R topics documented:

a.arise .................................................. 2
alter.unique ........................................... 3
analysis.window ....................................... 3
annotations ............................................. 4
back.gui .................................................. 4
bg.correct ............................................... 5
classes .................................................. 5
create.project ......................................... 6
cys.plot .................................................. 6
DataSet-class .......................................... 7
filter.spot ............................................... 7
genArise.init .......................................... 8
genArise .................................................. 8
genMerge ................................................ 9
get.values ............................................... 9
get.Zscore ............................................... 10
global.norm ............................................ 11
graphic.choose ........................................ 11
grid.narm ............................................... 12
help ...................................................... 12
i.arise .................................................. 13
imageLimma .............................................. 13
make.swap ............................................... 14
ma.plot ................................................ 15
m.arise .................................................. 16
meanUnique ............................................. 16
note ...................................................... 17
old.project ............................................. 17
post.analysis .......................................... 18
principal ................................................ 18
project.select ......................................... 19
r.arise .................................................. 19
read.dataset .......................................... 20
read.spot ................................................ 21
reset.history .......................................... 21
ri.plot .................................................. 22
set.grid.properties .................................... 22
**Description**

Extract A values from a Spot.

**Usage**

\[ a\text{.arise}(\text{mySpot}) \]

**Arguments**

- **mySpot**: Spot object for one microarray.

**Value**

List of A-values. \((\log_{2}(\text{cy3}) + \log_{2}(\text{cy5}))/2\)

**See Also**

- `m.arise`

**Examples**

```r
## read the spot from a file and save it in spot
data(Simon)
## Extract A from spot and save in a
a <- a.arise(mySpot = Simon)
```
**alter.unique**

### Remove Duplicates

**Description**

This function allows to remove from the spot repeated Id’s. Before moving one of the repeated Id’s the function compute the log ratio of both values with the same Id and delete the least absolute value if both of them are positive or negative. In other case delete both observations.

**Usage**

```r
alter.unique(mySpot)
```

**Arguments**

- `mySpot` Spot object for one microarray.

**Value**

Spot object without duplicates.

**Examples**

```r
data(Simon)
## filter the spot and save it in f.spot
f.spot <- filter.spot(Simon)
## remove duplicates and save it in u.spot
u.spot <- alter.unique(f.spot)
```

---

**analysis.window**

### Analysis.window

**Description**

Auxiliar function of genArise GUI, in this window you can apply operations to original data.

**Usage**

```r
analysis.window(texto, follow.wizard = FALSE, envir, swap)
```

**Arguments**

- `texto` Historial project string
- `follow.wizard` Boolean value, if this argument is TRUE, an data analysis are performed
- `envir` Environment where are the project variables
- `swap` Is this a swap analysis or an individual analysis

**Value**

tkwidget
annotations  Gene Annotations

Description

Performed an HTML file

Usage

annotations(specie.data, specie, column, symbol, output.file = "annotations.html")

Arguments

specie.data  A data frame  
specie       Name of specie  
column       Number of column where are the gene name in the data frame  
symbol       An optional symbol besides GenBank ID  
output.file  Name of output file

Value

HTML file with link for each spot in data frame

back.gui  Return to the last window

Description

Auxiliar function of genArise GUI.

Usage

back.gui(envir)

Arguments

envir  Environment where are the project variables

Value

tkwidget
**bg.correct**

**Background Correction**

**Description**

This function uses the background data to eliminate unwanted effects in signal. The background correction establishes the new Cy3 signal as Cy3 - BgCy3 and the new Cy5 as Cy5 - BgCy5.

**Usage**

```r
bg.correct(mySpot)
```

**Arguments**

- `mySpot`: Spot object for one microarray.

**Value**

Spot object with the background correction done.

**Examples**

```r
data(Simon)
## background correction and save it in c.spot
c.spot <- bg.correct(Simon)
```

**classes**

**Classes Defined by this Package**

**Description**

This package defines the following data classes.

- **Spot**: A class used to store spot data with the following attributes: Cy3, Cy5, BgCy3, BgCy5, Ids as they are read by `read.spot` or obtained from a function that return a spot object.

