlation and all the distances supported by <code>dist</code> , such as "euclidean", etc. If the value is none of the above it is assumed that a distance matrix is provided.
<code>clustering_distance_cols</code>	distance measure used in clustering columns. Possible values the same as for <code>clustering_distance_rows</code> .
<code>display_numbers</code>	logical, determining if the numeric values are also printed to the cells. If this is a matrix (with same dimensions as original matrix), the contents of the matrix are shown instead of original values.
<code>number_format</code>	format strings (C printf style) of the numbers shown in cells. For example "%0.2f" shows 2 decimal places and "%.1e" shows exponential notation (see more in <code>sprintf</code>).
<code>num_clusters_row</code>	number of clusters the rows are divided into, based on the hierarchical clustering (using <code>cutree</code>), if rows are not clustered, the argument is ignored.
<code>num_clusters_col</code>	similar to <code>num_clusters_row</code> , but for columns.
<code>cluster_rows</code>	logical, determining if the rows should be clustered; or a <code>hclust</code> object.
<code>cluster_cols</code>	similar to <code>cluster_rows</code> , but for columns.
<code>border_color</code>	color of cell borders on heatmap, use <code>NA</code> if no border should be drawn.
<code>annotate_new_clusters_col</code>	logical, to annotate cluster IDs (column) that will be identified.
<code>zero_white</code>	logical, to display 0 values as white in the colormap.
<code>color_palette</code>	vector of colors used in heatmap.
<code>show_rownames</code>	boolean, specifying if row names are be shown.
<code>show_colnames</code>	boolean, specifying if column names are be shown.
<code>min_data</code>	numeric, data value corresponding to minimum intensity in the <code>color_palette</code>
<code>max_data</code>	numeric, data value corresponding to maximum intensity in the <code>color_palette</code>
<code>treeheight_row</code>	the height of a tree for rows, if these are clustered. Default value is 50 points.
<code>treeheight_col</code>	the height of a tree for columns, if these are clustered. Default value is 50 points.

Value

`tree`, the hierarchical tree structure.

`cluster_IDs_row`, the (row) cluster identities of the markers.

`cluster_IDs_col`, the (column) cluster identities of the samples.

Examples

```
set.seed(1)
dat = setNames(as.data.frame(matrix(runif(10*10),10,10)),
row.names = paste('marker',1:10,sep='')), paste('sample',1:10,sep=''))
result = clusterData(dat)
```

dropMarkers

Filter out markers

Description

Filters out markers based on the percentage of missing values, low-expression and low-variability rates.

Usage

```
dropMarkers(dat, percent_NA = 0.2, low_mean_and_std = 0.05,
q_low_var = 0.25, force_drop = NULL)
```

Arguments

dat	an object of log2-normalized protein (or gene) expressions, containing markers in rows and samples in columns.
percent_NA	a constant in [0,1], the percentage of missing values that will be tolerated in the filtered data.
low_mean_and_std	a constant in [0,inf], the lower-bound of the mean or standard deviation of a marker in the filtered data.
q_low_var	a constant in [0,1], the quantile of marker variances which serves as a lower-bound of the marker variances in the filtered data.
force_drop	character array containing the marker names that user specifically wants to filter out.

Value

filtered data with the same format as the input data.

the row names (markers) of the data that are filtered out due to low-expression or low-variability.

Examples

```
dat = setNames(as.data.frame(matrix(1:(5*10),5,10)),
row.names = paste('marker',1:5,sep='')), paste('sample',1:10,sep=''))
dat[1,1:2] = NA # marker1 have 20% missing values
dropMarkers(dat, percent_NA = .2) # marker1 is filtered out
```

dysReg *Analyze dysregulated (protruding) events*

Description

For each marker processed, draws a scatter plot of matching values of observed vs imputed expressions.

Usage

```
dysReg(dat, dat.imp, marker.proc.list = NULL, verbose = FALSE)
```

Arguments

`dat` an object of log2-normalized protein (or gene) expressions, containing markers in rows and samples in columns.

`dat.imp` the imputed data that putatively represents the expressions of the markers in the (matched) normal states.

`marker.proc.list` character array, the row names of the data to be processed for dysregulation.

`verbose` logical, to show progress of the algorithm

Value

samples' distances to regression line (i.e., dysregulation) on the scatter plots.
the scatter plots.

