Package 'QTLRel'

February 19, 2020

Version 1.6

Date 2020-2-18

Title Tools for Mapping of Quantitative Traits of Genetically Related Individuals and Calculating Identity Coefficients from Pedigrees

Author Riyan Cheng [aut, cre]

Maintainer Riyan Cheng <riyancheng@hotmail.com>

Description This software provides tools for quantitative trait mapping in populations such as advanced intercross lines where relatedness among individuals should not be ignored. It can estimate background genetic variance components, impute missing genotypes, simulate genotypes, perform a genome scan for putative quantitative trait loci (QTL), and plot mapping results. It also has functions to calculate identity coefficients from pedigrees, especially suitable for pedigrees that consist of a large number of generations, or estimate identity coefficients from genotypic data in certain circumstances.

Depends R (>= 2.10)

Imports gdata, graphics, grDevices, lattice, stats

Suggests qtl

LazyLoad yes

LazyData no

License GPL (>= 2)

NeedsCompilation yes

Repository CRAN

Date/Publication 2020-02-19 05:20:02 UTC

R topics documented:

`aicVC` *AIC Model Selection*

Description

Select genetic variance components via Akaike's information criterion (AIC).

Usage

```
aicVC(y, x, v = list(E=diag(length(y))), initpar, k = 2, init = 1, keep = 1,
   direction = c("forward", "backward"), nit = 25, msg = FALSE,
   control = list(), hessian = FALSE)
```

Arguments

`y`	A numeric vector or a numeric matrix of one column (representing a phenotype for instance).
`x`	A data frame or matrix, representing covariates if not missing.
`v`	A list of variance components of interest. Note: `E` is reserved for residual (or environmental) variance and can be missed in `v`; it is considered to be an identify matirx if it is specified. `v` can be provided as a single matrix.

initpar	Optional initial parameter values.
k	Penalty on a parameter. The selection criterion is the known "AIC" if k = 2 and is "BIC" if k = log(n) where "n" is the sample size.
init	Indicates which variance components for the initial model. By default, E is included if it is missing in v.
keep	Indicator of which variance components should be forced into the final model. By default, E is kept in the final model if it is not specified in v.
direction	The mode of search. Either "forward" or "backward" with default "forward".
nit	Maximum number of iterations for optimization. Ignored if there are not more than two variance components.
msg	A logical variable. True if one wants to track the process for monitoring purpose.
control	A list of control parameters to be passed to optim.
hessian	Logical. Should a numerically differentiated Hessian matrix be returned?

Details

In genome-wide association studies (GWAS), random effects are usually added to a model to account for polygenic variation. Abney et al (2000) showed that five variance components including the most interesting additive and dominance variance components are potentially induced by polygenes. The above function is intended for selecting variance components that contribute "most" to a quantitative trait.

Function estVC is called by the above function to estimate the parameters and maximum likelihood in each model. Refer to estVC for more information.

Value

aic	AIC of the final model.
model	Gives parameter estimates, log-likihood, and other information.
lik	Log-likelihood of the model selected at each intermediate step.
trace	Indicates which variance components were selected at each intermediate step.

See Also

estVC for more information.

Examples

```
data(miscEx)

## Not run:
# forward selection
# any variance component will be selected
# if AIC improve by 1e-5 or larger
pheno<- pdatF8[!is.na(pdatF8$bwt) & !is.na(pdatF8$sex),]
ii<- match(rownames(pheno), rownames(gmF8$AA))
v<- list(A=gmF8$AA[ii,ii], D=gmF8$DD[ii,ii])
```

```
o<- aicVC(y=pheno$bwt, x=pheno$sex, k=0, v=v, msg=TRUE)
o

# forward selection
of<- aicVC(y=pheno$bwt, x=pheno$sex, v=v, k=1/2,
direction="for", msg=TRUE)
of

# backward elimination
ob<- aicVC(y=pheno$bwt, x=pheno$sex, v=v, k=1/2, init=1:2,
direction="back", msg=TRUE)
ob

## End(Not run)
```

`blup` *Best Linear Unbiased Prediction*

Description

Estimate the best linear unbiased prediction (BLUP) for various effects in the model.

Usage

```
blup(object)
```

Arguments

`object`	An object from `estVC` or `aicVC`.

Value

`fixed`	BLUP for fixed effects.
`R, etc.`	BLUP for random effects.

See Also

`estVC` and `aicVC`.

Examples

```
data(miscEx)

## Not run:
# only consider additive genetic variance component
pheno<- pdatF8[!is.na(pdatF8$bwt) & !is.na(pdatF8$sex),]
ii<- match(rownames(pheno), rownames(gmF8$AA))
v<- list(A=gmF8$AA[ii,ii],D=gmF8$DD[ii,ii])
```

```
vc<- estVC(y=pheno$bwt, x=pheno$sex, v=v)
b<- blup(vc)

## End(Not run)
```

`cic` *Calculate Jacquard condensed identity coefficients*

Description

Calculate Jacquard condensed identity coefficients from a pedigree.

Usage

```
cic(ped, ids, inter, df=3, ask = FALSE, msg = FALSE)
```

Arguments

`ped`	A pedigree, which is a data frame (id, father/sire, mother/dam, ...). If given, "generation" can be numeric 0, 1, 2, ... or non-numeric "F0", "F1", "F2", ... If "sex" is included, male should be "M", "Male" or 1, and female should be "F", "Female" or 2 (other than 0 and 1). If a founder is inbred, its ID should be tagged by character 'i' (e.g. 1i, 2i, etc.). Note: 0 is reserved for unknown `father`, `mother` or `sex`.
`ids`	IDs of the individuals for which to calculate the Jacquard condensed identity coefficients. If missing, all individuals in the pedigree `ped` will be considered.
`inter`	Intermediate generations, if given, where coefficients are calculated bottom-up.
`df`	If `inter` is missing, `df` is used to derive (optimal) `inter`. If `df = 0`, then there will no intermediate generations. If `df` is large (and free disk space is sufficient), then all generations will be used as intermediate generations.
`ask`	If true, users will be asked whether to proceed.
`msg`	If true, will print out some messages.

Details

The coefficients will be calculated for individuals with IDs specified by `ids`. All individuals will be considered if `ids` is missing. This is not recommended if the total number of individuals in the pedigree is large. Instead, it is recommended that `ids` is specified for interested individuals only

`df` is a tuning parameter. It should not be 0 (or smaller than 1) if the pedigree is large in depth (many generations) but the number of individuals is not small; otherwise, it can take forever to finish. It should not be `Inf` (or a large number) if the number of individuals in certain intermediate generation is very large.

Any individual without parent information is regarded as diallelic with two independent alleles. Users can add to their pedigree (e.g. 50 generations of selfing) if founders are inbred.

