Package ‘HTSeqGenie’

March 14, 2024

Imports BiocGenerics (>= 0.2.0), S4Vectors (>= 0.9.25), IRanges (>= 1.21.39), GenomicRanges (>= 1.23.21), Rsamtools (>= 1.8.5), Biostrings (>= 2.24.1), chipseq (>= 1.6.1), hwriter (>= 1.3.0), Cairo (>= 1.5.5), GenomicFeatures (>= 1.9.31), BiocParallel, parallel, tools, rtracklayer (>= 1.17.19), GenomicAlignments, VariantTools (>= 1.7.7), GenomeInfoDb, SummarizedExperiment, methods

Maintainer  Jens Reeder <reeder.jens@gene.com>
License      Artistic-2.0
Title        A NGS analysis pipeline.
Type         Package
LazyLoad      yes
Author        Gregoire Pau, Jens Reeder
Description   Libraries to perform NGS analysis.
Version       4.32.0
Depends       R (>= 3.0.0), gmapR (>= 1.8.0), ShortRead (>= 1.19.13), VariantAnnotation (>= 1.8.3)
Suggests      TxDb.Hsapiens.UCSC.hg19.knownGene, LungCancerLines, org.Hs.eg.db, RUnit
RoxygenNote   5.0.1

git_url https://git.bioconductor.org/packages/HTSeqGenie
git_branch  RELEASE_3_18
git_last_commit  9a84f3d
git_last_commit_date  2023-10-24
Repository    Bioconductor 3.18
Date/Publication  2024-03-13
## R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>alignReads</td>
<td>4</td>
</tr>
<tr>
<td>alignReadsChunk</td>
<td>5</td>
</tr>
<tr>
<td>analyzeVariants</td>
<td>6</td>
</tr>
<tr>
<td>annotateVariants</td>
<td>6</td>
</tr>
<tr>
<td>bamCountUniqueReads</td>
<td>7</td>
</tr>
<tr>
<td>buildConfig</td>
<td>7</td>
</tr>
<tr>
<td>buildGenomicFeaturesFromTxDb</td>
<td>8</td>
</tr>
<tr>
<td>buildShortReadReports</td>
<td>9</td>
</tr>
<tr>
<td>buildTallyParam</td>
<td>9</td>
</tr>
<tr>
<td>buildTP53FastaGenome</td>
<td>10</td>
</tr>
<tr>
<td>buildTP53GenomeTemplate</td>
<td>10</td>
</tr>
<tr>
<td>calculateCoverage</td>
<td>11</td>
</tr>
<tr>
<td>calculateTargetLengths</td>
<td>11</td>
</tr>
<tr>
<td>callVariantsGATK</td>
<td>12</td>
</tr>
<tr>
<td>checkConfig</td>
<td>12</td>
</tr>
<tr>
<td>checkGATKJar</td>
<td>13</td>
</tr>
<tr>
<td>checkPicardJar</td>
<td>13</td>
</tr>
<tr>
<td>computeBamStats</td>
<td>14</td>
</tr>
<tr>
<td>computeCoverage</td>
<td>14</td>
</tr>
<tr>
<td>countFeatures</td>
<td>15</td>
</tr>
<tr>
<td>countGenomicFeatures</td>
<td>16</td>
</tr>
<tr>
<td>countGenomicFeaturesChunk</td>
<td>16</td>
</tr>
<tr>
<td>createTmpDir</td>
<td>17</td>
</tr>
<tr>
<td>detectAdapterContam</td>
<td>17</td>
</tr>
<tr>
<td>detectQualityInFASTQFile</td>
<td>18</td>
</tr>
<tr>
<td>detectRRNA</td>
<td>18</td>
</tr>
<tr>
<td>excludeVariantsByRegions</td>
<td>19</td>
</tr>
<tr>
<td>FastQStreamer.getReads</td>
<td>20</td>
</tr>
<tr>
<td>FastQStreamer.init</td>
<td>20</td>
</tr>
<tr>
<td>FastQStreamer.release</td>
<td>21</td>
</tr>
<tr>
<td>filterByLength</td>
<td>22</td>
</tr>
<tr>
<td>filterQuality</td>
<td>22</td>
</tr>
<tr>
<td>findVariantFile</td>
<td>23</td>
</tr>
<tr>
<td>gatk</td>
<td>23</td>
</tr>
<tr>
<td>generateSingleGeneDERs</td>
<td>24</td>
</tr>
<tr>
<td>getAdapterSeqs</td>
<td>25</td>
</tr>
<tr>
<td>getBams</td>
<td>25</td>
</tr>
<tr>
<td>getChunkDirs</td>
<td>26</td>
</tr>
<tr>
<td>getConfig</td>
<td>26</td>
</tr>
<tr>
<td>getConfig.integer</td>
<td>27</td>
</tr>
<tr>
<td>getConfig.logical</td>
<td>27</td>
</tr>
<tr>
<td>getConfig.numeric</td>
<td>28</td>
</tr>
<tr>
<td>getConfig.vector</td>
<td>28</td>
</tr>
<tr>
<td>getEndNumber</td>
<td>29</td>
</tr>
<tr>
<td>getMemoryUsage</td>
<td>29</td>
</tr>
<tr>
<td>getNumberOfReadsInFASTQFile</td>
<td>30</td>
</tr>
</tbody>
</table>
R topics documented:

getNumericVectorDataFromFile ........................................ 30
getObjectFilename ....................................................... 31
getPackageFile ........................................................... 31
getRandomAlignCutoff ..................................................... 32
getRRNAIds ............................................................... 32
getTabDataFromFile ....................................................... 33
getTraceback .............................................................. 33
hashCoverage .............................................................. 34
hashVariants ............................................................... 34
hashVector ................................................................. 35
HTSeqGenie ................................................................. 35
initDirs ..................................................................... 36
initLog ...................................................................... 37
initLogger ................................................................. 37
initPipelineFromConfig .................................................. 38
initPipelineFromSaveDir ............................................... 38
isAboveQualityThresh .................................................... 39
isAdapter ................................................................. 39
isConfig ................................................................. 40
isFirstFragment .......................................................... 40
isSparse ................................................................. 41
listIterator.init .......................................................... 42
listIterator.next ......................................................... 42
loadConfig ............................................................... 43
logdebug ................................................................. 43
logerror ................................................................. 44
loginfo ................................................................. 44
logwarn ................................................................. 45
makeDir ................................................................. 45
makeRandomSreads ......................................................... 46
markDuplicates ............................................................ 46
markDups ................................................................. 47
mergeAlignReads .......................................................... 47
mergeCoverage ........................................................... 48
mergeLanes ............................................................... 49
mergePreprocessReads ................................................... 49
mergeSummaryAlignment .................................................. 50
parseDCF ................................................................. 51
parseSummaries ........................................................... 51
picard ................................................................. 52
plotDF ................................................................. 52
preprocessReads .......................................................... 53
preprocessReadsChunk ................................................... 53
processChunks ............................................................ 54
readInputFiles ............................................................ 55
readRNASEqEnds .......................................................... 55
realignIndels ............................................................. 56
realignIndelsGATK ........................................................ 56
alignReads

Align reads against genome

Align reads against genome
alignReadsChunk

Usage
alignReads()

Value
Nothing

Author(s)
Gregoire Pau

alignReadsChunk  Genomic alignment

Description
Genomic alignment using gsnap.

Usage
alignReadsChunk(fp1, fp2 = NULL, save_dir = NULL)

Arguments
fp1          Path to FastQ file
fp2          Path to second FastQ file if paired end data, NULL if single ended
save_dir     Save directory

Details
Aligns reads in fp1 and fp2 to genome specified via global config variable alignReads.genome. Gsnap output is converted into BAM files and sorted + indexed.

