

# Package ‘oligoClasses’

October 8, 2014

**Version** 1.24.0

**Title** Classes for high-throughput arrays supported by oligo and crlmm

**Author** Benilton Carvalho and Robert Scharpf

**Maintainer** Benilton Carvalho <Benilton.Carvalho@cancer.org.uk> and Robert Scharpf <rscharpf@jhsp.h.edu>

**Depends** R (>= 2.14), BiocGenerics (>= 0.3.2)

**Imports** BiocGenerics, Biobase (>= 2.17.8), methods, graphics, IRanges (>= 1.13.30), GenomicRanges, Biostrings (>= 2.23.6), affyio (>= 1.23.2), ff, foreach, BiocInstaller, utils

**Enhances** doMC, doMPI, doSNOW, doParallel, doRedis

**Suggests** RSQLite, hapmapsnp5, hapmap-snp6, pd.genomewidesnp.6, pd.genomewidesnp.5, pd.mapping50k.hind240, pd.mapping50k.xba240, pd.mapping250k.sty, pIaICE, RUnit, human370v1cCrlmm

**Description** This package contains class definitions, validity checks, and initialization methods for classes used by the oligo and crlmm packages.

**License** GPL (>= 2)

**LazyLoad** yes

**Collate** AllClasses.R AllGenerics.R utils-general.R utils-lds.R  
utils-parallel.R methods-gSet.R initialize-methods.R  
methods-AlleleSet.R methods-AnnotatedDataFrame.R  
methods-FeatureSet.R methods-AssayData.R methods-SnpFeatureSet.R methods-oligoSnpSet.R  
methods-CopyNumberSet.R methods-CNSet.R methods-PDInfo.R  
methods-RangedDataCNV.R methods-SnpSet.R  
methods-GenomeAnnotatedDataFrame.R methods-BeadStudioSet.R  
methods-BeadStudioSetList.R methods-gSetList.R  
methods-GRanges.R methods-SummarizedExperiment.R show-methods.R functions.R zzz.R

**biocViews** Infrastructure

**R topics documented:**

affyPlatforms . . . . .	3
AlleleSet-class . . . . .	4
annotationPackages . . . . .	5
AssayData-methods . . . . .	6
AssayDataList . . . . .	7
assayDataList-methods . . . . .	8
batch . . . . .	8
batchStatistics . . . . .	9
BeadStudioSet-class . . . . .	10
BeadStudioSetList-class . . . . .	11
celfileDate . . . . .	12
celfileName . . . . .	12
checkExists . . . . .	13
checkOrder . . . . .	14
chromosome-methods . . . . .	15
chromosome2integer . . . . .	16
CNSet-class . . . . .	17
CopyNumberSet-class . . . . .	18
CopyNumberSet-methods . . . . .	20
createFF . . . . .	21
db . . . . .	22
DBPDInfo-class . . . . .	23
efsExample . . . . .	23
exprs-methods . . . . .	24
featureDataList-methods . . . . .	24
FeatureSet-class . . . . .	24
ffdf-class . . . . .	25
ff_matrix-class . . . . .	26
ff_or_matrix-class . . . . .	27
fileConnections . . . . .	27
flags . . . . .	28
generics . . . . .	29
GenomeAnnotatedDataFrame-class . . . . .	29
GenomeAnnotatedDataFrameFrom-methods . . . . .	31
genomeBuild . . . . .	32
geometry . . . . .	32
getA . . . . .	33
getBar . . . . .	34
getSequenceLengths . . . . .	34
GRanges-methods . . . . .	35
gSet-class . . . . .	37
gSetList-class . . . . .	38
i2p . . . . .	39
initializeBigMatrix . . . . .	40
integerMatrix . . . . .	41
is.ffmatrix . . . . .	42

isPackageLoaded . . . . .	43
isSnp-methods . . . . .	44
kind . . . . .	44
ldSetOptions . . . . .	45
length-methods . . . . .	46
library2 . . . . .	46
list.celfiles . . . . .	47
ListClasses . . . . .	48
locusLevelData . . . . .	48
makeFeatureGRanges . . . . .	49
manufacturer-methods . . . . .	50
ocLapply . . . . .	50
ocSamples . . . . .	51
oligoSet . . . . .	52
oligoSnpSet-methods . . . . .	53
parStatus . . . . .	53
pdPkgFromBioC . . . . .	54
platform-methods . . . . .	55
pmFragmentLength-methods . . . . .	55
position-methods . . . . .	55
requireAnnotation . . . . .	56
requireClusterPkgSet . . . . .	57
sampleNames-methods . . . . .	57
scqsExample . . . . .	58
setCluster . . . . .	58
sfsExample . . . . .	59
SnpSet-methods . . . . .	60
SnpSet2-class . . . . .	61
SnpSuperSet-class . . . . .	62
splitIndicesByLength . . . . .	63
sqExample . . . . .	64
SummarizedExperiment-methods . . . . .	65
<b>Index</b>	<b>66</b>

---

affyPlatforms

*Available Affymetrix platforms for SNP arrays*


---

## Description

Provides a listing of available Affymetrix platforms currently supported by the R package oligo

## Usage

```
affyPlatforms()
```

**Value**

A vector of class character.

**Author(s)**

R. Scharpf

**Examples**

```
affyPlatforms()
```

---

AlleleSet-class	<i>Class "AlleleSet"</i>
-----------------	--------------------------

---

**Description**

A class for storing the locus-level summaries of the normalized intensities

**Objects from the Class**

Objects can be created by calls of the form `new("AlleleSet", assayData, phenoData, featureData, experimentData,`

**Slots**

```
assayData: Object of class "AssayData" ~~
phenoData: Object of class "AnnotatedDataFrame" ~~
featureData: Object of class "AnnotatedDataFrame" ~~
experimentData: Object of class "MIAME" ~~
annotation: Object of class "character" ~~
protocolData: Object of class "AnnotatedDataFrame" ~~
.___classVersion__.: Object of class "Versions" ~~
```

**Extends**

Class "[eSet](#)", directly. Class "[VersionedBiobase](#)", by class "eSet", distance 2. Class "[Versioned](#)", by class "eSet", distance 3.

**Methods**

```
allele signature(object = "AlleleSet"): extract allele specific summaries. For 50K (XBA and Hind) and 250K (Sty and Nsp) arrays, an additional argument (strand) must be used (allowed values: 'sense', 'antisense').
bothStrands signature(object = "AlleleSet"): tests if data contains allele summaries on both strands for a given SNP.
bothStrands signature(object = "SnpFeatureSet"): tests if data contains allele summaries on both strands for a given SnpFeatureSet.
```

**db** signature(object = "AlleleSet"): link to database connection.  
**getA** signature(object = "AlleleSet"): average intensities (across alleles)  
**getM** signature(object = "AlleleSet"): log-ratio (Allele A vs. Allele B)

**Author(s)**

R. Scharpf

**See Also**

[SnpSuperSet](#), [CNSet](#)

**Examples**

```
showClass("AlleleSet")
## an empty AlleleSet
x <- new("matrix")
new("AlleleSet", senseAlleleA=x, senseAlleleB=x, antisenseAlleleA=x, antisenseAlleleB=x)
##or
new("AlleleSet", alleleA=x, alleleB=x)
```

---

annotationPackages      *Annotation Packages*

---

**Description**

annotationPackages will return a character vector of the names of annotation packages.

**Usage**

```
annotationPackages()
```

**Value**

a character vector of the names of annotation packages

**Description**

Batch statistics used for estimating copy number are stored as AssayData in the 'batchStatistics' slot of the CNSet class. Each element in the AssayData must have the same number of rows and columns. Rows correspond to features and columns correspond to batch.

**Objects from the Class**

A virtual Class: No objects may be created from it.

**Methods**

**batchNames** signature(object = "AssayData"): ...  
**batchNames<-** signature(object = "AssayData"): ...  
**corr** signature(object = "AssayData", allele = "character"): ...  
**nu** signature(object = "AssayData", allele = "character"): ...  
**phi** signature(object = "AssayData", allele = "character"): ...

**Details**

lM: Extracts entire list of linear model parameters.  
 corr: The within-genotype correlation of log<sub>2</sub>(A) and log<sub>2</sub>(B) intensities.  
 nu: The intercept for the linear model. The linear model is fit to the A and B alleles independently.  
 phi: The slope for the linear model. The linear model is fit independently to the A and B alleles.

**See Also**

[CNSet-class](#)

**Examples**

```
library(crlmm)
library(Biobase)
data(cnSetExample, package="crlmm")
cnSet <- cnSetExample
isCurrent(cnSet)
assayDataElementNames(batchStatistics(cnSet))
## Accessors for linear model parameters
## -- Included here primarily as a check that accessors are working
## -- Values are all NA until CN estimation is performed using the crlmm package
##
## subsetting
cnSet[1:10, ]
```

```
## names of elements in the object
## accessors for parameters
nu(cnSet, "A")[1:10, ]
nu(cnSet, "B")[1:10, ]
phi(cnSet, "A")[1:10, ]
phi(cnSet, "B")[1:10, ]
```

---

AssayDataList

*Create a list of assay data elements*

---

## Description

The eSetList-derived classes have an assayDataList slot instead of an assayData slot.

## Usage

```
AssayDataList(storage.mode = c("lockedEnvironment", "environment", "list"), ...)
```

## Arguments

storage.mode    See assayDataNew.  
...            Named lists of matrices

## Value

environment

## Author(s)

R.Scharpf

## See Also

[assayDataNew](#)

## Examples

```
r <- replicate(5, matrix(rnorm(25),5,5), simplify=FALSE)
r <- lapply(r, function(x,dns) {dimnames(x) <- dns; return(x)}, dns=list(letters[1:5], LETTERS[1:5]))
ad <- AssayDataList(r=r)
ls(ad)
```

---

assayDataList-methods *Accessor for slot assayDataList in Package **oligoClasses***

---

### Description

Accessor for slot assayDataList in Package **oligoClasses**

### Methods

signature(object = "gSetList") An object inheriting from class gSetList.

signature(object = "oligoSetList") An object inheriting from class gSetList.

---

batch *The batch variable for the samples.*

---

### Description

Copy number estimates are susceptible to systematic differences between groups of samples that were processed at different times or by different labs. While 'batch' is often unknown, a useful surrogate is often the scan date of the arrays (e.g., the month of the calendar year) or the 96 well chemistry plate on which the samples were arrayed during lab processing.

### Usage

```
batch(object)
batchNames(object)
batchNames(object) <- value
```

### Arguments

object	An object of class CNSet.
value	For 'batchNames', the value must be a character string corresponding of the unique batch names.

### Value

The method 'batch' returns a character vector that has the same length as the number of samples in the CNSet object.

### Author(s)

R. Scharpf

### See Also

[CNSet-class](#)



### Examples

```
a <- matrix(1:25, 5, 5)
colnames(a) <- letters[1:5]
object <- new("CNSet", alleleA=a, batch=rep("batch1", 5))
batch(object)
batchNames(object)
```

---

batchStatistics	<i>Accessor for batch statistics uses for copy number estimation and storage of model parameters</i>
-----------------	--

---

### Description

The batchStatistics slot contains statistics estimated from each batch that are used to derive copy number estimates.

### Usage

```
batchStatistics(object)
batchStatistics(object) <- value
```

### Arguments

object	An object of class CNSet
value	An object of class AssayData

### Details

An object of class AssayData for slot batchStatistics is initialized automatically when creating a new CNSet instance. Required in the call to new is a factor called batch whose unique values determine the number of columns for each assay data element.

### Value

batchStatics is an accessor for the slot batchStatistics that returns an object of class AssayData.

### See Also

[CNSet-class](#), [batchNames](#), [batch](#)

---

BeadStudioSet-class    *Class "BeadStudioSet"*

---

### Description

A container for log R ratios and B allele frequencies from SNP arrays.

### Objects from the Class

Objects can be created by calls of the form `new("BeadStudioSet", assayData, phenoData, featureData, experimentD`

### Slots

featureData: Object of class "GenomeAnnotatedDataFrame" ~~  
 assayData: Object of class "AssayData" ~~  
 phenoData: Object of class "AnnotatedDataFrame" ~~  
 experimentData: Object of class "MIAXE" ~~  
 annotation: Object of class "character" ~~  
 protocolData: Object of class "AnnotatedDataFrame" ~~  
 genome: Object of class "character" ~~  
 .\_\_classVersion\_\_: Object of class "Versions" ~~

### Extends

Class "gSet", directly. Class "eSet", by class "gSet", distance 2. Class "VersionedBiobase", by class "gSet", distance 3. Class "Versioned", by class "gSet", distance 4.

### Methods

In the methods below, object has class BeadStudioSet.

`baf(object)`: accessor for the matrix of B allele frequencies.

`baf(object) <- value` replacement method for B allele frequencies: value must be a matrix of integers.

`as(object, "data.frame")`: coerce to data.frame with column headers 'lrr', 'baf', 'x' (physical position with unit Mb), 'id', and 'is.snp'. Used for plotting with lattice.

