Package ‘codelink’

March 28, 2017

Version  1.42.0
Date     2016-03-15
Title    Manipulation of Codelink microarray data
Author   Diego Diez
Maintainer Diego Diez <diego10ruiz@gmail.com>
Depends  R (>= 2.10), BiocGenerics (>= 0.3.2), methods, Biobase (>= 2.17.8), limma
Imports  annotate
Suggests genefilter, parallel, knitr
LazyLoad yes
Description This package facilitates reading, preprocessing and manipulating
             Codelink microarray data. The raw data must be exported as text file using the
             Codelink software.
License GPL-2
        plot.R  CodelinkSet-methods.R  CodelinkSet-tools.R
        CodelinkSetUnique-class.R  CodelinkSetUnique-methods.R  filter.R
        cluster.R
biocViews Microarray, OneChannel, DataImport, Preprocessing
ByteCompile yes
VignetteBuilder knitr
URL      https://github.com/ddiez/codelink
BugReports https://github.com/ddiez/codelink/issues
NeedsCompilation no

R topics documented:

  arrayNew                                  2
  arraySize                                 3
  as.matrix.Codelink                        3
  averageProbes                             4
  bkgdCorrect                              5
  Codelink-class                            6
arrayNew

Create a new x11 device

Description

Create a new x11 device with dimensions suited to be used with imageCodelink().

Usage

arrayNew(f=2, chip="rwgcod")

Arguments

chip character; Codelink chip to be used in imageCodelink.
f numerical; scaling factor.

Author(s)

Diego Diez
arraySize

Examples

```r
## Not run:
data(codelink.example)
arrayNew()
imageCodelink(codelink.example)

## End(Not run)
```

arraySize  

*Determine the size of the array*

Description

When loading Codelink arrays in text format (as exported from the Codelink software) this function retrieves the correct size of the array. This is useful because those files contain an indetermined number of empty lines at the end. Thus, reading the entire data matrix doesn’t work.

Note

Not meant to be used directly.

Author(s)

Diego Diez

as.matrix.Codelink  

*Return a matrix of intensity values*

Description

Takes a Codelink object and returns a matrix with the intensity values available.

Usage

```r
## S3 method for class 'Codelink'
as.matrix(x, ...)
```

Arguments

- `x`: an object of class "Codelink".
- `...`: additional arguments added to generic as.matrix since R-2.5.x

Value

A matrix with the intensity values.

Author(s)

Diego Diez
averageProbes

See Also

as.matrix

Examples

## Not run:
data(codelink.example)
mat <- as.matrix(codelink.example)
is(mat)
## End(Not run)

averageProbes

Description

Takes a CodelinkSet object and computes the average (mean) and sd of duplicated probes.

Usage

averageProbes(object, parallel = FALSE)

Arguments

object
  an object of class "CodelinkSet".

parallel
  whether to use a parallel version (requires package multicore).

Details

This function will compute the mean() and sd() on each duplicated probe (i.e. identical probe id as for probeNames()). CodelinkSet objects use the per-array index to enable unique identifiers needed for ExpressionSet derived objects. Although the method probeNames() provides access to Codelink probe ids, this is inconvenient when dealing with other packages that make use of featureNames() to obtain probe ids and feed them to the corresponding annotation package.

In such cases CodelinkSet objects are not compatible with methods working on ExpressionSet objects. To avoid this limitation it is possible to construct a CodelinkSet object containing unique identifiers (CodelinkSetUnique class) by averaging the intensities of all replicated probes. This is done by computing the mean(). The sd() is also computed and stored in the slot sd.

The current implementation takes a lot of time so a parallelized version of lapply() may be used through the package ‘multicore’. This is controlled by the argument ‘parallel’ which is FALSE by default.

According to the authors it is not actually possible to use ‘multicore’ in a GUI environment like the R.app Cocoa application in R, because this interferes with the events loop. Therefore it is advised to use option ‘parallel=TRUE’ in an R session running in a shell.

Author(s)

Diego Diez
bkgdCorrect

Examples

## Not run:
    data(codelink.example)
    foo <- averageProbes(codelink.example)

## End(Not run)

bkgdCorrect  

**Background correction of intensity values.**

Description

Takes a CodelinkSet or Codelink object with Spot mean and Bkgd median values and performs background correction using one of the methods available.

