Package ‘AgiMicroRna’

April 14, 2017

Version 2.24.0
Title Processing and Differential Expression Analysis of Agilent microRNA chips
Author Pedro Lopez-Romero <plopez@cnic.es>
Maintainer Pedro Lopez-Romero <plopez@cnic.es>
Imports Biobase
Depends R (>= 2.10),methods,Biobase,limma,affy (>= 1.22),preprocessCore,affycoretools
Suggests geneplotter,marray,gplots,gtools,gdata,codelink
LazyLoad yes
Description Processing and Analysis of Agilent microRNA data
License GPL-3
biocViews Microarray, AgilentChip, OneChannel, Preprocessing, DifferentialExpression
NeedsCompilation no

R topics documented:

- basicLimma .................................................. 2
- boxplotMicroRna ............................................ 4
- cvArray ..................................................... 5
- dd.micro ................................................... 6
- ddPROC .................................................... 7
- dim.uRNAList ............................................. 7
- dimnames.uRNAList ....................................... 8
- esetMicroRna ............................................... 8
- filterMicroRna ............................................. 9
- getDecideTests ............................................ 11
- HeatMapMicroRna .......................................... 12
- hierclusMicroRna ......................................... 13
- mvaBASIC .................................................. 14
- mvaMicroRna .............................................. 14
- PCAplotMicroRna .......................................... 15
- plotDensityMicroRna ..................................... 16
- pvalHistogram ............................................ 16
- qcPlots .................................................... 17
Linear models Using limma

Description
Differential expression analysis using the linear model features implemented in the limma package. A linear model is fitted to each miRNA gene so that the fold change between different experimental conditions and their standard errors can be estimated. Empirical Bayes methods are applied to obtain moderated statistics.

Usage
basicLimma(eset, design, CM, verbose = FALSE)

Arguments
eset ExpressionSet containing the processed log-expression values
design design matrix
CM contrast matrix
verbose logical, if TRUE prints out output

Details
In our data example (see the target file in Table 1 in vignette), we have used a paired design (by subject) to assess the differential expression between two treatments B and C vs a control treatment A. That is, we want to obtain the microRNAs that are differentially expressed between conditions A vs B and A vs C. The linear model that we are going to fit to every miRNA is defined by equation: 
\[ y = \text{Treatment} + \text{Subject} + \text{error term}. \] 
This model is going to estimate the treatment effect and then, the comparison between the different treatments are done in terms of contrasts between the estimates of the treatment effects. To fit the model, we need first to define a design matrix. The design matrix is an incidence matrix that relates each array/sample/file to its given experimental conditions, in our case, relates each file to one of the three treatments and with its particular subject. If treatment is a factor variable, we can define de desing matrix using model.matrix(~ -1 + treatment + subject). Then the linear model can be fitted using fit=lmFit(eset,design). This will get the treatment estimates for each microRNA in the eset object:
treatmentA treatmentB treatmentC subject2 hsa-miR-152 7.5721 7.656 7.566 -0.1157 hsa-miR-15a* 0.9265 1.066 1.211 -0.2242 hsa-miR-337-5p 6.2448 7.298 7.084 -0.4489
We can define the contrasts of interest using a contrast matrix as in

\[ CM = \text{cbind}(\text{MSC}_A \text{vs} \text{MSC}_B = c(1,-1,0), \text{MSC}_A \text{vs} \text{MSC}_C = c(1,0,-1)) \]

And then, we can estimate those contrasts using

\[ \text{fit2} = \text{contrasts.fit(fit,CM)} \]

Finally, we can obtain moderated statistics using

\[ \text{fit2} = \text{eBayes(fit2)} \]

The function ‘basicLimma’ implemented in AgiMicroRna produces the last fit2 object, that has in

\[ \text{fit2}\$\text{coeff} \] the M values, in \[ \text{fit2}\$\text{t} \] the moderated-t statistic of the contrasts, and in \[ \text{fit2}\$\text{p.value} \] the corresponding p value of each particular contrasts. Be aware that these p values must be corrected by multiple testing.

MSC_A vs MSC_B  MSC_A vs MSC_C  hsa-miR-152 0.67567761 0.977326746  hsa-miR-15a* 0.68019442 0.413657270  hsa-miR-337-5p 0.03737814 0.075248741

See limmaUsersGuide() for a complete description of the limma package.

Value

An MArrayLM object of the package limma

Author(s)

Pedro Lopez-Romero

References


See Also

An ‘RGList’ example containing processed data is in ddPROC and an overview of how the processed data is produced is given in filterMicroRna. The ExpressionSet object can be generated using esetMicroRna

Examples

```r
## Not run:
data(targets.micro)
data(ddPROC)
esetPROC=esetMicroRna(ddPROC,targets.micro,makePLOT=FALSE,verbose=FALSE)

elements.treatment=levels(factor(targets.micro$Treatment))
treatment=factor(as.character(targets.micro$Treatment),
               levels=elements.treatment)

levels.subject=levels(factor(targets.micro$Subject))
subject=factor(as.character(targets.micro$Subject),
               levels=levels.subject)

design=model.matrix(~ -1 + treatment + subject)

CM=cbind(MSC_AvsMSC_B=c(1,-1,0,0),
```
MSC_AvsMSC_C=c(1,0,-1,0))

fit2=basicLimma(esetPROC,design,CM,verbose=TRUE)

names(fit2)
head(fit2$coeff)
head(fit2$p.value)
plot(fit2$Amean,fit2$coeff[,1],xlab="A",ylab="M")
abline(h=0)
abline(h=c(-1,1),col="red")
plot(fit2$coeff[,1],fit2$p.value[,1], xlab="M",ylab="p value")

## End(Not run)

boxplotMicroRna

Boxplot

Description

It creates a Boxplot using the matrix columns as input

Usage

boxplotMicroRna(object, maintitle, colorfill, xlab, ylab)

