Package 'SNPtools'

July 13, 2025

Title S4 Tools for Reading and Organizing Genetic Data

Version 0.1.0

Maintainer Vinícius Junqueira <junqueiravinicius@hotmail.com>

Description SNPtools provides an integrated suite of tools for handling SNP genotype data in large-scale genetic studies. The package supports importing and merging genotype files, performing quality control on SNP markers and samples, and preparing data for downstream analyses using popular software such as FImpute and PLINK. It offers S4 classes and methods to efficiently encapsulate SNP data, along with utilities for generating genotype summary statistics and visualization. Additional functionalities in the current version include anticlustering approaches for batch effect control, automated script generation for external software, and streamlined workflows for large datasets commonly encountered in animal and plant breeding programs. The package is designed to facilitate reproducible and scalable SNP data analyses in quantitative and statistical genetics.

Logo man/figures/logo.png

Depends R (>= 4.1.0),
snpStats,
tidyverse,
dplyr

Imports methods,
reshape2,
ggplot2,
dplyr,
tidyverse,
data.table,
Rcpp,
stringi,
anticlust

LinkingTo Rcpp

Suggests knitr,
rmarkdown

VignetteBuilder knitr

Encoding UTF-8

License GPL-3 + file LICENSE

URL https://viniciusjunqueira.github.io/SNPtools

BugReports https://github.com/viniciusjunqueira/SNPtools/issues

RoxygenNote 7.3.2

Contents

cbind_SnpMatrix *Safe cbind for SnpMatrix preserving dimnames*

Description

This function performs a column-wise binding of multiple `SnpMatrix` objects, explicitly preserving row names and column names, avoiding unexpected "object has no names" warnings.

Usage

```
cbind_SnpMatrix(...)
```

Arguments

`...`	SnpMatrix objects to combine (must have identical row names).

Value

A single combined `SnpMatrix` with preserved row and column names.

Examples

```
## Not run:
cbind_SnpMatrix(matrix1, matrix2)

## End(Not run)
```

check.call.rate *Check SNP call rate*

Description

Identifies SNPs with call rates below a minimum threshold.

Usage

```
check.call.rate(summary, min.call.rate)
```

Arguments

`summary`	A data frame with SNP summary statistics (must contain ‘Call.rate‘ column).
`min.call.rate`	Numeric value specifying the minimum acceptable call rate.

Value

Character vector with SNP names below threshold. Returns ‘NULL‘ if none.

Author(s)

Roberto Higa

Examples

```
df <- data.frame(Call.rate = c(0.85, 0.95), row.names = c("SNP1", "SNP2"))
check.call.rate(df, 0.9)
```

---

check.ibs *Check Identity-By-State (IBS) for a genotype pair*

Description

Checks IBS status for two genotypes.

Usage

```
check.ibs(gen)
```

Arguments

gen	Numeric vector of length two with genotype codes.

Value

Integer: 2 if identical non-heterozygotes, 0 if opposite homozygotes, -1 otherwise.

Author(s)

Roberto Higa

Examples

```
check.ibs(c(1, 1))
check.ibs(c(1, 3))
```

---

check.identical.samples *Check identical samples based on distance*

Description

Identifies sample pairs considered identical based on genotype distances.

Usage

```
check.identical.samples(genotypes, threshold = 0)
```

Arguments

genotypes	Genotype matrix (samples x SNPs).
threshold	Numeric distance threshold. Default 0.

Value

List of identical sample pairs.

Author(s)

Roberto Higa

Examples

```
mat <- matrix(sample(0:2, 20, TRUE), nrow = 5)
rownames(mat) <- paste0("S", 1:5)
check.identical.samples(mat, 0.5)
```

`check.identical.samples.by.block`
Check identical samples by block

Description

Identifies identical samples within SNP blocks.

Usage

```
check.identical.samples.by.block(genotypes, blcsize, threshold = 0)
```

Arguments

`genotypes`	Genotype matrix.
`blcsize`	Block size (number of SNPs).
`threshold`	Distance threshold. Default 0.

Value

List of identical sample pairs.

