Package 'ArrayAnalysis'

July 22, 2021

Type Package

Title ArrayAnalysis

Version 1.0.0

Author Lars Eijssen <l.eijssen@maastrichtuniveristy.nl>

Maintainer Ammar Ammar <a.ammar@maastrichtuniversity.nl>,
Vishnu Karthik <vishnukarthik211@gmail.com>

Description A library for quality control and pre-processing analysis
of Affymetrix gene expression results by extending,
integrating and harmonizing functionality of Bioconductor packages.

Depends R (>= 3.6.0)

License Apache License (>= 2)

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

biocViews Software, AssayDomain, Transcription, GeneExpression, QualityControl

Imports Biobase, affy, affyPLM, affyQCReport, bioDist, biomaRt,
gcrma, gplots, plier, simpleaffy, yaqcaffy, grDevices, graphics, stats, utils, methods, gdata, R2HTML, limma

Suggests knitr

VignetteBuilder knitr

R topics documented:

`addStandardCDFenv` *Check that a cdf environment is loaded*

Description

This function (from functions_processing) makes sure that a cdf environment is loaded for the current chiptype. In some cases a cdf environment will already be available after reading the data with the ReadAffy function, then the function will detect this and return the object as is (unless overwrite is set to TRUE). In case no cdf environment has been assigned, it will try to search for a suitable one, and add this if found. If no suitable cdf can be found, a warning will be generated and the object returned as is.

Usage

```
addStandardCDFenv(Data, overwrite = FALSE)
```

Arguments

Data	(Status: required) The raw data object (datatype: AffyBatch)
overwrite	(Status: optional, Default: FALSE) Should the cdfName be overwritten if there is already a value assigned to the object passed to the function (datatype: logical)

Value

The object with a cdf annotation assigned if found (datatype: AffyBatch)

Examples

```
#rawData <- addStandardCDFenv(rawData)
```

---

addUpdatedCDFenv *Load an updated cdf environment from BrainArray*

Description

This function (from functions_processing) tries to find and load an updated cdf environment from BrainArray for the current chiptype. If this is not successful, a warning will be generated and the raw data object returned as is, i.e. with the cdf annotation that it already had, if any. Note that the IDs given to the reporters are artificially created by adding "_at" as a postfix to the ID from the entry in the database that the probeset corresponds to (c.f. also createNormDataTable). Note also that reannotation takes places at a probe level, not at a probeset level. This means that completely new probesets are constructed, not necessarily equal in size.

Usage

```
addUpdatedCDFenv(Data, species = NULL, type = "ENSG")
```

Arguments

Data	(Status: required) The raw data object (datatype: AffyBatch)
species	(Status: required, Default: NULL) The species associated with the chip type (datatype: character). Possible values: - Abbreviated: "Ag", "At", "Bt", "Ce", "Cf", "Dr", "Dm", "Gg", "Hs", "MAmu", "Mm", "Os", "Rn", "Sc", "Sp", "Ss" - Full names: "Anopheles gambiae", "Arabidopsis thaliana", "Bos taurus", "Caenorhabditis elegans", "Canis familiaris", "Danio rerio", "Drosophila melanogaster", "Gallus gallus", "Homo sapiens", "Macaca mulatta", "Mus musculus", "Oryza sativa", "Rattus norvegicus", "Saccharomyces cerevisiae", "Schizosaccharomyces pombe", "Sus scrofa".
type	(Status: optional, Default: "ENSG") The type of custom cdf requested This parameter indicates the database based on which an updated cdf environment should be selected (datatype: character). Possible values are: "ENTREZG", "REFSEQ", "ENSG", "ENSE", "ENST", "VEGAG", "VEGAE", "VEGAT", "TAIRG", "TAIRT", "UG", "MIRBASEF", "MIRBASEG".

Value

The object (of class AffyBatch) with a updated cdf annotation assigned if found

Examples

```
#By default, the script will call, if customCDF is TRUE
#(from within the \link[ArrayAnalysis]{normalizeData}
#and \link[ArrayAnalysis]{computePMAtable} functions,
#leaving the original Data intact and using Data.copy to further process):
#Data.copy <- Data
#Data.copy <- addUpdatedCDFenv(Data.copy, species, CDFtype)
```

`array.image` *Create several types of spatial images*

Description

This function (from functions_imagesQC.R) creates several types of spatial images. It does not make use of a PLMset object and is called when the PLM images cannot be computed due to time or memory constaints. Also, it offers more flexibility. By default, it creates a relative intensity plot versus the median array on a blue to red colour scale. Note that the median array will always be computed based on the complete data set, also when plots for a subset of arrays are requested. If the median array has to be computed on subsets (e.g. on experimental group), a data object with only those arrays can be provided to the function (without setting the arrays parameter). The color ranges will saturate at pcut percentage(s) of the data range. The color ranges can be modified by tuning col.mod. By default, for the relative plots, the arrays are first balanced for their overall intensity and a symmetrical color range is used. When there is less than 6 arrays, the median array is not used. Intensities are plotted using a virtual symetric color scale, from blue to red.

Usage

```
array.image(
  Data,
  pcut = NULL,
  relative = TRUE,
  symm = relative,
  balance = relative,
  quantitative = relative,
  col.mod = 1,
  postfix = "",
  arrays = NULL,
  WIDTH = 1000,
  HEIGHT = 1414,
  POINTSIZE = 24
)
```

Arguments

`Data`	(Status: required) The raw data object (datatype: AffyBatch or ExpressionSet)
`pcut`	(Status: optional, Default:NULL. This means that for relative plots a value of 0.001 will be used, and for other plots values of c(0.01, 0.05).) Either a numeric value or a vector of two numeric values, both in the interval [0, 0.5], used a color saturation limits as a percentage of the data range. If one value is provided, this is taken both a lower and upper saturation limit. (datatype: numeric)

relative	(Status: optional, Default:TRUE) Is the plot to be created a plot of each array relative to the median array, or not (i.e. an absolute plot)? (datatype: logical)
symm	(Status: optional, Default: TRUE if "relative" is TRUE, FALSE otherwise.) Should a symmetric color scale be used? (datatype: logical)
balance	(Status: optional, Default:Default: TRUE if "relative" is TRUE, FALSE otherwise.) Should the arrays first be balanced for their average intensities? (datatype: logical)
quantitative	(Status: optional, Default:TRUE if "relative"is TRUE, FALSE otherwise. Setting "quantitative" to TRUE has no effect if "relative" is FALSE, a warning will be produced.) Should the plot be quantitative or qualitative (i.e. only indicate the sign of the value)? (datatype: logical)
col.mod	(Status: optional, Default:1) A numeric value used as a modifier for the color range. A value of 1 means no modification (linear), smaller leads to faster saturation, larger to slower saturation. (datatype: numeric)
postfix	(Status: optional, Default:"") String to be attached to the file names produced. (datatype: character)
arrays	(Status: optional, Default:NULL (in which case all arrays are plotted)) Which arrays are to be plotted. NOTE: for relative plot types, the median array is still computed using all arrays in the dataset. (datatype: numeric)
WIDTH	(Status: optional, Default:1000) png image width (datatype: number)
HEIGHT	(Status: optional, Default:1414) png image height (datatype: number)
POINTSIZE	(Status: optional, Default:24) png image point size (datatype: number)

Value

Images containing each of the requested plots, one file per six arrays. Naming is "virtual_array_plots" followed by the postfix given by the user.

Examples

```
# There are two default calls by the script:
# 1/ relative intensity plot versus the median array on a blue to red colour scale:
# array.image(rawData)
# 2/ absolute intensity plot when there are less than 6 arrays in the dataset:
# array.image(rawData,relative=FALSE,col.mod=4,symm=TRUE)
# Other calls could be made, such as:
# spatial intensity plot on a virtual colour scale (red to yellow)
# array.image(rawData,relative=FALSE)
# signs of the relative intensities versus the median array
# (e.g. lower or higher) with blue as lower and red as higher.
# array.image(rawData,quantitative=FALSE)
# similar to the default plot, but not balanced within arrays
# array.image(rawData,balance=FALSE)
# or: per experimental group
# for (i in levels(experimentFactor)) {
# array.image(rawData[, experimentFactor == i], postfix = paste("_",i), balance=FALSE)
# }
```

backgroundPlot *Create an image of the background intensities and deviations*

Description

This function (from functions_imagesQC.R) creates an image of the background intensities and deviations for each array based on the qc (simpleaffy Bioconductor package) function, which generally only work for chiptypes with perfect match and mismatch probes, but even not for all of those. As such, when these statistics are not provided as parameters, trys are used in this function to compute them internally. Values for which the try fails are not computed, but the script will continue after giving a warning.

