Package 'qtl2pleio'

October 13, 2022

Type Package

Title Testing Pleiotropy in Multiparental Populations

Version 1.4.3

Description We implement an
adaptation of Jiang & Zeng's (1995) <https://www.genetics.org/content/140/3/1111> likelihood ratio test for testing
the null hypothesis of pleiotropy against the alternative hypothesis,
two separate quantitative trait loci. The test differs from that in Jiang & Zeng (1995) <https://www.genetics.org/content/140/3/1111>
and that in Tian et al. (2016) <doi:10.1534/genetics.115.183624> in
that our test accommodates multiparental populations.

License MIT + file LICENSE

URL https://github.com/fboehm/qtl2pleio

BugReports https://github.com/fboehm/qtl2pleio/issues

Depends R (>= 3.2)

Imports dplyr, gemma2, ggplot2, magrittr, MASS, Rcpp, rlang, tibble,
parallel

Suggests covr, mvtnorm, knitr, rmarkdown, testthat, broman, devtools,
qtl2, parallelly

LinkingTo Rcpp, RcppEigen

Encoding UTF-8

Language en-US

LazyData true

NeedsCompilation yes

RoxygenNote 7.1.1

VignetteBuilder knitr

Author Frederick J Boehm [aut, cre] (<https://orcid.org/0000-0002-1644-5931>)

Maintainer Frederick J Boehm <frederick.boehm@gmail.com>

Repository CRAN

Date/Publication 2020-12-02 22:50:02 UTC

R topics documented:

`add_pmap` *Add physical map contents to tibble*

Description

Add physical map contents to tibble

Usage

```
add_pmap(tib, pmap)
```

Arguments

tib	a tibble with 3 columns: marker, trace, and profile lod values, typically outputted by calc_profile_lods()
pmap	a physical map for a single chromosome

Value

a tibble with 4 columns: marker, trait, profile_lod, marker_position

Examples

```
pm <- 1:3
names(pm) <- as.character(paste0('m', 1:3))
expand.grid(paste0('m', 1:3), paste0('m', 1:3)) %>%
    tibble::as_tibble() %>%
    dplyr::mutate(log10lik = rgamma(9, 5)) %>%
    calc_profile_lods() %>%
    add_pmap(pm)
```

---

boot_pvl *Perform bootstrap sampling and calculate test statistic for each bootstrap sample*

Description

Create a bootstrap sample, perform multivariate QTL scan, and calculate log10 LRT statistic

Usage

```
boot_pvl(
  probs,
  pheno,
  addcovar = NULL,
  kinship = NULL,
  start_snp = 1,
  n_snp,
  pleio_peak_index,
  nboot = 1,
  max_iter = 10000,
  max_prec = 1/1e+08,
  cores = 1
)
```

Arguments

probs	founder allele probabilities three-dimensional array for one chromosome only (not a list)
pheno	n by d matrix of phenotypes
addcovar	n by c matrix of additive numeric covariates
kinship	a kinship matrix, not a list
start_snp	positive integer indicating index within probs for start of scan
n_snp	number of (consecutive) markers to use in scan
pleio_peak_index	positive integer index indicating genotype matrix for bootstrap sampling. Typically acquired by using ‘find_pleio_peak_tib‘.
nboot	number of bootstrap samples to acquire and scan
max_iter	maximum number of iterations for EM algorithm
max_prec	stepwise precision for EM algorithm. EM stops once incremental difference in log likelihood is less than max_prec
cores	number of cores to use when calling mclapply to parallelize the bootstrap analysis.

Details

Performs a parametric bootstrap method to calibrate test statistic values in the test of pleiotropy vs. separate QTL. It begins by inferring parameter values at the ‘pleio_peak_index‘ index value in the object ‘probs‘. It then uses these inferred parameter values in sampling from a multivariate normal distribution. For each of the ‘nboot‘ sampled phenotype vectors, a two-dimensional QTL scan, starting at the marker indexed by ‘start_snp‘ within the object ‘probs‘ and extending for a total of ‘n_snp‘ consecutive markers. The two-dimensional scan is performed via the function ‘scan_pvl_clean‘. For each two-dimensional scan, a log10 likelihood ratio test statistic is calculated. The outputted object is a vector of ‘nboot‘ log10 likelihood ratio test statistics from ‘nboot‘ distinct bootstrap samples.