- **DataSet**: A class used to store spot data with the following attributes: Cy3, Cy5, Ids, Z-score.
create.project

Description
 Auxiliar function for genAriseGUI

Usage
 create.project(project.name, results.file = "Results", graphics.file = "Graphics")

Arguments
 project.name
 results.file
 graphics.file

cys.plot

Description
 Data Visualization: log2(Cy3) vs log2(Cy5)

Usage
 cys.plot(mySpot, col = "green")

Arguments
 mySpot An Spot object
 col Color in which the points of the plot will be shown. This argument must be quoted and the possible values it can take are the same from the color function in the R base.

Examples
 data(Simon)
cys.plot(Simon)
**DataSet-class**  

**Description**  
A simple list-based class for storing red and green channel foreground, z-scores and the Ids.

**Creating Objects from the Class**  
Objects can be created by calls of the form `new("DataSet", sets, type)` where `sets` is a list containing Cy3, Cy5, Id and Zscore and `type` is "ri" or "ma". Objects are normally created by `read.spot`.

**Slots/List Components**  
This class contains no slots (other than `.Data`), but objects should contain the following list components:

- **Cy5**: numeric matrix containing the red (cy5) foreground intensities. Rows correspond to spots and columns to arrays.
- **Cy3**: numeric matrix containing the green (cy3) foreground intensities.
- **Id**: Ids from all the observations.
- **Zscore**: The result of `(R - mean) / sd` that define an intensity-dependent Z-score threshold to identify differential expression.

All of these matrices should have the same dimensions.

**Methods**  
This class inherits directly from class `list` so any operation appropriate for lists will work on objects of this class.

---

**filter.spot**  

**Intensity-based filtering of array elements**

**Description**  
This function keep only array elements with intensities that are 2 standard deviation above background.

**Usage**  

```
filter.spot(mySpot)
```

**Arguments**

- **mySpot**  
  Spot object for one microarray.

**Value**  
Array elements with intensities that are 2 standard deviation above background.
References

Examples
data(Simon)
## background correction and save it in c.spot
c.spot <- bg.correct(Simon)
## normalize spot
n.spot <- grid.norm(c.spot, nr = 23, nc = 24)
## filtering the spot
filter.spot(n.spot)

description
Auxiliar function of genArise GUI, this function show a principal menu of genAriseGUI

Usage
genArise.init(envir)

Arguments
envir Environment where are the project variables

Value
tkwidget

Description
This is the main function and display the GUI of genArise.

Usage
genArise()
**genMerge**

*genMerge: Post-Genomic Analysis*

**Description**

After we finished our slice analysis we get a up-regulated and down-regulated set. This will be the set of study genes for genMege. Given this set, genMerge retrieves functional genomic data for each gene and provides statistical rank scores for over-representation of particular functions in the dataset.

**Usage**

```r
genMerge(gene.association, description, population.genes, study.genes, output.file = "GenMerge.txt")
```

**Arguments**

- `gene.association` The gene-association file links gene names with a particular datum of information using a shorthand of gene-association IDS
- `description` The description file contains human-readable description of gene-association IDS
- `population.genes` Set of all genes detected on a array
- `study.genes` Set of genes may be those that are up-regulated or down-regulated or both of them.
- `output.file` The name of output file that includes all results obtained after this analysis.

**Note**

This function is completely based on GeneMerge from Cristian I. Castillo-Davis and Daniel L. Hartl

**References**


---

**get.values**

*Auxiliar function for post-analysis*

**Description**

This function get values from an DataSet object.

This is just a function for the GUI, and can not be used in the command line.

**Usage**

```r
get.values(list.values, genes.values, up.down, min.val, max.val)
```
get.Zscore

Arguments

list.values  Zscore values from DataSet object
genes.values  Ids values from DataSet object
up.down      If the analysis will be done with "up" or "down" regulated
min.val      Minimal value of the range
max.val      Maximal value of the range

Value

An Ids list

Description

Read both files, but only extract the interested columns and create a Spot object.

Usage

get.Zscore( spot, name, Zscore.min=NULL, Zscore.max=NULL, all=FALSE, envir)

Arguments

spot          a connection or a character string giving the name of the file to read where each
column represent the spot components.
name          a connection or a character string giving the name of the file to read where each
column represent the spot components.
Zscore.min    column that represent Cy3.
Zscore.max    column that represent Cy5.
all           column that represent BgCy3.
envir         Environment where are the genArise variables.

