Package 'bigsnpr'

July 5, 2020

Encoding UTF-8

Type Package

Title Analysis of Massive SNP Arrays

Version 1.4.4

Date 2020-07-03

Description Easy-to-use, efficient, flexible and scalable tools for analyzing massive SNP arrays <doi:10.1093/bioinformatics/bty185>.

License GPL-3

LazyData TRUE

Language en-US

ByteCompile TRUE

SystemRequirements Package 'bigsnpr' includes a few functions that wrap existing software such as 'PLINK' <www.cog-genomics.org/plink2>. Functions are provided to download these software. Note that these external software might not work for some operating systems (e.g. 'PLINK' might not work on Solaris).

Depends R (>= 3.3), bigstatsr (>= 1.2.2)

Imports bigassertr (>= 0.1.3), bigparallelr, bigsparser (>= 0.2.3), bigreadr, bigutilsr (>= 0.3), data.table, foreach, ggplot2, magrittr, Matrix, methods, Rcpp, stats

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

Suggests bindata, covr, dbplyr (>= 1.4), dplyr, gaston, glue, Hmisc, pcadapt (>= 4.1), rmutil, RSpectra, RSQLite, R.utils, spelling, testthat, xgboost

RoxygenNote 7.0.2

URL https://privefl.github.io/bigsnpr

BugReports https://github.com/privefl/bigsnpr/issues

NeedsCompilation yes

Author Florian Privé [aut, cre],
Michael Blum [ths],
Hugues Aschard [ths]

Maintainer Florian Privé <florian.prive.21@gmail.com>

Repository CRAN

Date/Publication 2020-07-05 18:40:16 UTC

R topics documented:

`bigsnpr-package` *bigsnpr: Analysis of Massive SNP Arrays*

Description

Easy-to-use, efficient, flexible and scalable tools for analyzing massive SNP arrays <doi:10.1093/bioinformatics/bty185>.

Arguments

`G`	A FBM.code256 (typically <bigSNP>$genotypes). **You shouldn't have missing values.** Also, remember to do quality control, e.g. some algorithms in this package won't work if you use SNPs with 0 MAF.
`Gna`	A FBM.code256 (typically <bigSNP>$genotypes). You can have missing values in these data.
`x`	A bigSNP.
`infos.chr`	Vector of integers specifying each SNP's chromosome. Typically <bigSNP>mapchromosome.
`infos.pos`	Vector of integers specifying the physical position on a chromosome (in base pairs) of each SNP. Typically <bigSNP>mapphysical.pos.
`nploidy`	Number of trials, parameter of the binomial distribution. Default is `2`, which corresponds to diploidy, such as for the human genome.

`ind.row`	An optional vector of the row indices (individuals) that are used. If not specified, all rows are used. **Don't use negative indices.**
`ind.col`	An optional vector of the column indices (SNPs) that are used. If not specified, all columns are used. **Don't use negative indices.**
`ncores`	Number of cores used. Default doesn't use parallelism. You may use nb_cores.
`is.size.in.bp`	Deprecated.
`obj.bed`	Object of type bed, which is the mapping of some bed file. Use `obj.bed <-bed(bedfile)` to get this object.

Author(s)

Maintainer: Florian Privé <florian.prive.21@gmail.com>

Other contributors:

- Michael Blum <michael.blum@univ-grenoble-alpes.fr> [thesis advisor]
- Hugues Aschard <hugues.aschard@pasteur.fr> [thesis advisor]

See Also

Useful links:

- https://privefl.github.io/bigsnpr
- Report bugs at https://github.com/privefl/bigsnpr/issues

`bed-class` *Class bed*

Description

A reference class for storing a pointer to a mapped version of a bed file.

Usage

```
bed(bedfile)
```

Arguments

`bedfile`	Path to file with extension ".bed" to read. You need the corresponding ".bim" and ".fam" in the same directory.

Details

A bed object has many field:

- $address: address of the external pointer containing the underlying C++ object, to be used internally as a XPtr<bed> in C++ code
- $extptr: use $address instead
- $bedfile: path to the bed file
- $bimfile: path to the corresponding bim file
- $famfile: path to the corresponding fam file
- $prefix: path without extension
- $nrow: number of samples in the bed file
- $ncol: number of variants in the bed file
- $map: data frame read from $bimfile
- $fam: data frame read from $famfile
- $.map: use $map instead
- $.fam: use $fam instead
- $light: get a lighter version of this object for parallel algorithms to not have to transfer e.g. $.map.

Examples

```
bedfile <- system.file("extdata", "example-missing.bed", package = "bigsnpr")
(obj.bed <- bed(bedfile))
```

---

bed-methods *Methods for the bed class*

Description

Methods for the bed class

Dimension methods for class bed. Methods `nrow()` and `ncol()` are automatically defined with `dim()`.

Usage

```
## S4 method for signature 'bed'
dim(x)

## S4 method for signature 'bed'
length(x)
```

Arguments

x	Object of type bed.

bed_clumping *LD clumping*

Description

For a bigSNP:

- snp_pruning(): LD pruning. Similar to "--indep-pairwise (size+1) 1 thr.r2" in PLINK. **This function is deprecated (see this article).**
- snp_clumping() (and bed_clumping()): LD clumping. If you do not provide any statistic to rank SNPs, it would use minor allele frequencies (MAFs), making clumping similar to pruning.
- snp_indLRLDR(): Get SNP indices of long-range LD regions for the human genome.

Usage

```
bed_clumping(
  obj.bed,
  ind.row = rows_along(obj.bed),
  S = NULL,
  thr.r2 = 0.2,
  size = 100/thr.r2,
  exclude = NULL,
  ncores = 1
)

snp_clumping(
  G,
  infos.chr,
  ind.row = rows_along(G),
  S = NULL,
  thr.r2 = 0.2,
  size = 100/thr.r2,
  infos.pos = NULL,
  is.size.in.bp = NULL,
  exclude = NULL,
  ncores = 1
)

snp_pruning(
  G,
  infos.chr,
  ind.row = rows_along(G),
  size = 49,
  is.size.in.bp = FALSE,
  infos.pos = NULL,
```

```
  thr.r2 = 0.2,
  exclude = NULL,
  nploidy = 2,
  ncores = 1
)

snp_indLRLDR(infos.chr, infos.pos, LD.regions = LD.wiki34)
```

Arguments

obj.bed	Object of type bed, which is the mapping of some bed file. Use obj.bed <-bed(bedfile) to get this object.
ind.row	An optional vector of the row indices (individuals) that are used. If not specified, all rows are used. **Don't use negative indices.**
S	A vector of column 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 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.
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.
ncores	Number of cores used. Default doesn't use parallelism. You may use nb_cores.
G	A FBM.code256 (typically <bigSNP>$genotypes). **You shouldn't have missing values.** Also, remember to do quality control, e.g. some algorithms in this package won't work if you use SNPs with 0 MAF.
infos.chr	Vector of integers specifying each SNP's chromosome. Typically <bigSNP>mapchromosome.
infos.pos	Vector of integers specifying the physical position on a chromosome (in base pairs) of each SNP. Typically <bigSNP>mapphysical.pos.
is.size.in.bp	Deprecated.
nploidy	Number of trials, parameter of the binomial distribution. Default is 2, which corresponds to diploidy, such as for the human genome.
LD.regions	A data.frame with columns "Chr", "Start" and "Stop". Default use the table of 34 long-range LD regions that you can find there.

Value

- `snp_clumping()` (and `bed_clumping()`): SNP indices that are **kept**.
- `snp_indLRLDR()`: SNP indices to be used as (part of) the 'exclude' parameter of `snp_clumping()`.

References

Price AL, Weale ME, Patterson N, et al. Long-Range LD Can Confound Genome Scans in Admixed Populations. Am J Hum Genet. 2008;83(1):132-135. http://dx.doi.org/10.1016/j.ajhg.2008.06.005

Examples

```
test <- snp_attachExtdata()
G <- test$genotypes

# clumping (prioritizing higher MAF)
ind.keep <- snp_clumping(G, infos.chr = test$map$chromosome,
                         infos.pos = test$map$physical.pos,
                         thr.r2 = 0.1)

# keep most of them -> not much LD in this simulated dataset
length(ind.keep) / ncol(G)
```

---

bed_counts *Counts*

Description

Counts the number of 0s, 1s, 2s and NAs by variants in the bed file.

Usage

```
bed_counts(
  obj.bed,
  ind.row = rows_along(obj.bed),
  ind.col = cols_along(obj.bed),
  byrow = FALSE,
  ncores = 1
)
```

Arguments

`obj.bed`	Object of type bed, which is the mapping of some bed file. Use `obj.bed <-bed(bedfile)` to get this object.
`ind.row`	An optional vector of the row indices (individuals) that are used. If not specified, all rows are used. **Don't use negative indices.**

ind.col	An optional vector of the column indices (SNPs) that are used. If not specified, all columns are used. **Don't use negative indices.**
byrow	Whether to count by individual rather than by variant? Default is FALSE (count by variant).
ncores	Number of cores used. Default doesn't use parallelism. You may use nb_cores.

Value

A matrix of with 4 rows and length(ind.col) columns.

Examples

```
bedfile <- system.file("extdata", "example-missing.bed", package = "bigsnpr")
obj.bed <- bed(bedfile)

bed_counts(obj.bed, ind.col = 1:5)

bed_counts(obj.bed, ind.row = 1:5, byrow = TRUE)
```

---

bed_cprodVec *Cross-product with a vector*

Description

Cross-product between a "bed" object and a vector.

Missing values are replaced by 0 (after centering), as if they had been imputed using parameter center.

Usage

```
bed_cprodVec(
  obj.bed,
  y.row,
  ind.row = rows_along(obj.bed),
  ind.col = cols_along(obj.bed),
  center = rep(0, length(ind.col)),
  scale = rep(1, length(ind.col)),
  ncores = 1
)
```

Arguments

obj.bed	A bed object.
y.row	A vector of same size as ind.row.
ind.row	An optional vector of the row indices (individuals) that are used. If not specified, all rows are used. **Don't use negative indices.**
ind.col	An optional vector of the column indices (SNPs) that are used. If not specified, all columns are used. **Don't use negative indices.**
center	Vector of same length of ind.col to subtract from columns of X.
scale	Vector of same length of ind.col to divide from columns of X.
ncores	Number of cores used. Default doesn't use parallelism. You may use nb_cores.

Value

$X^T \cdot y$.

Examples

```
bedfile <- system.file("extdata", "example.bed", package = "bigsnpr")
obj.bed <- bed(bedfile)

y.row <- rep(1, nrow(obj.bed))
str(bed_cprodVec(obj.bed, y.row))
```

---

bed_MAF *Allele frequencies*

Description

Allele frequencies of a bed object.

Usage

```
bed_MAF(
  obj.bed,
  ind.row = rows_along(obj.bed),
  ind.col = cols_along(obj.bed),
  ncores = 1
)
```

Arguments

obj.bed	Object of type bed, which is the mapping of some bed file. Use obj.bed <-bed(bedfile) to get this object.
ind.row	An optional vector of the row indices (individuals) that are used. If not specified, all rows are used. **Don't use negative indices.**
ind.col	An optional vector of the column indices (SNPs) that are used. If not specified, all columns are used. **Don't use negative indices.**
ncores	Number of cores used. Default doesn't use parallelism. You may use nb_cores.