Usage

```
backgroundPlot(
  Data,
  quality = NULL,
  experimentFactor = NULL,
  plotColors = NULL,
  legendColors = NULL,
  WIDTH = 1000,
  HEIGHT = 1414,
  POINTSIZE = 24,
  MAXARRAY = 41
)
```

Arguments

Data	(Status: required) The raw data object (datatype: AffyBatch)
quality	(Status: optional, Default:NULL) object obtained by calling the qc function (simpleaffy).When not provided, it is computed within the function.(datatype: QCStats)
experimentFactor	(Status: required, Default:NULL) The factor of groups. (datatype: factor)
plotColors	(Status: required, Default:NULL) Vector of colors assigned to each array. (datatype: character)
legendColors	(Status: required, Default:NULL) Vector of colors assigned to each experimental group. (datatype: character)
WIDTH	(Status: optional, Default:1000) png image width (datatype: number)
HEIGHT	(Status: optional, Default:1414) png image height (datatype: number)
POINTSIZE	(Status: optional, Default:24) png image point size (datatype: number)
MAXARRAY	(Status: optional, Default:41) threshold to adapt the image to the number of arrays (datatype: number)

Value

A PNG image of background intensities, called "RawDataBackgroundPlot"

Examples

```
# backgroundPlot(rawData, quality=quality, experimentFactor, plotColors,legendColors)
```

---

boxplotFun *Create an image with intensity boxplots for all arrays*

Description

This function (from functions_imagesQC.R) creates an image with intensity boxplots for all arrays in the raw or normalized dataset (depending on the object passed). It calls the boxplot function applied to an AffyBatch object.

Usage

```
boxplotFun(
  Data,
  experimentFactor = NULL,
  plotColors = NULL,
  legendColors = NULL,
  normMeth = "",
  WIDTH = 1000,
  HEIGHT = 1414,
  POINTSIZE = 24,
  MAXARRAY = 41
)
```

Arguments

Data	(Status: required) The raw data object (datatype: AffyBatch)
experimentFactor	(Status: required, Default:NULL) The factor of groups. (datatype: factor)
plotColors	(Status: required, Default:NULL) Vector of colors assigned to each array. (datatype: character)
legendColors	(Status: required, Default:NULL) Vector of colors assigned to each experimental group. (datatype: character)
normMeth	(Status: required when Data is a normalized data object, Default:"") String indicating the normalization method used (see normalizeData function for more information on the possible values). (datatype: character)
WIDTH	(Status: optional, Default:1000) png image width (datatype: number)
HEIGHT	(Status: optional, Default:1414) png image height (datatype: number)
POINTSIZE	(Status: optional, Default:24) png image point size (datatype: number)
MAXARRAY	(Status: optional, Default:41) threshold to adapt the image to the number of arrays (datatype: number)

Value

A PNG image of the raw or normalized boxplots of the arrays, called 'DataBoxplot'

Examples

```
# By default, before the normalization the script will call:
# boxplotFun(Data=rawData, experimentFactor, plotColors, legendColors)
# and after normalization:
# boxplotFun(Data=normData, experimentFactor, plotColors, legendColors, normMeth=normMeth)
```

---

clusterFun *Create a hierarchical clustering dendrogam*

Description

This function (from functions_imagesQC.R) creates a hierarchical clustering dendrogam of the arrays in the raw or normalized dataset (depending on the object passed). When the dataset consists of less than three arrays, no dendrogram is generated and a warning is given.

Usage

```
clusterFun(
  Data,
  experimentFactor = NULL,
  clusterOption1 = "pearson",
  clusterOption2 = "ward.D2",
  normMeth = "",
  plotColors = NULL,
  legendColors = NULL,
  plotSymbols = NULL,
  legendSymbols = NULL,
  WIDTH = 1000,
  HEIGHT = 1414,
  POINTSIZE = 24,
  MAXARRAY = 41
)
```

Arguments

Data	(Status: required) The raw data object (datatype: AffyBatch)
experimentFactor	(Status: required, Default:NULL) The factor of groups. (datatype: factor)
clusterOption1	(Status: optional, Default:"Pearson") String indicating the distance function to be used. Possible values are "pearson", "spearman", or "euclidean". (datatype: character)
clusterOption2	(Status: optional, Default:"ward") String indicating the hierarchical clustering function to be used. Possible values are "ward", "single", "complete", "average", "mcquitty", "median" or "centroid". (datatype: character)
normMeth	(Status: required when Data is a normalized data object, Default:"") String indicating the normalization method used (see normalizeData function for more information on the possible values). (datatype: character)
plotColors	(Status: required, Default:NULL) Vector of colors assigned to each array. (datatype: character)

legendColors	(Status: required, Default:NULL) Vector of colors assigned to each experimental group. (datatype: character)
plotSymbols	(Status: required, Default:NULL) symbol assigned to each array. (datatype: number)
legendSymbols	(Status: required, Default:NULL) symbol assigned to each experimental group. (datatype: number)
WIDTH	(Status: optional, Default:1000) png image width (datatype: number)
HEIGHT	(Status: optional, Default:1414) png image height (datatype: number)
POINTSIZE	(Status: optional, Default:24) png image point size (datatype: number)
MAXARRAY	(Status: optional, Default:41) threshold to adapt the image to the number of arrays (datatype: number)

Value

A PNG image with the clustering dendrogram of the arrays. Naming is 'DataCluster' followed by the name of the distance function used

Examples

```
# By default, before the normalization the script will call:
# clusterFun(Data=rawData, clusterOption1=clusterOption1,
# clusterOption2=clusterOption2)
# and after normalization:
# clusterFun(Data=normData, clusterOption1=clusterOption1,
# clusterOption2=clusterOption2, normMeth=normMeth)
```

---

colorsByFactor *Create colors for the plots and the legends*

Description

This function (from functions_processing) creates a list with two elements: a vector of colors, one for each array and a vector of one representative color for each group within the experiment, for use in legends. The colors are based on groups present in the dataset (as provided by the user in experimentFactor). If there is only one group, colors are chosen randomly over the rainbow palette. Otherwise arrays belonging to the same group get different shades of a similar color.

Usage

```
colorsByFactor(experimentFactor)
```

Arguments

experimentFactor	(Status: required) The factor of groups (datatype: factor)

Value

A list with two fields: plotColors contains a vector of colors, one for each array; legendColors contains a representative color for each group, for use in legends (datatype: list)

Examples

```
#By default, the script will call:
#colList <- colorsByFactor(experimentFactor)
#plotColors <- colList$plotColors
#legendColors <- colList$legendColors
#rm(colList)
```

`computeAdvancedStatistics` *Compute more advanced statistics*

Description

Compute more advanced statistics

Usage

```
computeAdvancedStatistics(
  normDataTable,
  descriptionFile,
  covariates_string,
  interaction_string,
  paired_string,
  plotVolcanoPlot,
  plotVennPlot,
  matfileName = NULL,
  keepAnnotation = FALSE,
  defaultContr = TRUE
)
```

Arguments

`normDataTable`	normalized data object (Status=required)
`descriptionFile`	(Status: required) The data.frame containing the description file information (column 1: file names; column 2: names to be used in the plots; column 3: experimental groups the samples belong to)(datatype: data.frame)
`covariates_string`	(Status=required)
`interaction_string`	(Status=required)
`paired_string`	(Status=required)
`plotVolcanoPlot`	(Status=required)
`plotVennPlot`	(Status=required)
`matfileName`	(datatype=character, Default=NULL)
`keepAnnotation`	(datatype=logical, Default=FALSE)
`defaultContr`	(datatype=logical, Default=TRUE)

Value

the contrast matrix file

Examples

```
#example here
```

`computePMAtable` *Prepare and return a table of PMA calls*

Description

This function (from functions_processing) computes a table of Absent (A), Marginal (M), Present (P) calls based on the MAS5 function (affy package). This function will only work for chiptypes that have mismatch probes and for which the detection.p.val (simpleaffy Bioconductor package) function that it calls works. A try construction will be used and if no table can be created a warning is given. Note that this function will always use the MAS5 algorithm, regardless of the normalization method used in the normalizeData function. In case customCDF is TRUE, annotation is updated using BrainArray custom cdf environments, before proceeding with the normalization (and summarisation of probe expressions into probeset expressions). To update the annotation, a sub call is made to the addUpdatedCDFenv function.

