Package: tidypopgen (via r-universe)

September 13, 2024

Title Tidy Population Genetics

Version 0.0.0.9016

Description We provide a tidy grammar of population genetics, facilitating the manipulation and analysis of data on biallelic single nucleotide polymorphisms (SNPs).

License GPL (>= 3)

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Depends R (>= 3.0.2), dplyr, tibble

Imports bigparallelr, bigsnpr, bigstatsr, forcats, foreach, generics, ggplot2, magrittr, methods, MASS, patchwork, rlang, stats, stringr, tidyselect, tidyr, utils, Rcpp, UpSetR, vctrs

Suggests adegenet, admixtools, broom, hierfstat, knitr, detectRUNS, LEA, rmarkdown, readr, testthat (>= 3.0.0), vcfR

Remotes uqrmaie1/admixtools

VignetteBuilder knitr

Config/testthat/edition 3

LinkingTo Rcpp, RcppArmadillo (>= 0.9.600), bigstatsr, rmio

LazyData true

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

RemoteUrl https://github.com/EvolEcolGroup/tidypopgen

RemoteRef main

RemoteSha 7c5ef31f704c7a6f59fcdeccf39dda6fd376c718

Contents

`as_q_matrix` *Convert a Q matrix into a* `q_matrix` *obejct*

Description

Takes a matrix of Q values, check its validity, and then formats it correctly to make sure it can then be processed and plotted correctly

Usage

```
as_q_matrix(x)
```

Arguments

`x`	a matrix

Value

a `q_matrix` object, which is a matrix with appropriate column names (.QX, where X is the component number) to use with plotting

augment.gt_dapc *Augment data with information from a gt_dapc object*

Description

Augment for gt_dapc accepts a model object and a dataset and adds scores to each observation in the dataset. Scores for each component are stored in a separate column, which is given name with the pattern ".fittedLD1", ".fittedLD2", etc. For consistency with broom::augment.prcomp, a column ".rownames" is also returned; it is a copy of 'id', but it ensures that any scripts written for data augmented with broom::augment.prcomp will work out of the box (this is especially helpful when adapting plotting scripts).

Usage

```
## S3 method for class 'gt_dapc'
augment(x, data = NULL, k = NULL, ...)
```

Arguments

x	A gt_dapc object returned by gt_dapc().
data	the gen_tibble used to run the PCA.
k	the number of components to add
...	Not used. Needed to match generic signature only.

Value

A gen_tibble containing the original data along with additional columns containing each observation's projection into PCA space.

See Also

gt_dapc() gt_dapc_tidiers

augment_gt_pca *Augment data with information from a gt_pca object*

Description

Augment for gt_pca accepts a model object and a dataset and adds scores to each observation in the dataset. Scores for each component are stored in a separate column, which is given name with the pattern ".fittedPC1", ".fittedPC2", etc. For consistency with broom::augment.prcomp, a column ".rownames" is also returned; it is a copy of 'id', but it ensures that any scripts written for data augmented with broom::augment.prcomp will work out of the box (this is especially helpful when adapting plotting scripts).

Usage

```
## S3 method for class 'gt_pca'
augment(x, data = NULL, k = NULL, ...)
```

Arguments

x	A gt_pca object returned by one of the gt_pca_* functions.
data	the gen_tibble used to run the PCA.
k	the number of components to add
...	Not used. Needed to match generic signature only.

Value

A gen_tibble containing the original data along with additional columns containing each observation's projection into PCA space.

See Also

gt_pca_autoSVD() gt_pca_tidiers

---

augment_loci *Augment the loci table with information from a analysis object*

Description

augment_loci add columns to the loci table of a gen_tibble related to information from a given analysis.

Usage

```
augment_loci(x, data, ...)
```

Arguments

x	An object returned by one of the gt_ functions (e.g. gt_pca()).
data	the gen_tibble used to run the PCA.
...	Additional parameters passed to the individual methods.

Value

A gen_tibble with additional columns added to the loci tibble (accessible with show_loci(). If data is missing, a tibble of the information, with a column .rownames giving the loci names.

augment_loci_gt_pca *Augment the loci table with information from a gt_pca object*

Description

Augment for gt_pca accepts a model object and a gen_tibble and adds loadings for each locus to the loci table. Loadings for each component are stored in a separate column, which is given name with the pattern ".loadingPC1", ".loadingPC2", etc. If data is missing, then a tibble with the loadings is returned.

Usage

```
## S3 method for class 'gt_pca'
augment_loci(x, data = NULL, k = NULL, ...)
```

Arguments

x	A gt_pca object returned by one of the gt_pca_* functions.
data	the gen_tibble used to run the PCA.
k	the number of components to add
...	Not used. Needed to match generic signature only.

Value

A gen_tibble with a loadings added to the loci tibble (accessible with show_loci(). If data is missing, a tibble of loadings.

See Also

gt_pca_autoSVD() gt_pca_tidiers

autoplot.gt_cluster_pca *Autoplots for* gt_cluster_pca *objects*

Description

For gt_cluster_pca, autoplot produces a plot of a metric of choice ('BIC', 'AIC' or 'WSS') against the number of clusters (k). This plot is can be used to infer the best value of k, which corresponds to the smallest value of the metric (the minimum in an 'elbow' shaped curve). In some cases, there is not 'elbow' and the metric keeps decreasing with increasing k; in such cases, it is customary to choose the value of k at which the decrease in the metric reaches as plateau. For a programmatic way of choosing k, use gt_cluster_pca_best_k().

Usage

```
## S3 method for class 'gt_cluster_pca'
autoplot(object, metric = c("BIC", "AIC", "WSS"), ...)
```

Arguments

object	an object of class gt_dapc
metric	the metric to plot on the y axies, one of 'BIC', 'AIC', or 'WSS' (with sum of squares)
...	not currently used.

Details

autoplot produces simple plots to quickly inspect an object. They are not customisable; we recommend that you use ggplot2 to produce publication ready plots.

Value

a ggplot2 object

---

autoplot.gt_dapc *Autoplots for* gt_dapc *objects*

Description

For gt_dapc, the following types of plots are available:

- screeplot: a plot of the eigenvalues of the discriminant axes
- scores a scatterplot of the scores of each individual on two discriminant axes (defined by ld)
- loadings a plot of loadings of all loci for a discriminant axis (chosen with ld)
- components a bar plot showing the probability of assignment to each cluster

Usage

```
## S3 method for class 'gt_dapc'
autoplot(
  object,
  type = c("screeplot", "scores", "loadings", "components"),
  ld = NULL,
  group = NULL,
  n_col = 1,
  ...
)
```

Arguments

object	an object of class gt_dapc
type	the type of plot (one of "screeplot", "scores" and "loadings")
ld	the principal components to be plotted: for scores, a pair of values e.g. c(1,2); for loadings either one or more values.
group	a vector of group memberships to order the individuals in "components" plot. If NULL, the clusters used for the DAPC will be used.
n_col	for loadings plots, if multiple LD axis are plotted, how many columns should be used.
...	not currently used.

Details

autoplot produces simple plots to quickly inspect an object. They are not customisable; we recommend that you use ggplot2 to produce publication ready plots.

Value

a ggplot2 object

---

autoplot.qc_report_indiv

Autoplots for qc_report_indiv *objects*

Description

For qc_report_indiv, the following types of plots are available:

- scatter: a plot of missingness and observed heterozygosity within individuals.
- relatedness: a histogram of paired kinship coefficients

Usage

```
## S3 method for class 'qc_report_indiv'
autoplot(
  object,
  type = c("scatter", "relatedness"),
  miss_threshold = NULL,
  kings_threshold = kings_threshold,
  ...
)
```

Arguments

object	an object of class qc_report_indiv
type	the type of plot (scatter,relatedness)
miss_threshold	a threshold for the accepted rate of missingness within individuals
kings_threshold	an optional numeric, a threshold of relatedness for the sample
...	not currently used.

Details

autoplot produces simple plots to quickly inspect an object. They are not customisable; we recommend that you use ggplot2 to produce publication ready plots.

Value

a ggplot2 object

---

autoplot.qc_report_loci

Autoplots for qc_report_loci *objects*

Description

For qc_report_loci, the following types of plots are available:

- overview: an UpSet plot, giving counts of snps over the threshold for missingness, minor allele frequency, and Hardy-Weinberg equilibrium P-value, and visualising the interaction between these
- all: a four panel plot, containing missing high maf, missing low maf, hwe, and significant hwe plots
- missing: a histogram of proportion of missing data
- missing low maf: a histogram of the proportion of missing data for snps with low minor allele freqency
- missing high maf:a histogram of the proportion of missing data for snps with high minor allele freqency
- maf: a histogram of minor allele frequency
- hwe: a histogram of HWE exact test p-values
- significant hwe: a histogram of significant HWE exact test p-values

Usage

```
## S3 method for class 'qc_report_loci'
autoplot(
  object,
  type = c("overview", "all", "missing", "missing low maf", "missing high maf", "maf",
    "hwe", "significant hwe"),
  maf_threshold = NULL,
  miss_threshold = NULL,
  hwe_p = NULL,
  ...
)
```

Arguments

object	an object of class qc_report_loci
type	the type of plot (one of overview, all, missing, missing low maf, missing high maf, maf, hwe, and significant hwe)
maf_threshold	a threshold for the accepted rate of minor allele frequency of loci
miss_threshold	a threshold for the accepted rate of missingness per loci
hwe_p	a threshold of significance for Hardy-Weinberg exact p-values
...	not currently used.

Details

autoplot produces simple plots to quickly inspect an object. They are not customisable; we recommend that you use ggplot2 to produce publication ready plots.

Value

a ggplot2 object

---

autoplot.q_matrix *Autoplots for* q_matrix *objects*

Description

Autoplots for q_matrix objects

Usage

```
## S3 method for class 'q_matrix'
autoplot(object, data = NULL, annotate_group = TRUE, ...)
```

Arguments

object	A Q matrix object (as returned by as_q_matrix()).
data	An associated tibble (e.g. a gen_tibble), with the individuals in the same order as the data used to generate the Q matrix
annotate_group	Boolean determining whether to annotate the plot with the group information
...	not currently used.

