Package: VCFArrow (via r-universe)

October 16, 2026

Title Fast and Memory-Efficient Manipulation and Conversion of VCF Files

Version 0.9.0

Description Reads Variant Call Format (VCF) files into an S4 object whose genotypes are stored on disk as an 'Apache Arrow' dataset and loaded lazily, allowing large datasets to be processed with little memory. Provides functions to filter variants and samples, subset and merge datasets, compute summary statistics and diagnostic plots, write VCF files, and convert data to input formats of population genetic and phylogenetic software such as 'STRUCTURE', 'ADMIXTURE', 'PLINK', 'EIGENSTRAT', 'Arlequin', 'GENEPOP', 'BayeScan', 'TreeMix', 'SNAPP' and 'adegenet'.

URL https://legallab.github.io/VCFArrow/, https://github.com/legalLab/VCFArrow

BugReports https://github.com/legalLab/VCFArrow/issues

Language en-US

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 4.1.0)

LinkingTo Rcpp

Imports Rcpp, arrow, cli, dplyr, ggplot2, methods, parallel, stats, stringr, tibble, utils, vctrs

Suggests adegenet, knitr, rmarkdown, svglite

VignetteBuilder knitr

SystemRequirements zlib

Config/roxygen2/version 8.1.0

NeedsCompilation yes

Author Tomas Hrbek [aut, cre, cph] (ORCID: <https://orcid.org/0000-0003-3239-7068>)

Maintainer Tomas Hrbek <hrbek@evoamazon.net>

Config/pak/sysreqs zlib1g-dev

Repository https://cran.r-universe.dev

Date/Publication 2026-10-16 13:20:38 UTC

RemoteUrl https://github.com/cran/VCFArrow

RemoteRef HEAD

RemoteSha f11404c41aadaedfe434c7c8582719203849669a

Contents

[,VCFArrow-method . . . 3
assess_vcf_coverage . . . 4
assess_vcf_missing_data . . . 5
read_vcf . . . 6
set_vcf_groups . . . 7
show,VCFArrow-method . . . 8
vcf_bind . . . 8
vcf_copy . . . 10
vcf_extract_groups . . . 11
vcf_extract_samples . . . 12
vcf_filter_adr . . . 13
vcf_filter_biallelic . . . 14
vcf_filter_coverage . . . 15
vcf_filter_hets . . . 16
vcf_filter_indels . . . 17
vcf_filter_invariant . . . 17
vcf_filter_maf . . . 18
vcf_filter_missing . . . 19
vcf_filter_missingness . . . 20
vcf_filter_multiSNV . . . 21
vcf_filter_oneSNV . . . 22
vcf_filter_paralogs . . . 23
vcf_filter_pass . . . 24
vcf_filter_quality . . . 25
vcf_filter_rank . . . 26
vcf_filter_rpt . . . 27
vcf_gc . . . 28
vcf_memory_estimate . . . 29
vcf_set_workers . . . 30
vcf_stats . . . 31
vcf_sub_SNVs . . . 32
vcf_sub_SNVs_stratified . . . 33
vcf_suggest_chunk_size . . . 34
vcf_theta . . . 34
vcf2admixture . . . 35
vcf2apparent . . . 37

vcf2arlequin 38
vcf2bayesass 39
vcf2bayescan 40
vcf2eigenstrat 41
vcf2fasta 42
vcf2fineradstructure 43
vcf2genepop 44
vcf2genlight 45
vcf2gt_long 46
vcf2migrate 47
vcf2nexus 48
vcf2plink_bed 49
vcf2plink_ped 50
vcf2related 51
vcf2smartsnp 52
vcf2snapp 53
vcf2snmf 54
vcf2structure 55
vcf2treemix 56
VCFArrow-class 57
write_vcf 57

Index **59**

`[,VCFArrow-method` *Subset method for VCFArrow*

Description

Subset a VCFArrow object by variants (rows) and samples (columns)

Usage

```
## S4 method for signature 'VCFArrow'
x[i, j, ..., drop = FALSE]
```

Arguments

`x`	A VCFArrow object
`i`	Variant (row) positions, numeric or logical
`j`	Column indices: numeric, logical, or sample name character vector
`...`	Ignored
`drop`	Ignored; kept for S4 compatibility

Details

This function is a method of the VCFArrow S4 class Method to subset by row and column of GT

Value

A new VCFArrow object containing the selected variants and samples

Author(s)

Tomas Hrbek April 2026

Examples

```
f <- system.file("extdata", "vaillantii_discosnp_sub.vcf.gz",
                 package = "VCFArrow")
vcf <- read_vcf(f)
vcf[1:100, 1:5]
```

---

assess_vcf_coverage *assess_vcf_coverage*

Description

Quantifying read depth of all samples in VCF

Usage

```
assess_vcf_coverage(
  vcf_arrow,
  res_path,
  species,
  project,
  details = TRUE,
  max_points_per_sample = 5000L
)
```

Arguments

`vcf_arrow`	-> VCFArrow object
`res_path`	-> path to results (directory for output dataframe and plots)
`species`	-> sample name for plot (character)
`project`	-> project name / base output file name (character)
`details`	-> flag for adding project name into figure title, default TRUE (Boolean)
`max_points_per_sample`	-> maximum number of SNVs from which to plot coverage, default 5000 (integer)

Details

This function generates a violin plot of read depths per sample.

Value

violin plot of read depth for each sample in VCF

Author(s)

Tomas Hrbek April 2026

Examples

```
f <- system.file("extdata", "vaillantii_discosnp_sub.vcf.gz",
                 package = "VCFArrow")
vcf <- read_vcf(f) |> set_vcf_groups(dirname(f))
assess_vcf_coverage(vcf, res_path = tempdir(),
                    species = "Phyllomedusa vaillantii",
                    project = "vaillantii")
```

---

assess_vcf_missing_data

assess_vcf_missing_data

Description

Quantifying missing data of all samples in VCF

Usage

```
assess_vcf_missing_data(vcf_arrow, res_path, species, project, details = TRUE)
```

Arguments

`vcf_arrow`	-> VCFArrow object
`res_path`	-> path to results (directory for output dataframe and plots)
`species`	-> sample name for plot (character)
`project`	-> project name / base output file name (character)
`details`	-> flag for adding project name into figure title, default TRUE (Boolean)

Details

This function generates a dataframe of absolute and relative missing data per sample, and a plot of relative missing data % per sample.

Value

dataframe and plot of missing data for each sample in VCF

Author(s)

Tomas Hrbek April 2026

Examples

```
f <- system.file("extdata", "vaillantii_discosnp_sub.vcf.gz",
                 package = "VCFArrow")
vcf <- read_vcf(f) |> set_vcf_groups(dirname(f))
assess_vcf_missing_data(vcf, res_path = tempdir(),
                        species = "Phyllomedusa vaillantii",
                        project = "vaillantii")
```

---

read_vcf *read_vcf*

Description

Read VCF file and store its content within a VCFArrow object

Usage

```
read_vcf(vcf_file, chunk_size = 50000)
```

Arguments

vcf_file	-> VCF file
chunk_size	-> number of variants to read in at a time, default 50000 (integer)

Details

This function read a VCF file into an S4 class object in chunks, returning a VCFArrow object. It accepts both uncompressed and gz compressed files. The GT field is stored as an Apache Arrow in Long format. Various metrics are precalculated for fast and easy filtering. GT slot content stored in a TEMP directory for lazy loading.

Value

VCFArrow object

Author(s)

Tomas Hrbek April 2026

Examples

```
f <- system.file("extdata", "vaillantii_discosnp_sub.vcf.gz",
                 package = "VCFArrow")
vcf <- read_vcf(f)
vcf
```

---

set_vcf_groups *set_vcf_groups*

Description

Associate samples with group info based on "strt" file

Usage

```
set_vcf_groups(vcf_arrow, data_path, strt = "strata")
```

Arguments

vcf_arrow	-> VCFArrow object
data_path	-> path to data (where strata are located)
strt	-> file with 2+ columns, one id column and one pop column (tsv file, header = TRUE)

Details

This function creates list of factors of sample-to-group assignments and groups based on a strata and group file. The strt is a tsv file with 2+ columns, with 2 columns named 'id' and 'pop'. By default strt is named 'strata' and found in the datapath where the VCF file is found.

Value

vcf_arrow object with groups slot filled

Author(s)

Tomas Hrbek April 2026

Examples

```
f <- system.file("extdata", "vaillantii_discosnp_sub.vcf.gz",
                 package = "VCFArrow")
vcf <- set_vcf_groups(read_vcf(f), data_path = dirname(f),
                      strt = "strata")
table(vcf@groups)
```

show,VCFArrow-method *Show method for VCFArrow*

Description

Print a summary of a VCFArrow object

Usage

```
## S4 method for signature 'VCFArrow'
show(object)
```

Arguments

object	A VCFArrow object

Details

This function is a method of the VCFArrow S4 class Method to show object content summary

Value

Invisibly returns `object`; called for its side effect of printing a summary.

