Package 'diaQTL'

June 6, 2021

Title QTL Analysis in Diallel Populations

Version 1.00

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 (>= 4.0)

License GPL-3 + file LICENSE

LazyData true

RoxygenNote 7.1.1

Roxygen list(markdown = TRUE)

Encoding UTF-8

Imports BGLR, ggplot2, methods, coda, Matrix, scam, parallel, arrangements, tidyr, ggfittext, ggdendro, labeling, rlang, reshape2, plyr

Suggests knitr, rmarkdown

VignetteBuilder knitr

R topics documented:

`BayesCI` *Bayesian Credible Interval for QTL position*

Description

Bayesian Credible Interval for QTL position

Usage

```
BayesCI(scan1_data, data, chrom, CI.prob = 0.9)
```

Arguments

`scan1_data`	data frame output from scan1
`data`	variable of class `diallel_geno_pheno`
`chrom`	chromosome
`CI.prob`	probability for the credible interval

Details

Parameter `CI.prob` sets the probability for the Bayesian credible interval (e.g., 0.90, 0.95) using the profile likelihood (posterior mean).

Value

subset of scan1_data with markers in the CI

Examples

```
## Not run:
  BayesCI(scan1_example, diallel_example, chrom="10", CI.prob=0.9)

## End(Not run)
```

`convert_mappoly` *Generate diaQTL input files from MAPpoly*

Description

Generates diaQTL input files from the output of calc_genoprob or calc_genoprob_error in the MAP-poly package. The argument `data` is a list containing the results for each linkage group. Map distances are rounded to 0.01 cM, and genotype probabilties are rounded to three decimal places.

Usage

```
convert_mappoly(data, ploidy, outstem = "")
```

Arguments

`data`	list of variables of class mappoly.genoprob (one for each linkage group)
`ploidy`	Either 2 or 4
`outstem`	prefix for the pedigree and genotype files for diaQTL

Examples

```
## Not run:
    # see MAPpoly tutorial for details on its functions
    MAP <- list(lg1, lg2, lg3) #list of linkage groups
    genoprob <- vector("list", 3) #if 3 linkage groups
    for(i in 1:length(genoprob))
       genoprob[[i]] <- mappoly::calc_genoprob_error(input.map = MAP[[i]], error = 0.05)
    convert_mappoly(genoprob, ploidy=4)

## End(Not run)
```

`convert_onemap` *Generate diaQTL input files from OneMap*

Description

Generate diaQTL input files from 'onemap_progeny_haplotypes' object class of OneMap R package (version >2.2.0)

Usage

```
convert_onemap(data, digits = 4, outstem = "")
```

Arguments

`data`	onemap_progeny_haplotypes object class
`digits`	how many rounding digits for the probabilities output (default=4)
`outstem`	prefix for the pedigree and genotype files for diaQTL

Examples

```
## Not run:
    map <- list(LG1_final, LG2_final)
    progeny_haplot <- onemap::progeny_haplotypes(map,
                                             most_likely = FALSE,
                                             ind = "all")
    convert_onemap(progeny_haplot)

## End(Not run)
```

`convert_polyorigin` *Create diaQTL input files from PolyOrigin output*

Description

Create diaQTL input files from PolyOrigin output

Usage

```
convert_polyorigin(
  filename,
  mapfile = NULL,
  remove.outliers = TRUE,
  outstem = ""
)
```

Arguments

`filename`	Name of polyancestry file
`mapfile`	Optional name of CSV file containing the physical map (marker, chrom, bp)
`remove.outliers`	Should offspring flagged as outliers be removed (default is TRUE)
`outstem`	prefix for output filenames

Details

Creates the pedigree (diaQTL_pedfile.csv) and genotype (diaQTL_genofile.csv) input files needed for `read_data` from the polyancestry output file generated by the PolyOrigin software. PolyOrigin outputs a genetic map in cM. To add a physical map in bp, use the option `mapfile`. The input file needed for `phased_parents` (diaQTL_parents.csv) is also created.

