Package 'vcfppR'

March 31, 2026

Title Rapid Manipulation of the Variant Call Format (VCF)

Version 0.8.3

Copyright See the file COPYRIGHTS for various htslib copyright details

Description The 'vcfpp.h' (<https://github.com/Zilong-Li/vcfpp>) provides an easy-to-use 'C++' 'API' of 'htslib', offering full functionality for manipulating Variant Call Format (VCF) files. The 'vcfppR' package serves as the R bindings of the 'vcfpp.h' library, enabling rapid processing of both compressed and uncompressed VCF files. Explore a range of powerful features for efficient VCF data manipulation.

Encoding UTF-8

Depends R (>= 3.6.0)

RoxygenNote 7.3.2

Suggests knitr, codetools, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

SystemRequirements libcurl: libcurl-devel (rpm) or libcurl4-openssl-dev (deb), GNU make.

Imports Rcpp, methods, stats, utils

LinkingTo Rcpp

URL https://github.com/Zilong-Li/vcfppR

BugReports https://github.com/Zilong-Li/vcfppR/issues

License MIT + file LICENSE

VignetteBuilder knitr

NeedsCompilation yes

Author Zilong Li [aut, cre] (ORCID: <https://orcid.org/0000-0001-5859-2078>), Bonfield, James K and Marshall, John and Danecek, Petr and Li, Heng and Ohan, Valeriu and Whitwham, Andrew and Keane, Thomas and Davies, Robert M [cph] (Authors of included htslib library)

Maintainer Zilong Li <zilong.dk@gmail.com>

Repository CRAN

Date/Publication 2026-03-31 18:50:02 UTC

Contents

`vcfppR-package` *vcfppR: Rapid Manipulation of the Variant Call Format (VCF)*

Description

The 'vcfpp.h' (https://github.com/Zilong-Li/vcfpp) provides an easy-to-use 'C++' 'API' of 'htslib', offering full functionality for manipulating Variant Call Format (VCF) files. The 'vcfppR' package serves as the R bindings of the 'vcfpp.h' library, enabling rapid processing of both compressed and uncompressed VCF files. Explore a range of powerful features for efficient VCF data manipulation.

Author(s)

Maintainer: Zilong Li <zilong.dk@gmail.com> (ORCID)

Other contributors:

- Bonfield, James K and Marshall, John and Danecek, Petr and Li, Heng and Ohan, Valeriu and Whitwham, Andrew and Keane, Thomas and Davies, Robert M (Authors of included htslib library) [copyright holder]

See Also

Useful links:

- https://github.com/Zilong-Li/vcfppR
- Report bugs at https://github.com/Zilong-Li/vcfppR/issues

`plot_variants_per_haplotype` *Plot variants on haplotypes across multiple samples*

Description

Visualizes variant positions and alleles on both haplotypes for multiple samples. Each sample is represented by two horizontal tracks (one per haplotype), with variants colored according to their type (SNP, insertion, deletion) and allele (reference or alternate). Large gaps between variants can be automatically compressed for better visualization.

Usage

```
plot_variants_per_haplotype(
  vcffiles,
  region,
  types = c("SNP", "DEL", "INS"),
  shrink_threshold = 1000,
  xlab = "Genomic position",
  ylab = "Haplotypes of each sample",
  main = NULL,
  ...
)
```

Arguments

`vcffiles`	Character vector of VCF/BCF file paths or URLs. Each file represents one sample.
`region`	Character string specifying the genomic region to visualize (e.g., "chr1:1000-5000").
`types`	Character vector of variant types to include in the plot. Valid options are "SNP" (single nucleotide polymorphisms), "DEL" (deletions), and "INS" (insertions). Default: c("SNP", "DEL", "INS").
`shrink_threshold`	Numeric value specifying the minimum gap size (in base pairs) between variants that will trigger compression. Gaps larger than this threshold are shrunk to improve visualization density. Default: 1000.
`xlab`	Character string for the x-axis label. Default: "Genomic position".
`ylab`	Character string for the y-axis label. Default: "Haplotypes of each sample".
`main`	Character string for the plot title. Default: NULL (no title).
`...`	Additional graphical parameters passed to the base plot function.