Author(s)

Tomas Hrbek April 2026

Examples

```
f <- system.file("extdata", "vaillantii_discosnp_sub.vcf.gz",
                 package = "VCFArrow")
vcf <- read_vcf(f)
show(vcf)
```

vcf_bind *vcf_bind*

Description

Bind two or more VCFArrow objects into a new VCFArrow object

Usage

```
vcf_bind(
  ...,
  mode = c("intersect", "union"),
  absent_as = c("missing", "hom_ref"),
  outgroup = FALSE,
  recover_loci = FALSE,
  .check = NULL
)
```

Arguments

`...`	-> a collection of VCFArrow objects
`mode`	-> "intersect" keeps variants present in all objects, "union" keeps variants present in any object, default "intersect" (character)
`absent_as`	-> for mode = "union" or outgroup = TRUE, how genotypes absent from a source object are coded, "missing" or "hom_ref", default "missing" (character)
`outgroup`	-> add the other objects (e.g. outgroups) to the first object (e.g. a filtered ingroup), keeping exactly the variants of the first object; mode and recover_loci are then not used, default FALSE (Boolean)
`recover_loci`	-> for mode = "union", restore the genotypes of variants an object removed by filtering, where its genotype files still hold them, instead of coding them according to absent_as, default FALSE (Boolean)
`.check`	-> deprecated and ignored; objects no longer need to share the same variants

Details

This function binds two or more VCFArrow objects with different sets of individuals, returning a new VCFArrow object. Sample names must be unique across objects. The objects need not have the same variants: mode = "intersect" keeps the variants present in all objects, mode = "union" keeps the variants present in any object and codes the genotypes an object lacks according to absent_as. For objects with the same variants both modes give the same result. In mode = "union", variants an object removed stay removed for it: a variant that one object removed by filtering but that another object contains is included, with the genotypes of the object that removed it coded according to absent_as. Binding in steps and binding all objects in one call then give the same result. With recover_loci = TRUE, the real genotypes of such variants are restored instead, read from the genotype files of the object that removed them (which keep every variant the object had when its files were written). This suits joining separately filtered datasets; binding in steps can then differ from binding in one call.

outgroup = TRUE is meant for adding outgroups to an ingroup that was filtered on its own (so that properties of the outgroup, such as its missingness, cannot remove ingroup variants): the result keeps exactly the variants of the first object, and variants removed from it stay removed. The other objects contribute their genotypes for these variants, read from their genotype files wherever present (including variants they removed, for example as invariant); genotypes they do not have are coded according to absent_as.

Variants are matched by CHROM, POS, REF and ALT, so the objects may come from different VCF files (e.g. separate sequencing runs); REF and ALT must be written the same way in all

of them (including the order of alternative alleles). This is assured when all datasets were called against the same reference genome assembly, which gives the same CHROM, POS and REF for the same locus. It is generally not possible for de novo (reference-free) SNP calling, for example with DiscoSnp, where locus names and positions are generated anew in each calling run, so the same locus will usually not match between separately called datasets; such datasets should be called together in a single run instead. Multi-allelic sites can list their ALT alleles in a different order in different files, so it is best to normalize the VCFs first, for example by splitting multi-allelic sites with `bcftools norm -m -any`. Variants keep the .row_id of the first object where its source VCF has them, other variants get new ids; the variant metadata of the first object and, in mode = "intersect", its record of removed invariant variants are kept, so bound objects can be bound again.

Value

VCFArrow object

Author(s)

Tomas Hrbek April 2026

Examples

```
f <- system.file("extdata", "vaillantii_discosnp_sub.vcf.gz",
                 package = "VCFArrow")
vcf <- read_vcf(f) |> set_vcf_groups(dirname(f))
ingroup <- vcf_extract_groups(vcf, c("GS", "BS", "WA"))
outgroup <- vcf_extract_groups(vcf, "OG")
vcf_bind(ingroup, outgroup, mode = "intersect")
vcf_bind(ingroup, outgroup, mode = "union", absent_as = "missing")
vcf_bind(vcf_filter_missingness(ingroup, 0.2), outgroup, outgroup = TRUE,
         absent_as = "missing")
```

`vcf_copy` *vcf_copy*

Description

Copy a VCFArrow object

Usage

```
vcf_copy(vcf_arrow)
```

Arguments

`vcf_arrow` -> VCFArrow object

Details

This function copies a VCFArrow object. It makes an actual physical copy of the object.

Value

VCFArrow object

Author(s)

Tomas Hrbek April 2026

Examples

```
f <- system.file("extdata", "vaillantii_discosnp_sub.vcf.gz",
                 package = "VCFArrow")
vcf <- read_vcf(f) |> set_vcf_groups(dirname(f))
vcf_copy(vcf_filter_maf(vcf, 0.05))
```

---

vcf_extract_groups *vcf_extract_groups*

Description

Extract groups of samples from a VCFArrow object

Usage

```
vcf_extract_groups(
  vcf_arrow,
  groups,
  keep = TRUE,
  f_invar = TRUE,
  verbose = TRUE
)
```

Arguments

vcf_arrow	-> VCFArrow object
groups	-> groups to retain/drop (character)
keep	-> retain (TRUE) or drop (FALSE) individuals flag, default TRUE (Boolean)
f_invar	-> filter invariant loci flag, default TRUE (Boolean)
verbose	-> report filtering stats, default TRUE (Boolean)

Details

This function removes groups of samples from a VCFArrow object, returning a new VCFArrow object. It uses the 'keep' flag to either keep or drop the groups in the list. By default will remove any loci that may have become invariant as the result of the removal of samples. By default will report removed samples, final % missing data, and number of retained samples after sample filtering.

Value

subsetted VCFArrow object

Author(s)

Tomas Hrbek April 2026

Examples

```
f <- system.file("extdata", "vaillantii_discosnp_sub.vcf.gz",
                 package = "VCFArrow")
vcf <- read_vcf(f) |> set_vcf_groups(dirname(f))
vcf_extract_groups(vcf, groups = c("GS", "BS"))
vcf_extract_groups(vcf, groups = "OG", keep = FALSE)
```

---

vcf_extract_samples *vcf_extract_samples*

Description

Extract samples from a VCFArrow object

Usage

```
vcf_extract_samples(
  vcf_arrow,
  samples,
  keep = TRUE,
  f_invar = TRUE,
  verbose = TRUE
)
```

Arguments

vcf_arrow	-> VCFArrow object
samples	-> individuals to retain/drop (character)
keep	-> retain (TRUE) or drop (FALSE) individuals flag, default TRUE (Boolean)
f_invar	-> filter invariant loci flag, default TRUE (Boolean)
verbose	-> report filtering stats, default TRUE (Boolean)

Details

This function removes samples from a VCFArrow object, returning a new VCFArrow object. It uses the 'keep' flag to either keep or drop the samples in the list. By default will remove any loci that may have become invariant as the result of the removal of samples. By default will report removed samples, final % missing data, and number of retained samples after sample filtering.

Value

subsetted VCFArrow object

Author(s)

Tomas Hrbek April 2026

Examples

```
f <- system.file("extdata", "vaillantii_discosnp_sub.vcf.gz",
                 package = "VCFArrow")
vcf <- read_vcf(f) |> set_vcf_groups(dirname(f))
vcf_extract_samples(vcf, samples = c("Pv14", "Pv27", "Pv28"))
vcf_extract_samples(vcf, samples = c("Pb2Jp", "Pb2Scx", "Pb1Rd"),
                    keep = FALSE)
```

---

vcf_filter_adr *vcf_filter_adr*

Description

Correct or remove genotypes with > % ALT/REF ratio from a VCFArrow object

Usage

```
vcf_filter_adr(
  vcf_arrow,
  mode = c("correct", "remove"),
  threshold = 0.1,
  f_invar = TRUE
)
```

Arguments

`vcf_arrow`	-> VCFArrow object
`mode`	-> switch between 'correct' or 'remove' mode (character)
`threshold`	-> decimal missing threshold, default 0.1 (numeric)
`f_invar`	-> filter invariant loci flag, default TRUE (Boolean)

Details

This function either changes genotypes or makes genotypes missing in a VCFArrow object if they have above/below threshold normalized ADR ratio, returning a new VCFArrow object. The ADR is calculated as ADR = ALT / (REF + ALT). If ADR > threshold, the genotype becomes ALT homozygous. If ADR < threshold, the genotype becomes REF homozygous. By default will remove any loci that may have become invariant as the result of the change/removal of genotypes

Value

subsetted VCFArrow object

Author(s)

Tomas Hrbek April 2026

Examples

```
f <- system.file("extdata", "vaillantii_discosnp_sub.vcf.gz",
                 package = "VCFArrow")
vcf <- read_vcf(f) |> set_vcf_groups(dirname(f))
vcf_filter_adr(vcf, mode = "correct")
```

---

vcf_filter_biallelic *vcf_filter_biallelic*

Description

Remove non-biallelic loci from a VCFArrow object

Usage

```
vcf_filter_biallelic(vcf_arrow)
```

Arguments

vcf_arrow	-> VCFArrow object

Details

This function removes non-biallelic loci from a VCFArrow object, returning a new VCFArrow object.