Author(s)

Roberto Higa

Examples

```
# See check.identical.samples example
```

check.mendelian.inconsistencies

Check Mendelian inconsistencies

Description

Identifies Mendelian inconsistencies between father-child pairs.

Usage

```
check.mendelian.inconsistencies(genotypes, father, child)
```

Arguments

genotypes	Genotype matrix.
father	Vector of father sample IDs.
child	Vector of child sample IDs.

Value

Data frame summarizing inconsistencies per pair.

Author(s)

Roberto Higa

Examples

```
# Requires proper parent-child genotype data
```

check.mendelian.inconsistencies.pair

Check Mendelian inconsistencies for a pair

Description

Calculates number of inconsistencies and total comparable SNPs for a parent-child pair.

Usage

```
check.mendelian.inconsistencies.pair(g1, g2)
```

Arguments

g1	Genotype vector for parent.
g2	Genotype vector for child.

Value

Numeric vector: [# inconsistencies, # comparable SNPs].

Author(s)

Roberto Higa

Examples

```
# Used internally by check.mendelian.inconsistencies
```

---

check.sample.call.rate

Check Sample Call Rate

Description

Identifies samples with call rate below a given threshold.

Usage

```
check.sample.call.rate(sample.summary, min.call.rate)
```

Arguments

sample.summary	A data frame with a "Call.rate" column for each sample.
min.call.rate	Minimum acceptable call rate (between 0 and 1).

Value

A character vector with the names of samples to remove.

---

check.sample.heterozygosity

Check sample heterozygosity

Description

Identifies samples with heterozygosity values deviating beyond a specified threshold.

Usage

```
check.sample.heterozygosity(sample.summary, max.dev)
```

Arguments

sample.summary	Data frame containing sample summary (must have ‘Heterozygosity‘ column).
max.dev	Maximum number of standard deviations allowed from mean.

Value

Character vector with sample names considered outliers. Returns ‘NULL‘ if none.

Author(s)

Roberto Higa

Examples

```
ss <- data.frame(Heterozygosity = c(0.2, 0.5, 0.7))
rownames(ss) <- c("Ind1", "Ind2", "Ind3")
check.sample.heterozygosity(ss, 1)
```

---

check.snp.chromo *Check SNP by chromosome*

Description

Filters SNP names belonging to specified chromosomes.

Usage

```
check.snp.chromo(snpmap, chromosomes)
```

Arguments

snpmap	Data frame with SNP map info (must contain columns ‘Chromosome‘ and ‘Name‘).
chromosomes	Vector of chromosome identifiers to filter.

Value

Character vector with SNP names.

Author(s)

Roberto Higa

Examples

```
snpmap <- data.frame(Chromosome = c(1, 1, 2), Name = c("SNP1", "SNP2", "SNP3"))
check.snp.chromo(snpmap, 1)
```

check.snp.hwe *Check SNP Hardy-Weinberg equilibrium deviation*

Description

Identifies SNPs deviating from HWE beyond a z-score threshold.

Usage

```
check.snp.hwe(snp.summary, max.dev)
```

Arguments

snp.summary	Data frame with SNP summary (must contain 'z.HWE' column).
max.dev	Maximum z-score allowed.

Value

Character vector with SNP names deviating from HWE. Returns 'NULL' if none.

Author(s)

Roberto Higa

Examples

```
df <- data.frame(z.HWE = c(2, 5), row.names = c("SNP1", "SNP2"))
check.snp.hwe(df, 3)
```

check.snp.hwe.chi2 *Check SNPs for Hardy-Weinberg equilibrium deviation using chi-square p-values*

Description

This function identifies SNP markers whose Hardy-Weinberg equilibrium (HWE) chi-square p-values indicate significant deviation beyond a specified threshold. It uses the p-values computed by `get.hwe.chi2` on the input summary data frame.

Usage

```
check.snp.hwe.chi2(snp.summary, max.dev)
```

Arguments

snp.summary	A data frame or matrix containing summary statistics for SNP markers. The row names should correspond to SNP identifiers. It must be compatible with the function `get.hwe.chi2`.
max.dev	A numeric value specifying the maximum acceptable p-value threshold. SNPs with p-values below this threshold are considered as deviating from HWE.

Details

Any SNP with missing p-value (NA) is treated as not failing (returned as FALSE).