Usage

```
computePMAtable(Data, customCDF = TRUE, species = NULL, CDFtype = NULL)
```

Arguments

`Data`	(Status: required) The raw data object (datatype: AffyBatch)
`customCDF`	(Status: optional, Default: TRUE) Should annotation of the chip be updated before normalizing the data (and building the probesets out of the separate probes)? If requested, this is done using BrainArray updated cdf environments, c.f. addUpdatedCDFenv (datatype: logical)
`species`	(Status: required when customCDF is TRUE, c.f. addUpdatedCDFenv, Default: NULL)The species associated with the chip type. (datatype: character)
`CDFtype`	(Status: required when customCDF is TRUE, c.f. addUpdatedCDFenv, Default: NULL) The type of custom cdf requested. (datatype: character)

Value

A data.frame table called 'PMAtable' containing the PMA values for each probeset and array

Examples

```
#By default, the script will call, if customCDF is TRUE:
#PMAtable <- computePMAtable(rawData,customCDF,species,CDFtype)
#or, if customCDF is FALSE:
#PMAtable <- computePMAtable(rawData,customCDF)
#After this call, the main script will save the result
#to a tab-delimited text file, called PMAtable.txt
```

computeStatistics *Compute the statistics*

Description

Compute the statistics

Usage

```
computeStatistics(
  normDataTable,
  descriptionFile,
  defaultContr = TRUE,
  matfileName = NULL,
  keepAnnotation = FALSE
)
```

Arguments

normDataTable	normalized data object (Status=required)
descriptionFile	(Status: required) The data.frame containing the description file information (column 1: file names; column 2: names to be used in the plots; column 3: experimental groups the samples belong to)(datatype: data.frame)
defaultContr	(datatype=logical, Default=TRUE)
matfileName	(datatype=character, Default=NULL)
keepAnnotation	(datatype=logical, Default=FALSE)

Value

the contrast matrix file

Examples

```
#example here
```

controlPlots *Create an image of the scale factors of the array*

Description

This function (from functions_imagesQC.R) creates images of the expression values of the affx controls, if present. One image shows the expression profiles over all samples, for each control separately. The other image shows the boxplots of all controls (together), for each sample.

Usage

```
controlPlots(
  Data,
  plotColors = NULL,
  experimentFactor = NULL,
  legendColors = NULL,
  affxplots = TRUE,
  boxplots = TRUE,
  WIDTH = 1000,
  HEIGHT = 1414,
  POINTSIZE = 24,
  MAXARRAY = 41
)
```

Arguments

Data	(Status: required) The raw data object (datatype: AffyBatch)
plotColors	(Status: required, Default:NULL) Vector of colors assigned to each array. (datatype: character)
experimentFactor	(Status: required, Default:NULL) The factor of groups. (datatype: factor)
legendColors	(Status: required, Default:NULL) Vector of colors assigned to each experimental group. (datatype: character)
affxplots	(Status: optional, Default:TRUE) Does the AFFX expression profiles have to be plotted? (datatype: logical)
boxplots	(Status: optional, Default:TRUE) Does the AFFX and other controls boxplots have to be plotted? (datatype: logical)
WIDTH	(Status: optional, Default:1000) png image width (datatype: number)
HEIGHT	(Status: optional, Default:1414) png image height (datatype: number)
POINTSIZE	(Status: optional, Default:24) png image point size (datatype: number)
MAXARRAY	(Status: optional, Default:41) threshold to adapt the image to the number of arrays (datatype: number)

Value

Two images. A PNG image of the expression profiles of the affx controls, called "Profiles_affx_controls" A PNG image of the boxplots of the affx controls, called "Boxplots_affx_controls"

Examples

```
# controlPlots(rawData,plotColors)
```

correlFun *Create an image of the intensity correlation values*

Description

This function (from functions_imagesQC.R) creates an image of the intensity correlation values of the arrays in the raw or normalized dataset (depending on the object passed). It calls correlationPlot (affyQCReport Bioconductor package).

Usage

```
correlFun(
  Data,
  clusterOption1 = "pearson",
  clusterOption2 = "ward.D2",
  normMeth = "",
  experimentFactor = NULL,
  legendColors = NULL,
  WIDTH = 1000,
  HEIGHT = 1414,
  POINTSIZE = 24,
  MAXARRAY = 41
)
```

Arguments

Data	(Status: required) The raw data object (datatype: AffyBatch)
clusterOption1	(Status: optional, Default:"Pearson") String indicating the distance function to be used. Possible values are "pearson", "spearman", or "euclidean". (datatype: character)
clusterOption2	(Status: optional, Default:"ward") String indicating the hierarchical clustering function to be used. Possible values are "ward", "single", "complete", "average", "mcquitty", "median" or "centroid". (datatype: character)
normMeth	(Status: required when Data is a normalized data object, Default:"") String indicating the normalization method used (see normalizeData function for more information on the possible values). (datatype: character)
experimentFactor	(Status: required, Default:NULL) The factor of groups. (datatype: factor)
legendColors	(Status: required, Default:NULL) Vector of colors assigned to each experimental group. (datatype: character)
WIDTH	(Status: optional, Default:1000) png image width (datatype: number)
HEIGHT	(Status: optional, Default:1414) png image height (datatype: number)
POINTSIZE	(Status: optional, Default:24) png image point size (datatype: number)
MAXARRAY	(Status: optional, Default:41) threshold to adapt the image to the number of arrays (datatype: number)

Value

A PNG image of the intensity correlation values of the arrays, called "DataArrayCorrelationPlot"

Examples

```
# By default, before the normalization the script will call:
# correlFun(Data=rawData)
# and after normalization:
# correlFun(Data=normData, normMeth=normMeth)
```

---

coverAndKeyPlot *Plot a cover sheet*

Description

This function (from functions_imagesQC.R) plots a cover sheet, and one or more key sheet indicating the links between CEL file names, array names used in the plots, and to which experimental group they belong based on the description file provided by the user, which has been loaded into the description variable earlier in the main script, passed as an argument (arrayGroup). For the key sheets, one sheet will be created for every 35 arrays in the experiment.

Usage

```
coverAndKeyPlot(description = NULL, refName = "", WIDTH = 1000, HEIGHT = 1414)
```

Arguments

description	(Status: required) The data.frame containing the description file information (column 1: file names; column 2: names to be used in the plots; column 3: experimental groups the samples belong to)(datatype: data.frame)
refName	(Status: optional, Default:"") dataset name. It is deduced from the name of the zip file containing the CEL files when used from arrayanalysis.org or as a GenePattern module.(datatype: character)
WIDTH	(Status: optional, Default:1000) png image width (datatype: number)
HEIGHT	(Status: optional, Default:1414) png image height (datatype: number)

Value

Two files, a file called 'Cover_1' that can be used as an opening image. And, file(s) called 'Description', if needed followed by an index number,that represent(s) the information as given in the input parameter.