Value

subsetted VCFArrow object

Author(s)

Tomas Hrbek April 2026

Examples

```
f <- system.file("extdata", "vaillantii_discosnp_sub.vcf.gz",
                 package = "VCFArrow")
vcf <- read_vcf(f) |> set_vcf_groups(dirname(f))
vcf_filter_biallelic(vcf)
```

vcf_filter_coverage *vcf_filter_coverage*

Description

Remove genotypes below read DP threshold from a VCFArrow object

Usage

```
vcf_filter_coverage(vcf_arrow, threshold = 10)
```

Arguments

vcf_arrow	-> VCFArrow object
threshold	-> DP threshold, default 10 (integer)

Details

This function removes genotypes below a DP threshold from a VCFArrow object, returning a new VCFArrow object.

Value

subsetted VCFArrow object

Author(s)

Tomas Hrbek April 2026

Examples

```
f <- system.file("extdata", "vaillantii_discosnp_sub.vcf.gz",
                 package = "VCFArrow")
vcf <- read_vcf(f) |> set_vcf_groups(dirname(f))
vcf_filter_coverage(vcf, threshold = 10)
```

vcf_filter_hets *vcf_filter_hets*

Description

Remove loci above a heterozigosity threshold from a VCFArrow object

Usage

```
vcf_filter_hets(vcf_arrow, threshold = 0.5)
```

Arguments

vcf_arrow	-> VCFArrow object
threshold	-> heterozigosity threshold, default 0.5 (numeric)

Details

This function removes loci above a heterozigosity threshold from a VCFArrow object, returning a new VCFArrow object. High heterozigosities are indicative of potential paralogs.

Value

subsetted VCFArrow object

Author(s)

Tomas Hrbek April 2026

Examples

```
f <- system.file("extdata", "vaillantii_discosnp_sub.vcf.gz",
                 package = "VCFArrow")
vcf <- read_vcf(f) |> set_vcf_groups(dirname(f))
vcf_filter_hets(vcf, threshold = 0.5)
```

vcf_filter_indels *vcf_filter_indels*

Description

Remove indels from a VCFArrow object

Usage

```
vcf_filter_indels(vcf_arrow)
```

Arguments

vcf_arrow	-> VCFArrow object

Details

This function removes indel loci from a VCFArrow object, returning a new VCFArrow object.

Value

subsetted VCFArrow object

Author(s)

Tomas Hrbek April 2026

Examples

```
f <- system.file("extdata", "vaillantii_discosnp_sub.vcf.gz",
                 package = "VCFArrow")
vcf <- read_vcf(f) |> set_vcf_groups(dirname(f))
vcf_filter_indels(vcf)
```

vcf_filter_invariant *vcf_filter_invariant*

Description

Remove invariant loci from a VCFArrow object

Usage

```
vcf_filter_invariant(vcf_arrow)
```

Arguments

`vcf_arrow`	-> VCFArrow object

Details

This function removes invariant loci from a VCFArrow object, returning a new VCFArrow object. This might be desirable after subsetting a VCFArrow object by individuals.

Value

subsetted VCFArrow object

Author(s)

Tomas Hrbek April 2026

Examples

```
f <- system.file("extdata", "vaillantii_discosnp_sub.vcf.gz",
                 package = "VCFArrow")
vcf <- read_vcf(f) |> set_vcf_groups(dirname(f))
vcf_filter_invariant(vcf)
```

`vcf_filter_maf` *vcf_filter_maf*

Description

Remove loci below MAF threshold from a VCFArrow object

Usage

```
vcf_filter_maf(vcf_arrow, threshold = 0.05)
```

Arguments

`vcf_arrow`	-> VCFArrow object
`threshold`	-> decimal MAF threshold, default 0.05 (numeric)

Details

This function removes loci below MAF threshold from a VCFArrow object, returning a new VCFArrow object.

Value

subsetted VCFArrow object

Author(s)

Tomas Hrbek April 2026

Examples

```
f <- system.file("extdata", "vaillantii_discosnp_sub.vcf.gz",
                 package = "VCFArrow")
vcf <- read_vcf(f) |> set_vcf_groups(dirname(f))
vcf_filter_maf(vcf, threshold = 0.05)
```

---

vcf_filter_missing *vcf_filter_missing*

Description

Remove samples with > % missing data from a VCFArrow object

Usage

```
vcf_filter_missing(vcf_arrow, threshold = 0.5, f_invar = TRUE, verbose = TRUE)
```

Arguments

`vcf_arrow`	-> VCFArrow object
`threshold`	-> decimal missing threshold, default 0.5 (numeric)
`f_invar`	-> filter invariant loci flag, default TRUE (Boolean)
`verbose`	-> report filtering stats, default TRUE (Boolean)

Details

This function removes samples from a VCFArrow object if they have above threshold missing loci, returning a new VCFArrow object. By default will remove any loci that may have become invariant as the result of the removal of samples. By default will report removed samples, final % missing data, and number of retained samples after sample filtering.

Value

subsetted VCFArrow object

Author(s)

Tomas Hrbek April 2026

Examples

```
f <- system.file("extdata", "vaillantii_discosnp_sub.vcf.gz",
                 package = "VCFArrow")
vcf <- read_vcf(f) |> set_vcf_groups(dirname(f))
vcf |>
  vcf_filter_missingness(threshold = 0.2) |>
  vcf_filter_missing(threshold = 0.3)
```

---

vcf_filter_missingness

vcf_filter_missingness

Description

Remove loci above missingness threshold from a VCFArrow object

Usage

```
vcf_filter_missingness(vcf_arrow, threshold = 0.1, verbose = TRUE)
```

Arguments

vcf_arrow	-> VCFArrow object
threshold	-> decimal missingness threshold, default 0.1 (numeric)
verbose	-> flag to report total % missing data after filtering, default TRUE (Boolean)

Details

This function removes loci above missingness threshold from a VCFArrow object, returning a new VCFArrow object. Missingness is a locus focused metric, i.e. missing data per locus.

Value

subsetted VCFArrow object

Author(s)

Tomas Hrbek April 2026

Examples

```
f <- system.file("extdata", "vaillantii_discosnp_sub.vcf.gz",
                 package = "VCFArrow")
vcf <- read_vcf(f) |> set_vcf_groups(dirname(f))
vcf_filter_missingness(vcf, threshold = 0.2)
```

vcf_filter_multiSNV *vcf_filter_multiSNV*

Description

Subset a VCFArrow object keeping only loci with 2+ SNVs per locus

Usage

```
vcf_filter_multiSNV(vcf_arrow, block_size = 10000, minSNV = 2, maxSNV = 5)
```

Arguments

`vcf_arrow`	-> VCFArrow object
`block_size`	-> size of linked SNV blocks, default 10000 bp (integer)
`minSNV`	-> minimum linked block size, default 2 (integer)
`maxSNV`	-> maximum number of selected linked SNVs per block, default 5 (integer)

Details

This function subsets a VCFArrow object keeping only loci with between min and max # of SNVs per locus, returning a new VCFArrow object. Default min = 2 and max = 5 SNVs per locus (recommended as input for fineRADstructure analyses). Locus is defined as a different chromosome or a block of the 'block_size' parameter value within a chromosome.

Value

subsetted VCFArrow object

Author(s)

Tomas Hrbek April 2026

Examples

```
f <- system.file("extdata", "vaillantii_discosnp_sub.vcf.gz",
                 package = "VCFArrow")
vcf <- read_vcf(f) |> set_vcf_groups(dirname(f))
vcf_filter_multiSNV(vcf, block_size = 10000, minSNV = 2, maxSNV = 5)
```

vcf_filter_oneSNV *vcf_filter_oneSNV*

Description

Subset a VCFArrow object keeping only 1 SNV per locus

Usage

```
vcf_filter_oneSNV(vcf_arrow, block_size = 10000)
```

Arguments

`vcf_arrow`	-> VCFArrow object
`block_size`	-> size of linked SNV blocks, default 10000 bp (integer)

Details

This function subsets a VCFArrow object keeping only 1 SNV per locus, returning a new VCFArrow object. Locus is defined as a different chromosome or a block of the 'block_size' parameter value within a chromosome. The first SNV independent of quality is taken (may modify this in the future).

