affycomp

November 11, 2009

R topics documented:

`affycomp.compfigs.auxiliary`
Auxiliary functions to create comparitive Figures

Description

These functions are auxiliary function to `affycompPlot`. These Figures are used to compare expression measures. They take lists with components created by the `assessDilution` and `assessSpikeIn` functions.

Usage

```
affycomp.compfig2(l, method.names = as.character(1:length(l)),
                  add.legend = TRUE, main = "Figure 2")

affycomp.compfig3(l, method.names = as.character(1:length(l)),
                  main = "Figure 3")

affycomp.compfig4a(l, method.names = as.character(1:length(l)),
                   add.legend = TRUE, main = "Figure 4a")

affycomp.compfig4b(l, method.names = as.character(1:length(l)),
                   add.legend = TRUE, main = "Figure 4b")

affycomp.compfig4c(l, method.names = as.character(1:length(l)),
                   add.legend = TRUE, rotate=TRUE, main = "Figure 4c")

affycomp.compfig5a(l, method.names = as.character(1:length(l)),
                  add.legend = TRUE, main = "Figure 5a", maxfp=100)

affycomp.compfig5b(l, method.names = as.character(1:length(l)),
                  add.legend = TRUE, main = "Figure 5b", maxfp=100)

affycomp.compfig5cde(l, method.names = as.character(1:length(l)),
                  add.legend = TRUE, main = "Figure 5c", maxfp=100,
                  type=c("low","med","high"))

affycomp.compfig5c(l, method.names = as.character(1:length(l)),
                  add.legend = TRUE, main = "Figure 5c", maxfp=100)

affycomp.compfig5d(l, method.names = as.character(1:length(l)),
                  add.legend = TRUE, main = "Figure 5d", maxfp=100)

affycomp.compfig5e(l, method.names = as.character(1:length(l)),
                  add.legend = TRUE, main = "Figure 5e", maxfp=100)
```

Arguments

`l`	a list of lists with the necessary components to create the Figure. See details.
`method.names`	a character vector with the names of the expression measures methodologies being compared.
`add.legend`	logical. If TRUE a legend is added.
`main`	title of the Figure.
`rotate`	in the case of compfig4c one can eiher show the actual local slopes or the bias (local slope minus 1).
`maxfp`	range of the false positives in ROC will be from 0 to `maxfp`
`type`	compfig5cdef is the engine for 5c, 5d, and 5e. `type` tells is which of these 4 to run.

Details

These are similar to the functions defined in `affycomp.figures.auxiliary`. Main difference is that here you send lists with the result of the assessment functions as components.

Value

Figures are produced.

Author(s)

Rafael A. Irizarry

Examples

```
data(rma.assessment)
data(mas5.assessment)
affycomp.compfig2(list(rma.assessment$Dilution,mas5.assessment$Dilution))
affycomp.compfig3(list(rma.assessment$Dilution,mas5.assessment$Dilution))
affycomp.compfig4a(list(rma.assessment$Signal,mas5.assessment$Signal))
affycomp.compfig4b(list(rma.assessment$Dilution,mas5.assessment$Dilution))
affycomp.compfig5a(list(rma.assessment$FC,mas5.assessment$FC))
affycomp.compfig5b(list(rma.assessment$FC2,mas5.assessment$FC2))
```

`affycomp.figures.auxiliary` *Auxiliary functions to create Figures*

Description

These functions are auxiliary function to `affycompPlot`. These Figures are used to assess an expression measure. They take components created by the `assessDilution` and `assessSpikeIn` functions.

Usage

```
affycomp.figure1(l,main="Figure 1",xlim=NULL,ylim=NULL)
affycomp.figure1b(l,main="Figure 1b",xlim=NULL,ylim=NULL,cex=0.85,all=FALSE)
affycomp.figure2(l,main="Figure 2")
affycomp.figure2b(l,main="Figure 2b")
affycomp.figure3(l, main = "Figure 3")
affycomp.figure4a(l, main = "Figure 4a",equal.lims=FALSE)
affycomp.figure4b(l, main = "Figure 4b")
affycomp.figure4c(l, rotate=TRUE, main = "Figure 4c")
affycomp.figure5a(l, main = "Figure 5a",maxfp=100)
affycomp.figure5b(l, main = "Figure 5b",maxfp=100)
affycomp.figure5c(l, main = "Figure 5c",maxfp=100)
affycomp.figure5d(l, main = "Figure 5d",maxfp=100)
affycomp.figure5e(l, main = "Figure 5e",maxfp=100)
affycomp.figure6a(l, main = "Figure 6a",xlim = NULL, ylim = NULL)
affycomp.figure6b(l, main = "Figure 6b",xlim = NULL, ylim = NULL)
```