Value
List of alignment files in BAM format
analyzeVariants  Calculate and process Variants

Description
Calculate and process Variants

Usage
analyzeVariants()

Value
Nothing

Author(s)
Jens Reeder

annotateVariants  Annotate variants via vep

Description
Annotate variants via vep

Usage
annotateVariants(vcf.file)

Arguments
vcf.file  A character vector pointing to a VCF (or gzipped VCF) file

Value
Path to a vcf file with variant annotations

Author(s)
Jens Reeder
bamCountUniqueReads

Uniquely count number of reads in bam file

Description
Uniquely count number of reads in bam file

Usage
bamCountUniqueReads(bam)

Arguments
bam Name of bam file

Value
number of reads

Author(s)
Jens Reeder

buildConfig
Build a configuration file based on a list of parameters

Description
Build a configuration file based on a list of parameters

Usage
buildConfig(config_filename, ...)

Arguments
config_filename The path of a configuration filename.
... A list of named value pairs.

Value
Nothing.

Author(s)
Gregoire Pau
buildGenomicFeaturesFromTxDb

Build genomic features from a TxDb object

Description

Build genomic features from a TxDb object

Usage

buildGenomicFeaturesFromTxDb(txdb)

Arguments

txdb

A TxDb object.

Value

A list named list of GRanges objects containing the biological entities to account for.

Author(s)

Gregoire Pau

Examples

```r
## Not run:
library("TxDb.Hsapiens.UCSC.hg19.knownGene")
txdb <- TxDb.Hsapiens.UCSC.hg19.knownGene
genomic_features <- buildGenomicFeaturesFromTxDb(txdb)
## End(Not run)
```
**buildShortReadReports**  
*Build a ShortRead report*

---

**Description**

Build a ShortRead report

**Usage**

buildShortReadReports(save_dir, paired_ends)

**Arguments**

- **save_dir**: Save directory of a pipeline run
- **paired_ends**: A logical, indicating whether reads are paired

**Value**

Nothing

**Author(s)**

Gregoire Pau

---

**buildTallyParam**  
*Build tally parameters*

---

**Description**

Build tally parameters

**Usage**

buildTallyParam()

**Value**

- a VariantTallyParam object

**Author(s)**

Gregoire Pau
buildTP53FastaGenome

Description
create fasta genome file of TP53 genome

Usage
buildTP53FastaGenome()

Value
Path to tp53 genome directory

Author(s)
Jens Reeder

buildTP53GenomeTemplate

Description
Create a tp53 config template

Usage
buildTP53GenomeTemplate()

Value
Path to tp53 template file

Author(s)
Jens Reeder
**calculateCoverage**

*Calculate read coverage*

**Description**
Calculate read coverage

**Usage**
calculateCoverage()

**Value**
Nothing

**Author(s)**
Jens Reeder

---

**calculateTargetLengths**

*Plot target length for paired end*

**Description**
Calculate and plot a histogram of mapped target lengths after trimming of trim/2 of the data points at the lower and upper end of the distribution

**Usage**
calculateTargetLengths(bamfile, save_dir, trim = 0.4)

**Arguments**

- **bamfile**
  Path to a bam file
- **save_dir**
  Path to a pipeline results dir
- **trim**
  Amount of data to be trimmed at the edges

**Value**
Target length table and writes two files in save_dir/reports/images/TargetLengths.[pdf|png]

**Author(s)**
Jens Reeder, Melanie Huntley
callVariantsGATK  
*Variant calling via GATK*

**Description**

Call variants via GATK using the pipeline framework. Requires a GATK compatible genome with a name matching the alignment genome to be installed in 'path.gatk_genome'

**Usage**

callVariantsGATK(bam.file)

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bam.file</td>
<td>Path to bam.file</td>
</tr>
</tbody>
</table>

**Value**

Path to variant file

**Author(s)**

Jens Reeder

checkConfig  
*Check configuration*

**Description**

Performs all configuration checks

**Usage**

checkConfig()

**Value**

Nothing. Individual checks will throw error instead.
**checkGATKJar**

*Check for the GATK jar file*

---

**Description**

Check for the GATK jar file

**Usage**

`checkGATKJar(path = getOption("gatk.path"))`

**Arguments**

- **path**
  
  Path to the GATK jar file

**Value**

TRUE if tool can be called, FALSE otherwise

---

**checkPicardJar**

*checkPicardJar*

---

**Description**

Check for a jar file from picard tools

**Usage**

`checkPicardJar(toolname, path = getOption("picard.path"))`

**Arguments**

- **toolname**
  
  Name of the Picard Tool, e.g. MarkDuplicates

- **path**
  
  Path to folder containing picard jars

**Details**

Call a tool from picard and see if it responds.

**Value**

TRUE if tool can be called, FALSE otherwise

**Author(s)**

Jens Reeder
computeBamStats  

Compute record statistics from a bam file

Description

Compute record statistics from a bam file

Usage

computeBamStats(bam)

Arguments

bam  
A character string containing an existing bam file

Details

The statistics are additive over chunks/lanes.

Value

A numeric vector

Author(s)

Gregoire Pau

computeCoverage  

Compute the coverage vector given a bamfile

Description

Compute the coverage vector given a bamfile

Usage

computeCoverage(bamfile, extendReads = FALSE, paired_ends = FALSE,  
fragmentLength = NULL, maxFragmentLength = NULL)
countFeatures

**Arguments**

- `bamfile` A character string indicating the path of bam file
- `extendReads` A logical, indicating whether reads should be extended
- `paired_ends` A logical, indicating whether reads are paired
- `fragmentLength` An integer, indicating the new size of reads when `extendReads` is TRUE and `paired_ends` is FALSE. If NULL, read size is estimated using `estimate.mean.fraglen` from the chipseq package.
- `maxFragmentLength` An optional integer, specifying the maximal size of fragments. Longer fragments will be disregarded when computing coverage.

**Value**

A SimpleRleList object containing the coverage

**Author(s)**

Gregoire Pau

---

**countFeatures** Count RNA-Seq Pipeline Genomic Features

**Description**

Given GRanges, counts number of hits by gene, exon, intergenic, etc

**Usage**

`countFeatures(reads, features)`

**Arguments**

- `reads` GRangesList object of interval, usually where reads aligned
- `features` A list of genome annotations as GRangesList

**Details**

Given a GRanges object, this function performs an overlap against a previously created set of genomic regions. These genomic regions include genes, coding portions of genes (CDS), exons, intergenic regions, and exon groups (which contain two or more exons)

**Value**

A list of counts by feature

**Author(s)**

Cory Barr
countGenomicFeatures

**Description**

Count overlaps with genomic features

**Usage**

```python
countGenomicFeatures()
```

**Value**

Nothing

**Author(s)**

Gegoire Pau

---

countGenomicFeaturesChunk

**Description**

Count reads by genomic Feature

**Usage**

```python
countGenomicFeaturesChunk(save_dir, genomic_features)
```

**Arguments**

- `save_dir`: Path to a pipeline run’s save dir
- `genomic_features`: A list of genomic features to tally

**Details**

given a BAM-file output from gsnap (with the MD tag), count hits to exons, genes, ncRNAs, etc. and quantify miRNA/ncRNA contaminatino

**Value**

Nothing
createTmpDir

Author(s)
Cory Barr

createTmpDir  Create a random directory with prefix in R temp dir

Description
Especially for testing code it is very helpful to have a temp directory with a defined prefix, so one knows which test produced which directory.

Usage
createTmpDir(prefix = NULL, dir = tempdir())

Arguments
prefix A string that will precede the directory name
dir Directory where the random dir will be created under. Defaults to tempdir()

Value
Name of temporary directory

detectAdapterContam  Detect sequencing adapter contamination

Description
For each read or pair of read, search for specific Illumina adapter sequences in the read. Flag if at least one read has significant overlap with adapter.

Usage
detectAdapterContam(lreads, save_dir = NULL)

Arguments
lreads List of reads as ShortRead objects
save_dir Save directory of a pipeline run

Value
Boolean vector indicating vector contamination for each read
detectQualityInFASTQFile

Detect quality protocol from a FASTQ file

**Description**

Detect quality protocol from a FASTQ file

**Usage**

detectQualityInFASTQFile(filename, nreads = 5000)

**Arguments**

- **filename**
  Path to a FASTQ or gzipped-FASTQ file
- **nreads**
  Number of reads to test quality on. Default is 5000.