Value

A character vector of SNP identifiers (rownames) that fail the HWE test (p-value < `max.dev`). If no SNPs fail, an empty vector is returned.

See Also

`get.hwe.chi2`

Examples

```
# Example usage (assuming snp.summary is precomputed and get.hwe.chi2 is defined)
# snps_failed <- check.snp.hwe.chi2(snp.summary, max.dev = 0.05)
```

`check.snp.maf` *Check SNP minor allele frequency*

Description

Identifies SNPs with minor allele frequency below a minimum threshold.

Usage

```
check.snp.maf(snp.summary, min.maf)
```

Arguments

`snp.summary`	Data frame with SNP summary (must contain ‘MAF‘ column).
`min.maf`	Minimum MAF allowed.

Value

Character vector with SNP names below threshold. Returns ‘NULL‘ if none.

Author(s)

Roberto Higa

Examples

```
df <- data.frame(MAF = c(0.01, 0.2), row.names = c("SNP1", "SNP2"))
check.snp.maf(df, 0.05)
```

check.snp.mgf *Check SNP missing genotype frequencies*

Description

Identifies SNPs with genotype frequencies below a minimum threshold.

Usage

```
check.snp.mgf(snp.summary, min.mgf)
```

Arguments

snp.summary	Data frame with columns 'P.AA', 'P.AB', 'P.BB'.
min.mgf	Minimum genotype frequency allowed.

Value

Character vector with SNP names below threshold. Returns 'NULL' if none.

Author(s)

Roberto Higa

Examples

```
df <- data.frame(P.AA = c(0.01, 0.5), P.AB = c(0.02, 0.4), P.BB = c(0.01, 0.1))
rownames(df) <- c("SNP1", "SNP2")
check.snp.mgf(df, 0.05)
```

check.snp.monomorf *Check SNP monomorphic status*

Description

Identifies SNPs considered monomorphic.

Usage

```
check.snp.monomorf(snp.summary)
```

Arguments

snp.summary	Data frame with columns 'P.AA', 'P.AB', 'P.BB'.

Value

Character vector with monomorphic SNP names. Returns 'NULL' if none.

Author(s)

Roberto Higa

Examples

```
df <- data.frame(P.AA = c(1, 0.5), P.AB = c(0, 0.5), P.BB = c(0, 0))
rownames(df) <- c("SNP1", "SNP2")
check.snp.monomorf(df)
```

---

check.snp.no.position *Check SNP no position*

Description

Identifies SNPs with position equal to zero in the SNP map.

Usage

```
check.snp.no.position(snpmap)
```

Arguments

snpmap	Data frame with columns 'Position' and 'Name'.

Value

Character vector with SNP names without position. Returns 'NULL' if none.

Author(s)

Roberto Higa

Examples

```
df <- data.frame(Position = c(0, 100), Name = c("SNP1", "SNP2"))
check.snp.no.position(df)
```

check.snp.same.position *Check SNPs mapped to the same position*

Description

Identifies groups of SNPs that are mapped to the exact same genomic position on each chromosome. Returns a list where each element corresponds to one group of overlapping SNPs.

Identifies SNPs that share the same position on the same chromosome.

Usage

```
check.snp.same.position(snpmap)

check.snp.same.position(snpmap)
```

Arguments

snpmap	Data frame with columns 'Chromosome', 'Position', and 'Name'.

Value

A list of character vectors, each with names of SNPs found at the same position.

List of SNP groups sharing positions.

Author(s)

Roberto Higa

Examples

```
df <- data.frame(Chromosome = c(1, 1, 2), Position = c(100, 100, 200), Name = c("SNP1", "SNP2", "SNP3"))
check.snp.same.position(df)
```

combinarSNPData *Combine multiple SNPDataLong objects*

Description

This function merges a list of SNPDataLong objects, typically representing different SNP panels or datasets, into a single unified SNPDataLong object. It ensures that all genotype matrices have the same set of SNPs (filling missing SNPs with NA), and merges the marker map information while removing duplicate SNP entries.

Usage

```
combinarSNPData(lista)
```

Arguments

lista	A list of SNPDataLong objects to be combined.

