

# Package ‘HELP’

May 14, 2024

**Version** 1.62.0

**Date** 2009-09-30

**Title** Tools for HELP data analysis

**Author** Reid F. Thompson <reid.thompson@gmail.com>, John M. Greally <john.greally@einstein.yu.edu>, with contributions from Mark Reimers <mreimers@vcu.edu>

**Maintainer** Reid F. Thompson <reid.thompson@gmail.com>

**Depends** R (>= 2.8.0), stats, graphics, grDevices, Biobase, methods

**Description** The package contains a modular pipeline for analysis of HELP microarray data, and includes graphical and mathematical tools with more general applications.

**License** GPL (>= 2)

**biocViews** CpGIsland, DNAMethylation, Microarray, TwoChannel, DataImport, QualityControl, Preprocessing, Visualization

**git\_url** <https://git.bioconductor.org/packages/HELP>

**git\_branch** RELEASE\_3\_19

**git\_last\_commit** a4beffc

**git\_last\_commit\_date** 2024-04-30

**Repository** Bioconductor 3.19

**Date/Publication** 2024-05-14

## Contents

base.stacking.thermodynamics . . . . .	2
calcGC . . . . .	3
calcGC-methods . . . . .	4
calcPrototype . . . . .	4
calcPrototype-methods . . . . .	5
calcTm . . . . .	6
calcTm-methods . . . . .	7
combineData . . . . .	7
combineData-methods . . . . .	9

createWiggle . . . . .	10
createWiggle-methods . . . . .	11
exprs2 . . . . .	12
exprs2-methods . . . . .	12
fuzzyMatches . . . . .	13
fuzzyMatches-methods . . . . .	14
getFeatures . . . . .	15
getFeatures-methods . . . . .	16
getSamples . . . . .	17
getSamples-methods . . . . .	18
plotBins . . . . .	19
plotBins-methods . . . . .	20
plotChip . . . . .	21
plotChip-methods . . . . .	22
plotFeature . . . . .	23
plotFeature-methods . . . . .	24
plotPairs . . . . .	25
plotPairs-methods . . . . .	26
quantileNormalize . . . . .	27
quantileNormalize-methods . . . . .	28
readDesign . . . . .	29
readDesign-methods . . . . .	31
readPairs . . . . .	32
readPairs-methods . . . . .	34
readSampleKey . . . . .	35

<b>Index</b>	<b>37</b>
--------------	-----------

---

base.stacking.thermodynamics

*Base-stacking thermodynamic parameters*

---

## Description

Unified thermodynamic parameters (delta-S and delta-H values) for nearest-neighbor base stacking calculations

## Usage

data(base.stacking.thermodynamics)

## Format

A matrix with 2 columns and 16 rows. Column 1 indicates enthalpic parameters (dH) and column 2 indicates entropic parameters (dS). Rows indicate all possible 2bp combinations of "A", "T", "C", and "G" (e.g. "AC")

**Source**

Allawi, H.T. and SantaLucia, J., Jr. (1997) Thermodynamics and NMR of internal G.T mismatches in DNA, *Biochemistry*, **36**, 10581-10594.

**Examples**

```
data(base.stacking.thermodynamics)
```

---

calcGC	<i>Calculate GC percent</i>
--------	-----------------------------

---

**Description**

Function to calculate GC percent from a nucleotide sequence input

**Usage**

```
calcGC(x, ...)
```

**Arguments**

x characters containing nucleotide sequence (ex: "ATCGGAA") or an object of class ExpressionSet

... Other arguments passed to methods:  
'allow' vector of characters specifying what other characters to allow in sequence (default is "N")

**Value**

Returns a numerical value (from '0' to '1') indicating the C+G content of the sequence, corresponding to the fraction of (C+G)/(A+T+C+G...). A value of 'NA' is returned if the function encounters an error that prevents proper calculation of GC percent.

**Author(s)**

Reid F. Thompson (<rthomps@aeocom.yu.edu>)

**See Also**

[calcGC-methods](#), [calcTm](#)

**Examples**

```
#demo(pipeline,package="HELP")  
  
calcGC("AAAACGCG")  
calcGC(sequence="cXgXcXgXcXgX",allow="X")
```

---

calcGC-methods	<i>Calculate GC percent (methods)</i>
----------------	---------------------------------------

---

### Description

Methods for calculating GC percent from oligonucleotide sequences

### Methods

**x = "missing"** Handle empty function call

**x = "NULL"** Handle empty function call

**x = "character"** Handle character input

**x = "ExpressionSet"** Handle input of an object of class ExpressionSet

### See Also

[calcGC](#)

---

calcPrototype	<i>Calculate prototype</i>
---------------	----------------------------

---

### Description

Calculates prototype (trimmed mean) across all samples

### Usage

```
calcPrototype(x, ...)
```

### Arguments

x	a numeric matrix, where each column represents a different sample
...	Arguments to be passed to methods (see <a href="#">calcPrototype-methods</a> ):
‘element’	which element of AssayData to use for a given ExpressionSet input (default is “exprs”)
‘samples’	which samples to use as data. Can be a vector of characters matching sample names, integers indicating which samples to choose, or a mixture of the two. If ‘NULL’ (default), all samples will be used.
‘center’	logical; if ‘TRUE’ (default) samples will be mean-centered before prototype calculation. If ‘FALSE’, this mean-centering step will be skipped
‘trim’	the fraction (0 to 0.5, default is 0.1) of observations to be trimmed from each end of each row and column in x before the mean is computed.
‘verbose’	logical; if ‘TRUE’ (default) progress will be output to screen. If ‘FALSE’, no output will be displayed.
‘dots’	other arguments to be passed to mean. See <a href="#">mean</a> .

**Value**

Returns a vector of numerical data, representing the prototype ([trimmed] mean) of all samples in *x*.

**Author(s)**

Mark Reimers (<mreimers@vcu.edu>), Reid F. Thompson (<rthomps@aeacom.yu.edu>)

**See Also**

[calcPrototype-methods](#), [mean](#)

**Examples**

```
#demo(pipeline, package="HELP")

x <- matrix(data=rep(1:1000,10),nrow=1000,ncol=10)
x <- x*(sample(1:100/100,size=10000,replace=TRUE))
x <- t(t(x)-1000*(1:10))
x[c(1:10,991:1000),]
x.avg <- calcPrototype(x)
x.avg[c(1:10,991:1000)]

#rm(x,x.avg)
```

---

calcPrototype-methods *Calculate prototype (methods)*

---

**Description**

Methods for calculating prototype ([trimmed] mean) across all samples

**Methods**

**x = "missing"** Handle empty function call

**x = "ExpressionSet"** Handle input of an object of class ExpressionSet. Derive data from AssayData.

**x = "vector"** Handle vector input as a matrix

**x = "matrix"** Handle matrix input

**Author(s)**

Mark Reimers (<mreimers@vcu.edu>), Reid F. Thompson (<rthomps@aeacom.yu.edu>)

**See Also**

[calcPrototype](#)

---

`calcTm`*Calculate Tm*

---

**Description**

Calculate melting temperature (Tm) using the nearest-neighbor base-stacking algorithm and the unified thermodynamic parameters.

**Usage**

```
calcTm(x, ...)
```

**Arguments**

<code>x</code>	characters containing nucleotide sequences (ex: "ATCGGAA") or an object of class ExpressionSet
<code>...</code>	Additional arguments passed to methods: <ul style="list-style-type: none"><li>'strand1.concentration' numeric value specifying concentration of strand 1 (default is 2e-07)</li><li>'strand2.concentration' numeric value specifying concentration of strand 2 (default is 2e-07)</li><li>'method' character value specifying the Tm algorithm to use (default is "nearest-neighbor"); currently not supported</li></ul>

**Value**

Returns a numerical value indicating the predicted melting temperature (Tm) of the sequence in degrees Celsius. A value of 'NA' is returned if the function encounters an error that prevents proper Tm calculation.

**Author(s)**

Reid F. Thompson (<rthomps@aeocom.yu.edu>)

**References**

Allawi, H.T. and SantaLucia, J., Jr. (1997) Thermodynamics and NMR of internal G.T mismatches in DNA, *Biochemistry*, **36**, 10581-10594.