Examples

```
dat = setNames(as.data.frame(matrix(1:(5*10),5,10),
row.names = paste('marker',1:5,sep='')), paste('sample',1:10,sep=''))
dat.imp = artImpute(dat, ku=2)
result = dysReg(dat, dat.imp)
```

markOut *Display outlying expressions*

Description

Mark outlying expressions on the scatter plot of a given marker

Usage

```
markOut(dat, dat.imp, dat.imp.test, dat.dys, dys.sig.thr.upp,
marker.proc.list = NULL, dataset = "", num.omit.fit = NULL,
draw.sc = TRUE, draw.vi = TRUE, conf.int = 0.95,
ylab = "Observed", xlab = "Inferred")
```

Arguments

<code>dat</code>	an object of log2-normalized protein (or gene) expressions, containing markers in rows and samples in columns.
<code>dat.imp</code>	the imputed data that putatively represents the expressions of the markers in the (matched) normal states.
<code>dat.imp.test</code>	marker's p-value of the statistical significance between its observed vs imputed values computed by the Kolmogorov-Smirnov test.
<code>dat.dys</code>	samples' distances to regression line (i.e., dysregulation) on the scatter plots.
<code>dys.sig.thr.upp</code>	the dysregulation score threshold to elucidate/mark significantly dysregulated outlier events.
<code>marker.proc.list</code>	character array, the row names of the data to be processed for outlier analyses and for plotting.
<code>dataset</code>	the cohort name to be used in the output files.
<code>num.omit.fit</code>	number of outlying events to ignore when fitting a marker's observed expressions to the imputed ones.
<code>draw.sc</code>	logical, to draw a scatter plot for every marker in <code>marker.proc.list</code> in a separate PDF file.
<code>draw.vi</code>	logical, to draw a violin plot for every marker in <code>marker.proc.list</code> in a separate PDF file.
<code>conf.int</code>	confidence interval to display around the regression line
<code>ylab</code>	a title for the y axis
<code>xlab</code>	a title for the x axis

Value

the scatter plots of the markers where the outlier dysregulation events are highlighted by red mark.

Examples

```
set.seed(1)
dat = setNames(as.data.frame(matrix(runif(10*10),10,10),
row.names = paste('marker',1:10,sep='')), paste('sample',1:10,sep=''))
dat.imp = artImpute(dat, ku=6)
dat.imp.test = statTest(dat, dat.imp)[[1]]
dat.dys = dysReg(dat, dat.imp)[[1]]
plots = markOut(dat, dat.imp, dat.imp.test, dat.dys, dys.sig.thr.upp = .25)
```

oppti

*Outlier protein and phosphosite target identification***Description**

Find outlying markers and events across cancer types.

Usage

```
oppti(data, mad.norm = FALSE, cohort.names = NULL, panel = "global",
panel.markers = NULL, tol.nas = 20, ku = 6, miss.pstat = 0.4,
demo.panels = FALSE, save.data = FALSE, draw.sc.plots = FALSE,
draw.vi.plots = FALSE, draw.sc.markers = NULL,
draw.ou.plots = FALSE, draw.ou.markers = NULL, verbose = FALSE)
```

Arguments

data	a list object where each element contains a proteomics data for a different cohort (markers in the rows, samples in the columns) or a character string defining the path to such data (in .RDS format).
mad.norm	logical, to normalize the proteomes to have a unit Median Absolute Deviation.
cohort.names	character array.
panel	a character string describing marker panel, e.g., 'kinases'. Use 'global' to analyze all markers quantified across cohorts (default). Use 'pancan' to analyze the markers commonly quantified across the cohorts.
panel.markers	a character array containing the set of marker names that user wants to analyze, e.g., panel.markers = c("AAK1", "AATK", "ABL1", "ABL2", ...).
tol.nas	a constant in [0,100], tolerance for the percentage of NAs in a marker, e.g., tol.nas = 20 will filter out markers containing 20% or more NAs across samples.
ku	an integer in [1,num.markers], upper bound on the number of nearest neighbors of a marker.
miss.pstat	a constant in [0,1], statistic to estimate potential outliers. See 'artImpute()'. demo.panels
demo.panels	logical, to draw demographics of the panel in each cohort.
save.data	logical, to save intermediate data (background inference and dysregulation measures).
draw.sc.plots	logical, to draw each marker's qqplot of observed vs inferred (imputed) expressions.
draw.vi.plots	logical, to draw each marker's violin plot of observed vs imputed expressions.
draw.sc.markers	character array, marker list to draw scatter plots
draw.ou.plots	logical, to draw each marker's outlier prevalence (by the percentage of outlying samples) across the cohorts.
draw.ou.markers	character array, marker list to draw pan-cancer outlier percentage plots
verbose	logical, to show progress of the algorithm.

Value

dysregulation scores of every marker for each sample.

the imputed data that putatively represents the expressions of the markers in the (matched) normal states.

the result of Kolmogorov-Smirnov tests that evaluates the statistical significance of each marker's outlier samples.

a data list containing, for each cohort, the percentage of outlier samples for every marker.

a data list containing, for each cohort, the outlier significance threshold.

