Package 'GenomicFiles'

October 7, 2014

Title Parallel queries distributed by file or by range

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

Version 1.0.1

Author Valerie Obenchain, Michael Love, Martin Morgan

Maintainer Bioconductor Package Maintainer <maintainer@bioconductor.org>

biocViews Infrastructure, DataImport

Depends R (>= 3.1.0), methods, BiocGenerics, BiocParallel, Rsamtools,rtracklayer (>= 1.23.16)

Imports GenomicAlignments

Suggests genefilter, deepSNV, Bioc-
Style, IRanges, RUnit,RNAseqData.HNRNPC.bam.chr14, TxDb.Hsapiens.UCSC.hg19.knownGene

License Artistic-2.0

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

```
if (.Platform$OS.type != "windows") {
  ## ---------------------------------------------------------------------
  ## BamFileView Objects
  ## ---------------------------------------------------------------------
  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)

## ----------------------------------------------------------------------
## Identify unique junctions (gaps in CIGAR) across files
## ----------------------------------------------------------------------

fls <- RNAseqData.HNRNPC.bam.chr14_BAMFILES
gr <- GRanges("chr14", IRanges(c(1.9e7, 2e7), width = 900000))
bfv <- BamFileViews(fls, fileRange=gr)
param <- ScanBamParam()

MAP = function(FILE, RANGE, ..., param) {
    require(GenomicAlignments)
    bamWhich(param) <- RANGE
    readGAlignmentPairs(FILE, param = param)
}
REDUCE = function(MAPPED, ...) {
    summarizeJunctions(do.call(c, unname(MAPPED)))
}
reduceByRange(bfv, MAP, REDUCE, param=param)

## ----------------------------------------------------------------------
## countBam(), scanBam(), summarizeOverlaps()
## ----------------------------------------------------------------------
## countBam(), scanBam() and summarizeOverlaps() are implemented
## for BamFileViews objects as a convenience. These do not use the
## reduceBy* and MAP/REDUCE functions but simply apply the function in
## parallel to each file in fileList defined by the ranges in fileRange.

## summarizeOverlaps()
library(TxDb.Hsapiens.UCSC.hg19.knownGene)
tx <- transcripts(TxDb.Hsapiens.UCSC.hg19.knownGene)

fls <- RNAseqData.HNRNPC.bam.chr14_BAMFILES
gr <- GRanges("chr14", IRanges(c(24656731, 104160127, 104160241),
                               width = 200))
bfv <- BamFileViews(fls, fileRange=gr)

res = summarizeOverlaps(tx, bfv, singleEnd = FALSE, inter.feature = FALSE)
colSums(assays(res)$counts)

## countBam()
head(countBam(bfv))

## scanBam()
```

```
  scn <- scanBam(bfv)
  lapply(scn[[1]], names)
}
```

`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)

## ----------------------------------------------------------------------
## reduceByRange() and reduceByFile()
## ----------------------------------------------------------------------
## reduceByRange() and reduceByFile() allow detailed control of how
## data are extracted and combined through the use of MAP and REDUCE
## functions.
```

```
  ## In this example a summary operation FUN is applied to the
  ## coverage from each range.
  coverage_summary <- function(X, MAP, REDUCE, FUN = mean, ...) {
      MAP <- function(FILE, RANGE, ...) {
          FUN(import(FILE, selection=RANGE, as="NumericList"))
      }
      REDUCE <- function(MAPPED, ...) unlist(MAPPED)
      reduceByRange(X, MAP, REDUCE)
  }

  coverage_summary(bwfv, MAP, REDUCE)
  coverage_summary(bwfv, MAP, REDUCE, FUN = median)
}
```

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 enviornment; 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

```
## ----------------------------------------------------------------------
## FaFileView Objects
## ----------------------------------------------------------------------

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

```
MAP <- function(FILE, RANGE, ...) {
    require(Rsamtools)
    scanFa(FILE, RANGE)
}
reduceByRange(ffv, MAP)
```

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.
```

---

reduceBy *Distribute parallel computations across files*

Description

Distribute parallel computations across files with the option to provide MAP and REDUCE functions across ranges and / or files.

Usage

```
## S4 method for signature GenomicFileViews
reduceByFile(X, MAP, REDUCE, ..., init, ITERATE=FALSE)

