Package 'RPPanalyzer'

January 25, 2024

Type Package

Title Reads, Annotates, and Normalizes Reverse Phase Protein Array Data

Version 1.4.9

Date 2024-01-25

Author Heiko Mannsperger with contributions of Stephan Gade

Maintainer Torsten Schoeps <torsten.schoeps@bioinf.med.uni-goettingen.de>

Depends R (>= 2.14)

Imports quantreg, limma, lattice, stats4, gam, gplots, grid, ggplot2, Hmisc, Biobase, methods, utils, graphics

ZipData no

Description Reads in sample description and slide description files and annotates the expression values taken from GenePix results files (text file format used by many microarray scanner and software providers). After normalization data can be visualized as boxplot, heatmap or dotplot.

License LGPL

LazyLoad yes

Repository CRAN

Date/Publication 2024-01-25 11:00:02 UTC

NeedsCompilation no

R topics documented:

`RPPanalyzer-package` *Read, annotate and normalize reverse phase protein array data and get a brief overview on the biological impact.*

Description

The package reads pheno and feature data of an RPPA experiment from textfiles and annotates the expression values in genepix result files (gpr files). For background correction the `backgroundcorrect` funktion from the limma package is used. After normalization data can be plotted to check quality control or to get a first impression on the biological relevance of the data set.

Author(s)

Maintainer: Heiko Mannsperger <h.mannsperger@dkfz.de>

Examples

```
## Not run:
data(dataI)

bgcorrected <- correctBG(dataI)
normalized <- normalizeRPPA(bgcorrected,method="proteinDye")
aggregated <- sample.median(normalized)

## End(Not run)
```

`averageData` *Average biological replicates over different slides.*

Description

The function assumes that each signal originates from an underlying true value which is scaled by a scaling factor depending on the slide and replicate. The method optimizes the scaling and truth parameters such that the distance between predicted and actual signals is minimized. There are aguments to specify what factors the scaling factors and truth parameters depend on.

Usage

```
averageData(subsample, scaling = c("slide", "replicate"),
distinguish = c("cellline", "treatment"))
```

Arguments

`subsample`	data.frame with columns "slide" (factor, the slide names), "ab" (factor, the antibody/target names), "time" (numeric, the time points), "signal" (numeric, signal values), "var0" (numeric, error parameter for the constant error), "varR" (numeric, error parameter for the relative error). The data.frame may contain further columns that can then be used in the `scaling` and `distinguish` arguments. The data.frame is a standard output of `getErrorModel`.
`scaling`	character. One scaling parameter ist estimated for each occurring combination of the corresponding factors.
`distinguish`	character. One truth parameter ist estimaed for each occuring combination of the factors "time", "ab" (antibody/target) and the factors in `distinguish`.

Details

Averaging is based on the assumption that for each level of `scaling` there is an underlying "true" antibody time-course for each level of `distinguish`. The signals of different scaling levels are assumed to differ by a scaling factor. Both, antibody time-course values and scaling parameters are estimated simulatenously by generalized least squares estimation:

$$GRSS = \sum_{i,j} (s_i S_{ij} - y_j/s_i)^2 / (\sigma^2_{ij,0} + (y_j/s_i)^2 \sigma^2_{ij,R})$$

where i, j correspond to the levels of `c("time", "ab", distinguish)` and the levels of `scaling`.

Value

data.frame	with columns "time", "ab", "signal" (the truth parameters returned by nls), "sigma" (the standard error of the truth parameter returned by nls), "connection" (integer, signals can only be compared on the same scale if they agree in "ab", and "connection") and one column for each entry of distinguish.

Author(s)

Daniel Kaschek, Physikalisches Institut, Uni Freiburg. Email: daniel.kaschek@physik.uni-freiburg.de

---

calcLinear *Calculates sample concentrations using linear model fit*

Description

calculates sample concentrations of a RPPA data set, using parameter of a linear model fitted to the dilution series.

Usage

```
calcLinear(x, sample.id = c("sample", "sample.n"), dilution = "dilution"
, method = "quantreg", plot = F, detectionLimit = T)
```

Arguments

x	List containing background corrected RPPA data set
sample.id	character vector refering to column names from which samples can be separated
dilution	column name from the column in feature data that describes the dilution steps of each sample
method	character string describing the method used for the linear fit
plot	logical. If true dilution curves are plotted
detectionLimit	logical. If true model is fitted on dilution steps above the detection limit. If false, all data points are used to fit the model

Value

expression	matrix with protein expression data
dummy	matrix with protein expression data
arraydescription	data frame with feature data
sampledescription	data frame with pheno data

Note

for calculation of serial diluted samples only

Author(s)

Heiko Mannsperger <h.mannsperger@dkfz.de>,Stephan Gade <s.gade@dkfz.de>

Examples

```
## Not run:
library(RPPanalyzer)
data(ser.dil.samples)

predicted.data <- calcLinear(ser.dil.samples,sample.id=c("sample","sample.n"),
dilution="dilution")

## End(Not run)
```

---

calcLogistic *Calculates sample concentrations using sigmoid model fit*

Description

Calculates sample concentrations of a RPPA data set, as wrapper for curveFitSigmoid.

Usage

```
calcLogistic(x, sample.id = c("sample", "sample.n"), dilution = "dilution",
xVal = NULL, plot = F, detectionLimit = F)
```

Arguments

x	x List containing RPPA data set
sample.id	character vector refering to column names from which samples can be separated
dilution	column name from the column in feature data that describes the dilution steps of each sample
xVal	defines the dilution value for which the concentration is calulated. If null the highest dilution value is used
plot	logical. If true dilution curves are plotted
detectionLimit	logical. If true model is fitted on dilution steps above the detection limit. If false, all data points are used to fit the model

Value

expression	matrix with protein expression data
dummy	matrix with protein expression data
arraydescription	data frame with feature data
sampledescription	data frame with pheno data

Author(s)

Heiko Mannsperger <h.mannsperger@dkfz.de>, Stephan Gade <s.gade@dkfz.de>

Examples

```
## Not run:
library(RPPanalyzer)
data(ser.dil.samples)

predicted.data <- calcLogistic(ser.dil.samples, sample.id=c("sample","sample.n"),
dilution="dilution")

## End(Not run)
```

`calcSdc` *Calculates the concentration of serial diluted samples*

Description

Calculates the protein concentration of a serial diluted sample stored in an RPPA data list using the serial dilution curve algorithm published by Zhang et.al, Bioinformatics 2009.

