Package ‘MultiPhen’

October 12, 2022

Type Package

Title A Package to Test for Multi-Trait Association

Version 2.0.3

Date 2020-02-06

Author Lachlan Coin, Paul O'Reilly, Yotsawat Pompyen, Clive Hoggart and Federico Calboli

Depends MASS, abind, epitools, meta

Imports HardyWeinberg, RColorBrewer

Maintainer Lachlan Coin <l.coin@imb.uq.edu.au>

Description Performs genetic association tests between SNPs (one-at-a-time) and multiple phenotypes (separately or in joint model).

License GPL-2

LazyLoad yes

NeedsCompilation no

Repository CRAN

Date/Publication 2020-02-09 13:50:02 UTC

R topics documented:

`MultiPhen-package` *MultiPhen, a package for the genetic association testing of multiple phenotypes*

Description

MultiPhen is the package containing the function mPhen, which performs association testing between genetic variants (SNPs; CNVs to be added soon) and multiple phenotypes. The primary purpose is for modelling and testing multiple phenotypes jointly by performing an ordinal regression where SNPs are treated as the outcome and multiple phenotypes are predictors; this can have large increases in statistical power to detect genotype-phenotype associations over the univariate approach. However, mPhen can also be used to perform standard univariate linear regression (SNP as predictor) and univariate ordinal regression (SNP as outcome) on the phenotypes under study. mPhen can be applied to genotyped or imputed data. From version 0.4 the option "multiPhenTest" is now called "JointModel", and its default is now "TRUE"

Details

Package:	MultiPhen
Type:	Package
Version:	2.0.0
Date:	2013-12-09
License:	GPL-2
LazyLoad:	yes

Author(s)

Lachlan Coin, Federico Calboli, Clive Hoggart, Paul O'Reilly, Yotsawat Pomyen.

Maintainer: Federico Calboli <f.calboli@imperial.ac.uk>

References

O'Reilly et al. 2012. MultiPhen: Joint model of multiple phenotypes can increase discovery in GWAS. http://dx.plos.org/10.1371/journal.pone.0034861 `mPhen` *A function for the genetic association testing of multiple phenotypes*

Description

mPhen performs association testing between genetic variants (SNPs; CNVs) and multiple phenotypes. The primary purpose is for modelling and testing multiple phenotypes jointly by performing an ordinal regression where SNPs are treated as the outcome and multiple phenotypes are predictors; this can have large increases in statistical power to detect genotype-phenotype associations over the univariate approach (method described in O'Reilly et al. 2012, see below). However, mPhen can also be used to perform standard univariate linear regression (SNP as predictor) and univariate ordinal regression (SNP as outcome) on the phenotypes under study. mPhen can be applied to directly genotyped or imputed data.

Usage

```
mPhen(genoData, phenoData, phenotypes = "all",
      covariates = NULL, resids = NULL, strats = NULL,
      opts = mPhen.options(c("regression","pheno.input")))
```

Arguments

`genoData`	Either a matrix (for directly measured genotypes) or 3 dimensional array (for imputed genotypes). The first dimension (rows) corresponds to individuals, and row.names are inidividual IDs. The second dimension corresponds to SNPS, with col.names equal to the snp identifiers. For directly measured genotypes, the value in each cell is a numeric genotype(i.e. AA = 0, AB=1, BB = 2). For imputed genotypes, the 3rd dimension corresponds to genotypes (with dimnames(genoData)[[3]]) equal to a numeric vector corresponding to genotype values. The values in these cells are the probability of each genotype multiplied by 1000. For copy number genotypes the numeric values correspond to numbers of copies. An example provided by 'snps' and 'snps.imputed'
`phenoData`	Matrix containing phenotype data, where each row corresponds to an individual and row.names are individual IDs. Each column contains data on a certain phenotype across the sample of individuals (can be quantitative, case/control or ordinal. Must be numeric); the column header provides the phenotype name. An example is provided by 'pheno'.
`phenotypes`	Vector of phenotype names, to be tested. If value is 'all' then all phenotypes are included after removing covariates and residuals.
`covariates`	Vector of phenotypes, from phenoData, to be considered as covariates to be controlled for in the regression (Default is no covariates).
`resids`	Vector of residuals, from phenoData, alternative way to adjust for covariates, which pre-calculates offset terms to use in the per SNP regression (Default is no residuals).
`strats`	Statification vector (i.e. cases/controls, exposed/not exposed, male/female etc), from phenoData (Default is no stratification).

