Package ‘PAA’

April 23, 2016

Version 1.4.1

Title PAA (Protein Array Analyzer)

Author Michael Turewicz [aut, cre], Martin Eisenacher [ctb, cre]

Maintainer Michael Turewicz <michael.turewicz@rub.de>, Martin Eisenacher <martin.eisenacher@rub.de>

Depends R (>= 3.2.0), Rcpp (>= 0.11.6)

Imports e1071, gplots, limma, MASS, mRMRe, randomForest, ROCR, sva

LinkingTo Rcpp

Suggests BiocStyle, RUnit, BiocGenerics, vsn

Description PAA imports single color (protein) microarray data that has been saved in gpr file format - esp. ProtoArray data. After pre-processing (background correction, batch filtering, normalization) univariate feature pre-selection is performed (e.g., using the ``minimum M statistic'' approach - hereinafter referred to as ``mMs''). Subsequently, a multivariate feature selection is conducted to discover biomarker candidates. Therefore, either a frequency-based backwards elimination aproach or ensemble feature selection can be used. PAA provides a complete toolbox of analysis tools including several different plots for results examination and evaluation.

License BSD_3_clause + file LICENSE

URL http://www.medizinisches-proteom-center.de/PAA

SystemRequirements C++ software package Random Jungle

biocViews Classification, Microarray, OneChannel, Proteomics

NeedsCompilation yes

R topics documented:

batchAdjust *Adjust microarray data for batch effects.*

Description

Adjusts `EListRaw` or `EList` data for batch/lot effects.

Usage

```
batchAdjust(elist=NULL, log=NULL)
```

Arguments

`elist`	`EList` or `EListRaw` object containing the data to be adjusted (mandatory).
`log`	string (`"TRUE"` or `"FALSE"`) indicating whether the data is in log scale (mandatory).

Details

This is a wrapper to `sva`'s function `ComBat()` for batch adjustment using the empirical Bayes approach. To use batchAdjust the targets information of the `EList` or `EListRaw` object must contain the columns `"Batch"` (containing batch/lot information for each particular array) and `"Group"` (containing experimental group information for each particular array).

Value

An `EListRaw` or `EList` object with the adjusted data in log scale is returned.

Note

The targets information of the `EListRaw` or `EList` object must contain the columns `"Batch"` and `"Group"`.

Author(s)

Michael Turewicz, <michael.turewicz@rub.de>

References

The package sva by Jeffrey T. Leek et al. can be downloaded from Bioconductor (http://www.bioconductor.org/).

Johnson WE, Li C, and Rabinovic A (2007) Adjusting batch effects in microarray expression data using empirical Bayes methods. Biostatistics 8:118-27.

Examples

```
cwd <- system.file(package="PAA")
load(paste(cwd, "/extdata/Alzheimer.RData", sep=""))
elist <- elist[elist$genes$Block < 10,]
elist <- batchAdjust(elist=elist, log=FALSE)
```

batchFilter *Remove differential features regarding array batches/lots.*

Description

Finds differential features regarding array batches/lots and removes them.

Usage

```
batchFilter(elist=NULL, lot1=NULL, lot2=NULL, p.thresh=0.05, fold.thresh=1.5,
output.path=NULL)
```

Arguments

elist	EList or EListRaw object (mandatory).
lot1	vector of column names for group 1 (mandatory).
lot2	vector of column names for group 2 (mandatory).
p.thresh	positive float number between 0 and 1 indicating the maximum Student's t-test p-value for features to be considered as differential (e.g., "0.5").
fold.thresh	float number indicating the minimum fold change for features to be considered as differential (e.g., "1.5").
output.path	string indicating a path for saving results (optional).

Details

This function takes an EList or EListRaw object (see limma documentation) and the batch-specific column name vectors lot1 and lot2 to find differential features regarding batches/lots. For this purpose, thresholds for p-values (Student's t-test) and fold changes can be defined. To visualize the differential features a volcano plot is drawn. Then, differential features are removed and the remaining data are returned. When an output path is defined (via output.path) volcano plots and result files are saved on the hard disk.

Value

An ELList or EListRaw object without differential features regarding array batches/lots.

Author(s)

Michael Turewicz, <michael.turewicz@rub.de>

Examples

```
cwd <- system.file(package="PAA")
load(paste(cwd, "/extdata/Alzheimer.RData", sep=""))
elist <- elist[elist$genes$Block < 10,]
lot1 <- elist$targets[elist$targets$Batch=='Batch1','ArrayID']
lot2 <- elist$targets[elist$targets$Batch=='Batch2','ArrayID']
elist <- batchFilter(elist=elist, lot1=lot1, lot2=lot2, p.thresh=0.001,
  fold.thresh=3)
```

diffAnalysis *Differential analysis.*

Description

Performs a univariate differential analysis.

Usage

```
diffAnalysis(input=NULL, label1=NULL, label2=NULL, class1=NULL, class2=NULL,
output.path=NULL, mMs.matrix1=NULL, mMs.matrix2=NULL, above=1500,
between=400, features=NULL, feature.names=NULL)
```

Arguments

input	EList$E- or EListRaw$E-matrix extended by row names comprising BRC-IDs of the corresponding features (mandatory).
label1	vector of column names for group 1 (mandatory).
label2	vector of column names for group 2 (mandatory).
class1	label of group 1 (mandatory).
class2	label of group 2 (mandatory).
output.path	string indicating a path for saving the results (optionally).
mMs.matrix1	pre-computed mMs reference matrix (see mMsMatrix()) for group 1 (mandatory).
mMs.matrix2	pre-computed mMs reference matrix (see mMsMatrix()) for group 2 (mandatory).
above	mMs above parameter (integer). Default is "1500".
between	mMs between parameter (integer). Default is "400".
features	vector of row indices (optional).
feature.names	vector of corresponding feature names (additionally to features).

