
`qb.BayesFactor` *Bayesian model selection via Bayes factors.*

Description

Model-averaged posteriors and Bayes factors computed for number and pattern of QTL, chromosomes and pairs of chromosomes showing epistasis.

Usage

```
qb.bf(...)
qb.BayesFactor(qbObject, items = c("nqtl","pattern","chrom","pairs"),
  cutoff.pattern, cutoff.pairs = 1, nmax = 15, epistasis = TRUE)
## S3 method for class 'qb.BayesFactor':
plot(x, ...)
## S3 method for class 'qb.BayesFactor':
summary(object, sort = TRUE, digits = 3, ...)
## S3 method for class 'qb.BayesFactor':
print(x, ...)
```

Arguments

<code>qbObject</code>	An object of class <code>qb</code> .
<code>object</code>	Object of class <code>qb.BayesFactor</code> .
<code>x</code>	Object of class <code>qb.BayesFactor</code> .
<code>items</code>	Items to include in model selection assessment.
<code>cutoff.pattern</code>	Percent cutoff for pattern inclusion in model selection. Default is 0.25 (0.5) if <code>epistasis</code> is <code>TRUE</code> (<code>FALSE</code>).
<code>cutoff.pairs</code>	Percent cutoff for epistatic pair inclusion in model selection.
<code>nmax</code>	Maximum number of model terms included per item (for <code>items</code> "pattern" and "pairs" only).
<code>epistasis</code>	Include epistasis in patterns if <code>TRUE</code> .
<code>sort</code>	Sort by Bayes factor if <code>TRUE</code> .
<code>digits</code>	Number of significant digits for summary.
<code>...</code>	Additional arguments passed to generic <code>plot</code> , <code>summary</code> or <code>print</code> .

Details

`qb.BayesFactor` (or `qb.bf` for short) creates model selection results for selected items. These are based on marginal posteriors and priors, averaged over all other model parameters. The posterior may be influenced by prior, while Bayes factors are empirically less sensitive for QTL model selection. The Bayes factors are computed relative to the smallest term for each item, using the ratios of `posterior/prior`. Any pair of model terms can be compared as the ratio of their Bayes factors. The `items` evaluated are:

nqtl Number of QTLs.

pattern Pattern of QTL across chromosomes. Identifiers are comma-separated chromosome numbers, with asterisk `n*` for multiple QTL per chromosome.

chrom Chromosome.

pairs Epistatic pairs of chromosomes.

Value

List with `items`, each containing:

<code>posterior</code>	Posterior frequency of MCMC samples.
<code>prior</code>	Prior frequency.
<code>bf</code>	Rank-ordered Bayes factors relative to smallest value.
<code>bfse</code>	Approximate standard error for <code>bf</code> computed using binomial variance of MCMC samples.

Author(s)

Brian S. Yandell, yandell@stat.wisc.edu

References

<http://www.qtlbim.org>

See Also

[plot.qb](#), [qb.mcmc](#)

Examples

```
data(qbExample)

temp <- qb.BayesFactor(qbExample)
summary(temp)
plot(temp)
```

`qb.BestPattern` *Proximity of common genetic architecture patterns.*

Description

Multidimensional scaling and hierarchical clustering of most common patterns of genetic architecture.

Usage

```
qb.best(...)
qb.BestPattern(qbObject, epistasis = TRUE,
  category = c("pattern", "nqt1"), cutoff, score.type =
  c("sq.atten", "attenuation", "variance", "recombination", "distance"),
  include = c("nested", "all", "exact"),
  center = c("median", "mean"), level = 5, ...)
## S3 method for class 'qb.BestPattern':
plot(x, type = c("mds", "hclust"),
  main, xlab, method = "complete", cluster = 3, cexmax = 5,
  colmax = 75, cex, col,
  symbol = c("pattern", "nqt1", "cluster", "c@n", "c@p", "n@p", "c@n@p"), ...)
## S3 method for class 'qb.BestPattern':
summary(object, method = "complete",
  cluster = 3, n.best = 1, ...)
```

Arguments

<code>qbObject</code>	Object of class <code>qb</code> .
<code>x, object</code>	Object of class <code>qb.BestPattern</code> .
<code>epistasis</code>	Include epistasis in patterns if <code>TRUE</code> .
<code>category</code>	Distances indexed by <code>nqt1</code> or <code>pattern</code> .
<code>cutoff</code>	Percent cutoff for pattern inclusion in model selection. Default is 0.25 (0.5) if <code>epistasis</code> is <code>TRUE</code> (<code>FALSE</code>).
<code>score.type</code>	Type of score to use as distance. See qb.close .
<code>type</code>	Plot dendrogram for <code>hclust</code> or 2-D multidimensional scaling projection for <code>mds</code> .
<code>main</code>	Main plot title as character string.
<code>xlab</code>	Character string for horizontal (x) axis.
<code>method</code>	Method for hierarchical clustering.
<code>cluster</code>	Number of clusters desired.
<code>n.best</code>	Number of better models to display.
<code>cexmax</code>	Maximum font size (minimum is set to 1); patterns are displayed in <code>mds</code> plot proportional to their posterior probability.
<code>colmax</code>	Maximum number of colors.
<code>cex</code>	Manual override of font size for <code>mds</code> plot; should be length 1 or the number of patterns exceeding <code>cutoff</code> .
<code>col</code>	Colors for plotting.
<code>symbol</code>	Plot symbol for <code>mds</code> plot. Shorthand using at sign @ signifies catenation of two or more symbols into one.
<code>include</code>	Action for model averaging of chromosome-specific locus and explained variance: use <code>all</code> MCMC samples that match the chromosome; use only MCMC samples for patterns that have the target pattern <code>nested</code> within them; or use only MCMC samples with the <code>exact</code> same target pattern.

<code>center</code>	Method of estimating the center for locus and explained variance.
<code>level</code>	Confidence level as percent between 0 and 100 for loci and variance contributions.
<code>...</code>	Parameters to methods.

Details

This uses the closeness measure from `qb.close` to compute a similarity matrix among patterns whose posterior probabilities exceed `cutoff`. Distance = 1 - similarity is used for hierarchical clustering or multidimensional scaling.

The `best` pattern is chosen as the one with highest posterior mean; all other patterns are compared to that pattern in terms of the `score.type`. This best pattern is a natural `target` for `qb.close`.

Author(s)

Brian S. Yandell, yandell@stat.wisc.edu

References

<http://www.qtlbim.org>

See Also

`qb.close`

Examples

```
data(qbExample)

temp <- qb.BestPattern(qbExample)
summary(temp, n.best = 3)
plot(temp, type = "hclust")
plot(temp)
plot(temp, symbol = "c@n")

best <- summary(temp)$best
temp <- qb.close(qbExample, best)
summary(temp)
plot(temp)
```

Bnapus

Cross structure for complete Brassica napus data

Description

Contains genotypes and phenotypes for Brassica napus study, including 0- 4- and 8-week vernalization, survival, and 19 chromosomes.

Usage

`data(Bnapus)`

Format

See `read.cross` in `library(qt1)` for format.

Details

Traits included are percent winter survival for 1992-3, 1993-4, 1994-5, 1997-8, and 1999-2000 (`surv92`, `surv93`, `surv94`, `surv97`, `surv99`, respectively), and days to flowering after no vernalization (`flower0`), 4 weeks vernalization (`flower4`) or 8 weeks vernalization (`flower8`). Percents are of plants alive in the Fall, taken from the middle of rows (totals unavailable). Days to flowering after transplant are averages over four replicates from a RCB design (values by block unavailable). First column has the trait name. The remaining columns identify individual DH line (302-455).

Marker genotype data for Major x Stellar double haploid (DH) population. Double haploids have the same relation of recombination to distance as backcrosses as there is just one meiosis tracked (in F1). However, DH are homozygous at every locus (usually mapped as RI0 lines). Marker genotypes are coded as M = Major, S = Stellar, - = missing. Data columns are

`chrom` = *B. napus* chromosome (N1, N2, etc.)

`order` = along chromosome

`cM d` = istance from proximal end

`marker` = marker name: E = AFLP; *ec, *tg, *wg = RFLP; *xxx = other markers from Arabidopsis: Lem, eru1, eru2, fad3, isoDia, isoIdh, isoPgi, isoLap, pr2, slg6, Aca1, cor15

Remaining columns are for individual DH lines (identifier 302-455).

Source

Thomas C. Osborn (<mailto:tcosborn@facstaff.wisc.edu>), Department of Agronomy, UW-Madison (<http://agronomy.wisc.edu>).

References

<http://www.stat.wisc.edu/~yandell/qt1/data/osborn/Bnapus>

Ferreira ME, Satagopan J, Yandell BS, Williams PH, Osborn TC (1995) Mapping loci controlling vernalization requirement and flowering time in *Brassica napus*. *Theor Appl Genet* 90: 727-732. [original source and analysis]

JM Satagopan, BS Yandell, MA Newton and TC Osborn (1996) Markov chain Monte Carlo approach to detect polygene loci for complex traits. *Genetics* 144: 805-816. <http://www.genetics.org/cgi/content/abstract/144/2/805> [first MCMC for experimental crosses; analysis of *B. napus* N2=LG9; see [vern](#) data]

Kole C, Thorman CE, Karlsson BH, Palta JP, Gaffney P, Yandell BS, Osborn TC (2001) Comparative mapping of loci controlling winter survival and related traits in oilseed Brassica rapa and B. napus. Molecular Breeding 1: 329-339. [refined map and reanalysis]

See Also

[read.cross,plot.qb](#)

Examples

```
data(Bnapus)
summary(Bnapus)
```

<code>qb.arch</code>	<i>Build genetic architecture with chromosomes, positions and epistatic pairs.</i>
----------------------	--

Description

These routines work in conjunction with [qb.hpdone](#), [qb.scantwo](#), [qb.sliceone](#) and [step.fitqtl](#) to infer the number, pattern and position of QTL from MCMC samples.

Usage

```
qb.arch(object, ...)
## Default S3 method:
qb.arch(object, chr, pos, tolerance = 10, ...)
## S3 method for class 'step.fitqtl':
qb.arch(object, main, epistasis, ...)
## S3 method for class 'qb.arch':
summary(object, ...)
## S3 method for class 'qb.arch':
print(x, ...)
```

Arguments

<code>object</code>	Object for appropriate method: summary of object of class qb.scantwo for default; object of class step.fitqtl .
<code>x</code>	Object of class qb.arch .
<code>chr</code>	Vector of chromosome numbers.
<code>pos</code>	Vector of positions on chromosomes (much be same length as <code>chr</code>).
<code>tolerance</code>	Minimum distance for two QTL to be considered distinct.
<code>main</code>	Vector of chromosome identifiers with only main effects.
<code>epistasis</code>	Data frame with a 2-element vector of chromosome identifiers for each epistatic pair.
<code>...</code>	Not used here.

Details

Extract architecture in terms of chromosomes and positions of main QTL and identifiers of epistatic pairs of QTL. The `step.fitqtl` approach is used to compare an automatic fit to a user-defined set of main chromosomes and epistatic pairs.

Value

<code>qtl</code>	Data frame with main QTL as <code>chr</code> and <code>pos</code> .
<code>by.num</code>	Data frame with epistatic pairs indexed by chromosome number, labeled <code>qtl_a</code> and <code>qtl_b</code> .
<code>by.chr</code>	List with elements <code>chr</code> and <code>pos</code> showing epistatic pairs. These elements are data frames with chromosomes and positions for each epistatic pair: rows are QTL number, columns are <code>qtl_a</code> and <code>qtl_b</code> .
<code>by.set</code>	List of connected sets of epistatic chromosomes.

Author(s)

Brian S. Yandell

References

<http://www.qtlbim.org>

See Also

`step.fitqtl`, `qb.sweave`

Examples

```
data(qbExample)

## Run qb.scantwo and get summary to use in qb.arch
temp <- summary(qb.scantwo(qbExample, type = "2logBF"),
  threshold = c(upper = 10))
## qb.arch default use.
cross.arch <- qb.arch(temp, chr = c(1,1,2,3), pos = c(15,45,12,15))
cross.arch
```

`qb.close`

Measures closeness of genetic architectures to target.

Description

Boxplots and summaries of how close MCMC samples of genetic architectures are to target architecture.

Usage

```
qb.close(qbObject, target = NULL, epistasis = TRUE, signed = FALSE,
  score.type = c("sq.atten","attenuation","variance","recombination",
  "distance"))
## S3 method for class 'qb.close':
plot(x, category = c("pattern", "nqt1"), xlab,
  cutoff, sort.pattern = c("percent","score"), ...)
## S3 method for class 'qb.close':
summary(object, cutoff, digits = 0,
  show = "score", ... )
```

Arguments

<code>qbObject</code>	Object of class <code>qb</code> .
<code>x,object</code>	Object of class <code>qb.close</code> .
<code>target</code>	Target architecture as data frame with columns <code>chrom</code> and <code>locus</code> . Extracted from <code>summary</code> if <code>target</code> is a qb.scanone object. If <code>target = NULL</code> , <code>score.type</code> is set to "variance" to be compared with null model.
<code>epistasis</code>	Include epistasis in patterns if <code>TRUE</code> .
<code>signed</code>	Sign score. Most useful to examine single chromosome.
<code>score.type</code>	Type of score to use as distance.
<code>category</code>	Boxplots indexed by <code>nqt1</code> or <code>pattern</code> .
<code>xlab</code>	Label for X axis (default taken from <code>x</code> object).
<code>cutoff</code>	Percent cutoff for pattern inclusion in model selection. Default is 0.25 (0.5) if <code>epistasis</code> is <code>TRUE</code> (<code>FALSE</code>).
<code>sort.pattern</code>	If <code>type = "pattern"</code> , sort by <code>percent</code> posterior or by median of <code>score</code> .
<code>digits</code>	Number of digits displayed for locus.
<code>show</code>	Character string with name from <code>object</code> to show.
<code>...</code>	Parameters to methods.