Value

subsetted VCFArrow object

Author(s)

Tomas Hrbek April 2026

Examples

```
f <- system.file("extdata", "vaillantii_discosnp_sub.vcf.gz",
                 package = "VCFArrow")
vcf <- read_vcf(f) |> set_vcf_groups(dirname(f))
vcf_filter_oneSNV(vcf)
```

vcf_filter_paralogs *vcf_filter_paralogs*

Description

Remove loci above REPEAT and below RANK threshold from a VCFArrow object

Usage

```
vcf_filter_paralogs(
  vcf_arrow,
  threshold_rk = 0.2,
  threshold_rpt = 0.5,
  keep_na = FALSE
)
```

Arguments

vcf_arrow	-> VCFArrow object
threshold_rk	-> decimal RANK threshold, loci below it are removed, default 0.2 (numeric)
threshold_rpt	-> decimal REPEAT threshold, loci above it are removed, default 0.5 (numeric)
keep_na	-> retain loci without a RANK or REPEAT value, default FALSE (Boolean)

Details

This function removes loci below the RANK threshold and above the REPEAT threshold from a VCFArrow object, returning a new VCFArrow object. It applies vcf_filter_rank() and then vcf_filter_rpt(). RANK and REPEAT metrics focus on different paralog signals, and used jointly maximize true positive and minimize false positive detections. Simulations indicate that removing loci with Rk < 0.2 or RPT > 0.5 gives the best results, with Rk < 0.4 or RPT > 0.5 being slightly more conservative.

Value

subsetted VCFArrow object

Author(s)

Tomas Hrbek September 2026

Examples

```
f <- system.file("extdata", "vaillantii_discosnp_sub.vcf.gz",
                 package = "VCFArrow")
vcf <- read_vcf(f) |> set_vcf_groups(dirname(f))
vcf_filter_paralogs(vcf, threshold_rk = 0.2, threshold_rpt = 0.5)
```

vcf_filter_pass *vcf_filter_pass*

Description

Remove loci that did not PASS FILTER in a VCFArrow object

Usage

```
vcf_filter_pass(vcf_arrow)
```

Arguments

vcf_arrow	-> VCFArrow object

Details

This function removes loci that did not PASS the FILTER in a VCFArrow object, returning a new VCFArrow object. PASS indicates that a variant has successfully passed all applied quality control filters. When no filters were applied and the FILTER field is '.' (Dot), vcf_filter_pass() defaults to passing the locus.

Value

subsetted VCFArrow object

Author(s)

Tomas Hrbek April 2026

Examples

```
f <- system.file("extdata", "vaillantii_discosnp_sub.vcf.gz",
                 package = "VCFArrow")
vcf <- read_vcf(f) |> set_vcf_groups(dirname(f))
vcf_filter_pass(vcf)
```

vcf_filter_quality *vcf_filter_quality*

Description

Remove loci below QUAL threshold from a VCFArrow object

Usage

```
vcf_filter_quality(vcf_arrow, threshold = 30)
```

Arguments

`vcf_arrow`	-> VCFArrow object
`threshold`	-> QUAL threshold, default 30 (integer)

Details

This function removes loci below QUAL threshold from a VCFArrow object, returning a new VCFArrow object. QUAL is a Phred-scaled quality metric assessing whether any variant exists at that position across all samples. QUAL is different from PL which represents the confidence in a genotype call of a specific sample. When QUAL not reported '.' (Dot), QUAL is assumed to be above threshold.

Value

subsetted VCFArrow object

Author(s)

Tomas Hrbek April 2026

Examples

```
f <- system.file("extdata", "vaillantii_discosnp_sub.vcf.gz",
                 package = "VCFArrow")
vcf <- read_vcf(f) |> set_vcf_groups(dirname(f))
vcf_filter_quality(vcf, threshold = 30)
```

`vcf_filter_rank` *vcf_filter_rank*

Description

Remove loci below RANK threshold from a VCFArrow object

Usage

```
vcf_filter_rank(vcf_arrow, threshold = 0.4, keep_na = FALSE)
```

Arguments

`vcf_arrow`	-> VCFArrow object
`threshold`	-> decimal rank threshold, default 0.4 (numeric)
`keep_na`	-> retain loci without a rank value, default FALSE (Boolean)

Details

This function removes loci below RANK threshold from a VCFArrow object, returning a new VCFArrow object. RANK is calculated by kissreads2 in DiscoSNP-RAD (Gauthier et. al. 2020) and registered as Rk in INFO. RANK is calculated as sqrt(chi-sqr/n) of allele read counts, and used for paralog detection -> very low rank values (<0.4) are indicative of paralogs.

Value

subsetted VCFArrow object

Author(s)

Tomas Hrbek April 2026

Examples

```
f <- system.file("extdata", "vaillantii_discosnp_sub.vcf.gz",
                 package = "VCFArrow")
vcf <- read_vcf(f) |> set_vcf_groups(dirname(f))
vcf_filter_rank(vcf, threshold = 0.5)
```

vcf_filter_rpt *vcf_filter_rpt*

Description

Remove loci above REPEAT threshold from a VCFArrow object

Usage

```
vcf_filter_rpt(vcf_arrow, threshold = 0.5, keep_na = FALSE)
```

Arguments

vcf_arrow	-> VCFArrow object
threshold	-> decimal REPEAT threshold, loci above it are removed, default 0.5 (numeric)
keep_na	-> retain loci without a REPEAT value, default FALSE (Boolean)

Details

This function removes loci above REPEAT threshold from a VCFArrow object, returning a new VCFArrow object. REPEAT is calculated by disco_haplotypes in DiscoSNP-RAD (Gauthier et. al. 2020) and registered as RPT in INFO. REPEAT is calculated as max of overlap, depth score and is 1 if in a giant-component bubble, and is used for paralog detection -> high repeat values (>0.5) are indicative of paralogs.

Value

subsetted VCFArrow object

Author(s)

Tomas Hrbek September 2026

Examples

```
f <- system.file("extdata", "vaillantii_discosnp_sub.vcf.gz",
                 package = "VCFArrow")
vcf <- read_vcf(f) |> set_vcf_groups(dirname(f))
vcf_filter_rpt(vcf, threshold = 0.5)
```

vcf_gc — *Garbage-collect VCFArrow temp directories*

Description

Triggers R's garbage collector (three full passes to handle the finalizer -> pending-queue -> unlink chain), then flushes any directories whose reference count has already reached zero.

Usage

```
vcf_gc(force = FALSE, verbose = TRUE)
```

Arguments

`force`	Logical. If TRUE, also force-deregister and delete directories for objects that are still nominally live in the registry. Use this when you have called `rm()` on all VCFArrow objects but the temp directories have not been cleaned up (common in RStudio, which can hold display references that delay GC).
`verbose`	Logical. Print a status message.

Value

Invisibly returns `NULL`; called for its side effect of deleting temporary directories.

Typical workflow

```
rm(vcf1, vcf2, vcf3)
vcf_gc()              # usually sufficient
vcf_gc(force = TRUE)  # if directories remain after rm()
```

Examples

```
f <- system.file("extdata", "vaillantii_discosnp_sub.vcf.gz",
                 package = "VCFArrow")
vcf <- read_vcf(f)
rm(vcf)
vcf_gc()
```

vcf_memory_estimate *Estimate RAM required for a VCFArrow export operation*

Description

Estimate RAM required for a VCFArrow export operation

Usage

```
vcf_memory_estimate(
  vcf_arrow,
  keep_groups = NULL,
  format = c("individual", "pop", "chunk"),
  chunk_size = 100000L,
  lowmem = FALSE
)
```

Arguments

vcf_arrow	A VCFArrow object.
keep_groups	Groups to export (NULL = all).
format	One of "individual" (Structure, Arlequin, FASTA, ...) or "pop" (BayesScan, Treemix, Migrate-N C, ...) or "chunk" (SmartSNP, fineRADstructure, sNMF, EIGENSTRAT, ...).
chunk_size	Feather chunk size used at read_vcf() time.
lowmem	If TRUE, estimate uses raw-byte matrices (vcf2*() lowmem variants); otherwise integer matrices.

Value

Invisibly, a list with elements:

- n_var: number of variants
- n_samples: number of samples retained
- n_pops: number of populations
- chunk_arrow_bytes: Arrow memory per chunk read
- matrix_bytes: size of the accumulation matrices
- peak_bytes: estimated peak memory

Examples

```
f <- system.file("extdata", "vaillantii_discosnp_sub.vcf.gz",
                 package = "VCFArrow")
vcf <- read_vcf(f) |> set_vcf_groups(dirname(f))
vcf_memory_estimate(vcf, format = "individual")
```

vcf_set_workers *vcf_set_workers*

Description

Set the number of parallel worker processes used by VCFArrow

Usage

```
vcf_set_workers(workers = 1L)
```

Arguments

workers	-> number of worker processes, default 1 (no parallelism) (integer)

Details

Functions that process the on-disk genotype chunks independently distribute the chunks over `workers` background R processes: read_vcf(), the genotype-based filters, sample and group extraction, vcf_stats() and the assess functions, vcf_bind(), write_vcf(), the vcf2*() exporters and vcf2gt_long() with CSV output (with feather or Parquet output it runs serially). Results do not depend on the number of workers. The workers are started once and reused until `vcf_set_workers(1)` is called or the session ends. Each worker holds one chunk in memory at a time, so memory use grows with the number of workers (see vcf_memory_estimate()). Useful values are up to the number of physical CPU cores; disk speed limits the gain beyond that.

