Package ‘seqminer’

September 2, 2023

Type Package

Title Efficiently Read Sequence Data (VCF Format, BCF Format, METAL Format and BGEN Format) into R

Version 9.1

Date 2023-9-1

Maintainer Xiaowei Zhan <zhanxw@gmail.com>

Description Integrate sequencing data (Variant call format, e.g. VCF or BCF) or meta-analysis results in R. This package can help you (1) read VCF/BCF/BGEN files by chromosomal ranges (e.g. 1:100-200); (2) read RareMETAL summary statistics files; (3) read tables from a tabix-indexed files; (4) annotate VCF/BCF files; (5) create customized workflow based on Makefile.

Copyright We have used the following software and made minimal necessary changes: tabix, Heng Li <lh3@live.co.uk> (MIT license), SQLite (Public Domain), Zstandard (BSD license). For tabix, we removed standard IO related functions, e.g. printf, fprintf ; also changed its un-safe pointer arithmetics. For zstandard, we removed compiler (clang, MSVC) specific preprocessing flags.

License GPL | file LICENSE

URL http://zhanxw.github.io/seqminer/

BugReports https://github.com/zhanxw/seqminer/issues

Repository CRAN

Suggests testthat, SKAT

SystemRequirements C++17, zlib headers and libraries, GNU make, optionally also bzip2 and POSIX-compliant regex functions.

NeedsCompilation yes

RoxygenNote 7.2.3

Encoding UTF-8

Author Xiaowei Zhan [aut, cre], Dajiang Liu [aut]
Attractive Chaos [cph] (We have used the following software and made
minimal necessary changes: Tabix, Heng Li <lh3@live.co.uk> (MIT
license). We removed standard IO related functions, e.g. printf,
fprintf ; also changed its un-safe pointer arithmetics.),
Broad Institute / Massachusetts Institute of Technology [cph],
Genome Research Ltd (GRL) [cph],
Facebook, Inc [cph],
D. Richard Hipp [cph]

Date/Publication 2023-09-02 06:50:02 UTC

R topics documented:

addJob .......................... 3
annotateGene .......................... 4
annotatePlain .......................... 5
annotateVcf .......................... 5
createSingleChromosomeBCFIndex .......................... 6
createSingleChromosomeVCFIndex .......................... 7
download.annotation.resource .......................... 7
getCovPair .......................... 8
getRefBase .......................... 8
isDirWritable .......................... 9
isInRange .......................... 9
isTabixRange ........................ 10
makeAnnotationParameter ........................ 10
newJob .......................... 11
ewWorkflow ........................ 11
openPlink ........................ 12
readBGENToListByGene ........................ 12
readBGENToListByRange ........................ 13
readBGENToMatrixByGene ........................ 14
readBGENToMatrixByRange ........................ 14
readPlinkToMatrixByIndex ........................ 15
readSingleChromosomeBCFToMatrixByRange ........................ 16
readSingleChromosomeVCFToMatrixByRange ........................ 17
readVCFToListByGene ........................ 17
readVCFToListByRange ........................ 19
readVCFToMatrixByGene ........................ 20
readVCFToMatrixByRange ........................ 20
rvmeta.readCovByRange ........................ 21
rvmeta.readDataByGene ........................ 22
rvmeta.readDataByRange ........................ 23
rvmeta.readNullModel ........................ 24
rvmeta.readScoreByRange ........................ 24
rvmeta.readSkewByRange ........................ 25
rvmeta.writeCovData ........................ 26
rvmeta.writeScoreData ........................ 26
addJob

Add a job to a workflow

Description

Add a job to a workflow

Usage

addJob(wf, job)

Arguments

wf a variable of workflow class

job a variable of job class

Examples

j1 <- newJob('id1', 'cmd out1', 'out1')
j2 <- newJob('id2', 'cmd out2', 'out2', depend = 'out1')
w <- newWorkflow("wf")
w <- addJob(w, j1)
w <- addJob(w, j2)

outFile <- file.path(tempdir(), "Makefile")
writeWorkflow(w, outFile)
cat("Outputted Makefile file are in the temp directory:", outFile, "\n")
annotateGene  

Annotate a test variant

Description

Annotate a test variant

Usage

annotateGene(param, chrom, position, ref, alt)

Arguments

- param: a list of annotation configuration (e.g. reference file, gene definition)
- chrom: a vector of chromosome names
- position: a vector of chromosome positions
- ref: a vector of reference alleles
- alt: a vector of alternative alleles

Value

annotated results in a data frame structure

See Also

makeAnnotationParameter

Examples

if (.Platform$endian == "little") {
  param <- list(reference = system.file("tabanno/test.fa", package = "seqminer"),
                 geneFile = system.file("tabanno/test.gene.txt", package = "seqminer"))
  param <- makeAnnotationParameter(param)
  print(param)
  annotateGene(param, c("1", "1"), c(3, 5), c("A", "C"), c("G", "C"))
} else {
  message("Tabix does not work well for big endian for now")
}
annotatePlain  