Value

numeric vector of (log) likelihood ratio test statistics from ‘nboot_per_job‘ bootstrap samples

References

Knott SA, Haley CS (2000) Multitrait least squares for quantitative trait loci detection. Genetics 156: 899–911.

Walling GA, Visscher PM, Haley CS (1998) A comparison of bootstrap methods to construct confidence intervals in QTL mapping. Genet. Res. 71: 171–180.

Examples

```
n <- 50
pheno <- matrix(rnorm(2 * n), ncol = 2)
rownames(pheno) <- paste0("s", 1:n)
colnames(pheno) <- paste0("tr", 1:2)
probs <- array(dim = c(n, 2, 5))
probs[ , 1, ] <- rbinom(n * 5, size = 1, prob = 0.2)
probs[ , 2, ] <- 1 - probs[ , 1, ]
rownames(probs) <- paste0("s", 1:n)
colnames(probs) <- LETTERS[1:2]
dimnames(probs)[[3]] <- paste0("m", 1:5)
boot_pvl(probs = probs, pheno = pheno,
        start_snp = 1, n_snp = 5, pleio_peak_index = 3, nboot = 1, cores = 1)
```

`calc_Bhat` *Calculate estimated allele effects, B matrix*

Description

Calculate estimated allele effects, B matrix

Usage

```
calc_Bhat(X, Sigma_inv, Y)
```

Arguments

`X`	dn by df block-diagonal design matrix that incorporates genetic info for d markers. Note that we can use the same marker data twice.
`Sigma_inv`	dn by dn inverse covariance matrix, often composed as the inverse of $K \otimes V_g + I_n \otimes V_e$
`Y`	dn by 1 matrix, ie, a column vector, of d phenotypes' measurements

Value

a df by 1 matrix of GLS-estimated allele effects

Examples

```
X1 <- as.matrix(rbinom(n = 100, size = 1, prob = 1 / 2))
X <- gemma2::stagger_mats(X1, X1)
Sigma_inv <- diag(200)
Y <- runif(200)
calc_Bhat(X, Sigma_inv, Y)
```

calc_covs *Calculate Vg and Ve from d-variate phenotype and kinship*

Description

Calculate Vg and Ve from d-variate phenotype and kinship

Usage

```
calc_covs(
  pheno,
  kinship,
  X1pre = rep(1, nrow(kinship)),
  max_iter = 1e+06,
  max_prec = 1/1e+08,
  covariates = NULL
)
```

Arguments

pheno	n by d matrix of phenotypes
kinship	a kinship matrix, n by n
X1pre	n by c design matrix. c = 1 to ignore genotypes
max_iter	maximum number of EM iterations
max_prec	maximum precision for stepwise increments in EM algorithm
covariates	a n by n.cov matrix of numeric covariates

Value

a list with 2 named components, Vg and Ve. Each is a d by d covariance matrix.

Examples

```
calc_covs(pheno = matrix(data = rnorm(100), nrow = 50, ncol = 2), kinship = diag(50))
```

calc_invsqrt_mat *Calculate matrix inverse square root for a covariance matrix*

Description

Calculate matrix inverse square root for a covariance matrix

Usage

```
calc_invsqrt_mat(A)
```

Arguments

A	covariance matrix

`calc_lrt_tib` *Calculate a likelihood ratio test statistic from the output of scan_pvl()*

Description

Calculate a likelihood ratio test statistic from the output of scan_pvl()

Usage

```
calc_lrt_tib(scan_pvl_out)
```

Arguments

scan_pvl_out	outputted tibble from scan_pvl

Value

a number, the (log) likelihood ratio test statistic

Examples

```
rep(paste0('Marker', 1:3), times = 3) -> marker1
rep(paste0('Marker', 1:3), each = 3) -> marker2
runif(9, -1, 0) -> ll
tibble::tibble(marker1, marker2, ll) -> scan_out
calc_lrt_tib(scan_out)
```

`calc_profile_lods` *Calculate profile lods for all traits*

Description

Calculate profile lods for all traits

Usage

```
calc_profile_lods(scan_pvl_out)
```

Arguments

scan_pvl_out	tibble outputted from scan_pvl

Value

a tibble with 3 columns, indicating 'marker identity, trace (pleiotropy or profile1, profile2, etc.), and value of the profile lod (base 10) for that trace at that marker.

