Package 'SNPfiltR'

September 4, 2025

Title Interactively Filter SNP Datasets

Version 1.0.7

Description Is designed to interactively and reproducibly visualize and filter SNP (single-nucleotide polymorphism) datasets. This R-based implementation of SNP and genotype filters facilitates an interactive and iterative SNP filtering pipeline, which can be documented reproducibly via 'rmarkdown'. 'SNPfiltR' contains functions for visualizing various quality and missing data metrics for a SNP dataset, and then filtering the dataset based on user specified cutoffs. All functions take 'vcfR' objects as input, which can easily be generated by reading standard vcf (variant call format) files into R using the R package 'vcfR' authored by Knaus and Grünwald (2017) <doi:10.1111/1755-0998.12549>. Each 'SNPfiltR' function can return a newly filtered 'vcfR' object, which can then be written to a local directory in standard vcf format using the 'vcfR' package, for downstream population genetic and phylogenetic analyses.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

Imports vcfR, ggplot2, Rtsne, cluster, adegenet, gridExtra, ggridges, stats, graphics, utils

Depends R (>= 2.10)

Suggests rmarkdown, knitr

VignetteBuilder knitr

NeedsCompilation no

Author Devon DeRaad [aut, cre] (ORCID: <https://orcid.org/0000-0003-3105-985X>)

Maintainer Devon DeRaad <devonderaad@gmail.com>

Repository CRAN

Date/Publication 2025-09-04 06:40:39 UTC

Contents

`assess_missing_data_pca` *Vizualise how missing data thresholds affect sample clustering*

Description

This function can be run in two ways: 1) Without 'thresholds' specified. This will run a PCA for the input vcf without filtering, and visualize the clustering of samples in two-dimensional space, coloring each sample according to a priori population assignment given in the popmap. 2) With 'thresholds' specified. This will filter your input vcf file to the specified missing data thresholds, and run a PCA for each filtering iteration. For each iteration, a 2D plot will be output showing clustering according to the specified popmap. This option is ideal for assessing the effects of missing data on clustering patterns.

Usage

```
assess_missing_data_pca(
  vcfR,
  popmap = NULL,
  thresholds = NULL,
  clustering = TRUE
)
```

Arguments

`vcfR`	a vcfR object
`popmap`	set of population assignments that will be used to color code the plots
`thresholds`	optionally specify a vector of missing data filtering thresholds to explore
`clustering`	use partitioning around medoids (PAM) to do unsupervised clustering on the output? (default = TRUE, max clusters = # of levels in popmap + 2)

Value

a series of plots showing the clustering of all samples in two-dimensional space

Examples

```
assess_missing_data_pca(vcfR = SNPfiltR::vcfR.example,
popmap = SNPfiltR::popmap,
thresholds = c(.6,.8))
```

---

assess_missing_data_tsne

Vizualise how missing data thresholds affect sample clustering

---

Description

This function can be run in two ways: 1) Without 'thresholds' specified. This will run t-SNE for the input vcf without filtering, and visualize the clustering of samples in two-dimensional space, coloring each sample according to a priori population assignment given in the popmap. 2) With 'thresholds' specified. This will filter your input vcf file to the specified missing data thresholds, and run a t-SNE clustering analysis for each filtering iteration. For each iteration, a 2D plot will be output showing clustering according to the specified popmap. This option is ideal for assessing the effects of missing data on clustering patterns.

Usage

```
assess_missing_data_tsne(
  vcfR,
  popmap = NULL,
  thresholds = NULL,
  perplexity = NULL,
  iterations = NULL,
  initial_dims = NULL,
  clustering = TRUE
)
```

Arguments

vcfR	a vcfR object
popmap	set of population assignments that will be used to color code the plots
thresholds	a vector specifying the missing data filtering thresholds to explore
perplexity	numerical value specifying the perplexity paramter during t-SNE (default: 5)
iterations	a numerical value specifying the number of iterations for t-SNE (default: 1000)
initial_dims	a numerical value specifying the number of initial_dimensions for t-SNE (default: 5)
clustering	use partitioning around medoids (PAM) to do unsupervised clustering on the output? (default = TRUE, max clusters = # of levels in popmap + 2)

Value

a series of plots showing the clustering of all samples in two-dimensional space

Examples

```
assess_missing_data_tsne(vcfR = SNPfiltR::vcfR.example,
popmap = SNPfiltR::popmap,
thresholds = .8)
```

`distance_thin` *Filter a vcf file based on distance between SNPs on a given scaffold*

Description

This function requires a vcfR object as input, and returns a vcfR object filtered to retain only SNPs greater than a specified distance apart on each scaffold. The function starts by automatically retaining the first SNP on a given scaffold, and then subsequently keeping the next SNP that is greater than the specified distance away, until it reaches the end of the scaffold/chromosome. This function scales well with an increasing number of SNPs, but poorly with an increasing number of scaffolds/chromosomes. For this reason, there is a built in progress bar, to monitor potentially long-running executions with many scaffolds. This type of filtering is often employed to reduce linkage among input SNPs, especially for downstream input to programs like structure, which require unlinked SNPs.