**Value**

A character vector containing the compatible qualities. NULL if none.

**Author(s)**

Jens Reeder

detectRRNA

Detect rRNA Contamination in Reads

**Description**

Returns a named vector indicating if a read ID has rRNA contamination or not

**Usage**

detectRRNA(lreads, remove_tmp_dir = TRUE, save_dir = NULL)

**Arguments**

- **lreads**
  A list of ShortReadQ objects
- **remove_tmp_dir**
  boolean indicating whether or not to delete temp directory of gsnap results
- **save_dir**
  Save directory

**Details**

Given a genome and fastq data, each read in the fastq data is aligned against the rRNA sequences for that genome
excludeVariantsByRegions

Value

  a named logical vector indicating if a read has rRNA contamination

Author(s)

  Cory Barr

Description

  Filter variants by regions

Usage

  excludeVariantsByRegions(variants, mask)

Arguments

  variants  Variants as Vranges, GRanges or VCF object
  mask      region to mask, given as GRanges

Details

  This function can be used to filter variants in a given region, e.g. low complexity and repeat regions

Value

  The filtered variants

Author(s)

  Jens Reeder
FastQStreamer.getReads

Get FastQ reads from the FastQ streamer

Description

Get FastQ reads from the FastQ streamer

Usage

FastQStreamer.getReads()

Value

A list of ShortRead object containing reads. NULL if there are no more reads to read.

Author(s)

Gregoire Pau

See Also

FastQStreamer.init

FastQStreamer.init

Open a streaming connection to a FastQ file

Description

Open a streaming connection to a FastQ file

Usage

FastQStreamer.init(input_file, input_file2 = NULL, chunk_size,
subsample_nbreads = NULL, max_nbchunks = NULL)

Arguments

input_file Path to a FastQ file
input_file2 Optional path to a FastQ file. Default is NULL.
chunk_size Number of reads per chunk
subsample_nbreads Optional number of reads to subsample (deterministic) from the input files. Default is NULL.
max_nbchunks Optional maximal number of chunks to read
FastQStreamer.release

Details

Only one FastQStreamer object can be open at any time.

Value

Nothing.

Author(s)

Gregoire Pau

See Also

FastQStreamer.getReads

---

FastQStreamer.release  Close the FastQStreamer

Description

Close the FastQStreamer

Usage

FastQStreamer.release()

Value

Nothing

Author(s)

Gregoire Pau

See Also

FastQStreamer.init
### filterByLength
*Filter reads by length*

**Description**
Checks whether reads have at least a length of `minlength`. Useful values are zero to rid of empty reads or 12 to match the gsnap k-mer size.

**Usage**

```r
filterByLength(lreads, minlength = 12, paired = FALSE)
```

**Arguments**
- `lreads` A set of reads as ShortReadQ object
- `minlength` Minimum length
- `paired` Indicates whether `lreads` has one of two elements

**Value**
A boolean vector indicating whether read passes filter

### filterQuality
*Filter reads by quality*

**Description**
Filtering reads by quality score. Discards reads that have more than a fraction of X nucleotides with a score below Y.

**Usage**

```r
filterQuality(lreads)
```

**Arguments**
- `lreads` A list of ShortReadQ objects

**Details**
X and Y are controlled by global config variables X: `filterQuality.minFrac` Y: `filterQuality.minQuality`

**Value**
A list of quality filtered ShortReadQ objects
**findVariantFile**

*Get a vcf filename given a HTSeqGenie directory*

**Description**

Get the filename of the variant file

**Usage**

```r
findVariantFile(save_dir)
```

**Arguments**

- `dir_path` A character string containing a dir path

**Details**

Depending on the variant caller used and the version of VariantAnotation used to create the file a file might have the ending vcf.gz, vcf.bgz. To function hides all this mess.

**Value**

A character vector containing an existing filename, stops if 0 or more than 1

**Author(s)**

Jens Reeder

---

**gatk**

*Run a command from the GATK*

**Usage**

```r
gatk(gatk.jar.path = getOption("gatk.path"), method, args, maxheap = "4g")
```

**Arguments**

- `gatk.jar.path` Path to the gatk jar file
- `method` Name of the gatk method, e.g. UnifiedGenotyper
- `args` additional args passed to gatk
- `maxheap` Maximal heap space allocated for java, GATK recommends 4G heap for most of its apps
generateSingleGeneDERs

Details

Execute the GATK jar file using the method specified as arg. Stops if the command executed fails.

Value

0 for success, stops otherwise

Author(s)

Jens Reeder

Description

Generate DEXSeq-ready exons

Usage

generateSingleGeneDERs(txdb)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>txdb</td>
<td>A transcript DB object</td>
</tr>
</tbody>
</table>

Details

generateSingleGeneDERs() generates exons by: 1) disjoining the whole exon set 2) keeping only the exons of coding regions 3) keeping only the exons that belong to unique genes

Value

single gene DERs
**getAdapterSeqs**

Read list of Illumina adapter seqs from package data

**Usage**

\[
\text{getAdapterSeqs}(\text{paired}\_\text{ends}, \text{force}\_\text{paired}\_\text{end}\_\text{adapter}, \text{pair}\_\text{num} = 1)
\]

**Arguments**

- `paired_ends` : Do we have paired ends reads?
- `force_paired_end_adapter` : Force paired end adapters for single end reads?
- `pair_num` : 1 for forward read, 2 for reverse read

**Value**

The adapter seq as string

---

**getBams**

Get bam files of a pipeline run

**Description**

Get bam files of a pipeline run

**Usage**

\[
\text{getBams}(\text{save}\_\text{dir})
\]

**Arguments**

- `save_dir` : Save directory of a pipeline run

**Value**

named list of bam files

**Author(s)**

Gregoire Pau
getChunkDirs  

Get the list of chunk directories

Description
Get the list of chunk directories

Usage
getChunkDirs()

Value
List of chunk directories

Author(s)
Gregoire Pau

getConfig  

Get a configuration parameter

Description
Get a configuration parameter

Usage
getConfig(p, stop.ifempty = FALSE)

Arguments
p  Name of parameter
stop.ifempty  throw error if value is not set, otherwise returns NULL

Value
If parameter is missing, return the config list otherwise return the value of the parameter name as a character string throws an exception if the parameter is not present in the config
getConfig.integer

Check if a config parameter is an integer

Description

Throws exception if value is no integer

Usage

getConfig.integer(p, tol = 1e-08, ...)

Arguments

<table>
<thead>
<tr>
<th>p</th>
<th>Name of parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>tol</td>
<td>Tolerance that controls how far a value can be from the next integer.</td>
</tr>
<tr>
<td>...</td>
<td>Additional parameters passed to getConfig()</td>
</tr>
</tbody>
</table>

Value

Value of parameter as integer

getConfig.logical

Check if a config parameter has a logical value

Description

Throws exception if value is not logical

Usage

getConfig.logical(p, ...)

Arguments

<table>
<thead>
<tr>
<th>p</th>
<th>Name of parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>extra params passed to getConfig</td>
</tr>
</tbody>
</table>

Value

Logical value of parameter
getConfig.numeric  
_Check if a config parameter is a numeric_

**Description**

Throws exception if value can't be cast into numeric

**Usage**

`getConfig.numeric(p, ...)`

**Arguments**

- `p` Name of parameter
- `...` Extra params passed to getConfig

**Value**

Value of parameter as numeric

---

getConfig.vector  
_Return values of a config variable as vector_

**Description**

Return values of a config variable as vector

**Usage**

`getConfig.vector(p, ...)`

**Arguments**

- `p` Name of parameter
- `...` extra params passed to getConfig

**Value**

value of config param as vector
**getEndNumber**  
*Get Read End Number*

**Description**

Returns the end number of an end from a paired-end read

**Usage**

```java
getEndNumber(int)
```

**Arguments**

- `int` an int from a SAM flag

**Details**

Given an integer from the BAM flag field, tells which end it is in a read

**Value**

1, 2

**Author(s)**

Cory Barr

---

**getMemoryUsage**  
*Returns memory usage in bytes*

**Description**

For debugging.

