Package 'SNPassoc'

February 19, 2015

Version 1.9-2

Date 2014-Apr-22

Depends R (>= 3.0.0), haplo.stats, survival, mvtnorm, parallel

Title SNPs-based whole genome association studies

Author Juan R González, Lluís Armengol, Elisabet Guinó, Xavier Solé, and Víctor Moreno

Maintainer Juan R González <jrgonzalez@creal.cat>

Description This package carries out most common analysis when performing whole genome association studies. These analyses include descriptive statistics and exploratory analysis of missing values, calculation of Hardy-Weinberg equilibrium, analysis of association based on generalized linear models (either for quantitative or binary traits), and analysis of multiple SNPs (haplotype and epistasis analysis). Permutation test and related tests (sum statistic and truncated product) are also implemented. Max-statistic and genetic risk-allele score exact distributions are also possible to be estimated.

License GPL (>= 2)

URL http://www.creal.cat/jrgonzalez/software.htm

Encoding latin1

NeedsCompilation no

Repository CRAN

Date/Publication 2014-04-23 09:27:48

R topics documented:

---

association   *Association analysis between a single SNP and a given phenotype*

Description

This function carries out an association analysis between a single SNP and a dependent variable (phenotype) under five different genetic models (inheritance patterns): codominant, dominant, recessive, overdominant and log-additive. The phenotype may be quantitative or categorical. In the second case (e.g. case-control studies) this variable must be of class 'factor' with two levels.

Usage

```
association(formula, data, model=c("all"), model.interaction=
        c("codominant"), subset, name.snp = NULL, quantitative =
        is.quantitative(formula,data), genotypingRate= 0,
        level = 0.95, ...)
```

Arguments

formula   a symbolic description of the model to be fited (a formula object). It might have
                 either a continuous variable (quantitative traits) or a factor variable (case-control
                 studies) as the response on the left of the ~ operator and a term correspond-
                 ing to the SNP on the right. This term must be of class snp (e.g. ~snp(var),
                 where var is a given SNP), and it is required. Terms with additional covari-
                 ates on the right of the ~ operator may be added to fit an adjusted model (e.g.,
                 ~var1+var2+...+varN+SNP). The formula allows to incorporate more than one

	object of class snp. In that case, the analysis is done for the first SNP which appears in the formula adjusted by the others covariates and other additional SNPs.
data	a required dataframe of class 'setupSNP' containing the variables in the model.
model	a character string specifying the type of genetic model (mode of inheritance) for the SNP. This indicates how the genotypes should be collapsed. Possible values are "codominant", "dominant", "recessive", "overdominant", "additive" or "all". The default is "all" that fits the 5 possible genetic models. Only the first words are required, e.g "co", "do", etc.
model.interaction	
	a character string specifying the type of genetic model (mode of inheritance) assumed for the SNP when it is included in a interaction term. Possible values are "codominant", "dominant", "recessive", "overdominant". The default is "codominant".
subset	an optional vector specifying a subset of observations to be used in the fitting process
name.snp	optional label of the SNP variable to be printed.
quantitative	logical value indicating whether the phenotype (that which is in the left of the operator ~ in 'formula' argument) is quantitative. The function 'is.quantitative' returns FALSE when the phenotype is a variable with two categories (i.e. indicating case-control status). Thus, it is not a required argument but it may be modified by the user.
genotypingRate	minimum percentage of genotype rate for the SNP to be analyzed. Default is 0% (e.g. all SNPs are analyzed). This parameter should not be changed. It is used in the function 'WGassociation'.
level	signification level for confidence intervals.
...	Other arguments to be passed through glm function

Details

This function should be called by the user when we are interested in analyzing an unique SNP. It is recommended to use [WGassociation](#) function when more than one SNP is studied.

Value

For each genetic model (codominant, dominant, recessive, overdominant, and log-additive) the function gives a matrix with sample size and percentages for each genotype, the Odds Ratio and its 95% confidence interval (taking the most frequent homozygous genotype as the reference), the p-value corresponding to the likelihood ratio test obtained from a comparison with the null model, and the Akaike Information Criterion (AIC) of each genetic model. In the case of analyzing a quantitative trait, the function returns a matrix with sample size, mean and standard errors for each genotype, mean difference and its 95% confidence interval with respect to the most frequent homozygous genotype, the p-value obtained from an overall gene effect and the Akaike Information Criterion (AIC) of each genetic model.

When an interaction term (a categorical covariate with an SNP) is included in the model, three different tables are given. The first one correponds to the full interaction matrix where the ORs (or mean differences if a quantitative trait is analyzed) are expressed with respect to the non variant genotype and the first category of the covariate. The other two tables show the ORs and their 95% confidence intervals for both marginal models. P values for interaction and trend are also showed in the output.

References

JR Gonzalez, L Armengol, X Sole, E Guino, JM Mercader, X Estivill, V Moreno. SNPassoc: an R package to perform whole genome association studies. Bioinformatics, 2007;23(5):654-5.

Iniesta R, Guino E, Moreno V. Statistical analysis of genetic polymorphisms in epidemiological studies. Gac Sanit. 2005;19(4):333-41.

Elston RC. Introduction and overview. Statistical methods in genetic epidemiology. Stat Methods Med Res. 2000;9:527-41.