`opts` A list of options, which is obtained from mPhen.options(c("regression","pheno.input")). To get more information about these options, type mPhen.options(c("regression","pheno.input"),descr=TF

Value

Returns a list, with two items. The first item (Results) is a Results is a 4 dimensional matrix, with dimensions [strata, snps, phenotypes, result_type], where result_type includes beta, pvalue and Nobs. The second item is a vector of minor allele frequencies.

Note

The user should remember that the genotype data file is always a matrix of at least a column, hence if taking a subset of 1 SNP in the non-imputed genotype data matrix, the option drop = FALSE should be used (see the example below)

Author(s)

Lachlan Coin, Federico Calboli, Clive Hoggart, Paul O'Reilly, Yotsawat Pomyen.

Maintainer, Federico Calboli <f.calboli@imperial.ac.uk>

References

O'Reilly et al. 2012. MultiPhen: Joint model of multiple phenotypes can increase discovery in GWAS. http://dx.plos.org/10.1371/journal.pone.0034861

Examples

```
data(snps); data(snps.imputed); data(pheno)
opts = mPhen.options(c("regression","pheno.input"))
res = mPhen(snps, pheno, phenotypes = "all",
      covariates = c('testPheno3', 'testPheno4'),opts = opts)
# performs a MultiPhen analysis, with snp as outcome,
# and phenotypes testPheno1, testPheno2 as predictors,
#with testPheno3 and testPheno4 as covariates using ordinal regression

res = mPhen(snps, pheno, phenotypes = c('testPheno1', 'testPheno2'),
      covariates = c('testPheno3', 'testPheno4'), resids = 'testPheno5', opts = opts)
# the same as above, with the fifth phenotype as residual

res = mPhen(snps[,2, drop = FALSE], pheno, phenotypes = c('testPheno1', 'testPheno2'),
      covariates = 'testPheno3',  opts = opts)
# please note the use use of drop = FALSE if analysing only one SNP


res = mPhen(snps.imputed, pheno, phenotypes = c('testPheno1', 'testPheno2'),
      covariates = 'testPheno3',  opts = opts)
# for imputed data
```

`mPhen.assoc` *A function for the genetic association testing of multiple phenotypes*

Description

This function is called by mPhen. If you are doing association on multiple batches of genotype data, it is more efficient to use this function, and to pre-prepare a 'phenoObject' object once and then use this function

Usage

```
mPhen.assoc(genoData, phenoObject,opts = mPhen.options("regression"),
 subinds =1:(dim(genoData)[1]))
```

Arguments

`phenoObject`	A phenotype object prepared by mPhen.preparePheno
`genoData`	This can be obtained from mPhen.readGenoConnection(...)$genoData. It is either a matrix (for directly measured genotypes) or 3 dimensional array (for imputed genotypes). The first dimension (rows) corresponds to individuals, and row.names are inidividual IDs. The second dimension corresponds to SNPS, with col.names equal to the snp identifiers. For directly measured genotypes, the value in each cell is a numeric genotype(i.e. AA = 0, AB=1, BB = 2). For imputed genotypes, the 3rd dimension corresponds to genotypes (with dimnames(genoData)[[3]]) equal to a numeric vector corresponding to genotype values. The values in these cells are the probability of each genotype multiplied by 1000. For copy number genotypes the numeric values correspond to numbers of copies. An example provided by 'snps'.
`opts`	A list of options, which is obtained from mPhen.options("regression"). To get more information about these options, type mPhen.options("regression",descr=TRUE)
`subinds`	This indicates the indices of individuals to include in the analysis. It is possible to have repeat indices (i.e. for bootstrap)

Value

Returns a list, with two items. The first item (Results) is a Results is a 4 dimensional matrix, with dimensions [strata, snps, phenotypes, result_type], where result_type includes beta, pvalue and Nobs. The second item is a vector of minor allele frequencies.

