Package ‘qtl’

February 1, 2020

Version 1.45-11
Date 2020-02-01
Title Tools for Analyzing QTL Experiments
Author Karl W Broman <broman@wisc.edu> and Hao Wu, with
ideas from Gary Churchill and Saunak Sen and contributions from
Danny Arends, Robert Corty, Timothee Flutre, Ritsert Jansen, Pjotr
Prins, Lars Ronnegard, Rohan Shah, Laura Shannon, Quoc Tran, Aaron
Wolen, and Brian Yandell
Maintainer Karl W Broman <broman@wisc.edu>
Description Analysis of experimental crosses to identify genes
(called quantitative trait loci, QTLs) contributing to variation in
quantitative traits.
Depends R (>= 2.14.0)
Imports parallel, graphics, stats, utils, grDevices
Suggests testthat
License GPL-3
BugReports https://github.com/kbroman/qtl/issues
Encoding UTF-8
ByteCompile true

R topics documented:

A starting point ................................................................. 6
add.cim.covar ................................................................. 10
add.threshold ................................................................. 12
addcovarint ................................................................. 13
addint ................................................................. 15
addloctocross ................................................................. 17
addmarker ................................................................. 18
<table>
<thead>
<tr>
<th>R topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>addpair</td>
<td>19</td>
</tr>
<tr>
<td>addqtl</td>
<td>22</td>
</tr>
<tr>
<td>addtoqtl</td>
<td>24</td>
</tr>
<tr>
<td>allchrsplits</td>
<td>26</td>
</tr>
<tr>
<td>argmax.geno</td>
<td>27</td>
</tr>
<tr>
<td>arithscan</td>
<td>29</td>
</tr>
<tr>
<td>arithscanperm</td>
<td>30</td>
</tr>
<tr>
<td>badorder</td>
<td>31</td>
</tr>
<tr>
<td>bayesint</td>
<td>32</td>
</tr>
<tr>
<td>bristle3</td>
<td>33</td>
</tr>
<tr>
<td>bristleX</td>
<td>34</td>
</tr>
<tr>
<td>c.cross</td>
<td>35</td>
</tr>
<tr>
<td>c.scanone</td>
<td>36</td>
</tr>
<tr>
<td>c.scanoneperm</td>
<td>37</td>
</tr>
<tr>
<td>c.scantwo</td>
<td>38</td>
</tr>
<tr>
<td>c.scantwoperm</td>
<td>39</td>
</tr>
<tr>
<td>calc.errorlod</td>
<td>40</td>
</tr>
<tr>
<td>calc.genoprob</td>
<td>42</td>
</tr>
<tr>
<td>calc.penalties</td>
<td>43</td>
</tr>
<tr>
<td>cbind.scanoneperm</td>
<td>45</td>
</tr>
<tr>
<td>cbind.scantwoperm</td>
<td>46</td>
</tr>
<tr>
<td>checkAlleles</td>
<td>47</td>
</tr>
<tr>
<td>chrlen</td>
<td>48</td>
</tr>
<tr>
<td>chrnames</td>
<td>49</td>
</tr>
<tr>
<td>cim</td>
<td>49</td>
</tr>
<tr>
<td>clean.cross</td>
<td>51</td>
</tr>
<tr>
<td>clean.scantwo</td>
<td>52</td>
</tr>
<tr>
<td>cleanGeno</td>
<td>53</td>
</tr>
<tr>
<td>comparecrosses</td>
<td>54</td>
</tr>
<tr>
<td>comparegeno</td>
<td>55</td>
</tr>
<tr>
<td>compareorder</td>
<td>56</td>
</tr>
<tr>
<td>condense.scantwo</td>
<td>57</td>
</tr>
<tr>
<td>convert.map</td>
<td>58</td>
</tr>
<tr>
<td>convert.scanone</td>
<td>59</td>
</tr>
<tr>
<td>convert.scantwo</td>
<td>60</td>
</tr>
<tr>
<td>convert2riself</td>
<td>61</td>
</tr>
<tr>
<td>convert2risib</td>
<td>62</td>
</tr>
<tr>
<td>convert2sa</td>
<td>63</td>
</tr>
<tr>
<td>countXO</td>
<td>64</td>
</tr>
<tr>
<td>drop.dupmarkers</td>
<td>65</td>
</tr>
<tr>
<td>drop.markers</td>
<td>66</td>
</tr>
<tr>
<td>drop.nullmarkers</td>
<td>66</td>
</tr>
<tr>
<td>dropfromqtl</td>
<td>67</td>
</tr>
<tr>
<td>droponemarker</td>
<td>68</td>
</tr>
<tr>
<td>effectplot</td>
<td>70</td>
</tr>
<tr>
<td>effectscan</td>
<td>72</td>
</tr>
<tr>
<td>est.map</td>
<td>74</td>
</tr>
<tr>
<td>est.rf</td>
<td>77</td>
</tr>
</tbody>
</table>
R topics documented:

fake.4way ............................................................... 78
fake.bc ................................................................. 79
fake.f2 ................................................................. 80
fill.geno ............................................................... 81
find.flanking ......................................................... 82
find.marker ........................................................... 83
find.markerindex .................................................... 84
find.markerpos ....................................................... 85
find.pheno ............................................................. 86
find.pseudomarker .................................................. 86
findDupMarkers ...................................................... 87
fitqtl ................................................................. 89
fitstahl ............................................................... 92
flip.order ............................................................ 94
formLinkageGroups ................................................ 95
formMarkerCovar .................................................... 96
geno.crosstab ....................................................... 97
geno.image .......................................................... 98
geno.table ........................................................... 99
getid ................................................................. 100
groupclusteredheatmap .......................................... 101
hyper ................................................................. 102
inferFounderHap ..................................................... 103
inferredpartitions .................................................. 104
interpPositions ...................................................... 106
jittermap ............................................................ 107
listeria ............................................................... 108
locateXO ............................................................. 109
locations ............................................................ 110
lodint ............................................................... 111
makeqtl ............................................................... 112
map10 ................................................................. 114
map2table ............................................................ 115
mapthis .............................................................. 116
markerlrt ............................................................ 117
markernames ....................................................... 117
max.scanone ......................................................... 118
max.scanPhyloQTL ................................................ 119
max.scantwo ........................................................ 121
movemarker ........................................................ 123
MQM ................................................................. 124
mqmaugment ........................................................ 126
mqmautocofactors ............................................... 128
mqmextractmarkers ............................................... 129
mqmfind.marker ................................................... 130
mqmgetmodel ....................................................... 131
mqmpermutation ................................................... 133
mqmplot.circle ..................................................... 135
R topics documented:

- `mqmplot.cistrans` ................................................. 137
- `mqmplot.clusteredheatmap` ...................................... 138
- `mqmplot.cofactors` ................................................ 139
- `mqmplot.directedqtl` ............................................. 140
- `mqmplot.heatmap` .................................................. 142
- `mqmplot.multitrait` ............................................... 143
- `mqmplot.permutations` ........................................... 144
- `mqmplot.singletrait` .............................................. 145
- `mqmprocesspermutation` ......................................... 146
- `mqmscan` ............................................................ 148
- `mqmscanall` ........................................................ 150
- `mqmscanfdr` ........................................................ 152
- `mqmsetcofactors` .................................................. 154
- `mqmtestnormal` .................................................... 155
- `multitrait` ........................................................... 156
- `nchr` ................................................................. 157
- `nind` ................................................................. 158
- `nmar` ................................................................. 159
- `nmissing` ............................................................ 159
- `nphe` ................................................................. 160
- `nqrank` .............................................................. 161
- `nqtl` ................................................................. 162
- `ntyped` .............................................................. 163
- `nullmarkers` ........................................................ 164
- `orderMarkers` ...................................................... 165
- `phenames` .......................................................... 166
- `pickMarkerSubset` .................................................. 167
- `plot.comparegeno` .................................................. 168
- `plot.cross` .......................................................... 169
- `plot.qtl` ............................................................. 170
- `plot.rfmatrix` ...................................................... 172
- `plot.scanone` ....................................................... 173
- `plot.scanoneboot` ................................................... 175
- `plot.scanoneperm` ................................................. 176
- `plot.scanPhyloQTL` ............................................... 177
- `plot.scantwo` ....................................................... 179
- `plot.scantwoperm` .................................................. 181
- `plotErrorlod` ....................................................... 182
- `plotGeno` ............................................................ 184
- `plotInfo` ............................................................ 185
- `plotLodProfile` .................................................... 187
- `plotMap` ............................................................. 189
- `plotMissing` ....................................................... 191
- `plotModel` .......................................................... 192
- `plotPheno` .......................................................... 194
- `plotPXG` ............................................................ 195
- `plotRF` ............................................................. 196
- `pull.argmaxgeno` ................................................... 198
<table>
<thead>
<tr>
<th>R topics documented:</th>
</tr>
</thead>
<tbody>
<tr>
<td>pull.draws</td>
</tr>
<tr>
<td>pull.geno</td>
</tr>
<tr>
<td>pull.genoprob</td>
</tr>
<tr>
<td>pull.map</td>
</tr>
<tr>
<td>pull.markers</td>
</tr>
<tr>
<td>pull.pheno</td>
</tr>
<tr>
<td>pull.rf</td>
</tr>
<tr>
<td>qtlversion</td>
</tr>
<tr>
<td>read.cross</td>
</tr>
<tr>
<td>readMWril</td>
</tr>
<tr>
<td>reduce2grid</td>
</tr>
<tr>
<td>refineqtl</td>
</tr>
<tr>
<td>reorderqtl</td>
</tr>
<tr>
<td>replace.map</td>
</tr>
<tr>
<td>replacemap.scanone</td>
</tr>
<tr>
<td>replacemap.scantwo</td>
</tr>
<tr>
<td>replaceqtl</td>
</tr>
<tr>
<td>rescalemap</td>
</tr>
<tr>
<td>ripple</td>
</tr>
<tr>
<td>scanone</td>
</tr>
<tr>
<td>scanoneboot</td>
</tr>
<tr>
<td>scanonevar</td>
</tr>
<tr>
<td>scanonevar.meanperm</td>
</tr>
<tr>
<td>scanonevar.varperm</td>
</tr>
<tr>
<td>scanPhyloQTL</td>
</tr>
<tr>
<td>scanqtl</td>
</tr>
<tr>
<td>scantwo</td>
</tr>
<tr>
<td>scantwopermhk</td>
</tr>
<tr>
<td>shiftmap</td>
</tr>
<tr>
<td>sim.cross</td>
</tr>
<tr>
<td>sim.geno</td>
</tr>
<tr>
<td>sim.map</td>
</tr>
<tr>
<td>simFounderSnps</td>
</tr>
<tr>
<td>simPhyloQTL</td>
</tr>
<tr>
<td>simulatemissingdata</td>
</tr>
<tr>
<td>stepwiseqtl</td>
</tr>
<tr>
<td>strippartials</td>
</tr>
<tr>
<td>subset.cross</td>
</tr>
<tr>
<td>subset.map</td>
</tr>
<tr>
<td>subset.scanone</td>
</tr>
<tr>
<td>subset.scanoneperm</td>
</tr>
<tr>
<td>subset.scantwo</td>
</tr>
<tr>
<td>subset.scantwoperm</td>
</tr>
<tr>
<td>summary.comparegeno</td>
</tr>
<tr>
<td>summary.cross</td>
</tr>
<tr>
<td>summary.fitqtl</td>
</tr>
<tr>
<td>summary.qtl</td>
</tr>
<tr>
<td>summary.ripple</td>
</tr>
</tbody>
</table>
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 (https://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 https://rqtl.org/rqtl1tour.pdf.
- Download the latest version of R/qtl from the R archive or from https://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 https://rqt1.org/rqt1tour.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, <broman@wisc.edu>, perhaps attaching examples of your data files. (Such data will be kept confidential.) Also see the sample data files and code at https://rqt1.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 plotMissing (to plot a matrix displaying missing genotypes) and plotMap (to plot the genetic maps), and also displays histograms or barplots of the phenotypes. The plotMissing function can plot individuals ordered by their phenotypes; you can see that for most markers, only individuals with extreme phenotypes were genotyped.

plot(hyper)
plotMissing(hyper)
plot Missing(hyper,reorder=TRUE)
plotMap(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)
plotRF(hyper)
plotRF(hyper,chr=c(1,4))

To re-estimate the genetic map for an experimental cross, use the function est.map. The function plotMap, 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.

```r
newmap <- est.map(hyper, error.prob=0.01, verbose=TRUE)
plotMap(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 returns 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).

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

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

```r
plotGeno(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.

```r
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`.

```r
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`.

```r
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.

```r
plot(out.em)
plot(out.hk, col="blue", add=TRUE)
plot(out.imp, col="red", add=TRUE)
plot(out.hk, out.imp, out.em, chr=c(1,4), lty=1,
     col=c("blue","red","black"))
```

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.

```r
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 chromosomes \((k, l)\), 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 be 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, <broman@wisc.edu>

References


---

(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, <broman@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

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.

Value

None.

Author(s)

Karl W Broman, <broman@wisc.edu>

See Also

scanone, plot.scanone, summary.scanoneperm, xaxisloc.scanone
Examples

```r
data(hyper)
hyper <- calc.genoprob(hyper)
out <- scanone(hyper, method="hk")
operm <- scanone(hyper, method="hk", n.perm=100, perm.Xsp=TRUE)

plot(out, chr=c(1,4,6,15,"X"))
add.threshold(out, chr=c(1,4,6,15,"X"), perms=operm, alpha=0.05)
add.threshold(out, chr=c(1,4,6,15,"X"), perms=operm, alpha=0.1,
            col="green", lty=2)
```

addcovarint

**Add QTL x covariate interaction to a multiple-QTL model**

Description

Try adding all QTL x covariate interactions, one at a time, to a multiple QTL model, for a given set of covariates.

Usage

```r
addcovarint(cross, pheno.col=1, qtl, covar=NULL, icovar, formula,
        method=c("imp","hk"), model=c("normal","binary"),
        verbose=TRUE, pvalues=TRUE, simple=FALSE, tol=1e-4,
        maxit=1000, require.fullrank=FALSE)
```

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. One may also give a character string matching a phenotype name. Finally, one may give a numeric vector of phenotypes, in which case it must have the length equal to the number of individuals in the cross, and there must be either non-integers or values < 1 or > no. phenotypes; this last case may be useful for studying transformations.
- **qtl**: An object of class `qtl`, as output from `makeqtl`.
- **covar**: A matrix or data.frame of covariates. These must be strictly numeric.
- **icovar**: Vector of character strings indicating the columns in `covar` to be considered for QTL x covariate interactions.
- **formula**: An object of class `formula` indicating the model to be fitted. (It can also be the character string representation of a formula.) 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 multiple imputation or Haley-Knott regression.
- **model**: The phenotype model: the usual model or a model for binary traits.
- **verbose**: If TRUE, will print a message if there are no interactions to test.
pvalues If FALSE, p-values will not be included in the results.
simple If TRUE, don’t include p-values or sums of squares in the summary.
tol Tolerance for convergence for the binary trait model.
maxit Maximum number of iterations for fitting the binary trait model.
require.fullrank If TRUE, give LOD=0 when covariate matrix in the linear regression is not of full rank.

Details

The formula is used to specific 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.

We enforce a hierarchical structure on the model formula: if a QTL or covariate is involved in an interaction, its main effect must also be included.

Value

An object of class addcovarint, with results as in the drop-one-term analysis from fitqtl. This is a data frame (given class “addcovarint”), with the following columns: 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.

QTL x covariate interactions already included in the input formula are not tested.

Author(s)

Karl W Broman, <broman@wisc.edu>

References


See Also

addint, fitqtl, makeqtl, scanqtl, refineqtl, addqtl, addpair
### 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 <- calc.genoprob(fake.f2, step=2, err=0.001)
qtl <- makeqtl(fake.f2, qc, qp, what="prob")

# use the sex phenotype as the covariate
covar <- data.frame(sex=fake.f2$pheno$sex)

# try all possible QTL x sex interactions, one at a time
addcovarint(fake.f2, pheno.col=1, qtl, covar, "sex", y~Q1+Q2+Q3,
             method="hk")
```

---

**addint**  
*Add pairwise interaction to a multiple-QTL model*

### Description

Try adding all possible pairwise interactions, one at a time, to a multiple QTL model.

### Usage

```r
addint(cross, pheno.col=1, qtl, covar=NULL, formula, method=c("imp","hk"),
       model=c("normal", "binary"), qtl.only=FALSE, verbose=TRUE,
       pvalues=TRUE, simple=FALSE, tol=1e-4, maxit=1000, require.fullrank=FALSE)
```

### 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. One may also give a character string matching a phenotype name. Finally, one may give a numeric vector of phenotypes, in which case it must have the length equal to the number of individuals in the cross, and there must be either non-integers or values < 1 or > no. phenotypes; this last case may be useful for studying transformations.
- **qtl**: An object of class `qtl`, as output from `makeqtl`.
- **covar**: A matrix or data.frame of covariates. These must be strictly numeric.
- **formula**: An object of class `formula` indicating the model to be fitted. (It can also be the character string representation of a formula.) QTLs are referred to as Q1, Q2, etc. Covariates are referred to by their names in the data frame `covar`. If the new QTL is not included in the formula, its main effect is added.
method Indicates whether to use multiple imputation or Haley-Knott regression.
model The phenotype model: the usual model or a model for binary traits
qtl.only If TRUE, only test QTL:QTL interactions (and not interactions with covariates).
verbose If TRUE, will print a message if there are no interactions to test.
pvalues If FALSE, p-values will not be included in the results.
simple If TRUE, don’t include p-values or sums of squares in the summary.
tol Tolerance for convergence for the binary trait model.
maxit Maximum number of iterations for fitting the binary trait model.
require.fullrank If TRUE, give LOD=0 when covariate matrix in the linear regression is not of full rank.

Details

The formula is used to specified 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.

We enforce a hierarchical structure on the model formula: if a QTL or covariate is in involved in an interaction, its main effect must also be included.

Value

An object of class addint, with results as in the drop-one-term analysis from fitqtl. This is a data frame (given class "addint", with the following columns: 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.

Pairwise interactions already included in the input formula are not tested.

Author(s)

Karl W Broman, <broman@wisc.edu>

References


See Also

addcovarint, fitqtl, makeqtl, scanqtl, refineqtl, addqtl, addpair
**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 <- calc.genoprob(fake.f2, step=2, err=0.001)
qtl <- makeqtl(fake.f2, qc, qp, what="prob")

# try all possible pairwise interactions, one at a time
addint(fake.f2, pheno.col=1, qtl, formula=y~Q1+Q2+Q3, method="hk")
```

---

**addloctocross**  
*Add phenotype location into a cross object*

**Description**

Add phenotype location(s) into a cross object (with eQTL/pQTL studies)

**Usage**

```r
addloctocross(cross, locations=NULL, locfile="locations.txt", verbose=FALSE)
```

**Arguments**

- `cross`: An object of class `cross`. See `read.cross` for details.
- `locations`: R variable holding location information
- `locfile`: load from a file, see the details section for the layout of the file.
- `verbose`: If TRUE, give verbose output

**Details**

inputfile layout: Num Name Chr cM 1 X3.Hydroxypropyl 4 50.0 Num is the number of the phenotype in the cross object Name is the name of the phenotype (will be checked against the name already in the cross object at position num Chr Chromosome cM postion from start of chromosome in cM

**Value**

The input cross object, with the locations added as an additional component `locations`

**Author(s)**

Ritsert C Jansen; Danny Arends; Pjotr Prins; Karl W Broman <broman@wisc.edu>
See Also

- `mqmplot.cistrans` - Cis/trans plot
- The MQM tutorial: https://rqtl.org/tutorials/MQM-tour.pdf
- `MQM` - MQM description and references
- `mqmscan` - Main MQM single trait analysis
- `mqmscanall` - Parallelized traits analysis
- `mqmaugment` - Augmentation routine for estimating missing data
- `mqmautocofactors` - Set cofactors using marker density
- `mqmsetcofactors` - Set cofactors at fixed locations
- `mqmpermutation` - Estimate significance levels
- `scanone` - Single QTL scanning

Examples

```r
## Not run:
data(multitrait)
data(locations)
multiloc <- addloctocross(multitrait, locations)
results <- scanall(multiloc)
mqmplot.cistrans(results, multiloc, 5, FALSE, TRUE)
## End(Not run)
```

addmarker

Add a marker to a cross

Description

Add a marker to a cross object.