Annotate a plain text file

Description
Annotate a plain text file

Usage
annotatePlain(inFile, outFile, params)

Arguments

- inFile: input file name
- outFile: output file name
- params: parameters

Value

0 if succeed

Examples

```r
param <- list(reference = system.file("tabanno/test.fa", package = "seqminer"),
               geneFile = system.file("tabanno/test.gene.txt", package = "seqminer"),
               inputFormat = "plain")
param <- makeAnnotationParameter(param)
inFile <- system.file("tabanno/input.test.plain.txt", package = "seqminer")
outFile <- file.path(tempdir(), "out.annotated.txt")
annotatePlain(inFile, outFile, param)
cat("Outputted annotation results are in the temp directory: ", outFile, 
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
createSingleChromosomeBCFIndex

Create a single chromosome index

Description

Create a single chromosome index

Usage

createSingleChromosomeBCFIndex(fileName, indexFileName = NULL)

Arguments

fileName character, represents an input BCF file (Bgzipped, with Tabix index)

indexFileName character, by default, create ‘fileName’.scIdx

Value

indexFileName if success, or NULL is failed

See Also

http://zhanxw.com/seqminer/ for online manual and examples

Examples

fileName = system.file("vcf/all.anno.filtered.extract.headerFixed.bcf.gz", package = "seqminer")
cfh <- createSingleChromosomeBCFIndex(fileName)
createSingleChromosomeVCFIndex

Create a single chromosome index

Description
Create a single chromosome index

Usage
createSingleChromosomeVCFIndex(fileName, indexFileName = NULL)

Arguments
fileName character, represents an input VCF file (Bgzipped, with Tabix index)
indexFileName character, by default, create ‘fileName’.scIdx

Value
indexFileName if success, or NULL is failed

See Also
http://zhanxw.com/seqminer/ for online manual and examples

Examples
fileName = system.file("vcf/all.anno.filtered.extract.vcf.gz", package = "seqminer")
cfh <- createSingleChromosomeVCFIndex(fileName)

download.annotation.resource

Download annotation resources to a directory

Description
Download annotation resources to a directory

Usage
download.annotation.resource(outputDirectory)

Arguments
outputDirectory
the directory to store annotation resources
getRefBase

Description
Annotate a test variant

Usage
getRefBase(reference, chrom, position, len = NULL)

getcovPair
Extract pair of positions by ranges

Description
Extract pair of positions by ranges

Usage
getcovPair(covData, rangeList1, rangeList2)

Arguments
covData a covariance matrix with positions as dimnames
rangeList1 character specify a range, 1-based index
rangeList2 character specify a range, 1-based index

Value
a covariance matrix covFileName = system.file("rvtests/rvtest.MetaCov.assoc.gz", package = "seqminer") cfh <- rmeta.readCovByRange(covFileName, "1:196621007-196716634") rangeList1 <- "1:196621007-196700000" rangeList2 <- "1:196700000-196716634" getCovPair(cfh, rangeList1, rangeList2)

getRefBase
Annotate a test variant

Description
Annotate a test variant

Usage
getRefBase(reference, chrom, position, len = NULL)
**isDirWritable**

**Arguments**
- `reference`: path to the reference genome file (.fa file)
- `chrom`: a vector of chromosome names
- `position`: a vector of chromosome positions
- `len`: a vector of length

**Value**

based extracted from the reference genome

---

**isDirWritable**

Test whether directory is writable

**Description**

Test whether directory is writable

**Usage**

```r
isDirWritable(outDir)
```

**Arguments**
- `outDir`: the name of the directory

**Value**

TRUE if the file is writable `isDirWritable("~")`

---

**isInRange**

Test whether a vector of positions are inside given ranges

**Description**

Test whether a vector of positions are inside given ranges

**Usage**

