Package 'GenomicScores'

February 22, 2019

Type Package

Title Infrastructure to work with genomewide position-specific scores

Description Provide infrastructure to store and access genomewide position-specific scores within R and Bioconductor.

Version 1.6.0

License Artistic-2.0

Depends R (>= 3.4), S4Vectors (>= 0.7.21), GenomicRanges, methods, BiocGenerics (>= 0.13.8)

Imports utils, XML, Biobase, IRanges (>= 2.3.23), Biostrings, BSgenome, GenomeInfoDb, AnnotationHub

Suggests BiocStyle, knitr, rmarkdown, BSgenome.Hsapiens.UCSC.hg19, phastCons100way.UCSC.hg19, MafDb.1Kgenomes.phase1.hs37d5, SNPlocs.Hsapiens.dbSNP144.GRCh37, VariantAnnotation, TxDb.Hsapiens.UCSC.hg19.knownGene, gwascat, RColorBrewer

VignetteBuilder knitr

URL https://github.com/rcastelo/GenomicScores

BugReports https://github.com/rcastelo/GenomicScores/issues

Encoding UTF-8

biocViews Infrastructure, Genetics, Annotation, Sequencing, Coverage

git_url https://git.bioconductor.org/packages/GenomicScores

git_branch RELEASE_3_8

git_last_commit 6644e90

git_last_commit_date 2018-10-30

Date/Publication 2019-02-21

Author Robert Castelo [aut, cre], Pau Puigdevall [ctb]

Maintainer Robert Castelo <robert.castelo@upf.edu>

R topics documented:

gscores *Accessing genomic gscores*

Description

Functions to access genomic gscores through GScores objects.

Usage

```
availableGScores()
getGScores(x)
## S4 method for signature 'GScores,GenomicRanges'
gscores(x, ranges, ...)
## S4 method for signature 'GScores,character'
gscores(x, ranges, ...)
## S4 method for signature 'MafDb,GenomicRanges'
gscores(x, ranges, ...)
## S4 method for signature 'GScores'
score(x, ..., simplify=TRUE)
## S4 method for signature 'MafDb'
score(x, ..., simplify=TRUE)
```

Arguments

x	For getGScores(), a character vector of length 1 specifiying the genomic scores resource to fetch. For gscores() and score(), a GScores object.
ranges	A GenomicRanges object with positions from where to retrieve genomic scores, or a character string vector with identifiers associated by the data producer to the genomic scores, e.g., dbSNP 'rs' identifiers.
...	In the call to the gscores() method one can additionally set the following arguments: • popCharacter string vector specifying the scores populations to query, when there is more than one. Use populations() to find out the available scores populations. • typeCharacter string specifying the type of genomic position being sought, which can be a single nucleotide range (snr), by default, or a nonsnr spanning multiple nucleotides. The latter is the case of indel variants in minor allele frequency data. • summaryFunFunction to summarize genomic scores when more than one position is retrieved. By default, this is set to the arithmetic mean, i.e., the mean() function. • quantizedFlag setting whether the genomic scores should be returned quantized (TRUE) or dequantized (FALSE, default). • refVector of reference alleles in the form of either a character vector, a DNAStringSet object or a DNAStringSetList object. This argument is used only when there are multiple scores per position. • altVector of alternative alleles in the form of either a character vector, a DNAStringSet object or a DNAStringSetList object. This argument is used only when there are multiple scores per position.

- minoverlapInteger value passed internally to the function `findOverlaps()` from the `IRanges` package, when querying genomic positions associated with multiple-nucleotide ranges (nonSNRs). By default, `minoverlap=1L`, which assumes that the sought nonSNRs are stored as in `VCF` files, using the nucleotide composition of the reference sequence. This argument is only relevant for genomic scores associated with nonSNRs.
- cachingFlag setting whether genomic scores per chromosome should be kept cached in memory (`TRUE`, default) or not (`FALSE`). The latter option minimizes the memory footprint but slows down the performance when the `gscores()` method is called multiple times.

`simplify` Flag setting whether the result should be simplified to a vector (`TRUE`, default) if possible. This happens when scores from a single population are queried.

