The qtl Package

October 9, 2007

Version 1.07-12

Date 9 October 2007

Title Tools for analyzing QTL experiments

Author Karl W Broman <kbroman@biostat.wisc.edu> and Hao Wu, with ideas from Gary Churchill and Saunak Sen and contributions from Brian Yandell

Maintainer Karl W Broman <kbroman@jhsph.edu>

Description Analysis of experimental crosses to identify genes (called quantitative trait loci, QTLs) contributing to variation in quantitative traits.

License GPL version 2 or later

URL http://www.rqtl.org

R topics documented:

A starting point .................................................. 3
add.cim.covar .................................................. 8
add.threshold ................................................... 9
argmax.geno .................................................... 10
arithscan ......................................................... 12
arithscanperm ................................................... 13
badorder ......................................................... 14
bayesint ........................................................ 15
bristle3 .......................................................... 16
bristleX .......................................................... 17
c.cross .......................................................... 18
c.scanone ....................................................... 19
c.scanoneperm .................................................. 20
c.scantwoperm .................................................. 21
calc.errorlod .................................................... 22
calc.genoprob ................................................... 23
cbind.scanoneperm ............................................. 25
checkAlleles ..................................................... 26
### R topics documented:

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>chrlen</td>
<td>27</td>
</tr>
<tr>
<td>cim</td>
<td>28</td>
</tr>
<tr>
<td>clean.cross</td>
<td>29</td>
</tr>
<tr>
<td>clean.scantwo</td>
<td>30</td>
</tr>
<tr>
<td>comparecrosses</td>
<td>31</td>
</tr>
<tr>
<td>comparegeno</td>
<td>32</td>
</tr>
<tr>
<td>condense.scantwo</td>
<td>33</td>
</tr>
<tr>
<td>convert.scanone</td>
<td>34</td>
</tr>
<tr>
<td>convert.scantwo</td>
<td>35</td>
</tr>
<tr>
<td>drop.markers</td>
<td>36</td>
</tr>
<tr>
<td>drop.nullmarkers</td>
<td>36</td>
</tr>
<tr>
<td>effectplot</td>
<td>37</td>
</tr>
<tr>
<td>effectscan</td>
<td>40</td>
</tr>
<tr>
<td>est.map</td>
<td>41</td>
</tr>
<tr>
<td>est.rf</td>
<td>43</td>
</tr>
<tr>
<td>fake.4way</td>
<td>44</td>
</tr>
<tr>
<td>fake.bc</td>
<td>46</td>
</tr>
<tr>
<td>fake.f2</td>
<td>47</td>
</tr>
<tr>
<td>fill.geno</td>
<td>48</td>
</tr>
<tr>
<td>find.flanking</td>
<td>49</td>
</tr>
<tr>
<td>find.marker</td>
<td>50</td>
</tr>
<tr>
<td>find.markerpos</td>
<td>51</td>
</tr>
<tr>
<td>find.pheno</td>
<td>51</td>
</tr>
<tr>
<td>find.pseudomarker</td>
<td>52</td>
</tr>
<tr>
<td>fitqtl</td>
<td>53</td>
</tr>
<tr>
<td>fitstahl</td>
<td>55</td>
</tr>
<tr>
<td>geno.image</td>
<td>57</td>
</tr>
<tr>
<td>geno.table</td>
<td>58</td>
</tr>
<tr>
<td>hyper</td>
<td>59</td>
</tr>
<tr>
<td>jittermap</td>
<td>60</td>
</tr>
<tr>
<td>listeria</td>
<td>61</td>
</tr>
<tr>
<td>lodint</td>
<td>62</td>
</tr>
<tr>
<td>makeqtl</td>
<td>63</td>
</tr>
<tr>
<td>map10</td>
<td>65</td>
</tr>
<tr>
<td>max.scanone</td>
<td>65</td>
</tr>
<tr>
<td>max.scantwo</td>
<td>67</td>
</tr>
<tr>
<td>movemarker</td>
<td>68</td>
</tr>
<tr>
<td>nmissing</td>
<td>69</td>
</tr>
<tr>
<td>plot.cross</td>
<td>70</td>
</tr>
<tr>
<td>plot.errorlod</td>
<td>71</td>
</tr>
<tr>
<td>plot.geno</td>
<td>72</td>
</tr>
<tr>
<td>plot.info</td>
<td>73</td>
</tr>
<tr>
<td>plot.map</td>
<td>75</td>
</tr>
<tr>
<td>plot.missing</td>
<td>76</td>
</tr>
<tr>
<td>plot.pheno</td>
<td>77</td>
</tr>
<tr>
<td>plot.pxg</td>
<td>78</td>
</tr>
<tr>
<td>plot.rf</td>
<td>80</td>
</tr>
<tr>
<td>plot.scanone</td>
<td>81</td>
</tr>
</tbody>
</table>
A starting point

Introductory comments on R/qtl

Description

A brief introduction to the R/qtl package, with a walk-through of an analysis.
New to R and/or R/qtl?

- In order to use the R/qtl package, you must type (within R) `library(qtl)`. You may wish to include this in a `.Rprofile` file.
- Documentation and several tutorials are available at the R archive (http://cran.r-project.org).
- Use the `help.start` function to start the html version of the R help.
- Type `library(help=qtl)` to get a list of the functions in R/qtl.
- Use the `example` function to run examples of the various functions in R/qtl.
- A tutorial on the use of R/qtl is distributed with the package and is also available at http://www.rqtl.org/rqtltutorial.pdf.
- Download the latest version of R/qtl from the R archive or from http://www.rqtl.org.

Walk-through of an analysis

Here we briefly describe the use of R/qtl to analyze an experimental cross. A more extensive tutorial on its use is distributed with the package and is also available at http://www.rqtl.org/rqtltutorial.pdf.

A difficult first step in the use of most data analysis software is the import of data. With R/qtl, one may import data in several different formats by use of the function `read.cross`. The internal data structure used by R/qtl is rather complicated, and is described in the help file for `read.cross`. We won’t discuss data import any further here, except to say that the comma-delimited format ("csv") is recommended. If you have trouble importing data, send an email to Karl Broman, (kbroman@biostat.wisc.edu), perhaps attaching examples of your data files. (Such data will be kept confidential.) Also see the sample data files and code at http://www.rqtl.org/sampledata.

We consider the example data `hyper`, an experiment on hypertension in the mouse, kindly provided by Bev Paigen and Gary Churchill. Use the `data` function to load the data.

```
data(hyper)
```

The `hyper` data set has class "cross". The function `summary.cross` gives summary information on the data, and checks the data for internal consistency. A number of other utility functions are available; hopefully these are self-explanatory.

```
summary(hyper)
nind(hyper)
nphe(hyper)
nchr(hyper)
nmar(hyper)
totmar(hyper)
```

The function `plot.cross` gives a graphical summary of the data; it calls `plot.missing` (to plot a matrix displaying missing genotypes) and `plot.map` (to plot the genetic maps), and also displays histograms or barplots of the phenotypes. The `plot.missing` function can plot individuals ordered by their phenotypes; you can see that for most markers, only individuals with extreme phenotypes were genotyped.

```
plot(hyper)
plot.missing(hyper)
```
plot.missing(hyper, reorder=TRUE)
plot.map(hyper)

Note that one marker (on chromosome 14) has no genotype data. The function drop.nullmarkers removes such markers from the data.

hyper <- drop.nullmarkers(hyper)
totmar(hyper)

The function est.rf estimates the recombination fraction between each pair of markers, and calculates a LOD score for the test of $r = 1/2$. This is useful for identifying markers that are placed on the wrong chromosome. Note that since, for these data, many markers were typed only on recombinant individuals, the pairwise recombination fractions show rather odd patterns.

hyper <- est.rf(hyper)
plot.rf(hyper)
plot.rf(hyper, chr=c(1,4))

To re-estimate the genetic map for an experimental cross, use the function est.map. The function plot.map, in addition to plotting a single map, can plot the comparison of two genetic maps (as long as they are composed of the same numbers of chromosomes and markers per chromosome). The function replace.map may be used to replace the genetic map in a cross with a new one.

newmap <- est.map(hyper, error.prob=0.01, verbose=TRUE)
plot.map(hyper, newmap)
hyper <- replace.map(hyper, newmap)

The function calc.errorlod may be used to assist in identifying possible genotyping errors; it calculates the error LOD scores described by Lincoln and Lander (1992). The calc.errorlod function return a modified version of the input cross, with error LOD scores included. The function top.errorlod prints the genotypes with values above a cutoff (by default, the cutoff is 4.0).

hyper <- calc.errorlod(hyper, error.prob=0.01)
top.errorlod(hyper)

The function plot.geno may be used to inspect the observed genotypes for a chromosome, with likely genotyping errors flagged.

plot.geno(hyper, chr=16, ind=c(24:34, 71:81))

Before doing QTL analyses, some intermediate calculations need to be performed. The function calc.genoprob calculates conditional genotype probabilities given the multipoint marker data. sim.geno simulates sequences of genotypes from their joint distribution, given the observed marker data.

As with calc.errorlod, these functions return a modified version of the input cross, with the intermediate calculations included. The step argument indicates the density of the grid on which the calculations will be performed, and determines the density at which LOD scores will be calculated.

hyper <- calc.genoprob(hyper, step=2.5, error.prob=0.01)
hyper <- sim.geno(hyper, step=2.5, n.draws=64, error.prob=0.01)

The function scanone performs a genome scan with a single QTL model. By default, it performs standard interval mapping (Lander and Botstein 1989): use of a normal model and the EM algorithm. If one specifies method="hk", Haley-Knott regression is performed (Haley and Knott 1992). These two methods require the results from calc.genoprob.
out.em <- scanone(hyper)
out.hk <- scanone(hyper, method="hk")

If one specifies method="imp", a genome scan is performed by the multiple imputation method of Sen and Churchill (2001). This method requires the results from sim.geno.
out.imp <- scanone(hyper, method="imp")

The output of scanone is a data.frame with class "scanone". The function plot.scanone may be used to plot the results, and may plot up to three sets of results against each other, as long as they conform appropriately.
plot(out.em)
plot(out.hk, col="blue", add=TRUE)
plot(out.imp, col="red", add=TRUE)

The function summary.scanone may be used to list information on the peak LOD for each chromosome for which the LOD exceeds a specified threshold.

summary(out.em)
summary(out.em, threshold=3)
summary(out.hk, threshold=3)
summary(out.imp, threshold=3)

The function max.scanone returns the maximum LOD score, genome-wide.

max(out.em)
max(out.hk)
max(out.imp)

One may also use scanone to perform a permutation test to get a genome-wide LOD significance threshold.
operm.hk <- scanone(hyper, method="hk", n.perm=1000)

The result has class "scanoneperm". The summary.scanoneperm function may be used to calculate LOD thresholds.

summary(operm.hk, alpha=0.05)

The permutation results may also be used in the summary.scanone function to calculate LOD thresholds and genome-scan-adjusted p-values.

summary(out.hk, perms=operm.hk, alpha=0.05, pvalues=TRUE)

We should say at this point that the function save.image will save your workspace to disk. You’ll wish you had used this if R crashes.

save.image()

The function scantwo performs a two-dimensional genome scan with a two-QTL model. Methods "em", "hk" and "imp" are all available. scantwo is considerably slower than scanone, and can require a great deal of memory. Thus, you may wish to re-run calc.genoprob and/or sim.geno with a more coarse grid.

hyper <- calc.genoprob(hyper, step=10, err=0.01)
hyper <- sim.geno(hyper, step=10, n.draws=64, err=0.01)

out2.hk <- scantwo(hyper, method="hk")
out2.em <- scantwo(hyper)
out2.imp <- scantwo(hyper, method="imp")

The output is an object with class scantwo. The function plot.scantwo may be used to plot
the results. The upper triangle contains LOD scores for tests of epistasis, while the lower triangle
contains LOD scores for the full model.

plot(out2.hk)
plot(out2.em)
plot(out2.imp)

The function summary.scantwo lists the interesting aspects of the output. For each pair of chro-
mosomes \((k, l)\), it calculates the maximum LOD score for the full model, \(M_f(k, l)\); a LOD score
indicating evidence for a second QTL, allowing for epistasis, \(M_{fv1}(k, l)\); a LOD score indicating
evidence for epistasis, \(M_i(k, l)\); the LOD score for the additive QTL model, \(M_a(k, l)\); and a LOD
score indicating evidence for a second QTL, assuming no epistasis, \(M_{av1}(k, l)\).

You must provide five LOD thresholds, corresponding to the above five LOD scores, and in that
order. A chromosome pair is printed if either (a) \(M_f(k, l) \geq T_f\) and \(M_{fv1}(k, l) \geq T_{fv1}\)
or \(M_i(k, l) \geq T_i\), or (b) \(M_a(k, l) \geq T_a\) and \(M_{av1}(k, l) \geq T_{av1}\).

summary(out2.em, thresholds=c(6.2, 5.0, 4.6, 4.5, 2.3))
summary(out2.em, thresholds=c(6.2, 5.0, Inf, 4.5, 2.3))

In the latter case, the interaction LOD score will be ignored.

The function max.scantwo returns the maximum joint and additive LODs for a two-dimensional
genome scan.

max(out2.em)

Permutation tests may also performed with scantwo; it may take a few days of CPU time. The
output is a list containing the maxima of the above five LOD scores for each of the imputations.

operm2 <- scantwo(hyper, method="hk", n.perm=100)
summary(operm2, alpha=0.05)

Citing R/qtl

To cite R/qtl in publications, use the Broman et al. (2003) reference listed below.

Author(s)

Karl W Broman, (kbroman@biostat.wisc.edu)

References

in line crosses using flanking markers. Heredity 69, 315–324.
using RFLP linkage maps. Genetics 121, 185–199.

---

### add.cim.covar

**Indicate marker covariates from composite interval mapping**

**Description**

Add dots at the locations of the selected marker covariates, for a plot of composite interval mapping results.

**Usage**

```r
add.cim.covar(cimresult, chr, gap=25, ...)
```

**Arguments**

- `cimresult`: Composite interval mapping results, as output from `cim`.
- `chr`: Optional vector specifying which chromosomes to plot. (The chromosomes must be specified by name.) This should be identical to that used in the call to `plot.scanone`.
- `gap`: Gap separating chromosomes (in cM). This should be identical to that used in the call to `plot.scanone`.
- `...`: Additional plot arguments, passed to the function `points`.

**Details**

One must first have used the function `plot.scanone` to plot the composite interval mapping results.

The arguments `chr` and `gap` must be identical to the values used in the call to `plot.scanone`. Dots indicating the locations of the selected marker covariates are displayed on the x-axis. (By default, solid red circles are plotted; this may be modified by specifying the graphics parameters `pch` and `col`.)

**Value**

A data frame indicating the marker covariates that were plotted.

**Author(s)**

Karl W Broman, ⟨kbroman@biostat.wisc.edu⟩

**See Also**

`cim`, `plot.scanone`
Examples

```r
## Not run: data(hyper)
hyper <- calc.genoprob(hyper, step=2.5)

out <- scanone(hyper)
out.cim <- cim(hyper, n.marcovar=3)
plot(out, out.cim, chr=c(1,4,6,15), col=c("blue", "red"))

add.cim.covar(out.cim, chr=c(1,4,6,15))## End(Not run)
```

---

**add.threshold**  
Add significance threshold to plot

---

**Description**

Add a significance threshold to a plot created by `plot.scanone`, using the permutation results.

**Usage**

```r
add.threshold(out, chr, perms, alpha=0.05, lodcolumn=1, gap=25, ...)
```

**Arguments**

- `out`  
  An object of class "scanone", as output by `scanone`. This must be identical to what was used in the call to `plot.scanone`.

- `chr`  
  Optional vector specifying which chromosomes to plot. If a selected subset of chromosomes were plotted, they must be specified here.

- `perms`  
  Permutation results from `scanone`, used to calculate the significance threshold.

- `alpha`  
  Significance level of the threshold.

- `lodcolumn`  
  An integer indicating which of column in the permutation results should be used.

- `gap`  
  Gap separating chromosomes (in cM). This must be identical to what was used in the call to `plot.scanone`.

- `...`  
  Passed to the function `abline` when it is called.

**Details**

This function allows you to add a horizontal line at the significance threshold to genome scan results plotted by `plot.scanone`.

The arguments `out`, `chr`, and `gap` must match what was used in the call to `plot.scanone`.

The argument `perms` must be specified. If X-chromosome-specific permutations were performed (via the argument `perm.Xsp` in the call to `scanone`), separate thresholds will be plotted for the autosomes and the X chromosome. These are calculated via the `summary.scanoneperm` function.
argmax.geno

Reconstruct underlying genotypes

Description

Uses the Viterbi algorithm to identify the most likely sequence of underlying genotypes, given the observed multipoint marker data, with possible allowance for genotyping errors.

Usage

argmax.geno(cross, step=0, off.end=0, error.prob=0.0001, map.function=c("haldane","kosambi","c-f","morgan"), stepwidth=c("fixed", "variable"))

Arguments

cross An object of class cross. See read.cross for details.
step Maximum distance (in cM) between positions at which the genotypes are reconstructed, though for step=0, genotypes are reconstructed only at the marker locations.
off.end Distance (in cM) past the terminal markers on each chromosome to which the genotype reconstructions will be carried.
error.prob Assumed genotyping error rate used in the calculation of the penetrance Pr(observed genotype | true genotype).
argmax.geno

map.function Indicates whether to use the Haldane, Kosambi, Carter-Falconer or Morgan map function when converting genetic distances into recombination fractions.

stepwidth Indicates whether the intermediate points should have fixed or variable step sizes. We strongly recommend using "fixed"; "variable" is included only for the qtlbim package (http://www.ssg.uab.edu/qtlbim).

Details

We use the Viterbi algorithm to calculate \( \arg\max_v \Pr(g = v | O) \) where \( g \) is the underlying sequence of genotypes and \( O \) is the observed marker genotypes.

This is done by calculating \( \gamma_k(v_k) = \max_{v_1, \ldots, v_{k-1}} \Pr(g_1 = v_1, \ldots, g_k = v_k, O_1, \ldots, O_k) \) for \( k = 1, \ldots, n \) and then tracing back through the sequence.

Value

The input cross object is returned with a component, argmax, added to each component of cross$geno. The argmax component is a matrix of size [n.ind x n.pos], where n.pos is the number of positions at which the reconstructed genotypes were obtained, containing the most likely sequences of underlying genotypes. Attributes "error.prob", "step", and "off.end" are set to the values of the corresponding arguments, for later reference.

Warning

The Viterbi algorithm can behave badly when step is small but positive. One may observe quite different results for different values of step.

The problem is that, in the presence of data like \( A----H \), the sequences \( AAAAAA \) and \( HHHHHH \) may be more likely than any one of the sequences \( AAAAAH, AAAAHH, AAAHHA, AAHHHH, AHHHHH, HHHAAA \). The Viterbi algorithm produces a single "most likely" sequence of underlying genotypes.

Author(s)

Karl W Broman, (kbroman@biostat.wisc.edu)

References


See Also

sim.geno, calc.genoprob, fill.geno

Examples

data(fake.f2)
fake.f2 <- argmax.geno(fake.f2, step=2, off.end=5, err=0.01)
Arithmetic Operators for scanone and scantwo results

Description
Add or subtract LOD scores in results from scanone or scantwo.

Usage
scan1+scan2
scan1-scan2

Arguments
scan1, scan2 Genome scan results on the same set of chromosomes and markers, as output by scanone or scantwo.

Details
This is used to calculate the sum or difference of LOD scores of two genome scan results. It is particularly useful for calculating the LOD scores for QTL-by-covariate interactions (see the example, below). Note that the degrees of freedom are also added or subtracted.

Value
The same type of data structure as the input objects, with LOD scores added or subtracted.

Author(s)
Karl W Broman, (kbroman@biostat.wisc.edu)

Examples
data(fake.bc)
fake.bc <- calc.genoprob(fake.bc, step=2.5)

# covariates
ac <- fake.bc$pheno[,c("sex","age")]
ic <- fake.bc$pheno[,"sex"]

# scan with additive but not the interactive covariate
out.acovar <- scanone(fake.bc, addcovar=ac)

# scan with interactive covariate
out.icovar <- scanone(fake.bc, addcovar=ac, intcovar=ic)

# plot the difference of with and without the interactive covariate
# This is a LOD score for a test of QTL x covariate interaction
plot(out.icovar-out.acovar)
**arithscanperm**  
*Arithmetic Operators for permutation results*

**Description**
Add or subtract LOD scores in permutation results from `scanone` or `scantwo`.

**Usage**

```r
perm1+perm2
perm1-perm2
```

**Arguments**

`perm1`, `perm2` Permutation results from `scanone` or `scantwo`, on the same set of chromosomes and markers.

**Details**
This is used to calculate the sum or difference of LOD scores of two sets of permutation results from `scanone` or `scantwo`. One must be careful to ensure that the permutations are perfectly linked, which will require the use of `set.seed`.

**Value**
The same data structure as the input objects, with LOD scores added or subtracted.

**Author(s)**
Karl W Broman, ⟨kbroman@biostat.wisc.edu⟩

**Examples**

```r
data(fake.bc)

fake.bc <- calc.genoprob(fake.bc, step=2.5)

# covariates
ac <- fake.bc$pheno[,c("sex","age")]
ic <- fake.bc$pheno[,"sex"]

# set seed
theseed <- round(runif(1, 1, 10^8))
set.seed(theseed)

# permutations with additive but not the interactive covariate
## Not run:
operm.acovar <- scanone(fake.bc, addcovar=ac, n.perm=1000)
## End(Not run)
```
# re-set the seed
set.seed(theseed)

# permutations with interactive covariate
## Not run:
operm.icovar <- scanone(fake.bc, addcovar=ac, intcovar=ic,
                       n.perm=1000)
## End(Not run)

# permutation results for the QTL x covariate interaction
operm.gxc <- operm.icovar - operm.acovar

# LOD thresholds
summary(operm.gxc)

---

badorder  An intercross with misplaced markers

Description
Simulated data for an intercross with some markers out of order.

Usage
data(badorder)

Format
An object of class cross. See read.cross for details.

Details
There are 250 F2 individuals typed at a total of 36 markers on four chromosomes. The data were simulated with QTLs at the center of chromosomes 1 and 3.

The order of several markers on chromosome 1 is incorrect. Markers on chromosomes 2 and 3 are switched.

Author(s)
Karl W Broman, ⟨kbroman@biostat.wisc.edu⟩

See Also
est.rf, ripple, est.map, sim.cross
bayesint

Examples

data(badorder)

# estimate recombination fractions
badorder <- est.rf(badorder)
plot.rf(badorder)

# re-estimate map
newmap <- est.map(badorder)
plot.map(badorder, newmap)

# assess marker order on chr 1
rip3 <- ripple(badorder, chr=1, window=3)
supply(rip3)

---

bayesint  \textit{Bayesian credible interval}

Description

Calculate an approximate Bayesian credible interval for a particular chromosome, using output from \texttt{scanone}.

Usage

\begin{verbatim}
bayesint(results, chr, prob=0.95, lodcolumn=1)
\end{verbatim}

Arguments

- \texttt{results} \hspace{1cm} Output from \texttt{scanone}.
- \texttt{chr} \hspace{1cm} A chromosome ID.
- \texttt{prob} \hspace{1cm} Probability coverage of the interval.
- \texttt{lodcolumn} \hspace{1cm} An integer, or vector of 3 integers, indicating which of the LOD score columns should be plotted (generally this is 1).

Details

We take $10^{LOD}$, rescale it to have area 1, and then calculate the connected interval with density above some threshold and having coverage matching the target probability.

Value

An object of class \texttt{scanone}, like the input, indicating the position with the maximum LOD, and indicating approximate endpoints for the Bayesian credible interval.

Author(s)

Karl W Broman, (kbroman@biostat.wisc.edu)
**bristle3**

*Data on bristle number in Drosophila*

**Description**

Data from bristle number in chromosome 3 recombinant isogenic lines of *Drosophila melanogaster*.

**Usage**

```r
data(bristle3)
```

**Format**

An object of class `cross`. See `read.cross` for details.

**Details**

There are 66 chromosome 3 recombinant isogenic lines, derived from inbred lines that were selected for low (A) and high (B) abdominal bristle numbers. A recombinant chromosome 3 was placed in an isogenic low background.

There are eight phenotypes: the average and SD of the number of abdominal and sternopleural bristles in males and females for each line.

Each line is typed at 29 genetic markers on chromosome 3.

**References**


**See Also**

`bristleX, listeria, fake.bc, fake.f2, fake.4way, hyper`

---

**See Also**

- `scanone, lodint`

**Examples**

```r
data(hyper)

hyper <- calc.genoprob(hyper, step=0.5)
out <- scanone(hyper, method="hk")
bayesint(out, chr=1)
bayesint(out, chr=4)
bayesint(out, chr=4, prob=0.99)
```
Examples

data(bristle3)
# Summaries
summary(bristle3)
plot(bristle3)

# genome scan for each of the average phenotypes
bristle3 <- calc.genoprob(bristle3, step=2)
out <- scanone(bristle3, pheno.col=c(1,3,5,7))

# Plot the results
  # maximum LOD score among four phenotypes
ym <- max(apply(out[,-(1:2)], 2, max))
plot(out, lod=1:3, ylim=c(0,ym))
plot(out, lod=4, add=TRUE, col="green")

bristleX

Data on bristle number in Drosophila

Description

Data from bristle number in chromosome X recombinant isogenic lines of *Drosophila melanogaster*.

Usage

data(bristleX)

Format

An object of class cross. See read.cross for details.

Details

There are 92 chromosome X recombinant isogenic lines, derived from inbred lines that were selected for low (A) and high (B) abdominal bristle numbers. A recombinant chromosome X was placed in an isogenic low background.

There are eight phenotypes: the average and SD of the number of abdominal and sternopleural bristles in males and females for each line.

Each line is typed at 17 genetic markers on chromosome 3.

References

See Also  

bristleX, listeria, fake.bc, fake.f2, fake.4way, hyper

Examples

data(bristleX)
# Summaries
summary(bristleX)
plot(bristleX)

# genome scan for each of the average phenotypes
bristleX <- calc.genoprob(bristleX, step=2)
out <- scanone(bristleX, pheno.col=c(1,3,5,7))

# Plot the results
  # maximum LOD score among four phenotypes
ym <- max(apply(out[,-(1:2)], 2, max))
plot(out, lod=1:3, ylim=c(0,ym))
plot(out, lod=4, add=TRUE, col="green")

---

**c.cross**  
*Combine data for QTL experiments*

Description

Concatenate the data for multiple QTL experiments.

Usage

```r
## S3 method for class 'cross':
c(...)
```

Arguments

...  
A set of objects of class `cross`. See `read.cross` for details. These must all either be of the same cross type or be a combination of backcrosses and intercrosses. All crosses must have the same number of chromosomes and chromosome names, and the same marker orders and positions, though the set of markers need not be precisely the same.