Details

The function reads variant data from multiple VCF files using `vcftable` with `collapse=FALSE` to preserve haplotype phasing information. Each sample is displayed as two horizontal tracks representing the two haplotypes (h1 and h2).

Variant types are distinguished by color:

- SNPs: Green (reference allele) or Yellow (alternate allele)
- Deletions: Light blue (reference allele only)
- Insertions: Dark blue (reference allele) or Dark orange (alternate allele)

When large gaps exist between variants (exceeding `shrink_threshold`), the function compresses these regions and marks them with black dashed lines and "..." text to indicate the compression. This feature helps visualize sparse variant distributions more effectively.

Value

Invisibly returns NULL. The function is called for its side effect of creating a plot.

See Also

`vcftable`, `vcfplot`

Examples

```
## Not run:
# Plot variants from three samples in a specific region
vcf_files <- c("sample1.vcf.gz", "sample2.vcf.gz", "sample3.vcf.gz")
plot_variants_per_haplotype(vcf_files, region = "chr20:1000000-1100000")

# Plot only SNPs and insertions with custom threshold
plot_variants_per_haplotype(vcf_files,
                            region = "chr20:1000000-1100000",
                            types = c("SNP", "INS"),
                            shrink_threshold = 5000)

# Customize plot appearance
plot_variants_per_haplotype(vcf_files,
                            region = "chr20:1000000-1100000",
                            main = "Variant Distribution",
                            xlab = "Position (bp)",
                            cex.axis = 0.8)

## End(Not run)
```

subset.vcftable *Subset a vcftable object*

Description

S3 method for subsetting vcftable objects by rows (variants) and columns (fields). Allows filtering variants based on logical conditions and selecting specific fields.

Usage

```
## S3 method for class 'vcftable'
subset(x, subset, select, drop = FALSE, ...)
```

Arguments

x	a vcftable object returned by vcftable
subset	logical expression indicating variants (rows) to keep. The expression is evaluated in the context of the vcftable object, allowing direct reference to fields like chr, pos, ref, alt, qual, etc. Missing values are treated as FALSE.
select	expression indicating which fields (columns) to select. If omitted, all fields except samples are selected. Note: the samples field is always kept and cannot be selected/deselected.
drop	logical. If TRUE, the result is coerced to the lowest possible dimension. Passed to the [operator when subsetting. Default FALSE.
...	Currently not used but can avoid S3 generic consistency warnings

Value

A vcftable object with the selected variants and fields.

Author(s)

Zilong Li <zilong.dk@gmail.com>

Examples

```
library('vcfppR')
vcffile <- system.file("extdata", "raw.gt.vcf.gz", package="vcfppR")
res <- vcftable(vcffile, "chr21:1-5050000")

# Subset by quality score
high_qual <- subset(res, qual > 100)

# Subset by position and select specific fields
region_subset <- subset(res, pos >= 5000000 & pos <= 5010000,
                        select = c(chr, pos, ref, alt))
```

```
region_subset <- subset(res, pos >= 5000000 & pos <= 5030400,
                        select = c(chr, pos, ref, alt))

# Subset SNPs (REF and ALT are single nucleotides)
snps <- subset(res, nchar(ref) == 1 & nchar(alt) == 1)
```

---

vcfcomp *Compare two VCF/BCF files reporting various statistics*

Description

Compare two VCF/BCF files reporting various statistics

Usage

```
vcfcomp(
  test,
  truth,
  formats = c("DS", "GT"),
  stats = "r2",
  by.sample = FALSE,
  by.variant = FALSE,
  flip = FALSE,
  names = NULL,
  bins = NULL,
  af = NULL,
  out = NULL,
  choose_random_start = FALSE,
  return_pse_sites = FALSE,
  ...
)
```