Value

A data.frame with

- $ac: allele counts,
- $mac: minor allele counts,
- $af: allele frequencies,
- $maf: minor allele frequencies,
- $N: numbers of non-missing values.

Examples

```
bedfile <- system.file("extdata", "example-missing.bed", package = "bigsnpr")
obj.bed <- bed(bedfile)

bed_MAF(obj.bed, ind.col = 1:5)
```

---

bed_prodVec *Product with a vector*

Description

Product between a "bed" object and a vector.

Missing values are replaced by 0 (after centering), as if they had been imputed using parameter center.

Usage

```
bed_prodVec(
  obj.bed,
  y.col,
  ind.row = rows_along(obj.bed),
  ind.col = cols_along(obj.bed),
  center = rep(0, length(ind.col)),
```

```
  scale = rep(1, length(ind.col)),
  ncores = 1
)
```

Arguments

obj.bed	A bed object.
y.col	A vector of same size as ind.col.
ind.row	An optional vector of the row indices (individuals) that are used. If not specified, all rows are used. **Don't use negative indices.**
ind.col	An optional vector of the column indices (SNPs) that are used. If not specified, all columns are used. **Don't use negative indices.**
center	Vector of same length of ind.col to subtract from columns of X.
scale	Vector of same length of ind.col to divide from columns of X.
ncores	Number of cores used. Default doesn't use parallelism. You may use nb_cores.

Value

$X \cdot y$.

Examples

```
bedfile <- system.file("extdata", "example.bed", package = "bigsnpr")
obj.bed <- bed(bedfile)

y.col <- rep(1, ncol(obj.bed))
str(bed_prodVec(obj.bed, y.col))
```

---

bed_projectPCA *Projecting PCA*

Description

Computing and projecting PCA of reference dataset to a target dataset.

Usage

```
bed_projectPCA(
  obj.bed.ref,
  obj.bed.new,
  k = 10,
  ind.row.new = rows_along(obj.bed.new),
  ind.row.ref = rows_along(obj.bed.ref),
```

```
  ind.col.ref = cols_along(obj.bed.ref),
  strand_flip = TRUE,
  join_by_pos = TRUE,
  match.min.prop = 0.5,
  build.new = "hg19",
  build.ref = "hg19",
  liftOver = NULL,
  ...,
  verbose = TRUE,
  ncores = 1
)
```

Arguments

`obj.bed.ref`	Object of type bed, which is the mapping of the bed file of the reference data. Use `obj.bed <-bed(bedfile)` to get this object.
`obj.bed.new`	Object of type bed, which is the mapping of the bed file of the target data. Use `obj.bed <-bed(bedfile)` to get this object.
`k`	Number of principal components to compute and project.
`ind.row.new`	Rows to be used in the target data. Default uses them all.
`ind.row.ref`	Rows to be used in the reference data. Default uses them all.
`ind.col.ref`	Columns to be potentially used in the reference data. Default uses all the ones in common with target data.
`strand_flip`	Whether to try to flip strand? (default is `TRUE`) If so, ambiguous alleles A/T and C/G are removed.
`join_by_pos`	Whether to join by chromosome and position (default), or instead by rsid.
`match.min.prop`	Minimum proportion of variants in the smallest data to be matched, otherwise stops with an error. Default is 50%.
`build.new`	Genome build of the target data. Default is `hg19`.
`build.ref`	Genome build of the reference data. Default is `hg19`.
`liftOver`	Path to liftOver executable. Binaries can be downloaded at `https://bit.ly/2KvHugi` for Mac and at `https://bit.ly/2TbSaEI` for Linux.
`...`	Arguments passed on to `bed_autoSVD`
	`fun.scaling` A function that returns a named list of `mean` and `sd` for every column, to scale each of their elements such as followed: $$\frac{X_{i,j} - mean_j}{sd_j}.$$ Default is `snp_scaleBinom()`.
	`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`.
	`thr.r2` Threshold over the squared correlation between two SNPs. Default is `0.2`. Use `NA` if you want to skip the clumping step.

`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`.

`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.

`verbose` Output some information on the iterations? Default is `TRUE`.

`ncores` Number of cores used. Default doesn't use parallelism. You may use nb_cores.

Value

A list of 3 elements:

- $obj.svd.ref: big_SVD object computed from reference data.
- $simple_proj: simple projection of new data into space of reference PCA.
- $OADP_proj: Online Augmentation, Decomposition, and Procrustes (OADP) projection of new data into space of reference PCA.

---

bed_projectSelfPCA *Projecting PCA*

Description

Projecting PCA using individuals from one dataset to other individuals from the same dataset.

Usage

```
bed_projectSelfPCA(
  obj.svd,
  obj.bed,
  ind.row,
  ind.col = attr(obj.svd, "subset"),
  ncores = 1
)
```

Arguments

`obj.svd` List with `v`, `d`, `center` and `scale`. Typically the an object of type "big_SVD".

`obj.bed` Object of type bed, which is the mapping of the bed file of the data containing both the individuals that were used to compute the PCA and the other individuals to be projected.

`ind.row` Rows (individuals) to be projected.

ind.col	Columns that were used for computing PCA. If bed_autoSVD was used, then attr(obj.svd,"subset") is automatically used by default. Otherwise (e.g. if bed_randomSVD was used), you have to pass ind.col.
ncores	Number of cores used. Default doesn't use parallelism. You may use nb_cores.

Value

A list of 3 elements:

- $obj.svd.ref: big_SVD object computed from reference data.
- $simple_proj: simple projection of new data into space of reference PCA.
- $OADP_proj: Online Augmentation, Decomposition, and Procrustes (OADP) projection of new data into space of reference PCA.

---

bed_randomSVD *Randomized partial SVD*

Description

Partial SVD (or PCA) of a genotype matrix stored as a PLINK (.bed) file.#'

Usage

```
bed_randomSVD(
  obj.bed,
  fun.scaling = bed_scaleBinom,
  ind.row = rows_along(obj.bed),
  ind.col = cols_along(obj.bed),
  k = 10,
  tol = 1e-04,
  verbose = FALSE,
  ncores = 1
)
```

Arguments

obj.bed	Object of type bed, which is the mapping of some bed file. Use obj.bed <-bed(bedfile) to get this object.
fun.scaling	A function that returns a named list of mean and sd for every column, to scale each of their elements such as followed: $$\frac{X_{i,j} - mean_j}{sd_j}.$$ Default doesn't use any scaling.
ind.row	An optional vector of the row indices (individuals) that are used. If not specified, all rows are used. **Don't use negative indices.**

ind.col	An optional vector of the column indices (SNPs) that are used. If not specified, all columns are used. **Don't use negative indices.**
k	Number of singular vectors/values to compute. Default is 10. **This algorithm should be used to compute only a few singular vectors/values.**
tol	Precision parameter of svds. Default is 1e-4.
verbose	Should some progress be printed? Default is FALSE.
ncores	Number of cores used. Default doesn't use parallelism. You may use nb_cores.

Value

A named list (an S3 class "big_SVD") of

- d, the singular values,
- u, the left singular vectors,
- v, the right singular vectors,
- niter, the number of the iteration of the algorithm,
- nops, number of Matrix-Vector multiplications used,
- center, the centering vector,
- scale, the scaling vector.

Note that to obtain the Principal Components, you must use predict on the result. See examples.

Examples

```
bedfile <- system.file("extdata", "example.bed", package = "bigsnpr")
obj.bed <- bed(bedfile)

str(bed_randomSVD(obj.bed))
```

---

bed_scaleBinom *Binomial(2, p) scaling*

Description

Binomial(2, p) scaling where p is estimated.

Usage

```
bed_scaleBinom(
  obj.bed,
  ind.row = rows_along(obj.bed),
  ind.col = cols_along(obj.bed),
  ncores = 1
)
```

Arguments

obj.bed	Object of type bed, which is the mapping of some bed file. Use obj.bed <-bed(bedfile) to get this object.
ind.row	An optional vector of the row indices (individuals) that are used. If not specified, all rows are used. **Don't use negative indices.**
ind.col	An optional vector of the column indices (SNPs) that are used. If not specified, all columns are used. **Don't use negative indices.**
ncores	Number of cores used. Default doesn't use parallelism. You may use nb_cores.

Details

You will probably not use this function as is but as parameter fun.scaling of other functions (e.g. bed_autoSVD and bed_randomSVD).

Value

A data frame with $center and $scale.

References

This scaling is widely used for SNP arrays. Patterson N, Price AL, Reich D (2006). Population Structure and Eigenanalysis. PLoS Genet 2(12): e190. http://dx.doi.org/10.1371/journal.pgen.0020190.

Examples

```
bedfile <- system.file("extdata", "example-missing.bed", package = "bigsnpr")
obj.bed <- bed(bedfile)

str(bed_scaleBinom(obj.bed))

str(bed_randomSVD(obj.bed, bed_scaleBinom))
```

---

bed_tcrossprodSelf *Tcrossprod*

Description

Compute $X.row X.row^T$ for a Filebacked Big Matrix X after applying a particular scaling to it.

Usage

```
bed_tcrossprodSelf(
  obj.bed,
  fun.scaling = bed_scaleBinom,
  ind.row = rows_along(obj.bed),
  ind.col = cols_along(obj.bed),
  block.size = block_size(length(ind.row))
)
```

Arguments

`obj.bed`	Object of type `bed`, which is the mapping of some bed file. Use `obj.bed <-bed(bedfile)` to get this object.
`fun.scaling`	A function that returns a named list of `mean` and `sd` for every column, to scale each of their elements such as followed: $$\frac{X_{i,j} - mean_j}{sd_j}.$$ Default doesn't use any scaling.
`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. Default uses block_size.

Value

A temporary FBM, with the following two attributes:

- a numeric vector `center` of column scaling,
- a numeric vector `scale` of column scaling.

Examples

```
bedfile <- system.file("extdata", "example.bed", package = "bigsnpr")
obj.bed <- bed(bedfile)

K <- bed_tcrossprodSelf(obj.bed)
K[1:4, 1:6] / ncol(obj.bed)
```

bigSNP-class *Class bigSNP*

Description

An S3 class for representing information on massive SNP arrays.

Details

A named list with at least 4 slots:

genotypes A FBM.code256 which is a special Filebacked Big Matrix encoded with type `raw` (one byte unsigned integer), representing genotype calls and possibly imputed allele dosages. Rows are individuals and columns are SNPs.

fam A `data.frame` containing some information on the SNPs (read from a ".fam" file).

map A `data.frame` giving some information on the individuals (read from a ".bim" file).

See Also

snp_readBed

CODE_012 *CODE_012: code genotype calls (3) and missing values.*

Description

CODE_012: code genotype calls (3) and missing values.

CODE_DOSAGE: code genotype calls and missing values (4), and imputed calls (3) and imputed allele dosages rounded to two decimal places (201).

CODE_IMPUTE_PRED: code genotype calls and missing values (4), and imputed calls (3).

