

# snapCGH

April 19, 2010

---

cbind

*Combine SegList Objects*

---

## Description

Combine a series of `SegList` objects.

## Usage

```
## S3 method for class 'SegList':  
cbind(..., deparse.level=1)
```

## Arguments

```
...           SegList objects  
deparse.level  
              not currently used, see cbind in the base package
```

## Details

`cbind` combines data objects assuming the same gene lists but different arrays.

For `cbind`, the matrices of expression data from the individual objects are cbinded. The `data.frames` of target information, if they exist, are rbinded. The combined data object will preserve any additional components or attributes found in the first object to be combined. It is not recommend to use the `is rbind` function for the `SegList` object. This is because it would require `SegLists` with mutually exclusive chromosomes or would result in combining multiple different segmentations for the same chromosome, which is pointless. If `rbind` is required perform it on an `MAList` and then segment it. It is currently only included as an internal function called within other library functions.

## Value

An `SegList` object holding data from all the arrays and all genes from the individual objects.

## Author(s)

Gordon Smyth, modified by Mike Smith for `SegList` object

## See Also

[cbind](#) in the base package.

---

`chrominfo.Mb`*Basic Chromosomal Information for UCSC Human Genome Assembly  
July 2003 freeze*

---

**Description**

This dataset contains basic chromosomal information for UCSC Human Genome Assembly July 2003 freeze.

**Usage**`chrominfo.basepair`**Format**

A data frame with 24 observations on the following 3 variables.

**chrom** Chromosomal index, X is coded as 23 and Y as 24.

**length** Length of each chromosome in megabases.

**centromere** Location of the centromere on the chromosome (Mb).

**Details**

This file is used for many plotting functions. The centromeric location is approximately estimated by taking mid-point between the last fish-mapped clone on the p-arm and the first fish-mapped clone on the q-arm using relevant UCSC freeze. For an alternative freeze, one needs to manually create a 3-column file of the format described above.

**Source**

<http://genome.ucsc.edu/cgi-bin/hgText>

---

`compareSegmentations`*Function for comparing segmentation methods to a known truth*

---

**Description**

This function takes a SegList and compares the breakpoints indicated in other SegLists with this original one.

**Usage**`compareSegmentations(TrueSeg, offset = 0, ...)`

**Arguments**

TrueSeg	An object of class <code>SegList</code> which is scored against. Normally the output from <code>simulateData</code> .
offset	Integer value between 0 and 2 specifying how close (in number of clones) to a true breakpoint the segmentation method must be before it is scored.
...	One or more objects of class <code>SegList</code> . These are compared to TrueSeg.

**Value**

The method returns a list containing two matrices. The first of these,  $\$TPR$ , contains the true positive rate, whilst the second,  $\$FDR$ , holds the false discovery rate. Both of these matrices are arranged such that a row represents a segmentation method and each column is an array.

**Author(s)**

John Marioni and Mike Smith

---

<code>convert.output</code>	<i>Converts the output from the simulation to a format which can be used by segmentation schemes available within R</i>
-----------------------------	-------------------------------------------------------------------------------------------------------------------------

---

**Description**

This function converts the output obtained by applying our simulation scheme into a format that can be used (either directly or indirectly) as the input to various segmentation schemes available within R. Additionally, we are in the process of submitting a library to CRAN which will enable the user to apply a number of the segmentation schemes available within R to datasets which have the same structure as that generated by this function.

**Usage**

```
convert.output(input)
```

**Arguments**

input	The output obtained upon applying the <code>sim.structure</code> function
-------	---------------------------------------------------------------------------

**Details**

This function outputs an object which is similar in structure/format to an RG or MA object used in Limma.

**Value**

This function outputs a list with entries

M	A matrix containing the $\log_2$ ratios
genes	A matrix containing the simulated midpoints and the chromosome which forms the template upon which the simulation is based.

**Author(s)**

Michael Smith, John Marioni

**Examples**

```
## The function is currently defined as
function(input){
  holder <- list()
  for (i in 1:length(input)){
    holder[[i]] <- list()
    for(i in 1:length(input)){
      holder[[i]]$genes <- matrix(NA, nrow = length(input[[i]]$clones$mid.point),
                                ncol = 2)
    }
    for(i in 1:length(input)){
      holder[[i]]$M <- as.matrix(input[[i]]$datamatrix)
      holder[[i]]$genes[,1] <- input[[i]]$clones$mid.point
      holder[[i]]$genes[,2] <- rep(input[[i]]$chrom,length(input[[i]]$clones$mid.point))
      colnames(holder[[i]]$genes) <- c("kb", "Chrom")
      holder[[i]] <- new("aCGHList", holder[[i]])
    }
  }
  holder
}
```

---

dimnames

*Retrieve the Dimension Names of an RGList, MAList or SegList Object*

---

**Description**

Retrieve the dimension names of a microarray data object.

**Usage**

```
## S3 method for class 'SegList':
dimnames(x)
```

**Arguments**

x                    An object of class SegList

**Details**

The dimension names of an microarray object are the same as those of the most important matrix component of that object.

A consequence is that `rownames` and `colnames` will work as expected.

**Value**

Either `NULL` or a list of length 2. If a list, its components are either `'NULL'` or a character vector the length of the appropriate dimension of `x`.

**Author(s)**

Gordon Smyth, edited by Mike Smith

**See Also**

[dimnames](#) in the base package.

---

dim

*Retrieve the Dimensions of an RGList, MAList or SegList Object*

---

**Description**

Retrieve the number of rows (genes) and columns (arrays) for an RGList, MAList or SegList object.

**Usage**

```
## S3 method for class 'SegList':  
dim(x)  
## S3 method for class 'SegList':  
length(x)
```

**Arguments**

x                    an object of class RGList, MAList or SegList

**Details**

Microarray data objects share many analogies with ordinary matrices in which the rows correspond to spots or genes and the columns to arrays. These methods allow one to extract the size of microarray data objects in the same way that one would do for ordinary matrices.

A consequence is that row and column commands `nrow(x)`, `ncol(x)` and so on also work.

**Value**

Numeric vector of length 2. The first element is the number of rows (genes) and the second is the number of columns (arrays).

**Author(s)**

Gordon Smyth, modified by Mike Smith for SegList object

**See Also**

[dim](#) in the base package.

**Examples**

```
M <- A <- matrix(11:14, 4, 2)
rownames(M) <- rownames(A) <- c("a", "b", "c", "d")
colnames(M) <- colnames(A) <- c("A1", "A2")
MA <- new("MAList", list(M=M, A=A))
dim(M)
ncol(M)
nrow(M)
length(M)
```

---

filterClones	<i>Filter clones from sample</i>
--------------	----------------------------------

---

**Description**

Function for filtering clones via a user defined function.

**Usage**

```
filterClones(MA, filterFunc, ...)
```

**Arguments**

MA	An object of class <code>MAList</code>
filterFunc	A user specified function that accepts an <code>MAList</code> and returns the indices of the clones to be removed.
...	Additional arguments to be passed to the filter function.

**Details**

Any clones identified by the filter function are turned into NA's. These are then removed or imputed within the `processCGH` function.

