Package 'qtlhot'

January 19, 2026

Version 1.2.10

Date 2026-01-19

Title Inference for QTL Hotspots

Description Functions to infer co-mapping trait hotspots and causal models.
Chaibub Neto E, Keller MP, Broman AF, Attie AD, Jansen RC, Broman KW, Yandell BS (2012) Quantile-based permutation thresholds for QTL hotspots. Genetics 191 : 1355-1365. <doi:10.1534/genetics.112.139451>.
Chaibub Neto E, Broman AT, Keller MP, Attie AD, Zhang B, Zhu J, Yandell BS (2013) Modeling causality for pairs of phenotypes in system genetics. Genetics 193 : 1003-1013. <doi:10.1534/genetics.112.147124>.

Maintainer Brian S. Yandell <brian.yandell@wisc.edu>

Depends stats,qtl,mnormt,utils,corpcor,broman, R (>= 3.5.0)

LazyLoad true

LazyData true

License GPL (>= 2)

URL https://github.com/byandell-sysgen/qtlhot

Encoding UTF-8

RoxygenNote 7.3.3

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Elias Chaibub Neto [aut],
Brian S Yandell [aut],
Brian S. Yandell [cre]

Repository CRAN

Date/Publication 2026-01-19 20:00:02 UTC

Contents

`add.phenos` *Add phenotypes to cross object.*

Description

Add phenotypes to cross object by checking index.

Usage

```
add.phenos(cross, newdata = NULL, index = NULL)
```

Arguments

`cross`	object of class `cross`; see `read.cross`
`newdata`	data frame with row names matching values of phenotype identified by `index` for object `cross`
`index`	character string name of phenotype in object `cross`; if `NULL`, then `newdata` must be of same size as `cross` with phenotypes in order

Details

The name `index` must be a phenotype in the `cross` object. The row names of `newdata` are matched with values of `index`.

Value

object of class cross with added phenotypes

Author(s)

Brian S. Yandell, <byandell@wisc.edu>

See Also

read.cross

Examples

```
## Not run:
data(hyper)
x <- data.frame(x = rnorm(nind(hyper)))
hyperx <- add.phenos(hyper, x)

## End(Not run)
```

CMSTCross *MST Sample Cross object*

Description

Created with 'SimCrossCausal()'.

Details

'CMSTCross' is created with 'SimCrossCausal()' and used in several examples.

Source

'SimCrossCausal'

Examples

```
data(CMSTCross)
```

CMSTtests *Perform CMST Tests on cross object*

Description

Performs 6 separate CMST tests (3 versions, 2 penalties).

Usage

```
CMSTtests(
  cross,
  pheno1,
  pheno2,
  Q.chr,
  Q.pos,
  addcov1 = NULL,
  addcov2 = NULL,
  intcov1 = NULL,
  intcov2 = NULL,
  method = c("par", "non.par", "joint", "all"),
  penalty = c("bic", "aic", "both"),
  verbose = FALSE,
  highobj = NULL,
  cand.reg = NULL,
  lod.thr = NULL
)

CMSTtestsList(
  cross,
  pheno1,
  pheno2,
  Q.chr,
  Q.pos,
  addcov1 = NULL,
  addcov2 = NULL,
  intcov1 = NULL,
  intcov2 = NULL,
  method = c("par", "non.par", "joint", "all"),
  penalty = c("bic", "aic", "both"),
  verbose = TRUE
)

GetCisCandReg(highobj, cand.reg, lod.thr = NULL)
```

Arguments

cross	object of class ‘cross‘

pheno1	first phenotype column number or character string name
pheno2	second phenotype column number or character string name; if more than one, then all phenotypes will be tested against 'pheno1'
Q.chr	QTL chromosome (number or label)
Q.pos	QTL position in cM
addcov1, addcov2	additive covariates for first and second phenotype, respectively
intcov1, intcov2	interactive covariates for first and second phenotype, respectively
method	test method; see details
penalty	type of penalty; see details
verbose	verbose printout if 'TRUE'
highobj	High LOD object used for CMST tests.
cand.reg	Candidate regions for QTL mapping.
lod.thr	LOD threshold for filtering significant regions.

Details

Explain method and penalty here.

References

Chaibub Neto E, Broman AT, Keller MP, Attie AD, Zhang B, Zhu J, Yandell BS, Causal model selection hypothesis tests in systems genetics. Genetics (in review).