Arguments

object       A matrix containing by columns the expression arrays in log2 scale
maintitle   character to indicate the title of the graph
colorfill   color to fill the boxplot
xlab        title for the x axe
ylab        title for the y axe

Author(s)

Pedro Lopez-Romero

Examples

data(dd.micro)
MMM=log2(dd.micro$meanS)
boxplotMicroRna(MMM,
maintitle="log2 Mean Signal",
colorfill="green",
xlab="Samples",
ylab="expression")
cvArray

Coefficient of variation of replicated probes within array

Description

Identifies replicated features at probe and at gene level and computes the coefficient of variation of the array

Usage

cvArray(ddDUP, foreground = c("MeanSignal", "ProcessedSignal"), targets, verbose=FALSE)

Arguments

- `ddDUP`: uRNAList, containing the output from `readMicroRnaAFE`
- `foreground`: Specifies the signal used, only "MeanSignal" or "ProcessedSignal" can be used
- `targets`: data.frame with the target structure
- `verbose`: logical, if TRUE prints out output

Details

In the Agilent microRNA platforms the features are replicated at a probe level and normally, a single microRNA is interrogated by either two or four sets of replicated probes. The replication of the probes allows computing the coefficient of variation (CV) for each array as a measure of the reproducibility of the array. The CV is computed for every set of replicated probes and the CV median is reported as the array CV. A lower array CV indicates a better array reproducibility.

Value

It prints out the results of the replication for the NON CONTROL FEATURES at a probe and gene level.

Author(s)

Pedro Lopez-Romero

Examples

```r
## Not run:
data(dd.micro)
data(targets.micro)
cvArray(dd.micro,"MeanSignal",targets.micro,verbose=TRUE)

graphics.off()
## End(Not run)
```
Description

Data, extracted from scanned images using Agilent Feature Extraction Software, are stored in a uRNAList object.

Usage

data(dd.micro)

Details

A data example is provided. The data example includes 3 experimental conditions with two replicates.

For these data, chips were scanned using the Agilent G2567AA Microarray Scanner System (Agilent Technologies) Image analysis and data collection were carried out using the Agilent Feature Extraction 9.1.3.1. (AFE).

Data, collected with the Agilent Feature Extraction Software, are stored in a uRNAList object with the following components:

- `uRNAList$TGS` matrix, ‘gTotalGeneSignal’
- `uRNAList$TPS` matrix, ‘gTotalProbeSignal’
- `uRNAList$meanS` matrix, ‘gMeanSignal’
- `uRNAList$procS` matrix, ‘gProcessedSignal’
- `uRNAList$targets` data.frame, ‘FileName’

For these data, chips were scanned using the Agilent G2567AA Microarray Scanner System (Agilent Technologies) Image analysis and data collection were carried out using the Agilent Feature Extraction 9.1.3.1. (AFE).

Data, collected with the Agilent Feature Extraction Software, are stored in a uRNAList object with the following components:

- `uRNAList$TGS` matrix, ‘gTotalGeneSignal’
- `uRNAList$TPS` matrix, ‘gTotalProbeSignal’
- `uRNAList$meanS` matrix, ‘gMeanSignal’
- `uRNAList$procS` matrix, ‘gProcessedSignal’
- `uRNAList$targets` data.frame, ‘FileName’

Details

A data example is provided. The data example includes 3 experimental conditions with two replicates.

For these data, chips were scanned using the Agilent G2567AA Microarray Scanner System (Agilent Technologies) Image analysis and data collection were carried out using the Agilent Feature Extraction 9.1.3.1. (AFE).

Data, collected with the Agilent Feature Extraction Software, are stored in a uRNAList object with the following components:

- `uRNAList$TGS` matrix, ‘gTotalGeneSignal’
- `uRNAList$TPS` matrix, ‘gTotalProbeSignal’
- `uRNAList$meanS` matrix, ‘gMeanSignal’
- `uRNAList$procS` matrix, ‘gProcessedSignal’
- `uRNAList$targets` data.frame, ‘FileName’

Details

A data example is provided. The data example includes 3 experimental conditions with two replicates.

For these data, chips were scanned using the Agilent G2567AA Microarray Scanner System (Agilent Technologies) Image analysis and data collection were carried out using the Agilent Feature Extraction 9.1.3.1. (AFE).

Data, collected with the Agilent Feature Extraction Software, are stored in a uRNAList object with the following components:

- `uRNAList$TGS` matrix, ‘gTotalGeneSignal’
- `uRNAList$TPS` matrix, ‘gTotalProbeSignal’
- `uRNAList$meanS` matrix, ‘gMeanSignal’
- `uRNAList$procS` matrix, ‘gProcessedSignal’
- `uRNAList$targets` data.frame, ‘FileName’

Author(s)

Pedro Lopez-Romero

See Also

readMicroRnaAFE.Rd
**ddPROC**

**Description**

Filtered and Normalized miRNA data stored in a uRNAList object.

**Usage**

```r
data(ddPROC)
```

**Details**

ddPROC is originated after the processing of the dd.micro raw data.

**Author(s)**

Pedro Lopez-Romero

**See Also**

An overview of how ddPROC is obtained is given in `filterMicroRna`

---

**dim.uRNAList**

**Description**

Retrieve the number of rows (genes) and columns (arrays) for an uRNAList object.

**Usage**

```r
## S3 method for class 'uRNAList'
dim(x)
## S3 method for class 'uRNAList'
length(x)
```

**Arguments**

- `x` an object of class uRNAList

**Details**

This function and this file, has been borrowed from the files created by Gordon Smyth for the limma package.

**Value**

Numeric vector of length 2. The first element is the number of rows (genes) and the second is the number of columns (arrays).
dimnames.uRNAList  Retrieve the Dimension Names of an uRNAList Object

Description

Retrieve the dimension names of a microarray data object.

