

Package ‘SummarizedExperiment’

April 15, 2017

Title SummarizedExperiment container

Description The SummarizedExperiment container contains one or more assays, each represented by a matrix-like object of numeric or other mode. The rows typically represent genomic ranges of interest and the columns represent samples.

Version 1.4.0

Encoding UTF-8

Author Martin Morgan, Valerie Obenchain, Jim Hester, Hervé Pagès

Maintainer Bioconductor Package Maintainer <maintainer@bioconductor.org>

biocViews Genetics, Infrastructure, Sequencing, Annotation, Coverage, GenomeAnnotation

Depends R (>= 3.2), methods, GenomicRanges (>= 1.23.15), Biobase

Imports utils, stats, Matrix, BiocGenerics (>= 0.15.3), S4Vectors (>= 0.11.7), IRanges (>= 2.7.2), GenomeInfoDb

Suggests annotate, AnnotationDbi, hgu95av2.db, GenomicFeatures, TxDb.Hsapiens.UCSC.hg19.knownGene, BiocStyle, knitr, rmarkdown, digest, jsonlite, rhdf5, airway, RUnit

VignetteBuilder knitr

License Artistic-2.0

Collate Assays-class.R SummarizedExperiment-class.R
RangedSummarizedExperiment-class.R intra-range-methods.R
inter-range-methods.R coverage-methods.R findOverlaps-methods.R
nearest-methods.R makeSummarizedExperimentFromExpressionSet.R
makeSummarizedExperimentFromDataFrame.R readKallisto.R zzz.R

NeedsCompilation no

R topics documented:

Assays-class	2
coverage-methods	4
findOverlaps-methods	5
inter-range-methods	6
intra-range-methods	8
makeSummarizedExperimentFromDataFrame	10
makeSummarizedExperimentFromExpressionSet	11

nearest-methods	13
RangedSummarizedExperiment-class	15
readKallisto	18
SummarizedExperiment-class	20

Index	26
--------------	-----------

Assays-class	<i>Assays objects</i>
--------------	-----------------------

Description

The Assays virtual class and its methods provide a formal abstraction of the assays slot of [SummarizedExperiment](#) objects.

`SimpleListAssays` and `ShallowSimpleListAssays` are concrete subclasses of `Assays` with the latter being currently the default implementation of `Assays` objects. Other implementations (e.g. disk-based) could easily be added.

Note that these classes are not meant to be used directly by the end-user and the material in this man page is aimed at package developers.

Details

`Assays` objects have a list-like semantics with elements having matrix- or array-like semantics (e.g., `dim`, `dimnames`).

The `Assays` API consists of:

- (a) The `Assays()` constructor function.
- (b) Lossless back and forth coercion from/to [SimpleList](#). The coercion method from `SimpleList` doesn't need (and should not) validate the returned object.
- (c) `length`, `names`, ``names<-``, `[[`, ``[[<-``, `dim`, `[[<-``, `rbind`, `cbind`.

An `Assays` concrete subclass needs to implement (b) (required) plus, optionally any of the methods in (c).

IMPORTANT: Methods that return a modified `Assays` object (a.k.a. endomorphisms), that is, `[[` as well as replacement methods `names<-``, `[[<-``, and `[[<-``, must respect the *copy-on-change contract*. With objects that don't make use of references internally, the developer doesn't need to take any special action for that because it's automatically taken care of by R itself. However, for objects that do make use of references internally (e.g. environments, external pointers, pointer to a file on disk, etc...), the developer needs to be careful to implement endomorphisms with copy-on-change semantics. This can easily be achieved (and is what the default methods for `Assays` objects do) by performing a full (deep) copy of the object before modifying it instead of trying to modify it in-place. Note that the full (deep) copy is not always necessary in order to achieve copy-on-change semantics: it's enough (and often preferable for performance reasons) to copy only the parts of the objects that need to be modified.

`Assays` has currently 3 implementations which are formalized by concrete subclasses `SimpleListAssays`, `ShallowSimpleListAssays`, and `AssaysInEnv`. `ShallowSimpleListAssays` is the default. `AssaysInEnv` is a *broken* alternative to `ShallowSimpleListAssays` that does NOT respect the *copy-on-change contract*. It is only provided for illustration purposes (see source file `Assays-class.R` for the details).

A little more detail about ShallowSimpleListAssays: a small reference class hierarchy (not exported from the **GenomicRanges** name space) defines a reference class ShallowData with a single field data of type ANY, and a derived class ShallowSimpleListAssays that specializes the type of data as [SimpleList](#), and contains=c("ShallowData", "Assays"). The assays slot of a [SummarizedExperiment](#) object contains an instance of ShallowSimpleListAssays.

Author(s)

Martin Morgan, mtmorgan@fhcrc.org

See Also

- [SummarizedExperiment](#) objects.
- [SimpleList](#) objects in the [S4Vectors](#) package.

Examples

```
## -----
## DIRECT MANIPULATION OF Assays OBJECTS
## -----
m1 <- matrix(runif(24), ncol=3)
m2 <- matrix(runif(24), ncol=3)
a <- Assays(SimpleList(m1, m2))
a

as(a, "SimpleList")

length(a)
a[[2]]
dim(a)

b <- a[-4, 2]
b
length(b)
b[[2]]
dim(b)

names(a)
names(a) <- c("a1", "a2")
names(a)
a[["a2"]]

rbind(a, a)
cbind(a, a)

## -----
## COPY-ON-CHANGE CONTRACT
## -----

## ShallowSimpleListAssays objects have copy-on-change semantics but not
## AssaysInEnv objects. For example:
ssla <- as(SimpleList(m1, m2), "ShallowSimpleListAssays")
aie <- as(SimpleList(m1, m2), "AssaysInEnv")

## No names on 'ssla' and 'aie':
names(ssla)
```

```

names(aie)

ssla2 <- ssla
aie2 <- aie
names(ssla2) <- names(aie2) <- c("A1", "A2")

names(ssla) # still NULL (as expected)

names(aie) # changed! (because the names<-,AssaysInEnv method is not
            # implemented in a way that respects the copy-on-change
            # contract)

```

coverage-methods

Coverage of a RangedSummarizedExperiment object

Description

This man page documents the coverage method for [RangedSummarizedExperiment](#) objects.

Usage

```

## S4 method for signature 'RangedSummarizedExperiment'
coverage(x, shift=0L, width=NULL, weight=1L,
         method=c("auto", "sort", "hash"))

```

Arguments

`x` A [RangedSummarizedExperiment](#) object.

`shift`, `width`, `weight`, `method`
 See [?coverage](#) in the **GenomicRanges** package.

Details

This method operates on the `rowRanges` component of the [RangedSummarizedExperiment](#) object, which can be a [GenomicRanges](#) or [GRangesList](#) object.

More precisely, on [RangedSummarizedExperiment](#) object `x`, `coverage(x, ...)` is equivalent to `coverage(rowRanges(x), ...)`.

See [?coverage](#) in the **GenomicRanges** package for the details of how coverage operates on a [GenomicRanges](#) or [GRangesList](#) object.

Value

See [?coverage](#) in the **GenomicRanges** package.

See Also

- [RangedSummarizedExperiment](#) objects.
- The [coverage](#) man page in the **GenomicRanges** package where the coverage methods for [GenomicRanges](#) and [GRangesList](#) objects are documented.