`copyNumber(object)`: accessor for log R ratios.

`copyNumber(object) <- value`: replacement method for the log R ratios

**initialize** signature(.Object = "BeadStudioSet"): constructs an instance of the class

`lrr(object)`: accessor for matrix of log R ratios

`lrr(object) <- value` replacement method for log R ratios: value should be a matrix or a `ff_matrix`.

`show(object)`: print a short summary of the BeadStudioSet object.

`updateObject(object)`: update a BeadStudioSet object.

**Author(s)**

R. Scharpf

**Examples**

```
new("BeadStudioSet")
```

---

BeadStudioSetList-class

*List classes with assay data listed by chromosome*

---

**Description**

Container for log R ratios and B allele frequencies stored by chromosome.

**Slots**

assayDataList: Object of class "AssayData" ~~

phenoData: Object of class "AnnotatedDataFrame" ~~

featureDataList: Object of class "list" ~~

chromosome: Object of class "integer" ~~

annotation: Object of class "character" ~~

genome: Object of class "character" indicating the genome build. Valid entries are "hg18" and "hg19".

**Methods defined for the class**

```
clone(object, id, prefix="",...)
```

Performs a deep copy of the ff objects in the assay data elements of object. A new object of the same class will be instantiated. The ff objects in the instantiated object will point to ff files on disk with prefix given by the argument prefix.

A use-case for such a function is that one may want to perform wave correction on the log R ratios in object, but keep a copy of the original unadjusted log R ratios. If object is not copied using clone prior to wave correction, the log R ratios will be updated on disk and the original, unadjusted log R ratios will no longer be available.

**Author(s)**

R. Scharpf

**See Also**

See supporting packages for methods defined for the class.

---

`celfileDate`*Cel file dates*

---

**Description**

Parses cel file dates from the header of .CEL files for the Affymetrix platform

**Usage**

```
celfileDate(filename)
```

**Arguments**

filename      Name of cel file

**Value**

character string

**Author(s)**

H. Jaffee

**Examples**

```
require(hapmapsnp6)
path <- system.file("celFiles", package="hapmapsnp6")
celfiles <- list.celfiles(path, full.names=TRUE)
dts <- sapply(celfiles, celfileDate)
```

---

`celfileName`*Extracts complete cel file name from a CNSet object*

---

**Description**

Returns the complete cel file (including path) for a CNSet object

**Usage**

```
celfileName(object)
```

**Arguments**

object      An object of class CNSet

**Value**

Character string vector.

**Note**

If the CEL files for an experiment are relocated, the `datadir` should be updated accordingly. See examples.

**Author(s)**

R. Scharpf

**Examples**

```
## Not run:
  if(require(crlmm)){
    data(cnSetExample, package="crlmm")
  }
celfileName(cnSetExample)
}

## End(Not run)
```

---

checkExists	<i>Checks to see whether an object exists and, if not, executes the appropriate function.</i>
-------------	---

---

**Description**

Only loads an object if the object name is not in the global environment. If not in the global environment and the file exists, the object is loaded (by default). If the file does not exist, the function FUN is run.

**Usage**

```
checkExists(.name, .path = ".", .FUN, .FUN2, .save.it=TRUE, .load.it, ...)
```

**Arguments**

.name	Character string giving name of object in global environment
.path	Path to where the object is saved.
.FUN	Function to be executed if <name> is not in the global environment and the file does not exist.
.FUN2	Not currently used.
.save.it	Logical. Whether to save the object to the directory indicated by path. This argument is ignored if the object was loaded from file or already exists in the .GlobalEnv.

`.load.it` Logical. If `load.it` is TRUE, we try to load the object from the indicated path. The returned object will replace the object in the `.GlobalEnv` unless the object is bound to a different name (symbol) when the function is executed.

`...` Additional arguments passed to FUN.

**Value**

Could be anything – depends on what FUN, FUN2 perform.

Future versions could return a 0 or 1 indicating whether the function performed as expected.

**Author(s)**

R. Scharpf

**Examples**

```
path <- tempdir()
dir.create(path)
x <- 3+6
x <- checkExists("x", .path=path, .FUN=function(y, z) y+z, y=3, z=6)
rm(x)
x <- checkExists("x", .path=path, .FUN=function(y, z) y+z, y=3, z=6)
rm(x)
x <- checkExists("x", .path=path, .FUN=function(y, z) y+z, y=3, z=6)
rm(x)
##now there is a file called x.rda in tempdir(). The file will be loaded
x <- checkExists("x", .path=path, .FUN=function(y, z) y+z, y=3, z=6)
rm(x)
unlink(path, recursive=TRUE)
```

---

checkOrder	<i>Checks whether a eSet-derived class is ordered by chromosome and physical position</i>
------------	---

---

**Description**

Checks whether a eSet-derived class (e.g., a SnpSet or CSet object) is ordered by chromosome and physical position

**Usage**

```
checkOrder(object, verbose = FALSE)
chromosomePositionOrder(object, ...)
```

**Arguments**

`object` A SnpSet or CopyNumberSet.

`verbose` Logical.

`...` additional arguments to order

**Details**

Checks whether the object is ordered by chromosome and physical position.

**Value**

Logical

**Author(s)**

R. Scharpf

**See Also**

[order](#)

**Examples**

```
data(oligoSetExample)
if(!checkOrder(oligoSet)){
  oligoSet <- chromosomePositionOrder(oligoSet)
}
checkOrder(oligoSet)
```

---

chromosome-methods

*Methods for function chromosome in package oligoClasses*

---

**Description**

Methods for function chromosome in package **oligoClasses** ~~

**Methods**

The methods for chromosome extracts the chromosome (represented as an integer) for each marker in a eSet-derived class or a AnnotatedDataFrame-derived class.

signature(object = "AnnotatedDataFrame") Accessor for chromosome.

signature(object = "eSet") If 'chromosome' is included in fvarLabels(object), the integer representation of the chromosome will be returned. Otherwise, an error is thrown.

signature(object = "GenomeAnnotatedDataFrame") Accessor for chromosome. If annotation was not available due to a missing or non-existent annotation package, the value returned by the accessor will be a vector of zero's.

(chromosome(object) <- value): Assign chromosome to the AnnotatedDataFrame slot of an eSet-derived object.

signature(object = "RangedDataCNV") Accessor for chromosome.

**Note**

Integer representation: chr X = 23, chr Y = 24, chr XY = 25. Symbols M, Mt, and MT are coded as 26.

**See Also**

[chromosome2integer](#)

**Examples**

```
chromosome2integer(c(1:22, "X", "Y", "XY", "M"))
```

---

chromosome2integer      *Converts chromosome to integer*

---

**Description**

Coerces character string for chromosome in the pd. annotation packages to integers

**Usage**

```
chromosome2integer(chrom)
integer2chromosome(intChrom)
```

**Arguments**

chrom	A one or 2 letter character string (e.g, "1", "X", "Y", "MT", "XY")
intChrom	An integer vector with values 1-25 possible

**Details**

This is useful when sorting SNPs in an object by chromosome and physical position – ensures that the sorting is done in the same way for different objects.

**Value**

integer2chromosome returns a vector of character string indicating the chromosome the same length as intChrom. chromosome2integer returns a vector of integers the same length as the number of elements in the chrom vector.

**Author(s)**

R. Scharpf

**Examples**

```
chromosome2integer(c(1:22, "X", "Y", "XY", "M"))
integer2chromosome(chromosome2integer(c(1:22, "X", "Y", "XY", "M")))
```



---

CNSet-class	<i>Class "CNSet"</i>
-------------	----------------------

---

### Description

CNSet is a container for intermediate data and parameters pertaining to allele-specific copy number estimation. Methods for CNSet objects, including accessors for linear model parameters and allele-specific copy number are included here.

### Objects from the Class

An object from the class is not generally intended to be initialized by the user, but returned by the genotype function in the crlmm package.

The following creates a very basic CNSet with assayData containing the required elements.

```
new(CNSet, alleleA=new("matrix"), alleleB=new("matrix"), call=new("matrix"), callProbability=new("matrix"))
```

### Slots

```
batch: Object of class "factor" ~~
batchStatistics: Object of class "AssayData" ~~
assayData: Object of class "AssayData" ~~
phenoData: Object of class "AnnotatedDataFrame" ~~
featureData: Object of class "AnnotatedDataFrame" ~~
experimentData: Object of class "MIAME" ~~
annotation: Object of class "character" ~~
protocolData: Object of class "AnnotatedDataFrame" ~~
datadir: Object of class "list"~~
mixtureParams: Object of class "matrix"~~
.__classVersion__: Object of class "Versions" ~~
```

### Methods

The argument object for the following methods is a CNSet.

`object[i, j]`: subset the CNSet object by markers (i) and/or samples (j).

`A(object)`: accessor for the normalized intensities of allele A

`A(object) <- value`: replace intensities for the A allele intensities by value. The object value must be a matrix, `ff_matrix`, or `ffdf`.

`allele(object, allele)`: accessor for the normalized intensities for the A or B allele. The argument for allele must be either 'A' or 'B'

`B(object)`: accessor for the normalized intensities of allele B

`B(object)` <- value: replace intensities for the B allele intensities by value. The object value must be a `matrix`, `ff_matrix`, or `ffdf`.  
`batch(object)`: vector of batch labels for each sample.  
`batchNames(object)`: the unique batch names  
`batchNames(object) <- value`: relabel the batches  
`calls(object)`: accessor for genotype calls coded as 1 (AA), 2 (AB), or 3 (BB). Nonpolymorphic markers are NA.  
`cnfs(object)`: accessor for the genotype confidence scores.  
`close(object)`: close any open file connections to `ff` objects stored in the `CNSet` object.  
`as(object, "oligoSnpSet")`: coerce a `CNSet` object to an object of class `oligoSnpSet` – a container for the total copy number and genotype calls.  
`corr(object)`: the correlation of the A and B intensities within each genotype.  
`flags(object)`: flags to indicate possible problems with the copy number estimation. Not fully implemented at this point.  
`new("CNSet")`: instantiating a `CNSet` object.  
`nu(object, allele)`: accessor for the intercept (background) for the A and B alleles. The value of `allele` must be 'A' or 'B'.  
`open(object)` open file connections for all `ff` objects stored in the `CNSet` object.  
`nu(object, allele)`: accessor for the slope for the A and B alleles. The value of `allele` must be 'A' or 'B'.  
`sigma2(object, allele)`: accessor for the within genotype variance  
`tau2(object, allele)`: accessor for background variance

**Author(s)**

R. Scharpf

**Examples**

```
new("CNSet")
```

---

CopyNumberSet-class    *Class "CopyNumberSet"*

---

**Description**

Container for storing total copy number estimates and confidence scores of the copy number estimates.

**Objects from the Class**

Objects can be created by calls of the form `new("CopyNumberSet", assayData, phenoData, featureData, experimentD`

**Slots**

assayData: Object of class "AssayData" ~~  
 phenoData: Object of class "AnnotatedDataFrame" ~~  
 featureData: Object of class "AnnotatedDataFrame" ~~  
 experimentData: Object of class "MIAXE" ~~  
 annotation: Object of class "character" ~~  
 protocolData: Object of class "AnnotatedDataFrame" ~~  
 .\_\_classVersion\_\_: Object of class "Versions" ~~

**Extends**

Class "eSet", directly. Class "VersionedBiobase", by class "eSet", distance 2. Class "Versioned", by class "eSet", distance 3.

**Methods**

**cnConfidence** signature(object = "CopyNumberSet"): ...  
**cnConfidence<-** signature(object = "CopyNumberSet", value = "matrix"): ...  
**coerce** signature(from = "CNSet", to = "CopyNumberSet"): ...  
**copyNumber** signature(object = "CopyNumberSet"): ...  
**copyNumber<-** signature(object = "CopyNumberSet", value = "matrix"): ...  
**initialize** signature(.Object = "CopyNumberSet"): ...

**Note**

This container is primarily for platforms for which genotypes are unavailable. As oligoSnpSet extends this class, methods related to total copy number that do not depend on genotypes can be defined at this level.

**Author(s)**

R. Scharpf

**See Also**

For genotyping platforms, total copy number estimates and genotype calls can be stored in the [oligoSnpSet](#) class.

**Examples**

```

showClass("CopyNumberSet")
cnset <- new("CopyNumberSet")
ls(Biobase::assayData(cnset))

```

---

CopyNumberSet-methods *Methods for class CopyNumberSet.*

---

## Description

Accessors and CopyNumberSet

## Usage

```
copyNumber(object, ...)
cnConfidence(object)
copyNumber(object) <- value
cnConfidence(object) <- value
```

## Arguments

object	CopyNumberSet object or derived class
...	Ignored for CopyNumberSet and oligoSnpSet.
value	matrix

## Value

copyNumber returns a matrix of copy number estimates or relative copy number estimates. Since the copy number estimates are stored as integers (copy number \* 100), the matrix returned by the copyNumber accessor will need to be divided by a factor of 100 to transform the measurements back to the original copy number scale.

cnConfidence returns a matrix of confidence scores for the copy number estimates. These are also represented as integers and will require a back-transformation to the original scale.

