Package 'diaQTL'

August 10, 2020

Title QTL Analysis in Diallel Populations

Version 0.71

Author Jeffrey B. Endelman and Rodrigo R. Amadeu

Maintainer Jeffrey Endelman <endelman@wisc.edu>

Description QTL analysis of diploid and autotetraploid diallel populations. Phenotypes are regressed on genotype probabilities, and the regression coefficients are random effects.

Depends R (>= 3.5.0)

License GPL-3

LazyData true

RoxygenNote 7.1.0

Roxygen list(markdown = TRUE)

Encoding UTF-8

Imports BGLR, ggplot2, methods, coda, Matrix, scam, parallel, arrangements

Suggests knitr, rmarkdown

VignetteBuilder knitr

R topics documented:

Amat *A matrix*

Description

Calculates the additive (A) relationship matrix from founder genotype probabilities

Usage

```
Amat(data, chrom = NULL)
```

Arguments

data	Variable inheriting from class diallel_geno
chrom	Optional, vector of chromosome names to include

Details

Additive relationships are calculated from kinship coefficients of order 2 (Gallais 2003). Can specify to use only a subset of the chromosomes (by default, all chromosomes are used). Calculated based on the marker bins.

Value

A matrix

References

Gallais, A. 2003. Quantitative Genetics and Breeding Methods in Autopolyploid Plants. Institut National de la Recherche Agronomique, Paris.

Examples

```
## Not run:
  Amat_example = Amat(data = diallel_example)
  Amat_example = Amat(data = diallel_example, chrom=c(1:11)) #leave chromosome 12 out

## End(Not run)
```

diallel_geno-class *S4 class with genotype data*

Description

S4 class with genotype data

Slots

`ploidy` Either 2 or 4

`dominance` Integer 1-4 indicating 1 = additive, 2 = digenic dominance, 3 = trigenic dominance, 4 = quadrigenic dominance.

`X.GCA` Incidence matrix for GCA effects

`map` data frame with marker,chrom, position (cM and/or bp) and bin

`geno` list of length equal to the number of marker bins. Each element is a list of length `dominance`. The elements in the nested list are sparse matrices with dimensions (id x effects).

`diallel_geno_pheno-class` *S4 class with genotype and phenotype data*

Description

S4 class with genotype and phenotype data

Slots

`ploidy` Either 2 or 4

`dominance` Integer 1-4 indicating 1 = additive, 2 = digenic dominance, 3 = trigenic dominance, 4 = quadrigenic dominance.

`X.GCA` Incidence matrix for GCA effects

`map` data frame with marker,chrom, position (cM and/or bp) and bin

`geno` list of length equal to the number of marker bins. Each element is a list of length ploidy corresponding to additive, digenic, trigenic, and quadrigenic effects. The elements in the nested list are sparse matrices with dimensions (id x effects).

`pheno` data frame of phenotypes

`X` incidence matrix for fixed effects

`Z` incidence matrix for individuals

`Dmat` *Dominance matrix*

Description

Calculates the dominance (D) relationship matrix from founder genotype probabilities

Usage

```
Dmat(data, chrom = NULL, dominance = 2)
```

Arguments

`data`	Variable inheriting from class `diallel_geno`
`chrom`	Optional, vector of chromosome names to include
`dominance`	Either 2, 3, or 4

Details

Parameter `dominance` refers to 2 = digenic, 3 = trigenic, 4 = quadrigenic (Gallais 2003). Can specify to use only a subset of the chromosomes (by default, all chromosomes are used). Calculated based on the marker bins.

Value

Dominance relationship matrix

References

Gallais, A. 2003. Quantitative Genetics and Breeding Methods in Autopolyploid Plants. Institut National de la Recherche Agronomique, Paris.

Examples

```
## Not run:
  Dmat_example = Dmat(data = diallel_example, dominance=2) #digenic dominance
  Dmat_example = Dmat(data = diallel_example, dominance=3) #trigenic dominance

## End(Not run)
```

F1codes *Genotype codes for F1 populations*

Description

Character vector with the 100 possible tetraploid genotypes for a F1 population. Maternal haplotypes are denoted 1,2,3,4 and paternal haplotypes 5,6,7,8.