Value

A matrix G with G[,j] being the j-th Jacquard identity coefficients.

Note

You may need the administrative privilege to run this function on systems such as Windows 7. It may require your operating system support "long long" integer type in C++. If you run this function in a windows system, make sure the working directory is under system volume C and you have the write privilege.

It is better to remove the working directory if the program is interrupted by external forces (e.g. killed by users).

Warning: you may need to run this program on a 64-bit machine in case of seeing such a message!

References

Abney, M., M. S. McPeek, and C. Ober (2000). Estimation of variance components of quantitative traits in inbred populations. Am. J. Hum. Genet. 141, 629-650.

See Also

pedRecode for more information.

Examples

```
data(miscEx)

ids<- sample(pedF8$id[300:500],20)

## Not run:
# run 'cic' for the sampled individuals
# top-down
oo<- cic(pedF8, ids=ids, df=Inf, msg=TRUE)
# bottom-up
o1<- cic(pedF8, ids=ids, df=0, msg=TRUE)
# hybrid of top-down and bottom-up
o2<- cic(pedF8, ids=ids, ask=TRUE, msg=TRUE)
# same results
c(sum(abs(oo-o1) >1e-7),sum(abs(o2-o1) >1e-7))

## End(Not run)
```

---

eigen.sym *Spectral decomposition of a matrix*

Description

Computes eigenvalues and eigenvectors of real symmetric matrices.

Usage

```
eigen.sym(x)
```

Arguments

x	A real symmetric matrix.

Details

This is to use the LAPACK routine 'DSYEVR' to perform spectral decomposition.

Value

values	a vector containing the eigenvalues of x, sorted in decreasing order.
vectors	a matrix whose columns contain the eigenvectors of x, corresponding to eigenvalues.

Note

Warning: symmetry is not checked by the program!

See Also

eigen for more information.

---

estVC *Estimate Variance Component Parameters*

Description

Estimate model parameters for covariates, genetic variance components and residual effect.

Usage

```
estVC(y, x, v = list(E=diag(length(y))), initpar, nit = 25,
   control = list(), hessian = FALSE)
```

Arguments

y	A numeric vector or a numeric matrix of one column (representing a phenotype for instance).
x	A data frame or matrix, representing covariates if not missing.
v	A list of matrices representing variance components of interest. Note: E is reserved for residual (or environmental) variance and can be missed in v; it is considered to be an identify matirx if it is missing. v can be provided as a single matrix, representing a variance component other than E.

initpar	Optional initial parameter values.
nit	Maximum number of iterations for optimization. Ignored if there are not more than two variance components.
control	A list of control parameters to be passed to optim.
hessian	Logical. Should a numerically differentiated Hessian matrix be returned?

Details

The optimization function optim is adopted in the above function to estimate the parameters and maximum likelihood. Several optimization methods are available for the optimization algorithm in optim, but we recommend "Nelder-Mead" for the sake of stability. Alternatively, one may choose other options, e.g., "BFGS" to initialize and speed up the estimation procedure and then the procedure will automatically turn to "Nelder-Mead" for final results.

Normality is assumed for the random effects. Input data should be free of missing values.

Value

par	estimates of the model parameters.
value	log-likelihood of the model.
y	y used.
x	associated with x used.
v	variance component matrices v used.
...	other information.

Note

Hessian matrix, if requested, pertains to -log-likelihood function.

See Also

optim and rem.

Examples

```
data(miscEx)

## Not run:
# no sex effect
pheno<- pdatF8[!is.na(pdatF8$bwt) & !is.na(pdatF8$sex),]
ii<- match(rownames(pheno), rownames(gmF8$AA))
v<- list(A=gmF8$AA[ii,ii], D=gmF8$DD[ii,ii])

o<- estVC(y=pheno$bwt, v=v)
o

# sex as fixed effect
fo<- estVC(y=pheno$bwt, x=pheno$sex, v=v)
fo
```

```
2*(fo$value-o$value) # log-likelihood test statistic

# sex as random effect
SM<- rem(~sex, data=pheno)
ro<- estVC(y=pheno$bwt, v=c(v,list(Sex=SM$sex)))
ro
2*(ro$value-o$value) # log-likelihood test statistic

## End(Not run)
```

---

genMatrix *Derive genetic matrices*

Description

Derive genetic matrices from Jacquard condensed identity coefficients or genotypic data.

Usage

```
genMatrix(x)
```

Arguments

x	An object of cic or ibs, or genotypic data in a matrix or a data frame with each row representing an individual and each column a marker locus and entry being "AA", "AB", "BB" (or 1, 2, 3) without missing genotypes.

Value

AA	Additive genetic matrix.
DD	Dominance genetic matrix.
AD,HH,MH	Other three genetic matrices (see Abney et. al. 2000).
ib	Inbreeding coefficients.

References

Abney, M., M. S. McPeek, and C. Ober (2000). Estimation of variance components of quantitative traits in inbred populations. Am. J. Hum. Genet. 141, 629-650.

See Also

cic

Examples

```
data(miscEx)

ids<- sample(pedF8$id[300:500],20)

## Not run:
# get condensed identity coefficients
oo<- cic(pedF8, ids=ids, df=0)
ksp<- kinship(pedF8, ids=ids) # kinship coefficients only
# extract genetic matrices
gm<- genMatrix(oo)
sum((gm$AA-2*ksp)>1e-7) # same results

## End(Not run)
```

`genoImpute` *Impute Genotypic Data*

Description

Impute missing genotypic data in advance intercross lines (AIL).

Usage

```
genoImpute(gdat, gmap, prd = NULL, step = Inf, gr = 2, pos = NULL,
   method = c("Haldane", "Kosambi"), na.str = "NA", msg = FALSE)
```

Arguments

`gdat`	Genotype data. Should be a matrix or a data frame, with each row representing an observation and each column a marker locus. The column names should be marker names. Genotypes can be 1, 2 and 3, or "AA", "AB" and "BB". Optional if an object `prd` from `genoProb` is used as an argument.
`gmap`	A genetic map. Should be data frame (snp, chr, dist,...), where "snp" is the SNP (marker) name, "chr" is the chromosome where the "snp" is, and "dist" is the genetic distance in centi-Morgan (cM) from the left of the chromosome.
`prd`	An object from `genoProb` if not `NULL`. See "details" for more information.
`step`	The maximum distance (in cM) between two adjacent loci for which the probabilities are calculated. The distance corresponds to the "cumulative" recombination rate at `gr`-th generation.
`gr`	The generation under consideration.
`pos`	Data frame (chr, dist, snp, ...). If given, `step` will be ignored.
`method`	Whether "Haldane" or "Kosambi" mapping function should be used.
`na.str`	String for missing values.
`msg`	A logical variable. If TRUE, certain information will be printed out during calculation.