Usage

`addmarker(cross, genotypes, markername, chr, pos)`

Arguments

- `cross` - An object of class `cross`. See `read.cross` for details.
- `genotypes` - Vector of numeric genotypes.
- `markername` - Marker name as character string.
- `chr` - Chromosome ID as character string.
- `pos` - Position of marker, as numeric value.
addpair

Details

Use this function with caution. It would be best to incorporate new data into a single file to be imported with read.cross.

But if you have genotypes on one or two additional markers that you want to add, you might load them with read.csv and incorporate them with this function.

Value

The input cross object with the single marker added.

Author(s)

Karl W Broman, <broman@wisc.edu>

See Also

pull.markers, drop.markers

Examples

data(fake.f2)

# genotypes for new marker
gi <- pull.geno(fill.geno(fake.f2))[, "D5M197"]

# add marker to cross
fake.f2 <- addmarker(fake.f2, gi, "D5M197imp", "5", 11)

addpair  Scan for an additional pair of QTL in a multiple-QTL model

Description

Scan for an additional pair of QTL in the context of a multiple QTL model.

Usage

addpair(cross, chr, pheno.col=1, qtl, covar=NULL, formula,
method=c("imp","hk"), model=c("normal", "binary"),
incl.markers=FALSE, verbose=TRUE, tol=1e-4, maxit=1000,
forceXcovar=FALSE)
Arguments

cross
Optional vector indicating the chromosomes to be scanned. If missing, all chromosomes are scanned. Refer to chromosomes by name. Refer to chromosomes with a preceding - to have all chromosomes but those considered. A logical (TRUE/FALSE) vector may also be used.

chr
Column number in the phenotype matrix to be used as the phenotype. One may also give a character string matching a phenotype name. Finally, one may give a numeric vector of phenotypes, in which case it must have the length equal to the number of individuals in the cross, and there must be either non-integers or values < 1 or > no. phenotypes; this last case may be useful for studying transformations.

pheno.col
Column number in the phenotype matrix to be used as the phenotype. One may also give a character string matching a phenotype name. Finally, one may give a numeric vector of phenotypes, in which case it must have the length equal to the number of individuals in the cross, and there must be either non-integers or values < 1 or > no. phenotypes; this last case may be useful for studying transformations.

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

covar
A matrix or data.frame of covariates. These must be strictly numeric.

formula
An object of class formula indicating the model to be fitted. (It can also be the character string representation of a formula.) QTLs are referred to as Q1, Q2, etc. Covariates are referred to by their names in the data frame covar. If the new QTL are not included in the formula, a two-dimensional scan as in scantwo is performed.

method
Indicates whether to use multiple imputation or Haley-Knott regression.

model
The phenotype model: the usual model or a model for binary traits.

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" or stepwidth="max", we force incl.markers=TRUE.

verbose
If TRUE, display information about the progress of calculations. If verbose is an integer > 1, further messages from scanqtl are also displayed.

tol
Tolerance for convergence for the binary trait model.

maxit
Maximum number of iterations for fitting the binary trait model.

forceXcovar
If TRUE, force inclusion of X-chr-related covariates (like sex and cross direction).

Details

The formula is used to specified 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.

We enforce a hierarchical structure on the model formula: if a QTL or covariate is in involved in an interaction, its main effect must also be included.

If neither of the two new QTL are indicated in the formula, we perform a two-dimensional scan as in scantwo. That is, for each pair of QTL positions, we fit two models: two additive QTL added to the formula, and two interacting QTL added to the formula.

If the both of the new QTL are indicated in the formula, that particular model is fit, with the positions of the new QTL allowed to vary across the genome. If just one of the QTL is indicated in the formula, a main effect for the other is added, and that particular model is fit, again with the
positions of both QTL varying. Note that in this case the LOD scores are not analogous to those produced by scantwo. Thus, there slightly modified forms for the plots (produced by plot.scantwo) and summaries (produced by summary.scantwo and max.scantwo). In the plot, the x-axis is to be interpreted as the position of the first of the new QTL, and the y-axis is to be interpreted as the position of the second of the new QTL. In the summaries, we give the single best pair of positions on each pair of chromosomes, and give LOD scores comparing that pair of positions to the base model (without each of these QTL), and to the base model plus one additional QTL on one or the other of the chromosomes.

Value
An object of class scantwo, as produced by scantwo.
If neither of the new QTL were indicated in the formula, the result is just as in scantwo, though with LOD scores relative to the base model (omitting the new QTL).
Otherwise, the results are contained in what would ordinarily be in the full and additive LOD scores, with the additive LOD scores corresponding to the case that the first of the new QTL is to the left of the second of the new QTL, and the full LOD scores corresponding to the case that the first of the new QTL is to the right of the second of the new QTL. Because the structure of the LOD scores in this case is different from those output by scantwo, we include, in this case, an attribute "addpair"=TRUE. (We also require results of single-dimensional scans, omitting each of the two new QTL from the formula, one at a time; these are included as attributes "lod.minus1" and "lod.minus2"). The results are then treated somewhat differently by summary.scantwo, max.scantwo, and plot.scantwo. See the Details section.

Author(s)
Karl W Broman, <broman@wisc.edu>

References

See Also
addint, addqtl, fitqtl, makeqtl, scanqtl, refineqtl, makeqtl, scantwo, addtoqtl

Examples
# A totally contrived example to show some of what you can do

# simulate backcross data with 3 chromosomes (names "17", "18", "19")
# one QTL on chr 17 at 40 cM
# one QTL on chr 18 at 30 cM
# two QTL on chr 19, at 10 and 40 cM
data(map10)
model <- rbind(c(1,40,0), c(2,30,0), c(3,10,0), c(3,40,0))
## Not run: fakebc <- sim.cross(map10[17:19], model=model, type="bc", n.ind=250)

# het at QTL on 17 and 1st QTL on 19 increases phenotype by 1 unit
# het at QTL on 18 and 2nd QTL on 19 decreases phenotype by 1 unit
qtlgeno <- fakebc$qtlgeno
phe <- rnorm(nind(fakebc))
w <- qtlgeno[,1]==2 & qtlgeno[,3]==2
phe[w] <- phe[w] + 1
w <- qtlgeno[,2]==2 & qtlgeno[,4]==2
phe[w] <- phe[w] - 1
fakebc$pheno[,1] <- phe

## Not run: fakebc <- calc.genoprob(fakebc, step=2, err=0.001)

# base model has QTLs on chr 17 and 18
qtl <- makeqtl(fakebc, chr=c("17", "18"), pos=c(40,30), what="prob")

# scan for an additional pair of QTL, one interacting with the locus
# on 17 and one interacting with the locus on 18
out.ap <- addpair(fakebc, qtl=qtl, formula = y~Q1*Q3 + Q2*Q4, method="hk")
max(out.ap)
summary(out.ap)
plot(out.ap)

addqtl

### Description
Scan for an additional QTL in a multiple-QTL model.

### Usage

```r
addqtl(cross, chr, pheno.col=1, qtl, covar=NULL, formula,
method=c("imp","hk"), model=c("normal", "binary"),
incl.markers=TRUE, verbose=FALSE, tol=1e-4, maxit=1000,
forceXcovar=FALSE, require.fullrank=FALSE)
```

### Arguments

- **cross**: An object of class cross. See `read.cross` for details.
- **chr**: Optional vector indicating the chromosomes to be scanned. If missing, all chromosomes are scanned. Refer to chromosomes by name. Refer to chromosomes with a preceding - to have all chromosomes but those considered. A logical (TRUE/FALSE) vector may also be used.
pheno.col  Column number in the phenotype matrix to be used as the phenotype. One may also give a character string matching a phenotype name. Finally, one may give a numeric vector of phenotypes, in which case it must have the length equal to the number of individuals in the cross, and there must be either non-integers or values < 1 or > no. phenotypes; this last case may be useful for studying transformations.

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

covar  A matrix or data.frame of covariates. These must be strictly numeric.

formula  An object of class formula indicating the model to be fitted. (It can also be the character string representation of a formula.) QTLs are referred to as Q1, Q2, etc. Covariates are referred to by their names in the data frame covar. If the new QTL is not included in the formula, its main effect is added.

method  Indicates whether to use multiple imputation or Haley-Knott regression.

model  The phenotype model: the usual model or a model for binary traits

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" or stepwidth="max", we force incl.markers=TRUE.

verbose  If TRUE, display information about the progress of calculations. If verbose is an integer > 1, further messages from scanqtl are also displayed.

tol  Tolerance for convergence for the binary trait model.

maxit  Maximum number of iterations for fitting the binary trait model.

forceXcovar  If TRUE, force inclusion of X-chr-related covariates (like sex and cross direction).

require.fullrank  If TRUE, give LOD=0 when covariate matrix in the linear regression is not of full rank.

Details

The formula is used to specified 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.

We enforce a hierarchical structure on the model formula: if a QTL or covariate is in involved in an interaction, its main effect must also be included.

If one wishes to scan for QTL that interact with another QTL, include it in the formula (with an index of one more than the number of QTL in the input qtl object).

Value

An object of class scanone, as produced by the scanone function. LOD scores are relative to the base model (with any terms that include the new QTL omitted).

Author(s)

Karl W Broman, <broman@wisc.edu>
addtoqtl

Add to a qtl object

Description

Add a QTL or multiple QTL to a qtl object.

Usage

addtoqtl(cross, qtl, chr, pos, qtl.name, drop.lod.profile=TRUE)
addtoqtl

Arguments

- **cross**: An object of class `cross`. See `read.cross` for details.
- **qtl**: The `qtl` object to which additional QTL are to be added.
- **chr**: Vector indicating the chromosome for each new QTL. (These should be character strings referring to the chromosomes by name.)
- **pos**: Vector (of same length as `chr`) indicating the positions on the chromosome for each new QTL. If there is no marker or pseudomarker at a position, the nearest position is used.
- **qtl.name**: Optional user-specified name for each new 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 Chromsome 1 at 10 cM.
- **drop.lod.profile**: If TRUE, remove any LOD profiles from the object.

Value

An object of class `qtl`, just like the input `qtl` object, but with additional QTL added. See `makeqtl` for details.

Author(s)

Karl W Broman, <broman@wisc.edu>

See Also

`makeqtl`, `fitqtl`, `dropfromqtl`, `replaceqtl`, `reorderqtl`

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 <- calc.genoprob(fake.f2, step=2, err=0.001)
qtl <- makeqtl(fake.f2, qc, qp, what="prob")
qtl <- addtoqtl(fake.f2, qtl, 14, 35)
allchsplits

Test all possible splits of a chromosome into two pieces

Description

In order to assess the support for a linkage group, this function splits the linkage groups into two pieces at each interval and in each case calculates a LOD score comparing the combined linkage group to the two pieces.

Usage

allchsplits(cross, chr, error.prob=0.0001,
        map.function=c("haldane","kosambi","c-f","morgan"),
        m=0, p=0, maxit=4000, tol=1e-6, sex.sp=TRUE,
        verbose=TRUE)

Arguments

cross     An object of class cross. See read.cross for details.
chr       A vector specifying which chromosomes to study. This should be a vector of character strings referring to chromosomes by name; numeric values are converted to strings. Refer to chromosomes with a preceding - to have all chromosomes but those considered. A logical (TRUE/FALSE) vector may also be used.
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. (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   If TRUE, print information on progress.
argmax.geno

Value

A data frame (actually, an object of class "scanone", so that one may use `plot.scanone`, `summary.scanone`, etc.) with each row being an interval at which a split is made. The first two columns are the chromosome ID and midpoint of the interval. The third column is a LOD score comparing the combined linkage group to the split into two linkage groups. A fourth column (gap) indicates the length of each interval.

The row names indicate the flanking markers for each interval.

Author(s)

Karl W Broman, `<broman@wisc.edu>`

See Also

`est.map`, `ripple.est.rf`, `switch.order`, `movemarker`

Examples

```r
data(fake.bc)
allchrplits(fake.bc, 7, error.prob=0, verbose=FALSE)
```

---

**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

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

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).
**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 recommend using "fixed"; "variable" was included for the qtlbim package (https://cran.r-project.org/src/contrib/Archive/qtlbim). The "max" option inserts the minimal number of intermediate points so that the maximum distance between points is step.

**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, AAAAAH, AAAHHH, AAHHHH, AAAAAH. The Viterbi algorithm produces a single "most likely" sequence of underlying genotypes.

**Author(s)**

Karl W Broman, <broman@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)
**Description**

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

**Usage**

```r
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, `<broman@wisc.edu>`

**Examples**

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

# covariates
ac <- pull.pheno(fake.bc, c("sex","age"))
ic <- pull.pheno(fake.bc, "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
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, <broman@wisc.edu>

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

# covariates
ac <- pull.pheno(fake.bc, c("sex","age"))
ic <- pull.pheno(fake.bc, "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)
# re-set the seed
cset.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
```r
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, <broman@wisc.edu>

### See Also
`est.rf`, `ripple`, `est.map`, `sim.cross`
Examples

data(badorder)

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

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

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

bayesint  Bayesian credible interval

Description
Calculate an approximate Bayesian credible interval for a particular chromosome, using output from scanone.

Usage
bayesint(results, chr, qtl.index, prob=0.95, lodcolumn=1, expandtomarkers=FALSE)

Arguments
results  Output from scanone, or a qtl object as output from refineqtl.
chr      A chromosome ID (if input results are from scanone (should have length 1).
qtl.index Numeric index for a QTL (if input results are from refineqtl (should have length 1).
prob     Probability coverage of the interval.
lodcolumn An integer indicating which of the LOD score columns should be considered (if input results are from scanone).
expandtomarkers  If TRUE, the interval is expanded to the nearest flanking markers.

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 scanone indicating the estimated QTL position and the approximate endpoints for the Bayesian credible interval.
Author(s)

Karl W Broman, <broman@wisc.edu>

See Also

scanone, lodint

Examples

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)
bayesint(out, chr=4, expandtomarkers=TRUE)

bristle3  Data on bristle number in Drosophila

Description

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

Usage

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

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.
Author(s)

Karl W Broman, <broman@wisc.edu>

See Also

subset.cross

Examples

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

---

c.scanone

Combine columns from multiple scanone results

Description

Concatenate the columns from different runs of scanone.

Usage

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

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

Arguments

... A set of objects of class scanone. (This can also be a list of scanone objects.)
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, <broman@wisc.edu>

See Also
summary.scanone, scanone, cbind.scanoneperm

Examples

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

## S3 method for class 'scanoneperm'
c(...)
## S3 method for class 'scanoneperm'
bind(...)

Arguments

... A set of objects of class scanoneperm. (This can also be a list of scanoneperm objects.) 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, <broman@wisc.edu>

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.scantwo

**Combine columns from multiple scantwo results**

**Description**

Concatenate the columns from different runs of **scantwo**.

**Usage**

```r
## S3 method for class 'scantwo'
c(...)
## S3 method for class 'scantwo'
cbind(...)
```

**Arguments**

...  
A set of objects of class scantwo. (This can also be a list of scantwo objects.) These are the results from **scantwo** (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.

**Details**

The aim of this function is to concatenate the results from multiple runs **scantwo**, generally for different phenotypes and/or methods.

**Value**

The concatenated input, as a scantwo object.
c.scantwoperm

Author(s)
Karl W Broman, <broman@wisc.edu>

See Also
summary.scantwo, scantwo, c.scanone

Examples

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

out2a <- scantwo(fake.bc, method="hk")
out2b <- scantwo(fake.bc, pheno.col=2, method="hk")

out2 <- c(out2a, out2b)

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. (This can also be a list of scantwoperm objects.) 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, <broman@wisc.edu>

See Also
summary.scantwoperm, scantwo, cbind.scantwoperm

Examples

data(fake.f2)

fake.f2 <- calc.genoprob(fake.f2)
## Not run: operm1 <- scantwo(fake.f2, method="hk", n.perm=50)
operm2 <- scantwo(fake.f2, method="hk", n.perm=50)
## End(Not run)

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 (preferred) version.
calc.errorlod

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 calculation 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 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, plotErrorlod to plot the error LOD scores for specified chromosomes, and plotGeno 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, <broman@wisc.edu>

References


See Also

plotErrorlod, top.errorlod, cleanGeno

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
plotGeno(hyper, chr=1, ind=160:200, cutoff=7, min.sep=2)
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","max"))

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(observed genotype \mid 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 with fixed or variable step sizes. We recommend using "fixed"; "variable" was included for the qtlbim package (https://cran.r-project.org/src/contrib/Archive/qtlbim). The "max" option inserts the minimal number of intermediate points so that the maximum distance between points is step.

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, <broman@wisc.edu>

References


See Also

sim.geno, argmax.geno, calc.errorlod

Examples

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)

calc.penalties (Calculate LOD penalties)

Description

Derive penalties for the penalized LOD scores (used by stepwiseqtl) on the basis of permutation results from a two-dimensional, two-QTL scan (obtained by scantwo).

Usage

calc.penalties(perms, alpha=0.05, lodcolumn)

Arguments

perms Permutation results from scantwo.
alpha Significance level.
lodcolumn If the scantwo permutation results contain LOD scores for multiple phenotypes, this argument indicates which to use in the summary. This may be a vector. If missing, penalties for all phenotypes are calculated.
Details

Thresholds derived from scantwo permutations (that is, for a two-dimensional, two-QTL genome scan) are used to calculate penalties on main effects and interactions.

The main effect penalty is the $1 - \alpha$ quantile of the null distribution of the genome-wide maximum LOD score from a single-QTL genome scan (as with scanone).

The "heavy" interaction penalty is the $1 - \alpha$ quantile of the null distribution of the maximum interaction LOD score (that is, the $\log_{10}$ likelihood ratio comparing the best model with two interacting QTL to the best model with two additive QTL) from a two-dimensional, two-QTL genome scan (as with scantwo).

The "light" interaction penalty is the difference between the "fv1" threshold from the scantwo permutations (that is, the $1 - \alpha$ quantile of the LOD score comparing the best model with two interacting QTL to the best single-QTL model) and the main effect penalty.

If the permutations results were obtained with perm.Xsp=TRUE, to give X-chr-specific results, six penalties are calculated: main effect for autosomes, main effect for X chr, heavy penalty on A:A interactions, light penalty on A:A interactions, penalty on A:X interactions, and penalty on X:X interactions.

Value

Vector of three values indicating the penalty on main effects and heavy and light penalties on interactions, or a matrix of such results, with each row corresponding to a different phenotype.

If the input permutations are X-chromosome-specific, the result has six values: main effect for autosomes, main effect for X chr, heavy penalty on A:A interactions, light penalty on A:A interactions, penalty on A:X interactions, and penalty on X:X interactions.

Author(s)

Karl W Broman, <broman@wisc.edu>

References


See Also

scantwo, stepwiseqtl

Examples

data(fake.f2)

fake.f2 <- calc.genoprob(fake.f2, step=5)

out.2dim <- scantwo(fake.f2, method="hk")

# permutations
cbind.scanoneperm

Combine columns from multiple scanone permutation results

Description

Concatenate the columns from different runs of \texttt{scanone} with \texttt{n.perm} \(> 0\).

Usage

\begin{verbatim}
## S3 method for class 'scanoneperm'
cbind(..., labels)
\end{verbatim}

Arguments

\begin{itemize}
  \item \texttt{...} A set of objects of class \texttt{scanoneperm}. These are the permutation results from \texttt{scanone} (that is, when \texttt{n.perm} \(> 0\)), generally run with different phenotypes or methods.
  \item \texttt{labels} A vector of character strings, of length 1 or of the same length as the input \ldots, to be appended to the column names in the output.
\end{itemize}

Details

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

Value

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

Author(s)

Karl W Broman, \texttt{<broman@wisc.edu>}

See Also

\texttt{summary.scanoneperm, scanone, c.scanoneperm, c.scanone}
cbind.scantwoperm

Combine scantwo permutations by column

Description

Column-bind permutations results from scantwo for multiple phenotypes or models.

Usage

## S3 method for class 'scantwoperm'
cbind(...)  

Arguments

...  