Details

Closeness for each loci is measured as $1-2r$, with r the recombination rate. Thus unlinked loci have measure 0. Loci between a MCMC sample architecture and the target architecture on the same chromosome are matched by closest distance in cM between subsets of the the same length (if target has 2 QTL on chr 3 and sample has 3, consider all pairs from sample to find closest pair in 2-D). Measure per sample is sum across all loci. A quick way to generate a target is to use [qb.BestPattern](#).

The `score.type` is "recombination" = r , the recombination rate; "attenuation" = $1-2r$; "sq.atten" = squared attenuation, "distance" in cM, or genetic "variance".

Author(s)

Brian S. Yandell, yandell@stat.wisc.edu

References

<http://www.qtlbim.org>

See Also

[qb.BestPattern](#), [qb.hpdone](#)

Examples

```
data(qbExample)

## Here target is known for simulated data.
target <- data.frame(chrom = c(1,1,2,3),
  locus = c(15,45,12,15))
temp <- qb.close(qbExample, target)
summary(temp)
plot(temp)
plot(temp, category = "nqtl")

## Or use target from qb.BestPattern
target <- summary(qb.BestPattern(qbExample))$best
qb.close(qbExample, target)
```

`qb.coda`

Coerce to an MCMC object for use with the coda package.

Description

This function creates an object of class `mcmc` from an object of class `qb` produced by the package `R/qtlbim`.

Usage

```
qb.coda(qbObject, element, variables)
```

Arguments

<code>qbObject</code>	An object of class <code>qb</code> returned by calling the function <code>qb.mcmc</code> .
<code>element</code>	A character string which has to one of "iterdiag", "mainloci", "pairloci", "covariates", or "gbye"; default is "iterdiag".
<code>variables</code>	A vector of integers specifying the columns or column names of <code>element</code> to be considered. Details about the columns can be found in <code>qb.mcmc</code>

Details

This package requires the package `coda`.

Value

An object of class `mcmc`. This object could be used to analyze the MCMC output using R/coda.

Author(s)

Dr. Nengjun Yi, et al., nyi@ms.ssg.uab.edu

References

<http://www.qtlbim.org>

See Also

`mcmc`, `qb.mcmc`, `plot.qb`

Examples

```
data(qbExample)

## Default plots for iteration diagnostics "iterdiag".
temp <- qb.coda(qbExample)
plot(temp)

## Summaries for some "mainloci" elements.
temp <- qb.coda(qbExample, "mainloci")
plot(temp)
```

`qb.confound`

Examine confounding of covariate with pseudomarkers.

Description

Covariates used in gene mapping may be correlated with covariates. These routines examine the pattern of confounding.

Usage

```
qb.confound(qbObject, covar = 1)
## S3 method for class 'qb.confound':
plot(x, ylim, main, ...)
## S3 method for class 'qb.confound':
print(x, ...)
## S3 method for class 'qb.confound':
summary(object, ...)
```

Arguments

<code>qbObject</code>	Object of class <code>qb</code> .
<code>x</code>	Object of class <code>qb.confound</code> .
<code>object</code>	Object of class <code>qb.confound</code> .
<code>covar</code>	Index to covariate
<code>ylim</code>	Limits for y (vertical) plotting axis.
<code>main</code>	Title for plot.
<code>...</code>	Additional parameters passed alone.

Details

This examines possible confounding between a covariate and pseudomarkers across the genome. Confounding, evidenced by large correlation with a marker, would raise suspicions about mapping in a genomic region, unless of course the covariate is a marker in that region. Blue curves are correlation with additive effect; red curves are correlation with dominance effect. Dashed lines at 5 percent significance limits.

Value

`qb.confound` returns a matrix with columns for:

<code>coradd</code>	Correlation with additive pseudomarker effect.
<code>cordom</code>	Correlation with dominance pseudomarker effect (if F2).
<code>chr</code>	Chromosome identifier.

The object inherits from `scanone` objects.

Author(s)

Brian S. Yandell

References

<http://www.qtlbim.org>

See Also

[qb.mcmc](#)

Examples

```
data(qbExample)

temp <- qb.confound(qbExample)
plot(temp)
```

qb.covar

Examine GxE effect of covariates on main genetic effects.

Description

Compare main effects with GxE effects to address correlation of estimates.

Usage

```
qb.covar(qbObject, element = "add", covar = 1, adjust.covar, chr, ...)
## S3 method for class 'qb.covar':
summary(object, percent = 5, digits = 3, ...)
## S3 method for class 'qb.covar':
print(x, ...)
## S3 method for class 'qb.covar':
plot(x, percent = 5, cex, include.zero = TRUE, ...)
```

Arguments

qbObject	Object of class qb.
object	Object of class qb.covar.
x	Object of class qb.covar.
element	Main effect to examine ("add" or "dom").
covar	Index to covariates used in MCMC samples.
adjust.covar	Adjustments to covariates. Default is NA, which adjusts by covariate mean values. Values are assumed to be in order of fixed covariates.
chr	Subset of chromosomes as integer vector.
percent	Percentile (0 to 100) for summaries.
digits	Number of significant digits to print.
cex	Character expansion for plots (default decreases with MCMC sample size).
include.zero	Include zero values in plot when TRUE.
...	Arguments passed through to inherited routines.

Details

The diagonal dark green line of points on plots by chromosome indicate adjustment for covariates that have not been centered. Main effects are generally less correlated with GxE when covariates are first centered to have mean zero.

Value

Objects of class `qb.covar` have three columns: main effect, GxE effect and chromosome. Summary objects have eight columns, three for main effect and GxE (mean, lower and upper percentile), followed by correlation and p-value. Summaries are done by chromosome.

Author(s)

Brian S. Yandell

References

<http://www.qtlbim.org>

See Also

[qb.mcmc](#)

Examples

```
data(qbExample)

temp <- qb.covar(qbExample)
summary(temp)
plot(temp)
```

<code>qb.data</code>	<i>Prepares data for qb.mcmc</i>
----------------------	----------------------------------

Description

This function selects trait(s) and covariates from a `cross` object to build a model ([qb.model](#)) for MCMC ([qb.mcmc](#)).

Usage

```
qb.data(cross, pheno.col = 1, trait = c("normal","binary","ordinal"),
        censor = NULL, fixcov = c(0), rancov = c(0), boxcox = FALSE,
        standardize = FALSE, ...)
```

Arguments

<code>cross</code>	An object of class <code>cross</code> . See read.cross for details.
<code>pheno.col</code>	the column number for the phenotype used by <code>model</code> . Currently, only one phenotype can be analyzed at a time.
<code>trait</code>	Type of the quantitative trait or dependent variable: "normal" or "binary" or "ordinal".
<code>censor</code>	Matrix of censor values with 2 columns and <code>nind(cross)</code> rows. Details needed here.
<code>fixcov</code>	list of fixed covariates. The column number(s) in <code>cross\$pheno</code> which is(are) considered as fixed covariates.
<code>rancov</code>	list of random covariates. The column number(s) in <code>cross\$pheno</code> which is(are) considered as random covariates.

<code>boxcox</code>	Indicates whether to use a Boxcox transformation for the dependent variable or not: TRUE or FALSE. Note: trait has to be "normal" and all phenotypic values have to be positive for using this option.
<code>standardize</code>	Indicates whether to standardize the dependent variable or not: TRUE or FALSE. Note: trait has to be "normal" to use this option.
<code>...</code>	Extra terms not used.

Details

This function picks the relevant part of the data from the `cross` object and prepares data for `qb.model` and `qb.mcmc`. It can also standardize or transform continuous data if specified.

Value

<code>yvalue</code>	vector of the values of the dependent variable.
<code>ncategory</code>	number of category type if it is non-normal data.
<code>envi</code>	environment effect: TRUE or FALSE.
<code>fixcov</code>	number of fixed covariates.
<code>rancov</code>	number of random covariates.
<code>fixcoef</code>	values of the fixed covariate(s) for all individuals.
<code>rancoeff</code>	values of the random covariate(s) for all individuals.
<code>nran</code>	number of categories defining the random covariate.
<code>lamda</code>	value of lamda, the transformation parameter for the <code>boxcox</code> transformation.

Note

This function returns a list and hence should have a different name from that of the `cross` object.

Author(s)

Dr. Nengjun Yi, et al., nyi@ms.ssg.uab.edu

References

<http://www.qtlbim.org>

See Also

`qb.genoprob`, `qb.model`, `qb.mcmc`

Examples

```
qbData <- qb.data(cross, pheno.col = 3, rancov = 2, fixcov = 1)
```

<code>plot.qb.diag</code>	<i>Marginal and model-conditional summaries of Bayesian interval mapping diagnostics</i>
---------------------------	--

Description

A density histogram is drawn for model-averaged summary diagnostics such as LOD, variance, or heritability.

Usage

```
qb.diag(qbObject, items= c("mean","envvar","var","herit"), ...)  
## S3 method for class 'qb.diag':  
plot(x, ... )  
## S3 method for class 'qb.diag':  
print(x, ... )  
## S3 method for class 'qb.diag':  
summary(object, digits = 5, ... )
```

Arguments

<code>qbObject</code>	Object of class <code>qb</code> .
<code>object</code>	Object of class <code>qb.diag</code> .
<code>x</code>	Object of class <code>qb.diag</code> .
<code>items</code>	Diagnostics to be summarized; must be name of a column in <code>element</code> .
<code>digits</code>	Number of significant digits.
<code>...</code>	Parameters to methods. Not used for <code>qb.diag</code> .

Details

Model-averaged density is smooth kernel estimate similar to ordinary histogram. A [boxplot](#) (without outliers) is overlaid for comparison with conditional boxplots. Conditional boxplots by number of QTL may show indication of model bias for small number of QTL. This and [qb.BayesFactor](#) can help suggest the minimal model. Diagnostic items that make sense to plot are "LOD", "envvar" (environmental variance), "herit" (heritability), "mean" (grand mean), "addvar" (variance of add), "domvar" (variance of add). Marginal and conditional medians are printed.

Author(s)

Brian S. Yandell, yandell@stat.wisc.edu

References

<http://www.qtlbim.org>

See Also

[plot.qb](#), [density](#), [boxplot](#), [qb.BayesFactor](#)

Examples

```
data(qbExample)

temp <- qb.diag(qbExample)
summary(temp)
plot(temp)
```

<code>plot.qb.epistasis</code>	<i>Density Plots for Models Showing Epistasis and GxE Interactions.</i>
--------------------------------	---

Description

Produces density plots of models showing epistasis (`qb.epistasis`) or GxE interactions (`qb.intcov`). The vertical axis shows magnitude of effect, horizontal axis shows chromosomes in epistatic pairs or covariate by chromosome. Paralell plots are produced for each of the entries in the `effects` parameter.

Usage

```
qb.epistasis(qbObject, effects = c("aa", "ad", "da", "dd"),
  cutoff = 1, maxpair = 5, pairs, ...)
qb.intcov(qbObject, covar, effects = c("add", "dom"),
  cutoff = 1, nmax = 5, cov.chr, ...)
## S3 method for class 'qb.epistasis':
plot(x, effects, cex = 0.5, main, ...)
## S3 method for class 'qb.epistasis':
print(x, ...)
## S3 method for class 'qb.epistasis':
summary(object, ...)
```

Arguments

<code>qbObject</code>	An object of class <code>qb</code> .
<code>object</code>	Object of class <code>qb.epistasis</code> .
<code>x</code>	Object of class <code>qb.epistasis</code> .
<code>cutoff</code>	The <code>cutoff</code> parameter for number of epistatic pairs.
<code>maxpair</code>	Maximum number of epistatic pairs shown.
<code>pairs</code>	A character vector of chromosome pairs to examining for epistatic pairs. Chromosome names are separated by a dot.
<code>covar</code>	Covariate(s) to include; default is <code>seq(nfixcov)</code> where <code>nfixcov</code> is taken from qb.data .

<code>nmax</code>	Maximum number of covariate chromosomes shown.
<code>cov.chr</code>	A character vector of covariate by chromosome pairs to examining for GxE effects. Covariate names and chromosome names are separated by a dot.
<code>effects</code>	Character string of model effects.
<code>cex</code>	Horizontal expansion factor for characters in the plot. See par .
<code>main</code>	Main titles for plots; default is <code>effects</code> .
<code>...</code>	Arguments passed to generic plot .

Value

Returns a table of counts of epistatic pairs with counts above the cutoff value.

Author(s)

Brian S. Yandell, yandell@stat.wisc.edu

References

<http://www.qtlbim.org>

Examples

```
data(qbExample)

temp <- qb.epistasis(qbExample)
summary(temp)
plot(temp)
temp <- qb.intcov(qbExample)
summary(temp)
plot(temp)
```

`fisch`

Eight QTL Stephens and Fisch simulated data

Description

Contains genotypes and phenotypes for data simulated using model in Stephens and Fisch (1998) but with 90 percent heritability.