Details

The missing genotypic value is randomly assigned with a probability conditional on the genotypes of the flanking SNPs (makers).

An object, `prd`, from `genoProb` alone can be used for the purpose of imputation. Then, the output (especially the putative loci) will be determined by `prd`. Optionally, it can be used together with `gdat` so that missing values in `gdat` will be imputed if possible, depending on whether loci in the columns of `gdat` can be identified in the third dimension of `prd`; this won't change the original genotypic data. See examples.

Value

A matrix with the number of rows being the same as `gdat` and with the number of columns depending on the SNP set in both `gdat` and `gmap` and the `step` length.

Note

Currently only suitable for advanced intercross lines.

See Also

`genoProb`

Examples

```
data(miscEx)

# briefly look at genotype data
sum(is.na(gdatF8))
gdatF8[1:5,1:5]

## Not run:
# run 'genoProb'
gdtmp<- gdatF8
   gdtmp<- replace(gdtmp,is.na(gdtmp),0)
prDat<- genoProb(gdat=gdtmp, gmap=gmapF8, step=Inf,
   gr=8, method="Haldane", msg=TRUE)

# imputation based on 'genoProb' object
tmp<- genoImpute(prd=prDat)
sum(is.na(tmp))
tmp[1:5,1:5]

# imputation based on both genotype data and 'genoProb' object
tmp<- genoImpute(gdatF8, prd=prDat)
sum(is.na(tmp))
tmp[1:5,1:5]

# imputation based on genotype data
tmp<- genoImpute(gdatF8, gmap=gmapF8, step=Inf,
   gr=8, na.str=NA)
sum(is.na(tmp))
```

```
tmp[1:5, 1:5]
# set "msg=TRUE" for more information
tmp<- genoImpute(gdatF8, gmap=gmapF8, step=Inf,
   gr=8, na.str=NA, msg=TRUE)
sum(is.na(tmp))
tmp[1:5, 1:5]

## End(Not run)
```

---

genoProb *Probability of a Genotype.*

Description

Calculate the probability of a genotype at a locus conditional on the genotypes of its flanking markers in advance intercross lines (AIL).

Usage

```
genoProb(gdat, gmap, step = Inf, gr = 2, pos = NULL, method=c("Haldane",
   "Kosambi"), msg = FALSE)
```

Arguments

gdat	Genotype data. Should be a matrix or a data frame, with each row representing an observation and each column a marker locus. The column names should be marker names. Each entry should be 1, 2, 3 or 0, corresponding to "AA", "AB", "BB" or missing genotype.
gmap	A genetic map. Should be data frame (snp, chr, dist,...), where "snp" is the SNP (marker) name, "chr" is the chromosome where the "snp" is, and "dist" is the genetic distance in centi-Morgan (cM) from the left of the chromosome.
step	The maximum "cumulative" distance (in cM) between two adjacent loci for which the probabilities are calculated. The distance corresponds to the "cumulative" recombination rate at gr-th generation.
gr	The generation under consideration.
pos	Data frame (chr, dist, snp, ...). If given, step will be ignored.
method	Whether "Haldane" or "Kosambi" mapping function should be used.
msg	A logical variable. If TRUE, certain information will be printed out during calculation.

Details

The "cumulative" genetic distance between any two adjacent loci for which probabilities are calculated is not larger than step. If step = Inf, probabilities will only be calculated at loci in both the columns of gdat and the rows of gmap. If step is small, a large set of putative loci will be considered, including all loci defined by the columns of gdat and the rows of gmap.

Value

Probabilities for genotypes as well as genetic map information (snp,chr,dist)

`pr` A 3-D array with the first dimension corresponding to that of `gdat`, the second to three genotype and the third to the putative loci. The probabilities will be -1 if not imputable, which happens when the genotype data is missing at all loci on the chromosome.

Note

Currently only suitable for advanced intercross lines.

Examples

```
data(miscEx)

## Not run:
# briefly look at genotype data
sum(is.na(gdatF8))
gdatF8[1:5,1:5]

gdtmp<- gdatF8
   gdtmp<- replace(gdtmp,is.na(gdtmp),0)
# In case an individual is not imputable, then
# one needs to assign genotypes manually
prDat<- genoProb(gdat=gdtmp, gmap=gmapF8, step=Inf,
   gr=8, method="Haldane", msg=TRUE)
prDat$pr[1:5,,1:5]

## End(Not run)
```

---

genoSim *Generate Genotypic Data*

Description

Simulate genotypic data from a pedigree in advanced intercross lines (AIL).

Usage

```
genoSim(ped, gmap, ids, hap, method = c("Haldane", "Kosambi"))
```

Arguments

`ped` A pedigree, which is a data frame (id, sex, father/sire, mother/dam, ...). In "sex", male should be "M", "Male" or 1, and female should be "F", "Female" or 2 (other than 0 and 1). If given, "generation" can be numeric 0, 1, 2, ... or non-numeric "F0", "F1", "F2", ..., which should be in an increasing order. Note that 0 is reserved for missing values. If a father/mother is an inbred founder, its ID should be tagged by character 'i' (e.g. 1i, 2i, etc.). See `pedRecode`.

gmap	A genetic map. Should be data frame (snp, chr, dist, ...), where "snp" is the SNP (marker) name, "chr" is the chromosome where the "snp" is, and "dist" is the genetic distance in centi-Morgan (cM) from the left of the chromosome. If gmap is missing but hap not, all but the first two columns of hap are ignored.
ids	Genotypic data are extracted only for individuals with IDs specified by ids. If missing, genotypic data are extracted for all individuals in the pedigree. If ped is an object of pedRecode, ids should be referred to "old" IDs.
hap	Founders' haplotype data if not missing. Rows correspond to founders as specified by row names, and columns correspond to loci in the genetic map gmap in the exact order. For an individual, the haplotype should be (f1 m1 f2 m2 ...) where fi is the allele from father at the i-th locus and mi is the allele from mother at the i-th locus. Elements should be non-negative integers that are not larger than 16384. If hap is not supplied, founders are assumed to be inbred.
method	Whether "Haldane" or "Kosambi" mapping function should be used. This will be ignored if the recombination rate recRate is a component of gmap.

Details

The pedigree should be in the same format as an output of pedRecode. Sex chromosome should be marked by 'x' or 'X'. Founders mean those whose parents have 0 or negative IDs after the pedigree is recoded by pedRecode. In addition, it is assumed that there are not more than two founders; otherwise, you may run hapSim and then extract genotypes manually.

Value

A matrix, with entry value s-1 where s is the summation of the numbers representing two alleles at a locus. For instance, 1, 2, and 3 representing genotypes "AA", "AB" and "BB" respectively if hap is not specified. Each row represent an observation, and each column corresponds to SNP in gmap.

