

oligoClasses

October 5, 2010

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 = "SnFeatureSet"): tests if data contains allele summaries on both strands for a given SnFeatureSet.

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")
```

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, ...)
open(con, ...)
close(con, ...)
```

Arguments

object	SnpQSet, SnpCnvQSet or TilingFeatureSet2 object.
con	AlleleSet or AlleleSet extension.
...	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: $(\text{antisenseThetaA} + \text{antisenseThetaB}) / 2$ 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 `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](#)

CNSet-class

Class "CNSet"

Description

A class to store copy number estimates as well as locus-level summaries of the normalized intensities

Objects from the Class

Objects can be created by calls of the form `new("CNSet", call, CA, CB, callProbability, alleleA, alleleB, phenoData, featureData, experimentData, protocolData, segmentData, emissionPr, position, chromosome, isSnp, annotation, ...)`.

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 "[SnpSuperSet](#)", directly. Class "[AlleleSet](#)", by class "SnpSuperSet", distance 2. Class "[SnpSet](#)", by class "SnpSuperSet", distance 2. Class "[eSet](#)", by class "SnpSuperSet", distance 3. Class "[VersionedBiobase](#)", by class "SnpSuperSet", distance 4. Class "[Versioned](#)", by class "SnpSuperSet", distance 5.

Methods

CA signature(object = "CNSet"): ...
CA<- signature(object = "CNSet", value = "matrix"): ...
CB signature(object = "CNSet"): ...
CB<- signature(object = "CNSet", value = "matrix"): ...
initialize signature(.Object = "CNSet"): ...

Author(s)

R. Scharpf

See Also

[AlleleSet](#), [SnpSuperSet](#)

Examples

```
## Not run:
theCalls <- matrix(2, nc=2, nrow=10)
A <- matrix(sample(1:1000, 20), 10, 2)
B <- matrix(sample(1:1000, 20), 10, 2)
CA <- matrix(rnorm(20, 1), nrow=10)
CB <- matrix(rnorm(20, 1), nrow=10)
p <- matrix(runif(20), nc=2)
theConfs <- round(-1000*log2(1-p))
obj <- new("CNSet",
  alleleA=A,
  alleleB=B,
  call=theCalls,
  callProbability=theConfs,
  CA=CA,
  CB=CB)

## Accessors
calls(obj)
confs(obj)
A(obj)
B(obj)
CA(obj)
CB(obj)

##with annotation
if(require("genomewidesnp6Crlmm")){
ids <- 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")
rownames(theCalls) <- rownames(p) <- rownames(theConfs) <- ids
rownames(A) <- rownames(B) <- rownames(CA) <- rownames(CB) <- ids
```

```

obj2 <- new("CNSet",
  alleleA=A,
  alleleB=B,
  call=theCalls,
  callProbability=theConfs,
  CA=CA,
  CB=CB, annotation="genomewidesnp6")
fvarLabels(obj2)

## additional accessors
isSnp(obj2)
chromosome(obj2)
position(obj2)
}

## End(Not run)

```

CopyNumberSet-methods

Methods for class CopyNumberSet.

Description

Accessors and CopyNumberSet

Usage

```

copyNumber(object)
cnConfidence(object)

```

Arguments

object CopyNumberSet object

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)
```

SnpSet-methods

Accessors for Calls and Confidences on a SnpSet object

Description

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

`confs` returns the confidences associated to the genotype calls. The current implementation of CRLMM stores the confidences as integers by using the transformation:

$$\text{conf} = \text{round}(-1000 \cdot \log_2(1-p)),$$

where 'p' is the posterior probability of the call.

Methods

`initialize(SnpSet)`: Object instantiation, used by `new`; not to be called directly by the user.

`calls(object)`: accessor for genotype calls

`confs(object)`: accessor for crlmm genotype confidence scores

See Also

[addFeatureAnnotation](#)

Examples

```

theCalls <- matrix(sample(1:3, 20, rep=TRUE), nc=2)
p <- matrix(runif(20), nc=2)
theConfs <- round(-1000*log2(1-p))
obj <- new("SnpSet", call=theCalls, callProbability=theConfs)
calls(obj)
confs(obj)

## if(require("genomewidesnp6Crlmm")){
## ids <- 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")
## rownames(theCalls) <- rownames(p) <- rownames(theConfs) <- ids
## obj <- new("SnpSet", call=theCalls, callProbability=theConfs,
## annotation="genomewidesnp6")
## obj2 <- annotate(obj)
## fvarLabels(obj2)
## }

```

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", assayData, phenoData, featureData, experimentData, annotation, protocolData, 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")
```

```
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 <- "WHERE 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

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)
```

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.