Examples

```
nrows <- 20; ncols <- 6
counts <- matrix(runif(nrows * ncols, 1, 1e4), nrows)
rowRanges <- GRanges(rep(c("chr1", "chr2"), c(5, 15)),
                    IRanges(sample(1000L, 20), width=100),
                    strand=Rle(c("+", "-"), c(12, 8)),
                    seqlengths=c(chr1=1800, chr2=1300))
colData <- DataFrame(Treatment=rep(c("ChIP", "Input"), 3),
                    row.names=LETTERS[1:6])
rse <- SummarizedExperiment(assays=SimpleList(counts=counts),
                          rowRanges=rowRanges, colData=colData)

cvg <- coverage(rse)
cvg
stopifnot(identical(cvg, coverage(rowRanges(rse))))
```

findOverlaps-methods *Finding overlapping ranges in RangedSummarizedExperiment objects*

Description

This man page documents the `findOverlaps` methods for [RangedSummarizedExperiment](#) objects.

[RangedSummarizedExperiment](#) objects also support `countOverlaps`, `overlapsAny`, and `subsetByOverlaps` thanks to the default methods defined in the **IRanges** package and to the `findOverlaps` methods defined in this package and documented below.

Usage

```
## S4 method for signature 'RangedSummarizedExperiment,Vector'
findOverlaps(query, subject,
             maxgap=0L, minoverlap=1L,
             type=c("any", "start", "end", "within", "equal"),
             select=c("all", "first", "last", "arbitrary"),
             ignore.strand=FALSE)
## S4 method for signature 'Vector,RangedSummarizedExperiment'
findOverlaps(query, subject,
             maxgap=0L, minoverlap=1L,
             type=c("any", "start", "end", "within", "equal"),
             select=c("all", "first", "last", "arbitrary"),
             ignore.strand=FALSE)
```

Arguments

`query`, `subject` One of these two arguments must be a [RangedSummarizedExperiment](#) object.

`maxgap`, `minoverlap`, `type`

See [?findOverlaps](#) in the **GenomicRanges** package.

`select`, `ignore.strand`

See [?findOverlaps](#) in the **GenomicRanges** package.

Details

These methods operate on the `rowRanges` component of the `RangedSummarizedExperiment` object, which can be a `GenomicRanges` or `GRangesList` object.

More precisely, if any of the above functions is passed a `RangedSummarizedExperiment` object through the `query` and/or `subject` argument, then it behaves as if `rowRanges(query)` and/or `rowRanges(subject)` had been passed instead.

See `?findOverlaps` in the `GenomicRanges` package for the details of how `findOverlaps` and family operate on `GenomicRanges` and `GRangesList` objects.

Value

See `?findOverlaps` in the `GenomicRanges` package.

See Also

- `RangedSummarizedExperiment` objects.
- The `findOverlaps` man page in the `GenomicRanges` package where the `findOverlaps` family of methods for `GenomicRanges` and `GRangesList` objects is documented.

Examples

```
nrows <- 20; ncols <- 6
counts <- matrix(runif(nrows * ncols, 1, 1e4), nrows)
rowRanges <- GRanges(rep(c("chr1", "chr2"), c(5, 15)),
                    IRanges(sample(1000L, 20), width=100),
                    strand=Rle(c("+", "-"), c(12, 8)))
colData <- DataFrame(Treatment=rep(c("ChIP", "Input"), 3),
                    row.names=LETTERS[1:6])
rse0 <- SummarizedExperiment(assays=SimpleList(counts=counts),
                             rowRanges=rowRanges, colData=colData)
rse1 <- shift(rse0, 100)

hits <- findOverlaps(rse0, rse1)
hits
stopifnot(identical(hits, findOverlaps(rowRanges(rse0), rowRanges(rse1))))
stopifnot(identical(hits, findOverlaps(rse0, rowRanges(rse1))))
stopifnot(identical(hits, findOverlaps(rowRanges(rse0), rse1)))
```

inter-range-methods *Inter range transformations of a RangedSummarizedExperiment object*

Description

This man page documents the *inter range transformations* that are supported on `RangedSummarizedExperiment` objects.

Usage

```
## S4 method for signature 'RangedSummarizedExperiment'
isDisjoint(x, ignore.strand=FALSE)

## S4 method for signature 'RangedSummarizedExperiment'
disjointBins(x, ignore.strand=FALSE)
```

Arguments

`x` A [RangedSummarizedExperiment](#) object.
`ignore.strand` See [?isDisjoint](#) in the **GenomicRanges** package.

Details

These transformations operate on the `rowRanges` component of the [RangedSummarizedExperiment](#) object, which can be a [GenomicRanges](#) or [GRangesList](#) object.

More precisely, any of the above functions performs the following transformation on [RangedSummarizedExperiment](#) object `x`:

```
f(rowRanges(x), ...)
```

where `f` is the name of the function and `...` any additional arguments passed to it.

See [?isDisjoint](#) in the **GenomicRanges** package for the details of how these transformations operate on a [GenomicRanges](#) or [GRangesList](#) object.

Value

See [?isDisjoint](#) in the **GenomicRanges** package.

See Also

- [RangedSummarizedExperiment](#) objects.
- The [isDisjoint](#) man page in the **GenomicRanges** package where *inter range transformations* of a [GenomicRanges](#) or [GRangesList](#) object are documented.

Examples

```
nrows <- 20; ncols <- 6
counts <- matrix(runif(nrows * ncols, 1, 1e4), nrows)
rowRanges <- GRanges(rep(c("chr1", "chr2"), c(5, 15)),
                    IRanges(sample(1000L, 20), width=100),
                    strand=Rle(c("+", "-"), c(12, 8)))
colData <- DataFrame(Treatment=rep(c("ChIP", "Input"), 3),
                    row.names=LETTERS[1:6])
rse0 <- SummarizedExperiment(assays=SimpleList(counts=counts),
                             rowRanges=rowRanges, colData=colData)
rse1 <- shift(rse0, 99*start(rse0))

isDisjoint(rse0) # FALSE
isDisjoint(rse1) # TRUE

bins0 <- disjointBins(rse0)
bins0
```

```
stopifnot(identical(bins0, disjointBins(rowRanges(rse0))))

bins1 <- disjointBins(rse1)
bins1
stopifnot(all(bins1 == bins1[[1]]))
```

intra-range-methods *Intra range transformations of a RangedSummarizedExperiment object*

Description

This man page documents the *intra range transformations* that are supported on [RangedSummarizedExperiment](#) objects.

Usage

```
## S4 method for signature 'RangedSummarizedExperiment'
shift(x, shift=0L, use.names=TRUE)

## S4 method for signature 'RangedSummarizedExperiment'
narrow(x, start=NA, end=NA, width=NA, use.names=TRUE)

## S4 method for signature 'RangedSummarizedExperiment'
resize(x, width, fix="start", use.names=TRUE,
        ignore.strand=FALSE)

## S4 method for signature 'RangedSummarizedExperiment'
flank(x, width, start=TRUE, both=FALSE,
       use.names=TRUE, ignore.strand=FALSE)

## S4 method for signature 'RangedSummarizedExperiment'
promoters(x, upstream=2000, downstream=200)

## S4 method for signature 'RangedSummarizedExperiment'
restrict(x, start=NA, end=NA, keep.all.ranges=FALSE,
         use.names=TRUE)

## S4 method for signature 'RangedSummarizedExperiment'
trim(x, use.names=TRUE)
```

Arguments

`x` A [RangedSummarizedExperiment](#) object.

`shift`, `use.names` See [?shift](#) in the **GenomicRanges** package.

`start`, `end`, `width`, `fix` See [?shift](#) in the **GenomicRanges** package.

