

Package ‘genoset’

September 24, 2012

Type Package

Title Provides classes similar to ExpressionSet for copy number analysis

Version 1.6.0

Date 2011-01-15

Author Peter M. Haverty

Maintainer Peter M. Haverty <phaverty@gene.com>

Description Load, manipulate, and plot copynumber and BAF data. GenoSet class extends eSet by adding a ‘locData’ slot for a RangedData object from the IRanges package. This object contains feature genome location data and provides for simple subsetting on genome location. CNSet and BAFSet extend GenoSet and require assayData matrices for Copy Number (cn) or Log-R Ratio (lrr) and B-Allele Frequency (baf) data. Implements and provides convenience functions for processing of copy number and B-Allele Frequency data.

License Artistic-2.0

LazyLoad yes

Depends R (>= 2.10), methods, BiocGenerics (>= 0.1.6), Biobase (>= 2.15.1), IRanges (>= 1.13.5), bigmemory, GenomicRanges

Imports methods, BiocGenerics, Biobase, DNACopy, graphics, IRanges, stats, GenomicRanges, Biostrings, BSgenome, bigmemory

Suggests RUnit

Enhances parallel, BSgenome.Hsapiens.UCSC.hg19

biocViews Infrastructure, DataRepresentation, Microarray, SNP, CopyNumberVariants

Collate ‘genoset-class.R’ ‘cnset-class.R’ ‘bafset-class.R’ ‘DataFrame-methods.R’ ‘bigmat.R’ ‘test_genoset_package.R’

ByteCompile TRUE

R topics documented:

genoset-package	3
asFactorMatrix	3
assayDataElement	4
assayDataElement<-	4
attachAssayDataElements	5
baf	5
baf2mbaf	6
BAFSet	7
bafset-class	8
BAFSet.to.ExpressionSets	9
boundingIndices	9
boundingIndices2	11
boundingIndicesByChr	12
chr	13
chrIndices	13
chrInfo	14
chrNames	15
chrOrder	15
cn	16
CNSet	16
cnset-class	17
colMeans	18
convertToBigMatrix	19
featureNames<-	19
gcCorrect	20
genomeAxis	21
genoPlot	22
genoPos	23
GenoSet	23
genoset-class	24
genoset-datasets	25
initGenoSet	26
isGenomeOrder	27
loadGC	27
locData	28
lrr	29
modeCenter	29
pos	30
rangeColMeans	30
rangeSampleMeans	31
readGenoSet	32
relocateAssayData	32
runCBS	33
segs2RangedData	34
segs2Rle	35
segs2RleDataFrame	36
segTable	36
space	37
subsetAssayData	39
toGenomeOrder	40

<i>genoset-package</i>	3
universe	40
Index	42

<i>genoset-package</i>	<i>GenoSet: An eSet for data with genome locations</i>
------------------------	--

Description

Load, manipulate, and plot copynumber and BAF data. *GenoSet* class extends *eSet* by adding a "locData" slot for a *RangedData* object from the *IRanges* package. This object contains feature genome location data and provides for simple subsetting on genome location. *CNSet* and *BAFSet* extend *GenoSet* and require *assayData* matrices for Copy Number (cn) or Log-R Ratio (lrr) and B-Allele Frequency (baf) data. Implements and provides convenience functions for processing of copy number and B-Allele Frequency data.

See Also

genoset-datasets *GenoSet* *CNSet* *BAFSet*

<i>asFactorMatrix</i>	<i>Make factor matrix from character matrix</i>
-----------------------	---

Description

Make factor matrix from character matrix for use with *convertToBigMatrix*. Makes an integer matrix with levels since *as.big.matrix* would make a factor matrix into a 1D object for some reason. Character matrices should be converted to factors with explicit levels as huge matrices are likely too big to unique.

Usage

```
asFactorMatrix(object, levels)
```

Arguments

object	matrix of characters
levels	character

Details

Caution: use *asFactorMatrix* on matrices already in an *eSet*. The *eSet* constructor will apparently wipe out the levels.

Value

factor with dimensions matching object

Author(s)

Peter M. Haverty <phaverty@gene.com>

assayDataElement *Get assayDataElement, attaching on-disk resource if necessary*

Description

Get assayDataElement, attaching on-disk resource if necessary

Usage

```
assayDataElement(object, elt)
```

Arguments

object	eSet
elt	character

Value

assayDataElement, matrix, DataFrame, or the like

Author(s)

Peter M. Haverty <phaverty@gene.com>

assayDataElement<- *Set assayDataElement, attaching on-disk resource if necessary*

Description

Set assayDataElement, attaching on-disk resource if necessary

Usage

```
assayDataElement(object, elt) <- value
```

Arguments

object	eSet
elt	character, assayDataElement name
value	input data to assayDataElement

Value

eSet

Author(s)

Peter M. Haverty <phaverty@gene.com>

`attachAssayDataElements`*Attach on-disk matrices into assayData*

Description

GenoSet objects can hold big.matrix objects in their assayData slot environment. After re-loading the GenoSet from disk, these objects will each need to be re-attached to their on-disk component using their resource locators stored in their "desc" attributes. This function checks each assayDataElement to see if it is an un-attached big.matrix object, re-attaching if necessary. All other assayDataElements are left untouched. In later releases this function will also handle other on-disk types, like HDF5-based matrices.

Usage

```
attachAssayDataElements(object)
```

Arguments

object eSet

Details

*** Intentional side-effects *** Environment type assayData objects, even "lockedEnvironment" objects, will be updated in place (same pointer). This allows for functions trying to access assayDataElements to attach before access, rather than crashing R.

Value

assayData in storage mode of input assayData, invisibly. Re-assignment back original eSet only necessary if using a list type assayData.

Author(s)

Peter M. Haverty <phaverty@gene.com>

`baf`*Get or Set the baf assayData slot*

Description

Get or Set the baf assayData slot

Arguments

object A BAFset object

Value

matrix

Author(s)

Peter M. Haverty

Examples

```
data(genoset)
baf(baf.ds) # Returns assayDataElement called "baf"
baf(baf.ds) <- baf2mbaf( baf(baf.ds) )
```

baf2mbaf

*Calculate mBAF from BAF***Description**

Calculate Mirrored B-Allele Frequency (mBAF) from B-Allele Frequency (BAF) as in Staaf et al., Genome Biology, 2008. BAF is converted to mBAF by folding around 0.5 so that is then between 0.5 and 1. HOM values are then made NA to leave only HET values that can be easily segmented. Values > hom.cutoff are made NA. Then, if genotypes (usually from a matched normal) are provided as the matrix 'calls' additional HOMs can be set to NA. The argument 'call.pairs' is used to match columns in 'calls' to columns in 'baf'.

Usage

```
baf2mbaf(baf, hom.cutoff = 0.95, calls = NULL,
         call.pairs = NULL)
```

Arguments

baf	numeric matrix of BAF values
hom.cutoff	numeric, values above this cutoff to be made NA (considered HOM)
calls	matrix of NA, CT, AG, etc. genotypes to select HETs (in normals). Dimnames must match baf matrix.
call.pairs	list, names represent target samples for HOMs to set to NA. Values represent columns in "calls" matrix.

Value

numeric matrix of mBAF values

Author(s)

Peter M. Haverty

Examples

```
data(genoset)
mbaf = baf2mbaf( baf(baf.ds), hom.cutoff=0.9 )
calls = matrix(sample(c("AT","AA","CG","GC","AT","GG"),(nrow(baf.ds) * 2),replace=TRUE),ncol=2,dimnames=
mbaf = baf2mbaf( baf(baf.ds), hom.cutoff=0.9, calls = calls, call.pairs = list(K="L",L="L") ) # Sample l
assayDataElement(baf.ds,"mbaf") = baf2mbaf( baf(baf.ds), hom.cutoff=0.9 ) # Put mbaf back into the BAFS
```

Description

This function is the preferred method for creating a new BAFSet object. Users are generally discouraged from calling "new" directly. This BAFSet function enforces the requirement for "lrr" and "baf" matrices. These and any other "..." arguments will become part of the assayData slot of the resulting object. "..." can be matrices or DataFrame objects (from the IRanges package). This function passes control to the "initGenoSet" method which performs argument checking including dimname matching among relevant slots and sets everything to genome order. Genome order can be disrupted by "[" or "[[" calls and will be checked by methods that require it.

