Package 'GenomicFiles'

April 10, 2015

Title Distributed computing by file or by range

Description This package provides infrastructure for parallel computations distributed 'by file' or 'by range'. User defined MAPPER and REDUCER functions provide added flexibility for data combination and manipulation.

Version 1.2.1

Author Valerie Obenchain, Michael Love, Martin Morgan

Maintainer Bioconductor Package Maintainer <`maintainer@bioconductor.org`>

biocViews Infrastructure, DataImport, Sequencing

Depends R (>= 3.1.0), methods, BiocGenerics (>= 0.11.2), Rsamtools (>= 1.17.29), GenomicRanges (>= 1.17.16), rtracklayer (>= 1.25.3), BiocParallel (>= 0.7.0)

Imports GenomicAlignments, IRanges,

Suggests BiocStyle, RUnit, genefilter, deepSNV, RNAseqData.HNRNPC.bam.chr14

License Artistic-2.0

Collate GenomicFiles-class.R GenomicFileViews-class.R BigWigFileViews-class.R BamFileViews-class.R FaFileViews-class.R utils.R reduceByFile-methods.R reduceByRange-methods.R reduceByYield.R reduceRanges.R reduceFiles.R pack-methods.R unpack-methods.R registry.R zzz.R

Video https://www.youtube.com/watch?v=3PK_jx44QTs

R topics documented:

`BamFileViews` *Views into a set of BAM files*

Description

Use `BamFileViews()` to reference a set of disk-based BAM files to be processed

Constructor

`BamFileViews(fileList,     fileSample=DataFrame(row.names=make.unique(basename(path(fileList)))),`
This constructor is a generic function with dispatch on argument `fileList`. Methods exist for BamFileList and character (vector of file names).

Accessors

All accessor-like methods defined for `GenomicFileViews` objects work on `BamFileViews` objects. See ?`GenomicFileViews` for details.

- fileList(x); fileList(x) <- value
- fileSample(x); fileSample(x) <- value
- fileRange(x); fileRange(x) <- value
- fileExperiment(x); fileExpermient(x) <- value
- yieldSize(x); yieldSize(x) <- value

Methods

`"["`: Subset the object by `fileRange` or `fileSample`.

reduceByFile Computations are distributed in parallel by files in `fileList` with the option to provide MAP and REDUCE functions across ranges and / or files.

reduceByRange Computations are distributed in parallel by ranges in `fileRange` with the option to provide MAP and REDUCE functions across ranges and / or files.

summarizeOverlaps Computations are distributed in parallel by files in `fileList`. Ranges in the `fileRange` slot take precedence over ranges in `param`. The return value is a `SummarizedExperiment` object.

countBam Computations are distributed in parallel by files in `fileList`. Ranges in the `fileRange` slot take precedence over ranges in `param`. The return value is a list of `data.frames`, one per file.

scanBam Computations are distributed in parallel by files in `fileList`. Ranges in the `fileRange` slot take precedence over ranges in `param`. The return value is a list of `lists`, one per file.

Arguments

`fileList`: A character() vector of BAM path names or a `BamFileList`.

`fileSample`: A DataFrame instance with as many rows as `length(fileList)`, containing sample information associated with each path.

`fileRange`: A `GRanges`, or missing instance with ranges defined on the spaces of the BAM files. Ranges are *not* validated against the BAM files.

`fileExperiment`: A list() containing additional information about the experiment.

`yieldSize`: An integer specifying number of records to process

`.views_on_file`: An enviornment; currently under development

`...`: Additional arguments.

`x`, `object`: An instance of `BamFileViews`.

`value`: An object of appropriate type to replace content.

`i`: During subsetting, a logical or numeric index into `fileRange`.

`j`: During subsetting, a logical or numeric index into `fileSample` and `fileList`.

`file`: An instance of `BamFileViews`.

`index`: Not used.

`param`: An optional `ScanBamParam` instance to further influence scanning or counting.

Slots

Inherited from `GenomicFileViews` class:

fileList
- fileSample
- fileRange
- fileExperiment
- yieldSize
- .views_on_file

Author(s)

Martin Morgan <mtmorgan@fhcrc.org.> and Valerie Obenchain <vobencha@fhcrc.org>

See Also

- GenomicFileViews-class class.
- reduceByFile and reduceByRange methods.