Usage

```
distance_thin(vcfR, min.distance = NULL)
```

Arguments

`vcfR`	a vcfR object
`min.distance`	a numeric value representing the smallest distance (in base-pairs) allowed between SNPs after distance thinning

Value

An identical vcfR object, except that SNPs separated by less than the specified distance have been removed from the file

Examples

```
distance_thin(vcfR = SNPfiltR::vcfR.example, min.distance = 1000)
```

filter_allele_balance *Filter out heterozygous genotypes failing an allele balance check*

Description

This function requires a vcfR object as input, and returns a vcfR object filtered to convert heterozygous sites with an allele balance falling outside of the specified ratio to 'NA'. If no ratio is specified, a default .25-.75 limit is imposed. From the dDocent filtering page "Allele balance: a number between 0 and 1 representing the ratio of reads showing the reference allele to all reads, considering only reads from individuals called as heterozygous, we expect that the allele balance in our data (for real loci) should be close to 0.5".

Usage

```
filter_allele_balance(vcfR, min.ratio = NULL, max.ratio = NULL)
```

Arguments

vcfR	a vcfR object
min.ratio	minumum allele ratio for a called het
max.ratio	maximum allele ratio for a called het

Value

An identical vcfR object, except that genotypes failing the allele balance filter have been converted to 'NA'.

Examples

```
filter_allele_balance(vcfR = SNPfiltR::vcfR.example)
```

filter_biallelic *Remove SNPs with more than two alleles*

Description

This function simply removes any SNPs from the vcf file which contains more than two alleles. Many downstream applications require SNPs to be biallelic, so this filter is generally a good idea during processing.

Usage

```
filter_biallelic(vcfR)
```

Arguments

vcfR	a vcfR object

Value

a vcfR object with SNPs containing more than two alleles removed

Examples

```
filter_biallelic(vcfR = SNPfiltR::vcfR.example)
```

---

hard_filter *Hard filter a vcf file by depth and genotype quality (gq)*

Description

This function requires a vcfR object as input. The user can then specify the minimum value for depth of coverage required to retain a called genotype (must be numeric). Additionally, the user can specify a minimum genotype quality required to retain a called genotype (again, must be numeric).

Usage

```
hard_filter(vcfR, depth = NULL, gq = NULL)
```

Arguments

vcfR	a vcfR object
depth	an integer representing the minimum depth for genotype calls that you wish to retain (e.g. 'depth = 5' would remove all genotypes with a sequencing depth of 4 reads or less)
gq	an integer representing the minimum genotype quality for genotype calls that you wish to retain (e.g. 'gq = 30' would remove all genotypes with a quality score of 29 or lower)

Value

The vcfR object input, with the sites failing specified filters converted to 'NA'

Examples

```
hard_filter(vcfR = SNPfiltR::vcfR.example, depth = 5)
hard_filter(vcfR = SNPfiltR::vcfR.example, depth = 5, gq = 30)
```

max_depth *Vizualise and filter based on mean depth across all called SNPs*

Description

This function can be run in two ways: 1) specify vcfR object only. This will visualize the distribution of mean depth per sample across all SNPs in your vcf file, and will not alter your vcf file. 2) specify vcfR object, and set 'maxdepth' = 'integer value'. This option will show you where your specified cutoff falls in the distribution of SNP depth, and remove all SNPs with a mean depth above the specified threshold from the vcf. Super high depth loci are likely multiple loci stuck together into a single paralogous locus. Note: This function filters on a 'per SNP' basis rather than a 'per genotype' basis, otherwise it would disproportionately remove genotypes from our deepest sequenced samples (because sequencing depth is so variable between samples).

Usage

```
max_depth(vcfR, maxdepth = NULL)
```

Arguments

vcfR	a vcfR object
maxdepth	an integer specifying the maximum mean depth for a SNP to be retained

Value

The vcfR object input, with SNPs above the 'maxdepth' cutoff removed

Examples

```
max_depth(vcfR = SNPfiltR::vcfR.example)
max_depth(vcfR = SNPfiltR::vcfR.example, maxdepth = 100)
```

min_mac *Vizualise, filter based on Minor Allele Count (MAC)*

Description

This function can be run in two ways: 1) Without 'min.mac' specified. This will return a folded site frequency spectrum (SFS), without performing any filtering on the vcf file. Or 2) With 'min.mac' specified. This will also print the folded SFS and show you where your specified min. MAC count falls. It will then return your vcfR object with SNPs falling below your min. MAC threshold removed. Note: previous filtering steps (especially removing samples) may have resulted in invariant SNPs (MAC =0). For this reason it's a good idea to run min_mac(vcfR, min.mac=1) before using a SNP dataset in downstream analyses.