**Usage**

```java
getMemoryUsage()
```

**Value**

Memory usage in bytes
getNumberOfReadsInFASTQFile

*Count reads in Fastq file*

**Description**

Count reads in Fastq file

**Usage**

getNumberOfReadsInFASTQFile(filename)

**Arguments**

- **filename**: Name of FastQ file

**Value**

Number of reads

**Author(s)**

Gregoire Pau

getNumericVectorDataFromFile

*Load data as numerical values*

**Description**

Load data as numerical values

**Usage**

getNumericVectorDataFromFile(dir_path, object_name)

**Arguments**

- **dir_path**: Save dir of a pipeline run
- **object_name**: Object name

**Value**

loaded data as table of numbers

**Author(s)**

Jens Reeder
**getObjectFilename**

*Get a filename given a directory and the object name*

**Description**

Get a filename given a directory and the object name

**Usage**

```
getObjectFilename(dir_path, object_name)
```

**Arguments**

- `dir_path`: A character string containing a dir path
- `object_name`: A character string containing the regular expression matching a filename in `dir_path`

**Value**

A character vector containing an existing filename, stops if 0 or more than 1

**Author(s)**

Gregoire Pau

---

**getPackageFile**

*Get a package file*

**Description**

Magically get package files from the inst directory, which will be in different location, depending on whether we run in: - local mode: if interactive() is TRUE - package mode: if interactive() is FALSE

**Usage**

```
getPackageFile(filename, package = "HTSeqGenie", mustWork = TRUE)
```

**Arguments**

- `filename`: Name of package file
- `package`: Name of the package the file is coming from
- `mustWork`: Boolean, will stop the code if set to TRUE and file not found otherwise returns Nothing.
getRandomAlignCutoff  
**Estimate an adapter alignment cutoff score**

**Description**
Empirically estimate a threshold that discrimiates random reads from reads with adapter contamination.

**Usage**
getRandomAlignCutoff(read_len, n)

**Arguments**
- read_len: The read length
- n: Number of samples

getRRNAIds  
**Detect reads that look like rRNA**

**Description**
Detect reads that look like rRNA.

**Usage**
geRRNAIds(file1, file2 = NULL, tmp_dir, rRNADb)

**Arguments**
- file1: FastQ file of forward reads
- file2: FastQ of reverse reads in paired-end sequencing, NULL otherwise
- tmp_dir: temporary directory used for storing the gsnaps results
- rRNADb: Name of the rRNA sequence database. Must exist in the gsnaps genome directory

**Value**
IDs of reads flagged as rRNA
getTabDataFromFile  

Load tabular data from the NGS pipeline result directory

Description
Load tabular data from the NGS pipeline result directory

Usage
getTabDataFromFile(save_dir, object_name)

Arguments
save_dir  A character string containing an NGS pipeline output directory.
object_name  A character string containing the regular expression matching a filename in dir_path

Value
A data frame.

getTraceback  

Get traceback from tryKeepTraceback()

Description
Get traceback from tryKeepTraceback()

Usage
getTraceback(mto)

Arguments
mto  An object of the try-error class

Value
Traceback as a string
hashCoverage

**Description**
Hashing function for coverage

**Usage**
hashCoverage(cov)

**Arguments**
cov  A SimpleRleList object

**Value**
A numeric

**Author(s)**
Gregoire Pau

hashVariants

**Description**
Hashing function for variants

**Usage**
hashVariants(var)

**Arguments**
var  A GRanges object

**Value**
A numeric

**Author(s)**
Gregoire Pau
hashVector

| hashVector | Hashing function for vector |

Description
Hashing function for vector

Usage
hashVector(x)

Arguments
x A vector

Value
A numeric

Author(s)
Gregoire Pau

HTSeqGenie

Package overview

Description
The HTSeqGenie package is a robust and efficient software to analyze high-throughput sequencing experiments in a reproducible manner. It supports the RNA-Seq and Exome-Seq protocols and provides: quality control reporting (using the ShortRead package), detection of adapter contamination, read alignment versus a reference genome (using the gmapR package), counting reads in genomic regions (using the GenomicRanges package), and read-depth coverage computation.

Package content
To run the pipeline:
• runPipeline
To access the pipeline output data:
• getTabDataFromFile
To build the genomic features object:
• buildGenomicFeaturesFromTxDb
• TP53GenomicFeatures
initDirs

Set up NGS output dir

Description

Set up NGS output dir (using save_dir from getConfig)

Usage

initDirs()

Value

Nothing
**initLog**

*Initialize the logger*

**Description**

Setup logging file in save_dir/progress.log and log sessionInfo and configuration

**Usage**

```python
initLog(save_dir, debug_level = "INFO")
```

**Arguments**

- `save_dir`  
  Save dir of a pipeline run
- `debug_level`  
  One of INFO, WARN, ERROR, FATAL

**Value**

Log file name

**initLogger**

*Init loggers*

**Description**

Init loggers (output dir log, using save_dir from getConfig, and console log)

**Usage**

```python
initLogger()
```

**Value**

Nothing

**Author(s)**

Gregoire Pau
initPipelineFromConfig

*Init pipeline environment*

**Description**

Init pipeline environment

**Usage**

```py
initPipelineFromConfig(config_filename, config_update)
```

**Arguments**

- `config_filename`: Name of config file
- `config_update`: List of name value pairs that will update the config parameters

**Value**

Nothing

**Author(s)**

Jens Reeder

---

initPipelineFromSaveDir

*Init Pipeline environment from previous run*

**Description**

Init Pipeline environment from previous run

**Usage**

```py
initPipelineFromSaveDir(save_dir, config_update)
```

**Arguments**

- `save_dir`: Save dir of a previous pipeline run
- `config_update`: List of name value pairs that will update the config parameters

**Details**

Loads the config file from a previous run stored in `[save_dir]/logs/config.txt`
**isAboveQualityThresh**  
*Check for high quality reads*

---

**Value**  
Nothing

**Author(s)**  
Gregoire Pau

---

**Description**  
Checks whether reads have more than a fraction of minFrac nucleotides with a score below min-quality.

**Usage**  
isAboveQualityThresh(reads, minquality, minfrac)

**Arguments**

- **reads**  
  A set of reads as ShortReadQ object
- **minquality**  
  Minimal quality score
- **minfrac**  
  Fraction of positions that need to be over minquality to be considered a good read.

**Value**  
A boolean vector indicating whether read is considered high quality.

---

**isAdapter**  
*Detect adapter contamination*

---

**Description**  
Does a Needleman-Wunsch like small-in-large alignment of the adapter vs each read. Flag read if score exceeds threshold

**Usage**  
isAdapter(reads, score_cutoff, adapter_seqs)
**isConfig**  
*Test the presence of the parameter in the current config*

**Description**  
Test the presence of the parameter in the current config

**Usage**  
isConfig(parameter)

**Arguments**  
parameter  
Name of parameter

**Value**  
TRUE if present, FALSE otherwise

---

**isFirstFragment**  
*Does a SAM flag indicate the first fragment*

**Description**  
Compute whether a SAM/BAM flag indicates a first fragment. Method is not foolproof, as it ignores a lot of SAM semantics. E.g the SAM spec says: "If 0x1 is unset, no assumptions can be made about 0x2, 0x8, 0x20, 0x40 and 0x80". For our purpose this should be enough, but we should keep an open eye for a more robust implementation in Rsamtools.

**Usage**  
isFirstFragment(flag)

**Arguments**  
flag  
A flag from the BAM/SAM file
isSparse

Value

Logical

Description

Check coverage for sparseness

Usage

isSparse(cov, threshold = 0.1)

Arguments

cov         A cov object as SimpleRleList
threshold    Fraction of number of runs over total length

Details

Some Rle related operations become very slow when they are dealing with data that violates their sparseness assumption. This method provides an estimate about whether the data is dense or sparse. More precisely it checks if the fraction of the number of runs over the total length is smaller than a threshold

Value

Boolean whether this object is dense or sparse

Author(s)

Jens Reeder
listIterator.init  
*Create a iterator on a list*

### Description
Create a iterator on a list

### Usage
listIterator.init(x)

### Arguments
- x: A list.

### Details
Only one listIterator object can be open at any time.

### Value
Nothing

### Author(s)
Gregoire Pau

---

listIterator.next  
*Get reads from the listIterator*

### Description
Get reads from the listIterator

### Usage
listIterator.next()

### Value
An object. NULL if there are no more objects in the listIterator.

### Author(s)
Gregoire Pau

### See Also
listIterator.init
loadConfig

Description
Loads the indicated configuration file. Creates and installs a global variable that should be accessed only via getConfig().