Examples

```
library(RNAseqData.HNRNPC.bam.chr14)
fls <- RNAseqData.HNRNPC.bam.chr14_BAMFILES
gr <- GRanges("chr14", IRanges(1:10*5, width = 2))
bfv <- BamFileViews(fls, fileRange = gr)
```

```
## Dimensions of the object are ranges by samples (files).
bfv
bfv[1:5,]
fileList(bfv)
fileRange(bfv)
```

`BigWigFileViews` *Views into a set of BigWig files*

Description

Use `BigWigFileViews()` to reference a set of disk-based BigWig files to be processed (e.g., queried using `coverage`) as a single 'experiment'.

Constructor

`BigWigFileViews(fileList, fileSample=DataFrame(row.names=make.unique(basename(path(fileList)))`
: This constructor is a generic function with dispatch on argument `fileList`. Methods exist for `BigWigFileList` and `character()` (vector of file names).

Accessors

All accessor-like methods defined for `GenomicFileViews` objects work on `BigWigFileViews` objects. See ?`GenomicFileViews` for details.

- fileList(x); fileList(x) <- value
- fileSample(x); fileSample(x) <- value
- fileRange(x); fileRange(x) <- value
- fileExperiment(x); fileExpermient(x) <- value
- yieldSize(x); yieldSize(x) <- value

Subsetting

`"["`: Subset the object by `fileRange` or `fileSample`.

Other methods

In the code snippets below, `x` and `object` are BigWigFileViews objects.

`coverage(x, ..., by = "file", summarize = TRUE, as = "RleList")`:
: Computes coverage with the `import` function from `rtracklayer` for each file in `fileList(x)` and each range in `fileRange(x)`. Work is divided in parallel as specified in the by argument. Results are returned as a list unless `summarize=TRUE` in which case the data are in the `assays` slot of a SummarizedExperiment object. Data type are controlled with the `as` argument. See ?`import,BigWigFile-method` for details.

`summary(object, ..., by = "file", summarize = TRUE)`: Computes summary statistics with the `summary` function from `rtracklayer` for each file in `fileList(object)` and each range in `fileRange(object)`. Work is divided in parallel as specified in the `by` argument. Results are returned as a list unless `summarize=TRUE` in which case the data are in the `assays` slot of a SummarizedExperiment object. Summary statistics are controlled with the `type` argument passed to `summary`. See ?`summary,BigWigFile-method` for details.

Arguments

`fileList`: A character() vector of BigWig path names or a `BigWigFileList`.

`fileSample`: A DataFrame instance with as many rows as `length(fileList)`, containing sample information associated with each path.

`fileRange`: A `GRanges`, or missing instance with ranges defined on the spaces of the BigWig files. Ranges are *not* validated against the BigWig files.

`fileExperiment`: A list() containing additional information about the experiment.

`yieldSize`: An integer specifying number of records to process

`.views_on_file`: An enviornment; currently under development

`...`: Additional arguments.

`x`, `object`: An instance of `BigWigFileViews`.

`value`: An object of appropriate type to replace content.

`i`: During subsetting, a logical or numeric index into `fileRange`.

`j`: During subsetting, a logical or numeric index into `fileSample` and `fileList`.

`file`: An instance of `BigWigFileViews`.

`index`: Not used.

Slots

Inherited from `GenomicFileViews` class:

fileList

- fileSample
- fileRange
- fileExperiment
- yieldSize
- .views_on_file

Author(s)

Michael Love <michaelisaiahlove@gmail.com>, Valerie Obenchain <vobencha@fhcrc.org>, Martin Morgan <mtmorgan@fhcrc.org.>

See Also

- GenomicFileViews-class.

Examples

```
if (.Platform$OS.type != "windows") {

  register(SerialParam())

  ## ---------------------------------------------------------------------
  ## BigWigFileView Objects
  ## ---------------------------------------------------------------------
  fl <- system.file("tests", "test.bw", package = "rtracklayer")
  gr <- GRanges(Rle(c("chr2", "chr19"), c(4, 2)),
                IRanges(1 + c(200, 250, 500, 550, 1450, 1750), width=100))
  bwfv <- BigWigFileViews(c(fl, fl), fileRange=gr)

  ## Object dimensions are range by sample (files).
  bwfv
  fileList(bwfv)
  fileRange(bwfv)

  ## Subset by range(s) or sample(s).
  bwfv[1:3,]
  bwfv[1:3,2]

  ## ---------------------------------------------------------------------
  ## coverage()
  ## ---------------------------------------------------------------------
  ## coverage() on a BigWigFileViews object returns coverage for all
  ## files in fileList() for the ranges in fileRange().

  ## When summarize = FALSE results are returned as a list. The
  ## by argument specifies grouping by range or by file.
  cv1 <- coverage(bwfv, by = "file", summarize = FALSE)
  length(cv1)
  elementLengths(cv1)

  cv2 <- coverage(bwfv, by = "range", summarize = FALSE)
  length(cv2)
  elementLengths(cv2)

  ## When summarize = TRUE output is a SummarizedExperiment.
  cv3 <- coverage(bwfv, summarize = TRUE)
  assays(cv3)

  ## FIXME: improve show method for Lists in a matrix
  ## FIXME: assays as List should respect withDimnames
  assays(cv3, withDimnames=TRUE)[[1]]

  ## ---------------------------------------------------------------------
  ## summary()
  ## ---------------------------------------------------------------------
  ## summary() on a BigWigFileViews object returns the type statistic
  ## for all files in fileList() for the ranges in fileRange().
  sm1 <- summary(bwfv, type = "mean", by = "file", summarize = TRUE)
  sm2 <- summary(bwfv, type = "max", by = "range", summarize = FALSE)
}
```