See Also

'CMSTCross', 'PrecTpFpMatrix', 'FitAllTests'

Examples

```
## Not run:
# Create CMSTCross object
example(SimCrossCausal)
nms <- names(CMSTCross$pheno)
out1 <- CMSTtests(CMSTCross,
                  pheno1 = nms[1],
                  pheno2 = nms[2],
                  Q.chr = 1,
                  Q.pos = 55,
                  addcov1 = NULL,
                  addcov2 = NULL,
                  intcov1 = NULL,
                  intcov2 = NULL,
                  method = "all",
                  penalty = "both")
out1[1:6]
out1[7]
```

```
out1[8:12]
out1[13:17]
## list of phenotypes
out2 <- CMSTtests(CMSTCross,
                  pheno1 = nms[1],
                  pheno2 = nms[-1],
                  Q.chr = 1,
                  Q.pos = 55,
                  addcov1 = NULL,
                  addcov2 = NULL,
                  intcov1 = NULL,
                  intcov2 = NULL,
                  method = "par",
                  penalty = "bic")
out2

## End(Not run)
```

`filter.threshold` *Summary of threshold results*

Description

Summary of threshold results.

Usage

```
filter.threshold(
  cross,
  pheno.col,
  latent.eff,
  res.var,
  lod.thrs,
  drop.lod = 1.5,
  s.quant,
  n.perm,
  alpha.levels,
 qh.thrs = summary(hotperm(cross, max(s.quant), n.perm, alpha.levels, lod.thrs, verbose
    = verbose)),
  ww.thrs = summary(ww.perm(highobj, n.perm, lod.thrs, alpha.levels)),
  addcovar = NULL,
  intcovar = NULL,
  verbose = FALSE,
  ...
)
```

Arguments

cross	object of class cross; see read.cross
pheno.col	phenotype columns used for filtering thresholds
latent.eff	ratio of latent effect SD to residual SD
res.var	residual variance (=SD^2)
lod.thrs	LOD threshold values for range of significance (alpha) levels
drop.lod	LOD drop from max LOD to keep in analysis
s.quant	vector of 1:Nmax with Nmax the maximum hotspot size to be considered
n.perm	number of permutations
alpha.levels	range of significance levels; same length as lod.thrs
qh.thrs	Results of call to hotperm
ww.thrs	Results of call to ww.perm
addcovar	additive covariates as vector or matrix; see scanone
intcovar	interactive covariates as vector or matrix; see scanone
verbose	verbose output if TRUE
...	arguments passed along to scanone

Value

List with items

NL.thrs

N.thrs

WW.thrs

NL

N.counts

WW.counts

References

Manichaikul A, Dupuis J, Sen S, Broman KW (2006) Poor performance of bootstrap confidence intervals for the location of a quantitative trait locus. Genetics 174: 481-489.

See Also

hotperm, ww.perm, scanone

`FitAllTests` *Determine false positive and true positive rates for known targets.*

Description

Determine how well different tests do to predict candidates of regulation.

Usage

```
FitAllTests(cross, pheno1, pheno2, Q.chr, Q.pos, verbose = TRUE)

CitTests(LL, GG, TT)

CombineTests(comap, file)

JoinTestOutputs(comap, tests, file = NULL)
```

Arguments

`cross`	object of class 'cross'
`pheno1`	first phenotype column number or character string name
`pheno2`	second phenotype column number or character string name; if more than one, then all phenotypes will be tested against 'pheno1'
`Q.chr`	QTL chromosome (number or label)
`Q.pos`	QTL position in cM
`verbose`	verbose printout if 'TRUE
`LL`, `GG`, `TT`	Numeric vectors corresponding to genotype
`comap`	list result of 'GetComappingTraits()'
`file`	prefix for file names when running 'FitAllTests()' in parallel and saving test results in separate files `p.adjust`
`tests`	list object as list of 'FitAllTests()' results, or of joined output created by 'JoinTestsOutputs()'

Details

'FitAllTests()' invokes 7 tests. The hidden routine 'CitTests()' is invoked by call to 'FitAllTests()'; this is hidden because we do not recommend its use.

'JoinTestOutputs()' joins results of 'FitAllTests()', either from a list 'tests' or from a collection of files prefixed by 'file'. The joined tests from 'JoinTestOutputs' are summarized with 'PrecTpFpMatrix()' using the biologically validated true positives, false positives and precision, for the inferred causal relations. We define a true positive as a statistically significant causal relation between a gene and a putative target gene when the putative target gene belongs to the known signature of the gene. Similarly, we define a false positive as a statistically significant causal relation between a gene and a putative target gene when the target gene does not belong to the signature. (For the AIC and BIC

methods that do not provide a p-value measuring the significance of the causal call, we simply use the detected causal relations in the computation of true and false positives). The validated precision is computed as the ratio of true positives by the sum of true and false positives. The 'PrecTpFpMatrix' computes these measures to both all genes, and to cis genes only. Simulations suggest only non-parametric tests need to be adjusted using Benjamini-Hochberg via 'p.adjust.np()'.