Usage
loadConfig(filename)

Arguments
filename Path to configuration file

Value
Nothing. Called for its side effect, which is setting the global config variable.

logdebug

Description
Log debug (with a try statement)

Usage
logdebug(msg)

Arguments
... Arguments passed to logging::logdebug

Value
Nothing

Author(s)
Gregoire Pau
logerror  

*Log info using the logging package*

Description

Log error (with a try statement)

Usage

logerror(msg)

Arguments

... Arguments passed to logging::loginfo

Value

Nothing

Author(s)

Gregoire Pau

loginfo  

*Log info using the logging package*

Description

Log info (with a try statement)

Usage

loginfo(msg)

Arguments

... Arguments passed to logging::loginfo

Value

Nothing

Author(s)

Gregoire Pau
logwarn

Log warning using the logging package

Description
Log warning (with a try statement)

Usage
logwarn(msg)

Arguments
...  Arguments passed to logging::logwarn

Value
Nothing

Author(s)
Gregoire Pau

makeDir
Make a directory after performing an existence check

Description
Throws an exception if file or directory with same name exist and overwrite is TRUE.

Usage
makeDir(dir, overwrite = "never")

Arguments
dir  Name of directory to create
overwrite  A character string: never (default), erase, overwrite

Value
Path to created directory
markDuplicates

Description

Mark duplicates in bam

Usage

markDuplicates(bamfile, outfile = NULL, path = getOption("picard.path"))

Arguments

bamfile | Name of input bam file
outfile | Name of output bam file
path    | Full path to MarkDuplicates jar

Details

Use MarkDuplicates from PicardTools to mark duplicate alignments in bam file.

makeRandomSreads | Generate a couple of random ShortReadQ, intended for testing

Description

Generate a couple of random ShortReadQ, intended for testing

Usage

makeRandomSreads(num, len)

Arguments

num | an integer
len | an integer

Value

a DNAStringSet

Author(s)

Gregoire Pau
markDups

Value
Path to output bam file

Author(s)
Jens Reeder

mergeAlignReads

Description
Merge BAMs and create summary alignment file

Usage
mergeAlignReads(indirs, outdir, prepend_str, num_cores)
mergeCoverage

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>indirs</td>
<td>A character vector, indicating which directories have to be merged</td>
</tr>
<tr>
<td>outdir</td>
<td>A character string indicating the output directory (which must exist)</td>
</tr>
<tr>
<td>prepend_str</td>
<td>A character string, containing a prefix going to be appended on all output result files</td>
</tr>
<tr>
<td>num_cores</td>
<td>Number of cores available for parallel processing (for the merge bam step)</td>
</tr>
</tbody>
</table>

Value

Nothing

Author(s)

Gregoire Pau

mergeCoverage(indirs, outdir, prepend_str)

Description

Merge coverage files

Usage

mergeCoverage(indirs, outdir, prepend_str)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>indirs</td>
<td>A character vector, indicating which directories have to be merged</td>
</tr>
<tr>
<td>outdir</td>
<td>A character string indicating the output directory (which must exist)</td>
</tr>
<tr>
<td>prepend_str</td>
<td>A character string, containing a prefix going to be appended on all output result files</td>
</tr>
</tbody>
</table>

Details

Merges coverage objects, usually SimpleRleLists, in a tree-reduce fashion. The coverage object dynamically switches to a SimpleIntegerList, once the data becomes too dense.

Value

Nothing

Author(s)

Jens Reeder
mergeLanes  

**Merge input lanes**

**Description**

Merge input lanes built by the NGS pipeline

**Usage**

```r
mergeLanes(indirs, outdir, prepend_str, num_cores, config_update, 
        preMergeChecks.do = TRUE, ignoreConfigParameters)
```

**Arguments**

- `indirs` A character vector of directory paths containing NGS pipeline output
- `outdir` A character string pointing to a non-existing output directory
- `prepend_str` A character string, containing a prefix going to be appended on all output result files
- `num_cores` Number of cores available for parallel processing (for the merge bam step)
- `config_update` List of name value pairs that will update the config parameters
- `preMergeChecks.do` A logical, indicating whether to perform pre merge checks
- `ignoreConfigParameters` A character vector containing the configuration parameters that are not required to be identical

**Value**

Nothing

**Author(s)**

greg

mergePreprocessReads  

**Merge after preprocessReads**

**Description**

Merge detectAdapterContam, merge preprocessed reads, create summary preprocess, build shortReadReport, remove processed

**Usage**

```r
mergePreprocessReads(indirs, outdir, prepend_str)
```
mergeSummaryAlignment

**Arguments**

- `indirs`: A character vector, indicating which directories have to be merged
- `outdir`: A character string indicating the output directory (which must exist)
- `prepend_str`: A character string, containing a prefix going to be appended on all output result files

**Value**

Nothing

**Author(s)**

Gregoire Pau

---

`mergeSummaryAlignment`  
*Merge summary alignments*

**Description**

Merge summary alignments

**Usage**

`mergeSummaryAlignment(indirs, outdir, prepend_str)`

**Arguments**

- `indirs`: A character vector, indicating which directories have to be merged
- `outdir`: A character string indicating the output directory (which must exist)
- `prepend_str`: A character string, containing a prefix going to be appended on all output result files

**Value**

Nothing

**Author(s)**

Gregoire Pau
parseDCF

Read and parse a configuration file

Description

From a file like x1: y1 x2: y2 extract field, using the rules: - split on ':' - first element of split id name of parameter, second is value - trailing whitespaces (tabs and spaces) are removed - comments (text flow starting with #) are removed

Usage

parseDCF(filename)

Arguments

filename File name

Value

Named list

parseSummaries

parse summary files from save dirs

Description

Parse a summary from a list of save_dirs

Usage

parseSummaries(save.dirs, summary.name)

Arguments

save.dirs list of result dirs
summary.name name of summary file e.g. summary_counts

Details

This function allows to parse a given summary from a list of pipeline results save_dirs

Value

data frame with summaries

Author(s)

Jens Reeder
picard

Description
Generic function to call all picard command line java tools

Usage
picard(tool, ..., path = getOption("picard.path"))

Arguments
- tool: Name of the Picard Tool, e.g. MarkDuplicates
- ...: Arguments forwarded to the picard tool
- path: full path to the picard tool jar file.

Value
Nothing

Author(s)
Jens Reeder, Michael Lawrence

plotDF
Make continuous plots of distribution function

Description
Make continuous plots of distribution function

Usage
plotDF(df, ylab, xlab, filename)

Arguments
- df: distribution function, given as absolute count and percent
- ylab: label of y axis
- xlab: label of x axis
- filename: plots will be saved under [filename].png and [filename].pdf
preprocessReads

Value
Nothing, creates two files instead

Author(s)
Jens Reeder

__________________________________________

**preprocessReads**  
**Pipeline preprocessing**

Description
The preprocessing for our NGS pipelines consists of:
- quality filtering
- check for adapter contamination
- filtering of rRNA reads
- read trimming
- shortRead report generation of surviving reads

Usage
preprocessReads()

Details
These steps are mostly controlled by the global config.

