Package 'genotypeeval'

October 12, 2016

Title QA/QC of a gVCF or VCF file

Version 1.2.2

Description Takes in a gVCF or VCF and reports metrics to assess quality of calls.

Depends R (>= 3.2.0), VariantAnnotation

Imports ggplot2, rtracklayer, BiocGenerics, GenomicRanges, GenomeInfoDb, IRanges, methods, BiocParallel

Collate GoldDataParam.R GoldData.R VCFData.R VCFQAParam.R VCFQAReport.R PopulationSummary.R

License file LICENSE

LazyData true

Suggests knitr, testthat, SNPlocs.Hsapiens.dbSNP141.GRCh38, TxDb.Hsapiens.UCSC.hg38.knownGene

VignetteBuilder knitr

Author Jennifer Tom [aut, cre]

Maintainer Jennifer Tom <tom.jennifer@gene.com>

biocViews Genetics, BatchEffect, Sequencing, SNP, VariantAnnotation, DataImport

NeedsCompilation no

R topics documented:

`didSamplePass` *Getter for VCFEvaluate class to check if Sample Passed. Using thresholds from VCFQAParam object return a list. First return whether each test was passed (TRUE) or failed (FALSE). Then return an overall pass (TRUE) or fail (FALSE).*

Description

Getter for VCFEvaluate class to check if Sample Passed. Using thresholds from VCFQAParam object return a list. First return whether each test was passed (TRUE) or failed (FALSE). Then return an overall pass (TRUE) or fail (FALSE).

Usage

```
didSamplePass(Object)
```

Arguments

`Object`	an object of type VCFQAReport

Value

Vector of True and False

Examples

```
vcffn <- system.file("ext-data", "chr22.GRCh38.vcf.gz", package="genotypeeval")
mydir <- paste(dirname(vcffn), "/", sep="")
myfile <-basename(vcffn)
svp <- ScanVcfParam(which=GRanges("22", IRanges(0,200e5)), geno="GT")
vcfparams <- VCFQAParam(count.limits=c(3014580000, Inf), readdepth.target = 30)
vcf <- ReadVCFData(mydir, myfile, "GRCh38")
ev <- VCFEvaluate(vcf, vcfparams)
didSamplePass(ev)
```

`didSamplePassOverall` *Getter for VCFEvaluate class to check if Sample Passed. Using thresholds from VCFQAParam object return a list. First return whether each test was passed (TRUE) or failed (FALSE). Then return an overall pass (TRUE) or fail (FALSE).*

Description

Getter for VCFEvaluate class to check if Sample Passed. Using thresholds from VCFQAParam object return a list. First return whether each test was passed (TRUE) or failed (FALSE). Then return an overall pass (TRUE) or fail (FALSE).

Usage

```
didSamplePassOverall(Object)
```

Arguments

`Object`	an object of type VCFQAReport

Value

True or False if sample passed all thresholds

Examples

```
vcffn <- system.file("ext-data", "chr22.GRCh38.vcf.gz", package="genotypeeval")
mydir <- paste(dirname(vcffn), "/", sep="")
myfile <-basename(vcffn)
svp <- ScanVcfParam(which=GRanges("22", IRanges(0,200e5)), geno="GT")
vcfparams <- VCFQAParam(count.limits=c(3014580000, Inf), readdepth.target = 30)
vcf <- ReadVCFData(mydir, myfile, "GRCh38")
ev <- VCFEvaluate(vcf, vcfparams)
didSamplePassOverall(ev)
```

`getName` *Getter for VCFQAReport class to return filename slot*

Description

Getter for VCFQAReport class to return filename slot

Usage

```
getName(Object)
```

Arguments

Object	Object of class VCFQAReport

Value

Name of file

Examples

```
vcffn <- system.file("ext-data", "chr22.GRCh38.vcf.gz", package="genotypeeval")
mydir <- paste(dirname(vcffn), "/", sep="")
myfile <-basename(vcffn)
svp <- ScanVcfParam(which=GRanges("22", IRanges(0,200e5)), geno="GT")
vcfparams <- VCFQAParam(count.limits=c(3014580000, Inf), readdepth.target = 30)
vcf <- ReadVCFData(mydir, myfile, "GRCh38")
ev <- VCFEvaluate(vcf, vcfparams)
getName(ev)
```

---

getPlots *Getter for VCFQAReport class to return plots slot.*

Description

Getter for VCFQAReport class to return plots slot.