**See Also**

[calcTm-methods](#), [base.stacking.thermodynamics](#), [calcGC](#)

**Examples**

```
#demo(pipeline,package="HELP")

calcTm("GTGTGGCTACAGGTGGGCCGTGGCGCACCTAAGTGAGGACAGAGAACAAC")
calcTm("GTGTGGCTACAGGTGGGCCGTGGCGCACCTAAGTGAGGACAGAGAACAAC",strand1.concentration=1E-5,strand2.concentration=2E-5)
```

---

calcTm-methods	<i>Calculate Tm (methods)</i>
----------------	-------------------------------

---

**Description**

Methods for calculating melting temperature (Tm) of nucleotide sequences

**Methods**

**x = "missing"** Handle empty function call  
**x = "NULL"** Handle empty function call  
**x = "character"** Handle character input  
**x = "ExpressionSet"** Handle input of an object of class ExpressionSet

**See Also**

[calcTm](#)

---

combineData	<i>Combine data</i>
-------------	---------------------

---

**Description**

Calculate trimmed and/or weighted means of groups of rows in a given data matrix.

**Usage**

```
combineData(x, y, w, ...)
```

**Arguments**

x	a numeric matrix containing the values whose trimmed and/or weighted mean is to be computed. Each column is treated independently.
y	a vector describing the discrete groups used to divide the elements of x. If y is missing then all elements of x are handled together.
w	a matrix of weights the same dimensions as x giving the weights to use for each element of x. If w is missing then all elements of x are given the same weight.
...	Arguments to be passed to methods (see <a href="#">combineData-methods</a> ):

- ‘trim’ the fraction (0 to 0.5, default is 0) of observations to be trimmed from each group of rows in x according to y.
- ‘na.rm’ logical; if ‘TRUE’, missing values are removed from x and y and z. If ‘FALSE’ any missing values cause an error.
- ‘element’ which element of AssayData to use for a given ExpressionSet input (default is “exprs”)
- ‘feature.group’ which element of featureData to use as binning variable (default is NULL). Can be a character matching varLabel or simply an integer indicating which feature to choose. See [getFeatures](#).
- ‘element.weight’ which element of AssayData to use for a given ExpressionSet input. If NULL (default), weighting is not performed.
- ‘feature.weight’ which element of featureData to use as weighting variable (default is NULL). Can be a character matching varLabel or simply an integer indicating which feature to choose. See [getFeatures](#).
- ‘samples’ which samples to use as data. Can be a vector of characters matching sample names, integers indicating which samples to choose, or a mixture of the two. If ‘NULL’ (default), all samples will be used.
- ‘dots’ other arguments not handled at this time.

### Value

Returns a matrix of combined numerical data, where each row represents the summary of a group of elements from the corresponding column in x.

### Note

Each column in data matrix treated separately.

### Author(s)

Reid F. Thompson (<rthomps@aecom.yu.edu>)

### See Also

[combineData-methods](#), [mean](#), [weighted.mean](#)

### Examples

```
#demo(pipeline,package="HELP")

x <- 1:100
combineData(x,w=x/100)
weighted.mean(x,w=x/100)

y <- sample(c("a","b","c",1:3),size=100,replace=TRUE)
combineData(cbind(x,x,2*x),y,trim=0.5)

#rm(x,y)
```



**Description**

Methods for calculating trimmed and/or weighted means of groups of rows in a given data matrix.

**Methods**

**x = "missing", y = "missing", w = "missing"** Handle empty function call

**x = "vector", y = "missing", w = "missing"** Handle partially empty function call. Reinterpret with default parameters instead of missing values.

**x = "vector", y = "missing", w = "vector"** Handle partially empty function call. Reinterpret with default parameters instead of missing values.

**x = "vector", y = "vector", w = "missing"** Handle partially empty function call. Reinterpret with default parameters instead of missing values.

**x = "vector", y = "vector", w = "vector"** Handle input of three vectors specifying data, grouping, and weighting information, respectively. Note that the data and weighting inputs are handled as matrices.

**x = "matrix", y = "vector", w = "missing"** Handle partially empty function call. Reinterpret with default parameters instead of missing values.

**x = "matrix", y = "vector", w = "matrix"** Handle input of one matrix, one vector, and one matrix specifying data, grouping, and weighting information, respectively.

**x = "ExpressionSet", y = "missing", w = "missing"** Handle input of an object of class `ExpressionSet`. Derive grouping and weighting data from `featureData` and `AssayDataElement`, respectively.

**x = "ExpressionSet", y = "vector", w = "missing"** Handle input of an object of class `ExpressionSet` and a vector specifying grouping information. Derive weighting data from `codeAssayDataElement`.

**Author(s)**

Reid F. Thompson (<rthomps@aeacom.yu.edu>)

**See Also**

[combineData](#)

---

 createWiggle

 Create wiggle track
 

---

### Description

Create and write a wiggle track (UCSC Genome Browser format) to flat file

### Usage

```
createWiggle(x, y, ...)
```

### Arguments

- |     |   |
|-----|---|
| x   | matrix of numerical data, where each column represents data for an individual wiggle track. x can also be of class "ExpressionSet".   |
| y   | an additional matrix of numerical data with columns corresponding to chr, start, and end, respectively.   |
| ... | Arguments to be passed to methods (see <a href="#">createWiggle-methods</a> ):  |
|     | ‘element’ which element of AssayData to use for a given ExpressionSet input (default is "exprs")  |
|     | ‘feature.chr’ which element of featureData to use as chromosomal information (default is "CHR"). Can be a character matching varLabel or simply an integer indicating which feature to choose.                      |
|     | ‘feature.start’ which element of featureData to use as start positions (default is "START"). Can be a character matching varLabel or simply an integer indicating which feature to choose.                          |
|     | ‘feature.stop’ which element of featureData to use as end positions (default is "STOP"). Can be a character matching varLabel or simply an integer indicating which feature to choose.                              |
|     | ‘samples’ which sample(s) to use as data. Can be a vector of characters matching sample names, integers indicating which samples to choose, or a mixture of the two. If ‘NULL’ (default), all samples will be used. |
|     | ‘colors’ vector of colors, indicates which colors to use for which wiggle track   |
|     | ‘file’ location of file to write wiggle track information; if "", wiggle track prints to the standard output connection: see <a href="#">cat</a> .  |
|     | ‘append’ logical; if ‘TRUE’, the output is appended to an existent wiggle track file. If ‘FALSE’ (default), a new file with a new header is created and any existing file of the same name is destroyed.            |
|     | ‘na.rm’ logical; if ‘TRUE’ (default), missing values are removed from data. If ‘FALSE’ any missing values cause an error  |
|     | ‘sep’ a string used to separate columns. Using ‘sep = "\t”’ (default) gives tab-delimited output.   |
|     | ‘dots’ other arguments to be passed to cat. See <a href="#">cat</a> .   |

**Author(s)**

Reid F. Thompson (<rthomps@aeocom.yu.edu>)

**References**

UCSC Genome Browser, <http://genome.ucsc.edu/goldenPath/help/customTrack.html>: Kent, W.J., Sugnet, C. W., Furey, T. S., Roskin, K.M., Pringle, T. H., Zahler, A. M., and Haussler, D. The Human Genome Browser at UCSC. *Genome Res.* **12**(6), 996-1006 (2002).

**See Also**

[write](#), [cat](#)

**Examples**

```
#demo(pipeline, package="HELP")

chr <- rep("chr1", 500)
start <- (1:500)*200
end <- start+199
data <- sample(5*(1:10000/10000)-2, size=500)
data <- cbind(data, abs(data), -1*data)
colnames(data) <- c("data", "abs", "opposite")
createWiggle(data, cbind(chr, start, end))

#rm(chr, start, end, data)
```

---

createWiggle-methods    *Create wiggle track (methods)*

---

**Description**

Methods for creating wiggle tracks

**Methods**

**x = "missing", y = "missing"** Handle empty function call

**x = "ExpressionSet", y = "missing"** Handle input of an object of class ExpressionSet. Derive features from FeatureData.