A set of objects of class scantwoperm. (This can also be a list of scantwoperm objects.) These are the permutation results from scantwo (that is, when n.perm > 0). These must all concern the same number of permutations.

Value

The column-binded input, as a scantwoperm object.

Author(s)

Karl W Broman, <broman@wisc.edu>

See Also

scantwo, c.scantwoperm, summary.scantwoperm

Examples

data(fake.bc)

fake.bc <- calc.genoprob(fake.bc)

## Not run: operm1 <- scantwo(fake.bc, pheno.col=1, method="hk", n.perm=50)  
## End(Not run)

operm2 <- scantwo(fake.bc, pheno.col=2, method="hk", n.perm=50)  

operm <- cbind(operm1, operm2)
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 recombination 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, <broman@wisc.edu>

See Also

est.rf, geno.crosstab, switchAlleles
Examples

data(fake.f2)

# switch homozygotes at marker D5M391
fake.f2 <- switchAlleles(fake.f2, "D5M391")

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

chrlen  Chromium 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, <broman@wisc.edu>

See Also
summaryMap, pull.map, summary.cross

Examples

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

map <- pull.map(fake.f2)
chrlen(map)
chrnames

Pull out the chromosome names from a cross

Description

Pull out the chromosome names from a cross object as one big vector.

Usage

chrnames(cross)

Arguments

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

Value

A vector of character strings (the chromosome names).

Author(s)

Karl W Broman, <broman@wisc.edu>

See Also

markernames, phenames

Examples

data(listeria)
chrnames(listeria)

---

cim

Composite interval mapping

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

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. 
                      One may also give a character string matching a phenotype name. Finally, one 
                      may give a numeric vector of phenotypes, in which case it must have the length 
                      equal to the number of individuals in the cross, and there must be either non- 
                      integers or values < 1 or > no. phenotypes; this last case may be useful for 
                      studying transformations.
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 geno- 
                      type 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 ob- 
                      served 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.perm is missing, the function returns the scan results as a data.frame with three columns: chromo- 
some, position, LOD score. Attributes indicate the names and positions of the chosen marker 
covariates.

If n.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, &lt;broman@wisc.edu&gt;
References


See Also

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

Examples

```r
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.
- `...` Ignored at this point.

**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, <broman@wisc.edu>

See Also
drop.nullmarkers, drop.markers, clean.scantwo

Examples

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 not separated by n.mar markers, or that are less than
distance cM apart, 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
## S3 method for class 'scantwo'
clean(object, n.mar=1, distance=0, ...)

Arguments

object         An object of class scantwo. See scantwo for details.
n.mar          Pairs of positions not separated by at least this many markers have LOD scores
                set to 0.
distance       Pairs of positions not separated by at least this distance have LOD scores set to
                0.
...            Ignored at this point.

Value
The input scanttwo object, with any negative or missing LOD scores replaced by 0, and LOD scores
for pairs of positions separated by fewer than n.mar markers, or less than distance cM, are set to
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.


**cleanGeno**

Delete genotypes that are possibly in error

**Description**

Delete genotypes from a cross that are indicated to be possibly in error, as they result in apparent tight double-crossovers.

**Usage**

cleanGeno(cross, chr, maxdist=2.5, maxmark=2, verbose=TRUE)

**Arguments**

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

chr Optional vector indicating the chromosomes to consider. This should be a vector of character strings referring to chromosomes by name; numeric values are converted to strings. Refer to chromosomes with a preceding - to have all chromosomes but those considered. A logical (TRUE/FALSE) vector may also be used.

maxdist A vector specifying the maximum distance between two crossovers.

maxmark A vector specifying the maximum number of typed markers between two crossovers.

verbose If TRUE, print information on the numbers of genotypes omitted from each chromosome.

**Examples**

data(fake.f2)

fake.f2 <- calc.genoprob(fake.f2, step=5)
out2 <- scantwo(fake.f2, method="hk")
out2 <- clean(out2)
out2cl2 <- clean(out2, n.mar=2, distance=5)
Details

We first use `locateXO` to identify crossover locations. If a pair of adjacent crossovers are separated by no more than `maxdist` and contain no more than `maxmark` genotyped markers, the intervening genotypes are omitted (that is, changed to NA).

The arguments `maxdist` and `maxmark` may be vectors. (If both have length greater than 1, they must have the same length.) If they are vectors, genotypes are omitted if they satisfy any one of the `((maxdist, maxmark))` pairs.

Value

The input cross object with suspect genotypes omitted.

Author(s)

Karl W Broman, <broman@wisc.edu>

See Also

`locateXO`, `countXO`, `calc.errorlod`

Examples

```r
data(hyper)
sum(ntyped(hyper))
hyperc <- cleanGeno(hyper, chr=4, maxdist=c(2.5, 10), maxmark=c(2, 1))
sum(ntyped(hyperc))
```

`comparecrosses`  
*Compare two cross objects*

Description

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

Usage

```r
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.
comparegeno

Author(s)
Karl W Broman, <broman@wisc.edu>

See Also
summary.cross

Examples

data(listeria)
comparecrosses(listeria, listeria)

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","both"))

Arguments

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

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

If called with what="both", the lower triangle contains the proportion and the upper triangle contains the number.

If called with what="proportion", the diagonal contains missing values. Otherwise, the diagonal contains the number of typed markers for each individual.

The output is given class "comparegeno" so that appropriate summary and plot functions may be used.

Author(s)
Karl W Broman, <broman@wisc.edu>
See Also

`comparegeno`, `summary.comparegeno`, `plot.comparegeno`

Examples

```r
data(listeria)

cg <- comparegeno(listeria)

summary(cg, 0.7)
plot(cg)
```

### Description

Compare the likelihood of an alternative order for markers on a chromosome to the current order.

### Usage

```r
compareorder(cross, chr, order, error.prob=0.0001,
             map.function=c("haldane","kosambi","c-f","morgan"),
             maxit=4000, tol=1e-6, sex.sp=TRUE)
```

### Arguments

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

- `chr`:
  - The chromosome to investigate. Only one chromosome is allowed. (This should be a character string referring to the chromosomes by name.)

- `order`:
  - The alternate order of markers on the chromosome: a numeric vector that is a permutation of the integers from 1 to the number of markers on the chromosome.

- `error.prob`:
  - Assumed genotyping error rate used in the calculation of the penetrance \( Pr(\text{observed genotype | true genotype}) \).

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

- `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

A data frame with two rows: the current order in the input cross object, and the revised order. The first column is the \( \log_{10} \) likelihood of the new order relative to the original one (positive values indicate that the new order is better supported). The second column is the estimated genetic length of the chromosome for each order. In the case of sex-specific maps, there are separate columns for the female and male genetic lengths.
condense.scantwo

Author(s)

Karl W Broman, <broman@wisc.edu>

See Also

ripple, switch.order, movemarker

Examples

data(badorder)
compareorder(badorder, chr=1, order=c(1:8,11,10,9,12))

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, <broman@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.map**  
*Change map function for a genetic map*

---

**Description**

Convert a genetic map from using one map function to another.

**Usage**

```r
## S3 method for class 'map'
convert(object, old.map.function=c("haldane", "kosambi", "c-f", "morgan"),
    new.map.function=c("haldane", "kosambi", "c-f", "morgan"), ...)
```

**Arguments**

- `object`  
  A genetic map object, of class "map": A list whose components are vectors of marker locations.

- `old.map.function`  
  The map function used in forming the map in `object`.

- `new.map.function`  
  The new map function to be used.

- `...`  
  Ignored at this point.

**Details**

The location of the first marker on each chromosome is left unchanged. Inter-marker distances are converted to recombination fractions with the inverse of the `old.map.function`, and then back to distances with the `new.map.function`.

**Value**

The same as the input, but with inter-marker distances changed to reflect a different map function.

**Author(s)**

Karl W Broman, <broman@wisc.edu>
convert.scanone

See Also

est.map, replace.map

Examples

data(listeria)
map <- pull.map(listeria)
map <- convert(map, "haldane", "kosambi")
listeria <- replace.map(listeria, map)

---

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

## S3 method for class 'scanone'
convert(object, ...)

Arguments

object  Output from the function scanone, for R/qtl version 0.97 and earlier.
...
    Ignored at this point.

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, <broman@wisc.edu>

See Also

scanone, convert.scantwo

Examples

## 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.
- `...`  
  Ignored at this point.

**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, <broman@wisc.edu>

**See Also**

- `scantwo`
- `convert.scanone`

**Examples**

```r
## Not run: out2.new <- convert(out2.old)
```
**convert2riself**  
*Convert a cross to RIL by selfing*

**Description**
Convert a cross to type "riself" (RIL by selfing).

**Usage**
```r
closest2riself(cross)
```

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

**Details**
If there are more genotypes with code 3 (BB) than code 2 (AB), we omit the genotypes with code==2 and call those with code==3 the BB genotypes.

If, instead, there are more genotypes with code 2 than code 3, we omit the genotypes with code==3 and call those with code==2 the BB genotypes.

Any chromosomes with class "X" (X chromosome) are changed to class "A" (autosomal).

**Value**
The input cross object, with genotype codes possibly changed and cross type changed to "riself".

**Author(s)**
Karl W Broman, <broman@wisc.edu>

**See Also**
`convert2risib`

**Examples**
```r
data(hyper)
hyper.as.riself <- convert2riself(hyper)
```
convert2risib

Convert a cross to RIL by sib mating

Description

Convert a cross to type "risib" (RIL by sib mating).

Usage

convert2risib(cross)

Arguments

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

Details

If there are more genotypes with code 3 (BB) than code 2 (AB), we omit the genotypes with code==2 and call those with code==3 the BB genotypes.

If, instead, there are more genotypes with code 2 than code 3, we omit the genotypes with code==3 and call those with code==2 the BB genotypes.

Value

The input cross object, with genotype codes possibly changed and cross type changed to "risib".

Author(s)

Karl W Broman, <broman@wisc.edu>

See Also

convert2riself

Examples

data(hyper)
hyper.as.risib <- convert2risib(hyper)
**convert2sa**  

Converting a sex-specific map to a sex-averaged one

### Description

Convert a sex-specific map to a sex-averaged one, assuming that the female and male maps are actually the same (that is, that the map was estimated assuming a common recombination rate in females and males).

### Usage

```
convert2sa(map, tol=1e-4)
```

### Arguments

- **map**  
  A map object with sex-specific locations (but assuming that the female and male maps are the same), as output by the function `est.map` for a 4-way cross, with argument `sex.sp=FALSE`.

- **tol**  
  Tolerance value for inspecting the differences between the female and male maps; if they differ by more than this tolerance, a warning is issued.

### Details

We pull out just the female marker locations, and give a warning if there are large differences between the female and male maps.

### Value

A map object, with sex-averaged distances.

### Author(s)

Karl W Broman, <broman@wisc.edu>

### See Also

`est.map`, `plotMap`

### Examples

```r
data(fake.4way)
## Not run: fake.4way <- subset(fake.4way, chr="-X")

nm <- est.map(fake.4way, sex.sp=FALSE)
plot(convert2sa(nm))
```
countXO  

Count number of obligate crossovers for each individual

Description

Count the number of obligate crossovers for each individual in a cross, either by chromosome or overall.

Usage

countXO(cross, chr, bychr=FALSE)

Arguments

cross An object of class cross. See read.cross for details.
chr Optional vector indicating the chromosomes to investigate. This should be a vector of character strings referring to chromosomes by name; numeric values are converted to strings. Refer to chromosomes with a preceding - to have all chromosomes but those considered. A logical (TRUE/FALSE) vector may also be used.
bychr If TRUE, return counts for each individual chromosome; if FALSE, return the overall number across the selected chromosomes.

Details

For each individual we count the minimal number of crossovers that explain the observed genotype data.

Value

If bychr=TRUE, a matrix of counts is returned, with rows corresponding to individuals and columns corresponding to chromosomes.
If bychr=FALSE, a vector of counts (the total number of crossovers across all selected chromosomes) is returned.

Author(s)

Karl W Broman, <broman@wisc.edu>

See Also

ripple, locateXO, cleanGeno

Examples

data(hyper)
plot(countXO(hyper))
drop.dupmarkers

Description

Drop markers with duplicate names; retaining the first of each set, with consensus genotypes

Usage

drop.dupmarkers(cross, verbose=TRUE)

Arguments

cross An object of class cross. See read.cross for details.
verbose If TRUE, print information on the numbers of genotypes and markers omitted. If > 1, give more detailed information on genotypes omitted.

Value

The input cross object, with any duplicate markers omitted (except for one). The marker retained will have consensus genotypes; if multiple versions of a marker have different genotypes for an individual, they will be replaced by NA.

Any derived data (such as produced by calc.genoprob) will be stripped off.

Author(s)

Karl W Broman, <broman@wisc.edu>

See Also

drop.nullmarkers, pull.markers, drop.markers, summary.cross, clean.cross

Examples

data(listeria)

listeria <- drop.dupmarkers(listeria)
drop.markers: Drop a set of markers

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

Usage
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, <broman@wisc.edu>

See Also
drop.nullmarkers, pull.markers, 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
drop.nullmarkers(cross)
**dropfromqtl**

### 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, <broman@wisc.edu>

### See Also

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

### Examples

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

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

---

**dropfromqtl**

*Drop a QTL from a qtl object*

### Description

Drop a QTL or multiple QTL from a QTL object

### Usage

```r
dropfromqtl(qtl, index, chr, pos, qtl.name, drop.lod.profile=TRUE)
```

### Arguments

- `qtl` A qtl object, as created by `makeqtl`.
- `index` Vector specifying the numeric indices of the QTL to be dropped.
- `chr` Vector indicating the chromosome for each QTL to drop.
- `pos` Vector (of same length as `chr`) indicating the positions of the QTL to be dropped.
- `qtl.name` Vector specifying the names of the QTL to be dropped.
- `drop.lod.profile` If TRUE, remove any LOD profiles from the object.
Details

Provide either chr and pos, or one of qtl.name or index.

Value

The input qtl object with the specified QTL omitted. See makeqtl for details on the format.

Author(s)

Karl W Broman, broman@wisc.edu

See Also

makeqtl, fitqtl, addtoqtl, replaceqtl, reorderqtl

Examples

data(fake.f2)

# take out several QTLs and make QTL object
cq <- c(1, 6, 13)
cp <- c(25.8, 33.6, 18.63)
fake.f2 <- subset(fake.f2, chr=cq)
fake.f2 <- calc.genoprob(fake.f2, step=2, err=0.001)
qtl <- makeqtl(fake.f2, cq, cp, what="prob")
newqtl <- dropfromqtl(qtl, chr=1, pos=25.8)
altqtl <- dropfromqtl(qtl, index=1)

---

droponemarker  

*Drop one marker at a time and determine effect on genetic map*

Description

Drop one marker at a time from a genetic map and calculate the change in log likelihood and in the chromosome length, in order to identify problematic markers.

Usage

droponemarker(cross, chr, error.prob=0.0001,
               map.function=c("haldane","kosambi","c-f","morgan"),
               m=0, p=0, maxit=4000, tol=1e-6, sex.sp=TRUE,
               verbose=TRUE)
Arguments

cross     An object of class cross. See read.cross for details.
chr       A vector specifying which chromosomes to test for the position of the marker. This should be a vector of character strings referring to chromosomes by name; numeric values are converted to strings. Refer to chromosomes with a preceding - to have all chromosomes but those considered. A logical (TRUE/FALSE) vector may also be used.
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. (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   If TRUE, print information on progress; if > 1, print even more information.

Value

A data frame (actually, an object of class "scanone", so that one may use plot.scanone, summary.scanone, etc.) with each row being a marker. The first two columns are the chromosome ID and position. The third column is a LOD score comparing the hypothesis that the marker is not linked to the hypothesis that it belongs at that position.

In the case of a 4-way cross, with sex.sp=TRUE, there are two additional columns with the change in the estimated female and male genetic lengths of the respective chromosome, upon deleting that marker. With sex.sp=FALSE, or for other types of crosses, there is one additional column, with the change in estimated genetic length of the respective chromosome, when the marker is omitted.

A well behaved marker will have a negative LOD score and a small change in estimated genetic length. A poorly behaved marker will have a large positive LOD score and a large change in estimated genetic length. But note that dropping the first or last marker on a chromosome could result in a large change in estimated length, even if they are not badly behaved; for these markers one should focus on the LOD scores, with a large positive LOD score being bad.

Author(s)

Karl W Broman, <broman@wisc.edu>

See Also

tryallpositions, est.map, ripple, est.rf, switch.order, movemarker, drop.markers
Examples

data(fake.bc)
droponemarker(fake.bc, 7, error.prob=0, verbose=FALSE)

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. One may also give a character string matching a phenotype name. Finally, one may give a numeric vector of phenotypes, in which case it must have the length equal to the number of individuals in the cross, and there must be either non-integers or values < 1 or > no. phenotypes; this last case may be useful for studying transformations.
mname1 Name for the first marker or pseudomarker. Pseudomarkers (that is, non-marker positions on the imputation grid) may be referred to in a form like "5@30.3", for position 30.3 on chromosome 5.
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.
geno1 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.
effectplot

- **legend.lab**: Optional title for the legend.
- **draw**: A logical value to indicate whether to 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, <broman@wisc.edu>

**See Also**

- `plotPXG`
- `find.marker`
- `effectscan`
- `find.pseudomarker`

**Examples**

```r
data(fake.f2)

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

########################################
# 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, mname=mname)

### output of the function contains the means and SEs
output <- effectplot(fake.f2, mname=mname)
output
```
### plot a phenotype

# Plot of sex-specific phenotype means,
# note that "sex" must be a phenotype name here

\[
effectplot(fake.f2, mname1="sex", geno1=c("F","M"))
\]

# alternatively:

\[
sex <- pull.pheno(fake.f2, "sex")
effectplot(fake.f2, mname1="Sex", mark1=sex, geno1=c("F","M"))
\]

########################################
# 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

##### refer to pseudomarkers by their positions

\[
effectplot(fake.f2, mname1="1@35", mname2="13@25")
\]

##### alternatively, find their names via find.pseudomarker

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

### Plot of sex- and genotype-specific phenotype means

\[
mname <- find.marker(fake.f2, 13, 24) # marker D13M254
# sex and a marker
\]

\[
effectplot(fake.f2, mname1=mname, mname2="Sex",
mark2=sex, geno2=c("F","M"))
\]

# Same as above, switch role of sex and the marker
# sex and marker

\[
effectplot(fake.f2, mname1="Sex", mark1=sex,
gen1=c("F","M"), mname2=mname)
\]

# X chromosome marker

\[
mname <- find.marker(fake.f2, "X", 14) # marker DXM66
\]

\[
effectplot(fake.f2, mname1=mname)
\]

# Two markers, including one on the X

\[
mname <- find.pseudomarker(fake.f2, c(13, "X"), c(24, 14))
effectplot(fake.f2, mname1=mname[1], mname2=mname[2])
\]
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. One may also give a character string matching a phenotype name.
chr Optional vector indicating the chromosomes to be drawn in the plot. This should be a vector of character strings referring to chromosomes by name; numeric values are converted to strings. Refer to chromosomes with a preceding - to have all chromosomes but those considered. A logical (TRUE/FALSE) vector may also be used.
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 an internal function `plot.effectscan` 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, <broman@wisc.edu>

References


See Also

effectplot, plotPXG, 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)

<table>
<thead>
<tr>
<th>est.map</th>
<th>Estimate genetic maps</th>
</tr>
</thead>
</table>

Description

Uses the Lander-Green algorithm (i.e., the hidden Markov model technology) to re-estimate the genetic map for an experimental cross.

Usage

est.map(cross, chr, error.prob=0.0001,
map.function=c("haldane","kosambi","c-f","morgan"),
m=0, p=0, maxit=10000, tol=1e-6, sex.sp=TRUE,
verbose=FALSE, omit.noninformative=TRUE, offset, n.cluster=1)
est.map

Arguments

cross An object of class cross. See read.cross for details.
chr Optional vector indicating the chromosomes to consider. This should be a vector of character strings referring to chromosomes by name; numeric values are converted to strings. Refer to chromosomes with a preceding - to have all chromosomes but those considered. A logical (TRUE/FALSE) vector may also be used.
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. (Ig-nored 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 If TRUE, print tracing information.
omit.noninformative If TRUE, on each chromosome, omit individuals with fewer than two typed markers, since they are not informative for linkage.
offset Defines the starting position for each chromosome. If missing, we use the starting positions that are currently present in the input cross object. This should be a single value (to be used for all chromosomes) or a vector with length equal to the number of chromosomes, defining individual starting positions for each chromosome. For a sex-specific map (as in a 4-way cross), we use the same offset for both the male and female maps.
n.cluster If the package snow is available calculations for multiple chromosomes are run in parallel using this number of nodes.