```r
isInRange(positions, rangeList)
```

**Arguments**
- `positions`: characters, positions. e.g. `c("1:2-3", "1:4")`
- `rangeList`: character, ranges, e.g. `"1:1-3,1:2-4"`, 1-based index
Value

logical vector, TRUE/FALSE/NA

Examples

positions <- c("1:2-3", "1:4", "XX")
ranges <- "1:1-3,1:2-4,1:5-10"
isInRange(positions, ranges)

isTabixRange

Check if the inputs are valid tabix range such as chr1:2-300

Description

Check if the inputs are valid tabix range such as chr1:2-300

Usage

isTabixRange(range)

Arguments

range character vector

Examples

valid <- isTabixRange(c("chr1:1-200", "X:1", "1:100-100", "chr1", "1:1-20,1:30-40"))
stopifnot(all(valid))
invalid <- isTabixRange(c(":1", "chr1::", "::"))
stopifnot(all(!invalid))
Value
  list, a complete list of supported parameters

newJob  Create a new job

Description
  Create a new job

Usage
  newJob(id, cmd, outFile, depend = NULL)

Arguments
  id   character, job ids.
  cmd  character, commands to run
  outFile  character, the output file names after command are run successfully
  depend  character vector, specify the prerequisite files (e.g. outFile from other jobs)

Examples
  j1 <- newJob("id1", "cmd out1", "out1")
  j2 <- newJob("id2", "cmd out2", "out2", depend = "out1")

newWorkflow  Create a new workflow

Description
  Create a new workflow

Usage
  newWorkflow(name)

Arguments
  name  character, specify the name of the workflow

Examples
  w <- newWorkflow("wf")
**openPlink**

*Open binary PLINK files*

**Description**
Open binary PLINK files

**Usage**
`openPlink(fileName)`

**Arguments**
- **fileName** character, represents the prefix of PLINK input file

**Value**
an PLINK file object with class name ("PlinkFile")

**Examples**
```r
code
fileName = system.file("plink/all.anno.filtered.extract.bed", package = "seqminer")
fileName = sub(fileName, pattern = ".bed", replacement = "")
plinkObj <- openPlink(fileName)
str(plinkObj)
```

---

**readBGENToListByGene**

*Read information from BGEN file in a given range and return a list*

**Description**
Read information from BGEN file in a given range and return a list

**Usage**
`readBGENToListByGene(fileName, geneFile, geneName)`

**Arguments**
- **fileName** character, represents an input BGEN file (Bgzipped, with Tabix index)
- **geneFile** character, a text file listing all genes in refFlat format
- **geneName** character vector, which gene(s) to be extracted

**Value**
a list of chrom, pos, varid, rsid, alleles, isPhased, probability, sampleId
readBGENToListByRange

See Also

http://zhanxw.com/seqminer/ for online manual and examples

Examples

fileName = system.file("bgen/all.anno.filtered.extract.bgen", package = "seqminer")
geneFile = system.file("vcf/refFlat_hg19_6col.txt.gz", package = "seqminer")
cfh <- readBGENToListByGene(fileName, geneFile, "CFH")

cfh <- readBGENToListByRange(fileName, "1:196621007-196716634")

description

Read information from BGEN file in a given range and return a list

Usage

readBGENToListByRange(fileName, range)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fileName</td>
<td>character, represents an input BGEN file (Bgzipped, with Tabix index)</td>
</tr>
<tr>
<td>range</td>
<td>character, a text indicating which range in the BGEN file to extract. e.g. 1:100-200, 1-based index</td>
</tr>
</tbody>
</table>

Value

a list of chrom, pos, varid, rsid, alleles, isPhased, probability, sampleId

See Also

http://zhanxw.com/seqminer/ for online manual and examples

Examples

fileName = system.file("bgen/all.anno.filtered.extract.bgen", package = "seqminer")
cfh <- readBGENToListByRange(fileName, "1:196621007-196716634")
readBGENToMatrixByGene

*Read a gene from BGEN file and return a genotype matrix*

**Description**

Read a gene from BGEN file and return a genotype matrix

**Usage**

```r
readBGENToMatrixByGene(fileName, geneFile, geneName)
```

**Arguments**

- `fileName`: character, represents an input BGEN file (Bgzipped, with Tabix index)
- `geneFile`: character, a text file listing all genes in refFlat format
- `geneName`: character vector, which gene(s) to be extracted

**Value**

genotype matrix

**See Also**

http://zhanxw.com/seqminer/ for online manual and examples

**Examples**

```r
fileName = system.file("bgen/all.anno.filtered.extract.bgen", package = "seqminer")
geneFile = system.file("vcf/refFlat_hg19_6col.txt.gz", package = "seqminer")
cfh <- readBGENToMatrixByGene(fileName, geneFile, "CFH")
```

---

readBGENToMatrixByRange

*Read a gene from BGEN file and return a genotype matrix*

**Description**

Read a gene from BGEN file and return a genotype matrix

**Usage**

```r
readBGENToMatrixByRange(fileName, range)
```
**Arguments**

- `fileName` character, represents an input BGEN file (Bgzipped, with Tabix index)
- `range` character, a text indicating which range in the BGEN file to extract. e.g. 1:100-200, 1-based index

**Value**

genotype matrix

**See Also**

http://zhanxw.com/seqminer/ for online manual and examples

**Examples**

```r
fileName = system.file("bgen/all.anno.filtered.extract.bgen", package = "seqminer")
cfh <- readBGENToMatrixByRange(fileName, "1:196621007-196716634")
```

---

**readPlinkToMatrixByIndex**

*Read from binary PLINK file and return a genotype matrix*

**Description**

Read from binary PLINK file and return a genotype matrix

**Usage**