Note

Sex may be used as a covariate if significance on x-chromosome is assessed by gene dropping through this function.

See Also

pedRecode for more information.

Examples

```
data(miscEx)

## Not run:
# simulate genotypes for F8 individuals
ids<- sapply(pedF8$id[pedF8$gen == "F8" & pedF8$sire != "32089"], as.character)
gdt<- genoSim(pedF8, gmapF8, ids=ids)
dim(gdt)
gdt[1:5,1:5]

## End(Not run)
```

gls *Generalized Least Squares Estimates*

Description

Obtain estimates using generalized least squares (gls).

Usage

```
gls(formula, data, vc = NULL, test=c("none","F"))
```

Arguments

formula	An object of class "formula": a symbolic description of the model to be fitted.
data	An data frame containing the variables in the model.
vc	An object from estVC or aicVC or an estimated variance-covariance matrix induced by relatedness and environment if not NULL.
test	Wheter F-test is performed.

Value

A matrix with columns: "Estimate", "Std. Error", "t value" and "Pr(>|t|)", or an ANOVA table if F-test is requested.

See Also

lm.

hapSim *Generate Genotypic Data*

Description

Simulate gametic data from a pedigree.

Usage

```
hapSim(ped, gmap, ids, hap, method = c("Haldane", "Kosambi"))
```

Arguments

ped	A pedigree, which is a data frame (id, sex, father/sire, mother/dam, ...). In "sex", male should be "M", "Male" or 1, and female should be "F", "Female" or 2 (other than 0 and 1). If given, "generation" can be numeric 0, 1, 2, ... or non-numeric "F0", "F1", "F2", ..., which should be in an increasing order. Note that 0 is reserved for missing values. If a father/mother is an inbred founder, its ID should be tagged by character 'i' (e.g. 1i, 2i, etc.). See pedRecode.
gmap	A genetic map. Should be data frame (snp, chr, dist, ...), where "snp" is the SNP (marker) name, "chr" is the chromosome where the "snp" is, and "dist" is the genetic distance in centi-Morgan (cM) from the left of the chromosome. If gmap is missing but hap not, all but the first two columns of hap are ignored.
ids	Genotypic data are extracted only for individuals with IDs specified by ids. If missing, genotypic data are extracted for all individuals in the pedigree. If ped is an object of pedRecode, ids should be referred to "old" IDs.
hap	Founders' haplotype data if not missing. Rows correspond to founders as specified by row names, and columns correspond to loci in the genetic map gmap in the exact order. For an individual, the haplotype should be (f1 m1 f2 m2 ...) where fi is the allele from father at the i-th locus and mi is the allele from mother at the i-th locus. Elements should be non-negative integers that are not larger than 16384. If hap is not supplied, founders are assumed to be inbred.
method	Whether "Haldane" or "Kosambi" mapping function should be used. This will be ignored if the recombination rate recRate is a component of gmap.

Details

The pedigree should be in the same format as an output of pedRecode. Founders mean those whose parents have 0 or negative IDs after the pedigree is recoded by pedRecode.

Value

A matrix giving haplotypes.

See Also

pedRecode for more information.

Examples

```
data(miscEx)

## Not run:
# prepare pedigree in desired format
pedR<- pedRecode(pedF8)
pedR[1:5,] # check to find out three founders
# fake founder haplotypes
hapDat<- rbind(rep(1:2,nrow(gmapF8)),rep(3:4,nrow(gmapF8)),rep(5:6,nrow(gmapF8)))
rownames(hapDat)<- c("32089","1","2")
# simulate hyplotypes for F8 individuals
```

```
hd<- hapSim(pedF8, gmapF8, ids=pedF8$id[pedF8$gen=="F8"], hap=hapDat)
dim(hd)
hd[1:5,1:10]

## End(Not run)
```

---

ibs *Estimate Jacquard condensed identity coefficients*

Description

Estimate Jacquard condensed identity coefficients by identity-by-state (IBS) from genotypic data.

Usage

```
ibs(x)
```

Arguments

x	Genotype data with genotypes ("AA", "AB", "BB", or, 1, 2, 3) and without missing data, or probabilities for these genotypes (e.g., obtained by using genoProb). In case of genotype data, rows represent individuals and columns represent SNPs.

Value

A matrix G with G[,j] being the j-th Jacquard identity coefficients.

Note

Currently only support the two-allele data.

See Also

genMatrix

`kinship` *Calculate kinship coefficients*

Description

Calculate kinship coefficients from a pedigree.

Usage

```
kinship(ped, ids, all = TRUE, msg = TRUE)
```

Arguments

`ped`	A pedigree, which a data frame (id, sire, dam, ...). If given, "generation" can be numeric 0, 1, 2, ... or non-numeric "F0", "F1", "F2", ... If "sex" is included, male should be "M", "Male" or 1, and female should be "F", "Female" or 2 (other than 0 and 1). If a founder is inbred, its ID should be tagged by character 'i' (e.g. 1i, 2i, etc.). Note that 0 is reserved for missing values.
`ids`	IDs of the individuals. If given, kinship coefficients are extracted for individuals with ID `ids`; otherwise, kinship coefficients are provided for all individuals in the pedigree.
`all`	If false, sires and dams with no parents are treated as unknown.
`msg`	If false, messages are suppressed.

Value

A matrix giving kinship coefficients.

Examples

```
data(miscEx)

ids<- sample(pedF8$id,10)
## Not run:
ksp<- kinship(pedF8,ids=ids)

## End(Not run)
```

`lodci` *Estimate LOD Support Intervals*

Description

Estimate LOD support intervals.

Usage

```
lodci(llk, cv = 0, lod = 1.5, drop = 3)
```

Arguments

`llk`	A data frame with components (chr, dist, y, ...), where "chr" is the chromosome on which the scanning locus is located, "dist" is the genetic or physical position of the scanning locus, and "y" is the test statistic.
`cv`	Threshold. Reported support intervals cover at least one scanning locus where `llk$y > cv`.
`lod`	`lod` (1.5 by default) LOD support intervals are reported when `llk$y` is converted to LOD score.
`drop`	3 by default. See "details".

Details

In case of multiple peaks on a chromosome, a peak has to satisfy: a) above the threshold `cv`; b) drops, e.g., 3 LOD on both sides except chromosome ends. So if two peaks close to each other but LOD between them doesn't drop, e.g., 3 LOD, only one of them is considered.