FaFileViews *Views into a set of Fasta files*

Description

Use `FaFileViews()` to reference a set of disk-based Fasta files to be processed

Constructor

`FaFileViews(fileList, fileSample=DataFrame(row.names=make.unique(basename(path(fileList)))),`
: This constructor is a generic function with dispatch on argument `fileList`. Methods exist for FaFileList and character (vector of file names).

Accessors

All accessor-like methods defined for `GenomicFileViews` objects work on `FaFileViews` objects. See ?`GenomicFileViews` for details.

- fileList(x); fileList(x) <- value
- fileSample(x); fileSample(x) <- value
- fileRange(x); fileRange(x) <- value
- fileExperiment(x); fileExpermient(x) <- value
- yieldSize(x); yieldSize(x) <- value

Methods

`"["`: Subset the object by `fileRange` or `fileSample`.

reduceByFile Parallel computations are distributed by files in `fileList` with the option to provide MAP and REDUCE functions across ranges and / or files.

reduceByRange Parallel computations are distributed by ranges in `fileRange` with the option to provide MAP and REDUCE functions across ranges and / or files.

Arguments

`fileList`: A character() vector of Fasta path names or a `FaFileList`.

`fileSample`: A DataFrame instance with as many rows as `length(fileList)`, containing sample information associated with each path.

`fileRange`: A `GRanges`, or missing instance with ranges defined on the spaces of the Fasta files. Ranges are *not* validated against the Fasta files.

`fileExperiment`: A list() containing additional information about the experiment.

`yieldSize`: An integer specifying number of records to process

`.views_on_file`: An environment; currently under development

`...`: Additional arguments.

`x`, `object`: An instance of `FaFileViews`.

`value`: An object of appropriate type to replace content.

`i`: During subsetting, a logical or numeric index into `fileRange`.

`j`: During subsetting, a logical or numeric index into `fileSample` and `fileList`.

`file`: An instance of `FaFileViews`.

`index`: Not used.

`param`: Unused option for `FaFileViews` object. `fileRange` are used to specify ranges to query.

Slots

Inherited from `GenomicFileViews` class:

fileList

- fileSample
- fileRange
- fileExperiment
- yieldSize
- .views_on_file

Author(s)

Martin Morgan <mtmorgan@fhcrc.org.> and Valerie Obenchain <vobencha@fhcrc.org>

See Also

- GenomicFileViews-class.

Examples

```
fa <- system.file("extdata", "ce2dict1.fa", package="Rsamtools",
                  mustWork=TRUE)
ffv <- FaFileViews(c(fa, fa), fileRange=scanFaIndex(fa))
ffv
```

`GenomicFiles` *GenomicFiles objects*

Description

The `GenomicFiles` class is a matrix-like container where rows represent ranges of interest and columns represent files. The class is designed for byFile or byRange queries and can be thought of as a precursor (or skeleton) of the `SummarizedExperiment` class.

`GenomicFiles` has slots for `files`, `rowData`, `colData` and `exptData`. In contrast to `SummarizedExperiment`, `GenomicFiles` does not contain an `assays` slot and the `rowData` is a `DataFrame` instead of a `GenomicRanges`.

Constructor

```
GenomicFiles(rowData, files, colData=DataFrame(),                 exptData=SimpleList(), ...):
```

Accessors

In the code below, `x` is a GenomicFiles object.

rowData, rowData(x) <- value Get or set the rowData on `x`. `value` can be a `GRanges` or `GRangesList` representing ranges or indices defined on the spaces (position) of the files.

files(x), files(x) <- value Get or set the files on `x`. `value` can be a character() of file paths or a List of file objects such as BamFile, BigWigFile etc.

colData, colData(x) <- value Get or set the colData on `x`. `value` must be a `DataFrame` instance describing the files. The number of rows must match the number of files. Row names, if present, become the column names of the `GenomicFiles`.

exptData, exptData(x) <- value Get or set the exptData on `x`. `value` must be a SimpleList of arbitrary content describing the overall experiment.

dimnames, dimnames(x) <- value Get or set the row and column names on `x`.