Usage

```r
## S3 method for class 'uRNAList'
dimnames(x)
## S3 replacement method for class 'uRNAList'
dimnames(x) <- value
```

Arguments

- `x`: an object of class `uRNAList`
- `value`: a possible value for `dimnames(x)`

Details

The dimension names of a microarray object are the same as those of the most important matrix component of that object. A consequence is that `rownames` and `colnames` will work as expected. This function and this file, has been borrowed from the files created by Gordon Smyth for the limma package.

Value

Either `NULL` or a list of length 2.

Author(s)

Pedro Lopez-Romero

esetMicroRna  ExpressionSet object from a uRNAList

Description

It creates an 'ExpressionSet' object from a 'uRNAList' with unique probe names. Typically, the 'uRNAList object' contains the Total Gene Processed data.

Usage

```r
esetMicroRna(uRNAList, targets, makePLOT=FALSE, verbose=FALSE)
```
Arguments

uRNAList  An uRNAList containing normally the processed data
targets    data.frame with the targets structure
makePLOT   logical, if TRUE it makes a 'heatmap' with the 100 greater variance genes, a 'hierarchical cluster' with all the genes and a pca plot
verbose    logical, if TRUE prints out output

Details

It creates an ExpressionSet object from a uRNAList. Usually this function is applied to an uRNAList object containing the Total Gene Processed data.

Value

An ExpressionSet object

Author(s)

Pedro Lopez-Romero

See Also

An 'uRNAList' example containing processed data is in ddPROC and an overview of how the processed data is produced is given in filterMicroRna

filterMicroRna  Filtering Genes

Description

Filter genes out according to their Quality Flag

Usage

filterMicroRna(ddNORM, 
   dd, 
   control, 
   IsGeneDetected, 
   wellaboveNEG, 
   limIsGeneDetected, 
   limNEG, 
   makePLOT, 
   targets, 
   verbose, 
   writeout)
Arguments

ddNORM uRNAList with the Total Gene Signal in log2 scale to be FILTERED out according to a Quality FLAG

dd uRNAList, containing the output from readMicroRnaAFE

control logical, if TRUE it removes controls

IsGeneDetected logical, if TRUE it filters genes according to glsGeneDetected Flag. Flag = 1, then gene is detected

wellaboveNEG logical, if TRUE it filter genes whose expression is not above a limit value defined by the expression of negative controls. Limit = Mean(negative) + 1.5 x sd(negative)

limIsGeneDetected for a given feature xi accros samples, is the minimum in at least one experimental condition with a IsGeneDetected-FLAG = 1 (Is Detected)

limNEG for a given feature xi accros samples, is the minimum in at least one experimental condition with intensity > Limit established for negative controls (Mean + 1.5 x SD)

makePLOT logical, if TRUE makes QC plots with the remaining signals

targets data.frame with the targets structure

verbose logical, if TRUE prints out output

writeout logical, if TRUE writes out output files

Details

Agilent Feature Extraction software provides a flag for each spot that identifies different quantification errors of the signal. Quantification flags were used to filter out signals that did not reach a minimum established criterion of quality.

Value

The function returns a uRNAList containing the FILTERED data. In order to allow the tracking of those microRNAs that may have been filtered out from the original raw data, the following files are given:

NOCtrl_exprs.txt: Log2 Normalized Total Gene Signals for the Non Control Genes
NOCtrl FlagIsGeneDetected.txt: IsGeneDetected Flag for the Non Control Genes, 1 = detected IsNOTGeneDetected.txt: Genes that not are not detected according to IsGeneDetected Flag

Author(s)

Pedro Lopez-Romero

References


Examples

data(dd.micro,verbose=FALSE)
data(targets.micro,verbose=FALSE)
ddTGS=tgsMicroRna(dd.micro,half=TRUE,makePLOT=FALSE,verbose=FALSE)
ddNORM=tgsNormalization(ddTGS,'quantile')
getDecideTests

Differential expression analysis an multiplicity of the tests

Description
It Uses the decideTests function of the 'limma' package to classify the list of genes as up, down or not significant after correcting by the multiplicity of the tests.

Usage
getDecideTests(fit2, DEmethod, MTestmethod, PVcut, verbose=FALSE)

Arguments
fit2 MArrayLM object
DEmethod method for decideTests, only 'separate' or 'nestedF' are implemented. see decideTests in limma package.
MTestmethod method for multiple test, choices are 'none', 'BH', 'BY', ... see p.adjust
PVcut p value threshold to declare significant features
verbose logical, if TRUE prints out output

Value
A 'TestResults' object of the 'limma' package. It prints out the number of UP and DOWN genes for every contrasts according to the p value limit specified.

Author(s)
Pedro Lopez-Romero

References

See Also
An overview of miRNA differential expression analysis is given in basicLimma
Examples

```r
## Not run:
DE=getDecideTests(fit2,
     DEmethod="separate",
     MTestmethod="BH",
     PVcut=0.10,
     verbose=TRUE)
## End(Not run)
```

---

Description

Creates a HeatMap graph using the `heatmap.2` function

Usage

```r
HeatMapMicroRna(object, size, maintitle)
```

Arguments

- `object`: A expression Matrix
- `size`: number of highest variance genes to be considered in the plot
- `maintitle`: title of the plot

Author(s)

Pedro Lopez-Romero

See Also

`heatmap.2`

Examples

```r
data(ddPROC)
HeatMapMicroRna(ddPROC$TGS,
    size=100,
    maintitle="100 High Var genes")
```
Hierarchical clustering

Description
Hierarchical cluster of samples using the 'hclust' function

Usage
hierclusMicroRna(object, GErep, methdis, methclu, sel, size)

Arguments
object An expression Matrix
GErep Numerical vector that relates each sample with its experimental condition
methdis the distance measure to be used. Options are 'euclidean' and 'pearson'. see 'dist' function
methclu the agglomeration method to be used by the 'hclust' function
sel logical, if TRUE selects the 'size' highest variance genes for the plot
size selects the 'size' highest variance genes for the plot if 'sel=TRUE'

Author(s)
Pedro Lopez-Romero

See Also
hclust,dist

Examples

data(targets.micro)
data(ddPROC)
hierclusMicroRna(ddPROC$TGS, GErep, methdis="euclidean",
methclu="complete",
    sel=FALSE,100)
### mvaBASIC

**MVA plot**

**Description**

For each array, the M value is computed for every spot as the difference between the spot intensity in the array and the averaged intensity for that feature over the whole set of arrays. It does not make a distinction between the different kind of features in the array as the mvaMicroRna() does.