**x = "ExpressionSet", y = "matrix"** Handle input of an object of class ExpressionSet. Derive features from matrix input

**x = "vector", y = "matrix"** Handle vector input

**x = "matrix", y = "matrix"** Handle matrix input

**Author(s)**

Reid F. Thompson (<rthomps@aeocom.yu.edu>)

**See Also**[createWiggle](#)


---

exprs2	<i>Retrieve microarray data (for signal channel 2) from ExpressionSets</i>
--------	--

---

**Description**

Access (and/or assign) data for signal channel 2 in a given ExpressionSet object

**Usage**

```
exprs2(object)
exprs2(object) <- value
```

**Arguments**

object	Object of class ExpressionSet
value	Matrix with rows representing features and columns representing samples

**Value**

exprs2 returns a (usually large!) matrix of values

**Author(s)**

Reid F. Thompson (<rthomps@aeacom.yu.edu>)

**Examples**

```
#demo(pipeline, package="HELP")
```

---

exprs2-methods	<i>Retrieve microarray data (for signal channel 2) from ExpressionSets (methods)</i>
----------------	--

---

**Description**

Methods for accessing (and/or assigning) data for signal channel 2 in a given ExpressionSet object

**Methods**

**object = "missing"** Handle empty function call  
**object = "ExpressionSet"** Handle input of an object of class ExpressionSet  
**object = "ExpressionSet", value = "missing"** Handle empty function call  
**object = "ExpressionSet", value = "matrix"** Handle input of an object of class ExpressionSet

**See Also**[exprs2](#)

fuzzyMatches

*Fuzzy matching***Description**

Match and reinterpret a vector in terms of a second vector, essentially using the second vector as a key to interpret the first.

**Usage**

```
fuzzyMatches(x, y, ...)
```

**Arguments**

- x            vector, the values to be matched.
- y            vector, the values to be matched against.
- ...          Arguments to be passed to methods (see [getSamples-methods](#)):
  - ‘strict’ logical indicating whether or not to allow restrict matching. If ‘FALSE’, numerical indices in x can reference values in y. If ‘TRUE’ (default), only exact matches of values in x and y will be used.
  - ‘keep’ logical indicating whether or not to preserve non-matching values from x (when strict = FALSE). If ‘TRUE’ (default), all values in x will be returned with those that match values in y replaced by the corresponding values in y. If ‘FALSE’, non-matching values will be removed.
  - ‘alias’ logical indicating whether or not to return aliased values (default is ‘TRUE’). If ‘FALSE’, numerical indices will be returned with a value of nomatch for non-matching values in x.
  - ‘match.all’ character value, specifying a special value to be interpreted as a match for ALL values in y (default is “\*”). Any occurrence of match.all in x will be replaced by all values in y.
  - ‘nomatch’ numerical, specifying a value for non-matching values in codex when strict = FALSE, keep = TRUE, and alias = FALSE.
  - ‘na.rm’ a logical value indicating whether NA values should be stripped before the computation proceeds (default is ‘TRUE’). If ‘FALSE’ any missing values in x will cause an error and missing values in codey may cause unexpected behavior.
  - ‘dots’ other arguments not handled at this time.

## Details

This function employs multiple stages of matching between two vectors. First, the values in `x` are matched against `y` to find any exact matches. Next, numerical values in `x` are used to retrieve the corresponding positions in `y`. All unmatched values may be retained or dropped (depending on the value of `keep`), and a list of unique values is returned. Note that a value of `match.all` in `x` will be interpreted as a match for ALL values in `y`, and therefore replaced with the contents of `y`.

## Value

Returns a vector of unique values in `x`, that match values in `y` according to the parameters described above.

## Author(s)

Reid F. Thompson (<rthomps@aeocom.yu.edu>)

## See Also

[fuzzyMatches-methods](#), [match](#)

## Examples

```
a <- c(1, "four", "missing")
b <- c("one", "two", "three", "four")
fuzzyMatches(a, b)
fuzzyMatches(a, b, strict=FALSE)
fuzzyMatches(a, b, strict=FALSE, alias=FALSE)
fuzzyMatches(a, b, strict=FALSE, keep=FALSE)
```

---

fuzzyMatches-methods    *Fuzzy matching (methods)*

---

## Description

Methods for matching and reinterpreting a vector in terms of a second vector, essentially using the second vector as a key to interpreting the first.

## Methods

**x = "missing", y = "missing"** Handle empty function call

**x = "vector", y = "missing"** Handle empty function call

**x = "vector", y = "NULL"** Handle empty function call

**x = "vector", y = "vector"** Handle input of two vectors.

**x = "NULL", y = "vector"** Handle empty function call

**Author(s)**

Reid F. Thompson (<rthomps@aeacom.yu.edu>)

**See Also**

[fuzzyMatches](#)

---

getFeatures

*Get features (methods)*

---

**Description**

Fetch a subset of features from a given data structure

**Usage**

```
getFeatures(x, y, ...)
```

**Arguments**

x	the matrix of feature data to subset. If x is a vector it is interpreted as a matrix. x can also be of class "ExpressionSet" or an "AnnotatedDataFrame" object.
y	which feature(s) to use. Can be a vector of characters matching feature names, integers indicating which features to choose, or a mixture of the two. If not supplied (or if equivalent to "*" ), all features will be used.
...	other arguments passed are not handled at this time.

**Value**

Returns a matrix of values corresponding to a subset of features from the data structure supplied, where columns correspond to features. Function halts if no features to return.

**Author(s)**

Reid F. Thompson (<rthomps@aeacom.yu.edu>)

**See Also**

[getFeatures-methods](#)

**Examples**

```
data(sample.ExpressionSet)
df <- data.frame(x=1:500,y=501:1000, row.names=featureNames(sample.ExpressionSet))
featureData(sample.ExpressionSet) <- new("AnnotatedDataFrame", data=df, dimLabels=c("featureNames", ""))
getFeatures(sample.ExpressionSet, "y")[1:10]
```

---

getFeatures-methods     *Get features (methods)*

---

## Description

Methods for fetching a subset of features from a given data structure

## Methods

**x = "missing", y = "missing"** Handle empty function call

**x = "ExpressionSet", y = "missing"** Handle input of an object of class ExpressionSet. Select all feature data.

**x = "ExpressionSet", y = "NULL"** Handle input of an object of class ExpressionSet. Select all feature data.

**x = "ExpressionSet", y = "vector"** Handle input of an object of class ExpressionSet. Select a subset of features.

**x = "AnnotatedDataFrame", y = "missing"** Handle input of an AnnotatedDataFrame object. Select all feature data.

**x = "AnnotatedDataFrame", y = "NULL"** Handle input of an AnnotatedDataFrame object. Select all feature data.

**x = "AnnotatedDataFrame", y = "vector"** Handle input of an AnnotatedDataFrame object. Select a subset of features.

**x = "vector", y = "missing"** Handle input of a vector (interpreted as a matrix). Select all feature data

**x = "vector", y = "NULL"** Handle input of a vector (interpreted as a matrix). Select all feature data

**x = "vector", y = "vector"** Handle input of two vectors specifying feature data and feature subset information, respectively.

**x = "matrix", y = "vector"** Handle input of a matrix and a vector specifying feature data and feature subset information, respectively.

## Author(s)

Reid F. Thompson (<rthomps@aecon.yu.edu>)

## See Also

[getFeatures](#)



---

`getSamples`*Get samples*

---

**Description**

Fetch a subset of samples from a given data structure

**Usage**

```
getSamples(x, y, ...)
```

**Arguments**

- |                  |  |
|------------------|--|
| <code>x</code>   | the matrix of sample data to subset. If <code>x</code> is a vector it is interpreted as a matrix. <code>x</code> can also be of class "ExpressionSet".   |
| <code>y</code>   | which sample(s) to use as data. Can be a vector of characters matching sample names, integers indicating which samples to choose, or a mixture of the two. If not supplied, all samples will be used.  |
| <code>...</code> | Arguments to be passed to methods (see <a href="#">getSamples-methods</a> ):<br>‘element’ which element of AssayData to use for a given ExpressionSet input (default is "exprs")<br>‘order’ vector of characters, specifying on which column(s) to order the sample data. If ‘NULL’ (default), the data will be returned without ordering enforced.<br>‘dots’ other arguments passed are not handled at this time. |

**Value**

Returns a matrix of values corresponding to a subset of samples from the data supplied, where columns correspond to samples. Function halts if no samples to return.

**Author(s)**

Reid F. Thompson (<[rthomps@aeacom.yu.edu](mailto:rthomps@aeacom.yu.edu)>)

**See Also**

[getSamples-methods](#)

**Examples**

```
data(sample.ExpressionSet)
se.ABC <- getSamples(sample.ExpressionSet, c("A", "B", "C"), element="se.exprs")
se.ABC[1:10,]
```

---

getSamples-methods      *Get samples (methods)*

---

### Description

Methods for fetching subsets of samples from various data structures

### Methods

**x = "missing", y = "missing"** Handle empty function call

**x = "ExpressionSet", y = "missing"** Handle input of an object of class ExpressionSet. Select data for all samples.

**x = "ExpressionSet", y = "NULL"** Handle input of an object of class ExpressionSet. Select data for all samples.

**x = "ExpressionSet", y = "vector"** Handle input of an object of class ExpressionSet. Select data for a subset of samples.