Usage

```
    calcSdc(x,sample.id=c("sample","sample.n"),
sel=c("measurement","control"), dilution="dilution",
D0=2,sensible.min=5, sensible.max=1.e9,minimal.err=5,
plot=T, r=1.2)
```

Arguments

`x`	RPPA data list with replicates aggregated with median
`sample.id`	Attributes to identify the samples
`sel`	The sample type that should be calculated. Has to be "measurements","control", "neg_control",or "blank".
`dilution`	Name of the column in the feature data matrix describing the dilution steps of the samples.
`D0`	Dilution factor.
`sensible.min`	Signals below this value are marked as undetected
`sensible.max`	Signals above the value are marked as saturated
`minimal.err`	Minimal valid estimate for the background noise
`plot`	Logical. If true, model fits are plotted
`r`	Constant factor used to determine the confidence interval for the saturation limit M and the background noise a, shoul be >1. Can be lower if accuracy of signals is improved.

Details

The method of Zhang et. al doesn't fit the dose response curve but a derive model describing the functional relationship between the signals of two consecutive dilution steps. Since this new model does not contain the protein concentration anymore all spots of one array can be used for the fit, allowing a much more robust estimation of the underlying paramters.

Value

`expression`	matrix with expression values
`error`	matrix with error values
`arraydescription`	data frame with feature data
`sampledescription`	data frame with pheno data

Author(s)

Heiko Mannsperger <h.mannsperger@dkfz.de>, Stephan Gade <s.gade@dkfz.de>

References

Zhang et. al, Bioinformatics 2009,Serial dilution curve: a new method for analysis of reverse phase protein array data

Examples

```
## Not run:
    library(RPPanalyzer)
    data(ser.dil.samples)

    ser.dil_median <- sample.median(ser.dil.samples)
   predicted.data <- calcSdc(ser.dil_median,D0=2,sel=c("measurement"), dilution="dilution")

## End(Not run)
```

`correctBG` *Corrects for background in an RPPA data set*

Description

Corrects for background in an RPPA data set using different algorithms (e.g. from the limma package) avoiding negative values

Usage

```
correctBG(x, method = "normexp")
```

Arguments

x	List with RPPA data set
method	any method from the function backgroundCorrect and addmin which adds a fix number to each value to avoid negative values

Details

This function is a wrapper for the backgroundCorrect function of the limma package. As additional method "addmin" is implemented.

Value

expression	matrix with background corrected expression data
background	matrix with background data
arraydescription	data frame with feature data
sampledescription	data frame with pheno data

Author(s)

Heiko Mannsperger <h.mannsperger@dkfz.de>, Stephan Gade <s.gade@dkfz.de>

References

Ritchie, ME, Silver, J, Oshlack, A, Holmes, M, Diyagama, D, Holloway, A, and Smyth, GK (2007). A comparison of background correction methods for two-colour microarrays. Bioinformatics 23, 2700-2707.

See Also

For detailed information about the background correction methods see: backgroundCorrect,

Examples

```
## Not run:
library(RPPanalyzer)
data(dataI)

dataBGcorrected <- correctBG(dataI,method="normexp")

## End(Not run)
```

`correctDilinterc` *Dilution series intersect correction*

Description

Consists of 3 functions: `getIntercepts()`, `analyzeIntercepts()` and `getSignals()`. The first one derives intercepts of dilution series in dependence of dilSeriesID (column in sampledescription.txt) and slide/pad/incubationRun/spottingRun number (colnames of arraydescription). A smoothing spline is used to extrapolate to 0. Nonparametric bootstrap is used to estimate uncertainty of the intercept estimate. The second function is used in the last one and does Analysis of Variances for nested models. The last one updates the original timeseries signal to (foreground expression - intercept).

Usage

```
correctDilinterc(dilseries, arraydesc, timeseries, exportNo)
    getIntercepts(dilseries, arraydesc)
    analyzeIntercepts(intercepts, test="F", export)
    getSignals(timeseries, intercepts, arraydesc, exportNo)
as.my(v)
```

Arguments

`dilseries`	foreground signal matrix as result of `write.Data` and import of resulting txt file, but just sample_type "control", i.e. dilution series
`arraydesc`	"arraydescription" matrix of the RPPA data set list
`timeseries`	foreground signal matrix as result of `write.Data` and import of resulting txt file, but just sample_type "measurement"
`exportNo`	integer of 1-4 which of the linear fits should be exported to the attribute of the result, variable for `analyzeIntercepts()`, 1: constant, 2: antibody, 3: antibody + slide (default) or antibody + slide + sample (`dilSeriesID`)
`intercepts`	output of `getIntercepts()`, data frame with columns for dilSeriesID and slide/pad/incubationRun/spotti number as well as antibody, estimated intercept and estimated error of intercept
`test`	test parameter for ANOVA (see documentation of `anova`), default is "F"
`export`	see `exportNo`
`v`	some variable

Value

matrix with adapted signal intensities via subtraction of dilution intercept at concentration 0

Author(s)

Daniel Kaschek, Silvia von der Heyde

Examples

```
## Not run:
library(RPPanalyzer)


# read data
dataDir <- system.file("extdata", package="RPPanalyzer")
setwd(dataDir)
rawdata <- read.Data(blocksperarray=12, spotter="aushon", printFlags=FALSE)
# write data
write.Data(rawdata,FileNameExtension="test_data")
# import raw data
fgRaw.tmp <- read.delim("test_dataexpression.txt",
stringsAsFactors=FALSE, row.names=NULL, header=TRUE)
fgRaw <- read.delim("test_dataexpression.txt", skip=max(which(fgRaw.tmp[,1]==""))+1,
stringsAsFactors=FALSE, row.names=NULL, header=TRUE)
# remove NAs
fgNAVec <- which(is.na(fgRaw[,"ID"]))
if(length(fgNAVec) > 0){
fgRaw <- fgRaw[-fgNAVec,]
}
colnames(fgRaw) <- sub("X","", gsub("\\.","-", colnames(fgRaw)))
# correct data for BG noise
correctedData <- correctDilinterc(dilseries=fgRaw[which(fgRaw$sample_type=="control" &
!is.na(fgRaw$dilSeriesID)),], arraydesc=rawdata$arraydescription,
timeseries=fgRaw[which(fgRaw$sample_type=="measurement"),], exportNo=2)

## End(Not run)
```

`curvePredictSigmoid` *Sigmoidal curve prediction.*

Description

3-parameter sigmoidal curve.

Usage

```
curvePredictSigmoid(x, params)
```

Arguments

`x`	Input value(s).
`params`	Parameter vector containing three parameters alpha, beta and gamma.