calc_Sigma *Calculate the phenotypes covariance matrix Sigma*

Description

Calculate the phenotypes covariance matrix Sigma

Usage

```
calc_Sigma(Vg, Ve, kinship = NULL, n_mouse = nrow(kinship))
```

Arguments

Vg	d by d genetic covariance matrix for the d phenotypes
Ve	d by d error covariance matrix for the d phenotypes
kinship	optional n by n kinship matrix. if NULL, Vg is not used.
n_mouse	number of subjects

Value

dn by dn covariance matrix

calc_sqrt_mat *Calculate matrix square root for a covariance matrix*

Description

Calculate matrix square root for a covariance matrix

Usage

```
calc_sqrt_mat(A)
```

Arguments

A	covariance matrix

check_identical *Check whether a vector, x, has all its entries equal to its first entry*

Description

Check whether a vector, x, has all its entries equal to its first entry

Usage

```
check_identical(x)
```

Arguments

x	a vector

Value

a logical indicating whether all vector entries are the same

Examples

```
x <- 1:5
check_identical(x)
y <- rep(1, 5)
check_identical(y)
```

check_missingness *Check for missingness in phenotypes or covariates*

Description

We use ‘is.finite‘ from base R to identify those subjects that have one or more missing values in ‘input_matrix‘. We then return a character vector of subjects that have no missingness in ‘input_matrix‘.

Usage

```
check_missingness(input_matrix)
```

Arguments

input_matrix	phenotypes or covariates matrix

Value

character vector of subjects that have no missingness

convert_to_scan1_output *Convert ‘scan_multi_oneqtl‘ output of ‘qtl2::scan1‘ output*

Description

We convert output of ‘scan_multi_oneqtl‘ into format outputted by ‘qtl2::scan1‘.

Usage

```
convert_to_scan1_output(sm_output, trait_name)
```

Arguments

sm_output	tibble output from scan_multi_oneqtl for one chromosome only
trait_name	character vector (of length one) specifying the trait names

Value

object of class ‘scan1‘

Examples

```
# read data
iron <- qtl2::read_cross2(system.file("extdata", "iron.zip", package="qtl2"))


# insert pseudomarkers into map
map <- qtl2::insert_pseudomarkers(iron$gmap, step=1)

# calculate genotype probabilities
probs <- qtl2::calc_genoprob(iron, map, error_prob=0.002)

# grab phenotypes and covariates; ensure that covariates have names attribute
pheno <- iron$pheno
covar <- match(iron$covar$sex, c("f", "m")) # make numeric
names(covar) <- rownames(iron$covar)
Xcovar <- qtl2::get_x_covar(iron)

aprobs <- qtl2::genoprob_to_alleleprob(probs)
sm_out <- scan_multi_oneqtl(probs = aprobs, pheno = pheno)
sm_to_s1 <- convert_to_scan1_output(sm_out[[1]], trait_name = "tr1and2")

# 95% Bayes credible interval for QTL on chr 7, first phenotype
qtl2::bayes_int(sm_to_s1, map)
```

find_pleio_peak_tib *Find the marker index corresponding to the peak of the pleiotropy trace in a tibble where the last column contains log likelihood values and the first d columns contain marker ids*

Description

Find the marker index corresponding to the peak of the pleiotropy trace in a tibble where the last column contains log likelihood values and the first d columns contain marker ids

Usage

```
find_pleio_peak_tib(tib, start_snp)
```

Arguments

tib	a (d+1) column tibble with first d columns containing marker ids and the last containing log likelihood values. Typically this is the output from 'scan_pvl'.
start_snp	positive integer, from the two-dimensional scan, that indicates where the scan started on the chromosome

Value

positive integer indicating marker index for maximum value of log lik under pleiotropy

Examples

```
marker1 <- rep(paste0('SNP', 1:3), times = 3)
marker2 <- rep(paste0('SNP', 1:3), each = 3)
loglik <- runif(9, -5, 0)
tibble::tibble(marker1, marker2, loglik) -> tib
find_pleio_peak_tib(tib, start_snp = 1)
```

fit1_pvl *Fit a model for a specified d-tuple of markers*

Description

'fit1_pvl' uses several functions in the package qtl2pleio to fit the linear mixed effects model for a single d-tuple of markers. Creation of 'fit1_pvl' - from code that originally resided in 'scan_pvl', enabled parallelization via the 'parallel' R package.