Usage

```
getPlots(Object)
```

Arguments

Object	Object of Class VCFQAReport

Value

List of named ggplots

Examples

```
vcffn <- system.file("ext-data", "chr22.GRCh38.vcf.gz", package="genotypeeval")
mydir <- paste(dirname(vcffn), "/", sep="")
myfile <-basename(vcffn)
svp <- ScanVcfParam(which=GRanges("22", IRanges(0,200e5)), geno="GT")
vcfparams <- VCFQAParam(count.limits=c(3014580000, Inf), readdepth.target = 30)
vcf <- ReadVCFData(mydir, myfile, "GRCh38")
ev <- VCFEvaluate(vcf, vcfparams)
getPlots(ev)
```

getResults *Getter for VCFQAReport class to return results. Return a list showing values that the sample was evaluated on.*

Description

Getter for VCFQAReport class to return results. Return a list showing values that the sample was evaluated on.

Usage

```
getResults(Object)
```

Arguments

Object	an object of type VCFQAReport

Value

numeric vector of results

Examples

```
vcffn <- system.file("ext-data", "chr22.GRCh38.vcf.gz", package="genotypeeval")
mydir <- paste(dirname(vcffn), "/", sep="")
myfile <-basename(vcffn)
svp <- ScanVcfParam(which=GRanges("22", IRanges(0,200e5)), geno="GT")
vcfparams <- VCFQAParam(count.limits=c(3014580000, Inf), readdepth.target = 30)
vcf <- ReadVCFData(mydir, myfile, "GRCh38")
ev <- VCFEvaluate(vcf, vcfparams)
getResults(ev)
```

getVR *getVr is a Getter. Returns vr slot.*

Description

getVr is a Getter. Returns vr slot.

Usage

```
getVR(x)
```

Arguments

x	VCFData object

Value

VRanges

Examples

```
vcffn <- system.file("ext-data", "chr22.GRCh38.vcf.gz", package="genotypeeval")
mydir <- paste(dirname(vcffn), "/", sep="")
myfile <-basename(vcffn)
svp <- ScanVcfParam(which=GRanges("22", IRanges(0,1e5)), geno="GT")
vcf <- ReadVCFData(mydir, myfile, "GRCh38")
getVR(vcf)
```

GoldData-class *Declare class Gold to store information from Gold" (1000 Genomes for example) along with the GoldDataParam*

Description

Declare class Gold to store information from Gold" (1000 Genomes for example) along with the GoldDataParam

Arguments

genome	Genome build, GRCh37 or GRCh38
track	Where the gold data is stored
goldparams	The Param file with the limits to be applied
track.rare	Stores the Gold data with MAF < 0.01 if MAF exists

Value

Object of class GoldData

GoldDataFromGRanges *User Constructor for class. Used to associate the gold params object with the gold granges and to check if MAF is present.*

Description

User Constructor for class. Used to associate the gold params object with the gold granges and to check if MAF is present.

Usage

```
GoldDataFromGRanges(genome, gold.granges, goldparams)
```

Arguments

genome	Genome build, GRCh37 or GRCh38
gold.granges	Gold file as GRanges
goldparams	GoldDataParam object setting thresholds for evaluation

Value

Object of class GoldData

Examples

```
gparam <-  GoldDataParam(percent.confirmed=0.792, percent.het.rare = 0.93)
gr <- GRanges(seqnames="22", IRanges(1e7,5e7))
gold <- GoldDataFromGRanges("GRCh38", gr, gparam)
```