**Author(s)**

Mike Smith

---

findBreakPoints	<i>Returns the start and end of segments.</i>
-----------------	-----------------------------------------------

---

**Description**

Function to returns the start and end of segments when given a `SegList` and an array. Currently only used within the `plotSegmentedGenome` function.

**Usage**

```
findBreakPoints(seg, array)
```

**Arguments**

seg	An object of class <code>SegList</code> .
array	Numeric value corresponding to a column in seg.

**Author(s)**

Mike Smith

---

find.param.five      *Yields the output in a model with five underlying states*

---

**Description**

This function is a workhorse of the process.data function. It outputs state means/variances and transitions matrices in the model with five states.

**Usage**

```
find.param.five(output.optim, var.fixed)
```

**Arguments**

output.optim	output of the optimisation with 5 underlying states
var.fixed	Logical variable - TRUE if you want to fix the variance to be the same across states

**Value**

Outputs the mean/variance, transition matrix, maximised likelihood and convergence information

**Author(s)**

John Marioni

---

find.param.four      *Yields output when there are 4 underlying states*

---

**Description**

This function provides the output (means/variances, transition matrix, likelihood) when the heterogeneous HMM is fitted with four underlying states. It is a workhorse of the process.data function.

**Usage**

```
find.param.four(output.optim, var.fixed)
```

**Arguments**

- `output.optim` The output from fitting a heterogeneous HMM when there are four underlying states
- `var.fixed` Logical variable - TRUE if you want the variance to be tied across states. Defaults to FALSE

**Value**

This function outputs the state means/variances, transition matrices, rate parameters, maximised likelihood and convergence information provided by fitting a heterogeneous HMM with four underlying states.

**Author(s)**

John Marioni

---

`find.param.one`      *Yields output when there is 1 underlying states*

---

**Description**

This function provides the output (means/variances, transition matrix, likelihood) when the heterogeneous HMM is fitted with one underlying state. It is a workhorse of the `process.data` function.

**Usage**

```
find.param.one(output.optim)
```

**Arguments**

- `output.optim` The output from fitting a heterogeneous HMM when there is one underlying states

**Value**

This function outputs the state means/variances, transition matrices, rate parameters, maximised likelihood and convergence information provided by fitting a heterogeneous HMM with one underlying state.

**Author(s)**

John Marioni



---

find.param.three     *Yields output when there are 3 underlying states*

---

**Description**

This function provides the output (means/variances, transition matrix, likelihood) when the heterogeneous HMM is fitted with three underlying states. It is a workhorse of the process.data function.

**Usage**

```
find.param.three(output.optim, var.fixed)
```

**Arguments**

`output.optim` The output from fitting a heterogeneous HMM when there are four underlying states

`var.fixed` Logical variable - TRUE if you want the variances to be tied across states. Defaults to FALSE

**Value**

This function outputs the state means/variances, transition matrices, rate parameters, maximised likelihood and convergence information provided by fitting a heterogeneous HMM with three underlying states.

**Author(s)**

John Marioni

---

find.param.two     *Yields output when there are 2 underlying states*

---

**Description**

This function provides the output (means/variances, transition matrix, likelihood) when the heterogeneous HMM is fitted with two underlying states. It is a workhorse of the process.data function.

**Usage**

```
find.param.two(output.optim, var.fixed)
```

**Arguments**

`output.optim` The output from fitting a heterogeneous HMM when there are two underlying states

`var.fixed` Logical variable - TRUE if you want to tie the variance across states. Defaults to FALSE

**Value**

This function outputs the state means/variances, transition matrices, rate parameters, maximised likelihood and convergence information provided by fitting a heterogeneous HMM with two underlying states.

**Author(s)**

John Marioni

---

fit.model	<i>Fitting a heterogeneous HMM to the log2 ratios on a particular chromosome.</i>
-----------	-----------------------------------------------------------------------------------

---

**Description**

This function fits five homogeneous HMMs to the log2 ratios on a particular chromosome. It then uses either the AIC or BIC to determine which of the five models is optimal before using a scaled version of the Viterbi algorithm to assign clones to states with the same underlying copy number.

**Usage**

```
fit.model(sample, chrom, dat, datainfo = clones.info, useCloneDists = TRUE, covariates = TRUE, aic = TRUE, bic = FALSE, delta = 1, var.fixed=FALSE, epsilon = 1e-06, numiter = 30000)
```

**Arguments**

sample	If there are multiple samples, the number of the sample to be segmented
chrom	The chromosome on which the segmentation is to be carried out on
dat	The log2 ratios obtained from the clones located on that chromosome
datainfo	A dataframe containing information about the clones on that chromosome (name, chromosome and location (in Mbs))
useCloneDists	Boolean stating whether the distance between clones should be incorporated into the HMM. If false then the HMM become homogeneous.
covariates	A matrix containing the covariate information for the clones located on the chromosome to be segmented. It should have length one less than the number of clones as covariate information is not used when segmenting the first clone on the chromosome.
aic	Set to true if you want to use the aic. This is the default. Only one of aic and bic should be set to true.
bic	Set to true if you want to use the bic.
delta	A parameter to be set if you want to use the BIC
var.fixed	Logical variable - TRUE if you want to tie the variance to be the same across all states. Defaults to FALSE
epsilon	.
numiter	Number of iterations to be used in the optimization algorithm.

**Value**

The output is in the same format as that obtained when the `nlm` function is applied.

**Author(s)**

John Marioni and Mike Smith

---

genomePlot

*Plots the genome*

---

**Description**

Basic plot of the log<sub>2</sub> ratios for each array ordered along the genome.

**Usage**

```
genomePlot(input, array = 1, naut = 22, Y = FALSE, X
           = FALSE, main = NA, status, values, pch, cex, col, chrominfo =
           chrominfo.Mb, ylim = c(-2, 2), ylb = "Log2Ratio",
           chrom.to.plot = NA, xlim = c(0,NA), ...)
```

**Arguments**

<code>input</code>	an object of class <code>MAList</code> or <code>SegList</code>
<code>array</code>	integer of the array (sample) to be plotted.
<code>naut</code>	number of autosomes in the organism
<code>Y</code>	TRUE if chromosome Y is to be plotted, FALSE otherwise
<code>X</code>	TRUE if chromosome X is to be plotted, FALSE otherwise
<code>main</code>	Provides the title of the plot
<code>status</code>	character vector giving the control status of each spot on the array, of same length as the number of rows of <code>log2ratios(input)</code> . If omitted, all points are plotted in the default color, symbol and size.
<code>values</code>	character vector giving values of <code>status</code> to be highlighted on the plot. Defaults to unique values of <code>status</code> . Ignored if there is no <code>status</code> vector.
<code>pch</code>	vector or list of plotting characters. Default to integer code 16. Ignored if there is no <code>status</code> vector.
<code>col</code>	numeric or character vector of colors, of the same length as <code>values</code> . Defaults to <code>1:length(values)</code> . Ignored if there is no <code>status</code> vector.
<code>cex</code>	numeric vector of plot symbol expansions, of the the same length as <code>values</code> . Defaults to 0.2 for the most common <code>status</code> value and 1 for the others. Ignored if there is no <code>status</code> vector.
<code>chrominfo</code>	a chromosomal information associated with the mapping of the data.
<code>ylim</code>	Minimum y-scale to use for plotting.
<code>chrom.to.plot</code>	Specify which chromosome to plot
<code>ylb</code>	label for the Y-axis.
<code>xlim</code>	limits for the x-axis
<code>...</code>	Any other parameters

**Details**

The status vector is intended to specify the control status of each spot, for example "gene", "ratio control", "house keeping gene", "buffer" and so on. The vector is usually computed using the function `controlStatus` and a spot-types file. However the function may be used to highlight any subset of spots.