Value

The concatenated input, as a `cross` object. Additional columns are added to the phenotype data indicating which cross an individual comes from; another column indicates cross type (0=BC, 1=intercross), if there are crosses of different types. The crosses are not required to have exactly the same set of phenotypes; phenotypes with the same names are assumed to be the same.

If the crosses have different sets of markers, we interpolate marker order, but the cM positions of markers that are in common between crosses must be precisely the same in the different crosses.
c.scanone

Author(s)
Karl W Broman, (kbroman@biostat.wisc.edu)

See Also
subset.cross

Examples

data(fake.f2)
junk <- fake.f2
junk <- c(fake.f2,junk)

Description
Concatenate the columns from different runs of scanone.

Usage

## S3 method for class 'scanone':
c(..., labels)

Arguments

... A set of objects of class scanone. These are the results from scanone (with
n.perm=0), generally run with different phenotypes or methods. All must
conform with each other, meaning that calc.genoprob and/or sim.geno
were run with the same values for step and off.end and with data having
the same genetic map.

labels A vector of character strings, of length 1 or of the same length as the input, to
be appended to the column names in the output.

Details
The aim of this function is to concatenate the results from multiple runs scanone, generally for
different phenotypes and/or methods, to be used in parallel with summary.scanone.

Value
The concatenated input, as a scanone object.

Author(s)
Karl W Broman, (kbroman@biostat.wisc.edu)
See Also

`summary.scanone, scanone, cbind.scanoneperm`

Examples

```r
data(fake.f2)
fake.f2 <- calc.genoprob(fake.f2)

out.hk <- scanone(fake.f2, method="hk")
out.np <- scanone(fake.f2, model="np")

out <- c(out.hk, out.np, labels=c("hk","np"))
plot(out, lod=1:2, col=c("blue", "red"))
```

c.scanoneperm

**Combine data from scanone permutations**

Description

Concatenate the data for multiple runs of `scanone` with `n.perm > 0`.

Usage

```r
## S3 method for class 'scanoneperm':
c(...)
## S3 method for class 'scanoneperm':
rbind(...)
```

Arguments

...  

A set of objects of class `scanoneperm`. These are the permutation results from `scanone` (that is, when `n.perm > 0`). These must all have the same number of columns. (That is, they must have been created with the same number of phenotypes, and it is assumed that they were generated in precisely the same way.)

Details

The aim of this function is to concatenate the results from multiple runs of a permutation test `scanone`, to assist with the case that such permutations are done on multiple processors in parallel.

Value

The concatenated input, as a `scanoneperm` object.

Author(s)

Karl W Broman, (kbroman@biostat.wisc.edu)
c.scantwoperm

See Also

summary.scanoneperm, scanone, cbind.scanoneperm, c.scantwoperm

Examples

data(fake.f2)

fake.f2 <- calc.genoprob(fake.f2)
operm1 <- scanone(fake.f2, method="hk", n.perm=100, perm.Xsp=TRUE)
operm2 <- scanone(fake.f2, method="hk", n.perm=50, perm.Xsp=TRUE)

operm <- c(operm1, operm2)

c.scantwoperm Combine data from scantwo permutations

Description

Concatenate the data for multiple runs of scantwo with n.perm > 0.

Usage

## S3 method for class 'scantwoperm':
c(...)
## S3 method for class 'scantwoperm':
rbind(...)

Arguments

... A set of objects of class scantwoperm. These are the permutation results from scantwo (that is, when n.perm > 0). These must all concern the same number of LOD columns. (That is, they must have been created with the same number of phenotypes, and it is assumed that they were generated in precisely the same way.)

Details

The aim of this function is to concatenate the results from multiple runs of a permutation test scantwo, to assist with the case that such permutations are done on multiple processors in parallel.

Value

The concatenated input, as a scantwoperm object.

Author(s)

Karl W Broman, (kbroman@biostat.wisc.edu)
calc.errorlod

See Also

summary.scantwoperm, scantwo, summary.scantwo, c.scanoneperm

Examples

data(fake.f2)
fake.f2 <- calc.genoprob(fake.f2)
operm1 <- scantwo(fake.f2, method="hk", n.perm=50)
operm2 <- scantwo(fake.f2, method="hk", n.perm=50)
operm <- c(operm1, operm2)

calc.errorlod Identify likely genotyping errors

Description

Calculates a LOD score for each genotype, measuring the evidence for genotyping errors.

Usage

calc.errorlod(cross, error.prob=0.01,
            map.function=c("haldane","kosambi","c-f","morgan"),
            version=c("new","old"))

Arguments

cross An object of class cross. See read.cross for details.
error.prob Assumed genotyping error rate used in the calculation of the penetrance Pr(observed
genotype | true genotype)
map.function Indicates whether to use the Haldane, Kosambi, Carter-Falconer, or Morgan map
function when converting genetic distances into recombination fractions.
version Specifies whether to use the original version of this function or the current (pre-
ferred) version.

Details

Calculates, for each individual at each marker, a LOD score measuring the strength of evidence for
a genotyping error, as described by Lincoln and Lander (1992).

In the latest version, evidence for a genotype being in error is considered assuming that all other
genotypes (for that individual, on that chromosome) are correct. The argument version allows
one to specify whether this new version is used, or whether the original (old) version of the calcu-
ation is performed.

Note that values below 4 are generally not interesting. Also note that if markers are extremely
tightly linked, recombination events can give large error LOD scores. The error LOD scores should
calc.genoprob

not be trusted blindly, but should be viewed as a tool for identifying genotypes deserving further study.

Use top.errorlod to print all genotypes with error LOD scores above a specified threshold, plot.errorlod to plot the error LOD scores for specified chromosomes, and plot.geno to view the observed genotype data with likely errors flagged.

Value

The input cross object is returned with a component, errorlod, added to each component of cross$geno. The errorlod component is a matrix of size (n.ind x n.mar). An attribute "error.prob" is set to the value of the corresponding argument, for later reference.

Author(s)

Karl W Broman, (kbroman@biostat.wisc.edu)

References


See Also

plot.errorlod, top.errorlod

Examples

data(hyper)

hyper <- calc.errorlod(hyper, error.prob=0.01)

# print those above a specified cutoff
top.errorlod(hyper, cutoff=4)

# plot genotype data, flagging genotypes with error LOD > cutoff
plot.geno(hyper, chr=1, ind=160:200, cutoff=7, min.sep=2)

calc.genoprob  Calculate conditional genotype probabilities

Description

Uses the hidden Markov model technology to calculate the probabilities of the true underlying genotypes given the observed multipoint marker data, with possible allowance for genotyping errors.

Usage

calc.genoprob(cross, step=0, off.end=0, error.prob=0.0001, map.function=c("haldane","kosambi","c-f","morgan"), stepwidth=c("fixed", "variable"))
Arguments

- **cross**: An object of class `cross`. See `read.cross` for details.
- **step**: Maximum distance (in cM) between positions at which the genotype probabilities are calculated, though for `step = 0`, probabilities are calculated only at the marker locations.
- **off.end**: Distance (in cM) past the terminal markers on each chromosome to which the genotype probability calculations will be carried.
- **error.prob**: Assumed genotyping error rate used in the calculation of the penetrance `Pr(\text{observed genotype} \mid \text{true genotype})`.
- **map.function**: Indicates whether to use the Haldane, Kosambi or Carter-Falconer map function when converting genetic distances into recombination fractions.
- **stepwidth**: Indicates whether the intermediate points should have fixed or variable step sizes. We strongly recommend using "fixed"; "variable" is included only for the qtlbim package (`http://www.ssg.uab.edu/qtlbim`).

Details

Let $O_k$ denote the observed marker genotype at position $k$, and $g_k$ denote the corresponding true underlying genotype.

We use the forward-backward equations to calculate $\alpha_{kv} = \log Pr(O_1,\ldots,O_k,g_k=v)$ and $\beta_{kv} = \log Pr(O_{k+1},\ldots,O_n \mid g_k = v)$

We then obtain $Pr(g_k \mid O_1,\ldots,O_n) = \exp(\alpha_{kv} + \beta_{kv})/s$ where $s = \sum_v \exp(\alpha_{kv} + \beta_{kv})$

In the case of the 4-way cross, with a sex-specific map, we assume a constant ratio of female:male recombination rates within the inter-marker intervals.

Value

The input `cross` object is returned with a component, `prob`, added to each component of `cross$geno`. `prob` is an array of size `[n.ind x n.pos x n.gen]` where `n.pos` is the number of positions at which the probabilities were calculated and `n.gen` = 3 for an intercross, = 2 for a backcross, and = 4 for a 4-way cross. Attributes "error.prob", "step", "off.end", and "map.function" are set to the values of the corresponding arguments, for later reference (especially by the function `calc.errorlod`).

Author(s)

Karl W Broman, (kbroman@biostat.wisc.edu)

References


See Also

- `sim.geno`, `argmax.geno`, `calc.errorlod`
Examples

```r
data(fake.f2)
fake.f2 <- calc.genoprob(fake.f2, step=2, off.end=5)

data(fake.bc)
fake.bc <- calc.genoprob(fake.bc, step=0, off.end=0, err=0.01)
```

**cbind.scanoneperm**  \hspace{1em} *Combine columns from multiple scanone permutation results*

**Description**

Concatenate the columns from different runs of `scanone` with `n.perm > 0`.

**Usage**

```r
## S3 method for class 'scanoneperm':
cbind(..., labels)
```

**Arguments**

- `...`  
  A set of objects of class `scanoneperm`. These are the permutation results from `scanone` (that is, when `n.perm > 0`), generally run with different phenotypes or methods.

- `labels`  
  A vector of character strings, of length 1 or of the same length as the input `...`, to be appended to the column names in the output.

**Details**

The aim of this function is to concatenate the results from multiple runs of a permutation test `scanone`, generally for different phenotypes and/or methods, to be used in parallel with `c.scanone`.

**Value**

The concatenated input, as a `scanoneperm` object. If different numbers of permutation replicates were used, those columns with fewer replicates are padded with missing values (`NA`).

**Author(s)**

Karl W Broman, (kbroman@biostat.wisc.edu)

**See Also**

`summary.scanoneperm, scanone, c.scanoneperm, c.scanone`
Examples

data(fake.f2)
fake.f2 <- calc.genoprob(fake.f2)

operml <- scanone(fake.f2, method="hk", n.perm=10, perm.Xsp=TRUE)
operm2 <- scanone(fake.f2, method="em", n.perm=5, perm.Xsp=TRUE)

operm <- cbind(operml, operm2, labels=c("hk","em"))
summary(operm)

checkAlleles Identify markers with switched alleles

Description

Identify markers whose alleles might have been switched by comparing the LOD score for linkage
to all other autosomal markers with the original data to that when the alleles have been switched.

Usage

checkAlleles(cross, threshold=3, verbose)

Arguments

cross An object of class cross. See read.cross for details.

threshold Only an increase in maximum 2-point LOD of at least this amount will lead to
a marker being flagged.

verbose If TRUE and there are no markers above the threshold, print a message.

Details

For each marker, we compare the maximum LOD score for the cases where the estimated recombi-
nation fraction > 0.5 to those where r.f. < 0.5. The function est.rf must first be run.

Note: Markers that are tightly linked to a marker whose alleles are switched are likely to also be
flagged by this method. The real problem markers are likely those with the biggest difference in
LOD scores.

Value

A data frame containing the flagged markers, having four columns: the marker name, chromosome
ID, numeric index within chromosome, and the difference between the maximum two-point LOD
score with the alleles switched to that from the original data.

Author(s)

Karl W Broman, (kbroman@biostat.wisc.edu)
See Also

est.rf, geno.table

Examples

data(fake.f2)

# switch homozygotes at marker D5M391
g <- fake.f2$geno[[5]]$data[,3]
g <- 4 - g
fake.f2$geno[[5]]$data[,3] <- g

fake.f2 <- est.rf(fake.f2)
checkAlleles(fake.f2)

---

chrlen

Chromosome lengths in QTL experiment

Description

Obtain the chromosome lengths in a cross or map object.

Usage

chrlen(object)

Arguments

object An object of class map or of class cross.

Value

Returns a vector of chromosome lengths. If the cross has sex-specific maps, it returns a 2-row matrix with the two lengths for each chromosome.

Author(s)

Karl W Broman, (kbroman@biostat.wisc.edu)

See Also

summary.map, pull.map, summary.cross

Examples

data(fake.f2)
chrlen(fake.f2)

map <- pull.map(fake.f2)
chrlen(map)
**Description**

Composite interval mapping by a scheme from QTL Cartographer: forward selection at the markers (here, with filled-in genotype data) to a fixed number, followed by interval mapping with the selected markers as covariates, dropping marker covariates if they are within some fixed window size of the location under test.

**Usage**

```r
cim(cross, pheno.col=1, n.marcovar=3, window=10, 
method=c("em", "imp", "hk", "ehk"),
imp.method=c("imp", "argmax"), error.prob=0.0001, 
map.function=c("haldane", "kosambi", "c-v", "morgan"), 
n.perm)
```

**Arguments**

- `cross` An object of class `cross`. See `read.cross` for details.
- `pheno.col` Column number in the phenotype matrix which should be used as the phenotype.
- `n.marcovar` Number of marker covariates to use.
- `window` Window size, in cM.
- `method` Indicates whether to use the EM algorithm, imputation, Haley-Knott regression, or the extended Haley-Knott method.
- `imp.method` Method used to impute any missing marker genotype data.
- `error.prob` Genotyping error probability assumed when imputing the missing marker genotype data.
- `map.function` Map function used when imputing the missing marker genotype data.
- `n.perm` If specified, a permutation test is performed rather than an analysis of the observed data. This argument defines the number of permutation replicates.

**Details**

We first use `fill.geno` to impute any missing marker genotype data, either via a simple random imputation or using the Viterbi algorithm.

We then perform forward selection to a fixed number of markers. These will be used (again, with any missing data filled in) as covariates in the subsequent genome scan.
Value

The function returns an object of the same form as the function scanone:

If \( n_{\text{perm}} \) is missing, the function returns the scan results as a data.frame with three columns: chromosome, position, LOD score. Attributes indicate the names and positions of the chosen marker covariates.

If \( n_{\text{perm}} > 0 \), the function results the results of a permutation test: a vector giving the genome-wide maximum LOD score in each of the permutations.

Author(s)

Karl W Broman, (kbroman@biostat.wisc.edu)

References


See Also

add.cim.covar, scanone, summary.scanone, plot.scanone, fill.geno

Examples

data(hyper)
hyper <- calc.genoprob(hyper, step=2.5)

out <- scanone(hyper)
out.cim <- cim(hyper, n.marcovar=3)
plot(out, out.cim, chr=c(1,4,6,15), col=c("blue", "red"))

add.cim.covar(out.cim, chr=c(1,4,6,15))

---

**clean.cross**

Remove derived data

Description

Remove any intermediate calculations from a cross object.
Usage

```r
## S3 method for class 'cross':
clean(object)
```

Arguments

- `object`: An object of class `cross`. See `read.cross` for details.

Value

The input object, with any intermediate calculations (such as is produced by `calc.genoprob`, `argmax.geno` and `sim.geno`) removed.

Author(s)

Karl W Broman, ⟨kbroman@biostat.wisc.edu⟩

See Also

drop.nullmarkers, drop.markers, clean.scantwo

Examples

```r
data(fake.f2)
names(fake.f2$geno)
fake.f2 <- calc.genoprob(fake.f2)
names(fake.f2$geno)
fake.f2 <- clean(fake.f2)
names(fake.f2$geno)
```

---

**clean.scantwo**

Clean up scantwo output

Description

In an object output from `scantwo`, replaces negative and missing LOD scores with 0, and replaces LOD scores for pairs of positions that are between markers with 0. Further, if the LOD for full model is less than the LOD for the additive model, the additive LOD is pasted over the full LOD.

Usage

```r
## S3 method for class 'scantwo':
clean(object)
```

Arguments

- `object`: An object of class `scantwo`. See `scantwo` for details.
Value

The input scantwo object, with any negative or missing LOD scores replaced by 0, and LOD scores for positions between markers by 0. Also, if the LOD for the full model is less than the LOD for the additive model, the additive LOD is used in place of the full LOD.

Author(s)

Karl W Broman, ⟨kbroman@biostat.wisc.edu⟩

See Also

scantwo, summary.scantwo

Examples

data(fake.f2)

fake.f2 <- calc.genoprob(fake.f2, step=5)
out2 <- scantwo(fake.f2, method="hk")
out2 <- clean(out2)

comparecrosses  

Compare two cross objects.

Description

Verify that two objects of class cross have identical classes, chromosomes, markers, genotypes, genetic maps, and phenotypes.

Usage

comparecrosses(cross1, cross2, tol=1e-5)

Arguments

cross1    An object of class cross (must be an intercross). See read.cross for details.
cross2    An object of class cross (must be an intercross). See read.cross for details.
tol       Tolerance value for comparing genetic map positions and numeric phenotypes.

Value

None.

Author(s)

Karl W Broman, ⟨kbroman@biostat.wisc.edu⟩
Compare individuals’ genotype data

Description
Count proportion of matching genotypes between all pairs of individuals, to look for unusually closely related individuals.

Usage
comparegeno(cross, what=c("proportion","number"))

Arguments
cross An object of class cross. See read.cross for details.
what Indicates whether to return the proportion or number of matching genotypes.

Value
A matrix whose (i,j)th element is the proportion or number of matching genotypes for individuals i and j.

Author(s)
Karl W Broman, (kbroman@biostat.wisc.edu)

See Also
nmissing

Examples
data(listeria)
output <- comparegeno(listeria)

# image of the proportions
n.ind <- nind(listeria)
image(1:n.ind, 1:n.ind, output, col=gray((0:99)/99),
     breaks=seq(0,1,len=101))
condense.scantwo

condense.scantwo Condense the output from a 2-d genome scan

Description

Produces a very condensed version of the output of scantwo.

Usage

## S3 method for class 'scantwo':
condense(object)

Arguments

object An object of class scantwo, the output of the function scantwo.

Details

This produces a very reduced version of the output of scantwo, for which a summary may still be created via summary.scantwo, though plots can no longer be made.

Value

An object of class scantwocondensed, containing just the maximum full, additive and interactive LOD scores, and the positions where they occurred, on each pair of chromosomes.

Author(s)

Karl W Broman, (kbroman@biostat.wisc.edu)

See Also

scantwo, summary.scantwo, max.scantwo
**Examples**

```r
data(fake.f2)
fake.f2 <- calc.genoprob(fake.f2)
out2 <- scantwo(fake.f2, method="hk")
out2c <- condense(out2)
summary(out2c, allpairs=FALSE)
max(out2c)
```

---

**convert.scanone**  
*Convert output from scanone for R/qtl version 0.98*

**Description**

Convert the output from scanone from the format used in R/qtl version 0.97 and earlier to that used in version 0.98 and later.

**Usage**

```r
## S3 method for class 'scanone':
convert(object)
```

**Arguments**

- `object`  
  Output from the function `scanone`, for R/qtl version 0.97 and earlier.

**Details**

Previously, inter-marker locations were named as, for example, `loc7.5.c3`; these were changed to `c3.loc7.5`.

**Value**

The same scanone output, but revised for use with R/qtl version 0.98 and later.

**Author(s)**

Karl W Broman, ⟨kbroman@biostat.wisc.edu⟩

**See Also**

- `scanone`, `convert.scantwo`

**Examples**

```r
## Not run: out.new <- convert(out.old)
```
**convert.scantwo**

*Convert output from scantwo for R/qtl version 1.03 and earlier*

**Description**

Convert the output from scantwo from the format used in R/qtl version 1.03 and earlier to that used in version 1.04 and later.

**Usage**

```r
## S3 method for class 'scantwo':
convert(object)
```

**Arguments**

- `object` Output from the function `scantwo`, for R/qtl version 1.03 and earlier.

**Details**

Previously, the output from `scantwo` contained the full and interaction LOD scores. In R/qtl version 1.04 and later, the output contains the LOD scores from the full and additive QTL models.

**Value**

The same scanone output, but revised for use with R/qtl version 1.03 and later.

**Author(s)**

Karl W Broman, (kbroman@biostat.wisc.edu)

**See Also**

`scantwo`, `convert.scanone`

**Examples**

```r
## Not run: out2.new <- convert(out2.old)
```
drop.markers  

Drop a set of markers

Description
Drop a vector of markers from the data matrices and genetic maps.

Usage

```r
drop.markers(cross, markers)
```

Arguments

- `cross` An object of class `cross`. See `read.cross` for details.
- `markers` A character vector of marker names.

Value

The input object, with any markers in the vector `markers` removed from the genotype data matrices, genetic maps, and, if applicable, any derived data (such as produced by `calc.genoprob`). (It might be a good idea to re-derive such things after using this function.)

Author(s)
Karl W Broman, (kbroman@biostat.wisc.edu)

See Also

- `drop.nullmarkers`, `geno.table`, `clean.cross`

Examples

```r
data(listeria)
listeria2 <- drop.markers(listeria, c("D10M44","D1M3","D1M75"))
```

---

drop.nullmarkers  

Drop markers without any genotype data

Description
Drop markers, from the data matrices and genetic maps, that have no genotype data.

Usage

```r
drop.nullmarkers(cross)
```
effectplot

Arguments

cross An object of class cross. See read.cross for details.

Value

The input object, with any markers lacking genotype data removed from the genotype data matrices, genetic maps, and, if applicable, any derived data (such as produced by calc.genoprob). (It might be a good idea to re-derive such things after using this function.)

Author(s)

Karl W Broman, ⟨kbroman@biostat.wisc.edu⟩

See Also

drop.markers, clean.cross, geno.table

Examples

# removes one marker from hyper
data(hyper)
hyper <- drop.nullmarkers(hyper)

# shouldn't do anything to listeria
data(listeria)
listeria <- drop.nullmarkers(listeria)

________________________

| effectplot |
| Plot phenotype means against genotypes at one or two markers. |

Description

Plot the phenotype means for each group defined by the genotypes at one or two markers (or the values at a discrete covariate).

Usage

effectplot(cross, pheno.col=1, mname1, mark1, geno1, mname2, mark2, geno2, main, ylim, xlab, ylab, col, add.legend=TRUE, legend.lab, draw=TRUE, var.flag=c("pooled","group"))

Arguments

cross An object of class cross.
pheno.col Column number in the phenotype matrix to be drawn in the plot.
mname1 Name for the first marker or pseudomarker.
mark1 Genotype data for the first marker. If unspecified, genotypes will be taken from the data in the input cross object, using the name specified in mname1.

gen1 Optional labels for the genotypes (or classes in a covariate).

mname2 Name for the second marker or pseudomarker (optional).

mark2 Like mark1 (optional).

geno2 Optional labels for the genotypes (or classes in a covariate).

main Optional figure title.

ylim Optional y-axis limits.
xlab Optional x-axis label.
ylab Optional y-axis label.
col Optional vector of colors for the different line segments.

add.legend A logical value to indicate whether to add a legend.

legend.lab Optional title for the legend.

draw A logical value to indicate generate the plot or not. If FALSE, no figure will be plotted and this function can be used to calculate the group means and standard errors.

var.flag The method to calculate the group variance. "pooled" means to use the pooled variance and "group" means to calculate from individual group.

Details

In the plot, the y-axis is the phenotype. In the case of one marker, the x-axis is the genotype for that marker. In the case of two markers, the x-axis is for different genotypes of the second marker, and the genotypes of first marker are represented by lines in different colors. Error bars are plotted at ± 1 SE.

The results of sim.geno are used; if they are not available, sim.geno is run with n.draws=16. The average phenotype for each genotype group takes account of missing genotype data by averaging across the imputations. The SEs take account of both the residual phenotype variation and the imputation error.

Value

A data.frame containing the phenotype means and standard errors for each group.

Author(s)

Hao Wu; Karl W Broman, ⟨kbroman@biostat.wisc.edu⟩

See Also

plot.pxg, find.marker, effectscan, find.pseudomarker
Examples

data(fake.f2)

# impute genotype data
## Not run:
fake.f2 <- sim.geno(fake.f2, step=5, n.draws=64)
## End(Not run)

########################################
# one marker plots
########################################
### plot of genotype-specific phenotype means for 1 marker
mname <- find.marker(fake.f2, 1, 37) # marker D1M437
effectplot(fake.f2, pheno.col=1, mname1=mname)

### plot a phenotype
# Plot of sex-specific phenotype means,
# note that "sex" must be a phenotype name here
effectplot(fake.f2, mname1="sex", genol=c("F","M"))
# alternatively:
sex <- fake.f2$pheno[,2]
effectplot(fake.f2, mname1="Sex", mark1=sex, genol=c("F","M"))

########################################
# two markers plots
########################################
### plot two markers
# plot of genotype-specific phenotype means for 2 markers
mname1 <- find.marker(fake.f2, 1, 37) # marker D1M437
mname2 <- find.marker(fake.f2, 13, 24) # marker D13M254
effectplot(fake.f2, mname1=mname1, mname2=mname2)

### plot two pseudomarkers
pmnames <- find.pseudomarker(fake.f2, chr=c(1, 13), c(35, 25))
effectplot(fake.f2, mname1=pmnames[1], mname2=pmnames[2])

### Plot of sex- and genotype-specific phenotype means
mname <- find.marker(fake.f2, 13, 24) # marker D13M254
# sex and a marker
effectplot(fake.f2, mname1=mname, mname2="Sex",
          mark2=fake.f2$pheno$sex, genol=c("F","M"))
# Same as above, switch role of sex and the marker
# sex and marker
effectplot(fake.f2, mname1="Sex", mark1=fake.f2$pheno$sex,
          genol=c("F","M"), mname2=mname)

# X chromosome marker
mname <- find.marker(fake.f2, "X", 14) # marker DXM66
effectplot(fake.f2, mname=mname)
# Two markers, including one on the X
mnames <- find.marker(fake.f2, c(13, "X"), c(24, 14))
effectplot(fake.f2, mname1=mnames[1], mname2=mnames[2])

---

**effectscan**  
*Plot estimated QTL effects across the whole genome*

**Description**

This function is used to plot the estimated QTL effects along selected chromosomes. For a backcross, there will be only one line, representing the additive effect. For an intercross, there will be two lines, representing the additive and dominance effects.

**Usage**

```r
effectscan(cross, pheno.col=1, chr, get.se=FALSE, draw=TRUE,  
gap=25, ylim, mtick=c("line","triangle"),  
add.legend=TRUE, alternate.chrid=FALSE, ...)
```

**Arguments**

- `cross`: An object of class `cross`.
- `pheno.col`: Column number in the phenotype matrix which to be drawn in the plot.
- `chr`: Chromosome(s) to be drawn in the plot (optional).
- `get.se`: If TRUE, estimated standard errors are calculated.
- `draw`: If TRUE, draw the figure.
- `gap`: Gap separating chromosomes (in cM).
- `ylim`: Y-axis limits (optional).
- `mtick`: Tick mark type for markers.
- `add.legend`: If TRUE, add a legend.
- `alternate.chrid`: If TRUE and more than one chromosome is plotted, alternate the placement of chromosome axis labels, so that they may be more easily distinguished.
- `...`: Passed to the function `plot` when it is called.

**Details**

The results of `sim.geno` are required for taking account of missing genotype information.