Value

a barplot of individuals, coloured by ancestry proportion

---

autoplot_gt_pca *Autoplots for* gt_pca *objects*

Description

For gt_pca, the following types of plots are available:

- screeplot: a plot of the eigenvalues of the principal components (currently it plots the singular value)
- scores a scatterplot of the scores of each individual on two principal components (defined by pc)
- loadings a plot of loadings of all loci for a given component (chosen with pc)

Usage

```
## S3 method for class 'gt_pca'
autoplot(object, type = c("screeplot", "scores", "loadings"), k = NULL, ...)
```

Arguments

object	an object of class gt_pca
type	the type of plot (one of "screeplot", "scores" and "loadings")
k	the principal components to be plotted: for scores, a pair of values e.g. c(1,2); for loadings either one or more values.
...	not currently used.

Details

autoplot produces simple plots to quickly inspect an object. They are not customisable; we recommend that you use ggplot2 to produce publication ready plots.

Value

a ggplot2 object

autoplot_gt_pcadapt *Autoplots for* gt_pcadapt *objects*

Description

For gt_pcadapt, the following types of plots are available:

- qq: a qunatile-quantile plot of the p-values from pcadapt (wrapping bigsnpr::snp_qq())
- manhattan a manhattan plot of the p-values from pcadapt (wrapping bigsnpr::snp_manhattan())

Usage

```
## S3 method for class 'gt_pcadapt'
autoplot(object, type = c("qq", "manhattan"), ...)
```

Arguments

object	an object of class gt_pcadapt
type	the type of plot (one of "qq", and "mnahattan")
...	further arguments to be passed to bigsnpr::snp_qq() or bigsnpr::snp_manhattan().

Details

autoplot produces simple plots to quickly inspect an object. They are not customisable; we recommend that you use ggplot2 to produce publication ready plots.

Value

a ggplot2 object

count_loci *Count the number of loci in a* gen_tibble

Description

Count the number of loci in gen_tibble (or directly from its genotype column).

Usage

```
count_loci(.x, ...)

## S3 method for class 'tbl_df'
count_loci(.x, ...)

## S3 method for class 'vctrs_bigSNP'
count_loci(.x, ...)
```

Arguments

.x	a gen_tibble, or a vector of class vctrs_bigSNP (usually the genotype column of a gen_tibble object).
...	currently unused.

Value

the number of loci

---

distruct_colours *Distruct colours*

Description

Colours in the palette used by distruct

Usage

```
distruct_colours
```

Format

A vector of 60 hex colours

---

filter_high_relatedness *Filter individuals based on a relationship threshold*

Description

This function takes a matrix of x by y individuals containing relatedness coefficients and returns the maximum set of individuals that contains no relationships above the given threshold.

Usage

```
filter_high_relatedness(
  matrix,
  .x = NULL,
  kings_threshold = NULL,
  verbose = FALSE
)
```

Arguments

matrix	a square symmetric matrix of individuals containing relationship coefficients
.x	a gen_tibble object
kings_threshold	a threshold over which
verbose	boolean whether to report to screen

Value

a list where '1' is individual ID's to retain, '2' is individual ID's to remove, and '3' is a boolean where individuals to keep are TRUE and individuals to remove are FALSE

---

gen_tibble *Constructor for a* gen_tibble

Description

A gen_tibble stores genotypes for individuals in a tidy format. DESCRIBE here the format

Usage

```
gen_tibble(
  x,
  ...,
  valid_alleles = c("A", "T", "C", "G"),
  missing_alleles = c("0", "."),
  backingfile = NULL,
  quiet = FALSE
)

## S3 method for class 'character'
gen_tibble(
  x,
  ...,
  parser = c("vcfR", "cpp"),
  chunk_size = NULL,
  valid_alleles = c("A", "T", "C", "G"),
  missing_alleles = c("0", "."),
  backingfile = NULL,
  quiet = FALSE
)

## S3 method for class 'matrix'
gen_tibble(
  x,
  indiv_meta,
```

```
  loci,
  ...,
  ploidy = 2,
  valid_alleles = c("A", "T", "C", "G"),
  missing_alleles = c("0", "."),
  backingfile = NULL,
  quiet = FALSE
)
```

Arguments

x	can be: • a string giving the path to a PLINK BED or PED file. The associated BIM and FAM files for the BED, or MAP for PED are expected to be in the same directory and have the same file name. • a string giving the path to a RDS file storing a bigSNP object from the bigsnpr package (usually created with bigsnpr::snp_readBed()) • a string giving the path to a vcf file. Note that we currently read the whole vcf in memory with vcfR, so only smallish vcf can be imported. Only biallelic SNPs will be considered. • a string giving the path to a packedancestry .geno file. The associated .ind and .snp files are expected to be in the same directory and share the same file name prefix. • a genotype matrix of dosages (0, 1, 2, NA) giving the dosage of the alternate allele.
...	if x is the name of a vcf file, additional arguments passed to vcfR::read.vcfR(). Otherwise, unused.
valid_alleles	a vector of valid allele values; it defaults to 'A','T', 'C' and 'G'.
missing_alleles	a vector of values in the BIM file/loci dataframe that indicate a missing value for the allele value (e.g. when we have a monomorphic locus with only one allele). It defaults to '0' and '.' (the same as PLINK 1.9).
backingfile	the path, including the file name without extension, for backing files used to store the data (they will be given a .bk and .RDS automatically). This is not needed if x is already an .RDS file. If x is a .BED file and backingfile is left NULL, the backing file will be saved in the same directory as the bed file, using the same file name but with a different file type (.bk rather than .bed). The same logic applies to .vcf files. If x is a genotype matrix and backingfile is NULL, then a temporary file will be created (but note that R will delete it at the end of the session!)
quiet	provide information on the files used to store the data
parser	the name of the parser used for VCF, either "cpp" to use a fast C++ parser, or "vcfR" to use the R package vcfR. The latter is slower but more robust; if "cpp" gives error, try using "vcfR" in case your VCF has an unusual structure.
chunk_size	the number of loci or individuals (depending on the format) processed at a time (currently used if x is a vcf or packedancestry file)

indiv_meta	a list, data.frame or tibble with compulsory columns 'id' and 'population', plus any additional metadata of interest. This is only used if x is a genotype matrix. Otherwise this information is extracted directly from the files.
loci	a data.frame or tibble, with compulsory columns 'name', 'chromosome', and 'position','genetic_dist', 'allele_ref' and 'allele_alt'. This is only used if x is a genotype matrix. Otherwise this information is extracted directly from the files.
ploidy	the ploidy of the samples (either a single value, or a vector of values for mixed ploidy). Only used if creating a gen_tibble from a matrix of data; otherwise, ploidy is determined automatically from the data as they are read.

Details

When loading packedancestry files, missing alleles will be converted from 'X' to NA

Value

an object of the class gen_tbl.

---

gt_as_genind *Convert a* gen_tibble *to a* genind *object from* adegenet

---

Description

This function converts a gen_tibble to a genind object from adegenet

Usage

```
gt_as_genind(x)
```

Arguments

x	a gen_tibble, with population coded as 'population'

Value

a genind object

gt_as_genlight *Convert a* gen_tibble *to a* genlight *object from* adegenet

Description

This function converts a gen_tibble to a genlight object from adegenet

Usage

```
gt_as_genlight(x)
```

Arguments

x	a gen_tibble, with population coded as 'population'

Value

a genlight object

gt_as_geno_lea *Convert a* gentibble *to a .geno file for sNMF from the LEA package*

Description

This function writes a .geno file fom a gen_tibble. Unless a file path is given, a file with suffix .geno is written in the same location as the .rds and .bk files that underpin the gen_tibble.

Usage

```
gt_as_geno_lea(x, file = NULL)
```

Arguments

x	a gen_tibble
file	the .geno filename with a path, or NULL (the default) to use the location of the backing files.

Value

the path of the .geno file

gt_as_hierfstat *Convert a* gen_tibble *to a data.frame compatible with* hierfstat

Description

This function converts a gen_tibble to a data.frame formatted to be used by hierfstat functions.

Usage

```
gt_as_hierfstat(x)
```

Arguments

x	a gen_tibble, with population coded as 'population'

Value

a data.frame with a column 'pop' and further column representing the genotypes (with alleles recoded as 1 and 2)

gt_as_plink *Export a* gen_tibble *object to PLINK bed format*

Description

This function exports all the information of a gen_tibble object into a PLINK bed, ped or raw file (and associated files, i.e. .bim and .fam for .bed; .fam for .ped).

Usage

```
gt_as_plink(x, file = NULL, type = c("bed", "ped", "raw"), overwrite = TRUE)
```

Arguments

x	a gen_tibble object
file	a character string giving the path to output file. If left to NULL, the output file will have the same path and prefix of the backingfile.
type	one of "bed", "ped" or "raw"
overwrite	boolean whether to overwrite the file.

Value

the path of the saved file

gt_as_vcf *Convert a* gen_tibble *to a VCF*

Description

This function write a VCF from a gen_tibble.

Usage

```
gt_as_vcf(x, file = NULL, chunk_size = NULL, overwrite = FALSE)
```

Arguments

x	a gen_tibble, with population coded as 'population'
file	the .vcf file name with a path, or NULL (the default) to use the location of the backing files.
chunk_size	the number of loci processed at a time. Automatically set if left to NULL
overwrite	logical, should the file be overwritten if it already exists?

Value

the path of the .vcf file

gt_cluster_pca *Run K-clustering on principal components*

Description

This function implements the clustering procedure used in Discriminant Analysis of Principal Components (DAPC, Jombart et al. 2010). This procedure consists in running successive K-means with an increasing number of clusters (k), after transforming data using a principal component analysis (PCA). For each model, several statistical measures of goodness of fit are computed, which allows to choose the optimal k using the function gt_cluster_pca_best_k(). See details for a description of how to select the optimal k and vignette("adegenet-dapc") for a tutorial.