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 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, <broman@wisc.edu>

**References**


**See Also**

`map2table`, `plotMap`, `replace.map`, `est.rf`, `fitstahl`

**Examples**

```r
data(fake.f2)

newmap <- est.map(fake.f2)
logliks <- sapply(newmap, attr, "loglik")
plotMap(fake.f2, newmap)
fake.f2 <- replace.map(fake.f2, newmap)
```
**Description**

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

**Usage**

```r
est.rf(cross, maxit=10000, tol=1e-6)
```

**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, <broman@wisc.edu>

**See Also**

`plotRF`, `pull.rf`, `plot.rfmatrix`, `est.map`, `badorder`, `checkAlleles`

**Examples**

```r
data(badorder)
badorder <- est.rf(badorder)
plotRF(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, <broman@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)
plotRF(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)
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, <broman@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 <- pull.pheno(fake.bc, c("sex","age"))
ic <- pull.pheno(fake.bc, "sex")
out.covar <- scanone(fake.bc, pheno.col=1:2,
        addcovar=ac, intcovar=ic)

# summaries
summary(out.nocovar, thr=3, format="allpeaks")
summary(out.covar, thr=3, format="allpeaks")

# plots
plot(out.nocovar, out.covar, chr=c(2,5,10), lod=1, col="blue",
    lty=1:2, ylim=c(0,13))
plot(out.nocovar, out.covar, chr=c(2,5,10), lod=2, col="red",
    lty=1:2, add=TRUE)

fake.f2

Simulated data for an F2 intercross

Description

Simulated data for an F2 intercross, obtained using sim.cross.

Usage

data(fake.f2)

Format

An object of class cross. See 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 (AxB)x(AxB), and 1=B, meaning the cross was (AxB)x(BxA)).

Note that the X chromosome genotypes are coded in a special way (see 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. 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).
fill.geno

Author(s)
Karl W Broman, <broman@wisc.edu>

See Also
sim.cross, fake.bc, fake.4way, listeria, hyper, bristle3, bristleX

Examples

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

Description
Replace the genotype data for a cross with a version imputed either by simulation with sim.geno, by the Viterbi algorithm with argmax.geno, or simply filling in genotypes between markers that have matching genotypes.

Usage
fill.geno(cross, method=c("imp","argmax", "no_dbl_XO", "maxmarginal"),
error.prob=0.0001,
map.function=c("haldane","kosambi","c-f","morgan"),
min.prob=0.95)

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, using the Viterbi algorithm, as implemented in argmax.geno, by simply filling in missing genotypes between markers with matching genotypes, or by choosing (at each marker) the genotype with maximal marginal probability.

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.

min.prob For method="maxmarginal", genotypes with probability greater than this value will be imputed; those less than this value will be made missing.
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.**

- **With method**="imp", a single random imputation is performed, using `sim.geno`.
- **With method**="argmax", for each individual the most probable sequence of genotypes, given the observed data (via `argmax.geno`), is used.
- **With method**="no_dbl_XO", non-recombinant intervals are filled in; recombinant intervals are left missing. For example, a sequence of genotypes like `A---A---H---H---A` (with `A` and `H` corresponding to genotypes AA and AB, respectively, and with `-` being a missing value) will be filled in as `AAAAA---HHHHH---A`.
- **With method**="maxmarginal", the conditional genotype probabilities are calculated with `calc.genoprob`, and then at each marker, the most probable genotype is determined. This is taken as the imputed genotype if it has probability greater than `min.prob`; otherwise it is made missing.
- **With method**="no_dbl_XO" and method="maxmarginal", some missing genotypes likely remain.
- **With method**="maxmarginal", some observed genotypes may be made missing.

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, <broman@wisc.edu>

See Also

`sim.geno`, `argmax.geno`

Examples

```r
data(hyper)
out.mr <- scantwo(fill.geno(hyper,method="argmax"), method="mr")
plot(out.mr)
```

**find.flanking**

`find.flanking` function documentation.

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, plotPXG, find.markerpos, find.pseudomarker

Examples

data(listeria)
find.flanking(listeria, 5, 28)
find.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, index)

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.
index A vector of numeric indices of the markers within chromosomes.
Details

Provide one of pos or index.

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

If pos is specified and 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, positions specified by pos 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 (if pos is specified) or to the input numeric indices (in index is specified).

Author(s)

Karl W Broman, <broman@wisc.edu>

See Also

find.flanking, plotPXG, find.pseudomarker, effectplot, find.markerpos

Examples

data(listeria)
find.marker(listeria, 5, 28)
find.marker(listeria, 5, index=6)
find.marker(listeria, c(1, 5, 13), c(81, 28, 26))
Value

A vector of numeric indices, from 1, 2, ..., \text{totmar(cross)}, with NA for markers not found.

Author(s)

Danny Arends; Karl W Broman <broman@wisc.edu>

See Also

find.markerpos

Examples

data(hyper)
mar <- find.marker(hyper, 4, 30)
find.markerindex(hyper, mar)

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 \texttt{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, <broman@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

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

find.pseudomarker(cross, chr, pos, where=c("draws", "prob"), addchr=TRUE)
findDupMarkers

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`.
- **addchr**: If `TRUE`, include something like "c5." at the beginning of the names of non-pseudomarker locations, as in the output of `scanone`; if `FALSE`, don't include this sort of string, as in the genotype probabilities from `calc.genoprob`.

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, <broman@wisc.edu>

See Also

`find.flanking`, `plotPXG`, `effectplot`, `find.marker`, `find.markerpos`

Examples

```r
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")
```

Description

Identify sets of markers with identical genotype data.

Usage

```r
findDupMarkers(cross, chr, exact.only=TRUE, adjacent.only=FALSE)
```
findDupMarkers

Arguments

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

chr Optional vector specifying which chromosomes to consider. This may be a logical, numeric, or character string vector.

exact.only If TRUE, look only for markers that have matching genotypes and the same pattern of missing data; if FALSE, also look for cases where one the observed genotypes at marker match those at another, and where the first marker has missing genotype whenever the genotype for the second marker is missing.

adjacent.only If TRUE, look only for sets of markers that are adjacent to each other.

Details

If exact.only=TRUE, we look only for groups of markers whose pattern of missing data and observed genotypes match exactly. One marker (chosen at random) is selected as the name of the group (in the output of the function).

If exact.only=FALSE, we look also for markers whose observed genotypes are contained in the observed genotypes of another marker. We use a pair of nested loops, working from the markers with the most observed genotypes to the markers with the fewest observed genotypes.

Value

A list of marker names; each component is a set of markers whose genotypes match one other marker, and the name of the component is the name of the marker that they match.

Author(s)

Karl W Broman, <broman@wisc.edu>

See Also

drop.nullmarkers, drop.markers, pickMarkerSubset

Examples

data(hyper)

hyper <- drop.nullmarkers(hyper)

dupmar <- findDupMarkers(hyper) # finds 4 pairs
dupmar.adjonly <- findDupMarkers(hyper, adjacent.only=TRUE) # finds 4 pairs

dupmar.nexact <- findDupMarkers(hyper, exact.only=FALSE, adjacent.only=TRUE) # finds 6 pairs

# one might consider dropping the extra markers
totmar(hyper) # 173 markers
hyper <- drop.markers(hyper, unlist(dupmar.adjonly))
totmar(hyper) # 169 markers
**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
```r
fitqtl(cross, pheno.col=1, qtl, covar=NULL, formula, method=c("imp", "hk"),
       model=c("normal", "binary"), dropone=TRUE, get.ests=FALSE,
       run.checks=TRUE, tol=1e-4, maxit=1000, forceXcovar=FALSE)
```

### 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. One may also give a character string matching a phenotype name. Finally, one may give a numeric vector of phenotypes, in which case it must have the length equal to the number of individuals in the cross, and there must be either non-integers or values < 1 or > no. phenotypes; this last case may be useful for studying transformations.
- **qtl**: An object of class `qtl`, as output from `makeqtl`.
- **covar**: A matrix or data.frame of covariates. These must be strictly numeric.
- **formula**: An object of class `formula` indicating the model to be fitted. (It can also be the character string representation of a formula.) 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 multiple imputation or Haley-Knott regression.
- **model**: The phenotype model: the usual model or a model for binary traits
- **dropone**: If TRUE, do drop-one-term analysis.
- **get.ests**: If TRUE, return estimated QTL effects and their estimated variance-covariance matrix.
- **run.checks**: If TRUE, check the input formula and check for individuals with missing phenotypes or covariates.
- **tol**: Tolerance for convergence for the binary trait model.
- **maxit**: Maximum number of iterations for fitting the binary trait model.
- **forceXcovar**: If TRUE, force inclusion of X-chr-related covariates (like sex and cross direction).
Details

The formula is used to specified 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.

We enforce a hierarchical structure on the model formula: if a QTL or covariate is involved in an interaction, its main effect must also be included.

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 estimated percent variances explained for the QTL are simplify transformations of the conditional LOD scores by the formula $h^2 = 1 - 10^{-2/\text{LOD}}$. While these may be reasonable for unlinked, additive QTL, they can be completely wrong in the case of linked QTL, but we don’t currently have any alternative.

For model="binary", a logistic regression model is used.

The part to get estimated QTL effects is not complete for the case of the X chromosome and 4-way crosses. The values returned in these cases are based on a design matrix that is convenient for calculations but not easily interpreted.

The estimated QTL effects for a backcross are derived by the coding scheme ± 1/2 for AA and AB, so that the additive effect corresponds to the difference between phenotype averages for the two genotypes. For doubled haploids and RIL, the coding scheme is ± 1 for AA and BB, so that the additive effect corresponds to half the difference between the phenotype averages for the two homozygotes.

For an intercross, the additive effect is derived from the coding scheme -1/0/+1 for genotypes AA/AB/BB, and so is half the difference between the phenotype averages for the two homozygotes. The dominance deviation is derived from the coding scheme 0/+1/0 for genotypes AA/AB/BB, and so is the difference between the phenotype average for the heterozygotes and the midpoint between the phenotype averages for the two homozygotes.

Epistatic effects and QTL × covariate interaction effects are obtained through the products of the corresponding additive/dominant effect columns.

Value

An object of class fitqtl. It may contains as many as four components:

- 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).
- lod is the LOD score from the fit of the full model.
- 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.
- ests contains the estimated QTL effects and standard errors.
When method="normal", residuals are saved as an attribute of the output, named "residuals" and accessible via the attr function.

The part to get estimated QTL effects is fully working only for the case of autosomes in a backcross, intercross, RIL or doubled haploids. 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; Karl W Broman, <broman@wisc.edu>

References

See Also
summary.fitqtl, makeqtl, scanqtl, refineqtl, addtoqtl, dropfromqtl, replaceqtl, reorderqtl

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 <- calc.genoprob(fake.f2, step=2, err=0.001)
qtl <- makeqtl(fake.f2, qc, qp, what="prob")

# fit model with 3 interacting QTLs interacting
# (performing a drop-one-term analysis)
lod <- fitqtl(fake.f2, pheno.col=1, qtl, formula=y~Q1*Q2*Q3, method="hk")
summary(lod)

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

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

# fit the same with an additive model
lod.sex.add <- fitqtl(fake.f2, pheno.col=1, qtl, formula=y~Q1+Q2+Q3+Sex,
```r
summary(lod.sex.add)
# residuals
residuals <- attr(lod.sex.add, "residuals")
plot(residuals)
## 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`: Optional vector indicating the chromosomes to consider. This should be a vector of character strings referring to chromosomes by name; numeric values are converted to strings. Refer to chromosomes with a preceding `-` to have all chromosomes but those considered. A logical (TRUE/FALSE) vector may also be 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 \( \text{max}m \) 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, <broman@wisc.edu>

References


See Also

`est.map, sim.cross`

Examples

```r
# Simulate genetic map: one chromosome of length 200 cM with
# a 2 cM marker spacing
mymap <- sim.map(200, 51, anchor.tel=TRUE, include.x=FALSE,
                 sex.sp=FALSE, eq.spacing=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
## Not run: 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)

---

flip.order | Flip the orders of markers on a set of chromosomes

### Description
Flip the orders of markers on a specified set of chromosome, so that the markers will be in the reverse order.

### Usage
```r
flip.order(cross, chr)
```

### Arguments
- **cross**: An object of class `cross`. See `read.cross` for details.
- **chr**: Vector indicating the chromosomes to flip. This should be a vector of character strings referring to chromosomes by name. A logical (TRUE/FALSE) vector may also be used.

### Details
If the cross contains results from `calc.genoprob`, `sim.geno`, `argmax.geno`, or `calc.errorlod`, those results are also updated.

Results of `est.rf` and `markerlrt` are deleted.

### Value
The input `cross` object, but with the marker order on the specified chromosomes flipped.

### Author(s)
Karl W Broman, <brroman@wisc.edu>
formLinkageGroups

See Also

switch.order

Examples

data(fake.f2)
fake.f2 <- flip.order(fake.f2, c(1, 5, 13))

Description

Use pairwise linkage information between markers (as calculated by est.rf) to partition markers into linkage groups.

Usage

formLinkageGroups(cross, max.rf=0.25, min.lod=3, reorgMarkers=FALSE, verbose=FALSE)

Arguments

cross An object of class cross. See read.cross for details.
max.rf Maximum recombination fraction for placing two markers in the same linkage group (see Details).
min.lod Minimum LOD score for placing two markers in the same linkage group (see Details).
reorgMarkers If TRUE, the output is a cross object, like the input, but with the markers organized into the inferred linkage groups. If FALSE, the output is a table indicating the initial chromosome assignments and the inferred linkage group partitions.
verbose If TRUE, display information about the progress of the calculations.

Details

Two markers are placed in the same linkage group if the estimated recombination fraction between them is \( \leq \text{max.rf} \) and the LOD score (for the test of the rec. frac. = 1/2) is \( \geq \text{min.lod} \). The transitive property (if A is linked to B and B is linked to C then A is linked to C) is used to close the groups.

Value

If reorgMarkers=FALSE (the default), the output is a data frame with rows corresponding to the markers and with two columns: the initial chromosome assignment and the inferred linkage group. Linkage groups are ordered by the number of markers they contain (from largest to smallest).

If reorgMarkers=TRUE, the output is a cross object, like the input, but with the markers reorganized into the inferred linkage groups. The marker order and marker positions within the linkage groups are arbitrary.
formMarkerCovar

Create matrix of marker covariates for QTL analysis

Description

Pull out a matrix of genotypes or genotype probabilities to use markers as covariates in QTL analysis.

Usage

formMarkerCovar(cross, markers, method=c("prob", "imp", "argmax"), ...)

Arguments

cross An object of class cross. See read.cross for details.
markers A vector of character strings of marker or pseudomarker names. Pseudomarker names may be of the form "5@21.5" (for chr 5 at 21.5 cM), but then all names must be of this form.
method If method="prob", the genotype probabilities from calc.genoprob are used; otherwise we use fill.geno to impute missing data, with this method.
... Passed to fill.geno, if necessary.

Value

A matrix containing genotype probabilities or genotype indicators, suitable for use as covariates in scanone.

Author(s)

Karl W Broman, <broman@wisc.edu>
See Also

pull.geno, pull.genoprob, fill.geno, scanone

Examples

data(hyper)
hyper <- calc.genoprob(hyper, step=0)
peakMarker <- "D4Mit164"
X <- formMarkerCovar(hyper, peakMarker)

out <- scanone(hyper, addcovar=X)

---

geno.crosstab Create table of two-locus genotypes

Description

Create a cross tabulation of the genotypes at a pair of markers.

Usage

geno.crosstab(cross, mname1, mname2, eliminate.zeros=TRUE)

Arguments

cross An object of class cross. See read.cross for details.
mname1 The name of the first marker (as a character string). (Alternatively, a vector with the two character strings, in which case mname2 should not be given.)
mname2 The name of the second marker (as a character string).
eliminate.zeros If TRUE, don’t show the rows and columns that have no data.

Value

A matrix containing the number of individuals having each possible pair of genotypes. Genotypes for the first marker are in the rows; genotypes for the second marker are in the columns.

Author(s)

Karl W Broman, <broman@wisc.edu>

See Also

geno.table, find.marker
Examples

```r
data(hyper)
geno.crosstab(hyper, "D1Mit123", "D1Mit156")
geno.crosstab(hyper, "DXMit22", "DXMit16")
geno.crosstab(hyper, c("DXMit22", "DXMit16"))
```

---

**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, col=NULL, ...)
```

**Arguments**

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

- `chr`  
  Optional vector indicating the chromosomes to be drawn in the plot. This should be a vector of character strings referring to chromosomes by name; numeric values are converted to strings. Refer to chromosomes with a preceding `-` to have all chromosomes but those considered. A logical (TRUE/FALSE) vector may also be used.

- `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.

- `col`  
  Vector of colors. The first is for missing genotypes, followed by colors for each of the genotypes. If NULL, a default set of colors are used.

- `...`  
  Passed to `image`.

**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.
geno.table

Value

None.

Author(s)

Karl W Broman, <broman@wisc.edu>

See Also

plot.cross, plotMissing, plotGeno, image

Examples

data(listeria)
geno.image(listeria)

---

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, scanone.output=FALSE)

Arguments

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

chr Optional vector indicating the chromosomes to consider. This should be a vector of character strings referring to chromosomes by name; numeric values are converted to strings. Refer to chromosomes with a preceding - to have all chromosomes but those considered. A logical (TRUE/FALSE) vector may also be used.

scanone.output If TRUE, give result in the form output by scanone, so that one may use plot.scanone, etc.

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.
getid

Value

If `scanone.output=FALSE`, the output is 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.

If `scanone.output=TRUE`, the output is of the form produced by `scanone`, with the first two columns being chromosome IDs and cM positions of the markers. The third column is $-\log_{10}(P)$ from chi-square tests of Mendelian segregation. The fourth column is the proportion of missing data. The remaining columns are the proportions of the different genotypes (among typed individuals).

Author(s)

Karl W Broman, <broman@wisc.edu>

See Also

`summary.cross`, `drop.markers`, `drop.nullmarkers`

Examples

```r
data(listeria)
geno.table(listeria)

geno.table(listeria, chr=13)

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

out <- geno.table(listeria, scanone.output=TRUE)
plot(out)
plot(out, lod=2)
```

---

**getid**

Pull out the individual identifiers from a cross

Description

Pull out the individual identifiers from a cross object.

Usage

```r
getid(cross)
```

Arguments

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

A vector of individual identifiers, pulled from the phenotype data (a column named id or ID). If there are no such identifiers in the cross, the function returns NULL.

**Author(s)**

Karl W Broman, <broman@wisc.edu>

**See Also**

subset.cross, top.errorlod

**Examples**

```r
data(fake.f2)
# create an ID column
fake.f2$pheno$id <- paste("ind", sample(nind(fake.f2)), sep="")
getid(fake.f2)
```

---

**groupclusteredheatmap**  *Retrieving groups of traits after clustering*

**Description**

Retrieving groups of clustered traits from the output of mqmplot.clusteredheatmap.