Value

A single SNPDataLong object containing the combined genotype matrix, merged map, and a concatenated path string.

Examples

```
## Not run:
combined <- combinarSNPData(list(snp_obj1, snp_obj2, snp_obj3))

## End(Not run)
```

---

doPCA *Do genome relationship matrix PCA*

Description

Performs PCA using the genome relationship matrix (GRM).

Usage

```
doPCA(genotypes)
```

Arguments

genotypes	Genotype matrix.

Value

List containing ‘pcs‘ (principal components) and ‘eigen‘ (eigenvalues).

Author(s)

Roberto Higa

Examples

```
# Requires matrix of numeric genotypes
```

`exploratory.plots` *Exploratory plots for SNP and sample summary*

Description

Generates exploratory plots: MAF histograms, HWE plots, heterozygosity scatter, MDS, and dendrogram.

Usage

```
exploratory.plots(
  snp.summary,
  snps.plot,
  sample.summary,
  samples.plot,
  distm,
  glabels,
  mds.plot,
  hierq.plot
)
```

Arguments

`snp.summary`	Data frame with SNP summary.
`snps.plot`	Filename for SNP histogram plot.
`sample.summary`	Data frame with sample summary.
`samples.plot`	Filename for heterozygosity plot.
`distm`	Distance matrix for samples.
`glabels`	Sample labels for plots.
`mds.plot`	Filename for MDS plot.
`hierq.plot`	Filename for hierarchical cluster plot.

Value

None. Plots are saved as JPEG files.

Author(s)

Roberto Higa

Examples

```
# Requires proper SNP and sample summary data frames
```

FImputeRunner *Build FImputeRunner object*

Description

A convenience function to construct a 'FImputeRunner' object from basic inputs.

Usage

```
FImputeRunner(object, path, exec_path = "FImpute3", name = "data")
```

Arguments

path	A character string indicating the directory to save FImpute files.
exec_path	Path to the FImpute executable (default = "FImpute3").
name	Name for the dataset (used internally, default = "gen_data").
geno	A SnpMatrix object.
map	A data.frame with SNP metadata (columns: Name, Chromosome, Position).

Value

An object of class 'FImputeRunner'.

genoToDF *Convert geno slot from SNPDataLong to a data.frame*

Description

Converts the genotype matrix (geno slot) of a SNPDataLong object to a data.frame, with optional centering and scaling per SNP (column).

Usage

```
genoToDF(object, center = FALSE, scale = FALSE)
```

Arguments

object	An object of class SNPDataLong.
center	Logical or numeric. If TRUE (default FALSE), center columns to mean zero.
scale	Logical or numeric. If TRUE (default FALSE), scale columns to standard deviation one.

Value

A data.frame with individuals as rows and SNPs as columns (numeric 0/1/2, or centered/scaled values).

Examples

```
## Not run:
df <- genoToDF(nelore_imputed, center = TRUE, scale = TRUE)
head(df[, 1:5])

## End(Not run)
```

`get.correl.fc` *Get correlation (fc method)*

Description

Calculates genotype correlation using a fast check (fc) method.

Usage

```
get.correl.fc(g1, g2)
```

Arguments

g1	Genotype vector.
g2	Genotype vector.

Value

Numeric value of correlation.

Author(s)

Roberto Higa

Examples

```
g1 <- sample(0:2, 10, TRUE)
g2 <- sample(0:2, 10, TRUE)
get.correl.fc(g1, g2)
```

`get.gender` *Get gender based on heterozygosity*

Description

Infers gender using heterozygosity thresholds.

Usage

```
get.gender(sample.summary, threshM, threshF)
```

Arguments

sample.summary	Data frame with ‘Heterozygosity‘ column.
threshM	Numeric threshold for males.
threshF	Numeric threshold for females.

Value

Data frame with columns ‘heterozygosity‘ and ‘sex‘.

Author(s)

Roberto Higa

Examples

```
df <- data.frame(Heterozygosity = c(0.1, 0.3, 0.6))
rownames(df) <- c("A", "B", "C")
get.gender(df, 0.2, 0.5)
```

---

get.hwe.chi2 *Get HWE chi-square p-values*

Description

Calculates Hardy-Weinberg equilibrium chi-square p-values for SNPs.