`ignore.strand`, `both` See [?shift](#) in the **GenomicRanges** package.

upstream, downstream

See [?shift](#) in the **GenomicRanges** package.

keep.all.ranges

See [?shift](#) in the **GenomicRanges** package.

Details

These transformations operate on the `rowRanges` component of the [RangedSummarizedExperiment](#) object, which can be a [GenomicRanges](#) or [GRangesList](#) object.

More precisely, any of the above functions performs the following transformation on [RangedSummarizedExperiment](#) object `x`:

```
rowRanges(x) <- f(rowRanges(x), ...)
```

where `f` is the name of the function and `...` any additional arguments passed to it.

See [?shift](#) in the **GenomicRanges** package for the details of how these transformations operate on a [GenomicRanges](#) or [GRangesList](#) object.

See Also

- [RangedSummarizedExperiment](#) objects.
- The [shift](#) man page in the **GenomicRanges** package where *intra range transformations* of a [GenomicRanges](#) or [GRangesList](#) object are documented.

Examples

```
nrows <- 20; ncols <- 6
counts <- matrix(runif(nrows * ncols, 1, 1e4), nrows)
rowRanges <- GRanges(rep(c("chr1", "chr2"), c(5, 15)),
                     IRanges(sample(1000L, 20), width=100),
                     strand=Rle(c("+", "-"), c(12, 8)))
colData <- DataFrame(Treatment=rep(c("ChIP", "Input"), 3),
                    row.names=LETTERS[1:6])
rse0 <- SummarizedExperiment(assays=SimpleList(counts=counts),
                             rowRanges=rowRanges, colData=colData)
```

```
rse1 <- shift(rse0, 1)
stopifnot(identical(
  rowRanges(rse1),
  shift(rowRanges(rse0), 1)
))
```

```
se2 <- narrow(rse0, start=10, end=-15)
stopifnot(identical(
  rowRanges(se2),
  narrow(rowRanges(rse0), start=10, end=-15)
))
```

```
se3 <- resize(rse0, width=75)
stopifnot(identical(
  rowRanges(se3),
  resize(rowRanges(rse0), width=75)
))
```

```
se4 <- flank(rse0, width=20)
```

```

stopifnot(identical(
  rowRanges(se4),
  flank(rowRanges(rse0), width=20)
))

se5 <- restrict(rse0, start=200, end=700, keep.all.ranges=TRUE)
stopifnot(identical(
  rowRanges(se5),
  restrict(rowRanges(rse0), start=200, end=700, keep.all.ranges=TRUE)
))

```

```
makeSummarizedExperimentFromDataFrame
```

Make a RangedSummarizedExperiment from a data.frame or DataFrame

Description

`makeSummarizedExperimentFromDataFrame` uses `data.frame` or `DataFrame` column names to create a `GRanges` object for the `rowRanges` of the resulting `SummarizedExperiment` object. It requires that non-range data columns be coercible into a numeric matrix for the `SummarizedExperiment` constructor. All columns that are not part of the row ranges attribute are assumed to be experiment data; thus, keeping metadata columns will not be supported. Note that this function only returns `SummarizedExperiment` objects with a single assay.

If metadata columns are to be kept, one can first construct the row ranges attribute by using the `makeGRangesFromDataFrame` function and subsequently creating the `SummarizedExperiment`.

Usage

```

makeSummarizedExperimentFromDataFrame(df,
                                     ...,
                                     seqinfo = NULL,
                                     starts.in.df.are.0based = FALSE)

```

Arguments

<code>df</code>	A <code>data.frame</code> or <code>DataFrame</code> object. If not, then the function first tries to turn <code>df</code> into a data frame with <code>as.data.frame(df)</code> .
<code>...</code>	Additional arguments passed on to <code>makeGRangesFromDataFrame</code>
<code>seqinfo</code>	Either <code>NULL</code> , or a <code>Seqinfo</code> object, or a character vector of seqlevels, or a named numeric vector of sequence lengths. When not <code>NULL</code> , it must be compatible with the genomic ranges in <code>df</code> i.e. it must include at least the sequence levels represented in <code>df</code> .
<code>starts.in.df.are.0based</code>	<code>TRUE</code> or <code>FALSE</code> (the default). If <code>TRUE</code> , then the start positions of the genomic ranges in <code>df</code> are considered to be <i>0-based</i> and are converted to <i>1-based</i> in the returned <code>GRanges</code> object. This feature is intended to make it more convenient to handle input that contains data obtained from resources using the "0-based start" convention. A notorious example of such resource is the UCSC Table Browser (http://genome.ucsc.edu/cgi-bin/hgTables).

Value

A [RangedSummarizedExperiment](#) object with rowRanges and a single assay

Author(s)

M. Ramos

See Also

- [makeGRangesFromDataFrame](#)

Examples

```
## -----
## BASIC EXAMPLES
## -----

# Note that rownames of the data.frame are also rownames of the result
df <- data.frame(chr="chr2", start = 11:15, end = 12:16,
                 strand = c("+", "-", "+", "*", "."), expr0 = 3:7,
                 expr1 = 8:12, expr2 = 12:16,
                 row.names = paste0("GENE", letters[5:1]))

df

exRSE <- makeSummarizedExperimentFromDataFrame(df)

exRSE

assay(exRSE)

rowRanges(exRSE)
```

```
makeSummarizedExperimentFromExpressionSet
```

Make a RangedSummarizedExperiment object from an ExpressionSet and vice-versa

Description

Coercion between [RangedSummarizedExperiment](#) and [ExpressionSet](#) is supported in both directions.

For going from [ExpressionSet](#) to [RangedSummarizedExperiment](#), the `makeSummarizedExperimentFromExpressionSet` function is also provided to let the user control how to map features to ranges.

Usage

```
makeSummarizedExperimentFromExpressionSet(from,
                                          mapFun=naiveRangeMapper,
                                          ...)

## range mapping functions
naiveRangeMapper(from)
probeRangeMapper(from)
geneRangeMapper(txDbPackage, key = "ENTREZID")
```

Arguments

from	An ExpressionSet object.
mapFun	A function which takes an ExpressionSet object and returns a GRanges , or GRangesList object which corresponds to the genomic ranges used in the ExpressionSet . The rownames of the returned GRanges are used to match the featureNames of the ExpressionSet . The naiveRangeMapper function is used by default.
...	Additional arguments passed to mapFun .
txDbPackage	A character string with the Transcript Database to use for the mapping.
key	A character string with the Gene key to use for the mapping.

Value

`makeSummarizedExperimentFromExpressionSet` takes an [ExpressionSet](#) object as input and a *range mapping function* that maps the features to ranges. It then returns a [RangedSummarizedExperiment](#) object that corresponds to the input.

The range mapping functions return a [GRanges](#) object, with the [rownames](#) corresponding to the [featureNames](#) of the [ExpressionSet](#) object.

Author(s)

Jim Hester, james.f.hester@gmail.com

See Also

- [RangedSummarizedExperiment](#) objects.
- [ExpressionSet](#) objects in the **Biobase** package.
- [TxDb](#) objects in the **GenomicFeatures** package.