GoldDataParam *User Constructor for class*

Description

User Constructor for class

Usage

```
GoldDataParam(titv.coding.confirmed.l = 0, titv.coding.confirmed.u = 5,
  titv.noncoding.confirmed.l = 0, titv.noncoding.confirmed.u = 5,
  titv.coding.unconfirmed.l = 0, titv.coding.unconfirmed.u = 5,
  titv.noncoding.unconfirmed.l = 0, titv.noncoding.unconfirmed.u = 5,
  percent.confirmed.limits = 0, percent.het.rare.limits = 0)
```

Arguments

titv.coding.confirmed.l	Lower limit of transition transversion ratio in coding confirmed
titv.coding.confirmed.u	upper limit of transition transverion ratio coding confirmed
titv.noncoding.confirmed.l	Lower limit of transition transversion ratio in noncoding confirmed
titv.noncoding.confirmed.u	upper limit of transition transverion ratio noncoding confirmed
titv.coding.unconfirmed.l	Lower limit of transition transversion ratio in coding unconfirmed
titv.coding.unconfirmed.u	upper limit of transition transverion ratio coding unconfirmed
titv.noncoding.unconfirmed.l	Lower limit of transition transversion ratio in noncoding unconfirmed

`titv.noncoding.unconfirmed.u`
: upper limit of transition transverion ratio noncoding unconfirmed

`percent.confirmed.limits`
: lower limit, upper limit, percent confirmed in Gold comparator

`percent.het.rare.limits`
: lower limit, upper limit, (Percent Het in Rare, MAF < 0.01 in Gold) / Total number of Heterozygotes

Value

Object of type GoldDataParam

Examples

```
gparam <- GoldDataParam(percent.confirmed=0.792, percent.het.rare = 0.93)
```

`GoldDataParam-class` *Declare class GoldDataParam which will store thresholds to apply to VCFEvaluate object. This is intended for use in batch mode when a large number of vcf files needs to be screened and individual vcf files that fail flagged. All limits follow the format lower limit than upper limit*

Description

Declare class GoldDataParam which will store thresholds to apply to VCFEvaluate object. This is intended for use in batch mode when a large number of vcf files needs to be screened and individual vcf files that fail flagged. All limits follow the format lower limit than upper limit

Arguments

`titv.confirmed.limits`
: lower limit coding, upper limit coding, lower limit noncoding, upper limit noncoding, Transition transversion ratios for confirmed snps

`titv.unconfirmed.limits`
: lower limit coding, upper limit coding, lower limit noncoding, upper limit noncoding, Transition transversion ratios for unconfirmed snps

`percent.confirmed.limits`
: lower limit, upper limit, percent confirmed in Gold comparator

`percent.het.rare.limits`
: lower limit, upper limit, (Percent Het in Rare, MAF < 0.01 in Gold) / Total number of Heterozygotes

Value

Object of type GoldDataParam

ReadGoldData *User Constructor for class*

Description

User Constructor for class

Usage

```
ReadGoldData(genome, vcffilename, goldparams)
```

Arguments

genome	Genome build, GRCh37 or GRCh38
vcffilename	path and filename of vcf file
goldparams	GoldDataParam object setting thresholds for evaluation

Value

Object of class GoldData

Examples

```
gparam <-  GoldDataParam(percent.confirmed=0.792, percent.het.rare = 0.93)
g1000fn <- system.file("ext-data", "example_gold_file.vcf", package="genotypeeval")
g1000 <- ReadGoldData("GRCh38", g1000fn, gparam)
```

ReadVCFData *User Constructor for class. Calls VCFData constructor: ReadVCFData is a wrapper for readVcfAsVRanges. It removes indels, GL chromosomes, and MULTI calls. It scans the header of the vcf file and adds in the following fields for analysis if present: AD, GT, DP, GQ. Looks for the "END" tag in the header and reads in file as gVCF if necessary.*

Description

User Constructor for class. Calls VCFData constructor: ReadVCFData is a wrapper for readVcfAsVRanges. It removes indels, GL chromosomes, and MULTI calls. It scans the header of the vcf file and adds in the following fields for analysis if present: AD, GT, DP, GQ. Looks for the "END" tag in the header and reads in file as gVCF if necessary.