**x = "vector", y = "missing"** Handle input of a vector (interpreted as a matrix). Select data for all samples.

**x = "vector", y = "NULL"** Handle input of a vector (interpreted as a matrix). Select data for all samples.

**x = "vector", y = "vector"** Handle input of two vectors specifying data and sample subset information, respectively.

**x = "matrix", y = "missing"** Handle input of a matrix. Select data for all samples.

**x = "matrix", y = "NULL"** Handle input of a matrix. Select data for all samples.

**x = "matrix", y = "vector"** Handle input of a matrix and a vector specifying data and sample subset information, respectively.

### Author(s)

Reid F. Thompson (<rthomps@aeacom.yu.edu>)

### See Also

[getSamples](#)

plotBins

*Plot bins***Description**

Plot densities of multiple bins of data, divided by a sliding window approach

**Usage**

```
plotBins(x, y, ...)
```

**Arguments**

**x** the vector of numerical data to be plotted. If **x** is a matrix it is interpreted as a vector. **x** can also be of class "ExpressionSet".

**y** an additional vector of numerical data to be used for binning. If **y** is a matrix it is interpreted as a vector. **y** can also be of class "ExpressionSet".

**...** Arguments to be passed to methods (see [plotBins-methods](#)):

- 'element' which element of AssayData to use for a given ExpressionSet input (default is "exprs")
- 'sample' which element of sampleNames to use as data (default is 1). Can be a character matching a sample name or simply an integer indicating which sample to choose. See [getSamples](#).
- 'feature' which element of featureData to use as binning variable (default is 1). Can be a character matching varLabel or simply an integer indicating which feature to choose. See [getFeatures](#).
- 'num.bins' number of bins (default is 10) used to divide the data
- 'num.steps' number of steps (default is 3) used to create bin offsets, resulting in bins of sliding windows
- 'mode' the binning mode to be used. This must be either "continuous" or "discrete". "continuous" mode will divide the data into density-dependent bins. "discrete" mode will divide the data uniformly by binning data values.
- 'show.avg' logical; if 'TRUE', plots overall density in addition to densities per bin. If 'FALSE' (default), overall density plot is omitted.
- 'main' an overall title for the plot: see [title](#).
- 'xlab' a title for the x axis: see [title](#).
- 'ylab' a title for the y axis: see [title](#).
- 'na.rm' logical; if 'TRUE' (default), missing values are removed from **x** and **y**. If 'FALSE' any missing values cause an error.
- 'dots' other arguments to be passed to plot. See [plot](#).

**Author(s)**

Reid F. Thompson (<rthomps@aeocom.yu.edu>)

**See Also**

[plotBins-methods](#), [density](#), [quantile](#)

**Examples**

```
#demo(pipeline,package="HELP")

x <- 1:1000
y <- sample(1:50,size=1000,replace=TRUE)
plotBins(x,y,show.avg=TRUE,main="Random binning data",xlab="1:1000")

#rm(x,y)
```

---

plotBins-methods

*Plot bins (methods)*

---

**Description**

Methods for plotting densities of multiple bins of data, divided by a sliding window approach

**Methods**

**x = "missing", y = "missing"** Handle empty function call

**x = "matrix", y = "missing"** Handle matrix input, reinterpret function call with two vector input if matrix has two columns, otherwise handle as empty function call

**x = "vector", y = "missing"** Handle empty function call

**x = "vector", y = "ExpressionSet"** Handle input of an object of class ExpressionSet. Derive binning information from this class but use data from a vector input.

**x = "vector", y = "vector"** Handle input of two vectors specifying data and binning information, respectively.

**x = "matrix", y = "matrix"** Handle matrix input, reinterpret function call with vector input

**x = "matrix", y = "vector"** Handle matrix input, reinterpret function call with vector input

**x = "ExpressionSet", y = "missing"** Handle input of an object of class ExpressionSet. Derive both data and binning information from a single object.

**x = "ExpressionSet", y = "vector"** Handle input of an object of class ExpressionSet. Derive data from this class but use binning information from a vector input.

**x = "ExpressionSet", y = "ExpressionSet"** Handle input of two objects of class ExpressionSet. Derive data and binning information from each one, respectively.

**Author(s)**

Reid F. Thompson (<rthomps@aecon.yu.edu>)

**See Also**

[plotBins](#)

---

plotChip

*Plot chip image*


---

### Description

Graphic display of spatially-linked data, particularly applicable for microarrays

### Usage

```
plotChip(x, y, z, ...)
```

### Arguments

- x            vector of numerical data determining x-coordinates of data on chip. x can also handle ExpressionSet (see [plotChip-methods](#) for more parameter details).
- y            vector of numerical data determining y-coordinates of data on chip
- z            the vector of numerical data to be plotted
- ...          Arguments to be passed to methods (see [plotChip-methods](#)):
  - ‘element’ which element of AssayData to use for a given ExpressionSet input (default is “exprs”)
  - ‘sample’ which element of sampleNames to use as data (default is 1). Can be a character matching a sample name or simply an integer indicating which sample to choose.
  - ‘feature.x’ which element of featureData to use as X coordinate (default is “X”). Can be a character matching varLabel or simply an integer indicating which feature to choose.
  - ‘feature.y’ which element of featureData to use as Y coordinate (default is “Y”). Can be a character matching varLabel or simply an integer indicating which feature to choose.
  - ‘na.rm’ logical; if ‘TRUE’, missing values are removed from x, y, and z. If ‘FALSE’ (default) any missing values cause an error.
  - ‘main’ an overall title for the plot: see [title](#).
  - ‘xlab’ a title for the x axis: see [title](#).
  - ‘ylab’ a title for the y axis: see [title](#).
  - ‘colors’ vector of colors specifying the color scheme to use (default is `rev(rainbow(n=20, start=0, end=1/3))`). Also determines the resolution of z such that the more colors that are used allow finer discrimination of differences in z.
  - ‘range’ vector of numerical data of length 2 (default is `c(NA, NA)`) specifying range used to color-code data in z
  - ‘nrows’ numerical input specifying the number of rows by which to divide the chip; default is ‘NULL’ which skips the division of data into blocks and results in individual spot resolution
  - ‘ncols’ numerical input specifying the number of columns by which to divide the chip; default is ‘NULL’ which skips the division of data into blocks and results in individual spot resolution
  - ‘dots’ other arguments to be passed to plot. See [plot](#).

**Author(s)**

Reid F. Thompson <rthompso@aecom.yu.edu>, Mark Reimers <mreimers@vcu.edu>

**See Also**

[plotChip-methods](#)

**Examples**

```
#demo(pipeline,package="HELP")

x <- rep(1:100,100)
y <- rep(1:100,each=100)
z <- x*(1001:11000/1000)
z <- z-mean(z)
z <- z*(sample(1:10000/10000)+1)
plotChip(x,y,z,main="Curved gradient",xlab="x",ylab="y")

plotChip(x,y,sample(1:10000,size=10000),colors=gray(0:50/50),range=c(1,10000),main="Random noise")

#rm(x,y,z)
```

---

plotChip-methods

*Plot chip image (methods)*

---

**Description**

Methods for graphic display of spatially-linked data, particularly applicable for microarrays

**Methods**

- x = "missing", y = "missing", z = "missing"** Handle empty function call
- x = "matrix", y = "missing", z = "missing"** Handle matrix input, extract information, and reinterpret function call with appropriate vectors
- x = "ExpressionSet", y = "missing", z = "missing"** Handle input of an object of class ExpressionSet. Derive both data and position information from a single object.
- x = "ExpressionSet", y = "vector", z = "missing"** Handle input of an object of class ExpressionSet. Derive position information from this object, but the corresponding data from vector input.
- x = "ExpressionSet", y = "ExpressionSet", z = "missing"** Handle input of two objects of class ExpressionSet. Derive position information and data from each one, respectively.
- x = "vector", y = "vector", z = "ExpressionSet"** Handle input of an object of class ExpressionSet. Derive data from this object, but the corresponding position information from vector input.
- x = "vector", y = "vector", z = "vector"** Handle input of three vectors. Derive X and Y positions and data from each one, respectively.