Value

List containing

Prec1, Prec2	matrix of precision with rows for significance level and columns for test; first is for all, second is for cis candidates only
Tp1, Tp2	matrix of true positive rate with rows for significance level and columns for test; first is for all, second is for cis candidates only
Fp1, Fp2	matrix of false positive rate with rows for significance level and columns for test; first is for all, second is for cis candidates only

Author(s)

Elias Chaibub Neto

See Also

p.adjust

Examples

```
## Not run:
example(GetCandReg)
## Suppose y1 is causal with targets y2 and y3.
targets <- list(y1 = c("y2","y3"))

tests <- list()
for(k in seq(names(comap.targets))) {
  tests[[k]] <- FitAllTests(CMSTCross, pheno1 = names(comap.targets)[k],
                      pheno2 = comap.targets[[k]],
                      Q.chr = cand.reg[k, 4],
                      Q.pos = cand.reg[k, 5])
}
names(tests) <- names(comap.targets)
tests <- JoinTestOutputs(comap.targets, tests)

PrecTpFpMatrix(alpha = seq(0.01, 0.10, by = 0.01),
  val.targets = targets, all.orfs = CMSThigh$names, tests = tests,
  cand.reg = cand.reg, cis.cand.reg = cis.cand.reg)

## End(Not run)
```

GetCandReg *Get genetic information on candidate regulators and co-mapping traits.*

Description

Get chromosome (phys.chr) and physical position in cM (phys.pos), along with the LOD score (peak.lod) at the peak position (peak.pos), and the chromosome where the peak is located (peak.chr). Some candidates may map to the same chromosome where they are physically located.

Usage

```
GetCandReg(highobj, annot, traits)

GetCoMappingTraits(highobj, cand.reg)
```

Arguments

highobj	data frame from 'highlod', which is sparse summary of high LODs in large scanone object
annot	data frame with annotation information; must have first column as unique identifier, third column as chromosome, and fifth column as position in cM; typically column 2 has gene name, and column 4 has position in Mb
traits	names of traits to examine as candidate regulators; names must correspond to phenotypes in 'cross' object
cand.reg	data frame with candidate regulator; see value section below

Details

Traits that map to positions close to their physical locations are said to map in cis (local linkages). Traits that map to positions away from their physical locations are said to map in trans (distal linkages). There is no unambiguous way to determine how close a trait needs to map to its physical location in order to be classified as cis. Our choice is to classify a trait as cis if the 1.5-LOD support interval (Manichaikul et al. 2006) around the LOD peak contains the trait's physical location, and if the LOD score at its physical location is higher the the LOD threshold. The function 'GetCisCandReg' determines which of the candidate regulators map in cis. The function 'GetCoMappingTraits' returns a list with the putative targets of each trait. A trait is included in the putative target list of a trait when its LOD peak is greater than 'lod.thr' and the 'drop' LOD support interval around the peak contains the location of the trait's QTL. The function 'JoinTestOutputs' currently relies on external files that contain results of 'FitAllTests'. It needs to be rewritten to save space.

Value

'GetCoMappingTraits' returns a list with each element being the names of co-mapping traits for a particular name in 'traits'. 'GetCandReg' returns a data frame while 'GetCisCandReg' returns a list with a similar candidate regulator data frame as the element 'cis.reg', and the index of trait

names as the element 'cis.index'. The elements of the candidate regulator data frame are as follows ('peak.pos.lower' and 'peak.pos.upper' only for 'GetCisCandReg'):

`gene`	name of trait, which might be a gene name
`phys.chr`	chromosome on which gene physically resides
`phys.pos`	physical position (in cM)
`peak.chr`	chromosome where peak LOD is located
`peak.pos`	position of peak (in cM)
`peak.lod`	LOD value at peak
`peak.pos.lower`, `peak.pos.upper`	lower and upper bounds of the 1.5-LOD support interval around 'peak.pos'

Author(s)

Elias Chaibub Neto

References

Manichaikul et al. (2006) Genetics

See Also

'highlod', 'FitAllTests', scanone

Examples

```
## Not run:
# Create CMSTCross object
example(SimCrossCausal)
# data(CMSTCross) loaded lazily
CMSTscan <- scanone(CMSTCross, pheno.col = 1:3, method = "hk")
CMSThigh <- highlod(CMSTscan)
traits <- names(CMSTCross$pheno)
annot <- data.frame(name = traits, traits = traits, chr = rep(1, 3),
 Mb.pos = c(55,10,100))
annot$cM.pos <- annot$Mb.pos
cand.reg <- GetCandReg(CMSThigh, annot, traits)
cis.cand.reg <- GetCisCandReg(CMSThigh, cand.reg)
comap.targets <- GetCoMappingTraits(CMSThigh, cand.reg)

## End(Not run)
```