Examples

```
## -----
## GOING FROM ExpressionSet TO RangedSummarizedExperiment
## -----

data(sample.ExpressionSet, package="Biobase")

# 2 equivalent ways of doing the naive coercion
makeSummarizedExperimentFromExpressionSet(sample.ExpressionSet)
as(sample.ExpressionSet, "RangedSummarizedExperiment")

# using probe range mapper
makeSummarizedExperimentFromExpressionSet(sample.ExpressionSet, probeRangeMapper)

# using the gene range mapper
makeSummarizedExperimentFromExpressionSet(sample.ExpressionSet,
                                           geneRangeMapper("TxDb.Hsapiens.UCSC.hg19.knownGene"))

## -----
## GOING FROM RangedSummarizedExperiment TO ExpressionSet
## -----
```

```
example(RangedSummarizedExperiment) # to create 'rse'
rse
as(rse, "ExpressionSet")
```

nearest-methods	<i>Finding the nearest range neighbor in RangedSummarizedExperiment objects</i>
-----------------	---

Description

This man page documents the nearest methods and family (i.e. precede, follow, distance, and distanceToNearest methods) for [RangedSummarizedExperiment](#) objects.

Usage

```
## S4 method for signature 'RangedSummarizedExperiment,ANY'
precede(x, subject, select=c("arbitrary", "all"),
        ignore.strand=FALSE)
## S4 method for signature 'ANY,RangedSummarizedExperiment'
precede(x, subject, select=c("arbitrary", "all"),
        ignore.strand=FALSE)

## S4 method for signature 'RangedSummarizedExperiment,ANY'
follow(x, subject, select=c("arbitrary", "all"),
        ignore.strand=FALSE)
## S4 method for signature 'ANY,RangedSummarizedExperiment'
follow(x, subject, select=c("arbitrary", "all"),
        ignore.strand=FALSE)

## S4 method for signature 'RangedSummarizedExperiment,ANY'
nearest(x, subject, select=c("arbitrary", "all"), ignore.strand=FALSE)
## S4 method for signature 'ANY,RangedSummarizedExperiment'
nearest(x, subject, select=c("arbitrary", "all"), ignore.strand=FALSE)

## S4 method for signature 'RangedSummarizedExperiment,ANY'
distance(x, y, ignore.strand=FALSE, ...)
## S4 method for signature 'ANY,RangedSummarizedExperiment'
distance(x, y, ignore.strand=FALSE, ...)

## S4 method for signature 'RangedSummarizedExperiment,ANY'
distanceToNearest(x, subject, ignore.strand=FALSE, ...)
## S4 method for signature 'ANY,RangedSummarizedExperiment'
distanceToNearest(x, subject, ignore.strand=FALSE, ...)
```

Arguments

x, subject	One of these two arguments must be a RangedSummarizedExperiment object.
select, ignore.strand	See ?nearest in the GenomicRanges package.
y	For the distance methods, one of x or y must be a RangedSummarizedExperiment object.
...	Additional arguments for methods.

Details

These methods operate on the `rowRanges` component of the `RangedSummarizedExperiment` object, which can be a `GenomicRanges` or `GRangesList` object.

More precisely, if any of the above functions is passed a `RangedSummarizedExperiment` object through the `x`, `subject`, and/or `y` argument, then it behaves as if `rowRanges(x)`, `rowRanges(subject)`, and/or `rowRanges(y)` had been passed instead.

See `?nearest` in the `GenomicRanges` package for the details of how `nearest` and family operate on `GenomicRanges` and `GRangesList` objects.

Value

See `?nearest` in the `GenomicRanges` package.

See Also

- `RangedSummarizedExperiment` objects.
- The `nearest` man page in the `GenomicRanges` package where the nearest family of methods for `GenomicRanges` and `GRangesList` objects is documented.

Examples

```
nrows <- 20; ncols <- 6
counts <- matrix(runif(nrows * ncols, 1, 1e4), nrows)
rowRanges <- GRanges(rep(c("chr1", "chr2"), c(5, 15)),
                    IRanges(sample(1000L, 20), width=100),
                    strand=Rle(c("+", "-"), c(12, 8)))
colData <- DataFrame(Treatment=rep(c("ChIP", "Input"), 3),
                    row.names=LETTERS[1:6])
rse0 <- SummarizedExperiment(assays=SimpleList(counts=counts),
                             rowRanges=rowRanges, colData=colData)
rse1 <- shift(rse0, 100)

res <- nearest(rse0, rse1)
res
stopifnot(identical(res, nearest(rowRanges(rse0), rowRanges(rse1))))
stopifnot(identical(res, nearest(rse0, rowRanges(rse1))))
stopifnot(identical(res, nearest(rowRanges(rse0), rse1)))

res <- nearest(rse0) # missing subject
res
stopifnot(identical(res, nearest(rowRanges(rse0))))

hits <- nearest(rse0, rse1, select="all")
hits
stopifnot(identical(
  hits,
  nearest(rowRanges(rse0), rowRanges(rse1), select="all")
))
stopifnot(identical(
  hits,
  nearest(rse0, rowRanges(rse1), select="all")
))
stopifnot(identical(
  hits,
```

```
nearest(rowRanges(rse0), rse1, select="all")
))
```

RangedSummarizedExperiment-class

RangedSummarizedExperiment objects

Description

The RangedSummarizedExperiment class is a matrix-like container where rows represent ranges of interest (as a [GRanges](#) or [GRangesList](#) object) and columns represent samples (with sample data summarized as a [DataFrame](#)). A RangedSummarizedExperiment contains one or more assays, each represented by a matrix-like object of numeric or other mode.

RangedSummarizedExperiment is a subclass of [SummarizedExperiment](#) and, as such, all the methods documented in [?SummarizedExperiment](#) also work on a RangedSummarizedExperiment object. The methods documented below are additional methods that are specific to RangedSummarizedExperiment objects.

Usage

```
## Constructor

SummarizedExperiment(assays, ...)
## S4 method for signature 'SimpleList'
SummarizedExperiment(assays, rowData=NULL, rowRanges=GRangesList(),
  colData=DataFrame(), metadata=list())
## S4 method for signature 'ANY'
SummarizedExperiment(assays, ...)
## S4 method for signature 'list'
SummarizedExperiment(assays, ...)
## S4 method for signature 'missing'
SummarizedExperiment(assays, ...)

## Accessors

rowRanges(x, ...)
rowRanges(x, ...) <- value

## Subsetting

## S4 method for signature 'RangedSummarizedExperiment'
subset(x, subset, select, ...)

## rowRanges access
## see 'GRanges compatibility', below
```

Arguments

`assays` A list or `SimpleList` of matrix-like elements, or a matrix-like object. All elements of the list must have the same dimensions, and dimension names (if

	present) must be consistent across elements and with the row names of <code>rowRanges</code> and <code>colData</code> .
<code>rowData</code>	A DataFrame object describing the rows. Row names, if present, become the row names of the <code>SummarizedExperiment</code> object. The number of rows of the DataFrame must equal the number of rows of the matrices in assays.
<code>rowRanges</code>	A GRanges or GRangesList object describing the ranges of interest. Names, if present, become the row names of the <code>SummarizedExperiment</code> object. The length of the GRanges or GRangesList must equal the number of rows of the matrices in assays. If <code>rowRanges</code> is missing, a SummarizedExperiment instance is returned.
<code>colData</code>	An optional DataFrame describing the samples. Row names, if present, become the column names of the <code>RangedSummarizedExperiment</code> .
<code>metadata</code>	An optional list of arbitrary content describing the overall experiment.
<code>...</code>	For <code>SummarizedExperiment</code> , S4 methods <code>list</code> and <code>matrix</code> , arguments identical to those of the <code>SimpleList</code> method. For <code>rowRanges</code> , ignored.
<code>x</code>	A <code>RangedSummarizedExperiment</code> object. The <code>rowRanges</code> setter will also accept a SummarizedExperiment object and will first coerce it to <code>RangedSummarizedExperiment</code> before it sets value on it.
<code>value</code>	A GRanges or GRangesList object.
<code>subset</code>	An expression which, when evaluated in the context of <code>rowRanges(x)</code> , is a logical vector indicating elements or rows to keep: missing values are taken as false.
<code>select</code>	An expression which, when evaluated in the context of <code>colData(x)</code> , is a logical vector indicating elements or rows to keep: missing values are taken as false.