Details

The function `availableGScores()` shows genomic score sets available as `AnnotationHub` online resources.

The method `gscores()` takes as first argument a `GScores-class` object that can be loaded from an annotation package or from an `AnnotationHub` resource. These two possibilities are illustrated in the examples below.

Value

The function `availableGScores()` returns a character vector with the names of the `AnnotationHub` resources corresponding to different available sets of genomic scores. The function `getGScores()` return a `GScores` object. The method `gscores()` returns a `GRanges` object with the genomic scores in a metadata column called `score`. The method `score()` returns a numeric vector with the genomic scores.

Author(s)

R. Castelo

See Also

`phastCons100way.UCSC.hg19` `MafDb.1Kgenomes.phase1.hs37d5`

Examples

```
## one genomic range of width 5
gr1 <- GRanges(seqnames="chr7", IRanges(start=117232380, width=5))
gr1

## five genomic ranges of width 1
gr2 <- GRanges(seqnames="chr7", IRanges(start=117232380:117232384, width=1))
gr2

## accessing genomic gscores from an annotation package
if (require(phastCons100way.UCSC.hg19)) {
  library(GenomicRanges)

  gsco <- phastCons100way.UCSC.hg19
  gsco
  gscores(gsco, gr1)
```

```
  score(gsco, gr1)
  gscores(gsco, gr2)
  populations(gsco)
  gscores(gsco, gr2, pop="DP2")
}

if (require(MafDb.1Kgenomes.phase1.hs37d5)) {
  mafdb <- MafDb.1Kgenomes.phase1.hs37d5
  mafdb
  populations(mafdb)

  ## lookup allele frequencies for SNP rs1129038, located at 15:28356859, a
  ## SNP associated to blue and brown eye colors as reported by Eiberg et al.
  ## Blue eye color in humans may be caused by a perfectly associated founder
  ## mutation in a regulatory element located within the HERC2 gene
  ## inhibiting OCA2 expression. Human Genetics, 123(2):177-87, 2008
  ## [http://www.ncbi.nlm.nih.gov/pubmed/18172690]
  gscores(mafdb, GRanges("15:28356859"), pop=populations(mafdb))
  gscores(mafdb, "rs1129038", pop=populations(mafdb))
}

## accessing genomic scores from AnnotationHub resources
## Not run:
availableGScores()
gsco <- getGScores("phastCons100way.UCSC.hg19")
gscores(gsco, gr1)

## End(Not run)
```

GScores-class *GScores objects*

Description

The goal of the GenomicScores package is to provide support to store and retrieve genomic scores associated to physical nucleotide positions along a genome. This is achieved through the GScores class of objects, which is a container for genomic score values.

Details

The GScores class attempts to provide a compact storage and efficient retrieval of genomic score values that have been typically processed and stored using some form of lossy compression. This class is currently based on a former version of the SNPlocs class defined in the BSgenome package, with the following slots:

provider (character), the data provider such as UCSC.

provider_version (character), the version of the data as given by the data provider, typically a date in some compact format.

download_url (character), the URL of the data provider from where the original data were downloaded.

download_date (character), the date on which the data were downloaded.

reference_genome (GenomeDescription), object with information about the reference genome whose physical positions have the genomic scores.

`data_pkgname` (`character`), name given to the set of genomic scores associated to a particular genome. When the genomic scores are stored within an annotation package, then this corresponds to the name of that package.

`data_dirpath` (`character`), absolute path to the local directory where the genomic scores are stored in one file per genome sequence.

`data_serialized_objnames` (`character`), named vector of filenames pointing to files containing the genomic scores in one file per genome sequence. The names of this vector correspond to the genome sequence names.

`data_group` (`character`), name denoting a category of genomic scores to which the scores stored in the object belong to. Typical values are "Conservation", "MAF", "Pathogenicity", etc.

`data_tag` (`character`), name identifying the genomic scores stored in the object and which can be used, for instance, to assign a column name storing these scores.

`data_pops` (`character`), vector of character strings storing score population names. The term "default" is reserved to denote a score set that is not associated to a particular population name and is used by default.