Value

A data frame with the following components:

`chr`	The chromosome
`lower`	The lower bound
`upper`	The upper bound
`index`	Indicates which scanning loci

Examples

```
data(miscEx)

## Not run:
# impute missing genotypes
pheno<- pdatF8[!is.na(pdatF8$bwt) & !is.na(pdatF8$sex),]
ii<- match(rownames(pheno), rownames(gdatF8))
geno<- gdatF8[ii,]
ii<- match(rownames(pheno), rownames(gmF8$AA))
```

```
v<- list(A=gmF8$AA[ii,ii], D=gmF8$DD[ii,ii])

gdtmp<- geno
   gdtmp<- replace(gdtmp,is.na(gdtmp),0)
# run 'genoProb'
prDat<- genoProb(gdat=gdtmp, gmap=gmapF8, step=Inf,
   gr=8, method="Haldane", msg=TRUE)
# estimate variance components
o<- estVC(y=pheno$bwt, x=pheno$sex, v=v)

# genome scan
llk.hk<- scanOne(y=pheno$bwt, x=pheno$sex, vc=o, prdat=prDat)

# extract LOD support intervals
tmp<- data.frame(y=llk.hk$p, chr=llk.hk$chr, dist=llk.hk$dist)
lodci(tmp, cv=10, lod=1.5, drop=3)

## End(Not run)
```

---

mAIC *Multiple QTL AIC*

Description

Multiple QTL model selection by AIC criterion.

Usage

```
mAIC(y, x, gdat, prdat = NULL, vc = NULL, chrIdx, xin, k = 2,
   direction = c("both","backward","forward"), ext = FALSE, msg = FALSE)
```

Arguments

y	A numeric vector or a numeric matrix of one column (representing a phenotype for instance).
x	A data frame or matrix, representing covariates if not missing.
gdat	Genotype data. Should be a matrix or a data frame, with each row representing an observation and each column a marker locus. The column names should be marker names. Numeric coding of genotype is treated as numeric. Ignored if prdat is an object from genoProb.
vc	An object from estVC or aicVC, or an estimated variance-covariance matrix induced by relatedness. The scan will assume no polygenic variation if vc is NULL.
prdat	An object from genoProb.
chrIdx	Chromsome index of markers in columns of gdat if given. Ignored if prdat is an object from genoProb.
xin	Vector indicating whether a locus is already in the model.

k	Penalty on a parameter. The selection criterion is the known "AIC" if k = 2 and is "BIC" if k = log(n) where "n" is the sample size.
direction	The mode of search: "both", "forward" or "backward" with default "both".
ext	A logical variable. True if ones wants more exhaustive search.
msg	A logical variable. True if ones wants to track the process for monitoring purpose.

Details

Makes use of "Haley-Knott" method (Haley and Knott 1992) if prdat is an object from genoProb.

Value

A list with the following components:

model: the resulting model;

aic: AIC of the model;

snp: selected SNPs.

xin: vector indicating whether a SNP is selected.

Note

Currently only suitable for advanced intercross lines (or diallelic data).

References

Haley, C. S., and S. A. Knott (1992). A simple regression method for mapping quantitative trait loci in line crosses using flanking markers. Heredity 69: 315-324.

See Also

optim, genoProb and aicVC.

Examples

```
data(miscEx)

## Not run:
# impute missing genotypes
pheno<- pdatF8[!is.na(pdatF8$bwt) & !is.na(pdatF8$sex),]
ii<- match(rownames(pheno), rownames(gdatF8))
geno<- gdatF8[ii,]
ii<- match(rownames(pheno), rownames(gmF8$AA))
v<- list(A=gmF8$AA[ii,ii], D=gmF8$DD[ii,ii])

gdat.imp<- genoImpute(geno, gmap=gmapF8, step=Inf,
   gr=8, na.str=NA)
# estimate variance components
o<- estVC(y=pheno$bwt, x=pheno$sex, v=v)
```

```
# run 'genoProb'
gdtmp<- geno
   gdtmp<- replace(gdtmp,is.na(gdtmp),0)
prDat<- genoProb(gdat=gdtmp, gmap=gmapF8, step=Inf,
   gr=8, method="Haldane", msg=TRUE)

# genome scan
llk.hk<- scanOne(y=pheno$bwt, x=pheno$sex, prdat=prDat, vc=o)
xin<- llk.hk$p > 10

# run 'mAIC' based on genome scan results
mg<- mAIC(y=pheno$bwt, x=pheno$sex, prdat=prDat, vc=o, xin=xin,
   k=5, direction="back", msg=TRUE)
mg$model$value # likelihood of the final model

## End(Not run)
```

`miscEx` *Genotype data, phenotype data, genetic map and pedigree.*

Description

AIL F8 data include the following:

"gmF8": A list with elements inbreeding coefficients "ib", additive genetic matrix "AA", dominance genetic matrix "DD" and other genetic matrices.

"pedF8": Pedigree data.

"pedF8.1", "pedF8.2": Alternative versions of pedigree pedF8.

"gmapF8: Genetic map.

"gdatF8": Genotype data.

"pdatF8": Phenotype data.

Usage

```
data(miscEx)
```

`misFct` *A collection of other functions.*

Description

A collection of other functions that are not needed by users.

nullSim *Simulate null distribution*

Description

Simulate distribution under the hypothesis of no QTL by permutation (of genotypic data) or gene dropping.

Usage

```
nullSim(y, x, gdat, prdat, ped, gmap, hap,
method = c("permutation","gene dropping"), vc = NULL, intc = NULL,
test = c("None","F","Chisq"), minorGenoFreq = 0.05, rmv = TRUE,
gr = 2, ntimes = 10)
```

Arguments

y	A numeric vector or a numeric matrix of one column (representing a phenotype for instance).
x	A data frame or matrix, representing covariates if not missing.
gdat	Genotype data without missing values. Should be a matrix or a data frame, with each row representing an observation and each column a marker locus. Ignored in the case of gene dropping.
prdat	An object from genoProb, or in the same form.
ped	A pedigree, which is a data frame (id, sex, father/sire, mother/dam, ...). In "sex", male should be "M", "Male" or 1, and female should be "F", "Female" or 2 (other than 0 and 1). If given, "generation" can be numeric 0, 1, 2, ... or non-numeric "F0", "F1", "F2", ... Note that 0 is reserved for missing values. Ignored in the case of permutation.
gmap	A genetic map. Should be data frame (snp, chr, dist, ...), where "snp" is the SNP (marker) name, "chr" is the chromosome where the "snp" is, and "dist" is the genetic distance in centi-Morgan (cM) from the left of the chromosome. Ignored in the case of permutation.
hap	Founders' haplotype data if not missing. Rows correspond to all founders, which should be in the first places in the pedigree ped, in the exact order and columns correspond to loci in the genetic map gmap in the exact order. For an individual, the haplotype should be (f1 m1 f2 m2 ...) where fi is the allele from father at the i-th locus and mi is the allele from mother at the i-th locus. Elements should be non-negative integers that are not larger than 16384. If missing, two founders with alleles 1 and 2 are assumed.
method	Permutation or gene dropping.
vc	An object from estVC or aicVC, or an estimated variance-covariance matrix induced by relatedness. The scan will assume no polygenic variation if vc is NULL.

intc	Covariates that interact with QTL.
test	"None", "F" or "Chisq".
minorGenoFreq	Specify the minimum tolerable minor genotype frequency at a scanning locus if gdat is used.
rmv	A logical variable. If true, then the scanning locus will be skipped if the minor genotype frequency at the locus is smaller than minorGenoFreq. Otherwise, the scanning process will stop and return with NULL.
gr	The generation under consideration.
ntimes	Number of simulations.