For a backcross, the additive effect is estimated as the difference between the phenotypic averages for heterozygotes and homozygotes.

For recombinant inbred lines, the additive effect is estimated as half the difference between the phenotypic averages for the two homozygotes.
For an intercross, the additive and dominance effects are estimated from linear regression on $a$ and $d$ with $a = -1, 0, 1$, for the AA, AB and BB genotypes, respectively, and $d = 0, 1, 0$, for the AA, AB and BB genotypes, respectively.

As usual, the X chromosome is a bit more complicated. We estimate separate additive effects for the two sexes, and for the two directions within females.

There is a function `plot.effects` that creates the actual plot by calling `plot.scanone`. In the case `get.se=TRUE`, colored regions indicate $\pm 1$ SE.

Value

The results are returned silently, as an object of class "effectscan", which is the same as the form returned by the function `scanone`, though with estimated effects where LOD scores might be. That is, it is a data frame with the first two columns being chromosome ID and position (in cM), and subsequent columns being estimated effects, and (if `get.se=TRUE`) standard errors.

Author(s)

Karl W. Broman, ⟨kbroman@biostat.wisc.edu⟩

References


See Also

effectplot, plot.pxl, sim.geno

Examples

data(fake.f2)
fake.f2 <- sim.geno(fake.f2, step=2.5, n.draws=16)

# allelic effect on whole genome
effectscan(fake.f2)

# on chromosome 13, include standard errors
effectscan(fake.f2, chr="13", mtick="triangle", get.se=TRUE)
Usage

```r
est.map(cross, error.prob=0.0001,
  map.function=c("haldane","kosambi","c-f","morgan"),
  m=0, p=0, maxit=4000, tol=1e-4, sex.sp=TRUE,
  verbose=FALSE)
```

Arguments

- `cross` An object of class `cross`. See `read.cross` for details.
- `error.prob` Assumed genotyping error rate used in the calculation of the penetrance $\text{Pr}$(observed genotype | true genotype).
- `map.function` Indicates whether to use the Haldane, Kosambi, Carter-Falconer, or Morgan map function when converting genetic distances into recombination fractions. (Ignored if $m > 0$.)
- `m` Interference parameter for the chi-square model for interference; a non-negative integer, with $m=0$ corresponding to no interference. This may be used only for a backcross or intercross.
- `p` Proportion of chiasmata from the NI mechanism, in the Stahl model; $p=0$ gives a pure chi-square model. This may be used only for a backcross or intercross.
- `maxit` Maximum number of EM iterations to perform.
- `tol` Tolerance for determining convergence.
- `sex.sp` Indicates whether to estimate sex-specific maps; this is used only for the 4-way cross.
- `verbose` Logical; indicates whether to print initial and final estimates of the recombination fractions for each chromosome.

Details

By default, the map is estimated assuming no crossover interference, but a map function is used to derive the genetic distances (though, by default, the Haldane map function is used).

For a backcross or intercross, inter-marker distances may be estimated using the Stahl model for crossover interference, of which the chi-square model is a special case.

In the chi-square model, points are tossed down onto the four-strand bundle according to a Poisson process, and every $(m + 1)$st point is a chiasma. With the assumption of no chromatid interference, crossover locations on a random meiotic product are obtained by thinning the chiasma process. The parameter $m$ (a non-negative integer) governs the strength of crossover interference, with $m = 0$ corresponding to no interference.

In the Stahl model, chiasmata on the four-strand bundle are a superposition of chiasmata from two mechanisms, one following a chi-square model and one exhibiting no interference. An additional parameter, $p$, gives the proportion of chiasmata from the no interference mechanism.

Value

A `map` object; a list whose components (corresponding to chromosomes) are either vectors of marker positions (in cM) or matrices with two rows of sex-specific marker positions. The maximized log likelihood for each chromosome is saved as an attribute named `loglik`. In the case that
est.rf

estimation was under an interference model (with m > 0), allowed only for a backcross, m and p are also included as attributes.

Author(s)

Karl W Broman, ⟨kbroman@biostat.wisc.edu⟩

References


See Also

plot.map, replace.map, est.rf, fitstahl

Examples

data(fake.f2)

newmap <- est.map(fake.f2)

logliks <- sapply(newmap, attr, "loglik")

plot.map(fake.f2, newmap)

fake.f2 <- replace.map(fake.f2, newmap)

---

**est.rf**

*Estimate pairwise recombination fractions*

**Description**

Estimate the sex-averaged recombination fraction between all pairs of genetic markers.

**Usage**

est.rf(cross, maxit=4000, tol=1e-4)
Arguments

cross   An object of class cross. See read.cross for details.
maxit   Maximum number of iterations for the EM algorithm (not used with backcrosses).
tol     Tolerance for determining convergence (not used with backcrosses).

Details

For a backcross, one can simply count recombination events. For an intercross or 4-way cross, a version of the EM algorithm must be used to estimate recombination fractions. (Since, for example, in an intercross individual that is heterozygous at two loci, it is not known whether there were 0 or 2 recombination events.) Note that, for the 4-way cross, we estimate sex-averaged recombination fractions.

Value

The input cross object is returned with a component, rf, added. This is a matrix of size (tot.mar x tot.mar). The diagonal contains the number of typed meioses per marker, the lower triangle contains the estimated recombination fractions, and the upper triangle contains the LOD scores (testing rf = 0.5).

Author(s)

Karl W Broman, ⟨kbroman@biostat.wisc.edu⟩

See Also

plot.rf, est.map, badorder, checkAlleles

Examples

data(badorder)
badorder <- est.rf(badorder)
plot.rf(badorder)

fake.4way Simulated data for a 4-way cross

Description

Simulated data for a phase-known 4-way cross, obtained using sim.cross.

Usage

data(fake.4way)
Format

An object of class `cross`. See `read.cross` for details.

Details

There are 250 individuals typed at 157 markers, including 8 on the X chromosome.

There are two phenotypes (including sex, for which 0=female and 1=male). The quantitative phenotype is affected by three QTLs: two on chromosome 2 at positions 10 and 25 cM on the female genetic map, and one on chromosome 7 at position 40 cM on the female map.

Author(s)

Karl W Broman, ⟨kbroman@biostat.wisc.edu⟩

See Also

`sim.cross`, `fake.bc`, `fake.f2`, `listeria`, `hyper`, `bristle3`, `bristleX`

Examples

data(fake.4way)

plot(fake.4way)
summary(fake.4way)

# estimate recombination fractions
fake.4way <- est.rf(fake.4way)
plot.rf(fake.4way)

# estimate genetic maps
ssmap <- est.map(fake.4way, verbose=TRUE)
samap <- est.map(fake.4way, sex.sp=FALSE, verbose=TRUE)
plot(ssmap, samap)

# error lod scores
fake.4way <- calc.genoprob(fake.4way, err=0.01)
fake.4way <- calc.errorlod(fake.4way, err=0.01)
top.errorlod(fake.4way, cutoff=2.5)

# genome scan
fake.4way <- calc.genoprob(fake.4way, step=2.5)
out.hk <- scanone(fake.4way, method="hk")
out.em <- scanone(fake.4way, method="em")
plot(out.em, out.hk, chr=c(2,7))
fake.bc  

Simulated data for a backcross

Description

Simulated data for a backcross, obtained using `sim.cross`.

Usage

data(fake.bc)

Format

An object of class `cross`. See `read.cross` for details.

Details

There are 400 backcross individuals typed at 91 markers and with two phenotypes and two covariates (sex and age).

The two phenotypes are due to four QTLs, with no epistasis. There is one on chromosome 2 (at 30 cM), two on chromosome 5 (at 10 and 50 cM), and one on chromosome 10 (at 30 cM). The QTL on chromosome 2 has an effect only in the males (sex=1); the two QTLs on chromosome 5 have effect in coupling for the first phenotype and in repulsion for the second phenotype. Age has an effect of increasing the phenotypes.

Author(s)

Karl W Broman, ⟨kbroman@biostat.wisc.edu⟩

See Also

`sim.cross`, `fake.4way`, `fake.f2`, `listeria`, `hyper`, `bristle3`, `bristleX`

Examples

data(fake.bc)

summary(fake.bc)
plot(fake.bc)

# genome scans without covariates
fake.bc <- calc.genoprob(fake.bc, step=2.5)
out.nocovar <- scanone(fake.bc, pheno.col=1:2)

# genome scans with covariates
ac <- fake.bc$pheno[,c("sex","age")]
ic <- fake.bc$pheno[,"sex"]
out.covar <- scanone(fake.bc, pheno.col=1:2,
        addcovar=ac, intcovar=ic)
fake.f2

Simulated data for an F2 intercross

Description
Simulated data for an F2 intercross, obtained using \texttt{sim.cross}.

Usage
\begin{verbatim}
data(fake.f2)
\end{verbatim}

Format
An object of class \texttt{cross}. See \texttt{read.cross} for details.

Details
There are 200 F2 individuals typed at 94 markers, including 3 on the X chromosome. There is one quantitative phenotype, along with an indication of sex (0=female, 1=male) and the direction of the cross (pgm = paternal grandmother, 0=A, meaning the cross was \((A\times B)\times(A\times B)\), and 1=B, meaning the cross was \((A\times B)\times(B\times A)\).

Note that the X chromosome genotypes are coded in a special way (see \texttt{read.cross}). For the individuals with pgm=0, sex=0, 1=AA and 2=AB; for individuals with pgm=0, sex=1, 1=A and 2=B (hemizygous); for individuals with pgm=1, sex=0, 1=BB and 2=AB; for individuals with pgm=1, sex=1, 1=A and 2=B. \textbf{This requires special care!}

The data were simulated using an additive model with three QTLs on chromosome 1 (at 30, 50 and 70 cM), one QTL on chromosome 13 (at 30 cM), and one QTL on the X chromosome (at 10 cM).

Author(s)
Karl W Broman, \{kbroman@biostat.wisc.edu\}

See Also
\texttt{sim.cross, fake.bc, fake.4way, listeria, hyper, bristle3, bristleX}
**Examples**

```r
data(fake.f2)
summary(fake.f2)
plot(fake.f2)
```

---

**fill.geno**  
*Fill holes in genotype data*

**Description**

Replace the genotype data for a cross with a version imputed either by simulation with `sim.geno` or by the Viterbi algorithm with `argmax.geno`.

**Usage**

```r
fill.geno(cross, method=c("imp","argmax"), error.prob=0.0001, map.function=c("haldane","kosambi","c-f","morgan"))
```

**Arguments**

- `cross`: An object of class `cross`. See `read.cross` for details.
- `method`: Indicates whether to impute using a single simulation replicate from `sim.geno` or using the Viterbi algorithm, as implemented in `argmax.geno`.
- `error.prob`: Assumed genotyping error rate used in the calculation of the penetrance \( \Pr(\text{observed genotype} | \text{true genotype}) \).
- `map.function`: Indicates whether to use the Haldane, Kosambi or Carter-Falconer map function when converting genetic distances into recombination fractions.

**Details**

This function is written so that one may perform rough genome scans by marker regression without having to drop individuals with missing genotype data. **We must caution the user that little trust should be placed in the results.**

**Value**

The input `cross` object with the genotype data replaced by an imputed version. Any intermediate calculations (such as is produced by `calc.genoprob`, `argmax.geno` and `sim.geno`) are removed.

**Author(s)**

Karl W Broman, (kbroman@biostat.wisc.edu)

**See Also**

`sim.geno`, `argmax.geno`
find.flanking  

Find flanking markers for a specified position

Description
Find the genetic markers flanking a specified position on a chromosome, as well as the marker that is closest to the specified position.

Usage

find.flanking(cross, chr, pos)

Arguments

cross An object of class cross. See read.cross for details.
chr A vector of chromosome identifiers, or a single such.
pos A vector of cM positions.

Value

A data.frame, each row corresponding to one of the input positions. The first column contains the left-flanking markers, the second column contains the right-flanking markers, and the third column contains the markers closest to the specified positions.

Author(s)
Brian Yandell

See Also

find.marker, plot.pxg, find.markerpos, find.pseudomarker

Examples

data(listeria)
fnd.flanking(listeria, 5, 28)
fnd.flanking(listeria, c(1, 5, 13), c(81, 28, 26))
find.marker

Find marker closest to a specified position

Description

Find the genetic marker closest to a specified position on a chromosome.

Usage

find.marker(cross, chr, pos)

Arguments

cross: An object of class cross. See read.cross for details.
chr: A vector of chromosome identifiers, or a single such.
pos: A vector of cM positions.

Details

If the input chr has length one, it is expanded to the same length as the input pos.
If multiple markers are exactly the same distance from the specified position, one is chosen at random from among those with the most genotype data.
For a cross with sex-specific maps, the input positions are assumed to correspond to the female genetic map.

Value

A vector of marker names (of the same length as the input pos), corresponding to the markers nearest to the specified chromosomes/positions.

Author(s)

Karl W Broman, ⟨kbroman@biostat.wisc.edu⟩

See Also

find.flanking, plot.pxp, find.pseudomarker, effectplot, find.markerpos

Examples

data(listeria)
find.marker(listeria, 5, 28)
find.marker(listeria, c(1, 5, 13), c(81, 28, 26))
find.markerpos Find position of a marker.

Description

Find the chromosome and cM position of a set of genetic markers.

Usage

find.markerpos(cross, marker)

Arguments

cross An object of class cross. See read.cross for details.
marker A vector of marker names.

Value

A data frame with two columns: the chromosome and position of the markers.

Author(s)

Karl W Broman, ⟨kbroman@biostat.wisc.edu⟩

See Also

find.flanking, find.marker, find.pseudomarker

Examples

data(hyper)
find.markerpos(hyper, "D4Mit164")
find.markerpos(hyper, c("D4Mit164", "D1Mit94"))

find.pheno Find column number for a particular phenotype.

Description

Find the column number corresponding to a particular phenotype name.

Usage

find.pheno(cross, pheno)
Arguments

- `cross` An object of class `cross`. See `read.cross` for details.
- `pheno` Vector of phenotype names (as character strings).

Value

A vector of numbers, corresponding to the column numbers of the phenotype in the input cross with the specified names.

Author(s)

Brian Yandell

Examples

```r
data(fake.bc)
find.pheno(fake.bc, "sex")
```

find.pseudomarker Find the pseudomarker closest to a specified position

Description

Find the pseudomarker closest to a specified position on a chromosome.

Usage

```r
find.pseudomarker(cross, chr, pos, where=c("draws", "prob"))
```

Arguments

- `cross` An object of class `cross`. See `read.cross` for details.
- `chr` A vector of chromosome identifiers, or a single such.
- `pos` A vector of cM positions.
- `where` Indicates whether to look in the `draws` or `prob` components of the input cross.

Details

If the input `chr` has length one, it is expanded to the same length as the input `pos`.

If multiple markers are exactly the same distance from the specified position, one is chosen at random from among those with the most genotype data.

For a cross with sex-specific maps, the input positions are assumed to correspond to the female genetic map.
Value

A vector of pseudomarker names (of the same length as the input pos), corresponding to the markers nearest to the specified chromosomes/positions.

Author(s)

Karl W Broman, ⟨kbroman@biostat.wisc.edu⟩

See Also

find.flanking, plot.pxg, effectplot, find.marker, find.markerpos

Examples

data(listeria)
listeria <- calc.genoprob(listeria, step=2.5)
find.pseudomarker(listeria, 5, 28, "prob")
find.pseudomarker(listeria, c(1, 5, 13), c(81, 28, 26), "prob")

fitqtl  

Fit a multiple-QTL model

Description

Fits a user-specified multiple-QTL model. If specified, a drop-one-term analysis will be performed.

Usage

fitqtl(pheno, qtl, covar=NULL, formula, method=c("imp"),
dropone=TRUE, get.ests=FALSE)

Arguments

pheno  
Phenotype data (a numeric vector).

qtl  
An object of class qtl, as output from makeqtl.

covar  
A data.frame of covariates

formula  
An object of class formula indicating the model to be fitted. QTLs are referred to as Q1, Q2, etc. Covariates are referred to by their names in the data frame covar.

method  
Indicates whether to use the EM algorithm or imputation. (Only imputation is implemented at this point.)

dropone  
If TRUE, do drop-one-term analysis.

get.ests  
If TRUE, return estimated QTL effects and their estimated variance-covariance matrix.
Details

In the drop-one-term analysis, for a given QTL/covariate model, all submodels will be analyzed. For each term in the input formula, when it is dropped, all higher order terms that contain it will also be dropped. The comparison between the new model and the full (input) model will be output.

The part to get estimated QTL effects is fully working only for the case of autosomes in a backcross or intercross. In other cases the values returned are based on a design matrix that is convenient for calculations but not easily interpreted.

Value

An object of class `fitqtl`. It may contains as many as three fields:

1. `result.full` is the ANOVA table as a matrix for the full model result. It contains the degree of freedom (df), Sum of squares (SS), mean square (MS), LOD score (LOD), percentage of variance explained (%var) and P value (Pvalue).
2. `result.drop` is a drop-one-term ANOVA table as a matrix. It contains degrees of freedom (df), Type III sum of squares (Type III SS), LOD score(LOD), percentage of variance explained (%var), F statistics (F value), and P values for chi square (Pvalue(chi2)) and F distribution (Pvalue(F)).

   Note that the degree of freedom, Type III sum of squares, the LOD score and the percentage of variance explained are the values comparing the full to the sub-model with the term dropped. Also note that for imputation method, the percentage of variance explained, the the F values and the P values are approximations calculated from the LOD score.
3. `ests` contains the estimated QTL effects and standard errors.

   The part to get estimated QTL effects is fully working only for the case of autosomes in a backcross or intercross. In other cases the values returned are based on a design matrix that is convenient for calculations but not easily interpreted.

Author(s)

Hao Wu

References


See Also

`summary.fitqtl`, `makeqtl`, `scanqtl`

Examples

```r
data(fake.f2)

# take out several QTLs and make QTL object
qc <- c(1, 8, 13)
qp <- c(26, 56, 28)
fake.f2 <- subset(fake.f2, chr=qc)
```
fake.f2 <- sim.geno(fake.f2, n.draws=8, step=2, err=0.001)
qtl1 <- makeqtl(fake.f2, qc, qp)

# fit model with 3 interacting QTLs interacting
# (performing a drop-one-term analysis)
 lod <- fitqtl(fake.f2$pheno[,1], qtl1, formula=y~Q1*Q2*Q3)
 summary(lod)

## Not run:
# fit an additive QTL model
 lod.add <- fitqtl(fake.f2$pheno[,1], qtl, formula=y~Q1+Q2+Q3)
 summary(lod.add)

# fit the model including sex as an interacting covariate
Sex <- data.frame(Sex=fake.f2$pheno$sex)
 lod.sex <- fitqtl(fake.f2$pheno[,1], qtl, formula=y~Q1*Q2*Q3*Sex, cov=Sex)
 summary(lod.sex)

# fit the same with an additive model
 lod.sex.add <- fitqtl(fake.f2$pheno[,1], qtl, formula=y~Q1+Q2+Q3+Sex, cov=Sex)
 summary(lod.sex.add)
## End(Not run)

---

**fitstahl**

*Fit Stahl interference model*

**Description**

Fit the Stahl model for crossover inference (or the chi-square model, which is a special case).

**Usage**

```r
fitstahl(cross, chr, m, p, error.prob=0.0001, maxit=4000, tol=1e-4, 
 maxm=15, verbose=TRUE)
```

**Arguments**

- `cross` An object of class `cross`. See `read.cross` for details.
- `chr` If specified, only selected chromosomes are used.
- `m` Interference parameter (a non-negative integer); if unspecified, this is estimated.
- `p` The proportion of chiasmata coming from the no interference mechanism in the Stahl model (0 <= p <= 1). p=0 gives the chi-square model. If unspecified, this is estimated.
- `error.prob` The genotyping error probability. If = NULL, it is estimated.
- `maxit` Maximum number of iterations to perform.
- `tol` Tolerance for determining convergence.
- `maxm` Maximum value of m to consider, if m is unspecified.
- `verbose` Logical; indicates whether to print tracing information.
Details

This function is currently only available for backcrosses and intercrosses.

The Stahl model of crossover interference (of which the chi-square model is a special case) is fit. In the chi-square model, points are tossed down onto the four-strand bundle according to a Poisson process, and every \((m + 1)\)st point is a chiasma. With the assumption of no chromatid interference, crossover locations on a random meiotic product are obtained by thinning the chiasma process. The parameter \(m\) (a non-negative integer) governs the strength of crossover interference, with \(m = 0\) corresponding to no interference.

In the Stahl model, chiasmata on the four-strand bundle are a superposition of chiasmata from two mechanisms, one following a chi-square model and one exhibiting no interference. An additional parameter, \(p\), gives the proportion of chiasmata from the no interference mechanism.

If all of \(m\), \(p\), and \(\text{error.prob}\) are specified, any of them with length > 1 must all have the same length.

If \(m\) is unspecified, we do a grid search starting at 0 and stop when the likelihood decreases (thus assuming a single mode), or \(\text{maxm}\) is reached.

Value

A matrix with four columns: \(m\), \(p\), \(\text{error.prob}\), and the log likelihood.

If specific values for \(m\), \(p\), \(\text{error.prob}\) are provided, the log likelihood for each set are given.

If some are left unspecified, the maximum likelihood estimates are provided in the results.

Author(s)

Karl W Broman, (kbroman@biostat.wisc.edu)

References


See Also

est.map, sim.cross

Examples

# Simulate genetic map: one chromosome of length 200 cM with
# a 2 cM marker spacing
mymap <- sim.map(200, 51, TRUE, FALSE, FALSE, TRUE)

# Simulate data under the chi-square model, no errors
mydata <- sim.cross(mymap, n.ind=250, type="bc",
                    error.prob=0, m=3, p=0)
# Fit the chi-square model for specified m's
output <- fitstahl(mydata, m=1:5, p=0, error.prob=0)
plot(output$m, output$loglik, lwd=2, type="b")

# Find the MLE of m in the chi-square model
## Not run: mle <- fitstahl(mydata, p=0, error.prob=0)
## Not run:
# Simulate data under the Stahl model, no errors
mydata <- sim.cross(mymap, n.ind=250, type="bc",
                   error.prob=0, m=3, p=0.1)

# Find MLE of m for the Stahl model with known p
mle.stahl <- fitstahl(mydata, p=0.1, error.prob=0)

# Fit the Stahl model with unknown p and m,
# get results for m=0, 1, 2, ..., 8
output <- fitstahl(mydata, m=0:8, error.prob=0)
plot(output$m, output$loglik, type="b", lwd=2)## End(Not run)

geno.image

---

**geno.image**

*Plot grid of genotype data*

**Description**

Plot a grid showing which the genotype data in a cross.

**Usage**

```r
geno.image(x, chr, reorder=FALSE, main="Genotype data",
            alternate.chrid=FALSE, ...)
```

**Arguments**

- `x`  
  An object of class `cross`. See `read.cross` for details.

- `chr`  
  Vector of chromosomes to plot. If missing, plot all chromosomes.

- `reorder`  
  Specify whether to reorder individuals according to their phenotypes.

  - `FALSE`  
    Don’t reorder

  - `TRUE`  
    Reorder according to the sum of the phenotypes

  - `n`  
    Reorder according to phenotype n

- `main`  
  Title to place on plot.

- `alternate.chrid`  
  If TRUE and more than one chromosome is plotted, alternate the placement of chromosome axis labels, so that they may be more easily distinguished.

- `...`  
  Ignored at this point.
Details

Uses `image` to plot a grid with the genotype data. The genotypes AA, AB, BB are displayed in the colors red, blue, and green, respectively. In an intercross, if there are genotypes “not BB” and “not AA”, these are displayed in purple and orange, respectively. White pixels indicate missing data.

Value

None.

Author(s)

Karl W Broman, ⟨kbroman@biostat.wisc.edu⟩

See Also

`plot.cross`, `plot.missing`, `plot.geno`, `image`

Examples

data(listeria)
geno.image(listeria)

data(listeria)
geno.table(listeria)

---

geno.table Create table of genotype distributions

Description

Create table showing the observed numbers of individuals with each genotype at each marker, including P-values from chi-square tests for Mendelian segregation.

Usage

geno.table(cross, chr)

Arguments

cross An object of class `cross`. See `read.cross` for details.
chr A vector specifying which chromosomes to consider. This may be a logical vector, a numeric vector, or a vector of character strings.

Details

The P-values are obtained from chi-square tests of Mendelian segregation. In the case of the X chromosome, the sexes and cross directions are tested separately, and the chi-square statistics combined, and so the test is of whether any of the groups show deviation from Mendel's rules.
Value

A matrix containing, for each marker, the number of individuals with each possible genotype, as well as the number that were not typed. The first column gives the chromosome ID, and the last column gives P-values from chi-square tests of Mendelian segregation.

Author(s)

Karl W Broman, ⟨kbroman@biostat.wisc.edu⟩

See Also

summary.cross, drop.markers, drop.nullmarkers

Examples

data(listeria)
geno.table(listeria)

geno.table(listeria, chr=13)

gt <- geno.table(listeria)
gt[gt$P.value < 0.01,]

 hyper  Data on hypertension

Description

Data from an experiment on hypertension in the mouse.

Usage

data(hyper)

Format

An object of class cross. See read.cross for details.

Details

There are 250 male backcross individuals typed at 174 markers (actually one contains only missing values), including 4 on the X chromosome, with one phenotype.

The phenotype is the blood pressure. See the reference below. Note that, for most markers, genotypes are available on only the individuals with extreme phenotypes. At many markers, only recombinant individuals were typed.
Source
Bev Paigen and Gary Churchill (The Jackson Laboratory, Bar Harbor, Maine) http://www.jax.org/research/churchill/datasets/qtl/qtlarchive

References

See Also
fake.bc, fake.f2, fake.4way, listeria, bristle3, bristleX

Examples
data(hyper)
summary(hyper)
plot(hyper)

# Note the selective genotyping
plot.missing(hyper, reorder=TRUE)

# A marker on c14 has no data; remove it
hyper <- drop.nullmarkers(hyper)

---

jittermap

Jitter marker positions in a genetic map

Description
Jitter the marker positions in a genetic map so that no two markers are on top of each other.

Usage
jittermap(object, amount=1e-6)

Arguments

object Either a cross (an object of class cross; see read.cross for details) or a map (an object of class map; see pull.map for details).

amount The amount by which markers should be moved.

Value
Either the input cross object or the input map, but with marker positions slightly jittered. If the input was a cross, the function clean is run to strip off any intermediate calculations.
Author(s)
Karl W Broman, (kbroman@biostat.wisc.edu)

See Also
pull.map, replace.map, summary.cross

Examples

data(hyper)
hyper <- jittermap(hyper)

listeria  Data on Listeria monocytogenes susceptibility

Description
Data from an experiment on susceptibility to Listeria monocytogenes infection in the mouse.

Usage
data(listeria)

Format
An object of class cross. See read.cross for details.

Details
There are 120 F2 individuals typed at 133 markers, including 2 on the X chromosome, with one phenotype.