Methods

```
signature(object = "eSet") ...
```

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.

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())
```

`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, nr, nc, vmode = "integer", initdata = NA)
initializeBigVector(name, n, 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

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

Description

Set/check large dataset options.

Usage

```
ldSetOptions(nsamples=100, nprobesets=1000, 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

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 [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())
}

```

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

oligoSnpSet-methods

Methods for oligoSnpSet class

Description

Methods for oligoSnpSet

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.

position	<i>Accessor to position information</i>
----------	---

Description

position will return the genomic position of a SNP.

Usage

```
position(object)
```

Arguments

object object inheriting from SnpLevelSet

Details

position will return genomic position of a SNP (number of basepairs from the 5-prime chromosomal end)

Value

an integer

Author(s)

R. Scharpf

requireAnnotation	<i>Helper function to load packages.</i>
-------------------	--

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
```

Package loaders for clusters.

Description

Package loaders for clusters.

Usage

```
requireClusterPkgSet (packages)
requireClusterPkg (...)
```

Arguments

packages	character vector with the names of the packages to be loaded on the compute nodes.
...	same arguments as <code>require</code>

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

*Topic **IO**

list.celfiles, 22

*Topic **classes**

AlleleSet-class, 1
CNSet-class, 3
DBPDInfo-class, 5
FeatureSet-class, 6
SnpSuperSet-class, 8

*Topic **datasets**

efsExample, 14
scqsExample, 14
sfsExample, 15
sqcExample, 15

*Topic **data**

pdPkgFromBioC, 24
requireAnnotation, 26

*Topic **list**

affyPlatforms, 10

*Topic **manip**

addFeatureAnnotation, 9
celfileDate, 11
chromosome2integer, 12
CopyNumberSet-methods, 5
createFF, 13
eSet-methods, 16
genomeBuild, 17
geometry, 7
getA, 2
getBar, 17
initializeBigMatrix, 20
is.ffmatrix, 18
isPackageLoaded, 19
kind, 19
ldSetOptions, 21
ocLapply, 23
parStatus, 24
position, 26
requireClusterPkgSet, 27
setCluster, 12
SnpSet-methods, 7
splitIndicesByLength, 28

*Topic **methods**

CopyNumberSet-methods, 5

db, 16

eSet-methods, 16
exprs-methods, 17
length-methods, 22
manufacturer-methods, 23
oligoSnpSet-methods, 24
platform-methods, 25
pmFragmentLength-methods, 25
sampleNames-methods, 28

*Topic **misc**

affyPlatforms, 10

*Topic **utilities**

list.celfiles, 22

A (*getA*), 2

A, AlleleSet-method (*getA*), 2

A<- (*getA*), 2

A<-, AlleleSet, matrix-method
(*getA*), 2

A<-, AlleleSet-method (*getA*), 2

addFeatureAnnotation, 8, 9

AffyExonPDInfo-class
(*DBPDInfo-class*), 5

AffyExpressionPDInfo-class
(*DBPDInfo-class*), 5

AffyGenePDInfo-class
(*DBPDInfo-class*), 5

affyPlatforms, 10

AffySNPCNVPDInfo-class
(*DBPDInfo-class*), 5

AffySNPPDInfo-class
(*DBPDInfo-class*), 5

AffySTPDInfo-class
(*DBPDInfo-class*), 5

AffyTilingPDInfo-class
(*DBPDInfo-class*), 5

allele (*AlleleSet-class*), 1

allele, AlleleSet-method
(*AlleleSet-class*), 1

AlleleSet, 4, 8, 9

AlleleSet-class, 1

annotation, DBPDInfo-method
(*DBPDInfo-class*), 5

annotationPackages, 11

- B (*getA*), 2
- B, *AlleleSet*-method (*getA*), 2
- B<- (*getA*), 2
- B<-, *AlleleSet*, matrix-method (*getA*), 2
- B<-, *AlleleSet*-method (*getA*), 2
- bothStrands (*AlleleSet*-class), 1
- bothStrands, *AlleleSet*-method (*AlleleSet*-class), 1
- bothStrands, *SnpFeatureSet*-method (*AlleleSet*-class), 1