Usage

```
data(fisch)
```

Format

`fisch` is `f2` (see [read.cross](#) for format).

Author(s)

Brian S. Yandell, <mailto:yandell@stat.wisc.edu>

Source

Patrick J. Gaffney (<mailto:paga@lubrizol.com>), Lubrizol Corp.

References

<http://www.qtlbim.org>

See Also

[read.cross](#), [plot.qb](#), [qb.mcmc](#)

Examples

```
data(fisch)
summary(fisch)
```

<code>step.fitqtl</code>	<i>Stepwise backward elimination and anova comparison.</i>
--------------------------	--

Description

These functions mimic `step` and `anova` but have reduced functionality. They are not truly methods, but can help study qtl model fits.

Usage

```
step.fitqtl(cross, qtl, pheno.col = 1, arch, cutoff = 0.05,
            trace = 1, steps = 100)
## S3 method for class 'step.fitqtl':
anova(object, object2, ...)
```

Arguments

<code>cross</code>	Object of class <code>cross</code> .
<code>qtl</code>	Object of class <code>qtl</code> , as output of makeqtl .
<code>pheno.col</code>	Column of phenotype (numeric).
<code>arch</code>	Object of class <code>qb.arch</code> from qb.arch .
<code>cutoff</code>	Significance cutoff for dropping terms.
<code>trace</code>	If positive, information is printed during the run. Values 1, 2, 3 give gradually more detailed information.
<code>steps</code>	Maximum number of steps to be considered.
<code>object</code>	Object of class <code>step.fitqtl</code> from <code>step.fitqtl</code> .
<code>object2</code>	Object of class <code>step.fitqtl</code> from <code>step.fitqtl</code> .
<code>...</code>	Currently not used.

Details

`step.fitqtl` is analogous to `step` applied to analysis with `fitqtl`. `anova.step.fitqtl` is an S3 method for `anova`. `anova.step.fitqtl` with one argument calls `summary.fitqtl`; with two arguments it attempts to conduct a general F comparison of anova fits.

Value

`step.fitqtl` returns an object of class `step.fitqtl` with

`fit` Object of class `fitqtl`.
`arch` Object of class `qb.arch`.

Author(s)

Brian S. Yandell, yandell@stat.wisc.edu

References

<http://www.qtlbim.org>

See Also

`qb.arch`, `fitqtl`, `summary.fitqtl`, `makeqtl`

Examples

```
cross <- sim.geno(cross, n.draws = 8, step = 2, error = 0.01)
qtl <- makeqtl(cross, chr = c(1,1,2,3), pos = c(15,45,12,15))
cross.step <- step.fitqtl(cross, qtl, pheno.col = 3, arch = cross.arch)
anova(cross.step)
cross.step$arch
```

`qb.genoprob`

Grid point and genotype probability computation method

Description

This function is used to compute putative QTL positions and genotypic probabilities at these positions. The genotypic probabilities for missing marker genotypes are also computed.

Usage

```
qb.genoprob(cross, map.function, step = 2,
            tolerance = 1e-6, stepwidth = "variable", ...)
```

Arguments

<code>cross</code>	An object of class <code>cross</code> . See read.cross for details.
<code>map.function</code>	Indicates what map function to use when converting genetic distances to recombination fractions. See calc.genoprob .
<code>step</code>	Distance (in cM) between positions at which putative QTL positions and their genotype probabilities are calculated. However, specifying <code>step = 0</code> would assume marker positions as putative QTL locations and genotypic probabilities would be calculated only for markers with missing genotype.
<code>tolerance</code>	Minimum separation of markers enforced by jittermap .
<code>stepwidth</code>	Type of stepwidth for calc.genoprob ; "variable" works best with MCMC sampling.
<code>...</code>	Extra arguments to pass to calc.genoprob .

Value

`qb.genoprob` first ensures marker separation is at least `tolerance`, and then computes genotype probabilities at pseudomarkers spaced approximately `step` units apart using [calc.genoprob](#). See [calc.genoprob](#) for value.

Author(s)

Dr. Nengjun Yi, et al., nyi@ms.ssg.uab.edu

References

<http://www.qtlbim.org>

See Also

[jittermap](#), [calc.genoprob](#).

Examples

```
## calculate grids and genotypic probabilities
cross <- qb.genoprob(cross, map.func="haldane", step=2)
```

<code>qb.hpdone</code>	<i>Highest probability density (HPD) region.</i>
------------------------	--

Description

Determine HPD region across genome, including position of posterior mode.

Usage

```
qb.hpdone(qbObject, level = 0.5, profile = "2logBF",
  effects = "cellmean", scan = "sum", chr, smooth = 3, ...)
## S3 method for class 'qb.hpdone':
summary(object, chr, digits = 3, ...)
## S3 method for class 'qb.hpdone':
print(x, ...)
## S3 method for class 'qb.hpdone':
plot(x, chr, ...)
```

Arguments

<code>qbObject</code>	Object of class <code>qb</code> .
<code>object</code>	Object of class <code>qb.hpdone</code> .
<code>x</code>	Object of class <code>qb.hpdone</code> .
<code>level</code>	Value between 0 and 1 of HPD coverage.
<code>scan</code>	Elements to scan; usually one of "sum", "mean", "epistasis", "GxE".
<code>smooth</code>	Degree of smoothing.
<code>chr</code>	Chromosomes to include; default determined by HPD region.
<code>effects</code>	Effects are "cellmean" for means by genotype; "estimate" for estimates of Cockerham main effects.
<code>profile</code>	Objective profile for plot; default is "2logBF"; other choices found in option <code>type</code> for <code>qb.scanone</code> .
<code>digits</code>	Number of digits for <code>round</code> .
<code>...</code>	Extra parameters passed along to plot.

Details

Determine $100 \times \text{level}$ percent HPD region. Subset chromosomes based on HPD region. Create genome scans for `profile` and `effects`.

Value

`qb.hpdone` is a list with a `hpd.region` summary matrix and `qb.scanone` objects for the `profile` and `effects`. A summary of a `qb.hpdone` object yields a matrix with columns for

<code>chr</code>	chromosome number
<code>n.qtl</code>	estimated number of QTL on chromosome
<code>pos</code>	estimated position of QTL
<code>lo.nn%</code>	lower nn% HPD limit
<code>hi.nn%</code>	upper nn% HPD limit
<code>profile</code>	Peak of profile, identified by the profile type.
<code>effects</code>	Columns for the effects, appropriately labeled.

Author(s)

Brian S. Yandell

References

<http://www.qtlbim.org>

See Also

[qb.scanone](#), [qb.hpdchr](#)

Examples

```
data(qbExample)

temp <- qb.hpdone(qbExample)
summary(temp)
plot(temp)
```

<code>qb.hpdchr</code>	<i>Find highest probability density (HPD) region by Chromosome.</i>
------------------------	---

Description

Find area under posterior by chromosome for genome-wide HPD region.

Usage

```
qb.hpdchr(qbObject, level = 0.5, height, hpd, chr, smooth = 3)
```

Arguments

<code>qbObject</code>	Object of class <code>qb</code> .
<code>level</code>	Value between 0 and 1 of HPD coverage. Ignored if <code>height</code> is supplied.
<code>height</code>	Height of posterior corresponding to <code>level</code> . Determined from <code>level</code> and <code>hpd</code> by default.
<code>hpd</code>	Object of class <code>qb.hpdone</code> . Created internally by default.
<code>chr</code>	Chromosomes to include; default determined by genome-wide HPD region.
<code>smooth</code>	Degree of smoothing.

Details

Determine $100 \times \text{level}$ percent HPD coverage by chromosome for a given `height` threshold.

Value

A List with the following elements:

`hpd.height` Height of HPD threshold. Name of `hpd.height` is the level.
`chr.posterior` Posterior probability by chromosome as percent.

Author(s)

Brian S. Yandell

References

<http://www.qtlbim.org>

See Also

[qb.scanone](#)

Examples

```
data(qbExample)
qb.hpdchr(qbExample)
```

`plot.qb.loci` *Jittered plot of Bayesian QTL loci samples by chromosome*

Description

Each point is one locus from the Bayesian QTL estimates, plotted vertically by chromosome, jittered to give a sense of density. Separate colored vertical bands by loci element.

Usage

```
qb.loci(qbObject, loci = c("main", "epistasis", "GxE"), covar, ...)
## S3 method for class 'qb.loci':
plot(x, loci, labels = FALSE, amount = 0.35, cex, col, ...)
## S3 method for class 'qb.loci':
print(x, ...)
## S3 method for class 'qb.loci':
summary(object, digit = 1, ...)
```

Arguments

<code>qbObject</code>	Object of class <code>qb</code> .
<code>loci</code>	Character string identifying MCMC sample elements; may include "main", "epistasis", "GxE" and "all".
<code>covar</code>	Fixed covariate(s) for "GxE" loci; default is all fixed covariates involved in GxE interactions.
<code>x</code>	Object of class <code>qb.loci</code> .
<code>object</code>	Object of class <code>qb.loci</code> .
<code>labels</code>	Include marker labels if TRUE.
<code>amount</code>	Amount of <code>jitter</code> (between 0 and .45)
<code>cex</code>	Character expansion (may be invisible if too small—default based on number of MCMC samples).
<code>col</code>	Character string with colors named by <code>loci</code> ; also includes color for marker lines.
<code>digit</code>	Number of digits for roundoff of loci quantiles.
<code>...</code>	Graphical parameters can be given as arguments to <code>plot</code> . Not used in <code>qb.loci</code> .

Details

Focuses attention on chromosome lengths and concentration of QTL loci estimates. Horizontal lines at markers. Separate bands by `loci` for each chromosome. Adjust `amount` and `cex` to modify look of plot. Most useful when looking at multiple chromosomes.

Author(s)

Brian S. Yandell, yandell@stat.wisc.edu

References

<http://www.qtlbim.org>

See Also

[jitter,subset.qb](#)

Examples

```
data(qbExample)

temp <- qb.loci(qbExample)
plot(temp)
summary(temp)
temp <- qb.loci(qbExample, "all")
plot(temp)
summary(temp)
```

Description

A computationally efficient MCMC algorithm using the Gibbs sampler or Metropolis-Hastings algorithm is used to produce posterior samples for QTL mapping.

Usage

```
qb.mcmc(cross, data, model, mydir = ".", n.iter = 3000, n.thin = 20,
        n.burnin = 0.01*n.iter*n.thin,
        genupdate = TRUE, seed = 0, verbose = TRUE, ...)
```

Arguments

<code>cross</code>	An object of class <code>cross</code> . See read.cross for details.
<code>data</code>	the list returned by calling the function qb.data .
<code>model</code>	the list returned by calling the function qb.model .
<code>mydir</code>	a directory to save output from <code>qb.mcmc</code> in several ‘*.dat’ files. A directory is created using the trait name and the system time and date. If no directory is specified, the default directory is the current working directory.
<code>n.iter</code>	number of iterations to be saved in <code>mydir</code> , the default being 3000. Note that, <code>n.iter</code> is not the total number of iterations performed but the number iterations saved or considered as posterior samples for future analysis. The actual number of iterations would be <code>n.burnin + n.iter*n.thin</code>
<code>n.thin</code>	the thinning number which must be a positive number (default=40)
<code>n.burnin</code>	the initial burn-in period, i.e number of iterations to discard at the beginning of the MCMC run default being <code>0.01*n.iter*n.thin</code> .
<code>genupdate</code>	=TRUE will update QTL genotypes and =FALSE will not do so and use the expected value of the QTL genotypes.
<code>seed</code>	Specifies the seed for the random number generator. Using the same seed for two runs of the <code>qb.mcmc</code> function will generate the exact same output. The <code>seed</code> needs to be an integer. The default value for seed is the system time.
<code>verbose</code>	=TRUE will force periodic output of the number of MCMC iterations saved. The location of the output directory where results are stored and the time taken for the MCMC run will also be displayed to the user.
<code>...</code>	Paramters passed to qb.data or qb.model if <code>data</code> or <code>model</code> , respectively, is not provided.

Details

A composite model space approach to develop a Bayesian model selection framework for identifying interacting QTL for complex traits in experimental crosses from two inbred lines. By placing a liberal constraint on the upper bound of the number of detectable QTL we restrict attention to models of fixed dimension. Either Gibbs sampler or Metropolis-Hastings algorithm can be applied to sample from the posterior distribution.

Value

The `qb.mcmc` function returns a list of class `qb`, the components of which contain input parameters from the `cross` object, `qb.data` and `qb.model`. Since, the parameters have already been described in their respective man pages we only include the components which have been added on top of these. However, the `qbObject$subset` component is a way to manipulate the Monte Carlo samples to make it ready for the high-end plotting routines and might not be of much importance to the average user.

`output.dir` directory used for saving all outputs generated by `qb.mcmc`.

`subset` `iterdiag` is a vector of integers from 1 to `n.iter`.
`mainloci` is a vector of length equaling the sum of the number of QTLs detected in each iteration. This vector is actually a permutation vector of the `mainloci.dat` file stored in `mydir` sorted with respect to the iteration number and ties are broken with the chromosome number and the locus of the putative QTL.
`pairloci` is a list of

- *order* is a vector of integers from 1 to the total number of epistatic effects in all iterations.
- *flip* is a logical vector = TRUE if chromosome no. 1 for the paired loci is greater than chromosome no.2 or if they are the same on the same chromosome then =TRUE when position of the first is greater than the position of the second.
- *left* is a character vector containing "chrom1" and "locus1" if there is a single TRUE in `flip`
- *right* is a character vector containing "chrom2" and "locus2" if there is a single TRUE in `flip`

`region` a data frame storing the first and last position of the marker map for each chromosome.