```r
readPlinkToMatrixByIndex(plinkFilePrefix, sampleIndex, markerIndex)
```

**Arguments**

- `plinkFilePrefix` a PlinkFileObject obtained by openPlink()
- `sampleIndex` integer, 1-based, index of samples to be extracted
- `markerIndex` integer, 1-based, index of markers to be extracted

**Value**

genotype matrix, marker by sample

**See Also**

http://zhanxw.com/seqminer/ for online manual and examples
Examples

```r
## these indice are nonsynonymous markers for 1:196621007-196716634
fileName = system.file("plink/all.anno.filtered.extract.bed", package = "seqminer")
fileName = sub(fileName, pattern = ".bed", replacement = "")
sampleIndex = seq(3)
markerIndex = c(14, 36)
cfh <- readPlinkToMatrixByIndex(fileName, sampleIndex, markerIndex)
```

---

**readSingleChromosomeBCFToMatrixByRange**

*Read a range from BCF file and return a genotype matrix*

**Description**

Read a range from BCF file and return a genotype matrix

**Usage**

```r
readSingleChromosomeBCFToMatrixByRange(fileName, range, indexFileName = NULL)
```

**Arguments**

- `fileName` character, represents an input BCF file (Bgzipped, with Tabix index)
- `range` character, a text indicating which range in the BCF file to extract. e.g. 1:100-200, 1-based index
- `indexFileName` character, index file, by default, it’s `fileName`.scIdx

**Value**

genotype matrix

**See Also**

http://zhanxw.com/seqminer/ for online manual and examples

**Examples**

```r
fileName = system.file("vcf/all.anno.filtered.extract.headerFixed.bcf.gz", package = "seqminer")
cfh <- readSingleChromosomeBCFToMatrixByRange(fileName, "1:196621007-196716634")
```
**Description**

Read a range from VCF file and return a genotype matrix

**Usage**

```r
readSingleChromosomeVCFToMatrixByRange(fileName, range, indexFileName = NULL)
```

**Arguments**

- `fileName`: character, represents an input VCF file (Bgzipped, with Tabix index)
- `range`: character, a text indicating which range in the VCF file to extract. e.g. 1:100-200, 1-based index
- `indexFileName`: character, index file, by default, it’s `fileName`.scIdx

**Value**

genotype matrix

**See Also**

http://zhanxw.com/seqminer/ for online manual and examples

**Examples**

```r
fileName = system.file("vcf/all.anno.filtered.extract.vcf.gz", package = "seqminer")
cfh <- readSingleChromosomeVCFToMatrixByRange(fileName, "1:196621007-196716634")
```

---

**readVCFToListByGene**

*Read information from VCF file in a given range and return a list*

**Description**

Read information from VCF file in a given range and return a list
readVCFToListByGene

Usage

readVCFToListByGene(
  fileName,
  geneFile,
  geneName,
  annoType,
  vcfColumn,
  vcfInfo,
  vcfIndv
)

Arguments

fileName character, represents an input VCF file (Bgzipped, with Tabix index)
geneFile character, a text file listing all genes in refFlat format
geneName character vector, which gene(s) to be extracted
annoType character, annotated types you would like to extract, such as "Nonsynonymous", "Synonymous". This can be left empty.
vcfColumn character vector, which vcf columns to extract. It can be chosen from CHROM, POS, ID, REF, ALT, QUAL, FILTER, INFO, FORMAT and etc.
vcfInfo character vector, which should be tags in the INFO columns to extrarct. Common choices include: DP, AC, AF, NS
vcfIndv character vector, which values to extract at individual level. Common choices are: GT, GQ, GD

Value

a list of genes, and each elements has specified vcfColumn, vcfinfo, vcfIndv

See Also

http://zhanxw.com/seqminer/ for online manual and examples

Examples

fileName = system.file("vcf/all.anno.filtered.extract.vcf.gz", package = "seqminer")
geneFile = system.file("vcf/refFlat_hg19_6col.txt.gz", package = "seqminer")
cfh <- readVCFToListByGene(fileName, geneFile, "CFH", "Synonymous",
                           c("CHROM", "POS"), c("AF", "AC"), c("GT") )
**readVCFToListByRange**  
*Read information from VCF file in a given range and return a list*

**Description**
Read information from VCF file in a given range and return a list

**Usage**
```r
readVCFToListByRange(fileName, range, annoType, vcfColumn, vcfInfo, vcfIndv)
```

**Arguments**
- **fileName**: character, represents an input VCF file (Bgzipped, with Tabix index)
- **range**: character, a text indicating which range in the VCF file to extract. e.g. 1:100-200, 1-based index
- **annoType**: character, annotated types you would like to extract, such as "Nonsynonymous", "Synonymous". This can be left empty.
- **vcfColumn**: character vector, which vcf columns to extract. It can be chosen from CHROM, POS, ID, REF, ALT, QUAL, FILTER, INFO, FORMAT and etc.
- **vcfInfo**: character vector, which should be tags in the INFO columns to extract. Common choices include: DP, AC, AF, NS
- **vcfIndv**: character vector, which values to extract at individual level. Common choices are: GT, GQ, GD

**Value**
a list of genes, and each element has specified vcfColumn, vcfinfo, vcfIndv

**See Also**
http://zhanxw.com/seqminer/ for online manual and examples

**Examples**
```r
fileName = system.file("vcf/all.anno.filtered.extract.vcf.gz", package = "seqminer")
cfh <- readVCFToListByRange(fileName, "1:196621007-196716634", "Nonsynonymous",
                          c("CHROM", "POS"), c("AF", "AC"), c("GT") )
```
readVCFToMatrixByGene  
*Read a gene from VCF file and return a genotype matrix*

**Description**
Read a gene from VCF file and return a genotype matrix

**Usage**
readVCFToMatrixByGene(fileName, geneFile, geneName, annoType)

**Arguments**
- `fileName` character, represents an input VCF file (Bgzipped, with Tabix index)
- `geneFile` character, a text file listing all genes in refFlat format
- `geneName` character vector, which gene(s) to be extracted
- `annoType` character, annotated types you would like to extract, such as "Nonsynonymous", "Synonymous". This can be left empty.

**Value**
genotype matrix

**See Also**
http://zhanxw.com/seqminer/ for online manual and examples

**Examples**
```r
fileName = system.file("vcf/all.anno.filtered.extract.vcf.gz", package = "seqminer")
geneFile = system.file("vcf/refFlat_hg19_6col.txt.gz", package = "seqminer")
cfh <- readVCFToMatrixByGene(fileName, geneFile, "CFH", "Synonymous")
```

---

readVCFToMatrixByRange  
*Read a gene from VCF file and return a genotype matrix*

**Description**
Read a gene from VCF file and return a genotype matrix

**Usage**
readVCFToMatrixByRange(fileName, range, annoType)

---
**rvmeta.readCovByRange**

**Description**

Read covariance by range from METAL-format files.

**Usage**