Details

The model is defined as `alpha + beta*(2^(x*gamma))/(1+2^(x*gamma)))`.

Value

The prediction f(x) of the input value(s).

Examples

```
## Not run:
x <- seq(-5, 5, by=0.1)
y <- curvePredictSigmoid(x, c(alpha=2, beta=1, gamma=1.5))
plot(x, y)

## End(Not run)
```

---

dataI *Reverse phase protein array rawdata, samples serially diluted*

Description

The data Set is a list of four elements. Expression and background are matrices containing signal intensities, the data frames arraydescription and sampledescription comprising feature and pheno-data.

Usage

```
data(dataI)
```

Format

list

Details

The data set is a list of four elements with data of a original reverse phase array experiment. The elements expression and background are 2304 times 26 matrices containing integers describing the signal intensities and local background for every spot of the experiment as generated with image analysis software. Arraydescription is a data frame, describing the incubation of every array refering the column of the expression and background matrix. Required rows are target and AB_ID with characters and array.id (four integers linked with "-"). Sampledescription is a data frame according to the rows of expression and background matrix and annotates the samples. Sampledescription requires the columns "ID", "sample_type", "sample", "concentration", and "dilution" as minimal information and "sample.n" to separate different sample groups.

Source

The data set contains original reverse phase protein array signals with randomized pheno and feature data.

Examples

```
data(dataI)
str(dataI)
```

`dataII` *Reverse phase protein array data, samples from a stimulation time course*

Description

The data Set is a list of four elements. `Sample.median` and `sample.mads` are matrices containing logged signal intensities and errors, the data frames arraydescription and sampledescription comprising feature and phenodata.

Usage

```
data(dataII)
```

Format

List

Details

The data set is a list of four elements with data of a original reverse phase array experiment. The elements `Sample.median` and `sample.mads` are 624 times 12 matrices containing logged signal intensities and errors for every sample of the experiment. The values are background corrected and normalized against total protein content. Arraydescription is a data frame, describing the incubation of every array refering the column of the matrices. Required rows are target and AB_ID with characters and array.id (four integers linked with "-"). Sampledescription is a data frame according to the rows of the matrices annotating the samples. The columns "sample", "stimulation", "inhibition", "stim_concentration", and "time" are describing the time course experiment.

Source

The data set contains original reverse phase protein array signals from a stimulation time course experiment with randomized pheno and feature data.

Examples

```
data(dataII)
str(dataII)
```

dataIII *Reverse phase protein array data from original cancer specimen*

Description

The data Set is a list of four elements. Expression and background are matrices containing signal intensities, the data frames arraydescription and sampledescription comprising feature and phenodata.

Usage

```
data(dataIII)
```

Format

List

Details

The data set is a list of four elements with data of a original reverse phase array experiment. The elements expression and background are 384 times 75 matrices containing integers describing the signal intensities and local background for every spot of the experiment as generated with image analysis software. Arraydescription is a data frame, describing the incubation of every array refering the column of the expression and background matrix. Required rows are target and AB_ID with characters and array.id (four integers linked with "-"). Sampledescription is a data frame according to the rows of expression and background matrix and annotates the samples.

Source

The data set contains original reverse phase protein array signals from cancer specimen with randomized pheno and feature data.

Examples

```
data(dataIII)
str(dataIII)
```

dataPreproc *Data preprocessing*

Description

Function for import, normalization and quality checks of data prior to the actual analysis. The preprocessing steps include subtraction of dilution series intercepts and FCF normalization. Additionally plots for quality checks are generated including dilutions and BLANK measurements.

Usage

```
dataPreproc(dataDir=getwd(), blocks=12, spot="aushon",
exportNo=3, correct="both", remove_flagged=NULL)
```

Arguments

dataDir	directory of gpr files, slidedescription.txt and sampledescription.txt, default is the current working directory
blocks	see blocksperarray in read.Data, default is 12
spot	see spotter in read.Data, default is "aushon"
exportNo	see exportNo in correctDilinterc, integer of 1-4 defining the linear fit to be used (1: constant, 2: antibody, 3: antibody + slide, 4: antibody + slide + sample), default is 3
correct	"both" applies correctDilinterc to all measurements, including FCF. "none" does not use this BG correction at all. "noFCF" applies correctDilinterc to all but not FCF measurements. The default is "both".
remove_flagged	Either NULL or an integer. If an integer, looks into column Flags of the gpr file and removes samples with flag value less than or equal -remove_flagged from the data tables.

Value

A list of 4 elements is returned.

rawdat	list of 4 raw data elements (expression and background matrices, arraydescription and sampledescription data frames) according to read.Data
cordat	list of 4 elements like rawdat with expression data corrected to dilution intercepts, in case of resulting negative values the absoulte minimum + 1 is added, expression data is without NAs and is reduced to the measurement sample type, background is not corrected to intercepts, as it is not used here. If correct is "noFCF", the FCF measurements stay as in rawdat. If correct is "none", the measurements stay as in rawdat.
normdat	list of 4 elements like cordat with expression as dilution intercept (correct "both" or "noFCF") and FCF normalized foreground data, the neglected background data are renamed here to dummy and should not be used

DIR directory for storing the generated outputs

All output files are stored in an analysis folder labeled by the date of analysis. The txt files Dataexpression and Databackground result from write.Data and store the raw data. The pdf files getIntercepts_Output and anovaIntercepts_Output result from correctDilinterc. getIntercepts_Output shows the derived intercepts and smoothing splines of dilution series in dependence of the dilSeriesID column in sampledescription.txt and the slide/pad/incubationRun/spottingRun columns of the arraydescription matrix. anovaIntercepts_Output.pdf results from the ANOVA in correctDilinterc, comparing different linear models of the dilution series intercepts. The barplot displays the residual sum of squares (RSS) of the individual model fits. It helps to choose the appropriate exportNo parameter. As RSS decreases, the model fits better. Finally, three pdf files for quality checking are returned. QC_dilutioncurve_raw.pdf plots target and blank (2nd antibody only) signals from serially diluted control samples of the raw RPPA data set, see plotQC. QC_targetVSblank_normed.pdf plots blank signals vs. target specific signals of dilution intercept corrected and FCF normalized RPPA data, see plotMeasurementsQC. QC_qqPlot_normed.pdf contains qq-plots of dilution intercept corrected and FCF normalized RPPA data, see plotqq.