See Also

write.spot.
global.norm

Description

This function normalize R and I values and fit the value of Cy5 from his argument. In this function
the normalize algorithm will be applied to all observations to get the lowess factor and then fit Cy5
with this factor. The observations. The observations with values R = 0 are deleted because they
have no change in their expression levels.

Usage

global.norm(mySpot)

Arguments

mySpot A spot object

Value

A new spot object but normalized, It means with a different Cy5 that is the result of the fit with the
lowess factor.

Examples

data(Simon)
# Background Correction
c.spot <- bg.correct(Simon)
#Normalized data
n.spot <- global.norm(c.spot)

graphic.choose

Description

This function show the plot of an spot object. This plot are identify with the graphic.type.value

Usage

graphic.choose(spot.object, graphic.type)

Arguments

spot.object An object ob Spot class
graphic.type representative integer of type graphic will be plot

Value

Plot device
grid.norm  

Normalization by grid of Spot

Description

This function normalize R and I values and fit the value of Cy5 for each grid in the spot that it receives as argument. In this function the dimension of grid is (meta-row * meta-column).

Usage

grid.norm(mySpot, nr, nc)

Arguments

mySpot  
Spot object for one microarray.

nr  
Total of meta-row.

nc  
Total of meta-column.

Value

Spot object with the grid normalization done.

Examples

data(Simon)
## background correction and save it in c.spot
c.spot <- bg.correct(Simon)
## normalization and save it in n.spot
n.spot <- grid.norm(c.spot, 23, 24)

help

Help of genArise

Description

Display the help of genArise in the GUI. This is just a function for the GUI, and can not be used in the command line.

Usage

help()
**i.arise**

*I Arise*

**Description**

Extract I from a Spot.

**Usage**

\[ i.arise(\text{mySpot}) \]

**Arguments**

- **mySpot**: Spot object for one microarray.

**Value**

List of I-values

**See Also**

`r.arise`.

**Examples**

```r
data(Simon)
## Extract I from spot and save in i
i.arise(Simon)
```

---

**imageLimma**

*Image Plot of Microarray*

**Description**

Plot an image of colours representing the log intensity ratio for each spot on the array. This function can be used to explore whether there are any spatial effects in the data.

**Usage**

\[
\text{imageLimma}(z, \text{row}, \text{column}, \text{meta.row}, \text{meta.column}, \
\text{low} = \text{NULL}, \text{high} = \text{NULL})
\]
Arguments

z: numeric vector or array. This vector can contain any spot statistics, such as log intensity ratios, spot sizes or shapes, or t-statistics. Missing values are allowed and will result in blank spots on the image.

row: rows in the microarray.

column: columns in the microarray.

meta.row: metarows in the microarray.

meta.column: metacolumns in the microarray.

low: color associated with low values of 'z'. May be specified as a character string such as "'green'", "'white'" etc, or as a rgb vector in which 'c(1,0,0)' is red, 'c(0,1,0)' is green and 'c(0,0,1)' is blue. The default value is "'green'" if 'zerocenter=T' or "'white'" if 'zerocenter=F'.

high: color associated with high values of 'z'. The default value is "'red'" if 'zerocenter=T' or "'blue'" if 'zerocenter=F'.

Note

This function is based in the imageplot function from limma package.

References


Examples

data(Simon)
spot.data <- attr(Simon, "spotData")
M <- log(spot.data$Cy5, 2) - log(spot.data$Cy3, 2)
imageLimma(z = M, row = 23, column = 24, meta.row = 2, meta.column = 2, low = NULL, high = NULL)

Description

Read both files, but only extract the interested columns and create a Spot object.

Usage

make.swap(spot1, spot2, Cy3, Cy5, BgCy3, BgCy5, Id, Symdesc, header = FALSE, is.ifc = FALSE, envir, nr, nc)
Arguments

- **spot1**: a connection or a character string giving the name of the file to read where each column represent the spot components.
- **spot2**: a connection or a character string giving the name of the file to read where each column represent the spot components.
- **Cy3**: column that represent Cy3.
- **Cy5**: column that represent Cy5.
- **BgCy3**: column that represent BgCy3.
- **BgCy5**: column that represent BgCy5.
- **Id**: column that represent Id.
- **Symdesc**: optional identifier besides the Id column.
- **header**: the logical value of the header input file.
- **is.ifc**: If is.ifc = TRUE this experiment was done in the Unit of Microarray from Cellular Physioloogy Institute.
- **envir**: Environment where are the genArise variables.
- **nr**: Total of meta-row.
- **nc**: Total of meta-column.