Usage

```
gt_cluster_pca(
  x = NULL,
  n_pca = NULL,
  k_clusters = c(1, round(nrow(x$u)/10)),
  method = c("kmeans", "ward"),
  n_iter = 1e+05,
  n_start = 10,
  quiet = FALSE
)
```

Arguments

x	a gt_pca object returned by one of the gt_pca_* functions.
n_pca	number of principal components to be fed to the LDA.
k_clusters	number of clusters to explore, either a single value, or a vector of length 2 giving the minimum and maximum (e.g. 1:5). If left NULL, it will use 1 to the number of pca components divided by 10 (a reasonable guess).
method	either 'kmeans' or 'ward'
n_iter	number of iterations for kmeans (only used if method="kmeans")
n_start	number of starting points for kmeans (only used if method="kmeans")
quiet	boolean on whether to silence outputting information to the screen (defaults to FALSE)

Value

a gt_cluster_pca object, which is a subclass of gt_pca with an additional element 'cluster', a list with elements:

- 'method' the clustering method (either kmeans or ward)
- 'n_pca' number of principal components used for clustering
- 'k' the k values explored by the function
- 'WSS' within sum of squares for each k
- 'AIC' the AIC for each k
- 'BIC' the BIC for each k
- 'groups' a list, with each element giving the group assignments for a given k

---

gt_cluster_pca_best_k *Find the best number of clusters based on principal components*

Description

This function selects the best k value based on a chosen metric and criterion. It is equivalent to plotting the metric against the *k* values, and selecting the *k* that fulfils a given criterion (see details for an explanation of each criterion). This function simply adds an element 'best_k' to the gt_cluster_pca returned by gt_cluster_pca(). The choice can be over-ridden simply by assigning a different value to that element (e.g. for an object x and a desired *k* of 8, simply use x$best_k <- 8)

Usage

```
gt_cluster_pca_best_k(
  x,
  stat = c("BIC", "AIC", "WSS"),
  criterion = c("diffNgroup", "min", "goesup", "smoothNgoesup", "goodfit"),
  quiet = FALSE
)
```

Arguments

x	a gt_cluster_pca object obtained with gt_cluster_pca()
stat	a statistics, one of "BIC", "AIC" or "WSS"
criterion	one of "diffNgroup", "min", "goesup", "smoothNgoesup", "goodfit", see details for a discussion of each approach.
quiet	boolean on whether to silence outputting information to the screen (defaults to FALSE)

Details

The analysis of data simulated under various population genetics models (see reference) suggested an ad-hoc rule for the selection of the optimal number of clusters. First important result is that BIC seems more efficient than AIC and WSS to select the appropriate number of clusters (see example). The rule of thumb consists in increasing K until it no longer leads to an appreciable improvement of fit (i.e., to a decrease of BIC). In the most simple models (island models), BIC decreases until it reaches the optimal K, and then increases. In these cases, the best rule amounts to choosing the lowest K. In other models such as stepping stones, the decrease of BIC often continues after the optimal K, but is much less steep, so a change in slope can be taken as an indication of where the best *k* lies.

This function provides a programmatic way to select *k*. Note that it is highly recommended to look at the graph of BIC versus the numbers of clusters, to understand and validate the programmatic selection. The criteria available in this function are:

- "diffNgroup": differences between successive values of the summary statistics (by default, BIC) are split into two groups using a Ward's clustering method (see ?hclust), to differentiate sharp decrease from mild decreases or increases. The retained K is the one before the first group switch. This criterion appears to work well for island/hierarchical models, and decently for isolation by distance models, albeit with some unstability. It can be confounded by an initial, very sharp decrease of the test statistics. IF UNSURE ABOUT THE CRITERION TO USE, USE THIS ONE.
- "min": the model with the minimum summary statistics (as specified by stat argument, BIC by default) is retained. Is likely to work for simple island model, using BIC. It is likely to fail in models relating to stepping stones, where the BIC always decreases (albeit by a small amount) as K increases. In general, this approach tends to over-estimate the number of clusters.
- "goesup": the selected model is the K after which increasing the number of clusters leads to increasing the summary statistics. Suffers from inaccuracy, since i) a steep decrease might follow a small 'bump' of increase of the statistics, and ii) increase might never happen, or happen after negligible decreases. Is likely to work only for clear-cut island models.
- "smoothNgoesup": a variant of "goesup", in which the summary statistics is first smoothed using a lowess approach. Is meant to be more accurate than "goesup" as it is less prone to stopping to small 'bumps' in the decrease of the statistics.
- "goodfit": another criterion seeking a good fit with a minimum number of clusters. This approach does not rely on differences between successive statistics, but on absolute fit. It selects the model with the smallest K so that the overall fit is above a given threshold.

Value

a 'gt_cluster_pca' object with an added element 'best_k'

References

Jombart T, Devillard S and Balloux F (2010) Discriminant analysis of principal components: a new method for the analysis of genetically structured populations. BMC Genetics 11:94. doi:10.1186/1471-2156-11-94

`gt_dapc` *Discriminant Analysis of Principal Components for gen_tibble*

Description

This function implements the Discriminant Analysis of Principal Components (DAPC, Jombart et al. 2010). This method describes the diversity between pre-defined groups. When groups are unknown, use `gt_cluster_pca()` to infer genetic clusters. See 'details' section for a succinct description of the method, and the vignette in the package `adegenet` ("adegenet-dapc") for a tutorial. This function returns objects of class `adegenet::dapc` which are compatible with methods from `adegenet`; graphical methods for DAPC are documented in adegenet::scatter.dapc (see ?scatter.dapc).

Usage

```
gt_dapc(
  x,
  pop = NULL,
  n_pca = NULL,
  n_da = NULL,
  loadings_by_locus = TRUE,
  pca_info = FALSE
)
```

Arguments

`x`	an object of class `gt_pca`, or its subclass `gt_cluster_pca`
`pop`	either a factor indicating the group membership of individuals; or an integer defining the desired *k* if x is a `gt_cluster_pca`; or NULL, if 'x' is a `gt_cluster_pca` and contain an element 'best_k', usually generated with `gt_cluster_pca_best_k()`, which will be used to select the clustering level.
`n_pca`	number of principal components to be used in the Discriminant Analysis. If NULL, k-1 will be used.
`n_da`	an integer indicating the number of axes retained in the Discriminant Analysis step.

`loadings_by_locus`
: a logical indicating whether the loadings and contribution of each locus should be stored (TRUE, default) or not (FALSE). Such output can be useful, but can also create large matrices when there are a lot of loci and many dimensions.

`pca_info`
: a logical indicating whether information about the prior PCA should be stored (TRUE, default) or not (FALSE). This information is required to predict group membership of new individuals using predict, but makes the object slightly bigger.

Details

The Discriminant Analysis of Principal Components (DAPC) is designed to investigate the genetic structure of biological populations. This multivariate method consists in a two-steps procedure. First, genetic data are transformed (centred, possibly scaled) and submitted to a Principal Component Analysis (PCA). Second, principal components of PCA are submitted to a Linear Discriminant Analysis (LDA). A trivial matrix operation allows to express discriminant functions as linear combination of alleles, therefore allowing one to compute allele contributions. More details about the computation of DAPC are to be found in the indicated reference.

Value

an object of class adegenet::dapc

References

Jombart T, Devillard S and Balloux F (2010) Discriminant analysis of principal components: a new method for the analysis of genetically structured populations. BMC Genetics 11:94. doi:10.1186/1471-2156-11-94 Thia, J. A. (2023). Guidelines for standardizing the application of discriminant analysis of principal components to genotype data. Molecular Ecology Resources, 23, 523–538. https://doi.org/10.1111/1755-0998.13706

`gt_extract_f2` *Compute and store blocked f2 statistics for ADMIXTOOLS 2*

Description

This function prepares data for various *ADMIXTOOLS 2* functions fro the package *ADMIXTOOLS 2*. It takes a `gen_tibble`, computes allele frequencies and blocked f2-statistics, and writes the results to `outdir`. It is equivalent to `admixtools::extract_f2()`.

Usage

```
gt_extract_f2(
  .x,
  outdir = NULL,
  blgsize = 0.05,
  maxmem = 8000,
```

```
  maxmiss = 0,
  minmaf = 0,
  maxmaf = 0.5,
  minac2 = FALSE,
  outpop = NULL,
  outpop_scale = TRUE,
  transitions = TRUE,
  transversions = TRUE,
  overwrite = FALSE,
  adjust_pseudohaploid = TRUE,
  fst = TRUE,
  afprod = TRUE,
  poly_only = c("f2"),
  apply_corr = TRUE,
  n_cores = 1,
  quiet = FALSE
)
```

Arguments

.x	a gen_tibble
outdir	Directory where data will be stored.
blgsize	SNP block size in Morgan. Default is 0.05 (5 cM). If blgsize is 100 or greater, if will be interpreted as base pair distance rather than centimorgan distance.
maxmem	Maximum amount of memory to be used. If the required amount of memory exceeds maxmem, allele frequency data will be split into blocks, and the computation will be performed separately on each block pair. This doesn't put a precise cap on the amount of memory used (it used to at some point). Set this parameter to lower values if you run out of memory while running this function. Set it to higher values if this function is too slow and you have lots of memory.
maxmiss	Discard SNPs which are missing in a fraction of populations higher than maxmiss
minmaf	Discard SNPs with minor allele frequency less than minmaf
maxmaf	Discard SNPs with minor allele frequency greater than than maxmaf
minac2	Discard SNPs with allele count lower than 2 in any population (default FALSE). This option should be set to TRUE when computing f3-statistics where one population consists mostly of pseudohaploid samples. Otherwise heterozygosity estimates and thus f3-estimates can be biased. minac2 == 2 will discard SNPs with allele count lower than 2 in any non-singleton population (this option is experimental and is based on the hypothesis that using SNPs with allele count lower than 2 only leads to biases in non-singleton populations). Note that, While the minac2 option discards SNPs with allele count lower than 2 in any population, the qp3pop function will only discard SNPs with allele count lower than 2 in the first (target) population (when the first argument is the prefix of a genotype file; i.e. it is applied directly to a genotype file, not via precomputing f2 from a gen_tibble).
outpop	Keep only SNPs which are heterozygous in this population

outpop_scale Scale f2-statistics by the inverse outpop heterozygosity (1/(p*(1-p))). Providing outpop and setting outpop_scale to TRUE will give the same results as the original *qpGraph* when the outpop parameter has been set, but it has the disadvantage of treating one population different from the others. This may limit the use of these f2-statistics for other models.

