Package 'SNPtools'

July 2, 2025

Title S4 Tools for Reading and Organizing Genetic Data

Version 0.1.0

Maintainer Vinícius Junqueira <junqueiravinicius@hotmail.com>

Description Provides S4-based structures to encapsulate the import, organization, and processing of genetic data from PLINK and FImpute files, with customizable arguments. Includes tools for combining SNP panels, summarizing genotype data, and facilitating downstream quality control and analysis workflows.

Depends R (>= 4.1.0),
snpStats,
tidyverse

Imports methods,
data.table,
fQC,
Rcpp

LinkingTo Rcpp

Suggests knitr,
rmarkdown

VignetteBuilder knitr

Encoding UTF-8

License GPL-3

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.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.

`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)
```

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)
```

`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

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)
```

Arguments

path	Character. Path to the folder containing 'output_fimpute' (e.g., "fimpute_run_nelore").

Value

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

Examples

```
## Not run:
imputed_obj <- importFImputeResults("fimpute_run_nelore")
head(imputed_obj@map)

## End(Not run)
```

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

Description

This function iterates over a list of configuration lists (each specifying parameters such as path, fields, separators, etc.), imports each genotype dataset using `getGeno()`, optionally subsets by individual IDs, and then combines them into a single `SNPDataLong` object.

Usage

```
import_geno_list(config_list)
```

Arguments

`config_list` A list of configuration lists. Each configuration must include at least:

`path` Character. Directory containing the genotype file (FinalReport.txt).

`fields` List. Specifies column indices in FinalReport for sample ID, SNP ID, allele1, allele2, and confidence. E.g., `list(sample = 2, snp = 1, allele1 = 7, allele2 = 8, confidence = 9)`.

Optional elements in each configuration include:

`codes` Character vector. Possible genotype allele codes (default: `c("A", "B")`).

`threshold` Numeric. Confidence threshold for genotype calling (default: 0.15).

`sep` Character. Field separator in the file (default: tab).

`skip` Integer. Number of lines to skip at the start of the file (default: 0).

`verbose` Logical. Whether to print progress messages (default: TRUE).

`subset` Character vector. Optional vector of sample IDs to keep (subsets individuals before merging).

Value

A unified `SNPDataLong` object containing combined genotype data from all configurations.

Examples

```
## Not run:
configs <- list(
  list(path = "panel1", fields = list(sample = 2, snp = 1, allele1 = 7, allele2 = 8, confidence = 9)),
  list(path = "panel2", fields = list(sample = 2, snp = 1, allele1 = 7, allele2 = 8, confidence = 9), threshold = 0.10
)
combined_data <- import_geno_list(configs)

## End(Not run)
```

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.

Usage

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

Arguments

`x` An object of class SNPDataLong.

`missing_ind` Maximum allowed proportion of missing data per individual (optional). *[Currently not implemented]*

`missing_snp` Maximum allowed proportion of missing data per SNP (optional). *[Currently not implemented]*

`min_snp_cr` Minimum acceptable call rate for SNPs (e.g., 0.95).

`min_maf` Minimum minor allele frequency allowed for SNPs (e.g., 0.05).

`hwe` p-value threshold for Hardy-Weinberg equilibrium test (e.g., 1e-6).

`snp_position` Logical. If TRUE, removes SNPs mapped to the same position, keeping the one with highest MAF.

`no_position` Logical. If TRUE, removes SNPs without defined map positions.

`snp_mono` Logical. If TRUE, removes monomorphic SNPs.

`remove_chr` Character vector of chromosomes to exclude (e.g., c("X", "Y")).

`action` One of "report" (list of removed SNPs), "filter" (returns filtered SNPDataLong), or "both" (both results).

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)

# Example with multiple filters and returning a detailed report
qcSNPs(x, min_snp_cr = 0.8, min_maf = 0.05, snp_mono = TRUE, no_position = TRUE, snp_position = TRUE, action = ”repo

## 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)
```

Arguments

`file`	Character. Path to the FImpute output directory (usually "output_fimpute").

Value

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

Examples

```
## Not run:
snp_long <- read.fimpute("output_fimpute")

## 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)
```

Arguments

`geno`	A ‘SnpMatrix‘ object.
`map`	A data.frame with columns ’Name’, ’Chromosome’, ’Position’.
`path`	Output directory.

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)
```

`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