Usage

```
BAFSet(locData, lrr = NULL, baf = NULL, pData = NULL,  
       annotation = "", universe = NULL, assayData = NULL,  
       ...)
```

Arguments

locData	A RangedData object specifying feature chromosome locations. Rownames are required to match featureNames.
lrr	numeric matrix of copy number data with rownames matching sampleNames and colnames matching sampleNames
baf	numeric matrix of B-Allele Frequency data with rownames matching sampleNames and colnames matching sampleNames
pData	A data frame with rownames matching all data matrices
annotation	character, string to specify chip/platform type
universe	character, a string to specify the genome universe for locData
assayData	assayData, usually an environment
...	More matrix or DataFrame objects to include in assayData slot

Value

A BAFSet object

Author(s)

Peter M. Haverty

See Also

bafset-class, genoset-class

Examples

```

test.sample.names = LETTERS[11:13]
probe.names = letters[1:10]
locData.rd = RangedData(ranges=IRanges(start=c(1,4,3,2,5:10),width=1,names=probe.names),space=c(rep("chr1",5)))
bs = BAFSet(
  locData=locData.rd,
  lrr=matrix(1:30,nrow=10,ncol=3,dimnames=list(probe.names,test.sample.names)),
  baf=matrix(31:60,nrow=10,ncol=3,dimnames=list(probe.names,test.sample.names)),
  pData=data.frame(matrix(LETTERS[1:15],nrow=3,ncol=5,dimnames=list(test.sample.names,letters[1:5]))),
  annotation="SNP6"
)

```

bafset-class

BAFSet class

Description

A BAFSet is an extension of GenoSet that requires 'baf' and 'lrr' assayData element

Extends

[GenoSet](#)

Author(s)

Peter M. Haverty

See Also

[bafset-class](#), [cnset-class](#)

Examples

```

## Creating a BAFSet
test.sample.names = LETTERS[11:13]
probe.names = letters[1:10]
locData.rd = RangedData(ranges=IRanges(start=c(1,4,3,2,5:10),width=1,names=probe.names),space=c(rep("chr1",5)))
bs = BAFSet(
  locData=locData.rd,
  lrr=matrix(1:30,nrow=10,ncol=3,dimnames=list(probe.names,test.sample.names)),
  baf=matrix(31:60,nrow=10,ncol=3,dimnames=list(probe.names,test.sample.names)),
  pData=data.frame(matrix(LETTERS[1:15],nrow=3,ncol=5,dimnames=list(test.sample.names,letters[1:5]))),
  annotation="SNP6"
)

```

 BAFSet.to.ExpressionSets

Make a pair of ExpressionSets from a BAFSet

Description

Often it is convenient to have a more standard "ExpressionSet" rather than a BAFSet. For example, when using infrastructure dependent on the ExpressionSet slots, like limma or ExpressionSetOnDisk. This will create a list of two ExpressionSets, one each for the baf and lrr data. To make a single ExpressionSet, with the lrr data in the exprs slot and the baf data as an additional member of assayData, use the standard coercion `eset = as(bafset,"ExpressionSet")`.

Usage

```
BAFSet.to.ExpressionSets(bs)
```

Arguments

`bs` A BAFset object

Value

A list with one ExpressionSet each for the baf and lrr data in the BAFSet object

Author(s)

Peter M. Haverty

Examples

```
data(genoset)
eset.list = BAFSet.to.ExpressionSets(baf.ds)
```

 boundingIndices

Find indices of features bounding a set of chromosome ranges/genes

Description

This function is similar to `findOverlaps` but it guarantees at least two features will be covered. This is useful in the case of finding features corresponding to a set of genes. Some genes will fall entirely between two features and thus would not return any ranges with `findOverlaps`. Specifically, this function will find the indices of the features (first and last) bounding the ends of a range/gene (start and stop) such that $first \leq start < stop \leq last$. Equality is necessary so that multiple conversions between indices and genomic positions will not expand with each conversion. Ranges/genes that are outside the range of feature positions will be given the indices of the corresponding first or last index rather than 0 or $n + 1$ so that genes can always be connected to some data.

Usage

```
boundingIndices(starts, stops, positions,
  valid.indices = TRUE, all.indices = FALSE, offset = 0)
```

Arguments

starts	integer vector of first base position of each query range
stops	integer vector of last base position of each query range
positions	Base positions in which to search
valid.indices	logical, TRUE assures that the returned indices don't go off either end of the array, i.e. 0 becomes 1 and n+1 becomes n
offset	integer, value to add to all returned indices. For the case where positions represents a portion of some larger array (e.g. a chr in a genome)
all.indices	logical, return a list containing full sequence of indices for each query

Details

This function uses some tricks from `findIntervals`, where is for k queries and n features it is $O(k * \log(n))$ generally and $\sim O(k)$ for sorted queries. Therefore will be dramatically faster for sets of query genes that are sorted by start position within each chromosome. The index of the stop position for each gene is found using the left bound from the start of the gene reducing the search space for the stop position somewhat. This function has important differences from `boundingIndices2`, which uses `findInterval`: `boundingIndices` does not check for NAs or unsorted data in the subject positions. Also, the positions are kept as integer, where `boundingIndices2` (and `findInterval`) convert them to doubles. These three once-per-call differences account for much of the speed improvement in `boundingIndices`. These three differences are meant for position info coming from `GenoSet` objects and `boundingIndices2` is safer for general use. `boundingIndices` works on integer positions and does not check that the positions are ordered. The starts and stops need not be sorted, but it will be much faster if they are.

Value

integer matrix of 2 columns for start and stop index of range in data or a list of full sequences of indices for each query (see `all.indices` argument)

Author(s)

Peter M. Haverty <phaverty@gene.com>

See Also

Other "range summaries": [boundingIndices2](#), [boundingIndicesByChr](#), [rangeColMeans](#), [rangeSampleMeans](#)

Examples

```
starts = seq(10,100,10)
boundingIndices( starts=starts, stops=starts+5, positions = 1:100 )
```

 boundingIndices2

Find indices of features bounding a set of chromosome ranges/genes

Description

This function is similar to `findOverlaps` but it guarantees at least two features will be covered. This is useful in the case of finding features corresponding to a set of genes. Some genes will fall entirely between two features and thus would not return any ranges with `findOverlaps`. Specifically, this function will find the indices of the features (first and last) bounding the ends of a range/gene (start and stop) such that $\text{first} \leq \text{start} \leq \text{stop} \leq \text{last}$. Equality is necessary so that multiple conversions between indices and genomic positions will not expand with each conversion. This function uses `findIntervals`, which is for k queries and n features is $O(k * \log(n))$ generally and $\sim O(k)$ for sorted queries. Therefore will be dramatically faster for sets of query genes that are sorted by start position within each chromosome. This should give performance for k genes and n features that is $\sim O(k)$ for starts and $O(k * \log(n))$ for stops and $\sim O(k * \log(n))$ overall. Ranges/genes that are outside the range of feature positions will be given the indices of the corresponding first or last index rather than 0 or $n + 1$ so that genes can always be connected to some data.