See Also

WGassociation

Examples

```
data(SNPs)

# first, we create an object of class 'setupSNP'
datSNP<-setupSNP(SNPs,6:40,sep="")

# case-control study, crude analysis
association(casco~snp10001, data=datSNP)

# case-control study, adjusted by sex and arterial blood pressure
association(casco~sex+snp10001+blood.pre, data=datSNP)


# quantitative trait, crude analysis
association(log(protein)~snp10001,data=datSNP)
# quantitative trait, adjusted by sex
association(log(protein)~snp10001+sex,data=datSNP)


#
# Interaction analysis
#

# Interaction SNP and factor
association(log(protein)~snp10001*sex+blood.pre, data=datSNP,
          model="codominant")

# Interaction SNP and SNP (codominant and codominant)
association(log(protein)~snp10001*factor(snp10002)+blood.pre,
          data=datSNP, model="codominant")
```

```
# Interaction SNP and SNP (dominant and recessive)
association(log(protein)~snp10001*factor(recessive(snp100019))+blood.pre,
          data=datSNP, model="dominant")
```

Bonferroni.sig	*Bonferroni correction of p values*

Description

This function shows the SNPs that are statistically significant after correcting for the number of tests performed (Bonferroni correction) for an object of class "WGassociation"

Usage

```
Bonferroni.sig(x, model = "codominant", alpha = 0.05,
    include.all.SNPs=FALSE)
```

Arguments

x	an object of class 'WGassociation'.
model	a character string specifying the type of genetic model (mode of inheritance). This indicantes how the genotypes should be collapsed when 'plot.summary' is TRUE. Possible values are "codominant", "dominant", "recessive", "overdominant", or "log-additive". The default is "codominant". Only the first words are required, e.g "co", "do", ...
alpha	nominal level of significance. Default is 0.05
include.all.SNPs	
	logical value indicating whether all SNPs are considered in the Bonferroni correction. That is, the number of performed tests is equal to the number of SNPs or equal to the number of SNPs where a p value may be computed. The default value is FALSE indicating that the number of tests is equal to the number of SNPs that are non Monomorphic and the rate of genotyping is greater than the percentage indicated in the GeneticModel.pval function.

Details

After deciding the genetic model, the function shows the SNPs that are statistically significant at alpha level corrected by the number of performed tests.

Value

A data frame with the SNPs and the p values for those SNPs that are statistically significant after Bonferroni correction

See Also

[WGassociation](#)

Examples

```
data(SNPs)
datSNP<-setupSNP(SNPs,6:40,sep="")
ans<-WGassociation(protein~1,data=datSNP,model="all")
Bonferroni.sig(ans, model="codominant", alpha=0.05, include.all.SNPs=FALSE)
```

GenomicControl *Population substructure*

Description

This function estimates an inflation (or deflation) factor, lambda, as indicated in the paper by Devlin et al. (2001) and corrects the p-values using this factor.

Usage

```
GenomicControl(x, snp.sel)
```

Arguments

x	an object of class 'WGassociation'.
snp.sel	SNPs used to compute lambda. Not required.

Details

This method is only valid for 2x2 tables. This means that the object of class 'WGassociation' might not have fitted the codominant model.

See reference for further details.

Value

The same object of class 'WGassociation' where the p-values have been corrected for genomic control.

References

B Devlin, K Roeder, and S.A. Bacanu. Unbiased Methods for Population Based Association Studies. Genetic Epidemiology (2001) 21:273-84

See Also

[qqpval](#), [WGassociation](#)

Examples

```
data(SNPs)
datSNP<-setupSNP(SNPs,6:40,sep="")
res<-WGassociation(casco,datSNP,model=c("do","re","log-add"))

# Genomic Control
resCorrected<-GenomicControl(res)
plot(resCorrected)
```

---

getSignificantSNPs *Extract significant SNPs from an object of class 'WGassociation'*

Description

Extract significant SNPs from an object of class 'WGassociation' when genomic information is available

Usage

```
getSignificantSNPs(x, chromosome, model, sig = 1e-15)
```

Arguments

x	an object of class 'WGassociation'
chromosome	chromosome from which SNPs are extracted
model	genetic model from which SNPs are extracted
sig	statistical significance level. The default is 1e-15

Value

A list with the following components:

names	the name of SNPs
column	the columns corresponding to the SNPs in the original data frame

...

See Also

[WGassociation](#)

Examples

```
data(HapMap)
# resHapMap contains the results for a log-additive genetic model

# to get the significant SNPs for chromosome 12
getSignificantSNPs(resHapMap,chromosome=12)
# to get the significant SNPs for chromosome 5
getSignificantSNPs(resHapMap,5)
# to get the significant SNPs for chromosome X at level 1e-8
getSignificantSNPs(resHapMap,5,sig=1e-8)
```

---

  haplo.interaction   *Haplotype interaction with a covariate*

Description

This function computes the ORs (or mean differences if a quantitative trait is analyzed) and their
95% confidence intervals corresponding to an interaction between the haplotypes and a categorical
covariate

Usage

```
haplo.interaction(formula, data, SNPs.sel, quantitative =
  is.quantitative(formula, data), haplo.freq.min = 0.05, ...)
```

Arguments

formula   a symbolic description of the model to be fitted (a formula object). It might have
                 either a continuous variable (quantitative traits) or a factor variable (case-control
                 studies) as the response on the left of the ~ operator and a term corresponding to
                 the interaction variable on the right indicated using 'interaction' function (e.g.
                 ~int(var), where var is a factor variable) and it is required. Terms with additional
                 covariates on the the right of the ~ operator may be added to fit an adjusted model
                 (e.g., ~var1+var2+...+varN+int(var)).

data   an object of class 'setupSNP' containing the variables in the model and the SNPs
                 that will be used to estimate the haplotypes.

SNPs.sel   a vector indicating the names of SNPs that are used to estimate the haplotypes

quantitative   logical value indicating whether the phenotype (which is on the left of the op-
                 erator ~ in 'formula' argument) is quantitative. The function 'is.quantitative'
                 returns FALSE when the phenotype is a variable with two categories (i.e. in-
                 dicating case-control status). Thus, it is not a required argument but it may be
                 modified by the user.

haplo.freq.min   control parameter for haplo.glm included in 'haplo.glm.control'. This parameter
                 corresponds to the minimum haplotype frequency for a haplotype to be included
                 in the regression model as its own effect. The haplotype frequency is based
                 on the EM algorithm that estimates haplotype frequencies independently of any
                 trait.

... additional parameters for 'haplo.glm.control'.

Details

The function estimates the haplotypes for the SNPs indicated in the 'SNPs.sel' argument. Then, usign 'haplo.glm' function (from 'haplo.stats' library) estimates the interaction between these haplotypes and the covariate indicated in the formula by means of 'interaction' function.