Details

This function takes an `EList$E`- or `EListRaw$E`-matrix (e.g., `temp <- elist$E`) extended by row names comprising BRC-IDs of the corresponding features. The BRC-IDs can be created via:
`brc <- paste(elist$genes[,1], elist$genes[,3], elist.$genes[,2])`.
The BRC-row names can be defined as follows: `rownames(temp) <- brc`. Furthermore, the corresponding column name vectors, group labels and mMs-parameters are needed to perform the univariate differential analysis. This analysis covers inter alia p-value computation, p-value adjustment (method: Benjamini & Hochberg, 1995), and fold change computation. Since the results table is usually large, a path for saving the results can be defined via `output.path`. Optionally, a vector of row indices (`features`) and additionally (not mandatory for subset analysis) a vector of corresponding feature names (`feature.names`) can be forwarded to perform the analysis for a feature subset.

Value

A matrix containing the analysis results is returned.

Author(s)

Michael Turewicz, <michael.turewicz@rub.de>

Examples

```
cwd <- system.file(package="PAA")
load(paste(cwd, "/extdata/Alzheimer.RData", sep=""))
elist <- elist[elist$genes$Block < 10,]
c1 <- paste(rep("AD",20), 1:20, sep="")
c2 <- paste(rep("NDC",20), 1:20, sep="")
mMs.matrix1 <- mMs.matrix2 <- mMsMatrix(x=20, y=20)
temp <- elist$E
rownames(temp) <- paste(elist$genes[,1], elist$genes[,3], elist$genes[,2])
diffAnalysis(input=temp, label1=c1, label2=c2, class1="AD", class2="NDC",
 mMs.matrix1=mMs.matrix1, mMs.matrix2=mMs.matrix2, above=1500,
 between=400)
```

`loadGPR` *A function to read raw data from gpr files.*

Description

Constructs an `EListRaw` object from a set of gpr files containing ProtoArray data or other one-color microarray data.

Usage

```
loadGPR(gpr.path = NULL, targets.path = NULL, array.type = "ProtoArray",
 protoarray.aggregation = "min", array.columns = list(E = "F635 Median",
 Eb = "B635 Median"),
 array.annotation = c("Block", "Column", "Row", "Description", "Name", "ID"))
```

Arguments

`gpr.path` string indicating the path to a folder containing gpr files (mandatory).

`targets.path` string indicating the path to targets file (see limma, mandatory).

`array.type` string indicating whether the gpr files contain ProtoArray data (default is `"ProtoArray"`). In any other case no duplicate aggregation will be performed (mandatory).

`protoarray.aggregation`
string indicating which type of ProtoArray spot duplicate aggregation should be performed. If `"min"` is chosen, the value for the corresponding feature will be the minimum of both duplicate values. If `"mean"` is chosen, the arithmetic mean will be computed. Alternatively, no aggregation will be performed, if `"none"` is chosen. The default is `"min"` (optional).

`array.columns` list containing the column names for foreground intensities (E) and background intensities (Eb) in the gpr files that is passed to `limma`'s `"read.maimages"` function (optional).

`array.annotation`
string vector containing further mandatory column names that are passed to limma (optional).

Details

This function is partially a wrapper to `limma`'s function `read.maimages()` featuring optional duplicate aggregation for ProtoArray data. Paths to a target file and to a folder containing gpr files (all gpr files in that folder that are listed in the targets file will be read) are mandatory. The folder `"R_HOME/library/PAA/extdata"` contains an exemplary target file that can be used as a template. If `array.type` is set to `"ProtoArray"` (default), duplicate spots can be aggregated. The corresponding method (`"min"`, `"mean"` or `"none"`) can be specified via `protoarray.aggregation`. As another ProtoArray-specific feature, control spot data and information will be stored in additional components of the returned object (see below). Arguments `array.columns` and `array.annotation` define the columns where `read.maimages()` will find foreground and background intensity values as well as other important columns. For `array.annotation` the default columns `"Block"`, `"Column"`, `"Row"`, `"Description"`, `"Name"` and `"ID"` are mandatory (exception: the `"Description"` column for ProtoArrays since a very general column can be constructed from the column `"Name"`).

Value

An extended object of class `EListRaw` (see the documentation of `limma` for details) is returned. If `array.type` is set to `"ProtoArray"` (default), the object provides additional components for control spot data: `C`, `Cb` and `cgenes` which are analogous to the probe spot data `E`, `Eb` and `genes`.

Note

Don't forget to check column names in your gpr files. They may differ from the default settings of `loadGPR()` and should be renamed to the default column names (see also the exemplary gpr files accompanying PAA as a reference for the default column names). At worst, important columns in your gpr files may be completely missing and should be added in order to provide all information needed by PAA.

Note that if array.type is not "ProtoArray", neither aggregation will be done nor controls components will be added to the returned object of class EListRaw.

Author(s)

Michael Turewicz, <michael.turewicz@rub.de>

References

The package limma by Gordon Smyth et al. can be downloaded from Bioconductor (http://www.bioconductor.org/).

Smyth, G. K. (2005). Limma: linear models for microarray data. In: Bioinformatics and Computational Biology Solutions using R and Bioconductor, R. Gentleman, V. Carey, S. Dudoit, R. Irizarry, W. Huber (eds.), Springer, New York, pages 397-420.

Examples

```
gpr <- system.file("extdata", package="PAA")
targets <- list.files(system.file("extdata", package="PAA"),
 pattern = "dummy_targets", full.names=TRUE)
elist <- loadGPR(gpr.path=gpr, targets.path=targets)
```

mMsMatrix *Compute a reference minimum M statistic (n1 x n2)-matrix.*

Description

Computes a reference minimum M statistic (n1 x n2)-matrix (mMs matrix).

Usage

```
mMsMatrix(x, y)
```

Arguments

x	integer, first dimension (i.e., number of samples in group 1) of the mMs matrix to be computed (mandatory).
y	integer, second dimension (i.e., number of samples in group 2) of the mMs matrix to be computed (mandatory).

Details

For feature pre-selection the "minimum M Statistic" (mMs) proposed by Love B. is used. The mMs is a univariate measure that is sensitive to population subgroups. To avoid redundant mMs computations for a large number of features (e.g., ca. 9500 features on ProtoArray v5) a reference matrix containing all relevant mMs values can be pre-computed. Therefore, only two parameters are needed: the number of samples in group 1 (n1) and the number of samples in group 2 (n2). According to mMs definition for each matrix element (i,m) a mMs value (= the probability of) for having m values in group 1 larger than the i-th largest value in group 2 is computed.