Usage

```
boundingIndices2(starts, stops, positions, offset = NULL)
```

Arguments

<code>starts</code>	numeric or integer, first base position of each query range
<code>stops</code>	numeric or integer, last base position of each query range
<code>positions</code>	Base positions in which to search
<code>offset</code>	integer, value to add to all returned indices. For the case where <code>positions</code> represents a portion of some larger array (e.g. a chr in a genome)

Value

integer matrix of 2 columns for start and stop index of range in data

Author(s)

Peter M. Haverty

See Also

Other "range summaries": [boundingIndices](#), [boundingIndicesByChr](#), [rangeColMeans](#), [rangeSampleMeans](#)

Examples

```
starts = seq(10,100,10)
boundingIndices2( starts=starts, stops=starts+5, positions = 1:100 )
```

boundingIndicesByChr *Find indices of features bounding a set of chromosome ranges/genes, across chromosomes*

Description

Finds subject ranges corresponding to a set of genes (query ranges), taking chromosome into account. Specifically, this function will find the indices of the features (first and last) bounding the ends of a range/gene (start and stop) such that $\text{first} \leq \text{start} < \text{stop} \leq \text{last}$. Equality is necessary so that multiple conversions between indices and genomic positions will not expand with each conversion. Ranges/genes that are outside the range of feature positions will be given the indices of the corresponding first or last index on that chromosome, rather than 0 or $n + 1$ so that genes can always be connected to some data. Checking the left and right bound for equality will tell you when a query is off the end of a chromosome.

Usage

```
boundingIndicesByChr(query, subject)
```

Arguments

query	GRanges or something coercible to GRanges
subject	RangedData

Details

This function uses some tricks from `findIntervals`, where is for k queries and n features it is $O(k * \log(n))$ generally and $\sim O(k)$ for sorted queries. Therefore will be dramatically faster for sets of query genes that are sorted by start position within each chromosome. The index of the stop position for each gene is found using the left bound from the start of the gene reducing the search space for the stop position somewhat. This function has important differences from `boundingIndices2`, which uses `findInterval`: `boundingIndices` does not check for NAs or unsorted data in the subject positions. Also, the positions are kept as integer, where `boundingIndices2` (and `findInterval`) convert them to doubles. These three once-per-call differences account for much of the speed improvement in `boundingIndices`. These three differences are meant for position info coming from `GenoSet` objects and `boundingIndices2` is safer for general use. `boundingIndices` works on integer positions and does not check that the positions are ordered. The starts and stops need not be sorted, but it will be much faster if they are.

This function differs from `boundingIndices` in that 1. it uses both start and end positions for the subject, and 2. query and subject start and end positions are processed in blocks corresponding to chromosomes.

Value

integer matrix with two columns corresponding to indices on left and right bound of queries in subject

Author(s)

Peter M. Haverty <phaverty@gene.com>

See Also

Other "range summaries": [boundingIndices](#), [boundingIndices2](#), [rangeColMeans](#), [rangeSampleMeans](#)

chr *Look up chromosome for each feature*

Description

Chromosome name for each feature

Arguments

object GRanges, RangedData or GenoSet

Details

Get chromosome name for each feature. Returns character, not the factor 'space'.

Value

character vector of chromosome positions for each feature

Author(s)

Peter Haverty

Examples

```
test.sample.names = LETTERS[11:13]
probe.names = letters[1:10]
gs = GenoSet(
  locData=RangedData(ranges=IRanges(start=1:10,width=1,names=probe.names),space=c(rep("chr1",4),rep("chr2",4)),
  cn=matrix(31:60,nrow=10,ncol=3,dimnames=list(probe.names,test.sample.names)),
  pData=data.frame(matrix(LETTERS[1:15],nrow=3,ncol=5,dimnames=list(test.sample.names,letters[1:5]))),
  annotation="SNP6"
)
chr(gs) # c("chr1","chr1","chr1","chr1","chr3","chr3","chrX","chrX","chrX","chrX")
chr(locData(gs)) # The same
```

chrIndices *Get a matrix of first and last index of features in each chromosome*

Description

Sometimes it is handy to know the first and last index for each chr. This is like chrInfo but for feature indices rather than chromosome locations. If chr is specified, the function will return a sequence of integers representing the row indices of features on that chromosome.

Arguments

object GenoSet, RangedData, or GRanges
chr character, specific chromosome name

Value

data.frame with "first" and "last" columns

Author(s)

Peter M. Haverty

Examples

```
data(genoset)
chrIndices(genoset.ds)
chrIndices(locData(genoset.ds)) # The same
```

chrInfo

Chromosome Information

Description

Get chromosome start and stop positions

Arguments

object A GenoSet object or similar

Details

Provides a matrix of start, stop and offset, in base numbers for each chromosome.

Value

list with start and stop position, by ordered chr

Author(s)

Peter Haverty

Examples

```
data(genoset)
chrInfo(genoset.ds)
chrInfo(locData(genoset.ds)) # The same
```

chrNames	<i>Get list of unique chromosome names</i>
----------	--

Description

Get list of unique chromosome names

Arguments

object RangedData or GenoSet

Value

character vector with names of chromosomes

Author(s)

Peter M. Haverty

Examples

```
test.sample.names = LETTERS[11:13]
probe.names = letters[1:10]
gs = GenoSet(
  locData=RangedData(ranges=IRanges(start=1:10,width=1,names=probe.names),space=c(rep("chr1",4),rep("chr2",4),rep("chr3",2)),
  cn=matrix(31:60,nrow=10,ncol=3,dimnames=list(probe.names,test.sample.names)),
  pData=data.frame(matrix(LETTERS[1:15],nrow=3,ncol=5,dimnames=list(test.sample.names,letters[1:5]))),
  annotation="SNP6"
)
chrNames(gs) # c("chr1","chr3","chrX")
chrNames(locData(gs)) # The same
```

chrOrder	<i>Order chromosome names in proper genome order</i>
----------	--

Description

Chromosomes make the most sense orded by number, then by letter.

Usage

```
chrOrder(chr.names)
```

Arguments

chr.names character, vector of unique chromosome names

Value

character vector of chromosome names in proper order

Author(s)

Peter M. Haverty

See Also

Other "genome ordering": [isGenomeOrder](#), [isGenomeOrder](#), [isGenomeOrder](#), [toGenomeOrder](#), [toGenomeOrder](#), [toGenomeOrder](#), [toGenomeOrder](#)

Examples

```
chrOrder(c("chr5", "chrX", "chr3", "chr7", "chrY")) # c("chr3", "chr5", "chr7", "chrX", "chrY")
```

cn	<i>Get or Set the cn assayData slot</i>
----	---

Description

Get or Set the cn assayData slot

Arguments

object	A BAFset object
--------	-----------------

Value

matrix

Author(s)

Peter M. Haverty

Examples

```
data(genoset)
cn(cn.ds) # Returns assayDataElement called "cn"
cn(cn.ds) <- cn(cn.ds) + 5
```

CNSet	<i>Create a CNSet object</i>
-------	------------------------------

Description

This function is the preferred method for creating a new CNSet object. Users are generally discouraged from calling "new" directly. This CNSet function enforces the requirement for a "cn" matrix. This and any other "..." arguments will become part of the assayData slot of the resulting object. "..." can be matrices or DataFrame objects (from the IRanges package). This function passes control to the "initGenoSet" method which performs argument checking including dimname matching among relevant slots and sets everything to genome order. Genome order can be disrupted by "[" or "[[" calls and will be checked by methods that require it.

Usage

```
CNSet(locData, cn = NULL, pData = NULL, annotation = "",
      universe = NULL, assayData = NULL, ...)
```

Arguments

locData	A RangedData object specifying feature chromosome locations. Rownames are required to match featureNames.
cn	numeric matrix of copy number data with rownames matching sampleNames and colnames matching sampleNames
pData	A data frame with rownames matching all data matrices
annotation	character, string to specify chip/platform type
universe	character, string to specify genome universe for locData
assayData	assayData, usually an environment
...	More matrix or DataFrame objects to include in assayData

Value

A CNSet object

Author(s)

Peter M. Haverty

Examples

```
test.sample.names = LETTERS[11:13]
probe.names = letters[1:10]
joe = CNSet(
  locData=RangedData(ranges=IRanges(start=1:10,width=1,names=probe.names),space=c(rep("chr1",4),rep("chr3",3)),
  cn=matrix(31:60,nrow=10,ncol=3,dimnames=list(probe.names,test.sample.names)),
  pData=data.frame(matrix(LETTERS[1:15],nrow=3,ncol=5,dimnames=list(test.sample.names,letters[1:5]))),
  annotation="SNP6"
)
```

cnset-class

CNSet class

Description

A CNSet is an extension of GenoSet that requires a 'cn' assayData element.