The phenotype is the survival time (in hours) following infection. Mice with phenotype 264 hours may be considered to have recovered from the infection. See the references below.

Source
Victor Boyartchuk and William Dietrich (Department of Genetics, Harvard Medical School and Howard Hughes Medical Institute) http://genetics.med.harvard.edu/~bdlab

References

See Also

`fake.bc, fake.f2, fake.4way, hyper, bristle3, bristleX`

Examples

```r
data(listeria)

# Summaries
summary(listeria)
plot(listeria)

# Take log of phenotype
listeria$pheno[,1] <- log2(listeria$pheno[,1])
plot(listeria)

# Genome scan with a two-part model, using log survival
listeria <- calc.genoprob(listeria, step=2)
out <- scanone(listeria, model="2part", method="em", upper=TRUE)

# Summary of the results
summary(out, thr=c(5,3,3), format="allpeaks")

# Plot LOD curves for interesting chromosomes
# (The two-part model gives three LOD scores)
plot(out, chr=c(1,5,6,13,15), lodcolumn=1:3,
     lty=1, col=c("black","red","blue"))
```

lodint

**LOD support interval**

Description

Calculate a LOD support interval for a particular chromosome, using output from scanone.

Usage

```r
lodint(results, chr, drop=1.5, lodcolumn=1)
```

Arguments

- `results`: Output from `scanone`.
- `chr`: A chromosome ID.
- `drop`: LOD units to drop to form the interval.
- `lodcolumn`: An integer, or vector of 3 integers, indicating which of the LOD score columns should be plotted (generally this is 1).
Value

An object of class scanone, like the input, indicating the position with the maximum LOD, and indicating approximate endpoints for the LOD support interval.

Author(s)

Karl W Broman, ⟨kbroman@biostat.wisc.edu⟩

See Also

scanone, bayesint

Examples

data(hyper)

hyper <- calc.genoprob(hyper, step=0.5)
out <- scanone(hyper, method="hk")
lodint(out, chr=1)
lodint(out, chr=4)
lodint(out, chr=4, drop=2)

makeqtl

Make a qtl object

Description

This function takes a cross object and specified chromosome numbers and positions and pulls out the genotype probabilities or imputed genotypes at the nearest pseudomarkers, for later use by the function fitqtl.

Usage

makeqtl(cross, chr, pos, qtl.name, what=c("draws","prob"))

Arguments

cross An object of class cross. See read.cross for details.
chr Vector indicating the chromosome for each QTL.
pos Vector (of same length as chr) indicating the positions on the chromosome to be taken. If there’s no marker or pseudomarker at a position, genotypes for the nearest positions are taken.
qtl.name The user-specified name for each QTL, used in the drop-one-term ANOVA table in fitqtl. If unspecified, the names will be of the form "Chr1@10" for a QTL on Chromosome 1 at 10 cM.
what Indicates whether to pull out the imputed genotypes or the genotype probabilities.
Details

This function will take out the genotype probabilities and imputed genotypes if they are present in the input cross object. If both fields are missing in the input object, the function will report an error. Before running this function, the user must have first run either sim.geno (for what="draws") or calc.genoprob (for what="prob").

Value

An object of class qtl with the following elements (though only one of geno and prob will be included, according to whether what is given as "draws" or "prob"):

- Imputed genotypes.
- Genotype probabilities.
- Input vector of chromosome numbers.
- Input vector of chromosome positions.
- Number of QTLs.
- Number of individuals.
- A vector indicating the number of genotypes for each QTL.

Author(s)

Hao Wu

See Also

fitqtl, calc.genoprob, sim.geno

Examples

data(fake.f2)

# take out several QTLs and make QTL object
qc <- c(1, 6, 13)
qp <- c(25.8, 33.6, 18.63)
fake.f2 <- subset(fake.f2, chr=qc)

fake.f2 <- sim.geno(fake.f2, n.draws=8, step=2, err=0.001)
qtl <- makeqtl(fake.f2, qc, qp, what="draws")
map10  

An example genetic map

Description

A genetic map corresponding approximately to the mouse genome with a 10 cM marker spacing.

Usage

data(map10)

Format

An object of class map: a list whose components are vectors of marker locations. This map approximates the mouse genome, with 20 chromosomes (including the X chromosome) and 187 markers at an approximately 10 cM spacing. The markers are equally spaced on each chromosome, but the spacings are a bit above or below 10 cM, so that the lengths match those in the Mouse Genome Database.

See Also

sim.map, plot.map, pull.map

Examples

data(map10)
plot(map10)

mycross <- sim.cross(map10, type="f2", n.ind=100)

max.scanone  

Maximum peak in genome scan

Description

Print the row of the output from scanone that corresponds to the maximum LOD, genome-wide.

Usage

## S3 method for class 'scanone':
max(object, chr, lodcolumn=1, df=FALSE, na.rm=TRUE, ...)

Arguments

- **object**: An object of the form output by the function `scanone`: a data.frame whose third column is the LOD score.
- **chr**: Numeric or character vector indicating chromosomes on which to find the maximum LOD score.
- **lodcolumn**: An integer, indicating which of the LOD score columns should be considered in pulling out the peak (these are indexed 1, 2, ...).
- **df**: If TRUE, the degrees of freedom associated with the LOD scores are shown.
- **na.rm**: A logical indicating whether missing values should be removed.
- **...**: Ignored.

Value

An object of class `summary.scanone`, to be printed by `print.summary.scanone`. This is a data.frame with one row, corresponding to the maximum LOD peak either genome-wide or for the particular chromosome specified.

Author(s)

Karl W Broman, \texttt{(kbroman@biostat.wisc.edu)}

See Also

- `scanone`
- `plot.scanone`
- `summary.scanone`

Examples

```r
data(listeria)
listeria <- calc.genoprob(listeria, step=2.5)
out <- scanone(listeria, model="2part", upper=TRUE)
# Maximum peak for LOD(p,mu)
max(out)
max(out, chr=13)

# Maximum peak for LOD(p)
max(out, lodcolumn=2)

# Maximum peak for LOD(mu)
max(out, lodcolumn=3)
```
max.scantwo

Maximum peak in two-dimensional genome scan

Description
Print the pair of loci with the largest LOD score in the results of scantwo.

Usage
## S3 method for class 'scantwo':
max(object, lodcolumn=1,
    what=c("best", "full", "add", "int"),
    df=FALSE, na.rm=TRUE, ...)

Arguments
- object: An object of class scantwo, the output of the function scantwo.
- lodcolumn: If the scantwo results contain LOD scores for multiple phenotypes, this argument indicates which to use.
- what: Indicates for which LOD score the maximum should be reported.
- df: If TRUE, the degrees of freedom associated with the LOD scores are shown.
- na.rm: Ignored.
- ...: Ignored.

Details
This is very similar to the summary.scantwo function, though this pulls out one pair of positions.
If what="best", we find the pair of positions at which the LOD score for the full model (2 QTL + interaction) is maximized, and then also print the positions on that same pair of chromosomes at which the additive LOD score is maximized.
In the other cases, we pull out the pair of positions with the largest LOD score; which LOD score is considered is indicated by the what argument.

Value
An object of class summary.scantwo, to be printed by print.summary.scantwo, with the pair of positions with the maximum LOD score. (Which LOD score is considered is indicated by the what argument.)

Author(s)
Karl W Broman, (kbroman@biostat.wisc.edu)
movemarker

See Also

scantwo.plot, scantwo, summary.scantwo

Examples

data(fake.f2)

fake.f2 <- calc.genoprob(fake.f2, step=10)
out.2dim <- scantwo(fake.f2, method="hk")
max(out.2dim)

---

movemarker  

Move a marker to a new chromosome

Description

Move a specified marker to a different chromosome.

Usage

movemarker(cross, marker, newchr, newpos)

Arguments

cross  

An object of class cross. See read.cross for details.

marker  

The name of the marker to be moved (a character string).

newchr

The chromosome to which the marker should be moved.

newpos

The position (in cM) at which the marker should be placed. If missing, the marker is placed at the end of the chromosome.

Value

The input cross object, but with the specified marker moved to the specified chromosome.

All intermediate calculations (such as the results of calc.genoprob and est.rf) are removed.

Author(s)

Karl W Broman, (kbroman@biostat.wisc.edu)

See Also

switch.order

Examples

data(badorder)
badorder <- movemarker(badorder, "D2M937", 3, 48.15)
badorder <- movemarker(badorder, "D3M160", 2, 28.83)
### nmissing

**Number of missing genotypes**

**Description**

Count the number of missing genotypes for each individual or each marker in a cross.

**Usage**

```r
nmissing(cross, what=c("ind","mar"))
```

**Arguments**

- `cross`: An object of class `cross`. See `read.cross` for details.
- `what`: Indicates whether to count missing genotypes for each individual or each marker.

**Value**

A vector containing the number of missing genotypes for each individual or for each marker.

**Author(s)**

Karl W Broman, ⟨kbroman@biostat.wisc.edu⟩

**See Also**

`summary.cross, nind, totmar`

**Examples**

```r
data(listeria)

# plot number of missing genotypes for each individual
plot(nmissing(listeria))

# plot number of missing genotypes for each marker
plot(nmissing(listeria, what="mar"))
```
plot.cross  

Plot various features of a cross object

Description

Plots grid of the missing genotypes, genetic map, and histograms or barplots of phenotypes for the data from an experimental cross.

Usage

```r
## S3 method for class 'cross':
plot(x, auto.layout=TRUE, pheno.col,
     alternate.chrid=TRUE, ...)
```

Arguments

- `x`  
  An object of class `cross`. See `read.cross` for details.
- `auto.layout`  
  If TRUE, `par(mfrow)` is set so that all plots fit within one figure.
- `pheno.col`  
  Vector of numbers or character strings corresponding to phenotypes that should be plotted. If unspecified, all phenotypes are plotted.
- `alternate.chrid`  
  If TRUE and more than one chromosome is plotted, alternate the placement of chromosome axis labels, so that they may be more easily distinguished.
- `...`  
  Ignored at this point.

Details

Calls `plot.missing`, `plot.map` and `plot.pheno` to plot the missing genotypes, genetic map, and histograms or barplots of all phenotypes.

If `auto.format=TRUE`, `par(mfrow)` is used with `ceiling(sqrt(n.phe+2))` rows and the minimum number of columns so that all plots fit on the plotting device.

Numeric phenotypes are displayed as histograms or barplots by calling `plot.pheno`.

Value

None.

Author(s)

Karl W Broman, ⟨kbroman@biostat.wisc.edu⟩; Brian Yandell

See Also

`plot.missing`, `plot.map`, `plot.pheno`
plot.errorlod

Examples

    data(fake.bc)
    plot(fake.bc)

plot.errorlod  Plot grid of error LOD values

Description

Plot a grid of the LOD scores indicating which genotypes are likely to be in error.

Usage

    plot.errorlod(x, chr, ind, breaks=c(-Inf, 2, 3, 4.5, Inf),
                    col=c("white", "gray85", "hotpink", "purple3"),
                    alternate.chrid=FALSE, ...)

Arguments

    x  An object of class cross. See read.cross for details.
    chr  The chromosomes for which the error LOD scores should be plotted.
    ind  Indicates the individuals for which the error LOD scores should be plotted.
    breaks  A set of breakpoints for the colors; must give one more breakpoint than color. Intervals are open on the left and closed on the right, except for the lowest interval.
    col  A vector of colors to appear in the image.
    alternate.chrid  If TRUE and more than one chromosome is plotted, alternate the placement of chromosome axis labels, so that they may be more easily distinguished.
    ...  Ignored at this point.

Details

Uses image to plot a grid with different shades of pixels to indicate which genotypes are likely to be in error.

Darker pixels have higher error LOD scores: \( LOD \leq 2 \) in white; \( 2 < LOD \leq 3 \) in gray; \( 3 < LOD \leq 4.5 \) in pink; \( LOD > 4.5 \) in purple.

Value

None.

Author(s)

Karl W Broman, (kbroman@biostat.wisc.edu)
References


See Also

`calc.errorlod`, `top.errorlod`, `image`

Examples

```r
data(hyper)

# Calculate error LOD scores
hyper <- calc.errorlod(hyper, error.prob=0.01)

# plot the error LOD scores; print those above a specified cutoff
plot.errorlod(hyper)
plot.errorlod(hyper, chr=1)
```

---

**plot.geno**

*Plot observed genotypes, flagging likely errors*

**Description**

Plot the genotypes on a particular chromosome for a set of individuals, flagging likely errors.

**Usage**

```
plot.geno(x, chr, ind, include.xo=TRUE, horizontal=FALSE,
          cutoff=4, min.sep=2, cex=1.2, ...)
```

**Arguments**

- `x` An object of class `cross`. See `read.cross` for details.
- `chr` Chromosome number to plot.
- `ind` Vector of individuals to plot. If missing, all individuals are plotted.
- `include.xo` If TRUE, plot X’s in intervals having a crossover. Not available for a 4-way cross.
- `horizontal` If TRUE, chromosomes are plotted horizontally.
- `cutoff` Genotypes with error LOD scores above this value are flagged as possible errors.
- `min.sep` Markers separated by less than this value (as a percent of the chromosome length) are pulled apart, so that they may be distinguished in the picture.
- `cex` Character expansion for the size of points in the plot. Larger numbers give larger points; see `par`.
- `...` Ignored at this point.
Details

Plots the genotypes for a set of individuals. Likely errors are indicated by red squares. In a backcross, genotypes AA and AB are indicated by white and black circles, respectively. In an intercross, genotypes AA, AB and BB are indicated by white, gray, and black circles, respectively, and the partially missing genotypes "not BB" (D in mapmaker) and "not AA" (C in mapmaker) are indicated by green and orange circles, respectively.

For a 4-way cross, two lines are plotted for each individual. The left or upper line indicates the allele A (white) or B (black); the right or lower line indicates the allele C (white) or D (black). For the case that genotype is known to be only AC/BD or AD/BC, we use green and orange, respectively.

Value

None.

Author(s)

Karl W Broman, ⟨kbroman@biostat.wisc.edu⟩

See Also

calc.errorlod, top.errorlod

Examples

data(hyper)

# Calculate error LOD scores
hyper <- calc.errorlod(hyper,error.prob=0.01)

# print those above a specified cutoff
top.errorlod(hyper,cutoff=4)

# plot genotype data, flagging genotypes with error LOD > cutoff
plot.geno(hyper, chr=1, ind=160:200, cutoff=7, min.sep=2)

plot.info

Plot the proportion of missing genotype information

Description

Plot a measure of the proportion of missing information in the genotype data.

Usage

plot.info(x, chr, method=c("both","entropy","variance"), step=1,
          off.end=0, error.prob=0.001,
          map.function=c("haldane","kosambi","c-f","morgan"),
          alternate.chrid=FALSE, ...)
Arguments

- **x**: An object of class `cross`. See `read.cross` for details.
- **chr**: Vector specifying the chromosomes to plot.
- **method**: Indicates whether to plot the entropy version of the information, the variance version, or both.
- **step**: Maximum distance (in cM) between positions at which the missing information is calculated, though for step=0, it is are calculated only at the marker locations.
- **off.end**: Distance (in cM) past the terminal markers on each chromosome to which the genotype probability calculations will be carried.
- **error.prob**: Assumed genotyping error rate used in the calculation of the penetrance Pr(observable genotype | true genotype).
- **map.function**: Indicates whether to use the Haldane, Kosambi or Carter-Falconer map function when converting genetic distances into recombination fractions.
- **alternate.chrid**: If TRUE and more than one chromosome is plotted, alternate the placement of chromosome axis labels, so that they may be more easily distinguished.
- **...**: Passed to `plot.scanone`.

Details

The entropy version of the missing information: for a single individual at a single genomic position, we measure the missing information as \( H = \sum p_g \log p_g / \log n \), where \( p_g \) is the probability of the genotype \( g \), and \( n \) is the number of possible genotypes, defining \( 0 \log 0 = 0 \). This takes values between 0 and 1, assuming the value 1 when the genotypes (given the marker data) are equally likely and 0 when the genotypes are completely determined. We calculate the missing information at a particular position as the average of \( H \) across individuals. For an intercross, we don’t scale by \( \log n \) but by the entropy in the case of genotype probabilities (1/4, 1/2, 1/4).

The variance version of the missing information: we calculate the average, across individuals, of the variance of the genotype distribution (conditional on the observed marker data) at a particular locus, and scale by the maximum such variance.

Calculations are done in C (for the sake of speed in the presence of little thought about programming efficiency) and the plot is created by a call to `plot.scanone`.

Note that `summary.scanone` may be used to display the maximum missing information on each chromosome.

Value

An object with class `scanone`: a data.frame with columns the chromosome IDs and cM positions followed by the entropy and/or variance version of the missing information.

Author(s)

Karl W Broman, (kbroman@biostat.wisc.edu)
plot.map

See Also

plot.scanone, plot.missing

Examples

data(hyper)

plot.info(hyper, chr=c(1,4))

# save the results and view maximum missing info on each chr
info <- plot.info(hyper)
summary(info)

Description

Plot genetic map of marker locations for all chromosomes.

Usage

## S3 method for class 'map':
plot(x, map2, chr, horizontal=FALSE, shift=TRUE,
    show.marker.names=FALSE, alternate.chrid=FALSE, ...)

Arguments

x
A list whose components are vectors of marker locations. A cross object may
be given instead, in which case the genetic map it contains is used.

map2
An optional second genetic map with the same number of chromosomes and
markers as the first. As with the first argument, a cross object may be given
instead. If this argument is given, a comparison of the two genetic maps is
plotted.

chr
Vector specifying which chromosomes to plot. (The chromosomes must be
specified by name.)

horizontal
Specifies whether the chromosomes should be plotted horizontally.

shift
If TRUE, shift the first marker on each chromosome to be at 0 cM.

show.marker.names
If TRUE, marker names are included.

alternate.chrid
If TRUE and more than one chromosome is plotted, alternate the placement of
chromosome axis labels, so that they may be more easily distinguished.

...
Ignored at this point.
plot.missing

Details

Plots the genetic map for each chromosome, or a comparison of the genetic maps if two maps are given.

For a sex-specific map, female and male maps are plotted against one another. For two sex-specific maps, the two female maps are plotted against one another and the two male maps are plotted against one another.

Value

None.

Author(s)

Karl W Broman, (kbroman@biostat.wisc.edu)

See Also

est.map, plot.cross

Examples

data(fake.bc)

plot.map(fake.bc)
plot.map(fake.bc, horizontal=TRUE)

newmap <- est.map(fake.bc)
plot(newmap)
plot.map(fake.bc, newmap)

plot.map(fake.bc, show.marker.names=TRUE)

plot.missing

Plot grid of missing genotypes

Description

Plot a grid showing which genotypes are missing.

Usage

plot.missing(x, chr, reorder=FALSE, main="Missing genotypes", alternate.chrid=FALSE, ...)

Plot grid of missing genotypes
Arguments

x
An object of class `cross`. See `read.cross` for details.

chr
Vector of chromosomes to plot. If missing, plot all chromosomes.

reorder
Specify whether to reorder individuals according to their phenotypes.

FALSE  Don’t reorder
TRUE  Reorder according to the sum of the phenotypes
n  Reorder according to phenotype n

main
Title to place on plot.

alternate.chrid
If TRUE and more than one chromosome is plotted, alternate the placement of chromosome axis labels, so that they may be more easily distinguished.

... Ignored at this point.

Details

Uses `image` to plot a grid with black pixels where the genotypes are missing. For intercross and 4-way cross data, gray pixels are plotted for the partially missing genotypes (for example, “not AA”).

Value

None.

Author(s)

Karl W Broman, ⟨kbroman@biostat.wisc.edu⟩

See Also

`plot.cross, geno.image, image`

Examples

data(fake.f2)
plot.missing(fake.f2)

---

`plot.pheno`  
*Plot a phenotype distribution*

Description

Plots a histogram or barplot of the data for a phenotype from an experimental cross.
Usage

```r
## S3 method for class 'pheno':
plot(x, pheno.col=1, ...)
```

Arguments

- `x`: An object of class `cross`. See `read.cross` for details.
- `pheno.col`: The phenotype column to plot: a numeric index, or the phenotype name as a character string.
- `...`: Passed to `hist` or `barplot`.

Details

Numeric phenotypes are displayed as histograms with approximately \(2\sqrt{n}\) bins. Phenotypes that are factors or that have very few unique values are displayed as barplots.

Value

None.

Author(s)

Karl W Broman, ⟨kbroman@biostat.wisc.edu⟩

See Also

`plot.cross`, `plot.map`, `plot.missing`, `hist`, `barplot`

Examples

```r
data(fake.bc)
plot(fake.bc, pheno.col=1)
plot(fake.bc, pheno.col=3)
```

Description

Plot phenotypes versus marker genotypes.

Usage

```r
plot.pxg(x, marker, pheno.col=1, jitter=1, infer=TRUE, 
         pch, ylab, main, col, ...)```
Arguments

- **x**: An object of class `cross`. See `read.cross` for details.
- **marker**: Marker name (a character string; can be a vector).
- **pheno.col**: Column number in the phenotype matrix which should be used as the phenotype.
- **jitter**: A positive number indicating how much to spread out the points horizontally. (Larger numbers correspond to greater spread.)
- **infer**: If TRUE, missing genotypes are filled in with a single random imputation and plotted in red; if FALSE, only individuals typed at the specified marker are plotted.
- **pch**: Plot symbol.
- **ylab**: Label for y-axis.
- **main**: Main title for the plot. If missing, the names of the markers are used.
- **col**: A vector of colors to use for the confidence intervals (optional).
- **...**: Ignored at this point.

Details

Plots the phenotype data against the genotypes at the specified marker. If infer=TRUE, the genotypes of individuals that were not typed is inferred based the genotypes at linked markers via a single imputation from `sim.geno`; these points are plotted in red. For each genotype, the phenotypic mean is plotted, with error bars at ±1 SE.

Value

A data.frame with initial columns the marker genotypes, then the phenotype data, then a column indicating whether any of the marker genotypes were inferred (1=at least one genotype inferred, 0=none were inferred).

Author(s)

- Karl W Broman, ⟨kbroman@biostat.wisc.edu⟩; Brian Yandell

See Also

- `find.marker`, `effectplot`, `find.flanking`, `effectscan`

Examples

data(listeria)
mname <- find.marker(listeria, 5, 28) # marker D5M357
plot.pxg(listeria, mname)

mname2 <- find.marker(listeria, 13, 26) # marker D13Mit147
plot.pxg(listeria, c(mname, mname2))
plot.pxg(listeria, c(mname2, mname))

data(fake.f2)
mname <- find.marker(fake.f2, 1, 37) # marker D1M437
plot.pxg(fake.f2, mname)
mname2 <- find.marker(fake.f2, "X", 14) # marker DXM66
plot.pxg(fake.f2, mname2)
plot.pxg(fake.f2, c(mname,mname2))
plot.pxg(fake.f2, c(mname2,mname))

---

plot.rf  

**Plot recombination fractions**

**Description**
Plot a grid showing the recombination fractions for all pairs of markers, and/or the LOD scores for tests of linkage between pairs of markers.

**Usage**

```r
plot.rf(x, chr, what=c("both","lod","rf"), alternate.chrid=FALSE,
...)
```

**Arguments**

- `x`  
  An object of class `cross`. See `read.cross` for details.

- `chr`  
  Vector specifying which chromosomes to plot (optional)

- `what`  
  Indicate whether to plot LOD scores, recombination fractions or both.

- `alternate.chrid`  
  If TRUE and more than one chromosome is plotted, alternate the placement of chromosome axis labels, so that they may be more easily distinguished.

- `...`  
  Ignored at this point.

**Details**

Uses `image` to plot a grid showing the recombination fractions and/or LOD scores for all pairs of markers. (The LOD scores are for a test of \( r = 1/2 \).) If both are plotted, the recombination fractions are in the upper left triangle while the LOD scores are in the lower right triangle. Red corresponds to a large LOD or a small recombination fraction, while blue is the reverse. Note that missing values appear in light gray.

Recombination fractions are transformed by \(-4(\log_2 r + 1)\) to make them on the same sort of scale as LOD scores. Values of LOD or the transformed recombination fraction that are above 12 are set to 12.

**Value**

None.
Description
Plot the LOD curve for a genome scan with a single-QTL model (the output of `scanone`).

Usage
```r
## S3 method for class 'scanone':
plot(x, x2, x3, chr, lodcolumn=1, incl.markers=TRUE, 
     xlim, ylim, lty=1, col=c("black","blue","red"), lwd=2, 
     add=FALSE, gap=25, mtick = c("line", "triangle"), 
     show.marker.names=FALSE, alternate.chrid=FALSE, ...)
```

Arguments
- `x`: An object of class "scanone", as output by `scanone`.
- `x2`: Optional second `scanone` object.
- `x3`: Optional third `scanone` object.
- `chr`: Optional vector specifying which chromosomes to plot. (The chromosomes must be specified by name.)
- `lodcolumn`: An integer, or vector of 3 integers, indicating which of the LOD score columns should be plotted (generally this is 1).
- `incl.markers`: Indicate whether to plot line segments at the marker locations.
xlim Limits for x-axis (optional).
ylim Limits for y-axis (optional).
lty Line types; a vector of length 1 or 3.
col Line colors; a vector of length 1 or 3.
lwd Line widths; a vector of length 1 or 3.
add If TRUE, add to a current plot.
gap Gap separating chromosomes (in cM).
mtick Tick mark type for markers (line segments or upward-pointing triangles).
show.marker.names If TRUE, show the marker names along the x axis.
alternate.chrid If TRUE and more than one chromosome is plotted, alternate the placement of chromosome axis labels, so that they may be more easily distinguished.

Details
This function allows you to plot the results of up to three genome scans against one another. Such objects must conform with each other.
One may alternatively use the argument add to add the plot of an additional genome scan to the current figure, but some care is required: the same chromosomes should be selected, and the results must concern crosses with the same genetic maps.
If a single scanone object containing multiple LOD score columns (for example, from different phenotypes) is input, up to three LOD curves may be plotted, by providing a vector in the argument lodcolumn. If multiple scanone objects are input (via x, x2 and x3), the LOD score columns to be plotted are chosen from the corresponding element of the lodcolumn argument.

Value
None.

Author(s)
Karl W Broman, ⟨kbroman@biostat.wisc.edu⟩

See Also

scanone, summary.scanone, par, colors

Examples

data(fake.f2)

fake.f2 <- calc.genoprob(fake.f2, step=2.5)
out.mr <- scanone(fake.f2, method="mr")
out.em <- scanone(fake.f2, method="em")
plot(out.mr)
plot.out.mr <- scanone(fake.f2, method="mr")
plot(out.mr, chr=c(1,13), lty=1, col=c("violetred","black"))

out.hk <- scanone(fake.f2, method="hk")
plot(out.hk, chr=c(1,13), add=TRUE, col="slateblue")
plot(out.hk, chr=13, show.marker.names=TRUE)

---

# Plot LOD scores for a two-dimensional genome scan

**Description**

Plot the results of a two-dimensional, two-QTL genome scan.

**Usage**