Usage

```r
bkgdCorrect(object, method = "half", preserve = FALSE, verbose = FALSE, offset = 0)
```

## S4 method for signature 'CodelinkSet'

codCorrect(object, method = "half", normexp.method="saddle", offset=0)

Arguments

- `object`: an object of class CodelinkSet or Codelink.
- `method`: the correction method to use, one of "none", "subtract", "half" and "normexp".
- `preserve`: logical; if Smean and Bmedian slots should be preserved.
- `verbose`: logical; if TRUE print some information with method normexp.
- `normexp.method`: method used with normexp correction.
- `offset`: numeric; value to add to intensities.

Details

Available methods are: . none: left intensities untouched. . subtract: simple subtraction of Bkgd median from Spot mean. . half: the same as above but aboid negative values setting all intensity values below zero to 0.5. . normexp: apply normexp background adjustment from package limma.

Value

An object of class Codelink with corrected intensity values, that is Ri slot.

Author(s)

Diego Diez

Examples

```r
data(codset)
codset <- codCorrect(codset, method = "half")
```
**Description**

This is the storage class for Codelink data

**Objects from the Class**

Object are created after reading text codelink files with readCodelink()

**Description**

Codelink objects contain a single "list", which contains the following elements:

- **sample** Object of class "character" containing the sample names
- **file** Object of class "character" containing the file names
- **name** Object of class "character" containing the probe ID
- **method** Object of class "list" containing log information
- **Smean** Object of class "matrix" containing spot mean intensities
- **Bmedian** Object of class "matrix" containing background median intensities
- **Ri** Object of class "matrix" containing raw intensities
- **Ni** Object of class "matrix" containing normalized intensities
- **snr** Object of class "matrix" containing signal to noise ratio values
- **flag** Object of class "character" containing assigned flags

**Note**

More details are in the package vignette

**Author(s)**

Diego Diez

**Examples**

```r
## Not run:
  data(codelink.example)

## End(Not run)
```
**codelink.example**

---

**codelink.example**  
*Dataset of class 'Codelink'*

---

**Description**

Dataset from a h20kcod (Codelink Human UniSet I 20k) array containing 2 samples and ~20000 probes.

**Usage**

```r
data(codelink.example)
```

**Format**

A Codelink object containing 20469 probes and 2 samples.

**Author(s)**

Diego Diez

---

**CodelinkSet-class**  
*Class CodelinkSet*

---

**Description**

This is the storage class for Codelink data

**Objects from the Class**

Object are created after reading text codelink files with `readCodelinkSet()`

**Description**

CodelinkSet objects are derived from ExpressionSet and therefore inherits all their methods. Additional methods are defined to extract information:

- Old Codelink-class objects can be converted into CodelinkSet instances using the function `Codelink2CodelinkSet`.

**Extends**

Directly extends class `ExpressionSet`.

**Methods**

Class-specific methods:

- `getWeight`  
  Get the matrix of weights.

**Author(s)**

Diego Diez
Examples

## Not run:
```
data(codset)
```
## End(Not run)

CodelinkSetUnique-class

Class CodelinkSetUnique

Description

This is the storage class for Codelink data– with unique probes.

Objects from the Class

Object are created after applying averageProbes() on an object of the class CodelinkSet.

Description

CodelinkSetUnique objects are derived from ExpressionSet and therefore inherits all the methods.

Note

More details are in the package vignette.

Author(s)

Diego Diez

Examples

## Not run:
```
data(codelink.exprs)
foo <- averageProbes(codelink.exprs)
```
## End(Not run)

codPlot

Diagnostic plots for CodelinkSet object.

Description

This function performs several types of diagnostic plots using information from CodelinkSet objects.

Usage

```
codPlot(object, array, what = "ma", ...)
```
Arguments

- **object**: a CodelinkSet object.
- **array**: The array to use for plotting (by default array=1).
- **what**: the type of plot (by default ma).
- **...**: additional arguments passed to the appropriate functions.

Details

Plot types:

- **scatter**: Scatter plot of the selected array intensities vs. the median array, or a second array if specified.
- **ma**: MA plot of the selected array intensities vs. the median array, or a second array if specified. In the x-axis the value of A (mean abundance; A = Array1 + Array2) is plotted. In the y-axis the value of M (difference; M = Array1 - Array2) is plotted.
- **density**: The distribution of intensities for each array in the dataset is plotted as a kernel density.
- **image**: If the logical position of the probes in the array are available, this plot produces a pseudoimage of the array. The values of the "intensity", "bg" or "snr" can be selected with the 'signal' parameter (default signal="bg"). Color range can be controled with 'high' and 'low' parameters.