**Author(s)**

John Marioni

**See Also**

[MAList](#) [SegList](#)

---

heatmapGenome      *clustering and heatmap*

---

**Description**

This function clusters samples and shows their heatmap

**Usage**

```
heatmapGenome(input, response = as.factor(rep("All", ncol(input))),
              chrominfo = chrominfo.Mb, cutoff = 1, lowCol =
              "blue", highCol = "yellow", midCol = "white", ncolors =
              50, byclass = FALSE, showaber = FALSE, amplif = 1,
              homdel = -0.75, samplenames = colnames(input),
              vecchrom = 1:22, titles = "Image Plot", methodS =
              "ward", categoricalPheno = TRUE, CENTROMERE = FALSE)
```

**Arguments**

<code>input</code>	object of class <a href="#">MAList</a> or <a href="#">SegList</a>
<code>response</code>	phenotype of interest. defaults to the same phenotype assigned to all samples
<code>chrominfo</code>	a chromosomal information associated with the mapping of the data
<code>cutoff</code>	maximum absolute value. all the values are floored to +/-cutoff depending on whether they are positive or negative. defaults to 1
<code>ncolors</code>	number of colors in the grid. input to <a href="#">maPalette</a> . defaults to 50
<code>lowCol</code>	color for the low (negative) values. input to <a href="#">maPalette</a> . defaults to "red"
<code>highCol</code>	color for the high (positive) values. input to <a href="#">maPalette</a> . defaults to "green"
<code>midCol</code>	color for the values close to 0. input to <a href="#">maPalette</a> . defaults to "black"
<code>byclass</code>	logical indicating whether samples should be clustered within each level of the phenotype or overall. defaults to F
<code>showaber</code>	logical indicating whether high level amplifications and homozygous deletions should be indicated on the plot. defaults to F
<code>amplif</code>	positive value that all observations equal or exceeding it are marked by yellow dots indicating high-level changes. defaults to 1

homdel	negative value that all observations equal or below it are marked by light blue dots indicating homozygous deletions. defaults to -0.75
samplenames	sample names
vecchrom	vector of chromosomal indices to use for clustering and to display. defaults to 1:23
titles	plot title. defaults to "Image Plots"
methodS	clustering method to cluster samples. defaults to "ward"
categoricalPheno	logical indicating whether phenotype is categorical. Continuous phenotypes are treated as "no groups" except that their values are displayed. defaults to TRUE.
CENTROMERE	logical indicating whether to plot the centromere

### Details

This function is a more flexible version of the [heatmap](#). It can cluster within levels of categorical phenotype as well as all of the samples while displaying phenotype levels in different colors. It also uses any combination of chromosomes that is requested and clusters samples based on these chromosomes only. It draws the chromosomal boundaries and displays high level changes and homozygous deletions. If phenotype is not categorical, its values may still be displayed but groups are not formed and `byclass = F`. Image plot has the samples reordered according to clustering order.

### See Also

[heatmap](#)

---

IDProbes

*Interactive version of genomePlot*

---

### Description

Interactive version of [genomePlot](#). Allows the user to click near a probe and the name of that probe will be displayed next to it.

### Usage

```
IDProbes(input, array = 1, naut = 22, Y = FALSE, X
         = FALSE, status, values, pch, cex, col, chrominfo =
         chrominfo.Mb, ylim = c(-2, 2), ylb = "Log2Ratio",
         chrom.to.plot = 1, xlim = c(0, NA))
```

### Arguments

input	an object of class <a href="#">MAList</a> or <a href="#">SegList</a>
array	integer of the array (sample) to be plotted.
naut	number of autosomes in the organism
Y	TRUE if chromosome Y is to be plotted, FALSE otherwise
X	TRUE if chromosome X is to be plotted, FALSE otherwise

<code>status</code>	character vector giving the control status of each spot on the array, of same length as the number of rows of <code>log2ratios(input)</code> . If omitted, all points are plotted in the default color, symbol and size.
<code>values</code>	character vector giving values of <code>status</code> to be highlighted on the plot. Defaults to unique values of <code>status</code> . Ignored if there is no <code>status</code> vector.
<code>pch</code>	vector or list of plotting characters. Default to integer code 16. Ignored if there is no <code>status</code> vector.
<code>col</code>	numeric or character vector of colors, of the same length as <code>values</code> . Defaults to <code>1:length(values)</code> . Ignored if there is no <code>status</code> vector.
<code>cex</code>	numeric vector of plot symbol expansions, of the the same length as <code>values</code> . Defaults to 0.2 for the most common <code>status</code> value and 1 for the others. Ignored if there is no <code>status</code> vector.
<code>chrominfo</code>	a chromosomal information associated with the mapping of the data.
<code>ylim</code>	Minimum y-scale to use for plotting.
<code>chrom.to.plot</code>	Specify which chromosome to plot
<code>ylb</code>	label for the Y-axis.
<code>xlim</code>	limits for the x-axis

**Author(s)**

Mike Smith

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


---

*imputeMissingValues*
*Imputing log2 ratios***Description**

Imputing log2 ratios

**Usage**

```
imputeMissingValues(seg, chrominfo = chrominfo.Mb, maxChrom =
23, smooth = 0.1)
```

**Arguments**

<code>seg</code>	Object of class <a href="#">SegList</a>
<code>chrominfo</code>	a chromosomal information associated with the mapping of the data
<code>maxChrom</code>	Highest chromosome to impute
<code>smooth</code>	smoothing parameter for the lowess procedure

**Details**

There are two main reasons to impute data. One is that given that imputation is reasonable, one can increase the analytical power and improve results. Another, more practical, is that at the moment many widely used functions in R do not support missing values. While procedures such as kNN imputations is widely used for gene expression data, it is more powerful to take advantage of the genomic structure of the array CGH data and use a smoother. Note that we perform only one pass of smoothing. If there still remain missing values, they are imputed by the median on the chromosome or chromosomal arm where applicable.

**Value**

Computes and returns the imputed log2 ratio matrix of the aCGH object.

**See Also**

[SegList](#)

---

LargeDataObject-class

*Large Data Object - class*

---

**Description**

A virtual class including the data classes `RGList`, `MAList` and `SegList`, all of which typically contain large quantities of numerical data in vector, matrices and `data.frames`.

**Methods**

A `show` method is defined for objects of class `LargeDataObject` which uses `printHead` to print only the leading elements or rows of components or slots which contain large quantities of data.

