oligoClasses

March 24, 2012

`AlleleSet-class` *Class "AlleleSet"*

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, annotation, protocolData, ...)`.

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)
```

---

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	SnpQSet, 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 (log2-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: `(antisenseThetaA+antisenseThetaB)/2` and `(senseThetaA+senseThetaB)/2`.

The average log-ratios are given by: `antisenseThetaA-antisenseThetaB` and `senseThetaA-senseThetaB`.

For Tiling data, `getM` and `getA` return the log-ratio and average log-intensities computed across channels: `M = log2(channel1)-log2(channel2) A = (log2(channel1)+log2(channel2))/2`

When large data support is enabled with the ff package, the AssayData elements of an `AlleleSet` object can be `ff_matrix` or `ffdf`, 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`

`AssayData-methods` *Methods for class AssayData in the oligoClasses package*

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 log2(A) and log2(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

```
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)))
isCurrent(test1)
assayDataElementNames(batchStatistics(test1))
## 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
test1[1:10, 1:5]
## names of elements in the object
## accessors for parameters
nu(test1, "A")[1:10, ]
nu(test1, "B")[1:10, ]
phi(test1, "A")[1:10, ]
phi(test1, "B")[1:10, ]
```

CNSet-class *Class "CNSet"*

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"), batch=new("factor"))
```

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" ~~

.__classVersion__: Object of class "Versions" ~~

Extends

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

Methods

[`signature(x = "CNSet")`: ...

A `signature(object = "CNSet")`: ...

A<- `signature(object = "CNSet")`: ...

allele `signature(object = "CNSet")`: ...

B `signature(object = "CNSet")`: ...

B<- `signature(object = "CNSet")`: ...

batch `signature(object = "CNSet")`: ...

batchNames `signature(object = "CNSet")`: ...

batchNames<- `signature(object = "CNSet")`: ...

close `signature(con = "CNSet")`: ...

coerce `signature(from="CNSetLM")`: ...

coerce `signature(from="CNSet")`: ...

corr `signature(object = "CNSet", allele = "character")`: ...

flags `signature(object="CNSet")`: SNP flags

initialize `signature(.Object = "CNSet")`: ...

nu `signature(object = "CNSet", allele = "character")`: ...

open `signature(con = "CNSet")`: ...

phi `signature(object = "CNSet", allele = "character")`: ...

sigma2 `signature(object = "CNSet", allele = "character")`: ...

tau2 `signature(object = "CNSet", allele = "character")`: ...

Author(s)

R. Scharpf

Examples

```
  if(require("genomewidesnp6Crlmm")){
require("genomewidesnp6Crlmm")
fns <- c("SNP_A-2131660", "SNP_A-1967418", "SNP_A-1969580", "SNP_A-4263484",
 "SNP_A-1978185", "SNP_A-4264431", "SNP_A-1980898", "SNP_A-1983139",
 "SNP_A-4265735", "SNP_A-1995832")
theCalls <- matrix(2, nc=2, nrow=10)
A <- matrix(sample(1:1000, 20), 10,2)
B <- matrix(sample(1:1000, 20), 10,2)
p <- matrix(runif(20), nc=2)
theConfs <- round(-1000*log2(1-p))
## Batch can be defined by the scan date of the array
##or the 96 well chemistry plate from which the
##samples were derived.  Here we indicate that the two
##samples were from the same batch.
batch <- rep(factor(1), ncol(A))
```

```
## each parameter is a R x C matrix, where the number
## of rows (R) corresponds to the number of features
## and the number of columns (C) corresponds to the
## number of batches.  In this toy example, the
## samples were assumed to be from the same batch.
## Ordinarily, one would have 50+ samples in a given
## batch.
dns <- list(fns, batch="1")
obj <- new("CNSet",
   alleleA=A,
   alleleB=B,
   call=theCalls,
   callProbability=theConfs,
   batch=as.character(rep(1, ncol(A))),
   annotation="genomewidesnp6")
assayDataElementNames(batchStatistics(obj))
featureNames(obj) <- fns
## Accessors
calls(obj)
confs(obj)
A(obj)
B(obj)
featureData(obj) <- addFeatureAnnotation(obj)
isSnp(obj)
chromosome(obj)
position(obj)
}
```

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, experimentData, annotation, protocolData, copyNumber, cnConfidence, ...)`.