Output Files

The following files are saved in `output.dir`:

1. **'iterdiag.dat'**

The iteration file saved in `mydir` has `n.iter` rows and 4 major columns:

column no 1 : iteration number.

column no 2 : number of putative QTLs included.

column no 3 : the overall mean.

column no 4 : the residual variance.

Depending on the type of cross, presence of covariates and epistatic effects there would be more columns in the following order: variance of additive effect, variance of dominant effect, variance of additive-additive interaction, variance of additive-dominant interaction, variance of dominant-additive interaction, variance of dominant-dominant interaction, variance of environment-additive interaction, variance of environment-dominant interaction, variance of environment effect, total genetic variance.

2. **'covariates.dat'**

The covariate file saved in `mydir` has `n.iter` rows and `L+M(length(fixcov)+length(rancov))` columns:

`L columns` : Coefficient of the fixed effect.

`M columns` : Variance of the random effect.

If an ordinal trait is analyzed, the cutoff points for the threshold model are also included in additional columns. There would be `C-3` bounded threshold values for an ordinal phenotype with `C` categories.

3. **'mainloci.dat'**

The mainloci file saved in `mydir` has the `N` rows (`N`=sum of number of QTLs detected in `n.iter` iterations) and 4 major columns:

`column no 1` : iteration number.

`column no 2` : number of putative QTLs included in the model.

`column no 3` : the chromosome number on which a putative QTL has been detected.

`column no 4` : the QTL position indicator in the grid.

Depending on the type of cross there would be more columns in the following order: additive effect, dominant effect, variance of additive effect, and variance of dominant effect.

4. **'pairloci.dat'**

The pairloci file saved in `mydir` has the `N` rows (`N`=sum of number of pairs of QTLs with epistatic effect detected) and 6 major columns:

`column no 1` : iteration number.

`column no 2` : number of pairs of QTLs detected to have epistatic effect.

`column no 3` : the chromosome number for the first one of each pair.

`column no 4` : the QTL position for this one.

`column no 5` : the chromosome number for the second one of each pair.

`column no 6` : the QTL position for this one.

Depending on the type of cross there would be more columns in the following order: additive-additive interaction effect, additive-dominant interaction effect, dominant-additive interaction effect, dominant-dominant interaction effect, variance of additive-additive interaction, variance of additive-dominant interaction, variance of dominant-additive interaction, variance of dominant-dominant interaction.

5. 'gbye.dat'

The gbye (Gene by Environment) file saved in `mydir` has 5 major columns:

```
column no 1 : iteration number.  
column no 2 : number of GxE interactions.  
column no 3 : fixed covariate number.  
column no 4 : chromosome number of the putative QTL in the GxE interaction.  
column no 5 : position of the corresponding QTL.
```

Depending on the type of cross there would be more columns in the following order: additive effect, dominant effect, variance of additive effect, and variance of dominant effect.

6. 'deviance.dat'

The deviance file saved in `mydir` has 1 column:

```
column no 1 : posterior deviance.
```

There is one deviance value for each iteration, or `n.iter` values. The last value is the deviance calculated at the posterior means, known as Dhat.

Author(s)

Nengjun Yi, nyi@ms.ssg.uab.edu

References

<http://www.qtlbim.org>

See Also

[qb.sim.cross](#), [qb.data](#), [qb.model](#), [qb.mcmc](#)

Examples

```
## Not run:  
example(qb.sim.cross)  
  
## Calculate grids and genotypic probabilities.  
cross <- qb.genoprob(cross, step=2)  
  
## Create MCMC samples
```

```
## First line as qb.data options; second line has qb.model options.
qbExample <- qb.mcmc(cross, pheno.col = 3, rancov = 2, fixcov = 1,
  chr.nqtl = rep(3,nchr(cross)), intcov = 1, interval = rep(10,3),
  n.iter = 1000, n.thin = 20)
## End(Not run)
```

qb.meancomp

Examine grand mean and covariate MCMC samples.

Description

Examine grand mean and covariate Monte Carlo samples to glean estimates of data center and importance of covariates.

Usage

```
qb.meancomp(qbObject, adjust.covar)
## S3 method for class 'qb.meancomp':
summary(object, percent = 5, ...)
## S3 method for class 'qb.meancomp':
print(x, ...)
## S3 method for class 'qb.meancomp':
plot(x, covar, percent = 5, cex, ...)
```

Arguments

<code>qbObject</code>	Object of class <code>qb</code> .
<code>adjust.covar</code>	Adjustments to covariates. Default is <code>NA</code> , which adjusts by covariate mean values. Values are assumed to be in order of fixed covariates.
<code>object</code>	Object of class <code>qb.meancomp</code> .
<code>x</code>	Object of class <code>qb.meancomp</code> .
<code>percent</code>	Percentile between 0 and 100 for summaries.
<code>covar</code>	Sequence of covariate identifiers for plot.
<code>cex</code>	Character expansion for plot symbols. Default shrinks with number of MCMC iterations.
<code>...</code>	Extra parameters passed along.

Details

Grand mean is adjusted to mean level of covariates. Diagonal of scatterplot matrix includes density plot. Setting `covar = 0` yields a density plot for the grand mean alone.

Value

`qb.meancomp` is a matrix with columns for the grand mean and for each fixed covariate. Summaries show mean and upper and lower percentiles.

Author(s)

Brian S. Yandell

References

<http://www.qtlbim.org>

See Also

[qb.mcmc](#)

Examples

```
data(qbExample)

temp <- qb.meancomp(qbExample)
summary(temp)
plot(temp)
```

qb.model	<i>Set up interacting QTL model for qb.mcmc</i>
----------	---

Description

This function sets up a genome-wide interacting QTL model by specifying global constraints on models and priors on unknowns.

Usage

```
qb.model(cross, epistasis = TRUE, main.nqtl = 3,
  mean.nqtl = main.nqtl + 3, max.nqtl = NULL, interval = NULL,
  chr.nqtl = NULL, intcov = c(0), depen = FALSE,
  prop = c(0.5, 0.1, 0.05), contrast = TRUE, ...)
```

Arguments

<code>cross</code>	An object of class <code>cross</code> . See read.cross for details.
<code>epistasis</code>	indicates if epistasis is included in the model: TRUE or FALSE
<code>main.nqtl</code>	prior expected number of main effect QTLs.
<code>mean.nqtl</code>	prior expected number for all QTLs on all chromosomes including QTLs with main effects, epistatic effects and gene-environment interactions.
<code>max.nqtl</code>	maximum number of QTLs allowed in the model. Default is $l + 3\sqrt{l}$ where l is <code>main.qtl</code> for non-epistatic model and <code>mean.qtl</code> for epistatic model.
<code>interval</code>	minimum distance between any two flanking QTLs for all chromosomes. Default is the average distance between markers in each chromosome.

<code>chr.nqtl</code>	list of the maximum number of QTLs allowed to be detected on each chromosome. Default is the length of the chromosome divided by <code>interval</code> .
<code>intcov</code>	logical or 0/1 vector for fixed covariates indicating which gene-environment interaction will be considered (default is all <code>FALSE</code> , no GxE).
<code>depen</code>	= <code>TRUE</code> will use dependent prior for indicator variables of epistatic effects.
<code>prop</code>	prior inclusion probabilities for epistatic effects in three different scenarios: when both (default 0.5), one (0.1) or none (0.05) of the main effects of the two interacting QTL are included in the model. Note that the sum of the probabilities need not be equal to 1 and <code>prop</code> should be specified only when <code>depen=TRUE</code> .
<code>contrast</code>	Use Cockerham model if <code>TRUE</code> ; otherwise estimate genotypic values.
<code>...</code>	Not used.

Details

This function defines the model for Bayesian QTL mapping using `qb.mcmc`. This model considers two-way interaction as the highest level of both gene-gene and gene-environment interactions.

Value

<code>qtl_envi</code>	Indicates if there is an interaction between the QTLs and environmental variables: <code>TRUE</code> or <code>FALSE</code> .
-----------------------	--

Note

This function returns a list and hence should have a different name from that of the `cross` object.

Author(s)

Dr. Nengjun Yi, et al., nyi@ms.ssg.uab.edu

References

<http://www.qtlbim.org>

See Also

[qb.data](#), [qb.genoprob](#), [qb.mcmc](#)

Examples

```
qbModel <- qb.model(cross, chr.nqtl = rep(3,nchr(cross)), intcov = 1,
  interval = rep(10,3))
```

`plot.qb.mainmodes` *Summaries of QTL modes by chromosome.*

Description

Determine number of QTL per chromosome and estimate peaks and valleys.

Usage

```
qb.split.chr(qbObject, split, ...)
qb.epimodes(qbObject, cutoff = 1, nqtl, n.iter, pairloci, ...)
qb.mainmodes(qbObject, cutoff = 25, nqtl, n.iter, mainloci, ...)
## S3 method for class 'qb.mainmodes':
summary(object, digits = 4, ...)
```

Arguments

<code>qbObject</code>	Object of class <code>qb</code> .
<code>split</code>	List of split locations; names of list must correspond to chromosome names in <code>qbObject</code> . Default set by call to <code>qb.mainmodes</code> with addition arguments in
<code>object</code>	Object of class <code>qb.mainmodes</code> .
<code>cutoff</code>	Cutoff for negligible number of QTL.
<code>nqtl</code>	Vector of number of QTL per chromosome.
<code>n.iter</code>	Number of iterations (for internal use only).
<code>mainloci</code>	Object containing <code>mainloci</code> data (for internal use only).
<code>pairloci</code>	Object containing <code>pairloci</code> data (for internal use only).
<code>digits</code>	Number of significant digits.
<code>...</code>	Parameters to <code>qb.mainmodes</code> or to methods.

Details

Cut off histogram of number of QTL per chromosome such that cumulative percent above number is less than `cutoff`. Once `nqtl` is determined or provided in call, divide MCMC samples using linear discriminant analysis ([lda](#)) and find peak locations per class. Use these peak locations to find locations of valleys between peaks. These valleys are used to divide MCMC samples into separate QTL for analysis. Currently this is used by [summary.qb.scanone](#) and [qb.multloci](#).

`qb.split.chr` sets up a `split` for chromosomes for at valleys between inferred multiple QTL as an attribute of a returned `qbObject`. This is done by default when `qbObject` is created, and is stored as an attribute available as `qb.get(qbObject, "split.chr")`.

Author(s)

Brian S. Yandell, yandell@stat.wisc.edu

References

<http://www.qtlbim.org>

See Also

[qb.multloci](#), [summary.qb.scanone](#)

Examples

```
data(qbExample)

temp <- qb.mainmodes(qbExample)
summary(temp)
```

<code>qb.multloci</code>	<i>Summaries of multiple loci on a chromosome.</i>
--------------------------	--

Description

Summaries and up to four plots showing loci found in MCMC samples for a chromosome.

Usage

```
qb.multloci(qbObject, chr = 1, cutoff = 25, nqtl, ...)
## S3 method for class 'qb.multloci':
plot(x, amount = 0.5, cex,
     split = TRUE, contour = TRUE, weight = TRUE, merge = TRUE, ...)
## S3 method for class 'qb.multloci':
summary(object, merge = TRUE, ...)
```

Arguments

<code>qbObject</code>	Object of class <code>qb</code> .
<code>x,object</code>	Object of class <code>qb.multloci</code> .
<code>chr</code>	Identifier for one chromosome.
<code>cutoff</code>	Smallest posterior probability for <code>nqtl</code> (ignored if <code>nqtl</code> provided).
<code>nqtl</code>	Number of QTL on chromosome (inferred by default).
<code>amount</code>	Amount to <code>jitter</code> points.
<code>cex</code>	Character expansion of plot symbols.
<code>split</code>	Split plots into four panels on one page if <code>TRUE</code> . Otherwise plot each panel separately. The <code>split</code> may be a numeric vector with values 1:4 signifying which panels to show. See details.
<code>contour</code>	Contour plot overlaid on pairs if <code>TRUE</code> .
<code>weight</code>	Inversely weight loci in density plot by number of QTL if <code>TRUE</code> .
<code>merge</code>	Merge across number of QTL if <code>TRUE</code> . Otherwise, show separate summary or plot by number of QTL. See details.
<code>...</code>	Parameters to qb.mainmodes or to methods.

Details

Find multiple loci in MCMC samples for chromosome `chr`. The number of QTL, `nqt1` is inferred from the histogram as the largest number of QTL above the percent `cutoff`.

The generic `plot` command produces the following plots: (1) density plot of main QTL grouped by QTL; (2) histogram of number of QTL; (3) density plot of epistatic pairs; (4) scatter plot of pairs of QTL. The density plots are divided into `nqt1` groups. The scatter plot shows pairs of main loci below diagonal and epistatic pairs above using codes corresponding to the number of QTL per sample; note that 3 QTL have 3 pairs, 4 QTL have 6, etc., and that solitary QTL are displayed along the diagonal.

`split` and `merge` control the manner of plotting. Setting `merge` to `FALSE` yields only density plots for main loci conditioned on the number of QTL per sample. Setting `split` to `FALSE` or to numbers between 1 and 4 yields plots on separate pages.