Usage

```
min_mac(vcfR, min.mac = NULL)
```

Arguments

vcfR	a vcfR object
min.mac	an integer specifying the minimum minor allele count for a SNP to be retained (e.g. 'min.mac=3' would remove all SNPs with a MAC of 2 or less)

Value

if 'min.mac' is not specified, the allele frequency spectrum is returned. If 'min.mac' is specified, SNPs falling below the MAC cutoff will be removed, and the filtered vcfR object will be returned.

Examples

```
min_mac(vcfR=SNPfiltR::vcfR.example)
```

---

missing_by_sample *Vizualise missing data per sample, remove samples above a missing data cutoff*

Description

This function can be run in two ways: 1) Without 'cutoff' specified. This will vizualise the amount of missing data in each sample across a variety of potential missing data cutoffs. Additionally, it will show you a dotplot ordering the amount of overall missing data in each sample. Based on these visualizations, you can make an informed decision on what you think might be an optimal cutoff to remove samples that are missing too much data to be retained for downstream analyses. 2) with 'cutoff' specified. This option will show you the dotplot with the cutoff you set, and then remove samples above the missing data cutoff you set, and return the filtered vcf to you.

Usage

```
missing_by_sample(vcfR, popmap = NULL, cutoff = NULL)
```

Arguments

vcfR	a vcfR object
popmap	if specifies, it must be a two column dataframe with columns names 'id' and 'pop'. IDs must match the IDs in the vcfR object
cutoff	a numeric value between 0-1 specifying the maximum proportion of missing data allowed in a sample to be retained for downstream analyses

Details

Note: This decision is highly project specific, but these visualizations should help you get a feel for how very low data samples cannot be rescued simply by a missing data SNP filter. If you want to remove specific samples from your vcf that cannot be specified with a simple cutoff refer to this great tutorial which is the basis for the code underlying this function.

Value

if 'cutoff' is not specified, will return a dataframe containing the average depth and proportion missing data in each sample. If 'cutoff' is specified, the samples falling above the missing data cutoff will be removed, and the filtered vcfR object will be returned.

Examples

```
missing_by_sample(vcfR = SNPfiltR::vcfR.example)
missing_by_sample(vcfR = SNPfiltR::vcfR.example, cutoff = .7)
```

---

missing_by_snp *Vizualise missing data per SNP, remove SNPs above a missing data cutoff*

Description

This function can be run in two ways: 1) Without 'cutoff' specified. This will vizualise the amount of missing data in each sample across a variety of potential missing data cutoffs. Additionally, it will show you dotplots visualizing the number of total SNPs retained across a variety of filtering cutoffs, and the total proportion of missing data. Based on these visualizations, you can make an informed decision on what you think might be an optimal cutoff to minimize the overall missingness of your dataset while still retaining an appropriate amount of SNPs for the downstream inferences you hope to make 2) with 'cutoff' specified. This option will show you the dotplots with the cutoff you set, and then remove SNPs above the missing data cutoff.

Usage

```
missing_by_snp(vcfR, cutoff = NULL)
```

Arguments

vcfR	a vcfR object
cutoff	a numeric value between 0-1 specifying the maximum proportion of missing data allowed in a SNP to be retained for downstream analyses

Value

if 'cutoff' is not specified, will return a dataframe containing the proportion missing data and the total SNPs retained across each filtering level. If 'cutoff' is specified, SNPs falling above the missing data cutoff will be removed, and the filtered vcfR object will be returned.

Examples

```
missing_by_snp(vcfR = SNPfiltR::vcfR.example)
missing_by_snp(vcfR = SNPfiltR::vcfR.example, cutoff = .6)
```

---

popmap *Popmap for example scrub-jay vcfR file*

Description

A dataset containing the sample name and population assignment for the 20 scrub-jay samples in SNPfilR::vcfR.example . The variables are as follows:

Usage

```
data(popmap)
```

Format

A data frame with 20 rows and 2 variables

Details

- id. unique sample identifier
- pop. population assignment for each individual

SNPfiltR *SNPfiltR: A package for interactively visualizing and filtering SNP datasets*

Description

The SNPfiltR package allows users to interactively visualize the effects of relevant filters on their datasets in order to optimize filtering parameters rather than simply following historical precedent. Each function takes a variant call format (vcf) file, stored in local memory as a vcfR object, as input. Most functions can be run without specified cutoffs, in order to visualize the distribution of the parameter of interest in your given dataset. Then the same function can be run with a specified cutoff, and a filtered vcfR object will be returned. For detailed documentation and vignettes showing fully implemented SNP filtering pipelines, please go to: devonderaad.github.io/SNPfiltR

vcfR.example *Example scrub-jay vcfR file*

Description

A vcfR object containing 500 SNPs for 20 individuals. Species assignments for each individual can be accessed via SNPfiltR::popmap

Usage

```
data(vcfR.example)
```

Format

A vcfR object containing 500 SNPs for 20 individuals

Index