Package ‘seqminer’

August 2, 2018

Type      Package
Title     Efficiently Read Sequence Data (VCF Format, BCF Format and METAL Format) into R
Version   6.1
Date      2018-08-01
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 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). We removed standard IO related functions, e.g. printf, fprintf; also changed its un-safe pointer arithmetics.

License   GPL file LICENSE

URL       http://seqminer.genomic.codes

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

Repository CRAN

Suggests testthat, SKAT

SystemRequirements zlib headers and libraries, optionally also bzip2 and POSIX-compliant regex functions.

NeedsCompilation yes

RoxygenNote 6.0.1

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]

Date/Publication 2018-08-02 17:20:03 UTC
addJob

### Description

Add a job to a workflow
**annotateGene**

**Usage**

```r
addJob(wf, job)
```

**Arguments**

- `wf` a variable of workflow class
- `job` a variable of job class

**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)
writeWorkflow(w, "Makefile")
```

---

**annotateGene**

Annotate a test variant

**Description**

Annotate a test variant

**Usage**

```r
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`
annotatePlain

Annotate a plain text file

Description
Annotate a plain text file

Usage
annotatePlain(inFile, outFile, params)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>inFile</td>
<td>input file name</td>
</tr>
<tr>
<td>outFile</td>
<td>output file name</td>
</tr>
<tr>
<td>params</td>
<td>parameters</td>
</tr>
</tbody>
</table>

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)
print(param)
annotateGene(param, c("1", "1"), c(3, 5), c("A", "C"), c("G", "C"))

annotatePlain(inFile, outFile, params)
```
annotateVcf

Annotate a VCF file

Description
Annotate a VCF file

Usage
annotateVcf(inVcf, outVcf, params)

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 <- paste0(getwd(), "/", "out.vcf")
annotateVcf (inVcf, outVcf, param)

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
Value

will not return anything

Examples

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

---

**getCovPair**

*Extract pair of positions by ranges*

---

**Description**

Extract pair of positions by ranges

**Usage**

```r
getCovPair(covData, rangeList1, rangeList2)
```

**Arguments**

- `covData`: a covariance matrix with positions as dimnames
- `rangeList1`: character specify a range
- `rangeList2`: character specify a range

**Value**

a covariance matrix

```r
covFileName = system.file("rvtests/rvtest.MetaCov.assoc.gz", package = "seqminer")
cfh <- rvmeta.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**

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

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>reference</td>
<td>path to the reference genome file (.fa file)</td>
</tr>
<tr>
<td>chrom</td>
<td>a vector of chromosome names</td>
</tr>
<tr>
<td>position</td>
<td>a vector of chromosome positions</td>
</tr>
<tr>
<td>len</td>
<td>a vector of length</td>
</tr>
</tbody>
</table>

**Value**

based extracted from the reference genome

**isDirWritable**  
Test whether directory is writable

**Description**

Test whether directory is writable

**Usage**

```r
isDirWritable(outDir)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>outDir</td>
<td>the name of the directory</td>
</tr>
</tbody>
</table>

**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**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>positions</td>
<td>characters, positions. e.g. c(&quot;1:2-3&quot;, &quot;1:4&quot;)</td>
</tr>
<tr>
<td>rangeList</td>
<td>character, ranges, e.g. &quot;1:1-3,1:2-4&quot;</td>
</tr>
</tbody>
</table>
makeAnnotationParameter

Description

Construct a usable set of annotation parameters

Usage

makeAnnotationParameter(param = NULL)

Arguments

param a list of annotation elements

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))
**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")
```
readVCFToListByGene  

Description

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

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 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 <- 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**
```
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
- **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 extarct. 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)`
rvmeta.readCovByRange

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
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

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

rvmeta.readCovByRange Read covariance by range from METAL-format files.

Description

Read covariance by range from METAL-format files.

Usage

rvmeta.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

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, covaraitve 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
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

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")

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

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.anne.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")
rmeta.writeScoreData(cfh, "cfh.MetaScore.assoc")

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 contain 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
```
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
```
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/ for online manual and examples
http://zhanxw.github.io/rvtests/ for rvtests

Examples

```r
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**

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

**Arguments**

- `bgzipVcfFile`: character, input vcf file

**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.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)
```
tabix.read.header

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")
}
```

---

**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

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

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

Validate annotate parameter is valid

Description

Validate annotate parameter is valid

Usage

  validateAnnotationParameter(param, debug = FALSE)
writeWorkflow

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)
writeWorkflow(w, "Makefile")
Index

addJob, 2
annotateGene, 3
annotatePlain, 4
annotateVcf, 5

download.annotation.resource, 5
getCovPair, 6
getRefBase, 6

isDirWritable, 7
isInRange, 7
isTabixRange, 8

makeAnnotationParameter, 8

newJob, 9
newWorkflow, 9

readVCFToListByGene, 10
readVCFToListByRange, 11
readVCFToMatrixByGene, 12
readVCFToMatrixByRange, 12
rvmeta.readCovByRange, 13
rvmeta.readDataByGene, 14
rvmeta.readDataByRange, 15
rvmeta.readNullModel, 16
rvmeta.readScoreByRange, 16
rvmeta.readSkewByRange, 17
rvmeta.writeCovData, 18
rvmeta.writeScoreData, 18

SeqMiner, 19
SeqMiner-package (SeqMiner), 19

tabix.createIndex, 20
tabix.createIndex.meta, 20
tabix.createIndex.vcf, 21
tabix.read, 21
tabix.read.header, 22
tabix.read.table, 23

validateAnnotationParameter, 23
verifyFilename, 24
writeWorkflow, 24