Examples

```
# By default, the script will call:
# coverAndKeyPlot(description)
```

`createCutOffTab` *For each comparison, create a cutoff table (P-value, LogFC and Average expression)*

Description

For each comparison, create a cutoff table (P-value, LogFC and Average expression)

Usage

```
createCutOffTab(
  files,
  cutOffPval = NULL,
  cutOfflogFC = NULL,
  cutOffAveExpr = NULL
)
```

Arguments

`files`	(Status=required) vector of fit-model file names
`cutOffPval`	(default=NULL)
`cutOfflogFC`	(default=NULL)
`cutOffAveExpr`	(default=NULL)

Value

several files (for each comparison)

Examples

```
#example here
```

`createFCHist` *For each comparison table, create a Fold Change histogram plot*

Description

For each comparison table, create a Fold Change histogram plot

Usage

```
createFCHist(files)
```

Arguments

`files`	(Status=required) vector of fit-model file names

Value

a PNG file for the histogram plot

Examples

```
#example here
```

`createNormDataTable` *Prepare and export the normalized data table*

Description

This function (from functions_processing) prepares a table suitable for visualisation and saving to disk from a normalized data object. In case of use of an updated cdf annotation (customCDF is TRUE), it will remove the artificial "_at" postfixes from all probeset IDs, apart from the affx controls, in order to get the real IDs from the database used to create the update. Also, in case an updated cdf environment based on Ensemble Gene ID ("ENSG") has been used, the function tries to connect to BioMart (using the biomaRt BioConductor package) in order to add two extra columns of information to the table: the common gene name, and a gene description. This will (for now) not be done for other updated cdf types, as there is no one-to-one mapping between BioMart entries and these IDs or it has not been sufficiently tested. Note that the BioMart connection may sometimes not be established (e.g. if the service is down or busy), it may in such cases be worthwhile to try again.

Usage

```
createNormDataTable(normData, customCDF = NULL, species = NULL, CDFtype = NULL)
```

Arguments

`normData`	(Status: required) The normalized data object (datatype: ExpressionSet)
`customCDF`	(Status: optional, Default: TRUE) Should annotation of the chip be updated before normalizing the data (and building the probesets out of the separate probes)? If requested, this is done using BrainArray updated cdf environments, c.f. addUpdatedCDFenv (datatype: logical)
`species`	(Status: required when customCDF is TRUE, c.f. addUpdatedCDFenv, Default: NULL)The species associated with the chip type. (datatype: character)
`CDFtype`	(Status: required when customCDF is TRUE, c.f. addUpdatedCDFenv, Default: NULL) The type of custom cdf requested. (datatype: character)

Value

A data.frame table with normalized data, and possible extra annotation.

Examples

```
#By default, the script will call:
#normDataTable <- createNormDataTable(normData, customCDF, species, CDFtype)
#After this function has been called, the normDataTable object is
#saved to a tab-delimited text file by the main script.
```

createPvalHist *For each comparison table, create a P-value histogram plot*

Description

For each comparison table, create a P-value histogram plot

Usage

```
createPvalHist(files)
```

Arguments

files	(Status=required) vector of fit-model file names

Value

a PNG file for the histogram plot

Examples

```
#example here
```

createPvalTab *Create significant pvalue table*

Description

Create significant pvalue table

Usage

```
createPvalTab(
  files,
  pvaluelist = c(0.001, 0.01, 0.05, 0.1),
  adjpvaluelist = 0.05,
  foldchangelist = c(1.1, 1.5, 3),
  html = FALSE
)
```

Arguments

files	(Status=required) vector of fit-model file names
pvaluelist	(datatype=vector, Default=c(0.001,0.01,0.05, 0.1))
adjpvaluelist	(datatype=numeric, Default=0.05)
foldchangelist	(datatype=vector, Default=c(1.1,1.5,3))
html	(datatype=logical, Default=FALSE)

Value

a summary table file

Examples

```
#example here
```

---

createVennPlot *create a venn diagram plot*

Description

create a venn diagram plot

Usage

```
createVennPlot(fit)
```

Arguments

fit	(Status=required) fit object

Value

a PNG file for the plot

Examples

```
#example here
```

---

createVolcanoPlot *create a volcano plot*

Description

create a volcano plot

Usage

```
createVolcanoPlot(fit)
```

Arguments

fit	(Status=required) fit object

Value

a PNG file for the plot

Examples

```
#example here
```

deduceSpecies *Find the species of the chiptype*

Description

This function (from functions_processing) tries to determine the species related to the current chiptype, if the species has not been provided by the user (and as such is set to ""). If the descr parameter is not provided or is empty, an empty string is returned as species. In other cases the function tries to load an annotation library depending on the chiptype to find the species. If not successful, it will be set by hand for some predefined chiptypes. If still not successful, the empty string is returned.

Usage

```
deduceSpecies(descr = NULL)
```

Arguments

descr	(Status: required, Default: NULL) A string indicating the chiptype, which can be obtained by getting the @annotation slot from an AffyBatch object (datatype: character)

Value

The species associated with the current chiptype, or "" if detection was unsuccessful (datatype: character)

Examples

```
#species <- deduceSpecies(rawData@annotation)
```

defaultMatrix *Default comparison (for less than 5 experimental factors)*

Description

Default comparison (for less than 5 experimental factors)

Usage

```
defaultMatrix(design)
```

Arguments

design	(Status=required)

Value

a dataframe for the contrast matrix

Examples

```
#example here
```

densityFun *Create an image with a density curve of the intensities*

Description

This function (from functions_imagesQC.R) creates an image with a density curve of the intensities for all arrays in the raw or normalized dataset (depending on the object passed).

Usage

```
densityFun(
  Data,
  plotColors = NULL,
  normMeth = "",
  WIDTH = 1000,
  HEIGHT = 1414,
  POINTSIZE = 24,
  MAXARRAY = 41
)
```

Arguments

Data	(Status: required) The raw data object (datatype: AffyBatch)
plotColors	(Status: required, Default:NULL) Vector of colors assigned to each array. (datatype: character)
normMeth	(Status: required when Data is a normalized data object, Default:"") String indicating the normalization method used (see normalizeData function for more information on the possible values). (datatype: character)
WIDTH	(Status: optional, Default:1000) png image width (datatype: number)
HEIGHT	(Status: optional, Default:1414) png image height (datatype: number)
POINTSIZE	(Status: optional, Default:24) png image point size (datatype: number)
MAXARRAY	(Status: optional, Default:41) threshold to adapt the image to the number of arrays (datatype: number)

Value

A PNG image of the raw or normalized density plots of the arrays, called "DensityHistogram"

Examples

```
# By default, before the normalization the script will call:
# densityFun(Data=rawData, plotColors)
# and after normalization:
# densityFun(Data=normData, plotColors, normMeth=normMeth)
```

`densityFunUnsmoothed` *Create an image with an unsmoothed density curve of the intensities*

Description

This function (from functions_imagesQC.R) creates an image with an unsmoothed density curve of the intensities for all arrays in the raw or normalized dataset (depending on the object passed).

Usage

```
densityFunUnsmoothed(
  Data,
  plotColors = NULL,
  normMeth = "",
  WIDTH = 1000,
  HEIGHT = 1414,
  POINTSIZE = 24,
  MAXARRAY = 41
)
```

Arguments

`Data`	(Status: required) The raw data object (datatype: AffyBatch)
`plotColors`	(Status: required, Default:NULL) Vector of colors assigned to each array. (datatype: character)
`normMeth`	(Status: required when Data is a normalized data object, Default:"") String indicating the normalization method used (see normalizeData function for more information on the possible values). (datatype: character)
`WIDTH`	(Status: optional, Default:1000) png image width (datatype: number)
`HEIGHT`	(Status: optional, Default:1414) png image height (datatype: number)
`POINTSIZE`	(Status: optional, Default:24) png image point size (datatype: number)
`MAXARRAY`	(Status: optional, Default:41) threshold to adapt the image to the number of arrays (datatype: number)

Value

A PNG image of the raw or normalized density plots of the arrays, called "DensityHistogram"

Examples

```
# By default, before the normalization the script will call:
# densityFunUnsmoothed(Data=rawData, plotColors)
# and after normalization:
# densityFunUnsmoothed(Data=normData, plotColors, normMeth=normMeth)
```

enterMatrix *Advanced comparison: enter custom contrasts*

Description

Advanced comparison: enter custom contrasts

Usage

```
enterMatrix(matfileName, groupNames)
```

Arguments

matfileName	(Status=required)
groupNames	(Status=required)

Value

a dataframe for the contrast matrix

Examples

```
#example here
```

getArrayType *Determine array type used*

Description

Determine array type used

Usage

```
getArrayType(Data)
```

Arguments

Data	An AffyBatch object

Value

A string indicating the type of the current chip, either "PMMM" for chips with perfect match and mismatch probes, or "PMonly" for chips with perfect match probes only

Examples

```
#aType <- getArrayType(rawData)
```

hybridPlot *Create an image of hybridisation controls*

Description

This function (from functions_imagesQC.R) creates an image of hybridisation controls based on the qc (simpleaffy Bioconductor package) function, which generally only work for chiptypes with perfect match and mismatch probes, but even not for all of those. As such, when these statistics are not provided as parameters, trys are used in this function to compute them internally. Values for which the try fails are not computed, but the script will continue after giving a warning.