Author(s)

Diego Diez

---

codset

**Dataset of class 'CodelinkSet'**

Description

Dataset from a h20kcod (Codelink Human UniSet I 20k) array containing 4 samples and ~20000 probes.

Usage

```r
data(codelink.example)
```

Format

A **Codelink** object containing 20469 probes and 4 samples.

Author(s)

Diego Diez
createWeights

Create weight for Codelink chips

Description
Create a weight matrix based on probe type and flags.

Usage
createWeights(object, type.weights = NULL, flag.weights = NULL)

Arguments
object an object of class "Codelink" or "CodelinkSet".
type.weights named character vector of type weights used.
flag.weights named character vector of flag weights used.

Author(s)
Diego Diez

Examples
data(codset)
w <- createWeights(codset)

cutCV

Calculate cutoff based in CV.

Description
Takes a Codelink object and calculate cutoff based in CV.

Usage
cutCV(object, subset=c(1:dim(object)[2]))

Arguments
object an object of class "Codelink".
subset subset of arrays to calculate cutoff with

Details
First it computes the median of CV for each gene over all arrays. Then it computes the mean and sd of all medians. Finally:
cutoff = mean + 3 * sd
Author(s)

Diego Diez.

Examples

```r
## Not run:
# data: Normalized Codelink object merged.
cutoff <- cutCV(data)
## End(Not run)
```

decDetect

Determine decimal type of Codelink files

Description

Determine decimal type of Codelink files.

Usage

```r
decDetect(file, nlines)
```

Arguments

- `file`: the file to be read.
- `nlines`: number of lines to skip.

Value

Decimal type.

Author(s)

Diego Diez

dim.Codelink

Return the dimension of a Codelink object.

Description

Takes a Codelink object and returns the dimension (genes x samples).

Usage

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

Arguments

- `x`: an object of class "Codelink".
Value
A numeric vector with the dimensions.

Author(s)
Diego Diez

Examples
```r
## Not run:
data(codelink.example)
dim(codelink.example)

## End(Not run)
```

---

**fc2Cond**

Select probes based on fold change calculation

Description
Takes a Codelink object and calculate fold changes (M) between two conditions (samples). Then select genes based on those who pass the passed cutoff.

Usage

```r
fc2Cond(object, cond1=NULL, cond2=NULL, fc=1.0, verbose=FALSE)
```

Arguments

- **object**: an object of class "Codelink".
- **cond1**: numeric or character; First condition to compute M.
- **cond2**: numeric or character; Second condition to compute M.
- **fc**: value of the fold change cutoff
- **verbose**: logical; if some information is printed on the console.

Details
Conditions can be passed as characters or as numeric index from the sample slot. The intensities are internally transformed to log2 if needed. The M value is computed as:

\[ M = \text{cond1} - \text{cond2} \]

Value
A logical vector indicating which genes pass the cutoff

Author(s)
Diego Diez
imageCodelink

**Image plot of Codelink arrays**

**Description**

Plot and image of a Codelink array if the layout information is found.

**Usage**

```r
imageCodelink(object, array = 1, what = "bg",
               low="black", high="white", mar=c(1,1,1,1),
               gr=1, gc=1, log.it=FALSE, ...)```

**Arguments**

- **object**: an object of class "Codelink".
- **array**: array to be used.
- **what**: with data plot: bg, smean, ri, ni.
- **low**: color used for low intensities.
- **high**: color used for high intensities.
- **mar**: character vector specifying margings.
- **gc**: numerical; number of grid columns.
- **gr**: numerical; number of grid rows.
- **log.it**: logical; if TRUE data is log2 transformed (if not yet).
- **...**: additional arguments passed to `image`.

**Author(s)**

Diego Diez

**Examples**

```r
## Not run:
data(codelink.example)
imageCodelink(codelink.example)

## End(Not run)```
logCodelink

Description
Takes a Codelink object and apply logCodelink to intensity values.

Usage
logCodelink(object)

Arguments
object an object of class "Codelink" or a list of genes.

Value
A Codelink object with logCodelink intensities.