Usage

```
CODE_012

CODE_DOSAGE

CODE_IMPUTE_PRED
```

Format

An object of class `numeric` of length 256.

coef_to_liab *Liability scale*

Description

Coefficient to convert to the liability scale. E.g. h2_liab = coef * h2_obs.

Usage

```
coef_to_liab(K_pop, K_gwas = 0.5)
```

Arguments

K_pop	Prevalence in the population.
K_gwas	Prevalence in the GWAS. You should provide this if you used (n_case + n_control) as sample size. If using the effective sample size 4 / (1 / n_case + 1 / n_control) instead, you should keep the default value of K_gwas = 0.5 as the GWAS case-control ascertainment is already accounted for in the effective sample size.

Value

Scaling coefficient to convert e.g. heritability to the liability scale.

Examples

```
h2 <- 0.2
h2 * coef_to_liab(0.02)
```

download_1000G *Download 1000G*

Description

Download 1000 genomes project (phase 3) data in PLINK bed/bim/fam format, including 2490 (mostly unrelated) individuals and ~1.7M SNPs in common with either HapMap3 or the UK Biobank.

Usage

```
download_1000G(dir, overwrite = FALSE, delete_zip = TRUE)
```

Arguments

dir	The directory where to put the downloaded files.
overwrite	Whether to overwrite files when downloading and unzipping? Default is FALSE.
delete_zip	Whether to delete zip after decompressing the file in it? Default is TRUE.

Value

The path of the downloaded bed file.

`download_beagle` *Download Beagle 4.1*

Description

Download Beagle 4.1 from https://faculty.washington.edu/browning/beagle/beagle.html

Usage

```
download_beagle(dir = tempdir())
```

Arguments

`dir`	The directory where to put the Beagle Java Archive. Default is a temporary directory.

Value

The path of the downloaded Beagle Java Archive.

`download_plink` *Download PLINK*

Description

Download PLINK 1.9 from http://www.cog-genomics.org/plink2.

Download PLINK 2.0 from http://www.cog-genomics.org/plink/2.0/.

Usage

```
download_plink(dir = tempdir(), overwrite = FALSE, verbose = TRUE)

download_plink2(
  dir = tempdir(),
  AVX2 = TRUE,
  overwrite = FALSE,
  verbose = TRUE
)
```

Arguments

dir	The directory where to put the PLINK executable. Default is a temporary directory.
overwrite	Whether to overwrite file? Default is FALSE.
verbose	Whether to output details of downloading. Default is TRUE.
AVX2	Whether to download the AVX2 version? This is only available for 64 bits architectures. Default is TRUE.

Value

The path of the downloaded PLINK executable.

LD.wiki34 *Long-range LD regions*

Description

34 long-range Linkage Disequilibrium (LD) regions for the human genome based on some wiki table.

Usage

```
LD.wiki34
```

Format

A data frame with 34 rows (regions) and 4 variables:

- Chr: region's chromosome
- Start: starting position of the region (in bp)
- Stop: stopping position of the region (in bp)
- ID: some ID of the region.

same_ref *Determine reference divergence*

Description

Determine reference divergence while accounting for strand flips. **This does not remove ambiguous alleles.**

Usage

```
same_ref(ref1, alt1, ref2, alt2)
```

Arguments

ref1	The reference alleles of the first dataset.
alt1	The alternative alleles of the first dataset.
ref2	The reference alleles of the second dataset.
alt2	The alternative alleles of the second dataset.

Value

A logical vector whether the references alleles are the same. Missing values can result from missing values in the inputs or from ambiguous matching (e.g. matching A/C and A/G).

See Also

snp_match()

Examples

```
same_ref(ref1 = c("A", "C", "T", "G", NA),
         alt1 = c("C", "T", "C", "A", "A"),
         ref2 = c("A", "C", "A", "A", "C"),
         alt2 = c("C", "G", "G", "G", "A"))
```

SCT *Stacked C+T (SCT)*

Description

Polygenic Risk Scores for a grid of clumping and thresholding parameters.

Stacking over many Polygenic Risk Scores, corresponding to a grid of many different parameters for clumping and thresholding.

Usage

```
snp_grid_clumping(
  G,
  infos.chr,
  infos.pos,
  lpS,
  ind.row = rows_along(G),
  grid.thr.r2 = c(0.01, 0.05, 0.1, 0.2, 0.5, 0.8, 0.95),
  grid.base.size = c(50, 100, 200, 500),
  infos.imp = rep(1, ncol(G)),
  grid.thr.imp = 1,
  groups = list(cols_along(G)),
  exclude = NULL,
  ncores = 1
)

snp_grid_PRS(
  G,
  all_keep,
  betas,
  lpS,
  n_thr_lpS = 50,
  grid.lpS.thr = 0.9999 * seq_log(max(0.1, min(lpS)), max(lpS), n_thr_lpS),
  ind.row = rows_along(G),
  backingfile = tempfile(),
  type = c("float", "double"),
  ncores = 1
)

snp_grid_stacking(
  multi_PRS,
  y.train,
  alphas = c(1, 0.01, 1e-04),
  ncores = 1,
  ...
)
```

Arguments

G	A FBM.code256 (typically <bigSNP>$genotypes). **You shouldn't have missing values.** Also, remember to do quality control, e.g. some algorithms in this package won't work if you use SNPs with 0 MAF.
infos.chr	Vector of integers specifying each SNP's chromosome. Typically <bigSNP>mapchromosome.
infos.pos	Vector of integers specifying the physical position on a chromosome (in base pairs) of each SNP. Typically <bigSNP>mapphysical.pos.

lpS	Numeric vector of -log10(p-value) associated with betas.
ind.row	An optional vector of the row indices (individuals) that are used. If not specified, all rows are used. **Don't use negative indices.**
grid.thr.r2	Grid of thresholds over the squared correlation between two SNPs for clumping. Default is c(0.01,0.05,0.1,0.2,0.5,0.8,0.95).
grid.base.size	Grid for base window sizes. Sizes are then computed as base.size / thr.r2 (in kb). Default is c(50,100,200,500).
infos.imp	Vector of imputation scores. Default is all 1 if you do not provide it.
grid.thr.imp	Grid of thresholds over infos.imp (default is 1), but you should change it (e.g. c(0.3,0.6,0.9,0.95)) if providing infos.imp.
groups	List of vectors of indices to define your own categories. This could be used e.g. to derive C+T scores using two different GWAS summary statistics, or to include other information such as functional annotations. Default just makes one group with all variants.
exclude	Vector of SNP indices to exclude anyway.
ncores	Number of cores used. Default doesn't use parallelism. You may use nb_cores.
all_keep	Output of snp_grid_clumping() (indices passing clumping).
betas	Numeric vector of weights (effect sizes from GWAS) associated with each variant (column of G). If alleles are reversed, make sure to multiply corresponding effects by -1.
n_thr_lpS	Length for default grid.lpS.thr. Default is 50.
grid.lpS.thr	Sequence of thresholds to apply on lpS. Default is a grid (of length n_thr_lpS) evenly spaced on a logarithmic scale, i.e. on a log-log scale for p-values.
backingfile	Prefix for backingfiles where to store scores of C+T. As we typically use a large grid, this can result in a large matrix so that we store it on disk. Default uses a temporary file.
type	Type of backingfile values. Either "float" (the default) or "double". Using "float" requires half disk space.
multi_PRS	Output of snp_grid_PRS().
y.train	Vector of phenotypes. If there are two levels (binary 0/1), it uses big_spLogReg() for stacking, otherwise big_spLinReg().
alphas	Vector of values for grid-search. See big_spLogReg(). Default for this function is c(1,0.01,0.0001).
...	Other parameters to be passed to big_spLogReg(). For example, using covar.train, you can add covariates in the model with all C+T scores. You can also use pf.covar if you do not want to penalize these covariates.

Value

snp_grid_PRS(): An FBM (matrix on disk) that stores the C+T scores for all parameters of the grid (and for each chromosome separately). It also stores as attributes the input parameters all_keep, betas, lpS and grid.lpS.thr that are also needed in snp_grid_stacking().

seq_log *Sequence, evenly spaced on a logarithmic scale*

Description

Sequence, evenly spaced on a logarithmic scale

Usage

```
seq_log(from, to, length.out)
```

Arguments

from	the starting and (maximal) end values of the sequence. Of length 1 unless just from is supplied as an unnamed argument.
to	the starting and (maximal) end values of the sequence. Of length 1 unless just from is supplied as an unnamed argument.
length.out	desired length of the sequence. A non-negative number, which for seq and seq.int will be rounded up if fractional.

Examples

```
seq_log(1, 1000, 4)
seq_log(1, 100, 5)
```

snp_asGeneticPos *Interpolate to genetic positions*

Description

Use genetic maps available at https://github.com/joepickrell/1000-genomes-genetic-maps/ to interpolate physical positions (in bp) to genetic positions (in cM).

Usage

```
snp_asGeneticPos(infos.chr, infos.pos, dir = tempdir(), ncores = 1)
```

Arguments

infos.chr	Vector of integers specifying each SNP's chromosome. Typically <bigSNP>mapchromosome.
infos.pos	Vector of integers specifying the physical position on a chromosome (in base pairs) of each SNP. Typically <bigSNP>mapphysical.pos.
dir	Directory where to download and decompress files. Default is `tempdir()`. Directly use files there if already present.
ncores	Number of cores used. Default doesn't use parallelism. You may use nb_cores.

Value

The new vector of genetic positions.

---

snp_assocBGEN *Compute quick association statistics from BGEN files*

Description

THIS FUNCTION WILL BE MODIFIED IN THE FUTURE.

Usage

```
snp_assocBGEN(
  bgenfiles,
  list_snp_id,
  y_row,
  ind_row,
  bgi_dir = dirname(bgenfiles),
  ncores = 1
)
```

Arguments

bgenfiles	Character vector of paths to files with extension ".bgen". The corresponding ".bgen.bgi" index files must exist.
list_snp_id	List (same length as the number of BGEN files) of character vector of SNP IDs to read. These should be in the form `"<chr>_<pos>_<a1>_<a2>"` (e.g. `"1_88169_C_T"` or `"01_88169_C_T"`). **This function assumes that these IDs are uniquely identifying variants.**
y_row	A vector corresponding to `ind_row` and representing the trait with which to compute correlations. Missing values in either `ind_row` or `y_row` are removed.
ind_row	A vector of the row indices (individuals) that are used. Missing values in either `ind_row` or `y_row` are removed. **Make sure to use indices corresponding to your training set only.**
bgi_dir	Directory of index files. Default is the same as `bgenfiles`.
ncores	Number of cores used. Default doesn't use parallelism. You may use `nb_cores()`.

Value

A list of vectors of log10(p-values) corresponding to the statistic $n \times r^2$, where r is the correlation of each variant with `y_row`.

`snp_attach` *Attach a "bigSNP" from backing files*

Description

Load a bigSNP from backing files into R.

Usage

```
snp_attach(rdsfile)
```

Arguments

`rdsfile`	The path of the ".rds" which stores the `bigSNP` object.