**Author(s)**

Gordon Smyth

**Examples**

```
# see normalizeBetweenArrays
```

---

log2ratios	<i>Extracting log2 ratios</i>
------------	-------------------------------

---

**Description**

This function extracts the log2 ratios from either an `MAList` object or a `SegList` object.

**Usage**

```
log2ratios(x)
```

**Arguments**

`x` An object of class `MAList` or `SegList`

**Author(s)**

Mike Smith

---

mergeStates	<i>Function to merge states based on their state means</i>
-------------	------------------------------------------------------------

---

**Description**

`mergeStates` takes the output of a segmentation algorithm in the form of a `SegList` and iteratively merges the states with means closer than a supplied threshold.

**Usage**

```
mergeStates(segList, MergeType = 1, pv.thres=0.0001, ansari.sign=0.01, minDiff =
```

**Arguments**

<code>segList</code>	Object of class <code>SegList</code> .
<code>MergeType</code>	Select either 1 or 2. 1 uses a new merging algorithm developed by Hanni Wilenbrock and Jane Fridlyand.
<code>pv.thres</code>	Significance threshold for Wilcoxon test for level merging. Used when <code>MergeType = 1</code> .
<code>ansari.sign</code>	Significance threshold for Ansari-Bradley test. Used when <code>MergeType = 1</code> .
<code>minDiff</code>	The states whose predicted values are less than <code>minDiff</code> apart are merged into one state and all the predicted values are recomputed. Used when <code>MergeType = 2</code> .

**Details**

This function is intended to reduce effect of the possible small magnitude technological artifacts on the structure determination.



**Value**

A `SegList` object is returned with the merged states stored in the `pred` list.

**Author(s)**

Jane Fridlyand

**References**

Application of Hidden Markov Models to the analysis of the array CGH data, Fridlyand et.al., *JMVA*, 2004

**See Also**

`SegList`, `runHomHMM`, `runGLAD`, `runDNAcopy`

---

non.zero.length.distr.non.tiled

*Empirical distribution of segment lengths in non-tiled regions with copy number gains or losses*

---

**Description**

This file contains the empirical distribution of segment lengths (of untiled regions and whose state mean indicates that they correspond to regions of copy number gain or loss) derived by applying the DNAcopy segmentation scheme (Olshen et al., 2004) to an unpublished breast cancer dataset. Instead of using the physical length of the segments we calculate the lengths as a proportion of the length of the untiled region.

**Usage**

```
data(non.zero.length.distr.non.tiled)
```

**Source**

The empirical distribution was derived using an unpublished breast cancer dataset.

---

non.zero.length.distr.tiled

*Empirical distribution of segment lengths in tiled regions with copy number gains or losses*

---

**Description**

This file contains the empirical distribution of segment lengths (of tiled regions and whose state mean indicates that they correspond to regions of copy number gain or loss) derived by applying the DNAcopy segmentation scheme (Olshen et al., 2004) to an unpublished breast cancer dataset. Instead of using the physical length of the segments we calculate the lengths as a proportion of the length of the tiled region.

**Usage**

```
data(non.zero.length.distr.tiled)
```

**Source**

The empirical distribution was derived using an unpublished breast cancer dataset.

---

```
plotSegmentedGenome
Plots the genome
```

---

**Description**

Basic plot of the log2 ratios for each array ordered along the genome.

**Usage**

```
plotSegmentedGenome(..., array = 1, naut = 22, Y = FALSE, X
                    = FALSE, status, values, pch, cex, col, chrominfo =
                    chrominfo.Mb, ylim = c(-2, 2), ylb = "Log2Ratio",
                    chrom.to.plot = NA, xlim = c(0,NA), colors = NULL,
                    mark.regions = FALSE, main = NA)
```

**Arguments**

...	Objects of class <a href="#">SegList</a>
array	integer of the array (sample) to be plotted.
naut	number of autosomes in the organism
Y	TRUE if chromosome Y is to be plotted, FALSE otherwise
X	TRUE if chromosome X is to be plotted, FALSE otherwise
status	character vector giving the control status of each spot on the array, of same length as the number of rows of <code>log2ratios(input)</code> . If omitted, all points are plotted in the default color, symbol and size.
values	character vector giving values of <code>status</code> to be highlighted on the plot. Defaults to unique values of <code>status</code> . Ignored if there is no <code>status</code> vector.
pch	vector or list of plotting characters. Default to integer code 16. Ignored if there is no <code>status</code> vector.
col	numeric or character vector of colors, of the same length as <code>values</code> . Defaults to <code>1:length(values)</code> . Ignored if there is no <code>status</code> vector.
cex	numeric vector of plot symbol expansions, of the the same length as <code>values</code> . Defaults to 0.2 for the most common <code>status</code> value and 1 for the others. Ignored if there is no <code>status</code> vector.
chrominfo	a chromosomal information associated with the mapping of the data.
ylim	Minimum y-scale to use for plotting.
chrom.to.plot	Specify which chromosome to plot
ylb	label for the Y-axis.

xlim	limits for the x-axis
colors	vector of colors to plot segmented states of each SegList passed to the function.
mark.regions	Boolean. If true will colour code the segmentation plot using the information stored in \\$.regions and generated by bayesCGH::nudSegmentation, or an equivalent method.
main	Specify the title of the plot

### Details

The status vector is intended to specify the control status of each spot, for example "gene", "ratio control", "house keeping gene", "buffer" and so on. The vector is usually computed using the function controlStatus and a spot-types file. However the function may be used to highlight any subset of spots.

### Author(s)

Mike Smith

### See Also

[genomePlot SegList](#)

---

processCGH                      *Process data in an MAList*

---

### Description

This function takes object of class MAList and it re-orders and filters clones based on their mapping information and proportion missing. It also average duplicated clones and imputes missing values for clones that are still NA after the filtering step. Note that imputation will only take place if duplicated clones are removed.

### Usage

```
processCGH(input, maxChromThreshold = 22, minChromThreshold = 1,
           method.of.averaging = NULL, ID = "ID", prop.missing = 0.1)
```

### Arguments

input	Object of class MAList or RGList
maxChromThreshold	Chromosomes are ordered and numbered as usual, except for X and Y chromosome, which in for Homo sapiens genome have numbers 23 and 24 respectively, in for Mus musculus 20 and 21, etc. Remove chromosomes from segmentation analysis which are greater than this value.
minChromThreshold	Chromosomes are ordered and numbered as usual, except for X and Y chromosome, which in for Homo sapiens genome have numbers 23 and 24 respectively, in for Mus musculus 20 and 21, etc. Remove chromosomes from segmentation analysis which are lower than this value.

method.of.averaging	If left as the default no combining of replicate spots takes place. Otherwise this should specify a function which takes a vector of duplicates and combines them into a single value.
ID	Name of column in <code>RG\$genes</code> corresponding to the clone names. For most data the default will work, however for affy data the value for ID should be "CloneName"
prop.missing	For each probe the proportion of NA's is calculated, and the probe is kept for further analysis if the proportion of NA's is less than <code>missing.prop</code>

**Value**

Object of class `SegList`

**Author(s)**

Jane Fridlyand, Peter Dimitrov, John Marioni and Mike Smith

**See Also**

`MAList`

---

`read.clonesinfo`     *Reading chromosome and positional information about each clone.*

---

**Description**

Function to read the chromosomal position information of each clone and incorporate it into the `genes.data.frame` of the relevant object.