```r
## S3 method for class 'scantwo':
plot(x, chr, incl.markers=FALSE, zlim, lodcolumn=1,
     lower = c("full", "add", "cond-int", "cond-add", "int"),
     upper = c("int", "cond-add", "cond-int", "add", "full"),
     nodiag=TRUE, contours=FALSE, main, zscale=TRUE, point.at.max=FALSE,
     col.scheme = c("redblue","cm","gray","heat","terrain","topo"),
     gamma=0.6, allow.neg=FALSE, alternate.chrid=FALSE, ...)
```

**Arguments**

- `x`: An object of class "scantwo", as output by `scantwo`
- `chr`: Optional vector specifying which chromosomes to plot.
- `incl.markers`: If FALSE, plot LOD scores on an evenly spaced grid (not including the results at the markers).
- `zlim`: A vector of length 2 (optional), indicating the z limits for the full LOD and the epistasis LOD scores, respectively. If missing, the maximum limits are used for each.
- `lodcolumn`: If the scantwo results contain LOD scores for multiple phenotypes, this argument indicates which to use in the plot.
- `lower`: Indicates which LOD scores should be plotted in the lower triangle. See the details below.
- `upper`: Indicates which LOD scores should be plotted in the upper triangle. See the details below.
- `nodiag`: If TRUE, suppress the plot of the scanone output (which is normally along the diagonal.)
- `contours`: If TRUE, add a contour to the plot at 1.5-LOD below its maximum, using a call to `contour`. If a numeric vector, contours are drawn at these values below the maximum LOD.
- `main`: An optional title for the plot.
plot.scantwo

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>zscale</td>
<td>If TRUE, a color scale is plotted at the right.</td>
</tr>
<tr>
<td>point.at.max</td>
<td>If TRUE, plot an X at the maximum LOD.</td>
</tr>
<tr>
<td>col.scheme</td>
<td>Name of color pallet.</td>
</tr>
<tr>
<td>gamma</td>
<td>Parameter affecting range of colors when col.scheme=&quot;gray&quot; or=&quot;redblue&quot;.</td>
</tr>
<tr>
<td>allow.neg</td>
<td>If TRUE, allow the plot of negative LOD scores; in this case, the z-limits are symmetric about 0. This option is chiefly to allow a plot of difference between LOD scores from different methods, calculated via -.scantwo.</td>
</tr>
<tr>
<td>alternate.chrid</td>
<td>If TRUE and more than one chromosome is plotted, alternate the placement of chromosome axis labels, so that they may be more easily distinguished.</td>
</tr>
</tbody>
</table>

... Ignored at this point.

Details

Uses `image` to plot a grid of LOD scores. The particular LOD scores plotted in the upper-left and lower-right triangles are selected via `upper` and `lower`, respectively. By default, the upper-left triangle contains the epistasis LOD scores ("int"), and the lower-right triangle contains the LOD scores for the full model ("full"). The diagonal contains either all zeros or the main effects LOD scores (from `scanone`).

The `scantwo` function calculates, for each pair of putative QTLs, \((q_1, q_2)\), the likelihood undering the null model \(L_0\), the likelihood under each of the single-QTL models, \(L(q_1)\) and \(L(q_2)\), the likelihood under an additive QTL model, \(L_a(q_1, q_2)\), and the likelihood under a full QTL model (including QTL-QTL interaction), \(L_f(q_1, q_2)\).

The five possible LOD scores that may be plotted are the following. The epistasis LOD scores ("int") are \(LOD_i = \log_{10} L_f(q_1, q_2) - \log_{10} L_a(q_1, q_2)\).

The full LOD scores ("full") are \(LOD_f = \log_{10} L_f(q_1, q_2) - \log_{10} L_0\).

The additive LOD scores ("add") are \(LOD_a = \log_{10} L_a(q_1, q_2) - \log_{10} L_0\).

In addition, we may calculate, for each pair of chromosomes, the difference between the full LOD score and the maximum single-QTL LOD scores for that pair of chromosomes ("cond-int").

Finally, we may calculate, for each pair of chromosomes, the difference between the additive LOD score and the maximum single-QTL LOD scores for that pair of chromosomes ("cond-add").

If a color scale is plotted `(zscale=TRUE)`, the axis on the left indicates the scale for the upper-left triangle, while the axis on the right indicates the scale for the lower-right triangle. Note that the axis labels can get screwed up if you change the size of the figure window; you’ll need to redo the plot.

Value

None.

Author(s)

Hao Wu; Karl W Broman, ⟨kbroman@biostat.wisc.edu⟩; Brian Yandell

See Also

`scantwo`, `summary.scantwo`, `plot.scanone`, `-.scantwo`
Examples

data(hyper)

hyper <- calc.genoprob(hyper, step=5)

# 2-d scan by EM and by Haley-Knott regression
out2.em <- scantwo(hyper, method="em")
out2.hk <- scantwo(hyper, method="hk")

# plot epistasis and full LOD scores
plot(out2.em)

# plot cond-int in upper triangle and full in lower triangle
# for chromosomes 1, 4, 6, 15
plot(out2.em, upper="cond-int", chr=c(1,4,6,15))

# plot cond-add in upper triangle and add in lower triangle
# for chromosomes 1, 4
plot(out2.em, upper="cond-add", lower="add", chr=c(1,4))

# plot the differences between the LOD scores from Haley-Knott
# regression and the EM algorithm
plot(out2.hk - out2.em, allow.neg=TRUE)

---

pull.geno  

Pull out the genotype data from a cross

Description

Pull out the genotype data from a cross object, as a single big matrix.

Usage

pull.geno(cross, chr)

Arguments

cross  An object of class cross. See read.cross for details.
chr  An optional vector specifying which chromosomes to keep or discard. This may be a logical vector, a numeric vector, or a vector of character strings.

Value

A matrix of size n.ind x tot.mar. The raw genotype data in the input cross object, with the chromosomes pasted together.
pull.map

Pull out the genetic map from a cross

Description

Pull out the map portion of a cross object.

Usage

pull.map(cross, chr)

Arguments

cross An object of class cross. See read.cross for details.
chr An optional vector specifying which chromosomes to keep or discard. This may be a logical vector, a numeric vector, or a vector of character strings.

Value

The genetic map: a list with each component containing the marker positions (in cM) for a chromosome. Each component has class A or X according to whether it is an autosome or the X chromosome. The components are either vectors of marker positions or, for a sex-specific map, 2-row matrices containing the female and male marker locations. The map itself is given class map.

Author(s)

Karl W Broman, (kbroman@biostat.wisc.edu)

See Also

replace.map, plot.map
Examples

```r
data(fake.f2)
map <- pull.map(fake.f2)
plot(map)
```

Internal qtl functions

Description

Internal qtl functions. These are generally not to be called by the user.

Usage

```r
addqtl(cross, qtl, add.chr, add.pos, add.name, map)
adjust.rf.ri(r, type=c("self","sib"), chrtype=c("A","X"), expand=TRUE)
calc.genoprob.special(cross, error.prob=0.0001,
  map.function=c("haldane","kosambi","c-f","morgan"))
calc.pairprob(cross, step=0, off.end=0, error.prob=0.0001,
  map.function=c("haldane","kosambi","c-f","morgan"), map)
checkcovar(cross, pheno.col, addcovar, intcovar, perm.strata, verbose=TRUE)
clean(object)
condense(object)
convert(object)
create.map(map, step, off.end, stepwidth=c("fixed", "variable"))
discan(cross, pheno, method=c("em","mr"),
  addcovar=NULL, intcovar=NULL, maxit=4000, tol=1e-4,
  verbose=FALSE, give.warnings=TRUE)
dropqtl(qtl, drop)
dropXcol(type=c("f2","bc"), sexpgm, cross.attr)
fitstahl.estp(cross, error.prob=0.0001, m=0, tol=1e-4, maxit=4000)
fitstahl.estp.sub(p, cross, error.prob=0.0001, m=0, thetol=1e-4, maxit=4000)
fitstahl.este(cross, m=0, p=0, tol=1e-4, maxit=4000)
fixX4write(geno,sex,pgm,crosstype)
fixXgeno.bc(cross)
fixXgeno.f2(cross)
forwsel(x, y, maxsize=7)
genotab.em(dat, tol=1e-6, maxit=10000, verbose=FALSE)
getid(cross)
getsex(cross)
getgenonames(type=c("f2","bc","riself","risib","4way"),
  chrtype=c("A","X"), expandX=c("simple","standard","full"),
  sexpgm, cross.attr)
grab.arg.names(...)
```
imf.cf(r)
imf.h(r)
imf.k(r)
imf.m(r)
imf.stahl(r, m=0, p=0, tol=1e-12, maxit=1000)
locatemarker(map, pos, chr, flag)
locate.xo(cross)
makeSSmap(cross)
## S3 method for class 'scantwocondensed':
max(object, lodcolumn=1,
    what=c("best", "full", "add", "int"),
    df=FALSE, na.rm=TRUE, ...)
mf.cf(d)
mf.h(d)
mf.k(d)
mf.m(d)
mf.stahl(d, m=0, p=0)
parseformula(formula, qtl.dimname, covar.dimname)
## S3 method for class 'effectscan':
plot(x, gap=25, ylim, mtick=c("line","triangle"),
    add.legend=TRUE, alternate.chrid=FALSE, ...)
## S3 method for class 'cross':
print(x, ...)
## S3 method for class 'map':
print(x, ...)
## S3 method for class 'qtl':
print(x, ...)
## S3 method for class 'scanoneboot':
print(x, ...)
## S3 method for class 'scantwo':
print(x, ...)
## S3 method for class 'summary.cross':
print(x, ...)
## S3 method for class 'summary.fitqtl':
print(x, ...)
## S3 method for class 'summary.map':
print(x, ...)
## S3 method for class 'summary.qtl':
print(x, ...)
## S3 method for class 'summary.ripple':
print(x, ...)
## S3 method for class 'summary.scanone':
print(x, ...)
## S3 method for class 'summary.scanoneperm':
print(x, ...)
## S3 method for class 'summary.scantwo':
print(x, ...)
## S3 method for class 'summary.scantwo.old':
print(x, ...)  
## S3 method for class 'summary.scantwoperm': print(x, ...)  
read.cro.qticart(file)  
read.cross.csv(dir, file, na.strings=c("-","NA"),  
genotypes=c("A","H","B","D","C"),  
estimate.map=TRUE, rotate=FALSE, ...)  
read.cross.csvs(dir, genfile, phefile, na.strings=c("-","NA"),  
genotypes=c("A","H","B","D","C"),  
estimate.map=TRUE, rotate=FALSE, ...)  
read.cross.gary(dir, genfile, mnamesfile, chridfile, phefile,  
pnamesfile, mapfile, estimate.map, na.strings)  
read.cross.karl(dir, genfile, mapfile, phefile)  
read.cross.mm(dir, rawfile, mapfile, estimate.map=TRUE)  
read.cross.qtlicart(dir, crofile, mapfile)  
read.map.qtlicart(file)  
read.maps.mm(mapfile)  
replaceqtl(cross, qtl, replace, by.chr, by.pos, by.name, map)  
revisecovar(sexpgm, covar)  
revisescantwodf(df)  
revisexdata(type=c("f2","bc"), expandX=c("simple","standard","full"),  
sexpgm, geno, prob, draws, pairprob, cross.attr)  
ripple.perm1(n)  
ripple.perm2(n)  
ripple.perm.sub(x, mat)  
scanone.perm(cross, pheno.col=1, model=c("normal","binary","2part","np"),  
method=c("em","imp","hk","ehk","mr","mr-imp","mr-argmax"),  
addcovar=NULL, intcovar=NULL, weights=NULL,  
use=c("all.obs","complete.obs"), upper=FALSE,  
ties.random=FALSE, start=NULL, maxit=4000,  
tol=1e-4, n.perm=1000, perm.Xsp=FALSE, perm.strata=NULL,  
verbose=TRUE)  
scanone.perm.engine(n.perm, cross, pheno.col, model,  
method, addcovar, intcovar, weights, use,  
upper, ties.random, start, maxit, tol,  
verbose, perm.strata)  
scanoneXnull(type, sexpgm)  
scantwo.perm(cross, pheno.col=1, model=c("normal","binary"),  
method=c("em","imp","hk","mr","mr-imp","mr-argmax"),  
addcovar=NULL, intcovar=NULL, weights=NULL,  
use=c("all.obs","complete.obs"),  
incl.markers=FALSE, clean.output=FALSE,  
maxit=4000, tol=1e-4, verbose=FALSE,  
n.perm=1000, perm.strata)  
scantwo.perm.engine(n.perm, cross, pheno.col, model,  
method, addcovar, intcovar, weights, use,  
incl.markers, clean.output, maxit, tol, verbose,
qtlversion

qtlversion

Description

Print the version number of the currently installed version of R/qtl.

Usage

qtlversion()

Value

A character string with the version number of the currently installed version of R/qtl.
Author(s)  
Karl W Broman, (kbroman@biostat.wisc.edu)

See Also  
version

Examples  
qtlversion()

---

read.cross  
Read data for a QTL experiment

Description  
Data for a QTL experiment is read from a set of files and converted into an object of class cross. The comma-delimited format (csv) is recommended. All formats require chromosome assignments for the genetic markers, and assume that markers are in their correct order.

Usage  
```r
read.cross(format=c("csv", "csvr", "csvs", "csvsr", "mm", "qtx", "qtlcart", "gary", "karl"),
  dir="", file, genfile, mapfile, phefile, chridfile,
  mnamesfile, pnamesfile, na.strings=c("-", "NA"),
  genotypes=c("A", "H", "B", "D", "C"), alleles=c("A", "B"),
  estimate.map=TRUE, convertXdata=TRUE, ...)
```

Arguments  
- `format` Specifies the format of the data.
- `dir` Directory in which the data files will be found. In Windows, use forward slashes ("/") or double backslashes ("\\") to specify directory trees.
- `file` The main input file for formats csv, csvr and mm.
- `genfile` File with genotype data (formats csvs, csvsr, karl and gary only).
- `mapfile` File with genotype data (formats csvs, csvsr, karl and gary only).
- `chridfile` File with chromosome ID for each marker (gary format only).
- `mnamesfile` File with marker names (gary format only).
- `pnamesfile` File with phenotype names (gary format only).
- `na.strings` A vector of strings which are to be interpreted as missing values (csv and gary formats only). For the csv formats, these are interpreted globally for the entire file, so missing value codes in phenotypes must not be valid genotypes, and vice versa. For the gary format, these are used only for the phenotype data.
genotypes A vector of character strings specifying the genotype codes (csv formats only). Generally this is a vector of length 5, with the elements corresponding to AA, AB, BB, not BB (i.e., AA or AB), and not AA (i.e., AB or BB). Note: Pay careful attention to the third and fourth of these; the order of these can be confusing. If you are trying to read 4-way cross data, your file must have genotypes coded as described below, and you need to set genotypes=NULL so that no re-coding gets done.

alleles A vector of two one-letter character strings (or four, for the four-way cross), to be used as labels for the two alleles.

estimate.map For all formats but qtlcart and karl: if TRUE and marker positions are not included in the input files, the genetic map is estimated using the function est.map.

convertXdata If TRUE, any X chromosome genotype data is converted to the internal standard, using columns sex and pgm in the phenotype data if they available or by inference if they are not. If FALSE, the X chromosome data is read as is.

Additional arguments, passed to the function read.table in the case of csv and csvr formats. In particular, one may use the argument sep to specify the field separator (the default is a comma) and dec to specify the character used for the decimal point (the default is a period).

Details
The available formats are comma-delimited (csv), rotated comma-delimited (csvr), comma-delimited with separate files for genotype and phenotype data (csvs), rotated comma-delimited with separate files for genotype and phenotype data (csvsr), Mapmaker (mm), Map Manager QTX (qtx), Gary Churchill’s format (gary) and Karl Broman’s format (karl). The required files and their specification for each format appears below. The comma-delimited formats are recommended. Note that most of these formats work only for backcross and intercross data.

The sampledata directory in the package distribution contains sample data files in all formats except Gary’s.

Value
An object of class cross, which is a list with two components:

geno
This is a list with elements corresponding to chromosomes. names(geno) contains the names of the chromosomes. Each chromosome is itself a list, and is given class A or X according to whether it is autosomal or the X chromosome. There are two components for each chromosome: data, a matrix whose rows are individuals and whose columns are markers, and map, either a vector of marker positions (in cM) or a matrix of dim (2 x n.mar) where the rows correspond to marker positions in female and male genetic distance, respectively. The genotype data for a backcross is coded as follows: NA = missing, 1 = AA, 2 = AB.

For an F2 intercross, the coding is NA = missing, 1 = AA, 2 = AB, 3 = BB, 4 = not BB (i.e. AA or AB; D in Mapmaker/qtl), 5 = not AA (i.e. AB or BB; C in Mapmaker/qtl).
For a 4-way cross, the mother and father are assumed to have genotypes AB and CD, respectively. The genotype data for the progeny is assumed to be phase-known, with the following coding scheme: NA = missing, 1 = AC, 2 = BC, 3 = AD, 4 = BD, 5 = A = AC or AD, 6 = B = BC or BD, 7 = C = AC or BC, 8 = D = AD or BD, 9 = AC or BD, 10 = AD or BC.

The genotypes for the X chromosome require special care!
The X chromosome should be given chromosome identifier X or x. If it is labeled by a number or by Xchr, it will be interpreted as an autosome.

The phenotype data should contain a column named "sex" which indicates the sex of each individual, either coded as 0=female and 1=male, or as a factor with levels female/male or f/m. Case will be ignored both in the name and in the factor levels. If no such phenotype column is included, it will be assumed that all individuals are of the same sex.

In the case of an intercross, the phenotype data may also contain a column named "pgm" (for "paternal grandmother") indicating the direction of the cross. It should be coded as 0/1 with 0 indicating the cross (AxB)x(AxB) or (BxA)x(AxB) and 1 indicating the cross (AxB)x(BxA) or (BxA)x(BxA). If no such phenotype column is included, it will be assumed that all individuals come from the same direction of cross.

The internal storage of X chromosome data is quite different from that of autosomal data. Males are coded 1=AA and 2=BB; females with pgm==0 are coded 1=AA and 2=AB; and females with pgm==1 are coded 1=BB and 2=AB. If the argument convertXdata is TRUE, conversion to this format is made automatically; if FALSE, no conversion is done, summary.cross will likely return a warning, and most analyses will not work properly.

Use of convertXdata=FALSE (in which case the X chromosome genotypes will not be converted to our internal standard) can be useful for diagnosing problems in the data, but will require some serious mucking about in the internal data structure.

CSV format
The input file is a comma-delimited text file. A different field separator may be specified via the argument sep, which will be passed to the function read.table. For example, in Europe, it is common to use a comma in place of the decimal point in numbers and a semi-colon in place of a comma as the field separator; such data may be read by using sep="; " and dec=" , ".
The first line should contain the phenotype names followed by the marker names. At least one phenotype must be included: for example, include a numerical index for each individual.
The second line should contain blanks in the phenotype columns, followed by chromosome identifiers for each marker in all other columns. If a chromosome has the identifier X or x, it is assumed to be the X chromosome; otherwise, it is assumed to be an autosome.

An optional third line should contain blanks in the phenotype columns, followed by marker positions, in cM.
Marker order is taken from the cM positions, if provided; otherwise, it is taken from the column order.

Subsequent lines should give the data, with one line for each individual, and with phenotypes followed by genotypes. If possible, phenotypes are made numeric; otherwise they are converted to factors.

The cross is determined to be a backcross if only the first two elements of the genotypes string are found; otherwise, it is assumed to be an intercross.

**CSVr format**

This is just like the csv format, but rotated (or really transposed), so that rows are columns and columns are rows.

**CSVs format**

This is like the csv format, but with separate files for the genotype and phenotype data.

The first column in the genotype data must specify individuals’ identifiers, and there must be a column in the phenotype data with precisely the same information, and the individuals must be in precisely the same order in the two files. These IDs will be included in the data as a phenotype. If the name id or ID is used, these identifiers will be used in top.errorlod, plot.errorlod, and plot.geno as identifiers for the individual.

In the genotype data file, the second row gives the chromosome IDs. The cell in the second row, first column, must be blank. A third row giving cM positions of markers may be included, in which case the cell in the third row, first column, must be blank.

There need be no blank rows in the phenotype data file.

**CSVsr format**

This is just like the csvs format, but with each file rotated (or really transposed), so that rows are columns and columns are rows.

**Mapmaker format**

This format requires two files. The so-called rawfile, specified by the argument file, contains the genotype and phenotype data. Rows beginning with the symbol # are ignored. The first line should be either data type f2 intercross or data type f2 backcross. The second line should begin with three numbers indicating the numbers of individuals, markers and phenotypes in the file. This line may include the word symbols followed by symbol assignments (see the documentation for mapmaker, and cross your fingers). The rest of the lines give genotype data followed by phenotype data, with marker and phenotype names always beginning with the * symbol.

A second file contains the genetic map information, specified with the argument mapfile. The map file may be in one of two formats. The function will determine which format of map file is presented.

The simplest format for the map file is not standard for the Mapmaker software, but is easy to create. The file contains two or three columns separated by white space and with no header row. The first column gives the chromosome assignments. The second column gives the marker names,
with markers listed in the order along the chromosomes. An optional third column lists the map positions of the markers.

Another possible format for the map file is the .maps format, which is produced by Mapmaker. The code for reading this format was written by Brian Yandell.

Marker order is taken from the map file, either by the order they are presented or by the cM positions, if specified.

Map Manager QTX format

This format requires a single file (that produced by the Map Manager QTX program).

QTL Cartographer format

This format requires two files: the .cro and .map files for QTL Cartographer (produced by the QTL Cartographer sub-program, Rmap and Rcross).

Note that the QTL Cartographer cross types are converted as follows: RF1 to riself, RF2 to risib, RF0 (doubled haploids) to bc, B1 or B2 to bc, RF2 or SF2 to f2.

Gary format

This format requires the six files. All files have default names, and so the file names need not be specified if the default names are used.

`genfile` (default = "geno.dat") contains the genotype data. The file contains one line per individual, with genotypes for the set of markers separated by white space. Missing values are coded as 9, and genotypes are coded as 0/1/2 for AA/AB/BB.

`mapfile` (default = "markerpos.txt") contains two columns with no header row: the marker names in the first column and their cM position in the second column. If marker positions are not available, use `mapfile=NULL`, and a dummy map will be inserted.

`phefile` (default = "pheno.dat") contains the phenotype data, with one row for each mouse and one column for each phenotype. There should be no header row, and missing values are coded as "-".

`chridfile` (default = "chrid.dat") contains the chromosome identifier for each marker.

`mnamesfile` (default = "mnames.txt") contains the marker names.

`pnamesfile` (default = "pnames.txt") contains the names of the phenotypes. If phenotype names file is not available, use `pnamesfile=NULL`; arbitrary phenotype names will then be assigned.

Karl format

This format requires three files; all files have default names, and so need not be specified if the default name is used.

`genfile` (default = "gen.txt") contains the genotype data. The file contains one line per individual, with genotypes separated by white space. Missing values are coded 0; genotypes are coded as 1/2/3/4/5 for AA/AB/BB/not BB/not AA.

`mapfile` (default = "map.txt") contains the map information, in the following complicated format:
n.chr
n.mar(1) rf(1,1) rf(1,2) ... rf(1,n.mar(1)-1)
mar.name(1,1)
mar.name(1,2)
...
mar.name(1,n.mar(1))
n.mar(2)
...
etc.

phefile (default = "phe.txt") contains a matrix of phenotypes, with one individual per line. The first line in the file should give the phenotype names.

Author(s)

Karl W Broman, (kbroman@biostat.wisc.edu); Brian S. Yandell

See Also

write.cross, sim.cross; the sampledata directory in the package distribution contains sample data files in all formats except Gary’s. Also see http://www.rqtl.org/sampledata.

Examples

## Not run:
# CSV format
dat1 <- read.cross("csv", dir="Mydata", file="mydata.csv")

# CSVS format
dat2 <- read.cross("csvs", dir="Mydata", genfile="mydata_gen.csv", phefile="mydata_phe.csv")

# Mapmaker format
dat3 <- read.cross("mm", dir="Mydata", file="mydata.raw", mapfile="mydata.map")

# Map Manager QTX format
dat4 <- read.cross("qtx", dir="Mydata", file="mydata.qtx")

# QTL Cartographer format
dat5 <- read.cross("qtlcart", dir="Mydata", file="qtlcart.cro", mapfile="qtlcart.map")

# Gary format
dat6 <- read.cross("gary", dir="Mydata", genfile="geno.dat", mapfile="markerpos.txt", phefile="pheno.dat", chridfile="chrid.dat", mnamesfile="mnames.txt", pnamesfile="pnames.txt")

# Karl format
**replace.map**

*Replace the genetic map of a cross*

**Description**

Replace the map portion of a cross object.

**Usage**

```r
replace.map(cross, map)
```

**Arguments**

- `cross`: An object of class `cross`. See `read.cross` for details.
- `map`: A list containing the new genetic map. This must be the same length and with the same marker names as that contained in `cross`.

**Value**

The input `cross` object with the genetic map replaced by the input `map`.

**Author(s)**

Karl W Broman, (kbroman@biostat.wisc.edu)

**See Also**

`pull.map`, `est.map`

**Examples**

```r
data(fake.f2)

newmap <- est.map(fake.f2)
plot.map(fake.f2, newmap)
fake.f2 <- replace.map(fake.f2, newmap)
```
### ripple

**Description**

Investigate different marker orders for a given chromosome, comparing all possible permutations of a sliding window of markers.

**Usage**