transitions Set this to FALSE to exclude transition SNPs

transversions Set this to FALSE to exclude transversion SNPs

overwrite Overwrite existing files in outdir

adjust_pseudohaploid
Genotypes of pseudohaploid samples are usually coded as 0 or 2, even though only one allele is observed. adjust_pseudohaploid ensures that the observed allele count increases only by 1 for each pseudohaploid sample. If TRUE (default), samples that don't have any genotypes coded as 1 among the first 1000 SNPs are automatically identified as pseudohaploid. This leads to slightly more accurate estimates of f-statistics. Setting this parameter to FALSE treats all samples as diploid and is equivalent to the *ADMIXTOOLS* inbreed: NO option. Setting adjust_pseudohaploid to an integer n will check the first n SNPs instead of the first 1000 SNPs.

fst Write files with pairwise FST for every population pair. Setting this to FALSE can make extract_f2 faster and will require less memory.

afprod Write files with allele frequency products for every population pair. Setting this to FALSE can make extract_f2 faster and will require less memory.

poly_only Specify whether SNPs with identical allele frequencies in every population should be discarded (poly_only = TRUE), or whether they should be used (poly_only = FALSE). By default (poly_only = c("f2")), these SNPs will be used to compute FST and allele frequency products, but not to compute f2 (this is the default option in the original ADMIXTOOLS).

apply_corr Apply small-sample-size correction when computing f2-statistics (default TRUE)

n_cores Parallelize computation across n_cores cores.

quiet Suppress printing of progress updates

Value

SNP metadata (invisibly)

---

gt_get_file_names *Get the names of files storing the genotypes of a* gen_tibble

---

Description

A function to return the names of the files used to store data in a gen_tibble. Specifically, this returns the .rds file storing the big

Usage

```
gt_get_file_names(x)
```

Arguments

x	a gen_tibble

Value

a character vector with the names and paths of the two files

---

gt_has_imputed *Checks if a* gen_tibble *has been imputed*

Description

This function checks if a dataset has been imputed. Note that having imputation does not mean that the imputed values are used.

Usage

```
gt_has_imputed(x)
```

Arguments

x	a gen_tibble

Value

boolean TRUE or FALSE depending on whether the dataset has been imputed

---

gt_impute_simple *Simple imputation based on allele frequencies*

Description

This function provides a very simple imputation algorithm for gen_tibble objects by using the mode, mean or sampling from the allele frequencies. Each locus is imputed independently (and thus linkage information is ignored). It is a wrapper around bigsnpr::snp_fastImputeSimple().

Usage

```
gt_impute_simple(x, method = c("mode", "mean0", "random"), n_cores = 1)
```

Arguments

x	a gen_tibble with missing data
method	one of • 'mode': the most frequent genotype • 'mean0': the mean rounded to the nearest integer • 'random': randomly sample a genotype based on the observed allele frequencies
n_cores	the number of cores to be used

Value

a gen_tibble with imputed genotypes

---

gt_king *Compute the KING-robust Matrix for a a* gen_tibble *object*

Description

This function computes the KING-robust estimator of kinship.

Usage

```
gt_king(
  x,
  as_matrix = FALSE,
  block_size = bigstatsr::block_size(length(loci_names(x)))
)
```

Arguments

x	a gen_tibble object.
as_matrix	boolean, determining whether the results should be a square symmetrical matrix (TRUE), or a tidied tibble (FALSE, the default)
block_size	maximum number of loci read at once. More loci should improve speed, but will tax memory.

Details

Note that monomorphic sites are currently considered. What does PLINK do???

gt_load *Load a gen_tibble*

Description

Load a `gen_tibble` previously saved with `gt_save()`. If the *.rds* and *.bk* files have not been moved, they should be found automatically. If they were moved, use `reattach_to` to point to the *.rds* file (the *.bk* file needs to be in the same directory as the *.rds* file).

Usage

```
gt_load(file = NULL, reattach_to = NULL)
```

Arguments

`file`	the file name, including the full path. If it does not end with *.gt*, the extension will be added.
`reattach_to`	the file name, including the full path, of the *.rds* file if it was moved. It assumes that the *.bk* file is found in the same path. You should be able to leave this to NULL unless you have moved the files.

Value

a `gen_tibble`

See Also

`gt_save()`

gt_pca *Principal Component Analysis for* `gen_tibble` *objects*

Description

There are a number of PCA methods available for `gen_tibble` objects. They are mostly designed to work on very large datasets, so they only compute a limited number of components. For smaller datasets, `gt_partialSVD` allows the use of partial (truncated) SVD to fit the PCA; this method is suitable when the number of individuals is much smaller than the number of loci. For larger dataset, `gt_randomSVD` is more appropriate. Finally, there is a method specifically designed for dealing with LD in large datasets, `gt_autoSVD`. Whilst this is arguably the best option, it is somewhat data hungry, and so only suitable for very large datasets (hundreds of individuals with several hundred thousands markers, or larger).

Details

NOTE: using gt_pca_autoSVD with a small dataset will likely cause an error, see man page for details.

gt_pcadapt *pcadapt analysis on a* gen_tibble *object*

Description

pcadapt is an algorithm that detects genetic markers under selection. It is based on the principal component analysis (PCA) of the genotypes of the individuals. The method is described in Luu et al. (2017), See the R package pcadapt, which provides extensive documentation and examples.

Usage

```
gt_pcadapt(x, pca, k, n_cores = 1)
```

Arguments

x	A gen_tibble object.
pca	a gt_pca object, as returned by gt_pca_partialSVD() or gt_pca_randomSVD().
k	Number of principal components to use in the analysis.
n_cores	Number of cores to use.

Details

Internally, this function uses the snp_pcadapt function from the bigsnpr package.

Value

An object of subclass gt_pcadapt, a subclass of mhtest.

gt_pca_autoSVD *PCA controlling for LD for* gen_tibble *objects*

Description

This function performs Principal Component Analysis on a gen_tibble, using a fast truncated SVD with initial pruning and then iterative removal of long-range LD regions. This function is a wrapper for bigsnpr::snp_autoSVD()

Usage

```
gt_pca_autoSVD(
  x,
  k = 10,
  fun_scaling = bigsnpr::snp_scaleBinom(),
  thr_r2 = 0.2,
  use_positions = TRUE,
  size = 100/thr_r2,
  roll_size = 50,
  int_min_size = 20,
  alpha_tukey = 0.05,
  min_mac = 10,
  max_iter = 5,
  n_cores = 1,
  verbose = TRUE
)
```

Arguments

x	a gen_tbl object
k	Number of singular vectors/values to compute. Default is 10. **This algorithm should be used to compute a few singular vectors/values.**
fun_scaling	Usually this can be left unset, as it defaults to bigsnpr::snp_scaleBinom(), which is the appropriate function for biallelic SNPs. Alternatively it is possible to use custom function (see bigsnpr::snp_autoSVD() for details.
thr_r2	Threshold over the squared correlation between two SNPs. Default is 0.2. Use NA if you want to skip the clumping step. size
use_positions	a boolean on whether the position is used to define size, or whether the size should be in number of SNPs. Default is TRUE
size	For one SNP, window size around this SNP to compute correlations. Default is 100 / thr_r2 for clumping (0.2 -> 500; 0.1 -> 1000; 0.5 -> 200). If not providing infos.pos (NULL, the default), this is a window in number of SNPs, otherwise it is a window in kb (genetic distance). I recommend that you provide the positions if available.
roll_size	Radius of rolling windows to smooth log-p-values. Default is 50.
int_min_size	Minimum number of consecutive outlier SNPs in order to be reported as long-range LD region. Default is 20.
alpha_tukey	Default is 0.1. The type-I error rate in outlier detection (that is further corrected for multiple testing).
min_mac	Minimum minor allele count (MAC) for variants to be included. Default is 10.
max_iter	Maximum number of iterations of outlier detection. Default is 5.
n_cores	Number of cores used. Default doesn't use parallelism. You may use bigstatsr::nb_cores().
verbose	Output some information on the iterations? Default is TRUE.

Details

Using gt_pca_autoSVD requires a reasonably large dataset, as the function iteratively removes regions of long range LD.

Value

a `gt_pca` object, which is a subclass of `bigSVD`; this is an S3 list with elements: A named list (an S3 class "big_SVD") of

- `d`, the eigenvalues (singular values, i.e. as variances),
- `u`, the scores for each sample on each component (the left singular vectors)
- `v`, the loadings (the right singular vectors)
- `center`, the centering vector,
- `scale`, the scaling vector,
- `method`, a string defining the method (in this case 'autoSVD'),
- `call`, the call that generated the object.

Note: rather than accessing these elements directly, it is better to use `tidy` and `augment`. See `gt_pca_tidiers`.

`gt_pca_partialSVD` *PCA for* `gen_tibble` *objects by partial SVD*

Description

This function performs Principal Component Analysis on a `gen_tibble`, by partial SVD through the eigen decomposition of the covariance. It works well if the number of individuals is much smaller than the number of loci; otherwise, `gt_pca_randomSVD()` is a better option. This function is a wrapper for `bigstatsr::big_SVD()`.

Usage

```
gt_pca_partialSVD(x, k = 10, fun_scaling = bigsnpr::snp_scaleBinom())
```

Arguments

`x`	a `gen_tbl` object
`k`	Number of singular vectors/values to compute. Default is `10`. **This algorithm should be used to compute a few singular vectors/values.**
`fun_scaling`	Usually this can be left unset, as it defaults to `bigsnpr::snp_scaleBinom()`, which is the appropriate function for biallelic SNPs. Alternatively it is possible to use custom function (see `bigsnpr::snp_autoSVD()` for details.

Value

a gt_pca object, which is a subclass of bigSVD; this is an S3 list with elements: A named list (an S3 class "big_SVD") of

- d, the eigenvalues (singular values, i.e. as variances),
- u, the scores for each sample on each component (the left singular vectors)
- v, the loadings (the right singular vectors)
- center, the centering vector,
- scale, the scaling vector,
- method, a string defining the method (in this case 'partialSVD'),
- call, the call that generated the object.

Note: rather than accessing these elements directly, it is better to use tidy and augment. See gt_pca_tidiers.