See Also

- `write.spot`

Data Visualization: M vs. A plot

Description

This function allows to plot M -vs- A in spot.

Usage

```r
ma.plot(mySpot, col = "green")
```

Arguments

- **mySpot**: Spot for one microarray.
- **col**: color of points in graphic

Examples

```r
data(Simon)
## plot the signals for spot.
ma.plot(Simon)
```
m.arise \hspace{1cm} M Arise

**Description**

Extract M values from a Spot.

**Usage**

```r
m.arise(mySpot)
```

**Arguments**

- `mySpot` Spot object for one microarray.

**Value**

List of M-values

**See Also**

`a.arise`.

**Examples**

```r
data(Simon)
## Extract M from spot and save in m
m <- m.arise(Simon)
```

---

**meanUnique **

**Remove Duplicates**

**Description**

This function allows to remove from the spot repeated Id’s. Before moving one of the repeated Id’s the function compute the average of Cy3 intensity and Cy5 intensity.

**Usage**

```r
meanUnique(mySpot)
```

**Arguments**

- `mySpot` Spot object for one microarray.

**Value**

Spot object without duplicates
Examples

data(Simon)
c.spot <- bg.correct(Simon)
n.spot <- global.norm(c.spot)
f.spot <- filter.spot(n.spot)
meanUnique(f.spot)

Description

Call a editor for note about actual experiment

Usage

note(envir)

Arguments

envir Environment where are the experiment variables

old.project

Open previous project

Description

Show the information that was obtained from the analysis of a previous project. This is just an auxiliar function for genAriseGUI, and can not be used in the command line.

Usage

old.project(project.name, envir, parent)

Arguments

project.name path of project file (PRJ)
envir Environment where are the genArise variables
parent The parent widget

Value

tkwidget
post.analysis  

Set-combinatorial analysis

Description

This function allows you to perform a set combinatorial analysis between the results previously obtained in different projects. This function is called post.analysis and it is mandatory that you have done the Zscore operation in all the selected projects. It is important to clarify that this function receives a list of files with extension prj as argument and for this reason you can’t use it if the results to compare was not obtained by the genArise GUI.

Usage

post.analysis(values, min.val, max.val, up.down, output)

Arguments

- **values**: A list of projects to compare
- **min.val**: The minimal value of the range
- **max.val**: The maximal value of the range
- **up.down**: If the analysis will be done with "up" or "down" regulated
- **output**: The directory that will contain all the output files

Value

Once obtained the ids list for each project a number of files with extension set are created in a directory. The name of this files consists in a sequence of 0 and 1. The number of digits in the file names is the same to the number of projects in the list passed as argument to the function. There is then, a relation between the number of digits in the file names and the projects. This relation is defined by the position specified in the file order.txt in the same directory you have passed as another argument in the function.

principal  

Principal window of genAriseGUI

Description

This function show a window with the information of experiment like name and dimensions, too plot an image of colours representing the log intensity ratio for each spot on the array. This is just an auxiliar function for genAriseGUI, and can not be used in the command line.

Usage

principal(envir, swap)

Arguments

- **envir**: Environment where are the genArise variables
- **swap**: Is this a swap analysis or an individual analysis
project.select

Value
tkwidget

Description
Previous window to post-analysis. In this window you can select one or several files (projects) and arguments to be used by post analisis function.

This is just an auxiliar function for genAriseGUI, and can not be used in the command line.

Usage
projects.select(envir, nombre)

Arguments

\begin{itemize}
\item envir Environment where are the genArise variables
\item nombre Name of directory where the post-analysis results will be placed.
\end{itemize}

Value
tkwidget

r.arise

Get R value

Description
Get the \textbf{R values} from an object of the Spot class.