Usage

```
data(F1codes)
```

Format

character vector

`fitQTL` *Fit a single QTL model*

Description

Fit a single QTL model

Usage

```
fitQTL(data, params, dominance = 1, trait, marker, cofactor = NULL)
```

Arguments

`data`	Variable of class `diallel_geno_pheno`
`params`	List containing the number of burn-in (burnIn) and total iterations (nIter)
`dominance`	Dominance degree (1-4). See Details.
`trait`	Name of trait
`marker`	Name of marker to fit as QTL
`cofactor`	Name of marker to fit as cofactor

Details

Standard errors of the posterior mean estimates are calculated by dividing the SD of the Markov Chain by the square root of the effective number of iterations, which is calculated by function `effectiveSize` in R package `coda`. The error bars on the plot of additive effects correspond to +/- 1.96*SE (95 percent confidence interval). For binary traits, R2 = the squared phi correlation. The additive and dominance variances are reported as a proportion of the total variance: h2=Va/(Va+Vd+Vresid) and d2=Vd/(Va+Vd+Vresid). Parameter `dominance` controls the genetic model: 1 = additive, 2 = digenic dominance, 3 = trigenic dominance, 4 = quadrigenic dominance.

Value

List containing

R2 Coefficient of determination

deltaDIC Deviance Information Criterion relative to null model

h2 Mean and SE for proportion of variance due to additive effects

effectsA Mean and SE of the additive effects for parental haplotypes

plotA ggplot object for additive effects

If dominance > 1, the list also contains

d2 Mean and SE for proportion of variance due to dominance (all orders)

effectsD Mean and SE of the digenic dominance effects

plotD ggplot object for digenic dominance effects

Examples

```
## Not run:
## additive effects
params1 <- set_params( diallel_example, trait = "tuber_shape" )

fit1 <- fitQTL( data = diallel_example,
                trait = "tuber_shape",
                params = params1,
                marker = "solcap_snp_c2_25522")

## additive + dominance effects
params2 <- set_params( diallel_example, trait = "tuber_shape", dominance = TRUE )

fit2 <- fitQTL( data = diallel_example,
                trait = "tuber_shape",
                params = params2,
                marker = "solcap_snp_c2_25522",
                dominance = 2)


## End(Not run)
```

`haplotypes` *Dosage of parental haplotypes*

Description

Dosage of parental haplotypes

Usage

```
haplotypes(data, marker = NULL, id = NULL)
```

Arguments

`data`	Variable inheriting from class `diallel_geno`
`marker`	Name of marker
`id`	Name of individual

Details

Function can be used to get parental haplotype dosage estimates at a single marker for all individuals (in which case `id` should be `NULL`) or for a single individual for all markers (in which case `marker` should be `NULL`)

Value

Matrix of (id or markers) x parental haplotypes

Examples

```
## Not run:
  haplo_example = haplotypes(data = diallel_example,
                             marker = "solcap_snp_c2_25522")
  haplo_example = haplotypes(data = diallel_example,
                             id = "W15263-8R")

## End(Not run)
```

`haplo_plot` *Plot parental haplotype dosage*

Description

Plot parental haplotype dosages across the chromosome for one individual

Usage

```
haplo_plot(data, id, chrom, position, marker = NULL)
```

Arguments

`data`	Variable inheriting from class `diallel_geno`
`id`	Name of individual
`chrom`	Name of chromosome
`position`	Either "cM" or "bp"
`marker`	Optional, marker to indicate with dashed line

Details

For "cM" plotting, only one marker per bin is displayed. For "bp" plotting, all markers are included.

Value

ggplot object

Examples

```
## Not run:
haplo_plot(data = diallel_example,
           id = "W15263-8R",
           chrom = 10)

haplo_plot(data = diallel_example,
           id = "W15263-8R",
           chrom = 10,
           marker = "solcap_snp_c2_25522")

## End(Not run)
```

haplo_switch *Find haplotype switches*

Description

Find haplotype switches

Usage

```
haplo_switch(data, marker, haplotype, position, jump = 0.8)
```

Arguments

data	Variable inheriting from class diallel_geno
marker	Name of focal marker
haplotype	Name of parental haplotype
position	Either "cM" or "bp"
jump	Change in dosage to designate haplotype switch

Details

Designed to help with fine mapping of QTL. Function returns the location of the nearest haplotype switch on both sides (left, right) of a marker and haplotype of interest that represent the presumed location of a QTL allele.