Value

Three different tables are given. The first one corresponds to the full interaction matrix where the ORs (or mean differences if a quantitative trait is analyzed) are expressed with respect to the most frequent haplotype and the first category of the covariate. The other two tables show the ORs (or mean differences if a quantitative trait is analyzed) and their 95% confidence intervals for both marginal models. P values for interaction are also showed in the output.

Examples

```
# not Run
# data(SNPs)
# datSNP<-setupSNP(SNPs,6:40,sep="")
# res<-haplo.interaction(log(protein)~int(sex), data=datSNP,
#     SNPs.sel=c("snp100019","snp10001","snp100029"))
# res
```

HapMap   *SNPs from HapMap project*

Description

Information about 9307 SNPs from the HapMap project belonging to 22 chromosomes. Information about two different population is available: European population (CEU) and Yoruba (YRI). The genomic information (names of SNPs, chromosomes and genetic position) is also available in a data frame called 'HapMap.SNPs.pos'.

Usage

```
data(HapMap)
```

Format

A data frame with 120 observations on the 9808 variables (SNPs) and one variable called 'group' indicating the population.

Source

HapMap project (http://www.hapmap.org)

Examples

```
data(HapMap)
```

---

inheritance	*Collapsing (or recoding) genotypes into different categories (generally two) depending on a given genetic mode of inheritance*

Description

codominant function recodifies a variable having genotypes depending on the allelic frequency in descending order.

dominant, recessive, and overdominant functions collapse the three categories of a given SNP into two categories as follows: Let 'AA', 'Aa', and 'aa' be the three genotypes. After determining the most frequent allele (let's suppose that 'A' is the major allele) the functions return a vector with to categories as follows. dominant: 'AA' and 'Aa-aa'; recessive: 'AA-Aa' and 'aa'; overdominant: 'AA-aa' vs 'Aa'.

additive function creates a numerical variable, 1, 2, 3 corresponding to the three genotypes sorted out by descending allelic frequency (this model is referred as log-additive).

Usage

```
codominant(o)
dominant(o)
recessive(o)
overdominant(o)
additive(o)
```

Arguments

o   categorical covariate having genotypes

Examples

```
data(SNPs)
dominant(snp(SNPs$snp10001,sep=""))
overdominant(snp(SNPs$snp10001,sep=""))
```

---

int   *Identify interaction term*

Description

This is a special function used for 'haplo.interaction' function. It identifies the variable that will interact with the haplotype estimates. Using int() in a formula implies that the interaction term between this variable and haplotypes is included in 'haplo.glm' function.

Usage

```
int(x)
```

Arguments

x   A factor variable.

Value

x

See Also

[haplo.interaction](#)

Examples

```
# Not Run
# data(SNPs)
# mod <- haplo.interaction(casco~int(sex)+blood.pre, data=SNPs,
# SNPs.sel=c("snp10001","snp10004","snp10005"))
#
```

---

interactionPval   *Two-dimensional SNP analysis for association studies*

Description

Perform a two-dimensional SNP analysis (interaction) for association studies with possible allowance for covariate

Usage

```
interactionPval(formula, data, quantitative =
          is.quantitative(formula, data), model = "codominant")
```

Arguments

formula   a formula object. It might have either a continuous variable (quantitative traits)
                  or a factor variable (case-control study) as the response on the left of the ~ op-
                  erator and the terms corresponding to the covariates to be adjusted. A crude
                  analysis is performed indicating ~1

data   a required object of class 'setupSNP'.

quantitative   logical value indicating whether the phenotype (those which is in the left of the
                  operator ~ in 'formula' argument) is quantitative. The function 'is.quantitative'
                  returns FALSE when the phenotype is a variable with two categories (i.e. in-
                  dicating case-control status). Thus, it is not a required argument but it may be
                  modified by the user.

model   a character string specifying the type of genetic model (mode of inheritance).
                  This indicates how the genotypes should be collapsed. Possible value are "codom-
                  inant", "dominant", "recessive", "overdominant" or "log-additive". The default
                  is "codominant". Only the first words are required, e.g "co", "do", "re", "ov",
                  "log"

Details

The 'interactionPval' function calculates, for each pair of SNPs (i,j), the likelihood underling the
null model L0, the likelihood under each of the single-SNP, L(i) and L(j), the likelihood under
an additive SNP model La(i,j), and the likelihood under a full SNP model (including SNP-SNP
interaction), Lf(i,j).

The upper triangle in matrix from this function contains the p values for the interaction (epistasis)
log-likelihood ratio test, LRT, LRTij = -2 (log Lf(i,j) - log La(i,j))

The diagonal contains the p values from LRT for the crude effect of each SNP, LRTii = -2 (log L(i)
- log L0)

The lower triangle contains the p values from LRT comparing the two-SNP additive likelihood to
the best of the single-SNP models, LRTji = -2 (log La(i,j) - log max(L(i),L(j)))

In all cases the models including the SNPs are adjusted by the covariates indicated in the 'formula'
argument. This method is used either for quantitative traits and dicotomous variables (case-control
studies).

Value

The 'interactionPval' function returns a matrix of class 'SNPinteraction' containing the p values
corresponding to the different likelihood ratio tests above describe.

Methods defined for 'SNPinteraction' objects are provided for print and plot. The plot method uses
'image' to plot a grid of p values. The upper triangle contains the interaction (epistasis) p values
from LRT. The content in the lower triangle is the p values from the LRT comparing the additive
model with the best single model. The diagonal contains the main effects pvalues from LRT. The
'plot.SNPinteraction' function also allows the user to plot the SNPs sorted by genomic position and
with the information about chromosomes as in the 'plotMissing' function.

Note

two-dimensional SNP analysis on a dense grid can take a great deal of computer time and memory.

References

JR Gonzalez, L Armengol, X Sole, E Guino, JM Mercader, X Estivill, V Moreno. SNPassoc: an R package to perform whole genome association studies. Bioinformatics, 2007;23(5):654-5.