Methods

In the code below, `x` is a GenomicFiles object.

[Subset the object by `fileRange` or `fileSample`.

show Compactly display the object.

reduceByFile Extract, manipulate and combine data defined in `rowData` within the files specified in `files`. See `?reduceByFile` for details.

reduceByRange Extract, manipulate and combine data defined in `rowData` across the files specified in `files`. See `?reduceByRange` for details.

Author(s)

Martin Morgan <mtmorgan@fhcrc.org> and Valerie Obenchain <vobencha@fhcrc.org>

See Also

- reduceByFile and reduceByRange methods.

Examples

```
library(RNAseqData.HNRNPC.bam.chr14)
fls <- RNAseqData.HNRNPC.bam.chr14_BAMFILES
gr <- GRanges("chr14", IRanges(2e5*1:3, width=1e5, names=LETTERS[1:3]))
colData <- DataFrame(method=rep("RNASeq", length(fls)),
                     format=rep("bam", length(fls)))
gf <- GenomicFiles(gr, fls, colData=colData)
gf

dimnames(gf)
dim(gf[1, 5:8])
```

`GenomicFileViews` *Views into a set of files*

Description

`GenomicFileViews` is a VIRTUAL class used to reference a set of disk-based files to be queried across views (ranges).

Objects from the Class

`GenomicFileViews` is a VIRTUAL class not intended for instantiation by the user. The class serves as a parent for concrete subclasses such as `BamFileViews`, `FaFileViews`, `TabixFileViews` etc.

Slots

fileList List of of length >= 2 containing the file path and index names. List names must include 'path' and 'index'.

fileSample A DataFrame instance with as many rows as `length(fileList)`, containing sample information associated with each path.

fileRange A `GRanges` instance with ranges defined on the spaces (genomic position) of the files.

fileExperiment A `list` containing additional information about the experiment.

yieldSize An integer specifying the data chunk size.

.views_on_file An environment. Under construction / future use.

Accessors

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

itemfileList(x), fileList(x) <- value Get or set the fileList on `x`. `value` must be a List with list elements appropriate for the subclass.

fileSample, fileSample(x) <- value Get or set the fileSample on `x`. `value` must be a DataFrame instance with as many rows as `length(fileList)`, containing sample information associated with each file.

fileRange, fileRange(x) <- value Get or set the fileSample on `x`. `value` must be a `GRanges` instance.

fileExperiment, fileExperiment(x) <- value Get or set the fileExperiment on `x`. `value` must be a list().

yieldSize, yieldSize(x) <- value Get or set the yieldSize on `x`. `value` must be an integer.

names, names(x) <- value Get or set the names on `x`. These are the column names of the `GenomicFileViews` instance corresponding to the paths in fileList.

dimnames, dimnames(x) <- value Get or set the row and column names on `x`.

Methods

In the code snippets below, x is a GenomicFileViews object.

[Subset the object by fileRange or fileSample.

show Compactly display the object.

reduceByFile Parallel computations are distributed by files in fileList with the option to provide MAP and REDUCE functions across ranges and / or files.

reduceByRange Parallel computations are distributed by ranges in fileRange with the option to provide MAP and REDUCE functions across ranges and / or files.

Author(s)

Martin Morgan <mtmorgan@fhcrc.org.> and Valerie Obenchain <vobencha@fhcrc.org>

See Also

- BamFileViews-class class.
- BigWigFileViews-class class.
- reduceByFile and reduceByRange methods.

Examples

```
fl <- system.file("extdata", "ex1.bam", package="Rsamtools",
                  mustWork=TRUE)
bfv <- BamFileViews(fl, fileRange = GRanges("seq1", IRanges(1, 10)))
showClass(class(bfv))

## See ?BamFileViews, ?BigWigFileViews, ?reduceBy for additional examples.
```

---

pack *Range transformations of a* GenomicRanges *object for optimal file queries.*

Description

Given a GRanges object, pack produces a GRangesList of the same ranges grouped and re-ordered.

Usage

```
## S4 method for signature GRanges
pack(x, ..., range_len = 1e9, inter_range_len = 1e7)
```

Arguments

x	A GRanges object.
range_len	A numeric specifying the max length allowed for ranges in x.
inter_range_len	A numeric specifying the max length allowed between ranges in x.
...	Arguments passed to other methods.