Value
A named vector containing the path to the preprocessed FastQ files and a few other statistics

__________________________________________

**preprocessReadsChunk**  
**Preprocess a chunk**

Description
Preprocess a chunk

Usage
preprocessReadsChunk(lreads, save_dir = NULL)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lreads</td>
<td>A list of GRanges objects, containing the reads</td>
</tr>
<tr>
<td>save_dir</td>
<td>Save directory of a pipeline run</td>
</tr>
</tbody>
</table>
**Value**

save_dir Save directory of a pipeline run

**Author(s)**

Gregoire Pau

---

**processChunks**

*Process chunk in the pipeline framework*

**Description**

Process chunk in the pipeline framework

**Usage**

```
processChunks(inext, fun, nb.parallel.jobs)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>inext</td>
<td>A function (without argument) returning an object to process; NULL if none</td>
</tr>
<tr>
<td>fun</td>
<td>Function to process the object returned by inext; this function is run in</td>
</tr>
<tr>
<td></td>
<td>children thread; this function is run in children thread.</td>
</tr>
<tr>
<td>nb.parallel.jobs</td>
<td>number of parallel jobs</td>
</tr>
</tbody>
</table>

**Details**

High-level pipeline-specific version of sclapply, with chunk loggers and safeExecute

**Value**

Nothing

**Author(s)**

Gregoire Pau
readInputFiles  Read FastQ input files

Description
Uses the global config to find input files

Usage
readInputFiles()

Value
Reads as list of ShortRead objects

readRNASeqEnds  Read single/paired End Bam Files

Description
Read single/paired end BAM files with requested columns from the BAM

Usage
readRNASeqEnds(filename, paired_ends, remove.strandness = TRUE)

Arguments
filename
Path to a bam file
paired_ends
A logical indicating whether the reads are paired
remove.strandness
A logical indicating whether read strands should be set to "*".

Value
GRangesList

Author(s)
Cory Barr
realignIndels

Description
Realign indels in pipeline context

Usage
realignIndels()

Details
High level function call to realign indels in the analyzed.bam file using GATK

Value
Nothing

Author(s)
Jens Reeder

realignIndelsGATK

Description
Realign indels using the GATK tools RealignerTargetCreator and IndelRealigner. Requires a GATK compatible genome with a name matching the alignment genome to be installed in 'path.gatk_genome'

Usage
realignIndelsGATK(bam.file)

Arguments
bam.file Path to bam.file

Details
Since GATKs IndelRealigner is not parallelized, we run it in parallel per chromosome.

Value
Path to realigned bam file
relativeBarPlot

Author(s)
Jens Reeder

relativeBarPlot Make relative bar plots

Description
Make relative bar plots

Usage
relativeBarPlot(data, total, labels, title, filename, ylab = "Percent", cex.names = 0.9, ymax = 100)

Arguments
data vector of raw, absolute counts
total number to normalize by, can be vector of same length as data
labels x-axes labels, category labels for data
title Title of the plot
filename plots will be saved under [filename].png and [filename].pdf
ylab label of y axis
cex.names scaling param of labels, passed to plot
ymax extent of y-axis

Value
Nothing, creates two files instead

removeChunkDir Remove chunk directories

Description
Remove chunk directories

Usage
removeChunkDir()
Details

A pipeline run processes the data in small chunks, which are eventually combined into the final result. Afterwards, this function can be called to remove the temporary results per chunk.

Value

Nothing

Author(s)

Jens Reeder

resource

Reload package source code

Description

When developing code this function can be used to quickly reload all of the packages code, without installing it.

Usage

resource(dirname = ".")

Arguments

dirname Directore with files to source

Value

Nothing

rpkm

Calculate RPKM

Description

Calculate RPKM

Usage

rpkm(counts, widths, nbreads)
runAlignment

Arguments

  counts      A vector of counts
  widths      vector of the width of each bin the counts were performed on
  nbreads     vector containing number of reads mapped to each bin

Value

  vector of RPKMs

Author(s)

  Gregoire Pau

Description

  Runs the read alignment step of the pipeline

Usage

  runAlignment(config_filename, config_update)

Arguments

  config_filename  Path to configuration file
  config_update   List of name value pairs that will update the config parameters

Value

  Nothing

Author(s)

  Jens Reeder
runPipeline

Run the NGS analysis pipeline

Description

Run the NGS analysis pipeline

Usage

runPipeline(...)

Arguments

... A list of parameters. See the vignette for details.

Details

This function starts the pipeline. It first preprocesses the input FASTQ reads, align them, count the read overlaps with genomic features and compute the coverage. See the vignette for details.

Value

The path to the NGS output directory.

Author(s)

Jens Reeder, Gregoire Pau

See Also

TP53Genome, TP53GenomicFeatures

Examples

## Not run:
## build genome and genomic features
tp53Genome <- TP53Genome()
tp53GenomicFeatures <- TP53GenomicFeatures()

## get the FASTQ files
fastq1 <- system.file("extdata/H1993_TP53_subset2500_1.fastq.gz", package="HTSeqGenie")
fastq2 <- system.file("extdata/H1993_TP53_subset2500_2.fastq.gz", package="HTSeqGenie")

## run the pipeline
save_dir <- runPipeline(
## input
  input_file=fastq1,
  input_file2=fastq2,
  paired_ends=TRUE,
runPipelineConfig

quality_encoding="illumina1.8",

## output
save_dir="test",
prepend_str="test",
overwrite_save_dir="erase",

## aligner
path.gsnap_genomes=path(directory(tp53Genome)),
alignReads.genome=genome(tp53Genome),
alignReads.additional_parameters="--indel-penalty=1 --nonsplicing=1 --distant-splice-penalty=1",

## gene model
path.genomic_features=dirname(tp53GenomicFeatures),
countGenomicFeatures.gfeatures=basename(tp53GenomicFeatures)
)

## End(Not run)

runPipelineConfig Run the NGS analysis pipeline

Description
Run the NGS analysis pipeline from a configuration file

Usage
runPipelineConfig(config_filename, config_update)

Arguments
config_filename
Path to a pipeline configuration file
config_update A list of name value pairs that will update the config parameters

Details
This is the launcher function for all pipeline runs. It will do some preprocessing steps, then aligns the reads, counts overlap with genomic Features such as genes, exons etc and applies a variant caller.

Value
Nothing
**safe.yield**

*Overloaded yield(...) method catching truncated exceptions for FastqStreamer*

**Description**

Overloaded yield(...) method catching truncated exceptions for FastqStreamer

**Usage**

```python
safe.yield(fqs)
```

**Arguments**

- `fqs` An instance from the FastqSampler or FastqStreamer class.

---

**runPreprocessReads**  
*Run the preprocessing steps of the pipeline*

**Description**

Runs the preprocessing steps of the pipeline

**Usage**

```python
runPreprocessReads(config_filename, config_update)
```

**Arguments**

- `config_filename` Path to configuration file
- `config_update` List of name value pairs that will update the config parameters

**Value**

Nothing

**Author(s)**

Jens Reeder

---

**safe.yield**

Overloaded yield(...) method catching truncated exceptions for FastqStreamer

**Description**

Overloaded yield(...) method catching truncated exceptions for FastqStreamer

**Usage**

```python
safe.yield(fqs)
```

**Arguments**

- `fqs` An instance from the FastqSampler or FastqStreamer class.
safeExecute

Value

Same as FastqStreamer::yield

Author(s)

Gregoire Pau

safeExecute  

Execute function in try catch with trace function

Description

Requires the logger to be set

Usage

safeExecute(expr, memtracer = TRUE, newthread = TRUE)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>expr</td>
<td>Expression to safely execute</td>
</tr>
<tr>
<td>memtracer</td>
<td>A boolean, to enable/disable a periodic memory tracer. Default is TRUE.</td>
</tr>
<tr>
<td>newthread</td>
<td>A boolean, indicating if a new thread should be used (to save memory from the main thread)</td>
</tr>
</tbody>
</table>

Value

Nothing

Author(s)

Gregoire Pau

safeGetObject  

Safely load a R data file

Description

Attempts to load a file given by object_name. Bails out if none or more than one files match the object name.

Usage

safeGetObject(dir_path, object_name)
safeUnlink

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dir_path</td>
<td>Save dir of a pipeline run</td>
</tr>
<tr>
<td>object_name</td>
<td>object name, can be a regexp</td>
</tr>
</tbody>
</table>

Value

loaded object

Description

Symlink-safe file/directory delete function

Usage

safeUnlink(path)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>path</td>
<td>A character string indicating which file/directory to delete.</td>
</tr>
</tbody>
</table>

Details

Unlike unlink(), safeUnlink() does not follow symlink directories for deletion.

Value

Nothing

Author(s)

Gregoire Pau
saveWithID  

**Save an R object**

**Description**

Exists so objects can be serialized and reloaded with the a unique identifier in the symbol. Stores the data object with a new name.

**Usage**