Details

The rows of a `RangedSummarizedExperiment` object represent ranges (in genomic coordinates) of interest. The ranges of interest are described by a [GRanges](#) or a [GRangesList](#) object, accessible using the `rowRanges` function, described below. The [GRanges](#) and [GRangesList](#) classes contains sequence (e.g., chromosome) name, genomic coordinates, and strand information. Each range can be annotated with additional data; this data might be used to describe the range or to summarize results (e.g., statistics of differential abundance) relevant to the range. Rows may or may not have row names; they often will not.

Constructor

`RangedSummarizedExperiment` instances are constructed using the `SummarizedExperiment` function with arguments outlined above.

Accessors

In the following code snippets, `x` is a `RangedSummarizedExperiment` object.

```
rowRanges(x), rowRanges(x) <- value: Get or set the row data. value is a GenomicRanges object. Row names of value must be NULL or consistent with the existing row names of x.
```


GRanges compatibility (rowRanges access)

Many [GRanges](#) and [GRangesList](#) operations are supported on `RangedSummarizedExperiment` objects, using `rowRanges`.

Supported operations include: `pcompare`, `duplicated`, `end`, `end<-`, `granges`, `is.unsorted`, `match`, `mcols`, `mcols<-`, `order`, `ranges`, `ranges<-`, `rank`, `seqinfo`, `seqinfo<-`, `seqnames`, `sort`, `start`, `start<-`, `strand`, `strand<-`, `width`, `width<-`.

See also `?shift`, `?isDisjoint`, `?coverage`, `?findOverlaps`, and `?nearest` for more *GRanges compatibility methods*.

Not all [GRanges](#) operations are supported, because they do not make sense for `RangedSummarizedExperiment` objects (e.g., `length`, `name`, `as.data.frame`, `c`, `splitAsList`), involve non-trivial combination or splitting of rows (e.g., `disjoin`, `gaps`, `reduce`, `unique`), or have not yet been implemented (`Ops`, `map`, `window`, `window<-`).

Subsetting

In the code snippets below, `x` is a `RangedSummarizedExperiment` object.

`subset(x, subset, select)`: Create a subset of `x` using an expression `subset` referring to columns of `rowRanges(x)` (including `'seqnames'`, `'start'`, `'end'`, `'width'`, `'strand'`, and `names(rowData(x))`) and `/` or `select` referring to column names of `colData(x)`.

Extension

`RangedSummarizedExperiment` is implemented as an S4 class, and can be extended in the usual way, using `contains="RangedSummarizedExperiment"` in the new class definition.

Author(s)

Martin Morgan, mtmorgan@fhcrc.org

See Also

- [SummarizedExperiment](#) objects.
- `shift`, `isDisjoint`, `coverage`, `findOverlaps`, and `nearest` for more *GRanges compatibility methods*.
- [GRanges](#) objects in the **GenomicRanges** package.

Examples

```
nrows <- 200; ncols <- 6
counts <- matrix(runif(nrows * ncols, 1, 1e4), nrows)
rowRanges <- GRanges(rep(c("chr1", "chr2"), c(50, 150)),
                     IRanges(floor(runif(200, 1e5, 1e6)), width=100),
                     strand=sample(c("+", "-"), 200, TRUE),
                     feature_id=sprintf("ID%03d", 1:200))
colData <- DataFrame(Treatment=rep(c("ChIP", "Input"), 3),
                    row.names=LETTERS[1:6])
rse <- SummarizedExperiment(assays=SimpleList(counts=counts),
                           rowRanges=rowRanges, colData=colData)
rse
dim(rse)
dimnames(rse)
assayNames(rse)
```

```

head(assay(rse))
assays(rse) <- endoapply(assays(rse), asinh)
head(assay(rse))

rowRanges(rse)
rowData(rse) # same as 'mcols(rowRanges(rse))'
colData(rse)

rse[, rse$Treatment == "ChIP"]

## cbind() combines objects with the same ranges but different samples:
rse1 <- rse
rse2 <- rse1[,1:3]
colnames(rse2) <- letters[1:ncol(rse2)]
cmb1 <- cbind(rse1, rse2)
dim(cmb1)
dimnames(cmb1)

## rbind() combines objects with the same samples but different ranges:
rse1 <- rse
rse2 <- rse1[1:50,]
rownames(rse2) <- letters[1:nrow(rse2)]
cmb2 <- rbind(rse1, rse2)
dim(cmb2)
dimnames(cmb2)

## Coercion to/from SummarizedExperiment:
se0 <- as(rse, "SummarizedExperiment")
se0

as(se0, "RangedSummarizedExperiment")

## Setting rowRanges on a SummarizedExperiment object turns it into a
## RangedSummarizedExperiment object:
se <- se0
rowRanges(se) <- rowRanges
se # RangedSummarizedExperiment

## Sanity checks:
stopifnot(identical(assays(se0), assays(rse)))
stopifnot(identical(dim(se0), dim(rse)))
stopifnot(identical(dimnames(se0), dimnames(rse)))
stopifnot(identical(rowData(se0), rowData(rse)))
stopifnot(identical(colData(se0), colData(rse)))

```

readKallisto

Input kallisto or kallisto bootstrap results.

Description

readKallisto inputs several kallisto output files into a single SummarizedExperiment instance, with rows corresponding to estimated transcript abundance and columns to samples. readKallistoBootstrap inputs kallisto bootstrap replicates of a single sample into a matrix of transcript x bootstrap abundance estimates.

Usage

```
readKallisto(files,
             json = file.path(dirname(files), "run_info.json"),
             h5 = any(grepl("\\.h5$", files)), what = KALLISTO_ASSAYS,
             as = c("SummarizedExperiment", "list", "matrix"))

readKallistoBootstrap(file, i, j)
```

Arguments

files	character() paths to kallisto ‘abundance.tsv’ output files. The assumption is that files are organized in the way implied by kallisto, with each sample in a distinct directory, and the directory containing files abundance.tsv, run_info.json, and perhaps abundance.h5.
json	character() vector of the same length as files specifying the location of JSON files produced by kallisto and containing information on the run. The default assumes that json files are in the same directory as the corresponding abundance file.
h5	character() vector of the same length as files specifying the location of HDF5 files produced by kallisto and containing bootstrap estimates. The default assumes that HDF5 files are in the same directory as the corresponding abundance file.
what	character() vector of kallisto per-sample outputs to be input. See KALLISTO_ASSAYS for available values.
as	character(1) specifying the output format. See Value for additional detail.
file	character(1) path to a single HDF5 output file.
i, j	integer() vector of row (i) and column (j) indexes to input.

Value

A SummarizedExperiment, list, or matrix, depending on the value of argument as; by default a SummarizedExperiment. The as="SummarizedExperiment" rowData(se) the length of each transcript; colData(se) includes summary information on each sample, including the number of targets and bootstraps, the kallisto and index version, the start time and operating system call used to create the file. assays() contains one or more transcript x sample matrices of parameters estimated by kallisto (see KALLISTO_ASSAYS).

as="list" return value contains information similar to SummarizedExperiment with row, column and assay data as elements of the list without coordination of row and column annotations into an integrated data container. as="matrix" returns the specified assay as a simple R matrix.

Author(s)

Martin Morgan martin.morgan@roswellpark.org

References

<http://pachterlab.github.io/kallisto> software for quantifying transcript abundance.