Author(s)

Brian S. Yandell, yandell@stat.wisc.edu

References

<http://www.qtlbim.org>

See Also

[plot.qb](#), [qb.scantwo](#), [qb.mainmodes](#)

Examples

```
data(qbExample)

temp <- qb.multloci(qbExample, 1)
summary(temp)
plot(temp)
plot(temp, merge = FALSE)
summary(temp, merge = FALSE)
```

`plot.qb.pairloci` *Summaries of epistatic pairs of loci.*

Description

Summaries and detailed scatterplot showing all MCMC samples for epistatic pairs for selected chromosomes.

Usage

```
qb.pairloci(qbObject, chr)
## S3 method for class 'qb.pairloci':
plot(x, main, cex = 0.75, ... )
## S3 method for class 'qb.pairloci':
print(x, ... )
## S3 method for class 'qb.pairloci':
summary(object, ... )
```

Arguments

qbObject	Object of class qb.
object	Object of class qb.pairloci.
x	Object of class qb.pairloci.
chr	Identifiers for one or two chromosomes.
main	Main title for plot.
cex	Character expansion of plot symbols.
...	Parameters to methods.

Details

Find pairs of loci in MCMC samples. Produce scatter plot with generic `plot` or show numerical `summary`. The plot provides position detail complementary to `qb.multloci` and `qb.scantwo`.

Author(s)

Brian S. Yandell, yandell@stat.wisc.edu

References

<http://www.qtlbim.org>

See Also

`plot.qb`, `qb.scantwo`, `qb.multloci`

Examples

```
data(qbExample)

tmpar <- par(mfrow = c(2,2))
temp <- qb.pairloci(qbExample, c(1,2))
summary(temp)
plot(temp)
temp <- qb.pairloci(qbExample, c(1,3))
summary(temp)
plot(temp)
par(tmpar)
```

Description

Diagnostic plots highlight putative QTL loci and effects as well as providing graphical model assessment tools.

Usage

```
## S3 method for class 'qb':  
plot(x, ask = dev.interactive(), verbose = TRUE, ...)  
## S3 method for class 'qb':  
print(x, ...)  
## S3 method for class 'qb':  
summary(object, cutoff = 1, ...)
```

Arguments

x	An object of class <code>qb</code> .
object	An object of class <code>qb</code> .
verbose	Verbose summaries if TRUE.
ask	Ask before each plot if TRUE.
cutoff	Cutoff passed to <code>qb.BayesFactor</code> .
...	graphical parameters can be given as arguments to <code>plot</code>

Details

This generic `plot` routine takes an object of class `qb` created by `qb.mcmc` and produces plots via calls to several other plot routines. The generic `summary` produces a summary, while the generic `print` passes through to `summary`.

Author(s)

Brian S. Yandell, yandell@stat.wisc.edu

References

<http://www.qtlbim.org>

See Also

`plot`, `print`, `summary`, `qb.mcmc`, `qb.coda`, `qb.loci`, `qb.BayesFactor`, `qb.hpdone`, `qb.epistasis`, `qb.diag`

Examples

```
data(qbExample)

plot(qbExample)
summary(qbExample)
```

<code>plot.qb.scanone</code>	<i>Plot or print qb.scanone object.</i>
------------------------------	---

Description

Plot or print marginal diagnostics of effects from a `qb.scanone` object.

Usage

```
## S3 method for class 'summary.qb.scanone':
print(x, digits = 3, ...)
## S3 method for class 'qb.scanone':
print(x, digits = 3, ...)
## S3 method for class 'qb.scanone':
plot(x, chr, scan, ylim, scan.name, ...)
```

Arguments

<code>x</code>	An object of class <code>qb.scanone</code> .
<code>digits</code>	Significant digits to round with print.
<code>chr</code>	Vector of chromosomes to plot. Must be integer.
<code>scan</code>	The model effects to include, the default is all those included in the <code>scanone</code> object <code>x</code> .
<code>ylim</code>	Vector of length 2 with vertical limits.
<code>scan.name</code>	Name of <code>scan</code> for plot; default is "effects" or comma-separated catenation of <code>scan</code> .
<code>...</code>	Other values passed to the generic plot function.

Details

This plot method uses `plot.scanone` as the engine to plot marginal posterior diagnostics created with `qb.scanone`. When there are multiple effects in `x`, these may be organized into one or several stacked plots using `scan`. The default for most diagnostics except counts is `scan = c("sum", "main", "epis")`. Counts and posterior diagnostics are typically plotted in two stacked plots. Individual columns from the `x` object can be plotted by specifying their names as a vector to option `scan`.

Arguments `col` and `lty` use keywords to names in `scan` argument. Default `main` and `sub` arguments are provided.

Value

Data frame with colors and line types used in plots.

Author(s)

Brian S. Yandell, yandell@stat.wisc.edu

References

<http://www.qtlbim.org>

See Also

[qb.scanone](#), [summary.qb.scanone](#), [plot.scanone](#)

Examples

```
example(qb.scanone)
```

<code>plot.qb.scantwo</code>	<i>Two dimensional LOD Plot of Epistasis/joint</i>
------------------------------	--

Description

Plots joint LOD for chromosomes on a two dimensional grid.

Usage

```
## S3 method for class 'qb.scantwo':  
plot(x, chr, smooth = 3, main, offset,  
     nodiag, slice = NULL, show.locus = TRUE,  
     weight = c("sqrt", "count", "none"), verbose = FALSE,  
     split.chr, ...)
```

Arguments

<code>x</code>	An object of class <code>qb.scantwo</code> .
<code>chr</code>	Vector of chromosomes to plot. Must be integer.
<code>smooth</code>	Perform smoothing if > 0 using weighted average over <code>smooth</code> adjacent points.
<code>main</code>	Main title.
<code>offset</code>	Offset to make all values non-negative (see below).
<code>nodiag</code>	If <code>TRUE</code> do not include diagonal in plot.
<code>slice</code>	Take 1-D slice through 2-D surface is not <code>NULL</code> (see below).
<code>show.locus</code>	If a <code>slice</code> , show locus estimate if <code>TRUE</code> .

<code>weight</code>	Weights to use for nearest neighbor smoothing. <code>sqrt</code> is square root of count per locus. Used only if <code>smooth > 0</code> .
<code>verbose</code>	Give verbose feedback if <code>TRUE</code> .
<code>split.chr</code>	Split summary by multiple QTL per chromosome (see details for <code>plot.qb.scanone</code>).
<code>...</code>	Other parameters passed to generic plot function.

Details

The `offset` is used only if `\line{qb.scantwo}` used `type = "estimate"` to make values for plotting all non-negative. Values are rescaled by `offset` so that the origin is at 1 and, by default, the min and max are at 0 and 2, respectively, for each half of the plot. We need this at this time because `plot.scantwo` does not allow negative values.

The `plot.scantwo` argument `nodia` is set to ensure values are all shown and not modified by `plot.scantwo`. Plots with different values for `nodia` or `lower` than the defaults may be non-sensical. For instance, passing `lower = "cond-int"` produces much white area on the image.

A non-null `slice` yields a 1-D view of the 2-D surface. The plots for slices use `plot.scantwo`. The elements of the slice vector are:

`chr` Chromosome number to slice on.

`upper` Focus on upper triangle of 2-D if `TRUE`.

`start` Start position in chromosome `chr` (default = 0).

`end` End position in chromosome `chr` (default = end of chromosome).

`weight` Type of weighted mean across `chr` by number of MCMC samples: 0 = unweighted; 1 = uniform weighting; 2 = position-specific weighting (default).

Value

The `scantwo` object being plotted.

Author(s)

Brian S. Yandell, yandell@stat.wisc.edu

References

<http://www.qtlbim.org>

See Also

`qb.scantwo`, `plot.scantwo`

Examples

```
example(qb.scantwo)
```

Description

These are internal qtlbim routines that are made visible in the namespace for technical use.

Usage

```

covar.mean(qbObject, adjust.covar, verbose = FALSE)
pull.grid(qbObject, offset, spacing, mask.region, cross, step,
  off.end, stepwidth, ...)
pull.loci(cross, step, off.end, stepwidth, region)
qb.cross(qbObject, genoprob = TRUE, ...)
qb.cross.class(qbObject)
qb.demo()
qb.get(qbObject, element, pheno.id = 1, warn = TRUE, ...)
qb.load(cross, qbObject, dir, file)
qb.save(cross, qbObject, dir, Name)
qb.reorder(qbObject, warn = FALSE)

```

Arguments

qbObject	Object of class qb.
adjust.covar	Adjustments to covariates. Default is NA, which adjusts by covariate mean values. Values are assumed to be in order of fixed covariates.
verbose	Verbose mode if TRUE.
cross	Object of class <code>cross</code> (see read.cross).
offset	Offset by first marker if TRUE.
step	Argument to create.map .
off.end	Argument to create.map .
stepwidth	Argument to create.map .
region	Argument to create.map .
genoprob	Compute genotype probabilities with qb.genoprob if TRUE.
spacing	Add columns for map, eq.spacing and xchr if TRUE. This corresponds to map element of a scantwo object.
mask.region	Subset genome regions if TRUE (see subset.qb).
element	Character string for element of qbObject to get. Typically this is a parameter to qb.data , qb.model or qb.mcmc , or it is one of the MCMC sample files in output.dir, from c("iterdiag", "mainloci", "pairloci", "covariates", "gbye").
pheno.id	Numeric or character identifier for phenotype. Useful eventually for multiple traits.

<code>warn</code>	Warn user if <code>qbObject</code> is legacy format if <code>TRUE</code> .
<code>dir</code>	Character string name of directory for load if <code>qbObject</code> does not exist.
<code>file</code>	Character string name of file for load if <code>qbObject</code> does not exist.
<code>Name</code>	Character string name of suffix for save if <code>qbObject</code> exists.
<code>...</code>	Additional arguments.

Details

These are all internal routines. But some may be useful beyond.

`qb.demo` is called in `demo(qb.tour)` and provides an interactive selection of the R/`qtlbim` demos.

`qb.cross` extracts the `cross` object associated with `qbObject`. `qb.get` is the internal main routine for extracting information from a `qbObject`. As stated elsewhere, currently `qbObject` refers to objects that are critical to it but not part of it: the `cross` object used to create it and the MCMC samples in files in `output.dir`.

`covar.mean` finds covariate means or adjusts them to user-supplied values.

`pull.grid` pulls the grid of pseudomarkers from the `cross` object associated with `qbObject`. The option `spacing` determines whether this is in a format similar to `scanone` (`FALSE`) or `scantwo` (`TRUE`). It is used `qb.get` when accessing external MCMC sample files and by several other routines that require pseudomarker information, notably genotype probabilities.

`qb.reorder` is called by `qb.mcmc` to create pointers to reorder the MCMC samples so that chromosome numbers and positions within chromosomes are in increasing order. It creates the `subset` element of a `qb` object.

`qb.save` and `qb.load` are deprecated. They were used to save and retrieve MCMC samples used in examples and vignettes. We now use `save` and `data`.

Examples

```
data(qbExample)

covar.mean(qbExample)
qb.get(qbExample, "output.dir")
summary(qb.cross(qbExample))
temp <- qb.get(qbExample, "iterdiag")
dim(temp)
names(temp)

## Not run:
## The following should have no effect.
qbExample <- qb.reorder(qbExampleb)

## You can call the following rather than demo() to get a tour.
qb.demo()
## End(Not run)
```

Description

Retrieve or recreate MCMC samples used in qtlbim.pdf document and demos.

Usage

```
data(qbExample)
```

Details

This example is used in vignettes and demos. See vignette [qtlbim.pdf](#) or see `qtlbim.Rnw` in doc folder of package.

See Also

[qb.genoprob](#), [qb.mcmc](#), [qb.sim.cross](#)

Examples

```
data(qbExample)
summary(qbExample)

## Not run:
## End(Not run)
## Fake BC Example.
set.seed(1234)
cross <- qb.sim.cross(len = rep(60,3), n.mar = 7, eq.spacing =FALSE,
  n.ind = 100, type = "bc", ordinal = c(0.3,0.3,0.2,0.2),
  missing.geno = 0.03, missing.pheno = 0.03,
  qtl.pos = rbind(qtl.1=c(chr=1,pos=15),qtl.2=c(1,45),qtl.3=c(2,12),qtl.4=c(3,15)),
  qtl.main = rbind(main.1=c(qtl=1,add=1.5),main.2=c(2,0),
    main3=c(3,-1),main4=c(4,0)),
  qtl.epis = rbind(epis1=c(qtl.a=2,qtl.b=3,aa=-2),
    epis2=c(2,4,3)),
  covariate = c(fix.cov=0.5,ran.cov=0.07),
  gbye = rbind(GxE.1=c(qtl=3,add=2)))
summary(cross$qtl)
cross <- qb.genoprob(cross, step=2)
qbExample <- qb.mcmc(cross, pheno.col = 3, rancov = 2, fixcov = 1,
  chr.nqtl = rep(3,nchr(cross)), intcov = 1, seed = 1216,
  interval = rep(10,3), n.iter = 1000, n.thin = 20)

save("qbExample", file = "qbExample.RData")
```

Description

Retrieve MCMC samples for hyper dataset from R/qtl. Code provided to recreate qbHyper if desired. These samples are used in vignettes and demos.