```r
saveWithID(data, orig_name, id, save_dir, compress = TRUE, format = "RData")
```

**Arguments**

- `data`  
  The data to store

- `orig_name`  
  The original name of the data

- `id`  
  A meaningful id that is prepended to the stored objects name

- `save_dir`  
  The directory where the data should be saved in

- `compress`  
  Save the data compressed or not

- `format`  
  Choice of 'RData' or 'tab'ular

**Value**

Name of the stored file

sclapply  

**Scheduled parallel processing**

**Description**

Scheduled parallel processing

**Usage**

```r
sclapply(inext, fun, max.parallel.jobs, ..., stop.onfail = TRUE,
         tracefun = NULL, tracefun.period = 60)
```
Arguments

inext A function (without argument) returning an object to process; NULL if none left; this function is run in the main thread

fun Function to process the object returned by inext; this function is run in children thread

max.parallel.jobs Number of jobs to start in parallel

... Further arguments passed to fun

stop.onfail Throw error if one

tracefun Callback function that will be executed in a separate thread

tracefun.period Time intervall between calls to tracefun

Value

Return value of applied function

---

setChunkDir Set the base directory for the chunks

---

Description

Set the base directory for the chunks

Usage

setChunkDir()

Value

path to chunk dir

Author(s)

Jens Reeder
**setUpDirs**  
*Create output directory and subdirectories for sequencing pipeline analysis outputs*

---

**Description**

Creates a directory with all needed subdirectories for pipeline outputs

**Usage**

```r
setUpDirs(save_dir, overwrite = "never")
```

**Arguments**

- `save_dir` path to the directory that will contain all needed subdirectories
- `overwrite` A character string: never (default), erase, overwrite

**Value**

Nothing. Called for its side effects

**Author(s)**

Cory Barr, Jens Reeder

---

**setupTestFramework**  
*setup test framework*

---

**Description**

setup test framework

**Usage**

```r
setupTestFramework(config.filename, config.update = list(),
                   testname = "test", package = "HTSeqGenie", use.TP53Genome = TRUE)
```

**Arguments**

- `config.filename` configuration file
- `config.update` update list of config values
- `testname` name of test case
- `package` name of package
- `use.TP53Genome` Boolean indicating the use of the TP53 genome as template config
Value
the created temp directory

statCountFeatures  Compute statistics on count features

Description
Compute statistics on count features

Usage
statCountFeatures(save_dir, feature = "counts_gene")

Arguments
save_dir  A character string containing a NGS analysis directory
feature  A character string containing a features name. Default is "counts_gene".

Value
A numeric vector containing statistics about features.

Author(s)
Gregoire Pau

TP53GenomicFeatures  Demo genomic features around the TP53 gene

Description
Build the genomic features of the TP53 demo region

Usage
TP53GenomicFeatures()

Details
Returns a list of genomic features (gene, exons, transcripts) annotating a region of UCSC hg19 sequence centered on the region of the TP53 gene, with 1 Mb flanking sequence on each side. This is intended as a test/demonstration to run the NGS pipeline in conjunction with the LungCancerLines data package.
### traceMem

**Value**

A list of GRanges objects containing the genomic features

**Author(s)**

Gregoire Pau

**See Also**

TP53Genome, buildGenomicFeaturesFromTxDb, runPipeline

---

### trimReads

**Description**

Trim/truncate a set of reads

**Usage**

```r
trimReads(lreads, trim_len = NULL, trim5 = 0)
```

**Arguments**

- `lreads`: A list of ShortReadQ objects
- `trim_len`: The length reads will be truncated to; default is NULL (no length truncation)
- `trim5`: The number of nucleotides to trim from the 5’-end; default is 0

**Value**

A list of truncated ShortReadQ objects
**trimTailsByQuality**  *Trim off low quality tail*

**Description**

The illuminsa manuals states: If a read ends with a segment of mostly low quality (Q15 or below), then all of the quality values in the segment are replaced with a value of 2 (encoded as the letter B in Illumina’s text-based encoding of quality scores)... This Q2 indicator does not predict a specific error rate, but rather indicates that a specific final portion of the read should not be used in further analyses.

**Usage**

```r
trimTailsByQuality(lreads, minqual = "#")
```

**Arguments**

- `lreads`: A list (usually a pair) of ShortReadQ object
- `minqual`: An ascii encoded quality score

**Details**

For illumina 1.8 the special char is encoded as ‘#’, which we choose as default here. For illumina 1.5 make sure to set the minqual to ‘B’

**Value**

A list of quality trimmed ShortReadQ objects

---

**truncateReads**  *Trim/truncate a set of reads*

**Description**

Trim/truncate a set of reads

**Usage**