GetCommonQtls *Get common QTLs for phenotypes*

Description

Perform joint QTL mapping for phenotypes with marginal LOD peak positions higher than LOD threshold and within set distance of each other

Usage

```
GetCommonQtls(
  cross,
  pheno1,
  pheno2,
  thr = 3,
  peak.dist = 5,
  addcov1 = NULL,
  addcov2 = NULL,
  intcov1 = NULL,
  intcov2 = NULL
)
```

Arguments

cross	object of class 'cross'
pheno1	first phenotype column number or character string name
pheno2	second phenotype column number or character string name; if more than one, then all phenotypes will be tested against 'pheno1'
thr	LOD threshold
peak.dist	maximal peak distance to be considered the same peak (in cM)
addcov1, addcov2	additive covariates for first and second phenotype, respectively
intcov1, intcov2	interactive covariates for first and second phenotype, respectively

References

Chaibub Neto E, Broman AT, Keller MP, Attie AD, Zhang B, Zhu J, Yandell BS, Causal model selection hypothesis tests in systems genetics. Genetics (in review).

Examples

```
## Not run:
# Create CMSTCross object
example(SimCrossCausal)
# data(CMSTCross) loaded lazily
```

```
commqtls <- GetCommonQtls(CMSTCross,
                          pheno1 = "y1",
                          pheno2 = "y3",
                          thr = 3,
                          peak.dist = 5,
                          addcov1 = NULL,
                          addcov2 = NULL,
                          intcov1 = NULL,
                          intcov2 = NULL)
commqtls

## End(Not run)
```

`highlod` *Pull high LOD values with chr and pos.*

Description

Pull high LOD values with chr and pos.

Usage

```
highlod(
  scans,
  lod.thr = 0,
  drop.lod = 1.5,
  extend = TRUE,
  restrict.lod = FALSE,
  ...
)

## S3 method for class 'highlod'
print(x, ...)

## S3 method for class 'highlod'
summary(object, ...)

## S3 method for class 'highlod'
plot(x, ..., quant.level = NULL, sliding = FALSE)

pull.highlod(object, chr, pos, ...)

## S3 method for class 'highlod'
max(x, lod.thr = NULL, window = NULL, quant.level = NULL, ...)
```

Arguments

scans	object of class scanone

lod.thr	LOD threshold
drop.lod	LOD drop from max to keep for support intervals
extend	extend support interval just past drop.lod; matches behavior of lodint when TRUE
restrict.lod	restrict to loci above LOD threshold if TRUE; matches behavior of lodint when FALSE (default)
...	arguments passed along
x, object	object of class highlod
quant.level	vector of LOD levels for 1 up to length(quant.level) size hotspots
sliding	plot as sliding hotspot if TRUE
chr	chromosome identifier
pos	position, or range of positions, in cM
window	size of window for smoothing hotspot size

Details

The highlod condenses a scanone object to the peaks above a lod.thr and/or within drop.lod of such peaks. The pull.highlod pulls out the entries at a particular genomic location or interval of locations. Summary, print, plot, max and quantile methods provide ways to examine a highlod object.

Value

Data frame with

row	row number in scanone object
phenos	phenotype column number
lod	LOD score for phenotype at locus indicated by row

Author(s)

Brian S Yandell and Elias Chaibub Neto

See Also

highlod, hotperm

Examples

```
ncross1 <- sim.null.cross(chr.len = rep(100, 4),
                          n.mar = 51,
                          n.ind = 100,
                          type = "bc",
                          n.phe = 1000,
                          latent.eff = 3,
                          res.var = 1,
                          init.seed = 123457)
```

```
cross1 <- include.hotspots(cross = ncross1,
                           hchr = c(2, 3, 4),
                           hpos = c(25, 75, 50),
                           hsize = c(100, 50, 20),
                           Q.eff = 2,
                           latent.eff = 3,
                           lod.range.1 = c(2.5, 2.5),
                           lod.range.2 = c(5, 8),
                           lod.range.3 = c(10, 15),
                           res.var = 1,
                           n.phe = 1000,
                           init.seed = 12345)
scan1 <- scanone(cross1, pheno.col = 1:1000, method = "hk")
high1 <- highlod(scan1, lod.thr = 2.11, drop.lod = 1.5)
pull.highlod(high1, chr = 2, pos = 24)
summary(high1, lod.thr = 2.44)
max(high1, lod.thr = seq(2.11, 3.11, by = .1))
```

---

hotperm *Conduct NL and N permutation tests*

Description

This set of functions compute the permutation LOD thresholds for the NL-method and the permutation hotspot size thresholds for the N-method. The output is a list with two elements: the NL- and the N-method's threshold matrices. The NL-method output is a nN (number of spurious hotspot sizes) by nalpha (number of significance levels) threshold matrix. Note that for the NL-method we have a single "alpha" since we use the same significance level for QTL mapping and permutation significance. The N-method output is a nlod (number of LOD thresholds) by nalpha (number of significance levels) threshold matrix. Note that here we have two "alphas", one for the QTL mapping (the LOD thresholds) and one for the permutation significance (alpha levels).