Value

Invisibly, the previous number of workers.

Author(s)

Tomas Hrbek September 2026

Examples

```
f <- system.file("extdata", "vaillantii_discosnp_sub.vcf.gz",
                 package = "VCFArrow")
# 2 workers (CRAN examples may use at most 2 cores)
vcf_set_workers(2)
vcf <- read_vcf(f)
vcf_set_workers(1)
```

vcf_stats *vcf_stats*

Description

Calculates basic stats of each samples from VCFArrow format data. Includes average read depth per individual, missing data per individual, Watterson's theta and pi.

Usage

```
vcf_stats(vcf_arrow, res_path, project, theta = FALSE)
```

Arguments

vcf_arrow	-> VCFArrow object
res_path	-> directory where to write results
project	-> base name of the project file
theta	-> flag to perform theta and pi calculation, default FALSE (Boolean)

Details

This function calculates average read depth, heterozygosity number of heterozygotes, number of reference and alternative homozygotes, missing data and total number SNPs of each sample in an VCFArrow object. Optionally calls vcf_theta() to get total and group Watterson's theta and pi.

Value

table of statistics

Author(s)

Tomas Hrbek April 2026

Examples

```
f <- system.file("extdata", "vaillantii_discosnp_sub.vcf.gz",
                 package = "VCFArrow")
vcf <- read_vcf(f) |> set_vcf_groups(dirname(f))
vcf_stats(vcf, res_path = tempdir(), project = "vaillantii",
          theta = TRUE)
```

vcf_sub_SNVs *vcf_sub_loci*

Description

Randomly subsets SNVs from a VCFArrow object

Usage

```
vcf_sub_SNVs(vcf_arrow, n_SNVs = 10000, seed = NULL)
```

Arguments

vcf_arrow	-> VCFArrow object
n_SNVs	-> number of SNVs to subset, default 10000 (integer)
seed	-> random number generator seed, default NULL (integer)

Details

This function subsets a VCFArrow object to specific number of SNVs, returning new VCFArrow object. The seed for random number generator is automatically generated unless otherwise specified.

Value

VCFArrow object

Author(s)

Tomas Hrbek April 2026

Examples

```
f <- system.file("extdata", "vaillantii_discosnp_sub.vcf.gz",
                 package = "VCFArrow")
vcf <- read_vcf(f) |> set_vcf_groups(dirname(f))
vcf_sub_SNVs(vcf, n_SNVs = 1000, seed = 42)
```

vcf_sub_SNVs_stratified

vcf_sub_loci_stratified

Description

Randomly subsets SNVs from a VCFArrow object

Usage

```
vcf_sub_SNVs_stratified(vcf_arrow, n_SNVs = 1000, seed = NULL)
```

Arguments

vcf_arrow	-> VCFArrow object
n_SNVs	-> number of SNVs to subset, default 1000 (integer)
seed	-> random number generator seed, default NULL (integer)

Details

This function subsets a VCFArrow object to specific number of SNVs, returning new VCFArrow object. The subsampling is stratified, i.e. the same proportion of SNVs per CHROM. The seed for random number generator is automatically generated unless specified.

Value

VCFArrow object

Author(s)

Tomas Hrbek April 2026

Examples

```
f <- system.file("extdata", "vaillantii_discosnp_sub.vcf.gz",
                 package = "VCFArrow")
vcf <- read_vcf(f) |> set_vcf_groups(dirname(f))
vcf_sub_SNVs_stratified(vcf, n_SNVs = 1000, seed = 42)
```

vcf_suggest_chunk_size

Suggest a chunk_size for read_vcf() given available RAM

Description

Suggest a chunk_size for read_vcf() given available RAM

Usage

```
vcf_suggest_chunk_size(available_gb, n_samples, n_columns = 5L)
```

Arguments

available_gb	RAM available in gigabytes.
n_samples	Number of samples.
n_columns	Number of columns per gt row (default 5: row_id, sample, a1, a2, phased).

Value

Invisibly, the suggested chunk size (numeric).

Examples

```
vcf_suggest_chunk_size(available_gb = 8, n_samples = 100)
```

vcf_theta

vcf_theta

Description

Calculates basic Watterson's theta and pi for all samples and for sample groups from VCFArrow format data.

Usage

```
vcf_theta(vcf_arrow, keep_groups = NULL)
```

Arguments

vcf_arrow	-> VCFArrow object
keep_groups	-> groups to retain, default NULL (character)

Details

This function calculates Watterson's theta and pi for the entire VCFArrow object, and for groups of individuals whose grouping is indicated by the groups slot in the VCFArrow object. The statistics are accumulated chunk by chunk as running sums per group, so memory use does not grow with the number of variants, and chunks are processed in parallel when workers are set with vcf_set_workers().

Value

list of statistics

Author(s)

Tomas Hrbek April 2026

Examples

```
f <- system.file("extdata", "vaillantii_discosnp_sub.vcf.gz",
                 package = "VCFArrow")
vcf <- read_vcf(f) |> set_vcf_groups(dirname(f))
vcf_theta(vcf)
```

---

vcf2admixture *vcf2admixture*

Description

Converts a VCFArrow object to a PLINK .bed format infile plus ADMIXTURE-specific .pop file

Usage

```
vcf2admixture(
  vcf_arrow,
  out_file,
  keep_groups = NULL,
  sex = NULL,
  pheno = NULL,
  chrom_code = c("zero", "auto", "index", "keep"),
  supervised = FALSE,
  reference_groups = NULL
)
```

Arguments

`vcf_arrow`	-> VCFArrow object
`out_file`	-> output file prefix; extensions are added automatically, no default (character)
`keep_groups`	-> groups to retain, default NULL (character)
`sex`	-> vector of sexes of samples, default NULL (character)
`pheno`	-> vector of phenotypes of samples, default NULL (character)
`chrom_code`	-> how CHROM is written: "auto" (keep numeric CHROM, otherwise 0), "index" (1..n by first appearance), "zero" (all 0) or "keep" (verbatim), default "zero" (character)
`supervised`	-> flag to generate .pop for use in ADMIXTURE's supervised mode, default FALSE (Boolean)
`reference_groups`	-> vector of ancestry group labels, default NULL (character)

Details

This function converts a VCFArrow object to an external PLINK .bed formatted file. Writing occurs in chunks whose size is determined by the read_vcf() function. Larger chunks result in faster writing speeds. If no groups are defined, the default behavior is to use all groups. Sex and phenotype vectors are optional. If not defined sex = 0, pheno = -9. For supervised analyses, the supervised flag needs to be true to generate .pop file needed for ADMIXTURE's supervised mode. The reference_groups vector specifies ancestral information of individuals. When NULL, or for any missing individual, ancestry is inferred.

Value

Invisibly returns the input VCFArrow object; called for its side effect of writing `out_file`.

Author(s)

Tomas Hrbek May 2026

Examples

```
f <- system.file("extdata", "vaillantii_discosnp_sub.vcf.gz",
                 package = "VCFArrow")
vcf <- read_vcf(f) |> set_vcf_groups(dirname(f))
vcf2admixture(vcf, out_file = file.path(tempdir(), "admixture_in"))
vcf2admixture(vcf, out_file = file.path(tempdir(), "admixture_sup"),
              supervised = TRUE,
              reference_groups = c("GS", "BS", "WA"))
```

vcf2apparent *vcf2apparent*

Description

Converts a VCFArrow object to an Apparent format infile

Usage

```
vcf2apparent(vcf_arrow, out_file, keep_groups = NULL, key = "All")
```

Arguments

vcf_arrow	-> VCFArrow object
out_file	-> name of file to output, no default (character)
keep_groups	-> groups to retain, default NULL (character)
key	-> relationship type (All, Pa, Mo, Fa, Off), default All (character)

Details

This function converts a VCFArrow object to an external SmartSNP formatted file. Writing occurs in chunks whose size is determined by the read_vcf() function. Larger chunks result in faster writing speeds. If no groups are defined, the default behavior is to use all groups. Possible relationships defined by the parameter 'kee' are All, Pa, Mo, Fa, Off.

Value

Invisibly returns the input VCFArrow object; called for its side effect of writing `out_file`.

Author(s)

Tomas Hrbek May 2026

Examples

```
f <- system.file("extdata", "vaillantii_discosnp_sub.vcf.gz",
                 package = "VCFArrow")
vcf <- read_vcf(f) |> set_vcf_groups(dirname(f))
vcf2apparent(vcf, out_file = tempfile(fileext = ".txt"))
```

vcf2arlequin *vcf2arlequin*

Description

Converts a VCFArrow object to an Arlequin infile

Usage

```
vcf2arlequin(vcf_arrow, out_file, keep_groups = NULL)
```

Arguments

vcf_arrow	-> VCFArrow object
out_file	-> name of file to output, no default (character)
keep_groups	-> groups to retain, default NULL (character)

Details

This function converts a VCFArrow object to an external Arlequin formatted file. Writing occurs in chunks whose size is determined by the read_vcf() function. Larger chunks result in faster writing speeds. If no groups are defined, the default behavior is to use all groups.