## Examples

```
library(Biobase)
data(locusLevelData)
path <- system.file("extdata", package="oligoClasses")
fd <- readRDS(file.path(path, "genomeAnnotatedDataFrameExample.rds"))
## the following command creates an oligoSnpSet object, storing
## an integer representation of the log2 copy number in the copyNumber element
## of the assayData. Genotype calls and genotype confidence scores are also stored
## in the assayData.
oligoSet <- new("oligoSnpSet",
  copyNumber=integerMatrix(log2(locusLevelData[["copynumber"]]/100), 100),
  call=locusLevelData[["genotypes"]],
  callProbability=integerMatrix(locusLevelData[["crlmmConfidence"]], 1),
  annotation=locusLevelData[["platform"]],
  featureData=fd,
  genome="hg19")
```

```

## There are several accessors for the oligoSnpSet class.
icn <- copyNumber(oligoSet)
range(icn) ## integer scale
lcn <- icn/100
range(lcn) ## log2 copy number

## confidence scores for the genotypes are also represented on an integer scale
ipr <- snpCallProbability(oligoSet)
range(ipr) ## integer scale

## for genotype confidence scores, the helper function i2p
## converts back to a probability scale
pr <- i2p(ipr)
range(pr)

## The helper function confs is a shortcut, extracting the
## integer-based confidence scores and transforming to the
## probability scale
pr2 <- confs(oligoSet)
all.equal(pr, pr2)

## To extract information on the annotation of the SNPs, one can use
position(oligoSet)
chromosome(oligoSet)
## the position and chromosome coordinates were extracted from build hg19
genomeBuild(oligoSet)

```

---

createFF

*Create ff objects.*


---

## Description

Creates ff objects (array-like) using settings (path) defined by oligoClasses.

## Usage

```
createFF(name, dim, vmode = "double", initdata = NULL)
```

## Arguments

name	Prefix for filename.
dim	Dimensions.
vmode	Mode.
initdata	NULL.

## Value

ff object.

**Note**

This function is meant to be used by developers.

**See Also**

ff

---

db

*Get the connection to the SQLite Database*

---

**Description**

This function will return the SQLite connection to the database associated to objects used in oligo.

**Usage**

```
db(object)
```

**Arguments**

object            Object of valid class. See methods.

**Value**

SQLite connection.

**Methods**

**object = "FeatureSet"** object of class FeatureSet

**object = "SnpCallSet"** object of class SnpCallSet

**object = "DBPDInfo"** object of class DBPDInfo

**object = "SnpLevelSet"** object of class SnpLevelSet

**Author(s)**

Benilton Carvalho

**Examples**

```
## db(object)
```

---

DBPDInfo-class	<i>Class "DBPDInfo"</i>
----------------	-------------------------

---

**Description**

A class for Platform Design Information objects, stored using a database approach

**Objects from the Class**

Objects can be created by calls of the form `new("DBPDInfo", ...)`.

**Slots**

`getdb`: Object of class "function"

`tableInfo`: Object of class "data.frame"

`manufacturer`: Object of class "character"

`genomebuild`: Object of class "character"

`geometry`: Object of class "integer" with length 2 (rows x columns)

**Methods**

**annotation** string describing annotation package associated to object

---

efsExample	<i>ExpressionFeatureSet Object</i>
------------	------------------------------------

---

**Description**

Example of ExpressionFeatureSet Object.

**Usage**

```
data(efsExample)
```

**Format**

Object belongs to ExpressionFeatureSet class.

**Examples**

```
data(efsExample)
class(efsExample)
```





**Slots**

manufacturer: Object of class "character"  
 assayData: Object of class "AssayData"  
 phenoData: Object of class "AnnotatedDataFrame"  
 featureData: Object of class "AnnotatedDataFrame"  
 experimentData: Object of class "MIAME"  
 annotation: Object of class "character"  
 .\_\_classVersion\_\_: Object of class "Versions"

**Methods**

**show** signature(.Object = "FeatureSet"): show object contents  
**bothStrands** signature(.Object = "SnpFeatureSet"): checks if object contains data for both strands simultaneously (50K/250K Affymetrix SNP chips - in this case it returns TRUE); if object contains data for one strand at a time (SNP 5.0 and SNP 6.0 - in this case it returns FALSE)

**Author(s)**

Benilton Carvalho

**See Also**

[eSet](#), [VersionedBiobase](#), [Versioned](#)

**Examples**

```

set.seed(1)
tmp <- 2^matrix(rnorm(100), ncol=4)
rownames(tmp) <- 1:25
colnames(tmp) <- paste("sample", 1:4, sep="")
efs <- new("ExpressionFeatureSet", exprs=tmp)

```

---

ffdf-class

*Class "ffdf"*


---

**Description**

Extended package ff's class definitions for ff to S4.

**Objects from the Class**

A virtual Class: No objects may be created from it.

**Slots**

.S3Class: Object of class ffdff ~~

**Extends**

Class "oldClass", directly. Class "list\_or\_ffdff", directly.

**Methods**

No methods defined with class "ffdff" in the signature.

---

ff_matrix-class	<i>Class "ff_matrix"</i>
-----------------	--------------------------

---

**Description**

~~ A concise (1-5 lines) description of what the class is. ~~

**Objects from the Class**

A virtual Class: No objects may be created from it.

**Slots**

.S3Class: Object of class "character" ~~

**Extends**

Class "oldClass", directly.

**Methods**

**annotatedDataFrameFrom** signature(object = "ff\_matrix"): ...

**Examples**

```
showClass("ff_matrix")
```

---

```
ff_or_matrix-class      Class "ff_or_matrix"
```

---

**Description**

A class union of 'ffdf', 'ff\_matrix', and 'matrix'

**Objects from the Class**

A virtual Class: No objects may be created from it.

**Methods**

**GenomeAnnotatedDataFrameFrom** signature(object = "ff\_or\_matrix"): ...

**Author(s)**

R. Scharpf

**See Also**

[ff](#), [ffdf](#)

**Examples**

```
showClass("ff_or_matrix")
```

---

```
fileConnections      Open and close methods for matrices and numeric vectors
```

---

**Description**

CNSet objects can contain ff-derived objects that contain pointers to files on disk, or ordinary matrices. Here we define open and close methods for ordinary matrices and vectors that simply pass back the original matrix/vector.

**Usage**

```
open(con, ...)
openff(object)
closeff(object)
```

**Arguments**

con	matrix or vector
object	A CNSet object.
...	Ignored

**Value**

not applicable

**Author(s)**

R. Scharpf

**Examples**

```
open(rnorm(15))
open(matrix(rnorm(15), 5, 3))
```

---

flags

*Batch-level summary of SNP flags.*

---

**Description**

Used to flag SNPs with low minor allele frequencies, or for possible problems during the CN estimation step. Currently, this is primarily more for internal use.

**Usage**

```
flags(object)
```

**Arguments**

object            An object of class CNSet

**Value**

A matrix or `ff_matrix` object with rows corresponding to markers and columns corresponding to batch.

**See Also**

[batchStatistics](#)

**Examples**

```
x <- matrix(runif(250*96*2, 0, 2), 250, 96*2)
test1 <- new("CNSet", alleleA=x, alleleB=x, call=x, callProbability=x,
            batch=as.character(rep(letters[1:2], each=96)))
dim(flags(test1))
```

---

generics	<i>Miscellaneous generics. Methods defined in packages that depend on oligoClasses</i>
----------	--

---

**Description**

Miscellaneous generics. Methods defined in packages that depend on oligoClasses

**Usage**

```
baf(object)
lrr(object)
```

**Arguments**

object            A eSet-derived class.

**Author(s)**

R. Scharpf

---

GenomeAnnotatedDataFrame-class	<i>Class "GenomeAnnotatedDataFrame"</i>
--------------------------------	---

---

**Description**

AnnotatedDataFrame with genomic coordinates (chromosome, position)

**Slots**

```
varMetadata: Object of class "data.frame" ~~
data: Object of class "data.frame" ~~
dimLabels: Object of class "character" ~~
. __classVersion__: Object of class "Versions" ~~
```

**Extends**

Class "[AnnotatedDataFrame](#)", directly. Class "[Versioned](#)", by class "AnnotatedDataFrame", distance 2.

**Coercion to or from other classes**

`as(from, "GenomeAnnotatedDataFrame"):`

Coerce an object of class `AnnotatedDataFrame` to a `GenomeAnnotatedDataFrame`.

`makeFeatureGRanges(object, genome, ...):`

Construct a `GRanges` instance from a `GenomeAnnotatedDataFrame` object. `genome` is a character string indicating the UCSC build. Supported builds are "hg18" and "hg19", but are platform specific. In particular, some platforms only support build hg19 at this time.

`updateObject(object):`

For updating a `GenomeAnnotatedDataFrame`

**Accessors**

`chromosome(object), chromosome(object) <- value`

Get or set chromosome.

`isSnp(object):`

Many platforms include polymorphic and nonpolymorphic markers. `isSnp` evaluates to TRUE if the marker is polymorphic.

`position(object):`

Physical position in the genome

`getArm(object, genome):`

Retrieve character vector indicating the chromosome arm of each marker in object. `genome` should indicate which genome build was used to define the chromosomal locations (currently, only UCSC genome builds 'hg18' and 'hg19' supported for this function).

**Author(s)**

R. Scharpf

**See Also**

[AnnotatedDataFrame](#)

**Examples**

```
new("GenomeAnnotatedDataFrame")
if(require("pd.mapping50k.hind240") && require("pd.mapping50k.xba240") && require("SNPchip")){
  data(locusLevelData)
  gd <- GenomeAnnotatedDataFrameFrom(locusLevelData[["genotypes"]],
    annotationPkg=locusLevelData[["platform"]],
    genome="hg19")
    arm <- getArm(gd, "hg19")
}
```

---

GenomeAnnotatedDataFrameFrom-methods

*Methods for Function GenomeAnnotatedDataFrameFrom in Package oligoClasses*

---

## Description

GenomeAnnotatedDataFrameFrom is a convenience for creating [GenomeAnnotatedDataFrame](#) objects.

## Methods

Use the method with `GenomeAnnotatedDataFrameFrom(object, annotationPkg, genome, ...)`; the argument `annotationPkg` *must* be specified for `matrix` and `AssayData` classes.

`signature(object="assayData")` This method creates an `GenomeAnnotatedDataFrame` using feature names and dimensions of an [AssayData](#) object as a template.

`signature(object="matrix")` This method creates an `GenomeAnnotatedDataFrame` using row names and dimensions of a `matrix` object as a template.

`signature(object="NULL")` This method (called with 'NULL' as the object) creates an empty `GenomeAnnotatedDataFrame`.

`signature(object="array")` This method (called with 'array' as the object) creates a `GenomeAnnotatedDataFrame` using the first dimension of the array (rows are the number of features).

## Author(s)

R Scharpf

## Examples

```
require(Biobase)
minReqVersion <- "1.0.2"
require(human370v1cCr1mm)
if (packageDescription("human370v1cCr1mm", fields=Version) >= minReqVersion){
  x <- matrix(1:25, 5, 5,
    dimnames=list(c("rs10000092", "rs1000055", "rs100016", "rs10003241", "rs10004197"), NULL))
  gd <- GenomeAnnotatedDataFrameFrom(x, annotationPkg="human370v1cCr1mm",
    genome="hg18")
  pData(gd)
  chromosome(gd)
  position(gd)
}
```

---

genomeBuild	<i>Genome Build Information</i>
-------------	---------------------------------

---

**Description**

Returns the genome build. This information comes from the annotation package and is given as an argument during the package creation process.

**Usage**

```
genomeBuild(object)
```

**Arguments**

object	Supported objects include PDInfo, FeatureSet, and any gSet-derived or eSetList-derived object.
--------	--

**Value**

character string

**Note**

Supported builds are UCSC genome builds are 'hg18' and 'hg19'.

**Examples**

```
showMethods("genomeBuild", where="package:oligoClasses")
```

---

geometry	<i>Array Geometry Information</i>
----------	-----------------------------------

---

**Description**

For a given array, geometry returns the physical geometry of it.

**Usage**

```
geometry(object)
```

**Arguments**

object	PDInfo or FeatureSet object
--------	-----------------------------

**Examples**

```
if (require(pd.mapping50k.xba240))  
  geometry(pd.mapping50k.xba240)
```



---

getA *Compute average log-intensities / log-ratios*

---

### Description

Methods to compute average log-intensities and log-ratios across alleles, within strand.