Value

A (n1 x n2)-matrix containing all mMs values for group 1 and group 2.

Note

To check whether a feature is more prevalent in group 1 or in group 2, PAA needs both the mMs for having m values in group 1 larger than the i-th largest element in group 2 as well as the mMs for having m values in group 2 larger than the i-th largest element in group 1. Hence, always both needs be computed `mMsMatrix(n1,n2)` and `mMsMatrix(n2,n1)`.

Author(s)

Michael Turewicz, <michael.turewicz@rub.de>

References

Love B: The Analysis of Protein Arrays. In: Functional Protein Microarrays in Drug Discovery. CRC Press; 2007: 381-402.

Examples

```
#exemplary computation for a group 1 comprising 10 arrays and a group 2
#comprising 12 arrays
mMs.matrix1 <- mMsMatrix(x=10, y=12)
mMs.matrix2 <- mMsMatrix(x=12, y=10)
```

normalizeArrays *Normalize microarray data.*

Description

Normalizes `EListRaw` data and returns an `EList` object containing normalized data in log scale.

Usage

```
normalizeArrays(elist = NULL, method = "quantile", cyclicloess.method = "pairs",
controls="internal", group1 = NULL, group2 = NULL, output.path=NULL)
```

Arguments

`elist` `EListRaw` object containing raw data to be normalized (mandatory).

`method` string indicating the normalization method (`"cyclicloess"`, `"quantile"` or `"vsn"`) to be used (mandatory).

`cyclicloess.method` string indicating which type of cyclicloess normalization (`"pairs"`, `"fast"`, `"affy"`) should be performed (optional).

`controls` sring indicating the ProtoArray controls for `rlm` normalization (optional). Valid options are `"internal"` (default), `"external"`, `"both"` or a regular expression defining a specific control or a specific set of controls.

`group1` vector of integers (column indices) indicating all group 1 samples (optional).

`group2` vector of integers (column indices) indicating all group 2 samples (optional).

`output.path` output.path for ProtoArray RLM normalization (optional).

Details

This function is partially a wrapper to `limma`'s function `normalizeBetweenArrays()` for inter-array normalization featuring optional groupwise normalization when the arguments `group1` AND `group2` are assigned. For more information on `"cyclicloess"`, `"quantile"` or `"vsn"` see the documentation of the `limma` package. Furthermore, for ProtoArrays robust linear normalization (`"rlm"`, see Sboner A. et al.) is provided.

For `rlm` normalization (`method = "rlm"`) the additional argument `controls` needs to be specified in order to select a set of controls used for normalization. Valid options are `"internal"` (default), `"external"` and `"both"` which refer to the following sets of ProtoArray controls:

- internal The set of all internal controls spotted on the ProtoArray. The human-IgG series and anti-human-IgG series, which respond to serum and secondary antibodies.
- external The V5-CMK1 series spotted on the ProtoArray which responds to exogenously added anti-V5 antibody (external control).
- both The combined set of both the internal and the external controls (i.e., the human-IgG and anti-human-IgG series and the V5-CMK1 series).

Moreover, via `controls` a regular expression can be passed in order to select a more specific group of controls. Please check the column `"Name"` in your gpr files in order to obtain the complete list of names of all controls spotted on the ProtoArray. In the following some examples of valid regular expressions are given:

- `"^HumanIg"` Only human IgGs and IgAs are selected (esp., no anti-human Igs).
- `"Anti-HumanIgA"` Only anti-human-IgAs are selected (esp., no human IgGs and IgAs).
- `"(Anti-HumanIg|^V5control|BSA|ERa)"` Only anti-human IgGs and anti-human IgAs, the V5-CMK1 series, BSA and ERa are selected.
- `"HumanIgG"` Only human IgGs and anti-human IgGs are selected.
- `"V5control"` Only the V5-CMK1 series is selected.

Value

An EList object with the normalized data in log scale is returned.

Note

Groupwise normalization will be performed when the arguments `group1` AND `group2` are assigned.

Author(s)

Michael Turewicz, `<michael.turewicz@rub.de>`

References

The package `limma` by Gordon Smyth et al. can be downloaded from Bioconductor (http://www.bioconductor.org/).

Smyth, G. K. (2005). Limma: linear models for microarray data. In: Bioinformatics and Computational Biology Solutions using R and Bioconductor, R. Gentleman, V. Carey, S. Dudoit, R. Irizarry, W. Huber (eds.), Springer, New York, pages 397-420.

Sboner A. et al., Robust-linear-model normalization to reduce technical variability in functional protein microarrays. J Proteome Res 2009, 8(12):5451-5464.

Examples

```
cwd <- system.file(package="PAA")
load(paste(cwd, "/extdata/Alzheimer.RData", sep=""))
elist <- elist[elist$genes$Block < 10,]
normalized.elist <- normalizeArrays(elist=elist, method="quantile")
```

`plotArray` *Plot ProtoArray expression intensities in the original arrangement mimicing the original scan image.*

Description

Uses the `"Block"`, `"Row"` and `"Column"` information of an `EList` or `EListRaw` object to resemble the original positions on the array(s). The resulting plot is similar to the original scan image of the considered array(s). Thus, this function is a visualization tool that can be used to visualize ProtoArrays for which the original scan image is not available. Visual inspection of the spatial expression pattern can then identify possible local tendencies and strong spatial biases. Moreover, the array can be inspected at all stages of the pre-processing workflow in order to check the impact of the particular methods that have been applied.

Usage

```
plotArray(elist=NULL, idx=NULL, data.type="fg", log=NULL, normalized=NULL,
  protoarray.aggregation=NULL, colpal="heat.colors", output.path=NULL)
```

Arguments

`elist` `EList` or `EListRaw` object (mandatory).

`idx` integer, vector of integers or the string `"all"` indicating the column indices of the sample(s) for drawing the plot(s) (mandatory).