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(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. cnConfidence returns a matrix of confidence scores for the copy number estimates.

DBPDInfo-class *Class "DBPDInfo"*

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

---

FeatureSet-class *"FeatureSet" and "FeatureSet" Extensions*

Description

Classes to store data from Expression/Exon/SNP/Tiling arrays at the feature level.

Objects from the Class

The FeatureSet class is VIRTUAL. Therefore users are not able to create instances of such class.

Objects for FeatureSet-like classes can be created by calls of the form: new(CLASSNAME, assayData, manufacturer, platform, exprs, phenoData, featureData, experimentData, annotation, ...). But the preferred way is using parsers like read.celfiles and read.xysfiles.

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)
```

`geometry` *Array Geometry Information*

Description

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

Usage

```
geometry(object)
```

Arguments

`object` `PDInfo` object

Examples

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

RangedData-classes *Classes in MinimumDistance for data on ranges*

Description

RangedDataCNV is a class extending the virtual class RangedDataCopyNumber. RangedDataCopyNumber extends RangedData.

RangedDataCBS extends RangedDataCNV and is useful for storing ranges from segmentation algorithms such as circular binary segmentation. In particular, the columns 'chrom', 'id', and 'num.mark' are required for instances of the class.

RangedDataHMM extends RangedDataHMM and is useful for storing ranges from hidden Markov models such as PennCNV or VanillaICE. A column labeled 'state' is required for the class.

Objects from the Class

See RangedDataCBS and RangedDataHMM constructors.

Slots

ranges: Object of class "RangesList" ~~

values: Object of class "SplitDataFrameList" ~~

elementType: Object of class "character" ~~

elementMetadata: Object of class "DataTableORNULL" ~~

metadata: Object of class "list" ~~

Extends

Class "RangedDataCNV", directly. Class "RangedDataCopyNumber", by class "RangedDataCNV", distance 2.

Class "RangedData", by class "RangedDataCNV", distance 3. Class "DataTable", by class "RangedDataCNV", distance 4. Class "List", by class "RangedDataCNV", distance 4. Class "DataTableORNULL", by class "RangedDataCNV", distance 5. Class "Vector", by class "RangedDataCNV", distance 5. Class "Annotated", by class "RangedDataCNV", distance 6.

Methods

coverage signature(object = "RangedDataCNV"): ...

state signature(object = "RangedDataCNV"): ...

todf signature(object = "RangedDataCNV", range = "ANY"): ...

Author(s)

R. Scharpf

See Also

See RangedData for a detailed description and additional methods.

Examples

```
showClass("RangedDataCNV")
```

RangedDataCNV-utils

Utility functions for RangedData extensions for storing ranged data on copy number variants.

Description

Mostly accessors for extracting data from RangedDataCBS and RangedDataHMM objects.

Usage

```
RangedDataCNV(ranges = IRanges(), values, start, end, chromosome, coverage, samp
RangedDataCBS(ranges = IRanges(), seg.mean = vector("numeric",  length(ranges)),
RangedDataHMM(ranges=IRanges(), state=vector("integer",length(ranges)),...)