**Author(s)**

Reid F. Thompson (<rthomps@aecom.yu.edu>), Mark Reimers (<mreimers@vcu.edu>)

**See Also**

[plotChip](#)

---

plotFeature

*Plot feature versus two-color intensity*

---

**Description**

Graphical display of featureData (ex: fragment size) versus two-color signal intensity data

**Usage**

```
plotFeature(x, y, ...)
```

**Arguments**

- x matrix of numerical data to be plotted, with two columns (one for each signal channel). x can also be of class "ExpressionSet".
- y an additional vector of numerical data to be used for feature. If y is missing, the function will attempt to fill a value from featureData in x.
- ... Arguments to be passed to methods (see [plotFeature-methods](#)):
  - 'element.x' which element of AssayData to use (for signal channel 1) for a given ExpressionSet input (default is "exprs")
  - 'element.y' which element of AssayData to use (for signal channel 2) for a given ExpressionSet input (default is "exprs2")
  - 'sample' which element of sampleNames to use as data (default is 1). Can be a character matching a sample name or simply an integer indicating which sample to choose.
  - 'feature' which element of featureData to use as plotting feature (default is 1). Can be a character matching varLabel or simply an integer indicating which feature to choose.
  - 'feature.random' which element of featureData to use to identify random probes (default is "TYPE"). Can be a character matching varLabel or simply an integer indicating which feature to choose.
  - 'which.random' an integer vector specifying which rows of data correspond to random probes. if 'NULL' (default), the function will attempt to identify random probes using featureData.
  - 'random.flag' a character specifying the label for random probes in feature.random from featureData. Default is "RAND".
  - 'na.rm' logical; if 'TRUE' (default), missing values are removed from x. If 'FALSE' any missing values cause an error.

- ‘limit’ numerical input specifying the maximum number of points to plot (default is 10,000). if ‘NULL’, all points will be used.
- ‘cutoff’ a numerical input specifying the value below which signal intensities from channel 1 can be considered "failed" probes. If ‘NULL’ (default), the function will attempt to calculate a cutoff from random probe information.
- ‘cutoff2’ a numerical input specifying the value below which signal intensities from channel 2 can be considered "failed" probes. If ‘NULL’ (default), the function will attempt to calculate a cutoff from random probe information.
- ‘main’ an overall title for the plot: see [title](#).
- ‘xlab’ a title for the x axis (default is “Fragment size (bp)”): see [title](#).
- ‘ylab’ a title for the y axis for signal channel 1 (default is “log(MspI)”): see [title](#).
- ‘ylab2’ a title for the y axis for signal channel 2 (default is “log(HpaII)”): see [title](#).
- ‘cex’ numerical value (default is 0.2) giving the amount by which plotting text and symbols should be scaled relative to the default.
- ‘dots’ other arguments to be passed to plot. See [plot](#).

**Author(s)**

Reid F. Thompson (<rthomps@aeacom.yu.edu>)

**See Also**

[plotFeature-methods](#)

**Examples**

```
#demo(pipeline,package="HELP")

msp1 <- sample(8000:16000/1000, size=1000)
msp1 <- msp1[order(msp1)]
hpa2 <- sample(8000:16000/1000, size=1000)
hpa2 <- hpa2[order(hpa2)]
size <- sample((1:1000)*1.8+200, size=1000)
rand <- which.min(abs(msp1-quantile(msp1, 0.25)))
plotFeature(cbind(msp1, hpa2), size, which.random=(rand-20):(rand+20), main="Random")

#rm(msp1, hpa2, size, rand)
```

---

plotFeature-methods     *Plot feature versus two-color intensity (methods)*

---

**Description**

Methods for plotting featureData (ex: fragment size) versus two-color signal intensity data



## Methods

- x = "missing", y = "missing"** Handle empty function call
- x = "ExpressionSet", y = "missing"** Handle input of an object of class ExpressionSet. Derive both data and feature information from a single object.
- x = "ExpressionSet", y = "vector"** Handle input of an object of class ExpressionSet. Derive data from this class but use feature values from a vector input.
- x = "matrix", y = "vector"** Handle matrix input, where each of two columns in matrix represents data from one signal channel. Feature data is derived from values from a vector input.

---

plotPairs

*Plot tree-pairs*

---

## Description

Pairwise comparison of samples producing a matrix of scatterplots and a corresponding dendrogram

## Usage

```
plotPairs(x, ...)
```

## Arguments

- x** a numeric matrix, where each column represents a different sample. x can also be of class "ExpressionSet".
- ...** Arguments to be passed to methods (see [plotPairs-methods](#)):
  - 'element'** which element of AssayData to use for a given ExpressionSet input (default is "exprs")
  - 'samples'** which samples to use as data. Can be a vector of characters matching sample names, integers indicating which samples to choose, or a mixture of the two. If 'NULL' (default), all samples will be used.
  - 'scale'** logical value indicating whether sample branch lengths should be scaled by distance (default is 'TRUE')
  - 'groups'** logical value indicating whether the samples should be organized and color-coded by group (default is 'TRUE')
  - 'dist.method'** the distance measure to be used. This must be one of "euclidean" (default), "maximum", "manhattan", "canberra", "binary" or "minkowski". Any unambiguous substring can be given: see [dist](#) for more details.
  - 'hclust.method'** the agglomeration method to be used. This should be (an unambiguous abbreviation of) one of "ward" (default), "single", "complete", "average", "mcquitty", "median" or "centroid": see [hclust](#) for more details.
  - 'k'** an integer scalar or vector with the desired number of groups. If 'NULL' (default), grouping will rely instead on distance measurements: see [cutree](#) for more details.

‘cor.method’ the correlation algorithm to be used. This must be one of “pearson” (default), “kendall”, or “spearman”. Any unambiguous substring can be given: see [cor](#) for more details.

‘dots’ other arguments to be passed to pairs or dist. See [pairs](#), [dist](#).

**Author(s)**

Reid F. Thompson (<rthomps@aeacom.yu.edu>)

**See Also**

[plotPairs-methods](#), [dist](#), [hclust](#), [dendrogram](#), [cutree](#), [pairs](#)

**Examples**

```
#demo(pipeline,package="HELP")

x <- sample(1:10000,size=10000)
x <- cbind(x,x+5,x*sample((1000:2000)/1000,size=10000,replace=TRUE),sample(-1*(1:10000),size=10000))
colnames(x) <- c("x","x+5","spread","random")
plotPairs(x)

#rm(x)
```

---

plotPairs-methods      *Plot tree-pairs (methods)*

---

**Description**

Methods for pairwise comparison of samples producing a matrix of scatterplots and a corresponding dendrogram

**Methods**

**x = "missing"** Handle empty function call

**x = "matrix"** Handle matrix input

**x = "ExpressionSet"** Handle input of an object of class ExpressionSet. Derive data from AssayData.

**Author(s)**

Reid F. Thompson (<rthomps@aeacom.yu.edu>)

**See Also**

[plotPairs](#)

---

quantileNormalize	<i>Quantile normalization</i>
-------------------	-------------------------------

---

### Description

Apply quantile normalization to multiple bins of data, divided by a sliding window approach

### Usage

```
quantileNormalize(x, y, ...)
```

### Arguments

- |     |  |
|-----|--|
| x   | the vector of numerical data to be normalized. If x is a matrix it is interpreted as a vector. x can also be of class "ExpressionSet".   |
| y   | an additional vector of numerical data to be used for binning. If y is a matrix it is interpreted as a vector. y can also be of class "ExpressionSet".   |
| ... | Arguments to be passed to methods (see <a href="#">quantileNormalize-methods</a> ):<br>'element' which element of AssayData to use for a given ExpressionSet input (default is "exprs")<br>'sample' which element of sampleNames to use as data (default is 1). Can be a character matching a sample name or simply an integer indicating which sample to choose. See <a href="#">getSamples</a> .<br>'feature' which element of featureData to use as binning variable (default is 1). Can be a character matching varLabel or simply an integer indicating which feature to choose. See <a href="#">getFeatures</a> .<br>'num.bins' number of bins (default is 10) used to divide the data<br>'num.steps' number of steps (default is 3) used to create bin offsets, resulting in bins of sliding windows<br>'mode' the binning mode to be used. This must be either "continuous" (default) or "discrete". "continuous" mode will divide the data into density-dependent bins. "discrete" mode will divide the data uniformly by binning data values.<br>'type' an integer between 1 and 9 (default is 7) selecting one of the nine quantile algorithms: see <a href="#">quantile</a> .<br>'na.rm' logical; if 'TRUE', missing values are removed from x and y. If 'FALSE' any missing values cause an error.<br>'dots' other arguments to be passed to quantile. See <a href="#">quantile</a> . |

### Value

Returns a vector of normalized numerical data according to input parameters.