See Also

setupSNP

Examples

```
data(SNPs)
datSNP<-setupSNP(SNPs,6:40,sep="")

ansCod<-interactionPval(log(protein)~sex,datSNP)
print(ansCod)
plot(ansCod)
```

---

intervals	*Print ORs and 95% confidence intervals for an object of class 'haplo.glm'*

Description

Print ORs and confidence intervals for an object of class 'haplo.glm'

Usage

```
intervals(o, level=.95, ...)
```

Arguments

o	object of class 'haplo.glm'
level	significance level. Default is 95 percent
...	other arguments

Examples

```
# Not Run
# data(SNPs)
# tag.SNPs<-c("snp100019","snp10001","snp100029")
# geno<-make.geno(SNPs,tag.SNPs)

# mod<-haplo.glm(casco~geno,data=SNPs,
#      family=binomial,
# locus.label=tag.SNPs,
# allele.lev=attributes(geno)$unique.alleles,
```

```
# control = haplo.glm.control(haplo.freq.min=0.05))

# intervals(mod)
# summary(mod)
```

LD   *max-statistic for a 2x3 table*

Description

Compute pairwise linkage disequilibrium between genetic markers

Usage

```
LD(g1, ...)

## S3 method for class 'snp'
LD(g1, g2, ...)

## S3 method for class 'setupSNP'
LD(g1, SNPs, ...)


LDplot(x, digits = 3, marker, distance, which = c("D", "D'",
    "r", "X^2", "P-value", "n", " "), ...)

LDtable(x, colorcut = c(0, 0.01, 0.025, 0.05, 0.1, 1),
        colors = heat.colors(length(colorcut)),
        textcol = "black", digits = 3, show.all = FALSE,
        which = c("D", "D'", "r", "X^2", "P-value", "n"),
        colorize = "P-value", cex, ...)
```

Arguments

g1	genotype object or dataframe containing genotype objects
g2	genotype object (ignored if g1 is a dataframe)
SNPs	columns containing SNPs
x	LD or LD.data.frame object
digits	Number of significant digits to display
which	Name(s) of LD information items to be displayed
colorcut	P-value cutoffs points for colorizing LDtable
colors	Colors for each P-value cutoff given in 'colorcut' for LDtable

textcol	Color for text labels for LDtable
marker	Marker used as 'comparator' on LDplot. If omitted separate lines for each marker will be displayed
distance	Marker location, used for locating of markers on LDplot.
show.all	If TRUE, show all rows/columns of matrix. Otherwise omit completely blank rows/columns.
colorize	LD parameter used for determining table cell colors
cex	Scaling factor for table text. If absent, text will be scaled to fit within the table cells.
...	Optional arguments ('plot.LD.data.frame' passes these to 'LDtable' and 'LDplot').

Value

None

Author(s)

functions adapted from LD, LDtable and LDplot in package genetics by Gregory Warnes et al. (warnes@bst.rochester.edu)

References

genetics R package by Gregory Warnes et al. (warnes@bst.rochester.edu)

See Also

[setupSNP](#) [snp](#)

---

make.geno	*Create a group of locus objects from some SNPs, assign to 'model.matrix' class.*

Description

This function prepares the CRITICAL element corresponding to matrix of genotypes necessary to be included in 'haplo.glm' function.

Usage

```
make.geno(data, SNPs.sel)
```

Arguments

data	an object of class 'setupSNP' containing the the SNPs that will be used to estimate the haplotypes.
SNPs.sel	a vector indicating the names of SNPs that are used to estimate the haplotypes

Value

the same as 'setupGeno' function, from 'haplo.stats' library, returns

See Also

[snp](#)

Examples

```
## Not run:
data(SNPs)
# first, we create an object of class 'setupSNP'
datSNP<-setupSNP(SNPs,6:40,sep="")
geno<-make.geno(datSNP,c("snp10001","snp10002","snp10003"))
## End(Not run)
```

---

maxstat   *max-statistic for a 2x3 table*

Description

Computes the asymptotic p-value for max-statistic for a 2x3 table

Usage

```
maxstat(x, ...)

## Default S3 method:
maxstat(x, y, ...)

## S3 method for class 'table'
maxstat(x, ...)

## S3 method for class 'setupSNP'
maxstat(x, y, colSNPs=attr(x,"colSNPs"), ...)

## S3 method for class 'matrix'
maxstat(x, ...)
```

Arguments

x   a numeric matrix with 2 rows (cases/controls) and 3 colums (genotypes) or a
                     vector with case/control status or an object of class 'setupSNP'.

y	an optional numeric vector containing the information for a given SNP. In this case 'x' argument must contain a vector indicarting case/control status. If 'x' argument is an object of class 'setupSNP' this argument migth be the name of the variable containing case/control information.
colSNPs	a vector indicating which columns contain those SNPs to compute max-statistic. By default max-statistic is computed for those SNPs specified when the object of class 'setupSNP' was created.
...	further arguments to be passed to or from methods.

Value

A matrix with the chi-square statistic for dominant, recessive, log-additive and max-statistic and its asymptotic p-value.

References

Gonzalez JR, Carrasco JL, Dudbridge F, Armengol L, Estivill X, Moreno V. Maximizing association statistics over genetic models (2007). Submitted

Sladek R, Rocheleau G, Rung J et al. A genome-wide association study identifies novel risk loci for type 2 diabetes (2007). Nature 445, 881-885

See Also

[setupSNP](#)

Examples

```
# example from Sladek et al. (2007) for the SNP rs1111875
 tt<-matrix(c(77,298,310,122,316,231),nrow=2,ncol=3,byrow=TRUE)
 maxstat(tt)

 data(SNPs)
 maxstat(SNPs$casco,SNPs$snp10001)
 myDat<-setupSNP(SNPs,6:40,sep="")
 maxstat(myDat,casco)
```

---

odds   *Extract odds ratios, 95% CI and pvalues*

Description

Extract odds ratios, 95

Usage

```
odds(x, model=c("log-additive", "dominant", "recessive", "overdominant", "codominant"),
        sorted=c("no","p-value","or"))
```

Arguments

x	an object of class 'WGassociation' output of WGassociation
model	model to be extracted. Only first one is used. The first letter is enough, low or upper case.
sorted	Sort the output by P value or OR.

Value

A matrix with OR 95% CI (lower, upper) and P value for the selected model. For codominant model, the OR and 95%CI are given for heterozygous and homozigous.