Usage

```
hotperm(
  cross,
  n.quant,
  n.perm,
  lod.thrs,
  alpha.levels,
  drop.lod = 1.5,
  window = NULL,
  verbose = FALSE,
  init.seed = 0,
  addcovar = NULL,
  intcovar = NULL,
  ...
```

```
)

## S3 method for class 'hotperm'
print(x, ...)

## S3 method for class 'hotperm'
summary(object, quant.levels, ...)

## S3 method for class 'summary.hotperm'
print(x, ...)

## S3 method for class 'hotperm'
plot(x, probs = seq(0.9, 0.99, by = 0.01), level = 0.95, ...)
```

Arguments

cross	object of class cross
n.quant	maximum of s.quant
n.perm	number of permutations
lod.thrs	vector of LOD thresholds
alpha.levels	vector of significance levels
drop.lod	LOD drop amount for support intervals
window	window size for smoothed hotspot size
verbose	verbose output if TRUE
init.seed	initial seed for pseudo-random number generation
addcovar	additive covariates as vector or matrix; see scanone
intcovar	interactive covariates as vector or matrix; see scanone
...	arguments passed along to scanone
x, object	object of class hotperm or summary.hotperm
quant.levels	quantile levels, as number of traits, to show in summary; default is 1, 2, 5, 10, ... up to maximum recorded
probs	probability levels for quantiles (1-probs if all > 0.5); default is alpha.levels
level	Significance level for hotspot detection.

Author(s)

Elias Chaibub Neto and Brian S Yandell

Examples

```
ncross1 <- sim.null.cross(chr.len = rep(100, 4),
                          n.mar = 51,
                          n.ind = 100,
                          type = "bc",
                          n.phe = 1000,
```

```
                            latent.eff = 3,
                            res.var = 1,
                            init.seed = 123457)
cross1 <- include.hotspots(cross = ncross1,
                            hchr = c(2, 3, 4),
                            hpos = c(25, 75, 50),
                            hsize = c(100, 50, 20),
                            Q.eff = 2,
                            latent.eff = 3,
                            lod.range.1 = c(2.5, 2.5),
                            lod.range.2 = c(5, 8),
                            lod.range.3 = c(10, 15),
                            res.var = 1,
                            n.phe = 1000,
                            init.seed = 12345)
pt.scanone <- scanone(ncross1, method = "hk", n.perm = 1000)
alphas <- seq(0.01, 0.10, by=0.01)
lod.thrs <- summary(pt.scanone, alphas)
# This takes awhile, so we save the object.
## Not run:
hotperm1 <- hotperm(cross = cross1,
                    n.quant = 300,
                    n.perm = 100,
                    lod.thrs = lod.thrs,
                    alpha.levels = alphas,
                    drop.lod = 1.5,
                    verbose = FALSE)
save(hotperm1, file = "hotperm1.RData", compress = TRUE)
# data(hotperm1)
summary(hotperm1)

## End(Not run)
```

`hotperm1` *Hotspot Permutation Sample*

Description

Created with ‘hotperm()‘.

Details

‘hotperm1‘ is created with ‘hotperm()‘ and used in several examples.

Source

‘SimCrossCausal‘

Examples

```
data(hotperm1)
```

hotsize *Hotspot size routines.*

Description

Determine hotspot sizes and display. Use individual threshold and quantile thresholds as provided.

Usage

```
hotsize(hotobject, ...)

## S3 method for class 'scanone'
hotsize(hotobject, lod.thr = NULL, drop.lod = 1.5, ...)

## S3 method for class 'highlod'
hotsize(hotobject, lod.thr = NULL, window = NULL, quant.level = NULL, ...)

## S3 method for class 'hotsize'
print(x, ...)

## S3 method for class 'hotsize'
summary(object, ...)

## S3 method for class 'hotsize'
plot(
  x,
  ylab = "counts",
  quant.axis = pretty(x$max.N),
  col = c("black", "red", "blue"),
  by.chr = FALSE,
  maps = NULL,
  title = "",
  ...
)

## S3 method for class 'hotsize'
max(x, ...)
```

Arguments

hotobject	object of class scanone or highlod
...	arguments passed along to scanone methods
lod.thr	LOD threshold
drop.lod	LOD drop from max to keep for support intervals
window	window width in cM for smoothing hotspot size; not used if 0 or NULL

quant.level	vector of LOD levels for 1 up to length(quant.level) size hotspots
x, object	object of class hotsize
ylab	label for vertical plot axis
quant.axis	hotspot sizes for quantile axis (vertical on right side of plot)
col	col of hotspot size, smoothed hotspot size, and sliding hotspot size
by.chr	separate plot by chromosome if TRUE
maps	if not NULL, list of objects of class map to use for rugs on top and bottom of plot
title	title for plot

Value

hotsize methods return an object of class hotsize, which is essentially an object of class summary.scanone with additional attributes for lod.thr, window, and quant.level.