```r
ripple(cross, chr, window=4, method=c("countxo","likelihood"),
       error.prob=0.0001, map.function=c("haldane","kosambi","c-f","morgan"),
       maxit=4000, tol=1e-4, sex.sp=TRUE, verbose=TRUE)
```

**Arguments**

- `cross`: An object of class `cross`. See `read.cross` for details.
- `chr`: The chromosome to investigate. Only one chromosome is allowed.
- `window`: Number of markers to include in the sliding window of permuted markers. Larger numbers result in the comparison of a greater number of marker orders, but will require a considerable increase in computation time.
- `method`: Indicates whether to compare orders by counting the number of obligate crossovers, or by a likelihood analysis.
- `error.prob`: Assumed genotyping error rate used in the calculation of the penetrance \( \text{Pr} (\text{observed genotype} \mid \text{true genotype}) \).
- `map.function`: Indicates whether to use the Haldane, Kosambi, Carter-Falconer, or Morgan map function when converting genetic distances into recombination fractions.
- `maxit`: Maximum number of EM iterations to perform.
- `tol`: Tolerance for determining convergence.
- `sex.sp`: Indicates whether to estimate sex-specific maps; this is used only for the 4-way cross.
- `verbose`: If TRUE, information about the number of orders (and, if `method="likelihood"`, about progress) are printed.

**Details**

For `method="likelihood"`, calculations are done by first constructing a matrix of marker orders and then making repeated calls to the R function `est.map`. Of course, it would be faster to do everything within C, but this was a lot easier to code.

For `method="countxo"`, calculations are done within C.
Value

A matrix, given class "ripple"; the first set of columns are marker indices describing the order. In the case of method="countxo", the last column is the number of obligate crossovers for each particular order. In the case of method="likelihood", the last two columns are LOD scores (log base 10 likelihood ratios) comparing each order to the initial order and the estimated chromosome length for the given order. Positive LOD scores indicate that the alternate order has more support than the original.

Author(s)

Karl W Broman, ⟨kbroman@biostat.wisc.edu⟩

See Also

summary.ripple, switch.order, est.map, est.rf

Examples

data(badorder)
ripl <- ripple(badorder, chr=1, window=3)
summary(ripl)

## Not run:
ripl <- ripple(badorder, chr=1, window=2, method="likelihood")
summary(ripl)
## End(Not run)

badorder <- switch.order(badorder, 1, ripl[2,])

---

Genome scan with a single QTL model

Description

Genome scan with a single QTL model, with possible allowance for covariates, using any of several possible models for the phenotype and any of several possible numerical methods.

Usage

scanone(cross, chr, pheno.col=1, model=c("normal","binary","2part","np"),
method=c("em","imp","hk","ehk","mr","mr-imp","mr-argmax"),
addcovar=NULL, intcovar=NULL, weights=NULL,
use=c("all.obs", "complete.obs"), upper=FALSE,
ties.random=FALSE, start=NULL, maxit=4000,
tol=1e-4, n.perm, perm.Xsp=FALSE, perm.strata=NULL, verbose)
Arguments

- **cross**: An object of class `cross`. See `read.cross` for details.
- **chr**: Optional vector indicating the chromosomes for which LOD scores should be calculated.
- **pheno.col**: Column number in the phenotype matrix which should be used as the phenotype. This can be a vector of integers; for methods "hk" and "imp" this can be considerably faster than doing them one at a time.
- **model**: The phenotypic model: the usual normal model, a model for binary traits, a two-part model or non-parametric analysis.
- **method**: Indicates whether to use the EM algorithm, imputation, Haley-Knott regression, the extended Haley-Knott method, or marker regression. Not all methods are available for all models. Marker regression is performed either by dropping individuals with missing genotypes ("mr"), or by first filling in missing data using a single imputation ("mr-imp") or by the Viterbi algorithm ("mr-argmax").
- **addcovar**: Additive covariates; allowed only for the normal and binary models.
- **intcovar**: Interactive covariates (interact with QTL genotype); allowed only for the normal and binary models.
- **weights**: Optional weights of individuals. Should be either NULL or a vector of length n.ind containing positive weights. Used only in the case `model="normal"`.
- **use**: In the case that multiple phenotypes are selected to be scanned, this argument indicates whether to use all individuals, including those missing some phenotypes, or just those individuals that have data on all selected phenotypes.
- **upper**: Used only for the two-part model; if true, the "undefined" phenotype is the maximum observed phenotype; otherwise, it is the smallest observed phenotype.
- **ties.random**: Used only for the non-parametric "model"; if TRUE, ties in the phenotypes are ranked at random. If FALSE, average ranks are used and a corrected LOD score is calculated.
- **start**: Used only for the EM algorithm with the normal model and no covariates. If NULL, use the usual starting values; if length 1, use random initial weights for EM; otherwise, this should be a vector of length n+1 (where n is the number of possible genotypes for the cross), giving the initial values for EM.
- **maxit**: Maximum number of iterations for methods "em" and "ehk".
- **tol**: Tolerance value for determining convergence for methods "em" and "ehk".
- **n.perm**: If specified, a permutation test is performed rather than an analysis of the observed data. This argument defines the number of permutation replicates.
- **perm.Xsp**: If n.perm > 0, so that a permutation test will be performed, this indicates whether separate permutations should be performed for the autosomes and the X chromosome, in order to get an X-chromosome-specific LOD threshold. In this case, additional permutations are performed for the X chromosome.
- **perm.strata**: If n.perm > 0, this may be used to perform a stratified permutation test. This should be a vector with the same number of individuals as in the cross data. Unique values indicate the individual strata, and permutations will be performed within the strata.
- **verbose**: In the case n.perm is specified, display information about the progress of the permutation tests.
Details

Use of the EM algorithm, Haley-Knott regression, and the extended Haley-Knott method require that multipoint genotype probabilities are first calculated using \texttt{calc.genoprob}. The imputation method uses the results of \texttt{sim.geno}.

Individuals with missing phenotypes are dropped.

In the case that \texttt{n.perm}>0, so that a permutation test is performed, the R function \texttt{scanone} is called repeatedly. If \texttt{perm.Xsp=TRUE}, separate permutations are performed for the autosomes and the X chromosome, so that an X-chromosome-specific threshold may be calculated. In this case, \texttt{n.perm} specifies the number of permutations used for the autosomes; for the X chromosome, \texttt{n.perm} \times L_A/L_X permutations will be run, where \(L_A\) and \(L_X\) are the total genetic lengths of the autosomes and X chromosome, respectively. More permutations are needed for the X chromosome in order to obtain thresholds of similar accuracy.

For further details on the models, the methods and the use of covariates, see below.

Value

If \texttt{n.perm} is missing, the function returns a data.frame whose first two columns contain the chromosome IDs and cM positions. Subsequent columns contain the LOD scores for each phenotype. In the case of the two-part model, there are three LOD score columns for each phenotype: LOD(\(p, \mu\)), LOD(\(p\)) and LOD(\(\mu\)). The result is given class "\texttt{scanone}" and has attributes "\texttt{model}", "\texttt{method}", "\texttt{df}" and "\texttt{type}" (the latter is the type of cross analyzed).

If \texttt{n.perm} is specified, the function returns the results of a permutation test and the output has class "\texttt{scanoneperm}". If \texttt{perm.Xsp=FALSE}, the function returns a matrix with \texttt{n.perm} rows, each row containing the genome-wide maximum LOD score for each of the phenotypes. In the case of the two-part model, there are three columns for each phenotype, corresponding to the three different LOD scores. If \texttt{perm.Xsp=TRUE}, the result contains separate permutation results for the autosomes and the X chromosome respectively, and an attribute indicates the lengths of the chromosomes and an indicator of which chromosome is X.

Models

The \textbf{normal model} is the standard model for QTL mapping (see Lander and Botstein 1989). The residual phenotypic variation is assumed to follow a normal distribution, and analysis is analogous to analysis of variance.

The \textbf{binary model} is for the case of a binary phenotype, which must have values 0 and 1. The proportions of 1’s in the different genotype groups are compared. Currently only methods \texttt{em} and \texttt{mr} are available for this model. See Xu and Atchley (1996) and Broman (2003).

The \textbf{two-part model} is appropriate for the case of a spike in the phenotype distribution (for example, metastatic density when many individuals show no metastasis, or survival time following an infection when individuals may recover from the infection and fail to die). The two-part model was described by Boyartchuk et al. (2001) and Broman (2003). Individuals with QTL genotype \(g\) have probability \(p_g\) of having an undefined phenotype (the spike), while if their phenotype is defined, it comes from a normal distribution with mean \(\mu_g\) and common standard deviation \(\sigma\). Three LOD scores are calculated: LOD(\(p, \mu\)) is for the test of the hypothesis that \(p_g = p\) and \(\mu_g = \mu\). LOD(\(p\)) is for the test that \(p_g = p\) while the \(\mu_g\) may vary. LOD(\(\mu\)) is for the test that \(\mu_g = \mu\) while the \(p_g\) may vary.
With the non-parametric "model", an extension of the Kruskal-Wallis test is used; this is similar to the method described by Kruglyak and Lander (1995). In the case of incomplete genotype information (such as at locations between genetic markers), the Kruskal-Wallis statistic is modified so that the rank for each individual is weighted by the genotype probabilities, analogous to Haley-Knott regression. For this method, if the argument `ties.random` is TRUE, ties in the phenotypes are assigned random ranks; if it is FALSE, average ranks are used and a corrected LOD score is calculate. Currently the `method` argument is ignored for this model.

Methods

- **em**: maximum likelihood is performed via the EM algorithm (Dempster et al. 1977), first used in this context by Lander and Botstein (1989).
- **imp**: multiple imputation is used, as described by Sen and Churchill (2001).
- **hk**: Haley-Knott regression is used (regression of the phenotypes on the multipoint QTL genotype probabilities), as described by Haley and Knott (1992).
- **ehk**: the extended Haley-Knott method is used (like H-K, but taking account of the variances), as described in Feenstra et al. (2006).
- **mr**: Marker regression is used. Analysis is performed only at the genetic markers, and individuals with missing genotypes are discarded. See Soller et al. (1976).

Covariates

Covariates are allowed only for the normal and binary models. The normal model is \( y = \beta_q + A\gamma + Z\delta_q + \epsilon \) where \( q \) is the unknown QTL genotype, \( A \) is a matrix of additive covariates, and \( Z \) is a matrix of covariates that interact with the QTL genotype. The columns of \( Z \) are forced to be contained in the matrix \( A \). The binary model is the logistic regression analog.

The LOD score is calculated comparing the likelihood of the above model to that of the null model \( y = \mu + A\gamma + \epsilon \).

Covariates must be numeric matrices. Individuals with any missing covariates are discarded.

X chromosome

The X chromosome must be treated specially in QTL mapping. See Broman et al. (in press).

If both males and females are included, male hemizygotes are allowed to be different from female homozygotes. Thus, in a backcross, we will fit separate means for the genotype classes AA, AB, AY, and BY. In such cases, sex differences in the phenotype could cause spurious linkage to the X chromosome, and so the null hypothesis must be changed to allow for a sex difference in the phenotype.

Numerous special cases must be considered, as detailed in the following table.

<table>
<thead>
<tr>
<th>BC</th>
<th>Sexes</th>
<th>Null</th>
<th>Alternative</th>
<th>df</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>both sexes</td>
<td>sex</td>
<td>AA/AB/AY/BY</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>all female</td>
<td>grand mean</td>
<td>AA/AB</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>all male</td>
<td>grand mean</td>
<td>AY/BY</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>F2</th>
<th>Direction</th>
<th>Sexes</th>
<th>Null</th>
<th>Alternative</th>
<th>df</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Both</td>
<td>both sexes</td>
<td>femaleF/femaleR/male</td>
<td>AA/ABf/ABr/BB/AY/BY</td>
<td>3</td>
</tr>
</tbody>
</table>
In the case that the number of degrees of freedom for the linkage test for the X chromosome is different from that for autosomes, a separate X-chromosome LOD threshold is recommended. Autosomal- and X-chromosome-specific LOD thresholds may be estimated by permutation tests with `scanone` by setting `n.perm>0` and using `perm.Xsp=TRUE`.

**Author(s)**

Karl W Broman, (kbroman@biostat.wisc.edu); Hao Wu

**References**


**See Also**

`plot.scanone`, `summary.scanone`, `scantwo`, `calc.genoprob`, `sim.geno`, `max.scanone`, `summary.scanoneperm`, `-scanone`, `+.scanone`

**Examples**

```r
# Normal Model

hyper <- calc.genoprob(hyper, step=2.5)
out.em <- scanone(hyper, method="em")
out.hk <- scanone(hyper, method="hk")

# Summarize results: peaks above 3
summary(out.em, thr=3)
summary(out.hk, thr=3)

# An alternate method of summarizing:
# patch them together and then summarize
out <- c(out.em, out.hk)
summary(out, thr=3, format="allpeaks")

# Plot the results
plot(out.hk, out.em)
plot(out.hk, out.em, chr=c(1,4), lty=1, col=c("blue","black"))

# Imputation; first need to run sim.geno
# Do just chromosomes 1 and 4, to save time
hyper.cl1n4 <- sim.geno(subset(hyper, chr=c(1,4)), step=2.5, n.draws=8)
out.imp <- scanone(hyper.cl1n4, method="imp")
summary(out.imp, thr=3)

# Plot all three results
plot(out.imp, out.hk, out.em, chr=c(1,4), lty=1, col=c("red","blue","black"))

# extended Haley-Knott
out.ehk <- scanone(hyper, method="ehk")
plot(out.hk, out.em, out.ehk, chr=c(1,4))

# Permutation tests
## Not run:
permo <- scanone(hyper, method="hk", n.perm=1000)
## End(Not run)
```
# Threshold from the permutation test
summary(permo, alpha=c(0.05, 0.10))

# Results above the 0.05 threshold
summary(out.hk, perms=permo, alpha=0.05)

# Stratified permutations
extremes <- (nmissing(hyper)/totmar(hyper) < 0.5)

# Not run:
operm.strat <- scanone(hyper, method="hk", n.perm=1000, perm.strata=extremes)
# End(Not run)
summary(operm.strat)

# X-specific permutations
fake.f2 <- calc.genoprob(fake.f2, step=2.5)

# genome scan
out <- scanone(fake.f2, method="hk")

# X-chr-specific permutations
# Not run:
operm <- scanone(fake.f2, method="hk", n.perm=1000, perm.Xsp=TRUE)
# End(Not run)

# thresholds
summary(operm)

# scanone summary with p-values
summary(out, perms=operm, alpha=0.05, pvalues=TRUE)

# Non-parametric
out.np <- scanone(hyper, model="np")
summary(out.np, thr=3)

# Plot with previous results
plot(out.np, chr=c(1,4), lty=1, col="green")
plot(out.imp, out.hk, out.em, chr=c(1,4), lty=1, col=c("red","blue","black"), add=TRUE)
# Two-part Model

data(listeria)

listeria <- calc.genoprob(listeria, step=2.5)
out.2p <- scanone(listeria, model="2part", upper=TRUE)
summary(out.2p, thr=c(5,3,3), format="allpeaks")

# Plot all three LOD scores together
plot(out.2p, out.2p, out.2p, lodcolumn=c(2,3,1), lty=1, chr=c(1,5,13),
     col=c("red","blue","black"))

# Permutation test
## Not run:
permo <- scanone(listeria, model="2part", upper=TRUE,
      n.perm=1000)
## End(Not run)

# Thresholds
summary(permo)

# Binary model

data(fake.bc)
plot(fake.bc)
fake.bc <- calc.genoprob(fake.bc, step=2.5)

# genome scans without covariates
out.nocovar <- scanone(fake.bc)
# genome scans with covariates
ac <- fake.bc$pheno[,c("sex","age")]
ic <- fake.bc$pheno[,"sex"]
out.covar <- scanone(fake.bc, pheno.col=1,
                  addcovar=ac, intcovar=ic)
summary(out.nocovar, thr=3)
summary(out.covar, thr=3)
plot(out.covar, out.nocovar, chr=c(2,5,10))

---

scanoneboot

**Bootstrap to get interval estimate of QTL location**

**Description**

Nonparametric bootstrap to get an estimated confidence interval for the location of a QTL, in the context of a single-QTL model.

**Usage**

```r
scanoneboot(cross, chr, pheno.col=1, model=c("normal","binary","2part","np"),
            method=c("em","imp","hk","ehk","mr","mr-imp","mr-argmax"),
            addcovar=NULL, intcovar=NULL, weights=NULL,
            use=c("all.obs", "complete.obs"), upper=FALSE,
            ties.random=FALSE, start=NULL, maxit=4000,
            tol=1e-4, n.boot=1000, verbose=FALSE)
```

**Arguments**

- `cross`: An object of class `cross`. See `read.cross` for details.
- `chr`: Indicates the chromosomes for which the bootstrap should be performed.
- `pheno.col`: Column number in the phenotype matrix which should be used as the phenotype. This can be a vector of integers; for methods "hk" and "imp" this can be considerably faster than doing them one at a time.
- `model`: The phenotypic model: the usual normal model, a model for binary traits, a two-part model or non-parametric analysis
- `method`: Indicates whether to use the EM algorithm, imputation, Haley-Knott regression, the extended Haley-Knott method, or marker regression. Not all methods are available for all models. Marker regression is performed either by dropping individuals with missing genotypes ("mr"), or by first filling in missing data using a single imputation ("mr-imp") or by the Viterbi algorithm ("mr-argmax").
- `addcovar`: Additive covariates; allowed only for the normal and binary models.
- `intcovar`: Interactive covariates (interact with QTL genotype); allowed only for the normal and binary models.
- `weights`: Optional weights of individuals. Should be either `NULL` or a vector of length `n.ind` containing positive weights. Used only in the case `model="normal"`. 
In the case that multiple phenotypes are selected to be scanned, this argument indicates whether to use all individuals, including those missing some phenotypes, or just those individuals that have data on all selected phenotypes.

**upper**

Used only for the two-part model; if true, the "undefined" phenotype is the maximum observed phenotype; otherwise, it is the smallest observed phenotype.

**ties.random**

Used only for the non-parametric "model"; if TRUE, ties in the phenotypes are ranked at random. If FALSE, average ranks are used and a corrected LOD score is calculated.

**start**

Used only for the EM algorithm with the normal model and no covariates. If NULL, use the usual starting values; if length 1, use random initial weights for EM; otherwise, this should be a vector of length n+1 (where n is the number of possible genotypes for the cross), giving the initial values for EM.

**maxit**

Maximum number of iterations for methods "em" and "ehk".

**tol**

Tolerance value for determining convergence for methods "em" and "ehk".

**n.boot**

Number of bootstrap replicates.

**verbose**

If TRUE, display information about the progress of the bootstrap.

**Details**

We recommend against the use of the bootstrap to derive a confidence interval for the location of a QTL; see Manichaikul et al. (2006). Use `lodint` or `bayesint` instead.

The bulk of the arguments are the same as for the `scanone` function. A single chromosome should be indicated with the `chr` argument; otherwise, we focus on the first chromosome in the input `cross` object.

A single-dimensional scan on the relevant chromosome is performed. We further perform a non-parametric bootstrap (sampling individuals with replacement from the available data, to create a new data set with the same size as the input cross; some individuals with be duplicated and some omitted). The same scan is performed with the resampled data; for each bootstrap replicate, we store only the location with maximum LOD score.

Use `summary.scanoneboot` to obtain the desired confidence interval.

**Value**

A vector of length `n.boot`, giving the estimated QTL locations in the bootstrap replicates. The results for the original data are included as an attribute, "results".

**Author(s)**

Karl W Broman, (kbroman@biostat.wisc.edu)

**References**


**scanqtl**  

**General QTL scan**

**Description**

Performs a multiple QTL scan for specified chromosomes and positions or intervals, with the possible inclusion of QTL-QTL interactions and/or covariates.

**Usage**

```r
scanqtl(cross, pheno.col=1, chr, pos, covar=NULL, formula, method="imp", incl.markers=FALSE, verbose=TRUE)
```

**Arguments**

- `cross`: An object of class `cross`. See `read.cross` for details.
- `pheno.col`: Column number in the phenotype matrix to be used as the phenotype.
- `chr`: Vector indicating the chromosome for each QTL.
- `pos`: List indicating the positions or intervals on the chromosome to be scanned. Each element should be either a single number (for a specific position) or a pair of numbers (for an interval).
- `covar`: A data frame of covariates.
- `formula`: An object of class `formula` indicating the model to be fitted. QTLs are indicated as Q1, Q2, etc. Covariates are indicated by their names in `covar`.
- `method`: Indicates whether to use the EM algorithm or imputation. (Only imputation is implemented at this point.)
- `incl.markers`: If FALSE, do calculations only at points on an evenly spaced grid.
- `verbose`: If TRUE, give feedback about progress.

**Examples**

```r
data(hyper)
hyper <- calc.genoprob(hyper, step=1, err=0.001)
## Not run: bootoutput <- scanoneboot(hyper, chr=4)

hist(bootoutput, breaks=100)
summary(bootoutput)
```

**See Also**

`scanone`, `summary.scanoneboot`, `lodint`, `bayesint`
Details

The formula is used to specify the model to be fit. In the formula, use Q1, Q2, etc., or q1, q2, etc., to represent the QTLs, and the column names in the covariate data frame to represent the covariates. Only the interaction terms need to be specified in the formula. The main effects of all input QTLs (as specified by chr and pos) and covariates (as specified by covar) will be included by default. For example, if the formula is y ~ Q1 * Q2 * Sex, and there are three elements in input chr and pos and Sex is one of the column names for input covariates, the formula used in genome scan will be y ~ Q1 + Q2 + Q3 + Sex + Q1:Q2 + Q1:Sex + Q2:Sex + Q1:Q2:Sex.

The input pos is a list or vector to specify the position/range of the input chromosomes to be scanned. If it is a vector, it gives the precise positions of the QTL on the chromosomes. If it is a list, it will contain either the precise positions or a range on the chromosomes. For example, consider the case that the input chr = c(1, 6, 13). If pos = c(9.8, 34.0, 18.6), it means to fit a model with QTL on chromosome 1 at 9.8cM, chromosome 6 at 34cM and chromosome 13 at 18.6cM. If pos = list(c(5,15), c(30,36), 18), it means to scan chromosome 1 from 5cM to 15cM, chromosome 6 from 30cM to 36cM, fix the QTL on chromosome 13 at 18cM.

Value

An object of class scanqtl. It is a multi-dimensional array of LOD scores, with the number of dimension equal to the number of QTLs specified.

Author(s)

Hao Wu

References


See Also

fitqtl, makeqtl

Examples

data(fake.f2)

# take out several QTLs
qc <- c(1, 8, 13)
fake.f2 <- subset(fake.f2, chr=qc)

# impute genotypes

## Not run:
fake.f2 <- sim.geno(fake.f2, n.draws=64, step=2, err=0.001)
## End(Not run)

# 2-dimensional genome scan with additive 3-QTL model
pos <- list(c(15,35), c(45,65), 28)
result <- scanqtl(fake.f2, pheno.col=1, chr=qc, pos=pos, 
     formula=y~Q1+Q2+Q3)

# image of the results
# chr locations
chr1 <- as.numeric(matrix(unlist(strsplit(colnames(result),"@")),
    ncol=2,byrow=TRUE)[,2])
chr8 <- as.numeric(matrix(unlist(strsplit(rownames(result),"@")),
    ncol=2,byrow=TRUE)[,2])

# image plot
image(chr1, chr8, t(result), las=1, col=rev(rainbow(256,start=0,end=2/3)))

# do the same, allowing the QTLs on chr 1 and 13 to interact
result2 <- scanqtl(fake.f2, pheno.col=1, chr=qc, pos=pos, 
     formula=y~Q1+Q2+Q3+Q1:Q3)

# image plot
image(chr1, chr8, t(result2), las=1, col=rev(rainbow(256,start=0,end=2/3)))

---

**scantwo**  
*Two-dimensional genome scan with a two-QTL model*

**Description**

Perform a two-dimensional genome scan with a two-QTL model, with possible allowance for covariates.

**Usage**

```r
scantwo(cross, chr, pheno.col=1, model=c("normal","binary"), 
    method=c("em","imp","hk","mr","mr-imp","mr-argmax"), 
    addcovar=NULL, intcovar=NULL, weights=NULL, 
    use=c("all.obs","complete.obs"), 
    incl.markers=FALSE, clean.output=FALSE, 
    maxit=4000, tol=1e-4, 
    verbose=TRUE, n.perm, perm.strata=NULL)
```

**Arguments**

- **cross**: An object of class `cross`. See `read.cross` for details.
- **chr**: Optional vector indicating the chromosomes for which LOD scores should be calculated.
- **pheno.col**: Column number in the phenotype matrix which should be used as the phenotype. This can be a vector of integers; for methods "hk" and "imp" this can be considerably faster than doing them one at a time.
- **model**: The phenotypic model: the usual normal model or a model for binary traits.
**method** Indicates whether to use the EM algorithm, imputation, Haley-Knott regression, or marker regression. Marker regression is performed either by dropping individuals with missing genotypes ("mr"), or by first filling in missing data using a single imputation ("mr-imp") or by the Viterbi algorithm ("mr-argmax").

**addcovar** Additive covariates.

**intcovar** Interactive covariates (interact with QTL genotype).

**weights** Optional weights of individuals. Should be either NULL or a vector of length n.ind containing positive weights. Used only in the case model="normal".

**use** In the case that multiple phenotypes are selected to be scanned, this argument indicates whether to use all individuals, including those missing some phenotypes, or just those individuals that have data on all selected phenotypes.

**incl.markers** If FALSE, do calculations only at points on an evenly spaced grid. If calc.genoprob or sim.geno were run with stepwidth="variable", we force incl.markers=TRUE.

**clean.output** If TRUE, clean the output with clean.scantwo, replacing LOD scores for pairs of positions that are between markers with 0. In permutations, this will be done for each permutation replicate. This can be important for the case of method="em", as there can be difficulty with algorithm convergence in these regions.

**maxit** Maximum number of iterations; used only with method "em".

**tol** Tolerance value for determining convergence; used only with method "em".

**verbose** If TRUE, display information about the progress of calculations. For method "em", if verbose is an integer above 1, further details on the progress of the algorithm will be displayed.

**n.perm** If specified, a permutation test is performed rather than an analysis of the observed data. This argument defines the number of permutation replicates.

**perm.strata** If n.perm > 0, this may be used to perform a stratified permutation test. This should be a vector with the same number of individuals as in the cross data. Unique values indicate the individual strata, and permutations will be performed within the strata.

### Details

Standard interval mapping (method="em") and Haley-Knott regression (method="hk") require that multipoint genotype probabilities are first calculated using calc.genoprob. The imputation method uses the results of sim.geno.

The method "em" is standard interval mapping by the EM algorithm (Dempster et al. 1977; Lander and Botstein 1989). Marker regression (method="mr") is simply linear regression of phenotypes on marker genotypes (individuals with missing genotypes are discarded). Haley-Knott regression (method="hk") uses the regression of phenotypes on multipoint genotype probabilities. The imputation method (method="imp") uses the pseudomarker algorithm described by Sen and Churchill (2001).

Individuals with missing phenotypes are dropped.

In the presence of covariates, the full model is

\[
y = \mu + \beta_{q_1} + \beta_{q_2} + \beta_{q_1 \times q_2} + A \gamma + Z \delta_{q_1} + Z \delta_{q_2} + Z \delta_{q_1 \times q_2} + \epsilon
\]
where $q_1$ and $q_2$ are the unknown QTL genotypes at two locations, $A$ is a matrix of covariates, and $Z$ is a matrix of covariates that interact with QTL genotypes. The columns of $Z$ are forced to be contained in the matrix $A$.

The above full model is compared to the additive QTL model,

$$y = \mu + \beta_{q_1} + \beta_{q_2} + A\gamma + Z\delta_{q_1} + Z\delta_{q_2} + \epsilon$$

and also to the null model, with no QTL,

$$y = \mu + A\gamma + \epsilon$$

In the case that n.perm is specified, the R function scantwo is called repeatedly.

**Value**

If n.perm is missing, the function returns a list with class "scantwo" and containing three components. The first component is a matrix of dimension [tot.pos x tot.pos]; the upper triangle contains the LOD scores for the additive model, and the lower triangle contains the LOD scores for the full model. The diagonal contains the results of scantwo. The second component of the output is a data.frame indicating the locations at which the two-QTL LOD scores were calculated. The first column is the chromosome identifier, the second column is the position in cM, the third column is a 1/0 indicator for ease in later pulling out only the equally spaced positions, and the fourth column indicates whether the position is on the X chromosome or not. The final component is a version of the results of scantwo including sex and/or cross direction as additive covariates, which is needed for a proper calculation of conditional LOD scores.

If n.perm is specified, the function returns a list with five different LOD scores from each of the permutation replicates. First, the maximum LOD score for the full model (two QTLs plus an interaction). Second, for each pair of chromosomes, we take the difference between the full LOD and the maximum single-QTL LOD for those two chromosomes, and then maximize this across chromosome pairs. Third, for each pair of chromosomes we take the difference between the maximum full LOD and the maximum additive LOD, and then maximize this across chromosome pairs. Fourth, the maximum LOD score for the additive QTL model. Finally, for each pair of chromosomes, we take the difference between the additive LOD and the maximum single-QTL LOD for those two chromosomes, and then maximize this across chromosome pairs.

**X chromosome**

The X chromosome must be treated specially in QTL mapping.

As in scantwo, if both males and females are included, male hemizygotes are allowed to be different from female homozygotes, and the null hypothesis must be changed in order to ensure that sex- or pgm-differences in the phenotype do not results in spurious linkage to the X chromosome. (See the help file for scantwo.)

**Author(s)**

Karl W Broman, ⟨kbroman@biostat.wisc.edu⟩; Hao Wu
References


See Also

`plot.scantwo`, `summary.scantwo`, `scanone`, `max.scantwo`, `summary.scantwoperm`, `c.scantwoperm`

Examples

data(fake.f2)

fake.f2 <- calc.genoprob(fake.f2, step=5)
out.2dim <- scantwo(fake.f2, method="hk")
plot(out.2dim)

# permutations

## Not run: permo.2dim <- scantwo(fake.f2, method="hk", n.perm=1000)
summary(permo.2dim, alpha=0.05)

# summary with p-values
summary(out.2dim, perms=permo.2dim, pvalues=TRUE,
       alphas=c(0.05, 0.10, 0.10, 0.05, 0.10))

# covariates
data(fake.bc)

fake.bc <- calc.genoprob(fake.bc, step=10)
ac <- fake.bc$pheno[,c("sex","age")]
ic <- fake.bc$pheno[,"sex"]
out <- scantwo(fake.bc, method="hk", pheno.col=1,
             addcovar=ac, intcovar=ic)
plot(out)
Simulate a QTL experiment

Description

Simulates data for a QTL experiment using a model in which QTLs act additively.

Usage