```r
truncateReads(reads, trim_len = NULL, trim5 = 0)
```

**Arguments**

- `reads`: A set of reads as ShortReadQ object
- `trim_len`: The length reads will be truncated to; default is NULL (no length truncation)
- `trim5`: The number of nucleotides to trim from the 5'-end; default is 0
tryKeepTraceback

Value
A truncated ShortReadQ object

Description
Wrapper around try-catch

Usage
tryKeepTraceback(expr)

Arguments
expr Expression to evaluate

Value
Result of expression or error if thrown

updateConfig

Description
Update the existing config

Usage
updateConfig(tconfig)

Arguments
tconfig List of configuration name value pairs

Value
Nothing.
## vcfStat  
*Compute stats on a VCF file*

### Description
Compute stats on a VCF file

### Usage
`vcfStat(vcf.filename)`

### Arguments
- `vcf.filename`: A character pointing to a VCF (or gzipped VCF) file

### Value
A numeric vector

### Author(s)
Gregoire Pau

## wrap.callVariants  
*Variant calling*

### Description
Call Variants in the pipeline framework

### Usage
`wrap.callVariants(bam.file)`

### Arguments
- `bam.file`: Aligned reads as bam file

### Details
A wrapper around VariantTools callVariant framework.

### Value
Variants as Vranges

### Author(s)
Jens Reeder
writeAudit

Write Session information

Description
Write Session information

Usage
writeAudit(filename)

Arguments
filename Optional name of file. If missing, prints session information on the standard output.

Value
Nothing

Author(s)
Gregoire Pau

writeConfig

Write a config file

Description
Writes the currently active configuration to file

Usage
writeConfig(config.filename)

Arguments
config.filename Optional name of output file. If missing, print the config file on the standard output.

Value
Name of saved file
writeFastQFiles  Write reads to file

Description
Write reads to file

Usage
writeFastQFiles(lreads, dir, filename1, filename2)

Arguments

  lreads       List of reads as ShortRead objects
  dir          Save directory
  filename1    Name of file 1
  filename2    Name of file 2

Value
Named list of filepaths

writeFeatureCountsHTML  writeFeatureCountsHTML

Description
writeFeatureCountsHTML

Usage
writeFeatureCountsHTML(outfile, dirPath, ExonsCoveredTable,
                         GenomicFeaturesTable, GenomicFeaturesDetectedTable)

Arguments

  outfile      a path
  dirPath      a path
  ExonsCoveredTable    a table
  GenomicFeaturesTable    a table
  GenomicFeaturesDetectedTable    a table
**writeGenomicFeaturesReport**

*Generate pipeline report*

**Value**

Nothing

**Author(s)**

Gregoire Pau

---

**writePreprocessAlignHTML**

**Description**

writePreprocessAlignHTML

**Usage**

writePreprocessAlignHTML(outfile, dirPath, sanity_check, readFilteringTable, ReadMappingsTable, targetLengthTable)

**Value**

Name of created HTML file

**Author(s)**

Melanie Huntley, Cory Barr, Jens Reeder
Arguments

- **outfile**: a path
- **dirPath**: a path
- **sanity_check**: a logical
- **readFilteringTable**: a table
- **ReadMappingsTable**: a table
- **targetLengthTable**: a table

Value

- Nothing

Author(s)

- Gregoire Pau

---

**writePreprocessAlignReport**

*Generate Pipeline Report*

---

Description

Generates a summary HTML for the preprocess and align step

Usage

**writePreprocessAlignReport()**

Value

- Name of created HTML file

Author(s)

- Melanie Huntley, Cory Barr, Jens Reeder
writeSummary

Write HTML summary

Description
Write html Summary for list of runs

Usage
writeSummary(dirs, cutoffs, outdir = "./"")

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dirs</td>
<td>List of pipeline result dirs</td>
</tr>
<tr>
<td>cutoffs</td>
<td>list, cutoffs for each plotting/QA function</td>
</tr>
<tr>
<td>outdir</td>
<td>Path to output directory. Does not create dir.</td>
</tr>
</tbody>
</table>

Value
Nothing, but writes file

Author(s)
Jens Reeder

writeVCF

writeVCF

Description
Write variants to VCF file

Usage
writeVCF(variants.vranges, filename)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>variants.vranges</td>
<td>Genomic Variants as VRanges object</td>
</tr>
<tr>
<td>filename</td>
<td>Name of vcf file to write</td>
</tr>
</tbody>
</table>

Value
VCF file name
Author(s)

Jens Reeder
Index

* internal

  alignReads, 4
  alignReadsChunk, 5
  bamCountUniqueReads, 7
  buildConfig, 7
  buildShortReadReports, 9
  buildTallyParam, 9
  buildTP53FastaGenome, 10
  buildTP53GenomeTemplate, 10
  calculateCoverage, 11
  calculateTargetLengths, 11
  checkConfig, 12
  checkPicardJar, 13
  computeBamStats, 14
  computeCoverage, 14
  countFeatures, 15
  countGenomicFeatures, 16
  countGenomicFeaturesChunk, 16
  createTimeDir, 17
  detectAdapterContam, 17
  detectQualityInFASTQFile, 18
  FastQStreamer.getReads, 20
  FastQStreamer.init, 20
  FastQStreamer.release, 21
  filterByLength, 22
  filterQuality, 22
  findVariantFile, 23
  getAdapterSeqs, 25
  getBams, 25
  getChunkDirs, 26
  getConfig, 26
 getConfig.integer, 27
  getConfig.logical, 27
  getConfig.numeric, 28
  getConfig.vector, 28
  getEndNumber, 29
  getMemoryUsage, 29
  getNumberOfReadsInFASTQFile, 30
  getNumericVectorDataFromFile, 30
  getPackageFile, 31
  getRandomAlignCutoff, 32
  getTraceback, 33
  initDirs, 36
  initLog, 37
  initLogger, 37
  initPipelineFromConfig, 38
  initPipelineFromSaveDir, 38
  isAboveQualityThresh, 39
  isAdapter, 39
  isConfig, 40
  isFirstFragment, 40
  listIterator.init, 42
  listIterator.next, 42
  loadConfig, 43
  logdebug, 43
  logerror, 44
  loginfo, 44
  logwarn, 45
  makeDir, 45
  makeRandomSreads, 46
  mergeAlignReads, 47
  mergeCoverage, 48
  mergeLanes, 49
  mergePreprocessReads, 49
  mergeSummaryAlignment, 50
  parseDCF, 51
  parseSummaries, 51
  picard, 52
  plotDF, 52
  preprocessReads, 53
  preprocessReadsChunk, 53
  processChunks, 54
  readInputFiles, 55
  readRNASeqEnds, 55
  relativeBarPlot, 57
  removeChunkDir, 57
  resource, 58
rpkm, 58
runAlignment, 59
runPreprocessReads, 62
safe.yield, 62
safeExecute, 63
safeGetObject, 63
safeUnlink, 64
saveWithID, 65
scapply, 65
setChunkDir, 66
setUpDirs, 67
statCountFeatures, 68
traceMem, 69
trimReads, 69
trimTailsByQuality, 70
truncateReads, 70
tryKeepTraceback, 71
updateConfig, 71
writeAudit, 73
writeConfig, 73
writeFastQFiles, 74
writeFeatureCountsHTML, 74
writeGenomicFeaturesReport, 75
writePreprocessAlignHTML, 75
writePreprocessAlignReport, 76
writeSummary, 77

* package
  HTSeqGenie, 35

alignReads, 4
alignReadsChunk, 5
analyzeVariants, 6
annotateVariants, 6

bamCountUniqueReads, 7
buildConfig, 7
buildGenomicFeaturesFromTxDb, 8
buildShortReadReports, 9
buildTallyParam, 9
buildTP53FastaGenome, 10
buildTP53GenomeTemplate, 10

calculateCoverage, 11
calculateTargetLengths, 11
callVariantsGATK, 12
checkConfig, 12
checkGATKJar, 13
checkPicardJar, 13
computeBamStats, 14

computeCoverage, 14
countFeatures, 15
countGenomicFeatures, 16
countGenomicFeaturesChunk, 16
createTmpDir, 17
detectAdapterContam, 17
detectQualityInFASTQFile, 18
detectRRNA, 18

ecludeVariantsByRegions, 19

FastQStreamer.getReads, 20
FastQStreamer.init, 20
FastQStreamer.release, 21
filterByLength, 22
filterQuality, 22
findVariantFile, 23

gatk, 23
generateSingleGeneDERs, 24
getAdapterSeqs, 25
getBams, 25
getChunkDirs, 26
getConfig, 26
getConfig.integer, 27
getConfig.logical, 27
getConfig.numeric, 28
getConfig.vector, 28
getEndNumber, 29
getMemoryUsage, 29
getNumberOfReadsInFASTQFile, 30
getNumericVectorDataFromFile, 30
getObjectFilename, 31
getPackageFile, 31
getRandomAlignCutoff, 32
getRRNAIds, 32
getTabDataFromFile, 33
getTraceback, 33

hashCoverage, 34
hashVariants, 34
hashVector, 35
HTSeqGenie, 35

initDirs, 36
initLog, 37
initLogger, 37
initPipelineFromConfig, 38
initPipelineFromSaveDir, 38
INDEX

isAboveQualityThresh, 39
isAdapter, 39
isConfig, 40
isFirstFragment, 40
isSparse, 41
listIterator.init, 42
listIterator.next, 42
loadConfig, 43
logdebug, 43
logerror, 44
loginfo, 44
logwarn, 45
makeDir, 45
makeRandomSreads, 46
markDuplicates, 46
mergeAlignReads, 47
mergeCoverage, 48
mergeLanes, 49
mergePreprocessReads, 49
mergeSummaryAlignment, 50
parseDCF, 51
parseSummaries, 51
picard, 52
plotDF, 52
preprocessReads, 53
preprocessReadsChunk, 53
processChunks, 54
readInputFiles, 55
readRNASeqEnds, 55
realignIndels, 56
realignIndelsGATK, 56
relativeBarPlot, 57
removeChunkDir, 57
resource, 58
rpkm, 58
runAlignment, 59
runPipeline, 60
runPipelineConfig, 61
runPreprocessReads, 62
safe.yield, 62
safeExecute, 63
safeGetObject, 63
safeUnlink, 64
saveWithID, 65
sclapply, 65
setChunkDir, 66
setUpDirs, 67
setupTestFramework, 67
statCountFeatures, 68
TP53GenomicFeatures, 68
traceMem, 69
trimReads, 69
trimTailsByQuality, 70
truncateReads, 70
tryKeepTraceback, 71
updateConfig, 71
vcfStat, 72
wrap.callVariants, 72
writeAudit, 73
writeConfig, 73
writeFastQFiles, 74
writeFeatureCountsHTML, 74
writeGenomicFeaturesReport, 75
writePreprocessAlignHTML, 75
writePreprocessAlignReport, 76
writeSummary, 77
writeVCF, 77