**Usage**

```r
mvaBASIC(object, colorfill, maintitle)
```

**Arguments**

- `object`: An expression matrix in log2 scale
- `colorfill`: color of the plot
- `maintitle`: title of the plot

**Author(s)**

Pedro Lopez-Romero

**Examples**

```r
data(dd.micro)
op=par(mfrow=c(1,1),ask=TRUE)
mvaBASIC(log2(dd.micro$meanS),
          colorfill="red",
          maintitle=" log2 Mean Signal")
par(op)
```

### mvaMicroRna

**MA plot**

**Description**

For each array, the M value is computed for every spot as the difference between the spot intensity in the array and the median intensity for that feature over the whole set of arrays. Every kind of feature is identified with different color (microRNA genes, positive controls, etc ...) The input must be an uRNAList object created by the user, in such a way that the uRNAList$meanS field contains the expression matrix that we want to use in log2 scale (see example below) The gProcessedSignal computed by the Agilent Feature Extaction software normally contains negative values, so a small constant has to be added to the signals before log tranformation.

**Usage**

```r
mvaMicroRna(uRNAList, maintitle, verbose=FALSE)
```
Arguments

uRNAList A uRNAList object. It uses the expression matrix stored in the uRNAList$meanS slot. Input expression matrix should be in log2 scale
maintitle character to indicate the title of the graph
verbose logical, if TRUE it prints details

Author(s)

Pedro Lopez-Romero

Examples

data(dd.micro)
op=par(mfrow=c(1,1),ask=TRUE)

MMM=dd.micro$procS ## gProcessedSignal

min=min(MMM) ## transforming gProcessedSignal to positive values
for(i in 1:dim(MMM)[2]){ ## before log2 transformation
    MMM[,i]=MMM[,i]+(abs(min)+ 5)
}

ddaux=dd.micro

ddaux$meanS=log2(MMM)
mvaMicroRna(ddaux,maintitle="ProcessedSignal",verbose=FALSE)

rm(ddaux)

par(op)

PCAplotMicroRna

Description

It is a wrapper for the 'plotPCA' of the 'affycoretools' package

Usage

PCAplotMicroRna(eset, targets)

Arguments

eset An Expression Set object
targets data.frame with the target structure

Author(s)

Pedro Lopez-Romero

Examples

data(targets.micro)
data(ddPROC)
esetPROC=eSetMicroRna(ddPROC,targets.micro,makePLOT=FALSE,verbose=FALSE)

PCAplotMicroRna(esetPROC,targets.micro)
plotDensityMicroRna  Density Plots of Intensity Signals

Description

Creates a density plot with the arrays intensities

Usage

plotDensityMicroRna(object, maintitle)

Arguments

object  An expression matrix, in log2 scale
maintitle  title of the plot

Author(s)

Pedro Lopez-Romero

Examples

data(dd.micro)
    plotDensity(log2(dd.micro$meanS),maintitle="log2 Mean Signal")

pvalHistogram  Histogram of the p values

Description

Creates an histogram of the pvalues. For multiple contrats, creates an histogram for every t.test pvalue (separate) or a single histogram for the F.test pvalue (nestedF). A uniform histogram will indicate no differential expression in the data set, whereas a right skewed histogram, will indicate some significant differential expression

Usage

pvalHistogram(fit2, DE, PVcut, DEmethod, MTestmethod, CM, verbose=FALSE)

Arguments

fit2  MAarrayLM object
DE  TestResults object
PVcut  limit p value to declare significant features
DEmethod  method for decideTests, only ’separate’ or ‘nestedF’ are implemented
MTestmethod  method for multiple test
CM  contrast matrix
verbose  logical, if TRUE prints out output
Author(s)

Pedro Lopez-Romero

See Also

An overview of miRNA differential expression analysis is given in basicLimma An example of how to get the 'TestResults' object is in getDecideTests

Examples

```r
## Not run:
pvalHistogram(fit2,DE,PVcut=0.10,
  DEmethod="separate",MTestmethod="BH",CM)
## End(Not run)
```

**qcPlots**

Plots for Quality Assessment

**Description**

It creates BoxPlots, Density Plots, MA plots, RLE plots and hierachical clustering plots with the sample data set.

**Usage**

```r
qcPlots(dd,
offset,
MeanSignal=TRUE,
ProcessedSignal=FALSE,
TotalProbeSignal=FALSE,
TotalGeneSignal=FALSE,
BGMedianSignal=FALSE,
BGUsed=FALSE,
targets)
```

**Arguments**

- `dd` A uRNAList object containing the output from readMicroRnaAFE
- `offset` numeric value to add to the intensities before log transforming
- `MeanSignal` logical, if TRUE "gMeanSignal" is used
- `ProcessedSignal` logical, if TRUE "gProcessedSignal" is used
- `TotalProbeSignal` logical, if TRUE "gTotalProbeSignal" is used
- `TotalGeneSignal` logical, if TRUE "gTotalGeneSignal" is used
- `BGMedianSignal` logical, if TRUE "gBGMedianSignal" is used
- `BGUsed` logical, if TRUE "gBGUsed" is used
- `targets` data.frame with the target structure
**Details**

The signals loaded from the AFE data files can be used for the quality assessment using the graphical utilities included in the `qcPlots` function. For the `gMeanSignal`, the BoxPlots, Density Plots, MA plots, RLE plots and hierarchical clustering plots are done. For the `gProcessedSignal` the same plots are done, except the hierarchical clustering. For the `gTotalProbeSignal` and the `gTotalGeneSignal` only the BoxPlots and Density Plots are done, and finally, for the Background signals only the Boxplots are done.