Value

Data frame with the locations of the nearest haplotype switch on the left and right side of the focal marker for all individuals with a haplotype switch

LODthresh *LOD thresholds for scan1*

Description

LOD thresholds for scan1

Usage

```
LODthresh(genome.size, num.parents, ploidy)
```

Arguments

genome.size	Genome size in Morgans (not centiMorgans)
num.parents	Number of parents
ploidy	2 or 4

Details

LOD thresholds to control the genome-wide false positive rate at 0.05 were determined via simulation for up to 20 parents and genome sizes up to 12 Morgans. A monotone increasing concave curve was fit to these results using R package `scam` and is used for prediction. (The LOD threshold does not depend on population size.)

Value

LOD threshold

`read_data` *Read data files*

Description

Reads genotype, pedigree, and phenotype data files

Usage

```
read_data(
  genofile,
  ploidy = 4,
  pedfile,
  phenofile = NULL,
  fixed = NULL,
  bin.markers = T,
  dominance = 2,
  n.core = 1
)
```

Arguments

`genofile`	File with map and genotype probabilities
`ploidy`	Either 2 or 4
`pedfile`	File with pedigree information (id,mother,father)
`phenofile`	File with phenotype data (optional)
`fixed`	If there are fixed effects, this is a character vector of "factor" or "numeric"
`bin.markers`	TRUE/FALSE whether to bin markers with the same cM position
`dominance`	Dominance degree (1-4). See Details.
`n.core`	Number of cores for parallel execution (only available from Linux or Mac command line)

Details

The first 3 or 4 columns of the genotype file are the map (marker, chrom, bp and/or cM), followed by the members of the population. The genotype information for each marker x individual combination is a string with the format "state|state|state...=>prob|prob|prob...", where "state" refers to the genotype state and "prob" is the genotype probability in decimal format. Only states with nonzero probabilities need to be listed. The encoding for the states in tetraploids is described in the documentation for the F1codes and S1codes datasets that come with the package. For diploids, there are 4 F1 genotype codes, 1,2,3,4, which correspond to haplotype combinations 1-3,1-4,2-3,2-4, respectively; the S1 genotype codes 1,2,3 correspond to 1-1,1-2,2-2, respectively. For the phenotype file, first column is id, followed by traits, and then any fixed effects. Pass a character vector for the function argument "fixed" to specify whether each effect is a factor or numeric covariate. The number of traits is deduced based on the number of columns. Binary traits must be coded N/Y and are converted to 0/1 internally for analysis by probit regression. Parameter `dominance` controls the genetic model: 1 = additive, 2 = digenic dominance, 3 = trigenic dominance, 4 = quadrigenic dominance.

Value

Variable of class `diallel_geno` if phenofile is NULL, otherwise `diallel_geno_pheno`

Examples

```
## Not run:
  ## Get the location of raw csv files examples
  genocsv = system.file( "tutorial", "potato_geno.csv", package = "diaQTL" )
  pedcsv = system.file( "tutorial", "potato_ped.csv", package = "diaQTL" )
  phenocsv = system.file( "tutorial", "potato_pheno.csv", package = "diaQTL" )

  ## Check their location in the system
  print(genocsv)
  print(pedcsv)
  print(phenocsv)

  ## Load them in R
  diallel_example <- read_data(genofile = genocsv,
                               ploidy = 4,
                               pedfile = pedcsv,
                               phenofile = phenocsv)

## End(Not run)
```

S1codes *Genotype codes for S1 populations*

Description

Character vector with the 35 possible tetraploid genotypes for a S1 population. Haplotypes are denoted 1,2,3,4.

Usage

```
data(S1codes)
```

Format

character vector

`scan1` *Single QTL scan*

Description

Performs a linear regression for each position in the map.

