Package ‘seqminer’

November 9, 2021

Type  Package
Title  Efficiently Read Sequence Data (VCF Format, BCF Format, METAL Format and BGEN Format) into R
Version  8.2
Date  2021-11-7
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 (MIT 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++11, zlib headers and libraries, GNU make, optionally also bzip2 and POSIX-compliant regex functions.
NeedsCompilation  yes
RoxygenNote  7.1.2
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]

**Date/Publication** 2021-11-09 10:20:02 UTC

### R topics documented:

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>addJob</td>
<td>3</td>
</tr>
<tr>
<td>annotateGene</td>
<td>4</td>
</tr>
<tr>
<td>annotatePlain</td>
<td>5</td>
</tr>
<tr>
<td>annotateVcf</td>
<td>5</td>
</tr>
<tr>
<td>createSingleChromosomeBCFIndex</td>
<td>6</td>
</tr>
<tr>
<td>createSingleChromosomeVCFIndex</td>
<td>7</td>
</tr>
<tr>
<td>download.annotation.resource</td>
<td>7</td>
</tr>
<tr>
<td>getCovPair</td>
<td>8</td>
</tr>
<tr>
<td>getRefBase</td>
<td>8</td>
</tr>
<tr>
<td>isDirWritable</td>
<td>9</td>
</tr>
<tr>
<td>isInRange</td>
<td>9</td>
</tr>
<tr>
<td>isTabixRange</td>
<td>10</td>
</tr>
<tr>
<td>makeAnnotationParameter</td>
<td>10</td>
</tr>
<tr>
<td>newJob</td>
<td>11</td>
</tr>
<tr>
<td>newWorkflow</td>
<td>11</td>
</tr>
<tr>
<td>openPlink</td>
<td>12</td>
</tr>
<tr>
<td>readBGENToListByGene</td>
<td>12</td>
</tr>
<tr>
<td>readBGENToListByRange</td>
<td>13</td>
</tr>
<tr>
<td>readBGENToMatrixByGene</td>
<td>14</td>
</tr>
<tr>
<td>readBGENToMatrixByRange</td>
<td>14</td>
</tr>
<tr>
<td>readPlinkToMatrixByIndex</td>
<td>15</td>
</tr>
<tr>
<td>readSingleChromosomeBCFToMatrixByRange</td>
<td>16</td>
</tr>
<tr>
<td>readSingleChromosomeVCFToMatrixByRange</td>
<td>17</td>
</tr>
<tr>
<td>readVCFToListByGene</td>
<td>17</td>
</tr>
<tr>
<td>readVCFToListByRange</td>
<td>19</td>
</tr>
<tr>
<td>readVCFToMatrixByGene</td>
<td>20</td>
</tr>
<tr>
<td>readVCFToMatrixByRange</td>
<td>20</td>
</tr>
<tr>
<td>rvmeta.readCovByRange</td>
<td>21</td>
</tr>
<tr>
<td>rvmeta.readDataByGene</td>
<td>22</td>
</tr>
<tr>
<td>rvmeta.readDataByRange</td>
<td>23</td>
</tr>
<tr>
<td>rvmeta.readNullModel</td>
<td>24</td>
</tr>
<tr>
<td>rvmeta.readScoreByRange</td>
<td>24</td>
</tr>
<tr>
<td>rvmeta.readSkewByRange</td>
<td>25</td>
</tr>
<tr>
<td>rvmeta.writeCovData</td>
<td>26</td>
</tr>
<tr>
<td>rvmeta.writeScoreData</td>
<td>26</td>
</tr>
<tr>
<td>SeqMiner</td>
<td>27</td>
</tr>
</tbody>
</table>
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, "\n")
```

---

**annotateVcf**

Annotate a VCF file

**Description**

Annotate a VCF file

**Usage**

annotateVcf(inVcf, outVcf, params)
createSingleChromosomeBCFIndex

Arguments

inVcf input VCF file name
outVcf output VCF file name
params parameters

Value

0 if succeed

Examples

param <- list(reference = system.file("tabanno/test.fa", package = "seqminer"),
geneFile = system.file("tabanno/test.gene.txt", package = "seqminer"))
param <- makeAnnotationParameter(param)
inVcf <- system.file("tabanno/input.test.vcf", package = "seqminer")
outVcf <- file.path(tempdir(), "/", "out.vcf")
annotateVcf (inVcf, outVcf, param)
cat('Annotated VCF files are in the temp directory:', outVcf, '"

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

Value

will not return anything

Examples

```r
## Not run:
download.annotation.resource("/tmp")

## End(Not run)
```

covFile = system.file("rvtests/rvtest.MetaCovassoc.gz", package = "seqminer")
cfh <- rvmeta.readCovByRange(covFile, "1:196621007-196716634")
rangeList1 <- "1:196621007-196700000"
rangeList2 <- "1:196700000-196716634"
getCovPair(cfh, rangeList1, rangeList2)

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

```r
covFile = system.file("rvtests/rvtest.MetaCovassoc.gz", package = "seqminer")
cfh <- rvmeta.readCovByRange(covFile, "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

Description

Test whether directory is writable

Usage

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

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

```r
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