Usage
r.arise(mySpot)

Arguments

\begin{itemize}
\item mySpot An object of the Spot class
\end{itemize}

Value
A vector containing the R value ( \( \log(Cy5/Cy3) \)) for each observation of the spot object.

See Also
i.arise.
Examples

data(Simon)
# Get R-value from an object of the Spot class and save the result
R <- r.arise(Simon)
# Show the R-values

read.dataset

Read Dataset from File

Description

Read all file and extract the interested columns to create a DataSet object (this file contain the zscore with all the genes after the duplicates filtering and makes not distinction between up-regulated and down-regulated. If you want to make this distinction you must write the data with the function write.dataSet, but there is no way to read this files with this function).

Usage

read.dataset(file.name, cy3 = 1, cy5 = 2, ids = 3, symdesc = NULL, zscore = 4, type = 6, header = FALSE, sep = "\t")

Arguments

file.name  a connection or a character string giving the name of the file to read where each column represent the dataset components.
cy3  column that represent Cy3.
cy5  column that represent Cy5.
ids  column that represent Id.
symdesc  optional identifier besides Id column.
zscore  column that represent the zscore value.
type  column that represent if the experiment was performed as R vs I or M vs A.
header  the logical value of the header input file
sep  the separator in the inputfile

See Also

write.zscore.
read.spot

Read Spot from File

Description

Read all file, but only extract the interested columns and create a Spot object.

Usage

read.spot(file.name, cy3, cy5, bg.cy3, bg.cy5, ids, symdesc, header = FALSE, sep = "\t", is.ifc = FALSE, envir)

Arguments

- file.name: a connection or a character string giving the name of the file to read where each column represent the spot components.
- cy3: column that represent Cy3.
- cy5: column that represent Cy5.
- bg.cy3: column that represent BgCy3.
- bg.cy5: column that represent BgCy5.
- ids: column that represent Id.
- symdesc: (optional) identifier besides Id column.
- header: the logical value of the header input file
- sep: the separator in the inputfile
- is.ifc: If is.ifc = TRUE this experiment was done in the Unit of Microarray from Cellular Phisiology Institute.
- envir: Environment where are the genArise variables. You don’t need to specify this argument.

See Also

write.spot.

reset.history

Reset the prj history file

Description

Clean all the operations saved in the prj history file.

Usage

reset.history(history.file, text)
Arguments

history.file  The name of the prj history file.
text        The new content of the prj history file.

Value

The history file without operations.

Description

This function allows to plot R-values vs I-values from a Spot object.

Usage

ri.plot(mySpot, col = "green")

Arguments

mySpot  Spot Object
col     Color in which the points of the plot will be shown. This argument must be quoted and the possible values it can ake ares the same from the colors funcion in the R base package.

See Also

colors()

Examples

data(Simon)
ri.plot(Simon)

Description

Auxiliary function for genAriseGUI

Usage

set.grid.properties(envir, name, nr, nc, nmr, nmc)
Arguments

- **envir**: Environment where the variables are stored
- **name**: The name of the experiment
- **nr**: Total rows in the array (each row represents a spot)
- **nc**: Total columns in the array
- **nmr**: Total of meta-rows
- **nmc**: Total of meta-columns

---

**set.history.project**

*Save the history of a project*

**Description**

Save in the history file each operation performed while the analysis. This is just to get the open this particular project in the future. This is just an auxiliary function for the GUI, and can not be used in the command line.

**Usage**

```r
set.history.project(history.file, id.name, data.file)
```

**Arguments**

- **history.file**: The name of the prj history file.
- **id.name**: The name of the operation.
- **data.file**: The file with the results of the operation.

**Value**

The history file with the new performed operation.

---

**set.path.project**

**Description**

Auxiliar function for genAriseGUI

**Usage**

```r
set.path.project(path, results.file, graphics.file, envir)
```

**Arguments**

- **path**: Project path value
- **results.file**: Name of directory where results file will be
- **graphics.file**: Name of directory where pdf graphics will be
- **envir**: Environment where are the experiment variables
Description

Auxiliar function for genAriseGUI

Usage

set.project.properties(envir)

Arguments

envir Environment where are the experiment variables

Simon

Dataset: Little fragment of a microarray from IFC UNAM

Description

This structure is a data fragment of a yeast microarray from the Microarrays Unit in IFC UNAM. The original microarray contains 6 meta-rows and 4 meta-columns, however this data just belongs to the first meta-row order in a way of 2 meta-rows and 2 meta-columns.