Details

This is often just a call to readRDS. But it also checks if you have moved the two (".bk" and ".rds") backing files to another directory.

Value

The `bigSNP` object.

Examples

```
(bedfile <- system.file("extdata", "example.bed", package = "bigsnpr"))

# Reading the bedfile and storing the data in temporary directory
rds <- snp_readBed(bedfile, backingfile = tempfile())

# Loading the data from backing files
test <- snp_attach(rds)

str(test)
dim(G <- test$genotypes)
G[1:8, 1:8]
```

snp_attachExtdata *Attach a "bigSNP" for examples and tests*

Description

Attach a "bigSNP" for examples and tests

Usage

```
snp_attachExtdata(bedfile = c("example.bed", "example-missing.bed"))
```

Arguments

bedfile	Name of one example bed file. Either • "example.bed" (the default), • "example-missing.bed".

Value

The example "bigSNP", filebacked in the "/tmp/" directory.

snp_autoSVD *Truncated SVD while limiting LD*

Description

Fast truncated SVD with initial pruning and that iteratively removes long-range LD regions.

Usage

```
snp_autoSVD(
  G,
  infos.chr,
  infos.pos = NULL,
  ind.row = rows_along(G),
  ind.col = cols_along(G),
  fun.scaling = snp_scaleBinom(),
  thr.r2 = 0.2,
  size = 100/thr.r2,
  k = 10,
  roll.size = 50,
  int.min.size = 20,
  alpha.tukey = 0.05,
  min.mac = 10,
  max.iter = 5,
```

```
  is.size.in.bp = NULL,
  ncores = 1,
  verbose = TRUE
)

bed_autoSVD(
  obj.bed,
  ind.row = rows_along(obj.bed),
  ind.col = cols_along(obj.bed),
  fun.scaling = bed_scaleBinom,
  thr.r2 = 0.2,
  size = 100/thr.r2,
  k = 10,
  roll.size = 50,
  int.min.size = 20,
  alpha.tukey = 0.05,
  min.mac = 10,
  max.iter = 5,
  ncores = 1,
  verbose = TRUE
)
```

Arguments

`G`	A FBM.code256 (typically <bigSNP>$genotypes). **You shouldn't have missing values.** Also, remember to do quality control, e.g. some algorithms in this package won't work if you use SNPs with 0 MAF.
`infos.chr`	Vector of integers specifying each SNP's chromosome. Typically <bigSNP>mapchromosome.
`infos.pos`	Vector of integers specifying the physical position on a chromosome (in base pairs) of each SNP. Typically <bigSNP>mapphysical.pos.
`ind.row`	An optional vector of the row indices (individuals) that are used. If not specified, all rows are used. **Don't use negative indices.**
`ind.col`	An optional vector of the column indices (SNPs) that are used. If not specified, all columns are used. **Don't use negative indices.**
`fun.scaling`	A function that returns a named list of `mean` and `sd` for every column, to scale each of their elements such as followed: $$\frac{X_{i,j} - mean_j}{sd_j}.$$ Default is `snp_scaleBinom()`.
`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	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.
k	Number of singular vectors/values to compute. Default is 10. **This algorithm should be used to compute a few singular vectors/values.**
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.
is.size.in.bp	Deprecated.
ncores	Number of cores used. Default doesn't use parallelism. You may use nb_cores.
verbose	Output some information on the iterations? Default is TRUE.
obj.bed	Object of type bed, which is the mapping of some bed file. Use obj.bed <-bed(bedfile) to get this object.

Details

If you don't have any information about SNPs, you can try using

- infos.chr = rep(1,ncol(G)),
- size = ncol(G) (if SNPs are not sorted),
- roll.size = 0 (if SNPs are not sorted).

Value

A named list (an S3 class "big_SVD") of

- d, the singular values,
- u, the left singular vectors,
- v, the right singular vectors,
- niter, the number of the iteration of the algorithm,
- nops, number of Matrix-Vector multiplications used,
- center, the centering vector,
- scale, the scaling vector.

Note that to obtain the Principal Components, you must use predict on the result. See examples.

Examples

```
ex <- snp_attachExtdata()

obj.svd <- snp_autoSVD(G = ex$genotypes,
                       infos.chr = ex$map$chromosome,
                       infos.pos = ex$map$physical.position)

str(obj.svd)
```

`snp_beagleImpute` *Imputation*

Description

Imputation using **Beagle** version 4.

Usage

```
snp_beagleImpute(
  beagle.path,
  plink.path,
  bedfile.in,
  bedfile.out = NULL,
  memory.max = 3,
  ncores = 1,
  extra.options = "",
  plink.options = "",
  verbose = TRUE
)
```

Arguments

`beagle.path`	Path to the executable of Beagle v4+.
`plink.path`	Path to the executable of PLINK 1.9.
`bedfile.in`	Path to the input bedfile.
`bedfile.out`	Path to the output bedfile. Default is created by appending `"_impute"` to `prefix.in` (`bedfile.in` without extension).
`memory.max`	Max memory (in GB) to be used. It is internally rounded to be an integer. Default is `3`.
`ncores`	Number of cores used. Default doesn't use parallelism. You may use nb_cores.
`extra.options`	Other options to be passed to Beagle as a string. More options can be found at Beagle's website.
`plink.options`	Other options to be passed to PLINK as a string. More options can be found at `http://www.cog-genomics.org/plink2/filter`.
`verbose`	Whether to show PLINK log? Default is `TRUE`.

Details

Downloads and more information can be found at the following websites

- PLINK,
- Beagle.

Value

The path of the new bedfile.

References

Browning, Brian L., and Sharon R. Browning. "Genotype imputation with millions of reference samples." The American Journal of Human Genetics 98.1 (2016): 116-126.

See Also

download_plink download_beagle

`snp_cor` *Correlation*

Description

Get significant correlations between nearby SNPs of the same chromosome (p-values are computed using a two-sided t-test).

Usage

```
snp_cor(
  Gna,
  ind.row = rows_along(Gna),
  ind.col = cols_along(Gna),
  size = 500,
  alpha = 1,
  fill.diag = TRUE,
  infos.pos = NULL,
  ncores = 1
)
```

Arguments

`Gna`	A FBM.code256 (typically <bigSNP>$genotypes). You can have missing values in these data.
`ind.row`	An optional vector of the row indices (individuals) that are used. If not specified, all rows are used. **Don't use negative indices.**

ind.col	An optional vector of the column indices (SNPs) that are used. If not specified, all columns are used. **Don't use negative indices.**
size	For one SNP, window size around this SNP to compute correlations. Default is 500. 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).
alpha	Type-I error for testing correlations. Default is 1 (no threshold is applied).
fill.diag	Whether to fill the diagonal with 1s (the default) or to keep it as 0s.
infos.pos	Vector of integers specifying the physical position on a chromosome (in base pairs) of each SNP. Typically <bigSNP>mapphysical.pos.
ncores	Number of cores used. Default doesn't use parallelism. You may use nb_cores.

Value

The correlation matrix. This is a sparse symmetric matrix.

Examples

```
test <- snp_attachExtdata()

corr <- snp_cor(test$genotypes, ind.col = 1:1000)
corr[1:10, 1:10]

# Sparsity
length(corr@x) / length(corr)
```

---

snp_fastImpute *Fast imputation*

Description

Fast imputation algorithm based on local XGBoost models.

Usage

```
snp_fastImpute(
  Gna,
  infos.chr,
  alpha = 1e-04,
  size = 200,
  p.train = 0.8,
  n.cor = nrow(Gna),
  seed = NA,
  ncores = 1
)
```

Arguments

Gna	A FBM.code256 (typically <bigSNP>$genotypes). You can have missing values in these data.
infos.chr	Vector of integers specifying each SNP's chromosome. Typically <bigSNP>mapchromosome.
alpha	Type-I error for testing correlations. Default is 1e-4.
size	Number of neighbor SNPs to be possibly included in the model imputing this particular SNP. Default is 200.
p.train	Proportion of non missing genotypes that are used for training the imputation model while the rest is used to assess the accuracy of this imputation model. Default is 0.8.
n.cor	Number of rows that are used to estimate correlations. Default uses them all.
seed	An integer, for reproducibility. Default doesn't use seeds.
ncores	Number of cores used. Default doesn't use parallelism. You may use nb_cores.

Value

An FBM with

- the proportion of missing values by SNP (first row),
- the estimated proportion of imputation errors by SNP (second row).

See Also

snp_fastImputeSimple()

Examples

```
## Not run:

fake <- snp_attachExtdata("example-missing.bed")
G <- fake$genotypes
CHR <- fake$map$chromosome
infos <- snp_fastImpute(G, CHR)
infos[, 1:5]

# Still missing values
big_counts(G, ind.col = 1:10)
# You need to change the code of G
# To make this permanent, you need to save (modify) the file on disk
fake$genotypes$code256 <- CODE_IMPUTE_PRED
fake <- snp_save(fake)
big_counts(fake$genotypes, ind.col = 1:10)

# Plot for post-checking
## Here there is no SNP with more than 1% error (estimated)
pvals <- c(0.01, 0.005, 0.002, 0.001); colvals <- 2:5
df <- data.frame(pNA = infos[1, ], pError = infos[2, ])
```

```
# base R
plot(subset(df, pNA > 0.001), pch = 20)
idc <- lapply(seq_along(pvals), function(i) {
  curve(pvals[i] / x, from = 0, lwd = 2,
        col = colvals[i], add = TRUE)
})
legend("topright", legend = pvals, title = "p(NA & Error)",
       col = colvals, lty = 1, lwd = 2)

# ggplot2
library(ggplot2)
Reduce(function(p, i) {
  p + stat_function(fun = function(x) pvals[i] / x, color = colvals[i])
}, x = seq_along(pvals), init = ggplot(df, aes(pNA, pError))) +
  geom_point() +
  coord_cartesian(ylim = range(df$pError, na.rm = TRUE)) +
  theme_bigstatsr()

## End(Not run)
```

`snp_fastImputeSimple` *Fast imputation*

Description

Fast imputation via mode, mean, sampling according to allele frequencies, or 0.

Usage

```
snp_fastImputeSimple(
  Gna,
  method = c("mode", "mean0", "mean2", "random"),
  ncores = 1
)
```

Arguments

`Gna`	A FBM.code256 (typically <bigSNP>$genotypes). You can have missing values in these data.
`method`	Either `"random"` (sampling according to allele frequencies), `"mean0"` (rounded mean), `"mean2"` (rounded mean to 2 decimal places), `"mode"` (most frequent call).
`ncores`	Number of cores used. Default doesn't use parallelism. You may use nb_cores.

Value

A new `FBM.code256` object (same file, but different code).

See Also

snp_fastImpute()

Examples

```
bigsnp <- snp_attachExtdata("example-missing.bed")
G <- bigsnp$genotypes
G[, 2]  # some missing values
G2 <- snp_fastImputeSimple(G)
G2[, 2]  # no missing values anymore
G[, 2]  # imputed, but still returning missing values
G$copy(code = CODE_IMPUTE_PRED)[, 2]  # need to decode imputed values

G$copy(code = c(0, 1, 2, rep(0, 253)))[, 2]  # "imputation" by 0
```

---

snp_gc *Genomic Control*

Description

Genomic Control

Usage

```
snp_gc(gwas)
```

Arguments

gwas	A `mhtest` object with the p-values associated with each SNP. Typically, the output of big_univLinReg, big_univLogReg or snp_pcadapt.