coverage2(object)
state(object)
```

Arguments

object	A RangedDataHMM or a RangedDataCNV object.
ranges	An instance of IRanges class.
values	A DataFrame object.
start	integer – physical position indicating start of copy number segment
end	integer – physical position indicating end of copy number segment
chromosome	integer indicating chromosome
sampleId	character string
startIndexInChromosome	index of marker within the chromosome for the beginning of the copy number segment. The physical position of this marker is given by 'start'. Optional
endIndexInChromosome	index of marker within the chromosome for the end of the copy number segment. The physical position of this marker is given by 'end'. Optional
seg.mean	Numeric – e.g., the mean copy number of a genomic interval
coverage	number of markers in a segment
state	typically an integer corresponding to the inferred copy number from a hidden markov model
...	Additional covariates that can be accessed by $ method

Details

RangedDataCNV, RangedDataHMM, and RangedDataCBS are constructors for the corresponding class.

Value

`coverage2` and `state` return a column (covariate) of the `RangedData` object. Coverage refers to the number of markers in the interval. State is a method for `RangedDataHMM` objects and returns the copy number state, typically estimated from a HMM.

Author(s)

R. Scharpf

See Also

See `RangedData` for additional details and methods for objects of the class.

`RangedDataHMM`

Examples

```
if(require("IRanges")){
  ranges <- IRanges(c(1,2,3),c(4,5,6))
  chrom <- 1:3
  id <- letters[1:3]
  num.mark <- rpois(3, 10)
  seg.mean <- rnorm(3)
  rd <- RangedDataCBS(ranges=ranges,
      chromosome=chrom,
      sampleId=id,
      coverage=num.mark,
      seg.mean=seg.mean)
  ## accessors
  chromosome(rd)
  sampleNames(rd)
  coverage2(rd)

  rd.hmm <- RangedDataHMM(ranges=ranges,
  chromosome=chrom,
  sampleId=id,
  coverage=num.mark,
  state=2L)
  ## accessors
  chromosome(rd.hmm)
  sampleNames(rd.hmm)
  coverage2(rd.hmm)
  state(rd.hmm)
}
```

`SnpSet-methods` *Accessors and methods for SnpSet objects*

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 SnpSet 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 ffdf, 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

See addFeatureAnnotation for adding chromosome, physical position, and an indicator for whether the marker is polymorphic to the featureData slot of a SnpSet object.

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("SnpSet", call=theCalls, callProbability=integerRepresentation)
calls(obj)
p2 <- confs(obj)
dimnames(p2) <- NULL
all.equal(p2, p) ## small differences due to rounding
        int <- snpCallProbability(obj) ## not necessarily a probability
        p3 <- i2p(int)  ## to convert back to a probability
        all.equal(p3, p2)

        int <- snpCallProbability(obj) ## not necessarily a probability
        p3 <- i2p(int)  ## to convert back to a probability
        all.equal(p3, p2)

## example using ff
if(require(ff)){
ldPath(tempdir())
integerRepresentation <- initializeBigMatrix("tmp", 10, 2)
for(j in 1:2) integerRepresentation[, j] <- as.integer(round(-1000*log(1-p[, j])))
integerRepresentation
obj <- new("SnpSet", call=theCalls, callProbability=integerRepresentation)
calls(obj)
res <- tryCatch(confs(obj), error=function(e) NULL)
is.null(res)
integerRepresentation <- snpCallProbability(obj)
##coerce to matrix by subsetting desired rows and columns (here we choose all rows and co
integerRepresentation <- integerRepresentation[,]
p3 <- oligoClasses:::i2p(integerRepresentation)
dimnames(p3) <- NULL
all.equal(p2, p3)
}
```

`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)
```

---

addFeatureAnnotation

Add genomic annotation (chromosome, position) for several SNP platforms.

Description

Adds chromosome, position, and an indicator for whether the locus is polymorphic.

Usage

```
addFeatureAnnotation(object)
```

Arguments

object	An object extending the eSet class.

Value

An AnnotatedDataFrame.

Author(s)

R. Scharpf

Examples