Examples

```

outputs <- system.file(package="SummarizedExperiment", "extdata",
  "kallisto")
files <- dir(outputs, pattern="abundance.tsv", full=TRUE, recursive=TRUE)
stopifnot(all(file.exists(files)))

## default: input 'est_counts'
(se <- readKallisto(files, as="SummarizedExperiment"))
str(readKallisto(files, as="list"))
str(readKallisto(files, as="matrix"))

## available assays
KALLISTO_ASSAYS
## one or more assay
readKallisto(files, what=c("tpm", "eff_length"))

## alternatively: read hdf5 files
files <- sub(".tsv", ".h5", files, fixed=TRUE)
readKallisto(files)

## input all bootstraps
xx <- readKallistoBootstrap(files[1])
ridx <- head(which(rowSums(xx) != 0), 3)
cidx <- c(1:5, 96:100)
xx[ridx, cidx]

## selective input of rows (transcripts) and/or bootstraps
readKallistoBootstrap(files[1], i=c(ridx, rev(ridx)), j=cidx)

```

SummarizedExperiment-class

SummarizedExperiment objects

Description

The SummarizedExperiment class is a matrix-like container where rows represent features of interest (e.g. genes, transcripts, exons, etc...) and columns represent samples (with sample data summarized as a [DataFrame](#)). A SummarizedExperiment object contains one or more assays, each represented by a matrix-like object of numeric or other mode.

Note that SummarizedExperiment is the parent of the [RangedSummarizedExperiment](#) class which means that all the methods documented below also work on a [RangedSummarizedExperiment](#) object.

Usage

```

## Constructor

# See ?RangedSummarizedExperiment for the constructor function.

## Accessors

```

```

assayNames(x, ...)
assayNames(x, ...) <- value
assays(x, ..., withDimnames=TRUE)
assays(x, ..., withDimnames=TRUE) <- value
assay(x, i, ...)
assay(x, i, ...) <- value
rowData(x, ...)
rowData(x, ...) <- value
colData(x, ...)
colData(x, ...) <- value
#dim(x)
#dimnames(x)
#dimnames(x) <- value

## Quick colData access

## S4 method for signature 'SummarizedExperiment'
x$name
## S4 replacement method for signature 'SummarizedExperiment'
x$name <- value
## S4 method for signature 'SummarizedExperiment,ANY,missing'
x[[i, j, ...]]
## S4 replacement method for signature 'SummarizedExperiment,ANY,missing'
x[[i, j, ...]] <- value

## Subsetting

## S4 method for signature 'SummarizedExperiment'
x[i, j, ..., drop=TRUE]
## S4 replacement method for signature 'SummarizedExperiment,ANY,ANY,SummarizedExperiment'
x[i, j] <- value

## Combining

## S4 method for signature 'SummarizedExperiment'
cbind(..., deparse.level=1)
## S4 method for signature 'SummarizedExperiment'
rbind(..., deparse.level=1)

```

Arguments

<code>x</code>	A SummarizedExperiment object.
<code>...</code>	For assay, ... may contain withDimnames, which is forwarded to assays. For rowData, arguments passed thru ... are forwarded to mcols. For cbind, rbind, ... contains SummarizedExperiment objects to be combined. For other accessors, ignored.
<code>i, j</code>	For assay, assay<-, i is an integer or numeric scalar; see 'Details' for additional constraints. For [, SummarizedExperiment, [, SummarizedExperiment<-, i, j are subscripts that can act to subset the rows and columns of x, that is the matrix

	elements of assays. For <code>[[, SummarizedExperiment</code> , <code>[[<-, SummarizedExperiment</code> , <code>i</code> is a scalar index (e.g., <code>character(1)</code> or <code>integer(1)</code>) into a column of <code>colData</code> .
<code>name</code>	A symbol representing the name of a column of <code>colData</code> .
<code>withDimnames</code>	A <code>logical(1)</code> , indicating whether <code>dimnames</code> should be applied to extracted assay elements. Setting <code>withDimnames=FALSE</code> increases the speed and memory efficiency with which assays are extracted. <code>withDimnames=TRUE</code> in the getter <code>assays<-</code> allows efficient complex assignments (e.g., updating names of assays, <code>names(assays(x, withDimnames=FALSE)) = ...</code> is more efficient than <code>names(assays(x)) = ...</code>); it does not influence actual assignment of <code>dimnames</code> to assays.
<code>drop</code>	A <code>logical(1)</code> , ignored by these methods.
<code>value</code>	An object of a class specified in the S4 method signature or as outlined in ‘Details’.
<code>deparse.level</code>	See <code>?base::cbind</code> for a description of this argument.

Details

The `SummarizedExperiment` class is meant for numeric and other data types derived from a sequencing experiment. The structure is rectangular like a `matrix`, but with additional annotations on the rows and columns, and with the possibility to manage several assays simultaneously.

The rows of a `SummarizedExperiment` object represent features of interest. Information about these features is stored in a `DataFrame` object, accessible using the function `rowData`. The `DataFrame` must have as many rows as there are rows in the `SummarizedExperiment` object, with each row of the `DataFrame` providing information on the feature in the corresponding row of the `SummarizedExperiment` object. Columns of the `DataFrame` represent different attributes of the features of interest, e.g., gene or transcript IDs, etc.

Each column of a `SummarizedExperiment` object represents a sample. Information about the samples are stored in a `DataFrame`, accessible using the function `colData`, described below. The `DataFrame` must have as many rows as there are columns in the `SummarizedExperiment` object, with each row of the `DataFrame` providing information on the sample in the corresponding column of the `SummarizedExperiment` object. Columns of the `DataFrame` represent different sample attributes, e.g., tissue of origin, etc. Columns of the `DataFrame` can themselves be annotated (via the `mcols` function). Column names typically provide a short identifier unique to each sample.

A `SummarizedExperiment` object can also contain information about the overall experiment, for instance the lab in which it was conducted, the publications with which it is associated, etc. This information is stored as a `list` object, accessible using the `metadata` function. The form of the data associated with the experiment is left to the discretion of the user.

The `SummarizedExperiment` container is appropriate for matrix-like data. The data are accessed using the `assays` function, described below. This returns a `SimpleList` object. Each element of the list must itself be a matrix (of any mode) and must have dimensions that are the same as the dimensions of the `SummarizedExperiment` in which they are stored. Row and column names of each matrix must either be `NULL` or match those of the `SummarizedExperiment` during construction. It is convenient for the elements of `SimpleList` of assays to be named.

Constructor

`SummarizedExperiment` instances are constructed using the `SummarizedExperiment` function documented in `?RangedSummarizedExperiment`.

Accessors

In the following code snippets, `x` is a `SummarizedExperiment` object.

`assays(x)`, `assays(x) <- value`: Get or set the assays. `value` is a list or `SimpleList`, each element of which is a matrix with the same dimensions as `x`.

`assay(x, i)`, `assay(x, i) <- value`: A convenient alternative (to `assays(x)[[i]]`, `assays(x)[[i]] <- value`) to get or set the `i`th (default first) assay element. `value` must be a matrix of the same dimension as `x`, and with dimension names `NULL` or consistent with those of `x`.

`assayNames(x)`, `assayNames(x) <- value`: Get or set the names of `assay()` elements.

`rowData(x)`, `rowData(x) <- value`: Get or set the row data. `value` is a `DataFrame` object. Row names of `value` must be `NULL` or consistent with the existing row names of `x`.