```r
cmeta.readCovByRange(covFile, tabixRange)
```

**Arguments**

- `covFile`: character, a covariance file (rvtests outputs using –meta cov)
- `tabixRange`: character, a text indicating which range in the VCF file to extract. e.g. 1:100-200

**Value**

a matrix of covariance within given range

**See Also**

http://zhanxw.com/seqminer/ for online manual and examples

**Examples**

```r
covFileName = system.file("rvtests/rvtest.MetaCov.assoc.gz", package = "seqminer")
cfh <- rvmeta.readCovByRange(covFileName, "1:196621007-196716634")
```
rvmeta.readDataByGene  Read association statistics by gene from METAL-format files. Both score statistics and covariance statistics will be extracted.

Description
Read association statistics by gene from METAL-format files. Both score statistics and covariance statistics will be extracted.

Usage
rvmeta.readDataByGene(
  scoreTestFiles,  
covFiles,  
geneFile,  
geneName,  
multiAllelic = FALSE
)

Arguments
scoreTestFiles  character vector, score test output files (rvtests outputs using –meta score)
covFiles  character vector, covaraite files (rvtests outputs using –meta cov)
geneFile  character, a text file listing all genes in refFlat format
geneName  character vector, which gene(s) to be extracted
multiAllelic  boolean, whether to read multi-allelic sites as multiple variants or not

Value
a list of statistics including chromosome, position, allele frequency, score statistics, covariance and annotation(if input files are annotated).

See Also
http://zhanxw.com/seqminer/ for online manual and examples

Examples
scoreFileName = system.file("rvtests/rvtest.MetaScore.assoc.anno.gz", package = "seqminer")
covFileName = system.file("rvtests/rvtest.MetaCov.assoc.gz", package = "seqminer")
geneFile = system.file("vcf/refFlat_hg19_6col.txt.gz", package = "seqminer")
cfh <- rvmeta.readDataByGene(scoreFileName, covFileName, geneFile, "CFH")
rvmeta.readDataByRange

Read association statistics by range from METAL-format files. Both score statistics and covariance statistics will be extracted.

Description

Read association statistics by range from METAL-format files. Both score statistics and covariance statistics will be extracted.

Usage

rvmeta.readDataByRange(scoreTestFiles, covFiles, ranges, multiAllelic = FALSE)

Arguments

- **scoreTestFiles** character vector, score test output files (rvtests outputs using \(-\)meta score)
- **covFiles** character vector, covariate files (rvtests outputs using \(-\)meta cov)
- **ranges** character, a text indicating which range in the VCF file to extract. e.g. 1:100-200, 1-based index
- **multiAllelic** boolean, whether to read multi-allelic sites as multiple variants or not

Value

a list of statistics including chromosome, position, allele frequency, score statistics, covariance and annotation(if input files are annotated).

See Also

http://zhanxw.com/seqminer/ for online manual and examples

Examples

scoreFileName = system.file("rvtests/rvtest.MetaScore.assoc.anno.gz", package = "seqminer")
covFileName = system.file("rvtests/rvtest.MetaCov.assoc.gz", package = "seqminer")
geneFile = system.file("vcf/refFlat_hg19_6col.txt.gz", package = "seqminer")
cfh <- rvmeta.readDataByRange(scoreFileName, covFileName, "1:196621007-196716634")
rvmeta.readNullModel  
Read null model statistics

Description
Read null model statistics

Usage
rvmeta.readNullModel(scoreTestFiles)

Arguments
scoreTestFiles  character vector, score test output files (rvtests outputs using –meta score)

Value
a list of statistics fitted under the null mode (without genetic effects)

See Also
http://zhanxw.com/seqminer/ for online manual and examples

Examples
scoreFileName = system.file("rvtests/rvtest.MetaScore.assoc.anno.gz", package = "seqminer")

rvmeta.readScoreByRange  
Read score test statistics by range from METAL-format files.

Description
Read score test statistics by range from METAL-format files.

Usage
rvmeta.readScoreByRange(scoreTestFiles, tabixRange)

Arguments
scoreTestFiles  character vector, score test output files (rvtests outputs using –meta score)
tabixRange  character, a text indicating which range in the VCF file to extract. e.g. 1:100-200

Value
score test statistics within given range
**rvmeta.readSkewByRange**

*Read skew by range from METAL-format files.*

**Description**

Read skew by range from METAL-format files.

**Usage**