## S4 method for signature GenomicFileViews
reduceByRange(X, MAP, REDUCE, ..., init, ITERATE=FALSE)
```

Arguments

X	A GenomicFileViews object.
MAP	A function applied to the files in fileList or ranges in fileRange. MAP should have (at minimum) the two arguments RANGE and FILE. Signature is as follows: `MAP = function(FILE, RANGE, ...)` MAP can be used without a REDUCE function. In this case results from reduceByRange and reduceByFile are the same but returned in different order with different geometry (lists and list elements are different lengths).

REDUCE	A function applied to the output of MAP which (usually) performs an aggregation. The function signature of REDUCE depends on when aggregation occurs. When ITERATE is FALSE REDUCE is applied after the mapping step is complete; the signature requires a single argument (MAPPED below). When ITERATE is TRUE data are aggregated while MAP iterates through the files or ranges; the signature requires two arguments (X and Y below).
	• ITERATE = FALSE : MAPPED is the complete output from MAP.
	`REDUCE = function(MAPPED, ...)`
	• ITERATE = TRUE : X is the result from the last reduce step and Y is the most recent yield from MAP.
	`REDUCE = function(X, Y, ...)`

init	An (optional) initial value for REDUCE. Applicable 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. See ?Reduce for details.
ITERATE	A logical specifying if reduction should be performed during the mapping step (TRUE) or after (FALSE).
...	Arguments passed to other methods.

Details

Computations are distributed in parallel by file or by range by specifying reduceByFile or reduceByRange. In the distributed step, both reduceByFile and reduceByRange use MAP and REDUCE functions to further process and combine the data. MAP and REDUCE are based on the list-processing combinators from functional programming in the same spirit as Map and Reduce in base R. The all-caps designation of MAP and REDUCE is used to distinguish the functions in GenomicFileViews.

In the case of reduceByRange the mapping occurs between a single range and all files. The reduce step combines the results across files within the single range. With reduceByFile the mapping occurs between a single file and all ranges. The reduce step combines results across ranges within a single file.

The ITERATE argument controls when the REDUCE function is applied. When TRUE, REDUCE is applied while MAP iterates through the files or ranges; when FALSE, REDUCE is applied after MAP has processed all files or ranges.

- reduceByRange Computations are distributed in parallel over the ranges in fileRange. Each worker lapply()s the MAP function to a single range over all files. The output is a list the length of fileRange with each list element the length of fileList.

ITERATE = FALSE: Each worker applies REDUCE to MAP output from a single range, across all files; independent combination across files.

ITERATE = TRUE: Each worker applies REDUCE as MAP iterates through each file, single range; dependent combination across files.

- reduceByFile Computations are distributed in parallel by file. Each worker lapply()s the MAP function over all ranges in a single file. The output is a list the length of fileList with each list element the length of fileRange.

  ITERATE = FALSE: Each worker applies REDUCE to MAP output from a single file, across all ranges; independent combination across ranges.

  ITERATE = TRUE: Each worker applies REDUCE as MAP iterates through each range, a single file; dependent combination across ranges.

Value

A list object.

Author(s)

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

See Also

- GenomicFileViews-class
- Map
- Reduce

Examples

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

  bw <- system.file("tests", "test.bw", package = "rtracklayer")

  ## ---------------------------------------------------------------------
  ## MAP only:
  ## ---------------------------------------------------------------------
  gr <- GRanges(c("chr2", "chr19", "chr19"),
                IRanges(c(1, 1400, 1700), width=10))
  bwfv <- BigWigFileViews(c(bw, bw), fileRange=gr)
  MAP = function(FILE, RANGE, ...) {
      import(FILE, selection=RANGE, as="NumericList")
  }

  ## reduceByRange():
  ## List length is the number of ranges, list element lengths are
  ## the number of files.
  reduceByRange(bwfv, MAP)

  ## reduceByFile():
  ## List length is the number of files, list element lengths are
  ## the number of ranges.
```

```
reduceByFile(bwfv, MAP)

## ---------------------------------------------------------------------
## MAP, REDUCE and ITERATE:
## ---------------------------------------------------------------------
## This simple example demonstrates the use of ITERATE to
## control when output from MAP is aggregated.

gr <- tileGenome(seqlengths(BigWigFile(bw)),
                 tilewidth = 1e7,
                 cut.last.tile.in.chrom = TRUE)
bwv <- BigWigFileViews(c(bw, bw), fileRange = gr)
MAP = function(FILE, RANGE, ...) TRUE

## REDUCE is applied after MAP is complete.
REDUCE = function(MAPPED, ...) sum(unlist(MAPPED))
result1 <- reduceByRange(bwv, MAP, REDUCE)

## REDUCE is applied while MAP iterates though each file.
REDUCE = function(X, Y, ...) X + Y
result2 <- reduceByRange(bwv, MAP, REDUCE, ITERATE = TRUE)

stopifnot(identical(result1, result2))

## ---------------------------------------------------------------------
## t-test at point locations across files:
## ---------------------------------------------------------------------

library(genefilter)
gr <- tileGenome(seqlengths(BigWigFile(bw)),
                 tilewidth = 1e7,
                 cut.last.tile.in.chrom=TRUE)
bwv <- BigWigFileViews(c(bw, bw, bw, bw), fileRange = gr)
grp <- factor(c(1, 2, 2, 1))

MAP = function(FILE, RANGE, ...) {
    stopifnot(length(RANGE) == 1)
    v <- Views(import(FILE, selection = RANGE, asRle = TRUE),
               as(RANGE, "RangesList"))
    as.vector(Filter(length, v)[[1]][[1]])
}
REDUCE = function(MAPPED, ..., grp) {
    m <- simplify2array(MAPPED)
    idx <- which(rowSums(m) != 0)
    df <- rowttests(m[idx,], grp)
    cbind(offset=idx - 1, df)
}
result3 <- reduceByRange(bwv, MAP, REDUCE, grp = grp)

## Each list element contains the data.frame output
## from rowttests():
result3[[1]]
```

```
  ## ------------------------------------------------------------------------
  ## mean per-file:
  ## ------------------------------------------------------------------------

  gr <- GRanges(c("chr2", "chr19", "chr19"),
                IRanges(c(1, 1400, 1700), width = 1000))
  bwv <- BigWigFileViews(c(bw, bw, bw), fileRange = gr)

  MAP = function(FILE, RANGE, ...) {
      v <- Filter(length, {
          Views(import(FILE, selection=RANGE, asRle = TRUE),
                as(RANGE, "RangesList"))
      })
      c(n = as.numeric(sum(unlist(width(v)))),
        sum = as.numeric(sum(sapply(v, sum))))
  }
  REDUCE = function(X, Y, ..., FILE)
      c(n = X[["n"]] + Y[["n"]], sum = X[["sum"]] + Y[["sum"]])

  result4 <- reduceByFile(bwv, MAP, REDUCE, ITERATE = TRUE)
}
```

---

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$")
```

Index