Author(s)

Silvia von der Heyde

Examples

```
## Not run:
library(RPPanalyzer)



# get output list
dataDir<-system.file("extdata",package="RPPanalyzer")
res<-dataPreproc(dataDir=dataDir,blocks=12,spot="aushon",exportNo=4,correct="both")

# get individual elements
# raw data
rawdat<-res$rawdat
# dilution intercept corrected data
cordat<-res$cordat
# dilution intercept corrected and FCF normalized data
normdat<-res$normdat
# output directory
DIR<-res$DIR


## End(Not run)
```

---

getErrorModel *Estimates error model parameters var0 (basal variance) and varR (relative variance) and produces a new data.frame with the signals and error model parameters.*

Description

The method is based on a maximum-likelihood estimation. The model prediction is the expected variance given the signal, depending on var0 and varR.

Usage

```
getErrorModel(dataexpression, verbose=FALSE)
```

Arguments

dataexpression	data.frame, standard output from RPPanalyzer's write.Data.
verbose	logical, if TRUE, the function prints out additional information and produces a PDF file in the working directory with the signal vs. variance plots.

Details

The empirical variance estimator is χ^2 distributed with $n-2$ degrees of freedom, where n is the number of technical replicates. The estimated error parameters maximize the corresponding log-likelihood function. At the moment, the code assumes $n = 3$. For cases $n > 3$, the error parameters are slightly overestimated, thus, providing a conservative result. The explicit error model is

$$\sigma^2(S) = \sigma_0^2 + S^2\sigma_R^2 = var0 + varRS^2$$

where S is the signal strength.

Value

data.frame	with columns "slide" (factor, the slide names), "ab" (factor, the antibody/target names), "time" (numeric, the time points), "signal" (numeric, signal values), "var0" (numeric, error parameter for the constant error, equivalent to sigma0^2), "varR" (numeric, error parameter for the relative error, equivalent to sigmaR^2) and other columns depending on the input data.frame

Author(s)

Daniel Kaschek, Physikalisches Institut, Uni Freiburg. Email: daniel.kaschek@physik.uni-freiburg.de

---

HKdata *Reverse phase protein array data of siRNA transfected cell line*

Description

The data Set is a list of four elements. Expression and background are matrices containing signal intensities, the data frames arraydescription and sampledescription comprising feature and pheno-data.

Usage

```
data(HKdata)
```

Format

List

Details

The data set is a list of four elements with data of a original reverse phase array experiment. The elements expression and background are 768 times 21 matrices containing integers describing the signal intensities and local background for every spot of the experiment as generated with image analysis software. Arraydescription is a data frame, describing the incubation of every array refering the column of the expression and background matrix. Required rows are target and AB_ID with characters and array.id (four integers linked with "-"). Sampledescription is a data frame according to the rows of expression and background matrix and annotates the samples.

Source

The data set contains original reverse phase protein array of siRNA transfected cell line with randomized pheno and feature data.

Examples

```
data(HKdata)
str(HKdata)
```

---

logList *Logarithmize (log2) the first two RPPA list elements, i.e. foreground and background signal intensities*

Description

Function to logarithmize (log2) the first two RPPA list elements, i.e. foreground and background signal intensities.

Usage

```
logList(x)
```

Arguments

x	list of 4 elements (`expression` and `background` data matrices, `arraydescription` and `sampledescription` data frames) according to `read.Data`

Value

x.log	list of 4 elements like the input but with log2 values of expression and background matrices

Author(s)

Silvia von der Heyde

Examples

```
## Not run:
library(RPPanalyzer)



# input data
dataDir<-system.file("extdata",package="RPPanalyzer")
x<-dataPreproc(dataDir=dataDir, blocks=12, spot="aushon", exportNo=4)
x.norm<-x$normdat

# get log2 list
x.log<-logList(x.norm)


## End(Not run)
```

---

normalizeRPPA *Normalizes data in an RPPA data list*

Description

Normalizes data in an RPPA data list. Four different normalization methods are provided: using externally measured protein concentration, signals from housekeeping proteins or protein dyes and row normalization.

Usage

```
normalizeRPPA(x, method = "row", normalizer = "housekeeping", useCol = "BCA",
writetable = F,vals="logged")
```

Arguments

x	List containing RPPA data set
method	character string: one of proteinDye,row, housekeeping,extValue
normalizer	character string describing the target in slidedescription that should be used for normalization using housekeeping
useCol	character string describing the column in sampledescription that should be used for normalization using the method extValue.

writetable	logical. If true data are exported as tab delimited text files to current working directory
vals	the data is returned at log2 scale with substracted normalizer value per default. If argument is set to native the median of the normalizer values is added after normalization and the data is returned at native scale.

Details

The function provides four different methods to normalize RPPA data to ensure that an optimal data quality. The default method row uses the expression matrix: after taking the logarithm the row median is substracted from each value of one row assuming that the median expression over all targets of one sample is representing total protein amount of the spots. For the method proteinDye arrays with the pattern protein in the target description are used for normalization. For every spotting run a separate protein slide is required. If the slides containing more than one array, the arrays will be normalized by the corresponding protein array. To use external protein assay data for normalization, a column containing the protein concentration has to be added to the sampledescription file. The name of this column is addressed via the useCol argument. To use any other target for normalization the method housekeeping can be used. The target for this method has to be addressed via the normalizer argument.

Value

expression	matrix with protein expression data
dummy	matrix with protein expression data
arraydescription	data frame with feature data
sampledescription	data frame with pheno data

Author(s)

Heiko Mannsperger <h.mannsperger@dkfz-heidelberg.de>

Examples

```
## Not run:
library(RPPanalyzer)
data(dataI)
dataI_bgcorr <- correctBG(dataI,method="normexp")
dataIb <- pick.high.conc(dataI_bgcorr,highest="dilution")
normRow <- normalizeRPPA(dataIb,method="row")
normDye <- normalizeRPPA(dataIb,method="proteinDye")
normPassay <- normalizeRPPA(dataIb,method="extValue",useCol="concentration")
normHK <-  normalizeRPPA(dataIb,method="housekeeping",normalizer="housekeeping")

## End(Not run)
```

`pick.high.conc` *Select the highest concentration from serialy diluted samples*

Description

Picks the dilution step with the value 1 from serialy diluted samples in an RPPA data set.

Usage

```
pick.high.conc(x, highest = ("dilution"), sample.id=c("sample","sample.n"))
```

Arguments

`x`	Any RPPA data list with 4 elements
`highest`	Character string describing the column that contains the dilution steps
`sample.id`	Attributes to identify the samples

Details

The function selects all spots or samples from a RPPA data set with the value 1 in the column of the sampledescription denoted in argument `highest`.

Value

An RPPA data list containing only the samples with the highest concentration of each dilution series.