Author(s)

Brian S Yandell and Elias Chaibub Neto

See Also

highlod, hotperm

Examples

```
example(highlod)
hots1 <- hotsize(high1)
summary(hots1)
plot(hots1)
```

---

include.hotspots *Generate hotspots for the simulated examples in the manuscript.*

Description

Generate hotspots for the simulated examples in the manuscript.

Usage

```
include.hotspots(
  cross,
  hchr,
  hpos,
  hsize,
  Q.eff,
  latent.eff,
```

```
  lod.range.1,
  lod.range.2,
  lod.range.3,
  res.var = 1,
  n.pheno,
  init.seed
)
```

Arguments

cross	cross object
hchr	character string for chromosome
hpos	position on chromosome
hsize	size
Q.eff	Q effect
latent.eff	latent effect
lod.range.1	lod range 1
lod.range.2	lod range 2
lod.range.3	lod range 3
res.var	residual variance
n.pheno	number of phenotypes
init.seed	initial seed

---

neqtl *neqtl.R Ported from http://github.com/kbroman/neqtl*

Description

neqtl.R Ported from http://github.com/kbroman/neqtl

Usage

```
neqtl(sigpos.out, chr, pos, win = 5)
```

Arguments

sigpos.out	position(s) of max
chr	chromosome as character
pos	position on chromosome
win	window width in cM

parallel.qtlhot *Code for parallelizing R/qtlhot.*

Description

Code for parallelizing R/qtlhot. See installed parallel directory for proper use. There is apparently an S3 parallel method, so doc has to be as shown below, even though it is called as parallel.qtlhot.

Usage

```
parallel.qtlhot(x, data = 1, ..., dirpath = ".")
```

Arguments

x	phase of parallel processing (1,2,3)
data	index for parallel processing (1,2,...)
...	additional arguments passed along
dirpath	directory path as character string

Author(s)

Brian S Yandell and Elias Chaibub Neto

See Also

read.cross

PrecTpFpMatrix *Compute Precision, True Positives, and False Positives*

Description

The ‘PrecTpFpMatrix‘ function calculates precision, true positives, and false positives for various tests. It is used to evaluate the performance of QTL mapping methods.

Usage

```
PrecTpFpMatrix(alpha, val.targets, all.orfs, tests, cand.reg, cis.cand.reg)
```

Arguments

alpha	A numeric vector of significance levels.
val.targets	A vector of validated target genes.
all.orfs	A vector of all open reading frames (ORFs).
tests	A list of test results for different methods.
cand.reg	Candidate regions for QTL mapping.
cis.cand.reg	Cis-regulatory candidate regions.

Value

A list of matrices containing precision, true positives, and false positives for each method and significance level.

Examples

```
## Not run:
results <- PrecTpFpMatrix(alpha = seq(0.01, 0.1, by = 0.01),
                          val.targets = validated_genes,
                          all.orfs = all_genes,
                          tests = test_results,
                          cand.reg = candidate_regions,
                          cis.cand.reg = cis_regions)

## End(Not run)
```

---

quantile_highlod *Compute Quantiles of High LOD Scores*

Description

The ‘quantile_highlod‘ function calculates quantiles of high LOD scores from a ‘highlod‘ object. It is used to summarize the distribution of LOD scores across the genome.

Usage

```
quantile_highlod(
  x,
  probs = NULL,
  lod.thr = NULL,
  n.quant,
  n.pheno,
  max.quantile = TRUE,
  ...
)
```

Arguments

x	A ‘highlod‘ object containing LOD scores.
probs	A numeric vector of probabilities for quantiles. If ‘NULL‘, quantiles are computed for all available data.
lod.thr	LOD threshold for filtering scores. If ‘NULL‘, no filtering is applied.
n.quant	Maximum number of quantiles to compute.
n.pheno	Number of phenotypes considered.
max.quantile	Logical; if ‘TRUE‘, returns only the maximum quantile values.
...	Additional arguments passed to internal functions.

Value

A numeric vector or matrix of quantiles, depending on the input parameters.