`data_nonsnrs` (`logical`), flag indicating whether the object stores genomic scores associated with non-single nucleotide ranges.

`data_nsites` (`integer`), number of sites in the genome associated with the genomic scores stored in the object.

`.data_cache` (`environment`), data structure where objects storing genomic scores are cached into main memory.

The goal of the design behind the `GScores` class is to load into main memory only the objects associated with the queried sequences to minimize the memory footprint, which may be advantageous in workflows that parallelize the access to genomic scores by genome sequence.

`GScores` objects are created either from `AnnotationHub` resources or when loading specific annotation packages that store genomic score values. Two such annotation packages are:

`phastCons100way.UCSC.hg19` Nucleotide-level phastCons conservation scores from the UCSC Genome Browser calculated from multiple genome alignments from the human genome version hg19 to 99 vertebrate species.

`phastCons100way.UCSC.hg38` Nucleotide-level phastCons conservation scores from the UCSC Genome Browser calculated from multiple genome alignments from the human genome version hg38 to 99 vertebrate species.

Constructor

`GScores(provider, provider_version, download_url, download_date, reference_genome, data`
Creates a `GScores` object. In principle, the end-user needs not to call this function.

`provider` character string, containing the data provider.

`provider_version` character string, containing the version of the data as given by the data provider.

`download_url` character string, containing the URL of the data provider from where the original data were downloaded.

`reference_genome` GenomeDescription, storing the information about the associated reference genome.

`data_pkgname` character string, name given to the set of genomic scores stored through this object.

`data_dirpath` character string, absolute path to the local directory where the genomic scores are stored.

`data_serialized_objname` character string vector, containing filenames where the genomic scores are stored.

`default_pop` character string, containing the name of the default scores population.

`data_group` character string, containing a name that indicates a category of genomic scores to which the scores in the object belong to. Typical names could be "Conservation", "MAF", etc.

`data_tag` character string, containing a tag that succintly labels genomic scores from a particular source. This can be used to automatically give, for instance, a name to a column storing genomic scores in data frame object. Its default value takes the prefix of the package name.

Accessors

`name(x)`: get the name of the set of genomic scores.

`type(x)`: get the substring of the name of the set of genomic scores comprised between the first character until the first period. This should typically match the type of genomic scores such as, `phastCons`, `phyloP`, etc.

`provider(x)`: get the data provider.

`providerVersion(x)`: get the provider version.

`organism(x)`: get the organism associated with the genomic scores.

`referenceGenome(x)`: get the `GenomeDescription` object associated with the genome on which the genomic scores are defined.

`seqlevelsStyle(x)`: get the genome sequence style.

`seqinfo(x)`: get the genome sequence information.

`seqnames(x)`: get the genome sequence names.

`seqlengths(x)`: get the genome sequence lengths.

`populations(x)`: get the identifiers of the available scores populations. If only one scores population is available, then it shows only the term `default`.

`defaultPopulation(x)`: get or set the default population of scores.

`gscoresGroup(x)`: get or set the genomic scores group label.

`gscoresTag(x)`: get or set the genomic scores tag label.

`gscoresNonSNRs(x)`: get whether there are genomic scores associated with non-single nucleotide ranges.

`nsites(x)`: get the number of sites in the genome with genomic scores.

`qfun(x)`: get the quantizer function.

`dqfun(x)`: get the dequantizer function.

`citation(x)`: get citation information for the genomic scores data in the form of a `bibentry` object.

Author(s)