**Usage**

```
read.clonesinfo(file, RG, path = NULL, sep="\t", quote="\"")
```

**Arguments**

<code>file</code>	Name of the file containing the chromosomal information.
<code>RG</code>	Object containing a <code>\\$genes.data.frame</code> that the information should be incorporated into.
<code>path</code>	Path to the chromosomal information file.
<code>sep</code>	Identifying the column separator in the designated file.
<code>quote</code>	Identifying the quotation character used in the designated file.

**Author(s)**

Mike Smith

---

```
readPositionalInfo readPositionalInfo
```

---

### Description

This function automatically inserts information about the chromosomal positional of a clone into the `\$genes` matrix of an `RGList` or `MAList`. This information is used in all the available segmentation methods as well as many of the plotting functions available in `snapCGH`.

### Usage

```
readPositionalInfo(input, source, path = NULL)
```

### Arguments

<code>input</code>	An object of class <code>RGList</code> or <code>MAList</code>
<code>source</code>	Defines which platform or technology this data was produced on. Currently supported options are: "aglient", "bluefuse", "nimblegen". This list will be expanded in time.
<code>path</code>	Optional parameter to specify where the original data is stored. Defaults to the current working directory. This option is only required for reading "bluefuse" data at the moment, as chromosome information isn't read by <code>limma</code> as default.

---

```
removeByWeights Remove clones based on a weights matrix
```

---

### Description

An example function to be used by the `filterClones` method. This function takes an MA list, a weights matrix and a threshold and returns the indices of any clones with weight below the threshold.

### Usage

```
removeByWeights(MA, weights=MA$weights, threshold = 0.2)
```

### Arguments

<code>MA</code>	An object of class <code>MAList</code>
<code>weights</code>	A matrix with the same dimensions as <code>MA</code> containing weight information.
<code>threshold</code>	Threshold value. Any clones with weight below this are removed.

### Author(s)

Mike Smith

### See Also

`filterClones`

runBioHMM

*This function implements the BioHMM***Description**

This function reads in a dataset of log2 ratios and the corresponding clone and covariate information. It calculates a heterogeneous HMM when there are 1,2,3,4 or 5 underlying states and chooses between them using either the AIC or BIC. It then assigns clones using a modified version of the Viterbi algorithm.

**Usage**

```
runBioHMM(input, useCloneDists = TRUE, covariates, criteria="AIC", delta=NA,
var.fixed=FALSE, epsilon = 1e-06, numiter = 30000)
```

**Arguments**

input	An object of class <a href="#">MAList</a> or <a href="#">SegList</a>
useCloneDists	Boolean stating whether the distance between clones should be incorporated into the HMM. If false then the HMM becomes homogeneous.
covariates	This is a dataframe containing information about covariate factors. The first two columns should be Chrom (giving the chromosome on which a clone is located) and Mb (giving the position of the chromosome along a particular chromosome in Megabases). The order should be the same as that described above with the following crucial difference. No covariate information about the first clone is used in the segmentation. Hence, for each chromosome, there should be one less row in the covariate dataframe than in the datainfo dataframe corresponding to this missing chromosome. This is important if the transition matrix is to be calculated correctly.
criteria	Options are AIC or BIC depending upon which we want to use to distinguish between the number of states
delta	A variable to be assigned if the BIC is used.
var.fixed	Logical variable - TRUE if you want to tie the variance to be the same across all states. Defaults to FALSE
epsilon	Stopping criterion for the optimization algorithm.
numiter	Number of iterations to be used in the optimization algorithm.

**Value**

The model returns an object of class [SegList](#).

**Author(s)**

John Marioni and Mike Smith

**References**

Marioni, J.C., Thorne, N.P., Tavaré, S., BioHMM: a heterogeneous Hidden Markov Model for segmenting array CGH data, submitted

---

runDNACopy	<i>Results of segmenting an MAList data object using the DNACopy library</i>
------------	------------------------------------------------------------------------------

---

### Description

The results of segmenting data from copy number array experiments from programs such as circular binary segmentation (CBS). This function requires the library DNACopy to be loaded.

### Usage

```
runDNACopy(input, smooth.region=2, outlier.SD.scale = 4, smooth.SD.scale = 2, trim = 0.05,
  "perm"), kmax = 25, nmin = 200, undo.splits = c("none", "prune", "sdundo"),
  undo.prune = 0.05, undo.SD = 3, nperm = 10000, eta = 0.05)
```

### Arguments

input	An object of class <code>MAList</code> or <code>SegList</code>
smooth.region	number of points to consider on the left and the right of a point to detect it as an outlier.
outlier.SD.scale	the number of SDs away from the nearest point in the smoothing region to call a point an outlier.
smooth.SD.scale	the number of SDs from the median in the smoothing region where a smoothed point is positioned.
trim	proportion of data to be trimmed for variance calculation for smoothing outliers and undoing splits based on SD.
alpha	significance levels for the test to accept change-points.
p.method	method used for p-value computation. For the "perm" method the p-value is based on full permutation. For the "hybrid" method the maximum over the entire region is split into maximum of max over small segments and max over the rest. Approximation is used for the larger segment max. Default is hybrid.
kmax	the maximum width of smaller segment for permutation in the hybrid method.
nmin	the minimum length of data for which the approximation of maximum statistic is used under the hybrid method.
undo.splits	A character string specifying how change-points are to be undone, if at all. Default is "none". Other choices are "prune", which uses a sum of squares criterion, and "sdundo", which undoes splits that are not at least this many SDs apart.
undo.prune	the proportional increase in sum of squares allowed when eliminating splits if undo.splits="prune".
undo.SD	the number of SDs between means to keep a split if undo.splits="sdundo".
nperm	number of permutations used for p-value computation.
eta	the probability to declare a change conditioned on the permuted statistic exceeding the observed statistic exactly $j$ ( $= 1, \dots, nperm * \alpha$ ) times.

**Value**

The function returns an object of class [SegList](#)

**Author(s)**

Mike Smith, based upon DNACopy help files written by E. S. Venkatraman and Adam Olshen

**See Also**

[segment](#) [MAList](#) [runHomHMM](#) [runGLAD](#) [SegList](#)

---

runGLAD

*Results of segmenting an aCGHList data object using the GLAD library*

---

**Description**

This function allows the detection of breakpoints in genomic profiles obtained by array CGH technology and affects a status (gain, normal or lost) to each clone. It requires that the library GLAD is loaded.