Value

Invisibly returns the input VCFArrow object; called for its side effect of writing `out_file`.

Author(s)

Tomas Hrbek May 2026

Examples

```
f <- system.file("extdata", "vaillantii_discosnp_sub.vcf.gz",
                 package = "VCFArrow")
vcf <- read_vcf(f) |> set_vcf_groups(dirname(f))
vcf2arlequin(vcf, out_file = tempfile(fileext = ".arp"))
```

`vcf2bayesass` *vcf2bayesass*

Description

Converts a VCFArrow object to BayesAss format infile

Usage

```
vcf2bayesass(vcf_arrow, out_file, keep_groups = NULL)
```

Arguments

`vcf_arrow`	-> VCFArrow object
`out_file`	-> name of file to output, no default (character)
`keep_groups`	-> groups to retain, default NULL (character)

Details

This function converts a VCFArrow object to an external BayesAss formatted file. Writing occurs in chunks whose size is determined by the read_vcf() function. Larger chunks result in faster writing speeds. If no groups are defined, the default behavior is to use all groups.

Value

Invisibly returns the input VCFArrow object; called for its side effect of writing `out_file`.

Author(s)

Tomas Hrbek May 2026

Examples

```
f <- system.file("extdata", "vaillantii_discosnp_sub.vcf.gz",
                 package = "VCFArrow")
vcf <- read_vcf(f) |> set_vcf_groups(dirname(f))
vcf2bayesass(vcf, out_file = tempfile(fileext = ".immanc"))
```

`vcf2bayescan` *vcf2bayescan*

Description

Converts a VCFArrow object to Bayescan format infile

Usage

```
vcf2bayescan(vcf_arrow, out_file, keep_groups = NULL)
```

Arguments

`vcf_arrow`	-> VCFArrow object
`out_file`	-> name of file to output, no default (character)
`keep_groups`	-> groups to retain, default NULL (character)

Details

This function converts a VCFArrow object to an external BayesAss formatted file. Writing occurs in chunks whose size is determined by the read_vcf() function. Larger chunks result in faster writing speeds. If no groups are defined, the default behavior is to use all groups.

Value

Invisibly returns the input VCFArrow object; called for its side effect of writing `out_file`.

Author(s)

Tomas Hrbek May 2026

Examples

```
f <- system.file("extdata", "vaillantii_discosnp_sub.vcf.gz",
                 package = "VCFArrow")
vcf <- read_vcf(f) |> set_vcf_groups(dirname(f))
vcf2bayescan(vcf, out_file = tempfile(fileext = ".txt"))
```

vcf2eigenstrat *vcf2eigenstrat*

Description

Converts a VCFArrow object to Eigenstrat format infile

Usage

```
vcf2eigenstrat(vcf_arrow, out_file, keep_groups = NULL, sex = NULL)
```

Arguments

vcf_arrow	-> VCFArrow object
out_file	-> output file prefix; extensions are added automatically, no default (character)
keep_groups	-> groups to retain, default NULL (character)
sex	-> sex of the individual, default = U (undefined) (character)

Details

This function converts a VCFArrow object to an external BayesAss formatted file. Writing occurs in chunks whose size is determined by the read_vcf() function. Larger chunks result in faster writing speeds. If no groups are defined, the default behavior is to use all groups.

Value

Invisibly returns the input VCFArrow object; called for its side effect of writing `out_file`.

Author(s)

Tomas Hrbek May 2026

Examples

```
f <- system.file("extdata", "vaillantii_discosnp_sub.vcf.gz",
                 package = "VCFArrow")
vcf <- read_vcf(f) |> set_vcf_groups(dirname(f))
vcf2eigenstrat(vcf, out_file = file.path(tempdir(), "eigenstrat_in"))
```

vcf2fasta *vcf2fasta*

Description

Converts a VCFArrow object to a FASTA format infile

Usage

```
vcf2fasta(vcf_arrow, out_file, keep_groups = NULL)
```

Arguments

vcf_arrow	-> VCFArrow object
out_file	-> name of file to output, no default (character)
keep_groups	-> groups to retain, default NULL (character)

Details

This function converts a VCFArrow object to an external FASTA formatted file. Writing occurs in chunks whose size is determined by the read_vcf() function. Larger chunks result in faster writing speeds. If no groups are defined, the default behavior is to use all groups.

Value

Invisibly returns the input VCFArrow object; called for its side effect of writing `out_file`.

Author(s)

Tomas Hrbek May 2026

Examples

```
f <- system.file("extdata", "vaillantii_discosnp_sub.vcf.gz",
                 package = "VCFArrow")
vcf <- read_vcf(f) |> set_vcf_groups(dirname(f))
vcf2fasta(vcf, out_file = tempfile(fileext = ".fas"))
```

vcf2fineradstructure *vcf2fineradstructure*

Description

Converts a VCFArrow object to FineRadStructure format infile

Usage

```
vcf2fineradstructure(vcf_arrow, out_file, keep_groups = NULL)
```

Arguments

`vcf_arrow`	-> VCFArrow object
`out_file`	-> name of file to output, no default (character)
`keep_groups`	-> groups to retain, default NULL (character)

Details

This function converts a VCFArrow object to an external Treemix formatted file. Writing occurs in chunks whose size is determined by the read_vcf() function. Larger chunks result in faster writing speeds. If no groups are defined, the default behavior is to use all groups.

Value

Invisibly returns the input VCFArrow object; called for its side effect of writing `out_file`.

Author(s)

Tomas Hrbek May 2026

Examples

```
f <- system.file("extdata", "vaillantii_discosnp_sub.vcf.gz",
                 package = "VCFArrow")
vcf <- read_vcf(f) |> set_vcf_groups(dirname(f))
vcf_linked <- vcf_filter_multiSNV(vcf)
vcf2fineradstructure(vcf_linked, out_file = tempfile(fileext = ".txt"))
```

vcf2genepop *vcf2genepop*

Description

Converts a VCFArrow object to a Genepop format infile

Usage

```
vcf2genepop(vcf_arrow, out_file, keep_groups = NULL)
```

Arguments

vcf_arrow	-> VCFArrow object
out_file	-> name of file to output, no default (character)
keep_groups	-> groups to retain, default NULL (character)

Details

This function converts a VCFArrow object to an external Genpop formatted file. Writing occurs in chunks whose size is determined by the read_vcf() function. Larger chunks result in faster writing speeds. If no groups are defined, the default behavior is to use all groups.

Value

Invisibly returns the input VCFArrow object; called for its side effect of writing `out_file`.

Author(s)

Tomas Hrbek May 2026

Examples

```
f <- system.file("extdata", "vaillantii_discosnp_sub.vcf.gz",
                 package = "VCFArrow")
vcf <- read_vcf(f) |> set_vcf_groups(dirname(f))
vcf2genepop(vcf, out_file = tempfile(fileext = ".gen"))
```

vcf2genlight *vcf2genlight*

Description

Converts a VCFArrow object to Genlight format infile

Usage

```
vcf2genlight(
  vcf_arrow,
  out_file = NULL,
  keep_groups = NULL,
  ploidy = 2L,
  save = FALSE
)
```

Arguments

`vcf_arrow`	-> VCFArrow object
`out_file`	-> name of .rds file to write, required when save = TRUE, default NULL (character)
`keep_groups`	-> groups to retain, default NULL (character)
`ploidy`	-> ploidy level, default = 2 (integer)
`save`	-> save as R data object, default = FALSE (Boolean)

Details

This function converts a VCFArrow object to an adegenet Genlight object. It requires the 'adegenet' package, which VCFArrow only suggests: install it with `install.packages("adegenet")` to use this function. Genotypes are read in chunks whose size is determined by the read_vcf() function. If no groups are defined, the default behavior is to use all groups. Genlight objects can encode polyploid genomes, by default diploid genomes are assumed. Genlight objects are in-memory S4 objects and are always returned; if save = TRUE, the object is also written to out_file with saveRDS(). Unlike the other exporters, out_file is optional because nothing is written unless save = TRUE.

Value

An adegenet `genlight` object.