`data.type` string indicating whether the foreground (`"fg"`) or background (`"bg"`) data should be plotted. The default is `"fg"` (optional).

`log` logical indicating whether the input data is logarithmized. If FALSE, the log2 will be plotted (mandatory).

`normalized` logical indicating whether `elist` was normalized (mandatory).

`protoarray.aggregation`
: string indicating whether the data stored in `elist` has been aggregated and, if this is the case, which method has been used by the function `loadGPR()`. Possible values are `"min"`, `"mean"` and `"none"`(mandatory).

`colpal`
: string indicating the color palette for the plot(s). The default is `"heat.colors"` (optional).

`output.path`
: string indicating the output path for the plots (optional).

Details

This function allows plotting of ProtoArray data using the `gplots` function `heatmap.2()` for visual quality control. The data obtained from an `EList` or `EListRaw` object is re-ordered and represented in the same way the spots are ordered on the actual ProtoArray. Consequently, the resulting plot is similar to the original scan image of the considered array. This allows for visual control and assessment of possible patterns in spatial distribution.

Mandatory arguments are `elist`, `idx`, `log`, `normalized` and `protoarray.aggregation`. While `elist` specifies the `EList` or `EListRaw` object to be used, `idx` designates the ProtoArray column index in `elist` to plot a single array from the `EList` object. Alternatively, a vector (e.g., `1:5`) or the string `"all"` can be designated to include multiple, respectively, all arrays that were imported.

Furthermore, `data.type` allows for plotting of `"fg"`, foreground data (i.e., `elist$E` and `elist$C`), which is the default or `"bg"`, background data (i.e., `elist$Eb` and `elist$Cb`).

The normalization approaches of PAA which comprise also data logarithmization do not include control data. With `normalized=TRUE` it is indicated that the input data was normalized, so the control data will be logarithmized before plotting as well. However, since the complete data (foreground and background values of protein features and control spots) can be logarithmized regardless of normalization the argument `log` states whether the designated data is already logarithmized.

The parameter `protoarray.aggregation` indicates whether the ProtoArray data has been aggregated by `loadGPR()` and, if so, which method has been used.

Moreover, the parameter `colpal` defines the color palette that will be used for the plot. Some exemplary values are `"heat.colors"` (default), `"terrain.colors"`, `"topo.colors"`, `"greenred"` and `"bluered"`.

Finally, the output path optionally can be specified with the argument `output.path` to save the plot(s). Then, one or more tiff file(s) containing the corresponding plot(s) are saved into the subfolder "array_plots".

Value

No value is returned.

Note

This function is designed to represent ProtoArray data. Please note the instructions of the PAA function `loadGPR()`. Note that the data has to be imported as `array.type = "ProtoArray"` (default) to include controls and that the data can be imported without aggregation by `loadGPR()` (when `protoarray.aggregation="none"`) in order to inspect the array visually before duplicate aggregation.

Author(s)

Daniel Bemmerl and Michael Turewicz <michael.turewicz@rub.de>

References

The package gplots by Gregory R. Warnes et al. can be downloaded from CRAN (http://CRAN.R-project.org/package=gplots).

Gregory R. Warnes, Ben Bolker, Lodewijk Bonebakker, Robert Gentleman, Wolfgang Huber, Andy Liaw, Thomas Lumley, Martin Maechler, Arni Magnusson, Steffen Moeller, Marc Schwartz and Bill Venables (2015). gplots: Various R Programming Tools for Plotting Data. R package version 2.17.0. http://CRAN.R-project.org/package=gplots

Examples

```
cwd <- system.file(package="PAA")
load(paste(cwd, "/extdata/BadData.RData", sep=""))
plotArray(elist=bad.elist, idx=1, data.type="bg", log=FALSE, normalized=FALSE,
  protoarray.aggregation="none")
```

plotFeatures *Plot intensities of selected features.*

Description

Plots intensities of all selected features (one sub-plot per feature) in group- specific colors.

Usage

```
plotFeatures(features = NULL, elist = NULL, n1 = NULL, n2 = NULL,
 group1 = "group1", group2 = "group2", output.path = NULL)
```

Arguments

features	vector containing the selected features as "BRC"-IDs (mandatory).
elist	EListRaw or EList object containing all intensity data in log scale (mandatory).
n1	integer indicating the sample size of group 1 (mandatory).
n2	integer indicating the sample size of group 2 (mandatory).
group1	class label of group 1.
group2	class label of group 2.
output.path	string indicating the folder where the figure will be saved (optional).

Details

Plots intensities of all features selected by the function selectFeatures() (one sub-plot per feature) in group-specific colors. All sub-plots are aggregated to one figure. When output.path is not NULL this figure will be saved in a tiff file in output.path. This function can be used to check whether the selected features are differential.

Value

No value is returned.

Author(s)

Michael Turewicz, `<michael.turewicz@rub.de>`

Examples

```
cwd <- system.file(package="PAA")
load(paste(cwd, "/extdata/Alzheimer.RData", sep=""))
#elist <- elist[elist$genes$Block < 10,]

#c1 <- paste(rep("AD",20), 1:20, sep="")
#c2 <- paste(rep("NDC",20), 1:20, sep="")

#pre.sel.results <- preselect(elist=elist, columns1=c1, columns2=c2, label1="AD",
# label2="NDC", discard.threshold=0.1, fold.thresh=1.9, discard.features=TRUE,
# method="tTest")
#elist <- elist[-pre.sel.results$discard,]

#selectFeatures.results <- selectFeatures(elist,n1=20,n2=20,label1="AD",
# label2="NDC",selection.method="rf.rfe",preselection.method="none",subruns=2,
# k=2,candidate.number=20,method="frequency")

load(paste(cwd, "/extdata/selectFeaturesResultsFreq.RData", sep=""))
plotFeatures(features=selectFeatures.results$features, elist=elist, n1=20,
 n2=20, group1="AD", group2="NDC")
```

`plotFeaturesHeatmap` *Plot feature intensities as a heatmap.*

Description

Plots intensities of given features as a heatmap.