References

JR Gonzalez, L Armengol, X Sole, E Guino, JM Mercader, X Estivill, V Moreno. SNPassoc: an R package to perform whole genome association studies. Bioinformatics, 2007;23(5):654-5.

Examples

```
data(SNPs)
datSNP<-setupSNP(SNPs,6:40,sep="")
ans<-WGassociation(casco~1,data=datSNP,model="all")
odds(ans)
```

---

permTest   *Permutation test analysis*

Description

This function extract the p values for permutation approach performed using scanWGassociation function

Usage

```
permTest(x, method="minimum", K)
```

Arguments

x	a required object of class 'WGassociation' with the attribute 'permTest'. See details
method	statistic used in the permutation test. The default is 'minimum' but 'rtp' (rank truncated product) is also available.
K	number of the K most significant p values from the total number of test performed (e.g number of SNPs) used to compute the rank truncated product. This argument is only required when method='rtp'. See references

Details

This function extract the p values from an object of class 'WGassociation'. This object migth be obtained using the funcion called 'scanWGassociation' indicating the number of permutations in the argument 'nperm'.

Value

An object of class 'permTest'.

'print' returns a summary indicating the number of SNPs analyzed, the number of valid SNPs (those non-Monomorphic and that pass the calling rate), the p value after Bonferroni correction, and the p values based on permutation approach. One of them is based on considering the empirical percentil for the minimum p values, and the another one on assuming that the minimum p values follow a beta distribution.

'plot' produces a plot of the empirical distribution for the minimum p values (histogram) and the expected distribution assuming a beta distribution. The corrected p value is also showed in the plot.

See examples for further illustration about all previous issues.

References

Dudbridge F, Gusnanto A and Koeleman BPC. Detecting multiple associations in genome-wide studies. Human Genomics, 2006;2:310-317.

Dudbridge F and Koeleman BPC. Efficient computation of significance levels for multiple associations in large studies of correlated data, including genomewide association studies. Am J Hum Genet, 2004;75:424-435.

JR Gonzalez, L Armengol, X Sole, E Guino, JM Mercader, X Estivill, V Moreno. SNPassoc: an R package to perform whole genome association studies. Bioinformatics, 2007;23(5):654-5.

See Also

scanWGassociation

Examples

```
data(SNPs)
datSNP<-setupSNP(SNPs,6:40,sep="")
ans<-scanWGassociation(casco~1,data=datSNP,model="co",nperm=1000)

# pPerm<-permTest(ans)
# print(pPerm)
# plot(pPerm)
```

---

plot.WGassociation   *Function to plot -log p values from an object of class 'WGassociation'*

Description

Function to plot -log p values from an object of class 'WGassociation'

Usage

```
## S3 method for class 'WGassociation'
plot(x, alpha = 0.05, plot.all.SNPs = FALSE,
  print.label.SNPs = TRUE, cutPval = c(0, 1e-10, 1), whole,
  ylim.sup=ifelse(is.null(attr(x,"fast")),1e-40, 1e-30),
  col.legend = c("red", "gray60"), sort.chromosome=TRUE,
  centromere, ...)
```

Arguments

x	an object of class 'WGassociation'
alpha	statistical significance nominal level. See details
plot.all.SNPs	are all SNPs plotted? If not, neither monomorphic nor SNPs with genotyping problems are plotted. The default is FALSE.
print.label.SNPs	
	are labels of SNPs printed? The default is TRUE
cutPval	when argument 'whole' is TRUE in the 'x' object (e.g. when whole genome analysis is carried out) 'cutPval divides the range of p values into intervals and codes these values according to which interval they fall (like 'cut' function). The default is c(0, 1e-10, 1). That is, the p values are divided in those less than 1e-10 and those greater than 1e-10.
whole	is a whole genome carried out? If TRUE 'dataSNPs.pos' argument in 'setup-SNP' is required.
ylim.sup	superior limit for each panel. This value helps to obtain nicer plots
col.legend	the color of the bar corresponding to p values plotted in each panel. The default is "red" for those p values less than 1e-10 and "gray60" for those greater than 1e-10.
sort.chromosome	
	should chromosome be sorted? Set this argument to FALSE when genomic information corresponds to different genes.
centromere	numeric vector specifying the centromere positions. If missing, the default centromere value of human genome are used.
...	other graphical parameters

Details

A panel with different plots (one for each mode of inheritance) are plotted. Each of them represents the -log(p value) for each SNP. Two horizontal lines are also plotted. One one them indicates the nominal statistical significance level (see 'alpha' argument) whereas the other one indicates the statistical significance level after Bonferroni correction.

A different plot is created when the argument 'whole' the object 'x' is TRUE (see setupSNP function). In that case a plot of p values in the -log sclae is plotted for each SNP and for each chromosome.

Value

No return value, just the plot

References

JR Gonzalez, L Armengol, X Sole, E Guino, JM Mercader, X Estivill, V Moreno. SNPassoc: an R package to perform whole genome association studies. Bioinformatics, 2007;23(5):654-5.

See Also

[association](#) [setupSNP](#) [WGassociation](#)

| plotMissing | *Plot of missing genotypes* |

Description

Plot a grid showing which genotypes are missing

Usage

```
plotMissing(x, print.labels.SNPs = TRUE,
        main = "Genotype missing data", ...)
```

Arguments

x	an object of class 'setupSNP'
print.labels.SNPs	
	should labels of SNPs be printed?
main	title to place on plot
...	extra arguments of 'image' function

Details

This function uses 'image' function to plot a grid with black pixels where the genotypes are missing.

Value

None

See Also

setupSNP

Examples

```
data(SNPs)
data(SNPs.info.pos)
ans<-setupSNP(SNPs,colSNPs=6:40,sep="")
plotMissing(ans)

# The same plot with the SNPs sorted by genomic position and
# showing the information about chromosomes

ans<-setupSNP(SNPs,colSNPs=6:40,sort=TRUE,SNPs.info.pos,sep="")
plotMissing(ans)
```

---

qqpval   *Functions for inspecting population substructure*

Description

This function plots ranked observed p values against the corresponding expected p values in -log scale.