Details

Packing ranges: The pack method attempts to re-package ranges in optimal form for extracting data from files. Ranges are not modified (made shorter or longer) but re-ordered and / or re-grouped according to the following criteria.

- order: Ranges are ordered by genomic position within chromosomes.
- distance: Ranges separted by a distance greater than the inter_range_len are packed in groups around the gap separating the distant ranges.
- length: Ranges longer than range_len are packed 'individually' (i.e., retrived from the file as a single range vs grouped with other ranges).

Utilities:

isPacked(x, ...): Returns a logical indicating if the ranges in x are packed. x must be a GRangesList object.

Value

A GRanges object.

See Also

- unpack for unpacking the result obtained with 'packed' ranges.

Examples

```
## Ranges are ordered by position within chromosome.
gr1 <- GRanges("chr1", IRanges(5:1*5, width = 3))
pack(gr1)

## Ranges separated by > inter_range_len are partitioned
## into groups defined by the endpoints of the gap.
gr2 <- GRanges("chr2", IRanges(c(1:3, 30000:30003), width = 1000))
pack(gr2, inter_range_len = 20000)

## Ranges exceeding range_len are isolated in a single element
## of the GRangesList.
gr3 <- GRanges("chr3", IRanges(c(1:4), width=c(45, 1e8, 45, 45)))
width(gr3)
pack(gr3, range_len = 1e7)
```

`reduceBy` *Parallel computations across files or ranges*

Description

Computations are distributed across files or ranges with the option to iteratively combine results.

Usage

```
## S4 method for signature GRanges,ANY
reduceByFile(ranges, files, MAP,
    REDUCE, ..., summarize=FALSE, iterate=TRUE, init)
## S4 method for signature GRangesList,ANY
reduceByFile(ranges, files, MAP,
    REDUCE, ..., summarize=FALSE, iterate=TRUE, init)
## S4 method for signature GenomicFiles,missing
reduceByFile(ranges, files, MAP,
    REDUCE, ..., summarize=FALSE, iterate=TRUE, init)

## S4 method for signature GRanges,ANY
reduceByRange(ranges, files, MAP,
    REDUCE, ..., summarize=FALSE, iterate=TRUE, init)
## S4 method for signature GRangesList,ANY
reduceByRange(ranges, files, MAP,
    REDUCE, ..., summarize=FALSE, iterate=TRUE, init)
## S4 method for signature GenomicFiles,missing
reduceByRange(ranges, files, MAP,
    REDUCE, ..., summarize=FALSE, iterate=TRUE, init)
```

Arguments

`ranges`	A `GRanges`, `GrangesList` or `GenomicFiles` object. When `ranges` is a `GenomicFiles`, the `files` argument is missing.
`files`	A `character` vector or `List` of filenames. When `ranges` is a `GenomicFiles` the `files` argument is missing; both ranges and files are extracted from the object.
`MAP`	A function executed on each worker. The signature must contain two arguments; the first represents the range(s) and the second the file(s). There is no restriction on the argument names and additional arguments may be provided. • `MAP = function(range, file, ...)`
`REDUCE`	An optional function that combines (reduces) output from the `MAP`. The first argument is the list output from MAP, additional arguments may be supplied. There are no restrictions on argument names. • `REDUCE = function(mapped, ...)`

	Reduction combines data from a single worker and is always performed as part of the distributed step. When `iterate=TRUE` `REDUCE` is applied after each `MAP` step; depending on the nature of `REDUCE`, iterative reduction can substantially decrease the data stored in memory. When `iterate=FALSE` reduction is applied to the list of `MAP` output applied to all files / ranges. When `REDUCE` is missing, output is a list from `MAP`.
`iterate`	A logical indicating if the `REDUCE` function should be applied iteratively to the output of `MAP`. When `REDUCE` is missing `iterate` is set to FALSE. Collapsing results iteratively is useful when the number of records to be procssed is large (maybe complete files) but the end result is a much reduced representation of all records. Iteratively applying `REDUCE` reduces the amount of data on each worker at any one time and can substantially reduce the memory footprint.
`summarize`	A logical indicating if results should be returned as a `SummarizedExperiment` object instead of a list. `SummarizedExperiment` requires matching dimensions across rows and columns of the slots. Becasue a `REDUCE` collapses one dimension (either ranges or files) the result cannnot be put in a `SummarizedExperiment`. When a `REDUCE` is provided `summarize` is ignored (i.e., set to FALSE).
`init`	An (optional) initial value for `REDUCE` when `iterate=TRUE`. `init` must be an object of the same type as the elements returned from `MAP`. `REDUCE` logically adds `init` to the start (when proceeding left to right) or end of results obtained with `MAP`.
`...`	Arguments passed to other methods.