```
if(require(pd.genomewidesnp.6)){
conn <- db(pd.genomewidesnp.6)
dbListTables(conn)
dbListFields(conn, "featureSet")
## get 5 snp identifiers
##sql <- "SELECT man_fsetid FROM featureSet WHERE man_fsetid LIKE 'SNP%' LIMIT 5"
sql <- "SELECT man_fsetid FROM featureSet LIMIT 5"
ids <- dbGetQuery(conn, sql)[[1]]
A <- B <- matrix(rnorm(25), 5, 5, dimnames=list(ids, LETTERS[1:5]))
obj <- new("AlleleSet",
   alleleA=A,
   alleleB=B,
   annotation="pd.genomewidesnp.6")
featureData(obj) <- addFeatureAnnotation(obj)
fData(obj)

##check against annotation package
##sql <- "SELECT man_fsetid, chrom, physical_pos FROM featureSet WHERE man_fsetid LIKE 'S
##dbGetQuery(conn, sql)
}
if(require(genomewidesnp6Crlmm)){
##alternatively, could use the Crlmm annotation package
obj2 <- new("AlleleSet",
   alleleA=A,
   alleleB=B,
   annotation="genomewidesnp6")
featureData(obj2) <- addFeatureAnnotation(obj2)
fData(obj2)
}
```

`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()
```

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

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. Analysis algorithms that do not adjust for batch effects are prone to spurious measures of association. While 'batch' is often unknown, a useful surrogates are the scan date of the arrays 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 factor that has the same length as the number of samples in the CNSet object.

The method 'batchNames' returns the unique batches as a character string. The batch labels for each element in the LinearModelParameter class can be reassigned using the 'batchNames<-' replacement method.

Author(s)

R. Scharpf

See Also

CNSet-class

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)))
batchNames(test1) ##unique batches
batch(test1)
test1[1:20, 1:10]
##just NA's
nu(test1, "A")[1:10, ]
## similarly for the B allele
##nu(test1, "B")
##phi(test1, "A")
##phi(test1, "B")
## using ff objects
if(require(ff)){
x2 <- initializeBigMatrix("smallx", nr=250, nc=96*2)
x2[,] <- as.numeric(x)
test2 <- new("CNSet", alleleA=x, alleleB=x, call=x, callProbability=x, batch=as.character
test2
batchNames(test2) ##unique batches
batch(test2)
## ff objects
class(nu(test2, "A"))
(test2.sub <- test2[1:20, 1:10])
## after subsetting, all elements are matrices
class(nu(test2.sub, "A"))
}
```

---

batchStatistics *Accessor for batch statistics uses for copy number estimation and storage of model parameters*

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

---

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)
```

`checkExists` *Checks to see whether an object exists and, if not, executes the appropriate function.*

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 indicaged 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` *Checks whether a eSet-derived class is ordered by chromosome and physical position*

Description

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

Usage

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

Arguments

`object`	A `SnpSet` or `CopyNumberSet`.
`verbose`	Logical.

Details

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

Value

Logical

Author(s)

R. Scharpf

See Also

`order`

Examples

```
data(oligoSetExample)
checkOrder(oligoSet)
oligoSet2 <- order(oligoSet)
checkOrder(oligoSet2)
```

chromosome2integer *Converts chromosome to integer*

Description

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

Usage

```
chromosome2integer(chrom)
```

Arguments

chrom	chromosome

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

integer character

Author(s)

R. Scharpf

Examples

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

setCluster *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()
ocSamples(n)
ocProbesets(n)
```

Arguments

...	arguments to be passed to `makeCluster` in the 'snow' package.
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 '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 <carvalho@bclab.org>

`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

---

efsExample *ExpressionFeatureSet Object*

Description

Example of ExpressionFeatureSet Object.

Usage

```
data(efsExample)
```

Format

Object belongs to ExpressionFeatureSet class.

Examples

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

---

scqsExample *SnpCnvQSet Example*

Description

Example of SnpCnvQSet object.

Usage

```
data(scqsExample)
```

Format

Object belongs to SnpCnvQSet class.

Examples

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

sfsExample *SnpFeatureSet Example*

Description

Example of SnpFeatureSet object.

Usage

```
data(sfsExample)
```

Format

Object belongs to SnpFeatureSet class

Examples

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

sqsExample *SnpQSet Example*

Description

Example of SnpQSet instance.

Usage

```
data(sqsExample)
```

Format

Belongs to SnpQSet class.

Examples

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

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)
```

eSet-methods — *Accessors for eSet extensions*

Description

Accessors for variables stored in the featureData slot of a class inheriting from eSet.