Usage

```
get.hwe.chi2(snp.summary)
```

Arguments

snp.summary	Data frame with columns ‘Calls‘, ‘P.AA‘, ‘P.AB‘, ‘P.BB‘.

Value

Numeric vector with p-values.

Author(s)

Roberto Higa

Examples

```
df <- data.frame(Calls = c(100, 100), P.AA = c(0.6, 0.4), P.AB = c(0.3, 0.4), P.BB = c(0.1, 0.2))
get.hwe.chi2(df)
```

getGeno — *Flexible and efficient genotype file reading with autodetection using fread*

Description

This generic and method allow flexible import of SNP genotype data from Illumina FinalReport files, supporting fast initial column detection using `data.table::fread`, followed by full genotype matrix construction via `snpStats::read.snps.long`.

Usage

```
getGeno(...)
```

Arguments

`path`	Path to the directory containing `FinalReport.txt`
`fields`	A list specifying column indices for sample, SNP, allele1, allele2, and confidence
`codes`	A character vector with allele codes (e.g., `c("A", "B")`)
`threshold`	Confidence threshold for genotype calling
`sep`	Field separator used in the files
`skip`	Number of lines to skip at the start of the file
`verbose`	Logical; if `TRUE`, displays progress messages
`every`	Frequency of progress update (number of SNPs)

Value

An `SNPDataLong` object containing the genotype matrix and map, or `NULL` if an error occurs

ibs.pair — *IBS pair statistics*

Description

Calculates IBS mean and standard deviation between two samples.

Usage

```
ibs.pair(g1, g2)
```

Arguments

`g1`	Genotype vector for first sample.
`g2`	Genotype vector for second sample.

Value

Numeric vector: [mean IBS, standard deviation].

Author(s)

Roberto Higa

Examples

```
g1 <- sample(0:2, 10, TRUE)
g2 <- sample(0:2, 10, TRUE)
ibs.pair(g1, g2)
```

---

importAllGenos *Import and combine multiple genotype configurations*

Description

This generic and method import genotype data from multiple configurations defined in an `SNPImportList` object, then combine them into a single unified `SNPDataLong` object.

Usage

```
importAllGenos(object)
```

Arguments

`object`	An object of class `SNPImportList` containing import configurations

Value

A single combined `SNPDataLong` object

---

importFImputeResults *Import imputed FImpute results from disk*

Description

Reads existing imputed results from a given path and returns an object of class SNPDataLong.

Usage

```
importFImputeResults(path, method = "R")
```

Arguments

`path`	Character. Path to the folder containing 'output_fimpute' (e.g., "fimpute_run_nelore").
`method`	Character. "R" (default) or "Rcpp". Passed to read.fimpute().

Value

An object of class SNPDataLong containing the imputed genotypes and SNP map.

import_geno_list *Import multiple genotype datasets from a list of configurations*

Description

Reads and imports multiple genotype datasets specified in a list of configurations. Each configuration must include the path to the genotype data and information on field mapping. Optionally, you can also specify codes, quality threshold, separator, lines to skip, and a subset of IDs to retain. The function automatically fills the ‘xref_path‘ slot per individual and combines maps into a single data.frame, adding a ‘SourcePath‘ column indicating their origin and removing duplicated SNP rows (by Name). Prints progress messages indicating the current path being loaded (with counter).

Usage

```
import_geno_list(config_list)
```

Arguments

config_list	A list of configuration lists. Each element should contain: - ‘path‘ (character): Path to the genotype file or folder. - ‘fields‘ (list): Named list defining the columns (e.g., SNP ID, sample ID, alleles, confidence). - ‘codes‘ (character vector, optional): Allele codes (default is c("A", "B")). - ‘threshold‘ (numeric, optional): Maximum allowed missingness or confidence threshold (default 0.15). - ‘sep‘ (character, optional): Field separator in the input file (default "tab-delimited"). - ‘skip‘ (integer, optional): Number of lines to skip at the beginning of the file (default 0). - ‘verbose‘ (logical, optional): Whether to print detailed messages (default TRUE). - ‘subset‘ (character vector, optional): Vector of sample IDs to retain after import.