Details

Two methods considered here are permutation test and gene dropping test as described as follows.

Permutation test. Depending on the genome-scan, one can provide either gdat or prdat respectively corresponding to single-marker analysis or interval mapping. Then only arguments in scanOne are needed in addition to method and ntimes.

Gene dropping test. If prdat is provided, then gdat will be ignored. The procedure will first call genoSim to generate new genotype data and then call genoProb to generate data for Haley-Knott interval mapping. If prdat is not provided, then gdat should be provided. The procedure will generate new genotype data and scan the genome using these generated genotype data. Haldane mapping function is used to generate data.

Value

A vector of numbers of length ntimes if minorGenoFreq > 0 and rmv = TRUE, each element of which is maximum of the test statistic over the genome scan (so test should be "None"), or a matrix of ntimes rows, each row of which records a genome scan.

See Also

genoSim, genoProb and scanOne.

Examples

```
data(miscEx)

## Not run:
# impute missing genotypes
pheno<- pdatF8[!is.na(pdatF8$bwt) & !is.na(pdatF8$sex),]
ii<- match(rownames(pheno), rownames(gdatF8))
geno<- gdatF8[ii,]
ii<- match(rownames(pheno), rownames(gmF8$AA))
v<- list(A=gmF8$AA[ii,ii], D=gmF8$DD[ii,ii])

gdatTmp<- genoImpute(geno, gmap=gmapF8, step=Inf,
   gr=8, na.str=NA)
# estimate variance components
o<- estVC(y=pheno$bwt, x=pheno$sex, v=v)
```

```
# scan marker loci & permutation
ex1<- nullSim(y=pheno$bwt, x=pheno$sex, gdat=gdatTmp,
method="permutation", vc=o, ntimes=10)
dim(ex1)

# scan marker loci & gene dropping
ex2<- nullSim(y=pheno$bwt, x=pheno$sex, gdat=gdatTmp, ped=ped,
gmap=gmapF8, method="gene", vc=o, ntimes=10)
dim(ex2)

# Haley-Knott method & permutation
gdtmp<- geno
   gdtmp<- replace(gdtmp,is.na(gdtmp),0)
prDat<- genoProb(gdat=gdtmp, gmap=gmapF8, step=Inf,
   gr=8, method="Haldane", msg=TRUE)
ex3<- nullSim(y=pheno$bwt, x=pheno$sex, prdat=prDat,
method="permutation", vc=o, ntimes=10)
dim(ex3)

# Haley-Knott method & gene dropping
ex4<- nullSim(y=pheno$bwt, x=pheno$sex, prdat=prDat, ped=ped,
gmap=gmapF8, method="gene", vc=o, gr=8, ntimes=10)
dim(ex4)

## End(Not run)
```

---

pedRecode *Recode a Pedigree*

Description

Prepare a pedigree in a format that is suitable for certain functions

Usage

```
pedRecode(ped, ids, all = TRUE, msg = TRUE)
```

Arguments

ped	A pedigree, which is a data frame (id, father/sire, mother/dam, ...). If "sex" is a component, male should be "M", "Male" or 1, and female should be "F", "Female" or 2 (other than 0 and 1). If given, "generation" can be numeric 0, 1, 2, ... or non-numeric "F0", "F1", "F2", ..., which should be in an increasing order. Note: 0 is reserved for unknown `father`, `mother` or `sex`. If a father/mother is an inbred founder, its ID should be tagged by character 'i' (e.g. 1i, 2i, etc.).
ids	If given, only individuals with `ids` and their ancestors are kept in the recoded pedigree.
all	If false, fathers and mothers with no parents are treated as unknown.
msg	If false, messages are suppressed.

Details

This function is used in `cic`, and it can be used for error checking with respect to sex and generation if sex and/or generation information is available. The actual values of generation can be anything but should correspond to the true order of generation; otherwise, `cic` may fail or we may get incorrect results. Information except `id`, `father` and `mother` is optional.

Value

A recoded pedigree.

See Also

`cic`.

Examples

```
data(miscEx)

pedF8[1:10,]
pedR<- pedRecode(pedF8)
pedR[1:10,]
dim(pedR)
pedR<- pedRecode(pedF8, ids=pedF8$id[pedF8$gener=="F8"])
dim(pedR)
```

---

plotit *Plotting*

Description

Plot mapping results.

Usage

```
## S3 method for class 'scanOne'
plot(x,...)

plotit(lrt, cv, bychr = FALSE, chr.labels = TRUE, type = "p", lty = NULL,
   col = NULL, pch = NULL, cex = NULL, ...)
```

Arguments

`x`	Object from `scanOne` or `scanTwo`.
`lrt`	A data frame with (chr, dist, y,...) or (chr, dist, y, group,...), where "chr" represents chromosome, "dist" position on the chromosome, "y" the test statistic.
`cv`	Threshold to be drawn on the plot.
`cex`	See `par`.

bychr — A logical variable. If true, the plot will be displayed per chromosomes.

chr.labels — A logical variable. If true, the chromosome names will be drawn.

type,lty,col,pch — See plot.default.

... — Other options passed to R plot function. To call plot to plot results of scanOne, one may need to provide a genetic map gmap that should be data frame (snp, chr, dist, ...), where "snp" is the SNP (marker) name, "chr" is the chromosome where the "snp" is, and "dist" is the genetic distance in centi-Morgan (cM) from the left of the chromosome.

Note

A genetic map 'gmap' may be needed to plot an object of scanOne or scanTwo. The color option may not give what is expected.