Author(s)

Tomas Hrbek May 2026

Examples

```
f <- system.file("extdata", "vaillantii_discosnp_sub.vcf.gz",
                 package = "VCFArrow")
vcf <- read_vcf(f) |> set_vcf_groups(dirname(f))
gl <- vcf2genlight(vcf)
gl
gl <- vcf2genlight(vcf, out_file = tempfile(fileext = ".rds"),
                   save = TRUE)
```

---

vcf2gt_long *vcf2gt_long*

Description

Converts a VCFArrow object to tidy long format infile

Usage

```
vcf2gt_long(
  vcf_arrow,
  out_file,
  keep_groups = NULL,
  format = c("feather", "parquet", "csv"),
  col_select = NULL
)
```

Arguments

vcf_arrow	-> VCFArrow object
out_file	-> name of file to output, no default (character)
keep_groups	-> groups to retain, default NULL (character)
format	-> one of three output formats (arrow, parquet, CSV) (character)
col_select	-> optional selection of columns to save, default ALL

Details

This function converts a VCFArrow object to an external SmartSNP formatted file. Writing occurs in chunks whose size is determined by the read_vcf() function. Larger chunks result in faster writing speeds. If no groups are defined, the default behavior is to use all groups. The tidy data can be saved in either Arrow, Parquet or CSV formats. File extension is added automatically if missing. Optionally, specific columns can be saved, by default all columns are saved. The gt long slot contains pre-calculated metrics in addition to just genotypes.

Value

Invisibly returns the path of the written file.

Author(s)

Tomas Hrbek May 2026

Examples

```
f <- system.file("extdata", "vaillantii_discosnp_sub.vcf.gz",
                 package = "VCFArrow")
vcf <- read_vcf(f) |> set_vcf_groups(dirname(f))
vcf2gt_long(vcf, out_file = tempfile(), format = "csv")
```

---

vcf2migrate *vcf2migrate*

Description

Converts a VCFArrow object to a MIGRATE-N format infile

Usage

```
vcf2migrate(
  vcf_arrow,
  out_file,
  keep_groups = NULL,
  block_size = 100L,
  method = "S"
)
```

Arguments

vcf_arrow	-> VCFArrow object
out_file	-> name of file to output, no default (character)
keep_groups	-> groups to retain, default NULL (character)
block_size	-> number of SNPs (method S) or base pairs (method N) per linked block, default 100 (integer)
method	-> "C" (allele counts), "S" (sequences, fixed blocks) or "N" (sequences, chromosome intervals), default "S" (character)

Details

This function converts a VCFArrow object to an external MIGRATE-N formatted file. Writing occurs in chunks whose size is determined by the read_vcf() function. Larger chunks result in faster writing speeds. If no groups are defined, the default behavior is to use all groups. The function implements two different SNP formats, new (un)linked SNPs (S) - fixed blocks, and new (un)linked SNPs (N) - chromosome intervals. Alleles are sequence data when SNPs are mapped to a reference. The S option generates blocks of a specific number of SNPs and treats them as

linked; this is appropriate if SNPs are extracted without a reference. The N format extracts all SNPs within a chromosome, within the block size and treats them as linked; this is appropriate if SNPs are mapped against a reference. The size of the linked block is determined by the block_size parameter. See the Migrate-N documentation (https://peterbeerli.com/migrate-html5/) for format detail.

Value

Invisibly returns the input VCFArrow object; called for its side effect of writing `out_file`.

Author(s)

Tomas Hrbek May 2026

Examples

```
f <- system.file("extdata", "vaillantii_discosnp_sub.vcf.gz",
                 package = "VCFArrow")
vcf <- read_vcf(f) |> set_vcf_groups(dirname(f))
vcf2migrate(vcf, out_file = tempfile(fileext = ".txt"))
```

---

vcf2nexus *vcf2nexus*

Description

Converts a VCFArrow object to a NEXUS format infile

Usage

```
vcf2nexus(vcf_arrow, out_file, keep_groups = NULL)
```

Arguments

`vcf_arrow`	-> VCFArrow object
`out_file`	-> name of file to output, no default (character)
`keep_groups`	-> groups to retain, default NULL (character)

Details

This function converts a VCFArrow object to an external NEXUS formatted file. Writing occurs in chunks whose size is determined by the read_vcf() function. Larger chunks result in faster writing speeds. If no groups are defined, the default behavior is to use all groups.

Value

Invisibly returns the input VCFArrow object; called for its side effect of writing `out_file`.

Author(s)

Tomas Hrbek May 2026

Examples

```
f <- system.file("extdata", "vaillantii_discosnp_sub.vcf.gz",
                 package = "VCFArrow")
vcf <- read_vcf(f) |> set_vcf_groups(dirname(f))
vcf2nexus(vcf, out_file = tempfile(fileext = ".nex"))
```

---

vcf2plink_bed *vcf2plink_bed*

Description

Converts a VCFArrow object to a PLINK .bed format infile

Usage

```
vcf2plink_bed(
  vcf_arrow,
  out_file,
  keep_groups = NULL,
  sex = NULL,
  pheno = NULL,
  chrom_code = c("auto", "index", "zero", "keep")
)
```

Arguments

vcf_arrow	-> VCFArrow object
out_file	-> output file prefix; extensions are added automatically, no default (character)
keep_groups	-> groups to retain, default NULL (character)
sex	-> vector of sexes of samples, default NULL (character)
pheno	-> vector of phenotypes of samples, default NULL (character)
chrom_code	-> how CHROM is written: "auto" (keep numeric CHROM, otherwise 0), "index" (1..n by first appearance), "zero" (all 0) or "keep" (verbatim), default "auto" (character)

Details

This function converts a VCFArrow object to an external PLINK .bed formatted file. Writing occurs in chunks whose size is determined by the read_vcf() function. Larger chunks result in faster writing speeds. If no groups are defined, the default behavior is to use all groups. Sex and phenotype vectors are optional. If not defined sex = 0, pheno = -9.

Value

Invisibly returns the input VCFArrow object; called for its side effect of writing `out_file`.

Author(s)

Tomas Hrbek May 2026

Examples

```
f <- system.file("extdata", "vaillantii_discosnp_sub.vcf.gz",
                 package = "VCFArrow")
vcf <- read_vcf(f) |> set_vcf_groups(dirname(f))
vcf2plink_bed(vcf, out_file = file.path(tempdir(), "plink_out"))
```

`vcf2plink_ped` *vcf2plink_ped*

Description

Converts a VCFArrow object to a PLINK .ped format infile

Usage

```
vcf2plink_ped(
  vcf_arrow,
  out_file,
  keep_groups = NULL,
  sex = NULL,
  pheno = NULL,
  chrom_code = c("auto", "index", "zero", "keep")
)
```

Arguments

`vcf_arrow`	-> VCFArrow object
`out_file`	-> output file prefix; extensions are added automatically, no default (character)
`keep_groups`	-> groups to retain, default NULL (character)
`sex`	-> vector of sexes of samples, default NULL (character)
`pheno`	-> vector of phenotypes of samples, default NULL (character)
`chrom_code`	-> how CHROM is written: "auto" (keep numeric CHROM, otherwise 0), "index" (1..n by first appearance), "zero" (all 0) or "keep" (verbatim), default "auto" (character)

Details

This function converts a VCFArrow object to an external PLINK .ped formatted file. Writing occurs in chunks whose size is determined by the read_vcf() function. Larger chunks result in faster writing speeds. If no groups are defined, the default behavior is to use all groups. Sex and phenotype vectors are optional. If not defined sex = 0, pheno = -9.

Value

Invisibly returns the input VCFArrow object; called for its side effect of writing `out_file`.

Author(s)

Tomas Hrbek May 2026

Examples

```
f <- system.file("extdata", "vaillantii_discosnp_sub.vcf.gz",
                 package = "VCFArrow")
vcf <- read_vcf(f) |> set_vcf_groups(dirname(f))
vcf2plink_ped(vcf, out_file = file.path(tempdir(), "plink_out"))
```

`vcf2related` *vcf2related*

Description

Converts a VCFArrow object to Related format infile

Usage

```
vcf2related(vcf_arrow, out_file, keep_groups = NULL)
```

Arguments

`vcf_arrow`	-> VCFArrow object
`out_file`	-> name of file to output, no default (character)
`keep_groups`	-> groups to retain, default NULL (character)

Details

This function converts a VCFArrow object to an external Related formatted file. Writing occurs in chunks whose size is determined by the read_vcf() function. Larger chunks result in faster writing speeds. If no groups are defined, the default behavior is to use all groups.

Value

Invisibly returns the input VCFArrow object; called for its side effect of writing `out_file`.

Author(s)

Tomas Hrbek May 2026

Examples

```
f <- system.file("extdata", "vaillantii_discosnp_sub.vcf.gz",
                 package = "VCFArrow")
vcf <- read_vcf(f) |> set_vcf_groups(dirname(f))
vcf2related(vcf, out_file = tempfile(fileext = ".txt"))
```

---

vcf2smartsnp *vcf2smartsnp*

Description

Converts a VCFArrow object to smartsnp tabular format infile

Usage

```
vcf2smartsnp(vcf_arrow, out_file, keep_groups = NULL)
```

Arguments

`vcf_arrow`	-> VCFArrow object
`out_file`	-> name of file to output, no default (character)
`keep_groups`	-> groups to retain, default NULL (character)

Details

This function converts a VCFArrow object to an external SmartSNP formatted file. Writing occurs in chunks whose size is determined by the read_vcf() function. Larger chunks result in faster writing speeds. If no groups are defined, the default behavior is to use all groups.