mPhen.cca *Sparse canonical correlation analysis*

Description

Carries out a sparse canonical correlation analysis

Usage

```
mPhen.cca(genoData, phenoObject, opts =mPhen.options("regression"),
         subinds = 1:(dim(genoData)[1]),
vs.G = opts$mPhen.variable.selection,
                 vs.P = opts$mPhen.variable.selection
)
```

Arguments

genoData	A 2 dimensional array. The first dimension (rows) corresponds to individuals, and row.names are inidividual IDs. The second dimension corresponds to SNPS, with col.names equal to the snp identifiers. The entries are either genotypes, or expected genotypes.
phenoObject	A phenotype object prepared by mPhen.preparePheno.
opts	A list of options, which is obtained from mPhen.options("regression"). To get more information about these options, type mPhen.options("regression",descr=TRUE).
vs.G	If true performs variable selection on genotypes. Is equal to opts$mPhen.variable_selection by default
vs.P	If true performs variable selection on phenotypes. Is equal to opts$mPhen.variable_selection by default
subinds	This indicates the indices of individuals to include in the analysis. It is possible to have repeat indices (i.e. for bootstrap)

Value

A list with following elements

betasp	Phenotype weights
betasg	Genotype weights
resultsGeno	Single (combined) phenotype analysis results against all genotypes
resultsPheno	Multiple phenotype analysis results against single (combined) genotype

---

`mPhen.defineOptions` *Defines and modifies options.*

Description

This reads options from the command-line, if provided. Also it can replace references to system variables in option values, such as '*' or \$WORK, with fully qualified values. Also translates coordinates which use Gb, Mb, Kb to integer values. All defined options of the form mPhen.xxx are examined and modified.

Usage

```
mPhen.defineOptions(file = NULL, getOptionsFromCommandLine = TRUE)
```

Arguments

`file`	specified if the options are in a script file and not set manually
`getOptionsFromCommandLine`	If running from a script using command Rscript, then this will read in command line options, such as \'–mPhen.logp=FALSE\'

Value

None

`mPhen.options` *Retrieves default mPhen options, and descriptions.*

Description

This command is used to get options which can be modified to control the behaviour of MultiPhen commands. It provides a list of options which are relevant to a particular command. For example, mPhen.assoc() has its behavioiur controlled by options in mPhen.options("regression"). In order to get a list of all options, you can type mPhen.options(descr=TRUE).

Usage

```
mPhen.options(type=c("regression", "plot", "geno.input",
"pheno.input","meta.analysis","misc"), descr = FALSE)
```

Arguments

`type`	A value which can take any of the following values: "regression", "plot","geno.input","pheno.input","meta
`descr`	If set to TRUE, then returns descriptions of all the options. If FALSE, then returns the values of all the options.

Value

A list of default option values. Note, the default value for opts = mPhen.options("regression"), has opts$inverseRegress =TRUE, opts$JointModel = TRUE and opts$geno.link = "ordinal", which is the standard multiPhen model.

`mPhen.plotCorrelation` *Plots correlation between phenotype values conditional on genotype*

Description

Plots the corrleation between phenotype values, with different genotypes coloured differently. Note that this will plot (dim(pheno_to_plot)[2] -1)* dim(geno)[2] plots

Usage

```
mPhen.plotCorrelation(pheno_to_plot,geno,title="",cex=0.25, cols =c(1, 2, 3))
```

Arguments

`pheno_to_plot`	the phenotype to plot
`geno`	Genotypes to use for stratifying samples
`title`	Title of plots
`cex`	Scaling of points
`cols`	thre colours to be used in the plot

Value

None

`mPhen.preparePheno` *Prepare phenotype data for analysis*

Description

This harmonises the phenotype data with genotype data, and also extract the relevant columns from a larger phenotype matrix, and also pre-calculates stratification indices and residuals. This is called by mPhen function, but quicker to do this just once for batched genotype data.