---

gt_pca_randomSVD *PCA for* gen_tibble *objects by randomized partial SVD*

Description

This function performs Principal Component Analysis on a gen_tibble, by randomised partial SVD based on the algorithm in RSpectra (by Yixuan Qiu and Jiali Mei).
This algorithm is linear in time in all dimensions and is very memory-efficient. Thus, it can be used on very large big.matrices. This function is a wrapper for bigstatsr::big_randomSVD()

Usage

```
gt_pca_randomSVD(
  x,
  k = 10,
  fun_scaling = bigsnpr::snp_scaleBinom(),
  tol = 1e-04,
  verbose = FALSE,
  n_cores = 1,
  fun_prod = bigstatsr::big_prodVec,
  fun_cprod = bigstatsr::big_cprodVec
)
```

Arguments

x	a gen_tbl object
k	Number of singular vectors/values to compute. Default is 10. **This algorithm should be used to compute a few singular vectors/values.**
fun_scaling	Usually this can be left unset, as it defaults to bigsnpr::snp_scaleBinom(), which is the appropriate function for biallelic SNPs. Alternatively it is possible to use custom function (see bigsnpr::snp_autoSVD() for details.

tol	Precision parameter of svds. Default is 1e-4.
verbose	Should some progress be printed? Default is FALSE.
n_cores	Number of cores used.
fun_prod	Function that takes 6 arguments (in this order): • a matrix-like object X, • a vector x, • a vector of row indices ind.row of X, • a vector of column indices ind.col of X, • a vector of column centers (corresponding to ind.col), • a vector of column scales (corresponding to ind.col), and compute the product of X (subsetted and scaled) with x.
fun_cprod	Same as fun.prod, but for the *transpose* of X.

Value

a gt_pca object, which is a subclass of bigSVD; this is an S3 list with elements: A named list (an S3 class "big_SVD") of

- d, the eigenvalues (singular values, i.e. as variances),
- u, the scores for each sample on each component (the left singular vectors)
- v, the loadings (the right singular vectors)
- center, the centering vector,
- scale, the scaling vector,
- method, a string defining the method (in this case 'randomSVD'),
- call, the call that generated the object.

Note: rather than accessing these elements directly, it is better to use tidy and augment. See gt_pca_tidiers.

---

gt_roh_window *Detect runs of homozygosity using a sliding-window approach*

Description

This function uses a sliding-window approach to look for runs of homozygosity (or heterozygosity) in a diploid genome. This function uses the package selectRUNS, which implements an approach equivalent to the one in PLINK.

Usage

```
gt_roh_window(
  x,
  window_size = 15,
  threshold = 0.05,
  min_snp = 3,
  heterozygosity = FALSE,
  max_opp_window = 1,
  max_miss_window = 1,
  max_gap = 10^6,
  min_length_bps = 1000,
  min_density = 1/1000,
  max_opp_run = NULL,
  max_miss_run = NULL
)
```

Arguments

x	a gen_tibble
window_size	the size of sliding window (number of SNP loci) (default = 15)
threshold	the threshold of overlapping windows of the same state (homozygous/heterozygous) to call a SNP in a RUN (default = 0.05)
min_snp	minimum n. of SNP in a RUN (default = 3)
heterozygosity	should we look for runs of heterozygosity (instead of homozygosity? (default = FALSE)
max_opp_window	max n. of SNPs of the opposite type (e.g. heterozygous snps for runs of homozygosity) in the sliding window (default = 1)
max_miss_window	max. n. of missing SNP in the sliding window (default = 1)
max_gap	max distance between consecutive SNP to be still considered a potential run (default = 10^6 bps)
min_length_bps	minimum length of run in bps (defaults to 1000 bps = 1 kbps)
min_density	minimum n. of SNP per kbps (defaults to 0.1 = 1 SNP every 10 kbps)
max_opp_run	max n. of opposite genotype SNPs in the run (optional)
max_miss_run	max n. of missing SNPs in the run (optional)

Details

This function returns a data frame with all runs detected in the dataset. This data frame can then be written out to a csv file. The data frame is, in turn, the input for other functions of the detectRUNS package that create plots and produce statistics from the results (see plots and statistics functions in this manual, and/or refer to the detectRUNS vignette).

If the gen_tibble is grouped, then the grouping variable is used to fill in the group table. Otherwise, the group 'column' is filled with the same values as the 'id' column

Value

A dataframe with RUNs of Homozygosity or Heterozygosity in the analysed dataset. The returned dataframe contains the following seven columns: "group", "id", "chrom", "nSNP", "from", "to", "lengthBps" (group: population, breed, case/control etc.; id: individual identifier; chrom: chromosome on which the run is located; nSNP: number of SNPs in the run; from: starting position of the run, in bps; to: end position of the run, in bps; lengthBps: size of the run)

Examples

```
# don't run the example
if (FALSE) {
sheep_ped <- system.file("extdata", "Kijas2016_Sheep_subset.ped",
    package="detectRUNS")
sheep_gt <- tidypopgen::gen_tibble(sheep_ped, backingfile = tempfile(),
    quiet=TRUE)
sheep_gt <- sheep_gt %>% group_by(population)
sheep_roh <- gt_roh_window(sheep_gt)
detectRUNS::plot_Runs(runs = sheep_roh)
}
```

---

gt_save *Save a gen_tibble*

Description

Save the tibble (and update the backing files). The `gen_tibble` object is saved to a file with extension *.gt*, togethe with update its *.rds* and *.bk* files. Note that multiple *.gt* files can be linked to the same *.rds* and *.bk* files; generally, this occurs when we create multiple subsets of the data. The *.gt* file then stores the information on what subset of the full dataset we are interested in, whilst the *.rds* and *.bk* file store the full dataset. To reload a `gen_tibble`, you can pass the name of the *.gt* file with `gt_load()`.

Usage

```
gt_save(x, file_name = NULL, quiet = FALSE)
```

Arguments

`x`	a `gen_tibble`
`file_name`	the file name, including the full path. If it does not end with *.gt*, the extension will be added.
`quiet`	boolean to suppress information about hte files

Value

the file name and path of the *.gt* file, together with the *.rds* and *.bk* files

See Also

gt_load()

`gt_set_imputed` *Sets a* `gen_tibble` *to use imputed data*

Description

This function sets or unsets the use of imputed data. For some analysis, such as PCA, that does not allow for missing data, we have to use imputation, but for other analysis it might be preferable to allow for missing data.

Usage

```
gt_set_imputed(x, set = NULL)
```

Arguments

`x`	a `gen_tibble`
`set`	a boolean defining whether imputed data should be used

`gt_uses_imputed` *Checks if a* `gen_tibble` *uses imputed data*

Description

This function checks if a dataset uses imputed data. Note that it is possible to have a dataset that has been imputed but it is currently not using imputation.

Usage

```
gt_uses_imputed(x)
```

Arguments

`x`	a `gen_tibble`

Value

boolean TRUE or FALSE depending on whether the dataset is using the imputed values

indiv_het_obs *Estimate individual observed heterozygosity*

Description

Estimate observed heterozygosity (H_obs) for each individual (i.e. the frequency of loci that are heterozygous in an individual).

Usage

```
indiv_het_obs(.x, ...)

## S3 method for class 'tbl_df'
indiv_het_obs(.x, ...)

## S3 method for class 'vctrs_bigSNP'
indiv_het_obs(.x, ...)

## S3 method for class 'grouped_df'
indiv_het_obs(.x, ...)
```

Arguments

.x	a vector of class vctrs_bigSNP (usually the genotype column of a gen_tibble object), or a gen_tibble.
...	currently unused.

Value

a vector of heterozygosities, one per individuals in the gen_tibble

indiv_missingness *Estimate individual missingness*

Description

Estimate missingness for each individual (i.e. the frequency of missing genotypes in an individual).

Usage

```
indiv_missingness(.x, as_counts = FALSE, ...)

## S3 method for class 'tbl_df'
indiv_missingness(.x, as_counts = FALSE, ...)
```

```
## S3 method for class 'vctrs_bigSNP'
indiv_missingness(.x, as_counts = FALSE, ...)

## S3 method for class 'grouped_df'
indiv_missingness(.x, as_counts = FALSE, ...)
```

Arguments

.x	a vector of class vctrs_bigSNP (usually the genotype column of a gen_tibble object), or a gen_tibble.
as_counts	boolean defining whether the count of NAs (rather than the rate) should be returned. It defaults to FALSE (i.e. rates are returned by default).
...	currently unused.

Value

a vector of heterozygosities, one per individuals in the gen_tibble

---

indiv_ploidy *Return individual ploidy*

Description

Returns the ploidy for each individual.

Usage

```
indiv_ploidy(.x, ...)

## S3 method for class 'tbl_df'
indiv_ploidy(.x, ...)

## S3 method for class 'vctrs_bigSNP'
indiv_ploidy(.x, ...)

## S3 method for class 'grouped_df'
indiv_ploidy(.x, ...)
```

Arguments

.x	a gen_tibble, or a vector of class vctrs_bigSNP (usually the genotype column of a gen_tibble object)
...	currently unused.

Value

a vector of ploidy, one per individuals in the gen_tibble

loci_alt_freq *Estimate allele frequencies at each each locus*

Description

Allele frequencies can be estimates as minimum allele frequencies (MAF) with `loci_maf()` or the frequency of the alternate allele (with `loci_alt_freq()`). The latter are in line with the genotypes matrix (e.g. as extracted by `show_loci()`). Most users will be in interested in the MAF, but the raw frequencies might be useful when computing aggregated statistics.

Usage

```
loci_alt_freq(.x, ...)

## S3 method for class 'tbl_df'
loci_alt_freq(.x, ...)

## S3 method for class 'vctrs_bigSNP'
loci_alt_freq(.x, ...)

## S3 method for class 'grouped_df'
loci_alt_freq(.x, n_cores = bigstatsr::nb_cores(), ...)

loci_maf(.x, ...)

## S3 method for class 'tbl_df'
loci_maf(.x, ...)

## S3 method for class 'vctrs_bigSNP'
loci_maf(.x, ...)

## S3 method for class 'grouped_df'
loci_maf(.x, n_cores = bigstatsr::nb_cores(), ...)
```

Arguments

`.x`	a vector of class `vctrs_bigSNP` (usually the `genotypes` column of a `gen_tibble` object), or a `gen_tibble`.
`...`	other arguments passed to specific methods, currently unused.
`n_cores`	number of cores to be used, it defaults to `bigstatsr::nb_cores()`

Value

a vector of frequencies, one per locus

loci_chromosomes *Get the chromosomes of loci in a* gen_tibble

Description

Extract the loci chromosomes from a gen_tibble (or directly from its genotype column).