Arguments

test	path to the comparison file (test), which can be a VCF/BCF file, vcftable object or saved RDS file.
truth	path to the baseline file (truth), which can be a VCF/BCF file, vcftable object or saved RDS file.
formats	character vector. the FORMAT tags to extract for the test and truth respectively. default c("DS", "GT") extracts 'DS' of the test and 'GT' of the truth.
stats	character. the statistics to be calculated. Supports the following options: **"r2"** the Pearson correlation coefficient squared (default) **"f1"** the F1-score, good balance between sensitivity and precision **"nrc"** the Non-Reference Concordance rate **"pse"** the Phasing Switch Error rate

"all" calculate r2, f1, and nrc together

"gtgq" genotype quality-based concordance analysis

"gtdp" depth-based concordance analysis

`by.sample` logical. calculate sample-wise concordance, which can be stratified by MAF bin.

`by.variant` logical. calculate variant-wise concordance, which can be stratified by MAF bin. If both by.sample and by.variant are FALSE, then do calculations for all samples and variants together in a bin.

`flip` logical. flip the ref and alt variants

`names` character vector. reset samples' names in the test VCF.

`bins` numeric vector. break statistics into allele frequency bins. If NULL (default), bins are automatically generated with fine resolution for rare variants and coarser resolution for common variants (ranging from 0 to 0.5).

`af` file path with allele frequency or a RDS file with a saved object for af. Format of the text file: a space-separated text file with five columns and a header named 'chr' 'pos' 'ref' 'alt' 'af'. If NULL, allele frequencies are calculated from the truth genotypes.

`out` output prefix for saving objects into RDS file. If provided, creates three files: out.af.rds, out.test.rds, and out.truth.rds

`choose_random_start` logical. choose random start for stats="pse". Defaults to FALSE.

`return_pse_sites` logical. return phasing switch error sites when stats="pse". Defaults to FALSE.

`...` additional options passed to `vcftable`, such as 'samples', 'region', or 'pass'.

Details

`vcfcomp` implements various statistics to compare two VCF/BCF files, e.g. report genotype concordance, correlation stratified by allele frequency.

Value

a list object of class "vcfcomp" containing:

samples character vector of sample names

stats the calculated statistics, named according to the 'stats' parameter. For stats="all", returns r2, f1, and nrc components.

Author(s)

Zilong Li <zilong.dk@gmail.com>

Examples

```
library('vcfppR')
# site-wise comparision stratified by allele frequency
test <- system.file("extdata", "imputed.gt.vcf.gz", package="vcfppR")
truth <- system.file("extdata", "raw.gt.vcf.gz", package="vcfppR")
samples <- "HG00673,NA10840"
res <- vcfcomp(test, truth, stats="r2", bins=c(0,1), samples=samples, setid=TRUE)
str(res)

# sample-wise comparision stratified by sample-level metrice e.g GQ
test <- system.file("extdata", "svupp.call.vcf.gz", package="vcfppR")
truth <- system.file("extdata", "platinum.sv.vcf.gz", package="vcfppR")
res <- vcfcomp(test, truth, stats = "gtgq", region = "chr1")
str(res)
```

`vcfinfo` *read a INFO tag in the VCF/BCF into R data structure*

Description

read a INFO tag in the VCF/BCF into R data structure

Usage

```
vcfinfo(
  vcffile,
  tag,
  region = "",
  vartype = "all",
  ids = NULL,
  qual = 0,
  pass = FALSE,
  setid = FALSE
)
```

Arguments

`vcffile`	path to the VCF/BCF file
`tag`	the INFO tag to extract.
`region`	region to subset in bcftools-like style: "chr1", "chr1:1-10000000"
`vartype`	restrict to specific type of variants. supports "snps","indels", "sv", "multisnps","multiallelics"
`ids`	character vector. restrict to sites with ID in the given vector. default NULL won't filter any sites.
`qual`	numeric. restrict to variants with QUAL > qual.
`pass`	logical. restrict to variants with FILTER = "PASS".
`setid`	logical. reset ID column as CHR_POS_REF_ALT.