Usage

```
mPhen.preparePheno(phenoData,
pcs = NULL, indiv = if (is.null(pcs)) rownames(phenoData) else rownames(pcs),
    opts = mPhen.options("regression"))
```

Arguments

`phenoData`	This is typically the output of mPhen.readPhenoFiles(...) or mPhen.simulate(...). It is a list containing the elements phenoData$pheno and phenoData$limit. The element phenoData$pheno is a matrix containing phenotype data, where each row corresponds to an individual and row.names are individual IDs. Each column contains data on a certain phenotype across the sample of individuals (can be quantitative, case/control or ordinal. Must be numeric); the column header provides the phenotype name. An example is provided by 'pheno'. The element phenoData$limit is a list which specifies which of the phenotypes to use as covariates, variables to associate, etc, in the following way: limit$phenotypes - vector of phenotypes to be tested. If set to 'all' then all phenotypes are used. limit$covariates - vector of phenotypes to be considered as covariates to be controlled for in the regression , limit$resids - vector of phenotypes to be considered as residuals, which is an alternative way to adjust for covariates, which pre-calculates offset terms to use in the per SNP regression limit$strats - statification vector (i.e. cases/controls, exposed/not exposed, male/female etc). limit$excls - Exclusion vector, ie. names of phenotypes which should be used as exclusion criteria respectively. Rows will be excluded if the value in any of exclusion columns is NA or 1 Alternatively, phenoData can simply be a matrix containing phenotype data, in which case, the default value of limit used is limit = list(phenotypes="all")
`pcs`	This is the genotype pcs which should be used in the analysis. If specified it should be in the same sample order as 'inidv'. The user still needs to specify in the phenoData$limit$covariate or phenoData$limit$resids the names of the PCs they wish to include (i.e. covariate = c("PC1","PC2")). These would typically be obtained from mPhen.readGenotypes.
`indiv`	individuals to be used. If unspecified the fucntion defaults to using all individuals
`opts`	A list of options, which is obtained from mPhen.options("regression"). To get more information about these options, type mPhen.options("regression",descr=TRUE)

Value

A list object, which can then be used in mPhen.assoc.

`mPhen.readGenotypes` *Open, and read from a read connection to a genotype file*

Description

Opens a read connection to a list of files which have a VCF-like format. Can read a .gz file. Also supports plink bed format. Also supports cnvPipe format. Also supports a .zip file format used by cnvHap.

Usage

```
mPhen.readGenotypes(genoConnection, indiv = NULL,opts =mPhen.options("geno.input" ))
```

Arguments

`genoConnection`	A list of paths to genotype files. Each file might represent a different cohort with overlapping snp set. These files can be .vcf .vcf.gz and .zip files. They can also be plink bed files. In the case that bed files are used, the root name of the file should be given (i.e. hapmap2 instead of hapmap2.bed). Alternatively, this can also be the object returned by mPhen.readGenotypes(...). This means that the connection only has to be established once, and can be read from multiple times in batches.
`indiv`	A list of individual ids. If provided the results will be in this order. If not provided the results will be in the order given by the genotype file
`opts`	A list of options, which is obtained from mPhen.options("geno.input"). To get more information about these options, type mPhen.options("geno.input",descr=TRUE)

Value

Returns a list object which can be used by subsequent calls to readGenotypes(..). This list also includes a value genoData, which contains the genotypes which have been read. This includes a connection to the underlying file (conn), a list of sample ids, and a flag indicating whether the file is zipped. It also includes 'pcs', which is a matrix of genotype pcs. These are only calculated is opts$mPhen.numGenoPCs>0, and are only calculated once all batches have been read (but will include all genotypes which have been read previously using the same connection).