```r
sim.cross(map, model=NULL, n.ind=100, type=c("f2", "bc", "4way"),
          error.prob=0, missing.prob=0, partial.missing.prob=0,
          keep.qtlgeno=TRUE, keep.errorind=TRUE, m=0, p=0,
          map.function=c("haldane","kosambi","c-f","morgan"))
```

Arguments

- `map`: A list whose components are vectors containing the marker locations on each of the chromosomes.
- `model`: A matrix where each row corresponds to a different QTL, and gives the chromosome number, cM position and effects of the QTL.
- `n.ind`: Number of individuals to simulate.
- `type`: Indicates whether to simulate an intercross (`f2`), a backcross (`bc`), or a phase-known 4-way cross (`4way`).
- `error.prob`: The genotyping error rate.
- `missing.prob`: The rate of missing genotypes.
- `partial.missing.prob`: When simulating an intercross or 4-way cross, this gives the rate at which markers will be incompletely informative (i.e., dominant or recessive).
- `keep.qtlgeno`: If TRUE, genotypes for the simulated QTLs will be included in the output.
- `keep.errorind`: If TRUE, and if `error.prob > 0`, the identity of genotyping errors will be included in the output.
- `m`: Interference parameter; a non-negative integer. 0 corresponds to no interference.
- `p`: Probability that a chiasma comes from the no-interference mechanism.
- `map.function`: Indicates whether to use the Haldane, Kosambi, Carter-Falconer, or Morgan map function when converting genetic distances into recombination fractions.

Details

Meiosis is assumed to follow the Stahl model for crossover interference (see the references, below), of which the no interference model and the chi-square model are special cases. Chiasmata on the four-strand bundle are a superposition of chiasmata from two different mechanisms. With probability `p`, they arise by a mechanism exhibiting no interference; the remainder come from a chi-square
model with interference parameter \( m \). Note that \( m=0 \) corresponds to no interference, and with \( p=0 \), one gets a pure chi-square model.

If a chromosomes has class \( X \), it is assumed to be the X chromosome, and is assumed to be segregating in the cross. Thus, in an intercross, it is segregating like a backcross chromosome. In a 4-way cross, a second phenotype, \( sex \), will be generated.

QTLs are assumed to act additively, and the residual phenotypic variation is assumed to be normally distributed with variance 1.

For a backcross, the effect of a QTL is a single number corresponding to the difference between the homozygote and the heterozygote.

For an intercross, the effect of a QTL is a pair of numbers, \((a, d)\), where \( a \) is the additive effect (half the difference between the homozygotes) and \( d \) is the dominance deviation (the difference between the heterozygote and the midpoint between the homozygotes).

For a four-way cross, the effect of a QTL is a set of three numbers, \((a, b, c)\), where, in the case of one QTL, the mean phenotype, conditional on the QTL genotyping being AC, BC, AD or BD, is \( a \), \( b \), \( c \) or 0, respectively.

Value

An object of class `cross`. See `read.cross` for details.

If `keep.qtlgeno` is TRUE, the cross object will contain a component `qtlgeno` which is a matrix containing the QTL genotypes (with complete data and no errors), coded as in the genotype data.

If `keep.errorind` is TRUE and errors were simulated, each component of `geno` will each contain a matrix `errors`, with 1’s indicating simulated genotyping errors.

Author(s)

Karl W Broman, (kbroman@biostat.wisc.edu)

References


See Also

`sim.map, read.cross, fake.f2, fake.bc, fake.4way`
sim.geno

Examples

```r
# simulate a genetic map
map <- sim.map()

# simulate 250 intercross individuals with 2 QTLs
fake <- sim.cross(map, type="f2", n.ind=250,
                   model = rbind(c(1,45,1,1),c(5,20,0.5,-0.5)))
```

Description

Uses the hidden Markov model technology to simulate from the joint distribution $Pr(g | O)$ where $g$ is the underlying genotype vector and $O$ is the observed multipoint marker data, with possible allowance for genotyping errors.

Usage

```r
sim.geno(cross, n.draws=16, step=0, off.end=0, error.prob=0.0001,
          map.function=c("haldane","kosambi","c-f","morgan"),
          stepwidth=c("fixed", "variable"))
```

Arguments

- **cross**: An object of class `cross`. See `read.cross` for details.
- **n.draws**: Number of simulation replicates to perform.
- **step**: Maximum distance (in cM) between positions at which the simulated genotypes will be drawn, though for `step=0`, genotypes are drawn only at the marker locations.
- **off.end**: Distance (in cM) past the terminal markers on each chromosome to which the genotype simulations will be carried.
- **error.prob**: Assumed genotyping error rate used in the calculation of the penetrance $Pr(\text{observed genotype} | \text{true genotype})$.
- **map.function**: Indicates whether to use the Haldane, Kosambi, Carter-Falconer, or Morgan map function when converting genetic distances into recombination fractions.
- **stepwidth**: Indicates whether the intermediate points should with fixed or variable step sizes. We strongly recommend using "fixed"; "variable" is included only for the `qtlbim` package (http://www.ssg.uab.edu/qtlbim).

Details

After performing the forward-backward equations, we draw from $Pr(g_1 = v | O)$ and then $Pr(g_{k+1} = v | O, g_k = u)$.

In the case of the 4-way cross, with a sex-specific map, we assume a constant ratio of female: male recombination rates within the inter-marker intervals.
Value

The input cross object is returned with a component, draws, added to each component of cross$geno. This is an array of size [n.ind x n.pos x n.draws] where n.pos is the number of positions at which the simulations were performed and n.draws is the number of replicates. Attributes "error.prob", "step", and "off.end" are set to the values of the corresponding arguments, for later reference.

Author(s)

Karl W Broman, ⟨kbroman@biostat.wisc.edu⟩

See Also

calc.genoprob, argmax.geno

Examples

data(fake.f2)

fake.f2 <- sim.geno(fake.f2, step=2, n.draws=8)

---

sim.map

Simulate a genetic map

Description

Simulate the positions of markers on a genetic map.

Usage

sim.map(len=rep(100,20), n.mar=10, anchor.tel=TRUE, include.x=TRUE, sex.sp=FALSE, eq.spacing=FALSE)

Arguments

len A vector specifying the chromosome lengths (in cM)
n.mar A vector specifying the number of markers per chromosome.
anchor.tel If true, markers at the two telomeres will always be included, so if n.mar = 1 or 2, we’ll give just the two telomeric markers.
include.x Indicates whether the last chromosome should be considered the X chromosome.
sex.sp Indicates whether to create sex-specific maps, in which case the output will be a vector of 2-row matrices, with rows corresponding to the maps for the two sexes.
eq.spacing If TRUE, markers will be equally spaced.
strip.partials

Details

Aside from the telomeric markers, marker positions are simulated as iid Uniform(0, L). If len or n.mar has just one element, it is expanded to the length of the other argument. If they both have just one element, only one chromosome is simulated.

If eq.spacing is TRUE, markers are equally spaced between 0 and L. If anchor.tel is FALSE, telomeric markers are not included.

Value

A list of vectors, each specifying the locations of the markers. Each component of the list is given class A or X, according to whether it is autosomal or the X chromosome.

Author(s)

Karl W Broman, ⟨kbroman@biostat.wisc.edu⟩

See Also

sim.cross, plot.map, replace.map, pull.map

Examples

# simulate 4 autosomes, each with 10 markers
map <- sim.map(c(100,90,80,40), 10, include.x=FALSE)
plot.map(map)

# equally spaced markers
map2 <- sim.map(c(100,90,80,40), 10, include.x=FALSE, eq.spacing=TRUE)
plot(map2)

strip.partials  Strip partially informative genotypes

Description

Replace all partially informative genotypes (e.g., dominant markers in an intercross) with missing values.

Usage

strip.partials(cross, verbose=TRUE)

Arguments

cross  An object of class cross. See read.cross for details.
verbose  If TRUE, print the number of genotypes removed.
subset.cross

**Value**

The same class `cross` object as in the input, but with partially informative genotypes made missing.

**Author(s)**

Karl W Broman, (kbroman@biostat.wisc.edu)

**See Also**

`plot.missing`, `plot.info`

**Examples**

```r
data(listeria)
sum(nmissing(listeria))
listeria <- strip.partials(listeria)
sum(nmissing(listeria))
```

---

**Description**

Pull out a specified set of chromosomes and/or individuals from a `cross` object.

**Usage**

```r
## S3 method for class 'cross':
subset(x, chr, ind, ...)
```

**Arguments**

- `x` An object of class `cross`. See `read.cross` for details.
- `chr` A vector specifying which chromosomes to keep or discard. This may be a logical vector, a numeric vector, or a vector of character strings.
- `ind` A vector specifying which individuals to keep or discard. This may be a logical or numeric vector.
- `...` Ignored at this point.

**Value**

The input `cross` object, but with only the specified subset of the data.

**Author(s)**

Karl W Broman, (kbroman@biostat.wisc.edu)
subset.scanone

See Also

pull.map, drop.markers

Examples

data(fake.f2)
fake.f2.A <- subset(fake.f2, chr=c("5","13"))
fake.f2.B <- subset(fake.f2, ind = -c(1,5,10))
fake.f2.C <- subset(fake.f2, chr=1:5, ind=1:50)

data(listeria)
y <- listeria$pheno[,1]
listeriaB <- subset(listeria, ind = (!is.na(y) & y < 264))

Description

Pull out a specified set of chromosomes and/or LOD columns from scanone output.

Usage

## S3 method for class 'scanone':
subset(x, chr, lodcolumn, ...)

Arguments

x An object of class scanone, output from scanone.
chr A character vector specifying which chromosomes to keep.
lodcolumn A vector specifying which LOD columns to keep (or, if negative), omit. These should be between 1 and the number of LOD columns in the input x.
... Ignored at this point.

Value

The input scanone object, but with only the specified subset of the data.

Author(s)

Karl W Broman, ⟨kbroman@biostat.wisc.edu⟩

See Also

summary.scanone, scanone
Examples

data(fake.bc)

fake.bc <- calc.genoprob(fake.bc, step=2.5)
out <- scantwo(fake.bc, method="hk", pheno.col=1:2)

summary(subset(out, chr=18:19), format="allpeaks")

subset.scantwo  Subsetting the results of a 2-d genome scan

Description

Pull out a specified set of chromosomes and/or LOD columns from scantwo output.

Usage

## S3 method for class 'scantwo':
subset(x, chr, lodcolumn, ...)

Arguments

x  An object of class scantwo, output from scantwo.
chr  A character vector specifying which chromosomes to keep.
lodcolumn  A vector specifying which LOD columns to keep (or, if negative), omit. These should be between 1 and the number of LOD columns in the input x.
...  Ignored at this point.

Value

The input scantwo object, but with only the specified subset of the data.

Author(s)

Karl W Broman, ⟨kbroman@biostat.wisc.edu⟩

See Also

summary.scantwo, scantwo

Examples

data(fake.bc)

fake.bc <- calc.genoprob(fake.bc)
out <- scantwo(fake.bc, method="hk", pheno.col=1:2)

summary(subset(out, chr=18:19))
summary.cross

Print summary of QTL experiment

Description

Print summary information about a cross object.

Usage

## S3 method for class 'cross':
summary(object, ...)
nind(object)
nmar(object)
totmar(object)
nchr(object)
nphe(object)

Arguments

object  An object of class cross. See read.cross for details.
...
Ignored at this point.

Value

The six functions described here return the following values.

summary.cross  An object of class summary.cross containing a variety of summary information about the cross (this is generally printed automatically).
nind  The number of individuals in the cross.
nmar  A vector containing the number of markers on each chromosome.
totmar  The total number of markers.
nchr  The number of chromosomes.
nphe  The number of phenotypes.

Author(s)

Karl W Broman, (kbroman@biostat.wisc.edu)

See Also

read.cross, plot.cross
Examples

data(fake.f2)
summary(fake.f2)
nind(fake.f2)
nmar(fake.f2)
totmar(fake.f2)
nchr(fake.f2)
nphe(fake.f2)

Description

Print summary information about the results of \texttt{fitqtl}.

Usage

\begin{verbatim}
## S3 method for class 'fitqtl':
summary(object, ...)
\end{verbatim}

Arguments

\begin{verbatim}
object Output from \texttt{fitqtl}.
...
\end{verbatim}

Value

An object of class \texttt{summary.fitqtl}, which is not all that different than the input, but when printed gives summary information about the results.

Author(s)

Hao Wu

References


See Also

\texttt{fitqtl,makeqtl,scanqtl}
**Examples**

data(fake.f2)

# take out several QTLs and make QTL object
qc <- c(1, 8, 13)
qp <- c(26, 56, 28)
fake.f2 <- subset(fake.f2, chr=qc)

fake.f2 <- sim.geno(fake.f2, n.draws=8, step=2, err=0.001)
qlt1 <- makeqtl(fake.f2, qc, qp)

# fit model with 3 interacting QTLs interacting
# (performing a drop-one-term analysis)
lod <- fitqtl(fake.f2$pheno[,1], qlt1, formula=y~Q1*Q2*Q3)
summary(lod)

---

**summary.map**

*Print summary of a genetic map*

**Description**

Print summary information about a `map` object.

**Usage**

```r
## S3 method for class 'map':
summary(object, ...)
```

**Arguments**

- `object` An object of class `map`, which is a list of vectors (or, for a sex-specific map, 2-row matrices), each specifying the locations of the markers. The object can also be of class `cross`, in which case the function `pull.map` is used to extract the genetic map from the object.
- `...` Ignored at this point.

**Value**

An object of class `summary.map`, which is just a data.frame containing the number of markers, length, and average inter-marker spacing for each chromosome and overall. An attribute `sexsp` indicates whether the map was sex-specific.

**Author(s)**

Karl W Broman, (kbroman@biostat.wisc.edu)

**See Also**

`chrlen`, `pull.map`, `summary.cross`
summary.qtl

Print summary of a QTL object

Description
Print summary information about a qtl object.

Usage

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

Arguments

object   An object of class qtl, created by makeqtl.
...
    Ignored at this point.

Value
An object of class summary.qtl, which is just a data.frame containing the chromosomes, positions, and number of possible genotypes for each QTL.

Author(s)
Karl W Broman, ⟨kbroman@biostat.wisc.edu⟩

See Also
makeqtl

Examples

data(fake.f2)

# take out several QTLs and make QTL object
cq <- c(1, 6, 13)
qg <- c(25.8, 33.6, 18.63)
fake.f2 <- subset(fake.f2, chr=cq)
fake.f2 <- sim.geno(fake.f2, n.draws=8, step=2, err=0.001)
qtl <- makeqtl(fake.f2, qc, qp, what="draws")

summary(qtl)
summary.ripple  Print summary of ripple results

Description

Print marker orders, from the output of the function `ripple`, for which the log10 likelihood relative to the initial order is above a specified cutoff.

Usage

## S3 method for class 'ripple':
summary(object, lod.cutoff = -1, ...)

Arguments

object   An object of class `ripple`, the output of the function `ripple`.
lod.cutoff Only marker orders with LOD score (relative to the initial order) above this cutoff will be displayed. For output of `ripple` in the case of minimization of the number of obligate crossovers, we double this argument and treat it as a cutoff for the number of obligate crossovers.
...
Ignored at this point.

Value

An object of class `summary.ripple`, whose rows correspond to marker orders with likelihood (or number of obligate crossovers) within some cutoff of the initial order. If no marker order, other than the initial one, has likelihood within the specified range, the initial and next-best orders are returned.

Author(s)

Karl W Broman, ⟨kbroman@biostat.wisc.edu⟩

See Also

`ripple`, `est.map`, `est.rf`

Examples

```
## Not run: data(badorder)
rip1 <- ripple(badorder, 1, 7)
summary(rip1)

rip2 <- ripple(badorder, 1, 2, method="likelihood")
summary(rip2)

badorder <- switch.order(badorder, 1, rip2[2,])
## End(Not run)
```
summary.scanone  

Summarize the results of a genome scans

Description

Print the rows of the output from scanone that correspond to the maximum LOD for each chromosome, provided that they exceed some specified thresholds.

Usage

```r
## S3 method for class 'scanone':
summary(object, threshold,
    format=c("onepheno", "allpheno", "allpeaks"),
    perms, alpha, lodcolumn=1, pvalues=FALSE, df=FALSE, ...)
```

Arguments

- **object**: An object output by the function `scanone`.
- **threshold**: LOD score thresholds. Only peaks with LOD score above this value will be returned. This could be a single number or (for formats other than "onepheno") a threshold for each LOD score column. If `alpha` is specified, `threshold` should not be.
- **format**: Format for the output; used only if there is more than one LOD score column in the input. See Details, below.
- **perms**: Optional permutation results used to derive thresholds or to calculate genome-scan-adjusted p-values. This must be consistent with the `object` input, in that it must have the same number of LOD score columns.
- **alpha**: If `perms` are included, this is the significance level used to calculate thresholds for determining which peaks to pull out. If `threshold` is specified, `alpha` should not be.
- **lodcolumn**: If `format="onepheno"`, this indicates the LOD score column to focus on. This should be a single number between 1 and the number of LOD columns in the object input.
- **pvalues**: If TRUE, include columns with genome-scan-adjusted p-values in the results. This requires that `perms` be provided.
- **df**: If TRUE, the degrees of freedom associated with the LOD scores are shown.
- **...**: Ignored at this point.

Details

This function is used to report loci deemed interesting from a one-QTL genome scan (by `scanone`). For `format="onepheno"`, we focus on a single LOD score column, indicated by `lodcolumn`. The single largest LOD score peak on each chromosome is extracted. If `threshold` is specified, only those peaks with LOD meeting the threshold will be returned. If `perms` and `alpha` are
specified, a threshold is calculated based on the permutation results in \texttt{perms} for the significance level \texttt{alpha}. If neither \texttt{threshold} nor \texttt{alpha} are specified, the peak on each chromosome is returned. Again note that with this format, only the LOD score column indicated by \texttt{lodcolumn} is considered in deciding which chromosomes to return, but the LOD scores from other columns, at the position with maximum LOD score in the \texttt{lodcolumn} column, are also returned.

For \texttt{format="allpheno"}, we consider all LOD score columns, and pull out the position, on each chromosome, showing the largest LOD score. The output thus may contain multiple rows for a chromosome. Here \texttt{threshold} may be a vector of LOD score thresholds, one for each LOD score column, in which case only those positions for which a LOD score column exceeded its threshold are given. If \texttt{threshold} is a single number, it is applied to all of the LOD score columns. If \texttt{alpha} is specified, it must be a single significance level, applied for all LOD score columns, and again \texttt{perms} must be specified, and these are used to calculate the LOD score threshold for the significance level \texttt{alpha}.

For \texttt{format="allpeaks"}, the output will contain, for each chromosome, the maximum LOD score for each LOD score column, at the position at which it achieved its maximum. Thus, the output will contain no more than one row per chromosome, but will contain the position and maximum LOD score for each of the LOD score columns. The arguments \texttt{threshold} and \texttt{alpha} may be specified as for the "allpheno" format. The results for a chromosome are returned if at least one of the LOD score columns exceeded its threshold.

If \texttt{pvalues=TRUE}, and \texttt{perms} is specified, genome-scan-adjusted p-values are calculated for each LOD score column, and there are additional columns in the output containing these p-values. In the case that X-chromosome specific permutations were performed (with \texttt{perm.Xsp=TRUE} in \texttt{scanone}), autosome- and X-chromosome specific thresholds and p-values are calculated by the method in Broman et al. (in press).

Value

An object of class \texttt{summary.scanone}, to be printed by \texttt{print.summary.scanone}.

Author(s)

Karl W Broman, \{kbroman@biostat.wisc.edu\}

References


See Also

\texttt{scanone,plot.scanone,max.scanone,subset.scanone,c.scanone,summary.scanoneperm,c.scanoneperm}

Examples

data(fake.bc)

fake.bc <- calc.genoprob(fake.bc, step=5)
# genome scan by Haley-Knott regression
out <- scanone(fake.bc, method="hk")

# permutation tests
## Not run:
operm <- scanone(fake.bc, method="hk", n.perm=100)
## End(Not run)

# peaks for all chromosomes
summary(out)

# results with LOD >= 3
summary(out, threshold=3)

# the same, but also showing the p-values
summary(out, threshold=3, perms=operm, pvalues=TRUE)

# results with LOD meeting the 0.05 threshold from the permutation results
summary(out, perms=operm, alpha=0.05)

# the same, also showing the p-values
summary(out, perms=operm, alpha=0.05, pvalues=TRUE)

##### summary with multiple phenotype results
out2 <- scanone(fake.bc, pheno.col=1:2, method="hk")

# permutations
## Not run:
operm2 <- scanone(fake.bc, pheno.col=1:2, method="hk", n.perm=1000)
## End(Not run)

# results with LOD >= 2 for the 1st phenotype and >= 1 for the 2nd phenotype
# using format="allpheno"
summary(out2, thr=c(2, 1), format="allpheno")

# The same with format="allpeaks"
summary(out2, thr=c(2, 1), format="allpeaks")

# The same with p-values
summary(out2, thr=c(2, 1), format="allpeaks", perms=operm2, pvalues=TRUE)

# results with LOD meeting the 0.05 significance level by the permutations
# using format="allpheno"
summary(out2, format="allpheno", perms=operm2, alpha=0.05)

# The same with p-values
summary(out2, format="allpheno", perms=operm2, alpha=0.05, pvalues=TRUE)

# The same with format="allpeaks"
summary(out2, format="allpeaks", perms=operm2, alpha=0.05, pvalues=TRUE)
**summary.scanoneboot**

*Bootstrap confidence interval for QTL location*

**Description**

Calculates a bootstrap confidence interval for QTL location, using the bootstrap results from `scanoneboot`.

**Usage**

```r
## S3 method for class 'scanoneboot':
summary(object, prob=0.95, ...)
```

**Arguments**

- `object` Output from `scanoneboot`.
- `prob` Desired coverage.
- `...` Ignored at this point.

**Value**

An object of class `scanone`, indicating the position with the maximum LOD, and indicating endpoints for the estimated bootstrap confidence interval.

**Author(s)**

Karl W Broman, ⟨kbroman@biostat.wisc.edu⟩

**See Also**

`scanoneboot, lodint, bayesint`

**Examples**

```r
## Not run: data(hyper)
hyper <- calc.genoprob(hyper, step=1, err=0.001)
bootoutput <- scanoneboot(hyper, chr=4)

hist(bootoutput, breaks=100)
summary(bootoutput)## End(Not run)
```
**summary.scanoneperm**

*LOD thresholds from scanone permutation results*

**Description**

Print the estimated genome-wide LOD thresholds on the basis of permutation results from `scanone` (with `n.perm > 0`).

**Usage**

```r
## S3 method for class 'scanoneperm':
summary(object, alpha=c(0.05, 0.10), df=FALSE, ...)
```

**Arguments**

- `object` Output from the function `scanone` with `n.perm > 0`.
- `alpha` Genome-wide significance levels.
- `df` If TRUE, the degrees of freedom associated with the LOD scores are shown.
- `...` Ignored at this point.

**Details**

If there were autosomal data only or `scanone` was run with `perm.Xsp=FALSE`, genome-wide LOD thresholds are given; these are the 1-\(\alpha\) quantiles of the genome-wide maximum LOD scores from the permutations.

If there were autosomal and X chromosome data and `scanone` was run with `perm.Xsp=TRUE`, autosome- and X-chromosome-specific LOD thresholds are given, by the method described in Bro- man et al. (in press). Let \(L_A\) and \(L_X\) be total the genetic lengths of the autosomes and X chromosome, respectively, and let \(L_T = L_A + L_X\) Then in place of \(\alpha\), we use

\[
\alpha_A = 1 - (1 - \alpha)^{L_A/L_T}
\]

as the significance level for the autosomes and

\[
\alpha_X = 1 - (1 - \alpha)^{L_X/L_T}
\]

as the significance level for the X chromosome. The result is a list with two matrices, one for the autosomes and one for the X chromosome.

**Value**

An object of class `summary.scanoneperm`, to be printed by `print.summary.scanoneperm`. If there were X chromosome data and `scanone` was run with `perm.Xsp=TRUE`, there are two matrices in the results, for the autosome and X-chromosome LOD thresholds.
**summary.scantwo**  
*Summarize the results of a two-dimensional genome scan*

**Description**

Summarize the interesting aspects of the results of `scantwo`.

**Usage**

```r
## S3 method for class 'scantwo':
summary(object, thresholds, what=c("best", "full", "add", "int"), perms, alphas, lodcolumn=1, pvalues=FALSE, df=FALSE, allpairs=TRUE, ...)
```

**Arguments**

- **object**: An object of class `scantwo`, the output of the function `scantwo`.
- **thresholds**: A vector of length 5, giving LOD thresholds for the full, conditional-interactive, interaction, additive, and conditional-additive LOD scores. See Details, below.

**Examples**