Examples

```
data(miscEx)

## Not run:
# impute missing genotypes
pheno<- pdatF8[!is.na(pdatF8$bwt) & !is.na(pdatF8$sex),]
ii<- match(rownames(pheno), rownames(gdatF8))
geno<- gdatF8[ii,]
ii<- match(rownames(pheno), rownames(gmF8$AA))
v<- list(A=gmF8$AA[ii,ii], D=gmF8$DD[ii,ii])

gdat.imp<- genoImpute(geno, gmap=gmapF8, step=Inf,
   gr=8, na.str=NA)
# estimate variance components
o<- estVC(y=pheno$bwt, x=pheno$sex, v=v)

# genome scan
llk<- scanOne(y=pheno$bwt, x=pheno$sex, vc=o, gdat=gdat.imp)

# plotting
plot(llk, gmap=gmapF8) # gmap is needed

# plotting in another way
idx<- match(colnames(gdat.imp), gmapF8$snp)
tmp<- data.frame(chr=gmapF8$chr[idx],dist=gmapF8$dist[idx],y=llk$pval)
plotit(tmp, main="Mapping Plot", xlab="Chromosome", ylab="LRT",
   col=as.integer(tmp$ch)%%2+2,type="p")

## End(Not run)
```

qqPlot *Quantile-Quantile Plots*

Description

Quantile-Quantile Plots With the Ability to Draw Confidence Bands.

Usage

```
qqPlot(y, x = "norm", ...,
   type = "p", xlim = NULL, ylim = NULL,
   xlab = if(is.numeric(x)) deparse(substitute(x)) else x,
   ylab = deparse(substitute(y)),main="Q-Q Plot",
   col = 1, lty = 2, lwd = 1, pch = 1, cex = 0.7, plot.it = TRUE,
   confidence = .95, qqline = c("observed","expected","none"),
   add = FALSE)
```

Arguments

y	A numeric vector of data values.
x	Either a numeric vector of data values, or a character string naming a distribution function such as "norm".
...	Parameters passed to the distribution specified by x (if non-numerical).
type	1-character string giving the type of plot desired.
xlim	The x limits.
ylim	The y limits.
xlab	A label for the x axis.
ylab	A label for the y axis.
main	A main title for the plot.
col	Color for points and lines.
lty	Line type.
lwd	Line width.
pch	Plotting character for points.
cex	Factor for expanding the size of plotted symbols.
plot.it	Whether or not to draw a plot. if plotting, points outside the confidence bands will be indicated by different a color.
confidence	Confidence level for the confidence band, or FALSE for no band.
qqline	Whether or not to draw a reference line. if "observed", the line passes through the first and third observed quartiles; if "expected", the point (x,y) is expected to fall on the line if x and y follow the same distribution; if "none", no reference line is drawn.
add	Add to an existing plot if true.

Details

If x is numeric, a two-sample test of the null hypothesis that x and y were drawn from the same continuous distribution is performed. Alternatively, x can be a character string naming a continuous distribution function. In such a case, a one-sample test is carried out of the null that y was draw from distribution x with parameters specified by "...".

Value

x	Quantiles of x
y	Quantiles of y
lower, upper	Lower and upper limits if confidence is specified

References

George Marsaglia, Wai Wan Tsang and Jingbo Wang (2003), Evaluating Kolmogorov's distribution. Journal of Statistical Software 8 (18): 1-4.

Vijayan N. Nair (1982). Q-Q plots with confidence bands for comparing several populations.

William J. Conover (1971). Practical Nonparametric Statistics. New York: John Wiley & Sons.

See Also

ks.test.

Examples

```
## Not run:
par(mfrow=c(1,2))
x<- rnorm(200, mean=0.7,sd=2); y<- rnorm(200, sd=2)
qqPlot(y,x,qqline="exp")
qqPlot(y=y,x="norm",sd=2)
ks.test(x,y)

## End(Not run)
```

---

qtl2rel *Convert data from R/qtl to QTLRel format*

Description

Convert the data for a QTL mapping experiment from the R/qtl format (see http://www.rqtl.org) to that used by QTLRel.

Usage

```
qtl2rel(cross)
```

Arguments

cross	An object of class "cross", as defined by the R/qtl package

Details

The input cross must by an intercross (class "f2").

Simple pedigree information is created, assuming the data are from a standard intercross.

Value

A list with four components: "ped" (pedigree information), "gdat" (genotype data), "pdat" (phenotype data), and "gmap" (genetic map), in the formats used by QTLRel.

Author(s)

Karl W Broman, <kbroman@biostat.wisc.edu>

See Also

rel2qtl

Examples

```
library(qtl)
data(listeria)
listeria <- listeria[as.character(1:19),]
reldat <- qtl2rel(listeria)
```

---

qtlVar *QTL Variance*

Description

Estimate variance in a quantitative trait induced by QTL.

Usage

```
qtlVar(lrt, prdat, simulation = FALSE, nsim = 25)
```

Arguments

lrt	A data frame (a, d, ...), where 'a' and 'd' are respectively additive and dominance effects.
prdat	A 3-D array that provides probabilities of genotypes "AA", "AB" and "BB". If prDat is an object of genoProb, then prdat can be prDat$pr.
simulation	Whether to use simulations to estimate the variance explained by QTL.
nsim	Number of simulations to perform if simulation is TRUE.

Value

A vector displaying the estimated variance at each loci.

Note

Correlations among observations are ignored, and this function should be used with caution.

See Also

scanOne and genoProb

Examples

```
data(miscEx)

## Not run:
# impute missing genotypes
pheno<- pdatF8[!is.na(pdatF8$bwt) & !is.na(pdatF8$sex),]
ii<- match(rownames(pheno), rownames(gdatF8))
geno<- gdatF8[ii,]
ii<- match(rownames(pheno), rownames(gmF8$AA))
v<- list(A=gmF8$AA[ii,ii], D=gmF8$DD[ii,ii])

gdtmp<- geno
   gdtmp<- replace(gdtmp,is.na(gdtmp),0)
# rung 'genoProb'
prDat<- genoProb(gdat=gdtmp, gmap=gmapF8, step=Inf,
   gr=8, method="Haldane", msg=TRUE)
# estimate variance components
o<- estVC(y=pheno$bwt, x=pheno$sex, v=v)

# genome scan
pv.hk<- scanOne(y=pheno$bwt, x=pheno$sex, prdat=prDat, vc=o)

# run 'qtlVar'
qef<- NULL
for(n in 1:length(pv.hk$par))
   qef<- rbind(qef,pv.hk$par[[n]][c("a","d")])
   qef<- as.data.frame(qef)
qtlVar(qef,prDat$pr)[1:3]

## End(Not run)
```

---

rel2qtl *Convert data from QTLRel to R/qtl format*

Description

Convert the data for a QTL mapping experiment from the QTLRel format to that used by R/qtl (http://www.rqtl.org).

Usage

```
rel2qtl(gdat, pdat, gmap)
```

Arguments

gdat	Genotype data
pdat	Phenotype data
gmap	Genetic map

Details

Pedigree information is ignored, and X chromosome data is omitted.

The data are treated as an intercross.

Value

A cross object for the R/qtl package (http://www.rqtl.org).