`mPhen.readPhenoFiles` *Read and merge phenotype files*

Description

This helper function merges multiple phenotype files into a single phenotype matrix, and applies missing value and exclusion criteria

Usage

```
mPhen.readPhenoFiles(phenoFiles,
limitFile = getOption("mPhen.limitFile","./limit.txt"),
 excludeFile =getOption("mPhen.excludeFile","./exclude.txt"),
opts = mPhen.options("pheno.input"),
indiv = NULL)
```

Arguments

`phenoFiles`	A list of paths to phenotype files (can be more than 1)
`excludeFile`	A path to a file which lists ids to exclude from further analysis, or alternatively is a two column file, with the first column of ids and a second column of numberical values which are used in conjunction with opts$quantileThresh
`limitFile`	As an alternative to specifying covariates, resids,strats and excl in mPhen.preparePheno(..), you can also specify this information via a limitfile, which is tab delimited file in which the first column specifies the type of variable to set (pheno,covar,resid,strat,excl), the second column specifies the phenotype name, and the third column optionally specifies a transformation Different lines can then be used for different values. The transformation syntax includes 'quantile' and 'factor', and also 'thresh_x_y' in which values less than x are coded 0 and greater than y are coded 1; and also 'toptail_x_y' where values less than x percentile are coded 0 and greater than y percentile are coded 1.
`opts`	A list of options, which is obtained from mPhen.options("pheno.input"). To get more information about these options, type mPhen.options("pheno.input",descr=TRUE)
`indiv`	A list of individual ids. If provided the phenotype matrix will be re-ordered to match

Value

An object consisting of a single merged phenotype matrix, and also a 'limit' object which specifies phenotypes to include in analyses. The limit is a list with the following entries. phenotypes - vector of phenotypes to be tested. If set to 'all' then all phenotypes are used. covariates - vector of phenotypes to be considered as covariates to be controlled for in the regression , resids - vector of phenotypes to be considered as residuals, which is an alternative way to adjust for covariates, which pre-calculates offset terms to use in the per SNP regression strats - statification vector (i.e. cases/controls, exposed/not exposed, male/female etc).

excls - Exclusion vector, ie. names of phenotypes which should be used as exclusion criteria respectively. Rows will be excluded if the value in any of exclusion columns is NA or 1

`mPhen.sampleCovar` *Generates a covariance matrix.*

Description

This function can be use to sample covariance matrices. This is useful when simulating data to test Multiphenotype based association strategies. This function lets the user decide on the orthoganality within 'blocks' and between 'blocks' of correlated variables/

Usage

```
mPhen.sampleCovar(noPhenos,blockSize, orthogAll = c(0.9,0.5),
dirichletScale = 50,  resample = FALSE,
sd = rgamma(noPhenos,shape=10,rate = 10))
```

Arguments

noPhenos	The number of phenotypes to simulate
blockSize	The number of phenotypes per covariance block
orthogAll	The orthogonality relationships between and within blocks expressed as a number on the interval (0,1). A number closer to one indicates closer to orthogonality, whereas 0 indicates non-orthogonality. First number is orthogonality between blocks, second is orthogonality within blocks.
dirichletScale	When sampling off diagonal elements of the cholesky decomposition, how much deviation from uniform to allow. Should be a number in interval (0,+Inf). Smaller value leads to greater variation
resample	Whether to randomly shuffle phenotype columns after sampling.
sd	Standard deviation for each phenotype

Value

Simulated covariance matrix

`mPhen.sampleGeno` *Sample genotypes*

Description

...

Usage

```
mPhen.sampleGeno(n = 100, sampSize = 100, chr="0",pos = 1:n,
snpids = paste(chr,pos,sep="_"),meanAlleleFreq=0.2, mu = 10,
samples =paste("id",1:sampSize,sep="_"),imputed = FALSE, dirichlet = 1)
```

Arguments

n	Number of genotypes to sample
sampSize	Number of individuals to sample
chr	Name of chromosome
pos	Positions of genotypes on chromosome
snpids	Ids of genotypes
meanAlleleFreq	The mean allele frequency to simulate
mu	A weight parameter which controls how close to the meanAlleleFreq the allele frequencies are sampled, via a beta distribution. A higher number implies allele frequencies stay closer to mean
samples	The sample ids
imputed	Whether to simulate imputed data
dirichlet	The weight of a dirichlet distribution used to simulated imputed data

Value

Returns matrix of genotypes, with individuals by rows, and snps by column, or a 3 dimensional array if imputed is TRUE

`mPhen.simulate` *Simulates phenotypes according to a correlation structure.*

Description

This function simulates phenotypes based on a pre-defined correlation structure (which can also be obtained from mPhen.sampleCovar), and a genetic effect x. The function works by sampling a phenotype from a correlation matrix in a linearly transformed space such that the genetic effect direction is only in the direction of the x-axis, then transforming back into the original space. If inverse is TRUE, then the phenotypes are sampled first with no genetic effect, then the genotype is sampled according to the effect direction.