**Usage**

```
runGLAD(input, smoothfunc="lawsglad", base=FALSE, sigma = NULL, bandwidth=10,
round=2, lambdabreak=8, lambdacluster=8, lambdaclusterGen=40,
type="tricubic", param=c(d=6), alpha=0.001, method="centroid",
nmax=8, verbose=FALSE, ...)
```

**Arguments**

input	An object of class <a href="#">MAList</a> or <a href="#">SegList</a>
smoothfunc	Type of algorithm used to smooth <code>LogRatio</code> by a piecewise constant function. Choose either <code>lawsglad</code> , <code>aws::aws</code> or <code>aws::laws</code> .
base	If TRUE, the position of clone is the physical position onto the chromosome, otherwise the rank position is used.
sigma	Value to be passed to either argument <code>sigma2</code> of <code>aws::aws</code> function or <code>shape</code> of <code>aws::laws</code> . If NULL, <code>sigma</code> is calculated from the data.
bandwidth	Set the maximal bandwidth <code>hmax</code> in the <code>aws::aws</code> or <code>aws::laws</code> function. For example, if <code>bandwidth=10</code> then the <code>hmax</code> value is set to $10 * X_N$ where $X_N$ is the position of the last clone.
round	The smoothing results are rounded or not depending on the <code>round</code> argument. The <code>round</code> value is passed to the argument <code>digits</code> of the <a href="#">round</a> function.
lambdabreak	Penalty term ( $\lambda'$ ) used during the <b>Optimization of the number of breakpoints</b> step.
lambdacluster	Penalty term ( $\lambda^*$ ) used during the <b>MSHR clustering by chromosome</b> step.
lambdaclusterGen	Penalty term ( $\lambda^*$ ) used during the <b>HCSR clustering throughout the genome</b> step.



type	Type of kernel function used in the penalty term during the <b>Optimization of the number of breakpoints</b> step, the <b>MSHR clustering by chromosome</b> step and the <b>HCSR clustering throughout the genome</b> step.
param	Parameter of kernel used in the penalty term.
alpha	Parameter for ?????.
method	The agglomeration method to be used during the <b>MSHR clustering by chromosome</b> and the <b>HCSR clustering throughout the genome</b> clustering steps.
nmax	Maximum number of clusters (N*max) allowed during the the <b>MSHR clustering by chromosome</b> and the <b>HCSR clustering throughout the genome</b> clustering steps.
verbose	If TRUE some information are printed
...	

### Details

For a detailed explanation of the GLAD algorithm please see the relevant section of the GLAD manual: [glad](#)

### Value

The function returns an object of class [SegList](#)

### See Also

[glad](#) [MAList](#) [runHomHMM](#) [runDNACopy](#) [SegList](#)

---

runHomHMM	<i>A function to fit unsupervised Hidden Markov model</i>
-----------	-----------------------------------------------------------

---

### Description

This function fits an unsupervised Hidden Markov model to a given [MAList](#) or [SegList](#)

### Usage

```
runHomHMM(input, vr = 0.01,
           maxiter = 100, criteria = "AIC", delta = NA,
           full.output = FALSE, eps = 0.01)
```

### Arguments

input	an object of class <a href="#">MAList</a> or <a href="#">SegList</a>
vr	Gets passed to the function <code>repeated: :hidden</code> as the <code>pshape</code> argument.
maxiter	Gets passed to the function <code>repeated: :hidden</code> as the <code>iterlim</code> argument.
criteria	Choice of which selection criteria should be used in the algorithm. The choices are either AIC or BIC.
delta	Delta value used of the BIC is selected. If no value is entered it defaults to 1.
full.output	if true the <a href="#">SegList</a> output includes a probability that a clone is in its assigned state and a smoothed value for the clone.
eps	parameter controlling the convergence of the EM algorithm.

**See Also**

[runDNACopy](#) [runGLAD](#) [SegList](#)

---

<code>runTilingArray</code>	<i>Results of segmenting an MAList data object using the Picard et al algorithm found in the tilingArray library</i>
-----------------------------	----------------------------------------------------------------------------------------------------------------------

---

**Description**

Wrapper calling the Tiling Array segmentation algorithm on an MAList object. This function requires the library DNACopy to be loaded.

**Usage**

```
runTilingArray(input, maxSeg = 5, maxk = 200, criteria = "BIC")
```

**Arguments**

<code>input</code>	An object of class <a href="#">MAList</a> or <a href="#">SegList</a>
<code>maxSeg</code>	integer of length 1, maximum number of segments (= 1 + maximum number of change points)
<code>maxk</code>	integer of length 1, maximum length of a single segment
<code>criteria</code>	Criteria for model selection. Options are "none", "AIC" and "BIC" (default)

**Value**

The function returns an object of class [SegList](#)

**Author(s)**

Mike Smith

**See Also**

[segment](#) [MAList](#) [runHomHMM](#) [runGLAD](#) [SegList](#)

---

<code>SegList-class</code>	<i>Segmentation States - class</i>
----------------------------	------------------------------------

---

**Description**

A list based class for storing the results of a segmentation algorithm. They are generally created by running one of the following functions [runHomHMM](#), [runGLAD](#) or [runDNACopy](#) on an [MAList](#) object.

**Slots/List Components**

Objects should contain the following list components:

pred: Predicted value of the state.  
 disp: Dispersion.  
 obs: Observed value.  
 state: Numeric value.  
 nstates.hmm: The number of states per chromosome. Each row represents a chromosome and each column is an array of states.  
 genes: data.frame that contains the chromosome and position on the chromosome for each clone. Used for plotting.

**Optional:**

rpred: Smoothed value for the clone.  
 prob: Probability of the clone being in the assigned state.

## Methods

SegLists can be subsetted and combined. They also return dimensions so functions such as `dim`, `nrow` and `ncol` are also defined. SegList inherits the `show` method from the Limma class `LargeDataObject`. This means that the SegList will print in a relatively compact way.

## Author(s)

Mike Smith

---

simulateData	<i>A function for simulating aCGH data and the corresponding clone layout</i>
--------------	-------------------------------------------------------------------------------

---

## Description

This simulation scheme operates in two stages. Initially, we simulate the layout of clones before using a modified version of the scheme developed by Willenbrock et al., 2005 to generate the  $\log_2$  ratios. For each simulated clone layout we generate 20 sets of simulated  $\log_2$  ratios from one of five templates. Additionally, we also take account of the cellularity of the test sample in our simulation.