**Usage**

```
groupclusteredheatmap(cross, clusteredheatmapresult, height)
```

**Arguments**

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

- `clusteredheatmapresult`  
  Resulting dendrogram object from `mqmplot.clusteredheatmap`

- `height`  
  Height at which to 'cut' the dendrogram, a higher cut-off gives less but larger groups. Height represents the maximum distance between two traits clustered together using hclust. The 'normal' behaviour of bigger groups when using a higher height cut-off depends on the tree structure and the amount of traits clustered using `mqmplot.clusteredheatmap`

**Value**

A list containing groups of traits which were clustered together with a distance less than `height`
Author(s)

Danny Arends <danny.arends@gmail.com>

See Also

- The MQM tutorial: https://rqtl.org/tutorials/MQM-tour.pdf
- MQM - MQM description and references
- mqmscan - Main MQM single trait analysis
- mqmscanall - Parallelized traits analysis
- mqaugment - Augmentation routine for estimating missing data
- mqmautocofactors - Set cofactors using marker density
- mqmsetcofactors - Set cofactors at fixed locations
- mqpperm - Estimate significance levels
- scanone - Single QTL scanning

Examples

data(multitrait)

multitrait <- fill.geno(multitrait) # impute missing genotype data
result <- mqmscanall(multitrait, logtransform=TRUE)
cresults <- mqmplot.clusteredheatmap(multitrait,result)
groupclusteredheatmap(multitrait,cresults,10)

---

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.
inferFounderHap

Crude reconstruction of founder haplotypes in multi-parent RIL

Description

Uses groups of adjacent markers to infer the founder haplotypes in SNP data on multi-parent recombinant inbred lines.

Usage

inferFounderHap(cross, chr, max.n.markers=15)

Arguments

cross An object of class cross. See read.cross for details.
chr Indicator of chromosome to consider. If multiple chromosomes are selected, only the first is used.
max.n.markers Maximum number of adjacent markers to consider.
We omit SNPs for which any of the founders are missing.

We then consider groups of adjacent SNPs, looking for founder haplotypes that are unique; RIL sharing such a unique haplotype are then inferred to have that founder’s DNA.

We consider each marker as the center of a haplotype, and consider haplotypes of size 1, 3, 5, ..., max.n.markers. We end the extension of the haplotypes when all founders have a unique haplotype.

A matrix of dimension nind(cross) × no. markers, with the inferred founder origin for each line at each marker.

Karl W Broman, <broman@wisc.edu>

**See Also**

*sim.geno, calc.genoprob, fill.geno, argmax.geno*

**Examples**

```r
map <- sim.map(100, n.mar=101, include.x=FALSE, eq.spacing=TRUE)
founderGeno <- simFounderSnps(map, "8")
ril <- sim.cross(map, n.ind=10, type="ri8sib", founderGeno=founderGeno)

h <- inferFounderHap(ril, max.n.markers=11)
mean(!is.na(h)) # proportion inferred
plot(map[[1]], h[[1]], ylim=c(0.5, 8.5), xlab="Position", ylab="Genotype")
```

**Description**

Identify inferred partitions in mapping QTL to a phylogenetic tree.

**Usage**

```r
inferredpartitions(output, chr, lodthreshold, probthreshold=0.9)
```
Arguments

output  An object output by the function `scanPhyloQTL`.

chr  A character string indicating the chromosome to consider. (It can also be a number, but it’s then converted to a character string.)

lodthreshold  LOD threshold; if maximum LOD score is less than this, the null model is considered.

probthreshold  Threshold on posterior probabilities. See Details below.

Details

We consider a single chromosome, and take the maximum LOD score for each partition on that chromosome. The presence of a QTL is inferred if at least one partition has LOD score greater than `lodthreshold`. In this case, we then convert the LOD scores for the partitions to approximate posterior probabilities by taking $10^{LOD}$ and then rescaling them to sum to 1. These are sorted from largest to smallest, and we then take as the inferred partitions the smallest set whose posterior probabilities cumulatively add up to at least `probthreshold`.

Value

A vector of character strings. If the null model (no QTL) is inferred, the output is "null". Otherwise, it is the set of inferred partitions.

Author(s)

Karl W Broman, <broman@wisc.edu>

References


See Also

`scanPhyloQTL`, `plot.scanPhyloQTL`, `summary.scanPhyloQTL`, `max.scanPhyloQTL`, `simPhyloQTL`

Examples

```r
# example map; drop X chromosome
data(map10)
map10 <- map10[1:19]

# simulate data
x <- simPhyloQTL(4, partition="AB|CD", crosses=c("AB", "AC", "AD"),
                     map=map10, n.ind=150,
                     model=c(1, 50, 0.5, 0))

# run calc.genoprob on each cross
```
## interpPositions

Interpolate positions from one map to another

### Description

On the basis of a pair of marker maps with common markers, take positions along one map and interpolate (or, past the terminal markers on a chromosome, extrapolate) their positions on the second map.

### Usage

```r
interpPositions(oldpositions, oldmap, newmap)
```

### Arguments

- `oldpositions`: A data frame with two columns: `chr` (chromosome identifiers) and `pos` (positions, along `oldmap`).
- `oldmap`: An object of class "map"; see `sim.map` for details.
- `newmap`: An object of class "map", with the same chromosomes and markers as `oldmap`.

### Details

In this explanation, take `oldmap` and `newmap` to be the physical and genetic maps, respectively. We use linear interpolation within each interval, assuming a constant recombination rate within the interval. Past the terminal markers, we use linear extrapolation, using the chromosome-wide average recombination rate.

### Value

The input data frame, `oldpositions`, with an additional column `newpos` with the interpolated positions along `newmap`.

### Author(s)

Karl W Broman, <brman@wisc.edu>
jittermap

See Also

shiftmap, rescalemap, pull.map

Examples

data(hyper)

# hyper genetic map
gmap <- pull.map(hyper)

# a fake physical map, with each chromosome starting at 0.
pmap <- shiftmap(rescalemap(gmap, 2))

# positions on pmap to determine location on gmap
tofind <- data.frame(chr=c(1, 5, 17, "X"), pos=c(220, 20, 105, 10))
rownames(tofind) <- paste("loc", 1:nrow(tofind), sep="")

interpPositions(tofind, pmap, gmap)

---

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, <broman@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)

References


See Also

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

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"))

locateXO  Estimate locations of crossovers

Description

Estimate the locations of crossovers for each individual on a given chromosome.

Usage

locateXO(cross, chr, full.info=FALSE)

Arguments

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

chr    Chromosome to investigate (if unspecified, the first chromosome is considered). This should be a character string referring to a chromosome by name; numeric values are converted to strings.

full.info If TRUE, output will include information on the left and right endpoints of the intervals to which recombination events are known, as well as the corresponding marker indices.
Details

For each individual we determine the locations of obligate crossovers, and estimate their location to be at the midpoint between the nearest flanking typed markers.

The function currently only works for a backcross, intercross, or recombinant inbred line.

Value

A list with one component per individual. Each component is either NULL or is a numeric vector with the estimated crossover locations.

If full.info=TRUE, in place of a numeric vector with estimated locations, there is a matrix that includes those locations, the left and right endpoints of the intervals to which crossovers can be placed, the marker indices corresponding to those endpoint, and genotype codes for the genotypes to the left and right of each crossover. The final column indicates the number of typed markers between the current crossover and the next one (useful for identifying potential genotyping errors).

Author(s)

Karl W Broman, <broman@wisc.edu>

See Also

countXO, cleanGeno

Examples

data(hyper)
xoloc <- locateXO(hyper, chr=4)
table(sapply(xoloc, length))

locations

A table with genetic locations of the traits in the multitrait dataset

Usage

data(locations)

Format

Each row is a trait with the following information: Name, Name of the trait (will be checked against the name in the cross object Chr, Chromosome of the trait cM, Location in cM from the start of the chromosome


Source
Additional information from the Arabidopsis RIL selfing experiment with Landsberg erecta (Ler) and Cape Verde Islands (Cvi) with 162 individuals scored (with errors at) 117 markers. Dataset obtained from GBIC - Groningen BioInformatics Centre

References


See Also

multitrait

Examples

```r
## Not run:
data(multitrait)
data(locations)
multiloc <- addloctocross(multitrait, locations)
results <- scanall(multiloc)
mqmplot.cistrans(results, multiloc, 5, FALSE, TRUE)
## End(Not run)
```

lodint  

**LOD support interval**

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

Usage

```r
lodint(results, chr, qtl.index, drop=1.5, lodcolumn=1, expandtomarkers=FALSE)
```

Arguments

- **results**: Output from `scanone`, or a `qtl` object as output from `refineqtl`.
- **chr**: A chromosome ID (if input results are from `scanone` (should have length 1).
- **qtl.index**: Numeric index for a QTL (if input results are from `refineqtl` (should have length 1).
- **drop**: LOD units to drop to form the interval.
lodcolumn An integer indicating which of the LOD score columns should be considered (if input results are from scanone).

expandtomarkers If TRUE, the interval is expanded to the nearest flanking markers.

Value
An object of class scanone indicating the estimated QTL position and the approximate endpoints for the LOD support interval.

Author(s)
Karl W Broman, <broman@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)
lodint(out, chr=4, expandtomarkers=TRUE)
makeqtl

Arguments

- **cross**: An object of class `cross`. See `read.cross` for details.
- **chr**: Vector indicating the chromosome for each QTL. (These should be character strings referring to the chromosomes by name.)
- **pos**: Vector (of same length as `chr`) indicating the positions on the chromosome to be taken. If there is no marker or pseudomarker at a position, the nearest position is used.
- **qtl.name**: Optional 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"):

- **geno**: Imputed genotypes.
- **prob**: Genotype probabilities.
- **name**: User-defined name for each QTL, or a name of the form "Chr1@10".
- **altname**: QTL names of the form "Q1", "Q2", etc.
- **chr**: Input vector of chromosome numbers.
- **pos**: Input vector of chromosome positions.
- **n.qtl**: Number of QTLs.
- **n.ind**: Number of individuals.
- **n.gen**: A vector indicating the number of genotypes for each QTL.

Author(s)

Hao Wu; Karl W Broman, <broman@wisc.edu>

See Also

`fitqtl, calc.genoprob, sim.geno, dropfromqtl, replaceqtl, addtoqtl, summary.qtl, reorderqtl`
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")
summary(qtl)

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 approx-
imates 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, plotMap, pull.map

Examples

data(map10)
plot(map10)

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

Convert a map object (as a list) to a table (as a data frame).

Usage

map2table(map, chr)

Arguments

map  A map object: a list whose components (corresponding to chromosomes) are either vectors of marker positions or matrices with two rows of sex-specific marker positions.

chr  Optional vector indicating the chromosomes to consider. This should be a vector of character strings referring to chromosomes by name; numeric values are converted to strings. Refer to chromosomes with a preceding - to have all chromosomes but those considered. A logical (TRUE/FALSE) vector may also be used.

Value

A data frame with two or three columns: chromosome and sex-averaged position, or chromosome, female position, and male position.

The row names are the marker names.

Author(s)

Karl W Broman, <broman@wisc.edu>

See Also

table2map, pull.map, est.map

Examples

data(fake.f2)
map <- pull.map(fake.f2)
map_as_tab <- map2table(map)
Simulated data for illustrating genetic map construction

Description

Simulated data for an F2 intercross, obtained using sim.cross, useful for illustrating the process of constructing a genetic map.

Usage

data(mapthis)

Format

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

Details

These are simulated data, consisting of 300 F2 individuals typed at 100 markers on five chromosomes. There are no real phenotypes, just a set of individual identifiers. The data were simulated for the purpose of illustrating the process of constructing a genetic map. The markers are all assigned to a single chromosome and in a random order, and there are a number of problematic markers and individuals.

See https://rqtl.org/tutorials/geneticmaps.pdf for a tutorial on how to construct a genetic map with these data.

Author(s)

Karl W Broman, <broman@wisc.edu>

References


See Also

fake.f2, est.rf, est.map, formLinkageGroups, orderMarkers

Examples

data(mapthis)
summary(mapthis)
plot(mapthis)
markerlrt

*General likelihood ratio test for association between marker pairs*

**Description**

Calculate a LOD score for a general likelihood ratio test for each pair of markers, to assess their association.

**Usage**

`markerlrt(cross)`

**Arguments**

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

**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 upper and lower triangles each contain the LOD scores.

**Author(s)**

Karl W Broman, <broman@wisc.edu>

**See Also**

- `plotRF`
- `est.rf`
- `badorder`

**Examples**

```r
data(badorder)
badorder <- markerlrt(badorder)
plotRF(badorder)
```

markernames

*Pull out the marker names from a cross*

**Description**

Pull out the marker names from a cross object as one big vector.

**Usage**

`markernames(cross, chr)`
Arguments

- **cross**: An object of class `cross`. See `read.cross` for details.
- **chr**: Optional vector indicating the chromosomes to consider. This should be a vector of character strings referring to chromosomes by name; numeric values are converted to strings. Refer to chromosomes with a preceding `-` to have all chromosomes but those considered. A logical (TRUE/FALSE) vector may also be used.

Value

A vector of character strings (the marker names).

Author(s)

Karl W Broman, <broman@wisc.edu>

See Also

- `pull.map`, `phenames`, `chrnames`

Examples

```r
data(listeria)
markernames(listeria, chr=5)
```

---

# max.scanone

## Maximum peak in genome scan

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

Usage

```r
## S3 method for class 'scanone'
max(object, chr, lodcolumn=1, 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**: Optional vector indicating the chromosomes to consider. This should be a vector of character strings referring to chromosomes by name; numeric values are converted to strings. Refer to chromosomes with a preceding `-` to have all chromosomes but those considered. A logical (TRUE/FALSE) vector may also be used.
max.scanPhyloQTL

lodcolumn  An integer, indicating which of the LOD score columns should be considered in pulling out the peak (these are indexed 1, 2, …).
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, <broman@wisc.edu>

See Also

scanone, plot.scanone, summary.scanone

Examples

data(listeria)

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

# Maximum peak for LOD(p,mu) on chr 5
max(out, chr=5)

# Maximum peak for LOD(p,mu) on chromosomes other than chr 13
max(out, chr="-13")

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

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

max.scanPhyloQTL  Maximum peak in genome scan to map a QTL to a phylogenetic tree

Description

Print the chromosome with the maximum LOD score across partitions, from the results of scanPhyloQTL.
Usage

## S3 method for class 'scanPhyloQTL'
max(object, chr, format=c("postprob", "lod"),
    ...)  

Arguments

  object  An object output by the function scanPhyloQTL.
  chr     Optional vector indicating the chromosomes to consider. This should be a vec-
          tor of character strings referring to chromosomes by name; numeric values are
          converted to strings. Refer to chromosomes with a preceding - to have all chro-
          mosomes but those considered. A logical (TRUE/FALSE) vector may also be
          used.
  format  Indicates whether to provide LOD scores or approximate posterior probabilities;
          see the help file for summary.scanPhyloQTL.
  ...     Ignored at this point.

Details

  The output, and the use of the argument format, is as in summary.scanPhyloQTL.

Value

  An object of class summary.scanPhyloQTL, to be printed by print.summary.scanPhyloQTL.

Author(s)

  Karl W Broman, <broman@wisc.edu>

References

  Broman, K. W., Kim, S., Ané, C. and Payseur, B. A. Mapping quantitative trait loci to a phyloge-

See Also

  scanPhyloQTL, plot.scanPhyloQTL, summary.scanPhyloQTL, max.scanone, inferredpartitions,
simPhyloQTL

Examples

## Not run:
# example map; drop X chromosome
data(map10)
map10 <- map10[1:19]

# simulate data
x <- simPhyloQTL(4, partition="AB|CD", crosses=c("AB", "AC", "AD"),
    map=map10, n.ind=150,
# run calc.genoprob on each cross
x <- lapply(x, calc.genoprob, step=2)

# scan genome, at each position trying all possible partitions
out <- scanPhyloQTL(x, method="hk")

# maximum peak
max(out, format="lod")

# approximate posterior probabilities at peak
max(out, format="postprob")

# all peaks above a threshold for LOD(best) - LOD(2nd best)
summary(out, threshold=1, format="lod")

# all peaks above a threshold for LOD(best), showing approx post'r prob
summary(out, format="postprob", threshold=3)

# plot of results
plot(out)

## End(Not run)

---

**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

```r
## S3 method for class 'scantwo'
max(object, lodcolumn=1,
    what=c("best", "full", "add", "int"),
    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.
- **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.)

Output of addpair

Note that, for output from `addpair` in which the new loci are indicated explicitly in the formula, the summary provided by `max.scantwo` is somewhat special.

All arguments (except, of course, the input object) are ignored.

If the formula is symmetric in the two new QTL, the output has just two LOD score columns: `lod.2v0` comparing the full model to the model with neither of the new QTL, and `lod.2v1` comparing the full model to the model with just one new QTL.

If the formula is not symmetric in the two new QTL, the output has three LOD score columns: `lod.2v0` comparing the full model to the model with neither of the new QTL, `lod.2v1b` comparing the full model to the model in which the first of the new QTL is omitted, and `lod.2v1a` comparing the full model to the model with the second of the new QTL omitted.

Author(s)

Karl W Broman, <broman@wisc.edu>

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, <broman@wisc.edu>

See Also

switch.order

Examples

data(badorder)
badorder <- movemarker(badorder, "D2M937", 3, 48.15)
badorder <- movemarker(badorder, "D3M160", 2, 28.83)
**Introduction to Multiple QTL Model (MQM) mapping**

**Description**

Overview of the MQM mapping functions

**Introduction**

Multiple QTL Mapping (MQM) provides a sensitive approach for mapping quantitative trait loci (QTL) in experimental populations. MQM adds higher statistical power compared to many other methods. The theoretical framework of MQM was introduced and explored by Ritsert Jansen, explained in the ‘Handbook of Statistical Genetics’ (see references), and used effectively in practical research, with the commercial ‘mapqtl’ software package. Here we present the first free and open source implementation of MQM, with extra features like high performance parallelization on multi-CPU computers, new plots and significance testing.

MQM is an automatic three-stage procedure in which, in the first stage, missing data is ‘augmented’. In other words, rather than guessing one likely genotype, multiple genotypes are modeled with their estimated probabilities. In the second stage important markers are selected by multiple regression and backward elimination. In the third stage a QTL is moved along the chromosomes using these pre-selected markers as cofactors, except for the markers in the window around the interval under study. QTL are (interval) mapped using the most ‘informative’ model through maximum likelihood. A refined and automated procedure for cases with large numbers of marker cofactors is included. The method internally controls false discovery rates (FDR) and lets users test different QTL models by elimination of non-significant cofactors.

R/qtl-MQM has the following advantages:

- Higher power to detect linked as well as unlinked QTL, as long as the QTL explain a reasonable amount of variation
- Protection against overfitting, because it fixes the residual variance from the full model. For this reason more parameters (cofactors) can be used compared to, for example, CIM
- Prevention of ghost QTL (between two QTL in coupling phase)
- Detection of negating QTL (QTL in repulsion phase)

**Note**

The current implementation of R/qtl-MQM has the following limitations: (1) MQM is limited to experimental crosses F2, BC, and selfed RIL, (2) MQM does not treat sex chromosomes differently from autosomal chromosomes - though one can introduce sex as a cofactor. Future versions of R/qtl-MQM may improve on these points. Check the website and change log (https://github.com/kbroman/qtl/blob/master/NEWS.md) for updates.

**Author(s)**

Ritsert C Jansen; Danny Arends; Pjotr Prins; Karl W Broman <broman@wisc.edu>
MQM

References


See Also

- The MQM tutorial: [https://rqtl.org/tutorials/MQM-tour.pdf](https://rqtl.org/tutorials/MQM-tour.pdf)
- MQM - MQM description and references
- mqmscan - Main MQM single trait analysis
- mqmscanall - Parallelized traits analysis
- mqmaugment - Augmentation routine for estimating missing data
- mqmautocofactors - Set cofactors using marker density
- mqmsetcofactors - Set cofactors at fixed locations
- mqmpermutation - Estimate significance levels
- scanone - Single QTL scanning

Examples

```r
data(map10)  # Genetic map modeled after mouse

# simulate a cross (autosomes 1–10)
qtl <- c(3,15,1,0)  # QTL model: chr, pos'n, add've & dom effects
cross <- sim.cross(map10[1:10],qtl,n=100,missing.prob=0.01)

# MQM
crossaug <- mqmaugment(cross)  # Augmentation
cat(crossaug$qm$Nind,'real individuals retained in dataset','
crossaug$qm$Naug,'individuals augmented\n')

result <- mqmscan(crossaug)  # Scan

# show LOD interval of the QTL on chr 3
lodint(result,chr=3)
```
mqmaugment

MQM augmentation

Description

Fill in missing genotypes for MQM mapping. For each missing or incomplete marker it fills in (or ‘augments’) all possible genotypes, thus creating new candidate ‘individuals’. The probability of each individual is calculated using information on neighbouring markers and recombination frequencies. When a genotype of an augmented genotype is less likely than the \( \text{minprob} \) parameter it is dropped from the dataset. The \textit{augmented} list of individuals is returned in a new cross object. For a full discussion on augmentation see the MQM tutorial online.