Extends

[GenoSet](#)

Author(s)

Peter M. Haverty

See Also

[bafset-class](#), [cnset-class](#)

Examples

```
test.sample.names = LETTERS[11:13]
probe.names = letters[1:10]
cn.ds = CNSet(
  locData=RangedData(ranges=IRanges(start=1:10,width=1,names=probe.names),space=c(rep("chr1",4),rep("chr3",3)),
  cn=matrix(31:60,nrow=10,ncol=3,dimnames=list(probe.names,test.sample.names)),
  pData=data.frame(matrix(LETTERS[1:15],nrow=3,ncol=5,dimnames=list(test.sample.names,letters[1:5]))),
  annotation="SNP6"
)
```

colMeans

Means of columns

Description

Calculate means of columns of a `DataFrame` as if it were a matrix. Allow `colmeans` in `rangeSampleMeans` for `DataTable` just like a real matrix. I'm sure there is much more clever way to do this using `aggregate`.

Arguments

<code>x</code>	<code>DataFrame</code>
<code>na.rm</code>	logical
<code>dims</code>	integer

Author(s)

Peter M. Haverty

Examples

```
df.ds = DataFrame( a = Rle(c(5,4,3),c(2,2,2)), b = Rle(c(3,6,9),c(1,1,4)) )
mat.ds = matrix( c(5,5,4,4,3,3,3,6,9,9,9,9), ncol=2, dimnames=list(NULL,c("a","b")))
## Not run: identical( colMeans(df.ds), colMeans(mat.ds) )
```

convertToBigMatrix *Make standard matrices in a GenoSet filebacked bigmatrix objects*

Description

Make standard matrices in a GenoSet filebacked bigmatrix objects. Something like a factor can be obtained using integer assayDataElements with a "levels" attribute. The levels attribute will be maintained. Such objects will be stored as char on disk if there are < 128 levels, and integer otherwise. "nlevels" and "levels" will work on these objects as they only require the levels attribute. The "as.character" functionality of a factor can be obtained like this: levels(assayDataElement(ds,"geno"))[ds[1:5,1:5,"geno"]] for a GenoSet called "ds" with a factor-like element called "geno".

Usage

```
convertToBigMatrix(object, prefix = "bigmat",
  path = "bigmat")
```

Arguments

object	GenoSet
prefix	character, prefix for all bigmatrix related files
path	character, directory to be created for all bigmatrix files, can be pre-existing.

Value

GenoSet or related, updated copy of "object"

Author(s)

Peter M. Haverty <phaverty@gene.com>

Examples

```
## Not run: ds = convertToBigMatrix(ds)
```

featureNames<- *Set featureNames*

Description

Set featureNames

Arguments

object	GenoSet
value	ANY

Details

Set featureNames including rownames of position info

Value

A new object of the class of supplied object

Author(s)

Peter M. Haverty

gcCorrect

cgCorrect

Description

Correct copy number for GC content

Usage

```
gcCorrect(ds, gc, retain.mean = TRUE)
```

Arguments

ds	numeric matrix of copynumber or log2ratio values, samples in columns
gc	numeric vector, GC percentage for each row of ds, must not have NAs
retain.mean	logical, center on zero or keep same mean?

Details

Copy number estimates from various platforms show "Genomic Waves" (Diskin et al., Nucleic Acids Research, 2008) where copy number trends with local GC content. This function regresses copy number on GC percentage and removes the effect (returns residuals). GC content should be smoothed along the genome in wide windows ≥ 100 kb.

Value

numeric matrix, residuals of ds regressed on gc

Author(s)

Peter M. Haverty

See Also

Other "gc content": [loadGC](#), [loadGC](#), [loadGC](#)

Examples

```
gc = runif(n=100, min=1, max=100)
ds = rnorm(100) + (0.1 * gc)
gcCorrect(ds, gc)
```

genomeAxis	<i>Label axis with base pair units</i>
------------	--

Description

Label an axis with base positions

Usage

```
genomeAxis(locs = NULL, side = 1, log = FALSE,
           do.other.side = TRUE)
```

Arguments

locs	RangedData to be used to draw chromosome boundaries, if necessary. Usually locData slot from a GenoSet.
side	integer side of plot to put axis
log	logical Is axis logged?
do.other.side	logical, label non-genome side with data values at tick marks?

Details

Label a plot with Mb, kb, bp as appropriate, using tick locations from axTicks

Value

nothing

Author(s)

Peter M. Haverty

See Also

Other "genome plots": [genoPlot](#), [genoPlot](#), [genoPlot](#), [genoPlot](#), [genoPlot](#), [genoPlot](#)

Examples

```
data(genoset)
  genoPlot(genoPos(baf.ds), baf(baf.ds)[,1])
  genomeAxis( locs=locData(baf.ds) ) # Add chromosome names and boundaries to a plot assuming genome along
  genomeAxis( locs=locData(baf.ds), do.other.side=FALSE ) # As above, but do not label y-axis with data va
  genomeAxis() # Add nucleotide position in sensible units assuming genome along x-axis
```

 genoPlot

Plot data along the genome

Description

For a `GenoSet` object, data for a specified sample in a specified `assayDataElement` can be plotted along the genome. One chromosome can be specified if desired. If more than one chromosome is present, the chromosome boundaries will be marked. Alternatively, for a numeric `x` and a numeric or `Rle` `y`, data in `y` can be plotted at genome positions `y`. In this case, chromosome boundaries can be taken from the argument `locs`. If data for `y`-axis comes from a `Rle`, either specified directly or coming from the specified `assayData` element and sample, lines are plotted representing segments.

Arguments

<code>sample</code>	A index or <code>sampleName</code> to plot
<code>element</code>	character, name of element in <code>assayData</code> to plot
<code>x</code>	<code>GenoSet</code> (or descendant) or numeric with chromosome or genome positions
<code>y</code>	numeric or <code>Rle</code> , values to be used for <code>y</code> -dimension, run start and stop indices or numeric with all values mapped to values in <code>x</code> for <code>x</code> -dimension or index of sample to be plotted if <code>x</code> is a <code>GenoSet</code> .
<code>element</code>	character, when <code>x</code> is a <code>GenoSet</code> , the name of the <code>assayDataElement</code> to plot from.
<code>locs</code>	<code>RangedData</code> , like <code>locData</code> slot of <code>GenoSet</code>
<code>chr</code>	Chromosome to plot, <code>NULL</code> by default for full genome
<code>add</code>	Add plot to existing plot
<code>xlab</code>	character, label for <code>x</code> -axis of plot
<code>ylab</code>	character, label for <code>y</code> -axis of plot
<code>col</code>	character, color to plot lines or points
<code>lwd</code>	numeric, line width for segment plots from an <code>Rle</code>
<code>pch</code>	character or numeric, printing character, see points
<code>...</code>	Additional plotting args

Value

nothing

Author(s)

Peter M. Haverty
 Peter M. Haverty

See Also

Other "genome plots": [genomeAxis](#)

Examples

```
data(genoset)
genoPlot( baf.ds,1,element="lrr")
genoPlot( genoPos(baf.ds), assayDataElement(baf.ds,"lrr")[,1], locs=locData(baf.ds) ) # The same
genoPlot( 1:10, Rle(c(rep(0,5),rep(3,4),rep(1,1))) )
```

genoPos	<i>Convert chromosome positions to positions from start of genome</i>
---------	---

Description

Get base positions of features in genome-scale units

Arguments

object A GenoSet object or a RangedData object

Details

Get base positions of array features in bases counting from the start of the genome. Chromosomes are ordered numerically, when possible, then lexically.