Usage

```
fit1_pvl(indices, start_snp, probs, addcovar, inv_S, S, pheno)
```

Arguments

indices	a vector of indices for extracting elements of ‘probs‘ array
start_snp	an integer to specify the index of the marker where the scan - in call to scan_pvl - starts. This argument is needed because ‘mytab‘ has only relative indices (relative to the ‘start_snp‘ marker)
probs	founder allele probabilities array
addcovar	additive covariates matrix
inv_S	inverse covariance matrix for the vectorized phenotype
S	covariance matrix for the vectorized phenotype, ie, the inverse of inv_S. By making this a function input, we avoid inverting the matrix many many times.
pheno	a n by d phenotypes matrix

Value

a number, the log-likelihood for the specified model

Examples

```
n <- 50
pheno <- matrix(rnorm(2 * n), ncol = 2)
Vg <- diag(2)
Ve <- diag(2)
Sigma <- calc_Sigma(Vg, Ve, diag(n))
Sigma_inv <- solve(Sigma)
probs <- array(dim = c(n, 2, 5))
probs[ , 1, ] <- rbinom(n * 5, size = 1, prob = 0.2)
probs[ , 2, ] <- 1 - probs[ , 1, ]
mytab <- prep_mytab(d_size = 2, n_snp = 5)
fit1_pvl(mytab[1, ], start_snp = 1,
probs = probs, addcovar = NULL, inv_S = Sigma_inv,
S = Sigma,
pheno = pheno
)
```

---

get_effects *Extract founder allele effects at a single marker from output of qtl2::scan1coef*

Description

Extract founder allele effects at a single marker from output of qtl2::scan1coef

Usage

```
get_effects(marker_index, allele_effects_matrix, map, columns = 1:8)
```

Arguments

marker_index	an integer indicating where in the ‘map‘ object the peak position (or position of interest) is located
allele_effects_matrix	output of ‘qtl2::scan1coef‘ for a single chromosome
map	a map object for the chromosome of interest
columns	which columns to choose within the ‘allele_effects_matrix‘. Default is 1:8 to reflect 8 founder alleles of Diversity Outbred mice

Value

a vector of 8 founder allele effects at a single marker

a vector of founder allele effects at a single marker

Examples

```
# set up allele effects matrix
ae <- matrix(dat = rnorm(100 * 8), ncol = 8, nrow = 100)
ae[, 8] <- - rowSums(ae[, 1:7])
colnames(ae) <- LETTERS[1:8]
rownames(ae) <- paste0(1, "_", 1:100)
# set up map
map <- 1:100
names(map) <- rownames(ae)
# call get_effects
get_effects(marker_index = 15, allele_effects_matrix = ae, map = map)
```

---

make_id2keep *Identify shared subject ids among all inputs: covariates, allele probabilities array, kinship, and phenotypes*

Description

We consider only those inputs that are not NULL. We then use ‘intersect‘ on pairs of inputs’ rownames to identify those subjects are shared among all non-NULL inputs.

Usage

```
make_id2keep(probs, pheno, addcovar = NULL, kinship = NULL)
```

Arguments

probs	an allele probabilities array
pheno	a phenotypes matrix
addcovar	a covariates matrix
kinship	a kinship matrix

Value

a character vector of subject IDs common to all (non-null) inputs

---

`plot_pvl` *Plot tidied results of a pvl scan*

Description

Plot tidied results of a pvl scan

Usage

```
plot_pvl(
  dat,
  units = "Mb",
  palette = c("#999999", "#E69F00", "#56B4E9"),
  linetype = c("solid", "longdash", "dotted")
)
```

Arguments

`dat`	a profile lod tibble
`units`	a character vector of length one to indicate units for physical or genetic map
`palette`	a character vector of length 3 containing strings for colors
`linetype`	a character vector of length 3 specifying the linetype values for the 3 traces

Value

a ggplot object with profile LODs

`prep_mytab` *Prepare mytab object for use within scan_pvl R code*

Description

Prepare mytab object for use within scan_pvl R code

Usage

```
prep_mytab(d_size, n_snp, pvl = TRUE)
```

Arguments

d_size	an integer, the number of traits
n_snp	an integer, the number of markers
pvl	logical indicating whether to output dataframe with all d-tuples for a d-QTL scan, or only those models that examine one marker at a time.

Value

a data.frame with d_size + 1 columns and (n_snp)^d_size rows. Last column is NA and named loglik.