Usage

data(Simon)

Format

A list that contains 1104 observations, because the dimensions of this example are: 2 meta-rows, 2 meta-columns, 23 rows, 24 columns.

Examples

data(Simon)

#A preview from the chip
datos <- attr(Simon, "spotData")
M <- log(datos$Cy3, 2) - log(datos$Cy5, 2)
imageLimma(M, 23, 24, 2, 2)
single.norm  

**Swap from Files**

**Description**
Read both files, but only extract the interested columns and create a Spot object.

**Usage**
```r
single.norm(envir)
```

**Arguments**
- `envir` Environment where are the genArise variables.

**See Also**
- `write.spot`

---

**Spot-class**

**Spot-class**

**Description**
A simple list-based class for storing red and green channel foreground and background intensities for a batch of spotted microarrays and the Ids.

**Creating Objects from the Class**
Objects can be created by calls of the form `new("Spot", spot)` where `spot` is a list. Objects are normally created by `read.spot`.

**Slots/List Components**
This class contains no slots (other than `.Data`), but objects should contain the following list components:

- `Cy5`: numeric matrix containing the red (cy5) foreground intensities. Rows correspond to spots and columns to arrays.
- `Cy3`: numeric matrix containing the green (cy3) foreground intensities.
- `BgCy5`: numeric matrix containing the red (cy5) background intensities.
- `BgCy3`: numeric matrix containing the green background intensities.
- `Id`: Ids from all the observations.

All of these matrices should have the same dimensions.

**Methods**
This class inherits directly from class `list` so any operation appropriate for lists will work on objects of this class.
spotUnique  Replicate filtering

Description
We consider replicate measures of two samples and adjust the log(ratio,2) measures for each gene so that the transformed values are equal. To do this we take the geometric mean. This procedure can be extended to averaging over \( n \) replicates.

Usage
spotUnique(mySpot)

Arguments
mySpot  Spot object for one microarray.

Value
Spot object without duplicates

Examples
data(Simon)
c.spot <- bg.correct(Simon)
f.spot <- filter.spot(c.spot)
spotUnique(mySpot = f.spot)

swap.select  Dye swap files selector

Description
This is just an auxiliary function for genAriseGUI, and can not be used in the command line.

Usage
swap.select(envir)

Arguments
envir  Environment where are the genArise variables

Value
tkwidget
trim

**Description**

Extract white spaces at the beginning or end of a word.

**Usage**

```r
trim(word)
```

**Arguments**

- `word` A string of characters possibly with white spaces at the beginning or end of the string.

**Value**

Returns a string of characters, with leading and trailing whitespace omitted.

**Examples**

```r
trim(" This is a String   ")
## return [1] "This is a String"
```

write.dataSet

**Write dataSet**

Write the values for observations of an object of DataSet class in an output file. This values are written in columns with the following order: Cy3, Cy5, Cy3 Background, Cy5 Background, Ids and finally the Zscore value. By default this output file has no header.

**Usage**

```r
write.dataSet(dataSet.spot, fileName, quote = FALSE, col.names = FALSE, row.names = FALSE, Zscore.min = NULL, Zscore.max = NULL, sep = "\t")
```

**Arguments**

- `dataSet.spot` An object of DataSet class
- `fileName` The name of the output file where the data will be written. This argument must be quoted.
- `quote` If `quote = TRUE`, all values in the file will be quoted.
- `col.names` If `col.names = TRUE`, an integer is written in every column as header. By default `col.names = FALSE`. 
row.names  If row.names = TRUE will be an extra column that numerates every rows in the file.

Zscore.min  The lower value in a range, if Zscore.min = NULL then the file will contain all values bellow Zscore.max

Zscore.max  The greater value in a range, if Zscore.max = NULL then file will be contain all values above Zscore.min. Both values, Zscore.min and Zscore.max can not be NULL

sep  Character to separate the columns in file. By default sep = "\t".