Usage

```
ReadVCFData(mydir, myfile, genome)
```

Arguments

mydir	Directory of vcf file
myfile	Filename of vcf file
genome	GRCh37 or GRCh38

Value

Object of class VCFData

Examples

```
vcffn <- system.file("ext-data", "chr22.GRCh38.vcf.gz", package="genotypeeval")
mydir <- paste(dirname(vcffn), "/", sep="")
myfile <-basename(vcffn)
vcf <- ReadVCFData(mydir, myfile, "GRCh38")
```

ReadVCFDataChunk

User Constructor for class. Calls VCFData constructor: ReadVCFDataChunk is a wrapper for readVcfAsVRanges. It removes indels, GL chromosomes, and MULTI calls. It scans the header of the vcf file and adds in the following fields for analysis if present: AD, GT, DP, GQ. Looks for the "END" tag in the header and reads in file as gVCF if necessary. This is a multi core version of readVCFData. Note, input file must have been zipped and have a corresponding tabix file. It will drop all hom ref sites not in the admixture file but retain the counts of homref and multi in the VCF file. This means that a few of the metrics and the hom ref plot can no longer be calculated in VCFQAReport. If the metrics can no longer be calculated, it will not be output. Please note that if using a filter on the data (eg gq.filter) this will not be applied to the hom ref and total number of calls. The filter is applied in the VCFQAReport step and the metrics number of hom ref and total number of calls is calculated while reading in the file. When calling this function keep in mind the memory requirements. For example, if numcores=6, then when submitting the job you may request 12 Gb each core (72 Gb total). However the VCF in memory will need to fit back onto a single core or else R will not be able to allocate the memory. The given example here does not make sense to run as it includes only chromosome 22.

Description

User Constructor for class. Calls VCFData constructor: ReadVCFDataChunk is a wrapper for readVcfAsVRanges. It removes indels, GL chromosomes, and MULTI calls. It scans the header of the vcf file and adds in the following fields for analysis if present: AD, GT, DP, GQ. Looks for the "END" tag in the header and reads in file as gVCF if necessary. This is a multi core version of readVCFData. Note, input file must have been zipped and have a corresponding tabix file. It

will drop all hom ref sites not in the admixture file but retain the counts of homref and multi in the VCF file. This means that a few of the metrics and the hom ref plot can no longer be calculated in VCFQAReport. If the metrics can no longer be calculated, it will not be output. Please note that if using a filter on the data (eg gq.filter) this will not be applied to the hom ref and total number of calls. The filter is applied in the VCFQAReport step and the metrics number of hom ref and total number of calls is calculated while reading in the file. When calling this function keep in mind the memory requirements. For example, if numcores=6, then when submitting the job you may request 12 Gb each core (72 Gb total). However the VCF in memory will need to fit back onto a single core or else R will not be able to allocate the memory. The given example here does not make sense to run as it includes only chromosome 22.

Usage

```
ReadVCFDataChunk(mydir, myfile, genome, admixture.ref, numcores)
```

Arguments

mydir	Directory of vcf file
myfile	Filename of vcf file (zipped)
genome	GRCh37 or GRCh38
admixture.ref	VRanges with MAF for superpopulations (EAS, AFR, EUR)
numcores	Number of cores to read in VCF (passed to bplapply)

Value

Object of type VCFData

Examples

```
vcffn <- system.file("ext-data", "chr22.GRCh38.vcf.gz", package="genotypeeval")
mydir <- paste(dirname(vcffn), "/", sep="")
myfile <-basename(vcffn)
svp <- ScanVcfParam(which=GRanges("22", IRanges(0,1e5)), geno="GT")
vcf <- ReadVCFData(mydir, myfile, "GRCh38")
admix.var <- getVR(vcf)[getVR(vcf)$GT %in% c("0|1", "1|0", "1|1"),][,1:2]
admix.var$EAS_AF <- ifelse(admix.var$GT %in% c("1|1"), 1, .5)
admix.var$AFR_AF<- 0
admix.var$EUR_AF<- 0
admix.hom <- getVR(vcf)[getVR(vcf)$GT %in% c("0|0"),][,1:2]
admix.hom$EAS_AF<- 0
admix.hom$AFR_AF<- 1
admix.hom$EUR_AF<- 1
admix.ref <- c(admix.var, admix.hom)
ReadVCFDataChunk(mydir, myfile, "GRCh38", admix.ref, numcores=2)
```