Examples

```
## Not run:
highlod_obj <- highlod(scan1, lod.thr = 2.5)
quantiles <- quantile_highlod(highlod_obj, probs = seq(0.1, 0.9, by = 0.1))

## End(Not run)
```

`quantile_hotperm` *Compute Quantiles for Hotperm Results*

Description

The ‘quantile_hotperm‘ function calculates quantiles for permutation test results. It is used to summarize the distribution of hotspot sizes and LOD scores.

Usage

```
quantile_hotperm(x, probs = attr(x, "alpha.levels"), ..., lod.thr = NULL)
```

Arguments

`x`	A ‘hotperm‘ object containing permutation test results.
`probs`	A numeric vector of probabilities for quantiles. Defaults to the alpha levels in the object.
`...`	Additional arguments passed to internal functions.
`lod.thr`	LOD threshold for filtering scores. If ‘NULL‘, no filtering is applied.

Value

A numeric vector or matrix of quantiles, depending on the input parameters.

Examples

```
## Not run:
hotperm_obj <- hotperm(cross, n.quant = 300, n.perm = 100, lod.thrs = c(2.5, 3.0))
quantiles <- quantile_hotperm(hotperm_obj, probs = seq(0.1, 0.9, by = 0.1))

## End(Not run)
```

`sim.hotspot` *Wrapper routine for simulations.*

Description

Simulate 'nSim' realizations of cross object with 'n.pheno' phenotypes with correlation 'latent.eff'. All simulations use the same genotypes in the 'cross' object.

Usage

```
sim.hotspot(
  nSim,
  cross,
  n.pheno,
  latent.eff,
  res.var = 1,
  n.quant,
  n.perm,
  alpha.levels,
  lod.thrs,
  drop.lod = 1.5,
  verbose = FALSE
)
```

Arguments

`nSim`	Number of simulated sets of phenotypes to create. See details.
`cross`	Object of class 'cross'. See `read.cross`.
`n.pheno`	Number of traits, or phenotypes, to simulate for cross object.
`latent.eff`	Strength of latent effect, which is included in all traits. See `sim.null.cross`.
`res.var`	Residual variance for traits. Should not affect results.
`n.quant`	maximum size of hotspots examined; ideally large enough to exceed the largest Breitling alpha critical value.
`n.perm`	Number of permutations to perform per realization. Good idea to do 1000, but this takes time.
`alpha.levels`	Vector of significance levels.
`lod.thrs`	Vector of LOD thresholds, typically single-trait permutation thresholds for various significance levels.
`drop.lod`	Drop in LOD score examined. LODs below this drop from the maximum for a chromosome will not be scored.
`verbose`	verbose output if `TRUE`

Value

'sim.null.cross' simulates an object of class 'cross'. 'sim.null.pheno.data' simulates a data frame of phenotypes. 'sim.hotspot' uses these other routines to simulate a hotspot, returning an list object.

Author(s)

Elias Chaibub Neto and Brian S. Yandell

See Also

sim.null.cross, read.cross.

Examples

```
ncross1 <- sim.null.cross(chr.len = rep(100, 4),
                          n.mar = 51,
                          n.ind = 100,
                          type = "bc",
                          n.phe = 1000,
                          latent.eff = 3,
                          res.var = 1,
                          init.seed = 123457)
cross1 <- include.hotspots(cross = ncross1,
                           hchr = c(2, 3, 4),
                           hpos = c(25, 75, 50),
                           hsize = c(100, 50, 20),
                           Q.eff = 2,
                           latent.eff = 3,
                           lod.range.1 = c(2.5, 2.5),
                           lod.range.2 = c(5, 8),
                           lod.range.3 = c(10, 15),
                           res.var = 1,
                           n.phe = 1000,
                           init.seed = 12345)
```

---

sim.null.cross *Generates a "null dataset" cross*

Description

Generates a "null dataset" cross

Usage

```
sim.null.cross(
  chr.len = rep(400, 16),
  n.mar = 185,
  n.ind = 112,
```

```
  type = "bc",
  n.pheno = 6000,
  latent.eff = 1.5,
  res.var = 1,
  init.seed = 92387475
)
```

Arguments

chr.len	vector with length of chromosomes
n.mar	number of markers
n.ind	number of individuals
type	cross type as character
n.pheno	number of phenotypes
latent.eff	latent effect
res.var	residual variance
init.seed	initial seed

SimCrossCausal *Simulate Cross for Causal Tests*

Description

Creates cross with certain pattern of dependence across phenotypes.