- CA (*CNSet*-class), 3
- CA, *CNSet*-method (*CNSet*-class), 3
- CA<- (*CNSet*-class), 3
- CA<-, *CNSet*, matrix-method (*CNSet*-class), 3
- CA<-, *CNSet*-method (*CNSet*-class), 3
- calls (*SnpSet*-methods), 7
- calls, *oligoSnpSet*-method (*oligoSnpSet*-methods), 24
- calls, *SnpSet*-method (*SnpSet*-methods), 7
- calls<- (*SnpSet*-methods), 7
- calls<-, *oligoSnpSet*, matrix-method (*oligoSnpSet*-methods), 24
- calls<-, *SnpSet*, matrix-method (*SnpSet*-methods), 7
- callsConfidence, *oligoSnpSet*-method (*oligoSnpSet*-methods), 24
- callsConfidence<-, *oligoSnpSet*, matrix-method (*oligoSnpSet*-methods), 24
- CB (*CNSet*-class), 3
- CB, *CNSet*-method (*CNSet*-class), 3
- CB<- (*CNSet*-class), 3
- CB<-, *CNSet*, matrix-method (*CNSet*-class), 3
- CB<-, *CNSet*-method (*CNSet*-class), 3
- celfileDate, 11
- chromosome (*eSet*-methods), 16
- chromosome, *eSet*-method (*eSet*-methods), 16
- chromosome2integer, 12
- chromosome<- (*eSet*-methods), 16
- chromosome<-, *eSet*-method (*eSet*-methods), 16
- close (*getA*), 2
- close, *AlleleSet*-method (*getA*), 2
- cnConfidence
 - (*CopyNumberSet*-methods), 5
- cnConfidence, *CopyNumberSet*-method (*CopyNumberSet*-methods), 5
- cnConfidence, *oligoSnpSet*-method (*oligoSnpSet*-methods), 24
- cnConfidence<-
 - (*CopyNumberSet*-methods), 5
- cnConfidence<-, *CopyNumberSet*, matrix-method (*CopyNumberSet*-methods), 5
- cnConfidence<-, *oligoSnpSet*, matrix-method (*oligoSnpSet*-methods), 24
- CNSet, 2
- CNSet-class, 3
- cnfs (*SnpSet*-methods), 7
- cnfs, *SnpSet*-method (*SnpSet*-methods), 7
- cnfs<- (*SnpSet*-methods), 7
- cnfs<-, *SnpSet*, matrix-method (*SnpSet*-methods), 7
- copyNumber
 - (*CopyNumberSet*-methods), 5
- copyNumber, *CopyNumberSet*-method (*CopyNumberSet*-methods), 5
- copyNumber, *oligoSnpSet*-method (*oligoSnpSet*-methods), 24
- copyNumber<-
 - (*CopyNumberSet*-methods), 5
- copyNumber<-, *CopyNumberSet*, matrix-method (*CopyNumberSet*-methods), 5
- copyNumber<-, *oligoSnpSet*, matrix-method (*oligoSnpSet*-methods), 24
- CopyNumberSet-methods, 5
- createFF, 13
- db, 16
- db, *AlleleSet*-method (*AlleleSet*-class), 1
- db, *DBPDInfo*-method (*db*), 16
- db, *eSet*-method (*db*), 16
- db, *FeatureSet*-method (*db*), 16
- db, *SnpcnvQSet*-method (*db*), 16
- db, *SnpQSet*-method (*db*), 16
- db-methods (*db*), 16
- DBPDInfo-class, 5
- delCluster (*setCluster*), 12

- efsExample, 14
- eSet, 1, 4, 7, 8
- eSet-methods, 16
- ExonFeatureSet-class (*FeatureSet*-class), 6
- ExpressionFeatureSet-class (*FeatureSet*-class), 6
- ExpressionPDInfo-class (*DBPDInfo*-class), 5