**Author(s)**

Pedro Lopez-Romero

**References**


**See Also**

`boxplotMicroRna`, `plotDensityMicroRna`, `RleMicroRna`, `mvaMicroRna` and `hierclusMicroRna`

**Examples**

```r
## Not run:
data(dd.micro)
qcPlots(dd.micro, offset=5,
       MeanSignal=TRUE,
       ProcessedSignal=TRUE,
       TotalProbeSignal=TRUE,
       TotalGeneSignal=TRUE,
       BGMedianSignal=TRUE,
       BGUsed=TRUE,
       targets.micro)
graphics.off()

## End(Not run)
```

---

**readMicroRnaAFE**

*Read Agilent Feature Extraction txt data files*

**Description**

Read the data files generated by the Agilent Feature Extraction image analysis software

**Usage**

```r
readMicroRnaAFE(targets, verbose=FALSE)
```
**Arguments**

**targets**
A data frame that specifies experimental conditions under which each sample has been obtained.

**verbose**
logical, if TRUE prints out output

**Details**

The function reads the *.txt files generated by the AFE Software using the `read.maimages` function of `limma` package.

Data, collected with the Agilent Feature Extraction Software, are stored in a `uRNAList` object with the following components:

- `dd.micro\$TGS` `gTotalGeneSignal`
- `dd.micro\$TPS` `gTotalProbeSignal`
- `dd.micro\$meanS` `gMeanSignal`
- `dd.micro\$procS` `gProcessedSignal`
- `dd.micro\$targets` `targets`
- `dd.micro\$ProbeName` `Probe Name`
- `dd.micro\$GeneName` `microRNA Name`
- `dd.micro\$ControlType` 'FLAG to specify the sort of feature'
- `dd.micro\$IsGeneDetected` 'FLAG IsGeneDetected'
- `dd.micro\$IsSaturated` 'FLAG IsSaturated'
- `dd.micro\$IsFeatNonUnifOL` 'FLAG IsFeatNonUnifOL'
- `dd.micro\$IsFeatPopnOL` 'FLAG IsFeatPopnOL'
- `dd.micro\$gBGMedianSignal` 'gBGMedianSignal'
- `dd.micro\$gBGUsed` 'gBGUsed'

**Value**

A `uRNAList` containing the following elements:

- `uRNAList\$TGS` matrix, 'gTotalGeneSignal'
- `uRNAList\$TPS` matrix, 'gTotalProbeSignal'
- `uRNAList\$meanS` matrix, 'gMeanSignal'
- `uRNAList\$procS` matrix, 'gProcessedSignal'
- `uRNAList\$targets` data.frame, 'FileName'
- `uRNAList\$ProbeName` character, 'Agilent Probe Name'
- `uRNAList\$GeneName` character, 'microRNA Name'
- `uRNAList\$ControlType` integer, '0'= Feature, '1'= Positive control, '-1'= Negative control
- `uRNAList\$IsGeneDetected` matrix, FLAG to classify signal if 'IsGeneDetected=1' or 'not=0'
- `uRNAList\$IsSaturated` matrix, FLAG to classify signal if 'IsSaturated = 1' or 'not=0'
- `uRNAList\$IsFeatPopnOL` matrix, FLAG to classify signal if 'gIsFeatPopnOL = 0' or 'not=1'
- `uRNAList\$IsFeatNonUnifOL` matrix, 'gIsFeatNonUnifOL'
- `uRNAList\$gBGMedianSignal` matrix, 'gBGMedianSignal'
- `uRNAList\$gBGUsed` matrix, 'gBGUsed'
readTargets

Author(s)
Pedro Lopez-Romero

References

See Also
A data example can be found in dd.micro See also readTargets to see how to build the target file and the example given in targets.micro

Examples
```r
## Not run:
data(targets.micro)
dd.micro = readMicroRnaAFE(targets.micro)
## End(Not run)
```

Description
The target file is a txt file created by the user where every input file (array, sample) is attached to a experimental condition

Usage
readTargets(infile,verbose=FALSE)

Arguments
infile name of the target file, for instance 'targets.micro.txt'
verbose logical, if TRUE prints out output

Details
In the 'target' file (see Table 1 in vignette) we specify the experimental conditions under which the data have been generated. The target file MUST contain the following mandatory columns:
- FileName : Name of the array data file -Treatment : Treatment effect -GErep : Treatment effect in numeric code, from '1' to 'n', being 'n' the number of the levels of the treatment effect
Other explanatory variables specifying the experimental conditions might be also included.

Value
A 'data.frame' containing by the columns specified in the input file targets.txt. This 'targets.txt' file must be created by the user.
**Author(s)**

Pedro Lopez-Romero

**References**


**See Also**

An example of a target file can be found in targets.micro

---

**RleMicroRna**  
*Relative Log Expression*

**Description**

RLE: Relative Log Expression

**Usage**

```r
RleMicroRna(object, maintitle, colorfill)
```

**Arguments**

- `object`: An expression matrix
- `maintitle`: title of the plot
- `colorfill`: color of the plot

**Details**

Each Boxplot corresponds to a sample and displays the Relative Log Expression computed for every spot in the array as the difference between the spot intensity and the median intensity for the same feature across all the arrays. Since majority of the spots are expected not to be differentially expressed, the plot should show boxplots centered around zero and all of them having approximately the same dispersion. An array showing greater dispersion than the other, or being not centered at zero could have quality problems.