Author(s)

Heiko Mannsperger <h.mannsperger@dkfz.de>, Stephan Gade <s.gade@dkfz.de>

Examples

```
## Not run:
   library(RPPanalyzer)
   data(ser.dil.samples)

   dataHighcon <- pick.high.conc(ser.dil.samples,highest="dilution")

## End(Not run)
```

plotMeasurementsQC *Scatter Plots from an RPPA data*

Description

Plots the blank signals and the target specific signals of an RPPA data list in a PDF file.

Usage

```
plotMeasurementsQC(x, file = "QC_plots.pdf", arrays2rm = c("protein"))
```

Arguments

x	RPPA data list as output from read.Data
file	name of the PDF file that will be exported
arrays2rm	character describing the arrays that dont have be plotted

Details

This function genrates scatter plots in a pdf file from not yet normalized samples (annotated as measurement in the sample_type column of the sampledescription file) of RPPA data to get an impression of the distance from the blank signal to the target specific signal. An array with blank as target description is needed.

Value

Genrates a PDF file

Author(s)

Heiko Mannsperger <h.mannsperger@dkfz.de>

Examples

```
## Not run:
library(RPPanalyzer)
data(dataIII)
   plotMeasurementsQC(dataIII,file="control_plot.pdf")

## End(Not run)
```

plotQC *Plot target and blank signal from RPPA control samples*

Description

Plots target and blank signal from control samples of an RPPA data set in one plot. Exports pdf file.

Usage

```
plotQC(x, file = "target_vs_blank.pdf", arrays2rm = c("protein"))
```

Arguments

x	RPPA data list as output from read.Data
file	name of the PDF file
arrays2rm	character describing the arrays that dont have be plotted

Details

This function genrates scatter plots in a pdf file from not yet normalized, serially diluted control samples (annotated as control in the sample_type column of the sampledescription file) of RPPA data to get an impression of the antibody dynamic. An array with blank as target description is needed.

Value

generates a PDF file

Author(s)

Heiko Mannsperger <h.mannsperger@dkfz.de>

Examples

```
## Not run:
library(RPPanalyzer)
data(dataIII)

plotQC(dataIII,file="plotQC.pdf")

## End(Not run)
```

plotqq *qq-plot and qq-line of an RPPA data set*

Description

Draws a qq-plot and qq-line from measurements samples of a RPPA data set

Usage

```
plotqq(x, fileName = "qqplot_and_line.pdf")
```

Arguments

x	RPPA data list as output from read.Data
fileName	name of the PDF file

Details

This function implements the functions qqnorm and qqline from stats package to get an impression of the data distribution in an RPPA data set.

Value

generates a PDF file.

Author(s)

Heiko Mannsperger <h.mannsperger@dkfz.de>

Examples

```
## Not run:
library(RPPanalyzer)
data(dataIII)
plotqq(dataIII,file="dataIII_qqplot.pdf")

## End(Not run)
```

plotTimeCourse *Draw time course from RPPA data*

Description

Draws time course data from a RPPA data list and calculates a mathematical model on the time course data.

Usage

```
plotTimeCourse(x, tc.identifier =
c("sample", "stimulation", "inhibition", "stim_concentration"),
tc.reference=NULL, plot.split = "experiment", file = NULL,
arrays2rm = c("protein", "Blank"), plotformat = "stderr",
log=TRUE, color=NULL, xlim = NULL, ylim = NULL)
```

Arguments

x	List containing RPPA data set
tc.identifier	character string describing the column names in the sampledescription that identifies the individual time course experiments
tc.reference	character string describing the sample that will be used as reference for the time course plots.
plot.split	character string describing the column names in sampledescription that defines the argument that devides between different plots
file	character string for the name of the exported file
arrays2rm	character strings identifying the targets that should be from the time course plots
plotformat	character string defining the plot type: rawdata for plotting the connected medians plus standard deviation of the data, spline and both for plotting the a spline fit through the data or both raw data and spline, as well as a confidence band showing the standard error of the spline fit; spline_noconf only plots the spline without confidence band. errbar will show the spline fit plus raw data without connecting the medians by a line, stderr will show a less crowded version of the spline plus standard error represented as simple error bars (which is the default).
log	logical, if true time courses signal intensities will be plotted at log2 scale
color	Vector holding the colors for the samples to be plot. If NULL, colors will be generated.
xlim	Limits for x-axis. If NULL (default) limits are generated for each timeseries plot. If a range (numeric vector of length 2) is given, this is used for all plots.
ylim	Analogous to xlim for y-axis limits.

Details

This function plots RPPA time course experiments from data sets with aggregated replicate spots. A column `time` containing numeric values is required in the sampledescription file. One or several column in the sampledescription file should be able to indentify the individual experiments described in argument `tc.identifier`. One column should provide a parameter `plot.split` to split the whole data set into different comparable time courses that have to be plotted together.

Different plotting options can be specified with the argument `plotformat`. Option `both` is most informative, since it shows the original data plus standard deviations at each time point, combined with a spline fit and the standard error of the fit.

Value

generates a PDF file

Author(s)

Heiko Mannsperger <h.mannsperger@dkfz.de

Examples

```
## Not run:
library(RPPanalyzer)
data(dataII)
plotTimeCourse(dataII,
tc.identifier=c("sample","stimulation","stim_concentration","inhibition")
        ,plot.split="experiment",plotformat="stderr")
    plotTimeCourse(dataII,
tc.identifier=c("sample","stimulation","stim_concentration","inhibition")
        ,plot.split="experiment",plotformat="errbar")
    plotTimeCourse(dataII,
tc.identifier=c("sample","stimulation","stim_concentration","inhibition")
        ,plot.split="experiment",plotformat="both")
    plotTimeCourse(dataII,
tc.identifier=c("sample","stimulation","stim_concentration","inhibition")
        ,plot.split="experiment",plotformat="rawdata")
    plotTimeCourse(dataII,
tc.identifier=c("sample","stimulation","stim_concentration","inhibition")
        ,plot.split="experiment",plotformat="spline")
    plotTimeCourse(dataII,
tc.identifier=c("sample","stimulation","stim_concentration","inhibition")
        ,plot.split="experiment",plotformat="spline_noconf")


## End(Not run)
```

plotTimeCourseII *Multiplot function for RPPA time course datasets*

Description

plotTimeCourseII creates multiplot rectangular PDF files for time course datasets. Page layout (number of plots per page, arrangement of plots) and plot layout can be customized within the function.