Usage

\[
mqmaugment(cross, \text{maxaugind}=82, \text{minprob}=0.1, \\
\text{strategy}=c("default","impute","drop"), \\
\text{verbose}=\text{FALSE})
\]

Arguments

- **cross**: An object of class \texttt{cross}. See \texttt{read.cross} for details.
- **maxaugind**: Maximum number of augmentations per individual. The default of 82 allows for six missing markers for an individual in a BC cross (\(2^6 = 64\)) and four missing markers in an F2 (\(3^4 = 81\)). When a large number of markers are missing this default number is quickly reached.
- **minprob**: Return individuals with augmented genotypes that have at least this probability of occurring. \texttt{minprob} is a value between 0 and 1. For example a value of 0.5 will drop all genotypes that are half as likely as the most likely genotype (candidate of the individual). The default value of 0.1 will drop all genotypes that are less likely of occurring than 1 in 10, compared against the most likely genotype. Use a value of 1.0 to return a single filled in genotype for each individual.
- **strategy**: When individuals have too much missing data and augmentation fails three options are provided: 1. "default": Calculate genotypes at missing marker positions, accounting for \texttt{minprob}, and add this individual to the set. 2. "impute": Calculate the most likely genotypes at missing marker positions and impute \texttt{maxaugind} individual-variants around the most likely genotype. 3. "drop": Drop individuals that cannot be augmented from the dataset, this option is not advised because information from the dropped individuals will be lost.
- **verbose**: If TRUE, give verbose output

Value

Returns the \texttt{cross} object with augmented individuals (many individuals from the data set will be repeated multiple times). Some individuals may have been dropped completely when the probability falls below \texttt{minprob}. An added component to the \texttt{cross} object named \texttt{mqm} contains information on exactly which individuals are retained and repeated.
mqmaugment

Note

The sex chromosome 'X' is treated like autosomes during augmentation. With an F2 the sex chromosome is not considered. This will change in a future version of MQM. Run with verbose=TRUE to verify how many individuals are augmented versus moved to the second augmentation round. This could have an effect on the resulting dataset or check the return cross$mqm values. Compare results by using minprob=1.

Author(s)

Ritsert C Jansen; Danny Arends; Pjotr Prins; Karl W Broman <broman@wisc.edu>

See Also

• fill.geno - Alternative routine for estimating missing data
• The MQM tutorial: https://rqtl.org/tutorials/MQM-tour.pdf
• MQM - MQM description and references
• mqmscan - Main MQM single trait analysis
• mqmscanall - Parallellized traits analysis
• mqmaugment - Augmentation routine for estimating missing data
• mqmautocofactors - Set cofactors using marker density
• mqmsetcofactors - Set cofactors at fixed locations
• mqmpermutation - Estimate significance levels
• scanone - Single QTL scanning

Examples

data(map10) # Genetic map modeled after mouse

# simulate a cross (autosomes 1-10)
qtl <- c(3,15,1,0) # QTL model: chr, pos'n, add've & dom effects
cross <- sim.cross(map10[1:10],qtl,n=100,missing.prob=0.01)

# MQM
crossaug <- mqmaugment(cross) # Augmentation
cat(crossaug$mqm$Nind,'real individuals retained in dataset',
   crossaug$mqm$Naug,'individuals augmented\n')

result <- mqmscan(crossaug) # Scan

# show LOD interval of the QTL on chr 3
lodint(result,chr=3)
Automatic setting of cofactors, taking marker density into account

Description

Sets cofactors, taking underlying marker density into account. Together with `mqmscan` cofactors are selected through backward elimination.

Usage

```r
mqmautocofactors(cross, num=50, distance=5, dominance=FALSE, plot=FALSE, verbose=FALSE)
```

Arguments

cross: An object of class `cross`. See `read.cross` for details.
num: Number of cofactors to set (warns when setting too many cofactors).
distance: Minimal distance between two cofactors, in cM.
dominance: If TRUE, create a cofactor list that is safe to use with the dominance scan mode of MQM. See `mqmscan` for details.
plot: If TRUE, plots a genetic map displaying the selected markers as cofactors.
verbose: If TRUE, give verbose output.

Value

A list of cofactors to be used with `mqmscan`.

Author(s)

Ritsert C Jansen; Danny Arends; Pjotr Prins; Karl W Broman <broman@wisc.edu>

See Also

- The MQM tutorial: [https://rqtl.org/tutorials/MQM-tour.pdf](https://rqtl.org/tutorials/MQM-tour.pdf)
- `MQM` - MQM description and references
- `mqmscan` - Main MQM single trait analysis
- `mqmscanall` - Parallelized traits analysis
- `mqmaugment` - Augmentation routine for estimating missing data
- `mqmautocofactors` - Set cofactors using marker density
- `mqmsetcofactors` - Set cofactors at fixed locations
- `mqmpermutation` - Estimate significance levels
- `scanone` - Single QTL scanning
Examples

```r
data(hyper)  # hyper dataset
hyperfilled <- fill.geno(hyper)
cofactors <- mqmautocofactors(hyperfilled,15)  # Set 15 Cofactors
result <- mqmscan(hyperfilled,cofactors)  # Backward model selection
mqmgetmodel(result)
```

Description

Extract the real markers from a cross object that includes pseudo markers

Usage

```r
mqmextractmarkers(mqmresult)
```

Arguments

- `mqmresult`: result from `mqmscan`, including pseudo markers

Value

Returns a `scanone` object with the pseudo markers removed

Author(s)

Ritsert C Jansen; Danny Arends; Pjotr Prins; Karl W Broman <broman@wisc.edu>

See Also

- The MQM tutorial: [https://rqtl.org/tutorials/MQM-tour.pdf](https://rqtl.org/tutorials/MQM-tour.pdf)
- MQM - MQM description and references
- `mqmscan` - Main MQM single trait analysis
- `mqmscanall` - Parallelized traits analysis
- `mqmaugment` - Augmentation routine for estimating missing data
- `mqmautocofactors` - Set cofactors using marker density
- `mqmsetcofactors` - Set cofactors at fixed locations
- `mqmpermutation` - Estimate significance levels
- `scanone` - Single QTL scanning
Examples

data(multitrait)
multitrait <- fill.geno(multitrait)
result <- mqmscan(multitrait)
newresult <- mqmextractmarkers(result)

mqmfind.marker

Fetch significant markers after permutation analysis

Description

Fetch significant makers after permutation analysis. These markers can be used as cofactors for
model selection in a forward stepwise approach.

Usage

mqmfind.marker(cross, mqmscan = NULL, perm = NULL, alpha = 0.05, verbose=FALSE)

Arguments

cross An object of class cross. See read.cross for details.
mqmscan Results from either scanone or mqmscan
perm a scanoneperm object
alpha Threshold value, everything with significance < alpha is reported
verbose Display more output on verbose=TRUE

Value

returns a matrix with at each row a significant marker (determined from the scanoneperm object)
and with columns: markername, chr and pos (cM)

Author(s)

Ritsert C Jansen; Danny Arends; Pjotr Prins; Karl W Broman <broman@wisc.edu>

See Also

- mqmprocesspermutation - Function called to convert results from an mqmprecision into
  an scanoneperm object
- The MQM tutorial: https://rqtl.org/tutorials/MQM-tour.pdf
- MQM - MQM description and references
- mqmscan - Main MQM single trait analysis
- mqmscanall - Parallellized traits analysis
mqmgetmodel

• **mqnaugment** - Augmentation routine for estimating missing data
• **mqmautocofactors** - Set cofactors using marker density
• **mqmsetcofactors** - Set cofactors at fixed locations
• **mqmpermutation** - Estimate significance levels
• **scanone** - Single QTL scanning

**Examples**

```r
# Use the multitrait dataset
data(multitrait)

# Set cofactors at each 3th marker
cof <- mqmsetcofactors(multitrait, 3)

# impute missing genotypes
multitrait <- fill.geno(multitrait)

# log transform the 7th phenotype
multitrait <- transformPheno(multitrait, 7)

# Bootstrap 50 runs in batches of 10
## Not run: result <- mqmpermutation(multitrait, scanfunction=mqmscan, cofactors=cof, pheno.col=7, n.perm=50, batchsize=10)
## End(Not run)

# Create a permutation object
f2perm <- mqmprocesspermutation(result)

# What LOD score is considered significant?
summary(f2perm)

# Find markers with a significant QTL effect (First run is original phenotype data)
marker <- mqmfind.marker(multitrait, result[[1]], f2perm)

# Print it to the screen
marker
```

---

**mqmgetmodel**

Retrieve the QTL model used in mapping from the results of an MQM scan

**Description**

Retrieves the QTL model used for scanning from the output of an MQM scan. The model only contains the selected cofactors significant at the specified cofactor.significance from the results of an mqm scan.
Usage

mqmgetmodel(scanresult)

Arguments

scanresult An object returned by mqmscan, including cofactors and QTL model.

Value

The function returns the multiple QTL model created, which consists of the cofactors selected during the modeling phase of the algorithm. This model was used when scanning for additional QTL in the mqmscan function. The format of the model is compatible with the makeqtl function. For more information about the format of the model see the makeqtl page. When no cofactor was selected in the modeling phase no model was created, then this function will return a NULL value.

Author(s)

Ritsert C Jansen; Danny Arends; Pjotr Prins; Karl W Broman &lt;broman@wisc.edu&gt;

See Also

- mqmsetcofactors - Setting multiple cofactors for backward elimination
- makeqtl - Make a qtl object
- The MQM tutorial: https://rqtl.org/tutorials/MQM-tour.pdf
- MQM - MQM description and references
- mqmscan - Main MQM single trait analysis
- mqmscanall - Parallellized traits analysis
- mqmaugment - Augmentation routine for estimating missing data
- mqmautocofactors - Set cofactors using marker density
- mqmsetcofactors - Set cofactors at fixed locations
- mqmpermutation - Estimate significance levels
- scanone - Single QTL scanning

Examples

data(hyper)

hyperfilled &lt;- fill.geno(hyper)
cofactors &lt;- mqmsetcofactors(hyperfilled,4)
result &lt;- mqmscan(hyperfilled,cofactors)
mqmgetmodel(result)
plot(mqmgetmodel(result))
mqmpermutation

Estimate QTL LOD score significance using permutations or simulations

Description

Two randomization approaches to obtain estimates of QTL significance:

- Random redistribution of traits (method='permutation')
- Random redistribution of simulated trait values (method='simulation')

Calculations can be parallelized using the SNOW package.

Usage

mqmpermutation(cross, scanfunction=scanone, pheno.col=1, multicore=TRUE,
              n.perm=10, batchsize=10, file="MQM_output.txt",
              n.cluster=1, method=c("permutation","simulation"),
              cofactors=NULL, plot=FALSE, verbose=FALSE, ...)

Arguments

cross An object of class cross. See read.cross for details.
scanfunction Function to use when mapping QTL's (either scanone, cim or mqm)
pheno.col Column number in the phenotype matrix which should be used as the phenotype.
            This can be a vector of integers.
multicore Use multicore (if available)
n.perm Number of permutations to perform (DEFAULT=10, should be 1000, or higher,
            for publications)
batchsize Batch size. The entire set is split in jobs. Each job contains b.size number of
            traits per job
file Name of the intermediate output file used
n.cluster Number of child processes to split the job into
method What kind permutation should occur: permutation or simulation
cofactors cofactors, only used when scanfunction is mqm. List of cofactors to be analysed
            in the QTL model. To set cofactors use mqmautocofactors or mqmsetcofactors.
plot If TRUE, make a plot
verbose If TRUE, print tracing information
... Parameters passed through to the scanone, cim or mqmscan functions

Details

Analysis of scanone, cim or mqmscan to scan for QTL in shuffled/randomized data. It is recommended to also install the snow library. The snow library allows calculations to run on multiple cores or even scale it up to an entire cluster, thus speeding up calculation.
Value

Returns a mqmmulti object. This object is a list of scanone objects that can be plotted using `plot.scanone(result[[trait]])`

Author(s)

Ritsert C Jansen; Danny Arends; Pjotr Prins; Karl W Broman <broman@wisc.edu>

References


See Also

- The MQM tutorial: [https://rqtl.org/tutorials/MQM-tour.pdf](https://rqtl.org/tutorials/MQM-tour.pdf)
- MQM - MQM description and references
- mqmscan - Main MQM single trait analysis
- mqmscanall - Parallelized traits analysis
- mqmaugment - Augmentation routine for estimating missing data
- mqmautocofactors - Set cofactors using marker density
- mqmsetcofactors - Set cofactors at fixed locations
- mqmpermutation - Estimate significance levels
- scanone - Single QTL scanning

Examples

```r
# Use the multitrait dataset
data(multitrait)

multitrait <- calc.genoprob(multitrait)
result <- mqmpermutation(multitrait, pheno.col=7, n.perm=2, batchsize=2)

## Not run: #Set 50 cofactors
cof <- mqmautocofactors(multitrait, 50)

## End(Not run)
multitrait <- fill.geno(multitrait)
```
result <- mqmpermutation(multitrait, scanfunction=mqmscan, cofactors=cof, 
    pheno.col=7, n.perm=2, batchsize=2, verbose=FALSE)

# Create a permutation object
f2perm <- mqmprocesspermutation(result)

# Get Significant LOD thresholds
summary(f2perm)

mqmplot.circle

Circular genome plot for MQM

Description
Circular genome plot - shows QTL locations and relations.

Usage
mqmplot.circle(cross, result, highlight=0, spacing=25, interactstrength=2, 
    axis.legend=TRUE, col.legend=FALSE, verbose=FALSE, transparency=FALSE)

Arguments
cross An object of class cross with optionally phenotype locations. See read.cross 
    for details on reading in cross objects, and optionally addloctocross for adding 
    phenotype locations.
result An object of class mqmmulti or scanone. See mqmscanall scanone for details.
highlight With a mqmmulti object, highlight this phenotype (value between one and the 
    number of results in the mqmmultiobject)
interactstrength When highlighting a trait, consider interactions significant they have a change 
    of more than interactstrength*SEs. A higher value will show less interactions. 
    However the interactions reported at higher interactstrength values will generaty 
    be more reliable.
spacing User defined spacing between chromosomes in cM
axis.legend When set to FALSE, suppresses the legends. (defaults to plotting legends be-
    sides the axis.
col.legend With a mqmmulti object, plots a legend for the non-highlighed version 
    transparency Use transparency when drawing the plots (defaults to no transparency)
verbose Be verbose

Details
Depending on the input of the result being either scanone or mqmmulti a different plot is drawn. 
If model information is present from mqmscan (by setting cofactors) This will be highlighted in red 
(see example). If phenotypes have genetic locations (e.g. eQTL) they will be plotted on the genome 
otherwise phenotypes will be plotted in the middle of the circle (with a small offset) Locations can be added by using the addloctocross function.
Value

Plotting routine, no return

Author(s)

Danny Arends <danny.arends@gmail.com>

See Also

- The MQM tutorial: [https://rqtl.org/tutorials/MQM-tour.pdf](https://rqtl.org/tutorials/MQM-tour.pdf)
- MQM - MQM description and references
- mqmscan - Main MQM single trait analysis
- mqmscanall - Parallelized traits analysis
- mqmaugment - Augmentation routine for estimating missing data
- mqmautocofactors - Set cofactors using marker density
- mqmsetcofactors - Set cofactors at fixed locations
- mqmperturbation - Estimate significance levels
- scanone - Single QTL scanning

Examples

data(multitrait)
data(locations)

multifilled <- fill.geno(multitrait) # impute missing genotypes
multicof  <- mqmsetcofactors(multitrait,10) # create cofactors
multiloc  <- addloctocross(multifilled,locations) # add phenotype information to cross
multires  <- mqmscanall(multifilled,cofactors=multicof) # run mqmscan for all phenotypes

#Basic mqmmulti, color = trait, round circle = significant
mqmplot.circle(multifilled,multires)

#mqmmulti with locations of traits in multiloc
mqmplot.circle(multiloc,multires)

#mqmmulti with highlighting
mqmplot.circle(multitrait,multires,highlight=3)

#mqmmulti with locations of traits in multiloc and highlighting
mqmplot.circle(multiloc,multires,highlight=3)
mqmplot.cistrans  

**Description**

Plot results for a genomsan using a multiple-QTL model. With genetic location for the traits it is possible to show cis- and trans- locations, and detect trans-bands

**Usage**

```r
mqmplot.cistrans(result, cross, threshold=5, onlyPEAK=TRUE,
    highPEAK=FALSE, cisarea=10, pch=22, cex=0.5,
    verbose=FALSE, ...)
```

**Arguments**

- **result**: An object of class `mqmmulti`. See `mqmscanall` for details.
- **cross**: An object of class `cross`. See `read.cross` for details.
- **threshold**: Threshold value in LOD. Markers that have a LOD score above this threshold are plotted as small squares (see `pch` parameter). The markers with LODscores below this threshold are not visible.
- **onlyPEAK**: Plot only the peak markers? (TRUE/FALSE) (Peak markers are markers that have a QTL likelihood above `threshold` and higher than other markers in the same region)
- **highPEAK**: Highlight peak markers? (TRUE/FALSE). When using this option peak markers (the marker with the highest LOD score in a region above the threshold gets an 25% increase in size and is displayed in red)
- **cisarea**: Adjust the two green lines around the line `y=x`
- **pch**: What kind of character is used in plotting of the figure (Default: 22, small square)
- **cex**: Size of the points plotted (default to 0.5 half of the original size)
- **verbose**: If TRUE, give verbose output
- **...**: Extra parameters will be passed to points

**Value**

Plotting routine, so no return

**Author(s)**

Danny Arends <danny.arends@gmail.com>
See Also

- The MQM tutorial: [https://rqtl.org/tutorials/MQM-tour.pdf](https://rqtl.org/tutorials/MQM-tour.pdf)
- **MQM** - MQM description and references
- **mqmscan** - Main MQM single trait analysis
- **mqmscanall** - Parallelized traits analysis
- **mqmaugment** - Augmentation routine for estimating missing data
- **mqmautocofactors** - Set cofactors using marker density
- **mqmsetcofactors** - Set cofactors at fixed locations
- **mqmpermutation** - Estimate significance levels
- **scanone** - Single QTL scanning

Examples

```r
data(multitrait)

data(locations)
multiloc <- addloctocross(multitrait, locations)
multiloc <- calc.genoprob(multiloc)
results <- scanall(multiloc, method="hk")
mqmplot.cistrans(results, multiloc, 5, FALSE, TRUE)
```

---

**mqmplot.clusteredheatmap**

*Plot clustered heatmap of MQM scan on multiple phenotypes*

Description

Plot the results from a MQM scan on multiple phenotypes.

Usage

```r
mqmplot.clusteredheatmap(cross, mqmresult, directed=TRUE, legend=FALSE, 
Colv=NA, scale="none", verbose=FALSE, 
breaks = c(-100,-10,-3,0,3,10,100), 
col = c("darkblue","blue","lightblue","yellow", 
"orange","red"), ...)
```

Arguments

- **cross** An object of class `cross`. See `read.cross` for details.
- **mqmresult** Result object from `mqmscanall`, the object needs to be of class `mqmmulti`
- **directed** Take direction of QTLs into account (takes more time because of QTL direction calculations)
- **legend** If TRUE, add a legend to the plot
**mqmplot.cofactors**

Cluster only the Rows, the columns (Markers) should not be clustered.

**scale**
Character indicating if the values should be centered and scaled in either the row direction or the column direction, or none. The default "none".

**verbose**
If TRUE, give verbose output.

**breaks**
Color break points for the LOD scores.

**col**
Colors used between breaks.

**...**
Additional arguments passed to heatmap.

**Author(s)**
Danny Arends <danny.arends@gmail.com>

**See Also**
- The MQM tutorial: [https://rqtl.org/tutorials/MQM-tour.pdf](https://rqtl.org/tutorials/MQM-tour.pdf)
- MQM - MQM description and references
- mqmscan - Main MQM single trait analysis
- mqmscanall - Parallelized traits analysis
- mqmaugment - Augmentation routine for estimating missing data
- mqmautocofactors - Set cofactors using marker density
- mqmsetcofactors - Set cofactors at fixed locations
- mqmpermutation - Estimate significance levels
- scanone - Single QTL scanning

**Examples**

```r
data(multitrait)

multitrait <- fill.geno(multitrait) # impute missing genotype data
result <- mqmscanall(multitrait, logtransform=TRUE)
cresults <- mqmplot.clusteredheatmap(multitrait,result)
groupclusteredheatmap(multitrait, cresults, 10)
```

---

**mqmplot.cofactors**

Plot cofactors on the genetic map.