Value

A `ggplot2` object. You can plot it using the `print` method. You can modify it as you wish by adding layers. You might want to read this chapter to get more familiar with the package **ggplot2**.

References

Devlin, B., & Roeder, K. (1999). Genomic control for association studies. Biometrics, 55(4), 997-1004.

Examples

```
set.seed(9)

test <- snp_attachExtdata()
G <- test$genotypes
y <- rnorm(nrow(G))

gwas <- big_univLinReg(G, y)
snp_qq(gwas)
gwas_gc <- snp_gc(gwas) # change attr(gwas_gc, "transfo")

snp_qq(gwas_gc)
# The next plot should be prettier with a real dataset
snp_manhattan(gwas_gc,
              infos.chr = test$map$chromosome,
              infos.pos = test$map$physical.pos)

p <- snp_qq(gwas_gc) + ggplot2::aes(text = asPlotlyText(test$map))
## Not run: plotly::ggplotly(p, tooltip = "text")
```

---

snp_getSampleInfos *Get sample information*

Description

Get information of individuals by matching from an external file.

Usage

```
snp_getSampleInfos(
  x,
  df.or.files,
  col.family.ID = 1,
  col.sample.ID = 2,
  col.infos = -c(1, 2),
  pair.sep = "-_-",
  ...
)
```

Arguments

x	A bigSNP.
df.or.files	Either • A data.frame, • A character vector of file names where to find at the information you want. You should have one column for family IDs and one for sample IDs.

col.family.ID	Index of the column containing the family IDs to match with those of the study. Default is the second one.
col.sample.ID	Index of the column containing the sample IDs to match with those of the study. Default is the first one.
col.infos	Indices of the column containing the information you want. Default is all but the first and the second columns.
pair.sep	Separator used for concatenation family and sample IDs in order to match easier. Default is "-_-".
...	Any additional parameter to pass to bigreadr::fread2(). Particularly, option header = FALSE is sometimes needed.

Value

The requested information as a data.frame.

See Also

list.files

Examples

```
test <- snp_attachExtdata()
# Just after reading
rle(test$fam$family.ID)
# Get populations clusters from external files
files <- system.file("extdata", paste0("cluster", 1:3), package = "bigsnpr")
bigreadr::fread2(files[1])
# need header option
bigreadr::fread2(files[1], header = FALSE)
infos <- snp_getSampleInfos(test, files, header = FALSE)
rle(infos[[1]])
```

---

snp_ldpred2_inf *LDpred2*

Description

LDpred2. Tutorial at https://bit.ly/ldpred2.

Usage

```
snp_ldpred2_inf(corr, df_beta, h2)

snp_ldpred2_grid(
  corr,
  df_beta,
```

```
  grid_param,
  burn_in = 50,
  num_iter = 100,
  ncores = 1
)

snp_ldpred2_auto(
  corr,
  df_beta,
  h2_init,
  vec_p_init = 0.1,
  burn_in = 1000,
  num_iter = 500,
  verbose = FALSE,
  ncores = 1
)
```

Arguments

`corr`	Sparse correlation matrix as an SFBM. If `corr` is a dgSMatrix, you can use `bigsparser::as_SFBM(as(corr,"dgCMatrix"))`.
`df_beta`	A data frame with 3 columns: • $beta: effect size estimates • $beta_se: standard errors of effect size estimates • $n_eff: sample size when estimating `beta` (in the case of binary traits, this is `4 / (1 / n_control + 1 / n_case)`)
`h2`	Heritability estimate.
`grid_param`	A data frame with 3 columns as a grid of hyper-parameters: • $p: proportion of causal variants • $h2: heritability (captured by the variants used) • $sparse: boolean, whether a sparse model is sought They can be run in parallel by changing `ncores`.
`burn_in`	Number of burn-in iterations.
`num_iter`	Number of iterations after burn-in.
`ncores`	Number of cores used. Default doesn't use parallelism. You may use nb_cores.
`h2_init`	Heritability estimate for initialization.
`vec_p_init`	Vector of initial values for p. Default is `0.1`.
`verbose`	Whether to print "p // h2" estimates at each iteration.

Value

`snp_ldpred2_inf`: A vector of effects, assuming an infinitesimal model.

`snp_ldpred2_grid`: A matrix of effect sizes, one vector (column) for each row of `grid_param`.

`snp_ldpred2_auto`: A list (over `vec_p_init`) of lists with

- $beta_est: vector of effect sizes
- $p_est: estimate of p, the proportion of causal variants
- $h2_est: estimate of the (SNP) heritability (also see coef_to_liab)
- $path_p_est: full path of p estimates (including burn-in); useful to check convergence of the iterative algorithm
- $path_h2_est: full path of h2 estimates (including burn-in); useful to check convergence of the iterative algorithm

---

`snp_ldsc` *LD score regression*

Description

LD score regression

Usage

```
snp_ldsc(
  ld_score,
  ld_size,
  chi2,
  sample_size,
  blocks = 200,
  intercept = NULL,
  chi2_thr1 = 30,
  chi2_thr2 = Inf,
  ncores = 1
)

snp_ldsc2(corr, df_beta, blocks = NULL, intercept = 1, ...)
```

Arguments

`ld_score`	Vector of LD scores.
`ld_size`	Number of variants used to compute `ld_score`.
`chi2`	Vector of chi-squared statistics.
`sample_size`	Sample size of GWAS corresponding to chi-squared statistics. Possibly a vector, or just a single value.
`blocks`	Either a single number specifying the number of blocks, or a vector of integers specifying the block number of each `chi2` value. Default is `200` for `snp_ldsc()`, dividing into 200 blocks of approximately equal size. `NULL` can also be used to skip estimating standard errors, which is the default for `snp_ldsc2()`.
`intercept`	You can constrain the intercept to some value (e.g. 1). Default is `NULL` in `snp_ldsc()` (the intercept is estimated) and is `1` in `snp_ldsc2()` (the intercept is fixed to 1). This is equivalent to parameter `--intercept-h2`.

chi2_thr1	Threshold on chi2 in step 1. Default is 30. This is equivalent to parameter --two-step.
chi2_thr2	Threshold on chi2 in step 2. Default is Inf (none).
ncores	Number of cores used. Default doesn't use parallelism. You may use nb_cores.
corr	Sparse correlation matrix.
df_beta	A data frame with 3 columns: • $beta: effect size estimates • $beta_se: standard errors of effect size estimates • $n_eff: sample size when estimating beta (in the case of binary traits, this is 4 / (1 / n_control + 1 / n_case))
...	Arguments passed on to snp_ldsc

Value

Vector of 4 values (or only the first 2 if blocks = NULL):

- [["int"]]: LDSC regression intercept,
- [["int_se"]]: SE of this intercept,
- [["h2"]]: LDSC regression estimate of (SNP) heritability (also see coef_to_liab),
- [["h2_se"]]: SE of this heritability estimate.

Examples

```
bigsnp <- snp_attachExtdata()
G <- bigsnp$genotypes
y <- bigsnp$fam$affection - 1
corr <- snp_cor(G, ind.col = 1:1000)

gwas <- big_univLogReg(G, y, ind.col = 1:1000)
df_beta <- data.frame(beta = gwas$estim, beta_se = gwas$std.err,
                      n_eff = 4 / (1 / sum(y == 0) + 1 / sum(y == 1)))

snp_ldsc2(corr, df_beta)
snp_ldsc2(corr, df_beta, blocks = 20, intercept = NULL)
```

---

snp_MAF *MAF*

Description

Minor Allele Frequency.

Usage

```
snp_MAF(
  G,
  ind.row = rows_along(G),
  ind.col = cols_along(G),
  nploidy = 2,
  ncores = 1
)
```

Arguments

G	A FBM.code256 (typically <bigSNP>$genotypes). **You shouldn't have missing values.** Also, remember to do quality control, e.g. some algorithms in this package won't work if you use SNPs with 0 MAF.
ind.row	An optional vector of the row indices (individuals) that are used. If not specified, all rows are used. **Don't use negative indices.**
ind.col	An optional vector of the column indices (SNPs) that are used. If not specified, all columns are used. **Don't use negative indices.**
nploidy	Number of trials, parameter of the binomial distribution. Default is 2, which corresponds to diploidy, such as for the human genome.
ncores	Number of cores used. Default doesn't use parallelism. You may use nb_cores.

Value

A vector of MAFs, corresponding to ind.col.

Examples

```
obj.bigsnp <- snp_attachExtdata()
str(maf <- snp_MAF(obj.bigsnp$genotypes))
```

---

snp_manhattan *Manhattan plot*

Description

Creates a manhattan plot.

Usage

```
snp_manhattan(
  gwas,
  infos.chr,
  infos.pos,
  colors = c("black", "grey60"),
  dist.sep.chrs = 1e+07,
  ind.highlight = integer(0),
  col.highlight = "red",
  labels = NULL,
  npoints = NULL,
  coeff = 1
)
```

Arguments

gwas	A mhtest object with the p-values associated with each SNP. Typically, the output of big_univLinReg, big_univLogReg or snp_pcadapt.
infos.chr	Vector of integers specifying each SNP's chromosome. Typically <bigSNP>mapchromosome.
infos.pos	Vector of integers specifying the physical position on a chromosome (in base pairs) of each SNP. Typically <bigSNP>mapphysical.pos.
colors	Colors used for each chromosome (they are recycled). Default is an alternation of black and gray.
dist.sep.chrs	"Physical" distance that separates two chromosomes. Default is 10 Mbp.
ind.highlight	Indices of SNPs you want to highlight (of interest). Default doesn't highlight any SNPs.
col.highlight	Color used for highlighting SNPs. Default uses red.
labels	Labels of the x axis. Default uses the number of the chromosome there are in infos.chr(sort(unique(infos.chr))). This may be useful to restrict the number of labels so that they are not overlapping.
npoints	Number of points to keep (ranked by p-value) in order to get a lighter object (and plot). Default doesn't cut anything. If used, the resulting object will have an attribute called subset giving the indices of the kept points.
coeff	Relative size of text. Default is 1.

Details

If you don't have information of chromosome and position, you should simply use plot instead.

Value

A ggplot2 object. You can plot it using the print method. You can modify it as you wish by adding layers. You might want to read this chapter to get more familiar with the package **ggplot2**.

Examples

```
set.seed(9)

test <- snp_attachExtdata()
G <- test$genotypes
y <- rnorm(nrow(G))

gwas <- big_univLinReg(G, y)
snp_qq(gwas)
gwas_gc <- snp_gc(gwas) # change attr(gwas_gc, "transfo")

snp_qq(gwas_gc)
# The next plot should be prettier with a real dataset
snp_manhattan(gwas_gc,
              infos.chr = test$map$chromosome,
              infos.pos = test$map$physical.pos)

p <- snp_qq(gwas_gc) + ggplot2::aes(text = asPlotlyText(test$map))
## Not run: plotly::ggplotly(p, tooltip = "text")
```

---

snp_match *Match alleles*

Description

Match alleles between summary statistics and SNP information. Match by ("chr", "a0", "a1") and ("pos" or "rsid"), accounting for possible strand flips and reverse reference alleles (opposite effects).