Value

numeric position of each feature in whole genome units, in original order

Author(s)

Peter M. Haverty

Examples

```
data(genoset)
genoPos(genoset.ds)
genoPos(locData(genoset.ds)) # The same
```

GenoSet	<i>Create a GenoSet object</i>
---------	--------------------------------

Description

This function is the preferred method for creating a new GenoSet object. Users are generally discouraged from calling "new" directly. Any "..." arguments will become part of the assayData slot of the resulting object. "..." can be matrices or DataFrame objects (from IRanges). This function passes control to the "initGenoSet" method which performs argument checking including dimname matching among relevant slots and sets everything to genome order. Genome order can be disrupted by "[" calls and will be checked by methods that require it.

Usage

```
GenoSet(locData, pData = NULL, annotation = "",
        universe = NULL, assayData = NULL, ...)
```

Arguments

locData	A RangedData object specifying feature chromosome locations. Rownames are required to match featureNames.
pData	A data frame with rownames matching all data matrices
annotation	character, string to specify chip/platform type
universe	character, a string to specify the genome universe for locData
assayData	assayData, usually an environment
...	More matrix or DataFrame objects to include in assayData

Value

A GenoSet object

Author(s)

Peter M. Haverty

Examples

```
test.sample.names = LETTERS[11:13]
probe.names = letters[1:10]
gs = GenoSet(
  locData=RangedData(ranges=IRanges(start=1:10,width=1,names=probe.names),space=c(rep("chr1",4),rep("chr3",3)),
  cn=matrix(31:60,nrow=10,ncol=3,dimnames=list(probe.names,test.sample.names)),
  pData=data.frame(matrix(LETTERS[1:15],nrow=3,ncol=5,dimnames=list(test.sample.names,letters[1:5]))),
  annotation="SNP6"
)
```

genoset-class

GenoSet class

Description

The genoset package offers an extension of the BioConductor eSet object for genome arrays. The package offers three classes. The first class is the GenoSet class which can hold an arbitrary number of equal-sized matrices in its assayData slot. The principal addition of the GenoSet class is a locData slot that holds a RangedData object from the IRanges package. The locData slot allows for quick subsetting by genome position.

Two classes extend GenoSet: CNSet and BAFSet. CNSet is the basic copy number object. It keeps its data in the cn slot, similar to the exprs slot of the ExpressionSet. BAFSet is intended to store LRR or Log-R Ratio and BAF or B-Allele Frequency data for SNP arrays. LRR and BAF come from the terms coined by Illumina. LRR is copynumber data processed on a per-snp basis to remove some variability using the expected log-ratio of normal samples with the same genotype. BAF represents the fraction of signal coming from the "B" allele, relative to the "A" allele, where A and B are arbitrarily assigned. BAF has the expected value of 0 or 1 for HOM alleles and 0.5 for HET alleles. Deviation from these expected values can be interpreted as Allelic Imbalance, which is a sign of gain, loss, or copy-neutral LOH.

Slots

locData: ([RangedData](#)) Contains a RangedData that holds probe locations

Extends

[eSet](#)

Author(s)

Peter M. Haverty

See Also

[bafset-class](#), [cnset-class](#)

Examples

```
## Creating a GenoSet
test.sample.names = LETTERS[11:13]
probe.names = letters[1:10]
gs = GenoSet(
  locData=RangedData(ranges=IRanges(start=1:10,width=1,names=probe.names),space=c(rep("chr1",4),rep("chr3",3)),
  cn=matrix(31:60,nrow=10,ncol=3,dimnames=list(probe.names,test.sample.names)),
  pData=data.frame(matrix(LETTERS[1:15],nrow=3,ncol=5,dimnames=list(test.sample.names,letters[1:5]))),
  annotation="SNP6"
)
```

genoset-datasets

Example GenoSet, BAFSet, and CNSet objects and the data to create them.

Description

Fake LRR, BAF, pData and location data were generated and saved as fake.lrr, fake.baf, fake.pData and locData.rd. These were used to construct the objects genoset.ds, baf.ds, and cn.ds

Usage

```
data(genoset)
```

Format

fake.lrr A matrix with some randomly generated LRR (log2ratio copynumber) data

fake.baf A matrix with some randomly generated BAF (B-Allele Frequency) data

fake.pData A data.frame of sample annotation to go with fake.lrr and fake.baf

locData.rd A RangedData object describing the genomic locations of the probes in fake.baf and fake.lrr

genoset.ds A GenoSet object created with fake.lrr as the "lrr" element, locData.rd as the locData, and fake.pData as the phenoData

baf.ds A BAFSet object created with fake.lrr as the "lrr" element, fake.baf as the "baf" element, locData.rd as the locData, and fake.pData as the phenoData

cn.ds A CNSet object created with fake.lrr as the "cn" element, locData.rd as the locData, and fake.pData as the phenoData

Source

Fake data generated using rnorm and the like.

initGenoSet	<i>Create a GenoSet or derivative object</i>
-------------	--

Description

This function is the preferred method for creating a new GenoSet object. Users are generally discouraged from calling "new" directly. The "..." argument is for any number of matrices of matching size that will become part of the assayData slot of the resulting object. This function passes control to the "genoSet" object which performs argument checking including dimname matching among relevant slots and sets everything to genome order. Genome order can be disrupted by "[" calls and will be checked by methods that require it.

Usage

```
initGenoSet(type, locData, pData = NULL, annotation = "",
            universe = NULL, assayData = NULL, ...)
```

Arguments

type	character, the type of object (e.g. GenoSet, BAFSet, CNSet) to be created
locData	A RangedData object specifying feature chromosome locations. Rownames are required to match featureNames.
pData	A data frame with rownames matching all data matrices
annotation	character, string to specify chip/platform type
universe	character, a string to specify the genome universe for locData
assayData	assayData, usually an environment
...	More matrix or DataFrame objects to include in assayData

Value

A GenoSet object or derivative as specified by "type" arg

Author(s)

Peter M. Haverty

Examples

```
test.sample.names = LETTERS[11:13]
probe.names = letters[1:10]
gs = GenoSet(
  locData=RangedData(ranges=IRanges(start=1:10,width=1,names=probe.names),space=c(rep("chr1",4),rep("chr2",4)),
  cn=matrix(31:60,nrow=10,ncol=3,dimnames=list(probe.names,test.sample.names)),
  pData=data.frame(matrix(LETTERS[1:15],nrow=3,ncol=5,dimnames=list(test.sample.names,letters[1:5]))),
  annotation="SNP6"
)
```

isGenomeOrder	<i>Check if a GRanges, GenoSet or RangedData is in genome order</i>
---------------	---

Description

Checks that rows in each chr are ordered by start. If strict=TRUE, then chromosomes must be in order specified by chrOrder. isGenomeOrder for GRanges differs from order in that it orders by chromosome and start position only, rather than chromosome, strand, start, and width.

Arguments

ds	GenoSet, GRanges, or RangedData
strict	logical, should space/chromosome order be identical to that from chrOrder?

Value

logical

Author(s)

Peter M. Haverty

See Also

Other "genome ordering": [chrOrder](#), [toGenomeOrder](#), [toGenomeOrder](#), [toGenomeOrder](#), [toGenomeOrder](#)

Examples

```
data(genoset)
isGenomeOrder( locData(genoset.ds) )
```

loadGC	<i>Load local GC percentage around features</i>
--------	---

Description

Local GC content can be used to remove GC artifacts from copynumber data see Diskin, 2008). GC added to the feature data. The dataset may be truncated to remove positions without GC information. GC data are accessible with locData(). Uses a cool BSgenome trick from Michael Lawrence. This takes 5.6 hours for 2Mb windows on 2.5M probes, so look for some custom C in future releases.