Value

An object of class ‘SNPDataLong‘ containing: - Combined genotype matrix (‘geno‘). - Combined map (‘map‘) as a single data.frame with ‘SourcePath‘ column and without duplicated rows. - Combined ‘xref_path‘ vector (one entry per individual). - ‘path‘ slot as a semicolon-separated string of all input dataset paths.

pairs2sets *Convert pairs to sets*

Description

Groups sample pairs into sets of related samples.

Usage

```
pairs2sets(pairs)
```

Arguments

pairs	Matrix or list of sample pairs.

Value

List of sets of samples.

Author(s)

Roberto Higa

Examples

```
pairs <- matrix(c("A", "B", "B", "C", "D", "E"), ncol = 2, byrow = TRUE)
pairs2sets(pairs)
```

`plotPCAgroups` *Plot PCA groups from anticlustering result*

Description

Plot PCA groups from anticlustering result

Usage

```
plotPCAgroups(pca_res, groups, pcs = c(1, 2), filename = NULL)
```

Arguments

`pca_res`	A prcomp object.
`groups`	A factor or vector of group assignments.
`pcs`	Vector of length 2 indicating which PCs to plot (default: c(1, 2)).
`filename`	Optional. If provided, saves plot to this file (e.g., "antic.png").

Value

A ggplot object (also prints to screen).

Examples

```
## Not run:
res <- runAnticlusteringPCA(nelore_imputed, K = 2, n_pcs = 20)
plotPCAgroups(res$pca, res$groups)

## End(Not run)
```

qcSamples *Quality control on samples*

Description

Applies quality control (QC) procedures to samples in a ‘SNPDataLong‘ object, based on heterozygosity and call rate thresholds.

Usage

```
qcSamples(x, ...)

## S4 method for signature 'SNPDataLong'
qcSamples(
  x,
  heterozygosity = NULL,
  smp_cr = NULL,
  action = c("report", "filter", "both")
)
```

Arguments

x	An object of class ‘SNPDataLong‘.
heterozygosity	A numeric threshold or range for heterozygosity. Samples outside this threshold are removed.
smp_cr	Minimum acceptable sample call rate (between 0 and 1). Samples below this value are removed.
action	Character string indicating the action to perform. One of: - ‘"report"‘: only returns a list of samples to remove and those kept; - ‘"filter"‘: returns a filtered object without reporting; - ‘"both"‘: performs filtering and returns the filtered object.

Value

Depending on the ‘action‘ argument: - ‘"report"‘: returns a list with removed and kept samples; - ‘"filter"‘: returns a new ‘SNPDataLong‘ object with filtered genotypes; - ‘"both"‘: returns a list with: - ‘filtered‘: the filtered ‘SNPDataLong‘ object; - ‘report‘: a list of removed and kept samples.

qcSNPs *Quality Control for SNPDataLong with optional criteria*

Description

Applies flexible quality control filters on an object of class SNPDataLong. Supports call rate filtering, minor allele frequency (MAF), Hardy-Weinberg equilibrium (HWE), removal of monomorphic SNPs, exclusion of specific chromosomes, optionally removing SNPs without positions, and optionally removing SNPs at the same genomic position (keeping the one with highest MAF).

Usage

```
qcSNPs(x, ...)
```

Arguments

x	An object of class SNPDataLong.
missing_ind	Maximum allowed proportion of missing data per individual (currently not implemented).
missing_snp	Maximum allowed proportion of missing data per SNP (currently not implemented).
min_snp_cr	Minimum acceptable call rate for SNPs (e.g., 0.95). SNPs below this threshold are removed.
min_maf	Minimum minor allele frequency allowed for SNPs (e.g., 0.05). SNPs with lower MAF are removed.
hwe	p-value threshold for Hardy-Weinberg equilibrium test (e.g., 1e-6). SNPs violating this are removed.
snp_position	Logical. If TRUE, removes SNPs mapped to the same position, retaining only the one with highest MAF.
no_position	Logical. If TRUE, removes SNPs without defined genomic positions.
snp_mono	Logical. If TRUE, removes monomorphic SNPs (with no variation).
remove_chr	Character vector of chromosomes to exclude (e.g., c("X", "Y")).
action	One of "report" (returns a list of removed SNPs), "filter" (returns filtered SNPDataLong), or "both" (returns both).