Usage

```
plotTimeCourseII(x,plotgroup="",filename="timeseries_multiplot.pdf",numpage=4,
cols=2,xname="time",yname="signal",legpos="top",legrow=2,legtitle="treatment",
legtitlepos="top",legtextsize=10,legtextcolor="black",legtitlesize=10,
legtitlecolor="black",legtitleface="bold",legitemsize=1,plottitlesize=12,
plottitleface="bold",xaxissize=10,yaxissize=10,xaxisface="bold",
yaxisface="bold",xaxistextsize=8,xaxistextangle=0,yaxistextsize=8,
linecolor="Set1")
```

Arguments

x	RPPA time course dataset preprocessed with the getErrorModel and averageData function
plotgroup	select the feature (eg. treatment) which should be plotted in one plot
filename	enter filename, DIR needs to be defined as your working directory, add .pdf to filename
numpage	number of plots per page
cols	number of plot columns per page
xname	title of the x axis
yname	title of the y axis
legpos	postion of the legend in context of the plot ("top","bottom","right","left"), "none" removes legend from plot
legrow	number of item rows within the legend
legtitle	title of the legend
legtitlepos	position of the legend title
legtextsize	font size of the legend text
legtextcolor	color of the legend text
legtitlesize	font size of the legend title
legtitlecolor	color of the legend title
legtitleface	font face of the legend title (eg. "bold")
legitemsize	size of the legend item pictures
plottitlesize	size of the plot title

plottitleface	font face of the plot title
xaxissize	font size of the x axis title
yaxissize	font size of the y axis title
xaxisface	font face of the x axis title
yaxisface	font face of the y axis title
xaxistextsize	font size of the x axis text
xaxistextangle	angle of the x axis text
yaxistextsize	font size of the y axis text
linecolor	color of the plot lines: either chose a scheme ("Set1","Dark2","Paired") or hand a vector of color names

Details

The plotTimeCourseII function plots RPPA timecourse datasets in multiple line charts. For each cell line and target protein a separate plot is created. The average foldchange values of different replicates and the error bars are visualized. In order to be visualized by the plotTimeCourseII function, the dataset needs to be preprocessed by the getErrorModel and averageData function from the RPPanalyzer package. Additionally the plotgroup needs to be defined if it is not named ?treatment?. The remaining arguments are optional.

Value

Generates a PDF file.

Author(s)

Johannes Bues (j.bues@dkfz-heidelberg.de)

Examples

```
## Not run:

# pre-process the data
dataDir <- system.file("extdata", package="RPPanalyzer")
res <- dataPreproc(dataDir=dataDir, blocks=12, spot="aushon", exportNo=2)
# remove arrays
normdat_rm <- remove.arrays(res$normdat, param="target", arrays2rm=c("protein","blank"))
# select samples and export data
sel_sampels_A549 <- select.sample.group(normdat_rm, params=list("cell_line"="A549"), combine= FALSE)
write.Data(sel_sampels_A549, FileNameExtension="HGF_sample_data_A549")
# read selected data
dataexpression_1 <- read.table("HGF_sample_data_A549expression.txt")
# use getErrorModel function
dataexpression_2 <- getErrorModel(dataexpression_1, verbose=FALSE)
# use averageData function
dataexpression_3 <- averageData(dataexpression_2, scaling=c("slide","replicate"),
distinguish=c("cell_line","treatment"))
# plot time course data
```

```
plotTimeCourseII(dataexpression_3, filename="timecourse_HGF_sample_data_A549.pdf",
legpos="top", xname="time [min]",
yname="signal [a.u.]", linecolor=c("red","green","blue","black","orange","grey"))


## End(Not run)
```

`read.Data` *Read and Annotate RPPA rawdata*

Description

reads sampledescription and slidedescription txt files and annotates the median expression value in GenePix result files stored in current working directory.

Usage

```
read.Data(blocksperarray = 4, spotter = "arrayjet", writetable = FALSE,
printFlags=FALSE,fileName="Flagged_spots.csv", remove_flagged=NULL, ...)
```

Arguments

`blocksperarray`	Integer describing the number of blocks in one array.
`spotter`	character strings: default `arrayjet` or `aushon`.
`writetable`	logical. If true data are exported as tab delimited text files to current working directory
`printFlags`	logical. If true flagged spots will exported as csv file
`fileName`	character string naming the csv file for the flagged spots
`remove_flagged`	Either NULL or an integer. If an integer, looks into column `Flags` of the gpr file and removes samples with flag value less than or equal `-remove_flagged` from the data tables.
`...`	any other arguments passed to read.gpr

Details

This function reads and annotates RPPA rawdata provided in three different kind of files. It is very important that these data files are in a correct format and stored in the same folder.

The file sampledescription.txt has to be a tab delimited text file with at least 6 columns named plate, column, row, sample_type, sample, concentration and in case of serially diluted samples a column dilution is required. The first 3 columns are describing the location of the sample in the source well plate. The 4th column describes the for different types of samples: measurement, control, neg_control or blank. In the column sample any character string describing the sample is possible. The column concentration has to contain only numerical values. Columns with further phenodata can be added.

The slidedescription.txt describes the array properties. Required columns are: gpr (describing the name of the corresponding gpr file), the columns pad, slide, incubation_run, spotting_run containing

integers are generating a unique array identifier. The column target describes the analyzed target and AB_ID the used antibody. Column with further feature data can be added.

The third kind of files are the gpr files as results from image analysis software GenePix using the galfile from a aushon or arrayjet spotter.

Value

`expression`	matrix with protein expression data
`background`	matrix with background data
`arraydescription`	data frame with feature data
`sampledescription`	data frame with pheno data

Author(s)

Heiko Mannsperger <h.mannsperger@dkfz-heidelberg.de>

Examples

```
## Not run:
  library(RPPanalyzer)


  dataDir <- system.file("extdata", package="RPPanalyzer")
  setwd(dataDir)
 rawdata <- read.Data(blocksperarray=12, spotter="aushon", printFlags=FALSE, remove_flagged=NULL)
  print(dim(rawdata$expression))

 rawdata <- read.Data(blocksperarray=12, spotter="aushon", printFlags=FALSE, remove_flagged=50)
  print(dim(rawdata$expression))

## End(Not run)
```

---

remove.arrays *Remove arrays from a RPPA data list*

Description

Removes arrays from the RPPA data set which are not used in following calculations.

Usage

```
remove.arrays(x, param = "target", arrays2rm = c("protein", "blank", "housekeeping"))
```

Arguments

x	List with RPPA data set
param	charater describing a row in the arraydescription (column in slidedescription file)
arrays2rm	character defining the arrays to remove

Value

The RPPA data list without the arrays specified by `arrays2rm`.