## Usage

```
simulateData(nArrays = 20, chrominfo = NULL, prb.short.tiled = 0.5,
             prb.long.tiled = 0.5, non.tiled.lower.res = 0.9,
             non.tiled.upper.res = 1.1, length.clone.lower = 0.05,
             length.clone.upper = 0.2, tiled.lower.res = -0.05,
             tiled.upper.res = 0, sd = NULL, output = FALSE,
             prb.proportion.tiled = c(0.2, 0.2, 0.2, 0.2, 0.2),
             zerolengthnontiled = NULL, zerolengthtiled = NULL,
             nonzerolengthnontiled = NULL, nonzerolengthtiled =
             NULL, seed = 1)
```

**Arguments**

nArrays	The number of arrays we want to simulate
chrominfo	The information about chromosome length/centromere location to be used. Defaults to the information provided in aCGH package of Jane Fridlyand and Peter Dimitrov.
prb.short.tiled	The probability of a tiled region on the short arm of the simulated chromosome (defaults to 0.5).
prb.long.tiled	The probability of a tiled region on the long arm of the simulated chromosome (defaults to 0.5).
non.tiled.lower.res	The lower limit for the distance (in Mbs) between adjacent clones in non-tiled regions of the genome (defaults to 0.9Mb).
non.tiled.upper.res	The upper limit for the distance (in Mbs) between adjacent clones in non-tiled regions of the genome (defaults to 1.1Mb).
length.clone.lower	The lower limit for the length (in Mbs) of a clone (this defaults to 0.05Mb).
length.clone.upper	The upper limit for the length (in Mbs) of a clone (this defaults to 0.2Mb).
tiled.lower.res	The lower limit for the distance (in Mbs) between adjacent clones in tiled regions of the genome (defaults to -0.05Mb).
tiled.upper.res	The upper limit for the distance (in Mbs) between adjacent clones in tiled regions of the genome (defaults to 0Mb).
sd	The standard deviation of the simulated data in each of the states. Defaults to being randomly sampled between 0.1 and 0.2.
output	A logical variable which is TRUE if you want the output to be written to txt files in the present working directory. Defaults to FALSE.
prb.proportion.tiled	Given that an arm of a chromosome contains a tiled region this variable (which is a vector of length 5) gives the probability that 20,30,40,50 or 100% of the chromosome is tiled. It defaults to (0.2,0.2,0.2,0.2,0.2)
zerolengthnontiled	The empirical distribution for regions of the genome which are non-tiled and contain no copy number gains or losses. Defaults to zero.length.distr.non.tiled
zerolengthtiled	The empirical distribution for regions of the genome which are tiled and contain no copy number gains or losses. Defaults to zero.length.distr.tiled
nonzerolengthnontiled	The empirical distribution for regions of the genome which are non-tiled and contain no copy number gains or losses. Defaults to non.zero.length.distr.non.tiled
nonzerolengthtiled	The empirical distribution for regions of the genome which are tiled and contain copy number gains or losses. Defaults to non.zero.length.distr.tiled
seed	Seed value allowing simulation to be reproduced if the same seed value is set.

**Details**

For more details see the article by Marioni and Thorne published in Bioinformatics.

**Value**

The function returns a list containing the following elements.

<code>clones</code>	Gives the start, end and midpoint of the simulated clones.
<code>class.output</code>	A list of the true underlying state clones are assigned to for each of the twenty simulations associated with each clone layout.
<code>class.matrix</code>	Defines the true underlying state clones are assigned to in each of the five classes
<code>classes</code>	Which of the five class outputs has been used to simulate the $\log_2$ ratios
<code>datamatrix</code>	A matrix containing twenty columns each of which contains the simulated $\log_2$ ratios associated with each of the simulations for a particular clone layout.
<code>samples</code>	Gives information about the cellularity associated with each of the samples.

**Author(s)**

John Marioni and Natalie Thorne

**References**

See the relevant article in Bioinformatics or the following website: [www.damtp.cam.ac.uk/user/jcm68](http://www.damtp.cam.ac.uk/user/jcm68)

---

<code>Viterbi.five</code>	<i>A scaled Viterbi algorithm for allocating clones to one of five underlying states.</i>
---------------------------	-------------------------------------------------------------------------------------------

---

**Description**

A work horse of the `fit.model` function. It uses a scaled version of the Viterbi algorithm to allocate clones to one of five underlying states as fitted using a heterogeneous HMM.

**Usage**

```
Viterbi.five(y, BFGS.output, BFGS.trans.mat)
```

**Arguments**

<code>y</code>	the data to be allocated to states
<code>BFGS.output</code>	The output obtained from the <code>find.param.five</code> function
<code>BFGS.trans.mat</code>	A list of the heterogeneous transition matrices

**Value**

A vector of numbers indicating to which state clones are allocated to.

**Author(s)**

John Marioni

---

Viterbi.four	<i>A scaled Viterbi algorithm for allocating clones to one of four underlying states.</i>
--------------	-------------------------------------------------------------------------------------------

---

**Description**

A work horse of the fit.model function. It uses a scaled version of the Viterbi algorithm to allocate clones to one of four underlying states as fitted using a heterogeneous HMM.

**Usage**

```
Viterbi.four(y, BFGS.output, BFGS.trans.mat)
```

**Arguments**

y	the data to be allocated to states
BFGS.output	The output obtained from the find.param.four function
BFGS.trans.mat	A list of the heterogeneous transition matrices

**Value**

A vector of numbers indicating to which state clones are allocated to.

**Author(s)**

John Marioni

---

Viterbi.three	<i>A scaled Viterbi algorithm for allocating clones to one of two underlying states.</i>
---------------	------------------------------------------------------------------------------------------

---

**Description**

A work horse of the fit.model function. It uses a scaled version of the Viterbi algorithm to allocate clones to one of three underlying states as fitted using a heterogeneous HMM.

**Usage**

```
Viterbi.three(y, BFGS.output, BFGS.trans)
```

**Arguments**

y	the data to be allocated to states
BFGS.output	The output obtained from the find.param.three function
BFGS.trans	A list of the heterogeneous transition matrices

**Value**

A vector of numbers indicating to which state clones are allocated to.

**Author(s)**

John Marioni

---

Viterbi.two	<i>A scaled Viterbi algorithm for allocating clones to one of two underlying states.</i>
-------------	------------------------------------------------------------------------------------------

---

**Description**

A work horse of the fit.model function. It uses a scaled version of the Viterbi algorithm to allocate clones to one of two underlying states as fitted using a heterogeneous HMM.

**Usage**

```
Viterbi.two(y, BFGS.output, BFGS.trans.mat)
```

**Arguments**

y	the data to be allocated to states
BFGS.output	The output obtained from the find.param.two function
BFGS.trans.mat	A list of the heterogeneous transition matrices

**Value**

A vector of numbers indicating to which state clones are allocated to.

**Author(s)**

John Marioni

---

zero.length.distr.non.tiled	<i>Empirical distribution of segment lengths in non-tiled regions with no copy number gains or losses</i>
-----------------------------	-----------------------------------------------------------------------------------------------------------

---

**Description**

This file contains the empirical distribution of segment lengths (of untiled regions and whose state mean indicates that they correspond to regions of no copy number gain or loss) derived by applying the DNACopy segmentation scheme (Olshen et al., 2004) to an unpublished breast cancer dataset. Instead of using the physical length of the segments we calculate the lengths as a proportion of the length of the untiled region.

**Usage**

```
data(zero.length.distr.non.tiled)
```

**Source**

The empirical distribution was derived using an unpublished breast cancer dataset.

---

```
zero.length.distr.tiled
```

*Empirical distribution of segment lengths in tiled regions with no copy number gains or losses*

---

### Description

This file contains the empirical distribution of segment lengths (of tiled regions and whose state mean indicates that they correspond to regions of no copy number gain or loss) derived by applying the DNACopy segmentation scheme (Olshen et al., 2004) to an unpublished breast cancer dataset. Instead of using the physical length of the segments we calculate the lengths as a proportion of the length of the tiled region.