`reformatData` *Take in the results from the population data and re-format it*

Description

Take in the results from the population data and re-format it

Usage

```
reformatData(results)
```

Arguments

`results`	The list of results from running the package using BatchJobs

Value

list, data frame of logical (passed or not), data frame of numeric (all results)

`VCFData-class` *Declare class Reads in VCF using readVCFAsVRanges*

Description

Declare class Reads in VCF using readVCFAsVRanges

Arguments

`mydir`	Directory of vcf file
`myfile`	Filename of vcf file
`vr.homref`	All SNPs from VCF with INDELs, MULTIs (seperately removed for variant and non variant), weird chromosomes removed
`genoString`	A character vector of all genotype fields present (looks for AD, GQ, GT, DP)
`infoString`	A character vector looking for "END" tag indicating file is a gVCF
`genome`	Declare if the genome is GRCh37 or GRCh38
`n.dup`	Counts the number of MULTIs removed
`chunked`	Whether data was read in using ReadVCFDataChunk which means hom refs not in the admixture file were dropped

Value

Object of class VCFData

VCFEvaluate	*Constructor for class. Calls constructor for class. Using the GENO fields present in the vcf header will evaluate the vcf file using metrics and generate plots. Each metric will be tested against the params specified in the params class. For example, if Read Depth is in the GENO header will calculate median read depth, percent in target (50 percent to 200 percent of the target specified in the params file) and generate a histogram of Read Depth.*

Description

Constructor for class. Calls constructor for class. Using the GENO fields present in the vcf header will evaluate the vcf file using metrics and generate plots. Each metric will be tested against the params specified in the params class. For example, if Read Depth is in the GENO header will calculate median read depth, percent in target (50 percent to 200 percent of the target specified in the params file) and generate a histogram of Read Depth.

Usage

```
VCFEvaluate(myvcf, vcfparams, gold.ref = NA, cds.ref = NA,
  masked.ref = NA, admixture.ref = NA)
```

Arguments

myvcf	Vcf file to evaluate
vcfparams	object of VCFQAParam class. Sets thresholds to evaluate the VCF File against.
gold.ref	Object of class Gold that contains the 1000 Genomes reference
cds.ref	Coding Region as GRanges
masked.ref	optional regions as GRanges to mask eg repeats, self chain, paralogs, etc.
admixture.ref	VRanges with MAF for superpopulations (EAS, AFR, EUR)

Value

Object of VCFQAReport.

Examples

```
vcffn <- system.file("ext-data", "chr22.GRCh38.vcf.gz", package="genotypeeval")
mydir <- paste(dirname(vcffn), "/", sep="")
myfile <-basename(vcffn)
svp <- ScanVcfParam(which=GRanges("22", IRanges(0,200e5)), geno="GT")
vcfparams <- VCFQAParam(count.limits=c(3014580000, Inf), readdepth.target = 30)
vcf <- ReadVCFData(mydir, myfile, "GRCh38")
ev <- VCFEvaluate(vcf, vcfparams)
```

VCFQAParam *User Constructor for class. Call limits are set as default to pass.*

Description

User Constructor for class. Call limits are set as default to pass.