Usage

```
data(qbHyper)
```

See Also

[hyper](#), [qb.genoprob](#), [qb.mcmc](#)

Examples

```
data(qbHyper)
summary(qbHyper)
## Not run:
## Here is code to generate qbHyper.

## Get data from R/qtl.
data(hyper)

## Restrict to autosomes.
hyper <- subset(hyper, chr = 1:19)

## Calculate genotype probabilities.
hyper <- qb.genoprob(hyper, step=2)

## Create MCMC samples.
qbHyper <- qb.mcmc(hyper, n.thin = 40, seed = 1616)

## The next line saves qbHyper as an external binary file.
save("qbHyper", file = "qbHyper.RData")
## End(Not run)
```

Description

Retrieve or recreate MCMC samples used in scan.pdf document.

Usage

```
data(qbSimMain)
data(qbSimEpi)
```

Details

Both calls to `data` create `qb` objects names `qbSim`. See vignette `scan.pdf` or see `scan.Rnw` in `doc` folder of package.

See Also

[qb.genoprob](#), [qb.mcmc](#), [qb.sim.cross](#)

Examples

```
data(qbSimMain)
summary(qbSim)

data(qbSimEpi)
summary(qbSim)

## Not run:
## Setup for Simulated Data used in scan.pdf.
n.ind <- 100          ## number of individuals
n.mark <- 200        ## number of markers
by.mark <- 1         ## cM spacing between markers
qtl.positions <- n.mark / 2    ## position of QTL
markers <- seq(0, n.mark, by = by.mark)
names(markers) <- paste("M", markers, sep = "")
sim.map <- list(ch1 = markers)
sim.model <- matrix(c(1, qtl.positions, qtl.effect / 2), 1, 3)
colnames(sim.model) <- c("chromosome", "qtl-position", "effect-size")
n.iter <- 1000       ## number of iterations for MCMC
qb.random.seed <- 1626 ## random seed for MCMC

## Genetic architecture for scan simulations: 3 QTL.
qtl.positions <- rbind(qtl1 = c(chromosome = 1, locus = 5),
                      qtl2 = c(chromosome = 1, locus = 50),
                      qtl3 = c(chromosome = 2, locus = 33) )

qtl.positions
qtl.main.model <-
  rbind(qtl1.main.effect = c(qtl = 1, main.effect.size = 0),
        qtl2.main.effect = c(qtl = 2, main.effect.size = 0),
        qtl3.main.effect = c(qtl = 3, main.effect.size = 0))

qtl.main.model
qtl.epi.model <- rbind(qtl1.and.qtl3.epi.effect =
  c(qtl1 = 1, qtl2 = 3, epi.effect.size = 10))
qtl.epi.model

## SimEpi
```

```

set.seed(1234)
sim <- qb.sim.cross(len = rep(100, 2), n.mar = 10, eq.spacing = TRUE,
                   n.ind = 100, type = "bc", missing.geno = 0.03,
                   qtl.pos = qtl.positions,
                   qtl.main = qtl.main.model,
                   qtl.epis = qtl.epi.model)
sim <- qb.genoprob(sim)
qbSim <- qb.mcmc(sim, n.iter = n.iter, verbose = FALSE, n.thin = 40,
                 seed = qb.random.seed)

## The next line saves qbSim as an external binary file.
save("qbSim", file = "qbSimEpi.RData")

## SimMain
qtl.main.model[2, "main.effect.size"] = 10
set.seed(1234)
sim <- qb.sim.cross(len = rep(100, 2), n.mar = 10, eq.spacing = TRUE,
                   n.ind = 100, type = "bc", missing.geno = 0.03,
                   qtl.pos = qtl.positions,
                   qtl.main = qtl.main.model,
                   qtl.epis = NULL)

## After the data is simulated call qb.genoprob to fill in
## missing data.
sim <- qb.genoprob(sim, step = 2)

## Call qb.mcmc and then analysis code.
qbSim <- qb.mcmc(sim, n.iter = n.iter, verbose = FALSE, n.thin = 40,
                 seed = qb.random.seed)

## The next line saves qbSim as an external binary file.
save("qbSim", file = "qbSimMain.RData")
## End(Not run)

```

<code>qb.remove</code>	<i>Legacy update, remove or recover of qb object and associated MCMC samples</i>
------------------------	--

Description

Old (2006, R/qtlbim version 2.6) style `qb` objects had important external objects, namely the `cross` object and the MCMC samples in flat files that were loaded as needed. `qb.legacy` upgrades to the new (2007) style `qb` object. The commands `qb.remove` and `qb.recover` are useful to remove and restore old style `qb` objects.

Usage

```

qb.legacy(qbObject, remove = FALSE)
qb.remove(qbObject, verbose = TRUE, external.only = FALSE)
qb.recover(cross, traitName, output.dir, n.thin = 40, n.burnin,
           algorithm = "M-H", genoudate = FALSE, ...)

```

Arguments

<code>qbObject</code>	Object of class <code>qb</code> (see qb.mcmc).
<code>remove</code>	Remove external MCMC samples if <code>TRUE</code> . This will not remove the <code>cross</code> object associated with the <code>qbObject</code> . Be sure to remove the <code>qbObject</code> itself separately.
<code>verbose</code>	Print warning if <code>TRUE</code> .
<code>external.only</code>	Remove only external MCMC samples if <code>TRUE</code> .
<code>cross</code>	Object of class <code>cross</code> (see read.cross).
<code>traitName</code>	Character string name of trait to recover.
<code>output.dir</code>	Character string with name of output directory (inferred if missing).
<code>n.thin</code>	Thinning of MCMC chain used in qb.mcmc .
<code>n.burnin</code>	Burnin of MCMC chain used in qb.mcmc .
<code>algorithm</code>	Algorithm of MCMC chain used in qb.mcmc .
<code>genoupdate</code>	Genotype update flag for MCMC chain used in qb.mcmc .
<code>...</code>	Options passed to qb.data and qb.model .

Details

At the present time, [qb.mcmc](#) stores MCMC samples in external files located in directory `output.dir`, whose name is typically the `traitName` followed by the date. `qb.remove` removes this directory along with the `qbObject`. `qb.recover` attempts to recover the use of an orphaned `output.dir` after a crash of R. These are fragile routines.

Author(s)

Brian S. Yandell

References

<http://www.qtlbim.org>

See Also

[qb.mcmc](#), [qb.genoprob](#)

Examples

```
## Not run:
## Upgrade legacy qb object.
myqbObject <- qb.legacy(myqbObject)

## Recover qbExample for trait "bp" of cross "hyper" using default output.dir.
qbExample <- qb.recover(hyper, "bp")

## Remove internal qbExample and external output.dir.
qb.remove(qbExample)
## End(Not run)
```

Description

This method extracts iteration diagnostics and mainloci from the `qb` object and returns a data frame (of class `qb.scanone`) containing information on environmental variance, explained variance components, non-epistatic variance components.

Usage

```
qb.scanone(qbObject, epistasis = TRUE, scan, type.scan, covar, adjust.covar,
  chr, sum.scan = "yes", min.iter = 1, aggregate = TRUE, smooth = 3,
  weight = c("sqrt","count","none"), split.chr,
  center.type = c("mode","mean","scan"), half = FALSE, verbose = FALSE)
```

Arguments

<code>qbObject</code>	An object of class <code>qb</code> .
<code>epistasis</code>	If <code>TRUE</code> then information about epistasis is included.
<code>scan</code>	Vector of diagnostics to scan (see below).
<code>type.scan</code>	Type of scan; default is "heritability" (see below).
<code>covar</code>	Covariate(s) to include; default is <code>seq(nfixcov)</code> where <code>nfixcov</code> is taken from <code>qb.data</code> . Set to 0 to exclude any covariates.
<code>adjust.covar</code>	Adjustments to covariates. Default is <code>NA</code> , which adjusts by covariate mean values. Values are assumed to be in order of fixed covariates.
<code>chr</code>	Chromosomes to subset on if not <code>NULL</code> .
<code>sum.scan</code>	Sum over <code>scan</code> diagnostics if "yes" or "only"; only report <code>sum</code> if "only".
<code>min.iter</code>	Include only samples at loci if minimum number of iterations is at least <code>min.iter</code> ; default is to include all (<code>min.iter = 1</code>).
<code>aggregate</code>	Aggregate effects into main, epis, gbye if <code>TRUE</code> .
<code>half</code>	Cut epistatic effects in half if <code>TRUE</code> .
<code>smooth</code>	Degree of nearest neighbor smoothing to determine maxima.
<code>weight</code>	Weights to use for nearest neighbor smoothing. <code>sqrt</code> is square root of count per locus. Used only if <code>smooth > 0</code> .
<code>center.type</code>	Method to find QTL loci. See details.
<code>split.chr</code>	Split summary by multiple QTL per chromosome (see details).
<code>verbose</code>	Give verbose feedback if <code>TRUE</code> .

Details

The `type.scan` specifies what type of scan is performed. Scan produces marginal estimates of diagnostics at each potential loci across the genome. That is, values are adjusted for other possible QTL simply by taking the marginal average over MCMC samples. Choices of `type.scan` are "heritability", "LPD", "LR", "deviance", "detection", "variance", "estimate", "cellmean", "count", "log10", "posterior", "logposterior" (i.e. $\log_{10}(\text{posterior})$), "BF", "2logBF" (i.e. $2 \cdot \ln(\text{BF})$), and "nqtl" (number of linked QTL). Default is "LPD".

Type "heritability" is actually R-squared at this point, not the theoretical heritability. Types "LPD", "LR" and "deviance" are all proportional to each other in the usual sense; "LPD" is computed to agree with `lod` from `scanone` if models were restricted to one QTL and missing genotypes are imputed. Detection is the marginal posterior probability of detection of a QTL at a locus. Types "variance" and "estimate" yield, respectively, the marginal variance components and the marginal parameter estimates at each loci. Type "cellmean" gives marginal estimates for A, H, B genotypes (these are single character codes for AA, AB, BB, respectively). The remaining count types provide diagnostics. Types "count" and "log10" report on number of MCMC samples in raw or logged scale. Type "posterior" ("logposterior") yields the marginal (log) posterior probability. Type "BF" ("2logBF") gives the marginal Bayes factor per loci; both are proportional to "count". Type "nqtl" gives the average number of linked loci, which can be useful in sorting out multiple linked loci.

The `scan` specifies the model effects to include for all types except the counts. Aggregated effects (default except for type "cellmean") are "main", "epistasis" and "GxE" (genotype by environment). Individual model effects can be requested as "add", "dom", "aa", "ad", "da", "dd". In addition, GxE terms, if present are included automatically if `covar` is not 0. For type "estimate", main effects for "add" and "dom" are adjusted for any covariate GxE effects. The `sum.scan` is used for all types but the counts to get a summary across `scan` effects.

The "mode" and "mean" centering rely on the mode and mean, respectively, of the posterior; the "scan" centering uses the mode of the actual type used to create the `qb.scanone` object. The "scan" agrees with an `scanone` summary method for "pos" and "sum" columns. However, the mode for a "scan" could be in a region of low posterior mass and may not be reliable as such. Note that `mean` can be biased when there are linked loci. Only used in `qb.scanone` summary.

Evidence for linked loci leads to multiple summary lines per chromosome. By default, a `qbObject` has inferred chromosome splits based on MCMC samples (see `qb.split.chr`). This is determined in a similar manner to `qb.multloci`. In particular, the arguments `cutoff` and `nqtl` documented in `qb.multloci` would adjust whether and how many linked loci may be considered. These apply across all chromosomes being summarized.

Value

Returns an object of class `qb.scanone` (a data frame) containing effects selected according to `type.scan` and `scan`.

Author(s)

Brian S. Yandell, yandell@stat.wisc.edu

References

<http://www.qtlbim.org>

See Also

[summary.qb.scanone](#), [plot.qb.scanone](#)

Examples

```
data(qbExample)

temp <- qb.scanone(qbExample)
summary(temp)
plot(temp)
```

qb.scantwo

Genome Scan for Pairs of Loci Involved in Phenotypic Trait

Description

This method extracts iteration diagnostics and pair loci from the `qb` object and returns a data frame (of class `qb.scanone`) containing information on environmental variance, explained variance components, epistatic and non-epistatic variance components.

Usage

```
qb.scantwo(qbObject, epistasis = TRUE, scan, type.scan,
  upper.scan = "epistasis", lower.scan = "full", covar,
  adjust.covar, chr, min.iter = 1, verbose = FALSE)
```

Arguments

<code>qbObject</code>	An object of class <code>qb</code> .
<code>epistasis</code>	If <code>TRUE</code> information on epistasis is included in the return value.
<code>scan</code>	List of diagnostics to scan (see below).
<code>type.scan</code>	Vector of two scan types; default is "heritability" (see below).
<code>upper.scan</code>	Vector of diagnostics to scan for upper triangle (see below).
<code>lower.scan</code>	Vector of diagnostics to scan for lower triangle (see below).
<code>covar</code>	Covariate(s) to include; default is <code>seq(nfixcov)</code> where <code>nfixcov</code> is taken from <code>qb.data</code> . Set to 0 to exclude any covariates.
<code>adjust.covar</code>	Adjustments to covariates. Default is <code>NA</code> , which adjusts by covariate mean values. Values are assumed to be in order of fixed covariates.
<code>chr</code>	Chromosomes to subset on if not <code>NULL</code> .
<code>min.iter</code>	Include only samples at loci if minimum number of iterations is at least <code>min.iter</code> ; default is to include all (<code>min.iter = 1</code>).
<code>verbose</code>	Give verbose feedback if <code>TRUE</code> .