Arguments

object	A GenoSet object or derivative
expand	numeric, expand each feature location by this many bases on each side
bsgenome,	sequence db object from BSgenome (e.g. Hsapiens)

Value

An updated object, with GC percentage information added to the locData slot.

Author(s)

Peter M. Haverty

See Also

Other "gc content": [gcCorrect](#)

 locData

Get and set probe set info

Description

Access the feature genome position info

Set locData

Arguments

object	GenoSet
object	A GenoSet object
object	GenoSet
value	RangedData describing features

Details

The position information for each probe/feature is stored as an IRanges RangedData object. The locData functions allow this data to be accessed or re-set.

Set locData

Value

A GenoSet object

Author(s)

Peter M. Haverty

Peter Haverty

Examples

```
data(genoset)
rd = locData(genoset.ds)
locData(genoset.ds) = rd
```

lrr *Get or Set the lrr assayData slot*

Description

Get or Set the lrr assayData slot

Arguments

object A BAFset object

Value

matrix

Author(s)

Peter M. Haverty

Examples

```
data(genoset)
lrr(baf.ds) # Returns assayDataElement called "lrr"
lrr(baf.ds) <- lrr(baf.ds) + 0.1
```

modeCenter *Center continuous data on mode*

Description

Copynumber data distributions are generally multi-modal. It is often assumed that the tallest peak represents "normal" and should therefore be centered on a log2ratio of zero. This function uses the density function to find the mode of the dominant peak and subtracts that value from the input data.

Usage

```
modeCenter(ds)
```

Arguments

ds numeric matrix

Value

numeric matrix

Author(s)

Peter M. Haverty

Examples

```
modeCenter( matrix( rnorm(150, mean=0), ncol=3 ))
```

pos	<i>Positions for features</i>
-----	-------------------------------

Description

Chromosome position of features

Arguments

object GRanges, RangedData or GenoSet

Details

Get chromosome position of features/ranges. Defined as floor of mean of start and end.

Value

numeric vector of feature positions within a chromosome

Author(s)

Peter Haverty

Examples

```
test.sample.names = LETTERS[11:13]
probe.names = letters[1:10]
gs = GenoSet(
  locData=RangedData(ranges=IRanges(start=1:10,width=1,names=probe.names),space=c(rep("chr1",4),rep("chr2",4)),
  cn=matrix(31:60,nrow=10,ncol=3,dimnames=list(probe.names,test.sample.names)),
  pData=data.frame(matrix(LETTERS[1:15],nrow=3,ncol=5,dimnames=list(test.sample.names,letters[1:5]))),
  annotation="SNP6"
)
pos(gs) # 1:10
pos(locData(gs)) # The same
```

rangeColMeans	<i>Calculate column means for multiple ranges</i>
---------------	---

Description

Essentially colMeans with a loop, all in a .Call. Designed to take a 2-column matrix of row indices, bounds, for a matrix, x, and calculate mean for each range in each column (or along a single vector). bounds matrix need not cover all rows.

Usage

```
rangeColMeans(bounds, x)
```

Arguments

bounds	A two column integer matrix of row indices
x	A numeric matrix with rows corresponding to indices in bounds.

Value

A numeric matrix or vector, matching the form of x. One row for each row in bounds, one col for each col of x and appropriate dimnames. If x is a vector, just a vector with names from the rownames of bounds.

Author(s)

Peter M. Haverty <phaverty@gene.com>

See Also

Other "range summaries": [boundingIndices](#), [boundingIndices2](#), [boundingIndicesByChr](#), [rangeSampleMeans](#)

rangeSampleMeans	<i>Average features in ranges per sample</i>
------------------	--

Description

This function takes per-feature genomic data and returns averages for each of a set of genomic ranges. The most obvious application is determining the copy number of a set of genes. The features corresponding to each gene are determined with boundingIndices such that all features with the bounds of a gene (overlaps). The features on either side of the gene unless those positions exactly match the first or last base covered by the gene. Therefore, genes falling between two features will at least cover two features. This is similar to rangeSampleMeans, but it checks the subject positions for being sorted and not being NA and also treats them as doubles, not ints. Range bounding performed by the boundingIndices function.

Usage

```
rangeSampleMeans(query.rd, subject, assay.element)
```

Arguments

query.rd	RangedData object representing genomic regions (genes) to be averaged.
subject	A GenoSet object or derivative
assay.element	character, name of element in assayData to use to extract data

Value

numeric matrix of features in each range averaged by sample

Author(s)

Peter M. Haverty

See Also

Other "range summaries": [boundingIndices](#), [boundingIndices2](#), [boundingIndicesByChr](#), [rangeColMeans](#)

Examples

```
data(genoset)
my.genes = RangedData( ranges=IRanges(start=c(35e6,128e6),end=c(37e6,129e6),names=c("HER2","CMYC")), spa
rangeSampleMeans( my.genes, baf.ds, "lrr" )
```

readGenoSet	<i>Load a GenoSet from a RData file</i>
-------------	---

Description

Given a RData file with one object (a GenoSet or related object), load it, and return.

Usage

```
readGenoSet(path)
```

Arguments

path character, path to RData file

Value

GenoSet or related object (only object in RData file)

Author(s)

Peter M. Haverty <phaverty@gene.com>

Examples

```
## Not run: ds = readGenoSet("/path/to/genoset.RData")
```

relocateAssayData	<i>Update "desc" attributes for big.matrix assayDataElement to new location</i>
-------------------	---

Description

Update "desc" attributes for big.matrix assayDataElement to new location. Assumes files have already been moved on the filesystem. Assumes names of description and data files are the same.

Usage

```
relocateAssayData(ds, new.bigmat.dir)
```


Arguments

ds eSet
 new.bigmat.dir character, path to directory holding desc and data files

Value

eSet

Author(s)

Peter M. Haverty <phaverty@gene.com>

runCBS *Run CBS Segmentation*

Description

Utility function to run CBS's three functions on one or more samples

Usage

```
runCBS(data, locs, return.segs = FALSE, n.cores = 1,
        smooth.region = 2, outlier.SD.scale = 4,
        smooth.SD.scale = 2, trim = 0.025, alpha = 0.001)
```

Arguments

data numeric matrix with continuous data in one or more columns
 locs RangeData, like locData slot of GenoSet
 return.segs logical, if true list of segment data.frames return, otherwise a DataFrame of Rle vectors. One Rle per sample.
 n.cores numeric, number of cores to ask mclapply to use
 smooth.region number of positions to left and right of individual positions to consider when smoothing single point outliers
 outlier.SD.scale number of SD single points must exceed smooth.region to be considered an outlier
 smooth.SD.scale floor used to reset single point outliers
 trim fraction of sample to smooth
 alpha pvalue cutoff for calling a breakpoint

Details

Takes care of running CBS segmentation on one or more samples. Makes appropriate input, smooths outliers, and segment

Value

data frame of segments from CBS

Author(s)

Peter M. Haverty

See AlsoOther "segmented data": [segs2RangedData](#), [segs2Rle](#), [segs2RleDataFrame](#), [segTable](#), [segTable](#), [segTable](#)**Examples**

```

sample.names = paste("a",1:2,sep="")
probe.names = paste("p",1:30,sep="")
ds = matrix(c(c(rep(5,20),rep(3,10)),c(rep(2,10),rep(7,10),rep(9,10))),ncol=2,dimnames=list(probe.names,
locs = RangedData(ranges=IRanges(start=c(1:20,1:10),width=1,names=probe.names),space=paste("chr",c(rep(
seg.rle.result = DataFrame( a1 = Rle(c(rep(5,20),rep(3,10))), a2 = Rle(c(rep(2,10),rep(7,10),rep(9,10))),
seg.list.result = list(
  a1 = data.frame( ID=rep("a1",2), chrom=factor(c("chr1","chr2")), loc.start=c(1,1), loc.end=c(20,10),
  a2 = data.frame( ID=rep("a2",3), chrom=factor(c("chr1","chr1","chr2")), loc.start=c(1,11,1), loc.end=
)

runCBS(ds,locs) # Should give seg.rle.result
runCBS(ds,locs,return.segs=TRUE) # Should give seg.list.result

```

segs2RangedData

Make a RangedData from segments

Description

Starting from a data.frame of segments, like from CBS and segTable, organize as a RangedData. Label data "score", so it can easily be made into various genome browser formats using rtracklayer.