### Author(s)

Reid F. Thompson (<rthomps@aeocom.yu.edu>)

**See Also**

[quantileNormalize-methods](#), [quantile](#)

**Examples**

```
#demo(pipeline,package="HELP")

x <- rep(1:100,10)+10*rep(1:10,each=100)
y <- rep(1:20,each=50)
d <- density(quantileNormalize(x,y,num.bins=20,num.steps=1,mode="discrete"))
plot(density(x))
lines(d$x,d$y/3,col="red")

#rm(x,y,d)
```

---

quantileNormalize-methods

*Quantile normalization (methods)*

---

**Description**

Methods for applying quantile normalization to multiple bins of data, divided by a sliding window approach

**Methods**

- x = "missing", y = "missing"** Handle empty function call
- x = "matrix", y = "missing"** Handle matrix input, reinterpret function call with two vector input if matrix has two columns, otherwise handle as empty function call
- x = "vector", y = "missing"** Handle empty function call
- x = "vector", y = "ExpressionSet"** Handle input of an object of class ExpressionSet. Derive binning information from this class but use data from a vector input.
- x = "vector", y = "vector"** Handle input of two vectors specifying data and binning information, respectively.
- x = "ExpressionSet", y = "missing"** Handle input of an object of class ExpressionSet. Derive both data and binning information from a single object.
- x = "ExpressionSet", y = "vector"** Handle input of an object of class ExpressionSet. Derive data from this class but use binning information from a vector input.
- x = "ExpressionSet", y = "ExpressionSet"** Handle input of two objects of class ExpressionSet. Derive data and binning information from each one, respectively.

**Author(s)**

Reid F. Thompson (<rthomps@aecon.yu.edu>)

**See Also**[quantileNormalize](#)


---

readDesign	<i>Read NimbleGen design files</i>
------------	------------------------------------

---

**Description**

Function to extract array design information from corresponding files in the Nimblegen .ndf and .ngd formats.

**Usage**

```
readDesign(x, y, z, ...)
```

**Arguments**

- |     |  |
|-----|--|
| x   | path to the Nimblegen design file (.ndf). Each line of the file is interpreted as a single spot on the array design. If it does not contain an absolute path, the file name is relative to the current working directory, <a href="#">getwd()</a> . Tilde-expansion is performed where supported. Alternatively, x can be a readable connection (which will be opened for reading if necessary, and if so closed at the end of the function call). 'file' can also be a complete URL.  |
| y   | path to the Nimblegen gene descriptions file (.ngd). Each line of the file is interpreted as a single locus. If it does not contain an absolute path, the file name is relative to the current working directory, <a href="#">getwd()</a> . Tilde-expansion is performed where supported. Alternatively, y can be a readable connection (which will be opened for reading if necessary, and if so closed at the end of the function call). 'file' can also be a complete URL.  |
| z   | object in which to store design information from files. Can be an ExpressionSet, in which case design information will be stored in featureData.   |
| ... | Arguments to be passed to methods (see <a href="#">readDesign-methods</a> ):<br>'path' a character vector containing a single full path name to which filenames will be appended. If 'NULL', filenames (x and y) are treated as is.<br>'comment.char' character: a character vector of length one containing a single character or an empty string (default is "\#"). Use "" to turn off the interpretation of comments altogether.<br>'sep' the field separator character (default is "\t"). Values on each line of the file are separated by this character. If 'sep = "" the separator is "white space", that is one or more spaces, tabs, newlines or carriage returns.<br>'quote' the set of quoting characters (default is "\""). To disable quoting altogether, use quote = "". See <a href="#">scan</a> for the behavior on quotes embedded in quotes. Quoting is only considered for columns read as character, which is all of them unless 'colClasses' is specified.<br>'eSet' ExpressionSet input (default is new("ExpressionSet")) in which to store design information in featureData<br>'dots' other arguments to be passed to read.table. See <a href="#">read.table</a> . |

**Value**

Returns an ExpressionSet filled with featureData containing the following featureColumns:

<code>\option{SEQ_ID}</code>	a vector of characters with container IDs, linking each probe to a parent identifier
<code>\option{PROBE_ID}</code>	a vector of characters containing unique ID information for each probe
<code>\option{X}</code>	vector of numerical data determining x-coordinates of probe location on chip
<code>\option{Y}</code>	vector of numerical data determining y-coordinates of probe location on chip
<code>\option{TYPE}</code>	a vector of characters defining the type of probe, e.g. random background signals ("RAND") or usable data ("DATA").
<code>\option{CHR}</code>	a matrix of characters containing unique ID and chromosomal positions for each container
<code>\option{START}</code>	a matrix of characters containing unique ID and chromosomal positions for each container
<code>\option{STOP}</code>	a matrix of characters containing unique ID and chromosomal positions for each container
<code>\option{SIZE}</code>	a matrix of characters containing unique ID and chromosomal positions for each container
<code>\option{SEQUENCE}</code>	a vector of characters containing sequence information for each probe
<code>\option{WELL}</code>	a vector of characters containing multiplex well location for each probe (if present in design files)

**Author(s)**

Reid F. Thompson (<rthomps@aeacom.yu.edu>)

**See Also**

[readDesign-methods](#), [read.table](#)

**Examples**

```
#demo(pipeline, package="HELP")

chr <- rep("chr1", 500)
start <- (1:500)*200
stop <- start+199
x <- 1:500
seqids <- sample(1:50, size=500, replace=TRUE)
cat("#COMMENT\nSEQ_ID\tCHROMOSOME\tSTART\tSTOP\n", file="./read.design.test.ngd")
table.ngd <- cbind(seqids, chr, start, stop)
write.table(table.ngd, file="./read.design.test.ngd", append=TRUE, col.names=FALSE, row.names=FALSE, quote=FALSE)
cat("#COMMENT\nSEQ_ID\tX\tY\tPROBE_ID\tCONTAINER\tPROBE_SEQUENCE\tPROBE_DESIGN_ID\n", file="./read.design.test")
sequence <- rep("NNNNNNNN", 500)
table.ndf <- cbind(seqids, x, x, x, x, sequence, x)
```

```
write.table(table.ndf, file="./read.design.test.ndf", append=TRUE, col.names=FALSE, row.names=FALSE, quote=FALSE)
x <- readDesign("./read.design.test.ndf", "./read.design.test.ngd")
seqids[1:10]
pData(featureData(x))$"SEQ_ID"[1:10]

#rm(table.ngd, table.ndf, chr, start, stop, x, seqids, sequence)
#file.remove("./read.design.test.ngd")
#file.remove("./read.design.test.ndf")
```

---

readDesign-methods      *Read NimbleGen design files (methods)*

---

## Description

Methods for extracting array design information from corresponding files in the Nimblegen .ndf and .ngd formats.

## Methods

- x = "missing", y = "missing", z = "missing"** Handle empty function call
- x = "vector", y = "missing", z = "missing"** Handle single vector input. If two values specified in vector, reinterpret function call with two character inputs. Otherwise, handle as empty function call.
- x = "vector", y = "vector", z = "missing"** Handle two vector input. If vectors of unit length, reinterpret function call with two character inputs. Otherwise, handle as improper function call.
- x = "character", y = "character", z = "ExpressionSet"** Handle two character vector inputs, each specifying a filename to use when reading design information. Design information will be written to an ExpressionSet.
- x = "character", y = "character", z = "character"** Handle two character vector inputs, each specifying a filename to use when reading design information. Design information will be written to a database.

## Author(s)

Reid F. Thompson (<rtthomps@aecon.yu.edu>)

## See Also

[readDesign](#)

---

 readPairs

*Read Nimblegen .pair files*


---

## Description

Function to extract data from corresponding files in the Nimblegen .pair format.

## Usage

```
readPairs(x, y, z, ...)
```

## Arguments

- x the name of the file containing data from signal channel 1. Each line of the file is interpreted as a single data point. If it does not contain an absolute path, the file name is relative to the current working directory, `getwd()`. Tilde-expansion is performed where supported. Alternatively, x can be a readable connection (which will be opened for reading if necessary, and if so closed at the end of the function call). x can also be a complete URL.
- y the name of the file containing data from signal channel 1. Each line of the file is interpreted as a single data point. If it does not contain an absolute path, the file name is relative to the current working directory, `getwd()`. Tilde-expansion is performed where supported. Alternatively, y can be a readable connection (which will be opened for reading if necessary, and if so closed at the end of the function call). y can also be a complete URL.
- z object in which to store pair information from files. Can be an `ExpressionSet`, in which case pair data will be stored in `featureData`.
- ... Arguments to be passed to methods (see [readPairs-methods](#)):
  - 'name' character input specifying a sample name to assign to the data from specified pair files. If 'NULL' (default), a value will be extracted from the file-name specified in x.
  - 'element.x' which element of `AssayData` (default is "exprs") in which to store signal channel 1 data.
  - 'element.y' which element of `AssayData` (default is "exprs2") in which to store signal channel 2 data.
  - 'match.probes' logical specifying whether to match data from pair files by "PROBE\_ID" with any pre-existing data. if 'TRUE' (default), order of values will be rearranged so long as there are already "PROBE\_ID"s specified in `featureData`.
  - 'path' a character vector containing a single full path name to which filenames will be appended. If 'NULL', filenames (x and y) are treated as is.
  - 'comment.char' character: a character vector of length one containing a single character or an empty string (default is "\#"). Use "" to turn off the interpretation of comments altogether.