Author(s)
Diego Diez

See Also
log2

Examples
## Not run:
data(codelink.example)
codelink.example <- logCodelink(codelink.example)
## End(Not run)

mergeArray

Description
Merge data in a Codelink Object corresponding to same samples. Need a vector indicating the classes and an optional vector indicating the labels of the merged samples.

Usage
mergeArray(object, class, names=NULL, method="mean", log.it=FALSE, merge.snr=TRUE)
na2false

Arguments

object an object of class "Codelink".
class a numeric vector indicating the classes.
names an optional character vector indicating labels for each class.
method the method used to summarize. Currently only "mean" supported.
log.it logical; a logical indicating if log2 values should be returned.
merge.snr logical; a logical indicating if SNR values should be merged.

Value

An object of class "Codelink".

Author(s)

Diego Diez

Examples

## Not run:
data(codelink.example)
codelink.example <- bkgdCorrect(codelink.example)
codelink.example <- normalize(codelink.example, log.it = FALSE)
codelink.example <- mergeArray(codelink.example, class = c(1,1), names = "SAMPLE", log.it = TRUE)
## End(Not run)

na2false Set NAs to FALSE

Description

Takes a logical vector as input and set all NAs to FALSE. This may happens when comparison is done on NA values.

Usage

na2false(x)

Arguments

x a logical vector.

Value

A logical vector without NAs.

Author(s)

Diego Diez
### Examples

```r
## Not run:
a <- c(1, 2, 3, NA, 5)
b <- c(5, 4, NA, 2, 1)
sel <- a > b
sel <- na2false(sel)

## End(Not run)
```

---

#### normalize

**Normalization wrapper for Codelink/CodelinkSet objects.**

**Description**

Takes a CodelinkSet or Codelink object and applies normalization to intensity values.

**Usage**

```r
## S4 method for signature 'Codelink'
normalize(object, method="quantiles", log.it=TRUE, preserve=FALSE, weights=NULL, loess.method="fast")

## S4 method for signature 'CodelinkSet'
normalize(object, method="quantile", log.it=TRUE, weights=NULL, loess.method="fast")

## S4 method for signature 'CodelinkSet'
codNormalize(object, method="quantile", log.it=TRUE, weights=NULL, loess.method="fast")
```

**Arguments**

- `object`: an object of class CodelinkSet or Codelink.
- `...`: further arguments passed to the normalization functions.
- `method`: method to use in normalization.
- `log.it`: logical; if data should be log2.
- `preserve`: logical; if Ri slot should be preserved.
- `weights`: weights for method CyclicLoess.
- `loess.method`: loess method to be used, default to 'fast'.

**Details**

Currently supported methods include "median", "quantile" and "loess". Normalization functions are borrowed from the limma package.

If a matrix of weights is passed for CyclicLoess normalization, a vector is constructed taking the lowest weight for each probe.

**Value**

A CodelinkSet or Codelink object with normalized intensity values.
Author(s)

Diego Diez.

Examples

data(codset)
# Background correction.
codset <- codCorrect(codset, method = "half")
# Normalization.
codset <- normalize(codset, method = "quantile")

plotCorrelation

Plot correlation scatterplot between two arrays

Description

Takes a Codelink object as argument and plot Correlation scatterplot of two arrays.

Usage

plotCorrelation(object, x=1, y=2, cutoff=FALSE, label="type", title=NULL, xlim=NULL, ylim=NULL)

Arguments

object an object of class "Codelink".
x array to be used in x axis.
y array to be used in y axis.
cutoff cutoff used to show fold change.
label labels to shown.
title The title of the plot.
xlim range for the X axis.
ylim range for the Y axis.

Author(s)

Diego Diez

See Also

plot

Examples

## Not run:
data(codelink.example)
plotCorrelation(codelink.example)

## End(Not run)
plotCV

Plot of CV

Description
Takes a Codelink object and plot the distribution of CV after applying mergeCodelink.

Usage
plotCV(object, subset=c(1:dim(object)[2]), cutoff=NULL, title=NULL, legend.cex=1)

Arguments
- object: an object of class "Codelink".
- subset: subset of arrays to plot
- cutoff: cutoff of CV to be shown.
- title: title of the plot.
- legend.cex: factor to apply to the fonts in the legend to fit.

Author(s)
Diego Diez

plotDensities

Plot Densities

Description
Takes a Codelink object and plot the distributions of intensities.

Usage
plotDensities(object, what = NULL, title = NULL, col = NULL, legend.title = NULL, legend.cex=1, ...)