```r
isTabixRange(range)
```

Arguments

- `range` character vector

Examples

```r
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))
```

makeAnnotationParameter

Construct a usable set of annotation parameters

Description

Construct a usable set of annotation parameters

Usage

```r
makeAnnotationParameter(param = NULL)
```

Arguments

- `param` a list of annotation elements
\textit{newJob} \hfill 11

\textbf{Value}

list, a complete list of supported parameters

\begin{itemize}
\item \texttt{newJob} Create a \textit{new job}
\end{itemize}

\textbf{Description}

Create a new job

\textbf{Usage}

\texttt{newJob(id, cmd, outFile, depend = NULL)}

\textbf{Arguments}

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

\textbf{Examples}

\begin{verbatim}
j1 <- newJob('id1', 'cmd out1', 'out1')
j2 <- newJob('id2', 'cmd out2', 'out2', depend = 'out1')
\end{verbatim}

\begin{itemize}
\item \texttt{newWorkflow} Create a \textit{new workflow}
\end{itemize}

\textbf{Description}

Create a new workflow

\textbf{Usage}

\texttt{newWorkflow(name)}

\textbf{Arguments}

- \texttt{name} character, specify the name of the workflow

\textbf{Examples}

\begin{verbatim}
w <- newWorkflow("wf")
\end{verbatim}
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
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")

readBGENToListByRange  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

readBGENToListByRange(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

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”,
## refer to the readVCFToMatrixByRange()
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 is `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")
```

---

**Description**

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

**Usage**

```r
readVCFToListByGene
```

**Description**

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

Usage

readVCToListByGene(
  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 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 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 <- readVCToListByGene(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 elements 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)
Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fileName</td>
<td>character, represents an input VCF file (Bgzipped, with Tabix index)</td>
</tr>
<tr>
<td>range</td>
<td>character, a text indicating which range in the VCF file to extract. e.g. 1:100-200, 1-based index</td>
</tr>
<tr>
<td>annoType</td>
<td>character, annotated types you would like to extract, such as &quot;Nonsynonymous&quot;, &quot;Synonymous&quot;. This can be left empty.</td>
</tr>
</tbody>
</table>

Value

genotype matrix

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")
```

Description

Read covariance by range from METAL-format files.

Usage

rvmeta.readCovByRange(covFile, tabixRange)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>covFile</td>
<td>character, a covariance file (rvtests outputs using –meta cov)</td>
</tr>
<tr>
<td>tabixRange</td>
<td>character, a text indicating which range in the VCF file to extract. e.g. 1:100-200</td>
</tr>
</tbody>
</table>

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,  # 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 = 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

```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, "1:196621007-196716634")
```
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

See Also

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

Examples

scoreFileName = system.file("rvtests/rvtest.MetaScore.assoc.anno.gz", package = "seqminer")
cfh <- rvmeta.readScoreByRange(scoreFileName, "1:196621007-196716634")

skewFileName = system.file("rvtests/rvtest.MetaSkew.assoc.gz", package = "seqminer")
cfh <- rvmeta.readSkewByRange(skewFileName, "1:196621007-196716634")

rvmeta.readSkewByRange

Read skew by range from METAL-format files.

Description

Read skew by range from METAL-format files.

Usage

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

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

outFile <- file.path(tempdir(), "cfh.MetaCov.assoc.gz")
rvmeta.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
rvmeta.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  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, '
")

SeqMiner

Efficiently Read Sequencing Data (VCF format, METAL format) into R

Description

SeqMiner provides functions to easily load Variant Call Format (VCF) or METAL format into R

Details

The aim of this package is to save your time parsing large text file. That means data processing time can be saved for other researches. This packages requires Bgzip compressed and Tabix indexed files as input. If input files containaions annotation by TabAnno (), it is possible to extract information at the unit of genes.
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
fileName = system.file("vcf/all.anno.filtered.extract.vcf.gz", package = "seqminer")
tabix.createIndex(fileName, 1, 2, 0, '#', 0)
```
tabix.createIndex.meta

Create tabix index for bgzipped MetaScore/MetaCov file

Description
Create tabix index for bgzipped MetaScore/MetaCov file

Usage

```
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

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

---

tabix.createIndex.vcf

Create tabix index for bgzipped VCF file

Description
Create tabix index for bgzipped VCF file

Usage

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

Arguments

- `bgzipVcfFile` character, input vcf file

See Also

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

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

---

**tabix.read**

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

---

**Description**

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

**Usage**

```r
tabix.read(tabixFile, tabixRange)
```

**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

**Value**

character vector, each elements is an individual line

**See Also**

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

**Examples**

```r
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

```r
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

```r
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

```r
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

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

validateAnnotationParameter(param, debug = FALSE)

Validate annotate parameter is valid

Description

Validate annotate parameter is valid

Usage

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**

```r
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

## 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/ for online manual and examples

Examples

## 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
getcovPair, 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
rvmeta.readDataByGene, 22
rvmeta.readDataByRange, 23
rvmeta.readNullModel, 24
rvmeta.readScoreByRange, 24
rvmeta.readSkewByRange, 25
rvmeta.writeCovData, 26
rvmeta.writeScoreData, 26
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