Usage

```
snp_match(
  sumstats,
  info_snp,
  strand_flip = TRUE,
  join_by_pos = TRUE,
  match.min.prop = 0.5
)
```

Arguments

sumstats	A data frame with columns "chr", "pos", "a0", "a1" and "beta".
info_snp	A data frame with columns "chr", "pos", "a0" and "a1".
strand_flip	Whether to try to flip strand? (default is TRUE) If so, ambiguous alleles A/T and C/G are removed.
join_by_pos	Whether to join by chromosome and position (default), or instead by rsid.
match.min.prop	Minimum proportion of variants in the smallest data to be matched, otherwise stops with an error. Default is 50%.

Value

A single data frame with matched variants.

See Also

snp_modifyBuild

Examples

```
sumstats <- data.frame(
  chr = 1,
  pos = c(86303, 86331, 162463, 752566, 755890, 758144),
  a0 = c("T", "G", "C", "A", "T", "G"),
  a1 = c("G", "A", "T", "G", "A", "A"),
  beta = c(-1.868, 0.250, -0.671, 2.112, 0.239, 1.272),
  p = c(0.860, 0.346, 0.900, 0.456, 0.776, 0.383)
)

info_snp <- data.frame(
  id = c("rs2949417", "rs115209712", "rs143399298", "rs3094315", "rs3115858"),
  chr = 1,
  pos = c(86303, 86331, 162463, 752566, 755890),
  a0 = c("T", "A", "G", "A", "T"),
  a1 = c("G", "G", "A", "G", "A")
)

snp_match(sumstats, info_snp)
snp_match(sumstats, info_snp, strand_flip = FALSE)
```

---

snp_MAX3 *MAX3 statistic*

Description

Compute the MAX3 statistic, which tests for three genetic models (additive, recessive and dominant).

Usage

```
snp_MAX3(Gna, y01.train, ind.train = rows_along(Gna), val = c(0, 0.5, 1))
```

Arguments

Gna	A FBM.code256 (typically <bigSNP>$genotypes). You can have missing values in these data.
y01.train	Vector of responses, corresponding to ind.train. **Must be only 0s and 1s.**
ind.train	An optional vector of the row indices that are used, for the training part. If not specified, all rows are used. **Don't use negative indices.**

`val` Computing $\max_{x \in val} Z^2_{CATT}(x)$.

- Default is `c(0,0.5,1)` and corresponds to the *MAX3* statistic.
- Only `c(0,1)` corresponds to *MAX2*.
- And only `0.5` corresponds to the Armitage trend test.
- Finally, `seq(0,1,length.out = L)` corresponds to *MAXL*.

Details

P-values associated with returned scores are in fact the minimum of the p-values of each test separately. Thus, they are biased downward.

Value

An object of classes `mhtest` and `data.frame` returning one score by SNP. See `methods(class = "mhtest")`.

References

Zheng, G., Yang, Y., Zhu, X., & Elston, R. (2012). Robust Procedures. Analysis Of Genetic Association Studies, 151-206. http://dx.doi.org/10.1007/978-1-4614-2245-7_6.

Examples

```
set.seed(1)

# constructing a fake genotype big.matrix
N <- 50; M <- 1200
fake <- snp_fake(N, M)
G <- fake$genotypes
G[] <- sample(as.raw(0:3), size = length(G), replace = TRUE)
G[1:8, 1:10]

# Specify case/control phenotypes
fake$fam$affection <- rep(1:2, each = N / 2)

# Get MAX3 statistics
y01 <- fake$fam$affection - 1
str(test <- snp_MAX3(fake$genotypes, y01.train = y01))
# p-values are not well calibrated
snp_qq(test)
# genomic control is not of much help
snp_qq(snp_gc(test))

# Armitage trend test (well calibrated because only one test)
test2 <- snp_MAX3(fake$genotypes, y01.train = y01, val = 0.5)
snp_qq(test2)
```

snp_modifyBuild *Modify genome build*

Description

Modify the physical position information of a data frame when converting genome build using executable *liftOver*.

Usage

```
snp_modifyBuild(info_snp, liftOver, from = "hg18", to = "hg19")
```

Arguments

info_snp	A data frame with columns "chr" and "pos".
liftOver	Path to liftOver executable. Binaries can be downloaded at https://bit.ly/2KvHugi for Mac and at https://bit.ly/2TbSaEI for Linux.
from	Genome build to convert from. Default is hg18.
to	Genome build to convert to. Default is hg19.

Value

Input data frame info_snp with column "pos" in the new build.

References

Hinrichs, Angela S., et al. "The UCSC genome browser database: update 2006." Nucleic acids research 34.suppl_1 (2006): D590-D598.

snp_pcadapt *Outlier detection*

Description

Method to detect genetic markers involved in biological adaptation. This provides a statistical tool for outlier detection based on Principal Component Analysis. This corresponds to the statistic based on mahalanobis distance, as implemented in package **pcadapt**.

Usage

```
snp_pcadapt(
  G,
  U.row,
  ind.row = rows_along(G),
  ind.col = cols_along(G),
  ncores = 1
)

bed_pcadapt(
  obj.bed,
  U.row,
  ind.row = rows_along(obj.bed),
  ind.col = cols_along(obj.bed),
  ncores = 1
)
```

Arguments

G	A FBM.code256 (typically <bigSNP>$genotypes). **You shouldn't have missing values.** Also, remember to do quality control, e.g. some algorithms in this package won't work if you use SNPs with 0 MAF.
U.row	Left singular vectors (not scores, $U^T U = I$) corresponding to `ind.row`.
ind.row	An optional vector of the row indices (individuals) that are used. If not specified, all rows are used. **Don't use negative indices.**
ind.col	An optional vector of the column indices (SNPs) that are used. If not specified, all columns are used. **Don't use negative indices.**
ncores	Number of cores used. Default doesn't use parallelism. You may use nb_cores.
obj.bed	Object of type bed, which is the mapping of some bed file. Use `obj.bed <-bed(bedfile)` to get this object.

Value

An object of classes `mhtest` and `data.frame` returning one score by SNP. See `methods(class = "mhtest")`.

References

Luu, K., Bazin, E., & Blum, M. G. (2017). pcadapt: an R package to perform genome scans for selection based on principal component analysis. Molecular ecology resources, 17(1), 67-77.

See Also

snp_manhattan, snp_qq and snp_gc.

Examples

```
test <- snp_attachExtdata()
G <- test$genotypes
obj.svd <- big_SVD(G, fun.scaling = snp_scaleBinom(), k = 10)
plot(obj.svd) # there seems to be 3 "significant" components
pcadapt <- snp_pcadapt(G, obj.svd$u[, 1:3])
snp_qq(pcadapt)
```

---

snp_plinkIBDQC *Identity-by-descent*

Description

Quality Control based on Identity-by-descent (IBD) computed by **PLINK 1.9** using its method-of-moments.

Usage

```
snp_plinkIBDQC(
  plink.path,
  bedfile.in,
  bedfile.out = NULL,
  pi.hat = 0.08,
  ncores = 1,
  pruning.args = c(100, 0.2),
  do.blind.QC = TRUE,
  extra.options = "",
  verbose = TRUE
)
```

Arguments

`plink.path`	Path to the executable of PLINK 1.9.
`bedfile.in`	Path to the input bedfile.
`bedfile.out`	Path to the output bedfile. Default is created by appending `"_norel"` to `prefix.in` (`bedfile.in` without extension).
`pi.hat`	PI_HAT value threshold for individuals (first by pairs) to be excluded. Default is `0.08`.
`ncores`	Number of cores used. Default doesn't use parallelism. You may use nb_cores.
`pruning.args`	A vector of 2 pruning parameters, respectively the window size (in variant count) and the pairwise r^2 threshold (the step size is fixed to 1). Default is `c(100,0.2)`.

do.blind.QC	Whether to do QC with pi.hat without visual inspection. Default is TRUE. If FALSE, return the data.frame of the corresponding ".genome" file without doing QC. One could use ggplot2::qplot(Z0,Z1,data = mydf,col = RT) for visual inspection.
extra.options	Other options to be passed to PLINK as a string (for the IBD part). More options can be found at http://www.cog-genomics.org/plink/1.9/ibd.
verbose	Whether to show PLINK log? Default is TRUE.

Value

The path of the new bedfile. If no sample is filter, no new bed/bim/fam files are created and then the path of the input bedfile is returned.

References

Chang, Christopher C, Carson C Chow, Laurent CAM Tellier, Shashaank Vattikuti, Shaun M Purcell, and James J Lee. 2015. *Second-generation PLINK: rising to the challenge of larger and richer datasets.* GigaScience 4 (1): 7. http://dx.doi.org/10.1186/s13742-015-0047-8.

See Also

download_plink snp_plinkQC snp_plinkKINGQC

Examples

```
## Not run:

bedfile <- system.file("extdata", "example.bed", package = "bigsnpr")
plink <- download_plink()

bedfile <- snp_plinkIBDQC(plink, bedfile,
                          bedfile.out = tempfile(fileext = ".bed"),
                          ncores = 2)

df_rel <- snp_plinkIBDQC(plink, bedfile, do.blind.QC = FALSE, ncores = 2)
str(df_rel)

library(ggplot2)
qplot(Z0, Z1, data = df_rel, col = RT)
qplot(y = PI_HAT, data = df_rel) +
  geom_hline(yintercept = 0.2, color = "blue", linetype = 2)
snp_plinkRmSamples(plink, bedfile,
                   bedfile.out = tempfile(fileext = ".bed"),
                   df.or.files = subset(df_rel, PI_HAT > 0.2))

## End(Not run)
```

snp_plinkKINGQC *Relationship-based pruning*

Description

Quality Control based on KING-robust kinship estimator. More information can be found at http://www.cog-genomics.org/plink/2.0/distance#king_cutoff.

Usage

```
snp_plinkKINGQC(
  plink2.path,
  bedfile.in,
  bedfile.out = NULL,
  thr.king = 2^-3.5,
  make.bed = TRUE,
  ncores = 1,
  extra.options = "",
  verbose = TRUE
)
```

Arguments

plink2.path	Path to the executable of PLINK 2.
bedfile.in	Path to the input bedfile.
bedfile.out	Path to the output bedfile. Default is created by appending "_norel" to prefix.in (bedfile.in without extension).
thr.king	Note that KING kinship coefficients are scaled such that duplicate samples have kinship 0.5, not 1. First-degree relations (parent-child, full siblings) correspond to ~0.25, second-degree relations correspond to ~0.125, etc. It is conventional to use a cutoff of ~0.354 (2^-1.5, the geometric mean of 0.5 and 0.25) to screen for monozygotic twins and duplicate samples, ~0.177 (2^-2.5) to remove first-degree relations as well, and ~0.0884 (2^-3.5, **default**) to remove second-degree relations as well, etc.
make.bed	Whether to create new bed/bim/fam files (default). Otherwise, returns a table with coefficients of related pairs.
ncores	Number of cores used. Default doesn't use parallelism. You may use nb_cores.
extra.options	Other options to be passed to PLINK2 as a string.
verbose	Whether to show PLINK log? Default is TRUE.