`colData(x)`, `colData(x) <- value`: Get or set the column data. `value` is a `DataFrame` object. Row names of `value` must be `NULL` or consistent with the existing column names of `x`.

`metadata(x)`, `metadata(x) <- value`: Get or set the experiment data. `value` is a list with arbitrary content.

`dim(x)`: Get the dimensions (features of interest \times samples) of the `SummarizedExperiment`.

`dimnames(x)`, `dimnames(x) <- value`: Get or set the dimension names. `value` is usually a list of length 2, containing elements that are either `NULL` or vectors of appropriate length for the corresponding dimension. `value` can be `NULL`, which removes dimension names. This method implies that `rownames`, `rownames<-`, `colnames`, and `colnames<-` are all available.

Subsetting

In the code snippets below, `x` is a `SummarizedExperiment` object.

`x[i, j]`, `x[i, j] <- value`: Create or replace a subset of `x`. `i`, `j` can be numeric, logical, character, or missing. `value` must be a `SummarizedExperiment` object with dimensions, dimension names, and assay elements consistent with the subset `x[i, j]` being replaced.

Additional subsetting accessors provide convenient access to `colData` columns

`x$name`, `x$name <- value` Access or replace column name in `x`.

`x[[i, ...]]`, `x[[i, ...]] <- value` Access or replace column `i` in `x`.

Combining

In the code snippets below, `...` are `SummarizedExperiment` objects to be combined.

`cbind(...)`: `cbind` combines objects with the same features of interest but different samples (columns in assays). The `colnames` in `colData(SummarizedExperiment)` must match or an error is thrown. Duplicate columns of `rowData(SummarizedExperiment)` must contain the same data.

Data in assays are combined by name matching; if all assay names are `NULL` matching is by position. A mixture of names and `NULL` throws an error.

`metadata` from all objects are combined into a list with no name checking.

`rbind(...)`: `rbind` combines objects with the same samples but different features of interest (rows in assays). The `colnames` in `rowData(SummarizedExperiment)` must match or an error is thrown. Duplicate columns of `colData(SummarizedExperiment)` must contain the same data.

Data in assays are combined by name matching; if all assay names are `NULL` matching is by position. A mixture of names and `NULL` throws an error.

`metadata` from all objects are combined into a list with no name checking.

Implementation and Extension

This section contains advanced material meant for package developers.

SummarizedExperiment is implemented as an S4 class, and can be extended in the usual way, using `contains="SummarizedExperiment"` in the new class definition.

In addition, the representation of the assays slot of SummarizedExperiment is as a virtual class Assays. This allows derived classes (`contains="Assays"`) to easily implement alternative requirements for the assays, e.g., backed by file-based storage like NetCDF or the `ff` package, while re-using the existing SummarizedExperiment class without modification. See [Assays](#) for more information.

The current assays slot is implemented as a reference class that has copy-on-change semantics. This means that modifying non-assay slots does not copy the (large) assay data, and at the same time the user is not surprised by reference-based semantics. Updates to non-assay slots are very fast; updating the assays slot itself can be 5x or more faster than with an S4 instance in the slot. One useful technique when working with assay or assays function is use of the `withDimnames=FALSE` argument, which benefits speed and memory use by not copying dimnames from the row- and colData elements to each assay.

Author(s)

Martin Morgan, mtmorgan@fhcrc.org

See Also

- [RangedSummarizedExperiment](#) objects.
- [DataFrame](#), [SimpleList](#), and [Annotated](#) objects in the **S4Vectors** package.
- The [metadata](#) and [mcols](#) accessors in the **S4Vectors** package.

Examples

```
nrows <- 200; ncols <- 6
counts <- matrix(runif(nrows * ncols, 1, 1e4), nrows)
colData <- DataFrame(Treatment=rep(c("ChIP", "Input"), 3),
                    row.names=LETTERS[1:6])
se0 <- SummarizedExperiment(assays=SimpleList(counts=counts),
                          colData=colData)

se0
dim(se0)
dimnames(se0)
assayNames(se0)
head(assay(se0))
assays(se0) <- endoapply(assays(se0), asinh)
head(assay(se0))

rowData(se0)
colData(se0)

se0[, se0$Treatment == "ChIP"]

## cbind() combines objects with the same features of interest
## but different samples:
se1 <- se0
se2 <- se1[,1:3]
colnames(se2) <- letters[1:ncol(se2)]
```



```
cmb1 <- cbind(se1, se2)
dim(cmb1)
dimnames(cmb1)

## rbind() combines objects with the same samples but different
## features of interest:
se1 <- se0
se2 <- se1[1:50,]
rownames(se2) <- letters[1:nrow(se2)]
cmb2 <- rbind(se1, se2)
dim(cmb2)
dimnames(cmb2)
```


- assayNames<- , SummarizedExperiment, character-method (SummarizedExperiment-class), 20
- Assays, 24
- Assays (Assays-class), 2
- assays (SummarizedExperiment-class), 20
- assays, SummarizedExperiment-method (SummarizedExperiment-class), 20
- Assays-class, 2
- assays<- (SummarizedExperiment-class), 20
- assays<- , SummarizedExperiment, list-method (SummarizedExperiment-class), 20
- assays<- , SummarizedExperiment, SimpleList-method (SummarizedExperiment-class), 20
- AssaysInEnv (Assays-class), 2
- AssaysInEnv-class (Assays-class), 2

- cbind, 22
- cbind, Assays-method (Assays-class), 2
- cbind, SummarizedExperiment-method (SummarizedExperiment-class), 20
- class: Assays (Assays-class), 2
- class: AssaysInEnv (Assays-class), 2
- class: RangedSummarizedExperiment (RangedSummarizedExperiment-class), 15
- class: ShallowData (Assays-class), 2
- class: ShallowSimpleListAssays (Assays-class), 2
- class: SimpleListAssays (Assays-class), 2
- class: SummarizedExperiment (SummarizedExperiment-class), 20
- class: SummarizedExperiment0 (SummarizedExperiment-class), 20
- coerce, AssaysInEnv, SimpleList-method (Assays-class), 2
- coerce, ExpressionSet, RangedSummarizedExperiment-method (makeSummarizedExperimentFromExpressionSet), 11
- coerce, RangedSummarizedExperiment, ExpressionSet-method (makeSummarizedExperimentFromExpressionSet), 11
- coerce, RangedSummarizedExperiment, SummarizedExperiment-method (RangedSummarizedExperiment-class), 15
- coerce, ShallowSimpleListAssays, SimpleList-method (Assays-class), 2
- coerce, SimpleList, AssaysInEnv-method (Assays-class), 2
- coerce, SimpleList, ShallowSimpleListAssays-method (Assays-class), 2
- coerce, SummarizedExperiment, ExpressionSet-method (makeSummarizedExperimentFromExpressionSet), 11
- coerce, SummarizedExperiment, RangedSummarizedExperiment-method (RangedSummarizedExperiment-class), 15
- colData (SummarizedExperiment-class), 20
- colData, SummarizedExperiment-method (SummarizedExperiment-class), 20
- colData<- (SummarizedExperiment-class), 20
- colData<- , SummarizedExperiment, DataFrame-method (SummarizedExperiment-class), 20
- Compare, ANY, RangedSummarizedExperiment-method (RangedSummarizedExperiment-class), 15
- Compare, RangedSummarizedExperiment, ANY-method (RangedSummarizedExperiment-class), 15
- Compare, RangedSummarizedExperiment, RangedSummarizedExperiment-method (RangedSummarizedExperiment-class), 15
- coverage, 4, 17
- coverage (coverage-methods), 4
- coverage, RangedSummarizedExperiment-method (coverage-methods), 4
- coverage-methods, 4
- DataFrame, 10, 15, 16, 20, 22–24
- dim, Assays-method (Assays-class), 2
- dim, SummarizedExperiment-method (SummarizedExperiment-class), 20
- dimnames, RangedSummarizedExperiment-method (RangedSummarizedExperiment-class), 15
- dimnames, SummarizedExperiment-method (SummarizedExperiment-class), 20
- dimnames<- , RangedSummarizedExperiment, list-method (RangedSummarizedExperiment-class), 15
- dimnames<- , SummarizedExperiment, list-method (SummarizedExperiment-class), 20