Value

Depending on the action argument: - "report": list of SNPs removed by each filter and SNPs retained. - "filter": filtered SNPDataLong object. - "both": list containing the filtered object and detailed report.

Examples

```
## Not run:
set.seed(123)
mat <- matrix(sample(c(0, 1, 2, NA), 100, replace = TRUE, prob = c(0.4, 0.4, 0.15, 0.05)),
              nrow = 10, ncol = 10)
colnames(mat) <- paste0("snp", 1:10)
rownames(mat) <- paste0("ind", 1:10)
map <- data.frame(Name = colnames(mat), Chromosome = 1, Position = 1:10)
x <- new("SNPDataLong", geno = mat, map = map, path = "dummy_path", xref_path = rep("chip1", 10))

# Example using multiple filters
qcSNPs(x, min_snp_cr = 0.8, min_maf = 0.05, snp_mono = TRUE, no_position = TRUE, snp_position = TRUE, action = "f

## End(Not run)
```

qc_header *Formatted header message*

Description

Prints a formatted message with a border for section titles in the console.

Usage

```
qc_header(title)
```

Arguments

title	Character string to be printed inside the header box.

Value

No return value. Used for side effects (message).

Examples

```
qc_header("Quality Control on Samples")
```

rbindSnpFlexible *Faster row-bind for SnpMatrix objects with differing columns*

Description

Combines multiple SnpMatrix objects by rows, automatically handling differing SNP columns, optimized for large matrices.

Usage

```
rbindSnpFlexible(...)
```

Arguments

...	One or more SnpMatrix objects.

Value

A single SnpMatrix object with all rows combined.

Examples

```
## Not run:
combined <- rbindSnpFlexible(brangus_geno, batch_BM@geno)

## End(Not run)
```

rbind_SnpMatrix *Safe rbind for SnpMatrix preserving dimnames*

Description

This function performs a row-wise binding of multiple SnpMatrix objects, explicitly preserving row names and column names, avoiding unexpected "object has no names" warnings.

Usage

```
rbind_SnpMatrix(...)
```

Arguments

...	SnpMatrix objects to combine (must have identical column names).

Value

A single combined SnpMatrix with preserved row and column names.

Examples

```
## Not run:
rbind_SnpMatrix(matrix1, matrix2)

## End(Not run)
```

read.fimpute *Read imputed genotypes from FImpute output and return SNPDataLong object*

Description

Reads imputed genotypes and SNP information from FImpute output, builds a SnpMatrix and a corresponding map, and returns an SNPDataLong object.

Usage

```
read.fimpute(file, method = c("R", "Rcpp"))
```

Arguments

file	Character. Path to the FImpute output directory (usually "output_fimpute").
method	Character. "R" (default) for vectorized R implementation, or "Rcpp" for compiled C++ implementation.

Value

An object of class SNPDataLong containing the imputed genotypes and SNP map.

Examples

```
## Not run:
snp_long <- read.fimpute("output_fimpute", method = "R")

## End(Not run)
```

`runAnticlusteringPCA` *Run PCA and Anticlustering on SNPDataLong*

Description

Converts a SNPDataLong object to a data.frame, runs PCA, and performs anticlustering grouping.

Usage

```
runAnticlusteringPCA(object, K = 2, n_pcs = 20, center = TRUE, scale = TRUE)
```

Arguments

`object`	An object of class SNPDataLong.
`K`	Number of groups for anticlustering.
`n_pcs`	Number of top principal components to use (default: 20).
`center`	Logical or numeric. Center columns before PCA (default: TRUE).
`scale`	Logical or numeric. Scale columns before PCA (default: TRUE).

Value

A list with: - groups: vector with group assignments. - pca: the PCA result object (prcomp). - pcs: matrix of top PCs used in anticlustering.

Examples

```
## Not run:
res <- runAnticlusteringPCA(nelore_imputed, K = 2, n_pcs = 20)
table(res$groups)

## End(Not run)
```

runFImpute *Run FImpute from a FImputeRunner object*

Description

This function runs the external FImpute software using a 'FImputeRunner' object, ensuring that all required input files are present and the results are imported.