Details

The reduceBy* functions offer two approaches to working with data subsets from multiple files. `reduceByFile` enables the extraction, manipulation and combination of data within files while `reduceByRanges` works across files.

Both `MAP` and `REDUCE` functions can be provided but only the `MAP` is required. The first two arguments to `MAP` are ‘range’ and ‘file’; the first argument to `REDUCE` is the list of output from the `MAP`.

Both `MAP` and `REDUCE` are applied in a distributed step. Currently there is no ’built-in’ ability to combine results across workers in the distributed step.

Value

Output is a `list` when `summarize=FALSE` (default) and a `SummarizedExperiment` when `summarize=TRUE`. Note that if `REDUCE` is provided `summarize` is ignored (i.e., set to FALSE).

When `ranges` is a `GenomicFiles` object and `summarize=TRUE`, data from `rowData`, `colData` and `exptData` are transferred to the `SummarizedExperiment`.

Author(s)

Martin Morgan <mtmorgan@fhcrc.org> and Valerie Obenchain <vobencha@fhcrc.org>

See Also

- GenomicFiles-class

Examples

```
if (all(require(RNAseqData.HNRNPC.bam.chr14) &&
        require(GenomicAlignments))) {
  fls <- RNAseqData.HNRNPC.bam.chr14_BAMFILES  ## 8 bam files

  ## -----------------------------------------------------------------------
  ## Basics of reduceByFile() and reduceByRange():
  ## -----------------------------------------------------------------------

  ## In this first example we provide a MAP only (no REDUCE).

  ## Ranges of interest.
  gr <- GRanges("chr14", IRanges(c(19100000, 106000000), width=1e7))

  ## The MAP counts the number of junctions in each range
  ## (i.e., N operations in the CIGAR).
  MAP <- function(range, file, ...) {
      library(GenomicAlignments)
      param = ScanBamParam(which=range)
      gal = readGAlignments(file, param=param)
                table(njunc(gal))
  }

  ## Length of the output corresponds to the number of files and
  ## the elementLengths to the number of ranges.
  rbf <- reduceByFile(gr, fls, MAP)
  length(rbf)          ## 8 files
  elementLengths(rbf)  ## 2 ranges

  ## Each list element contains a table of counts, one for each range.
  rbf[[1]]

  ## In contrast, reduceByRange() extracts data across files.
  rbr <- reduceByRange(gr, fls, MAP)

  ## Output length corresponds to the number of ranges.
  length(rbr)          ## 2 ranges
  elementLengths(rbr)  ## 8 files

  ## Each list element contains a table of counts, one for each file.
  do.call(rbind, rbr[[1]])

  ## Output a SummarizedExperiment instead of list:
  se <- reduceByRange(gr, fls, MAP, summarize=TRUE)
  assays(se)

  ## -----------------------------------------------------------------------
  ## Computing coverage across files:
  ## -----------------------------------------------------------------------

  ## Use reduceByRange() to compute coverage for a group of ranges
  ## across files.

  ## Regions of interest.
  gr <- GRanges("chr14", IRanges(c(62262735, 63121531, 63980327),
                width=214700))

  ## The MAP computes the pileups ...
  MAP <- function(range, file, ...) {
      library(GenomicRanges)
      param = ScanBamParam(which=range)
      coverage(file, param=param)[range]
  }

  ## and the REDUCE adds the last and current results.
  REDUCE <- function(mapped, ...)
      Reduce("+", mapped)

  ## Each call to coverage() produces an RleList which accumulate
  ## on the workers. When the REDUCE is applied iteratively the
  ## current result is collapsed with the last resulting in a
  ## maximum of 2 RleLists on a worker at a time.
  cov1 <- reduceByRange(gr, fls, MAP, REDUCE, iterate=TRUE)
  cov1[[1]]

  ## If memory use is not a concern (or if MAP output
  ## is not large) the REDUCE can be applied non-iteratively.
  cov2 <- reduceByRange(gr, fls, MAP, REDUCE, iterate=FALSE)

  ## Results match those obtained with the iterative REDUCE.
  cov2[[1]]

  ## -----------------------------------------------------------------------
  ## Organizing runs with the GenomicFiles class:
  ## -----------------------------------------------------------------------

  ## The GenomicFiles class is a light-weight form of SummarizedExperiment
  ## that does not have an assays slot.
  colData <- DataFrame(method=rep("RNASeq", length(fls)),
                       format=rep("bam", length(fls)))
  gf <- GenomicFiles(files=fls, rowData=gr, colData=colData)
  gf

  ## The object can be subset on ranges or files for different
  ## experimental runs.
  dim(gf)
  gf_sub <- gf[2, 3:4]
  dim(gf_sub)

  ## When summarize = TRUE and no REDUCE is provided the reduceBy*
  ## functions output a SummarizedExperiment object.
  se <- reduceByFile(gf, MAP=MAP, summarize=TRUE)
  se
```

```
  ## Data from the rowData, colData and exptData slots in the
  ## GenomicFiles are transferred to the SummarizedExperiment.
  colData(se)

  ## Results are in the assays slot.
  assays(se)
}
```