Author(s)

Heiko Mannsperger <h.mannsperger@dkfz.de>

Examples

```
## Not run:
        library(RPPanalyzer)
        data(dataIII)

        DT <- remove.arrays(dataIII, param = "target", arrays2rm = c("protein"))

## End(Not run)
```

---

rppa2boxplot *Draws boxplots of groups of an RPPA data set including wilcox or kruskal test.*

Description

Draws boxplots of groups of an RPPA data set and compares the expression values to a reference group (control) if provided (wilcox.test). Otherwise a test on general differences is performed (kruskal.test). Additionally a grouping order for plotting can be provided here.

Usage

```
rppa2boxplot(x, param, control=NULL, orderGrp=NULL, file = "boxplot_groups.pdf")
```

Arguments

x	List with RPPA data with aggregated replicate spots
param	Character value of one of the columns of the sampledescription matrix, i.e. x[[4]], describing the phenodata that should be analyzed
control	Character value of one of the columns of the sampledescription matrix, i.e. x[[4]], describing the sample group of `param` that serves as reference in the wilcoxon test. In case of NULL (default) the general kruskal.test is performed instead.

orderGrp	defines the ordering of the subgroups in param, i.e. vector of specifically ordered values of param
file	Title of the file that will be exported.

Value

Generates a PDF file

Author(s)

Silvia von der Heyde, Heiko Mannsperger

Examples

```
## Not run:
library(RPPanalyzer)

data(dataIII)
dataIII_median <- sample.median(dataIII)
rppa2boxplot(x=dataIII_median, param="rank", control="vx",
orderGrp=c("vx","zx","yzr","rxi"), file="wilcoxonBoxplot.pdf")
rppa2boxplot(x=dataIII_median, param="rank", control=NULL,
orderGrp=c("vx","zx","yzr","rxi"), file="kruskalBoxplot.pdf")

## End(Not run)
```

---

rppaList2ExpressionSet

Convert RPPA data into Expression Set

Description

Converts a RPPA data list into an Expression Set

Usage

```
rppaList2ExpressionSet(x)
```

Arguments

x	List with RPPA data set

Details

This function builds an Expression Set from RPPA data. Due to the design of RPPA experiments, pheno and feature data are inverted compared to DNA/RNA array data sets.

Value

object of class Expressionset

Author(s)

Heiko Mannsperger <h.mannsperger@dkfz.de>

Examples

```
## Not run:
library(RPPanalyzer)
data(dataI)
   dataI_bgcorr <- correctBG(dataI,method="normexp")
dataI_median <- sample.median(dataI_bgcorr)
expr.set <- rppaList2ExpressionSet(dataI_median)

## End(Not run)
```

`rppaList2Heatmap` *Draw a heatmap with column side colors from a RPPA data*

Description

Draws a heatmap from an RPPA data set and adds column side colors visualizing groups of selected phenodata.

Usage

```
rppaList2Heatmap(x, sampledescription = "sample", side.color = "tissue",
remove = c("blank", "protein", "Abmix"), distance = "eucsq",
dendros = "both", cutoff = 0.005, fileName = NULL,
cols = colorpanel(100, low = "blue", mid = "yellow", high = "red"),
hclust.method="ward", scale = "row")
```

Arguments

`x`	List with RPPA data set, aggregatedreplicates
`sampledescription`	character describing the sample identifier
`side.color`	character describing the parameter for the side colors of the heatmap
`remove`	character describing the arrays that should removed from the heatmap data
`distance`	character describing the method for the dendrogram
`dendros`	character: "both" for row and column dendrogram
`cutoff`	numeric describing the percentage that are identified as outliers for the heatmap color distribution

fileName	character for the file where the pdf file will be stored. If NULL, plot to standard plotting device.
cols	color key for the heatmap
hclust.method	The method to be used for cluster agglomeration. Defaults to ward. See help of hclust for options.
scale	String. Either row, column, both or none for row or column, both or no scaling, respectively.

Value

generates a PDF file

Author(s)

Heiko Mannsperger <h.mannsperger@dkfz.de>

Examples

```
## Not run:
library(RPPanalyzer)
data(dataIII)
dataIII_median <- sample.median(dataIII)

rppaList2Heatmap(dataIII_median)

## End(Not run)
```

S1.gpr *GenePix result files*

Description

GenePix result files are tab delimited text files exported from the commonly used microarray image analysis tool GenePix.

Format

tab delimeted text file

Source

The GenePix result files are files from original reverse phase protein arrays

`sample.median` *Aggregate the replicates in an RPPA data set*

Description

Aggregates the replicates in an RPPA data list using the median function.

Usage

```
sample.median(x)
```

Arguments

`x`	List with RPPA data set

Value

`expression`	matrix with protein expression data
`error_mad`	matrix with error values
`arraydescription`	data frame with feature data
`sampledescription`	data frame with pheno data

Author(s)

Heiko Mannsperger <h.mannsperger@dkfz.de>

Examples

```
## Not run:
library(RPPanalyzer)

   data(dataI)
   dataI_bgcorr <- correctBG(dataI,method="normexp")

data.median <- sample.median(dataI_bgcorr)

## End(Not run)
```

sampledescription.old *sample description file*

Description

The sample description file contains all information concerning the samples of a reverse phase protein experiment.

Format

tab delimeted text file

Details

The sample description file contains information for sample annotation and data analysis. To identify the sample in the source well plate the columns `plate`, `row`, `column` are obligatory. It is neccessary that every well that is spottet is described. The columns `sample_type` and `sample` as well as `concentration` and for serially diluted samples `dilution` are required for data analysis. To fit a model to serial dilution e.g. using the `calcSdc` function, it is neccessary to indicate the highest concentration in the `dilution` column with the value 1. Any additionally column can be added to describe further phenodata of interest.

Source

The data set contains original reverse phase protein array signals with randomized pheno and feature data.

sampledescription.txt *sample description file*

Description

The sample description file contains all information concerning the samples of a reverse phase protein experiment.

Format

tab delimeted text file

Details

The sample description file contains information for sample annotation and data analysis. To identify the sample in the source well plate the columns `plate`, `row`, `column` are obligatory. It is neccessary that every well that is spottet is described. The columns `sample_type` and `sample` as well as `concentration` and for serially diluted samples `dilution` are required for data analysis. The column `dilSeriesID` is required for background correction based on serial dilutions. Any additionally column can be added to describe further phenodata of interest.

Source

The data set contains original reverse phase protein array signals. A549 cells were starved for 24 h and subsequently stimulated with six different HGF concentrations ranging from 0 - 100 ng/ml. Samples were obtained at six different time points ranging from 0 - 120 min. The experiment was done in triplicates, and the samples were analysed by RPPA using antibodies directed against proteins and phosphoproteins of MET receptor signalling.