Usage

```
segs2RangedData(segs)
```

Arguments

segs data.frame, like from segment in DNACopy or segTable

Value

RangedData

Author(s)

Peter M. Haverty <phaverty@gene.com>

See Also

Other "segmented data": [runCBS](#), [segs2Rle](#), [segs2RleDataFrame](#), [segTable](#), [segTable](#), [segTable](#)

segs2Rle	<i>Make Rle from segments for one sample</i>
----------	--

Description

Take output of CBS, make Rle representing all features in 'locs' ranges. CBS output contains run length and run values for genomic segments, which could very directly be converted into a Rle. However, as NA values are often removed, especially for mBAF data, these run lengths do not necessarily cover all features in every sample. Using the start and top positions of each segment and the location of each feature, we can make a Rle that represents all features.

Usage

```
segs2Rle(segs, locs)
```

Arguments

segs	data.frame of segments, formatted as output of segment function from DNACopy package
locs	RangedData, like locData slot of a GenoSet

Value

Rle with run lengths and run values covering all features in the data set.

Author(s)

Peter M. Haverty <phaverty@gene.com>

See Also

Other "segmented data": [runCBS](#), [segs2RangedData](#), [segs2RleDataFrame](#), [segTable](#), [segTable](#), [segTable](#)

Examples

```
data(genoset)
segs = runCBS( lrr(baf.ds), locData(baf.ds), return.segs=TRUE )
segs2Rle( segs[[1]], locData(baf.ds) ) # Take a data.frame of segments, say from DNACopy's segment func
```

segs2RleDataFrame	<i>CBS segments to probe matrix</i>
-------------------	-------------------------------------

Description

Given segments, make a DataFrame of Rle objects for each sample

Usage

```
segs2RleDataFrame(seg.list, locs)
```

Arguments

seg.list	list, list of data frames, one per sample, each is result from CBS
locs	locData from a GenoSet object

Details

Take table of segments from CBS, convert DataTable of Rle objects for each sample.

Value

DataFrame of Rle objects with nrows same as locs and one column for each sample

Author(s)

Peter Haverty

See Also

Other "segmented data": [runCBS](#), [segs2RangedData](#), [segs2Rle](#), [segTable](#), [segTable](#), [segTable](#)

Examples

```
data(genoset)
seg.list = runCBS( lrr(baf.ds), locData(baf.ds), return.segs=TRUE )
segs2RleDataFrame( seg.list, locData(baf.ds) ) # Loop segs2Rle on list of data.frames in seg.list
```

segTable	<i>Convert Rle objects to tables of segments</i>
----------	--

Description

Like the inverse of segs2Rle and segs2RleDataFrame. Takes a Rle or a DataFrame with Rle columns and the locData RangedData both from a GenoSet object and make a list of data.frames each like the result of CBS's segment. Note the loc.start and loc.stop will correspond exactly to probe locations in locData and the input to segs2RleDataFrame are not necessarily so. For a DataFrame, the argument stack combines all of the individual data.frames into one large data.frame and adds a "Sample" column of sample ids.

Arguments

object	Rle or list/DataFrame of Rle vectors
locs	RangedData with rows corresponding to rows of df
chr.ind	matrix, like from chrIndices method
start	integer, vector of feature start positions
end	integer, vector of feature end positions
stack	logical, rbind list of segment tables for each sample and add "Sample" column?

Details

For a Rle, the user can provide locs or chr.ind, start and stop. The latter is surprisingly much faster and this is used in the DataFrame version.

Value

one or a list of data.frames with columns chrom, loc.start, loc.end, num.mark, seg.mean

Author(s)

Peter M. Haverty

See Also

Other "segmented data": [runCBS](#), [segs2RangedData](#), [segs2Rle](#), [segs2RleDataFrame](#)

Examples

```
data(genoset)
seg.list = runCBS( lrr(baf.ds), locData(baf.ds), return.segs=TRUE )
df = segs2RleDataFrame( seg.list, locData(baf.ds) ) # Loop segs2Rle on list of data.frames in seg.list
assayDataElement( baf.ds, "lrr.segs" ) = df
segTable( df, locData(baf.ds) )
segTable( assayDataElement(baf.ds,"lrr.segs"), locData(baf.ds) )
segTable( assayDataElement(baf.ds,"lrr.segs")[,1], locData(baf.ds), sampleNames(baf.ds)[1] )
```

space

Get space factor for GenoSet

Description

locData slot holds a RangedData, which keeps the chromosome of each feature in a factor names 'space'.

locData slot holds a RangedData.

locData slot holds a RangedData.

locData slot holds a RangedData.

Get chromosome names

Get ranges from locData slot

Get elementLengths from locData slot

Arguments

x	GenoSet
x	GenoSet
x	GenoSet
x	GenoSet
x	GenoSet
x	GenoSet
x	GenoSet
x	GenoSet
i	character, RangedData, RangesList, logical, integer
j	character, RangedData, RangesList, logical, integer
k	character or integer
drop	logical drop levels of space factor?
...	additional subsetting args

Details

Get chromosome names, which are the names of the locData slot.

Get ranges from locData slot

Get elementLengths from locData slot

Value

factor

integer

integer

integer

character

character

character

Author(s)

Peter M. Haverty

Peter M. Haverty

Peter M. Haverty

Peter M. Haverty

Peter Haverty

Peter Haverty

Peter Haverty

Examples

```

data(genoset)
space(genoset.ds)
start(genoset.ds)
end(genoset.ds)
chrNames(genoset.ds)
ranges(genoset.ds) # Returns a RangesList
elementLengths(genoset.ds) # Returns the number of probes per chromosome
data(genoset)
  genoset.ds[1:5,2:3] # first five probes and samples 2 and 3
  genoset.ds[ , "K"] # Sample called K
  rd = RangedData(ranges=IRanges(start=seq(from=15e6,by=1e6,length=7),width=1),names=letters[8:14],space=r)
  genoset.ds[ rd, "K" ] # sample K and probes overlapping those in rd, which overlap specified ranges on ch

```

subsetAssayData	<i>Subset assayData</i>
-----------------	-------------------------

Description

Subset or re-order assayData

Usage

```
subsetAssayData(orig, i, j, ..., drop = FALSE)
```

Arguments

orig	assayData environment
i	row indices
j	col indices
...	Additional args to give to subset operator
drop	logical, drop dimensions when subsetting with single value?

Details

Subset or re-order assayData locked environment, environment, or list. Shamelessly stolen from "[" method in Biobase version 2.8 along with guts of assayDataStorageMode()

Value

assayData data structure

Author(s)

Peter M. Haverty

Examples

```

data(genoset)
ad = assayData(genoset.ds)
small.ad = subsetAssayData(ad,1:5,2:3)

```

toGenomeOrder	<i>Set a GRanges, GenoSet, or RangedData to genome order</i>
---------------	--

Description

Returns a re-ordered object sorted by chromosome and start position. If `strict=TRUE`, then chromosomes must be in order specified by `chrOrder`. If `ds` is already ordered, no re-ordering is done. Therefore, checking order with `isGenomeOrder`, is unnecessary if order will be corrected if `isGenomeOrder` is `FALSE`.