```r
rvmeta.readSkewByRange(skewFile, tabixRange)
```

**Arguments**

- `skewFile`: character, a skew file (rvtests outputs using -meta skew)
- `tabixRange`: character, a text indicating which range in the VCF file to extract. e.g. 1:100-200

**Value**

an 3-dimensional array of skewness within given range

**See Also**

http://zhanxw.com/seqminer/ for online manual and examples

**Examples**

```r
skewFileName = system.file("rvtests/rvtest.MetaSkew.assoc.gz", package = "seqminer")
cfh <- rvmeta.readSkewByRange(skewFileName, "1:196621007-196716634")
```
**rvmeta.writeCovData**  
Write covariance association statistics files.

### Description
Write covariance association statistics files.

### Usage
```r
rmeta.writeCovData(rvmetaData, outName)
```

### Arguments
- **rvmetaData**: a list vector. It’s usually read by `rvmeta.readDataByRange` or `rvmeta.readDataByGene` function
- **outName**: character, a text indicating output file prefix

### Value
TRUE only if succeed

### See Also
http://zhanxw.com/seqminer/ for online manual and examples

### Examples
```r
scoreFileName = system.file("rvtests/rvtest.MetaScore.assoc.anno.gz", package = "seqminer")
covFileName = system.file("rvtests/rvtest.MetaCov.assoc.gz", package = "seqminer")
geneFile = system.file("vcf/refFlat_hg19_6col.txt.gz", package = "seqminer")
cfh <- rvmeta.readDataByRange(scoreFileName, covFileName, "1:19662007-196716634")

outFile <- file.path(tempdir(), "cfh.MetaCov.assoc.gz")
rmeta.writeCovData(cfh, outFile)
cat("Outputted MetaCov file are in the temp directory:", outFile, "\n")
```

**rvmeta.writeScoreData**  
Write score-based association statistics files.

### Description
Write score-based association statistics files.

### Usage
```r
rmeta.writeScoreData(rvmetaData, outName, createIndex = FALSE)
```

### Arguments
- **rvmetaData**: a list vector. It’s usually read by `rvmeta.readDataByRange` or `rvmeta.readDataByGene` function
- **outName**: character, a text indicating output file prefix
- **createIndex**: logical, if TRUE, an index file will be created

### Value
TRUE only if succeed

### See Also
http://zhanxw.com/seqminer/ for online manual and examples

### Examples
```r
scoreFileName = system.file("rvtests/rvtest.MetaScore.assoc.anno.gz", package = "seqminer")
covFileName = system.file("rvtests/rvtest.MetaCov.assoc.gz", package = "seqminer")
geneFile = system.file("vcf/refFlat_hg19_6col.txt.gz", package = "seqminer")
cfh <- rvmeta.readDataByRange(scoreFileName, covFileName, "1:19662007-196716634")