Arguments

`l`	A list with the necessary components to create the Figure. See details.
`main`	Title for the Figure.
`maxfp`	range of the false positives in ROC will be from 0 to `maxfp`
`xlim`	x-axis limits.
`ylim`	y-axis limits.
`cex`	size of numbers in figure 1b.
`all`	logical. If `TRUE` all spikeins are shown. Otherwise, only those resulting in smaller, realistic, fold changes are shown.
`equal.lims`	logical. If `TRUE` the limits of x-axis and y-axis will have same range.
`rotate`	in the case of compfig4c one can eiher show the actual local slopes or the bias (local slope minus 1).

Details

Read the vignette for more details on what each Figure is. You can read `assessSpikeIn` and `assessDilution` to see which assessments are needed.

Value

Figures are produced.

Author(s)

Rafael A. Irizarry

Examples

```
data(rma.assessment)
affycomp.figure1(rma.assessment$MA)
affycomp.figure2(rma.assessment$Dilution)
affycomp.figure3(rma.assessment$Dilution)
affycomp.figure4a(rma.assessment$Signal)
affycomp.figure4b(rma.assessment$Dilution)
affycomp.figure5a(rma.assessment$FC)
affycomp.figure5b(rma.assessment$FC2)
affycomp.figure6a(rma.assessment$FC)
affycomp.figure6b(rma.assessment$FC)
```

`affycompPlot` *Assessment Plots*

Description

Function that makes assessment plot

Usage

```
affycompPlot(...,assessment.list=NULL,method.names=NULL,
             figure1.xlim=c(-4,15),figure1.ylim=c(-10,12),
             figure1b.xlim=c(-2,14),figure1b.ylim=c(-6,5),
             figure6a.xlim=c(-12,12),figure6a.ylim=c(-12,12),
             figure6b.xlim=c(-3,3),figure6b.ylim=c(-6,6))

affycomp.compfigs(l, method.names = NULL, figure1.xlim = c(-4, 15),
                  figure1.ylim = c(-10, 12), figure1b.xlim = c(-4, 15),
                  figure1b.ylim = c(-4, 4), figure6a.xlim = c(-12, 12),
                  figure6a.ylim = c(-12, 12), figure6b.xlim = c(-3, 3),
                  figure6b.ylim = c(-6, 6))
affycomp.figures(l)
affycomp.figure.calls(what)
affycomp.compfigs.calls(what)
```

Arguments

`...`	lists produced by the assessment functions (one for each method) separated by commas.
`assessment.list`	Alternatively, one can also send a list of lists produced by one of the assessment functions
`method.names`	A character vector with the names of the epxression measure methodology.
`figure1.xlim`	x-axis lim used for the plots in Figure 1.
`figure1.ylim`	y-axis lim used for the plots in Figure 1.
`figure1b.xlim`	x-axis lim used for the plots in Figure 1b.
`figure1b.ylim`	y-axis lim used for the plots in Figure 1b.
`figure6a.xlim`	x-axis lim used for the plots in Figure 6a.
`figure6a.ylim`	y-axis lim used for the plots in Figure 6a.
`figure6b.xlim`	x-axis lim used for the plots in Figure 6b.
`figure6b.ylim`	y-axis lim used for the plots in Figure 6b.
`l`	list with assessment lists as components.
`what`	a dummy variable to know what function call to create.

Details

Read the vignette for more details on what each Figure is. Once an assessment is used this function knows what to do. You can call any of the assessment functions described in `assessSpikeIn`, `assessDilution` and `assessSD`.

`affycomp.figures`, `affycomp.figure.calls`, `affycomp.compfigs.calls` are auxiliary functions.

Value

Figures are produced.

Author(s)

Rafael A. Irizarry

Examples

```
data(rma.assessment)
data(mas5.assessment)
affycompPlot(rma.assessment,mas5.assessment)
affycompPlot(rma.assessment$FC)
affycompPlot(rma.assessment$Dilution,mas5.assessment$Dilution)
affycompPlot(rma.assessment$Dilution,mas5.assessment$Dilution)
affycompPlot(rma.assessment$Signal,mas5.assessment$Signal)
affycompPlot(rma.assessment$Dilution,mas5.assessment$Dilution)
affycompPlot(rma.assessment$FC2,mas5.assessment$FC2)
```

`affycompTable` *Expression Assessment Table*

Description

These functions takes as an argument the output of the asessment functions.