Usage

```
plotFeaturesHeatmap(features = NULL, elist = NULL, n1 = NULL, n2 = NULL,
 output.path = NULL, description=FALSE)
```

Arguments

`features` vector containing the selected features as `"BRC"`-IDs (mandatory).

`elist` `EListRaw` or `EList` object containing all intensity data in log scale (mandatory).

`n1` integer indicating the sample size of group 1 (mandatory).

`n2` integer indicating the sample size of group 2 (mandatory).

`output.path` path for saving the heatmap as a tiff file (default: NULL).

`description` if TRUE, features will be described via protein names instead of uniprot accessions (default: FALSE).

Details

Plots intensities of all features given in the vector `features` via their corresponding `"BRC"`-IDs as a heatmap. If `description` is TRUE (default: FALSE), features will be described via protein names instead of uniprot accessions. Furthermore, if `output.path` is not NULL, the heatmap will be saved as a tiff file in `output.path`. This function can be used to check whether the selected features are differential.

Value

No value is returned.

Author(s)

Michael Turewicz, `<michael.turewicz@rub.de>`

Examples

```
cwd <- system.file(package="PAA")
load(paste(cwd, "/extdata/Alzheimer.RData", sep=""))
#elist <- elist[elist$genes$Block < 10,]

#c1 <- paste(rep("AD",20), 1:20, sep="")
#c2 <- paste(rep("NDC",20), 1:20, sep="")

#pre.sel.results <- preselect(elist=elist, columns1=c1, columns2=c2, label1="AD",
# label2="NDC", discard.threshold=0.1, fold.thresh=1.9, discard.features=TRUE,
# method="tTest")
#elist <- elist[-pre.sel.results$discard,]

#selectFeatures.results <- selectFeatures(elist,n1=20,n2=20,label1="AD",
# label2="NDC",selection.method="rf.rfe",preselection.method="none",subruns=2,
# k=2,candidate.number=20,method="frequency")

load(paste(cwd, "/extdata/selectFeaturesResultsFreq.RData", sep=""))
plotFeaturesHeatmap(features=selectFeatures.results$features, elist=elist,
 n1=20, n2=20, description=TRUE)
```

plotMAPlots *Check normalization results with MA plots.*

Description

Draws MA plots of raw data and data after all kinds of normalization provided by PAA.

Usage

```
plotMAPlots(elist = NULL, idx="all", include.rlm=FALSE, controls="internal",
output.path = NULL)
```

Arguments

`elist` `EListRaw` object containing raw data (mandatory).

`idx` integer indicating the column index of the sample for drawing MA plots or the string 'all' for drawing MA plots for all samples (default: all).

`include.rlm` boolean indicating whether RLM normalization should be included (for ProtoArrays only, deafault: FALSE).

`controls` sring indicating the ProtoArray controls for `rlm` normalization (optional). Valid options are `"internal"` (default), `"external"`, `"both"` or a regular expression defining a specific control or a specific set of controls.

`output.path` string indicating the folder where the tiff files will be saved (mandatory when idx='all').

Details

When `idx="all"` (default) for each microarray a tiff file containing MA plots for raw data, cyclicoess normalized data, quantile normalized data and vsn normalized data (and, optionally, for ProtoArrays, rlm normalized data) will be created. When `idx` is an integer indicating the column index of a particular sample, MA plots only for this sample will be created. For A and M value computation the artificial median array is used as reference signal. All figures can be saved in `output.path` (mandatory when `idx="all"`). The resulting MA plots can be used to compare the results of the different normalization methods.

Value

No value is returned.

Author(s)

Michael Turewicz, <michael.turewicz@rub.de>

Examples

```
cwd <- system.file(package="PAA")
load(paste(cwd, "/extdata/Alzheimer.RData", sep=""))
elist <- elist[elist$genes$Block == 1,]
plotMAPlots(elist=elist, idx=1)
```

plotNormMethods *Check normalization results with boxplots.*

Description

Draws boxplots (one boxplot per sample) of raw data and data after all kinds of normalization provided by PAA.

Usage

```
plotNormMethods(elist = NULL, include.rlm=FALSE, controls="internal",
output.path = NULL)
```

Arguments

`elist`	`EListRaw` object containing raw data (mandatory).
`include.rlm`	boolean indicating whether RLM normalization should be included (for ProtoArrays only, deafault: FALSE).
`controls`	sring indicating the ProtoArray controls for `rlm` normalization (optional). Valid options are `"internal"` (default), `"external"`, `"both"` or a regular expression defining a specific control or a specific set of controls.
`output.path`	string indicating a folder for saving the boxplots as tiff files (optional).

Details

For each normalization approach sample-wise boxplots are created. All boxplots can be saved as high-quality tiff files (when an output path has been specified via the argument `output.path`). The resulting boxplots can be used to compare the results of different normalization methods.

Value

No value is returned.

Author(s)

Michael Turewicz, <michael.turewicz@rub.de>

Examples

```
cwd <- system.file(package="PAA")
load(paste(cwd, "/extdata/Alzheimer.RData", sep=""))
elist <- elist[elist$genes$Block == 1,]
plotNormMethods(elist=elist)
```

`preselect` *Score and preselect features.*

Description

Iterates all features to score them via mMs, Student's t-test, or mRMR. Optionally, a list of not informative features can be obtained (for discarding them).

Usage

```
preselect(elist=NULL, columns1=NULL, columns2=NULL, label1="A", label2="B",
 discard.threshold=0.5, fold.thresh=1.5, discard.features=TRUE,
 mMs.above=1500, mMs.between=400, mMs.matrix1=NULL,
 mMs.matrix2=NULL, method=NULL)
```

Arguments

`elist` EListRaw or EList object (mandatory).

`columns1` column name vector (string vector) of group 1 (mandatory).

`columns2` column name vector (string vector) of group 2 (mandatory).

`label1` class label of group 1.

`label2` class label of group 2.

`discard.threshold` positive numeric between 0 and 1 indicating the maximum mMs or, respectively, the maximum t-test p-value for features to be included for further analysis. Default is "0.5".