Usage

```
loci_chromosomes(.x, ...)

## S3 method for class 'tbl_df'
loci_chromosomes(.x, ...)

## S3 method for class 'vctrs_bigSNP'
loci_chromosomes(.x, ...)
```

Arguments

.x	a gen_tibble, or a vector of class vctrs_bigSNP (usually the genotype column of a gen_tibble object).
...	currently unused.

Value

a character vector of chromosomes

loci_hwe *Test Hardy-Weinberg equilibrium at each locus*

Description

Return the p-value from an exact test of HWE.

Usage

```
loci_hwe(.x, ...)

## S3 method for class 'tbl_df'
loci_hwe(.x, mid_p = TRUE, ...)

## S3 method for class 'vctrs_bigSNP'
loci_hwe(.x, mid_p = TRUE, ...)

## S3 method for class 'grouped_df'
loci_hwe(.x, ...)
```

Arguments

.x	a vector of class vctrs_bigSNP (usually the genotypes column of a gen_tibble object), or a gen_tibble.
...	not used.
mid_p	boolean on whether the mid-p value should be computed. Default is TRUE, as in PLINK.

Details

This function uses the original C++ algorithm from PLINK 1.90.

NOTE There are no tests for this function yet! Unit tests are needed.

Value

a vector of probabilities from HWE exact test, one per locus

Author(s)

the C++ algorithm was written by Christopher Chang for PLINK 1.90, based on original code by Jan Wigginton (the code was released under GPL3).

---

loci_ld_clump *Clump loci based on a Linkage Disequilibrium threshold*

Description

This function uses clumping to remove SNPs at high LD. When used with its default options, clumping based on MAF is similar to standard pruning (as done by PLINK with "–indep-pairwise (size+1) 1 thr.r2", but it results in a better spread of SNPs over the chromosome.

Usage

```
loci_ld_clump(.x, ...)

## S3 method for class 'tbl_df'
loci_ld_clump(.x, ...)

## S3 method for class 'vctrs_bigSNP'
loci_ld_clump(
  .x,
  S = NULL,
  thr_r2 = 0.2,
  size = 100/thr_r2,
  exclude = NULL,
  use_positions = TRUE,
  n_cores = 1,
```

```
  return_id = FALSE,
  ...
)

## S3 method for class 'grouped_df'
loci_ld_clump(.x, ...)
```

Arguments

.x	a gen_tibble object
...	currently not used.
S	A vector of loci statistics which express the importance of each SNP (the more important is the SNP, the greater should be the corresponding statistic). For example, if S follows the standard normal distribution, and "important" means significantly different from 0, you must use abs(S) instead. **If not specified, MAFs are computed and used.**
thr_r2	Threshold over the squared correlation between two SNPs. Default is 0.2.
size	For one SNP, window size around this SNP to compute correlations. Default is 100 / thr.r2 for clumping (0.2 -> 500; 0.1 -> 1000; 0.5 -> 200). If use_positions = FALSE, this is a window in number of SNPs, otherwise it is a window in kb (genetic distance). Ideally, use positions, as they provide a more sensible approach.
exclude	Vector of SNP indices to exclude anyway. For example, can be used to exclude long-range LD regions (see Price2008). Another use can be for thresholding with respect to p-values associated with S.
use_positions	boolean, if TRUE (the default), size is in kb, if FALSE size is the number of SNPs.
n_cores	number of cores to be used
return_id	boolean on whether the id of SNPs to keep should be returned. It defaults to FALSE, which returns a vector of booleans (TRUE or FALSE)

Details

Any missing values in the genotypes of a gen_tibble passed to loci_ld_clump will cause an error. To deal with missingness, see gt_impute_simple().

Value

a boolean vector indicating whether the SNP should be kept (if 'return_id = FALSE', the default), else a vector of SNP indices to be kept (if 'return_id = TRUE')

loci_missingness *Estimate missingness at each locus*

Description

Estimate the rate of missingness at each locus.

Usage

```
loci_missingness(.x, as_counts = FALSE, ...)

## S3 method for class 'tbl_df'
loci_missingness(.x, as_counts = FALSE, ...)

## S3 method for class 'vctrs_bigSNP'
loci_missingness(.x, as_counts = FALSE, ...)

## S3 method for class 'grouped_df'
loci_missingness(.x, as_counts = FALSE, n_cores = bigstatsr::nb_cores(), ...)
```

Arguments

.x	a vector of class vctrs_bigSNP (usually the genotypes column of a gen_tibble object), or a gen_tibble.
as_counts	boolean defining whether the count of NAs (rather than the rate) should be returned. It defaults to FALSE (i.e. rates are returned by default).
...	other arguments passed to specific methods.
n_cores	number of cores to be used, it defaults to bigstatsr::nb_cores()

Value

a vector of frequencies, one per locus

loci_names *Get the names of loci in a* gen_tibble

Description

Extract the loci names from a gen_tibble (or directly from its genotype column).

Usage

```
loci_names(.x, ...)

## S3 method for class 'tbl_df'
loci_names(.x, ...)

## S3 method for class 'vctrs_bigSNP'
loci_names(.x, ...)
```

Arguments

.x	a vector of class vctrs_bigSNP (usually the genotype column of a gen_tibble object), or a gen_tibble.
...	currently unused.

Value

a character vector of names

---

loci_transitions *Find transitions*

Description

Use the loci table to define which loci are transitions

Usage

```
loci_transitions(.x, ...)

## S3 method for class 'tbl_df'
loci_transitions(.x, ...)

## S3 method for class 'vctrs_bigSNP'
loci_transitions(.x, ...)

## S3 method for class 'grouped_df'
loci_transitions(.x, ...)
```

Arguments

.x	a vector of class vctrs_bigSNP (usually the genotype column of a gen_tibble object), or a gen_tibble.
...	other arguments passed to specific methods.

Value

a logical vector defining which loci are transitions

loci_transversions *Find transversions*

Description

Use the loci table to define which loci are transversions

Usage

```
loci_transversions(.x, ...)

## S3 method for class 'tbl_df'
loci_transversions(.x, ...)

## S3 method for class 'vctrs_bigSNP'
loci_transversions(.x, ...)

## S3 method for class 'grouped_df'
loci_transversions(.x, ...)
```

Arguments

.x	a vector of class vctrs_bigSNP (usually the genotype column of a gen_tibble object), or a gen_tibble.
...	other arguments passed to specific methods.

Value

a logical vector defining which loci are transversions

pairwise_allele_sharing *Compute the Pairwise Allele Sharing Matrix for a* gen_tibble *object*

Description

This function computes the Allele Sharing matrix. Estimates Allele Sharing (matching in hierfstat)) between pairs of individuals (for each locus, gives 1 if the two individuals are homozygous for the same allele, 0 if they are homozygous for a different allele, and 1/2 if at least one individual is heterozygous. Matching is the average of these 0, 1/2 and 1s)

Usage

```
pairwise_allele_sharing(
  x,
  as_matrix = FALSE,
  block_size = bigstatsr::block_size(count_loci(x))
)
```

Arguments

x	a gen_tibble object.
as_matrix	boolean, determining whether the results should be a square symmetrical matrix (TRUE), or a tidied tibble (FALSE, the default)
block_size	maximum number of loci read at once. More loci should improve speed, but will tax memory.

Value

a matrix of allele sharing between all pairs of individuals

---

pairwise_ibs *Compute the Identity by State Matrix for a* gen_tibble *object*

Description

This function computes the IBS matrix.

Usage

```
pairwise_ibs(
  x,
  as_matrix = FALSE,
  type = c("proportion", "adjusted_counts", "raw_counts"),
  block_size = bigstatsr::block_size(count_loci(x))
)
```

Arguments

x	a gen_tibble object.
as_matrix	boolean, determining whether the results should be a square symmetrical matrix (TRUE), or a tidied tibble (FALSE, the default)
type	one of "proportion" (equivalent to "ibs" in PLINK), "adjusted_counts" ("distance" in PLINK), and "raw_counts" (the counts of identical alleles and non-missing alleles, from which the two other quantities are computed)
block_size	maximum number of loci read at once. More loci should improve speed, but will tax memory.

Details

Note that monomorphic sites are currently counted. Should we filter them beforehand? What does plink do?

Value

a bigstatsr::FBM of proportion or adjusted counts, or a list of two bigstatsr::FBM matrices, one of counts of IBS by alleles, and one of number of valid alleles (i.e. 2*n_loci* - 2missing_loci)

---

pairwise_pop_fst *Compute pairwise population Fst*

Description

This function computes pairwise Fst. The following methods are implemented:

- 'Hudson': Hudson's formulation, as derived in Bhatia et al (2013) for diploids.
- 'Nei86' : Gst according to Nei (1986), as derived in Bhatia et al (2013) for diploids.
- 'Nei87' : Fst according to Nei (1987) - this is equivalent to `hierfstat::pairwise.neifst()`, and includes the correction for heterozygosity when computing Ht
- 'WC84' : Weir and Cockerham (1984), as derived in Bhatia et al (2013) for diploids.

Usage

```
pairwise_pop_fst(
  .x,
  by_locus = FALSE,
  method = c("Hudson", "Nei87", "Nei86", "WC84"),
  n_cores = bigstatsr::nb_cores()
)
```

Arguments

`.x`	a grouped `gen_tibble` (as obtained by using `dplyr::group_by()`)
`by_locus`	boolean, determining whether Fst should be returned by locus(TRUE), or as a single genome wide value obtained by taking the ratio of the mean numerator and denominator (FALSE, the default).
`method`	one of 'Hudson', 'Nei86', 'Nei87', and 'WC84'
`n_cores`	number of cores to be used, it defaults to `bigstatsr::nb_cores()`

Details

For all formulae, the genome wide estimate is obtained by taking the ratio of the mean numerators and denominators over all relevant SNPs.

Value

a tibble of genome-wide pairwise Fst values with each pairwise combination as a row if "by_locus=FALSE", else a list including the tibble of genome-wide values as well as a matrix with pairwise Fst by locus with loci as rows and and pairwise combinations as columns.

References

Bhatia G, Patterson N, Sankararaman S, Price AL. Estimating and Interpreting FST: The Impact of Rare Variants. Genome Research. 2013;23(9):1514–1521.