### Usage

```
getA(object)
getM(object)
A(object, ...)
B(object, ...)
```

### Arguments

object	SnqQSet, SnpCnvQSet or TilingFeatureSet2 object.
...	arguments to be passed to allele - 'sense' and 'antisense' are valid values if the array is pre-SNP_5.0

### Details

For SNP data, SNPRMA summarizes the SNP information into 4 quantities (log<sub>2</sub>-scale):

- antisenseThetaAantisense allele A. (Not applicable for Affymetrix 5.0 and 6.0 platforms.)
- antisenseThetaBantisense allele B. (Not applicable for Affymetrix 5.0 and 6.0 platforms.)
- senseThetaAsense allele A. (Not applicable for Affymetrix 5.0 and 6.0 platforms.)
- senseThataBsense allele B. (Not applicable for Affymetrix 5.0 and 6.0 platforms.)
- alleleAAffymetrix 5.0 and 6.0 platforms
- alleleBAffymetrix 5.0 and 6.0 platforms

The average log-intensities are given by:  $(\text{antisenseThetaA} + \text{antisenseThetaB}) / 2$  and  $(\text{senseThetaA} + \text{senseThetaB}) / 2$ .

The average log-ratios are given by:  $\text{antisenseThetaA} - \text{antisenseThetaB}$  and  $\text{senseThetaA} - \text{senseThetaB}$ .

For Tiling data, getM and getA return the log-ratio and average log-intensities computed across channels:  $M = \log_2(\text{channel1}) - \log_2(\text{channel2})$   $A = (\log_2(\text{channel1}) + \log_2(\text{channel2})) / 2$

When large data support is enabled with the ff package, the AssayData elements of an AlleleSet object can be ff\_matrix or ffd, in which case pointers to the ff object are stored in the assay data. The functions open and close can be used to open or close the connection, respectively.

### Value

A 3-dimensional array (SNP's x Samples x Strand) with the requested measure, when the input SNP data (50K, 250K).

A 2-dimensional array (SNP's x Samples), when the input is from SNP 5.0 and SNP 6.0 arrays.

A 2-dimensional array if the input is from Tiling arrays.

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

---

getBar	<i>Gets a bar of a given length.</i>
--------	--------------------------------------

---

**Description**

Gets a bar of a given length.

**Usage**

```
getBar(width = getOption("width"))
```

**Arguments**

width                    desired length of the bar.

**Value**

character string.

**Author(s)**

Benilton S Carvalho

**Examples**

```
message(getBar())
```

---

getSequenceLengths	<i>Load chromosome sequence lengths for UCSC genome build hg18 or hg19</i>
--------------------	--

---

**Description**

Load chromosome sequence lengths for UCSC genome build hg18 or hg19

**Usage**

```
getSequenceLengths(build)
```

**Arguments**

build                    character string: "hg18" or "hg19"

**Details**

The chromosome sequence lengths for UCSC builds hg18 and hg19 were extracted from the packages BSgenome.Hsapiens.UCSC.hg18 and BSgenome.Hsapiens.UCSC.hg19, respectively.

**Value**

Names integer vector of chromosome lengths.

**Author(s)**

R. Scharpf

**Examples**

```
getSequenceLengths("hg18")
getSequenceLengths("hg19")

if(require("GenomicRanges")){
  ## from GenomicRanges
  sl <- getSequenceLengths("hg18")[c("chr1", "chr2", "chr3")]
  gr <-
  GRanges(seqnames =
    Rle(c("chr1", "chr2", "chr1", "chr3"), c(1, 3, 2, 4)),
    ranges =
    IRanges(1:10, width = 10:1, names = head(letters,10)),
    strand =
    Rle(strand(c("-", "+", "*+", "+", "-")),
      c(1, 2, 2, 3, 2)),
    score = 1:10,
    GC = seq(1, 0, length=10),
    seqlengths=sl)
  metadata(gr) <- list(genome="hg18")
  gr
  metadata(gr)
}
```

**Description**

Methods for GRanges objects

**findOverlaps methods**

`findOverlaps(query, subject, ...)`:

Find the feature indices in `subject` that overlap the genomic intervals in `query`, where `query` is a `GRanges` object and `subject` is a `gSet`-derived object. Additional arguments to the `findOverlaps` method in the package **IRanges** can be passed through the `...` operator.

**Accessors**

`object` is an instance of the `GRanges` class.

`coverage2(object)`:

For the `GRanges` and `GRangesList` objects returned by the hidden Markov model implemented in the "VanillaICE" package and the segmentation algorithm in the "MinimumDistance" package, the intervals are annotated by the number of probes (markers) for SNPs and nonpolymorphic regions. `coverage2` and `numberProbes` are convenient accessors for these annotations.

`genomeBuild(object)`:

Accessor for the UCSC genome build.

`numberProbes(object)`:

Integer vector indicating the number of probes (markers) for each range in `object`. Equivalent to `coverage2`.

`state(object)`:

Accessor for the `elementMetadata` column 'state', when applicable. State is used to contain the index of the inferred copy number state for various hmm methods defined in the **VanillaICE**.

**See Also**

[GRanges](#)

**Examples**

```
library(IRanges)
library(GenomicRanges)
gr1 <- GRanges(seqnames = "chr2", ranges = IRanges(3, 6),
state=3L, numberProbes=100L)
## convenience functions
state(gr1)
numberProbes(gr1)

gr2 <- GRanges(seqnames = c("chr1", "chr1"),
ranges = IRanges(c(7,13), width = 3),
state=c(2L, 2L), numberProbes=c(200L, 250L))
gr3 <- GRanges(seqnames = c("chr1", "chr2"),
ranges = IRanges(c(1, 4), c(3, 9)),
state=c(1L, 4L), numberProbes=c(300L, 350L))
## Ranges organized by sample
grl <- GRangesList("sample1" = gr1, "sample2" = gr2, "sample3" = gr3)
sampleNames(grl) ## same as names(grl)
numberProbes(grl)
chromosome(grl)
```

```

state(gr1)
gr <- stack(gr1)
sampleNames(gr)
chromosome(gr)
state(gr)

```

---

gSet-class

*Container for objects with genomic annotation on SNPs*


---

### Description

Container for objects with genomic annotation on SNPs

### Objects from the Class

A virtual Class: No objects may be created from it.

### Slots

```

featureData: Object of class "GenomeAnnotatedDataFrame" ~~
assayData: Object of class "AssayData" ~~
phenoData: Object of class "AnnotatedDataFrame" ~~
experimentData: Object of class "MIAXE" ~~
annotation: Object of class "character" ~~
protocolData: Object of class "AnnotatedDataFrame" ~~
genome: Object of class "character" ~~
. __classVersion__: Object of class "Versions" ~~

```

### Extends

Class "eSet", directly. Class "VersionedBiobase", by class "eSet", distance 2. Class "Versioned", by class "eSet", distance 3.

### Methods

The object for the below methods is a class that extends the virtual class gSet.

checkOrder(object): checks that the object is ordered by chromosome and physical position.  
Returns logical.

chromosome(object): accessor for chromosome in the GenomeAnnotatedDataFrame slot.

chromosome(object) <- value: replacement method for chromosome in the GenomeAnnotatedDataFrame slot. value must be an integer vector.

db(object): database connection

genomeBuild(object), genomeBuild(object) <- value:  
 Get or set the UCSC genome build. Supported builds are hg18 and hg19.

getArm(object): Character vector indicating the chromosomal arm for each marker in object.

isSnp(object): whether the marker is polymorphic. Returns a logical vector.

makeFeatureGRanges(object): Construct an instance of the GRanges class from a GenomeAnnotatedDataFrame.

position(object): integer vector of the genomic position

show(object):  
 Print a concise summary of object.

**Author(s)**

R. Scharpf

**See Also**

[chromosome](#), [position](#), [isSnp](#)

**Examples**

```
showClass("gSet")
```

---

gSetList-class      *Virtual Class for Lists of eSets*

---

**Description**

Virtual Class for Lists of eSets.

**Objects from the Class**

A virtual Class: No objects may be created from it.

**Slots**

assayDataList: Object of class "AssayData" ~~  
 phenoData: Object of class "AnnotatedDataFrame" ~~  
 protocolData: Object of class "AnnotatedDataFrame" ~~  
 experimentData: Object of class "MIAME" ~~  
 featureDataList: Object of class "list" ~~  
 chromosome: Object of class "vector" ~~  
 annotation: Object of class "character" ~~  
 genome: Object of class "character" ~~

**Accessors**

object is an instance of a gSetList-derived class.

annotation(object):

character string indicating the package used to provide annotation for the features on the array.

chromosome(object):

Returns the chromosome corresponding to each element in the gSetList object

elementLengths(object): Returns the number of rows for each list of assays. In most gSetList-derived classes, the assays are organized by chromosome and elementLengths returns the number of markers for each chromosome.

genomeBuild(object), genomeBuild(object) <- value:

Get or set the UCSC genome build. Supported builds are hg18 and hg19.

**Coercion**

object is an instance of a gSetList-derived class.

makeFeatureGRanges(object, ...):

Create a GRanges object for the featureData. The featureData is stored as a list. This method stacks the featureData from each list element. Metadata columns in the GRanges object include physical position ('position'), a SNP indicator ('isSnp'), and the chromosome. The genome build is extracted from object using the method genomeBuild.

**Author(s)**

R. Scharpf

**See Also**

[oligoSetList](#), [BeadStudioSetList](#)

**Examples**

```
showClass("gSetList")
```

---

i2p

*Functions to convert probabilities to integers, or integers to probabilities.*

---

**Description**

Probabilities estimated in the cr1mm package are often stored as integers to save memory. We provide a few utility functions to go back and forth between the probability and integer representations.

**Usage**

```
i2p(i)
p2i(p)
```

**Arguments**

i	A matrix or vector of integers.
p	A matrix or vector of probabilities.

**Value**

The value returned by `i2p` is

$$1 - \exp(-i/1000)$$

The value returned by `p2i` is

$$\text{as.integer}(-1000 * \log(1-p))$$
**See Also**

[confs](#)

**Examples**

```
i2p(693)
p2i(0.5)
i2p(p2i(0.5))
```

---

`initializeBigMatrix`    *Initialize big matrices/vectors.*

---

**Description**

Initialize big matrices or vectors appropriately (conditioned on the status of support for large datasets - see Details).

**Usage**

```
initializeBigMatrix(name=basename(tempfile()), nr=0L, nc=0L, vmode = "integer", initdata = NA)
initializeBigVector(name=basename(tempfile()), n=0L, vmode = "integer",
  initdata = NA)
initializeBigArray(name=basename(tempfile()), dim=c(0L,0L,0L),
  vmode="integer", initdata=NA)
```



**Arguments**

name	prefix to be used for file stored on disk
nr	number of rows
nc	number of columns
n	length of the vector
vmode	mode - "integer", "double"
initdata	Default is NA
dim	Integer vector indicating the dimensions of the array to initialize

**Details**

These functions are meant to be used by developers. They provide means to appropriately create big vectors or matrices for packages like oligo and crlmm (and friends). These objects are created conditioned on the status of support for large datasets.

**Value**

If the 'ff' package is loaded (in the search path), then an 'ff' object is returned. A regular R vector or array is returned otherwise.

**Examples**

```
x <- initializeBigVector("test", 10)
class(x)
x
if (isPackageLoaded("ff"))
  finalizer(x) <- "delete"
rm(x)
initializeBigMatrix(nr=5L, nc=5L)
initializeBigArray(dim=c(10, 5, 3))
```

---

integerMatrix	<i>Coerce numeric matrix (or array) to a matrix (array) of integers, retaining dimnames.</i>
---------------	--

---

**Description**

Coerce numeric matrix to matrix of integers, retaining dimnames.

**Usage**

```
integerMatrix(x, scale = 100)
integerArray(x, scale=100)
```

**Arguments**

`x` a matrix or array

`scale` scalar (numeric). If not 1, `x` is multiplied by `scale` prior to coercing to a matrix of integers.

**Value**

A matrix or array of integers.

**Author(s)**

R. Scharpf

**Examples**

```
x <- matrix(rnorm(10), 5, 2)
rownames(x) = letters[1:5]
i <- integerMatrix(x, scale=100)
```

---

`is.ffmatrix`

*Check if object is an ff-matrix object.*

---

**Description**

Check if object is an ff-matrix object.

**Usage**

```
is.ffmatrix(object)
```

**Arguments**

`object` object to be checked

**Value**

Logical.

**Note**

This function is meant to be used by developers.

**Examples**

```
if (isPackageLoaded("ff")){
  x1 <- ff(vmode="double", dim=c(10, 2))
  is.ffmatrix(x1)
}
x1 <- matrix(0, nr=10, nc=2)
is.ffmatrix(x1)
```

---

isPackageLoaded	<i>Check if package is loaded.</i>
-----------------	------------------------------------

---

**Description**

Checks if package is loaded.

**Usage**

```
isPackageLoaded(pkg)
```

**Arguments**

pkg                    Package to be checked.

**Details**

Checks if package name is in the search path.

**Value**

Logical.

**See Also**

search

**Examples**

```
isPackageLoaded("oligoClasses")
isPackageLoaded("ff")
isPackageLoaded("snow")
```

---

isSnp-methods                      *Methods for Function isSnp in package oligoClasses*~~

---

### Description

~~ Methods for function isSnp in package **oligoClasses** ~~

### Methods

Return an indicator for whether the marker is polymorphic (value 1) or nonpolymorphic (value 0).