Usage

```
VCFQAParam(homref.limits = c(-Inf, Inf), het.limits = c(-Inf, Inf),
  homvar.limits = c(-Inf, Inf), percenthets.limits = c(-Inf, Inf),
  titv.noncoding.limits = c(-Inf, Inf), titv.coding.limits = c(-Inf, Inf),
  readdepth.target = -1, readdepth.limits = c(-Inf, Inf),
  readdepth.percent.limits = 0, gq.limit = 0, masked.limits = c(-Inf,
  Inf), non.masked.limits = c(-Inf, Inf), het.gap.limits = rep(Inf, 24),
  count.limits = c(-Inf, Inf), gq.filter = 0, dp.filter = -1)
```

Arguments

homref.limits	lower limit, upper limit, number of homozygous reference
het.limits	lower limit, upper limit, number of heterozygous calls
homvar.limits	lower limit, upper limit, number of homozygous alternative
percenthets.limits	lower limit, upper limit, Number of Heterozgyous / (Total Number of Counts) or percent het
titv.noncoding.limits	lower limit, upper limit, Transition transversion ratio in noncoding regions
titv.coding.limits	lower limit, upper limit, Transition transversion ratio in coding regions
readdepth.target	The sequencing depth target (eg 30x)
readdepth.limits	lower limit, upper limit, Mean read depth
readdepth.percent.limits	lower limit, upper limit, Percent read depth in target (50 percent to 200 percent of target read depth)
gq.limit	lower limit, Mean genotype quality (does not make sense to have an upper limit)
masked.limits	lower limit, upper limit, (Number of heterozygous in self chained regions)/(Total number of heterozygotes)
non.masked.limits	lower limit, upper limit, (Number of heterozygous in non-self chained regions)/(Total number of heterozygotes)
het.gap.limits	lower limit, upper limit, Largest gap within chromosome between two heterozygous calls

count.limits	lower limit, upper limit, total number of counts
gq.filter	filter for the VCF file on genotype quality (eg only GQ > 90)
dp.filter	filter for the VCF file on read depth (eg only DP > 0)

Value

Object of class VCFQAParam

Examples

```
vcfparams <- VCFQAParam(count.limits=c(3014580000, Inf), readdepth.target = 30)
```

VCFQAParam-class *Declare class VCFQAParam which will store thresholds to apply to VCFEvaluate object. This is intended for use in batch mode when a large number of vcf files needs to be screened and individual vcf files that fail flagged. All limits follow the format lower limit than upper limit*

Description

Declare class VCFQAParam which will store thresholds to apply to VCFEvaluate object. This is intended for use in batch mode when a large number of vcf files needs to be screened and individual vcf files that fail flagged. All limits follow the format lower limit than upper limit

Arguments

homref.limits	lower limit, upper limit, number of homozygous reference
het.limits	lower limit, upper limit, number of heterozygous calls
homvar.limits	lower limit, upper limit, number of homozygous alternative
count.limits	lower limit, upper limit, total number of counts
percenthets.limits	lower limit, upper limit, Number of Heterozgyous / (Total Number of Counts) or percent het
titv.noncoding.limits	lower limit, upper limit, Transition transversion ratio in noncoding regions
titv.coding.limits	lower limit, upper limit, Transition transversion ratio in coding regions
readdepth.target	The sequencing depth target (eg 30x)
readdepth.limits	lower limit, upper limit, Mean read depth
readdepth.percent.limits	lower limit, upper limit, Percent read depth in target (50 percent to 200 percent of target read depth)

gq.limit	lower limit, Mean genotype quality (does not make sense to have an upper limit)
masked.limits	lower limit, upper limit, (Number of heterozygous in masked regions)/(Total number of heterozygotes)
non.masked.limits	lower limit, upper limit, (Number of heterozygous in non-self chained regions)/(Total number of heterozygotes)
het.gap.limits	lower limit, upper limit, Largest gap within chromosome between two heterozygous calls
gq.filter	filter for the VCF file on genotype quality (eg only GQ > 90)
dp.filter	filter for the VCF file on read depth (eg only DP > 0)

Value

VCFQAParam object

VCFQAReport-class *Declare class VCFQAReport which will evaluate a VCF stored as a ReadData object.*

Description

Declare class VCFQAReport which will evaluate a VCF stored as a ReadData object.

Arguments

printnames	List of tests applied to VCF
results	Numeric vector of metrics calculated from VCF file
plots	List of plots created from VCF File
tests	TRUE (passed) or FALSE (failed) logical vector of whether VCF passed metrics using thresholds from VCFQAParam
fn	Filename of VCF evaluated (for plot titles)

Index