```r
data(fake.f2)
fake.f2 <- calc.genoprob(fake.f2, step=2.5)
operm1 <- scantwo(fake.f2, n.perm=100, method="hk")
summary(operm1)
operm2 <- scantwo(fake.f2, n.perm=100, method="hk", perm.Xsp=TRUE)
summary(operm2)
```

**Author(s)**

Karl W Broman, (kbroman@biostat.wisc.edu)

**References**


**See Also**

`scanone`, `summary.scanone`, `quantile`
what

Indicates for which LOD score the maximum should be reported. See Details, below.

perms

Optional permutation results used to derive thresholds or to calculate genome-scan-adjusted p-values. This must be consistent with the object input, in that it must have the same number of LOD score columns.

alphas

If perms are included, these are the significance levels used to calculate thresholds for determining which peaks to pull out. It should be a vector of length 5, giving significance levels for the full, conditional-interactive, interaction, additive, and conditional-additive LOD scores. (It can also be a single number, in which case it is assumed that the same value is used for all five LOD scores.) If thresholds is specified, alphas should not be.

lodcolumn

If the scantwo results contain LOD scores for multiple phenotypes, this argument indicates which to use in the summary. Only one LOD score column may be considered at a time.

pvalues

If TRUE, include columns with genome-scan-adjusted p-values in the results. This requires that perms be provided.

df

If TRUE, the degrees of freedom associated with the LOD scores are shown.

allpairs

If TRUE, all pairs of chromosomes are considered. If FALSE, only self-self pairs are considered, so that one may more conveniently check for possible linked QTL.

... Ignored at this point.

Details

If what="best", we calculate, for each pair of chromosomes, the maximum LOD score for the full model (two QTL plus interaction) and the maximum LOD score for the additive model. The difference between these is a LOD score for a test for interaction. We also calculate the difference between the maximum full LOD and the maximum single-QTL LOD score for the two chromosomes; this is the LOD score for a test for a second QTL, allowing for epistasis, which we call either the conditional-interactive or "fv1" LOD score. Finally, we calculate the difference between the maximum additive LOD score and the maximum single-QTL LOD score for the two chromosomes; this is the LOD score for a test for a second QTL, assuming that the two QTL act additively, which we call either the conditional-additive or "av1" LOD score. Note that the maximum full LOD and additive LOD are allowed to occur in different places.

If what="full", we find the maximum full LOD and extract the additive LOD at the corresponding pair of positions; we derive the other three LOD scores for that fixed pair of positions.

If what="add", we find the maximum additive LOD and extract the full LOD at the corresponding pair of positions; we derive the other three LOD scores for that fixed pair of positions.

If what="int", we find the pair of positions for which the difference between the full and additive LOD scores is largest, and then calculate the five LOD scores at that pair of positions.

If thresholds or alphas is provided (and note that when alphas is provided, perms must also), we extract just those pairs of chromosomes for which either (a) the full LOD score exceeds its thresholds and either the conditional-interactive LOD or the interaction LOD exceed their threshold, or (b) the additive LOD score exceeds its threshold and the conditional-additive LOD exceeds its threshold. The thresholds or alphas must be given in the order full, cond-int, int, add, cond-add.
Thresholds may be obtained by a permutation test with `scantwo`, but these are extremely time-consuming. For a mouse backcross, we suggest the thresholds (6.0, 4.7, 4.4, 4.7, 2.6) for the full, conditional-interactive, interaction, additive, and conditional-additive LOD scores, respectively. For a mouse intercross, we suggest the thresholds (9.1, 7.1, 6.3, 6.3, 3.3) for the full, conditional-interactive, interaction, additive, and conditional-additive LOD scores, respectively. These were obtained by 10,000 simulations of crosses with 250 individuals, markers at a 10 cM spacing, and analysis by Haley-Knott regression.

**Value**

An object of class `summary.scantwo`, to be printed by `print.summary.scantwo`;

**Note**

The previous version of this function is still available, though it is now named `summary.scantwo.old`. We much prefer the revised function. However, while we are confident that this function (and the permutations in `scantwo`) are calculating the relevant statistics, the appropriate significance levels for these relatively complex series of statistical tests is not yet completely clear.

**Author(s)**

Karl W Broman, (kbroman@biostat.wisc.edu)

**See Also**

`scantwo`, `plot.scantwo`, `max.scantwo`, `condense.scantwo`

**Examples**

```r
data(fake.f2)
fake.f2 <- calc.genoprobs(fake.f2, step=5)
out.2dim <- scantwo(fake.f2, method="hk")

# All pairs of chromosomes
summary(out.2dim)

# Chromosome pairs meeting specified criteria
summary(out.2dim, thresholds=c(9.1, 7.1, 6.3, 6.3, 3.3))

# Similar, but ignoring the interaction LOD score in the rule
summary(out.2dim, thresholds=c(9.1, 7.1, Inf, 6.3, 3.3))

# Pairs having largest interaction LOD score, if it's > 4
summary(out.2dim, thresholds=c(0, Inf, 4, Inf, Inf), what="int")

# permutation test to get thresholds; run in two batches
# and then combined with c.scantwoperm
## Not run:
operm.2dimA <- scantwo(fake.f2, method="hk", n.perm=500)
operm.2dimB <- scantwo(fake.f2, method="hk", n.perm=500)
```
summary.scantwo.old

Summarize the results of a two-dimensional genome scan

Description

Summarize the interesting aspects of the results of `scantwo`; this is the version of `summary.scantwo` that was included in R/qtl version 1.03 and earlier.

Usage

```r
## S3 method for class 'scantwo.old':
summary(object, thresholds = c(0, 0, 0), lodcolumn=1,
   type = c("joint","interaction"), ...)
```

Arguments

- **object**: An object of class `scantwo`, the output of the function `scantwo`.
- **thresholds**: A vector of length three, giving LOD thresholds for the joint LOD, interaction LOD and single-QTL conditional LOD. Negative threshold values are taken relative to the maximum joint, interaction, or individual QTL LOD, respectively.
- **lodcolumn**: If the scantwo results contain LOD scores for multiple phenotypes, this argument indicates which to use in the summary.
- **type**: Indicates whether to pick peaks with maximal joint or interaction LOD.
- **...**: Ignored at this point.
Details

For each pair of chromosomes, the pair of loci for which the LOD score (either joint or interaction LOD, according to the argument type) is a maximum is considered. The pair is printed only if its joint LOD score exceeds the joint threshold and either (a) the interaction LOD score exceeds its threshold or (b) both of the loci have conditional LOD scores that are above the conditional LOD threshold, where the conditional LOD score for locus $q_1$, $LOD(q_1|q_2)$, is the log$_{10}$ likelihood ratio comparing the model with $q_1$ and $q_2$ acting additively to the model with $q_2$ alone.

In the case the results of scanone are not available, the maximum locus pair for each chromosome is printed whenever its joint LOD exceeds the joint LOD threshold.

The criterion used in this summary is due to Gary Churchill and Saunak Sen, and deserves careful consideration and possible revision.

Value

An object of class summary.scantwo.old, to be printed by print.summary.scantwo.old. Pairs of loci meeting the specified criteria are printed, with their joint LOD, interaction LOD, and the conditional LOD for each locus, along with single-point P-values calculated by the $\chi^2$ approximation. P-values are printed as $-\log_{10}(P)$.

If the input scantwo object does not include the results of scanone, the interaction and conditional LOD thresholds are ignored, and all pairs of loci for which the joint LOD exceeds its threshold are printed, though without their conditional LOD scores.

Author(s)

Hao Wu; Karl W Broman, (kbroman@biostat.wisc.edu); Brian Yandell

See Also

summary.scantwo, scantwo, plot.scantwo, max.scantwo

Examples

data(fake.f2)

fake.f2 <- calc.genoprob(fake.f2, step=5)
out.2dim <- scantwo(fake.f2, method="hk")

# All pairs of loci
summary.scantwo.old(out.2dim)

# Pairs meeting specified criteria
summary.scantwo.old(out.2dim, c(7, 3, 3))

# Pairs with both conditional LODs > 2
summary.scantwo.old(out.2dim,c(0,1000,2))

# Pairs with interaction LOD is above 3
summary.scantwo.old(out.2dim,c(0,3,1000))
Summary.scantwoperm

LOD thresholds from scantwo permutation results

Description
Print the estimated genome-wide LOD thresholds on the basis of permutation results from scantwo (with n.perm > 0).

Usage
## S3 method for class 'scantwoperm':
summary(object, alpha=c(0.05, 0.10), df=FALSE, ...)

Arguments
- object: Output from the function scantwo with n.perm > 0.
- alpha: Genome-wide significance levels.
- df: If TRUE, the degrees of freedom associated with the LOD scores are shown.
- ...: Ignored at this point.

Details
We take the 1 − α quantiles of the individual LOD scores.

Value
An object of class summary.scantwoperm, to be printed by print.summary.scantwoperm.

Author(s)
Karl W Broman, ⟨kbroman@biostat.wisc.edu⟩

References

See Also
scantwo, summary.scantwo, quantile
Examples

```
data(fake.f2)

fake.f2 <- calc.genoprob(fake.f2, step=0)

operm <- scantwo(fake.f2, n.perm=100, method="hk")
summary(operm)
```

```
switch.order  Switch the order of markers on a chromosome

Description

Switch the order of markers on a specified chromosome to a specified new order.

Usage

```
switch.order(cross, chr, order, error.prob=0.0001,
            map.function=c("haldane","kosambi","c-f","morgan"),
            maxit=4000, tol=1e-4, sex.sp=TRUE)
```

Arguments

cross An object of class cross. See read.cross for details.
chr The chromosome for which the marker order is to be switched.
order A vector of numeric indices defining the new marker order. The vector may have
length two more than the number of markers, for easy in use with the output of
the function ripple.
error.prob Assumed genotyping error rate (passed to est.map).
map.function Map function to be used (passed to est.map).
maxit Maximum number of EM iterations to perform.
tol Tolerance for determining convergence.
sex.sp Indicates whether to estimate sex-specific maps; this is used only for the 4-way
cross.

Value

The input cross object, but with the marker order on the specified chromosome updated, and with
any derived data removed (except for recombination fractions, if present, which are not removed);
the genetic map for the relevant chromosome is re-estimated.

Author(s)

Karl W Broman, (kbroman@biostat.wisc.edu)
See Also

ripple, clean

Examples

data(fake.f2)
fake.f2 <- switch.order(fake.f2, 1, c(1,3,2,4:7))

top.errorlod  List genotypes with large error LOD scores

Description

Prints those genotypes with error LOD scores above a specified cutoff.

Usage

top.errorlod(cross, chr, cutoff=4, msg=TRUE)

Arguments

cross An object of class cross. See read.cross for details.
chr A vector specifying for which chromosomes the error LOD scores should be inspected.
cutoff Only those genotypes with error LOD scores above this cutoff will be listed.
msg If TRUE, print a message if there are no apparent errors.

Value

A data.frame with 4 columns, whose rows correspond to the genotypes that are possibly in error. The four columns give the chromosome number, individual number, marker name, and error LOD score.

Author(s)

Karl W Broman, (kbroman@biostat.wisc.edu)

See Also

calc.errorlod, plot.geno
Examples

data(hyper)

# Calculate error LOD scores
hyper <- calc.errorlod(hyper, error.prob=0.01)

# Print those above a specified cutoff
top.errorlod(hyper, cutoff=4)

write.cross(data)  # Write data for a QTL experiment to a file

Description

Data for a QTL experiment is written to a file (or files).

Usage

write.cross(cross, format=c("csv", "csvr", "csvs", "csvsr", "mm", "qtlcart", "gary"),
            filestem="data", chr, digits=5)

Arguments

cross An object of class cross. See read.cross for details.
format Specifies whether to write the data in comma-delimited, rotated comma-delimited,
         Mapmaker, QTL Cartographer, or Gary Churchill’s format.
filestem A character string giving the first part of the output file names (the bit before the
         dot). In Windows, use forward slashes ("/") or double backslashes ("\") to
         specify directory trees.
chr A vector specifying for which chromosomes genotype data should be written.
digits Number of digits to which phenotype values should be rounded.

Details

Comma-delimited formats: a single csv file is created in the formats "csv" or "csvr". Two files
are created (one for the genotype data and one for the phenotype data) for the formats "csvs"
and "csvsr": if filestem="file", the two files will be names "file_gen.csv" and
"file_phe.csv". See the help file for read.cross for details on these formats.

Mapmaker format: Data is written to two files. Suppose filestem="file". Then "file.raw"
will contain the genotype and phenotype data, and "file.prep" will contain the necessary code
for defining the chromosome assignments, marker order, and inter-marker distances.

QTL Cartographer format: Data is written to two files. Suppose filestem="file". Then
"file.cro" will contain the genotype and phenotype data, and "file.map" will contain the
genetic map information. Note that cross types are converted to QTL Cartographer cross types as
follows: riself to RF1, risib to RF2, bc to B1 and f2 to RF2.
Gary’s format: Data is written to six files. They are:
"geno.data" - genotype data;
"pheno.data" - phenotype data;
"chrid.dat" - the chromosome identifier for each marker;
"mnames.txt" - the marker names;
"markerpos.txt" - the marker positions;
"pnames.txt" - the phenotype names

Author(s)
Karl W Broman, ⟨kbroman@biostat.wisc.edu⟩; Hao Wu; Brian S. Yandell

See Also
read.cross

Examples
## Not run: data(fake.bc)

# comma-delimited format
write.cross(fake.bc, "csv", "Data/fakebc", c(1,5,13))

# rotated comma-delimited format
write.cross(fake.bc, "csvr", "Data/fakebc", c(1,5,13))

# split comma-delimited format
write.cross(fake.bc, "csvs", "Data/fakebc", c(1,5,13))

# split and rotated comma-delimited format
write.cross(fake.bc, "csvsr", "Data/fakebc", c(1,5,13))

# Mapmaker format
write.cross(fake.bc, "mm", "Data/fakebc", c(1,5,13))

# QTL Cartographer format
write.cross(fake.bc, "qtlcart", "Data/fakebc", c(1,5,13))

# Gary's format
write.cross(fake.bc, "gary", c(1,5,13))## End(Not run)
Index

*Topic IO
   read.cross, 89
   write.cross, 139
*Topic arith
   arithscan, 10
   arithscanperm, 11
*Topic datagen
   sim.cross, 113
   sim.map, 116
*Topic datasets
   badorder, 12
   bristle3, 14
   bristleX, 15
   fake.4way, 42
   fake.bc, 44
   fake.f2, 45
   hyper, 57
   listeria, 59
   map10, 63
*Topic hplot
   add.cim.covar, 6
   add.threshold, 7
   effectplot, 35
   effectscan, 38
   geno.image, 55
   plot.cross, 68
   plot.errorlod, 69
   plot.geno, 70
   plot.info, 71
   plot.map, 73
   plot.missing, 74
   plot.pheno, 75
   plot.pxy, 76
   plot.rf, 78
   plot.scanone, 79
   plot.scantwo, 81
*Topic internal
   qtl-internal, 85
*Topic manip
   c.cross, 16
   c.scanone, 17
   c.scanoneperm, 18
   c.scantwoperm, 19
   cbind.scanoneperm, 23
   clean.cross, 27
   clean.scantwo, 28
   convert.scanone, 32
   convert.scantwo, 33
   drop.markers, 34
   drop.nullmarkers, 34
   jittermap, 58
   movemarker, 66
   replace.map, 95
   strippartials, 117
   subset.cross, 118
   subset.scanone, 119
   subset.scantwo, 120
   switch.order, 137

*Topic models
   A starting point, 1
   cim, 26
   fitqtl, 51
   fitstahl, 53
   scanone, 97
   scantwo, 107
   scantwo, 109

*Topic print
   chrlen, 25
   condense.scantwo, 31
   max.scanone, 63
   max.scantwo, 65
   qtlversion, 88
   summary.cross, 121
   summary.fitqtl, 122
   summary.map, 123
   summary.qtl, 124
   summary.ripple, 125
   summary.scanone, 126
summary.scanoneperm, 130
summary.scantwo, 131
summary.scantwo.old, 134
summary.scan twoperm, 136
top.errorlod, 138

*Topic univar
plot.info, 71

*Topic utilities
argmax.geno, 8
bayesint, 13
calc.errorlod, 20
calc.genoprob, 21
checkAlleles, 24
comparecros ses, 29
comparegeno, 30
est.rf, 41
fill.gen o, 46
find.flanking, 47
find.marker, 48
find.markerpos, 49
find.pheno, 49
find.pseudomarker, 50
genotypeable, 56
lodint, 60
makeqtl, 61
nmissing, 67
pull.gen o, 83
pull.map, 84
ripple, 96
scanoneboot, 105
sim.gen o, 115
summary.scanoneboot, 129
+.scanone, 102
+.scanone (arithscan), 10
+.scanoneperm (arithscanperm), 11
+.scantwo (arithscan), 10
+.scantwoperm (arithscanperm), 11
-.scanone (arithscan), 10
-.scanoneperm (arithscanperm), 11
-.scantwo (arithscan), 10
-.scantwoperm (arithscanperm), 11
-.scanone, 102
-.scantwo, 82
.Rprofile, 1

A starting point, 1
abline, 7
add.cim.covar, 6, 27
add.threshold, 7
addqtl (qtl-internal), 85
adjust.rf.ri (qtl-internal), 85
argmax.geno, 8, 22, 28, 46, 116
arithscan, 10
arithscanperm, 11
badorder, 12, 42, 79
barplot, 76
bayesint, 13, 61, 106, 107, 129
bristle3, 14, 43–45, 58, 60
bristleX, 14, 15, 16, 43–45, 58, 60
c.cross, 16
c.scanone, 17, 23, 127
c.scanoneperm, 18, 20, 23, 127
c.scantwoperm, 19, 19, 112
calc.errorlod, 3, 20, 22, 70, 71, 138
calc.genoprob, 3, 4, 9, 17, 21, 28, 34, 35, 46, 62, 66, 99, 102, 110, 116
calc.genoprob.special
(qtl-internal), 85
cbind.pairprob (qtl-internal), 85
cbind.scanoneperm, 18, 19, 23
ccheckAlleles, 24, 42
ccheckcovar (qtl-internal), 85
chrlen, 25, 123
cim, 6, 26
clean, 58, 138
clean (qtl-internal), 85
clean.cross, 27, 34, 35
clean.scantwo, 28, 28, 110
colors, 80
comparecros ses, 29
comparegeno, 30
condense (qtl-internal), 85
condense.scantwo, 31, 133
contour, 81
cconvert (qtl-internal), 85
cconvert.scanone, 32, 33
cconvert.scantwo, 32, 33
create.map (qtl-internal), 85
data, 2
discan (qtl-internal), 85
drop.markers, 28, 34, 35, 57, 119
drop.nullmarkers, 2, 28, 34, 34, 57
dropqtl (qtl-internal), 85
dropXcol (qtl-internal), 85
INDEX

find.pheno, 58
find.pseudomarker, 36, 47–49, 50
fitqtl, 51, 61, 62, 108, 122
fitstahl, 41, 53
fitstahl.este (qtl-internal), 85
fitstahl.estp (qtl-internal), 85
fitstahl.estpe (qtl-internal), 85
fixX4write (qtl-internal), 85
fixX4geno.bc (qtl-internal), 85
fixX4geno.f2 (qtl-internal), 85
formula, 51, 107
forwsel (qtl-internal), 85
jittermap, 58
listeria, 14, 16, 43–45, 58, 59
locate.xo (qtl-internal), 85
locatemarker (qtl-internal), 85
lodint, 14, 60, 106, 107, 129
makeqtl, 51, 52, 61, 108, 122, 124
makeSSMap (qtl-internal), 85
map10, 63
max.scanone, 4, 63, 102, 127
max.scanone, 5, 31, 65, 112, 133, 135
max.scantwo (qtl-internal), 85
max.scantwocondensed (qtl-internal), 85
mf.cf (qtl-internal), 85
mf.h (qtl-internal), 85
mf.k (qtl-internal), 85
mf.m (qtl-internal), 85
mf.stahl (qtl-internal), 85
movemaker, 66
nchr (summary.cross), 121
nind, 67
nind (summary.cross), 121
nmissing, 30, 67
nphe (summary.cross), 121
par, 70, 80
parseformula (qtl-internal), 85
plot, 38, 80
plot.cross, 2, 56, 68, 74–76, 121
plot.estimate, 39
plot.effects, 39
plot.effects (qtl-internal), 85
plot.errorlod, 21, 69, 91, 92
plot.info, 71, 118
plot.map, 2, 3, 41, 63, 68, 73, 76, 84, 117
plot.missing, 2, 56, 68, 73, 74, 76, 118
plot.pheno, 68, 75
plot.pwg, 36, 39, 47, 48, 51, 76
plot.rf, 42, 78
plot.scanone, 3, 6–8, 27, 64, 72, 73, 79, 82, 102, 127
plot.scantwo, 4, 66, 81, 112, 133, 135
points, 6
print.cross (qtl-internal), 85
print.map (qtl-internal), 85
print.qtl (qtl-internal), 85

INDEX

effectplot, 35, 39, 48, 51, 77
effectplot.calmeanse (qtl-internal), 85
effectplot.getmark (qtl-internal), 85
effectscan, 36, 38, 77
est.map, 3, 12, 39, 42, 54, 74, 90, 95–97, 125, 137
est.rf, 2, 12, 24, 25, 41, 41, 46, 66, 79, 97, 125
example, 2
fake.4way, 14, 16, 42, 44, 45, 58, 60, 114
fake.bc, 14, 16, 43, 44, 45, 58, 60, 114
fake.f2, 14, 16, 43, 44, 45, 58, 60, 114
fill.genotype, 9, 26, 27, 46
find技能培训, 47, 48, 49, 51, 77
find.marker, 36, 47, 48, 49, 51, 77
find.markerpos, 47, 48, 49, 51
find.pheno, 49
forwsel (qtl-internal), 85
image, 56, 69, 70, 75, 78, 79, 82
imf.cf (qtl-internal), 85
imf.h (qtl-internal), 85
imf.k (qtl-internal), 85
imf.m (qtl-internal), 85
imf.stahl (qtl-internal), 85

INDEX

INDEX

INDEX

INDEX

INDEX

INDEX

INDEX

INDEX

INDEX

INDEX

INDEX

INDEX

INDEX

INDEX

INDEX

INDEX

INDEX

INDEX

INDEX

INDEX

INDEX

INDEX

INDEX

INDEX

INDEX

INDEX

INDEX

INDEX

INDEX

INDEX

INDEX

INDEX

INDEX

INDEX

INDEX

INDEX

INDEX

INDEX

INDEX

INDEX

INDEX

INDEX

INDEX

INDEX

INDEX

INDEX
print.scanoneboot (qtl-internal), 85
print.scantwo (qtl-internal), 85
print.summary.cross (qtl-internal), 85
print.summary.fitqtl (qtl-internal), 85
print.summary.map (qtl-internal), 85
print.summary.qtl (qtl-internal), 85
print.summary.ripple (qtl-internal), 85
print.summary.scanone (qtl-internal), 85
print.summary.scanoneperm (qtl-internal), 85
print.summary.scantwo (qtl-internal), 85
print.summary.scantwoperm (qtl-internal), 85
pull.geno, 83
pull.map, 25, 58, 59, 63, 84, 84, 95, 117, 119, 123
qtl-internal, 85
qtl-package (A starting point), 88
quantile, 131, 136
rbind.scanoneperm (c.scanoneperm), 18
rbind.scantwoperm (c.scantwoperm), 19
read.cro.qtlcart (qtl-internal), 85
read.cross, 2, 8, 12, 14–16, 20, 22, 24, 26, 28–30, 34, 35, 40, 42–50, 53, 55–59, 61, 66–70, 72, 75–78, 83, 84, 89, 95, 96, 98, 105, 107, 109, 114, 115, 117, 118, 121, 137–140
read.cross.csv (qtl-internal), 85
read.cross.csvs (qtl-internal), 85
read.cross.gary (qtl-internal), 85
read.cross.karl (qtl-internal), 85
read.cross.mm (qtl-internal), 85
read.cross.qtlcart (qtl-internal), 85
read.cross.qtx (qtl-internal), 85
read.map.qtlcart (qtl-internal), 85
read.maps.mm (qtl-internal), 85
read.table, 90, 91
replace.map, 3, 41, 59, 84, 95, 117
replaceqtl (qtl-internal), 85
revisecovar (qtl-internal), 85
revisescantwo (qtl-internal), 85
revisexcovar (qtl-internal), 85
ripple, 12, 79, 96, 125, 137, 138
ripple.perm.sub (qtl-internal), 85
ripple.perm1 (qtl-internal), 85
ripple.perm2 (qtl-internal), 85
save.image, 4
scanone, 3, 4, 7, 8, 10, 11, 13, 14, 17–19, 23, 27, 32, 39, 60, 61, 63, 64, 79, 80, 82, 97, 106, 107, 111, 112, 119, 126, 127, 130, 131, 135
scanone.perm (qtl-internal), 85
scanoneboot, 105, 129
scanoneXnull (qtl-internal), 85
scanqtl, 52, 107, 122
scantwo, 4, 5, 10, 11, 19, 20, 28, 29, 31, 33, 65, 66, 81, 82, 102, 109, 120, 131, 133–136
scantwo.perm (qtl-internal), 85
dset.seed, 11
sim.bcq (qtl-internal), 85
sim.cc (qtl-internal), 85
sim.cross, 12, 42–45, 54, 94, 113, 117
sim.cross.4way (qtl-internal), 85
sim.cross.bc (qtl-internal), 85
sim.cross.f2 (qtl-internal), 85
sim.gen, 3, 4, 9, 17, 22, 28, 38, 39, 46, 62, 77, 99, 102, 110, 115
sim.map, 63, 114, 116
sim.ril (qtl-internal), 85
strip.partials, 117
subrousummaryscantwo (qtl-internal), 85
subset.cross, 17, 118
subset.scanone, 119, 127
subset.scantwo, 120
summary.cross, 2, 25, 30, 57, 59, 67, 91, 121, 123
summary.fitqtl, 52, 122
summary.map, 25, 123
summary.qtl, 124
INDEX

summary.ripple, 97, 125
summary.scanone, 4, 17, 18, 27, 64, 72, 80, 102, 119, 126, 131
summary.scanoneboot, 106, 107, 129
summary.scanoneperm, 4, 7, 8, 19, 23, 102, 127, 130
summary.scantwoold, 133, 134
summary.scantwocondensed
(qtl-internal), 85
summary.scantwoperm, 20, 112, 136
switch.order, 66, 97, 137
top.errorlod, 3, 21, 70, 71, 91, 92, 138
totmar, 67
totmar(summary.cross), 121
vbscan(qtl-internal), 85
version, 89
write.cross, 94, 139
write.cross.csv(qtl-internal), 85
write.cross.gary(qtl-internal), 85
write.cross.mm(qtl-internal), 85
write.cross.qtlcart
(qtl-internal), 85