**Author(s)**

Pedro Lopez-Romero

**References**

Examples

data(dd.micro)
RleMicroRna(log2(dd.micro$meanS),
maintitle="log2 Mean Signal RLE",
colorfill="orange")

rmaMicroRna  Getting the Total Gene Signal by RMA algorithm

Description

The function creates an uRNAList containing the TotalGeneSignal computed by the RMA algorithm. This signal can be used for the statistical analysis.

Usage

rmaMicroRna(dd, normalize, background)

Arguments

dd  uRNAList, containing the output from readMicroRnaAFE
normalize  logical, if TRUE the signal is normalized between arrays using the 'quantile' method
background  logical, if TRUE the signal is background corrected by fitting a normal + exponential convolution model to a vector of observed intensities

Details

The function creates an uRNAList output that contains in the uRNAList$TGS, uRNAList$TPS, uRNAList$meanS & uRNAList$procS slots the Total Gene Signal (TGS) computed by the RMA algorithm. The function uses the robust multiarray average (RMA) method from the 'affy' package. RMA obtains an estimate of the expression measure for each gene using all the replicated probes for that gene. First, RMA obtains a background corrected intensity by fitting a normal + exponential convolution model to a vector of observed intensities. The normal part represents the background and the exponential part represents the signal intensities. Then the arrays are normalized using 'quantile' normalization. Finally, for each probe set that interrogates the same microRNA, RMA fits a linear model to the background-corrected, normalized and log2 transformed probe intensities. This model produces an estimate of the gene signal taking into account the probe effect. The model parameters estimates are obtained by median polish. The estimates of the gene expression signals are referred as RMA estimates. Normally, each microRNA is interrogated by 16 probes either using 2 different probes, each of them replicated 8 times, or using 4 different probes replicated 4 times. First, function rmaMicroRna obtains a background corrected signal using the rma.background.correct function of the package preprocessCore, then the signal is normalized between arrays using the limma function normalizeBetweenArrays with the 'quantile' method. Then, the median of the replicated probes is obtained, leading to either 2 or 4 different measures for each gene. These measures correspond to different probes for the same genes that are summarized into a single RMA linear model described above.

Value

uRNAList containing the Total Gene Signal computed by the RMA algorithm in log 2 scale.
Author(s)

Pedro Lopez-Romero

References


Bolstad B. M. (). preprocessCore: A collection of pre-processing functions. R package version 1.4.0


Examples

data(dd.micro)
ddTGS.rma=rmaMicroRna(dd.micro, normalize=TRUE, background=TRUE)
dim(ddTGS.rma)
RleMicroRna(ddTGS.rma$TGS,"RLE TGS.rma","blue")

significantMicroRna   Summarize Differential Expression Results

Description

The function summarizes the results from the differential expression analysis using the different objects that are obtained after 'limma' has been used, such as the 'MArrayLM' object with the statistics and the 'TestResults' object highlighting the significant features.

Usage

significantMicroRna(eset, ddset, targets, fit2, CM, DE, DEmethod, MTestmethod, PVcut, Mcut, verbose=FALSE)

Arguments

eset   ExpressionSet containing the Total Gene processed data
ddset   An uRNAList object containing the Total Gene processed data
targets  data.frame with the target structure
fit2    MArrayLM object from eBayes 'limma' function
CM      Contrast matrix
DE      TestResults object
DEmethod method used in decideTests, only 'separate' or 'nestedF' are implemented
MTestmethod method for multiple test
PVcut   p value threshold to declare significant features
Mcut    M value threshold to select within significant features
verbose logical, if TRUE prints out output
Details

A list containing the genes with their statistics is generated. The significant genes above the PVcut p values are also given in a html file that links the selected miRNAs to the miRBase [http://microrna.sanger.ac.uk/](http://microrna.sanger.ac.uk/). A MA plots indicating the differentially expressed genes are also displayed.

When multiple contrasts are done, the method for the selection of the significant genes can be either 'separated' or 'nestedF'. See decideTests in package limma [limma](https://www.bioconductor.org/packages/release/bioc/html/limma.html) for a detailed description on these two methods. When 'separated' is used a list with all the genes that have been analyzed in limma is given. The list includes de following columns:

- **PROBE** - Probe name (one of the probes interrogating the gene)
- **GENE** - miRNA name
- **chr\_coord** - Agilent chromosomal location
- **M** - Fold change
- **A** - Mean of the intensity for that miRNA
- **t** - moderated t-statistic
- **pval** - p value of the t-statistic
- **adj.pval** - p value adjusted by 'MTest-method'
- **fdr.pval** - p value adjusted by fdr

Some times, the user can be set 'MTestmethod = none', in this case, it might be interesting to still see the fdr value, despite of the fact that the user has decided not apply any multiple testing correction.

If the 'nestedF' is used, then two lists are provided for each contrasts. A first containing the selected significant genes, and a second list containing the rest of the genes that have been analyzed. The columns given in this case is:

- **PROBE** - Probe name (one of the probes interrogating the gene)
- **GENE** - miRNA name
- **chr\_coord** - Agilent chromosomal location
- **M** - Fold change
- **A** - Mean of the intensity for that miRNA
- **t** - moderated t-statistic
- **pval** - p value of the t-statistic
- **F** - F statistic (null hypothesis: Ci = Cj, for all contrasts i, j)
- **adj.F.pval** - F p value adjusted by 'MTestmethod'
- **fdr.F.pval** - F p value adjusted by fdr

The html files, both for the 'separated' and 'nestedF' method, includes only the selected as significant genes.