**Description**
Plots cofactors as created by **mqmsetcofactors** or **mqmautocofactors** on the genetic map.

**Usage**

```r
mqmplot.cofactors(cross, cofactors, ...)
```
Arguments

- **cross**: An object of class `cross`. See `read.cross` for details.
- **cofactors**: List of cofactors to be analysed in the QTL model. To set cofactors use `mqmautocofactors` or `mqmsetcofactors`.
- **...**: Passed to `plot.qtl`

Author(s)

Danny Arends <danny.arends@gmail.com>

See Also

- The MQM tutorial: [https://rqtl.org/tutorials/MQM-tour.pdf](https://rqtl.org/tutorials/MQM-tour.pdf)
- **MQM** - MQM description and references
- **mqmscan** - Main MQM single trait analysis
- **mqmscanall** - Parallelized traits analysis
- **mqmaugment** - Augmentation routine for estimating missing data
- **mqmautocofactors** - Set cofactors using marker density
- **mqmsetcofactors** - Set cofactors at fixed locations
- **mqmpermutation** - Estimate significance levels
- **scanone** - Single QTL scanning

Examples

data(multitrait)
cof1 <- mqmsetcofactors(multitrait,20)
cof2 <- mqmsetcofactors(multitrait,10)
op <- par(mfrow=c(2,1))
mqmplot.cofactors(multitrait,cof1,col="blue")
mqmplot.cofactors(multitrait,cof2,col="blue")
op <- par(mfrow=c(1,1))

mqmplot.directedqtl **Plot LOD*Effect curves of a multiple-QTL model**

Description

Plot the LOD*Effect curve for a genome scan with a multiple-QTL model (the output of `mqmscan`).

Usage

`mqmplot.directedqtl(cross, mqmresult, pheno.col=1, draw = TRUE)`
Arguments

cross        An object of class cross. See read.cross for details.
mqmresult    Results from mqmscan of type scanone
pheno.col    From which phenotype in the cross object are the result calculated
draw         If TRUE, draw the figure.

Value

Returns a scanone object, with added the effectsign calculated internally by the function effect.scan. For more info on the scanone object see: scanone

Author(s)

Danny Arends <danny.arends@gmail.com>

See Also

- The MQM tutorial: https://rqtl.org/tutorials/MQM-tour.pdf
- MQM - MQM description and references
- mqmscan - Main MQM single trait analysis
- mqmscanall - Parallellized traits analysis
- mqmaugment - Augmentation routine for estimating missing data
- mqnautocofactors - Set cofactors using marker density
- mqnsetcofactors - Set cofactors at fixed locations
- mqmpermutation - Estimate significance levels
- scanone - Single QTL scanning

Examples

```r
# Simulated F2 Population
f2qtl <- c(3,15,1,0) # QTL at chromosome 3
data(map10) # Mouse genetic map

f2cross <- sim.cross(map10,f2qtl,n=100,type="f2") # Simulate a F2 Cross
f2cross <- fill.geno(f2cross) # Fill in missing genotypes
f2result <- mqmscan(f2cross) # Do a MQM scan of the genome
mqmplot.directedqtl(f2cross,f2result)
```
mqmplot.heatmap  

*Heatmap of a genome of MQM scan on multiple phenotypes*

---

**Description**

Plotting routine to display a heatmap of results obtained from a multiple-QTL model on multiple phenotypes (the output of `mqmscanall`)

**Usage**

```r
mqmplot.heatmap(cross, result, directed=TRUE, legend=FALSE, breaks = c(-100,-10,-3,0,3,10,100), col = c("darkblue","blue","lightblue","yellow","orange","red"), ...)```

**Arguments**

- `cross` An object of class `cross`. See `read.cross` for details.
- `result` Result object from `mqmscanall`, the object needs to be of class `mqmmulti`.
- `directed` Take direction of QTLs into account (takes more time because of QTL direction calculations)
- `legend` If TRUE, add a legend to the plot
- `breaks` Color break points for the LOD scores
- `col` Colors used between breaks
- `...` Additional arguments passed to the `image` function

**Author(s)**

Danny Arends <danny.arends@gmail.com>

**See Also**

- The MQM tutorial: [https://rqtl.org/tutorials/MQM-tour.pdf](https://rqtl.org/tutorials/MQM-tour.pdf)
- MQM - MQM description and references
- `mqmscan` - Main MQM single trait analysis
- `mqmscanall` - Parallelized traits analysis
- `mqmaugment` - Augmentation routine for estimating missing data
- `mqmautocofactors` - Set cofactors using marker density
- `mqmsetcofactors` - Set cofactors at fixed locations
- `mqmpermutation` - Estimate significance levels
- `scanone` - Single QTL scanning
Examples

```r
data(multitrait)

multitrait <- fill.geno(multitrait) # impute missing genotype data
result <- mqmscanall(multitrait, logtransform=TRUE)
mqmplot.heatmap(multitrait,result)
```

Description

Plotting routine to display the results from a multiple-QTL model on multiple phenotypes. It supports four different visualizations: a contourmap, heatmap, 3D graph or a multiple QTL plot created by using `plot.scanone` on the `mqmmulti` object.

Usage

```r
mqmplot.multitrait(result, type=c("lines","image","contour","3Dplot"),
group=NULL, meanprofile=c("none","mean","median"),
theta=30, phi=15, ...)
```

Arguments

- `result`: Result object from `mqmscanall`
- `type`: Selection of the plot method to visualize the data: "lines" (default plotting option), "image", "contour" and "3Dplot"
- `group`: A numeric vector indicating which traits to plot. NULL means no grouping
- `meanprofile`: Plot a mean/median profile from the group selected
- `theta`: Horizontal axis rotation in a 3D plot
- `phi`: Vertical axis rotation in a 3D plot
- `...`: Additional arguments passed to `plot`.

Author(s)

Danny Arends <danny.arends@gmail.com>

See Also

- The MQM tutorial: [https://rqtl.org/tutorials/MQM-tour.pdf](https://rqtl.org/tutorials/MQM-tour.pdf)
- MQM - MQM description and references
- `mqmscan` - Main MQM single trait analysis
- `mqmscanall` - Parallellized traits analysis
• \texttt{mqmaugment} - Augmentation routine for estimating missing data
• \texttt{mqmautocofactors} - Set cofactors using marker density
• \texttt{mqmsetcofactors} - Set cofactors at fixed locations
• \texttt{mqmpermutation} - Estimate significance levels
• \texttt{scanone} - Single QTL scanning

\textbf{Examples}

\begin{verbatim}
data(multitrait)
multitrait <- fill.geno(multitrait) # impute missing genotype data
result <- mqmscanall(multitrait, logtransform=TRUE)
mqmplot.multitrait(result,"lines")
mqmplot.multitrait(result,"contour")
mqmplot.multitrait(result,"image")
mqmplot.multitrait(result,"3Dplot")
\end{verbatim}

\textbf{mqmplot.permutations} \hspace{1cm} \textit{Plot results from mqmpermutation}

\begin{verbatim}
Description
Plotting routine to display the results from a permutation QTL scan. (the output of \texttt{mqmpermutation})

Usage
mqmplot.permutations(permutationresult, ...)

Arguments
permutationresult
mqmmulti object returned by \texttt{mqmpermutation} permutation analysis.
...
Extra arguments passed to \texttt{polyplot}

Value
No value returned (plotting routine)

Author(s)
Danny Arends <danny.arends@gmail.com>, Rutger Brouwer
\end{verbatim}
See Also

- The MQM tutorial: https://rqtl.org/tutorials/MQM-tour.pdf
- MQM - MQM description and references
- mqmscan - Main MQM single trait analysis
- mqmscanall - Parallellized traits analysis
- mqmaugment - Augmentation routine for estimating missing data
- mqmautocofactors - Set cofactors using marker density
- mqmsetcofactors - Set cofactors at fixed locations
- mqmpermutation - Estimate significance levels
- scanone - Single QTL scanning

Examples

```r
# Simulated F2 Population
# QTL at chromosome 3
f2qtl <- c(3,15,1,0)

# Mouse genetic map
data(map10)

# Simulate a F2 Cross
f2cross <- sim.cross(map10,f2qtl,n=100,type="f2")
f2cross <- calc.genoprob(f2cross)
## Not run: # Permutations to obtain significance threshold
f2result <- mqmpermutation(f2cross, n.perm=1000, method="permutation")
## End(Not run)

# Plot results
mqmplot.permutations(f2result)
```

mqmplot.singletrait  Plot LOD curves of a multiple-QTL model

Description

Plot the LOD curve for a genome scan for a single trait, with a multiple-QTL model (the output of
mqmscan).

Usage

mqmplot.singletrait(result, extended = 0 ,...
mqmprocesspermutation

Arguments

result : mqmscan result.
extended : Extended plotting of the information content
... : Extra arguments passed to plot.scanone

Author(s)

Danny Arends <danny.arends@gmail.com>

See Also

- The MQM tutorial: https://rqtl.org/tutorials/MQM-tour.pdf
- MQM - MQM description and references
- mqmscan - Main MQM single trait analysis
- mqmscanall - Parallellized traits analysis
- mqmaugment - Augmentation routine for estimating missing data
- mqmautocofactors - Set cofactors using marker density
- mqmsetcofactors - Set cofactors at fixed locations
- mqmpermutation - Estimate significance levels
- scanone - Single QTL scanning

Examples

# Simulated F2 Population
f2qtl <- c(3,15,1,0) # QTL at chromosome 3
data(map10) # Mouse genetic map
f2cross <- sim.cross(map10,f2qtl,n=100,type="f2") # Simulate a F2 Cross
f2cross <- mqmaugment(f2cross)
f2result <- mqmscan(f2cross) # Do a MQM scan of the genome
mqmplot.singletrait(f2result) # Use our fancy plotting routine

Description

Function to convert mqmmulti objects into a scanoneperm object, this allows the use of R/qtl methods for permutation analysis that do not support the output of a multiple QTL scan using mqm’s outputstructure.

Usage

mqmprocesspermutation(mqmpermutationresult = NULL)
**Arguments**

`mqmpermutationresult`

mqmmulti object obtained after performing permutations on a single trait using the function `mqmpermutation`.

**Value**

Output of the algorithm is a `scanoneperm` object. See also: `summary.scanoneperm`

**Author(s)**

Ritsert C Jansen; Danny Arends; Pjotr Prins; Karl W Broman <broman@wisc.edu>

**See Also**

- The MQM tutorial: [https://rqtl.org/tutorials/MQM-tour.pdf](https://rqtl.org/tutorials/MQM-tour.pdf)
- MQM - MQM description and references
- `mqmscan` - Main MQM single trait analysis
- `mqmscanall` - Parallelized traits analysis
- `mqnaugment` - Augmentation routine for estimating missing data
- `mqmautocofactors` - Set cofactors using marker density
- `mqmsetcofactors` - Set cofactors at fixed locations
- `mqmpermutation` - Estimate significance levels
- `scanone` - Single QTL scanning

**Examples**

```r
# QTL at chromosome 3
f2qtl <- c(3,15,1,0)

# Mouse genetic map
data(map10)

# Simulate a F2 Cross
f2cross <- sim.cross(map10,f2qtl,n=100,type="f2")
## Not run: # Bootstrap MQM mapping on the f2cross
f2result <- mqmpermutation(f2cross,scanfunction=mqmscan)
## End(Not run)

# Create a permutation object
f2perm <- mqmprocesspermutation(f2result)

# What LOD score is considered significant?
summary(f2perm)
```
mqmscan

Genome scan with a multiple QTL model (MQM)

Description

Genome scan with a multiple QTL model.

Usage

mqmscan(cross, cofactors=NULL, pheno.col = 1,
model=c("additive","dominance"), forceML=FALSE,
cofactor.significance=0.02, em.iter=1000,
window.size=25.0, step.size=5.0,
logtransform = FALSE, estimate.map = FALSE,
plot=FALSE, verbose=FALSE, outputmarkers=TRUE,
multicore=TRUE, batchsize=10, n.clusters=1, test.normality=FALSE, off.end=0)

Arguments

cross An object of class cross. See read.cross for details.
cofactors List of cofactors to be analysed as cofactors in backward elimination procedure when building the QTL model. See mqmsetcofactors on how-to manually set cofactors for backward elimination. Or use mqmautocofactors for automatic selection of cofactors. Only three kind of (integer) values are allowed in the cofactor list. (0: no cofactor at this marker, 1: Use this marker as an additive cofactor, 2: Use this marker as an sexfactor (Dominant cofactor))
pheno.col Column number in the phenotype matrix which should be used as the phenotype. This can be a vector of integers; One may also give a character strings matching the phenotype names. Finally, one may give a numeric vector of phenotypeIDs. This should consist of integers with 0 < value < no. phenotypes.
model When scanning for QTLs should haplotype dominance be considered in an F2 intercross. Using the dominance model we scan for additive effects but also allow an additional effect where AA+AB versus BB and AA versus AB+BB. This setting is ignored for BC and RIL populations
forceML Specify which statistical method to use to estimate variance components to use when QTL modeling and mapping. Default usage is the Restricted maximum likelihood approach (REML). With this option a user can disable REML and use maximum likelihood.
cofactor.significance Significance level at which a cofactor is considered significant. This is estimated using an analysis of deviance, and compared to the level specified by the user. The cofactors that dont reach this level of statistical significance are NOT used in the mapping stage. Value between 0 and 1
em.iter Maximum number of iterations for the EM algorithm to converge
window.size Window size for mapping QTL locations, this parameter is used in the interval mapping stage. When calculating LOD scores at a genomic position all cofactors within window.size are dropped to estimate the (unbiased) effect of the location under interest.

step.size Step size used in interval mapping. A lower step.size parameter increases the number of output points, this creates a smoother QTL profile.

off.end Distance (in cM) past the terminal markers on each chromosome to which the genotype simulations will be carried.

logtransform Indicate if the algorithm should do a log transformation on the trait data in the pheno.col.

estimate.map Should Re-estimation of the marker locations on the genetic map occur before mapping QTLs. This method is deprecated rather use the est.map function in R/qtl. This is because no map is returned into the crossobject. The old map remains in the cross object.

plot plot the results (default FALSE)

verbose verbose output

outputmarkers If TRUE (the default), the results include the marker locations as well as along a grid of pseudomarkers; if FALSE, the results include only the grid positions.

multicore Use multicore (if available).

batchsize Number of traits being analyzed as a batch.

n.clusters Number of child processes to split the job into.

test.normality If TRUE, test whether the phenotype follows a normal distribution via mqmtestnormal.

Value

When scanning a single phenotype the function returns a scanone object. The object contains a matrix of three columns for LOD scores, information content and LOD*information content with pseudo markers sorted in increasing order. For more information on the scanone object see: scanone

Note

The resulting scanone object itself can be visualized using the standard R/qtl plotting routines (plot.scanone) or specialized function to show the mqm model (mqmplot.singletrait) and QTL profile. If cofactors were specified the QTL model used in scanning is also returned as a named attribute of the scanone object called mqmmodel. It can be extracted from the resulting scanone object by using the mqmgetmodel function or the attr function.

Also note the estimate.map parameter does not return its re-estimated genetic map, although it is used internally. When scanning multiple genotypes a mqmmulti object is created. This object is just a list composed of scanone objects. The results for a single trait can be obtained from the mqmmulti object, in scanone format.

Author(s)

Ritsert C Jansen; Danny Arends; Pjotr Prins; Karl W Broman <broman@wisc.edu>
See Also

- The MQM tutorial: https://rqtl.org/tutorials/MQM-tour.pdf
- MQM - MQM description and references
- mqmscan - Main MQM single trait analysis
- mqmscanall - Parallelized traits analysis
- mqmaugment - Augmentation routine for estimating missing data
- mqmautocofactors - Set cofactors using marker density
- mqmsetcofactors - Set cofactors at fixed locations
- mqmpermutation - Estimate significance levels
- scanone - Single QTL scanning

Examples

data(map10)  # Genetic map modeled after mouse

# simulate a cross (autosomes 1-10)
qtl <- c(3, 15, 1, 0)  # QTL model: chr, pos'n, add'Ve & dom effects
cross <- sim.cross(map10[1:10], qtl, n=100, missing.prob=0.01)

crossaug <- mqmaugment(cross)  # Augmentation
cat(crossaug$mqm$Nind, 'real individuals retained in dataset',
crossaug$mqm$Naug, 'individuals augmented

result <- mqmscan(crossaug)  # Scan

# show LOD interval of the QTL on chr 3
lodint(result, chr=3)

mqmscanall

Parallelized MQM on multiple phenotypes in a cross object

Description

Parallelized QTL analysis using MQM on multiple phenotypes in a cross object (uses SNOW)

Usage

mqmscanall(cross, multicore=TRUE, n.clusters = 1, batchsize=10, cofactors=NULL, ...)

mqmscanall
mqmscanall

Arguments

cross              An object of class cross. See read.cross for details.
multicore          Use multiple cores (only if the package SNOW is available, otherwise this setting will be ignored)
n.clusters         Number of parallel processes to spawn, recommended is setting this lower than the number of cores in the computer
batchsize          Batch size. The entire set is split in jobs to reduce memory load per core. Each job contains batchsize number of traits per job.
cofactors          cofactors, only used when scanfunction is mqmscan. List of cofactors to be analysed in the QTL model. To set cofactors use mqmautocofactors or mqmsetcofactors.
...                Parameters passed through to the mqmscan function used in scanning for QTLs

Details

Uses mqmscan to scan for QTL’s for each phenotype in the cross object. It is recommended that the package SNOW is installed before using this function on large numbers of phenotypes.

Value

Returns a MQMmulti object. This object is a list of scanone objects that can be plotted using plot.scanone(result[[trait]]) or using mqmplot.multitrait(result)

Author(s)

Ritsert C Jansen; Danny Arends; Pjotr Prins; Karl W Broman <broman@wisc.edu>

References


See Also

- The MQM tutorial: [https://rqtl.org/tutorials/MQM-tour.pdf](https://rqtl.org/tutorials/MQM-tour.pdf)
- MQM - MQM description and references
- mqmscan - Main MQM single trait analysis
- mqmscanall - Parallelized traits analysis
- mqmaugment - Augmentation routine for estimating missing data
- mqmautocofactors - Set cofactors using marker density
- mqmsetcofactors - Set cofactors at fixed locations
- mqmpermutation - Estimate significance levels
- scanone - Single QTL scanning
**Examples**

```r
# Doing a multitrait analysis
data(multitrait)
multitrait <- calc.genoprob(multitrait)
cof <- mqmsetcofactors(multitrait,3)
multitrait <- fill.geno(multitrait)
result <- mqmscanall(multitrait, cofactors=cof, batchsize=5)
mqmplot.multitrait(result, "lines")
```

---

**mqmscanfdr**

*Estimate FDR for multiple trait QTL analysis*

**Description**

Estimate the false discovery rate (FDR) for multiple trait analysis

**Usage**

