Package ‘HTSeqGenie’

March 22, 2017

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  4.4.0

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

RoxygenNote  5.0.1

NeedsCompilation  no

R topics documented:

analyzeVariants .......................................................... 2
annotateVariants .......................................................... 2
buildGenomicFeaturesFromTxDb ........................................ 3
callVariantsGATK ........................................................... 4
checkGATKJar ............................................................... 4
detectRRNA ................................................................. 5
excludeVariantsByRegions ................................................. 5
gatk ................................................................. 6
generateSingleGeneDERs ............................................... 7
getRRNAIds ............................................................ 7
getTabDataFromFile .................................................... 8
hashCoverage ............................................................ 8
analyzeVariants

**Description**

Calculate and process Variants

**Usage**

analyzeVariants()

**Value**

Nothing

**Author(s)**

Jens Reeder

---

annotateVariants

**Description**

Annotate variants via vep

**Usage**

annotateVariants(vcf.file)

**Arguments**

 vc.f.ile A character vector pointing to a VCF (or gzipped VCF) file
**buildGenomicFeaturesFromTxDb**

Build genomic features from a TxDb object

**Description**

Build genomic features from a TxDb object

**Usage**

```r
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)
```
**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**
bam.file  
Path to bam.file

**Value**
Path to variant file

**Author(s)**
Jens Reeder

---

**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
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

Value
a named logical vector indicating if a read has rRNA contamination

Author(s)
Cory Barr

excludeVariantsByRegions

Filter variants by regions

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

---

**Description**

Run a command from the GATK

**Usage**

```
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

**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
**generateSingleGeneDERs**

**Description**
Generate DEXSeq-ready exons

**Usage**
generateSingleGeneDERs(txdb)

**Arguments**
- **txdb**
  A transcript DB object

**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

---

**getRRNAIds**

**Detect reads that look like rRNA**

**Description**
Detect reads that look like rRNA

**Usage**
getRRNAIds(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 gsnap results
- **rRNADb**
  Name of the rRNA sequence database. Must exist in the gsnap 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.

---

hashCoverage  
*Hashing function for coverage*

**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

**Description**
Hashing function for vector

**Usage**
hashVector(x)

**Arguments**
- **x**: A vector

**Value**
A numeric

**Author(s)**
Gregoire Pau
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.

To run the pipeline:
- runPipeline

To access the pipeline output data:
- getTabDataFromFile

To build the genomic features object:
- buildGenomicFeaturesFromTxDb
- 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,
  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 --novelsplicing=1 --distant-splice-penalty=1",
)```
isSparse

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

## End(Not run)

### 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

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.

Value

Path to output bam file

Author(s)

Jens Reeder

Description

Mark duplicates in pipeline context

Usage

markDups()

Details

High level function call to mark duplicates in the analyzed.bam file of a pipeline run.

Value

Nothing

Author(s)

Jens Reeder
**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**
Realigning 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 GATK's IndelRealigner is not parallelized, we run it in parallel per chromosome.

**Value**
Path to realigned bam file

**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")
faspq2 <- 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,
  quality_encoding="illumina1.8",

  ## output
  save_dir="test",
  prepend_str="test"),

runPipelineConfig

## Run the NGS analysis pipeline

**Description**

Run the NGS analysis pipeline from a configuration file

**Usage**

```r
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

**Author(s)**

Jens Reeder, Gregoire Pau
setupTestFramework  setup test framework

Description
setup test framework

Usage
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

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.

Value
A list of GRanges objects containing the genomic features

Author(s)
Gregoire Pau
See Also
TP53Genome, buildGenomicFeaturesFromTxDb, runPipeline

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
Description

Write variants to VCF file

Usage

writeVCF(variants.vranges, filename)

Arguments

variants.vranges
  Genomic Variants as VRanges object
filename
  Name of vcf file to write

Value

VCF file name

Author(s)

Jens Reeder
Index

*Topic package*
  HTSeqGenie, 10

analyzeVariants, 2
annotateVariants, 2

buildGenomicFeaturesFromTxDb, 3

callVariantsGATK, 4
checkGATKJar, 4

detectRRNA, 5
excludeVariantsByRegions, 5

gatk, 6
generateSingleGeneDERs, 7
getRRNAIds, 7
getTabDataFromFile, 8

hashCoverage, 8
hashVariants, 9
hashVector, 9
HTSeqGenie, 10

isSparse, 11
markDuplicates, 11
markDups, 12

realignIndels, 13
realignIndelsGATK, 13
runPipeline, 14
runPipelineConfig, 15

setupTestFramework, 16

TP53GenomicFeatures, 16

vcfStat, 17

wrap.callVariants, 17
writeVCF, 18