`reduceByYield` *Iterate through a BAM (or other) file, reducing output to a single result.*

Description

Rsamtools files can be created with a 'yieldSize' argument that influences the number of records (chunk size) input at one time (see, e.g,. `BamFile`). `reduceByYield` iterates through the file, processing each chunk and reducing it with previously input chunks. This is a memory efficient way to process large data files, especially when the final result fits in memory.

Usage

```
reduceByYield(X, YIELD, MAP, REDUCE,
              DONE = function(x) is.null(x) || length(x) == 0L,
              ..., parallel = FALSE, iterate = TRUE, init)
```

Arguments

`X`	A `BamFile` instance (or other class for which `isOpen`, `open`, `close` methods are defined, and which support extraction of sequential chunks).
`YIELD`	A function name or user-supplied function that operates on `X` to produce a `VALUE` that is passed to `DONE` and `MAP`. Generally `YIELD` will be a data extractor such as `readGAlignments`, `scanBam`, `yield`, etc. and `VALUE` is a chunk of data. • YIELD(X)
`MAP`	A function of one or more arguments that operates on the chunk of data from `YIELD`. • MAP(VALUE, ...)
`REDUCE`	A function of one (`iterate=FALSE` or two (`iterate=TRUE`) arguments, returning the reduction (e.g., addition) of the argument(s). If missing, `REDUCE` is `c` (when `iterate=TRUE`) or `identity` when (when `iterate=FALSE`). • REDUCE(all.mapped, ...) ## iterate=FALSE • REDUCE(x, y, ...) ## iterate=TRUE
`DONE`	A function of one argument, the `VALUE` output of the most recent call to `YIELD(X, ...)`. If missing, `DONE` is `function(VALUE) length(VALUE) == 0`.
`...`	Additional arguments, passed to `MAP`.

`iterate`	logical(1) determines whether the call to `REDUCE` is iterative (`iterate=TRUE`) or cumulative (`iterate=FALSE`).
`parallel`	logical(1) determines if the `MAP` step is run in parallel.
`init`	(Optional) Initial value used for `REDUCE` when `iterate=TRUE`.

Details

When `iterate=TRUE`, `REDUCE` is initially invoked with either the `init` value and the value of the first call to `MAP` or, if `init` is missing, the values of the first two calls to `MAP`.

When `iterate=FALSE`, `REDUCE` is invoked with a list containing a list with as many elements as there were calls to `MAP`. Each element the result of an invocation of `MAP`.

Value

The return value is the value returned by the final invocation of `REDUCE`, or `init` if provided and no data were yield'ed, or `list()` if `init` is missing and no data were yield'ed.

Author(s)

Martin Morgan `mtmorgan@fhcrc.org`

See Also

- `BamFile` and `TabixFile` for examples of 'X'.
- `reduceByFile` and `reduceByRange`