Details

The `scan` and `type.scan` are similar to those used in [qb.scanone](#). However, here `scan` is a list and `type.scan` is a vector, each with elements "lower" and "upper". You can either specify `scan` as a list, or provide `upper.scan` and `lower.scan` separately.

The `scan` defaults for types other than counts to `list(upper = "epistasis", lower = "full")`; you can modify the list `scan` or the separate options `upper.scan` and `lower.scan`. The string "epistasis" is short-hand for the epistatic effects, `c("aa", "ad", "da", "dd")`. The string "full" is shorthand for the epistatic effects plus main effects, `c("add", "dom")`, plus any GxE terms.

The `type.scan` defaults to `c(upper = "LPD", lower = "LPD")`. See [qb.scanone](#) for the range of possible types. Mostly the 2-D version of `type.scan` provides marginal summaries for pairs of loci. However, for type "nqt1", the marginal summaries involving main effects (e.g. with `scan` values "full" or "main" or "add" or "dom") show, for each pair of chromosomes, the average number of QTL at both chromosomes.

Value

Returns an object of class `qb.scantwo` (a data frame) containing effects selected according to `type.scan` and `scan`.

Author(s)

Brian S. Yandell, yandell@stat.wisc.edu

References

<http://www.qtlbim.org>

See Also

[plot.qb.scanone](#)

Examples

```
data(qbExample)

temp <- qb.scantwo(qbExample)
summary(temp)
plot(temp)
```

`qb.sim.cross`

Simulates QTL related data for an F2 or BC cross.

Description

This function is used to simulate genotypic, phenotypic and covariate data for BC and F2 populations. The underlying genetic model is Cockerham's model and data for both continuous (normally distributed only) and ordinal traits can be generated.

Usage

```
qb.sim.cross(len = rep(100,20), n.mar = 11, eq.spacing = TRUE,
  n.ind = 400, type = c("f2","bc","riself","risib"), missing.geno = 0.0,
  missing.pheno = 0.0, ordinal = c(0.5,0.5),
  qtl.pos = NULL, qtl.main = NULL, qtl.epis = NULL,
  covariate = NULL, gbye = NULL, seed = NULL )
## S3 method for class 'qb.sim':
summary(object, ...)
```

Arguments

<code>len</code>	defines the length (in cM) of each chromosome and number of chromosomes. Thus <code>len = c(80,90,44)</code> would represent a model with three chromosomes of lengths 80, 90, and 44 respectively.
<code>n.mar</code>	The number of markers per chromosome. This can be specified as a single number or as a vector. If a single number is specified, all the chromosomes will have the same number of markers. If <code>n.mar</code> is a vector then it must have the same number of entries as there are chromosomes. For example, if <code>n.mar = c(10,11,9)</code> then we have a three chromosome model in which the first chromosome has 10 markers, the second has 11 and the third has 9. a vector specifying the number of markers per chromosome.
<code>eq.spacing</code>	if TRUE, markers will be equally spaced. Default is TRUE. If FALSE, markers are generated uniformly over the chromosome.
<code>n.ind</code>	specifies the number of individuals.
<code>type</code>	indicates whether to simulate an intercross ("f2") or a backcross ("bc").
<code>missing.geno</code>	the frequency of missing genotypes.
<code>missing.pheno</code>	the frequency of missing phenotypic values.
<code>ordinal</code>	define the probabilities of each ordinal category and the number of elements in the vector will determine the number of categories. The elements must be positive and the should sum up to 1
<code>qtl.pos</code>	This parameter specifies the positions of qtl as a matrix with dimensions (number of qtl) x 2. Note that the row dimension is the number of qtl and is not the number of chromosomes. Each row identifies a qtl, the first column entries represent the chromosome's index, the second column entries represent the location on the chromosome of the qtl. The (row) order in which qtl are listed in this parameter is the index by which they are identified later on in the parameters <code>qtl.main</code> and <code>qtl.epi</code> .
<code>qtl.main</code>	The parameter <code>qtl.main</code> is a matrix specifying the main effects of QTLs. The first column gives the qtl-index (the row index of the qtl in the <code>qtl.pos</code> parameter.), the second and third column gives the additive and dominance effects, respectively. There are two or three columns depending on <code>type</code> being "bc" or "f2".
<code>qtl.epis</code>	It is a matrix specifying epistatic effects. There are 3 or 6 columns depending on <code>type</code> being "bc" or "f2". Each row gives an epistatic pair. The first entry in a row gives the first qtl index, the second entry represents the index of the second qtl. The other entries give the value of

	the epistatic effects (additive-additive, additive-dominance, dominance-additive and dominance-dominance) of the two qtls. The indices used to represent the qtl are the row indices of the <code>qtl.pos</code> matrix which correspond to the first and second qtl in each epistatic pair.
<code>covariate</code>	A vector of two elements, the first being the true value of the coefficient for the fixed covariate and the second the true value for the standard deviation of the random covariate.
<code>gbye</code>	A matrix specifying the interaction between the fixed covariate and QTL main effect. The first column is the index of the QTL, the other column(s) is(are) the value(s) of interaction(s).
<code>seed</code>	Set pseudo-random number seed with <code>set.seed</code> if not NULL.
<code>object</code>	An object of class <code>qb.sim</code> , typically the <code>qtl</code> element of a <code>cross</code> object created by <code>qb.sim.cross</code> .
<code>...</code>	Not used here.

Details

The most important difference of this simulation function from others is that it computes phenotype values with full genetic model. i.e. both additive, dominance, and epistatic effects are considered. Furthermore, environmental effects and gene-environment interactions can be included in the model to simulate phenotypes.

The outputted genotypes for markers and qtls will be coded as 1 and 2 for BC and 1,2, and 3 for F2. Missing data will be coded as NA.

Value

`qb.sim.cross` will return an object of class `cross`. See `read.cross` for details. In addition, a component `qtl` of class `qb.sim` is added which is a list of at most 6 components depending on the options specified.

<code>geno</code>	is a matrix of true QTL genotypes for every individual and each locus. The genotypes are defined following <code>read.cross</code> .
<code>pos</code>	is a matrix of true QTL position. Same as <code>qtl.pos</code> .
<code>herit.main</code>	is a matrix of the heritability of main effects. <code>nrow(\$qtl\$herit.main)=no. of QTLs</code> and <code>ncol(\$qtl\$herit.main)=2 or 3</code> depending on the type of genetic cross ("bc" or "f2"). The first column being the QTL index and the others being additive and dominant heritability respectively.
<code>herit.epis</code>	is a matrix of the heritability of epistatic effects. <code>nrow(\$qtl\$herit.epis)=no. of QTLs pairs interacting</code> and <code>ncol(\$qtl\$herit.main)=3 or 6</code> depending on the type of genetic cross ("bc" or "f2"). The first column being the QTL index and the others being additive-additive, additive-dominant, dominant-additive and dominant-dominant heritability respectively.
<code>herit.cov</code>	is a vector of length 2 containing the heritability of the fixed and random covariate.

herit.gbye is a matrix of heritability of GxE interactions. `nrow(qtlherit.gbye)`= no. of GxE interactions and `ncol(qtlherit.gbye)`= 2 or 3 depending on the type of genetic cross ("bc" or "f2"). The first column being the GxE index and the others being additive and dominant GxE interaction heritability.

Author(s)

Dr. Nengjun Yi, et al., nyi@ms.ssg.uab.edu

References

<http://www.qtlbim.org>

See Also

[qb.genoprob](#), [qb.data](#) [qb.model](#), [qb.mcmc](#), [sim.cross](#)

Examples

```
## Not run:
## Simulate large intercross.
cross <- qb.sim.cross(len = rep(100,20), n.mar = 11, eq.spacing =FALSE,
  n.ind = 500, type = "f2", ordinal = c(0.3,0.3,0.2,0.2),
  missing.geno = 0.03, missing.pheno = 0.03,
  qtl.pos = rbind(qtl.1=c(chr=1,pos=15),qtl.2=c(1,45),qtl.3=c(3,12),
    qtl.4=c(5,15),qtl.5=c(7,15),qtl.6=c(10,15),qtl.7=c(12,35),qtl.8=c(19,15)),
  qtl.main = rbind(main.1=c(qtl=1,add=0.5,dom=0),main.2=c(2,0,0.7),
    main3=c(3,-0.5,0),main4=c(4,0.5,-0.5)),
  qtl.epis = rbind(epis1=c(qtl.a=4,qtl.b=5,aa=-0.7,ad=0,da=0,dd=0),
    epis2=c(6,8,0,1.2,0,0)),
  covariate = c(fix.cov=0.5,ran.cov=0.07),
  gbye = rbind(GxE.1=c(qtl=7,add=0.8,dom=0)) )

## Examine simulation information.
summary(cross$qtl)
## End(Not run)

## Simulate small backcross.
cross <- qb.sim.cross(len = rep(60,3), n.mar = 7, eq.spacing =FALSE,
  n.ind = 100, type = "bc", ordinal = c(0.3,0.3,0.2,0.2),
  missing.geno = 0.03, missing.pheno = 0.03,
  qtl.pos = rbind(qtl.1=c(chr=1,pos=15), qtl.2=c(1,45),
    qtl.3=c(2,12), qtl.4=c(3,15)),
  qtl.main = rbind(main.1=c(qtl=1,add=1.5), main.2=c(2,0),
    main3=c(3,-1), main4=c(4,0)),
  qtl.epis = rbind(epis1=c(qtl.a=2,qtl.b=3,aa=-2), epis2=c(2,4,3)),
  covariate = c(fix.cov=0.5,ran.cov=0.07),
  gbye = rbind(GxE.1=c(qtl=3,add=2)))
summary(cross$qtl)
```

Description

This method extracts iteration diagnostics and mainloci from the `qb` object and returns a data frame (of class `qb.sliceone`). Generic summary and plot can be used for display.

Usage

```
qb.sliceone(qbObject, slice, epistasis = TRUE, scan, type.scan, covar,
  adjust.covar, chr, sum.scan = "yes", min.iter = 1,
  aggregate = TRUE, smooth = 3, weight = c("sqrt","count","none"),
  split.chr, center.type = c("mode","mean","scan"), verbose = FALSE)
## S3 method for class 'qb.sliceone':
summary(object, chr, ...)
## S3 method for class 'qb.sliceone':
print(x, ...)
## S3 method for class 'qb.sliceone':
plot(x, ..., scan, auto.par = TRUE)
```

Arguments

<code>qbObject</code>	An object of class <code>qb</code> .
<code>object</code>	Object of class <code>qb.sliceone</code> .
<code>x</code>	Object of class <code>qb.sliceone</code> .
<code>slice</code>	Chromosomes to slice upon.
<code>epistasis</code>	If <code>TRUE</code> then information about epistasis is included.
<code>scan</code>	Vector of diagnostics to scan (see below).
<code>type.scan</code>	Type of scan; default is "heritability" (see below).
<code>covar</code>	Covariate(s) to include; default is <code>seq(nfixcov)</code> where <code>nfixcov</code> is taken from <code>qb.data</code> . Set to 0 to exclude any covariates.
<code>adjust.covar</code>	Adjustments to covariates. Default is <code>NA</code> , which adjusts by covariate mean values. Values are assumed to be in order of fixed covariates.
<code>chr</code>	Chromosomes to subset on if not <code>NULL</code> .
<code>sum.scan</code>	Sum over <code>scan</code> diagnostics if "yes" or "only"; only report <code>sum</code> if "only".
<code>min.iter</code>	Include only samples at loci if minimum number of iterations is at least <code>min.iter</code> ; default is to include all (<code>min.iter = 1</code>).
<code>aggregate</code>	Aggregate effects into main, epis, gbye if <code>TRUE</code> .
<code>smooth</code>	Degree of nearest neighbor smoothing to determine maxima.
<code>weight</code>	Weights to use for nearest neighbor smoothing. <code>sqrt</code> is square root of count per locus. Used only if <code>smooth > 0</code> .

<code>split.chr</code>	Split summary by multiple QTL per chromosome (see details for plot.qb.scanone).
<code>center.type</code>	Method to find QTL loci. See details.
<code>verbose</code>	Give verbose feedback if TRUE.
<code>auto.par</code>	Automatic setting of plot parameters for multiple plots if TRUE.
<code>...</code>	Arguments to be passed along.

Details

All arguments except `slice` agree with [qb.scanone](#). The `slice` specifies a chromosome upon which to slice, yielding a 1-D scan of what might be seen on a 2-D scan using [qb.scantwo](#). One advantage of [qb.sliceone](#) is that you can get 2-QTL cell means for the slice chromosome and the scanned chromosomes.

The summary invokes [summary.qb.scanone](#) to summarize slice by chromosome. The plot will by default give separate plots for each slice genotype and use [plot.qb.scanone](#) to scan the chromosomes. If `scan` is specified for [plot.qb.sliceone](#), then those elements will be plotted. For instance, `plot(x,scan="slice")` will plot the running average locus on the slice chromosome with respect to the other chromosomes.

Value

`qb.sliceone` returns an object of class `qb.sliceone` (a data frame) containing effects selected according to `type.scan` and `scan`.