Usage

```
scan1(
  data,
  trait,
  params,
  dominance = 1,
  chrom = NULL,
  cofactor = NULL,
  n.core = 1
)
```

Arguments

`data`	Variable of class `diallel_geno_pheno`
`trait`	Name of trait
`params`	List containing burnIn and nIter
`dominance`	Dominance degree (1-4). See Details.
`chrom`	Names of chromosomes to scan (default is all)
`cofactor`	Optional name of marker to include as cofactor in the scan
`n.core`	Number of cores for parallel execution (only available from Linux or Mac command line)

Details

For non-binary traits, R2 is the proportion of variance explained by the regression. For binary traits, R2 is the squared phi correlation. LOD score is the difference between the log-likelihood of the model with QTL and the null model (no QTL); higher values are better. deltaDIC is the difference between the DIC of the model with QTL minus the DIC of the null model; lower values are better. Parameter `dominance` controls the genetic model: 1 = additive, 2 = digenic dominance, 3 = trigenic dominance, 4 = quadrigenic dominance. Parameter `params` can be estimated using `set_params`.

Value

Data frame containing the map, LOD, R2 and deltaDIC results.

Examples

```
## Not run:
  par1 <- set_params(data = diallel_example,
                     trait = "tuber_shape")

  scan1_example <- scan1(data = diallel_example,
                chrom = 10,
                trait = "tuber_shape",
                params = par1)

## End(Not run)
```

`scan1_permute` *Permutation test for scan1*

Description

Permutation test for scan1

Usage

```
scan1_permute(
  data,
  trait,
  params,
  n.permute = 1000,
  chrom = NULL,
  dominance = 1,
  cofactor = NULL,
  n.core = 1
)
```

Arguments

`data`	Variable of class `diallel_geno_pheno`
`trait`	Name of trait
`params`	List containing burnIn and nIter
`n.permute`	Number of permutations
`chrom`	Names of chromosomes to scan (default is all)
`dominance`	Dominance degree (1-4)
`cofactor`	Optional name of marker to include as cofactor in the scan
`n.core`	Number of cores for parallel execution (only available from Linux or Mac command line)

Value

Data frame with maximum LOD and minimum deltaDIC for each iteration

Examples

```
## Not run:
  par1 <- set_params(data = diallel_example,
                     trait = "tuber_shape")

  ans1_permut <- scan1_permute(data = diallel_example,
                               chrom = 10,
                               trait = "tuber_shape",
                               params = par1,
                               n.permute = 100)


## End(Not run)
```

---

scan1_summary *Summary of scan1 result*

Description

Summary of scan1 result

Usage

```
scan1_summary(scan1_data, thresh = NULL, chrom = NULL, position)
```

Arguments

scan1_data	output from scan1
thresh	optional, LOD threshold for plotting
chrom	optional, subset of chromosomes to plot
position	Either "cM" or "bp"

Value

List containing

peaks Data frame of the markers with the highest LOD score per chromosome

plot ggplot object

Examples

```
## Not run:
  scan1_summary( scan1_example )
  scan1_summary( scan1_example, chrom = "10" )
  scan1_summary( scan1_example, chrom = c( "10", "12" ) )

## End(Not run)
```

`set_params` *Determine parameters for scan1*

Description

Determine parameters for scan1

Usage

```
set_params(data, trait, tol = 0.1, burnIn = 50, nIter = 1000)
```

Arguments

`data`	Variable of class `diallel_geno_pheno`
`trait`	Name of trait
`tol`	tolerance for estimating the median
`burnIn`	initial value for burnIn parameter
`nIter`	initial value for nIter parameter

Details

The burn-in and total number of iterations are determined using the Raftery and Lewis diagnostic from R package `coda`, based on a 95% probability that the estimated median of the additive effects is between the quantiles (0.5-tol) to (0.5+tol). For greater precision, decrease the `tol` parameter.

Value

List containing

burnIn Number of burn-in iterations

nIter Total number of iterations

Examples

```
## Not run:
  par1 <- set_params(data = diallel_example,
                     trait = "tuber_shape")

## End(Not run)
```

Index