Details

vcfinfo uses the C++ API of vcfpp, which is a wrapper of htslib, to read VCF/BCF files. Thus, it has the full functionalities of htslib, such as restrict to specific variant types, samples and regions. For the memory efficiency reason, the vcfinfo is designed to parse only one tag at a time in the INFO column of the VCF. Currently it does not support parsing a vector of values for a given INFO tag.

Value

Return a list containing the following components:

chr : character vector;
the CHR column in the VCF file

pos : character vector;
the POS column in the VCF file

id : character vector;
the ID column in the VCF file

ref : character vector;
the REF column in the VCF file

alt : character vector;
the ALT column in the VCF file

qual : character vector;
the QUAL column in the VCF file

filter : character vector;
the FILTER column in the VCF file

tag : vector of either integer, numberic or character values depending on the tag to extract;
a specifiy tag in the INFO column to be extracted

Author(s)

Zilong Li <zilong.dk@gmail.com>

Examples

```
library('vcfppR')
vcffile <- system.file("extdata", "raw.gt.vcf.gz", package="vcfppR")
res <- vcfinfo(vcffile, "AF", region = "chr21:1-5050000", vartype = "snps", pass = TRUE)
str(res)
```

vcfplot *Make sensible and beautiful plots based on various objects in vcfppR*

Description

Make sensible and beautiful plots based on various objects in vcfppR

Usage

```
vcfplot(
  obj,
  which.sample = NULL,
  which.format = 10,
  variant = c("SNP", "INDEL"),
  pop = NULL,
  ...
)
```

Arguments

obj	object returned by vcftable, vcfcomp, vcfsummary
which.sample	which sample to be plotted. NULL will aggregate all samples.
which.format	which FORMAT field to be plotted. Defaults will use the 10-th names.
variant	which types of variant are desired
pop	file contains population information
...	parameters passed to graphics

vcfpopgen *count the heterozygous sites per sample in the VCF/BCF*

Description

count the heterozygous sites per sample in the VCF/BCF

Usage

```
vcfpopgen(
  vcffile,
  region = "",
  samples = "-",
  pass = FALSE,
  qual = 0,
  fun = "heterozygosity"
)
```

Arguments

vcffile	path to the VCF/BCF file
region	region to subset like bcftools
samples	samples to subset like bcftools
pass	restrict to variants with FILTER==PASS
qual	restrict to variants with QUAL > qual.
fun	which popgen function to run. available functions are "heterozygosity".

Value

vcfpopgen a list containing the following components:

samples : character vector;
the samples ids in the VCF file after subsetting

hets : integer vector;
the counts of heterozygous sites of each sample in the same order as samples

Author(s)

Zilong Li <zilong.dk@gmail.com>

Examples

```
library('vcfppR')
vcffile <- system.file("extdata", "raw.gt.vcf.gz", package="vcfppR")
res <- vcfpopgen(vcffile)
str(res)
```

---

vcfpp_calc_info_persite

Calculate INFO score from GP after genotype imputation

Description

Calculate INFO score from GP after genotype imputation

Usage

```
vcfpp_calc_info_persite(GP)
```

Arguments

GP	vector of length a multiple of 3

vcfreader *API for manipulating the VCF/BCF.*

Description

Type the name of the class to see the details and methods

Value

A C++ class with the following fields/methods for manipulating the VCF/BCF

Fields

`new` Constructor given a vcf file

- Parameter: vcffile - The path of a vcf file

`new` Constructor given a vcf file and the region

- Parameter: vcffile - The path of a vcf file
- Parameter: region - The region to be constrained

`new` Constructor given a vcf file, the region and the samples

- Parameter: vcffile - The path of a vcf file
- Parameter: region - The region to be constrained
- Parameter: samples - The samples to be constrained. Comma separated list of samples to include (or exclude with "^" prefix).

`setRegion` try to set specific region to work with. will throw errors if no index or region found. Use getStatus to check if the region is valid or empty!

`getStatus` return 1: region is valid and not empty. 0: region is valid but empty. -1: no index file. -2: region not found or invalid region form

`variant` Try to get next variant record. return FALSE if there are no more variants or hit the end of file, otherwise TRUE.