Usage

```
qqpval(p, pch=16, col=4, ...)
```

Arguments

p	a vector of p values
pch	symbol to use for points
col	color for points
...	other plot arguments

Value

A plot

See Also

[GenomicControl,](GenomicControl) WGassociation

Examples

```
data(SNPs)
datSNP<-setupSNP(SNPs,6:40,sep="")
res<-WGassociation(casco,datSNP,model=c("do","re","log-add"))

# observed vs expected p values for recessive model
qqpval(recessive(res))
```

---

scanWGassociation   *Whole genome association analysis*

Description

This function also performs a whole genome association analysis as the function [WGassociation](#) but only computes the p values for each SNP corresponding to likelihood ratio test.

Usage

```
scanWGassociation(formula, data, model = c("all"), nperm,
    quantitative = is.quantitative(formula, data), genotypingRate = 80)
```

Arguments

formula
: either a symbolic description of the model to be fited (a formula object) without the SNP or the name of response variable in the case of fitting single models (e.g. unadjusted models). It might have either a continuous variable (quantitative traits) or a factor variable (case-control studies) as the response on the left of the ~ operator and terms with additional covariates on the right of the ~. Currently only is possible to write ~1 (that is, non-adjusted analysis). See [WGassociation](#) if you are interested in including additional covariates.

data
: a required dataframe of class 'setupSNP' containing the variables in the model and the SNPs

model
: a character string specifying the type of genetic model (mode of inheritance) for the SNP. This indicates how the genotypes should be collapsed. Possible values are "codominant", "dominant", "recessive", "overdominant", "log-additive" or "all". The default is "all" that fits the 5 possible genetic models. Only the first words are required, e.g "co", "do", etc.

nperm
: number of permutations to simulate the null hypotesis (e.g OR=1), conditioning on the empirical correlation structure. Only required to perform a permutation test. Currently this test is only available for binary traits.

quantitative
: logical value indicating whether the phenotype (that which is in the left of the operator ~ in 'formula' argument) is quantitative. The function 'is.quantitative' returns FALSE when the phenotype is a variable with two categories (i.e. indicating case-control status). Thus, it is not a required argument but it may be modified by the user.

> genotypingRate   minimum percentage of genotype rate for a given SNP to be included in the analysis. Default is 80%.

Value

An object of class 'WGassociation'.

The function 'print' is used to print the results. The p values are saved in the attribute 'pvalues' as a matrix. They may be obtained using attr(,"pvalues") (see examples). The first column indicates whether a problem with genotyping is present.

The function 'plot' is used to obtain a plot of p values in the -log scale. See plot.WGassociation for further details

References

JR Gonzalez, L Armengol, X Sole, E Guino, JM Mercader, X Estivill, V Moreno. SNPassoc: an R package to perform whole genome association studies. Bioinformatics, 2007;23(5):654-5.

See Also

[WGassociation](#) [getSignificantSNPs](#) [association](#) [setupSNP](#) [plot.WGassociation](#) [permTest](#)

Examples

```
# Next steps may be very time consuming. So they are not executed

#data(HapMap)
#myDat<-setupSNP(HapMap, colSNPs=3:9307, sort = TRUE,
#    info=HapMap.SNPs.pos, sep="")
#resHapMap<-scanWGassociation(group~1, data=myDat, model="log")
```

---

setupSNP   *Convert columns in a dataframe to class 'snp'*

Description

setupSNP Convert columns in a dataframe to class 'snp'

summary.setupSNP gives a summary for an object of class 'setupSNP' including allele names, major allele frequencie, an exact thest of Hardy-Weinberg equilibrium and percentage of missing genotypes

Usage

```
setupSNP(data, colSNPs, sort = FALSE, info, sep = "/", ...)
```

Arguments

`data`	dataframe containing columns with the SNPs to be converted
`colSNPs`	Vector specifying which columns contain SNPs data
`sort`	should SNPs be sorted. Default is FALSE
`info`	if sort is TRUE a dataframe containing information about the SNPs regarding their genomic position and the gene where they are located
`sep`	character separator used to divide alleles in the genotypes
`...`	optional arguments

Value

a dataframe of class 'setupSNP' containing converted SNP variables. All other variables will be unchanged.

References

JR Gonzalez, L Armengol, X Sole, E Guino, JM Mercader, X Estivill, V Moreno. SNPassoc: an R package to perform whole genome association studies. Bioinformatics, 2007;23(5):654-5.

See Also

[snp](snp)

Examples

```
 data(SNPs)
 myDat<-setupSNP(SNPs,6:40,sep="")


#sorted SNPs and having genomic information
 data(SNPs.info.pos)
 myDat.o<-setupSNP(SNPs,6:40,sep="",sort=TRUE, info=SNPs.info.pos)

# summary
 summary(myDat.o)

# plot one SNP
  plot(myDat,which=2)
```

---

snp   *SNP object*

Description

snp creates an snp object

is returns TRUE if x is of class 'snp'

as attempts to coerce its argument into an object of class 'snp'

reorder change the reference genotype

summary gives a summary for an object of class 'snp' including genotype and allele frequencies and an exact thest of Hardy-Weinberg equilibrium

plot gives a summary for an object of class 'snp' including genotype and allele frequencies and an exact thest of Hardy-Weinberg equilibrium in a plot. Barplot or pie are allowed

[.snp is a copy of [.factor modified to preserve all attributes

Usage

```
snp(x, sep = "/", name.genotypes, reorder="common",
  remove.spaces = TRUE, allow.partial.missing = FALSE)

is.snp(x)

as.snp(x, ...)

## S3 method for class 'snp'
additive(o)
```

Arguments

x	either an object of class 'snp' or an object to be converted to class 'snp'
sep	character separator used to divide alleles when x is a vector of strings where each string holds both alleles. The default is "/". See below for details.
name.genotypes	the codes for the genotypes. This argument may be useful when genotypes are coded using three different codes (e.g., 0,1,2 or hom1, het, hom2)
reorder	how should genotypes within an individual be reordered. Possible values are 'common' or 'minor'. The default is reorder="common". In that case, alleles are sorted within each individual by common homozygous.
remove.spaces	logical indicating whether spaces and tabs will be removed from the genotypes before processing
allow.partial.missing	
	logical indicating whether one allele is permitted to be missing. When set to 'FALSE' both alleles are set to 'NA' when either is missing.

o   an object of class 'snp' to be coded as a linear covariate: 0,1,2

...   optional arguments

Details

SNP objects hold information on which gene or marker alleles were observed for different individuals. For each individual, two alleles are recorded.