Examples

```
if (all(require(RNAseqData.HNRNPC.bam.chr14) &&
        require(GenomicAlignments))) {

  ## -----------------------------------------------------------------------
  ## Nucleotide frequency of mapped reads
  ## -----------------------------------------------------------------------

  ## In this example nucleotide frequency of mapped reads is computed
  ## for a single file. The MAP step is run in parallel and REDUCE
  ## is iterative.

  fl <- system.file(package="Rsamtools", "extdata", "ex1.bam")
  bf <- BamFile(fl, yieldSize=500) ## typically, yieldSize=1e6

  param <- ScanBamParam(
      flag=scanBamFlag(isUnmappedQuery=FALSE),
      what="seq")
  YIELD <- function(X, ...) scanBam(X, param, ...)[[1]][[seq]]
  MAP <- function(value, ...)
      alphabetFrequency(value, collapse=TRUE)
  REDUCE <- +        # add successive alphabetFrequency matrices
```

```
    reduceByYield(bf, YIELD, MAP, REDUCE, param=param, parallel=TRUE)

    ## ----------------------------------------------------------------------
    ## Coverage
    ## ----------------------------------------------------------------------

    ## reduceByYield() can be applied to multiple files by combining it
    ## with bplapply().

    ## FUN will be run on each worker; it contains the necessary arguments
    ## to reduceByYield() as well as a call to the function itself.
    ## reduceByYield() could also be run in parallel (parallel=TRUE)
    ## but in this example it is not.
    FUN <- function(bf) {
      library(GenomicAlignments)
      library(GenomicFiles)
      YIELD <- readGAlignments
      MAP <- function(value, ...) coverage(value)[["chr14"]]
      REDUCE <- +
      reduceByYield(bf, YIELD, MAP, REDUCE)
    }

    ## BAM files are distributed across Snow workers and each worker applies
    ## reduceByYield().
    bfl <- BamFileList(RNAseqData.HNRNPC.bam.chr14_BAMFILES[1:3])
    bplapply(bfl, FUN, BPPARAM = SnowParam(3))
  }
```

registry-utils — *Functions for creating and searching a registry of file types.*

Description

Functions for creating and searching a registry of file types based on file extension.

Usage

```
registerFileType(type, package, regex)
findTypeRegistry(fnames)
makeFileType(fnames, ..., regex=findTypeRegistry(fnames))
```

Arguments

`type`	The List class the file is associated with such as BamFileList, BigWigFileList, FaFileList.
`package`	The package where the List class (`type`) is defined.
`regex`	A regular expression that uniquely identifies the file extension.
`fnames`	A `character` vector of file names.
`...`	Additional arguments passed to the List-class constructor (e.g., yieldSize for BamFileList).

Details

- registerFileType The `registerFileType` function adds entries to the file type register created at load time. The point of the register is for discovery of file type (class) by file extension. These are List-type classes (e.g., BamFileList) that occupy the `fileList` slot of a GenomicFileViews class (e.g., BamFileViews).

  Each List class entry in the register is associated with (1) a regular expression that identifies the file extension, (2) a class and (3) the package where the class is defined. At load time the register is populated with classes known to GenomicFileViews. New classes / file types can be added to the register with `registerFileType` by providing these three pieces of information.
- findTypeRegistry Searches the registry for a match to the extension of `fname`. Internal use only.
- makeFileType Performs a look-up in the file registry based on the supplied regular expression; returns an object of the associated class. Internal use only.

Value

`registerFileType`: NULL

`findTypeRegistry`: The regular expression associated with the file.

`makeFileType`: A List-type object defined in the registry.

Examples

```
## At load time the registry is populated with file types
## known to GenomicFileViews.
sapply(as.list(.fileTypeRegistry), "[", "type")

## Add a new class to the file register.
## Not run: registerFileType(NewClassList, NewPackage, "\.NewExtension$")
```

`unpack` *Un-pack results obtained with a pack()ed group of ranges*

Description

unpack returns results obtained with pack()ed ranges to the geometry of the original, unpacked ranges.

Usage

```
## S4 method for signature list,GRangesList
unpack(flesh, skeleton, ...)
## S4 method for signature List,GRangesList
unpack(flesh, skeleton, ...)
```

Arguments

`flesh`	A `List` object to be unpacked; the result from querying a file with `skeleton`.
`skeleton`	The `GRangesList` created with ‘pack(x)‘.
`...`	Arguments passed to other methods.

Details

`unpack` returns a `List` obtained with packed ranges to the geometry and order of the original, unpacked ranges.

Value

A unpacked form of `flesh`.

See Also

- `pack` for packing ranges.

Examples

```
fl <- system.file("extdata", "ex1.bam", package = "Rsamtools")
gr <- GRanges(c(rep("seq2", 3), "seq1"),
              IRanges(c(75, 1, 100, 1), width = 2))

## Ranges are packed by order within chromosome and grouped
## around gaps greater than inter_range_len. See ?pack for details.
pk <- pack(gr, inter_range_len = 25)

## FUN computes coverage for the range passed as rng.
FUN <- function(rng, fl, param) {
    bamWhich(param) <- rng
    coverage(fl, param=param)[rng]
}

## Compute coverage on the packed ranges.
dat <- bplapply(as.list(pk), FUN, fl = fl, param = ScanBamParam())

## The result list contains RleLists of coverage.
lapply(dat, class)

## unpack() transforms the results back to the order of
## the original ranges (i.e., unpacked gr).
unpack(dat, pk)
```

Index