`convert_rabbit` *Generate diaQTL input files from RABBIT MagicReconstruct*

Description

Generate diaQTL input files from RABBIT MagicReconstruct

Usage

```
convert_rabbit(rabbit.outfile, ped.file, outstem)
```

Arguments

`rabbit.outfile`	name of RABBIT output file
`ped.file`	name of RABBIT pedigree file
`outstem`	prefix for the pedigree and genotype files for diaQTL

`diallel_geno-class` *S4 class with genotype data*

Description

S4 class with genotype data

Slots

`ploidy` Either 2 or 4

`input` matrix of character strings from the genotype input file, one row per bin

`Xa` list of matrices (one for each offspring) with the expected haplotype dosage (rows) for each parental origin genotype (columns)

`dominance` Maximum dosage stored in slot `geno`. 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), containing the dosage for each effect.

`A` list with the additive relationship matrix for each chromosome

`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

`input` matrix of character strings from the genotype input file

`Xa` list of matrices with the expected haplotype dosage (rows) for each parental origin genotype (columns)

`dominance` Maximum dosage stored in slot `geno`. 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), containing the dosage for each effect.

`A` list with the additive relationship matrix for each chromosome

`pheno` data frame of phenotypes

`X` incidence matrix for fixed effects

`Z` incidence matrix for individuals

`DIC_thresh` *delta DIC thresholds for scan1*

Description

delta DIC thresholds for scan1

Usage

```
DIC_thresh(genome.size, num.parents, ploidy, alpha = 0.05, dominance = 1)
```

Arguments

`genome.size`	Genome size in Morgans (not centiMorgans)
`num.parents`	Number of parents (2 to 10)
`ploidy`	2 or 4
`alpha`	significance level (0.01, 0.05, 0.10, or 0.20)
`dominance`	1 (additive) or 2 (digenic dominance)

Details

Thresholds to control the genome-wide false positive rate at `alpha` were determined for half-diallel mating designs with up to 10 parents.

Value

-deltaDIC threshold

Examples

```
## Not run: 
  DIC_thresh(genome.size=10,
             num.parents=4,
             ploidy=4,
             dominance=1,
             alpha=0.05)

## End(Not run)
```

`diplo_freq` *Diplotype frequencies*

Description

Plot the frequency of individuals with diplotype dosage above a threshold

Usage

```
diplo_freq(data, diplotypes, dosage, position, chrom = NULL)
```

Arguments

`data`	Variable inheriting from class `diallel_geno`
`diplotypes`	Names of diplotypes
`dosage`	Dosage threshold
`position`	Either "cM" or "bp" for plotting
`chrom`	Names of chromosomes (default is all)

Details

Useful for visualizing selection in selfed populations.

Value

List containing

result Data frame with the map and frequency

plot ggplot object

`diplo_get` *Dosage of parental diplotypes*

Description

Dosage of parental diplotypes

Usage

```
diplo_get(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 diplotype 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 diplotypes

Examples

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

## 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

`fine_map` *Visualize haplotype switches for fine mapping*

Description

Visualize haplotype switches for fine mapping

Usage

```
fine_map(data, haplotype, interval, trait = NULL, marker = NULL)
```

Arguments

`data`	Variable inheriting from class `diallel_geno`
`haplotype`	Name of parental haplotype
`interval`	2-vector with marker names
`trait`	Name of trait to plot (optional)
`marker`	Optional, marker to indicate with dashed line

Details

Function returns graphic for all individuals with a haplotype switch (defined as change in dosage from 0 to ≥ 1 or vice versa) for `haplotype` within `interval`. If `trait` is included, the trait values for each individual are displayed on the right side. The function requires map positions in bp to be included in `data`.

Value

ggplot2 variable

Examples

```
## Not run:
  fine_map(data = diallel_example,
           haplotype = "W6511-1R.2",
           interval = c("solcap_snp_c2_40766","solcap_snp_c1_15225"))

  fine_map(data = diallel_example,
           haplotype = "W6511-1R.2",
           interval = c("solcap_snp_c2_40766","solcap_snp_c1_15225"),
           marker = "solcap_snp_c2_25522")