The snp class considers the stored alleles to be unordered , i.e., "C/T" is equivalent to "T/C". It assumes that the order of the alleles is not important.

When snp is called, x is a character vector, and it is assumed that each element encodes both alleles. In this case, if sep is a character string, x is assumed to be coded as "Allele1<sep>Allele2". If sep is a numeric value, it is assumed that character locations 1:sep contain allele 1 and that remaining locations contain allele 2.

additive.snp recodes the SNPs for being analyzed as a linear covariate (codes 0,1,2)

Value

The snp class extends "factor" where the levels is a character vector of possible genotype values stored coded by paste(allele1, "", allele2, sep="/")

References

JR Gonzalez, L Armengol, X Sole, E Guino, JM Mercader, X Estivill, V Moreno. SNPassoc: an R package to perform whole genome association studies. Bioinformatics, 2007;23(5):654-5.

See Also

association

Examples

```
# some examples of snp data in different formats

dat1  <- c("21", "21", "11", "22", "21",
                    "22", "22", "11", "11", NA)
ans1  <- snp(dat1,sep="")
ans1

dat2 <- c("A/A","A/G","G/G","A/G","G/G",
                    "A/A","A/A","G/G",NA)
ans2  <- snp(dat2,sep="/")
ans2

dat3 <- c("C-C","C-T","C-C","T-T","C-C",
                    "C-C","C-C","C-C","T-T",NA)
ans3 <- snp(dat3,sep="-")
ans3


dat4 <- c("het","het","het","hom1","hom2",
```

```
                      "het","het","hom1","hom1",NA)
ans4 <- snp(dat4,name.genotypes=c("hom1","het","hom2"))
ans4


# summary
summary(ans3)

# plots

plot(ans3)
plot(ans3,type=pie)
plot(ans3,type=pie,label="SNP 10045")
```

SNPs   *SNPs in a case-control study*

Description

SNPs data.frame contains selected SNPs and other clinical covariates for cases and controls in a case-control study

SNPs.info.pos data.frame contains the names of the SNPs included in the data set 'SNPs' including their chromosome and their genomic position

Usage

```
data(SNPs)
```

Format

'SNPs' data.frame contains the following columns:

id	identifier of each subject
casco	case or control status: 0-control, 1-case
sex	gender: Male and Female
blood.pre	arterial blood presure
protein	protein levels
snp10001	SNP 1
snp10002	SNP 2
...	...
snp100036	SNP 36

'SNPs.info.pos' data.frame contains the following columns: A data frame with 35 observations on the following 3 variables.

snp name of SNP

chr name of chromosome

pos genomic position

Source

Data obtained from our department. The reference and details will be supplied after being published.

---

sortSNPs   *Sort a vector of SNPs by genomic position*

Description

This function sorts a vector with the position of SNPs in a data frame using another data frame which contains information about SNPs, their chromosome, and their genomic position

Usage

```
sortSNPs(data, colSNPs, info)
```

Arguments

data	a required data frame with the SNPs
colSNPs	a required vector indicating which columns of 'data' contains genotype data
info	a required data frame with genomic information for the SNPs (chromosome and position). The first column must have the SNPs, the second one the chromosome and the third one the genomic position.

Details

First of all, the function obtains a vector with the SNPs sorted using the data frame with the genomic positions (see 'dataSNPs.pos' argument). Then, the columns which indicate where the information about the genotypes is in our data frame, are sorted using this vector.

This information is useful when [WGassociation](#) function is called since it allow the user to summaryze the results with the SNPs sorted by genomic position

Value

a vector indicating the colums where the SNPs are recorded in our data frame ('data' argument), sorted using the genomic positions listed in 'dataSNPs.pos' argument)

Examples

```
#
# data(SNPs)
# data(SNPs.info.pos)
# colSNPs.order<-sortSNPs(SNPs,c(6:40),SNPs.info.pos)
#
```

Table.mean.se *Descriptive sample size, mean, and standard error*

Description

This function computes sample size, mean and standard error of a quantitative trait for each genotype (or combination of genotypes)

Usage

```
Table.mean.se(var, dep, subset = !is.na(var))
```

Arguments

var   quantitative trait

dep   variable with genotypes or any combination of them

subset   an optional vector specifying a subset of observations to be used in the descriptive analysis

Value

tp   A matrix giving sample size (n), median (me) and standard error (se) for each genotype

See Also

Table.N.Per

Examples

```
data(SNPs)
# sample size, mean age and standard error for each genotype
#  Table.mean.se(SNPs$snp10001,SNPs$protein)

# The same table for a subset (males)
#  Table.mean.se(SNPs$snp10001,SNPs$protein,SNPs$sex=="Male")

# The same table assuming a dominant model
#  Table.mean.se(dominant(snp(SNPs$snp10001,sep="")),SNPs$protein,SNPs$sex=="Male")
```

Table.N.Per   *Descriptive sample size and percentage*

Description

This function computes sample size and percentage for each category of a categorical trait (e.g. case-control status) for each genotype (or combination of genotypes).

Usage

```
Table.N.Per(var, dep, subset = !is.na(var))
```

Arguments

var   categorical trait.

dep   variable with genotypes or any combination of them

subset   an optional vector specifying a subset of observations to be used in the descrip-
                    tive analysis.