See Also

[artImpute()] for how to set 'miss.pstat' and 'ku'

Examples

```
set.seed(1)
dat = setNames(as.data.frame(matrix(runif(10*10),10,10)),
row.names = paste('marker',1:10,sep='')), paste('sample',1:10,sep=''))
result = opti(dat)
```

outScores

Analyze putative outliers

Description

Calculates a statistical measure of each data entry being a putative outlier

Usage

```
outScores(dat)
```

Arguments

dat an object of log2-normalized protein (or gene) expressions, containing markers in rows and samples in columns.

Value

outlier p-statistics

Examples

```
dat = setNames(as.data.frame(matrix(1:(5*10),5,10)),
row.names = paste('marker',1:5,sep='')), paste('sample',1:10,sep=''))
result = outScores(dat)
```

plotDen *Draw densities*

Description

Draw column densities of an object over multiple plots by using `limma::plotDensities()` function.

Usage

```
plotDen(dat, name = "", per.plot = 8, main = NULL, group = NULL,
legend = TRUE)
```

Arguments

<code>dat</code>	an object of log ₂ -normalized protein (or gene) expressions, containing markers in rows and samples in columns.
<code>name</code>	name tag for the output file.
<code>per.plot</code>	number of densities to be drawn on a single plot. If <code>NULL</code> , <code>ncol(object)</code> will be used.
<code>main</code>	character string, an overall title for the plot.
<code>group</code>	vector or factor classifying the arrays into groups. Should be same length as <code>ncol(object)</code> .
<code>legend</code>	character string giving position to place legend. See ‘legend’ for possible values. Can also be logical, with <code>FALSE</code> meaning no legend.

Value

pdf plot(s).

Examples

```
dat = setNames(as.data.frame(matrix(1:(5*10),5,10),
row.names = paste('marker',1:5,sep='')), paste('sample',1:10,sep=''))
plotDen(dat, name = 'myresults')
```

rankPerOut *Rank markers by the percentage of outlying events*

Description

Ranks markers in the order of decreasing percentage of outlying events.

Usage

```
rankPerOut(dat.dys, marker.proc.list = NULL, dys.sig.thr.upp)
```

Arguments

`dat.dys` samples' distances to regression line (i.e., dysregulation) on the scatter plots.

`marker.proc.list` character array, the row names of the data to be processed for outlier analyses.

`dys.sig.thr.upp` the dysregulation score threshold to elucidate/mark significantly dysregulated outlier events.

Value

markers rank-ordered by the percentage of outliers over the samples.

the percentages of outliers corresponding to ranked markers.

Examples

```
set.seed(1)
dat = setNames(as.data.frame(matrix(runif(10*10),10,10)),
row.names = paste('marker',1:10,sep=' '), paste('sample',1:10,sep=' '))
dat.imp = artImpute(dat, ku=6)
dat.dys = dysReg(dat, dat.imp)[[1]]
result = rankPerOut(dat.dys, dys.sig.thr.upp = .25)
```

statTest	<i>Analyze dysregulation significance</i>
----------	---

Description

Rank-order markers by the significance of deviation of the observed expressions from the (matched) imputed expressions based on the Kolmogorov-Smirnov (KS) test.

Usage

```
statTest(dat, dat.imp, marker.proc.list = NULL, pval.insig = 0.2)
```

Arguments

`dat` an object of log2-normalized protein (or gene) expressions, containing markers in rows and samples in columns.

`dat.imp` the imputed data that putatively represents the expressions of the markers in the (matched) normal states.

`marker.proc.list` character array, the row names of the data to be processed for dysregulation significance.

`pval.insig` p-value threshold to determine spurious (null) dysregulation events.

Value

each marker's p-value of the statistical significance between its observed vs imputed values computed by the KS test.

ranked p-values (KS test) of the significant markers, which are lower than pval.insig.

ranked significantly dysregulated markers with p-values lower than pval.insig.

ranked p-values (KS test) of the insignificant markers, which are greater than pval.insig.

ranked insignificantly dysregulated markers (spurious dysregulations) with p-values greater than pval.insig.

Examples

```
set.seed(1)
dat = setNames(as.data.frame(matrix(runif(10*10),10,10)),
row.names = paste('marker',1:10,sep=''), paste('sample',1:10,sep=''))
dat.imp = artImpute(dat, ku=6)
result = statTest(dat, dat.imp) # the dysregulations on marker4 is
# statistically significant with p-value 0.05244755.
```

Index

artImpute, 2
clusterData, 3
dropMarkers, 5
dysReg, 6
markOut, 6
oppti, 8
outScores, 9
plotDen, 10
rankPerOut, 10
statTest, 11