R. Castelo

See Also

`gscores()` `score()` `phastCons100way.UCSC.hg19`

Examples

```
## one genomic range of width 5
gr1 <- GRanges(seqnames="chr7", IRanges(start=117232380, width=5))
gr1

## five genomic ranges of width 1
gr2 <- GRanges(seqnames="chr7", IRanges(start=117232380:117232384, width=1))
gr2

## supporting annotation packages with genomic scores
if (require(phastCons100way.UCSC.hg19)) {
  library(GenomicRanges)

  gsco <- phastCons100way.UCSC.hg19
  gsco
  gscores(gsco, gr1)
  score(gsco, gr1)
  gscores(gsco, gr2)
  populations(gsco)
  gscores(gsco, gr2, pop="DP2")
}

## supporting AnnotationHub resources
## Not run:
availableGScores()
gsco <- getGScores("phastCons100way.UCSC.hg19")
gsco
gscores(gsco, gr1)

## End(Not run)

## metadata from a GScores object
name(gsco)
type(gsco)
provider(gsco)
providerVersion(gsco)
organism(gsco)
referenceGenome(gsco)
seqlevelsStyle(gsco)
seqinfo(gsco)
head(seqnames(gsco))
head(seqlengths(gsco))
gscoresTag(gsco)
populations(gsco)
defaultPopulation(gsco)
qfun(gsco)
dqfun(gsco)
citation(gsco)
```

`MafDb-class` *MafDb class*

Description

Class for annotation packages storing minor allele frequency data.

Usage

```
## S4 method for signature 'MafDb'
mafByOverlaps(x, ranges, pop="AF", type=c("snvs", "nonsnvs"), maf.only=FALSE, caching=TRUE)
## S4 method for signature 'MafDb'
mafById(x, ids, pop="AF", maf.only=FALSE, caching)
## S4 method for signature 'MafDb'
populations(x)
```

Arguments

x	A MafDb object.
ranges	Either a GRanges object, a GPos object or a character string vector with the format "CHR:START[-END]".
ids	A character string vector with variant identifiers annotated by the MAF data source, typically dbSNP 'rs' identifiers. Note that the mapping of these identifiers to genomic positions and MAF values might be a subset of the most up to date dbSNP 'rs' identifier assignment to variants. To access the latter, please use the snpsById() method from the BSgenome package with the desired SNPlocs.* package.
pop	Character string vector with the populations for which we want to retrieve MAF values.
type	Character string setting the type of variant to seek, which can be either 'snvs' (default) when we seek single nucleotide variants or 'nonsnvs', otherwise.
maf.only	Flag set to FALSE (default) when MAF values are returned in metadata columns from the input GenomicRanges object. When set to TRUE, a DataFrame object is returned with the MAF values.
caching	logical; TRUE (default) indicates that the function stores into main memeory the MAF data as it gets loaded from disk, improving performance; FALSE forces this function to load MAF data from disk each time, decreasing performance and memory requirements.

Details

This class has been deprecated in Bioconductor 3.7, is being replaced by the GScores-class and will become defunct and unavailable in Bioconductor 3.8.

The MafDb class is derived from the GScores class and it serves the purpose of providing support to store and access minor allele frequency (MAF) data from R and Bioconductor. Two annotation packages using the MafDb class are:

MafDb.1Kgenomes.phase1.hs37d5	MAF values from the 1000 Genomes Project Phase 1.
MafDb.1Kgenomes.phase3.hs37d5	MAF values from the 1000 Genomes Project Phase 3.

This object class tries to reduce the disk space required to store MAF values for millions of SNPs by coding their double-precision values, which range between 0 and 1, into a single-byte raw object type. To achieve this, the original MAF values are rounded to one significant digit for AF < 0.1 and two significant digits for AF >= 0.1. When a variant has multiple alternate alleles, only the largest MAF value is stored.

Author(s)

R. Castelo

Examples

```
## Not run:
 ## lookup allele frequencies for rs1129038, a SNP associated to blue and brown eye colors
 ## as reported by Eiberg et al. Blue eye color in humans may be caused by a perfectly associated
 ## founder mutation in a regulatory element located within the HERC2 gene inhibiting OCA2 expression.
  ## Human Genetics, 123(2):177-87, 2008 [http://www.ncbi.nlm.nih.gov/pubmed/18172690]

 if (require(MafDb.1Kgenomes.phase1.hs37d5)) {
   mafdb <- MafDb.1Kgenomes.phase1.hs37d5
   mafdb

   ## specialized interface
   populations(mafdb)

   rng <- GRanges("15", IRanges(28356859, 28356859))
   mafByOverlaps(mafdb, rng)
   mafByOverlaps(mafdb, "15:28356859-28356859")
   mafByOverlaps(mafdb, "15:28356859")
   mafById(mafdb, "rs1129038")
 }

## End(Not run)
```

Index