Nei, M. (1987) Molecular Evolutionary Genetics. Columbia University Press

---

`pop_fis` *Compute population specific FIS*

Description

This function computes population specific FIS (as computed by `hierfstat::fis.dosage()`).

Usage

```
pop_fis(.x, include_global = FALSE, allele_sharing_mat = NULL)
```

Arguments

`.x`	a grouped `gen_tibble` (as obtained by using `dplyr::group_by()`)
`include_global`	boolean determining whether, besides the population specific fis, a global fis should be appended. Note that this will return a vector of n populations plus 1 (the global value)
`allele_sharing_mat`	optional, the matrix of Allele Sharing returned by `pairwise_allele_sharing()` with `as_matrix=TRUE`. As a number of statistics can be derived from the Allele Sharing matrix, it it sometimes more efficient to pre-compute this matrix.

Value

a vector of population specific fis (plus the global value if `include_global=TRUE`)

pop_fst *Compute population specific Fst*

Description

This function computes population specific Fst (as computed by `hierfstat::fst.dosage()`).

Usage

```
pop_fst(.x, include_global = FALSE, allele_sharing_mat = NULL)
```

Arguments

`.x`	a grouped `gen_tibble` (as obtained by using `dplyr::group_by()`)
`include_global`	boolean determining whether, besides the population specific Fst, a global Fst should be appended. Note that this will return a vector of n populations plus 1 (the global value)
`allele_sharing_mat`	optional, the matrix of Allele Sharing returned by `pairwise_allele_sharing()` with `as_matrix=TRUE`. As a number of statistics can be derived from the Allele Sharing matrix,

Value

a vector of population specific Fst (plus the global value if `include_global=TRUE`)

predict.gt_pca *Predict scores of a PCA*

Description

Predict the PCA scores for a `gt_pca`, either for the original data or projecting new data.

Usage

```
## S3 method for class 'gt_pca'
predict(
  object,
  new_data = NULL,
  project_method = c("none", "simple", "OADP", "least_squares"),
  lsq_pcs = c(1, 2),
  block_size = NULL,
  n_cores = 1,
  ...
)
```

Arguments

object	the gt_pca object
new_data	a gen_tibble if scores are requested for a new dataset
project_method	a string taking the value of either "simple", "OADP" (Online Augmentation, Decomposition, and Procrustes (OADP) projection), or "least_squares" (as done by SMARTPCA)
lsq_pcs	a vector of length two with the values of the two principal components to use for the least square fitting. Only relevant if project_method = 'least_squares'
block_size	number of loci read simultaneously (larger values will speed up computation, but require more memory)
n_cores	number of cores
...	no used

Value

a matrix of predictions, with samples as rows and components as columns. The number of components depends on how many were estimated in the gt_pca object.

References

Zhang et al (2020). Fast and robust ancestry prediction using principal component analysis 36(11): 3439–3446.

---

qc_report_indiv *Create a Quality Control report for individuals*

Description

#' Return QC information to assess loci (Observed heterozygosity and missingness).

Usage

```
qc_report_indiv(.x, kings_threshold = NULL, ...)
```

Arguments

.x	a gen_tibble object.
kings_threshold	an optional numeric, a threshold of relatedness for the sample
...	further arguments to pass

Value

a tibble with 2 elements: het_obs and missingness

qc_report_loci *Create a Quality Control report for loci*

Description

Return QC information to assess loci (MAF, missingness and HWE test).

Usage

```
qc_report_loci(.x, ...)
```

Arguments

.x	a gen_tibble object.
...	currently unused the HWE test.

Value

a tibble with 3 elements: maf, missingness and hwe_p

rbind.gen_tbl *Combine two gen_tibbles*

Description

This function combined two gen_tibbles. By defaults, it subsets the loci and swaps ref and alt alleles to make the two datasets compatible (this behaviour can be switched off with as_is). The first object is used as a "reference" , and SNPs in the other dataset will be flipped and/or alleles swapped as needed. SNPs that have different alleles in the two datasets (i.e. triallelic) will also be dropped. There are also options (NOT default) to attempt strand flipping to match alleles (often needed in human datasets from different SNP chips), and remove ambiguous alleles (C/G and A/T) where the correct strand can not be guessed.

Usage

```
## S3 method for class 'gen_tbl'
rbind(
  ...,
  as_is = FALSE,
  flip_strand = FALSE,
  use_position = FALSE,
  quiet = FALSE,
  backingfile = NULL
)
```

Arguments

...	two gen_tibble objects. Note that this function can not take more objects, rbind has to be done sequentially for large sets of objects.
as_is	boolean determining whether the loci should be left as they are before merging. If FALSE (the defaults), rbind will attempt to subset and swap alleles as needed.
flip_strand	boolean on whether strand flipping should be checked to match the two datasets. If this is set to TRUE, ambiguous SNPs (i.e. A/T and C/G) will also be removed. It defaults to FALSE
use_position	boolean of whether a combination of chromosome and position should be used for matching SNPs. By default, rbind uses the locus name, so this is set to FALSE. When using 'use_position=TRUE', make sure chromosomes are coded in the same way in both gen_tibbles (a mix of e.g. 'chr1', '1' or 'chromosome1' can be the reasons if an unexpectedly large number variants are dropped when merging).
quiet	boolean whether to omit reporting to screen
backingfile	the path and prefix of the files used to store the merged data (it will be a .RDS to store the bigSNP object and a .bk file as its backing file for the FBM)

Details

rbind differs from merging data with plink, which swaps the order of allele1 and allele2 according to minor allele frequency when merging datasets. rbind flips and/or swaps alleles according to the reference dataset, not according to allele frequency.

Value

a gen_tibble with the merged data.

---

rbind_dry_run *Generate a report of what would happen to each SNP in a merge*

Description

This function provides an overview of the fate of each SNP in two gen_tibble objects in the case of a merge. Only SNPs found in both objects will be kept. One object is used as a reference, and SNPs in the other dataset will be flipped and/or alleles swapped as needed. SNPs that have different alleles in the two datasets will also be dropped.

Usage

```
rbind_dry_run(
  ref,
  target,
  use_position = FALSE,
  flip_strand = FALSE,
  quiet = FALSE
)
```

Arguments

ref	either a gen_tibble object, or the path to the PLINK bim file; the alleles in this objects will be used as template to flip the ones in target and/or swap their order as necessary.
target	either a gen_tibble object, or the path to the PLINK bim file
use_position	boolean of whether a combination of chromosome and position should be used for matching SNPs. By default, rbind uses the locus name, so this is set to FALSE. When using 'use_position=TRUE', make sure chromosomes are coded in the same way in both gen_tibbles (a mix of e.g. 'chr1', '1' or 'chromosome1' can be the reasons if an unexpectedly large number variants are dropped when merging).
flip_strand	boolean on whether strand flipping should be checked to match the two datasets. Ambiguous SNPs (i.e. A/T and C/G) will also be removed. It defaults to FALSE
quiet	boolean whether to omit reporting to screen

Value

a list with two data.frames, named target and ref. Each data.frame has nrow() equal to the number of loci in the respective dataset, a column id with the locus name, and boolean columns to_keep (the valid loci that will be kept in the merge), alleles_mismatched (loci found in both datasets but with mismatched alleles, leading to those loci being dropped), to_flip (loci that need to be flipped to align the two datasets, only found in target data.frame) and to_swap (loci for which the order of alleles needs to be swapped to align the two datasets, target data.frame)

---

read_q_matrix_list *Tidy ADMXITURE output files into plots*

Description

Takes the name of a directory containing .Q file outputs, and produces a list of tidied tibbles ready to plot.

Usage

```
read_q_matrix_list(x, data)
```

Arguments

x	the name of a directory containing .Q files
data	An associated tibble (e.g. a gen_tibble), with the individuals in the same order as the data used to generate the Q matrix

Value

a list of q_matrix objects to plot

scale_fill_distruct *Scale constructor using the distruct colours*

Description

A wrapper around ggplot2::scale_fill_manual(), using the distruct colours from distruct_colours.

Usage

```
scale_fill_distruct(guide = "none", ...)
```

Arguments

guide	guide function passed to ggplot2::scale_fill_manual(). Defaults to "none", set to "legend" if a legend is required.
...	further parameters to be passed to ggplot2::scale_fill_manual()

Value

a scale constructor to be used with ggplot

select_loci *The* select *verb for* loci

Description

An equivalent to dplyr::select() that works on the genotype column of a gen_tibble, using the mini-grammar available for tidyselect. The select-like evaluation only has access to the names of the loci (i.e. it can select only based on names, not summary statistics of those loci; look at select_loci_if() for that feature.

Usage

```
select_loci(.data, .sel_arg)
```

Arguments

.data	a gen_tibble
.sel_arg	one unquoted expression, using the mini-grammar of dplyr::select() to select loci. Variable names can be used as if they were positions in the data frame, so expressions like x:y can be used to select a range of variables.

Details

Note that the select_loci verb does not modify the backing FBM files, but rather it subsets the list of loci to be used stored in the gen_tibble.

Value

a gen_tibble with a subset of the loci.

---

select_loci_if *The* select_if *verb for* loci

Description

An equivalent to dplyr::select_if() that works on the genotype column of a gen_tibble. This function has access to the genotypes (and thus can work on summary statistics to select), but not the names of the loci (look at select_loci() for that feature.

Usage

```
select_loci_if(.data, .sel_logical)
```

Arguments

.data	a gen_tibble
.sel_logical	a logical vector of length equal to the number of loci, or an expression that will tidy evaluate to such a vector. Only loci for which .sel_logical is TRUE will be selected; NA will be treated as FALSE.

Details

#' Note that the select_loci_if verb does not modify the backing FBM files, but rather it subsets the list of loci to be used stored in the gen_tibble.

---

show_genotypes *Show the genotypes of a* gen_tibble

Description

Extract the genotypes (as a matrix) from a gen_tibble.

Usage

```
show_genotypes(.x, indiv_indices = NULL, loci_indices = NULL, ...)

## S3 method for class 'tbl_df'
show_genotypes(.x, indiv_indices = NULL, loci_indices = NULL, ...)

## S3 method for class 'vctrs_bigSNP'
show_genotypes(.x, indiv_indices = NULL, loci_indices = NULL, ...)
```

Arguments

.x	a vector of class vctrs_bigSNP (usually the genotype column of a gen_tibble object), or a gen_tibble.
indiv_indices	indices of individuals
loci_indices	indices of loci
...	currently unused.