- ‘sep’ the field separator character (default is “\t”). Values on each line of the file are separated by this character. If ‘sep = “ ”’ the separator is "white space", that is one or more spaces, tabs, newlines or carriage returns.
- ‘quote’ the set of quoting characters (default is “\””). To disable quoting altogether, use quote = “ ”. See [scan](#) for the behaviour on quotes embedded in quotes. Quoting is only considered for columns read as character, which is all of them unless ‘colClasses’ is specified.
- ‘eSet’ ExpressionSet input (default is new("ExpressionSet")) in which to store pair information in assayData
- ‘verbose’ logical; if ‘TRUE’ (default) progress will be output to screen. If ‘FALSE’, no output will be displayed.
- ‘dots’ other arguments to be passed to read.table. See [read.table](#).

### Value

Returns an ExpressionSet filled with assayData containing matrices of data from both signal channels.

and featureData containing the following featureColumns:

`\option{SEQ_ID}`

a vector of characters with container IDs, linking each probe to a parent identifier

`\option{PROBE_ID}`

a vector of characters containing unique ID information for each probe

and phenoData containing the following sampleColumns:

`\option{CHIPS}` a vector of characters with .pair file locations for signal channel 1 data

`\option{CHIPS2}`

a vector of characters with .pair file locations for signal channel 2 data

### Author(s)

Reid F. Thompson (<rthomps@aeocom.yu.edu>)

### See Also

[readPairs-methods](#), [read.table](#)

### Examples

```
#demo(pipeline, package="HELP")

x <- 1:500
y <- rev(x)
data <- sample(8000:10000/1000, size=500)
seqids <- sample(1:50, size=500, replace=TRUE)
cat("#COMMENT\nSEQ_ID\tPROBE_ID\tX\tY\tPM\n", file="./read.pair.test.1")
table.1 <- cbind(seqids, y, x, x, data)
write.table(table.1, file="./read.pair.test.1", append=TRUE, col.names=FALSE, row.names=FALSE, quote=FALSE, sep="\t")
cat("#COMMENT\nSEQ_ID\tPROBE_ID\tX\tY\tPM\n", file="./read.pair.test.2")
```

```

table.2 <- cbind(seqids,y,x,x,rev(data))
write.table(table.2,file="./read.pair.test.2",append=TRUE,col.names=FALSE,row.names=FALSE,quote=FALSE,sep="\t")
x <- readPairs("./read.pair.test.1","./read.pair.test.2")
c(seqids[1],y[1],data[1],rev(data)[1])
pData(featureData(x))$"SEQ_ID"[1]
pData(featureData(x))$"PROBE_ID"[1]
assayDataElement(x, "exprs")[1]
assayDataElement(x, "exprs2")[1]

#rm(table.1,table.2,x,y,data,seqids)
#file.remove("./read.pair.test.1")
#file.remove("./read.pair.test.2")

```

---

readPairs-methods

*Read Nimblegen .pair files (methods)*


---

## Description

Methods for extracting data from corresponding files in the Nimblegen .pair format.

## Methods

- x = "missing", y = "missing", z = "missing"** Handle empty function call
- x = "vector", y = "missing", z = "missing"** Handle single vector input. If two values specified in vector, reinterpret function call with two character inputs. Otherwise, handle as empty function call.
- x = "vector", y = "vector", z = "missing"** Handle two vector input. If vectors of unit length, reinterpret function call with two character inputs. Otherwise, handle as improper function call.
- x = "character", y = "character", z = "ExpressionSet"** Handle two character vector inputs, each specifying a filename to use when reading pair information. Pair data will be written to an ExpressionSet object.
- x = "character", y = "character", z = "character"** Handle two character vector inputs, each specifying a filename to use when reading pair information. Pair data will be written to a database.

## Author(s)

Reid F. Thompson (<rthomps@aeocom.yu.edu>)

## See Also

[readPairs](#)

---

readSampleKey	<i>Read sample key</i>
---------------	------------------------

---

**Description**

Function to extract sample key data from a file and link chip ID information with aliases if they exist.

**Usage**

```
readSampleKey(file = NULL, chips = NULL, comment.char = "#", sep = "\\t")
```

**Arguments**

file	the name of the file containing sample key information. Each line of the file is interpreted as a single chip-to-sample map. If it does not contain an absolute path, the file name is relative to the current working directory, <code>getwd()</code> . Tilde-expansion is performed where supported. Alternatively, file can be a readable connection (which will be opened for reading if necessary, and if so closed at the end of the function call). 'file' can also be a complete URL.
chips	a character vector specifying a specific chip ID lookup in the sample key, for which the function will return the appropriate sample aliases
comment.char	character: a character vector of length one containing a single character or an empty string. Use <code>""</code> to turn off the interpretation of comments altogether.
sep	the field separator character. Values on each line of the file are separated by this character. If <code>sep = ""</code> the separator is "white space", that is one or more spaces, tabs, newlines or carriage returns.

**Value**

Returns a character vector of sample alias information corresponding to the chips present in the sample key or a subset thereof, specified by the chips input.

**Author(s)**

Reid F. Thompson (<[rthomps@aeacom.yu.edu](mailto:rthomps@aeacom.yu.edu)>)

**See Also**

[read.table](#)

**Examples**

```
#demo(pipeline,package="HELP")

cat("#COMMENT\nCHIP_ID\tSAMPLE\n",file="./sample.key.txt")
write.table(cbind(1:10,1001:1010),file="./sample.key.txt",append=TRUE,col.names=FALSE,row.names=FALSE,quote=FALSE)
readSampleKey(file="./sample.key.txt")
readSampleKey(file="./sample.key.txt",chips=c(7:10,"NA1","NA2"))

#file.remove("./sample.key.txt")
```

# Index

- \* **IO**
    - readDesign, 29
    - readPairs, 32
    - readSampleKey, 35
  - \* **arith**
    - calcGC, 3
    - calcTm, 6
    - combineData, 7
  - \* **array**
    - calcPrototype, 4
    - quantileNormalize, 27
  - \* **attribute**
    - getFeatures, 15
    - getSamples, 17
  - \* **datasets**
    - base.stacking.thermodynamics, 2
  - \* **file**
    - readDesign, 29
    - readPairs, 32
    - readSampleKey, 35
  - \* **hplot**
    - plotBins, 19
    - plotChip, 21
    - plotFeature, 23
    - plotPairs, 25
  - \* **manip**
    - exprs2, 12
    - fuzzyMatches, 13
  - \* **methods**
    - calcGC-methods, 4
    - calcPrototype-methods, 5
    - calcTm-methods, 7
    - combineData-methods, 9
    - createWiggle-methods, 11
    - exprs2-methods, 12
    - fuzzyMatches-methods, 14
    - getFeatures-methods, 16
    - getSamples-methods, 18
    - plotBins-methods, 20
    - plotChip-methods, 22
    - plotFeature-methods, 24
    - plotPairs-methods, 26
    - quantileNormalize-methods, 28
    - readDesign-methods, 31
    - readPairs-methods, 34
  - \* **print**
    - createWiggle, 10
  - \* **utilities**
    - calcGC, 3
    - calcTm, 6
- base.stacking.thermodynamics, 2, 6
- calcGC, 3, 4, 6
- calcGC, character-method (calcGC-methods), 4
- calcGC, ExpressionSet-method (calcGC-methods), 4
- calcGC, missing-method (calcGC-methods), 4
- calcGC, NULL-method (calcGC-methods), 4
- calcGC-methods, 4
- calcPrototype, 4, 5
- calcPrototype, ExpressionSet-method (calcPrototype-methods), 5
- calcPrototype, matrix-method (calcPrototype-methods), 5
- calcPrototype, missing-method (calcPrototype-methods), 5
- calcPrototype, vector-method (calcPrototype-methods), 5
- calcPrototype-methods, 5
- calcTm, 3, 6, 7
- calcTm, character-method (calcTm-methods), 7
- calcTm, ExpressionSet-method (calcTm-methods), 7
- calcTm, missing-method (calcTm-methods), 7