Usage

```
mPhen.simulate(x,sample_names, covar,effDir,
varexp,inverse=FALSE, geno.link="gaussian",
effDirInReverseEigenspace=FALSE, freq = 0.1)
```

Arguments

`x`	A vector of genotype effect. If a single SNP has an effect, this will just be genotypes at this SNP
`sample_names`	Vector of sample names, should have same length as x
`covar`	Covariance of phenotypes, should be an n by n matrix, where n is the number of phenotypes to simulate.
`effDir`	Direction in phenotype space in which to simulate the effect
`varexp`	The proportion of variance of phenotype variation in the target direction explained by the genotypic effect overall.
`inverse`	If TRUE, then simulates correlated phenotypes, and then simulates genotypes from phenotypes in specified direction. Otherwise, simulates correlated phenotypes with direction of effect based on input genotypes.
`geno.link`	Only applicable if inverse = TRUE, in which case it specifies a link function for genotypes. Can be binomial, gaussian or ordinal.
`effDirInReverseEigenspace`	If TRUE, then effDir is interpreted as eigenvector weights, ordered from eigenvector with smallest eigenvalue to eigenvector with biggest eigenvector (i.e. in reverse direction). This is useful if you want to simulate directs which are in the least variable axis of variation.
`freq`	If inverse = TRUE, then this is target allele frequency.

Value

Returns a list of three elements: pheno - a matrix of phenotype values, with phenotypes in columns and samples in rows; and geno- a vector genotypes, either sampled if inverse=TRUE, or the original genetic effect x; and limit - this is a default list of phenotypes to include in subsequent association analysis, as well as covariates (empty) and residuals (empty). This object can then be used by mPhen.preparePheno

---

`mPhen.writeOutput` *Prepares output files and plots from MultiPhen results*

Description

Writes output to files defined in mPhen.openOutputConnection, and extracts pvalues and betas for further plots.

Usage

```
mPhen.writeOutput(results,
output = getOption("mPhen.resultsName","resultsDir/"),geno = NULL,
towrite = list(long.txt = getOption("mPhen.writeLong",TRUE),
qc.txt =  getOption("mPhen.writeQC",FALSE),
wide.txt = getOption("mPhen.writeWide",TRUE)),
toplot = list(.manh = TRUE, .qq = TRUE,.heatm = TRUE,

              .fprint = !is.null(geno)),
opts = mPhen.options("plot"))
```

Arguments

`results`	Output of mPhen.assoc
`output`	Directory to write results, or object returned by mPhen.writeOutput(..)
`towrite`	List specifying which formats to write output - long.txt and wide.txt for standard results; qc.txt for per-sample qc output.
`toplot`	List specifying which formats to plot output - .qq for qq plot, .manh for manhattan; .heatm for pvalue heatmap; .fprint for fingerprint plot
`geno`	Genotype matrix. Note that attr(geno,"closeConnection") controls whether plots are produced, as this indicates whether all batches of genotype data have been analysed
`opts`	A list of options, which is obtained from mPhen.options("plot"). To get more information about these options, type mPhen.options("plot",descr=TRUE)

Value

Returns an outputConnection, which can be used to write further results.

pheno	*A dummy phenotype dataset that provides an example of the input phenotype data used by the package*

Description

A dummy dataset of 5 phenotypes measured in 150 individuals. The data has been generated to yield significant results for SNP1 and SNP2 of the snps dataset. The first two columns have been generated as alpha + beta1*snp + beta2*snp2 + error (with different alphas, betas and errors for each phenotype), the third has been generated as alpha + beta1*testPheno2 + beta2*snp3 + error, the fourth column is the results of sample of a binomial distributioni correlated with testPheno3, and the final column is the 1st PC of the principal component analysis of the snps matrix.