outFile <- file.path(tempdir(), "cfh.MetaCov.assoc.gz")
rmeta.writeScoreData(cfh, outFile, createIndex = TRUE)
cat("Outputted MetaCov file are in the temp directory:", outFile, "\n")
```
Arguments

rvmetaData a list vector. It’s usually read by rvmeta.readDataByRange or rvmeta.readDataByGene function
outName character, a text indicating output file prefix
createIndex boolean, (default FALSE), whether or not to create the index

Value

TRUE only if succeed

See Also

http://zhanxw.com/seqminer/ for online manual and examples

Examples

scoreFileName = system.file("rvtests/rvtest.MetaScore.assoc.anno.gz", package = "seqminer")
covFileName = system.file("rvtests/rvtest.MetaCov.assoc.gz", package = "seqminer")
geneFile = system.file("vcf/refFlat_hg19_6col.txt.gz", package = "seqminer")
cfh <- rvmeta.readDataByRange(scoreFileName, covFileName, "1:196621007-196716634")

outFile <- file.path(tempdir(), "cfh.MetaScore.assoc")
rvmeta.writeScoreData(cfh, outFile)
cat('Outputted MetaScore file are in the temp directory:', outFile, 'n')
tabix.createIndex

- Attractive Chaos <attractor@live.co.uk> (We have used the following software and made minimal necessary changes: Tabix, Heng Li <lh3@live.co.uk> (MIT license). We removed standard IO related functions, e.g. printf, fprintf; also changed its un-safe pointer arithmetics.) [copyright holder]
- Broad Institute / Massachusetts Institute of Technology [copyright holder]
- Genome Research Ltd (GRL) [copyright holder]
- Facebook, Inc [copyright holder]
- D. Richard Hipp [copyright holder]

See Also
Useful links:
- http://zhanxw.github.io/seqminer/
- Report bugs at https://github.com/zhanxw/seqminer/issues

---
tabix.createIndex  Create tabix index file, similar to running tabix in command line.

Description
Create tabix index file, similar to running tabix in command line.

Usage
```r
tabix.createIndex(
  bgzipFile,
  sequenceColumn = 1,
  startColumn = 4,
  endColumn = 5,
  metaChar = "#",
  skipLines = 0
)
```

Arguments
- `bgzipFile` character, an tabix indexed file
- `sequenceColumn` integer, sequence name column
- `startColumn` integer, start column
- `endColumn` integer, end column
- `metaChar` character, symbol for comment/meta lines
- `skipLines` integer, first this number of lines will be skipped

See Also
http://zhanxw.com/seqminer/ for online manual and examples
Examples

```r
define(fileName = system.file("vcf/all.anno.filtered.extract.vcf.gz", package = "seqminer"))
    tabix.createIndex(fileName, 1, 2, 0, 
                      quotesingle.Var
                      quotesingle.Var
                      
                      quotesingle.Var
                      quotesingle.Var
                      
                      quotesingle.Var
                      quotesingle.Var
                      
                      quotesingle.Var
                      quotesingle.Var
```

Description

Create tabix index for bgzipped MetaScore/MetaCov file

Usage

```r
tabix.createIndex.meta(bgzipFile)
```

Arguments

- `bgzipFile` character, input vcf file

See Also

- [http://zhanxw.com/seqminer/](http://zhanxw.com/seqminer/) for online manual and examples

Examples

```r
define(fileName = system.file("rvtests/rvtest.MetaScore.assoc.anno.gz", package = "seqminer"))
    tabix.createIndex.meta(fileName)
```

Description

Create tabix index for bgzipped VCF file

Usage

```r
tabix.createIndex.vcf(bgzipVcfFile)
```

Arguments

- `bgzipVcfFile` character, input vcf file
tabix.read

See Also

http://zhanxw.com/seqminer/ for online manual and examples

Examples

fileName = system.file("vcf/all.anno.filtered.extract.vcf.gz", package = "seqminer")
tabix.createIndex.vcf(fileName)

if (.Platform$endian == "little") {
  fileName = system.file("vcf/all.anno.filtered.extract.vcf.gz", package = "seqminer")
  snp <- tabix.read(fileName, "1:196623337-196632470")
} else {
  message("Tabix does not work well for big endian for now")
}
tabix.read.header  Read tabix file, similar to running tabix in command line.

Description
Read tabix file, similar to running tabix in command line.

Usage
tabix.read.header(tabixFile, skippedLine = FALSE)

Arguments
  tabixFile  character, an tabix indexed file
  skippedLine  logical, whether to read tabix skipped lines (when used 'tabix -S NUM')

Value
a list

See Also
http://zhanxw.com/seqminer/ for online manual and examples

Examples
fileName = system.file("vcf/all.anno.filtered.extract.vcf.gz", package = "seqminer")
snp <- tabix.read.header(fileName)

---

tabix.read.table  Read tabix file, similar to running tabix in command line.

Description
Read tabix file, similar to running tabix in command line.

Usage
tabix.read.table(
  tabixFile,
  tabixRange,
  col.names = TRUE,
  stringsAsFactors = FALSE
)

---

Usage
tabix.read.table(
  tabixFile,
  tabixRange,
  col.names = TRUE,
  stringsAsFactors = FALSE
)
validateAnnotationParameter

Arguments

- **tabixFile**: character, an tabix indexed file
- **tabixRange**: character, a text indicating which range in the VCF file to extract. e.g. 1:100-200
- **col.names**: logical, use tabix file header as result headers (default: TRUE)
- **stringsAsFactors**: logical, store loaded data as factors (default: FALSE)

Value

data frame, each elements is an individual line

See Also

http://zhanxw.com/seqminer/ for online manual and examples

Examples