Value

See parameter make-bed.

References

Manichaikul, Ani, Josyf C. Mychaleckyj, Stephen S. Rich, Kathy Daly, Michele Sale, and Wei-Min Chen. "Robust relationship inference in genome-wide association studies." Bioinformatics 26, no. 22 (2010): 2867-2873.

See Also

download_plink2 snp_plinkQC

Examples

```
## Not run:

bedfile <- system.file("extdata", "example.bed", package = "bigsnpr")
plink2 <- download_plink2(AVX2 = FALSE)

bedfile2 <- snp_plinkKINGQC(plink2, bedfile,
                            bedfile.out = tempfile(fileext = ".bed"),
                            ncores = 2)

df_rel <- snp_plinkKINGQC(plink2, bedfile, make.bed = FALSE, ncores = 2)
str(df_rel)

## End(Not run)
```

---

snp_plinkQC *Quality Control*

Description

Quality Control (QC) and possible conversion to *bed/bim/fam* files using **PLINK 1.9**.

Usage

```
snp_plinkQC(
  plink.path,
  prefix.in,
  file.type = "--bfile",
  prefix.out = paste0(prefix.in, "_QC"),
  maf = 0.01,
  geno = 0.1,
  mind = 0.1,
  hwe = 1e-50,
  autosome.only = FALSE,
  extra.options = "",
  verbose = TRUE
)
```

Arguments

plink.path	Path to the executable of PLINK 1.9.
prefix.in	Prefix (path without extension) of the dataset to be QCed.
file.type	Type of the dataset to be QCed. Default is "--bfile" and corresponds to bed/bim/fam files. You can also use "--file" for ped/map files or "--vcf" for a VCF file. More information can be found at http://www.cog-genomics.org/plink/1.9/input.
prefix.out	Prefix (path without extension) of the bed/bim/fam dataset to be created. Default is created by appending "_QC" to prefix.in.
maf	Minimum Minor Allele Frequency (MAF) for a SNP to be kept. Default is 0.01.
geno	Maximum proportion of missing values for a SNP to be kept. Default is 0.1.
mind	Maximum proportion of missing values for a sample to be kept. Default is 0.1.
hwe	Filters out all variants which have Hardy-Weinberg equilibrium exact test p-value below the provided threshold. Default is 1e-50.
autosome.only	Whether to exclude all unplaced and non-autosomal variants? Default is FALSE.
extra.options	Other options to be passed to PLINK as a string. More options can be found at http://www.cog-genomics.org/plink2/filter. If using PLINK 2.0, you could e.g. use "--king-cutoff 0.0884" to remove some related samples at the same time of quality controls.
verbose	Whether to show PLINK log? Default is TRUE.

Value

The path of the newly created bedfile.

References

Chang, Christopher C, Carson C Chow, Laurent CAM Tellier, Shashaank Vattikuti, Shaun M Purcell, and James J Lee. 2015. *Second-generation PLINK: rising to the challenge of larger and richer datasets.* GigaScience 4 (1): 7. http://dx.doi.org/10.1186/s13742-015-0047-8.

See Also

download_plink snp_plinkIBDQC

Examples

```
## Not run:

bedfile <- system.file("extdata", "example.bed", package = "bigsnpr")
prefix  <- sub_bed(bedfile)
plink <- download_plink()
test <- snp_plinkQC(plink.path = plink,
                    prefix.in = prefix,
                    prefix.out = tempfile(),
                    file.type = "--bfile",  # the default (for ".bed")
```

```
                    maf = 0.05,
                    geno = 0.05,
                    mind = 0.05,
                    hwe = 1e-10,
                    autosome.only = TRUE)
test

## End(Not run)
```

`snp_plinkRmSamples` *Remove samples*

Description

Create new *bed*/*bim*/*fam* files by removing samples with PLINK.

Usage

```
snp_plinkRmSamples(
  plink.path,
  bedfile.in,
  bedfile.out,
  df.or.files,
  col.family.ID = 1,
  col.sample.ID = 2,
  ...,
  verbose = TRUE
)
```

Arguments

`plink.path`	Path to the executable of PLINK 1.9.
`bedfile.in`	Path to the input bedfile.
`bedfile.out`	Path to the output bedfile.
`df.or.files`	Either • A `data.frame`, • A character vector of file names where to find at the information you want. You should have one column for family IDs and one for sample IDs.
`col.family.ID`	Index of the column containing the family IDs to match with those of the study. Default is the second one.
`col.sample.ID`	Index of the column containing the sample IDs to match with those of the study. Default is the first one.
`...`	Any additional parameter to pass to `bigreadr::fread2()`. Particularly, option `header = FALSE` is sometimes needed.
`verbose`	Whether to show PLINK log? Default is `TRUE`.

Value

The path of the new bedfile.

See Also

download_plink

---

snp_PRS *PRS*

Description

Polygenic Risk Scores with possible clumping and thresholding.

Usage

```
snp_PRS(
  G,
  betas.keep,
  ind.test = rows_along(G),
  ind.keep = cols_along(G),
  same.keep = rep(TRUE, length(ind.keep)),
  lpS.keep = NULL,
  thr.list = 0
)
```

Arguments

G	A FBM.code256 (typically <bigSNP>$genotypes). **You shouldn't have missing values.** Also, remember to do quality control, e.g. some algorithms in this package won't work if you use SNPs with 0 MAF.
betas.keep	Numeric vector of weights associated with each SNP corresponding to ind.keep. You may want to see big_univLinReg or big_univLogReg.
ind.test	The individuals on whom to project the scores. Default uses all.
ind.keep	Column (SNP) indices to use (if using clumping, the output of snp_clumping). Default doesn't clump.
same.keep	A logical vector associated with betas.keep whether the reference allele is the same for G. Default is all TRUE (for example when you train the betas on the same dataset). Otherwise, use same_ref.
lpS.keep	Numeric vector of -log10(p-value) associated with betas.keep. Default doesn't use thresholding.
thr.list	Threshold vector on lpS.keep at which SNPs are excluded if they are not significant enough. Default doesn't use thresholding.

Value

A matrix of scores, where rows correspond to `ind.test` and columns correspond to `thr.list`.

Examples

```
test <- snp_attachExtdata()
G <- big_copy(test$genotypes, ind.col = 1:1000)
CHR <- test$map$chromosome[1:1000]
POS <- test$map$physical.position[1:1000]
y01 <- test$fam$affection - 1

# PCA -> covariables
obj.svd <- snp_autoSVD(G, infos.chr = CHR, infos.pos = POS)

# train and test set
ind.train <- sort(sample(nrow(G), 400))
ind.test <- setdiff(rows_along(G), ind.train) # 117

# GWAS
gwas.train <- big_univLogReg(G, y01.train = y01[ind.train],
                             ind.train = ind.train,
                             covar.train = obj.svd$u[ind.train, ])
# clumping
ind.keep <- snp_clumping(G, infos.chr = CHR,
                         ind.row = ind.train,
                         S = abs(gwas.train$score))
# -log10(p-values) and thresolding
summary(lpS.keep <- -predict(gwas.train)[ind.keep])
thrs <- seq(0, 4, by = 0.5)
nb.pred <- sapply(thrs, function(thr) sum(lpS.keep > thr))

# PRS
prs <- snp_PRS(G, betas.keep = gwas.train$estim[ind.keep],
               ind.test = ind.test,
               ind.keep = ind.keep,
               lpS.keep = lpS.keep,
               thr.list = thrs)

# AUC as a function of the number of predictors
aucs <- apply(prs, 2, AUC, target = y01[ind.test])
library(ggplot2)
qplot(nb.pred, aucs) +
  geom_line() +
  scale_x_log10(breaks = nb.pred) +
  labs(x = "Number of predictors", y = "AUC") +
  theme_bigstatsr()
```

---

snp_qq *Q-Q plot*

Description

Creates a quantile-quantile plot from p-values from a GWAS study.

Usage

```
snp_qq(gwas, lambdaGC = TRUE, coeff = 1)
```

Arguments

gwas	A mhtest object with the p-values associated with each SNP. Typically, the output of big_univLinReg, big_univLogReg or snp_pcadapt.
lambdaGC	Add the Genomic Control coefficient as subtitle to the plot?
coeff	Relative size of text. Default is 1.

Value

A ggplot2 object. You can plot it using the print method. You can modify it as you wish by adding layers. You might want to read this chapter to get more familiar with the package **ggplot2**.

Examples

```
set.seed(9)

test <- snp_attachExtdata()
G <- test$genotypes
y <- rnorm(nrow(G))

gwas <- big_univLinReg(G, y)
snp_qq(gwas)
gwas_gc <- snp_gc(gwas) # change attr(gwas_gc, "transfo")

snp_qq(gwas_gc)
# The next plot should be prettier with a real dataset
snp_manhattan(gwas_gc,
              infos.chr = test$map$chromosome,
              infos.pos = test$map$physical.pos)

p <- snp_qq(gwas_gc) + ggplot2::aes(text = asPlotlyText(test$map))
## Not run: plotly::ggplotly(p, tooltip = "text")
```

---

snp_readBed *Read PLINK files into a "bigSNP"*

Description

Functions to read bed/bim/fam files into a bigSNP.

Usage

```
snp_readBed(bedfile, backingfile = sub_bed(bedfile))

snp_readBed2(
  bedfile,
  backingfile = sub_bed(bedfile),
  ind.row = rows_along(obj.bed),
  ind.col = cols_along(obj.bed),
  ncores = 1
)
```

Arguments

bedfile	Path to file with extension ".bed" to read. You need the corresponding ".bim" and ".fam" in the same directory.
backingfile	The path (without extension) for the backing files for the cache of the bigSNP object. Default takes the bedfile without the ".bed" extension.
ind.row	An optional vector of the row indices (individuals) that are used. If not specified, all rows are used. **Don't use negative indices.**
ind.col	An optional vector of the column indices (SNPs) that are used. If not specified, all columns are used. **Don't use negative indices.**
ncores	Number of cores used. Default doesn't use parallelism. You may use nb_cores.

Details

For more information on these formats, please visit PLINK webpage. For other formats, please use PLINK to convert them in bedfiles, which require minimal space to store and are faster to read. For example, to convert from a VCF file, use the `--vcf` option. See snp_plinkQC.

Value

The path to the RDS file that stores the `bigSNP` object. Note that this function creates one other file which stores the values of the Filebacked Big Matrix.
You shouldn't read from PLINK files more than once. Instead, use snp_attach to load the "bigSNP" object in any R session from backing files.

Examples

```
(bedfile <- system.file("extdata", "example.bed", package = "bigsnpr"))

# Reading the bedfile and storing the data in temporary directory
rds <- snp_readBed(bedfile, backingfile = tempfile())

# Loading the data from backing files
test <- snp_attach(rds)
```

```
str(test)
dim(G <- test$genotypes)
G[1:8, 1:8]
```

`snp_readBGEN` *Read BGEN files into a "bigSNP"*

Description

Function to read the UK Biobank BGEN files into a bigSNP.