Value

a matrix of counts of the alternative alleles (see show_loci()) to extract information on the alleles for those loci from a gen_tibble.

---

show_loci *Show the loci information of a* gen_tibble

Description

Extract and set the information on loci from a gen_tibble.

Usage

```
show_loci(.x, ...)

## S3 method for class 'tbl_df'
show_loci(.x, ...)

## S3 method for class 'vctrs_bigSNP'
show_loci(.x, ...)

show_loci(.x) <- value

## S3 replacement method for class 'tbl_df'
show_loci(.x) <- value

## S3 replacement method for class 'vctrs_bigSNP'
show_loci(.x) <- value
```

Arguments

.x	a vector of class vctrs_bigSNP (usually the genotype column of a gen_tibble object), or a gen_tibble.
...	currently unused.
value	a data.frame or tibble of loci information to replace the current one.

Value

a tibble::tibble of information (see gen_tibble for details on compulsory columns that will always be present)

---

show_ploidy *Show the ploidy information of a* gen_tibble

Description

Extract the ploidy information from a gen_tibble. NOTE that this function does not return the ploidy level for each individual (that is obtained with indiv_ploidy); instead, it returns an integer which is either the ploidy level of all individuals (e.g. 2 indicates all individuals are diploid), or a 0 to indicate mixed ploidy.

Usage

```
show_ploidy(.x, ...)

## S3 method for class 'tbl_df'
show_ploidy(.x, ...)

## S3 method for class 'vctrs_bigSNP'
show_ploidy(.x, ...)
```

Arguments

.x	a vector of class vctrs_bigSNP (usually the genotype column of a gen_tibble object), or a gen_tibble.
...	currently unused.

Value

the ploidy (0 indicates mixed ploidy)

---

snp_allele_sharing *Compute the Pairwise Allele Sharing Matrix for a bigSNP object*

Description

This function computes the Allele Sharing matrix. Estimates Allele Sharing (matching in hierfstat)) between pairs of individuals (for each locus, gives 1 if the two individuals are homozygous for the same allele, 0 if they are homozygous for a different allele, and 1/2 if at least one individual is heterozygous. Matching is the average of these 0, 1/2 and 1s)

Usage

```
snp_allele_sharing(
  X,
  ind.row = bigstatsr::rows_along(X),
  ind.col = bigstatsr::cols_along(X),
  block.size = bigstatsr::block_size(nrow(X))
)
```

Arguments

X	a bigstatsr::FBM.code256 matrix (as found in the genotypes slot of a bigsnpr::bigSNP object).
ind.row	An optional vector of the row indices that are used. If not specified, all rows are used. Don't use negative indices.
ind.col	An optional vector of the column indices that are used. If not specified, all columns are used. Don't use negative indices.
block.size	maximum number of columns read at once. Note that, to optimise the speed of matrix operations, we have to store in memory 3 times the columns.

Value

a matrix of allele sharing between all pairs of individuals

---

snp_ibs *Compute the Identity by State Matrix for a bigSNP object*

Description

This function computes the IBS matrix.

Usage

```
snp_ibs(
  X,
  ind.row = bigstatsr::rows_along(X),
  ind.col = bigstatsr::cols_along(X),
  type = c("proportion", "adjusted_counts", "raw_counts"),
  block.size = bigstatsr::block_size(nrow(X))
)
```

Arguments

X	a bigstatsr::FBM.code256 matrix (as found in the genotypes slot of a bigsnpr::bigSNP object).
ind.row	An optional vector of the row indices that are used. If not specified, all rows are used. Don't use negative indices.

ind.col	An optional vector of the column indices that are used. If not specified, all columns are used. Don't use negative indices.
type	one of "proportion" (equivalent to "ibs" in PLINK), "adjusted_counts" ("distance" in PLINK), and "raw_counts" (the counts of identical alleles and non-missing alleles, from which the two other quantities are computed)
block.size	maximum number of columns read at once. Note that, to optimise the speed of matrix operations, we have to store in memory 3 times the columns.

Details

Note that monomorphic sites are currently counted. Should we filter them beforehand? What does plink do?

Value

if as.counts = TRUE function returns a list of two bigstatsr::FBM matrices, one of counts of IBS by alleles (i.e. 2*n loci), and one of valid alleles (i.e. 2 * n_loci - 2 * missing_loci). If as.counts = FALSE returns a single matrix of IBS proportions.

---

snp_king *Compute the KING-robust Matrix for a bigSNP object*

Description

This function computes the KING-robust estimator of kinship.

Usage

```
snp_king(
  X,
  ind.row = bigstatsr::rows_along(X),
  ind.col = bigstatsr::cols_along(X),
  block.size = bigstatsr::block_size(nrow(X)) * 4
)
```

Arguments

X	a bigstatsr::FBM.code256 matrix (as found in the genotypes slot of a bigsnpr::bigSNP object).
ind.row	An optional vector of the row indices that are used. If not specified, all rows are used. Don't use negative indices.
ind.col	An optional vector of the column indices that are used. If not specified, all columns are used. Don't use negative indices.
block.size	maximum number of columns read at once.

Details

The last step is not optimised yet, as it does the division of the num by the den all in memory (on my TODO list...).

---

summary.rbind_report *Print a summary of a merge report*

Description

This function creates a summary of the merge report generated by `rbind_dry_run()`

Usage

```
## S3 method for class 'rbind_report'
summary(object, ..., ref_label = "reference", target_label = "target")
```

Arguments

`object`	a list generated by `rbind_dry_run()`
`...`	unused (necessary for compatibility with generic function)
`ref_label`	the label for the reference dataset (defaults to "reference")
`target_label`	the label for the target dataset (defaults to "target")

Value

NULL (prints a summary to the console)

---

theme_distruct *A theme to match the output of distruct*

Description

A theme to remove most plot decorations, matching the look of plots created with distruct.

Usage

```
theme_distruct()
```

Value

a ggplot2::theme

`tidy.gt_dapc` *Tidy a* `gt_dapc` *object*

Description

This summarizes information about the components of a `gt_dapc` from the `tidypopgen` package. The parameter `matrix` determines which element is returned.

Usage

```
## S3 method for class 'gt_dapc'
tidy(x, matrix = "eigenvalues", ...)
```

Arguments

`x`	A `gt_dapc` object (as returned by `gt_dapc()`).
`matrix`	Character specifying which component of the DAPC should be tidied. • `"samples"`, `"scores"`, or `"x"`: returns information about the map from the original space into the least discriminant axes. • `"v"`, `"rotation"`, `"loadings"` or `"variables"`: returns information about the map from discriminant axes space back into the original space (i.e. the genotype frequencies). Note that this are different from the loadings linking to the PCA scores (which are available in the element $loadings of the dapc object). • `"d"`, `"eigenvalues"` or `"lds"`: returns information about the eigenvalues.
`...`	Not used. Needed to match generic signature only.

Value

A tibble::tibble with columns depending on the component of DAPC being tidied.

If `"scores"` each row in the tidied output corresponds to the original data in PCA space. The columns are:

`row`	ID of the original observation (i.e. rowname from original data).
`LD`	Integer indicating a principal component.
`value`	The score of the observation for that particular principal component. That is, the location of the observation in PCA space.

If `matrix` is `"loadings"`, each row in the tidied output corresponds to information about the principle components in the original space. The columns are:

`row`	The variable labels (colnames) of the data set on which PCA was performed.
`LD`	An integer vector indicating the principal component.
`value`	The value of the eigenvector (axis score) on the indicated principal component.

If "eigenvalues", the columns are:

LD	An integer vector indicating the discriminant axis.
std.dev	Standard deviation (i.e. sqrt(eig/(n-1))) explained by this DA (for compatibility with prcomp.
cumulative	Cumulative variation explained by principal components up to this component (note that this is NOT phrased as a percentage of total variance, since many methods only estimate a truncated SVD.

See Also

gt_dapc() augment.gt_dapc()

---

tidy.gt_pca *Tidy a* gt_pca *object*

Description

This summarizes information about the components of a gt_pca from the tidypopgen package. The parameter matrix determines which element is returned. Column names of the tidied output match those returned by broom::tidy.prcomp, the tidier for the standard PCA objects returned by stats::prcomp.

Usage

```
## S3 method for class 'gt_pca'
tidy(x, matrix = "eigenvalues", ...)
```

Arguments

x	A gt_pca object returned by one of the gt_pca_* functions.
matrix	Character specifying which component of the PCA should be tidied. • "samples", "scores", or "x": returns information about the map from the original space into principle components space (this is equivalent to product of *u* and *d*). • "v", "rotation", "loadings" or "variables": returns information about the map from principle components space back into the original space. • "d", "eigenvalues" or "pcs": returns information about the eigenvalues.
...	Not used. Needed to match generic signature only.

Value

A tibble::tibble with columns depending on the component of PCA being tidied.

If "scores" each row in the tidied output corresponds to the original data in PCA space. The columns are:

row	ID of the original observation (i.e. rowname from original data).
PC	Integer indicating a principal component.
value	The score of the observation for that particular principal component. That is, the location of the observation in PCA space.

If matrix is "loadings", each row in the tidied output corresponds to information about the principle components in the original space. The columns are:

row	The variable labels (colnames) of the data set on which PCA was performed.
PC	An integer vector indicating the principal component.
value	The value of the eigenvector (axis score) on the indicated principal component.

If "eigenvalues", the columns are:

PC	An integer vector indicating the principal component.
std.dev	Standard deviation (i.e. sqrt(eig/(n-1))) explained by this PC (for compatibility with prcomp.
cumulative	Cumulative variation explained by principal components up to this component (note that this is NOT phrased as a percentage of total variance, since many methods only estimate a truncated SVD.

See Also

gt_pca_autoSVD() augment_gt_pca

---

tidy.q_matrix *Tidy a Q matrix*

Description

Takes a q_matrix object, which is a matrix, and returns a tidied tibble.

Usage

```
## S3 method for class 'q_matrix'
tidy(x, data, ...)
```

Arguments

x	A Q matrix object (as returned by LEA::Q()).
data	An associated tibble (e.g. a gen_tibble), with the individuals in the same order as the data used to generate the Q matrix
...	not currently used

Value

A tidied tibble

Index