Usage

```
hybridPlot(
  Data,
  quality = NULL,
  plotColors = NULL,
  WIDTH = 1000,
  HEIGHT = 1414,
  POINTSIZE = 24,
  MAXARRAY = 41
)
```

Arguments

Data	(Status: required) The raw data object (datatype: AffyBatch)
quality	(Status: optional, Default:NULL) object obtained by calling the qc function (simpleaffy).When not provided, it is computed within the function.(datatype: QCStats)
plotColors	(Status: required, Default:NULL) Vector of colors assigned to each array. (datatype: character)
WIDTH	(Status: optional, Default:1000) png image width (datatype: number)
HEIGHT	(Status: optional, Default:1414) png image height (datatype: number)
POINTSIZE	(Status: optional, Default:24) png image point size (datatype: number)
MAXARRAY	(Status: optional, Default:41) threshold to adapt the image to the number of arrays (datatype: number)

Value

Two images. A PNG image of hybridisation controls, called "RawDataSpike-inPlot"

Examples

```
# hybridPlot(rawData,quality=quality,plotColors)
```

maFun *Create MA plots for each array versus the median array for the raw or normalized dataset*

Description

This function (from functions_imagesQC.R) creates MA plots for each array versus the median array for the raw or normalized dataset. The median array is computed for the whole data set (if perGroup is FALSE) of per experimental group (perGroup is TRUE). In the script this setting will depend on the setting op the MAOption1 parameter, which can have the values "dataset" or "group".

Usage

```
maFun(
  Data,
  experimentFactor = NULL,
  perGroup = FALSE,
  normMeth = "",
  aType = NULL,
  WIDTH = 1000,
  HEIGHT = 1414,
  MAXARRAY = 41
)
```

Arguments

Data	(Status: required) The raw data object (datatype: AffyBatch or ExpressionSet)
experimentFactor	(Status: required when perGroup is TRUE, Default:NULL) The factor of groups. (datatype: factor)
perGroup	(Status: optional, Default:FALSE) Are MA plots to be made for each experimental group separately or not? (datatype: logical)
normMeth	(Status: required when Data is a normalized data object, Default:"") String indicating the normalization method used (see normalizeData function for more information on the possible values). (datatype: character)
aType	(Status: optional, Default:NULL) String indicating the type of the current chip, either "PMMM" for chips with perfect match and mismatch probes, or "PMonly" for chips with perfect match probes only. Required when Data is a raw data object. (datatype: character)
WIDTH	(Status: optional, Default:1000) png image width (datatype: number)
HEIGHT	(Status: optional, Default:1414) png image height (datatype: number)
MAXARRAY	(Status: optional, Default:41) threshold to adapt the image to the number of arrays (datatype: number)

Value

Images of the MA plots of each array versus the median array, each file contains MA plots for six arrays. The files contain the string 'MAplot" and a number if more than one are needed; in case of groupwise computation, the name of the group is also included in the filename.

Examples

```
# By default, before the normalization the script will call:
# maFun(Data=rawData, experimentFactor, perGroup=(MAOption1=="group"), aType=aType)
# and after normalization:
# maFun(Data=normData, experimentFactor, perGroup=(MAOption1=="group"), normMeth=normMeth)
```

`normalizeData` *Normalize the data set*

Description

This function (from functions_processing) normalizes the data in the AffyBatch object provided. Currently, GCRMA, RMA, and PLIER normalization are supported. For GCRMA, fast normalization is not used, as this gives unreliable results. For PLIER, justPlier (plier Bioconductor package) is used, with the "together" option for arrays having perfect match and mismatch probes, and the "PMonly" option for arrays with perfect match probes only. When normalization per experimental group is selected, the function makes sure that still one normalized data object including all arrays is returned. In case customCDF is TRUE, annotation is updated using BrainArray custom cdf environments, before proceeding with the normalization (and summarisation of probe expressions into probeset expressions). To update the annotation, a sub call is made to the addUpdatedCDFenv function.

Usage

```
normalizeData(
  Data,
  normMeth = "",
  perGroup = FALSE,
  experimentFactor = NULL,
  customCDF = TRUE,
  species = NULL,
  CDFtype = NULL,
  aType = NULL,
  isOligo = FALSE,
  WIDTH = 1000,
  HEIGHT = 1414
)
```

Arguments

`Data`	(Status: required) The raw data object (datatype: AffyBatch)
`normMeth`	(Status: required, Default: "") String indicating the normalization method used. Possible values: RMA, GCRMA, PLIER or none. (datatype: character)
`perGroup`	(Status: optional, Default: FALSE) Should normalization be performed per experimental group (e.g. when global differences are expected between groups) or for the dataset as a whole? (datatype: logical)
`experimentFactor`	(Status: required if perGroup is TRUE , Default: NULL) The factor of groups. (datatype: factor)

`customCDF`	(Status: optional, Default: TRUE) Should annotation of the chip be updated before normalizing the data (and building the probesets out of the separate probes)? If requested, this is done using BrainArray updated cdf environments, c.f. addUpdatedCDFenv (datatype: logical)
`species`	(Status: required when customCDF is TRUE, c.f. addUpdatedCDFenv, Default: NULL) The species associated with the chip type. (datatype: character)
`CDFtype`	(Status: required when customCDF is TRUE, c.f. addUpdatedCDFenv, Default: NULL) The type of custom cdf requested. (datatype: character)
`aType`	(Status: optional, Default: NULL) String indicating the type of the current chip, either "PMMM" for chips with perfect match and mismatch probes, or "PMonly" for chips with perfect match probes only. Required if normMeth is "PLIER". (datatype: character)
`isOligo`	(Status: optional, Default:FALSE) is Oligo array or not (datatype: logical)
`WIDTH`	(Status: optional, Default:1000) png image width (datatype: number)
`HEIGHT`	(Status: optional, Default:1414) png image height (datatype: number)

Value

A normalized data object (datatype: ExpressionSet). It also returns A reference sheet (PNG file) indicating the cdf annotation (cdfName) used, the normalization method used, and if this is the case stating that normalization has been performed per experimental group

Examples

```
#By default, the script will call, if customCDF is TRUE:
#normData <- normalizeData(rawData, normMeth,
#perGroup=(normOption1=="group"), experimentFactor, customCDF,
#species, CDFtype)
#or, if customCDF is FALSE:
#normData <- normalizeData(rawData, normMeth,
#perGroup=(normOption1=="group"), experimentFactor, customCDF)
```

`nuseFun` *Create an image with boxplots of normalized Unscaled Standard Errors (NUSE)*

Description

This function (from functions_imagesQC.R) creates an image with boxplots of normalized Unscaled Standard Errors (NUSE) for each array. It calls the NUSE function (affyPLM Bioconductor package). An object obtained by calling fitPLM (affyPLM) is needed. If this object is not provided, it will be computed internally within this function.