Arguments
- object: an object of class "Codelink"
- what: what data to plot, one of "bg", "smean", "snr", "ri" or "ni"
- title: title of the plot
- col: vector of colors
- legend.title: if provided, a title for the legend
- legend.cex: font factor use in legend
- ...: further arguments to be passed to plot() and line() functions (e.g. lwd, etc.)

Author(s)
Diego Diez
Examples

```r
## Not run:
data(codelink.example)
plotDensities(codelink.example)

## End(Not run)
```

plotMA  

*MA plot*

Description

Takes a Codelink object and plot M vs A.

Usage

```r
plotMA(object, array1 = 1, array2 = NULL, cutoff = c(-1, 1), label = NULL,
type = NULL, high.list = NULL, high.col = "blue", high.pch = 21,
high.bg = "cyan", snr = NULL, snr.cutoff = 1, legend.x = NULL, pch = ".",
subset = NULL, title = NULL, xlim = NULL, ylim = NULL)
```

Arguments

- **object**: an object of class "Codelink" or "MAarrayLM".
- **array1**: first array to be used.
- **array2**: second array to be used.
- **cutoff**: cutoff to be used as fold change marker.
- **label**: type of labeling used in legend.
- **type**: spot type information.
- **high.list**: list of genes highlighted.
- **high.col**: color used for high genes.
- **high.pch**: pch used for high genes.
- **high.bg**: background color used for high genes.
- **snr**: vector with SNR values, usually, taking rowMeans() from a SNR matrix.
- **snr.cutoff**: SNR cutoff used for label spots.
- **legend.x**: relative position of the legend.
- **pch**: pch style used to main spots.
- **subset**: subset of spots used to plot based on 'type' slot.
- **title**: title of the plot.
- **xlim**: range for the X axis.
- **ylim**: range for the Y axis.
**Details**

This function has suffered recent re-working, to increase the usability and to clean a little bit the code.

If array2 is NULL a median array is computed using all available arrays. Then the values of M and A are computed using the following formula:

\[ M = \text{array2} - \text{array1} \]
\[ A = (\text{array2} + \text{array1}) / 2 \]

If type information is available in the Codelink object, or provided through the `type` argument, spots are colored based on that. DISCOVERY spots are plotted black with `pch = .` whereas the other classes are plotted with different background colors, using gray as border to increase contrasts. For that `pch = 21` is used. If `snr` is specified as `label` option, the SNR is used to label spots, if available in the Codelink object. In this case, the mean SNR across all arrays is used when `array2 = NULL`.

Some parameters may not be working right now, as the new function is using a different method to labels spots.

The legend is 'automagically' located, but this can be overridden with the `legend.x` argument.

In addition, a subset of the spots can be plotted based on type information when available. This allows, for example, to plot only DISCOVERY spots.

**Author(s)**

Diego Diez

**Examples**

```r
## Not run:
data(codelink.example)
plotMA(codelink.example)
## End(Not run)
```

---

### printHead

**Print briefly a Codelink object**

**Description**

Takes a Codelink object and print a summary information of the data stored. It is based on ```printHead()``` from package ```limma```.

**Usage**

`printHead(x)`

**Arguments**

- `x`: an object of class "Codelink".

**Author(s)**

Diego Diez
**Description**

Read data exported as text by Codelink Software. It reads values (normalized by Codelink Software or not) flags and information about probes.

**Usage**

```r
readCodelink(files = list.files(pattern = "TXT"), sample.name = NULL, flag, flag.weights, type.weights, dec = NULL, type = "Spot", preserve = FALSE, verbose = 2, file.type = "Codelink", check = TRUE, fix = FALSE, old = FALSE)
```

**Arguments**

- `files`: list of files to read.
- `sample.name`: vector of same length as files with sample names.
- `flag`: list with values to assign based on Flag quality values.
- `flag.weights`: weights assigned to each probe flag.
- `type.weights`: weights assigned to each probe type.
- `dec`: character indicating the decimal character used in the files.
- `type`: character indicating which base value to read from files.
- `preserve`: logical, if TRUE Bkgd_stdev slot is not removed (if present).
- `verbose`: numerical, set the level of information. Level 2 set as old behaviour. Level > 2 output some debug info.
- `file.type`: exported file type, currently Codelink or XLS file formats supported.
- `check`: logical, check for probe order consistency.
- `fix`: logical, try to fix probe order consistency.
- `old`: logical, whether NA is assigned to intensities based on flags.
- `filename`: list of files.
- `path`: path to the files.
- `phenoData`: phenoData object with sample information.
- `...`: further arguments to be passed.