Arguments

<code>ds</code>	GenoSet, GRanges, or RangedData
<code>strict</code>	logical, should chromosomes be in order specified by <code>chrOrder</code> ?

Details

`toGenomeOrder` for `GRanges` differs from `sort` in that it orders by chromosome and start position only, rather than chromosome, strand, start, and width.

Value

re-ordered `ds`

Author(s)

Peter M. Haverty

See Also

Other "genome ordering": [chrOrder](#), [isGenomeOrder](#), [isGenomeOrder](#), [isGenomeOrder](#)

Examples

```
data(genoset)
toGenomeOrder( baf.ds, strict=TRUE )
toGenomeOrder( baf.ds )
toGenomeOrder( locData(baf.ds) )
```

universe	<i>Get and set the genome universe annotation.</i>
----------	--

Description

Genome universe for `locData`

Set genome universe

Arguments

x	GenoSet
x	GenoSet
value	character, new universe string, e.g. hg19

Details

The genome positions of the features in locData. The UCSC notation (e.g. hg18, hg19, etc.) should be used.

Value

character, e.g. hg19
A GenoSet object

Author(s)

Peter M. Haverty
Peter Haverty

Examples

```
data(genoset)
universe(genoset.ds)
universe(genoset.ds) = "hg19"
```

Index

- *Topic **classes**
 - bafset-class, 8
 - cnset-class, 17
 - genoset-class, 24
- *Topic **datasets**
 - genoset-datasets, 25
- *Topic **methods**
 - bafset-class, 8
 - cnset-class, 17
 - genoset-class, 24
- [(space), 37
- [, GenoSet, ANY, ANY, ANY-method (genoset-class), 24
- [, GenoSet, RangedData, ANY, ANY-method (genoset-class), 24
- [, GenoSet, RangesList, ANY, ANY-method (genoset-class), 24
- [, GenoSet, character, ANY, ANY-method (genoset-class), 24
- [<- (space), 37
- [<-, GenoSet, ANY, ANY, ANY-method (space), 37

- asFactorMatrix, 3
- assayDataElement, 4
- assayDataElement<-, 4
- attachAssayDataElements, 5

- baf, 5
- baf, BAFSet-method (bafset-class), 8
- baf.ds (genoset-datasets), 25
- baf2mbaf, 6
- baf<- (baf), 5
- baf<-, BAFSet, matrix-method (bafset-class), 8
- BAFSet, 7
- BAFSet-class (bafset-class), 8
- bafset-class, 8, 8, 18, 25
- BAFSet.to.ExpressionSets, 9
- boundingIndices, 9, 11, 13, 31, 32
- boundingIndices2, 10, 11, 13, 31, 32
- boundingIndicesByChr, 10, 11, 12, 31, 32

- chr, 13
- chr, GenoSet-method (genoset-class), 24
- chr, GRanges-method (genoset-class), 24
- chr, RangedData-method (genoset-class), 24
- chrIndices, 13
- chrIndices, RangedDataOrGenoSetOrGRanges-method (genoset-class), 24
- chrInfo, 14
- chrInfo, RangedDataOrGenoSetOrGRanges-method (genoset-class), 24
- chrNames, 15
- chrNames, GenoSet-method (genoset-class), 24
- chrNames, GRanges-method (genoset-class), 24
- chrNames, RangedData-method (genoset-class), 24
- chrOrder, 15, 27, 40
- class:BAFSet (bafset-class), 8
- class:CNSet (cnset-class), 17
- class:GenoSet (genoset-class), 24
- cn, 16
- cn, CNSet-method (cnset-class), 17
- cn.ds (genoset-datasets), 25
- cn<- (cn), 16
- cn<-, CNSet, matrix-method (cnset-class), 17
- CNSet, 16
- CNSet-class (cnset-class), 17
- cnset-class, 8, 17, 18, 25
- coerce, CNSet, BAFSet-method (bafset-class), 8
- coerce, ExpressionSet, BAFSet-method (bafset-class), 8
- coerce, ExpressionSet, CNSet-method (cnset-class), 17
- colMeans, 18
- colMeans, DataFrame-method (genoset-class), 24
- convertToBigMatrix, 19

- elementLengths (space), 37
- elementLengths, GenoSet-method (genoset-class), 24

- elementLengths, GRanges-method
(genoset-class), 24
- end (space), 37
- end, GenoSet-method (genoset-class), 24
- eSet, 25
- fake.baf (genoset-datasets), 25
- fake.lrr (genoset-datasets), 25
- fake.pData (genoset-datasets), 25
- featureNames<-, 19
- featureNames<-, GenoSet-method
(genoset-class), 24
- gcCorrect, 20, 28
- genomeAxis, 21, 22
- genoPlot, 21, 22
- genoPlot, BAFSet, ANY-method (genoPlot),
22
- genoPlot, CNSet, ANY-method (genoPlot), 22
- genoPlot, GenoSet, ANY-method (genoPlot),
22
- genoPlot, numeric, numeric-method
(genoPlot), 22
- genoPlot, numeric, Rle-method (genoPlot),
22
- genoPos, 23
- genoPos, RangedDataOrGenoSet-method
(genoset-class), 24
- GenoSet, 8, 17, 23
- genoset (genoset-package), 3
- GenoSet-class (genoset-class), 24
- genoset-class, 24
- genoset-datasets, 25
- genoset-package, 3
- genoset.ds (genoset-datasets), 25
- initGenoSet, 26
- isGenomeOrder, 16, 27, 40
- isGenomeOrder, GRanges-method
(isGenomeOrder), 27
- isGenomeOrder, RangedDataOrGenoSet-method
(isGenomeOrder), 27
- loadGC, 20, 27
- loadGC, GenoSet, numeric, BSgenome-method
(loadGC), 27
- loadGC, RangedData, numeric, BSgenome-method
(loadGC), 27
- locData, 28
- locData, GenoSet-method (genoset-class),
24
- locData.rd (genoset-datasets), 25
- locData<- (locData), 28
- locData<- , GenoSet, RangedData-method
(genoset-class), 24
- lrr, 29
- lrr, BAFSet-method (bafset-class), 8
- lrr<- (lrr), 29
- lrr<- , BAFSet, matrix-method
(bafset-class), 8
- modeCenter, 29
- names (space), 37
- names, GenoSet-method (genoset-class), 24
- orderedChrs, RangedDataOrGenoSet-method
(genoset-class), 24
- pos, 30
- pos, RangedDataOrGenoSetOrGRanges-method
(genoset-class), 24
- rangeColMeans, 10, 11, 13, 30, 32
- RangedData, 25
- ranges (space), 37
- ranges, GenoSet-method (genoset-class),
24
- rangeSampleMeans, 10, 11, 13, 31, 31
- readGenoSet, 32
- relocateAssayData, 32
- runCBS, 33, 34–37
- segs2RangedData, 34, 34, 35–37
- segs2Rle, 34, 35, 36, 37
- segs2RleDataFrame, 34, 35, 36, 37
- segTable, 34–36, 36
- segTable, DataFrame-method (segTable), 36
- segTable, Rle-method (segTable), 36
- show, GenoSet-method (genoset-class), 24
- space, 37
- space, GenoSet-method (genoset-class), 24
- start (space), 37
- start, GenoSet-method (genoset-class), 24
- subsetAssayData, 39
- toGenomeOrder, 16, 27, 40
- toGenomeOrder , GenoSet-method
(toGenomeOrder), 40
- toGenomeOrder , GRanges-method
(toGenomeOrder), 40
- toGenomeOrder , RangedData-method
(toGenomeOrder), 40
- uniqueChrs, RangedDataOrGenoSet-method
(genoset-class), 24
- universe, 40

universe, GenoSet-method
 (genoset-class), [24](#)
universe<- (universe), [40](#)
universe<- ,GenoSet-method
 (genoset-class), [24](#)

width (space), [37](#)