Examples

```
prep_mytab(2, 10)
```

---

prep_X_list *Create a list of component X matrices for input to stagger_mats, to ultimately create design matrix*

Description

Create a list of component X matrices for input to stagger_mats, to ultimately create design matrix

Usage

```
prep_X_list(indices, start_snp, probs, covariates)
```

Arguments

indices	a vector of integers
start_snp	an integer denoting the index (within genotype probabilities array) where the scan should start
probs	a three-dimensional array of genotype probabilities for a single chromosome
covariates	a matrix of covariates

Value

a list of design matrices, ultimately useful when constructing the (multi-locus) design matrix

Examples

```
pp <- array(rbinom(n = 200, size = 1, prob = 0.5), dim = c(10, 2, 10))
prep_X_list(1:3, 1, probs = pp, covariates = NULL)
```

process_inputs *Process inputs to scan functions*

Description

Process inputs to scan functions

Usage

```
process_inputs(
  probs,
  pheno,
  addcovar,
  kinship,
  n_snp = dim(probs)[3],
  start_snp = 1,
  max_iter = 10^4,
  max_prec = 1/10^8
)
```

Arguments

probs	a three-dimensional array of founder allele probabilities
pheno	a matrix of d trait values
addcovar	a matrix of covariates
kinship	a kinship matrix
n_snp	number of markers
start_snp	index number of start position in the probs object.
max_iter	max number of iterations for EM
max_prec	max precision for stopping EM

qtl2pleio *qtl2pleio.*

Description

Testing pleiotropy vs. separate QTL in multiparental populations

rcpp_calc_Bhat *Estimate allele effects matrix, B hat, with Rcpp functions*

Description

Estimate allele effects matrix, B hat, with Rcpp functions

Usage

```
rcpp_calc_Bhat(X, Sigma_inv, Y)
```

Arguments

X	dn by df block-diagonal design matrix that incorporates genetic info for two markers. Note that we can use the same marker data twice.
Sigma_inv	dn by dn inverse covariance matrix, where its inverse, ie, Sigma, is often composed as $K \otimes V_g + I_n \otimes V_e$
Y	dn by 1 matrix, ie, a column vector, of d phenotypes' measurements

Value

a df by 1 matrix of GLS-estimated allele effects

Examples

```
X1 <- as.matrix(rbinom(n = 100, size = 1, prob = 1 / 2))
X <- gemma2::stagger_mats(X1, X1)
Sigma_inv <- diag(200)
Y <- runif(200)
rcpp_calc_Bhat(X = X, Sigma_inv = Sigma_inv, Y = Y)
```

rcpp_calc_Bhat2 *Estimate allele effects matrix, B hat, with Rcpp functions*

Description

Estimate allele effects matrix, B hat, with Rcpp functions

Usage

```
rcpp_calc_Bhat2(X, Y, Sigma_inv)
```

Arguments

X	dn by df block-diagonal design matrix that incorporates genetic info for two markers. Note that we can use the same marker data twice.
Y	dn by 1 matrix, ie, a column vector, of d phenotypes' measurements
Sigma_inv	dn by dn inverse covariance matrix, often composed as inverse of $K \otimes V_g + I_n \otimes V_g$

Value

a df by 1 matrix of GLS-estimated allele effects

Examples

```
X1 <- as.matrix(rbinom(n = 100, size = 1, prob = 1 / 2))
X <- gemma2::stagger_mats(X1, X1)
Sigma_inv <- diag(200)
Y <- runif(200)
rcpp_calc_Bhat2(X = X, Y = Y, Sigma_inv = Sigma_inv)
```

---

rcpp_log_dmvnorm2 *Calculate log likelihood for a multivariate normal*

Description

Calculate log likelihood for a multivariate normal

Usage

```
rcpp_log_dmvnorm2(inv_S, mu, x, S)
```

Arguments

inv_S	inverse covariance matrix
mu	mean vector
x	data vector
S	covariance matrix, ie, the inverse of inv_S

`scan_multi_onechr` *Perform multivariate, one-QTL model fitting for markers on one chromosome*

Description

'scan_multi_onechr' calculates log likelihood for d-variate phenotype model fits. Inputted parameter 'start_snp' indicates where in the 'probs' object to start the scan.