## End(Not run)
```

`fitQTL` *Fit multiple QTL model*

Description

Fit multiple QTL model

Usage

```
fitQTL(
  data,
  trait,
  qtl,
  epistasis = NULL,
  polygenic = FALSE,
  params = list(burnIn = 100, nIter = 5000),
  CI.prob = 0.9
)
```

Arguments

`data`	variable of class `diallel_geno_pheno`
`trait`	name of trait
`qtl`	data frame, see Details
`epistasis`	optional data frame, see Details
`polygenic`	TRUE/FALSE whether to include additive polygenic effect
`params`	list containing the number of burn-in (burnIn) and total iterations (nIter)
`CI.prob`	probability for Bayesian credible interval

Details

Argument `qtl` is a data frame with columns `marker` and `dominance` to specify the marker name and highest order effect (1 = additive, 2 = digenic dominance, 3 = trigenic dominance, 4 = quadrigenic dominance). All effects up to the value in `dominance` are included. Optional argument `epistasis` is a data frame with columns `marker1` and `marker2`, where each row specifies an additive x additive epistatic interaction. The number of burn-in and total iterations in `params` can be estimated using `set_params`. Parameter `CI.prob` sets the probability (e.g., 0.90, 0.95) for the Bayesian credible interval for the estimated effects (to disable plotting of the CI, use `CI.prob=NULL`).

Value

List containing

deltaDIC DIC relative to model with GCA but no QTL effects

resid residuals

var matrix with proportion of variance for the effects

effects list with two matrices, `additive` and `digenic`, with markers on the rows and effects on the columns

plots list of ggplot objects, one for each marker, containing elements `additive` and `digenic`. The digenic plot has digenic effects above the diagonal and the sum of additive and digenic effects below the diagonal.

Examples

```
## Not run:
## getting minimum burnIn and nIter for one qtl
set_params(data = diallel_example,
           trait = "tuber_shape",
           q = 0.05,
           r = 0.025,
           qtl = data.frame(marker="solcap_snp_c2_25522",dominance=2),
           polygenic = TRUE)

## additive effects
fit1 <- fitQTL(data = diallel_example,
               trait = "tuber_shape",
               params = list(burnIn=100,nIter=5000),
               qtl = data.frame(marker="solcap_snp_c2_25522",dominance=1),
               CI.prob = 0.9)

## additive + digenic dominance effects
fit2 <- fitQTL(data = diallel_example,
               trait = "tuber_shape",
               params = list(burnIn=100,nIter=5000),
               qtl = data.frame(marker="solcap_snp_c2_25522",dominance=2),
               CI.prob=0.9)

## getting minimum burnIn and nIter for two qtl with epistasis
set_params(data = diallel_example,
           trait = "tuber_shape",
           q = 0.05,
           r = 0.025,
           qtl = data.frame(marker=c("PotVar0099535","solcap_snp_c2_25522"),
                            dominance=c(2,1)),
          epistasis = data.frame(marker1="solcap_snp_c2_25522",marker2="PotVar0099535"),
           polygenic = TRUE)

## additive + digenic dominance effects for both QTL
fit3 <- fitQTL(data = diallel_example, trait = "tuber_shape",
               params = list(burnIn=100,nIter=5000),
               qtl = data.frame(marker=c("PotVar0099535","solcap_snp_c2_25522"),
                                dominance=c(2,2)),
               polygenic = TRUE, CI.prob = 0.9)

## additive + digenic dominance effects for both QTL + their epistatic effects
fit4 <- fitQTL(data = diallel_example, trait = "tuber_shape",
               params = list(burnIn=100,nIter=5000),
               qtl = data.frame(marker=c("PotVar0099535","solcap_snp_c2_25522"),
                                dominance=c(2,2)),
          epistasis = data.frame(marker1="solcap_snp_c2_25522",marker2="PotVar0099535"),
               polygenic = TRUE, CI.prob = 0.9)