**Value**

readCodelink returns an object of class "Codelink", whereas readCodelinkSet() returns a CodelinkSet object.

**Author(s)**

Diego Diez
Examples

```r
## Not run:
# specify a different one.
  f <- list.files(pattern = "TXT")
  codset <- readCodelinkSet(filename = f)

# read a targets file with phenotypic information
# including sample names, file names and sample groups.
  pdata = read.AnnotationDataFrame("targets.txt")
  codset = readCodelinkSet(pdata$FileNames, phenoData=pdata)

data(codset)
```

## End(Not run)

---

### readHeader

**Read Header from Codelink Bioarrays Files**

**Description**

Read the header of Codelink files and obtain useful information.

**Usage**

```r
readHeader(file, dec=FALSE)
```

**Arguments**

- `file` File to read.
- `dec` logical; If TRUE determine decimal point.

**Value**

A list with header and other useful information.

**Author(s)**

Diego Diez

**Examples**

```r
## Not run:
  files <- list.files(pattern = "TXT")
  head <- readHeader(files[[1]])

## End(Not run)
```
**readHeaderXLS**

*Read Header from XLS exported Codelink Bioarrays Files*

**Description**
Read the header of Codelink files and obtain useful information.

**Usage**
```r
readHeaderXLS(file, dec=FALSE)
```

**Arguments**
- `file`: File to read.
- `dec`: logical; If TRUE determine decimal point.

**Details**
This function is not meant to be used by normal users.

**Value**
A list with header and other useful information.

**Author(s)**
Diego Diez

**Examples**
```r
## Not run:
files <- list.files(pattern = "TXT")
head <- readHeaderXLS(files[1])
## End(Not run)
```

---

**reportCodelink**

*Write a report of genes selected in HTML*

**Description**
Takes a list of genes as argument and writes an HTML page containing information about these genes: Unigene, Genbank, Entrez Gene, etc.

**Usage**
```r
reportCodelink(object, chip, filename = NULL, title = "Main title",
               probe.type = FALSE, other = NULL, other.ord = NULL)
```
Arguments

object an object of class "Codelink" or a list of genes.
chip the chip description package.
filename file name used in the report.
title title used in the report.
probe.type logical; if TRUE Probe type information is written.
other list of vectors containing additional values to add to the report.
other.ord slot name in other to order genes by.

Value

Nothing, only the HTML file generated.

Author(s)

Diego Diez

See Also

htmlpage

Description

Takes a Codelink object and select genes based on CV cutoff.

Usage

selCV(object, cutoff)

Arguments

object an object of class "Codelink".
cutoff cutoff normaly calculated with cutCV()

Value

A logical vector.

Author(s)

Diego Diez
**SNR**

**Calculate SNR**

**Description**

Compute SNR inside read.Codelink.

**Usage**

```r
SNR(Smean, Bmedian, Bstdev)
```

**Arguments**

- `Smean` matrix of Smean intensities.
- `Bmedian` matrix of background median intensities.
- `Bstdev` matrix of background standard deviation.

**Author(s)**

Diego Diez

---

**writeCodelink**

**Write a Codelink object to file.**

**Description**

Export of the data from a codelink object to a text file.

**Usage**

```r
writeCodelink(object, file, dec = ".", sep = "\t", flag = FALSE, chip)
```

**Arguments**

- `object` an object of class "Codelink".
- `file` filename to write object to.
- `dec` decimal character to use.
- `sep` delimiter character to use.
- `flag` should the Codelink flags be written.
- `chip` chip package to use, normally guessed.

**Details**

By default, intensities and SNR are wrote to the file. If set, the flag are also output. The header have "INTENSITY\_", "SNR\_" and "FLAG\_" respectively appendend to the sample name for those values. The default delimiter is the tab character, but that can be set with the sep argument. The default decimal character is the point.
Author(s)

Diego Diez.