Usage

```
runFImpute(object, verbose = TRUE)

## S4 method for signature 'FImputeRunner'
runFImpute(object, verbose = TRUE)
```

Arguments

object	An object of class 'FImputeRunner'.
verbose	Logical. If TRUE (default), FImpute output will be printed to the console.

Value

An updated 'FImputeRunner' object with the 'results' slot populated (SnpMatrix).

Examples

```
## Not run:
# Example: Running FImpute from a FImputeRunner object

path_fimpute <- "fimpute_run_example"
param_file <- file.path(path_fimpute, "fimpute.par")
fimpute_exec <- "FImpute3"  # assuming it is in PATH

export_obj <- new("FImputeExport",
                  geno = geno_obj@geno,
                  map = geno_obj@map,
                  path = path_fimpute)

runner <- new("FImputeRunner",
              export = export_obj,
              par_file = param_file,
              exec_path = fimpute_exec)

runner <- runFImpute(runner, verbose = TRUE)
head(runner@results)

## End(Not run)
```

saveFImpute *Save genotype and map files in FImpute format*

Description

S4 method to export genotype (.gen), map (.map), and parameter (fimpute.par) files compatible with [FImpute](https://www.aps.uoguelph.ca/~msargol/fimpute/).

Usage

```
saveFImpute(object, ...)

## S4 method for signature 'FImputeExport'
saveFImpute(object)

## S4 method for signature 'SNPDataLong'
saveFImpute(object, path = NULL)
```

Arguments

object	An object of class ‘FImputeExport‘ or ‘SNPDataLong‘.
...	Additional arguments passed to methods.
path	Output directory (default: "fimpute_run" for SNPDataLong).

Value

No return value. Files are saved to disk.

saveFImputeRaw *Export genotypes and map using basic arguments*

Description

Convenience function to export FImpute files directly from a ‘SnpMatrix‘ and map ‘data.frame‘.

Usage

```
saveFImputeRaw(geno, map, path, xref = NULL)
```

Arguments

geno	A ‘SnpMatrix‘ object.
map	A data.frame with columns ’Name’, ’Chromosome’, ’Position’, and ’SourcePath’.
path	Output directory.
xref	Optional vector of identifiers per individual (used to assign numeric chip IDs).

savePlink *Save SNPDataLong object to PLINK format*

Description

Saves genotype and map data from an SNPDataLong object in PLINK format (.ped/.map and optionally binary files).

Usage

```
savePlink(
  object,
  path = "plink_out",
  name = "plink_data",
  run_plink = TRUE,
  chunk_size = 1000
)
```

Arguments

object	An object of class SNPDataLong.
path	Character. Directory where files will be saved.
name	Character. Base name for PLINK output files.
run_plink	Logical. If TRUE (default), runs PLINK1 to convert to binary files. If FALSE, only .ped and .map files are saved.
chunk_size	Integer. Number of individuals per chunk for writing .ped file (default: 1000).

Value

No return value. Files are saved to disk.

Examples

```
## Not run:
savePlink(genotypes_qc, path = "plink_out", name = "nelore_qc", run_plink = TRUE, chunk_size = 2000)

## End(Not run)
```

Subset *Subset method for SNPDataLong*

Description

Subset method for SNPDataLong

Usage

```
Subset(object, index, margin = 1, keep = TRUE)
```

Arguments

object	A SNPDataLong object.
index	Character vector with row (individual) or column (SNP) names to filter.
margin	Integer: 1 = rows (individuals), 2 = columns (SNPs).
keep	Logical: TRUE to keep specified levels, FALSE to discard them.

Value

A new SNPDataLong object, subsetted accordingly.

---

summary,SNPDataLong-method

Summary for SNPDataLong objects

Description

Provides a detailed summary of an SNPDataLong object, including sample and SNP counts, proportion of missing data, and SNP distribution by chromosome if mapping information is available.

Usage

```
## S4 method for signature 'SNPDataLong'
summary(object, ...)
```

Arguments

object	An object of class SNPDataLong.

Value

Prints a summary to the console. Returns NULL (invisible).

Index