Author(s)

Karl W Broman, <kbroman@biostat.wisc.edu>

See Also

qtl2rel

Examples

```
data(miscEx)
f8 <- rel2qtl(gdatF8, pdatF8, gmapF8)
summary(f8)
```

---

rem *Random effect matrices*

Description

Construct matrices associated with random effects.

Usage

```
rem(formula,data)
```

Arguments

formula	A formula of the form: ~ Z | G1/.../Gk + ..., corresponding to random effects Z*G_i + Z*G_{ij} + ... in a mixed effect model. If Z=1 as in most cases, then it can be ~ G1/.../Gk + ...
data	A data frame that contains all the variables in the formula.

Value

A list of matrices that are associated with random effects.

Examples

```
## Not run:
# make-up example
dat<- data.frame(
   group=c("A","A","A","A","A","A","B","B","B","B"),
   sex=c("F","F","F","M","M","M","F","F","M","M"),
   pass=c("Y","N","N","Y","Y","Y","Y","N","N","Y"),
   z=1:10)

# random effect pass, group and sex, where sex is nested
# within group:
# y_{ijk} = x_{ij}*b + group_i + sex_{ij} + z*pass_{ij}
#           + e_{ijk}
rem(~ group/sex + z|pass,data=dat)

## End(Not run)
```

---

scanOne *Genome Scan for QTL*

Description

Evaluate likelihood ratio test statistics or P-values at scanning loci along the genome.

Usage

```
scanOne(y, x, gdat, prdat = NULL, vc = NULL, intc = NULL,
   numGeno = FALSE, test = c("None","F","LRT"),
   minorGenoFreq = 0, rmv = TRUE)
```

Arguments

y	A numeric vector or a numeric matrix of one column (representing a phenotype for instance).
x	A data frame or matrix, representing covariates if not missing.

gdat	Genotype data. Commonly, should be a matrix or a data frame, with each row representing an observation and each column a marker locus. The column names should be marker names. Optional if an object prdat from genoProb is used as an argument. If gdat is not numeric, there can be more than three genotypes. However, all scanning loci should have the same number of genotypes. Otherwise, we can split gdat into sub-matrices that each have the same number of genotypes and run the analysis for these sub-matrices one after another.
prdat	An object from genoProb, or in the same form. It should have a class "addEff" if allelic effects are assumed to be additive (see example below).
vc	An object from estVC or aicVC, or an estimated variance-covariance matrix induced by relatedness and environment.
intc	Covariates that interact with QTL.
numGeno	Whether to treat numeric coding of genotypes as numeric. If true, minorGenoFreq will be ignored.
test	"None", "F" or "LRT".
minorGenoFreq	Specify the minimum tolerable minor genotype frequency at a scanning locus if gdat is used.
rmv	A logical variable. If true, then the scanning locus will be skipped if the minor genotype frequency at the locus is smaller than minorGenoFreq. Otherwise, the scanning process will stop and return with NULL.

Details

The test at a scanning locus under the assumption of no QTL effect versus the assumption of QTL effect is performed by conditioning on the estimated polygenic genetic variance-covariance matrix. Normality is assumed for the random effects.

It is possible to extend the Haley-Knott approach to multiple-allelic cases under the assumption that allele effects are all additive. Then, prdat should be provided and be of class "addEff".

Value

A list with at least the following components:

F or LRT	the F-test or likelihood ratio test (LRT) statistic at the SNP (marker) if test is "F" or otherwise
pval	P-value at the snp (marker) if test is "F" or "LRT"
v	Variation explained by the SNP (marker)
parameters	Estimated parameters at all scanning loci, including additive effect a and dominance effect d if prdat is not NULL

References

Haley, C. S., and S. A. Knott (1992). A simple regression method for mapping quantitative trait loci in line crosses using flanking markers. Heredity 69: 315-324.

See Also

genoImpute and genoProb.

Examples

```
data(miscEx)

## Not run:
# impute missing genotypes
pheno<- pdatF8[!is.na(pdatF8$bwt) & !is.na(pdatF8$sex),]
ii<- match(rownames(pheno), rownames(gdatF8))
geno<- gdatF8[ii,]
ii<- match(rownames(pheno), rownames(gmF8$AA))
v<- list(A=gmF8$AA[ii,ii], D=gmF8$DD[ii,ii])

# estimate variance components
o<- estVC(y=pheno$bwt, x=pheno$sex, v=v)

# impute missing genotypes
gdtmp<- genoImpute(geno, gmap=gmapF8, step=Inf,
        gr=8, na.str=NA, msg=FALSE)
# genome scan and plotting
lrt<- scanOne(y=pheno$bwt, x=pheno$sex, gdat=gdtmp, vc=o)
lrt
plot(lrt,gmap=gmapF8)

# Haley-Knott method
gdtmp<- geno; unique(unlist(gdtmp))
   gdtmp<- replace(gdtmp,is.na(gdtmp),0)
prDat<- genoProb(gdat=gdtmp, gmap=gmapF8, step=Inf,
   gr=8, method="Haldane", msg=TRUE)
pv.hk<- scanOne(y=pheno$bwt, intc=pheno$sex, prdat=prDat, vc=o, test="F")
pv.hk
plot(pv.hk, gmap=gmapF8)

# assume additive allelic effects
class(prDat)<- c(class(prDat), "addEff")
lrt.hk<- scanOne(y=pheno$bwt, intc=pheno$sex, prdat=prDat, vc=o)
lrt.hk

## End(Not run)
```

---

scanTwo *Genome Scan for Epistasis*

Description

Evaluate log-likelihood ratio test statistic for epistasis (QTL by QTL interaction).

Usage

```
scanTwo(y, x, gdat, prdat = NULL, vc = NULL, numGeno = FALSE,
   minorGenoFreq = 0, rmv = TRUE)
```

Arguments

y	A numeric vector or a numeric matrix of one column (representing a phenotype for instance).
x	A data frame or matrix, representing covariates if not missing.
gdat	Genotype data. Should be a matrix or a data frame, with each row representing an observation and each column a marker locus. The column names should be marker names. Optional if an object prdat from genoProb is used as an argument.
prdat	An object from genoProb.
vc	An object from estVC or aicVC, or an estimated variance-covariance matrix induced by relatedness and environment.
numGeno	Whether to treat numeric coding of genotypes as numeric. If true, minorGenoFreq will be ignored.
minorGenoFreq	Specify the minimum tolerable minor genotype frequency at a scanning locus if gdat is used.
rmv	A logical variable. If true, then the scanning locus will be skipped if the minor genotype frequency at the locus is smaller than minorGenoFreq. Otherwise, the scanning process will stop and return with NULL.

Value

A matrix whose entry in the upper triangle is the log-likelihood test statistic for epistatic effect.

See Also

scanOne.

Index