## additive + digenic dominance effects for three QTL + all their epistatic effects
fit5 <- fitQTL(data = diallel_example, trait = "tuber_shape",
               params = list(burnIn=100,nIter=5000),
               qtl = data.frame(marker=c("PotVar0099535",
                                         "solcap_snp_c1_6427",
                                         "solcap_snp_c2_25522"),
                                dominance=c(2,2,2)),
               epistasis = data.frame(marker1=c("solcap_snp_c2_25522",
                                                "solcap_snp_c2_25522",
                                                "PotVar0099535"),
                                      marker2=c("PotVar0099535",
                                                "solcap_snp_c1_6427",
                                                "solcap_snp_c1_6427")),
               polygenic = TRUE, CI.prob = 0.9)

## End(Not run)
```

`get_map` *Get map summary from diallel_geno object*

Description

Get map summary from diallel_geno object

Usage

```
get_map(data, summary = TRUE)
```

Arguments

`data`	Variable inheriting from class `diallel_geno`
`summary`	logical, if TRUE (default) returns total sizes per chromosome, if FALSE returns the map

Value

data frame with map summary or the map

Examples

```
## Not run:
  get_map(diallel_example)

## End(Not run)
```

`haplo_cluster` *Cluster parental haplotypes*

Description

Cluster parental haplotypes

Usage

```
haplo_cluster(filename, marker, haplotypes = NULL)
```

Arguments

`filename`	Name of diaQTL_parents input file
`marker`	Either target marker or marker interval (see Details).
`haplotypes`	Vector of haplotype names (default is all)

Details

The argument `marker` can be either a single marker or vector of two markers. If a single marker, the function finds the smallest interval containing that marker such that the phased SNP haplotypes are all unique. If two markers are provided, that interval is used. Clustering utilizes hclust(method="average"). See also `phased_parents` for an additional visualization tool.

Value

List containing

haplo Data frame of haplotypes

dendro Dendrogram

`haplo_freq` *Haplotype frequencies*

Description

Plots the frequency of individuals with haplotype dosage above a threshold

Usage

```
haplo_freq(
  data,
  haplotypes,
  dosage,
  id = NULL,
  position = "cM",
  chrom = NULL,
  markers = NULL
)
```

Arguments

`data`	Variable inheriting from class `diallel_geno`
`haplotypes`	Names of haplotypes
`dosage`	Dosage threshold
`id`	Vector of id names (default is entire population)
`position`	Either "cM" (default) or "bp" for plotting
`chrom`	Names of chromosomes (default is all)
`markers`	Optional, markers to indicate with dashed line. Only available when plotting a single chromosome.

Details

Useful for visualizing selection in selfed populations. For multiple chromosomes, each haplotype is shown in its own panel using facet_wrap. For one chromosome, the haplotypes are shown on the same set of axes.

Value

List containing

result Data frame with the map and frequency

plot ggplot object

`haplo_get` *Dosage of parental haplotypes*

Description

Dosage of parental haplotypes

Usage

```
haplo_get(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 = haplo_get(data = diallel_example,
                            marker = "solcap_snp_c2_25522")
  haplo_example = haplo_get(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 = "cM", markers = NULL)
```

Arguments

data	Variable inheriting from class diallel_geno
id	Name of individual
chrom	Name of chromosome
position	Either "cM" (default) or "bp"
markers	Optional, markers 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)
```

IBDmat *Realized IBD relationship*

Description

Calculates realized relationship matrices from founder genotype probabilities

Usage

```
IBDmat(
  data,
  dominance = 1,
  epistasis = FALSE,
  spacing = 1,
  chrom = NULL,
  n.core = 1
)
```

Arguments

`data`	Variable inheriting from class `diallel_geno`
`dominance`	One of 1,2,3,4
`epistasis`	TRUE/FALSE
`spacing`	spacing between marker bins, in cM
`chrom`	Optional, vector of chromosome names to include
`n.core`	number of cores for parallel execution

Details

Parameter `dominance` refers to 1 = additive, 2 = digenic, 3 = trigenic, 4 = quadrigenic. Can specify to use only a subset of the chromosomes (by default, all chromosomes are used). If `epistasis` is TRUE, then `dominance` must be 1 (additive x additive epistasis). Only pairs of markers on different chromosomes are used for epistasis.