`fold.thresh` numeric indicating the minimum fold change for features to be included for further analysis. Default is "1.5".

`discard.features` boolean indicating whether merely feature scores (i.e., mMs or t-test p-values) (="FALSE") or feature scores and a discard list "TRUE" should be returned. Default is "FALSE".

`mMs.above` mMs above parameter (integer). Default is "1500".

`mMs.between` mMs between parameter (integer). Default is "400".

`mMs.matrix1` pre-computed mMs reference matrix (see mMsMatrix()) for group 1 (mandatory).

`mMs.matrix2` pre-computed mMs reference matrix (see mMsMatrix()) for group 2 (mandatory).

`method` preselection method ("mMs", "tTest", "mrmr"). Default is "mMs".

Details

This function takes an `EListRaw` or `EList` object and group-specific column vectors. Furthermore, the class labels of group 1 and group 2 are needed. If `discard.features` is `"TRUE"` (default), all features that are considered as not differential will be collected and returned for discarding.

If `method = "mMs"`, additionally pre-computed mMs reference matrices (see `mMsMatrix()`) for group 1 and group 2 will be needed to compute mMs values (see Love B.) as scoring method. All mMs parameters (`mMs.above` and `mMs.between`) can be set. The defaults are `"1500"` for `mMs.above` and `"400"` for `mMs.between`. Features having an mMs value larger than `discard.threshold` (here: numeric between 0.0 and 1.0) or do not satisfy the minimal absolute fold change `fold.thresh` are considered as not differential.

If `method = "tTest"`, Student's t-test will be used as scoring method. Features having a p-value larger than `discard.threshold` (here: numeric between 0.0 and 1.0) or do not satisfy the minimal absolute fold change `fold.thresh` are considered as not differential.

If `method = "mrmr"`, mRMR scores for all features will be computed as scoring method (using the function `mRMR.classic()` of the CRAN R package `mRMRe`). Features that are not the `discard.threshold` (here: integer indicating a number of features) best features regarding their mRMR score are considered as not differential.

Value

If `discard.features` is `"FALSE"`: matrix containing metadata, feature scores and intensity values for the whole data set.

If `discard.features` is `"TRUE"`, a list containing:

`results`	matrix containing metadata, feature scores and intensity values for the whole data set.
`discard`	vector containing row indices (= features) for discarding features considered as not differential.

Author(s)

Michael Turewicz, `<michael.turewicz@rub.de>`

References

Love B: The Analysis of Protein Arrays. In: Functional Protein Microarrays in Drug Discovery. CRC Press; 2007: 381-402.

The software `"Prospector"` for ProtoArray analysis can be downloaded from Life Technologies web site (`http://www.lifetechnologies.com/`).

The R package mRMRe can be downloaded from CRAN. See also: De Jay N, Papillon-Cavanagh S, Olsen C, El-Hachem N, Bontempi G, Haibe-Kains B. mRMRe: an R package for parallelized mRMR ensemble feature selection. Bioinformatics 2013.

The package `limma` by Gordon Smyth et al. can be downloaded from Bioconductor (`http://www.bioconductor.org/`).

Smyth, G. K. (2005). Limma: linear models for microarray data. In: Bioinformatics and Computational Biology Solutions using R and Bioconductor, R. Gentleman, V. Carey, S. Dudoit, R. Irizarry, W. Huber (eds.), Springer, New York, pages 397-420.

Examples

```
cwd <- system.file(package="PAA")
load(paste(cwd, "/extdata/Alzheimer.RData", sep=""))
elist <- elist[elist$genes$Block < 10,]
c1 <- paste(rep("AD",20), 1:20, sep="")
c2 <- paste(rep("NDC",20), 1:20, sep="")
preselect(elist, columns1=c1, columns2=c2, label1="AD", label2="NDC",
 discard.threshold=0.5, fold.thresh=1.5, discard.features=TRUE, method="tTest")
```

printFeatures *Print biomarker candidate panel into a table.*

Description

Creates a table containing the selected biomarker candidate panel.

Usage

```
printFeatures(features = NULL, elist = NULL, output.path = NULL)
```

Arguments

features	vector containing the selected features as "BRC"-IDs (mandatory).
elist	EListRaw or EList object containing all intensity data in log scale (mandatory).
output.path	string indicating the folder where the table will be saved as a txt file (optional).

Details

Creates a table containing the selected biomarker candidate panel as well as additional information. When output.path is defined this table will be saved in a txt file ("candidates.txt").

Value

Table containing the selected biomarker panel.

Author(s)

Michael Turewicz, <michael.turewicz@rub.de>

Examples

```
cwd <- system.file(package="PAA")
load(paste(cwd, "/extdata/Alzheimer.RData", sep=""))
#elist <- elist[elist$genes$Block < 10,]

#c1 <- paste(rep("AD",20), 1:20, sep="")
#c2 <- paste(rep("NDC",20), 1:20, sep="")
```

```
#pre.sel.results <- preselect(elist=elist, columns1=c1, columns2=c2, label1="AD",
# label2="NDC", discard.threshold=0.1, fold.thresh=1.9, discard.features=TRUE,
# method="tTest")
#elist <- elist[-pre.sel.results$discard,]

#selectFeatures.results <- selectFeatures(elist,n1=20,n2=20,label1="AD",
# label2="NDC",selection.method="rf.rfe",preselection.method="none",subruns=2,
# k=2,candidate.number=20,method="frequency")

load(paste(cwd, "/extdata/selectFeaturesResultsFreq.RData", sep=""))
printFeatures(features=selectFeatures.results$features, elist=elist)
```

pvaluePlot *Draw a p-value plot.*

Description

Draws a p-value plot to visualize the p-values for all features stored in a `EList` or `EListRaw` object.

Usage

```
pvaluePlot(elist=NULL, group1=NULL, group2=NULL, method="tTest",
output.path=NULL, tag="", mMs.matrix1=NULL, mMs.matrix2=NULL, above=1500,
between=400, adjust=FALSE)
```

Arguments

`elist` `EList` or `EListRaw` object (mandatory).