Value

Invisibly returns the input VCFArrow object; called for its side effect of writing `out_file`.

Author(s)

Tomas Hrbek May 2026

Examples

```
f <- system.file("extdata", "vaillantii_discosnp_sub.vcf.gz",
                 package = "VCFArrow")
vcf <- read_vcf(f) |> set_vcf_groups(dirname(f))
vcf2smartsnp(vcf, out_file = tempfile(fileext = ".txt"))
```

vcf2snapp *vcf2snapp*

Description

Converts a VCFArrow object to a SNAPP encoded NEXUS format infile

Usage

```
vcf2snapp(vcf_arrow, out_file, keep_groups = NULL)
```

Arguments

vcf_arrow	-> VCFArrow object
out_file	-> name of file to output, no default (character)
keep_groups	-> groups to retain, default NULL (character)

Details

This function converts a VCFArrow object to an external SNAPP/SNAPPER formatted file. Writing occurs in chunks whose size is determined by the read_vcf() function. Larger chunks result in faster writing speeds. If no groups are defined, the default behavior is to use all groups.

Value

Invisibly returns the input VCFArrow object; called for its side effect of writing `out_file`.

Author(s)

Tomas Hrbek May 2026

Examples

```
f <- system.file("extdata", "vaillantii_discosnp_sub.vcf.gz",
                 package = "VCFArrow")
vcf <- read_vcf(f) |> set_vcf_groups(dirname(f))
vcf2snapp(vcf, out_file = tempfile(fileext = ".nex"))
```

vcf2snmf *vcf2snmf*

Description

Converts a VCFArrow object to an sNMF format infile

Usage

```
vcf2snmf(vcf_arrow, out_file, keep_groups = NULL)
```

Arguments

vcf_arrow	-> VCFArrow object
out_file	-> name of file to output, no default (character)
keep_groups	-> groups to retain, default NULL (character)

Details

This function converts a VCFArrow object to an external sNMF formatted file. Writing occurs in chunks whose size is determined by the read_vcf() function. Larger chunks result in faster writing speeds. If no groups are defined, the default behavior is to use all groups.

Value

Invisibly returns the input VCFArrow object; called for its side effect of writing `out_file`.

Author(s)

Tomas Hrbek May 2026

Examples

```
f <- system.file("extdata", "vaillantii_discosnp_sub.vcf.gz",
                 package = "VCFArrow")
vcf <- read_vcf(f) |> set_vcf_groups(dirname(f))
vcf2snmf(vcf, out_file = tempfile(fileext = ".geno"))
```

vcf2structure *vcf2structure*

Description

Converts a VCFArrow object to a Structure or FastStructure infile

Usage

```
vcf2structure(vcf_arrow, out_file, keep_groups = NULL, method = "S")
```

Arguments

`vcf_arrow`	-> VCFArrow object
`out_file`	-> name of file to output, no default (character)
`keep_groups`	-> groups to retain, default NULL (character)
`method`	-> flag for Structure/FastStructure formats, default 'S' (character)

Details

This function converts a VCFArrow object to an external Structure or FastStructure formatted file. Writing occurs in chunks whose size is determined by the read_vcf() function. Larger chunks result in faster writing speeds. If no groups are defined, the default behavior is to use all groups. The flag parameter controls whether Structure (flag = 'S') or FastStructure (flag = 'F') formatted output is written out.

Value

Invisibly returns the input VCFArrow object; called for its side effect of writing `out_file`.

Author(s)

Tomas Hrbek May 2026

Examples

```
f <- system.file("extdata", "vaillantii_discosnp_sub.vcf.gz",
                 package = "VCFArrow")
vcf <- read_vcf(f) |> set_vcf_groups(dirname(f))
vcf2structure(vcf, out_file = tempfile(fileext = ".str"))
vcf2structure(vcf, out_file = tempfile(fileext = ".fstr"),
              method = "F")
```

vcf2treemix *vcf2treemix*

Description

Converts a VCFArrow object to Treemix format infile

Usage

```
vcf2treemix(vcf_arrow, out_file, keep_groups = NULL)
```

Arguments

vcf_arrow	-> VCFArrow object
out_file	-> name of file to output, no default (character)
keep_groups	-> groups to retain, default NULL (character)

Details

This function converts a VCFArrow object to an external Treemix formatted file. Writing occurs in chunks whose size is determined by the read_vcf() function. Larger chunks result in faster writing speeds. If no groups are defined, the default behavior is to use all groups.

Value

Invisibly returns the input VCFArrow object; called for its side effect of writing `out_file`.

Author(s)

Tomas Hrbek May 2026

Examples

```
f <- system.file("extdata", "vaillantii_discosnp_sub.vcf.gz",
                 package = "VCFArrow")
vcf <- read_vcf(f) |> set_vcf_groups(dirname(f))
vcf2treemix(vcf, out_file = tempfile(fileext = ".txt"))
```

VCFArrow-class — *VCFArrow S4 class*

Description

S4 class holding VCF data. Variant metadata are kept in memory while genotypes are stored on disk as an Apache Arrow dataset and loaded lazily. Objects are created with `read_vcf()`.

Slots

`header` Complete VCF header (character).

`info` INFO field of each variant (character).

`format` FORMAT field of each variant (data.frame).

`variants` CHROM, POS, ID, REF, ALT, QUAL, FILTER and precalculated per-variant metrics (data.frame).

`gt` Arrow dataset of genotypes in long format.

`samples` Sample names (character).

`groups` Group assignment of each sample (character).

`path` Location of the on-disk Arrow dataset (character).

`finalizer_env` Environment used to clean up `path` on garbage collection.

`loci` CHROM, POS, REF and ALT of every variant of the source VCF, row i describing .row_id i; never filtered, so variants can be identified (e.g. by vcf_bind()) even after they were removed from `variants` (data.frame).

`invariant_removed` Variants removed as invariant, with their metadata (data.frame).

Author(s)

Tomas Hrbek April 2026

write_vcf — *write_vcf*

Description

Write a VCFArrow object to an external VCF file

Usage

```
write_vcf(vcf_arrow, out_file, gzip = FALSE)
```

Arguments

`vcf_arrow`	-> VCFArrow object
`out_file`	-> name of the VCF file to be written to, no default (character)
`gzip`	-> a flag to GZIP VCF when writing, default FALSE (Boolean)

Details

This function writes a VCFArrow object to an external VCF file. Writing occurs in chunks whose size is determined by the read_vcf() function. Larger chunks result in faster writing speeds. It writes both uncompressed and gz compressed files. Compressing increases writing time be about 50%. For large files, it is recommended to output an uncompressed VCF file, and then compress with GZIP or PIGZ.

Value

Invisibly returns the path of the written file.

Author(s)

Tomas Hrbek April 2026

Examples

```
f <- system.file("extdata", "vaillantii_discosnp_sub.vcf.gz",
                 package = "VCFArrow")
vcf <- read_vcf(f) |> set_vcf_groups(dirname(f))
write_vcf(vcf, out_file = tempfile(fileext = ".vcf"))
write_vcf(vcf, out_file = tempfile(fileext = ".vcf.gz"),
          gzip = TRUE)
```

Index

[,VCFArrow-method, 3

assess_vcf_coverage, 4
assess_vcf_missing_data, 5

read_vcf, 6
read_vcf(), *57*

set_vcf_groups, 7
show,VCFArrow-method, 8

vcf2admixture, 35
vcf2apparent, 37
vcf2arlequin, 38
vcf2bayesass, 39
vcf2bayescan, 40
vcf2eigenstrat, 41
vcf2fasta, 42
vcf2fineradstructure, 43
vcf2genepop, 44
vcf2genlight, 45
vcf2gt_long, 46
vcf2migrate, 47
vcf2nexus, 48
vcf2plink_bed, 49
vcf2plink_ped, 50
vcf2related, 51
vcf2smartsnp, 52
vcf2snapp, 53
vcf2snmf, 54
vcf2structure, 55
vcf2treemix, 56
vcf_bind, 8
vcf_copy, 10
vcf_extract_groups, 11
vcf_extract_samples, 12
vcf_filter_adr, 13
vcf_filter_biallelic, 14
vcf_filter_coverage, 15
vcf_filter_hets, 16
vcf_filter_indels, 17
vcf_filter_invariant, 17
vcf_filter_maf, 18
vcf_filter_missing, 19
vcf_filter_missingness, 20
vcf_filter_multiSNV, 21
vcf_filter_oneSNV, 22
vcf_filter_paralogs, 23
vcf_filter_pass, 24
vcf_filter_quality, 25
vcf_filter_rank, 26
vcf_filter_rpt, 27
vcf_gc, 28
vcf_memory_estimate, 29
vcf_set_workers, 30
vcf_stats, 31
vcf_sub_SNVs, 32
vcf_sub_SNVs_stratified, 33
vcf_suggest_chunk_size, 34
vcf_theta, 34
VCFArrow-class, 57

write_vcf, 57