Value

Relationship matrix

Examples

```
## Not run:
  IBD_example = IBDmat(data = diallel_example, dominance=1) #additive
  IBD_example = IBDmat(data = diallel_example, dominance=2) #digenic dominance
 IBD_example = IBDmat(data = diallel_example, epistasis=TRUE) #additive x additive epistasis

## End(Not run)
```

`phased_parents` *Visualize phased SNPs of parents*

Description

Visualize phased SNPs of parents

Usage

```
phased_parents(filename, interval, markers, parents)
```

Arguments

`filename`	Name of CSV input file
`interval`	Vector of length 2 with the first and last marker names
`markers`	Vector of marker names to plot
`parents`	Vector of parent names to plot

Details

The solid circles in the figure represent the allele counted by dosage.

Value

ggplot2 object

`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 = TRUE,
  dominance = NULL,
  n.core = 1
)
```

Arguments

`genofile`	File with map and genotype probabilities
`ploidy`	Either 2 or 4
`pedfile`	File with pedigree data (id,parent1,parent2)
`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`	Maximum value of dominance that will be used for analysis. Default = ploidy.
`n.core`	Number of cores for parallel execution

Details

The first 3 columns of the genotype file should be the genetic map (labeled marker, chrom, cM), and a fourth column for a reference genome position (labeled bp) can also be included. The map is followed by the members of the population. The genotype data 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. Missing data in the phenotype file should be coded as NA. The parameter `dominance` specifies the maximum value of dominance that can be used in subsequent analysis: 1 = additive, 2 = digenic dominance, 3 = trigenic dominance, 4 = quadrigenic dominance. The default is dominance = ploidy, which allows the full range of dominance models in functions such as `scan1` and `fitQTL`, but this requires the most RAM. Output files from the BGLR package are stored in a folder named 'tmp' in the current directory.

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( "vignette_data", "potato_geno.csv", package = "diaQTL" )
  pedcsv = system.file( "vignette_data", "potato_ped.csv", package = "diaQTL" )
  phenocsv = system.file( "vignette_data", "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 = list(burnIn = 100, nIter = 1000),
  dominance = 1,
  covariate = NULL,
  chrom = NULL,
  n.core = 1
)
```

Arguments

`data`	variable of class `diallel_geno_pheno`
`trait`	name of trait
`params`	list containing burnIn and nIter
`dominance`	maximum dominance for the scan, see Details
`covariate`	optional, to include markers as covariates. See Details
`chrom`	names of chromosomes to scan (default is all)
`n.core`	number of cores for parallel execution

Details

Parameter `dominance` has possible values of 1 = additive, 2 = digenic dominance, 3 = trigenic dominance, 4 = quadrigenic dominance. MCMC `params` can be estimated using `set_params`. Optional argument `covariate` is used to include other markers in the model during the scan, which can improve statistical power with multiple QTL. It is a data frame with three columns: marker = name of the marker, dominance = 1 to 4, and epistasis = TRUE/FALSE. Function returns deltaDIC = DIC for the QTL model relative to null model with only GCA effects for the parents, as well as LL = posterior mean of the log-likelihood, which is used by `BayesCI`.

Value

Data frame containing the map, LL, and deltaDIC.