`chr` Return the CHROM field of current variant

`pos` Return the POS field of current variant

`id` Return the CHROM field of current variant

`ref` Return the REF field of current variant

`alt` Return the ALT field of current variant

`qual` Return the QUAL field of current variant

`filter` Return the FILTER field of current variant

`info` Return the INFO field of current variant

`infoInt` Return the tag value of integer type in INFO field of current variant

- Parameter: tag - The tag name to retrieve in INFO

`infoFloat` Return the tag value of float type in INFO field of current variant

- Parameter: tag - The tag name to retrieve in INFO

`infoStr` Return the tag value of string type in INFO field of current variant

- Parameter: tag - The tag name to retrieve in INFO

`infoIntVec` Return the tag value in a vector of integer type in INFO field of current variant

- Parameter: tag - The tag name to retrieve in INFO

`infoFloatVec` Return the tag value in a vector of float type in INFO field of current variant

- Parameter: tag - The tag name to retrieve in INFO

`genotypes` Return the genotype values in a vector of integers

- Parameter: collapse - Boolean value indicates wheather to collapse the size of genotypes, eg, return diploid genotypes.

`formatInt` Return the tag value of integer type for each sample in FORAMT field of current variant

- Parameter: tag - The tag name to retrieve in FORAMT

`formatFloat` Return the tag value of float type for each sample in FORAMT field of current variant

- Parameter: tag - The tag name to retrieve in FORAMT

`formatStr` Return the tag value of string type for each sample in FORAMT field of current variant

- Parameter: tag - The tag name to retrieve in FORAMT

`isSNP` Test if current variant is exculsively a SNP or not

`isIndel` Test if current variant is exculsively a INDEL or not

`isSV` Test if current variant is exculsively a SV or not

`isMultiAllelics` Test if current variant is exculsively a Multi Allelics or not

`isMultiAllelicSNP` Test if current variant is exculsively a Multi Biallelics (SNPs) or not

`hasSNP` Test if current variant has a SNP or not

`hasINDEL` Test if current variant has a INDEL or not

`hasINS` Test if current variant has a INS or not

`hasDEL` Test if current variant has a DEL or not

`hasMNP` Test if current variant has a MNP or not

`hasBND` Test if current variant has a BND or not

`hasOTHER` Test if current variant has a OTHER or not

`hasOVERLAP` Test if current variant has a OVERLAP or not

`nsamples` Return the number of samples

`samples` Return a vector of samples id

`header` Return the raw string of the vcf header

`string` Return the raw string of current variant including newline

`line` Return the raw string of current variant without newline

`output` Init an output object for streaming out the variants to another vcf

`updateSamples` update samples name in the output VCF

- Parameter: s - A comma-seperated string for new samples names

`write` Streaming out current variant the output vcf

`close` Close the connection to the output vcf

`setCHR` Modify the CHR of current variant

- Parameter: s - A string for CHR

`setID` Modify the ID of current variant

- Parameter: s - A string for ID

`setPOS` Modify the POS of current variant

- Parameter: pos - An integer for POS

`setRefAlt` Modify the REF and ALT of current variant

- Parameter: s - A string reperated by comma

`setInfoInt` Modify the given tag of INT type in the INFO of current variant

- Parameter: tag - A string for the tag name
- Parameter: v - An integer for the tag value

`setInfoFloat` Modify the given tag of FLOAT type in the INFO of current variant

- Parameter: tag - A string for the tag name
- Parameter: v - A double for the tag value

`setInfoStr` Modify the given tag of STRING type in the INFO of current variant

- Parameter: tag - A string for the tag name
- Parameter: s - A string for the tag value

`setPhasing` Modify the phasing status of each sample

- Parameter: v - An integer vector with size of the number of samples. only 1s and 0s are valid.

`setGenotypes` Modify the genotypes of current variant

- Parameter: v - An integer vector for genotypes. Use NA or -9 for missing value.