Return an indicator for whether the vector of marker identifiers in object is polymorphic. pkgname must be one of the supported annotation packages specific to the platform.

signature(object = "character", pkgname = "character") signature(object = "eSet", pkgname = "ANY")  
If 'isSnp' is included in fvarLabels(object), an indicator for polymorphic markers is returned. Otherwise, an error is thrown.

signature(object = "GenomeAnnotatedDataFrame", pkgname = "ANY") Accessor for indicator of whether the marker is polymorphic. If annotation was not available due to a missing or non-existent annotation package, the value returned by the accessor will be a vector of zero's.

---

kind                                      *Array type*

---

### Description

Retrieves the array type.

### Usage

```
kind(object)
```

### Arguments

object                      FeatureSet or DBPDIInfo object

### Value

String: "Expression", "Exon", "SNP" or "Tiling"

### Examples

```
if (require(pd.mapping50k.xba240)){
  data(sfsExample)
  Biobase::annotation(sfsExample) <- "pd.mapping50k.xba240"
  kind(sfsExample)
}
```

---

ldSetOptions	<i>Set/check large dataset options.</i>
--------------	---

---

**Description**

Set/check large dataset options.

**Usage**

```
ldSetOptions(nsamples=100, nprobesets=20000, path=getwd(), verbose=FALSE)
ldStatus(verbose=FALSE)
ldPath(path)
```

**Arguments**

nsamples	number of samples to be processed at once.
nprobesets	number of probesets to be processed at once.
path	path where to store large dataset objects.
verbose	verbosity (logical).

**Details**

Some functions in oligo/crlmm can process data in batches to minimize memory footprint. When using this feature, the 'ff' package resources are used (and possibly combined with cluster resources set in options() via 'snow' package).

Methods that are executed on a sample-by-sample manner can use ocSamples() to automatically define how many samples are processed at once (on a compute node). Similarly, methods applied to probesets can use ocProbesets(). Users should set these options appropriately.

ldStatus checks the support for large datasets.

ldPath checks where ff files are stored.

**Author(s)**

Benilton S Carvalho

**See Also**

ocSamples, ocProbesets

**Examples**

```
ldStatus(TRUE)
```

---

length-methods	<i>Number of samples for FeatureSet-like objects.</i>
----------------	---

---

**Description**

Number of samples for FeatureSet-like objects.

**Methods**

`x = "FeatureSet"` Number of samples

---

library2	<i>Supress package startup messages when loading a library</i>
----------	--

---

**Description**

Supress package startup messages when loading a library

**Usage**

```
library2(...)
```

**Arguments**

... arguments to library

**Author(s)**

R. Scharpf

**See Also**

[library](#)

**Examples**

```
library2("Biobase")
```

---

list.celfiles	<i>List CEL files.</i>
---------------	------------------------

---

**Description**

Function used to get a list of CEL files.

**Usage**

```
list.celfiles(..., listGzipped=FALSE)
```

**Arguments**

...	Passed to <a href="#">list.files</a>
listGzipped	Logical. List .CEL.gz files?

**Value**

Character vector with filenames.

**Note**

Quite often users want to use this function to pass filenames to other methods. In this situations, it is safer to use the argument 'full.names=TRUE'.

**See Also**

[list.files](#)

**Examples**

```
if (require(hapmapsnp5)){
  path <- system.file("celFiles", package="hapmapsnp5")

  ## only the filenames
  list.celfiles(path)

  ## the filenames with full path...
  ## very useful when genotyping samples not in the working directory
  list.celfiles(path, full.names=TRUE)
}else{
  ## this wont return anything
  ## if in the working directory there isnt any CEL
  list.celfiles(getwd())
}
```

---

ListClasses	<i>eSetList class</i>
-------------	-----------------------

---

**Description**

Initialization method for eSetList virtual class.

---

locusLevelData	<i>Basic data elements required for the HMM</i>
----------------	---

---

**Description**

This object is a list containing the basic data elements required for the HMM

**Usage**

```
data(locusLevelData)
```

**Format**

A list

**Details**

The basic assay data elements that can be used for fitting the HMM are:

1. a mapping of platform identifiers to chromosome and physical position
2. (optional) a matrix of copy number estimates
3. (optional) a matrix of confidence scores for the copy number estimates (e.g., inverse standard deviations)
4. (optional) a matrix of genotype calls
5. (optional) CRLMM confidence scores for the genotype calls

At least (2) or (4) is required. The locusLevelData is a list that contains (1), (2), (4), and (5).

**Source**

A HapMap sample on the Affymetrix 50k platform. Chromosomal alterations were simulated. The last 100 SNPs on chromosome 2 are, in fact, a repeat of the first 100 SNPs on chromosome 1 – this was added for internal use.

**Examples**

```
data(locusLevelData)
str(locusLevelData)
```



---

makeFeatureGRanges	<i>Construct a GRanges object from several possible feature-level classes</i>
--------------------	---

---

### Description

Construct a GRanges object from several possible feature-level classes. The conversion is useful for subsequent ranged-data queries, such as `findOverlaps`, `countOverlaps`, etc.

### Usage

```
makeFeatureGRanges(object, ...)
```

### Arguments

object	A gSet-derived object containing chromosome and physical position for the markers on the array.
...	See the makeFeatureGRanges method for GenomeAnnotatedDataFrame.

### Value

A GRanges object.

### Author(s)

R. Scharpf

### See Also

[findOverlaps](#), [GRanges](#), [GenomeAnnotatedDataFrame](#)

### Examples

```
if(require("VanillaICE")){
  library(oligoClasses)
  library(GenomicRanges)
  library(Biobase)
  library(foreach)
  registerDoSEQ()
  data(oligoSetExample, package="oligoClasses")
  oligoSet <- oligoSet[chromosome(oligoSet) == 1, ]
  gr1 <- hmm(oligoSet,TAUP=1e10)
  class(gr1)## GRangesList
  gr <- gr1[[1]]
  (frange <- makeFeatureGRanges(oligoSet))
  ## which features overlap with the second range in sample NA06993
  subsetByOverlaps(frange, gr[2,])
}
```

---

manufacturer-methods    *Manufacturer ID for FeatureSet-like objects.*

---

### Description

Manufacturer ID for FeatureSet-like and DBPDInfo-like objects.

### Methods

**object = "FeatureSet"** Manufacturer ID

**object = "PDInfo"** Manufacturer ID

---

ocLapply                    *lapply-like function that parallelizes code when possible.*

---

### Description

ocLapply is an lapply-like function that checks if ff/snow are loaded and if the cluster variable is set to execute FUN on a cluster. If these requirements are not available, then lapply is used.

### Usage

ocLapply(X, FUN, ..., neededPkgs)

### Arguments

X	first argument to FUN.
FUN	function to be executed.
...	additional arguments to FUN.
neededPkgs	packages needed to execute FUN on the compute nodes.

### Details

neededPkgs is needed when parallel computing is expected to be used. These packages are loaded on the compute nodes before the execution of FUN.

### Value

A list of length length(X).

### Author(s)

Benilton S Carvalho

### See Also

lapply, parStatus

---

`ocSamples`*Cluster and large dataset management utilities.*

---

## Description

Tools to simplify management of clusters via 'snow' package and large dataset handling through the 'bigmemory' package.

## Usage

```
ocSamples(n)
ocProbesets(n)
```

## Arguments

`n` integer representing the maximum number of samples/probesets to be processed simultaneously on a compute node.

## Details

Some methods in the `oligo/crlmm` packages, like `backgroundCorrect`, `normalize`, `summarize` and `rma` can use a cluster (set through the 'foreach' package). The use of cluster features is conditioned on the availability of the 'ff' (used to provide shared objects across compute nodes) and 'foreach' packages.

To use a cluster, 'oligo/crlmm' checks for three requirements: 1) 'ff' is loaded; 2) an adaptor for the parallel backend (like 'doMPI', 'doSNOW', 'doMC') is loaded and registered.

If only the 'ff' package is available and loaded (in addition to the caller package - 'oligo' or 'crlmm'), these methods will allow the user to analyze datasets that would not fit in RAM at the expense of performance.

In the situations above (large datasets and cluster), `oligo/crlmm` uses the options `ocSamples` and `ocProbesets` to limit the amount of RAM used by the machine(s). For example, if `ocSamples` is set to 100, steps like background correction and normalization process (in RAM) 100 samples simultaneously on each compute node. If `ocProbesets` is set to 10K, then summarization processes 10K probesets at a time on each machine.

## Warning

In both scenarios (large dataset and/or cluster use), there is a penalty in performance because data are written to disk (to either minimize memory footprint or share data across compute nodes).

## Author(s)

Benilton Carvalho

## Examples

```
if(require(doMC)) {
  registerDoMC()
  ## tasks like summarize()
}
```

---

oligoSet

*An example instance of oligoSnpSet class*

---

## Description

An example instance of the oligoSnpSet class

## Usage

```
data(oligoSetExample)
```

## Source

Created from the simulated locusLevelData provided in this package.

## See Also

[locusLevelData](#)

## Examples

```
## Not run:
## oligoSetExample created by the following
data(locusLevelData)
oligoSet <- new("oligoSnpSet",
  copyNumber=integerMatrix(log2(locusLevelData[["copynumber"]]/100), 100),
  call=locusLevelData[["genotypes"]],
  callProbability=locusLevelData[["crLmmConfidence"]],
  annotation=locusLevelData[["platform"]],
  genome="hg19")
oligoSet <- oligoSet[!is.na(chromosome(oligoSet)), ]
oligoSet <- oligoSet[chromosome(oligoSet) < 3, ]

## End(Not run)
data(oligoSetExample)
oligoSet
```

---

oligoSnpSet-methods    *Methods for oligoSnpSet class*

---

### Description

Methods for oligoSnpSet class

### Methods

In the following code, object is an instance of the oligoSnpSet class.

`new("oligoSnpSet", ...)`: Instantiates an object of class oligoSnpSet. The assayData elements of the oligoSnpSet class can include matrices of genotype calls, confidence scores for the genotype calls, B allele frequencies, absolute or relative copy number, and confidence scores for the copy number estimates. Each matrix should be coerced to an integer scale prior to assignment to the oligoSnpSet object. Validity methods defined for the class will fail if the matrices are not integers. See examples for additional details.

`baf(object)`: Accessor for integer representation of the B allele frequencies. The value returned by this method can be divided by 1000 to obtain B allele frequencies on the original [0,1] scale.

`baf(object) <- value`: Assign an integer representation of the B allele frequencies to the 'baf' element of the assayData slot. value must be a matrix of integers. See the examples for help converting BAFs to a matrix of integers.

---

`parStatus`                    *Checks if oligo/crlmm can use parallel resources.*

---

### Description

Checks if oligo/crlmm can use parallel resources (needs ff and snow package, in addition to options(cluster=makeCluster(...)).

### Usage

```
parStatus()
```

### Value

logical

### Author(s)

Benilton S Carvalho

---

`pdPkgFromBioC`*Get packages from BioConductor.*

---

**Description**

This function checks if a given package is available on BioConductor and installs it, in case it is.

**Usage**

```
pdPkgFromBioC(pkgname, lib = .libPaths()[1], verbose = TRUE)
```

**Arguments**

<code>pkgname</code>	character. Name of the package to be installed.
<code>lib</code>	character. Path where to install the package at.
<code>verbose</code>	logical. Verbosity flag.

**Details**

Internet connection required.

**Value**

Logical: TRUE if package was found, downloaded and installed; FALSE otherwise.

**Author(s)**

Benilton Carvalho

**See Also**

`download.packages`

**Examples**

```
## Not run:  
pdPkgFromBioC("pd.mapping50k.xba240")  
  
## End(Not run)
```

---

platform-methods      *Platform Information*

---

**Description**

Platform Information

**Methods**

**object = "FeatureSet"** platform information

---

pmFragmentLength-methods  
*Information on Fragment Length*

---

**Description**

This method will return the fragment length for PM probes.

**Methods**

**object = "AffySNPPDInfo"** On AffySNPPDInfo objects, it will return the fragment length that contains the SNP in question.

---

position-methods      *Methods for function position in Package oligoClasses*

---

**Description**

Methods for function position in package **oligoClasses**

**Methods**

The methods for position extracts the physical position stored as an integer for each marker in a eSet-derived class or a AnnotatedDataFrame-derived class.

signature(object = "AnnotatedDataFrame") Accessor for physical position.

signature(object = "eSet") If 'position' is included in fvarLabels(object), the physical position will be returned. Otherwise, an error is thrown.

signature(object = "GenomeAnnotatedDataFrame") Accessor for physical position. If annotation was not available due to a missing or non-existent annotation package, the value returned by the accessor will be a vector of zero's.

---

requireAnnotation      *Helper function to load packages.*

---

### Description

This function checks the existence of a given package and loads it if available. If the package is not available, the function checks its availability on BioConductor, downloads it and installs it.

### Usage

```
requireAnnotation(pkgname, lib=.libPaths()[1], verbose = TRUE)
```

### Arguments

pkgname            character. Package name (usually an annotation package).  
lib                character. Path where to install packages at.  
verbose            logical. Verbosity flag.