`group1` vector of column names for group 1 (mandatory).

`group2` vector of column names for group 2 (mandatory).

`method` method for p-value computation: `"tTest"` or `"mMs"`. Default is `"tTest"`.

`output.path` string indicating a path for saving the plot (optional).

`tag` string that can be used for tagging the saved plot (optional).

`mMs.matrix1` pre-computed M score reference matrix (see `mMsMatrix()`) for group 1 (mandatory when method = "mMs").

`mMs.matrix2` pre-computed M score reference matrix (see `mMsMatrix()`) for group 2 (mandatory when method = "mMs").

`above` M score above parameter (integer). Default is `"1500"`.

`between` M score between parameter (integer). Default is `"400"`.

`adjust` Boolean indicating whether p-values should be adjusted. Default is `FALSE`.

Details

This function takes an `EList` or `EListRaw` object and the corresponding column name vectors to draw a plot of p-values for all features stored in `elist` (sorted in increasing order and in log2 scale). The p-value computation method (`"tTest"` or `"mMs"`) can be set via the argument `method`. Furthermore, when `adjust=TRUE` adjusted p-values (method: Benjamini & Hochberg, 1995, computed via `p.adjust()`) will be used. When an output path is defined (via `output.path`) the plot will be saved as a tiff file.

Value

No value is returned.

Author(s)

Michael Turewicz, `<michael.turewicz@rub.de>`

Examples

```
cwd <- system.file(package="PAA")
load(paste(cwd, "/extdata/Alzheimer.RData", sep=""))
elist <- elist[elist$genes$Block < 10,]
c1 <- paste(rep("AD",20), 1:20, sep="")
c2 <- paste(rep("NDC",20), 1:20, sep="")
pvaluePlot(elist=elist, group1=c1, group2=c2, method="tTest", tag="_tTest",
 adjust=FALSE)
```

`selectFeatures` *Select features using frequency-based or ensemble feature selection.*

Description

Performs a multivariate feature selection using frequency-based feature selection (based on RF-RFE, RJ-RFE or SVM-RFE) or ensemble feature selection (based on SVM-RFE).

Usage

```
selectFeatures(elist = NULL, n1 = NULL, n2 = NULL, label1 = "A", label2 = "B",
 cutoff = 10, selection.method = "rf.rfe", preselection.method = "mMs",
 subruns = 100, k = 10, subsamples = 10, bootstraps = 10,
 candidate.number = 300, above=1500, between=400,
 panel.selection.criterion="accuracy", importance.measure="MDA", ntree = 500,
 mtry = NULL, plot = FALSE, output.path = NULL, verbose = FALSE,
 method = "frequency")
```

Arguments

`elist` `EListRaw` or `EList` object containing all microarray data (mandatory).

`n1` integer indicating the sample number in group 1 (mandatory).

`n2` integer indicating the sample number in group 2 (mandatory).

`label1` class label of group 1 (default: "A").

`label2` class label of group 2 (default: "B").

`cutoff` integer indicating how many features will be selected (default: 10).

`selection.method`
string indicating the feature selection method: `"rf.rfe"` (default), `"svm.rfe"` or `"rj.rfe"`.

`preselection.method`
string indicating the feature pre-selection method: `"mMs"` (default), `"tTest"`, `"mrmr"` or `"none"`.

`subruns` integer indicating the number of re-sampling repeats to be performed (default: 100).

`k` integer indicating the number of k-fold cross validation subsets (default: 10, i.e., 10-fold CV).

`subsamples` integer indicating the number of subsamples for ensemble feature selection (default: 10).

`bootstraps` integer indicating the number of bootstrap samples for ensemble feature selection (default: 10).

`candidate.number`
integer indicating how many features shall be pre- selected. Default is `"300"`.

`above` mMs above parameter (integer). Default is `"1500"`.

`between` mMs between parameter (integer). Default is `"400"`.

`panel.selection.criterion`
string indicating the panel selection criterion: `"accuracy"` (default), `"sensitivity"` or `"specificity"`.

`importance.measure`
string indicating the random forest importance measure: `"MDA"` (default) or `"MDG"`.

`ntree` random forest parameter ntree (default: `"500"`).

`mtry` random forest parameter mtry (default: `sqrt(p)` where p is the number of predictors).

`plot` indicates whether performance plots shall be plotted (default: FALSE).

`output.path` string indicating the results output folder (optional).

`verbose` logical, indicating whether additional information shall be printed to the console (default: FALSE).

`method` the feature selection method: "frequency" (default) for frequency-based and "ensemble" for ensemble feature selection.

Details

This function takes an `EListRaw` or `EList` object, group-specific sample numbers, group labels and parameters choosing and setting up a univariate feature pre-selection method as well as a multivariate feature selection method (frequency-based or ensemble feature selection) to select a panel of differential features. When an output path is defined (via `output.path`) results will be saved on the hard disk and when `verbose` is TRUE additional information will be printed to the console.

Frequency-based feature selection (`method="frequency"`): The whole data is splitted in k cross validation training and test set pairs. For each training set a multivariate feature selection procedure is performed. The resulting k feature subsets are tested using the corresponding test sets (via classification). As a result, `selectFeatures()` returns the average k-fold cross validation classification accuracy as well as the selected feature panel (i.e., the union set of the k particular feature subsets). As multivariate feature selection methods random forest recursive feature elimination (RF-RFE), random jungle recursive feature elimination (RJ-RFE) and support vector machine recursive feature elimination (SVM-RFE) are supported. To reduce running times, optionally, univariate feature pre-selection can be performed (usage via `preselection.method`). As univariate pre-selection methods mMs `"mMs"`, Student's t-test `"tTest"` and mRMR `"mrmr"` are supported. Alternatively, no pre-selection can be chosen `"none"`. This approach is similar to the method proposed in Baek et al.