`select.measurements` *Selects the measurement samples from an RPPA data list*

Description

Selects the measurement samples defined as "measurement" in sample_type from an RPPA data list

Usage

```
select.measurements(x)
```

Arguments

`x`	List with RPPA data set

Value

`expression`	matrix with protein expression data
`background`	matrix with protein background data or error values dependend on the input files
`arraydescription`	data frame with feature data
`sampledescription`	data frame with pheno data

Author(s)

Heiko Mannsperger <h.mannsperger@dkfz.de>

Examples

```
## Not run:
library(RPPanalyzer)
data(dataIII)
dataIII_median <- sample.median(dataIII)
measures <- select.measurements(dataIII_median)

## End(Not run)
```

`select.sample.group` *Selects samples from RPPA data*

Description

Selects samples from an RPPA data list according to the selected parameter.

Usage

```
select.sample.group(x, params=list("tissue" = c("T", "N")), combine = F )
```

Arguments

`x`	List with RPPA data set
`params`	List of parameters the selection of samples is bases on. The names of the list describes the columns of the sampledescription matrix. The according values corresponds to the values in these columns that will be selected.
`combine`	Logical value. Indicates wheter the samples should match at least one criterion given in the params list (`combine=TRUE`) or if all criteria should be met (`combine=FALSE`). Default is `FALSE`.

Value

An RPPA data list containing only these samples that match the criteria given in the `params` list.

Author(s)

Heiko Mannsperger <h.mannsperger@dkfz.de>, Stephan Gade <s.gade@dkfz.de>

Examples

```
## Not run:
library(RPPanalyzer)
data(dataII)

selectedData <- select.sample.group(dataII,params=list("stimulation"=c("A","B")))

## End(Not run)
```

ser.dil.samples *Reverse phase protein array rawdata, samples serially diluted*

Description

The data Set is a list of four elements. Expression and background are matrices containing signal intensities, the data frames arraydescription and sampledescription comprising feature and pheno-data.

Usage

```
data(ser.dil.samples)
```

Format

list

Details

The data set is a subset of the data set dataI to shorten the running time during the R CMD check process. The data set contains information about the localization of the samples.

Source

The data set contains original reverse phase protein array signals with randomized pheno and feature data.

Examples

```
## Not run:
data(ser.dil.samples)
str(ser.dil.samples)

## End(Not run)
```

simpleBoxplot *Draws boxplots of groups of an RPPA data set.*

Description

Draws boxplots of groups of an RPPA data set. Additionally a grouping order for plotting can be provided here.

Usage

```
simpleBoxplot(x, param, orderGrp=NULL, file = "boxplot_groups.pdf")
```

Arguments

x	List with RPPA data with aggregated replicate spots
param	Character value of one of the columns of the sampledescription matrix, i.e. x[[4]], describing the phenodata that should be analyzed
orderGrp	defines the ordering of the subgroups in param, i.e. vector of specifically ordered values of param
file	Title of the file that will be exported.

Value

Generates a PDF file

Author(s)

Silvia von der Heyde, Heiko Mannsperger

Examples

```
## Not run:
library(RPPanalyzer)

data(dataIII)
dataIII_median <- sample.median(dataIII)
simpleBoxplot(x=dataIII_median, param="rank",
orderGrp=c("vx","zx","yzr","rxi"), file="simpleBoxplot.pdf")

## End(Not run)
```

---

slidedescription.old *slide description file*

Description

The slide description file contains all information concerning the arrays of a reverse phase protein experiment.

Format

tab delimeted text file

Details

The slide description file contains information for array annotation and data analysis. To find the GenePix result files (gpr files) in current working directory it is neccesssary that the names of the gpr files are matching with the gpr column. To identify the array on the slides the columns pad, slide, spotting_run, incubation_run are obligatory. It is neccessary that every well that is spottet is described. The columns sample_type and sample as well as concentration and (for

serially diluted samples) dilution are required for data analysis. The columns target describes the analyzed proteins and AB_ID contains a indentifier for the antibody used for the detection. Any additionally column can be added to describe further phenodata of interest.

Source

The data set contains the incubation data from reverse phase protein arrays with randomized feature data.

---

slidedescription.txt *slide description file*

Description

The slide description file contains all information concerning the arrays of a reverse phase protein experiment.

Format

tab delimeted text file

Details

The slide description file contains information for array annotation and data analysis. To find the GenePix result files (gpr files) in current working directory it is neccesssary that the names of the gpr files are matching with the gpr column. To identify the array on the slides the columns pad, slide, spotting_run, incubation_run are obligatory. It is neccessary that every well that is spottet is described. The columns sample_type and sample as well as concentration and (for serially diluted samples) dilution are required for data analysis. The columns target describes the analyzed proteins and AB_ID contains a indentifier for the antibody used for the detection. Any additionally column can be added to describe further phenodata of interest.

Source

The data set contains the incubation data from reverse phase protein arrays for the HGF data set. These are 3 sample slides plus one slide for FCF normalization.

`test.correlation` *Tests for correlations in RPPA data*

Description

Tests for correlation between protein expression value and any continuous data using cor.test.

Usage

```
test.correlation(x, param, method.cor = "kendall",
method.padj = "BH", file = "correlation_plot.pdf")
```

Arguments

`x`	List containing RPPa data set
`param`	character describing the parameter
`method.cor`	character string describing the correlation
`method.padj`	character string describing the method for the p-value correction for multiple testing.
`file`	character string

Value

generates a pdf file

Author(s)

Heiko Mannsperger <h.mannsperger@dkfz.de>

See Also

For information about the argument method.cor see `cor.test`, informations about methods.padj can be found under `p.adjust`

Examples

```
## Not run:
library(RPPanalyzer)
data(dataIII)
dataIII_median <- sample.median(dataIII)
test.correlation(dataIII_median,param="staging")


## End(Not run)
```

`write.Data` *writes an RPPA data list into csv file*

Description

Writes the 3 or 4 elements of an RPPA data list into one or two csv files which can easily imported into spreadsheet software

Usage

```
write.Data(x,FileNameExtension="Data")
```

Arguments

`x`	List with RPPA data set
`FileNameExtension`	character string which will be added to the name of the exported file

Value

one or two csv files dependend from the length of the RPPA data list

Author(s)

Heiko Mannsperger <h.mannsperger@dkfz.de>

Examples

```
## Not run:
library(RPPanalyzer)
data(dataII)

write.Data(dataII)

## End(Not run)
```

Index