Examples

```r
data(WT.dataset)
write.dataSet(dataSet.spot = WT.dataset, fileName = "Example.csv", Zscore.min = 1,
Zscore.max = 1.5, sep = "\t")
```

write.spot  Write Spot

Description

Write the values for observations of an object of Spot class in an output file. This values are writen in columns with the follow order: Cy3, Cy5, Cy3 Background, Cy5 Background and finally Ids. By default this file has no header.

Usage

```r
write.spot(spot, fileName, quote = FALSE, sep = "\t",
col.names = FALSE, row.names = FALSE)
```

Arguments

- **spot**  An object of Spot class
- **fileName**  The name of the output file where the data will be writen. This argument must be quoted.
- **quote**  If quote = TRUE, all values in the file will be quoted.
- **sep**  Character to separate the columns in file. By default sep = "\t".
- **col.names**  If col.names = TRUE, an integer is writen in every column as header. By default col.names = FALSE.
- **row.names**  If row.names = TRUE will be an extra column that numerates every rows in the file.

Examples

```r
data(Simon)
write.spot(spot = Simon, fileName = "Example.csv", quote = FALSE, sep = "\t",
col.names = FALSE, row.names = FALSE)
```
**write.zscore**

*Write Z-score data*

**Description**

Write the values for observations of an object of `DataSet` class in an output file. This values are written in columns tab separated with the following order: Cy3, Cy5, Cy3 Background, Cy5 Background, Ids and finally the z-score value. The header of the output file is the selected type for the z-score (ri or ma).

**Usage**

```r
code
```

**Arguments**

- `dataSet.spot`: An object of `DataSet` class
- `fileName`: The name of the output file where the data will be written. This argument must be quoted.
- `sep`: Character to separate the columns in file. By default `sep = "\t"`.

**Examples**

```r
data(WT.dataset)
write.zscore(dataSet.spot = WT.dataset, fileName = "Zscore.csv", sep = "\t")
```

---

**WT.dataset**

*Microarray from the IFC*

**Description**

This data set is a Microarray from the IFC.

**Usage**

```r
data(WT.dataset)
```

**Format**

A vector containing 4036 observations.

**Examples**

```r
data(WT.dataset)
Zscore.plot(WT.dataset)
```
Zscore.plot  

**Z-score Data Visualization: R vs I or M vs A**

**Description**

This function allows to plot **R-values** vs **I-values** or **M-values** vs **A-values** for identifying differential expression.

**Usage**

```r
Zscore.plot(dataSet.spot, Zscore.min, Zscore.max, all, col)
```

**Arguments**

- `dataSet.spot`  
  Spot Object
- `Zscore.min`  
  The lower value in a range, if `Zscore.min = NULL` then the file will contain all values below `Zscore.max`
- `Zscore.max`  
  The greater value in a range, if `Zscore.max = NULL` then file will be contain all values above `Zscore.min`. Both values, `Zscore.min` and `Zscore.max` can not be `NULL`
- `all`  
  Plot all the observations in four sets: $Z < 1, 1 < Z < 1.5, 1.5 < Z < 2, Z > 2$
- `col`  
  Color in which the points of the plot will be shown where only the points from center are plot. This argument must be quoted and the possible values it can take are the same from the colors function in the R base package.

**See Also**

`colors()`

**Examples**

```r
data(WT.dataset)
Zscore.plot(WT.dataset, Zscore.min = 1, Zscore.max = 2)
```

---

Zscore.points  

**Z-score Window**

**Description**

This function display the window that show the results after the Z-score. This window allow:

1. Show the plots of the up and down generated with the function `Zscore.plot` regulated spots in:  
   - `Zscore < 1 sd`
   - `1 sd < Zscore < 1.5 sd`
   - `1.5 sd < Zscore < 2 sd`
   - `Zscore > 2 sd` and All the points
2. Save the plots in pdf and save the results in an output file

This is just a function for the GUI, and can not be used in the command line.
Zscore

Usage

\texttt{Zscore.points(type,text,envir, swap)}

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>Type of analysis done: &quot;ri&quot; is for a R-I analysis and &quot;ma&quot; is for M-A analysis</td>
</tr>
<tr>
<td>text</td>
<td>The text for the text area of the history of the project</td>
</tr>
<tr>
<td>envir</td>
<td>Environment where the variables are stored</td>
</tr>
<tr>
<td>swap</td>
<td>Is this a swap analysis or an individual analysis</td>
</tr>
</tbody>
</table>

Zscore

\textit{Z-scores for identifying differential expression}

Description

This function identify differential expressed genes by calculating an intensity-dependent Z-score. This function use a sliding window to calculate the mean and standard deviation within a window surrounding each data point, and define a Z-score where Z measures the number of standard deviations a data point is from the mean.