Ensemble feature selection (`method="ensemble"`): From the whole data the previously defined number of subsamples is drawn defining pairs of training and test sets. Moreover, for each training set a previously defined number of bootstrap samples is drawn. Then, for each bootstrap sample SVM-RFE is performed and a feature ranking is obtained. To obtain a final ranking for a particular training set, all associated bootstrap rankings are aggregated to a single ranking. To score the `cutoff` best features, for each subsample a classification of the test set is performed (using a svm trained with the `cutoff` best features from the training set) and the classification accuracy is determined. Finally, the stability of the subsample-specific panels is assessed (via Kuncheva index, Kuncheva LI, 2007), all subsample-specific rankings are aggregated, the top n features (defined by `cutoff`) are selected, the average classification accuracy is computed, and all these results are returned in a list. This approach has been proposed and is described in Abeel et al.

Value

If `method` is `"frequency"`, the results list contains the following elements:

`accuracy`	average k-fold cross validation accuracy.
`sensitivity`	average k-fold cross validation sensitivity.
`specificity`	average k-fold cross validation specificity.
`features`	selected feature panel.
`all.results`	complete cross validation results.

If `method` is `"ensemble"`, the results list contains the following elements:

`accuracy`	average accuracy regarding all subsamples.
`sensitivity`	average sensitivity regarding all subsamples.
`specificity`	average specificity regarding all subsamples.
`features`	selected feature panel.
`all.results`	all feature ranking results.

`stability`	stability of the feature panel (i.e., Kuncheva index for the subrun-specific panels).

Author(s)

Michael Turewicz, `<michael.turewicz@rub.de>`

References

Baek S, Tsai CA, Chen JJ.: Development of biomarker classifiers from high- dimensional data. Brief Bioinform. 2009 Sep;10(5):537-46.

Abeel T, Helleputte T, Van de Peer Y, Dupont P, Saeys Y: Robust biomarker identification for cancer diagnosis with ensemble feature selection methods. Bioinformatics. 2010 Feb 1;26(3):392-8.

Kuncheva, LI: A stability index for feature selection. Proceedings of the IASTED International Conference on Artificial Intelligence and Applications. February 12-14, 2007. Pages: 390-395.

Examples

```
cwd <- system.file(package="PAA")
load(paste(cwd, "/extdata/Alzheimer.RData", sep=""))
elist <- elist[elist$genes$Block < 10,]

c1 <- paste(rep("AD",20), 1:20, sep="")
c2 <- paste(rep("NDC",20), 1:20, sep="")

pre.sel.results <- preselect(elist=elist, columns1=c1, columns2=c2, label1="AD",
 label2="NDC", discard.threshold=0.1, fold.thresh=1.9, discard.features=TRUE,
 method="tTest")
elist <- elist[-pre.sel.results$discard,]

selectFeatures.results <- selectFeatures(elist,n1=20,n2=20,label1="AD",
    label2="NDC",selection.method="svm.rfe",subsamples=2,bootstraps=1,
    candidate.number=20,method="ensemble")
```

`shuffleData` *Shuffles class labels to obtain random groups.*

Description

Shuffles class labels of an `EList` or `EListRaw` object randomly to obtain two random groups (e.g. "A" and "B").

Usage

```
shuffleData(elist=NULL, n1=NULL, n2=NULL, label1="A", label2="B")
```

Arguments

`elist` EList or EListRaw object (mandatory).

`n1` sample size of random group 1 (mandatory).

`n2` sample size of random group 2 (mandatory).

`label1` class label of random group 1 (default: "A").

`label2` class label of random group 2 (default: "B").

Details

Shuffles class labels of an EList or EListRaw object randomly to obtain two random groups (e.g. "A" and "B").

Value

EList or EListRaw object with random groups.

Author(s)

Michael Turewicz, <michael.turewicz@rub.de>

Examples

```
cwd <- system.file(package="PAA")
load(paste(cwd, "/extdata/Alzheimer.RData", sep=""))
shuffleData(elist=elist, n1=20, n2=20, label1="A", label2="B")
```

---

volcanoPlot *Draw a volcano plot.*

Description

Draws a volcano plot to visualize differential features.

Usage

```
volcanoPlot(elist=NULL, group1=NULL, group2=NULL, method="tTest", p.thresh=NULL,
fold.thresh=NULL, output.path=NULL, tag="", mMs.matrix1=NULL, mMs.matrix2=NULL,
above=1500, between=400)
```

Arguments

`elist` `EList` or `EListRaw` object (mandatory).

`group1` vector of column names for group 1 (mandatory).

`group2` vector of column names for group 2 (mandatory).

`method` method for p-value computation: `"tTest"` or `"mMs"`. Default is `"tTest"`.

`p.thresh` positive float number between 0 and 1 indicating the maximum p-value for features to be considered as differential (e.g., `"0.5"`). This argument is optional.

`fold.thresh` float number indicating the minimum fold change for features to be considered as differential (e.g., `"1.5"`). This argument is optional.

`output.path` string indicating a path for saving the plot (optional).

`tag` string that can be used for tagging the saved plot (optional).

`mMs.matrix1` a pre-computed M score reference matrix (see `mMsMatrix()`) for group 1 (mandatory when method = "mMs").

`mMs.matrix2` a pre-computed M score reference matrix (see `mMsMatrix()`) for group 2 (mandatory when method = "mMs").

`above` M score above parameter (integer). Default is `"1500"`.

`between` M score between parameter (integer). Default is `"400"`.

Details

This function takes an `EList` or `EListRaw` object and the corresponding column name vectors to draw a volcano plot. To visualize differential features, thresholds for p-values and fold changes can be defined. Furthermore, the p-value computation method (`"mMs"` or `"tTest"`) can be set. When an output path is defined (via `output.path`) the plot will be saved as a tiff file.

Value

No value is returned.

Author(s)

Michael Turewicz, `<michael.turewicz@rub.de>`

Examples

```
cwd <- system.file(package="PAA")
load(paste(cwd, "/extdata/Alzheimer.RData", sep=""))
elist <- elist[elist$genes$Block < 10,]
c1 <- paste(rep("AD",20), 1:20, sep="")
c2 <- paste(rep("NDC",20), 1:20, sep="")
volcanoPlot(elist=elist, group1=c1, group2=c2, method="tTest", p.thresh=0.01,
 fold.thresh=2)
```

Index