Usage

```
scan_multi_onechr(
  probs,
  pheno,
  kinship = NULL,
  addcovar = NULL,
  start_snp = 1,
  n_snp = dim(probs)[3],
  max_iter = 10000,
  max_prec = 1/1e+08,
  cores = 1
)
```

Arguments

`probs`	an array of founder allele probabilities for a single chromosome
`pheno`	a matrix of phenotypes
`kinship`	a kinship matrix for one chromosome
`addcovar`	a matrix, n subjects by c additive covariates
`start_snp`	index of where to start the scan within probs
`n_snp`	the number of (consecutive) markers to include in the scan
`max_iter`	maximum number of iterations for EM algorithm
`max_prec`	stepwise precision for EM algorithm. EM stops once incremental difference in log likelihood is less than max_prec
`cores`	number of cores for parallelization

Value

a tibble with d + 1 columns. First d columns indicate the genetic data (by listing the marker ids) used in the design matrix; last is log10 likelihood

References

Knott SA, Haley CS (2000) Multitrait least squares for quantitative trait loci detection. Genetics 156: 899–911.

Jiang C, Zeng ZB (1995) Multiple trait analysis of genetic mapping for quantitative trait loci. Genetics 140: 1111-1127.

Zhou X, Stephens M (2014) Efficient multivariate linear mixed model algorithms for genome-wide association studies. Nature methods 11:407-409.

Broman KW, Gatti DM, Simecek P, Furlotte NA, Prins P, Sen S, Yandell BS, Churchill GA (2019) R/qtl2: software for mapping quantitative trait loci with high-dimensional data and multi-parent populations. GENETICS https://www.genetics.org/content/211/2/495.

Examples

```
# read data
n <- 50
pheno <- matrix(rnorm(2 * n), ncol = 2)
rownames(pheno) <- paste0("s", 1:n)
colnames(pheno) <- paste0("tr", 1:2)
probs <- array(dim = c(n, 2, 5))
probs[ , 1, ] <- rbinom(n * 5, size = 1, prob = 0.2)
probs[ , 2, ] <- 1 - probs[ , 1, ]
rownames(probs) <- paste0("s", 1:n)
colnames(probs) <- LETTERS[1:2]
dimnames(probs)[[3]] <- paste0("m", 1:5)
scan_multi_onechr(probs = probs, pheno = pheno, kinship = NULL, cores = 1)
```

`scan_multi_oneqtl` *Perform multivariate, one-QTL model fitting for markers on all chromosomes*

Description

The function first discards individuals with one or more missing phenotypes or missing covariates. It then infers variance components, Vg and Ve. Both Vg and Ve are d by d covariance matrices. It uses an expectation maximization algorithm, as implemented in the 'gemma2' R package. 'gemma2' R package is an R implementation of the GEMMA algorithm for multivariate variance component estimation (Zhou & Stephens 2014 Nature methods). Note that variance components are fitted on a model that uses the d-variate phenotype but contains no genetic information. This model does, however, use the specified covariates (after dropping dependent columns in the covariates matrix). These inferred covariance matrices, $\hat{Vg}$ and $\hat{Ve}$, are then used in subsequent model fitting via generalized least squares. Generalized least squares model fitting is applied to every marker on every chromosome. For a single marker, we fit the model:

$$vec(Y) = Xvec(B) + vec(G) + vec(E)$$

where

$$G \sim MN(0, K, \hat{Vg})$$

and

$$E \sim MN(0, I, \hat{V}e)$$

where MN denotes the matrix-variate normal distribution with three parameters: mean matrix, covariance among rows, and covariance among columns. vec denotes the vectorization operation, ie, stacking by columns. K is a kinship matrix, typically calculated by leave-one-chromosome-out methods. Y is the n by d phenotypes matrix. X is a block-diagonal nd by fd matrix consisting of d blocks each of dimension n by f. Each n by f block (on the diagonal) contains a matrix of founder allele probabilities for the n subjects at a single marker. The off-diagonal blocks have only zero entries. The log-likelihood is returned for each model. The outputted object is a tibble with d + 1 columns. The first d columns specify the markers used in the corresponding model fit, while the last column specifies the log-likelihood value at that d-tuple of markers.

Usage

```
scan_multi_oneqtl(
  probs_list,
  pheno,
  kinship_list = NULL,
  addcovar = NULL,
  cores = 1
)
```

Arguments

probs_list	an list of arrays of founder allele probabilities
pheno	a matrix of phenotypes
kinship_list	a list of kinship matrices, one for each chromosome
addcovar	a matrix, n subjects by c additive covariates
cores	number of cores for parallelization via parallel::mclapply()

Value

a tibble with d + 1 columns. First d columns indicate the genetic data (by listing the marker ids) used in the design matrix; last is log10 likelihood

References

Knott SA, Haley CS (2000) Multitrait least squares for quantitative trait loci detection. Genetics 156: 899–911.