Usage

\texttt{Zscore(spot.object,type,window.size)}

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spot.object</td>
<td>A spot object</td>
</tr>
<tr>
<td>type</td>
<td>Type of analysis: &quot;ri&quot; is for a R-I analysis and &quot;ma&quot; is for M-A analysis</td>
</tr>
<tr>
<td>window.size</td>
<td>Size of the sliding window</td>
</tr>
</tbody>
</table>

Value

A \texttt{dataSet} object with attributes \texttt{Cy3}, \texttt{Cy5}, \texttt{Id}, Z-score.

Examples

\begin{verbatim}
data(Simon)
# Background Correction
c.spot <- bg.correct(Simon)
#Normalized data
n.spot <- grid.norm(c.spot,23,24)
#Filter spot
f.spot <- filter.spot(n.spot)
#Replicate filtering
u.spot <- spotUnique(f.spot)
#Zscore analysis
s.spot <- Zscore(u.spot)
\end{verbatim}
Index

**Topic IO**
- back.gui, 3
- set.grid.properties, 21

**Topic aplot**
- cys.plot, 5
- ma.plot, 14
- ri.plot, 21
- Zscore.plot, 29
- Zscore.points, 29

**Topic arith**
- a.arise, 1
- alter.unique, 2
- create.project, 5
- filter.spot, 6
- genArise, 7
- get.values, 8
- global.norm, 10
- grid.norm, 11
- m.arise, 15
- ma.plot, 14
- note, 16
- r.arise, 18
- set.path.project, 22
- set.project.properties, 23
- Zscore, 30

**Topic character**
- trim, 26

**Topic classes**
- DataSet-class, 6
- Spot-class, 24

**Topic color**
- imageLimma, 12
- principal, 17

**Topic datasets**
- Simon, 23
- WT.dataset, 28

**Topic data**
- DataSet-class, 6
- Spot-class, 24

**Topic documentation**
- classes, 4
- help, 11

**Topic file**
- get.Zscore, 9
- make.swap, 13
- old.project, 16
- post.analysis, 17
- read.dataset, 19
- read.spot, 20
- reset.history, 20
- set.history.project, 22
- single.norm, 24
- swap.select, 25
- write.dataSet, 26
- write.spot, 27
- write.zscore, 28

**Topic graphs**
- cys.plot, 5

**Topic math**
- a.arise, 1
- alter.unique, 2
- analysis.window, 2
- annotations, 3
- bg.correct, 4
- filter.spot, 6
- genArise, 7
- genArise.init, 7
- genMerge, 8
- global.norm, 10
- graphic.choose, 10
- grid.norm, 11
- i.arise, 12
- m.arise, 15
- meanUnique, 15
- r.arise, 18
- spotUnique, 25
- Zscore, 30
INDEX

classes, 4
create.project, 5
cys.plot, 5

DataSet, 4
DataSet-class, 6

filter.spot, 6
genArise, 7
genArise.init, 7
genMerge, 8
get.values, 8
get.Zscore, 9
global.norm, 10
graphic.choose, 10
grid.norm, 11

help, 11

i.arise, 12, 18
imageLimma, 12

m.arise, 1, 15
ma.plot, 14
make.swap, 13
meanUnique, 15

note, 16

old.project, 16

post.analysis, 17
principal, 17
project.select, 18
projects.select(project.select), 18

r.arise, 12, 18
read.dataset, 19
read.spot, 6, 20, 24, 27
reset.history, 20
ri.plot, 21

set.grid.properties, 21
set.history.project, 22
set.path.project, 22
set.project.properties, 23
Simon, 23
single.norm, 24
Spot, 4
Spot-class, 24
spotUnique, 25
swap.select, 25

trim, 26
write.dataSet, 26
write.spot, 9, 14, 20, 24, 27
write.zscore, 19, 28
WT.dataset, 28
Zscore, 30
Zscore.plot, 29
Zscore.points, 29