### Value

Logical: TRUE if package is available or FALSE if package unavailable for download.

### Author(s)

Benilton Carvalho

### See Also

install.packages

### Examples

```
## Not run:  
requirePackage("pd.mapping50k.xba240")  
  
## End(Not run)
```



---

requireClusterPkgSet *DEPRECATED FUNCTIONS. Package loaders for clusters.*

---

**Description**

Package loaders for clusters.

**Usage**

```
requireClusterPkgSet(packages)
requireClusterPkg(pkg, character.only)
```

**Arguments**

packages            character vector with the names of the packages to be loaded on the compute nodes.

pkg                 name of a package given as a name or literal character string

character.only     a logical indicating whether 'pkg' can be assumed to be a character string

**Details**

requireClusterPkgSet applies require for a set of packages on the cluster nodes.

requireClusterPkg applies require for \*ONE\* package on the cluster nodes and accepts every argument taken by require.

**Value**

Logical.

**Author(s)**

Benilton S Carvalho

**See Also**

require

---

sampleNames-methods    *Sample names for FeatureSet-like objects*

---

**Description**

Returns sample names for FeatureSet-like objects.

**Methods**

**object = "FeatureSet"** Sample names

---

`scqsExample`*SnpCnvQSet Example*

---

**Description**

Example of SnpCnvQSet object.

**Usage**

```
data(scqsExample)
```

**Format**

Object belongs to SnpCnvQSet class.

**Examples**

```
data(scqsExample)
class(scqsExample)
```

---

`setCluster`*DEPRECATED FUNCTIONS. Cluster and large dataset management utilities.*

---

**Description**

Tools to simplify management of clusters via 'snow' package and large dataset handling through the 'bigmemory' package.

**Usage**

```
setCluster(...)
getCluster()
delCluster()
```

**Arguments**

... arguments to be passed to makeCluster in the 'snow' package.

## Details

Some methods in the oligo/crlmm packages, like backgroundCorrect, normalize, summarize and rma can use a cluster (set through 'snow' package). The use of cluster features is conditioned on the availability of the 'bigmemory' (used to provide shared objects across compute nodes) and 'snow' packages.

To use a cluster, 'oligo/crlmm' checks for three requirements: 1) 'ff' is loaded; 2) 'snow' is loaded; and 3) the 'cluster' option is set (e.g., via options(cluster=makeCluster(...)) or setCluster(...)).

If only the 'ff' package is available and loaded (in addition to the caller package - 'oligo' or 'crlmm'), these methods will allow the user to analyze datasets that would not fit in RAM at the expense of performance.

In the situations above (large datasets and cluster), oligo/crlmm uses the options ocSamples and ocProbesets to limit the amount of RAM used by the machine(s). For example, if ocSamples is set to 100, steps like background correction and normalization process (in RAM) 100 samples simultaneously on each compute node. If ocProbesets is set to 10K, then summarization processes 10K probesets at a time on each machine.

## Warning

In both scenarios (large dataset and/or cluster use), there is a penalty in performance because data are written to disk (to either minimize memory footprint or share data across compute nodes).

## Author(s)

Benilton Carvalho

---

sfsExample

*SnpFeatureSet Example*

---

## Description

Example of SnpFeatureSet object.

## Usage

```
data(sfsExample)
```

## Format

Object belongs to SnpFeatureSet class

## Examples

```
data(sfsExample)
class(sfsExample)
```

**Description**

Utility functions for accessing data in SnpSet objects.

**Usage**

```
calls(object)
calls(object) <- value
confs(object, transform=TRUE)
confs(object) <- value
```

**Arguments**

object	A <a href="#">SnpSet</a> object.
transform	Logical. Whether to transform the integer representation of the confidence score (for memory efficiency) to a probability. See details.
value	A matrix.

**Details**

calls returns the genotype calls. CRLMM stores genotype calls as integers (1 - AA; 2 - AB; 3 - BB).

confs returns the confidences associated with the genotype calls. The current implementation of CRLMM stores the confidences as integers to save memory on disk by using the transformation:

```
round(-1000*log2(1-p)),
```

where 'p' is the posterior probability of the call. confs is a convenience function that transforms the integer representation back to a probability. Note that if the assayData elements of the SnpSet objects are ff\_matrix or ffd, the confs function will return a warning. For such objects, one should first subset the ff object and coerce to a matrix, then apply the above conversion. The function `snpCallProbability` for the `callProbability` slot of SnpSet objects. See the examples below.

checkOrder checks whether the object is ordered by chromosome and physical position, evaluating to TRUE or FALSE.

**Note**

Note that the replacement method for `confs<-` expects a matrix of probabilities and will automatically convert the probabilities to an integer representation. See details for the conversion.

The accessor `snpCallProbability` is an accessor for the 'callProbability' element of the assayData. The name can be misleading, however, as the accessor will not return a probability if the call probabilities are represented as integers.

**See Also**

The helper functions `p2i` converts probabilities to integers and `i2p` converts integers to probabilities. See `order` and `checkOrder`.

**Examples**

```
theCalls <- matrix(sample(1:3, 20, rep=TRUE), nc=2)
p <- matrix(runif(20), nc=2)
integerRepresentation <- matrix(as.integer(round(-1000*log(1-p))), 10, 2)
obj <- new("SnpSet2", call=theCalls, callProbability=integerRepresentation)
calls(obj)
  confs(obj) ## coerces to probability scale
  int <- Biobase::snpCallProbability(obj) ## not necessarily a probability
  p3 <- i2p(int) ## to convert back to a probability
```

---

SnpSet2-class

Class "SnpSet2"

---

**Description**

A container for genotype calls and confidence scores. Similar to the `SnpSet` class in **Biobase**, but `SnpSet2` extends `gSet` directly whereas `SnpSet` extends `eSet`. Useful properties of `gSet` include the genome slot and the `GenomeAnnotatedDataFrame`.

**Objects from the Class**

Objects can be created by calls of the form `new("SnpSet2", assayData, phenoData, featureData, experimentData, a`

**Slots**

**genome:** Object of class "character" indicating the UCSC genome build. Supported builds are 'hg18' and 'hg19'.  
**assayData:** Object of class "AssayData".  
**phenoData:** Object of class "AnnotatedDataFrame".  
**featureData:** Object of class "AnnotatedDataFrame".  
**experimentData:** Object of class "MIAXE".  
**annotation:** Object of class "character" ~~  
**protocolData:** Object of class "AnnotatedDataFrame" ~~  
**.\_\_classVersion\_\_:** Object of class "Versions" ~~

**Extends**

Class "`gSet`", directly. Class "`eSet`", by class "`gSet`", distance 2. Class "`VersionedBiobase`", by class "`gSet`", distance 3. Class "`Versioned`", by class "`gSet`", distance 4.

**Accessors**

The argument object for the following methods is an instance of the SnpSet2 class.

```
calls(object): calls(object) <- value:
  Gets or sets the genotype calls. value can be a matrix or a ff_matrix.
confs(object): confs(object) <- value:
  Gets or sets the genotype confidence scores. value can be a matrix or a ff_matrix.
snpCall(object): snpCallProbability(object) <- value:
  Gets or sets the genotype confidence scores.
```

**Author(s)**

R. Scharpf

**See Also**

[SnpSet](#)

**Examples**

```
showClass("SnpSet2")
new("SnpSet2")
```

---

SnpSuperSet-class      *Class "SnpSuperSet"*

---

**Description**

A class to store locus-level summaries of the quantile normalized intensities, genotype calls, and genotype confidence scores

**Objects from the Class**

```
new("SnpSuperSet", alleleA=alleleA, alleleB=alleleB, call=call, callProbability, ...).
```

**Slots**

```
assayData: Object of class "AssayData" ~~
phenoData: Object of class "AnnotatedDataFrame" ~~
featureData: Object of class "AnnotatedDataFrame" ~~
experimentData: Object of class "MIAME" ~~
annotation: Object of class "character" ~~
protocolData: Object of class "AnnotatedDataFrame" ~~
.___classVersion__: Object of class "Versions" ~~
```

**Extends**

Class "[AlleleSet](#)", directly. Class "[SnpSet](#)", directly. Class "[eSet](#)", by class "AlleleSet", distance 2. Class "[VersionedBiobase](#)", by class "AlleleSet", distance 3. Class "[Versioned](#)", by class "AlleleSet", distance 4.

**Methods**

No methods defined with class "SnpSuperSet" in the signature.

**Author(s)**

R. Scharpf

**See Also**

[AlleleSet](#)

**Examples**

```
showClass("SnpSuperSet")
## empty object from the class
x <- new("matrix")
new("SnpSuperSet", alleleA=x, alleleB=x, call=x, callProbability=x)
```

---

splitIndicesByLength *Tools to distribute objects across nodes or by length.*

---

**Description**

Tools to distribute objects across nodes or by length.

**Usage**

```
splitIndicesByLength(x, lg, balance=FALSE)
splitIndicesByNode(x)
```

**Arguments**

x	object to be split
lg	length
balance	logical. Currently ignored

**Details**

splitIndicesByLength splits x in groups of length lg.

splitIndicesByNode splits x in N groups (where N is the number of compute nodes available).

**Value**

List.

**Author(s)**

Benilton S Carvalho

**See Also**

split

**Examples**

```
x <- 1:100
splitIndicesByLength(x, 8)
splitIndicesByLength(x, 8, balance=TRUE)
splitIndicesByNode(x)
```

---

sqsExample

*SnpQSet Example*

---

**Description**

Example of SnpQSet instance.

**Usage**

```
data(sqsExample)
```

**Format**

Belongs to SnpQSet class.

**Examples**

```
data(sqsExample)
class(sqsExample)
```



---

SummarizedExperiment-methods

*Methods for SummarizedExperiment objects*

---

## Description

Methods for [SummarizedExperiment](#).

## Usage

```
## S4 method for signature SummarizedExperiment
baf(object)
## S4 method for signature SummarizedExperiment
chromosome(object, ...)
## S4 method for signature SummarizedExperiment
isSnp(object, ...)
## S4 method for signature SummarizedExperiment
lrr(object)
```

## Arguments

object	A <a href="#">SummarizedExperiment</a> object.
...	ignored

## Details

baf and lrr are accessors for the B allele frequencies and log R ratio assays (matrices or arrays), respectively,

chromosome returns the seqnames of the rowData.

isSnp returns a logical vector for each marker in rowData indicating whether the marker targets a SNP (nonpolymorphic regions are FALSE).

## See Also

[SummarizedExperiment](#)