```r
fileName = system.file("vcf/all.anno.filtered.extract.vcf.gz", package = "seqminer")
snp <- tabix.read.table(fileName, "1:196623337-196632470")
```

---

validateAnnotationParameter

*Validate annotate parameter is valid*

Description

Validate annotate parameter is valid

Usage

```r
validateAnnotationParameter(param, debug = FALSE)
```

Arguments

- **param**: a list of annotation elements
- **debug**: show extra debug information or not

Value

list, first element is TRUE/FALSE if parameter is valid/invalid;
verifyFilename

validate the inVcf can be created, and outVcf can be write to. will stop if any error occurs

Description
validate the inVcf can be created, and outVcf can be write to. will stop if any error occurs

Usage
verifyFilename(inVcf, outVcf)

Arguments
inVcf input file
outVcf output file

writeWorkflow Export workflow to Makefile

Description
Export workflow to Makefile

Usage
writeWorkflow(wf, outFile)

Arguments
wf a variable workflow class
outFile character, typically named "Makefile"

Examples
j1 <- newJob('id1', 'cmd out1', 'out1')
j2 <- newJob('id2', 'cmd out2', 'out2', depend = 'out1')
w <- newWorkflow("wf")
w <- addJob(w, j1)
w <- addJob(w, j2)

outFile <- file.path(tempdir(), "Makefile")
writeWorkflow(w, outFile)
cat('Outputted Makefile file are in the temp directory:', outFile, '\n')
Description

Read from binary PLINK file and return a genotype matrix

Usage

```r
## S3 method for class 'PlinkFile'
plinkFileObject[sampleIndex, markerIndex]
```

Arguments

- `plinkFileObject` a `PlinkFileObject` obtained by `openPlink()`
- `sampleIndex` integer, 1-based, index of samples to be extracted
- `markerIndex` integer, 1-based, index of markers to be extracted

Value

genotype matrix, marker by sample

See Also

- [http://zhanxw.com/seqminer/](http://zhanxw.com/seqminer/) for online manual and examples

Examples

```r
## these indices are nonsynonymous markers for 1:196621007-196716634",
## refer to the readVCFToMatrixByRange()
fileName = system.file("plink/all.anno.filtered.extract.bed", package = "seqminer")
filePrefix = sub(fileName, pattern = "\..bed\", replacement = "")
plinkObj = openPlink(filePrefix)
sampleIndex = seq(3)
markerIndex = c(14, 36)
cfh <- plinkObj[sampleIndex, markerIndex]
```
Index

[.PlinkFile, 34
addJob, 3
annotateGene, 4
annotatePlain, 5
annotateVcf, 5
createSingleChromosomeBCFIndex, 6
createSingleChromosomeVCFIndex, 7
download.annotation.resource, 7
ggetCovPair, 8
getRefBase, 8
isDirWritable, 9
isInRange, 9
isTabixRange, 10
makeAnnotationParameter, 10
newJob, 11
newWorkflow, 11
openPlink, 12
readBGENToListByGene, 12
readBGENToListByRange, 13
readBGENToMatrixByGene, 14
readBGENToMatrixByRange, 14
readPlinkToMatrixByIndex, 15
readSingleChromosomeBCFToMatrixByRange, 16
readSingleChromosomeVCFToMatrixByRange, 17
readVCFToListByGene, 17
readVCFToListByRange, 19
readVCFToMatrixByGene, 20
readVCFToMatrixByRange, 20
rvmeta.readCovByRange, 21
rvmata.readDataByGene, 22
rvmeta.readDataByRange, 23
rvmeta.readNullModel, 24
rvmeta.readScoreByRange, 24
rvmeta.readSkewByRange, 25
rvmeta.writeCovData, 26
rvmeta.writeScoreData, 26
SEQMINER, 27
seqminer (SEQMINER), 27
seqminer-package (SEQMINER), 27
tabix.createIndex, 28
tabix.createIndex.meta, 29
tabix.createIndex.vcf, 29
tabix.read, 30
tabix.read.header, 31
tabix.read.table, 31
validateAnnotationParameter, 32
verifyFilename, 33
writeWorkflow, 33