- exprs, FeatureSet-method
(*exprs-methods*), 17
- exprs-methods, 17
- FeatureSet-class, 6
- GeneFeatureSet-class
(*FeatureSet-class*), 6
- genomeBuild, 17
- genomeBuild, DBPDPInfo-method
(*genomeBuild*), 17
- genomeBuild, FeatureSet-method
(*genomeBuild*), 17
- geometry, 7
- geometry, DBPDPInfo-method
(*geometry*), 7
- getA, 2
- getA, AlleleSet-method
(*AlleleSet-class*), 1
- getA, SnpCnvQSet-method (*getA*), 2
- getA, SnpQSet-method (*getA*), 2
- getA, TilingFeatureSet2-method
(*getA*), 2
- getBar, 17
- getCluster (*setCluster*), 12
- getM (*getA*), 2
- getM, AlleleSet-method
(*AlleleSet-class*), 1
- getM, SnpCnvQSet-method (*getA*), 2
- getM, SnpQSet-method (*getA*), 2
- getM, TilingFeatureSet2-method
(*getA*), 2
- initialize, CNSet-method
(*CNSet-class*), 3
- initialize, CopyNumberSet-method
(*CopyNumberSet-methods*), 5
- initialize, DBPDPInfo-method
(*DBPDPInfo-class*), 5
- initialize, FeatureSet-method
(*FeatureSet-class*), 6
- initialize, oligoSnpSet-method
(*oligoSnpSet-methods*), 24
- initialize, SnpSuperSet-method
(*SnpSuperSet-class*), 8
- initializeBigMatrix, 20
- initializeBigVector
(*initializeBigMatrix*), 20
- is.ffmatrix, 18
- isPackageLoaded, 19
- isSnp (*eSet-methods*), 16
- isSnp, eSet-method (*eSet-methods*),
16
- kind, 19
- kind, AffyExonPDPInfo-method
(*kind*), 19
- kind, AffyExpressionPDPInfo-method
(*kind*), 19
- kind, AffyGenePDPInfo-method
(*kind*), 19
- kind, AffySNPCNVDPInfo-method
(*kind*), 19
- kind, AffySNPPDPInfo-method (*kind*),
19
- kind, ExpressionPDPInfo-method
(*kind*), 19
- kind, FeatureSet-method (*kind*), 19
- kind, TilingPDPInfo-method (*kind*),
19
- ldPath (*ldSetOptions*), 21
- ldSetOptions, 21
- ldStatus (*ldSetOptions*), 21
- length, FeatureSet-method
(*length-methods*), 22
- length-methods, 22
- list.celfiles, 22
- list.files, 22
- manufacturer
(*manufacturer-methods*), 23
- manufacturer, DBPDPInfo-method
(*manufacturer-methods*), 23
- manufacturer, FeatureSet-method
(*manufacturer-methods*), 23
- manufacturer-methods, 23
- NgsExpressionPDPInfo-class
(*DBPDPInfo-class*), 5
- NgsTilingPDPInfo-class
(*DBPDPInfo-class*), 5
- ocLapply, 23
- ocProbesets (*setCluster*), 12
- ocSamples (*setCluster*), 12
- oligoSnpSet-class
(*oligoSnpSet-methods*), 24
- oligoSnpSet-methods, 24
- open (*getA*), 2
- open, AlleleSet-method (*getA*), 2
- parStatus, 24
- pdPkgFromBioC, 24
- platform (*platform-methods*), 25
- platform, FeatureSet-method
(*platform-methods*), 25

- platform-methods, [25](#)
- pmFragmentLength
 - (*pmFragmentLength-methods*), [25](#)
- pmFragmentLength, AffySNPPDInfo-method
 - (*pmFragmentLength-methods*), [25](#)
- pmFragmentLength-methods, [25](#)
- position, [26](#)
- position, eSet-method
 - (*eSet-methods*), [16](#)

- read.celfiles, [6](#)
- read.xysfiles, [6](#)
- requireAnnotation, [26](#)
- requireClusterPkg
 - (*requireClusterPkgSet*), [27](#)
- requireClusterPkgSet, [27](#)

- sampleNames, FeatureSet-method
 - (*sampleNames-methods*), [28](#)
- sampleNames-methods, [28](#)
- scqsExample, [14](#)
- se.exprs, FeatureSet-method
 - (*exprs-methods*), [17](#)
- setCluster, [12](#)
- sfsExample, [15](#)
- show, DBPDInfo-method
 - (*DBPDInfo-class*), [5](#)
- show, FeatureSet-method
 - (*FeatureSet-class*), [6](#)
- SnpcnvFeatureSet-class
 - (*FeatureSet-class*), [6](#)
- SNPCNVPDInfo-class
 - (*DBPDInfo-class*), [5](#)
- SnpcnvFeatureSet-class
 - (*FeatureSet-class*), [6](#)
- SNPPDInfo-class (*DBPDInfo-class*), [5](#)
- snprma, [3](#)
- SnpSet, [4, 8](#)
- SnpSet-methods, [7](#)
- SnpSuperSet, [2, 4](#)
- SnpSuperSet-class, [8](#)
- splitIndicesByLength, [28](#)
- splitIndicesByNode
 - (*splitIndicesByLength*), [28](#)
- sqqsExample, [15](#)

- TilingFeatureSet-class
 - (*FeatureSet-class*), [6](#)
- TilingFeatureSet2-class
 - (*FeatureSet-class*), [6](#)
- TilingPDInfo-class
 - (*DBPDInfo-class*), [5](#)
- Versioned, [1, 4, 7, 8](#)
- VersionedBiobase, [1, 4, 7, 8](#)