`setFormatInt` Modify the given tag of INT type in the FORMAT of current variant

- Parameter: tag - A string for the tag name
- Parameter: v - An integer for the tag value

`setFormatFloat` Modify the given tag of FLOAT type in the FORMAT of current variant

- Parameter: tag - A string for the tag name
- Parameter: v - A double for the tag value

`setFormatStr` Modify the given tag of STRING type in the FORMAT of current variant

- Parameter: tag - A string for the tag name
- Parameter: s - A string for the tag value

`rmInfoTag` Remove the given tag from the INFO of current variant

- Parameter: s - A string for the tag name

`clearInfo` Remove all INFO tags from the current variant, making INFO column empty

`rmFormatTag` Remove the given tag from the FORMAT of current variant

- Parameter: s - A string for the tag name

`setVariant` Modify current variant by adding a vcf line

- Parameter: s - A string for one line in the VCF

`addINFO` Add a INFO in the header of the vcf

- Parameter: id - A string for the tag name
- Parameter: number - A string for the number
- Parameter: type - A string for the type
- Parameter: desc - A string for description of what it means

`addFORMAT` Add a FORMAT in the header of the vcf

- Parameter: id - A string for the tag name
- Parameter: number - A string for the number
- Parameter: type - A string for the type
- Parameter: desc - A string for description of what it means

Examples

```
vcffile <- system.file("extdata", "raw.gt.vcf.gz", package="vcfppR")
br <- vcfreader$new(vcffile)
res <- rep(0L, br$nsamples())
while(br$variant()) {
  if(br$isSNP()) {
  gt <- br$genotypes(TRUE) == 1
  gt[is.na(gt)] <- FALSE
  res <- res + gt
  }
}
```

`vcfsummary` *summarize the various variant types at both variant level and sample level.*

Description

summarize the various variant types at both variant level and sample level.

Usage

```
vcfsummary(
  vcffile,
  region = "",
  samples = "-",
  pass = FALSE,
  qual = 0,
  svtype = FALSE
)
```

Arguments

vcffile	path to the VCF/BCF file
region	region to subset like bcftools
samples	samples to subset like bcftools
pass	restrict to variants with FILTER==PASS
qual	restrict to variants with QUAL > qual.
svtype	summarize the variants with SVTYPE

Details

bcftools view -s "id01,id02" input.bcf.gz chr1:100000-20000

Value

vcfsummary a list containing the following components:

summary : named integer vector;
summarize the counts of each variant type

samples : character vector;
the samples ids in the VCF file after subsetting

vartype : integer vector;
the counts of the variant type at sample level in the same order as samples

Author(s)

Zilong Li <zilong.dk@gmail.com>

Examples

```
library('vcfppR')
svfile <- system.file("extdata", "sv.vcf.gz", package="vcfppR")
res <- vcfsummary(svfile, region = "chr21:1-10000000", svtype = TRUE)
str(res)
```

---

vcftable *read VCF/BCF contents into R data structure*

Description

The swiss army knife for reading VCF/BCF into R data types rapidly and easily.

Usage

```
vcftable(
  vcffile,
  region = "",
  samples = "-",
  vartype = "all",
  format = "GT",
  ids = NULL,
  qual = 0,
  pass = FALSE,
  info = TRUE,
  collapse = TRUE,
  setid = FALSE,
  mac = 0,
  rmdup = FALSE
)
```