Usage

```
affycompTable(...,Table=NULL,assessment.list=NULL,method.names=NULL)

tableAll(...,assessment.list=NULL,method.names=NULL)

tableDilution(l, method.names=NULL)

tableFC(l, method.names=NULL)

tableFC2(l, method.names=NULL)

tableSignal(l, method.names=NULL)

tableLS(l, method.names=NULL)

tableSpikeInSD(l, method.names=NULL)

tableMA2(l, method.names=NULL)

tableOverallSNR(...,assessment.list=NULL,method.names=NULL,ngenes=12626)

tableRanks(...,assessment.list=NULL,method.names=NULL,ngenes=12626,rank=TRUE)
```

Arguments

`...`	lists produced by the assessment functions
`Table`	If `TableAll` was used one can send it through this argument
`assessment.list`	Alternatively, one can also send a list of lists produced by `tableAll`.
`method.names`	A character vector with the names of the epxression measure methodology.
`l`	list of assessments.
`rank`	if `TRUE` `tableRanks` will present ranks instead of local slopes.
`ngenes`	when computing ranks, out of how many genes should we do it?

Details

Read the vignette for more details on what the entries of the table are. `affycompTable` has a few entries per graph. `tableAll` has more entries. Once an assessment is used this function knows what to do. You can call any of the assessment functions described in `assessSpikeIn`, `assessDilution`, `assessSD`, `assessLS`, `assessMA2`, and `assessSpikeInSD`.

Note `tableRanks` and `tableOverallSNR` work on the results from `assessSpikeIn2`.

Value

A matrix. One column per each method and one row for each comparison. tableOverallSNR is an exception. Where rows represnt methods.

Author(s)

Rafael A. Irizarry

Examples

```
data(rma.assessment) ##this was produced with affycomp.assess
data(mas5.assessment) ##this one too
tmp <- affycompTable(mas5.assessment,rma.assessment)
format(tmp,digit=2)
```

`assessAll` *Assessment functions*

Description

Assessment functions. Takes a couple of `ExpressionSet-class`, one for spike in another for the dilution and returns a list with necessary information to create assessment plots.

Usage

```
assessAll(d,s,method.name=NULL,verbose=TRUE)

affycomp(d,s,method.name=NULL,verbose=TRUE,return.it=TRUE)
```

Arguments

`d`	An `ExpressionSet-class` containing the expression measures for the Gene Logic's dilution experiment.
`s`	An `ExpressionSet-class` containing expression measures for the Affymetrix's spike-in experiment.
`method.name`	Name of expression measure being assessed.
`verbose`	verbosity flag.
`return.it`	if `TRUE` returns assessment lists.

Details

`assessAll` performs assessments for Figures 1-6. It is a wrapper for `assessDilution` and `assessSpikeIn`.

`affycomp` is a wrapper that does it all... including the plotting and creation of table.

Value

Lists with the necessary information to make the Figures.

Author(s)

Rafael A. Irizarry

`assessDilution` *Assessment functions for Dilution Data*

Description

Assessment function. Takes an `ExpressionSet-class` and returns a list with necessary information to create assessment plots.

Usage

```
assessDilution(exprset,method.name=NULL)
```

Arguments

`exprset`	An `ExpressionSet-class` containing expression measures for Affymetrix's spike-in experiment.
`method.name`	Name of expression measure being assessed.

Details

`assessDilution` performs the assessment for the plots related to Dilution (Figures 2, 3, 4b)

Value

Lists with the necessary information to make the Figures.

Author(s)

Rafael A. Irizarry

`assessSD` *SD Assessment functions*

Description

Assessment function for standard deviation estimates. Takes a dilution data `ExpressionSet-class` and returns a list with necessary information to create assessment plot.

Usage

```
assessSD(exprset,method.name=NULL,logx=FALSE)
```

Arguments

`exprset`	An `ExpressionSet-class` containing expression measures for Affymetrix's spike-in experiment.
`method.name`	Name of expression measure being assessed.
`logx`	Logical. If `TRUE` expression is logged for plot. See details.

Details

`assessSD` does the assessment for Figure 7. This requires the `ExpressionSet` to have standard error estimates for the expression measure. Some expression (such as dChip) will have SEs for the original scale. Others, like RMA will have them for the log scale. For original scales, making `logx=TRUE` is recommended.

Value

Lists with the necessary information to make the Figures.