# Index

## \*Topic **IO**

list.celfiles, 47

## \*Topic **attribute**

getSequenceLengths, 34

## \*Topic **classes**

AlleleSet-class, 4

AssayData-methods, 6

BeadStudioSet-class, 10

BeadStudioSetList-class, 11

CNSet-class, 17

CopyNumberSet-class, 18

DBPDInfo-class, 23

FeatureSet-class, 24

ff\_matrix-class, 26

ff\_or\_matrix-class, 27

ffdf-class, 25

GenomeAnnotatedDataFrame-class, 29

gSet-class, 37

gSetList-class, 38

ListClasses, 48

SnpsSet2-class, 61

SnpsSuperSet-class, 62

## \*Topic **datasets**

efsExample, 23

locusLevelData, 48

oligoSet, 52

scqsExample, 58

sfsExample, 59

sqExample, 64

## \*Topic **data**

pdPkgFromBioC, 54

requireAnnotation, 56

## \*Topic **list**

affyPlatforms, 3

## \*Topic **manip**

AssayDataList, 7

assayDataList-methods, 8

batchStatistics, 9

celfileDate, 12

celfileName, 12

checkExists, 13

checkOrder, 14

chromosome2integer, 16

CopyNumberSet-methods, 20

createFF, 21

featureDataList-methods, 24

fileConnections, 27

flags, 28

genomeBuild, 32

geometry, 32

getA, 33

getBar, 34

i2p, 39

initializeBigMatrix, 40

integerMatrix, 41

is.ffmatrix, 42

isPackageLoaded, 43

kind, 44

ldSetOptions, 45

library2, 46

makeFeatureGRanges, 49

ocLapply, 50

ocSamples, 51

parStatus, 53

requireClusterPkgSet, 57

setCluster, 58

SnpsSet-methods, 60

splitIndicesByLength, 63

## \*Topic **methods**

assayDataList-methods, 8

batch, 8

batchStatistics, 9

chromosome-methods, 15

CopyNumberSet-methods, 20

db, 22

exprs-methods, 24

featureDataList-methods, 24

flags, 28

- GenomeAnnotatedDataFrameFrom-methods, 31
- GRanges-methods, 35
- isSnp-methods, 44
- length-methods, 46
- manufacturer-methods, 50
- oligoSnpSet-methods, 53
- platform-methods, 55
- pmFragmentLength-methods, 55
- position-methods, 55
- sampleNames-methods, 57
- SummarizedExperiment-methods, 65
- \*Topic **misc**
  - affyPlatforms, 3
  - generics, 29
- \*Topic **utilities**
  - list.celfiles, 47
- [,CNSet-method (CNSet-class), 17
- [,gSetList-method (gSetList-class), 38
- [[,BafLrrSetList,ANY,ANY-method (BeadStudioSetList-class), 11
- [[,BeadStudioSetList,ANY,ANY-method (BeadStudioSetList-class), 11
- [[<- ,BafLrrSetList,ANY,ANY,BafLrrSet-method (BeadStudioSetList-class), 11
- [[<- ,gSetList,ANY,ANY,BafLrrSet-method (gSetList-class), 38
- \$,gSetList-method (gSetList-class), 38
- \$<- ,gSetList-method (gSetList-class), 38
- A (getA), 33
- A,AlleleSet-method (getA), 33
- A,CNSet-method (CNSet-class), 17
- A<- (getA), 33
- A<- ,AlleleSet,matrix-method (getA), 33
- A<- ,AlleleSet-method (getA), 33
- A<- ,CNSet-method (CNSet-class), 17
- AffyExonPDInfo (DBPDInfo-class), 23
- AffyExonPDInfo-class (DBPDInfo-class), 23
- AffyExpressionPDInfo (DBPDInfo-class), 23
- AffyExpressionPDInfo-class (DBPDInfo-class), 23
- AffyGenePDInfo (DBPDInfo-class), 23
- AffyGenePDInfo-class (DBPDInfo-class), 23
- affyPlatforms, 3
- AffySNPCNVPDInfo (DBPDInfo-class), 23
- AffySNPCNVPDInfo-class (DBPDInfo-class), 23
- AffySNPPDInfo (DBPDInfo-class), 23
- AffySNPPDInfo-class (DBPDInfo-class), 23
- AffySTPDInfo (DBPDInfo-class), 23
- AffySTPDInfo-class (DBPDInfo-class), 23
- AffyTilingPDInfo (DBPDInfo-class), 23
- AffyTilingPDInfo-class (DBPDInfo-class), 23
- allele (AlleleSet-class), 4
- allele,AlleleSet-method (AlleleSet-class), 4
- allele,CNSet-method (CNSet-class), 17
- allele,SnpFeatureSet-method (AlleleSet-class), 4
- AlleleSet, 63
- AlleleSet (AlleleSet-class), 4
- AlleleSet-class, 4
- AnnotatedDataFrame, 29, 30
- annotatedDataFrameFrom,ff\_matrix-method (ff\_matrix-class), 26
- annotation,DBPDInfo-method (DBPDInfo-class), 23
- annotation,gSetList-method (gSetList-class), 38
- annotationPackages, 5
- AssayData, 31
- AssayData-methods, 6
- AssayDataList, 7
- assayDataList (assayDataList-methods), 8
- assayDataList,gSetList-method (gSetList-class), 38
- assayDataList,oligoSetList-method (assayDataList-methods), 8
- assayDataList-methods, 8
- assayDataNew, 7
- B (getA), 33
- B,AlleleSet-method (getA), 33
- B,CNSet-method (CNSet-class), 17
- B<- (getA), 33
- B<- ,AlleleSet,matrix-method (getA), 33
- B<- ,AlleleSet-method (getA), 33
- B<- ,CNSet-method (CNSet-class), 17
- baf (generics), 29
- baf,BeadStudioSet-method (BeadStudioSet-class), 10
- baf,BeadStudioSetList-method (BeadStudioSetList-class), 11

- baf, oligoSetList-method  
(BeadStudioSetList-class), 11
- baf, oligoSnpSet-method  
(oligoSnpSet-methods), 53
- baf, SummarizedExperiment-method  
(SummarizedExperiment-methods),  
65
- baf<- (BeadStudioSet-class), 10
- baf<- ,BeadStudioSet-method  
(BeadStudioSet-class), 10
- baf<- ,oligoSnpSet-method  
(oligoSnpSet-methods), 53
- BafLrrSet-class (BeadStudioSet-class),  
10
- BafLrrSetList-class  
(BeadStudioSetList-class), 11
- batch, 8, 9
- batch, CNSet-method (CNSet-class), 17
- batchNames, 9
- batchNames (batch), 8
- batchNames, AssayData-method  
(AssayData-methods), 6
- batchNames, CNSet-method (CNSet-class),  
17
- batchNames<- (batch), 8
- batchNames<- ,AssayData-method  
(AssayData-methods), 6
- batchNames<- ,CNSet-method  
(CNSet-class), 17
- batchStatistics, 9, 28
- batchStatistics, CNSet-method  
(CNSet-class), 17
- batchStatistics<- (batchStatistics), 9
- batchStatistics<- ,CNSet, AssayData-method  
(CNSet-class), 17
- BeadStudioSet (BeadStudioSet-class), 10
- BeadStudioSet-class, 10
- BeadStudioSetList, 39
- BeadStudioSetList-class, 11
- bothStrands (AlleleSet-class), 4
- bothStrands, AlleleSet-method  
(AlleleSet-class), 4
- bothStrands, SnpFeatureSet-method  
(AlleleSet-class), 4
- calls (SnpSet-methods), 60
- calls, CNSet-method (CNSet-class), 17
- calls, oligoSetList-method  
(BeadStudioSetList-class), 11
- calls, oligoSnpSet-method  
(oligoSnpSet-methods), 53
- calls, SnpSet-method (SnpSet-methods), 60
- calls, SnpSet2-method (SnpSet2-class), 61
- calls<- (SnpSet-methods), 60
- calls<- ,CNSet, matrix-method  
(CNSet-class), 17
- calls<- ,oligoSnpSet, matrix-method  
(oligoSnpSet-methods), 53
- calls<- ,SnpSet, matrix-method  
(SnpSet-methods), 60
- calls<- ,SnpSet2, matrix-method  
(SnpSet2-class), 61
- callsConfidence, oligoSnpSet-method  
(oligoSnpSet-methods), 53
- callsConfidence<- ,oligoSnpSet, matrix-method  
(oligoSnpSet-methods), 53
- celfileDate, 12
- celfileName, 12
- checkExists, 13
- checkOrder, 14, 61
- checkOrder, CopyNumberSet-method  
(CopyNumberSet-class), 18
- checkOrder, gSet-method (gSet-class), 37
- checkOrder, SnpSet-method  
(SnpSet-methods), 60
- chromosome, 38
- chromosome (chromosome-methods), 15
- chromosome, AnnotatedDataFrame-method  
(chromosome-methods), 15
- chromosome, GenomeAnnotatedDataFrame-method  
(chromosome-methods), 15
- chromosome, GRanges-method  
(chromosome-methods), 15
- chromosome, GRangesList-method  
(chromosome-methods), 15
- chromosome, gSet-method  
(chromosome-methods), 15
- chromosome, gSetList-method  
(gSetList-class), 38
- chromosome, SnpSet-method  
(chromosome-methods), 15
- chromosome, SummarizedExperiment-method  
(SummarizedExperiment-methods),  
65
- chromosome-methods, 15
- chromosome2integer, 16, 16
- chromosome<- (chromosome-methods), 15

- chromosome<-, GenomeAnnotatedDataFrame, integer-method  
 (chromosome-methods), 15
- chromosome<-, gSet, integer-method  
 (chromosome-methods), 15
- chromosome<-, SnpSet, integer-method  
 (chromosome-methods), 15
- chromosomePositionOrder (checkOrder), 14
- clone (BeadStudioSetList-class), 11
- clone, BafLrrSetList-method  
 (BeadStudioSetList-class), 11
- close (fileConnections), 27
- close, AlleleSet-method (getA), 33
- close, array-method (fileConnections), 27
- close, CNSet-method (CNSet-class), 17
- close, matrix-method (fileConnections),  
 27
- close, numeric-method (fileConnections),  
 27
- closeff (fileConnections), 27
- closeff, CNSet-method (fileConnections),  
 27
- cnConfidence (CopyNumberSet-methods), 20
- cnConfidence, CopyNumberSet-method  
 (CopyNumberSet-class), 18
- cnConfidence, oligoSnpSet-method  
 (oligoSnpSet-methods), 53
- cnConfidence<- (CopyNumberSet-methods),  
 20
- cnConfidence<-, CopyNumberSet, matrix-method  
 (CopyNumberSet-class), 18
- cnConfidence<-, oligoSnpSet, matrix-method  
 (oligoSnpSet-methods), 53
- CNSet, 5
- CNSet (CNSet-class), 17
- CNSet-class, 17
- coerce, AnnotatedDataFrame, GenomeAnnotatedDataFrame-method  
 (GenomeAnnotatedDataFrame-class),  
 29
- coerce, BeadStudioSet, data.frame-method  
 (BeadStudioSet-class), 10
- coerce, CNSet, CopyNumberSet-method  
 (CNSet-class), 17
- coerce, CNSet, oligoSnpSet (CNSet-class),  
 17
- coerce, CNSet, oligoSnpSet-method  
 (CNSet-class), 17
- coerce, CNSetLM, CNSet-method  
 (CNSet-class), 17
- coerce, gSetList, list-method  
 (gSetList-class), 38
- coerce, oligoSnpSet, data.frame-method  
 (oligoSnpSet-methods), 53
- confs, 40
- confs (SnpSet-methods), 60
- confs, CNSet-method (CNSet-class), 17
- confs, SnpSet-method (SnpSet-methods), 60
- confs, SnpSet2-method (SnpSet2-class), 61
- confs<- (SnpSet-methods), 60
- confs<-, CNSet, matrix-method  
 (CNSet-class), 17
- confs<-, SnpSet, matrix-method  
 (SnpSet-methods), 60
- confs<-, SnpSet2, matrix-method  
 (SnpSet2-class), 61
- copyNumber (CopyNumberSet-methods), 20
- copyNumber, BeadStudioSet-method  
 (BeadStudioSet-class), 10
- copyNumber, CopyNumberSet-method  
 (CopyNumberSet-class), 18
- copyNumber, oligoSetList-method  
 (BeadStudioSetList-class), 11
- copyNumber, oligoSnpSet-method  
 (oligoSnpSet-methods), 53
- copyNumber<- (CopyNumberSet-methods), 20
- copyNumber<-, BeadStudioSet, ANY-method  
 (BeadStudioSet-class), 10
- copyNumber<-, CopyNumberSet, matrix-method  
 (CopyNumberSet-class), 18
- copyNumber<-, oligoSnpSet, matrix-method  
 (oligoSnpSet-methods), 53
- CopyNumberSet (CopyNumberSet-class), 18
- CopyNumberSet-class, 18
- CopyNumberSet-methods, 20
- corr (AssayData-methods), 6
- cPP, CNSet, character-method  
 (CNSet-class), 17
- coverage2 (GRanges-methods), 35
- coverage2, GRanges-method  
 (GRanges-methods), 35
- coverage2, GRangesList-method  
 (GRanges-methods), 35
- createFF, 21
- db, 22
- db, AlleleSet-method (AlleleSet-class), 4
- db, DBPDInfo-method (db), 22
- db, FeatureSet-method (db), 22

- db, gSet-method (gSet-class), 37
- db, SnpCnvQSet-method (db), 22
- db, SnpQSet-method (db), 22
- db, SnpSet-method (db), 22
- db-methods (db), 22
- DBPDInfo (DBPDInfo-class), 23
- DBPDInfo-class, 23
- delCluster (setCluster), 58
- delCluster-deprecated (setCluster), 58
- dims, gSetList-method (gSetList-class), 38
- efsExample, 23
- elementLengths, gSetList-method (gSetList-class), 38
- eSet, 4, 10, 19, 25, 37, 61, 63
- ExonFeatureSet (FeatureSet-class), 24
- ExonFeatureSet-class (FeatureSet-class), 24
- ExpressionFeatureSet (FeatureSet-class), 24
- ExpressionFeatureSet-class (FeatureSet-class), 24
- ExpressionPDInfo (DBPDInfo-class), 23
- ExpressionPDInfo-class (DBPDInfo-class), 23
- exprs, FeatureSet-method (exprs-methods), 24
- exprs, SnpSet2-method (SnpSet2-class), 61
- exprs-methods, 24
- featureDataList (featureDataList-methods), 24
- featureDataList, gSetList-method (gSetList-class), 38
- featureDataList-methods, 24
- FeatureSet (FeatureSet-class), 24
- FeatureSet-class, 24
- ff, 27
- ff\_matrix-class, 26
- ff\_or\_matrix-class, 27
- ffdf, 27
- ffdf-class, 25
- fileConnections, 27
- findOverlaps, 49
- findOverlaps, GRanges, gSet-method (GRanges-methods), 35
- findOverlaps, GRangesList, gSet-method (GRanges-methods), 35
- flags, 28
- flags, AssayData-method (AssayData-methods), 6
- flags, CNSet-method (CNSet-class), 17
- GeneFeatureSet (FeatureSet-class), 24
- GeneFeatureSet-class (FeatureSet-class), 24
- generics, 29
- GenomeAnnotatedDataFrame, 31, 49
- GenomeAnnotatedDataFrame (GenomeAnnotatedDataFrame-class), 29
- GenomeAnnotatedDataFrame-class, 29
- GenomeAnnotatedDataFrameFrom (GenomeAnnotatedDataFrameFrom-methods), 31
- GenomeAnnotatedDataFrameFrom, array-method (GenomeAnnotatedDataFrameFrom-methods), 31
- GenomeAnnotatedDataFrameFrom, AssayData-method (GenomeAnnotatedDataFrameFrom-methods), 31
- GenomeAnnotatedDataFrameFrom, ff\_or\_matrix-method (GenomeAnnotatedDataFrameFrom-methods), 31
- GenomeAnnotatedDataFrameFrom, list-method (GenomeAnnotatedDataFrameFrom-methods), 31
- GenomeAnnotatedDataFrameFrom, NULL-method (GenomeAnnotatedDataFrameFrom-methods), 31
- GenomeAnnotatedDataFrameFrom-methods, 31
- genomeBuild, 32
- genomeBuild, DBPDInfo-method (genomeBuild), 32
- genomeBuild, FeatureSet-method (genomeBuild), 32
- genomeBuild, GRanges-method (GRanges-methods), 35
- genomeBuild, gSet-method (gSet-class), 37
- genomeBuild, gSetList-method (gSetList-class), 38
- genomeBuild<- (genomeBuild), 32
- genomeBuild<-, gSet, character-method (gSet-class), 37
- genomeBuild<-, gSetList, character-method (gSetList-class), 38