Author(s)

Brian S. Yandell, yandell@stat.wisc.edu

References

<http://www.qtlbim.org>

See Also

[summary.qb.scanone](#), [plot.qb.scanone](#)

Examples

```
data(qbExample)

## Get profile of heritability.
temp <- qb.sliceone(qbExample, slice = 1, chr = 2:3)
summary(temp)
plot(temp)

## Get profile of cell means.
temp <- qb.sliceone(qbExample, slice = 1, chr = 2:3, type.scan = "cellmean")
summary(temp)
plot(temp)
```

qb.slicetwo

Slices for epistatic pairs.

Description

These routines refine QTL positions for epistatic pairs and show plots to reveal the nature of epistasis.

Usage

```
qb.slicetwo(qbObject, chr, pos, type.scan = "2logBF", width = 10)
## S3 method for class 'qb.slicetwo':
summary(object, ...)
## S3 method for class 'qb.slicetwo':
print(x, ...)
## S3 method for class 'qb.slicetwo':
plot(x, byrow = TRUE, figs, auto.par = TRUE, ...)
```

Arguments

qbObject	Object of class qb.
object	Object of class qb.slicetwo.
x	Object of class qb.slicetwo.
chr	Chromosome vector.
pos	Position vector corresponding to chr.
type.scan	Type of profile scan; see qb.scanone .
width	Width of slice.
byrow	Arrange plots by row (for slides) if TRUE.
figs	Plot only selected figures. Full set of c("profile", "effects", "cellmean", "effectplot") is default.
auto.par	Automatic setting of plot parameters for multiple plots if TRUE.
...	Extra plot options.

Author(s)

Brian S. Yandell, yandell@stat.wisc.edu

References

<http://www.qtlbim.org>

See Also

[qb.scantwo](#), [qb.sliceone](#)

Examples

```
data(qbExample)

temp <- qb.slicetwo(qbExample, chr = c(1,2), pos = c(45,12))
summary(temp)
plot(temp)
```

subset.qb

Subsetting Bayesian interval mapping data

Description

Subset Bayesian interval mapping iterations on number of QTL and/or chromosome pattern of QTL, using exact match or inclusive subsetting.

Usage

```
## S3 method for class 'qb':
subset(x, nqtl=1, pattern=NULL, exact=FALSE, chr,
       region, offset = TRUE, restrict.pair = TRUE, pheno.id = 1, ...)
```

Arguments

<code>x</code>	object of class <code>qb</code>
<code>nqtl</code>	subset on number of QTL
<code>pattern</code>	subset on chromosome pattern of QTL
<code>exact</code>	subset on exact pattern or number of QTL if true
<code>chr</code>	subset of chromosomes to plot (numerical indices, logical or chromosome names)
<code>region</code>	list containing <code>chr</code> , and <code>start</code> and <code>end</code> positions, for regions to include
<code>offset</code>	indicates that <code>start</code> and <code>end</code> are in cM position if <code>TRUE</code> ; otherwise they are in distance from first marker
<code>restrict.pair</code>	Restrict <code>chr</code> and <code>region</code> selection to linked loci all in selected subset.
<code>pheno.id</code>	Numeric or character identifier for phenotype. Useful eventually for multiple traits.
<code>...</code>	additional arguments to <code>subset</code>

Details

Subset to include only iterations with at least `nqtl` number of QTL and at least the `pattern` across chromosomes. `pattern` is a vector of chromosome identices, with repeats for multiple linked QTL on a chromosome. If `exact=FALSE`, then all iterations with at least the given `pattern` and `nqtl` are included. `nqtl` will be reset to `length(pattern)` if it is smaller than this value. Note that `pattern` should be number codes corresponding to those used in the `x` object. Further subsets to only include QTL from these iterations that are on chromosomes `chr`.

Author(s)

Brian S. Yandell, yandell@stat.wisc.edu

References

<http://www.qtlbim.org>

See Also

[read.cross](#)

Examples

```
data(qbExample)

## Subset to chr 1,2, and to within 10 cM of QTL on chr 1,2.
qbSubset <- subset(qbExample, chr = c(1,2),
  region = data.frame(chr = c(1,2), start = c(35,2), end = c(55,22)))
```

`summary.qb.scanone` *Summary of qb.scanone or qb.scantwo object.*

Description

Summary of a `qb.scanone` or `qb.scantwo` object.

Usage

```
## S3 method for class 'qb.scanone':
summary(object, chr, threshold = 0,
  sort = "no", n.qtl = 0.05, ...)
## S3 method for class 'qb.scantwo':
summary(object, chr, threshold = 0,
  sort = "no", which.pos = "upper", min.iter,
  refine = FALSE, width = 10, smooth = 3, n.qtl = 0.05,
  weight = c("sqrt","count","none"), ...)
## S3 method for class 'summary.qb.scantwo':
print(x, digits = 3, ...)
```

Arguments

<code>object</code>	A <code>qb.scanone</code> or <code>qb.scantwo</code> object.
<code>x</code>	An object of class <code>qb.scantwo</code> .
<code>chr</code>	Chromosomes to include in summary (must be integers for now).
<code>threshold</code>	Threshold(s) for inclusion in summary (see below).
<code>sort</code>	Sort by selected column of <code>object</code> ("no" indicates sort by chromosome).

<code>which.pos</code>	Base position estimate on this summary for maximal statistics such as LOD.
<code>min.iter</code>	Minimum number of iterations included at each position (default gleaned from <code>object</code>).
<code>refine</code>	Refine estimates if TRUE.
<code>width</code>	Window width for refinement.
<code>smooth</code>	Degree of nearest neighbor smoothing to determine maxima.
<code>n.qtl</code>	Minimum number of estimated QTL per chromosome or chromosome pair.
<code>weight</code>	Weights to use for nearest neighbor smoothing. <code>sqrt</code> is square root of count per locus. Used only if <code>smooth > 0</code> .
<code>digits</code>	Significant digits to round with print.
<code>...</code>	Additional arguments for multiple linked loci (see details).

Details

These summary method report estimates by chromosome (or chromosome pair) at the maximum poster. Threshold can be used to condense summary to a subset of chromosomes (or chromosome pairs). Threshold is a vector with names corresponding to a subset of column names of `object`. Positive threshold values select chromosomes where that column average is above given value; negative threshold values select chromosomes with mean value within that value of the maximum across chromosomes. Thresholding is inclusive rather than exclusive.

It can be helpful to use `summary.qb.scanone` as an initial screen of chromosomes worth a further look. Since marginal summaries can include effects of multiple QTL and epistasis. Subsets based on 1-D scans can be used for 2-D subsequent screens. See [qb.demo](#) for an example.

Value

Matrix with chromosome `chr`, estimated position `pos` (or chromosome pairs `chr1` and `chr2` and two columns for `pos1` and `pos2` in the case of `summary.qb.scantwo`) and means or modes of each column of `object`. Means are weighted by a smooth average of the number of MCMC sample iterations.

Author(s)

Brian S. Yandell, yandell@stat.wisc.edu

References

<http://www.qtlbim.org>

See Also

[qb.scanone](#), [plot.qb.scanone](#), [qb.split.chr](#)

Examples

```
data(qbExample)

temp <- qb.scanone(qbExample)
summary(temp, threshold = c(sum=15), sort = "sum")

temp <- qb.scantwo(qbExample)
summary(temp, threshold = c(upper=3), sort = "upper")
```

`qb.sweave`

Run sweave to automate QTL search with MCMC samples.

Description

This routine runs a separate Sweave file (*.Rnw) of commands, making substitutions for the user-supplied data and thresholds. It can be used to automate the search for genetic architecture.

Usage

```
qb.sweave(cross, pheno.col = 1, n.iter = 3000, n.draws = 64,
  scan.type = "2logBF", hpd.level = 0.5,
  upper.threshold, SweaveFile, SweaveExtra, PDFDir, remove.qb = TRUE)
```

Arguments

<code>cross</code>	Object of class <code>cross</code> .
<code>pheno.col</code>	Phenotype column in object <code>cross</code> .
<code>n.iter</code>	Number of MCMC iterations to be stored.
<code>n.draws</code>	Number of MC draws to use for <code>fitqtl</code> .
<code>scan.type</code>	Type of 1-D and 2-D scan to perform; see qb.scanone .
<code>hpd.level</code>	Highest probability density level for scan; see qb.hpdone .
<code>upper.threshold</code>	Threshold for upper triangle (epistasis) in 2-D scan; see qb.scantwo .
<code>SweaveFile</code>	Name of Sweave file (default is <code>system.file("doc", "prototype_qtl_hyper_data.Rnw", package = "qtlbim")</code>).
<code>SweaveExtra</code>	Name of user-supplied extra Sweave file (default is <code>NULL</code>).
<code>PDFDir</code>	Name of directory to store PDF files (default is <code>phenoPDF</code> , where <code>pheno</code> is the name associated with phenotype <code>pheno.col</code>).
<code>remove.qb</code>	Remove constructed objects if <code>TRUE</code> .

Details

This is a simple shell around the [Sweave](#) routine to create customized documents with embedded QTL analysis. The default file `system.file("doc", "prototype.qtl.hyper.slides.Rnw", package = "qtlbim")` creates a "beamer" style PDF slide show. An alternative file `system.file("external", "prototype.qtl.hyper.paper.Rnw", package = "qtlbim")` creates a preprint document. Both require post-processing with `pdflatex`.

A user-defined section can be added to the automated documents, using the `SweaveExtra` option. We have provided `system.file("external", "hyper.slide.extra.Rnw", package = "qtlbim")` for the slide version and `system.file("external", "hyper.paper.extra.Rnw", package = "qtlbim")` for the preprint version.

Author(s)

Brian S. Yandell, yandell@stat.wisc.edu

References

<http://www.qtlbim.org>

See Also

[Sweave](#)

Examples

```
## Not run:
data(hyper)

## Create default slide show LaTeX source without extra section.
qb.sweave(hyper)

## Turn LaTeX into PDF. Run twice to get outline correct.
## Need pdflatex on your system.
system("pdflatex prototype.qtl.hyper.slides")
system("pdflatex prototype.qtl.hyper.slides")

## Create document form, with extra section.
qb.sweave(hyper,
  SweaveFile = system.file("external", "prototype.qtl.hyper.paper.Rnw", package = "qtlbim"),
  SweaveExtra = system.file("external", "hyper.paper.extra.Rnw",
    package = "qtlbim"))
system("pdflatex prototype.qtl.hyper.paper")
system("pdflatex prototype.qtl.hyper.paper")
## End(Not run)
```

Description

These routines extract and summarize variance components for Bayesian multiple QTL. Variance components are averaged over genome loci. Covariates and GxE may be included.

Usage

```
qb.varcomp(qbObject, scan, aggregate = TRUE)
## S3 method for class 'qb.varcomp':
summary(object, ...)
## S3 method for class 'qb.varcomp':
print(x, ...)
## S3 method for class 'qb.varcomp':
plot(x, log = TRUE, percent = 5, cex, ...)
```

Arguments

qbObject	Object of class <code>qb</code> .
object	Object of class <code>qb.varcomp</code> .
x	Object of class <code>qb.varcomp</code> .
scan	Aggregated terms to include in created object (see below).
aggregate	Sum over individual components of aggregated terms if <code>TRUE</code> .
log	Use <code>log10</code> of variances in plot if <code>TRUE</code> .
percent	Percentile between 0 and 100 for summaries.
cex	Character expansion for plot symbols. Default shrinks with number of MCMC iterations.
...	Arguments to pass along.

Details

Variance components are organized as "main" ("add" and "dom"), "epistasis" ("aa", etc.), "fixcov" (for all fixed covariate terms), "rancov" (random covariates), and "GxE" (genotype by environment, including additive and dominance terms). Any subset of these may be chosen.

Value

`qb.varcomp` creates a matrix with columns of samples for the variance components. Each row represents an MCMC iteration. Values are averaged over loci.

Author(s)

Brian S. Yandell

References

<http://www.qtlbim.org>

See Also

[qb.mcmc](#)

Examples

```
data(qbExample)

temp <- qb.varcomp(qbExample)
summary(temp)
plot(temp)
```

vern

Eight week vernalization data for Brassica napus

Description

Contains genotypes and phenotypes for 8-week vernalization study used in Satagopan et al. (1996).

Usage

```
data(vern)
```

Format

See [read.cross](#) for format of `vern`.

Source

Thomas C. Osborn (<mailto:tcosborn@facstaff.wisc.edu>), Department of Agronomy, UW-Madison.

References

<http://www.stat.wisc.edu/~yandell/qtl/data/osborn/Bnapus>

Ferreira ME, Satagopan J, Yandell BS, Williams PH, Osborn TC (1995) Mapping loci controlling vernalization requirement and flowering time in *Brassica napus*. *Theor Appl Genet* 90: 727-732. [original source and analysis]

Kole C, Thorman CE, Karlsson BH, Palta JP, Gaffney P, Yandell BS, Osborn TC (2001) Comparative mapping of loci controlling winter survival and related traits in oilseed *Brassica rapa* and *B. napus*. *Molecular Breeding* 1: 329-339. [refined map and reanalysis]

JM Satagopan, BS Yandell, MA Newton and TC Osborn (1996) Markov chain Monte Carlo approach to detect polygene loci for complex traits. *Genetics* 144: 805-816. <http://www.genetics.org/cgi/content/abstract/144/2/805> [first MCMC for experimental crosses; analysis of *B. napus* N2=LG9; see `vern` data]

See Also

`read.cross`, `plot.qb`, `qb.mcmc`

Examples

```
data(vern)
summary(vern)
```