Usage

```
nuseFun(
  Data,
  Data.pset = NULL,
  experimentFactor = NULL,
  plotColors = NULL,
```

```
  legendColors = NULL,
  WIDTH = 1000,
  HEIGHT = 1414,
  POINTSIZE = 24,
  MAXARRAY = 41
)
```

Arguments

Data	(Status: required) The raw data object (datatype: AffyBatch)
Data.pset	(Status: optional, Default:NULL) An object obtained by calling fitPLM (affyPLM), used for each but the raw plot. When not provided, it is computed within the function.(datatype: PLMset)
experimentFactor	(Status: required, Default:NULL) The factor of groups. (datatype: factor)
plotColors	(Status: required, Default:NULL) Vector of colors assigned to each array. (datatype: character)
legendColors	(Status: required, Default:NULL) Vector of colors assigned to each experimental group. (datatype: character)
WIDTH	(Status: optional, Default:1000) png image width (datatype: number)
HEIGHT	(Status: optional, Default:1414) png image height (datatype: number)
POINTSIZE	(Status: optional, Default:24) png image point size (datatype: number)
MAXARRAY	(Status: optional, Default:41) threshold to adapt the image to the number of arrays (datatype: number)

Value

A PNG image containing boxplots of the NUSE values per array, called "RawDataNUSEplot"

Examples

```
# By default, the script will call:
# nuseFun(rawData, Data.pset=rawData.pset, experimentFactor, plotColors, legendColors)
# where rawData.pset has been constructed by calling:
# rawData.pset <- fitPLM(rawData)
```

---

pcaFun *Create a Principal Component Analysis (PCA) plot*

Description

This function (from functions_imagesQC.R) creates a Principal Component Analysis (PCA) plot of the arrays in the raw or normalized dataset (depending on the object passed). When the dataset consists of less than three arrays, no PCA plot is generated and a warning is given. Before computing the PCA each probeset's expression values are centred on zero. If scaled_pca is TRUE, they will also be rescaled to unit variance. When the maximum length of an array (sample)name is ten characters, and there are no more than 16 samples, the array (sample)names are put within the plot, otherwise they are put in the legend. Since computing a PCA (using the prcomp function) can be memory intensive, a try is used. Furthermore, in cases where scaling is not possible due to loss of any variation, a second attempt is done using no scaling (when scaled_pca had been set to TRUE), and a warning is given. When no PCA can be computed the image is not created, and a warning is given.

Usage

```
pcaFun(
  Data,
  experimentFactor = NULL,
  normMeth = "",
  scaled_pca = TRUE,
  plotColors = NULL,
  legendColors = NULL,
  plotSymbols = NULL,
  legendSymbols = NULL,
  namesInPlot = FALSE,
  WIDTH = 1000,
  HEIGHT = 1414,
  POINTSIZE = 24
)
```

Arguments

Data	(Status: required) The raw data object (datatype: AffyBatch)
experimentFactor	(Status: required, Default:NULL) The factor of groups. (datatype: factor)
normMeth	(Status: required when Data is a normalized data object, Default:"") String indicating the normalization method used (see normalizeData function for more information on the possible values). (datatype: character)
scaled_pca	(Status: optional, Default:TRUE) Should each probeset's expression be scaled to unit variance before proceeding? Note that the expression is centred on zero in any case. (datatype: logical)
plotColors	(Status: required, Default:NULL) Vector of colors assigned to each array. (datatype: character)
legendColors	(Status: required, Default:NULL) Vector of colors assigned to each experimental group. (datatype: character)
plotSymbols	(Status: required, Default:NULL) symbol assigned to each array. (datatype: number)
legendSymbols	(Status: required, Default:NULL) symbol assigned to each experimental group. (datatype: number)
namesInPlot	(Status: optional, Default:FALSE) Should the array (sample)names be put within the plot, or in the legend? (datatype: logical)
WIDTH	(Status: optional, Default:1000) png image width (datatype: number)
HEIGHT	(Status: optional, Default:1414) png image height (datatype: number)
POINTSIZE	(Status: optional, Default:24) png image point size (datatype: number)

Value

A PNG image with PCA plot of the arrays. Naming is either 'Raw' or the normalization method, followed by "DataPCAanalysis"

Examples

```
# By default, before the normalization the script will call:
# pcaFun(Data=rawData, experimentFactor=experimentFactor,
# plotColors=plotColors, legendColors=legendColors,
# namesInPlot=((max(nchar(sampleNames(rawData)))<=10)&&
#  (length(sampleNames(rawData))<=16))
# and after normalization:
# pcaFun(Data=normData, experimentFactor=experimentFactor,
# normMeth=normMeth, plotColors=plotColors,
# legendColors=legendColors,
# namesInPlot=((max(nchar(sampleNames(rawData)))<=10)&&
#  (length(sampleNames(rawData))<=16))
```

---

percPresPlot *Creates an image of the percentage present values of each array*

Description

This function (from functions_imagesQC.R) creates an image of the percentage present values of each array based on the qc (simpleaffy Bioconductor package) function, which generally only work for chiptypes with perfect match and mismatch probes, but even not for all of those. As such, when these statistics are not provided as parameters, trys are used in this function to compute them internally. Values for which the try fails are not computed, but the script will continue after giving a warning.

Usage

```
percPresPlot(
  Data,
  quality = NULL,
  experimentFactor = NULL,
  plotColors = NULL,
  legendColors = NULL,
  WIDTH = 1000,
  HEIGHT = 1414,
  POINTSIZE = 24,
  MAXARRAY = 41
)
```

Arguments

Data	(Status: required) The raw data object (datatype: AffyBatch)
quality	(Status: optional, Default:NULL) object obtained by calling the qc function (simpleaffy).When not provided, it is computed within the function.(datatype: QCStats)
experimentFactor	(Status: required, Default:NULL) The factor of groups. (datatype: factor)
plotColors	(Status: required, Default:NULL) Vector of colors assigned to each array. (datatype: character)
legendColors	(Status: required, Default:NULL) Vector of colors assigned to each experimental group. (datatype: character)

WIDTH	(Status: optional, Default:1000) png image width (datatype: number)
HEIGHT	(Status: optional, Default:1414) png image height (datatype: number)
POINTSIZE	(Status: optional, Default:24) png image point size (datatype: number)
MAXARRAY	(Status: optional, Default:41) threshold to adapt the image to the number of arrays (datatype: number)

Value

A PNG image of percent present values, called "RawDataPercentPresentPlot"

Examples

```
# percPresPlot(rawData, quality=quality, experimentFactor, plotColors, legendColors)
```

---

plotArrayLayout *Create an image of the layout of the current chiptype*

Description

This function (from functions_imagesQC.R) creates an image of the layout of the current chiptype. Thus, this plot does not plot any data, but shows where control and regular perfect match (PM) and (if applicable) mismatch (MM) probes are present on the array. Note: due to resolution issues, banding may seem different from the real situation, e.g. normally on classical chiptypes, PM and MM are present in alternate lines, but patterns may appear due to image resolution. This function tries to load annotation libraries, depending on the chiptype.

Usage

```
plotArrayLayout(
  Data,
  aType = NULL,
  WIDTH = 1000,
  HEIGHT = 1414,
  POINTSIZE = 24
)
```

Arguments

Data	(Status: required) The raw data object (datatype: AffyBatch or ExpressionSet)
aType	(Status: optional, Default:NULL) String indicating the type of the current chip, either "PMMM" for chips with perfect match and mismatch probes, or "PMonly" for chips with perfect match probes only. Required when Data is a raw data object. (datatype: character)
WIDTH	(Status: optional, Default:1000) png image width (datatype: number)
HEIGHT	(Status: optional, Default:1414) png image height (datatype: number)
POINTSIZE	(Status: optional, Default:24) png image point size (datatype: number)

Value

A PNG image of the expression profiles of the affx controls, called "Array_layout_plot"

Examples

```
# plotArrayLayout(rawData,aType)
```

PNdistrPlot — *Create an image with boxplots of positive and negative control intensities*

Description

This function (from functions_imagesQC.R) creates an image with boxplots of positive and negative control intensities for each array. It calls the borderQC1 function (affyQCReport Bioconductor package).

Usage

```
PNdistrPlot(Data, WIDTH = 1000, HEIGHT = 1414, POINTSIZE = 24)
```

Arguments

Data	(Status: required) The raw data object (datatype: AffyBatch)
WIDTH	(Status: optional, Default:1000) png image width (datatype: number)
HEIGHT	(Status: optional, Default:1414) png image height (datatype: number)
POINTSIZE	(Status: optional, Default:24) png image point size (datatype: number)

Value

A PNG image of boxplots of positive and negative control intensities, called "RawDataPosNegDistribPlot"

Examples

```
# PNdistrPlot(rawData)
```

PNposPlot — *Create an image showing the centres of intensity of the positive and negative border elements*

Description

This function (from functions_imagesQC.R) creates an image showing the centres of intensity of the positive and negative border elements for each array. It calls the borderQC2 function (affyQCReport Bioconductor package).