Usage

```
snp_readBGEN(
  bgenfiles,
  backingfile,
  list_snp_id,
  ind_row = NULL,
  bgi_dir = dirname(bgenfiles),
  read_as = c("dosage", "random"),
  ncores = 1
)
```

Arguments

`bgenfiles`	Character vector of paths to files with extension ".bgen". The corresponding ".bgen.bgi" index files must exist.
`backingfile`	The path (without extension) for the backing files for the cache of the bigSNP object.
`list_snp_id`	List (same length as the number of BGEN files) of character vector of SNP IDs to read. These should be in the form `"<chr>_<pos>_<a1>_<a2>"` (e.g. `"1_88169_C_T"` or `"01_88169_C_T"`). **This function assumes that these IDs are uniquely identifying variants.**
`ind_row`	An optional vector of the row indices (individuals) that are used. If not specified, all rows are used. **Don't use negative indices.**
`bgi_dir`	Directory of index files. Default is the same as `bgenfiles`.
`read_as`	How to read BGEN probabilities? Currently implemented: • as dosages (rounded to two decimal places), the default, • as hard calls, randomly sampled based on those probabilities (similar to PLINK option '--hard-call-threshold random').
`ncores`	Number of cores used. Default doesn't use parallelism. You may use `nb_cores()`.

Details

For more information on this format, please visit BGEN webpage.

This function is designed to read UK Biobank imputation files. This assumes that variants have been compressed with zlib, that there are only 2 possible alleles, and that each probability is stored on 8 bits.

Value

The path to the RDS file that stores the `bigSNP` object. Note that this function creates one other file which stores the values of the Filebacked Big Matrix.
You shouldn't read from BGEN files more than once. Instead, use snp_attach to load the "bigSNP" object in any R session from backing files.

Examples

```
# See e.g. https://github.com/privefl/UKBiobank/blob/master/10-get-dosages.R
```

`snp_readBGI` *Read variant info from one BGI file*

Description

Read variant info from one BGI file

Usage

```
snp_readBGI(bgifile, snp_id)
```

Arguments

`bgifile`	Path to one file with extension ".bgi".
`snp_id`	Character vector of SNP IDs. These should be in the form `"<chr>_<pos>_<a1>_<a2>"` (e.g. `"1_88169_C_T"` or `"01_88169_C_T"`). **This function assumes that these IDs are uniquely identifying variants.**

Value

A data frame containing variant information.

snp_save *Save modifications*

Description

Save a bigSNP after having made some modifications to it. As bigSNP is an S3 class, you can add any slot you want to an object of this class, then use snp_save to save these modifications in the corresponding ".rds" backing file.

Usage

```
snp_save(x, version = NULL)
```

Arguments

x	A bigSNP.
version	the workspace format version to use. NULL specifies the current default version (3). The only other supported value is 2, the default from R 1.4.0 to R 3.5.0.

Value

The (saved) bigSNP.

Examples

```
set.seed(1)

# Reading example
test <- snp_attachExtdata()

# I can add whatever I want to an S3 class
test$map$`p-values` <- runif(nrow(test$map))
str(test$map)

# Reading again
test.savedIn <- sub_bk(test$genotypes$backingfile, ".rds")
test2 <- snp_attach(rdsfile = test.savedIn)
str(test2$map) # new slot wasn't saved

# Save it
test <- snp_save(test)

# Reading again
test3 <- snp_attach(rdsfile = test.savedIn)
str(test3$map) # it is saved now

# The complicated code of this function
snp_save
```

`snp_scaleBinom` *Binomial(n, p) scaling*

Description

Binomial(n, p) scaling where `n` is fixed and `p` is estimated.

Usage

```
snp_scaleBinom(nploidy = 2)
```

Arguments

`nploidy`	Number of trials, parameter of the binomial distribution. Default is `2`, which corresponds to diploidy, such as for the human genome.

Details

You will probably not use this function as is but as the `fun.scaling` parameter of other functions of package `bigstatsr`.

Value

A new **function** that returns a data.frame of two vectors "center" and "scale" which are of the length of `ind.col`.

References

This scaling is widely used for SNP arrays. Patterson N, Price AL, Reich D (2006). Population Structure and Eigenanalysis. PLoS Genet 2(12): e190. http://dx.doi.org/10.1371/journal.pgen.0020190.

Examples

```
set.seed(1)

a <- matrix(0, 93, 170)
p <- 0.2
a[] <- rbinom(length(a), 2, p)
X <- add_code256(big_copy(a, type = "raw"), code = c(0, 1, 2, rep(NA, 253)))
X.svd <- big_SVD(X, fun.scaling = snp_scaleBinom())
str(X.svd)
plot(X.svd$center)
abline(h = 2 * p, col = "red")
plot(X.svd$scale)
abline(h = sqrt(2 * p * (1 - p)), col = "red")
```

`snp_simuPheno` *Simulate phenotypes*

Description

Simulate phenotypes using a linear model. When a prevalence is given, the liability threshold is used to convert liabilities to a binary outcome. The genetic and environmental liabilities are scaled such that the variance of the genetic liability is equality the requested heritability, and the variance of the total liability is 1.

Usage

```
snp_simuPheno(
  G,
  h2,
  M,
  K = NULL,
  ind.row = rows_along(G),
  ind.possible = cols_along(G),
  effects.dist = c("gaussian", "laplace"),
  ncores = 1
)
```

Arguments

`G`	A FBM.code256 (typically <bigSNP>$genotypes). **You shouldn't have missing values.** Also, remember to do quality control, e.g. some algorithms in this package won't work if you use SNPs with 0 MAF.
`h2`	Heritability.
`M`	Number of causal variants.
`K`	Prevalence. Default is `NULL`, giving a continuous trait.
`ind.row`	An optional vector of the row indices (individuals) that are used. If not specified, all rows are used. **Don't use negative indices.**
`ind.possible`	Indices of possible causal variants.
`effects.dist`	Distribution of effects. Either `"gaussian"` (the default) or `"laplace"`.
`ncores`	Number of cores used. Default doesn't use parallelism. You may use nb_cores.

Value

A list with 3 elements:

- $pheno: vector of phenotypes,
- $set: indices of causal variants,
- $effects: effect sizes corresponding to `set`.

snp_split *Split-parApply-Combine*

Description

A Split-Apply-Combine strategy to parallelize the evaluation of a function on each SNP, independently.

Usage

```
snp_split(infos.chr, FUN, combine, ncores = 1, ...)
```

Arguments

infos.chr	Vector of integers specifying each SNP's chromosome. Typically <bigSNP>mapchromosome.
FUN	The function to be applied. It must take a FBM.code256 as first argument and ind.chr, an another argument to provide subsetting over SNPs. You can access the number of the chromosome by using attr(ind.chr,"chr").
combine	function that is used by foreach to process the tasks results as they generated. This can be specified as either a function or a non-empty character string naming the function. Specifying 'c' is useful for concatenating the results into a vector, for example. The values 'cbind' and 'rbind' can combine vectors into a matrix. The values '+' and '*' can be used to process numeric data. By default, the results are returned in a list.
ncores	Number of cores used. Default doesn't use parallelism. You may use nb_cores.
...	Extra arguments to be passed to FUN.

Details

This function splits indices for each chromosome, then apply a given function to each part (chromosome) and finally combine the results.

Value

The result of foreach.

Examples

```
# parallelize over chromosomes made easy
# examples of functions from this package
snp_pruning
snp_clumping
snp_fastImpute
```

snp_subset *Subset a bigSNP*

Description

Subset (copy) of a bigSNP, also stored on disk.

Usage

```
snp_subset(
  x,
  ind.row = rows_along(x$fam),
  ind.col = rows_along(x$map),
  backingfile = NULL
)

## S3 method for class 'bigSNP'
subset(
  x,
  ind.row = rows_along(x$fam),
  ind.col = rows_along(x$map),
  backingfile = NULL,
  ...
)
```

Arguments

x	A bigSNP.
ind.row	Indices of the rows (individuals) to keep. Negative indices **can** be used to exclude row indices. Default: keep them all.
ind.col	Indices of the columns (SNPs) to keep. Negative indices **can** be used to exclude column indices. Default: keep them all.
backingfile	Prefix of the two new files created (".bk" and ".rds"). By default, it is automatically determined by appending "_sub" and a number to the prefix of the input bigSNP backing files.
...	Not used.

Value

The path to the RDS file that stores the bigSNP object.

See Also

bigSNP

Examples

```
str(test <- snp_attachExtdata())

# keep only first 50 samples and SNPs
rdsfile <- snp_subset(test, ind.row = 1:50, ind.col = 1:50)
str(snp_attach(rdsfile))

# remove only first 50 samples and SNPs
rdsfile2 <- snp_subset(test, ind.row = -(1:50), ind.col = -(1:50))
str(snp_attach(rdsfile2))
```

---

snp_writeBed *Write PLINK files from a "bigSNP"*

Description

Function to write bed/bim/fam files from a bigSNP. This will use the slot `code` **rounded** to write 0s, 1s, 2s or NAs.

Usage

```
snp_writeBed(x, bedfile, ind.row = rows_along(G), ind.col = cols_along(G))
```

Arguments

`x`	A bigSNP.
`bedfile`	Path to file with extension ".bed" to create.
`ind.row`	An optional vector of the row indices (individuals) that are used. If not specified, all rows are used. **Don't use negative indices.**
`ind.col`	An optional vector of the column indices (SNPs) that are used. If not specified, all columns are used. **Don't use negative indices.**

Value

The input `bedfile` path.

Examples

```
N <- 17
M <- 911

fake <- snp_fake(N, M)
G <- fake$genotypes
G[] <- sample(as.raw(0:3), size = length(G), replace = TRUE)
```

```
# Write the object as a bed/bim/fam object
tmp <- tempfile(fileext = ".bed")
bed <- snp_writeBed(fake, tmp)

# Read this new file for the first time
rds <- snp_readBed(bed, backingfile = tempfile())
# Attach object in R session
fake2 <- snp_attach(rds)

# Same content
all.equal(fake$genotypes[], fake2$genotypes[])
all.equal(fake$fam, fake2$fam)
all.equal(fake$map, fake2$map)

# Two different backingfiles
fake$genotypes$backingfile
fake2$genotypes$backingfile
```

---

sub_bed *Replace extension '.bed'*

Description

Replace extension '.bed'

Usage

```
sub_bed(path, replacement = "", stop_if_not_ext = TRUE)
```

Arguments

path	String with extension '.bed'.
replacement	Replacement of '.bed'. Default replaces by nothing. Can be useful to replace e.g. by '.bim' or '.fam'.
stop_if_not_ext	If replacement != "", whether to error if replacement is not an extension (starting with a '.').

Value

String with extension '.bed' replaced by replacement.

Examples

```
path <- "toto.bed"
sub_bed(path)
sub_bed(path, ".bim")
sub_bed(path, ".fam")
sub_bed(path, "_QC", stop_if_not_ext = FALSE)
```

Index