Usage

```
chromosome(object,na.rm=FALSE)
chromosome(object) <- value
position(object,na.rm=FALSE)
isSnp(object,pkgname)
##snpNames(object)
```

Arguments

object	A eSet object or AnnotatedDataFrame.
na.rm	Logical. Whether to exlude NA's.
value	A integer vector.
pkgname	A character string indicating the annotation package.

Value

position returns an integer vector of the genomic position position (units are base pairs).

chromosome returns an integer vector of the chromosome. See chromosome2integer for the integer coding of non-autosomal chromosomes.

isSnp returns a logical vector that indicates which markers are polymorphic. If object is character vector of feature names, the pkgname must be supplied. For nonpolymorphic markers, isSnp evaluates to FALSE.

See Also

chromosome2integer, annotationPackages

---

exprs-methods *Accessor for the 'exprs' slot*

Description

Accessor for the 'exprs'/'se.exprs' slot of FeatureSet-like objects

Methods

object = "ExpressionSet" Expression matrix for objects of this class. Usually results of preprocessing algorithms, like RMA.

object = "FeatureSet" General container 'exprs' inherited from eSet

object = "SnpSet" General container 'exprs' inherited from eSet, not yet used.

---

featuresInRange *Find feature index in a genomic interval*

Description

Find feature index of a SnpSet object in a RangedDataCNV object.

Usage

```
featuresInRange(object, range, FRAME = 0, FRAME.LEFT, FRAME.RIGHT, ...)
```

Arguments

object	A SnpSet.
range	A RangedDataCNV object.
FRAME	Integer (basepairs). The distance from start and end coordinates of the genomic interval. Useful for retrieving indices that 'frame' the genomic interval of interest.
FRAME.LEFT	Integer. If missing, set equal to FRAME.
FRAME.RIGHT	Integer. If missing, set equal to FRAME.
...	Ignored

Details

Extract marker indices in object that occur within a genomic interval.

Value

Vector of integers.

Author(s)

R. Scharpf

---

ff_matrix-class *Class "ff_matrix"*

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")
```

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 `ffdf` ~~

Extends

Class "`oldClass`", directly. Class "`list_or_ffdf`", directly.

Methods

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

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 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))
```

`findOverlaps` *Find markers that overlap with a query set of ranges*

Description

Methods for finding overlapping genomic ranges.

Usage

```
findOverlaps(query, subject, maxgap = 0L, minoverlap = 1L, type = c("any", "star
```

Arguments

`query`	A `RangedDataCNV` object.
`subject`	A `SnpSet`, a `CNSet` or the `featureData` from these classes. The `featureData` must be an `AnnotatedDataFrame`.
`maxgap`	Passed to `findOverlaps` method for a `IRanges` query and a `IRanges` subject.
`minoverlap`	Passed to `findOverlaps` method for a `IRanges` query and a `IRanges` subject.
`type`	Passed to `findOverlaps` method for a `IRanges` query and a `IRanges` subject.
`select`	Passed to `findOverlaps` method for a `IRanges` query and a `IRanges` subject.
`...`	Passed to `findOverlaps` method for a `IRanges` query and a `IRanges` subject.

Details

When both `query` and `subject` are `RangedDataCNV` objects, we require that the overlapping ranges have the same chromosome and sample id. If `query` and `subject` are `RangedDataHMM` objects we additionally require that the `state` value in the `RangedDataHMM` objects match. For example, if the same genomic interval for a subject is called a 'deletion' in `query` and 'normal' in `subject`, the interval is not matched. The primary purpose of such queries is to assess the concordance of hidden Markov models applied to the same dataset.