Jiang C, Zeng ZB (1995) Multiple trait analysis of genetic mapping for quantitative trait loci. Genetics 140: 1111-1127.

Zhou X, Stephens M (2014) Efficient multivariate linear mixed model algorithms for genome-wide association studies. Nature methods 11:407-409.

Broman KW, Gatti DM, Simecek P, Furlotte NA, Prins P, Sen S, Yandell BS, Churchill GA (2019) R/qtl2: software for mapping quantitative trait loci with high-dimensional data and multi-parent populations. GENETICS https://www.genetics.org/content/211/2/495.

Examples

```
# read data
n <- 50
pheno <- matrix(rnorm(2 * n), ncol = 2)
rownames(pheno) <- paste0("s", 1:n)
colnames(pheno) <- paste0("tr", 1:2)
probs <- array(dim = c(n, 2, 5))
probs[ , 1, ] <- rbinom(n * 5, size = 1, prob = 0.2)
probs[ , 2, ] <- 1 - probs[ , 1, ]
rownames(probs) <- paste0("s", 1:n)
colnames(probs) <- LETTERS[1:2]
dimnames(probs)[[3]] <- paste0("m", 1:5)
scan_multi_oneqtl(probs_list = list(probs, probs), pheno = pheno, cores = 1)
```

`scan_multi_oneqtl_perm`

Permute the phenotypes matrix and then scan the genome. Record the genomewide greatest LOD score for each permuted data set.

Description

Permute the phenotypes matrix and then scan the genome. Record the genomewide greatest LOD score for each permuted data set.

Usage

```
scan_multi_oneqtl_perm(
  probs_list,
  pheno,
  kinship_list = NULL,
  addcovar = NULL,
  n_perm = 1,
  cores = 1
)
```

Arguments

`probs_list`	a list of founder allele probabilities, one array per chromosome
`pheno`	a matrix of trait values
`kinship_list`	a list of kinship matrices, one per chromosome
`addcovar`	a matrix of covariate values
`n_perm`	positive integer for the number of permuted data sets to scan.
`cores`	number of cores for parallelization

Value

a vector of ‘n_perm‘ max lod statistics

scan_pvl *Perform model fitting for all ordered pairs of markers in a genomic region of interest*

Description

'scan_pvl' calculates log likelihood for d-variate phenotype model fits. Inputted parameter 'start_snp' indicates where in the 'probs' object to start the scan.

Usage

```
scan_pvl(
  probs,
  pheno,
  kinship = NULL,
  addcovar = NULL,
  start_snp = 1,
  n_snp,
  max_iter = 10000,
  max_prec = 1/1e+08,
  cores = 1
)
```

Arguments

probs	an array of founder allele probabilities for a single chromosome
pheno	a matrix of phenotypes
kinship	a kinship matrix for one chromosome
addcovar	a matrix, n subjects by c additive covariates
start_snp	index of where to start the scan within probs
n_snp	the number of (consecutive) markers to include in the scan
max_iter	maximum number of iterations for EM algorithm
max_prec	stepwise precision for EM algorithm. EM stops once incremental difference in log likelihood is less than max_prec
cores	number of cores to use when parallelizing via parallel::mclapply. Set to 1 for no parallelization.

Details

The function first discards individuals with one or more missing phenotypes or missing covariates. It then infers variance components, Vg and Ve. Both Vg and Ve are d by d covariance matrices. It uses an expectation maximization algorithm, as implemented in the 'gemma2' R package. 'gemma2' R package is an R implementation of the GEMMA algorithm for multivariate variance component estimation (Zhou & Stephens 2014 Nature methods). Note that variance components are fitted on a model that uses the d-variate phenotype but contains no genetic information. This model does,

however, use the specified covariates (after dropping dependent columns in the covariates matrix). These inferred covariance matrices, $\hat{V}g$ and $\hat{V}e$, are then used in subsequent model fitting via generalized least squares. Generalized least squares model fitting is applied to every d-tuple of markers within the specified genomic region for 'scan_pvl'. For a single d-tuple of markers, we fit the model:

$$vec(Y) = Xvec(B) + vec(G) + vec(E)$$

where

$$G \sim MN(0, K, \hat{V}g)$$

and

$$E \sim MN(0, I, \hat{V}e)$$

where MN denotes the matrix-variate normal distribution with three parameters: mean matrix, covariance among rows, and covariance among columns. vec denotes the vectorization operation, ie, stacking by columns. K is a kinship matrix, typically calculated by leave-one-chromosome-out methods. Y is the n by d phenotypes matrix. X is a block-diagonal nd by fd matrix consisting of d blocks each of dimension n by f. Each n by f block (on the diagonal) contains a matrix of founder allele probabilities for the n subjects at a single marker. The off-diagonal blocks have only zero entries. The log-likelihood is returned for each model. The outputted object is a tibble with d + 1 columns. The first d columns specify the markers used in the corresponding model fit, while the last column specifies the log-likelihood value at that d-tuple of markers.