Author(s)

Rafael A. Irizarry

`assessSpikeIn2` *New Assessment functions for Spike In Data*

Description

These functions are assessment functions. Each takes an `ExpressionSet-class` and returns a list with necessary information to create assessment plots.

Usage

```
assessSpikeIn2(s,method.name=NULL,verbose=TRUE)

assessSpikeInSD(exprset,method.name=NULL,span=1/3)
assessLS(exprset,method.name=NULL)
assessMA2(exprset,method.name=NULL)
```

Arguments

`s`	An `ExpressionSet-class` containing expression measures for Affymetrix's spike-in experiment.
`exprset`	An `ExpressionSet-class` containing expression measures for Affymetrix's spike-in experiment.
`method.name`	Name of expression measure being assessed.
`verbose`	logical. If `TRUE` show messages.
`span`	span used in call to `loess`.

Details

`assessMA2` performs the assessment for the second MA-plot (Figure 1b), `assessLS` performs the assessment for signal detection plot (Figure 4c), `assessMA2` also performs assessments used by fold-change related plots (Figures 5c-f). `assessSpikeInSD` is for the standard deviation assessment of Figure 2b. `assessSpikeIn2` is a wrapper for all these and returns a list of lists.

Value

Lists with the necessary information to make the Figures.

Author(s)

Rafael A. Irizarry

`assessSpikeIn` *Assessment functions for Spike In Data*

Description

These functions are assessment functions. Each takes an `ExpressionSet-class` and returns a list with necessary information to create assessment plots.

Usage

```
assessSpikeIn(s,method.name=NULL,verbose=TRUE)

assessMA(exprset,method.name=NULL)
assessSignal(exprset,method.name=NULL)
assessFC(exprset,method.name=NULL)
assessFC2(exprset,method.name=NULL)
```

Arguments

`s`	An `ExpressionSet-class` containing expression measures for Affymetrix's spike-in experiment.
`exprset`	An `ExpressionSet-class` containing expression measures for Affymetrix's spike-in experiment.
`method.name`	Name of expression measure being assessed.
`verbose`	logical. If `TRUE` show messages.

Details

`assessMA` performs the assessment for the MA-plot (Figure 1), `assessSignal` performs the assessment for signal detection plot (Figure 4a), `assessFC` performs assessments used by fold-change related plots (Figures 5a, 6a, 6b). `assessFC2` is for the ROC for genes with nominal fold changes of 2 (Figure 5b). `assessSpikeIn` is a wrapper for all these and returns a list of lists.

Value

Lists with the necessary information to make the Figures.

Author(s)

Rafael A. Irizarry

`dilution.phenodata` *Phenotypic Information for Dilution Study*

Description

This objact is of class `phenoData` with necessary information for the assessemnts.

Usage

```
data(dilution.phenodata)
```

Format

An object of class `phenoData`

Source

Two sources of cRNA A (human liver tissue) and B (Central Nervous System cell line) have been hybridized to human array (HGU95Av2) in a range of proportions and dilutions. This object described these.

For more information see Irizarry, R.A., et al. (2001) `http://www.biostat.jhsph.edu/~ririzarr/papers/index.html`

`exprset.log` *Take log base 2 of Expression*

Description

Take log base 2 of the expression matrix in an `ExpressionSet`

Usage

```
exprset.log(exprset)
```

Arguments

`exprset`	`ExpressionSet`

Details

This functions takes log base 2 of the expression matrix in an `ExpressionSet`. Negatives are converted to the smallest non-negative entry.

Value

An `ExpressionSet`

Author(s)

Rafael A. Irizarry

`hgu133a.spikein.phenodata` *phenotypic information for HGU133A spike in study*

Description

This object is of class `phenoData` with necessary information for the assessemnts.

Usage

```
data(hgu133a.spikein.phenodata)
```

Format

An object of class `phenoData`

Source

This comes from an experiments where 16 different cRNA fragments have been added to the hybridization mixture of the GeneChip arrays at different pM concentrations. For more information see Irizarry, R.A., et al. (2001) `http://www.biostat.jhsph.edu/~ririzarr/papers/index.html`

`lw.sd.assessment` *An example of the result of an SD assessment*

Description

The Dilution files were processed with the dChip package (using PM-only) and then the function `assessSD` was applied.

Usage

```
data(lw.sd.assessment)
```

Format

A list.

`mas5.assessment` *An example of the result of the assessments*

Description

The Dilution and both (HGU95 and HGU133) Spike-in cel files were processed with MAS 5.0 software and then the functions `assessAll` and `assessSpikeIn2` were applied.