Usage

```
PNposPlot(Data, WIDTH = 1000, HEIGHT = 1414, POINTSIZE = 24)
```

Arguments

Data	(Status: required) The raw data object (datatype: AffyBatch)
WIDTH	(Status: optional, Default:1000) png image width (datatype: number)
HEIGHT	(Status: optional, Default:1414) png image height (datatype: number)
POINTSIZE	(Status: optional, Default:24) png image point size (datatype: number)

Value

An image of centres of intensity of positive and negative border elements for each array, called "RawDataPosNegPositionPlot"

Examples

```
# PNposPlot(rawData)
```

QCtablePlot *Compute and plot a table of QC statistics*

Description

This function (from functions_imagesQC.R) computes and plots a table of QC statistics based on the qc (simpleaffy Bioconductor package) and yaqc (yaqcaffy Bioconductor package) functions, which generally only work for chiptypes with perfect match and mismatch probes, but even not for all of those. As such, when these statistics are not provided as parameters, trys are used in this function to compute them internally. Values for which the try fails are not computed, but the script will continue after giving a warning.

Usage

```
QCtablePlot(
  Data,
  quality = NULL,
  sprep = NULL,
  lys = NULL,
  samplePrep = TRUE,
  ratio = TRUE,
  hybrid = TRUE,
  percPres = TRUE,
  bgPlot = TRUE,
  scaleFact = TRUE,
  WIDTH = 1000,
  HEIGHT = 1414,
  POINTSIZE = 24
)
```

Arguments

Data	(Status: required) The raw data object (datatype: AffyBatch)
quality	(Status: optional, Default:NULL) object obtained by calling the qc function (simpleaffy).When not provided, it is computed within the function.(datatype: QCStats)
sprep	(Status: optional, Default:NULL) A matrix of 3'probe intensities for dap, thr, lys, and phe, taken from an object of class YAQCStats (yaqc function, yaqcaffy). When not provided, it is computed within the function. (datatype: matrix)
lys	(Status: optional, Default:NULL) Matrix of A, M, P calls for the 3' probeset of Lys on each array, based on results from the detection.p.val function (simpleaffy) . When not provided, it is computed within the function. (datatype: matrix)
samplePrep	(Status: optional, Default:TRUE) Does the table have to contain sample preparation QC statistics? (datatype: logical)
ratio	(Status: optional, Default:TRUE) Does the table have to contain 3'/5' ratio statistics? (datatype: logical)
hybrid	(Status: optional, Default:TRUE) Does the table have to contain hybridisation QC statistics? (datatype: logical)
percPres	(Status: optional, Default:TRUE) Does the table have to contain percentage present QC statistics? (datatype: logical)
bgPlot	(Status: optional, Default:TRUE) Does the table have to contain background signal intensity QC statistics? (datatype: logical)
scaleFact	(Status: optional, Default:TRUE) Does the table have to contain scale factor QC statistics?(datatype: logical)
WIDTH	(Status: optional, Default:1000) png image width (datatype: number)
HEIGHT	(Status: optional, Default:1414) png image height (datatype: number)
POINTSIZE	(Status: optional, Default:24) png image point size (datatype: number)

Value

A PNG file called 'QCtable' with the requested statistics

Examples

```
# By default, the script will call:
# computeQCtable(rawData, quality, sprep, lys, samplePrep = samplePrep,
# ratio = ratio, hybrid = hybrid, percPres = percPres, bgPlot = bgPlot, scaleFact = scaleFact)
```

---

ratioPlot *Create an image of beta-actin and GAPDH 3'/5' ratios*

Description

This function (from functions_imagesQC.R) creates an image of beta-actin and GAPDH 3'/5' ratios based on the qc (simpleaffy Bioconductor package) function, which generally only work for chip-types with perfect match and mismatch probes, but even not for all of those. As such, when these statistics are not provided as parameters, trys are used in this function to compute them internally. Values for which the try fails are not computed, but the script will continue after giving a warning.

Usage

```
ratioPlot(
  Data,
  quality = NULL,
  experimentFactor = NULL,
  plotColors = NULL,
  legendColors = NULL,
  WIDTH = 1000,
  HEIGHT = 1414,
  POINTSIZE = 24,
  MAXARRAY = 41
)
```

Arguments

Data	(Status: required) The raw data object (datatype: AffyBatch)
quality	(Status: optional, Default:NULL) object obtained by calling the qc function (simpleaffy).When not provided, it is computed within the function.(datatype: QCStats)
experimentFactor	(Status: required, Default:NULL) The factor of groups. (datatype: factor)
plotColors	(Status: required, Default:NULL) Vector of colors assigned to each array. (datatype: character)
legendColors	(Status: required, Default:NULL) Vector of colors assigned to each experimental group. (datatype: character)
WIDTH	(Status: optional, Default:1000) png image width (datatype: number)
HEIGHT	(Status: optional, Default:1414) png image height (datatype: number)
POINTSIZE	(Status: optional, Default:24) png image point size (datatype: number)
MAXARRAY	(Status: optional, Default:41) threshold to adapt the image to the number of arrays (datatype: number)

Value

Two images. A PNG image of 3'/5'ratios for beta actin, called 'RawData53ratioPlot_beta-actin'. A PNG image of 3'/5' ratios for GAPDH, called 'RawData53ratioPlot_GAPDH'

Examples

```
# ratioPlot(rawData, quality=quality, experimentFactor, plotColors, legendColors)
```

---

readDescFile *Read desc file. Return array names and groups*

Description

Read desc file. Return array names and groups

Usage

```
readDescFile(descfile, outprint = FALSE)
```

Arguments

descfile	(Status=required)
outprint	(datatype=logical, Default=FALSE)

Value

a dataframe with description matrix

Examples

```
#example here
```

---

readExtFile *Read norm file, depending on the file type (supports: txt, csv, xls, xlsx)*

Description

Read norm file, depending on the file type (supports: txt, csv, xls, xlsx)

Usage

```
readExtFile(filename, extension, outprint = FALSE)
```

Arguments

filename	(Status=required)
extension	(Status=required)
outprint	(datatype=logical, Default=FALSE)

Value

a dataframe with the normalized gene expression data

Examples

```
#example here
```

`rleFun` *Create an image with boxplots of Relative Log Expression (RLE) values*

Description

This function (from functions_imagesQC.R) creates an image with boxplots of Relative Log Expression (RLE) values for each array. It calls the RLE function (affyPLM Bioconductor package). An object obtained by calling fitPLM (affyPLM) is needed. If this object is not provided, it will be computed internally within this function.

Usage

```
rleFun(
  Data,
  Data.pset = NULL,
  experimentFactor = NULL,
  plotColors = NULL,
  legendColors = NULL,
  WIDTH = 1000,
  HEIGHT = 1414,
  POINTSIZE = 24,
  MAXARRAY = 41
)
```

Arguments

`Data`	(Status: required) The raw data object (datatype: AffyBatch)
`Data.pset`	(Status: optional, Default:NULL) An object obtained by calling fitPLM (affyPLM), used for each but the raw plot. When not provided, it is computed within the function.(datatype: PLMset)
`experimentFactor`	(Status: required, Default:NULL) The factor of groups. (datatype: factor)
`plotColors`	(Status: required, Default:NULL) Vector of colors assigned to each array. (datatype: character)
`legendColors`	(Status: required, Default:NULL) Vector of colors assigned to each experimental group. (datatype: character)
`WIDTH`	(Status: optional, Default:1000) png image width (datatype: number)
`HEIGHT`	(Status: optional, Default:1414) png image height (datatype: number)
`POINTSIZE`	(Status: optional, Default:24) png image point size (datatype: number)
`MAXARRAY`	(Status: optional, Default:41) threshold to adapt the image to the number of arrays (datatype: number)

Value

A PNG image containing boxplots of the NUSE values per array, called "RawDataRLEplot"

Examples

```
# By default, the script will call:
# rleFun(rawData, Data.pset=rawData.pset, experimentFactor, plotColors, legendColors
# where rawData.pset has been constructed by calling:
# rawData.pset <- fitPLM(rawData)
```

RNAdegPlot *Creates an image of overall RNA degradation for all arrays*

Description

This function (from functions_imagesQC.R) creates an image of overall RNA degradation for all arrays. It calls the function AffyRNAdeg (affy Bioconductor package).