- mcols<-, RangedSummarizedExperiment-method
(RangedSummarizedExperiment-class),
15
- metadata, 24
- naiveRangeMapper
(makeSummarizedExperimentFromExpressionSet),
11
- names, Assays-method (Assays-class), 2
- names, AssaysInEnv-method
(Assays-class), 2
- names, RangedSummarizedExperiment-method
(RangedSummarizedExperiment-class),
15
- names, SummarizedExperiment-method
(SummarizedExperiment-class),
20
- names<-, Assays-method (Assays-class), 2
- names<-, AssaysInEnv-method
(Assays-class), 2
- names<-, RangedSummarizedExperiment-method
(RangedSummarizedExperiment-class),
15
- names<-, SummarizedExperiment-method
(SummarizedExperiment-class),
20
- narrow (intra-range-methods), 8
- narrow, RangedSummarizedExperiment-method
(intra-range-methods), 8
- nearest, 13, 14, 17
- nearest (nearest-methods), 13
- nearest, ANY, RangedSummarizedExperiment-method
(nearest-methods), 13
- nearest, RangedSummarizedExperiment, ANY-method
(nearest-methods), 13
- nearest, RangedSummarizedExperiment, RangedSummarizedExperiment-method
(nearest-methods), 13
- nearest-methods, 13
- order, 17
- order, RangedSummarizedExperiment-method
(RangedSummarizedExperiment-class),
15
- pcompare, 17
- pcompare, ANY, RangedSummarizedExperiment-method
(RangedSummarizedExperiment-class),
15
- pcompare, RangedSummarizedExperiment, ANY-method
(RangedSummarizedExperiment-class),
15
- pcompare, RangedSummarizedExperiment, RangedSummarizedExperiment-method
(RangedSummarizedExperiment-class),
15
- precede (nearest-methods), 13
- precede, ANY, RangedSummarizedExperiment-method
(nearest-methods), 13
- precede, RangedSummarizedExperiment, ANY-method
(nearest-methods), 13
- precede, RangedSummarizedExperiment, RangedSummarizedExperiment-method
(nearest-methods), 13
- probeRangeMapper
(makeSummarizedExperimentFromExpressionSet),
11
- promoters (intra-range-methods), 8
- promoters, RangedSummarizedExperiment-method
(intra-range-methods), 8
- RangedSummarizedExperiment, 4-9, 11-14,
20, 22, 24
- RangedSummarizedExperiment
(RangedSummarizedExperiment-class),
15
- RangedSummarizedExperiment-class, 15
- ranges, 17
- ranges, RangedSummarizedExperiment-method
(RangedSummarizedExperiment-class),
15
- ranges<-, RangedSummarizedExperiment-method
(RangedSummarizedExperiment-class),
15
- rank, 17
- rank, RangedSummarizedExperiment-method
(RangedSummarizedExperiment-class),
15
- rbind, Assays-method (Assays-class), 2
- rbind, SummarizedExperiment-method
(SummarizedExperiment-class),
20
- readKallisto, 18
- readKallistoBootstrap (readKallisto), 18
- replaceROWS, SummarizedExperiment-method
(SummarizedExperiment-class),
20
- resize (intra-range-methods), 8
- resize, RangedSummarizedExperiment-method
(intra-range-methods), 8
- restrict (intra-range-methods), 8
- restrict, RangedSummarizedExperiment-method
(intra-range-methods), 8
- rowData (SummarizedExperiment-class), 20
- rowData, SummarizedExperiment-method
(SummarizedExperiment-class),
20
- rowData<- (SummarizedExperiment-class),
20

- rowData<- , SummarizedExperiment-method
(SummarizedExperiment-class),
20
- rownames, 12
- rowRanges
(RangedSummarizedExperiment-class),
15
- rowRanges, RangedSummarizedExperiment-method
(RangedSummarizedExperiment-class),
15
- rowRanges<-
(RangedSummarizedExperiment-class),
15
- rowRanges<- , SummarizedExperiment, GenomicRanges-method
(RangedSummarizedExperiment-class),
15
- rowRanges<- , SummarizedExperiment, GRangesList-method
(RangedSummarizedExperiment-class),
15

- Seqinfo, 10
- seqinfo, 17
- seqinfo, RangedSummarizedExperiment-method
(RangedSummarizedExperiment-class),
15
- seqinfo<- , RangedSummarizedExperiment-method
(RangedSummarizedExperiment-class),
15
- seqnames, 17
- seqnames, RangedSummarizedExperiment-method
(RangedSummarizedExperiment-class),
15
- ShallowData (Assays-class), 2
- ShallowData-class (Assays-class), 2
- ShallowSimpleListAssays (Assays-class),
2
- ShallowSimpleListAssays-class
(Assays-class), 2
- shift, 8, 9, 17
- shift (intra-range-methods), 8
- shift, RangedSummarizedExperiment-method
(intra-range-methods), 8
- show, SummarizedExperiment-method
(SummarizedExperiment-class),
20
- SimpleList, 2, 3, 22, 24
- SimpleListAssays (Assays-class), 2
- SimpleListAssays-class (Assays-class), 2
- sort, 17
- sort, RangedSummarizedExperiment-method
(RangedSummarizedExperiment-class),
15

- split, RangedSummarizedExperiment, ANY-method
(RangedSummarizedExperiment-class),
15
- split, RangedSummarizedExperiment-method
(RangedSummarizedExperiment-class),
15
- start, 17
- start, RangedSummarizedExperiment-method
(RangedSummarizedExperiment-class),
15
- start<- , RangedSummarizedExperiment-method
(RangedSummarizedExperiment-class),
15
- strand, RangedSummarizedExperiment-method
(RangedSummarizedExperiment-class),
15
- strand<- , RangedSummarizedExperiment, ANY-method
(RangedSummarizedExperiment-class),
15
- subset, RangedSummarizedExperiment-method
(RangedSummarizedExperiment-class),
15
- SummarizedExperiment, 2, 3, 10, 15–17
- SummarizedExperiment
(RangedSummarizedExperiment-class),
15
- SummarizedExperiment, ANY-method
(RangedSummarizedExperiment-class),
15
- SummarizedExperiment, list-method
(RangedSummarizedExperiment-class),
15
- SummarizedExperiment, missing-method
(RangedSummarizedExperiment-class),
15
- SummarizedExperiment, SimpleList-method
(RangedSummarizedExperiment-class),
15
- SummarizedExperiment-class, 20
- SummarizedExperiment0
(SummarizedExperiment-class),
20
- SummarizedExperiment0-class
(SummarizedExperiment-class),
20

- trim, RangedSummarizedExperiment-method
(intra-range-methods), 8
- TxDb, 12
- updateObject, SummarizedExperiment-method
(RangedSummarizedExperiment-class),

15

width, 17

width,RangedSummarizedExperiment-method
(RangedSummarizedExperiment-class),
15

width<-,RangedSummarizedExperiment-method
(RangedSummarizedExperiment-class),
15