Usage

```
data(rma.assessment)
data(rma.assessment.133)
data(rma.assessment2)
data(rma.assessment2.133)
```

Format

A list of list.

`readin` *Read Expression Date Sets*

Description

Reads a comma-delimited file containing the expression values of the dilution and spike-in data sets and creates a `ExpressionSet`

Usage

```
read.dilution(filename)
read.spikein(filename,cdfName=c("hgu95a","hgu133a"),remove.xhyb=TRUE)
read.newspikein(filename)
```

Arguments

`filename`	character containing the filename to be read.
`cdfName`	are we reading data from the hgu95a or hgu133a spike-in experiment?
`remove.xhyb`	logical. If `TRUE` possible cross hybridizers are removed from the HGU133A spikein. See `remove.hgu133a.xhyb`.

Details

The file to be read must be comma-delimited with the first row containing the cel filenames (case sensitive). The first column must be the Affymetrix gene identifiers. `read.dilution` will put things in the right place.

`read.newspikein` is a wrapper to read results from the hgu133a spikein experiment.

Value

An `ExpressionSet`.

Author(s)

Rafael A. Irizarry

`remove.hgu133a.xhyb` *Remove crosshybridizers*

Description

This functions removes possible cross hybridizers from Affymetrix HGU133A spike-in experiment

Usage

```
remove.hgu133a.xhyb(s, bp = c("200", "150", "100"))
```

Arguments

`s`	an `ExpressionSet` containing the HGU133A spike-in
`bp`	number of base pair matches needed to define a possible cross hybridizer

Details

Some details are contained in the help file for `hgu133a.spikein.xhyb`

Value

An `ExpressionSet` with probeset removed

See Also

`hgu133a.spikein.xhyb`

`rma.assessment` *An example of the result of an assessment*

Description

The Dilution and Spike-in cel files were processed with the `affy` version 1.0 package rma add-on function and then the functions `assessAll` and `assessSpikeIn2` were applied.

Usage

```
data(rma.assessment)
data(rma.assessment.133)
data(rma.assessment2)
data(rma.assessment2.133)
```

Format

A list of list.

`rma.sd.assessment` *An example of the result of an SD assessment*

Description

The Dilution files were processed with the `affy` version 1.0 package rma add-on function and then the function `assessSD` was applied.

Usage

```
data(rma.sd.assessment)
```

Format

A list.

`SD` *SD Assessment Functions*

Description

These functions create assessments, figures, and tables for expression standard errors

Usage

```
affycomp.figure7(l,main="Figure 7")
affycomp.compfig7(l,method.names=as.character(1:length(l)),
                  main="Figure 7")
tableSD(l,method.names=NULL)
```

Arguments

`l`	a list of lists with the necessary components to create the Figure. See details.
`method.names`	a character vector with the names of the expression measures methodologies being compared.
`main`	title of the Figure.

Details

This uses the dilution data. The exprsets need to have standard error estimates in the `assayDataElement(exprset,`
Read the vignette for more details. The functions work similarly to those assessing expression measures.

All these files need the result of `assessSD`

Value

Depends on the call.

Author(s)

Rafael A. Irizarry

Examples

```
data(rma.sd.assessment) ##this was produced with affycomp.assess
data(lw.sd.assessment) ##this one too
affycomp.compfig7(list(rma.sd.assessment,lw.sd.assessment))
affycomp.figure7(rma.sd.assessment)
```

`spikein.phenodata` *phenotypic information for spike in study*

Description

This objact is of class `phenoData` with necessary information for the assessemnts.

Usage

```
data(spikein.phenodata)
```

Format

An object of class `phenoData`

Source

This comes from an experiments where 16 different cRNA fragments have been added to the hybridization mixture of the GeneChip arrays at different pM concentrations. For more information see Irizarry, R.A., et al. (2001) `http://www.biostat.jhsph.edu/~ririzarr/papers/index.html` `hgu133a.spikein.xhyb` *Cross hybridizers*

Description

Probe Sets likely to crosshybridize to spiked-in probesets in the Affymetrix HGU133A spike in.

This objact is list. Each component of the list contains probeset names of possible crosshybridizers. The sequences of each spiked-in clone were collected and blasted against all HG-U133A target sequences. Target sequences are the ~600bp regions from which probes were selected. Thresholds of 100, 150 and 200bp were used and define the three components of the list.

Usage

```
data(hgu133a.spikein.xhyb)
```

Format

A list

Source

Simon Cawley <simon_cawley@affymetrix.com>

Index