Format

A matrix with 150 phenotype observations.

`testPheno1`	a numeric vector
`testPheno2`	a numeric vector
`testPheno3`	a numeric vector
`testPheno4`	a numeric vector
`testPheno5`	a numeric vector

Details

Please note the following IMPORTANT issue: the 'pheno' matrix has both column names and row names! the column names MUST be the names of the phenotypes and the row names MUST be the codes representing each individual in the pheno matrix, one individual for each row. Both row names and column names are extracted by the main function and are therefore mandatory

Examples

```
data(pheno)
head(pheno)
dimnames(pheno)[[1]] # the row names
dimnames(pheno)[[2]] # the column names
```

`read.plink` *A function to read (small) binary PLINK binary files in a R session*

Description

`read.plink` is a convenience function designed to read PLINK binary files (i.e. files that end with the suffix “.bed”) in a R session. Please be aware that binary PLINK files are binary for a reason, i.e. to store genotype data in a compact way. Once they are imported in R they exist in R in a un “unpacked” form, and can therefore be very big. If the .bed file is big, or very big, the result will be that R will run out of memory and crash, or make the whole system slow or unresponsive. It is MANDATORY that in the directory containing the binary file also reside two accesory files, with the same name as the binary file but with extensions .fam and .bim, both produced by PLINK.

Usage

```
read.plink(root,indiv = NULL, opts = mPhen.options("geno.input"))
```

Arguments

`root`	filename of the dataset in PLINK binary format, WITHOUT the .bed extension.
`indiv`	List of individuals, results will be in this order
`opts`	List of options, use mPhen.options("geno.input",descr=TRUE) for more details of each option.

Details

Please note that, if the binary file is listed a “mydata.bed”, the filename is “mydata”, and the extension is “.bed”. In this case “mydata” would be used as `root` value.

Value

A matrix of dimesions n by m, with n rows corresponding to the n individuals in the dataset, and m columns corresponding to the m markers. The colnames are retrived from the .fam file, and (should) correspond to the markers’ names.

Note

Please do note that the concept of a “big” binary file, or a binary file that is “too big” is purely dependent on the computer on which the code is running. A computer with 512MB of RAM will stop being able to read in a whole binary file well before a 16GB RAM machine.

Author(s)

Federico Calboli <`f.calboli@imperial.ac.uk`>

References

The plink homepage is at: `https://www.cog-genomics.org/plink2`

snps — *A dummy snp dataset that provides an example of the input snp data used by the package*

Description

A dummy dataset of three SNPs, as a matrix of 3 column and 150 rows. The genotypes are in 0/1/2 format (0 for "AA", 1 for "Aa" and 2 for "aa", where A and a correspond, arbitrarily, to the two alleles). The data has been randomly generated, for instructional purposes only, and do not yield a significant association with any of the example phenotypes.

Usage

```
data(snps)
```

Format

A data frame with 150 genotype observations.

`rsID1`	a numeric vector
`rsID2`	a numeric vector
`rsID3`	a numeric vector

Details

The 150 genotypes in 'snp' correspond to the phenotype data on 150 individuals in 'pheno', i.e. one individual for each line. Please note the following important points: genotype data must be in matrix format, with one row for each individual and as many columns for each SNP. In the case of one single genotype the data must still conform to this format, as a matrix of as many rows as individuals and one single column for the one genotype present. A second important point is that the column names must be the rsID of the SNP for genotypes in the 0/1/2 format. Further options of genotype format (incorporating raw genotype data, and CNV genotypes) will be available and documented in future releases.

Examples

```
data(snps)
dim(snps)
colnames(snps)
```

`snps.imputed` *Imputed SNP dataset*

Description

A toy dataset of three imputed SNP, for 150 individuals. For each individuals, and for each SNP, the first column is the probability of a minor allele homozygote genotype (genotype “0”), the second column is the probability of an heterozygote genotype (genotype “1”) and the third and last column is the probability of a major allele homozygote (genotype “2”).

Usage

```
data(snps.imputed)
```

Format

A matrix of 150 rows and nine columns.

Examples

```
data(snps.imputed)
```

Index