If `subject` is a `SnpSet`, `CNSet`, or `AnnotatedDataFrame` with 'chromosome', and 'position' `varLabels`, the method returns a `RangesMatching` object indicating which markers in `subject` overlap with the `query` ranges. This method can be useful for finding the set of markers in `subject` that reside within a given genomic interval in the `query`.

Value

A RangesMatching object.

Author(s)

R. Scharpf

Examples

```
if(require("VanillaICE")){
data(hmmResults, package="VanillaICE")
data(oligoSetExample)
## Find markers in a oligoSnpSet object that overlap with the
## 2nd range:
rmatching <- findOverlaps(hmmResults[2, ], oligoSet)
index.in.range <- matchMatrix(rmatching)[,2]
features.in.range <- featureNames(oligoSet)[index.in.range]
##
## For two RangedDataHMM objects, returns only the ranges
## that have the same sample name, chromosome, and HMM state
## A trivial example:
hmmResults2 <- hmmResults
hmmResults2$state <- rep(3, nrow(hmmResults))
findOverlaps(hmmResults, hmmResults2)
}
```

---

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` *Miscellaneous generics. Methods defined in packages that depend on oligoClasses*

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

`genomeBuild` *Genome Build Information*

Description

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

Usage

```
genomeBuild(object)
```

Arguments

`object`	`PDInfo` or `FeatureSet` object.

getBar — *Gets a bar of a given length.*

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())
```

i2p — *Functions to convert probabilities to integers, or integers to probabilities.*

Description

Probabilities estimated in the crlmm 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 `2pi` is

as.integer(-1000*log(1-p))

See Also

`confs`

Examples

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

---

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 *Check if package is loaded.*

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")
```

kind *Array type*

Description

Retrieves the array type.

Usage

```
kind(object)
```

Arguments

object	FeatureSet or DBPDInfo object

Value

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

Examples

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

---

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",
initializeBigVector(name=basename(tempfile()), n=0L, vmode = "integer", initdata
```

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

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/matrix is returned otherwise.

Examples

```
x <- initializeBigVector("test", 10)
class(x)
x
if (isPackageLoaded("ff"))
  finalizer(x) <- "delete"
rm(x)
```

`ldSetOptions` *Set/check large dataset options.*

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 *Number of samples for FeatureSet-like objects.*

Description

Number of samples for FeatureSet-like objects.

Methods

x = "FeatureSet" Number of samples

list.celfiles *List CEL files.*

Description

Function used to get a list of CEL files.

Usage

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

Arguments

`...`	Passed to `list.files`
`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 won't return anything
  ## if in the working directory there isn't any CEL
  list.celfiles(getwd())
}
```

`locusLevelData` *Basic data elements required for the HMM*

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)
```

`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, setCluster, parStatus

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=log2(locusLevelData[["copynumber"]]/100),
call=locusLevelData[["genotypes"]],
callProbability=locusLevelData[["crlmmConfidence"]],
annotation=locusLevelData[["platform"]])
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

order *Methods for CopyNumberSet objects*

Description

Methods for objects of class `CopyNumberSet`.

Usage

```
order(..., na.last=TRUE, decreasing=FALSE)
```

Arguments

`...`	blah blah
`na.last`	Ignored
`decreasing`	Ignored

Details

Reorders the object by chromosome and physical position.

Value

An object of the same class as the first element in ....

See Also

`chromosome`, `position`

Examples

```
data(oligoSetExample)
oligoSet2 <- order(oligoSet)
```

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

pkgname	character. Name of the package to be installed.
lib	character. Path where to install the package at.
verbose	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.

requireAnnotation

Helper function to load packages.

Description

This function checkes 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` *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

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

Description

Tools to distribute objects across nodes or by length.

Usage

```
splitIndicesByLength(x, lg)
splitIndicesByNode(x)
```

Arguments

`x`	object to be split
`lg`	length

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)
splitIndicesByNode(x)
```

Index