Usage

```
SimCrossCausal(
  n.ind,
  len,
  n.mar,
  beta,
  add.eff,
  dom.eff,
  sig2.1 = 1,
  sig2.2 = 1,
  eq.spacing = FALSE,
  cross.type = c("bc", "f2"),
  normalize = FALSE
)
```

Arguments

n.ind	number of individuals to simulate
len	vector specifying the chromosome lengths (in cM)
n.mar	vector specifying the number of markers per chromosome
beta	causal effect (slope) of first phenotype on others
add.eff	additive genetic effect
dom.eff	dominance genetic effect
sig2.1	residual variance for first phenotype
sig2.2	residual variance for all other phenotypes
eq.spacing	if TRUE, markers will be equally spaced
cross.type	type of cross (bc and f2 for now)
normalize	normalize values if TRUE

References

Chaibub Neto E, Broman AT, Keller MP, Attie AD, Zhang B, Zhu J, Yandell BS, Causal model selection hypothesis tests in systems genetics. Genetics (in review).

Examples

```
## Not run:
set.seed(987654321)
CMSTCross <- SimCrossCausal(n.ind = 100,
  len = rep(100, 3), n.mar = 101,
  beta = rep(0.5, 2), add.eff = 1, dom.eff = 0,
  sig2.1 = 0.4, sig2.2 = 0.1, eq.spacing = FALSE,
  cross.type = "bc", normalize = TRUE)
CMSTCross <- calc.genoprob(CMSTCross, step = 1)
save(CMSTCross, file = "CMSTCross.RData", compress = TRUE)
class(CMSTCross)

## End(Not run)
```

---

ww.perm *Conduct West-Wu (Q) permutation tests*

Description

This function computes the West/Wu permutation thresholds. The output is a nlod (number of LOD thresholds) by nalpha (number of significance levels) matrix, where each entry shows the hotspot size significance threshold of the West/Wu approach. Note we have two "alphas" here, one for the QTL mapping (the LOD thresholds) and one for the permutation significance (alpha levels of lod.thrs).

Usage

```
ww.perm(highobj, n.perm, lod.thrs, alpha.levels, verbose = FALSE)

## S3 method for class 'ww.perm'
print(x, ...)

## S3 method for class 'ww.perm'
summary(object, alpha.levels = attr(object, "alpha.levels"), ...)
```

Arguments

highobj	object of class highlod
n.perm	number of permutations
lod.thrs	vector of LOD thresholds
alpha.levels	vector of significance levels
verbose	verbose output if TRUE
x, object	object of class ww.perm
...	ignored

Details

Note that I separated the original ww.permutations() into a piece that do the actual permutations [ww.perm.matrix() function] and a piece that summarizes it [the ww.summary() function] in the same way you did with the NL.N.permutations() function.

Perform permutation tests to assess the statistical significance of the hotspots detected using the West-Wu Q-method permutations. The ww.perm function implements the Q-method's permutation scheme (see the Method's section of Chaibub Neto et a. 2012, for details). The n.perm parameter specifies the number of simulations. Here we set it to 100 in order to save time. In practice, we recommend at least 1,000 permutations. The function's output is a matrix with 100 rows representing the permutations, and 10 columns representing the QTL mapping thresholds. Each entry ij, represents the maximum number of significant linkages across the entire genome detected at permutation i, using the LOD threshold j. The ww.summary function computes the Q-method's hotspot size permutation thresholds, that is, the 1-alpha quantiles for each one of the QTL mapping LOD thrsholds in lod.thrs. For instance, the entry at row 10 and column 1 of the Q.1.thr matrix tells us that the 99% percentile of the permutation distribution of genome wide maximum hotspot size based on a QTL mapping threshold of 2.11 is 27.00. In other words, any hotspot greater than 27 is considered statistically significant at a 0.01 significance level when QTL mapping is done using a 2.11 LOD threshold. In general, we are often interested in using the same error rates for the QTL mapping and hotspot analysis. That is, if we adopt a QTL mapping threshold that controls GWER at a 1% level (in our case, 3.11) we will also want to consider alpha = 0.01 for the hotspot analysis, leading to a hotspot threshold of 12.00. Therefore, we are usually more interested in the diagonal of Q.1.thr. We adopted a GWER of 5%, and the corresponding Q-method's permutation threshold is 18. According to this threshold, all hotspots are significant.

Author(s)

Elias Chaibub Neto and Brian S Yandell

Examples

```
## Not run:
## All unspecified objects come from vignette qtlhot.
set.seed(12345)
Q.1 <- ww.perm(high1, n.perm = 100, lod.thrs, alphas)
Q.1.thr <- summary(Q.1, alphas)
Q.1.thr
diag(Q.1.thr)

set.seed(12345)
Q.2 <- ww.perm(high2, 100, lod.thrs, alphas)
Q.2.thr <- summary(Q.2, alphas)

## End(Not run)
```

Index