- calcTm, NULL-method (calcTm-methods), 7
- calcTm-methods, 7
- cat, 10, 11
- combineData, 7, 9
- combineData, ExpressionSet, missing, missing-method (combineData-methods), 9
- combineData, ExpressionSet, vector, missing-method (combineData-methods), 9
- combineData, matrix, vector, matrix-method (combineData-methods), 9
- combineData, matrix, vector, missing-method (combineData-methods), 9
- combineData, missing, missing, missing-method (combineData-methods), 9
- combineData, vector, missing, missing-method (combineData-methods), 9
- combineData, vector, missing, vector-method (combineData-methods), 9
- combineData, vector, vector, missing-method (combineData-methods), 9
- combineData, vector, vector, vector-method (combineData-methods), 9
- combineData-methods, 9
- cor, 26
- createWiggle, 10, 12
- createWiggle, ExpressionSet, matrix-method (createWiggle-methods), 11
- createWiggle, ExpressionSet, missing-method (createWiggle-methods), 11
- createWiggle, matrix, matrix-method (createWiggle-methods), 11
- createWiggle, missing, missing-method (createWiggle-methods), 11
- createWiggle, vector, matrix-method (createWiggle-methods), 11
- createWiggle-methods, 11
- cutree, 25, 26
  
- dendrogram, 26
- density, 20
- dist, 25, 26
  
- exprs2, 12, 13
- exprs2, ExpressionSet-method (exprs2-methods), 12
- exprs2, missing-method (exprs2-methods), 12
- exprs2-methods, 12
- exprs2<- (exprs2), 12
- exprs2<-, ExpressionSet, matrix-method (exprs2-methods), 12
- exprs2<-, ExpressionSet, missing-method (exprs2-methods), 12
- fuzzyMatches, 13, 15
- fuzzyMatches, missing, missing-method (fuzzyMatches-methods), 14
- fuzzyMatches, NULL, vector-method (fuzzyMatches-methods), 14
- fuzzyMatches, vector, missing-method (fuzzyMatches-methods), 14
- fuzzyMatches, vector, NULL-method (fuzzyMatches-methods), 14
- fuzzyMatches, vector, vector-method (fuzzyMatches-methods), 14
- fuzzyMatches-methods, 14
- getFeatures, 8, 15, 16, 19, 27
- getFeatures, AnnotatedDataFrame, missing-method (getFeatures-methods), 16
- getFeatures, AnnotatedDataFrame, NULL-method (getFeatures-methods), 16
- getFeatures, AnnotatedDataFrame, vector-method (getFeatures-methods), 16
- getFeatures, ExpressionSet, missing-method (getFeatures-methods), 16
- getFeatures, ExpressionSet, NULL-method (getFeatures-methods), 16
- getFeatures, ExpressionSet, vector-method (getFeatures-methods), 16
- getFeatures, matrix, vector-method (getFeatures-methods), 16
- getFeatures, missing, missing-method (getFeatures-methods), 16
- getFeatures, vector, missing-method (getFeatures-methods), 16
- getFeatures, vector, NULL-method (getFeatures-methods), 16
- getFeatures, vector, vector-method (getFeatures-methods), 16
- getFeatures-methods, 16
- getSamples, 17, 18, 19, 27
- getSamples, ExpressionSet, missing-method (getSamples-methods), 18
- getSamples, ExpressionSet, NULL-method (getSamples-methods), 18
- getSamples, ExpressionSet, vector-method (getSamples-methods), 18

- getSamples,matrix,missing-method  
(getSamples-methods), 18
- getSamples,matrix,NULL-method  
(getSamples-methods), 18
- getSamples,matrix,vector-method  
(getSamples-methods), 18
- getSamples,missing,missing-method  
(getSamples-methods), 18
- getSamples,vector,missing-method  
(getSamples-methods), 18
- getSamples,vector,NULL-method  
(getSamples-methods), 18
- getSamples,vector,vector-method  
(getSamples-methods), 18
- getSamples-methods, 18
- getwd, 29, 32, 35
  
- hclust, 25, 26
  
- match, 14
- mean, 4, 5, 8
  
- pairs, 26
- plot, 19, 21, 24
- plotBins, 19, 20
- plotBins,ExpressionSet,ExpressionSet-method  
(plotBins-methods), 20
- plotBins,ExpressionSet,missing-method  
(plotBins-methods), 20
- plotBins,ExpressionSet,vector-method  
(plotBins-methods), 20
- plotBins,matrix,missing-method  
(plotBins-methods), 20
- plotBins,missing,missing-method  
(plotBins-methods), 20
- plotBins,vector,ExpressionSet-method  
(plotBins-methods), 20
- plotBins,vector,missing-method  
(plotBins-methods), 20
- plotBins,vector,vector-method  
(plotBins-methods), 20
- plotBins-methods, 20
- plotChip, 21, 23
- plotChip,ExpressionSet,ExpressionSet,missing-method  
(plotChip-methods), 22
- plotChip,ExpressionSet,missing,missing-method  
(plotChip-methods), 22
- plotChip,ExpressionSet,vector,missing-method  
(plotChip-methods), 22
- plotChip,matrix,missing,missing-method  
(plotChip-methods), 22
- plotChip,missing,missing,missing-method  
(plotChip-methods), 22
- plotChip,vector,vector,ExpressionSet-method  
(plotChip-methods), 22
- plotChip,vector,vector,vector-method  
(plotChip-methods), 22
- plotChip-methods, 22
- plotFeature, 23
- plotFeature,ExpressionSet,missing-method  
(plotFeature-methods), 24
- plotFeature,ExpressionSet,vector-method  
(plotFeature-methods), 24
- plotFeature,matrix,vector-method  
(plotFeature-methods), 24
- plotFeature,missing,missing-method  
(plotFeature-methods), 24
- plotFeature-methods, 24
- plotPairs, 25, 26
- plotPairs,ExpressionSet-method  
(plotPairs-methods), 26
- plotPairs,matrix-method  
(plotPairs-methods), 26
- plotPairs,missing-method  
(plotPairs-methods), 26
- plotPairs-methods, 26
  
- quantile, 20, 27, 28
- quantileNormalize, 27, 29
- quantileNormalize,ExpressionSet,ExpressionSet-method  
(quantileNormalize-methods), 28
- quantileNormalize,ExpressionSet,missing-method  
(quantileNormalize-methods), 28
- quantileNormalize,ExpressionSet,vector-method  
(quantileNormalize-methods), 28
- quantileNormalize,matrix,missing-method  
(quantileNormalize-methods), 28
- quantileNormalize,missing,missing-method  
(quantileNormalize-methods), 28
- quantileNormalize,vector,ExpressionSet-method  
(quantileNormalize-methods), 28
- quantileNormalize,vector,missing-method  
(quantileNormalize-methods), 28
- quantileNormalize,vector,vector-method  
(quantileNormalize-methods), 28
- quantileNormalize-methods, 28
  
- read.table, 29, 30, 33, 35

readDesign, [29](#), [31](#)  
readDesign, character, character, character-method  
    (readDesign-methods), [31](#)  
readDesign, character, character, ExpressionSet-method  
    (readDesign-methods), [31](#)  
readDesign, missing, missing, missing-method  
    (readDesign-methods), [31](#)  
readDesign, vector, missing, missing-method  
    (readDesign-methods), [31](#)  
readDesign, vector, vector, missing-method  
    (readDesign-methods), [31](#)  
readDesign-methods, [31](#)  
readPairs, [32](#), [34](#)  
readPairs, character, character, character-method  
    (readPairs-methods), [34](#)  
readPairs, character, character, ExpressionSet-method  
    (readPairs-methods), [34](#)  
readPairs, missing, missing, missing-method  
    (readPairs-methods), [34](#)  
readPairs, vector, missing, missing-method  
    (readPairs-methods), [34](#)  
readPairs, vector, vector, missing-method  
    (readPairs-methods), [34](#)  
readPairs-methods, [34](#)  
readSampleKey, [35](#)  
  
scan, [29](#), [33](#)  
  
title, [19](#), [21](#), [24](#)  
  
weighted.mean, [8](#)  
write, [11](#)