### Usage

```
data(zero.length.distr.tiled)
```

### Source

The empirical distribution was derived using an unpublished breast cancer dataset.

---

```
zoomChromosome
```

*Interactive plot of a single chromosome*

---

### Description

Plot splitting the screen into two. The top window displays the entire chromosome, whilst the bottom plots a selected region. The plot is interactive allowing the user to click twice on a chromosome in the upper plot and have it the region between the two clicks displayed below.

### Usage

```
zoomChromosome(..., array = 1, chrom.to.plot, colors = NULL, chrominfo = chrominfo.Mb, ylim = c(-2, 2))
```

### Arguments

<code>...</code>	Objects of type <code>MAList</code> or <code>SegList</code>
<code>array</code>	Specify which array should be plotted.
<code>chrom.to.plot</code>	Which chromosome should be plotted
<code>colors</code>	Vector specify the colors for each of the <code>SegLists</code>
<code>chrominfo</code>	chromosomal information associated with the mapping of the data.
<code>ylim</code>	Specify the limits of the y-axis

### Details

If `colors` is unspecified then all `SegLists` passed to this function will be plotted in blue. Since this makes it quite hard to tell which is which it is highly recommended to specify the colors vector if more than one object is being passed to this function.



**Author(s)**

Mike Smith

---

`zoomGenome`*Interactive plot of the whole genome*

---

**Description**

Plot splitting the screen into two. The top window displays the entire genome, whilst the bottom window plots a single chromosome. The plot is interactive allowing the user to click on a chromosome in the upper plot and have it displayed below. Clicking to either side of the plot borders ends the interactivity.

**Usage**

```
zoomGenome(..., array = 1, colors = NULL, chrominfo = chrominfo.Mb)
```

**Arguments**

<code>...</code>	Objects of type <a href="#">SegList</a>
<code>array</code>	Specify which array should be plotted.
<code>colors</code>	Vector specify the colors for each of the SegLists
<code>chrominfo</code>	chromosomal information associated with the mapping of the data.

**Details**

If `colors` is unspecified then all objects passed to this function will be plotted in blue. Since this makes it quite hard to tell which is which it is highly recommended to specify the colors vector if more than one object is being passed to this function.

**Author(s)**

Mike Smith

# Index

- \*Topic **array**
  - dim, 5
  - dimnames, 4
- \*Topic **classes**
  - LargeDataObject-class, 15
  - SegList-class, 26
- \*Topic **cluster**
  - heatmapGenome, 12
- \*Topic **datasets**
  - chrominfo.Mb, 2
  - convert.output, 3
  - log2ratios, 16
  - non.zero.length.distr.non.tiled, 17
  - non.zero.length.distr.tiled, 17
  - simulateData, 27
  - zero.length.distr.non.tiled, 31
  - zero.length.distr.tiled, 32
- \*Topic **data**
  - LargeDataObject-class, 15
  - SegList-class, 26
- \*Topic **file**
  - processCGH, 19
- \*Topic **hplot**
  - genomePlot, 11
  - heatmapGenome, 12
  - IDProbes, 13
  - plotSegmentedGenome, 18
  - zoomChromosome, 32
  - zoomGenome, 33
- \*Topic **manip**
  - cbind, 1
  - compareSegmentations, 2
  - readPositionalInfo, 21
- \*Topic **methods**
  - filterClones, 6
  - findBreakPoints, 6
  - read.clonesinfo, 20
  - removeByWeights, 21
  - runDNACopy, 23
  - runGLAD, 24
  - runTilingArray, 26
- \*Topic **misc**
  - find.param.five, 7
  - find.param.four, 7
  - find.param.one, 8
  - find.param.three, 9
  - find.param.two, 9
  - Viterbi.five, 29
  - Viterbi.four, 30
  - Viterbi.three, 30
  - Viterbi.two, 31
- \*Topic **models**
  - fit.model, 10
  - imputeMissingValues, 14
  - mergeStates, 16
  - runBioHMM, 22
  - runHomHMM, 25
  - [.SegList (*SegList-class*), 26
- cbind, 1, 1
- chrominfo.Mb, 2
- combine.func (*mergeStates*), 16
- compareBreakPoints
  - (*compareSegmentations*), 2
- compareSegmentations, 2
- convert.output, 3
- dim, 5, 5, 27
- dim, SegList-method (*dim*), 5
- dim.MAList (*dim*), 5
- dim.RGList (*dim*), 5
- dim.SegList (*dim*), 5
- dimnames, 4, 5
- dimnames, SegList-method (*dimnames*), 4
- dimnames.SegList (*dimnames*), 4
- filterClones, 6, 21
- find.param.five, 7
- find.param.four, 7
- find.param.one, 8
- find.param.three, 9
- find.param.two, 9
- findBreakPoints, 6

- fit.model, 10
- floor.func (*heatmapGenome*), 12
- generate.data (*simulateData*), 27
- genomePlot, 11, 13, 14, 19
- glad, 25
- heatmap, 13
- heatmapGenome, 12
- IDProbes, 13
- imputeMissingValues, 14
- LargeDataObject, 27
- LargeDataObject-class, 15
- length (*dim*), 5
- length, SegList-method (*dim*), 5
- length.MAList (*dim*), 5
- length.RGList (*dim*), 5
- length.SegList (*dim*), 5
- log2ratios, 16
- MAList, 1, 6, 11–13, 16, 19–26, 32
- maPalette, 12
- maPalette (*heatmapGenome*), 12
- MergeLevels.new (*mergeStates*), 16
- MergeLevels.old (*mergeStates*), 16
- mergeStates, 16
- ncol, 27
- non.zero.length.distr.non.tiled, 17
- non.zero.length.distr.tiled, 17
- nrow, 27
- plotChrom (*heatmapGenome*), 12
- plotSegmentedGenome, 6, 18
- plotValChrom (*heatmapGenome*), 12
- plotvalChrom.func (*heatmapGenome*), 12
- plotValGenome (*heatmapGenome*), 12
- print.SegList (*SegList-class*), 26
- processCGH, 6, 19
- prop.na (*processCGH*), 19
- rbind.SegList (*cbind*), 1
- read.clonesinfo, 20
- readPositionalInfo, 21
- removeByWeights, 21
- RGList, 19, 21
- round, 24
- run.nelder (*fit.model*), 10
- runBioHMM, 22
- runDNAcopy, 17, 23, 25, 26
- runGLAD, 17, 24, 24, 26
- runHomHMM, 17, 24, 25, 25, 26
- runTilingArray, 26
- sample.names (*genomePlot*), 11
- SegList, 1, 3, 7, 11–20, 22–26, 32, 33
- SegList-class, 26
- segment, 24, 26
- show, 27
- show, LargeDataObject-method (*LargeDataObject-class*), 15
- show, SegList-method (*SegList-class*), 26
- simulateData, 3, 27
- states.hmm.func (*runHomHMM*), 25
- Viterbi.five, 29
- Viterbi.four, 30
- Viterbi.three, 30
- Viterbi.two, 31
- zero.length.distr.non.tiled, 31
- zero.length.distr.tiled, 32
- zoomChromosome, 32
- zoomGenome, 33