Examples

```
## Not run:
## getting minimum burnIn and nIter
  set_params(data = diallel_example,
             trait = "tuber_shape")

  scan1_example <- scan1(data = diallel_example,
                         chrom = "10",
                         trait = "tuber_shape",
                         params = list(burnIn=60,nIter=600))

## 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,
  covariate = 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)
covariate	optional, to include markers as covariates. See Details.
n.core	Number of cores for parallel execution

Value

Data frame with maximum LOD and minimum deltaDIC for each iteration

Examples

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

  ans1_permut <- scan1_permute(data = diallel_example,
                               chrom = 10,
                               trait = "tuber_shape",
                               params = list(burnIn=60,nIter=600),
                               n.permute = 100)

  ## computing permutation threshold for alpha=0.05
  quantile(ans1_permut$min_deltaDIC, 0.05)


## End(Not run)
```

scan1_summary   *Summary of scan1 result*

Description

Summary of scan1 result

Usage

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

Arguments

scan1_data	output from scan1
thresh	optional -deltaDIC threshold for plotting
chrom	optional, subset of chromosomes to plot
position	Either "cM" (default) or "bp"

Details

Plots the "score" (-deltaDIC) for each marker vs. genome position. The `thresh` argument should be a positive number.

Value

List containing

peaks data frame of markers with the highest score on each 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" ) )
  scan1_summary( scan1_example, chrom = "10", thresh = 20)

## End(Not run)
```

`set_params` *Determine number of iterations for MCMC*

Description

Determine number of iterations for MCMC

Usage

```
set_params(
  data,
  trait,
  qtl = NULL,
  epistasis = NULL,
  polygenic = FALSE,
  q = 0.5,
  r = 0.1,
  nIter = 2000
)
```

Arguments

`data`	variable of class `diallel_geno_pheno`
`trait`	name of trait
`qtl`	optional data frame, see Examples
`epistasis`	optional data frame, see Example
`polygenic`	TRUE/FALSE whether to include additive polygenic effect
`q`	quantile to estimate
`r`	tolerance for quantile
`nIter`	number of iterations

Details

Determines the burn-in and total number of iterations using the Raftery and Lewis diagnostic from R package coda, based on a 95% probability that the estimate for quantile `q` of the additive genetic variance is within the interval `(q-r,q+r)`. If `marker=NULL` (default), the first marker of each chromosome is analyzed, and the largest value across this set is returned. Parameter `dominance` specifies which genetic model (1 = additive, 2 = digenic dominance, 3 = trigenic dominance, 4 = quadrigenic dominance) to use when determining the number of iterations, but this parameter must still be specified when calling functions such as `scan1` or `fitQTL`. The default values of q=0.5 and r=0.1 are recommended for `scan1` based on the idea of estimating the posterior mean. For estimating the 90% Bayesian CI with `fitQTL`, suggested values are q=0.05, r=0.025. Parameter `nIter` sets the number of iterations used to apply the Raftery and Lewis diagnostic; the default value is 2000, and if a larger number is needed, an error will be generated with this information.

Value

matrix showing the number of burn-in and total iterations for the genetic variances in the model

Examples

```
## Not run:
  # Parameters for scan1
  par1 <- set_params(data = diallel_example,
                     trait = "tuber_shape",
                     q=0.5,
                     r=0.1)

  # Parameters for fitQTL (specify the position)
  set_params(data = diallel_example,
             trait = "tuber_shape",
             q=0.05,
             r=0.025,
             qtl=data.frame(marker="solcap_snp_c2_25522",dominance=2))

  # Parameters for fitQTL (specify the position) with polygenic effects
  set_params(data = diallel_example,
             trait = "tuber_shape",
             q=0.05,
             r=0.025,
             qtl=data.frame(marker="solcap_snp_c2_25522",dominance=2),
             polygenic=TRUE)

  # Parameters for fitQTL with 2 QTLs
  set_params(data = diallel_example,
             trait = "tuber_shape",
             q=0.05,
             r=0.025,
         qtl=data.frame(marker=c("solcap_snp_c2_25522","solcap_snp_c2_14750"),dominance=c(2,1)))

  # Parameters for fitQTL with epistasis
  set_params(data = diallel_example,
             trait = "tuber_shape",
             q=0.05,
             r=0.025,
         epistasis = data.frame(marker1="solcap_snp_c2_25522",marker2="solcap_snp_c2_14750"),
         qtl=data.frame(marker=c("solcap_snp_c2_25522","solcap_snp_c2_14750"),dominance=c(2,1)))
```

```
## End(Not run)
```

Index