Author(s)

Pedro Lopez-Romero

References


miRBase: the home of microRNA data [http://microrna.sanger.ac.uk/](http://microrna.sanger.ac.uk/)

See Also

A 'uRNAList' example containing proccesed data is in ddPROC and an overview of how the processed data is produced is given in filterMicroRna. The ExpressionSet object can be generated using esetMicroRna An overview of miRNA differential expression analysis is given in basicLimma An example of how to get the 'TestResults' object is in getDecideTests

Examples

data(targets.micro)
data(ddPROC)
esetPROC=esetMicroRna(ddPROC,targets.micro,makePLOT=FALSE)
levels.treatment=levels(factor(targets.micro$Treatment))
treatment=factor(as.character(targets.micro$Treatment),
    levels=levels.treatment)

levels.subject=levels(factor(targets.micro$Subject))
subject=factor(as.character(targets.micro$Subject),
    levels=levels.subject)

design=model.matrix(~ -1 + treatment + subject )

CM=cbind(MSC_AvsMSC_B=c(1,-1,0,0),
    MSC_AvsMSC_C=c(1,0,-1,0))

fit2=basicLimma(esetPROC,design,CM,verbose=TRUE)

DE=getDecideTests(fit2,
    DEmethod="separate",
    MTestmethod="BH",
    PVcut=0.10)

significantMicroRna(esetPROC,
    ddPROC,
    targets.micro,
    fit2,
    CM,
    DE,
    DEmethod="separate",
    MTestmethod="BH",
    PVcut=0.10,
    Mcut=0,
    verbose=TRUE)

---

**summary.uRNAList**

**Summaries of Microarray Data Objects**

**Description**

Briefly summarize microarray data objects.

**Usage**

```r
## S3 method for class 'uRNAList'
summary(object, ...)
```

**Arguments**

- `object` : an object of class uRNAList
- `...` : other arguments are not used

**Details**

The data objects are summarized as if they were lists, i.e., brief information about the length and type of the components is given. This function and this file, has been borrowed from the files created by Gordon Smyth for the limma package.
Value
A table.

Author(s)
Pedro Lopez-Romero

Example of target file

Description
Example of target file

Usage
data(targets.micro)

Format
A data frame with 4 observations on the following 5 variables.
FileName names of the Files Ast.txt Bst.txt Aunst.txt Bunst.txt
Treatment Assigns level for Treatment Effect to each File (mandatory)
GErep a numeric vector that numerates the FACTOR of the Treatment Effect (mandatory)
Subject Assigns level for Subject Effect to each File

Details
It is a tab-delimited text format file. The target file is created by the user with the intention of carrying out a differential expression analysis in future steps using 'limma'. Here is where the factors that are going to be included in the linear model that is fitted to each gen are specified. The targets file assigns each data file to a particular experimental conditions. First column 'FileName' is mandatory and includes the image data files names. Second column 'Treatment' is also mandatory and includes the image data files names. Second column 'Treatment' is also mandatory and includes the treatment effect. Third column 'GErep' is also mandatory, and includes the treatment effect in a numeric code, from 1 to n, being n the number of Treatment effect levels.

Author(s)
Pedro Lopez-Romero

References

See Also
readTargets
**tgsMicroRna**

---

**Getting the Total Gene Signal**

**Description**

The function creates an uRNAList containing the TotalGeneSignal computed by the Agilent Feature Extraction software. This signal can be used for the statistical analysis after a possible normalization step.

**Usage**

```r
tgsMicroRna(dd, offset, half, makePLOT=FALSE, verbose=FALSE)
```

**Arguments**

- `dd`: uRNAList, containing the output from `readMicroRnaAFE`
- `offset`: integer. To use this option set `half = FALSE`
- `half`: logical, if `TRUE` half option is used
- `makePLOT`: logical, if `TRUE` QC plots with the Total Gene Signal are displayed
- `verbose`: logical, if `TRUE` prints out some summary results

**Details**

The function creates a uRNAList object that contains in the uRNAList$TGS, uRNAList$TPS, uRNAList$meanS & uRNAList$procS the Total Gene Signal (TGS) as computed by the Agilent Feature Extraction algorithms. This TGS is not in log2 scale. All the replicated genes have the same estimated TGS, and the function simply picks one gene from each set of replicated genes. To maintain the format of the uRNAList, every selected gene retains a probe name attach to them. This probe name is not meaningful any more, since the signal corresponds to the total gene signal and not to the probe signal. The TGS processed by AFE contains some negative values. To get signals with positive values we can either add a positive small constant to all the signals (offset) or we can select the 'half' option, which set to 0.5 all the values that are smaller than 0.5. To use the offset option we have to set `half=FALSE`, otherwise the half method is used by default. The offset option, adds to each signal the quantity (abs( min(ddTGS$TGS)) + offset), where ddTGS$TGS is the matrix that contains the TotalGeneSignal.

**Value**

uRNAList containing the TotalGeneSignal computed by the Agilent Feature Extraction software. Optionally, it can generate a boxplot, a density plot and a MA plot with the Total Gene Signal.

**Author(s)**

Pedro Lopez-Romero

**References**

Examples

data(dd.micro)
data(targets.micro)
ddTGS = tgsMicroRna(dd.micro, half = TRUE, makePLOT = FALSE, verbose = FALSE)

tgsNormalization

Normalization Between Arrays

Description

Normalization between arrays of the Total Gene Signal. The function is a wrapper of the 'limma' 'normalizeBetweenArrays' with ('none','quantile','scale') methods

Usage

tgsNormalization(ddTGS, NORMmethod = "quantile", makePLOTpre = FALSE, makePLOTpost = FALSE, targets, verbose = FALSE)

Arguments

ddTGS uRNAList, containing the output from tgsMicroRna
NORMmethod character specifying the normalization method, 'none','quantile','scale'. The default is quantile
makePLOTpre logical, if TRUE QC plots with the Raw Total Gene Signal are displayed
makePLOTpost logical, if TRUE QC plots with the Normalized Total Gene Signal are displayed
targets data.frame with the target structure
verbose logical, if TRUE prints out output

Value

A uRNAList object containing the Normalized Total Gene Signal in log 2 scale

Author(s)