Examples

```r
## Not run:
data(codelink.example)
writeCodelink(codelink.example, file = "foo.txt")

## End(Not run)
```
Index

*Topic **classes**  
  Codelink-class, 6  
  CodelinkSet-class, 7  
  CodelinkSetUnique-class, 8

*Topic **datasets**  
  codelink.example, 7  
  codset, 9

*Topic **documentation**  
  arrayNew, 2  
  arraySize, 3  
  as.matrix.Codelink, 3  
  averageProbes, 4  
  bkgdCorrect, 5  
  createWeights, 10  
  cutCV, 10  
  decDetect, 11  
  dim.Codelink, 11  
  fc2Cond, 12  
  imageCodelink, 13  
  logCodelink, 14  
  mergeArray, 14  
  na2false, 15  
  normalize, 16  
  plotCorrelation, 17  
  plotCV, 18  
  plotDensities, 18  
  plotMA, 19  
  printHead, 20  
  readCodelink, 21  
  readHeader, 22  
  readHeaderXLS, 23  
  reportCodelink, 23  
  selCV, 24  
  SNR, 25  
  writeCodelink, 25

[,Codelink,ANY,ANY-method  
  (Codelink-class), 6

arrayNew, 2  
arraySize, 3  
as.matrix, 4  
as.matrix.Codelink, 3  
averageProbes, 4  
averageProbes,CodelinkSet-method  
  (averageProbes), 4

bkgdCorrect, 5

class::CodelinkSet (CodelinkSet-class), 7

class::CodelinkSetUnique  
  (CodelinkSetUnique-class), 8

class::Codelink (Codelink-class), 6  
  codCorrect (bkgdCorrect), 5  
  codCorrect,CodelinkSet-method  
  (bkgdCorrect), 5  
  Codelink, 7, 9  
  Codelink (Codelink-class), 6
INDEX

Codelink-class, 6
codelink.example, 7
Codelink2CodelinkSet
 (CodelinkSet-class), 7
CodelinkSet (CodelinkSet-class), 7
CodelinkSet-class, 7
CodelinkSetUnique
 (CodelinkSetUnique-class), 8
CodelinkSetUnique-class, 8
codNormalize (normalize), 16
codNormalize, CodelinkSet-method
 (normalize), 16
codPlot, 8
codPlot, CodelinkSet-method (codPlot), 8
codPlot, MArrayLM-method (codPlot), 8
codPreprocess (CodelinkSet-class), 7
codPreprocess, CodelinkSet-method
 (CodelinkSet-class), 7
codset, 9
createWeights, 10
createWeights, Codelink-method
 (createWeights), 10
createWeights, CodelinkSet-method
 (createWeights), 10
cutCV, 10
decDetect, 11
dim.Codelink, 11
ExpressionSet, 7
exprs, CodelinkSet-method
 (CodelinkSet-class), 7
fc2Cond, 12
getBkg (CodelinkSet-class), 7
getBkg, CodelinkSet-method
 (CodelinkSet-class), 7
getFlag (CodelinkSet-class), 7
getFlag, CodelinkSet-method
 (CodelinkSet-class), 7
getInfo (CodelinkSet-class), 7
getInfo, CodelinkSet-method
 (CodelinkSet-class), 7
getInt (CodelinkSet-class), 7
getInt, CodelinkSet-method
 (CodelinkSet-class), 7
getSNR (CodelinkSet-class), 7
getSNR, CodelinkSet-method
 (CodelinkSet-class), 7
getWeight (CodelinkSet-class), 7
getWeight, CodelinkSet-method
 (CodelinkSet-class), 7
htmlpage, 24
imageCodelink, 13
log2, 14
logCodelink, 14
mergeArray, 14
na2false, 15
normalize, 16
normalize, Codelink-method (normalize), 16
normalize, CodelinkSet-method
 (normalize), 16
plot, 17
plotCorrelation, 17
plotCV, 18
plotDensities, 18
plotDensities, Codelink-method
 (plotDensities), 18
plotMA, 19
printHead, 20
probeNames (CodelinkSet-class), 7
probeNames, CodelinkSet-method
 (CodelinkSet-class), 7
probeNames, CodelinkSetUnique-method
 (CodelinkSet-class), 7
probeTypes (CodelinkSet-class), 7
probeTypes, CodelinkSet-method
 (CodelinkSet-class), 7
readCodelink, 21
readCodelinkSet (readCodelink), 21
readHeader, 22
readHeaderXLS, 23
reportCodelink, 23
selCV, 24
show, Codelink-method (Codelink-class), 6
SNR, 25
writeCodelink, 25
writeCodelink, CodelinkSet-method
 (writeCodelink), 25