Arguments

vcffile	path to the VCF/BCF file
region	region to subset in bcftools-like style: "chr1", "chr1:1-10000000"
samples	samples to subset in bcftools-like style. comma separated list of samples to include (or exclude with "^" prefix). e.g. "id01,id02", "^id01,id02".
vartype	restrict to specific type of variants. supports "snps","indels", "sv", "multisnps","multiallelics"
format	the FORMAT tag to extract. default "GT" is extracted.
ids	character vector. restrict to sites with ID in the given vector. default NULL won't filter any sites.
qual	numeric. restrict to variants with QUAL > qual.
pass	logical. restrict to variants with FILTER = "PASS".
info	logical. drop INFO column in the returned list.
collapse	logical. It acts on the FORMAT. If the FORMAT to extract is "GT", the dim of raw genotypes matrix of diploid is (M, 2 * N), where M is #markers and N is #samples. default TRUE will collapse the genotypes for each sample such that the matrix is (M, N). Set this to FALSE if one wants to maintain the phasing order, e.g. "1|0" is parsed as c(1, 0) with collapse=FALSE. If the FORMAT to extract is not "GT", then with collapse=TRUE it will try to turn a list of the extracted vector into a matrix. However, this raises issues when one variant is mutliallelic resulting in more vaules than others.
setid	logical. reset ID column as CHR_POS_REF_ALT.
mac	integer. restrict to variants with minor allele count higher than the value.
rmdup	logical. remove duplicated sites by keeping the first occurrence of POS. (default: FALSE)

Details

`vcftable` uses the C++ API of vcfpp, which is a wrapper of htslib, to read VCF/BCF files. Thus, it has the full functionalities of htslib, such as restrict to specific variant types, samples and regions. For the memory efficiency reason, the `vcftable` is designed to parse only one tag at a time in the FORMAT column of the VCF. In default, only the matrix of genotypes, i.e. "GT" tag, are returned by `vcftable`, but there are many other tags supported by the `format` option.

Value

Return a list containing the following components:

samples : character vector;
the samples ids in the VCF file after subsetting

chr : character vector;
the CHR column in the VCF file

pos : character vector;
the POS column in the VCF file

id : character vector;
the ID column in the VCF file

ref : character vector;
the REF column in the VCF file

alt : character vector;
the ALT column in the VCF file

qual : character vector;
the QUAL column in the VCF file

filter : character vector;
the FILTER column in the VCF file

info : character vector;
the INFO column in the VCF file

format : matrix of either integer or numberic values depending on the tag to extract;
a specifiy tag in the FORMAT column to be extracted

Author(s)

Zilong Li <zilong.dk@gmail.com>

Examples

```
library('vcfppR')
vcffile <- system.file("extdata", "raw.gt.vcf.gz", package="vcfppR")
res <- vcftable(vcffile, "chr21:1-5050000", vartype = "snps")
str(res)
```

`vcfwriter` *API for writing the VCF/BCF.*

Description

Type the name of the class to see the details and methods

Value

A C++ class with the following fields/methods for writing the VCF/BCF

Fields

`new` Constructor given a vcf file

- Parameter: vcffile - The path of a vcf file. don't start with "~"
- Parameter: version - The version of VCF specification

`addContig` Add a Contig in the header of the vcf

- Parameter: str - A string for the CONTIG name

`addFILTER` Add a FILTER in the header of the vcf

- Parameter: id - A string for the FILTER name
- Parameter: desc - A string for description of what it means

`addINFO` Add a INFO in the header of the vcf

- Parameter: id - A string for the tag name
- Parameter: number - A string for the number
- Parameter: type - A string for the type
- Parameter: desc - A string for description of what it means

`addFORMAT` Add a FORMAT in the header of the vcf

- Parameter: id - A string for the tag name
- Parameter: number - A string for the number
- Parameter: type - A string for the type
- Parameter: desc - A string for description of what it means

`addSample` Add a SAMPLE in the header of the vcf

- Parameter: str - A string for a SAMPLE name

`addLine` Add a line in the header of the vcf

- Parameter: str - A string for a line in the header of VCF

`writeline` Write a variant record given a line

- Parameter: line - A string for a line in the variant of VCF. Not ended with "newline"

`close` Close and save the vcf file

Examples

```
outvcf <- file.path(paste0(tempfile(), ".vcf.gz"))
bw <- vcfwriter$new(outvcf, "VCF4.1")
bw$addContig("chr20")
bw$addFORMAT("GT", "1", "String", "Genotype");
bw$addSample("NA12878")
s1 <- "chr20\t2006060\t.\tG\tC\t100\tPASS\t.\tGT\t1|0"
bw$writeline(s1)
bw$close()
```

Index