Pedro Lopez-Romero

References


## Examples

```r
## Not run:
data(dd.micro)
data(targets.micro)
ddTGS=tgsMicroRna(dd.micro,half=TRUE,makePLOT=FALSE,verbose=FALSE)
ddNORM=tgsNormalization(ddTGS, 'quantile',
    makePLOTpre=FALSE,makePLOTpost=TRUE,targets.micro,verbose=TRUE)
graphics.off()
## End(Not run)
```

### uRNAList-class

#### uRNAList - class

**Description**

A list-based class (similar to the RList class in limma package) for the storing of Agilent chips microRNA data. `uRNAList` objects are created by `read.agiMicroRna`

**uRNAList Components**

`uRNAList` objects are created by `new("uRNAList", Newagi)` where `Newagi` is a list with the following components:

- `uRNAList$TGS` matrix, 'gTotalGeneSignal'
- `uRNAList$TPS` matrix, 'gTotalProbeSignal'
- `uRNAList$meanS` matrix, 'gMeanSignal'
- `uRNAList$procS` matrix, 'gProcessedSignal'
- `uRNAList$targets` data.frame, 'FileName'
- `uRNAList$genes$ProbeName` vector of characters, 'AGilent Probe Name'
- `uRNAList$genes$GeneName` vector of characters, 'microRNA Name'
- `uRNAList$genes$ControlType` vector of integers, '0'= Feature, '1'= Positive control, '-1'= Negative control
- `uRNAList$other$gIsGeneDetected` matrix, FLAG to classify signal if 'IsGeneDetected=1' or 'not=0'
- `uRNAList$other$gIsSaturated` matrix, FLAG to classify signal if 'IsSaturated = 1' or 'not=0'
- `uRNAList$other$gIsFeatPopnOL` matrix, FLAG to classify signal if 'IsFeatPopnOL = 0' or 'not=1'
- `uRNAList$other$gIsFeatNonUnifOL` matrix, FLAG to classify signal if 'gIsFeatNonUnifOL = 0' or 'not=1'
- `uRNAList$other$gBGMedianSignal` matrix, gBGMedianSignal
- `uRNAList$other$gBGUsed` matrix, gBGUsed

**Author(s)**

Pedro Lopez-Romero
writeEset

## Not run:
data(dd.micro)

## End(Not run)

writeEset

Writes the expression data matrix of an ExpressionSet object in a file

Description

Writes the expression data matrix of an ExpressionSet object in a file.

Usage

writeEset(eset, ddPROC, targets, verbose = FALSE)

Arguments

eset An Expression object, normally containing the processed data
ddPROC An RGLlist object, normally containing the processed data
targets data.frame with the targets structure
verbose logical, if TRUE prints out output

Details

Writes the expression data matrix of an ExpressionSet object in a file.

Author(s)

Pedro Lopez-Romero

See Also

An 'RGList' example containing processed data is in ddPROC and an overview of how the processed data is produced is given in filterMicroRna. The ExpressionSet object can be generated using esetMicroRna

Examples

## Not run:
data(ddPROC)
data(targets.micro)
esetPROC = esetMicroRna(ddPROC, targets.micro, makePLOT = TRUE, verbose = FALSE)
writeEset(esetPROC, ddPROC, targets.micro, verbose = TRUE)

## End(Not run)
Index

*Topic array
  dim.uRNAList, 7
  dimnames.uRNAList, 8
*Topic classes
  uRNAList-class, 29
*Topic datasets
  dd.micro, 6
  ddPROC, 7
  targets.micro, 26
*Topic documentation
  basicLimma, 2
  boxplotMicroRna, 4
  cvArray, 5
  esetMicroRna, 8
  filterMicroRna, 9
  getDecideTests, 11
  HeatMapMicroRna, 12
  hierclusMicroRna, 13
  mvaBASIC, 14
  mvaMicroRna, 14
  PCAplotMicroRna, 15
  plotDensityMicroRna, 16
  pvalHistogram, 16
  qcPlots, 17
  readMicroRnaAFE, 18
  readTargets, 20
  RleMicroRna, 21
  rmaMicroRna, 22
  significantMicroRna, 23
  tgsMicroRna, 27
  tgsNormalization, 28
  writeEset, 30
*Topic methods
  summary.uRNAList, 25
*Topic utilities
  basicLimma, 2
  boxplotMicroRna, 4
  cvArray, 5
  esetMicroRna, 8
  filterMicroRna, 9
  getDecideTests, 11
  HeatMapMicroRna, 12
  hierclusMicroRna, 13
  mvaBASIC, 14
  mvaMicroRna, 14
  PCAplotMicroRna, 15
  plotDensityMicroRna, 16
  pvalHistogram, 16
  qcPlots, 17
  readMicroRnaAFE, 18
  readTargets, 20
  RleMicroRna, 21
  rmaMicroRna, 22
  significantMicroRna, 23
  tgsMicroRna, 27
  tgsNormalization, 28
  writeEset, 30
<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>readMicroRnaAFE</td>
<td>18</td>
</tr>
<tr>
<td>readTargets</td>
<td>20</td>
</tr>
<tr>
<td>RleMicroRNA</td>
<td>21</td>
</tr>
<tr>
<td>rmaMicroRNA</td>
<td>22</td>
</tr>
<tr>
<td>show,uRNAList-method</td>
<td>29</td>
</tr>
<tr>
<td>significantMicroRNA</td>
<td>23</td>
</tr>
<tr>
<td>summary,uRNAList</td>
<td>25</td>
</tr>
<tr>
<td>targets.micro</td>
<td>26</td>
</tr>
<tr>
<td>tgsMicroRNA</td>
<td>27</td>
</tr>
<tr>
<td>tgsNormalization</td>
<td>28</td>
</tr>
<tr>
<td>uRNAList-class</td>
<td>29</td>
</tr>
<tr>
<td>writeEset</td>
<td>30</td>
</tr>
</tbody>
</table>