Value

tp   A matrix giving sample size (n),and the percentage (for each genotype

See Also

Table.mean.se

Examples

```
data(SNPs)
#sample size and percentage of cases and controls for each genotype
#  Table.N.Per(SNPs$snp10001,SNPs$casco)

# The same table for a subset (males)
#  Table.N.Per(SNPs$snp10001,SNPs$casco,SNPs$sex=="Male")

# The same table assuming a dominant model
#  Table.N.Per(dominant(snp(SNPs$snp10001,sep="")),SNPs$casco,SNPs$sex=="Male")
```

---

tableHWE   *Test for Hardy-Weinberg Equilibrium*

Description

Test the null hypothesis that Hardy-Weinberg equilibrium holds in cases, controls and both populations.

`print` print the information. Number of digits, and significance level can be changed

Usage

```
tableHWE(x, strata, ...)
```

Arguments

x	an object of class 'setupSNP'
strata	a factor variable for a stratified analysis
...	optional arguments

Details

This function calculates the HWE test for those variables of class 'snp' in the object x of class 'setupSNP'

Value

A matrix with p values for Hardy-Weinberg Equilibrium

Author(s)

This function is based on an R function which computes an exact SNP test of Hardy-Weinberg Equilibrium written by Wigginton JE, Cutler DJ and Abecasis GR available at http://www.sph.umich.edu/csg/abecasis/Exact/r_instruct.html

References

Wigginton JE, Cutler DJ and Abecasis GR (2005). A note on exact tests of Hardy-Weinberg equilibrium. Am J Hum Genet 76:887-93

See Also

setupSNP

Examples

```
data(SNPs)
ans<-setupSNP(SNPs,6:40,sep="")
res<-tableHWE(ans)
print(res)
#change the significance level showed in the flag column
print(res,sig=0.001)


#stratified analysis
res<-tableHWE(ans,ans$sex)
print(res)
```

WGassociation   *Whole genome association analysis*

Description

This function carries out a whole genome association analysis between the SNPs and a dependent
variable (phenotype) under five different genetic models (inheritance patterns): codominant, domi-
nant, recessive, overdominant and log-additive. The phenotype may be quantitative or categorical.
In the second case (e.g. case-control studies) this variable must be of class 'factor' with two levels.

Usage

```
WGassociation(formula, data, model = c("all"),
              quantitative = is.quantitative(formula, data),
              genotypingRate = 80, level = 0.95, ...)
```

Arguments

formula   either a symbolic description of the model to be fited (a formula object) without
                 the SNP or the name of response variable in the case of fitting single models (e.g.
                 unadjusted models). It might have either a continuous variable (quantitative
                 traits) or a factor variable (case-control studies) as the response on the left of
                 the ~ operator and terms with additional covariates on the right of the ~ operator
                 may be added to fit an adjusted model (e.g., ~var1+var2+...+varN+SNP). See
                 details

data   a required dataframe of class 'setupSNP' containing the variables in the model
                 and the SNPs

model   a character string specifying the type of genetic model (mode of inheritance) for
                 the SNP. This indicates how the genotypes should be collapsed. Possible values
                 are "codominant", "dominant", "recessive", "overdominant", "log-additive" or
                 "all". The default is "all" that fits the 5 possible genetic models. Only the first
                 words are required, e.g "co", "do", etc.

quantitative   logical value indicating whether the phenotype (that which is in the left of the
                operator ~ in 'formula' argument) is quantitative. The function 'is.quantitative'
                returns FALSE when the phenotype is a variable with two categories (i.e. in-
                dicating case-control status). Thus, it is not a required argument but it may be
                modified by the user.

genotypingRate minimum percentage of genotype rate for a given SNP to be included in the
                analysis. Default is 80%.

level   signification level for confidence intervals. Defaul 95%.

...   Other arguments to be passed through glm function

Details

This function assesses the association between the response variable included in the left side in the
'formula' and the SNPs included in the 'data' argument adjusted by those variables included in the
right side of the 'formula'. Different genetic models may be analyzed using 'model' argument.

Value

An object of class 'WGassociation'.

'summary' returns a summary table by groups defined in info (genes/chromosomes).

'WGstats' returns a detailed output, similar to the produced by association.

'pvalues' and 'print' return a table of p-values for each genetic model for each SNP. The first column
indicates whether a problem with genotyping is present.

'plot' produces a plot of p values in the -log scale. See plot.WGassociation for further details.

'labels' returns the names of the SNPs analyzed.

The functions 'codominat', 'dominant', 'recessive', 'overdominant' and 'additive' are used to obtain
the p values under these genetic models.

See examples for further illustration about all previous issues.

References

JR Gonzalez, L Armengol, X Sole, E Guino, JM Mercader, X Estivill, V Moreno. SNPassoc: an R
package to perform whole genome association studies. Bioinformatics, 2007;23(5):654-5.

See Also

scanWGassociation getSignificantSNPs association WGstats setupSNP plot.WGassociation

Examples

```
data(SNPs)
datSNP<-setupSNP(SNPs,6:40,sep="")
ansAll<-WGassociation(protein~1,data=datSNP,model="all")

# In that case the formula is not required. You can also write:
# ansAll<-WGassociation(protein,data=datSNP,model="all")
```

```
#only codominant and log-additive
ansCoAd<-WGassociation(protein~1,data=datSNP,model=c("co","log-add"))

#for printing p values
print(ansAll)
print(ansCoAd)

#for obtaining a matrix with the p palues
pvalAll<-pvalues(ansAll)
pvalCoAd<-pvalues(ansCoAd)

# when all models are fitted and we are interested in obtaining
# p values for different genetic models

# codominant model
pvalCod<-codominant(ansAll)

# recessive model
pvalRec<-recessive(ansAll)

# and the same for additive, dominant or overdominant


#summary
summary(ansAll)

#for a detailed report
WGstats(ansAll)

#for plotting the p values
plot(ansAll)


#
# Whole genome analysis
#

data(HapMap)
# Next steps may be very time consuming. So they are not executed

#myDat<-setupSNP(HapMap, colSNPs=3:9809, sort = TRUE,
#    info=HapMap.SNPs.pos, sep="")
#resHapMap<-WGassociation(group~1, data=myDat, model="log")


# However, the results are saved in the object "resHapMap"
# to illustrate print, summary and plot functions
summary(resHapMap)
plot(resHapMap)
print(resHapMap)
```

Index