Usage

```
RNAdegPlot(
  Data,
  Data.rnadeg = NULL,
  plotColors = NULL,
  WIDTH = 1000,
  HEIGHT = 1414,
  POINTSIZE = 24,
  MAXARRAY = 41
)
```

Arguments

Data	(Status: required) The raw data object (datatype: AffyBatch)
Data.rnadeg	(Status: optional, Default:NULL) List as obtained by calling the AffyRNAdeg function (affy). When not provided, it is computed internally within this function. (datatype: list)
plotColors	(Status: required, Default:NULL) Vector of colors assigned to each array. (datatype: character)
WIDTH	(Status: optional, Default:1000) png image width (datatype: number)
HEIGHT	(Status: optional, Default:1414) png image height (datatype: number)
POINTSIZE	(Status: optional, Default:24) png image point size (datatype: number)
MAXARRAY	(Status: optional, Default:41) threshold to adapt the image to the number of arrays (datatype: number)

Value

A PNG image of overall RNA degradation, called 'RawDataRNAdegradationPlot'

Examples

```
# RNAdegPlot(rawData,plotColors=plotColors)
```

samplePrepPlot *Create an image of sample preparation controls*

Description

This function (from functions_imagesQC.R) creates an image of sample preparation controls based on the yaqc (yaqcaffy Bioconductor package) function, which generally only work for chiptypes with perfect match and mismatch probes, but even not for all of those. As such, when these statistics are not provided as parameters, trys are used in this function to compute them internally. Values for which the try fails are not computed, but the script will continue after giving a warning.

Usage

```
samplePrepPlot(
  Data,
  sprep = NULL,
  lys = NULL,
  plotColors = NULL,
  WIDTH = 1000,
  HEIGHT = 1414,
  POINTSIZE = 24,
  MAXARRAY = 41
)
```

Arguments

Data	(Status: required) The raw data object (datatype: AffyBatch)
sprep	(Status: optional, Default:NULL) A matrix of 3'probe intensities for dap, thr, lys, and phe, taken from an object of class YAQCStats (yaqc function, yaqcaffy). When not provided, it is computed within the function. (datatype: matrix)
lys	(Status: optional, Default:NULL) Matrix of A, M, P calls for the 3' probeset of Lys on each array, based on results from the detection.p.val function (simpleaffy) . When not provided, it is computed within the function. (datatype: matrix)
plotColors	(Status: required, Default:NULL) Vector of colors assigned to each array. (datatype: character)
WIDTH	(Status: optional, Default:1000) png image width (datatype: number)
HEIGHT	(Status: optional, Default:1414) png image height (datatype: number)
POINTSIZE	(Status: optional, Default:24) png image point size (datatype: number)
MAXARRAY	(Status: optional, Default:41) threshold to adapt the image to the number of arrays (datatype: number)

Value

A PNG image of sample preparation controls, called "RawDataSamplePrepControl"

Examples

```
# samplePrepPlot(rawData,sprep,lys,plotColors)
```

saveComparison *For each contrast, save the comparison into files*

Description

For each contrast, save the comparison into files

Usage

```
saveComparison(
  cont.matrix,
  fit,
  normDataTable,
  annotation = NULL,
  firstColumn = NULL
)
```

Arguments

cont.matrix	(Status=required)
fit	(Status=required)
normDataTable	(Status=required)
annotation	(default=NULL)
firstColumn	(default=NULL)

Value

several files (for each comparison)

Examples

```
#example here
```

scaleFactPlot *Create an image of the scale factors of the array*

Description

This function (from functions_imagesQC.R) creates an image of the scale factors of the array based on the qc (simpleaffy Bioconductor package) function, which generally only work for chiptypes with perfect match and mismatch probes, but even not for all of those. As such, when these statistics are not provided as parameters, trys are used in this function to compute them internally. Values for which the try fails are not computed, but the script will continue after giving a warning.

Usage

```
scaleFactPlot(
  Data,
  quality = NULL,
  experimentFactor = NULL,
  plotColors = NULL,
  legendColors = NULL,
  WIDTH = 1000,
  HEIGHT = 1414,
  POINTSIZE = 24,
  MAXARRAY = 41
)
```

Arguments

Data	(Status: required) The raw data object (datatype: AffyBatch)
quality	(Status: optional, Default:NULL) object obtained by calling the qc function (simpleaffy).When not provided, it is computed within the function.(datatype: QCStats)
experimentFactor	(Status: required, Default:NULL) The factor of groups. (datatype: factor)
plotColors	(Status: required, Default:NULL) Vector of colors assigned to each array. (datatype: character)
legendColors	(Status: required, Default:NULL) Vector of colors assigned to each experimental group. (datatype: character)
WIDTH	(Status: optional, Default:1000) png image width (datatype: number)
HEIGHT	(Status: optional, Default:1414) png image height (datatype: number)
POINTSIZE	(Status: optional, Default:24) png image point size (datatype: number)
MAXARRAY	(Status: optional, Default:41) threshold to adapt the image to the number of arrays (datatype: number)

Value

A PNG image of scale factors, called "RawDataScaleFactorsPlot"

Examples

```
# scaleFactPlot(rawData, quality=quality, experimentFactor, plotColors,legendColors)
```

---

spatialImages *Create an image per array, containing one to four spatial images*

Description

This function (from functions_imagesQC.R) creates an image per array, containing one to four spatial images. For any but the raw images, an object obtained by calling fitPLM (affyPLM Bioconductor package) is used. If this object is not provided, it will be computed internally within this function.

Usage

```
spatialImages(
  Data,
  Data.pset = NULL,
  Resid = TRUE,
  ResSign = TRUE,
  Raw = TRUE,
  Weight = TRUE,
  WIDTH = 1000,
  HEIGHT = 1414,
  POINTSIZE = 24
)
```

Arguments

Data	(Status: required) The raw data object (datatype: AffyBatch or ExpressionSet)
Data.pset	(Status: optional, Default:NULL) An object obtained by calling fitPLM (affyPLM), used for each but the raw plot. When not provided, it is computed within the function.(datatype: PLMset)
Resid	(Status: optional, Default:TRUE) Should a residual plot be made? (datatype: logical)
ResSign	(Status: optional, Default:TRUE) Should a residual sign plot be made? (datatype: logical)
Raw	(Status: optional, Default:TRUE) Should a raw plot be made? (datatype: logical)
Weight	(Status: optional, Default:TRUE) Should a weight plot be made? (datatype: logical)
WIDTH	(Status: optional, Default:1000) png image width (datatype: number)
HEIGHT	(Status: optional, Default:1414) png image height (datatype: number)
POINTSIZE	(Status: optional, Default:24) png image point size (datatype: number)

Value

Images containing each of the requested plots, one file per array. Naming is 'virtual_image' followed by the (sample)name of the array.

Examples

```
# There are two default calls by the script:
# 1/ Compute only the images showing the residuals of the PLM (if spatialImage parameter is TRUE):
# spatialImages(rawData, Data.pset=rawData.pset, Resid=TRUE, ResSign=FALSE, Raw=FALSE, Weight=FALSE)
# 2/ Compute the four images for all arrays (if PLMimage parameter is TRUE):
# spatialImages(rawData, Data.pset=rawData.pset)
# where rawData.pset has been constructed by calling:
# rawData.pset <- fitPLM(rawData)
```

`toptable` *Create a toptable and save it into a file*

Description

Create a toptable and save it into a file

Usage

```
toptable(contrast.fit, i, normDataTable, fileName, firstColumn, annotation)
```

Arguments

`contrast.fit`	(Status=required)
`i`	(Status=required) coefficient
`normDataTable`	(Status=required)
`fileName`	(Status=required) contrast file name
`firstColumn`	(Status=required)
`annotation`	(Status=required)

Value

the file name where the toptable results are saved

Examples

```
#example here
```

Index