Value

a tibble with d + 1 columns. First d columns indicate the genetic data (by listing the marker ids) used in the design matrix; last is log10 likelihood

References

Knott SA, Haley CS (2000) Multitrait least squares for quantitative trait loci detection. Genetics 156: 899–911.

Jiang C, Zeng ZB (1995) Multiple trait analysis of genetic mapping for quantitative trait loci. Genetics 140: 1111-1127.

Zhou X, Stephens M (2014) Efficient multivariate linear mixed model algorithms for genome-wide association studies. Nature methods 11:407-409.

Broman KW, Gatti DM, Simecek P, Furlotte NA, Prins P, Sen S, Yandell BS, Churchill GA (2019) R/qtl2: software for mapping quantitative trait loci with high-dimensional data and multi-parent populations. GENETICS https://www.genetics.org/content/211/2/495.

Examples

```
# read data
n <- 50
pheno <- matrix(rnorm(2 * n), ncol = 2)
rownames(pheno) <- paste0("s", 1:n)
colnames(pheno) <- paste0("tr", 1:2)
probs <- array(dim = c(n, 2, 5))
probs[ , 1, ] <- rbinom(n * 5, size = 1, prob = 0.2)
probs[ , 2, ] <- 1 - probs[ , 1, ]
rownames(probs) <- paste0("s", 1:n)
```

```
colnames(probs) <- LETTERS[1:2]
dimnames(probs)[[3]] <- paste0("m", 1:5)
scan_pvl(probs = probs, pheno = pheno, kinship = NULL,
start_snp = 1, n_snp = 5, cores = 1)
```

`sim1` *Simulate a single multivariate data set consisting of n subjects and d phenotypes for each*

Description

Simulate a single multivariate data set consisting of n subjects and d phenotypes for each

Usage

```
sim1(X, B, Sigma)
```

Arguments

`X`	design matrix (incorporating genotype probabilities from two loci), dn by df
`B`	a matrix of allele effects, f rows by d columns
`Sigma`	dn by dn covariance matrix

Value

a vector of length dn. The first n entries are for trait 1, the second n for trait 2, etc.

Examples

```
n_mouse <- 20
geno <- rbinom(n = n_mouse, size = 1, prob = 1 / 2)
X <- gemma2::stagger_mats(geno, geno)
B <- matrix(c(1, 2), ncol = 2, nrow = 1)
sim1(X, B, Sigma = diag(2 * n_mouse))
```

`subset_input` *Subset an input object - allele probabilities array or phenotypes matrix or covariates matrix. Kinship has its own subset function*

Description

An inputted matrix or 3-dimensional array is first subsetted - by rownames - to remove those subjects who are not in ‘id2keep‘. After that, the object’s rows are ordered to match the ordering of subject ids in the vector ‘id2keep‘. This (possibly reordered) object is returned.

Usage

```
subset_input(input, id2keep)
```

Arguments

input	a matrix of either phenotypes or covariates or array of allele probabilities
id2keep	a character vector of subject ids to identify those subjects that are shared by all inputs

Value

an object resulting from subsetting of ‘input‘. Its rows are ordered per ‘id2keep‘

Examples

```
# define s_id
s_id <- paste0("s", 1:10)
# set up input matrix
foo <- matrix(data = rnorm(10 * 3), nrow = 10, ncol = 3)
rownames(foo) <- s_id
subset_input(input = foo, id2keep = s_id)
```

---

subset_kinship *Subset a kinship matrix to include only those subjects present in all inputs*

Description

Since a kinship matrix has subject ids in both rownames and colnames, so we need to remove rows and columns according to names in ‘id2keep‘. We first remove rows and columns of subjects that are not in ‘id2keep‘. We then order rows and columns of the resulting matrix by the ordering in ‘id2keep‘.

Usage

```
subset_kinship(kinship, id2keep)
```

Arguments

kinship	a kinship matrix
id2keep	a character vector of subject ids to identify those subjects that are shared by all inputs

Index