- geometry, 32
- geometry, DBPDIInfo-method (geometry), 32
- geometry, FeatureSet-method (geometry), 32
- getA, 33
- getA, AlleleSet-method (AlleleSet-class), 4
- getA, SnpCnvQSet-method (getA), 33
- getA, SnpQSet-method (getA), 33
- getA, TilingFeatureSet2-method (getA), 33
- getArm (gSet-class), 37
- getArm, GenomeAnnotatedDataFrame-method (GenomeAnnotatedDataFrame-class), 29
- getArm, gSet-method (gSet-class), 37
- getBar, 34
- getCluster (setCluster), 58
- getCluster-deprecated (setCluster), 58
- getM (getA), 33
- getM, AlleleSet-method (AlleleSet-class), 4
- getM, SnpCnvQSet-method (getA), 33
- getM, SnpQSet-method (getA), 33
- getM, TilingFeatureSet2-method (getA), 33
- getSequenceLengths, 34
- GRanges, 36, 49
- GRanges-methods, 35
- gSet, 10, 61
- gSet (gSet-class), 37
- gSet-class, 37
- gSetList-class, 38
- i2p, 39, 61
- initialize, BeadStudioSet-method (BeadStudioSet-class), 10
- initialize, BeadStudioSetList-method (BeadStudioSetList-class), 11
- initialize, CNSet-method (CNSet-class), 17
- initialize, CNSetLM-method (CNSet-class), 17
- initialize, CopyNumberSet-method (CopyNumberSet-class), 18
- initialize, DBPDIInfo-method (DBPDIInfo-class), 23
- initialize, eSetList-method (ListClasses), 48
- initialize, FeatureSet-method (FeatureSet-class), 24
- initialize, GenomeAnnotatedDataFrame-method (GenomeAnnotatedDataFrame-class), 29
- initialize, gSet-method (gSet-class), 37
- initialize, gSetList-method (gSetList-class), 38
- initialize, oligoSnpSet-method (oligoSnpSet-methods), 53
- initialize, SnpSet2-method (SnpSet2-class), 61
- initialize, SnpSuperSet-method (SnpSuperSet-class), 62
- initializeBigArray (initializeBigMatrix), 40
- initializeBigMatrix, 40
- initializeBigVector (initializeBigMatrix), 40
- integer2chromosome (chromosome2integer), 16
- integerArray (integerMatrix), 41
- integerMatrix, 41
- is.ffmatrix, 42
- isPackageLoaded, 43
- isSnp, 38
- isSnp (isSnp-methods), 44
- isSnp, character-method (isSnp-methods), 44
- isSnp, GenomeAnnotatedDataFrame-method (isSnp-methods), 44
- isSnp, gSet-method (isSnp-methods), 44
- isSnp, SnpSet-method (isSnp-methods), 44
- isSnp, SummarizedExperiment-method (SummarizedExperiment-methods), 65
- isSnp-methods, 44
- kind, 44
- kind, AffyExonPDIInfo-method (kind), 44
- kind, AffyExpressionPDIInfo-method (kind), 44
- kind, AffyGenePDIInfo-method (kind), 44
- kind, AffySNPCNVPIInfo-method (kind), 44
- kind, AffySNPPDIInfo-method (kind), 44
- kind, ExpressionPDIInfo-method (kind), 44
- kind, FeatureSet-method (kind), 44
- kind, TilingPDIInfo-method (kind), 44
- ldPath (ldSetOptions), 45
- ldSetOptions, 45

- ldStatus (ldSetOptions), 45
- length, FeatureSet-method
  - (length-methods), 46
- length, gSetList-method
  - (gSetList-class), 38
- length-methods, 46
- library, 46
- library2, 46
- list.celfiles, 47
- list.files, 47
- list\_or\_ffdf, 26
- list\_or\_ffdf-class (ffdf-class), 25
- ListClasses, 48
- locusLevelData, 48, 52
- lrr (generics), 29
- lrr, BeadStudioSet-method
  - (BeadStudioSet-class), 10
- lrr, BeadStudioSetList-method
  - (BeadStudioSetList-class), 11
- lrr, SummarizedExperiment-method
  - (SummarizedExperiment-methods), 65
- lrr<- (BeadStudioSet-class), 10
- lrr<-, BafLrrSet, ANY-method
  - (BeadStudioSet-class), 10
- lrr<-, BafLrrSet-method
  - (BeadStudioSet-class), 10
- lrr<-, BafLrrSetList, matrix-method
  - (BeadStudioSetList-class), 11
- lrr<-, BeadStudioSet, ANY-method
  - (BeadStudioSet-class), 10
- lrr<-, BeadStudioSet-method
  - (BeadStudioSet-class), 10
  
- makeFeatureGRanges, 49
- makeFeatureGRanges, GenomeAnnotatedDataFrame-method
  - (GenomeAnnotatedDataFrame-class), 29
- makeFeatureGRanges, gSet-method
  - (gSet-class), 37
- makeFeatureGRanges, gSetList-method
  - (gSetList-class), 38
- manufacturer (manufacturer-methods), 50
- manufacturer, DBPDIInfo-method
  - (manufacturer-methods), 50
- manufacturer, FeatureSet-method
  - (manufacturer-methods), 50
- manufacturer-methods, 50
- matrix, 31
  
- NgsExpressionPDInfo (DBPDIInfo-class), 23
- NgsExpressionPDInfo-class
  - (DBPDIInfo-class), 23
- NgsTilingPDInfo (DBPDIInfo-class), 23
- NgsTilingPDInfo-class (DBPDIInfo-class), 23
  
- nu (AssayData-methods), 6
- nu, AssayData, character-method
  - (AssayData-methods), 6
- nu, CNSet, character-method
  - (CNSet-class), 17
- numberProbes (GRanges-methods), 35
- numberProbes, GRanges-method
  - (GRanges-methods), 35
- numberProbes, GRangesList-method
  - (GRanges-methods), 35
  
- ocLapply, 50
- ocProbesets (ocSamples), 51
- ocSamples, 51
- oldClass, 26
- oligoSet, 52
- oligoSetList, 39
- oligoSetList-class
  - (BeadStudioSetList-class), 11
- oligoSnpSet, 19
- oligoSnpSet-class
  - (oligoSnpSet-methods), 53
- oligoSnpSet-methods, 53
- open (fileConnections), 27
- open, AlleleSet-method (getA), 33
- open, array-method (fileConnections), 27
- open, CNSet-method (CNSet-class), 17
- open, matrix-method (fileConnections), 27
- open, numeric-method (fileConnections), 27
- openff (fileConnections), 27
- openff, CNSet-method (fileConnections), 27
  
- order, 15, 61
  
- p2i, 61
- p2i (i2p), 39
- parStatus, 53
- pdPkgFromBioC, 54
- phi (AssayData-methods), 6
- phi, AssayData, character-method
  - (AssayData-methods), 6



- phi, CNSet, character-method  
(CNSet-class), 17
- platform(platform-methods), 55
- platform, FeatureSet-method  
(platform-methods), 55
- platform-methods, 55
- pmFragmentLength  
(pmFragmentLength-methods), 55
- pmFragmentLength, AffySNPPDInfo-method  
(pmFragmentLength-methods), 55
- pmFragmentLength-methods, 55
- position, 38
- position(position-methods), 55
- position, AnnotatedDataFrame-method  
(position-methods), 55
- position, GenomeAnnotatedDataFrame-method  
(position-methods), 55
- position, gSet-method  
(position-methods), 55
- position, gSetList-method  
(gSetList-class), 38
- position, SnpSet-method  
(position-methods), 55
- position-methods, 55
- position<-  
(GenomeAnnotatedDataFrame-class),  
29
- position<-, GenomeAnnotatedDataFrame, integer-method  
(GenomeAnnotatedDataFrame-class),  
29
- position<-, oligoSnpSet, integer-method  
(oligoSnpSet-methods), 53
  
- read.celfiles, 24
- read.xysfiles, 24
- requireAnnotation, 56
- requireClusterPkg  
(requireClusterPkgSet), 57
- requireClusterPkg-deprecated  
(requireClusterPkgSet), 57
- requireClusterPkgSet, 57
- requireClusterPkgSet-deprecated  
(requireClusterPkgSet), 57
  
- sampleNames, FeatureSet-method  
(sampleNames-methods), 57
- sampleNames, GRanges-method  
(GRanges-methods), 35
- sampleNames, GRangesList-method  
(GRanges-methods), 35
- sampleNames, gSetList-method  
(gSetList-class), 38
- sampleNames-methods, 57
- sampleNames<-, gSetList, character-method  
(gSetList-class), 38
- scqsExample, 58
- se.exprs, FeatureSet-method  
(exprs-methods), 24
- setCluster, 58
- setCluster-deprecated(setCluster), 58
- sfsExample, 59
- show, BeadStudioSet-method  
(BeadStudioSet-class), 10
- show, CNSet-method (CNSet-class), 17
- show, DBPDPInfo-method (DBPDPInfo-class),  
23
- show, FeatureSet-method  
(FeatureSet-class), 24
- show, gSet-method (gSet-class), 37
- show, gSetList-method (gSetList-class),  
38
- sigma2, CNSet, character-method  
(CNSet-class), 17
- snpCallProbability, 60
- snpCallProbability, CNSet-method  
(CNSet-class), 17
- SnpCnvFeatureSet (FeatureSet-class), 24
- SnpCnvFeatureSet-class  
(FeatureSet-class), 24
- SNPCNVDPInfo (DBPDPInfo-class), 23
- SNPCNVDPInfo-class (DBPDPInfo-class), 23
- SnpFeatureSet (FeatureSet-class), 24
- SnpFeatureSet-class (FeatureSet-class),  
24
- SNPPDInfo (DBPDPInfo-class), 23
- SNPPDInfo-class (DBPDPInfo-class), 23
- snpRma, 34
- SnpSet, 60, 62, 63
- SnpSet-methods, 60
- SnpSet2-class, 61
- SnpSuperSet, 5
- SnpSuperSet (SnpSuperSet-class), 62
- SnpSuperSet-class, 62
- splitIndicesByLength, 63
- splitIndicesByNode  
(splitIndicesByLength), 63

sqExample, [64](#)  
state (GRanges-methods), [35](#)  
state, GRanges-method (GRanges-methods),  
[35](#)  
state, GRangesList-method  
(GRanges-methods), [35](#)  
SummarizedExperiment, [65](#)  
SummarizedExperiment-methods, [65](#)  
  
tau2, CNSet, character-method  
(CNSet-class), [17](#)  
TilingFeatureSet (FeatureSet-class), [24](#)  
TilingFeatureSet-class  
(FeatureSet-class), [24](#)  
TilingFeatureSet2 (FeatureSet-class), [24](#)  
TilingFeatureSet2-class  
(FeatureSet-class), [24](#)  
TilingPDInfo (DBPDInfo-class), [23](#)  
TilingPDInfo-class (DBPDInfo-class), [23](#)  
  
updateObject, BeadStudioSet-method  
(BeadStudioSet-class), [10](#)  
updateObject, BeadStudioSetList-method  
(BeadStudioSetList-class), [11](#)  
updateObject, CNSet-method  
(CNSet-class), [17](#)  
updateObject, GenomeAnnotatedDataFrame-method  
(GenomeAnnotatedDataFrame-class),  
[29](#)  
updateObject, oligoSnpSet-method  
(oligoSnpSet-methods), [53](#)  
  
Versioned, [4](#), [10](#), [19](#), [25](#), [29](#), [37](#), [61](#), [63](#)  
VersionedBiobase, [4](#), [10](#), [19](#), [25](#), [37](#), [61](#), [63](#)