```r
mqmscanfdr(cross, scanfunction=mqmscanall, 
thresholds=c(1,2,3,4,5,7,10,15,20), n.perm=10, 
verbose=FALSE, ...)
```

**Arguments**

- `cross`: An object of class `cross`. See `read.cross` for details.
- `scanfunction`: QTL mapping function. Note: Must use scanall or mqmscanall. Otherwise this will not produce useful results. Reason: We need a function that maps all traits because of the correlation structure which is not changed (between traits) during permutation (Valis options: scanall or mqmscanall).
- `thresholds`: False discovery rate (FDR) is calculated for peaks above these LOD thresholds (DEFAULT=Range from 1 to 20, using 10 thresholds) Parameter is a list of LOD scores at which FDR is calculated.
- `n.perm`: Number of permutations (DEFAULT=10 for quick analysis, however for publications use 1000, or higher)
- `verbose`: verbose output
- `...`: Parameters passed to the mapping function

**Details**

This function wraps the analysis of `scanone`, `cim` and `mqmscan` to scan for QTL in shuffled/randomized data. It is recommended to also install the snow library for parallelization of calculations. The snow library allows calculations to run on multiple cores or even scale it up to an entire cluster, thus speeding up calculation by the number of computers used.
Value

Returns a data.frame with 3 columns: FalsePositives, FalseNegatives and False Discovery Rates. In the rows the userspecified thresholds are with scores for the 3 columns.

Author(s)

Ritsert C Jansen; Danny Arends; Pjotr Prins; Karl W Broman <broman@wisc.edu>

References


See Also

- The MQM tutorial: https://rqtl.org/tutorials/MQM-tour.pdf
- MQM - MQM description and references
- mqmscan - Main MQM single trait analysis
- mqmscanall - Parallelized traits analysis
- mqmnaugment - Augmentation routine for estimating missing data
- mqaoutocofactors - Set cofactors using marker density
- mqmsetcofactors - Set cofactors at fixed locations
- mqmpermutation - Estimate significance levels
- scanone - Single QTL scanning

Examples

data(multitrait)

# impute missing genotype data
multitrait <- fill.geno(multitrait)
## Not run: # Calculate the thresholds
result <- mqmscanfdr(multitrait, threshold=10.0, n.perm=1000)

## End(Not run)
mqmsetcofactors

Set cofactors at fixed intervals, to be used with MQM

Description

Set cofactors, at fixed marker intervals. Together with mqmscan cofactors are selected through backward elimination.

Usage

mqmsetcofactors(cross, each = NULL, cofactors=NULL, sexfactors=NULL, verbose=FALSE)

Arguments

cross An object of class cross. See read.cross for details.
each Every 'each' marker will be used as a cofactor, when each is used the cofactors and sexfactors parameter is ignored
cofactors List of cofactors to be analysed in the QTL model. To set cofactors use mqmautocofactors or mqmsetcofactors; when each is set, this parameter is ignored
sexfactors list of markers which should be treated as dominant cofactors (sexfactors), when each is set, this parameter is ignored
verbose If TRUE, print tracing information.

Value

An list of cofactors to be passed into mqmscan.

Author(s)

Ritsert C Jansen; Danny Arends; Pjotr Prins; Karl W Broman <broman@wisc.edu>

See Also

• The MQM tutorial: https://rqtl.org/tutorials/MQM-tour.pdf
• MQM - MQM description and references
• mqmscan - Main MQM single trait analysis
• mqmscanall - Parallelized traits analysis
• mqmaugment - Augmentation routine for estimating missing data
• mqmautocofactors - Set cofactors using marker density
• mqmsetcofactors - Set cofactors at fixed locations
• mqmpermutation - Estimate significance levels
• scanone - Single QTL scanning
**Examples**

```r
data(hyper)  # Hyper dataset
hyperfilled <- fill.geno(hyper)
# Automatic cofactors every third marker
cofactors <- mqmsetcofactors(hyperfilled,3)
result <- mqmscan(hyperfilled,cofactors)  # Backward model selection
mqmgetmodel(result)
# Manual cofactors at markers 3,6,9,12,40 and 60
cofactors <- mqmsetcofactors(hyperfilled,cofactors=c(3,6,9,12,40,60))
result <- mqmscan(hyperfilled,cofactors)  # Backward model selection
mqmgetmodel(result)
```

---

**mqmtestnormal**  
*Shapiro normality test used for MQM*

**Description**

Wraps a shapiro’s normality test from the nortest package. This function is used in MQM to test the normality of the trait under investigation.

**Usage**

```r
mqmtestnormal(cross, pheno.col = 1, significance=0.05, verbose=FALSE)
```

**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. This can be a vector of integers.
- `significance`: Significance level used in the normality test. Lower significance levels will accept larger deviations from normality.
- `verbose`: If TRUE, print result as well as return it.

**Details**

For augmented data (as from `mqmaugment`), the cross is first reduced to distinct individuals. Furthermore the shapiro used to test normality works only for 3 \leq nind(cross) \leq 5000.

**Value**

Boolean indicating normality of the trait in `pheno.col`. (FALSE when not normally distributed.)

**Author(s)**

Danny Arends <danny.arends@gmail.com>
See Also

- `shapiro.test` - Function wrapped by our `mqmtestnormal`
- The MQM tutorial: [https://rqtl.org/tutorials/MQM-tour.pdf](https://rqtl.org/tutorials/MQM-tour.pdf)
- `mqm` - MQM description and references
- `mqmscan` - Main MQM single trait analysis
- `mqmscanall` - Parallelized traits analysis
- `mqmaugment` - Augmentation routine for estimating missing data
- `mqmautocofactors` - Set cofactors using marker density
- `mqmsetcofactors` - Set cofactors at fixed locations
- `mqmpermutation` - Estimate significance levels
- `scanone` - Single QTL scanning

Examples

```r
data(multitrait)

# test normality of 7th phenotype
mqmtestnormal(multitrait, pheno.col=7)

# take log
multitrait <- transformPheno(multitrait, pheno.col=7, transf=log)

# test again
mqmtestnormal(multitrait, pheno.col=7)
```

multitrait

Example Cross object from R/QTL with multiple traits

Description

Cross object from R/QTL, an object of class `cross` from R/QTL. See `read.cross` for details.

Usage

```r
data(multitrait)
```

Format

Cross object from R/QTL

Details

Arabidopsis recombinant inbred lines by selfing. There are 162 lines, 24 phenotypes, and 117 markers on 5 chromosomes.
Source

Part of the Arabidopsis RIL selfing experiment with Landsberg erecta (Ler) and Cape Verde Islands (Cvi) with 162 individuals scored (with errors at) 117 markers. Dataset obtained from GBIC - Groningen BioInformatics Centre

References


Examples

data(multitrait) # Load dataset
multitrait <- fill.geno(multitrait) # impute missing genotype data
result <- mqmscanall(multitrait, logtransform=TRUE) # Analyse all 24 traits

---

`nchr`

*Determine the number of chromosomes*

**Description**

Determine the number of chromosomes in a cross or map object.

**Usage**

`nchr(object)`

**Arguments**

- `object` : An object of class `cross` (see `read.cross` for details) or `map` (see `sim.map` for details).

**Value**

The number of chromosomes in the input.

**Author(s)**

Karl W Broman, <brroman@wisc.edu>
nind

See Also

read.cross, plot.cross, summary.cross, nind, totmar, nmar, nphe

Examples

data(fake.f2)
nchr(fake.f2)
map <- pull.map(fake.f2)
nchr(map)

nind

Determine the number of individuals QTL experiment

Description

Determine the number of individuals in cross object.

Usage

nind(object)

Arguments

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

Value

The number of individuals in the input cross object.

Author(s)

Karl W Broman, <broman@wisc.edu>

See Also

read.cross, plot.cross, summary.cross, nmar, nchr, totmar, nphe

Examples

data(fake.f2)
nind(fake.f2)
**nmar**

*Determine the numbers of markers on each chromosome*

**Description**

Determine the number of markers on each chromosome in a cross or map object.

**Usage**

```r
nmar(object)
```

**Arguments**

- `object` An object of class `cross` (see `read.cross` for details) or `map` (see `sim.map` for details).

**Value**

A vector with the numbers of markers on each chromosome in the input.

**Author(s)**

Karl W Broman, <broman@wisc.edu>

**See Also**

`read.cross, plot.cross, summary.cross, nind, nchr, totmar, nphe`

**Examples**

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

---

**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, <broman@wisc.edu>

See Also

- `ntyped`, `summary.cross`, `nind`, `totmar`

Examples

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"))
nqrank

See Also

read.cross, plot.cross, summary.cross, nmar, nchr, totmar, nind

Examples

data(fake.f2)
nphe(fake.f2)

nqrank

Transform a vector of quantitative values to the corresponding normal quantiles

Description

Transform a vector of quantitative values to the corresponding normal quantiles (preserving the mean and SD).

Usage

nqrank(x, jitter)

Arguments

x A numeric vector
jitter If TRUE, randomly jitter the values to break ties.

Value

A numeric vector; the input x is converted to ranks and then to normal quantiles.

Author(s)

Karl W Broman, <broman@wisc.edu>

See Also

rank, qnorm, transformPheno

Examples

data(hyper)

hyper <- transformPheno(hyper, pheno.col=1, transf=nqrank)
nqtl

Determine the number of QTL in a QTL object

Description

Determine the number of QTL in a QTL object.

Usage

nqtl(qtl)

Arguments

qtl

An object of class qtl. See makeqtl for details.

Value

The number of QTL in the input QTL object.

Author(s)

Karl W Broman, <broman@wisc.edu>

See Also

makeqtl, fitqtl, dropfromqtl, replaceqtl, addtoqtl, summary.qtl, reorderqtl

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 <- calc.genoprob(fake.f2, step=2, err=0)
qtl1 <- makeqtl(fake.f2, qc, qp, what="prob")
nqtl(qtl1)
ntyped

Description

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

Usage

ntyped(cross, what=c("ind","mar"))

Arguments

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

Value

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

Author(s)

Karl W Broman, <broman@wisc.edu>

See Also

nmissing, summary.cross, nind, totmar

Examples

data(listeria)

# plot number of genotypes for each individual
plot(ntyped(listeria))

# plot number of genotypes for each marker
plot(ntyped(listeria, what="mar"))
nullmarkers

Identify markers without any genotype data

Description

Identify markers in a cross that have no genotype data.

Usage

nullmarkers(cross)

Arguments

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

Value

Marker names (a vector of character strings) with no genotype data.

Author(s)

Karl W Broman, <broman@wisc.edu>

See Also

drop.nullmarkers

Examples

# one marker with no data
data(hyper)
nullmarkers(hyper)

# nothing in listeria
data(listeria)
nullmarkers(listeria)
Find an initial order for markers within chromosomes

**Description**

Establish initial orders for markers within chromosomes by a greedy algorithm, adding one marker at a time with locations of previous markers fixed, in the position giving the minimum number of obligate crossovers.

**Usage**

```
orderMarkers(cross, chr, window=7, use.ripple=TRUE, error.prob=0.0001,
            map.function=c("haldane","kosambi","c-f","morgan"),
            maxit=4000, tol=1e-4, sex.sp=TRUE, verbose=FALSE)
```

**Arguments**

- `cross` An object of class `cross`. See `read.cross` for details.
- `chr` Optional vector indicating the chromosomes to consider. This should be a vector of character strings referring to chromosomes by name; numeric values are converted to strings. Refer to chromosomes with a preceding - to have all chromosomes but those considered. A logical (TRUE/FALSE) vector may also be used.
- `window` If `use.ripple=TRUE`, this indicates the 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.
- `use.ripple` If TRUE, the initial order is refined by a call to the function `ripple`.
- `error.prob` Assumed genotyping error rate used in the final estimated map.
- `map.function` Indicates the map function to use in the final estimated map.
- `maxit` Maximum number of EM iterations to perform in the final estimated map.
- `tol` Tolerance for determining convergence in the final estimated map.
- `sex.sp` Indicates whether to estimate sex-specific maps in the final estimated map; this is used only for the 4-way cross.
- `verbose` If TRUE, information about the progress of the calculations is displayed; if > 1, even more information is given.

**Details**

Markers within a linkage group are considered in order of decreasing number of genotyped individuals. The first two markers are placed in an arbitrary order. Additional markers are considered one at a time, and each possible placement of a marker is compared (with the order of the previously placed markers taken as fixed) via the number of obligate crossovers (that is, the minimal number of crossovers that would explain the observed data). The marker is placed in the position giving
the minimal number of obligate crossovers. If multiple positions give the same number of obligate
crossovers, a single location (among those positions) is chosen at random.

If use.ripple=TRUE, the final order is passed to ripple with method="countxo" to refine the
marker order. If use.ripple=TRUE and the number of markers on a chromosome is ≤ the argument
window, the initial greedy algorithm is skipped and all possible marker orders are compared via
ripple.

Value

The output is a cross object, as in the input, with orders of markers on selected chromosomes
revised.

Author(s)

Karl W Broman, <broman@wisc.edu>

See Also

formLinkageGroups, ripple, est.map, countX0

Examples

data(listeria)
pull.map(listeria, chr=3)
revcross <- orderMarkers(listeria, chr=3, use.ripple=FALSE)
pull.map(revcross, chr=3)

---

**phenames**

*Pull out the phenotypes names from a cross*

Description

Pull out the phenotype names from a cross object as a vector.

Usage

`phenames(cross)`

Arguments

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

Value

A vector of character strings (the phenotype names).

Author(s)

Karl W Broman, <broman@wisc.edu>
See Also
markednames, chrnames

Examples

```r
data(listeria)
phenames(listeria)
```

---

### Description

Identify the largest subset of markers for which no two adjacent markers are separated by less than some specified distance; if weights are provided, find the marker subset for which the sum of the weights is maximized.

### Usage

```r
pickMarkerSubset(locations, min.distance, weights)
```

### Arguments

- `locations`: A vector of marker locations.
- `min.distance`: Minimum distance between adjacent markers in the chosen subset.
- `weights`: (Optional) vector of weights for the markers. If missing, we take `weights == 1`.

### Details

Let $d_i$ be the location of marker $i$, for $i \in 1, \ldots, M$. We use the dynamic programming algorithm of Broman and Weber (1999) to identify the subset of markers $i_1, \ldots, i_k$ for which $d_{i_{j+1}} - d_{i_j} \leq \text{min.distance}$ and $\sum w_{i_j}$ is maximized.

If there are multiple optimal subsets, we pick one at random.

### Value

A vector of marker names.

### Author(s)

Karl W Broman, <broman@wisc.edu>

### References

See Also

drop.markers, pull.markers, findDupMarkers

Examples

data(hyper)

# subset of markers on chr 4 spaced >= 5 cM
pickMarkerSubset(pull.map(hyper)[[4]], 5)

# no. missing genotypes at each chr 4 marker
n.missing <- nmissing(subset(hyper, chr=4), what="mar")

# weight by -log(prop'n missing), but don't let 0 missing go to +Inf
wts <- -log( (n.missing+1) / (nind(hyper)+1) )

# subset of markers on chr 4 spaced >= 5 cM, with weights = -log(prop'n missing)
pickMarkerSubset(pull.map(hyper)[[4]], 5, wts)

plot.comparegeno  

Description

Plot the results of the comparison of all pairs of individuals' genotypes. A histogram of the proportion of matching genotypes, with tick marks at individual values below, via rug.

Usage

## S3 method for class 'comparegeno'
plot(x, breaks=NULL, main="", xlab="Proportion matching genotypes", ...)

Arguments

x An object of class "comparegeno", as produced by comparegeno.
breaks Passed to hist, with the default $2\sqrt{n}$ where $n$ is the number of pairs of individuals.
main Title for the plot.
xlab x-axis label for the plot.
... Passed to hist.

Details

Creates a histogram with hist with ticks at individual values using rug.
Value
None.

Author(s)
Karl W Broman, <broman@wisc.edu>

See Also
comparegeno, summary.comparegeno

Examples

data(fake.f2)
cg <- comparegeno(fake.f2)
plot(cg)

---

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 `plotMissing`, `plotMap` and `plotPheno` 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 `plotPheno`.

Value

None.

Author(s)

Karl W Broman, <broman@wisc.edu>; Brian Yandell

See Also

`plotMissing`, `plotMap`, `plotPheno`

Examples

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

---

**plot.qtl**

*Plot QTL locations*

Description

Plot the locations of the QTL against a genetic map

Usage

```r
## S3 method for class 'qtl'
plot(x, chr, horizontal=FALSE, shift=TRUE,
     show.marker.names=FALSE, alternate.chrid=FALSE, justdots=FALSE,
     col="red", ...)
```

Arguments

- **x**
  - An object of class "qtl", as produced by `makeqtl`.

- **chr**
  - Optional vector indicating the chromosomes to plot. This should be a vector of character strings referring to chromosomes by name; numeric values are converted to strings. Refer to chromosomes with a preceding ~ to have all chromosomes but those considered. A logical (TRUE/FALSE) vector may also be used.
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.
justdots If FALSE, just plot dots at the QTL, rather than arrows and QTL names.
col Color used to plot indications of QTL
... Passed to plotMap.

Details
Creates a plot, via plotMap, and indicates the locations of the QTL in the input QTL object, x.

Value
None.

Author(s)
Karl W Broman, <broman@wisc.edu>

See Also
plotMap, makeqtl

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.f22 <- subset(fake.f2, chr=qc)

fake.f22 <- calc.genoprob(fake.f22, step=2, err=0.001)
qtl <- makeqtl(fake.f22, qc, qp, what="prob")
plot(qtl)
plot(qtl, justdots=TRUE, col="seagreen")
plot.rfmatrix  
Plot recombination fractions or LOD scores for a single marker

Description
Plot a slice (corresponding to a single marker) through the pairwise recombination fractions or LOD scores calculated by est.rf and extracted with pull.rf.

Usage
```r
## S3 method for class 'rfmatrix'
plot(x, marker, ...)
```

Arguments
- `x`: An object of class `rfmatrix`, as output by `pull.rf`.
- `marker`: A single marker name, as a character string.
- `...`: Optional arguments passed to `plot.scanone`.

Value
An object of class "scanone" (as output by `scanone`, and which may be summarized by `summary.scanone` or plotted with `plot.scanone`), containing the estimated recombination fractions or LOD scores for the input marker against all others.

Author(s)
Karl W Broman, <broman@wisc.edu>

See Also
- `pull.rf`, `est.rf`, `plotRF`

Examples
```r
data(fake.f2)
fake.f2 <- est.rf(fake.f2)
marker <- markernames(fake.f2, chr=5)[6]
 lod <- pull.rf(fake.f2, "lod")
plot(lod, marker, bandcol="gray70")
```
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,
     bandcol=NULL, type="l", cex=1, pch=1, bg="transparent",
     bgrect=NULL, ...)  
```

Arguments

- **x**: An object of class "scanone", as output by `scanone`.
- **x2**: Optional second `scanone` object.
- **x3**: Optional third `scanone` object.
- **chr**: Optional vector indicating the chromosomes to plot. This should be a vector of character strings referring to chromosomes by name; numeric values are converted to strings. Refer to chromosomes with a preceding - to have all chromosomes but those considered. A logical (TRUE/FALSE) vector may also be used.
- **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.
bandcol

Optional color for alternating bands to indicate chromosomes. If NULL (the default), no bands are plotted. A good choice might be bandcol="gray70".

type

Type of plot (see plot): for example, type="l" for lines or type="p" for points only, may be of length 1 or 3.

cex

Point size expansion, for example if type="p" is used. May be of length 1 or 3.

pch

Point type, for example if type="p" is used. See points. May be of length 1 or 3.

bg

Background color for points, for example if type="p" and pch=21 are used. See points. May be of length 1 or 3.

bgrect

Optional background color for the rectangular plotting region.

...

Passed to the function plot when it is called.

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, <broman@wisc.edu>

See Also

scanone, summary.scanone, par, colors, add.threshold, xaxisloc.scanone

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, out.em, 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 results of bootstrap for QTL position

Description

Plot a histogram of the results of a nonparametric bootstrap to assess uncertainty in QTL position.

Usage

## S3 method for class 'scanoneboot'
plot(x, ...)

Arguments

x An object of class "scanoneboot", as output by scanoneboot.
...
Passed to the function hist when it is called.

Details

The function plots a histogram of the bootstrap results obtained by scanoneboot. Genetic marker locations are displayed by vertical lines at the bottom of the plot.

Value

None.

Author(s)

Karl W Broman, <broman@wisc.edu>

See Also

scanone, summary.scanoneboot
Examples

```r
data(fake.f2)
fake.f2 <- calc.genoprob(fake.f2, step=1)

# Not run: out.boot <- scanoneboot(fake.f2, chr=13, method="hk")

summary(out.boot)
plot(out.boot)
```

plot.scanoneperm  
*Plot permutation results for a single-QTL genome scan*

Description

Plot a histogram of the permutation results from a single-QTL genome scan.

Usage

```r
## S3 method for class 'scanoneperm'
plot(x, lodcolumn=1, ...)
```

Arguments

- `x`: An object of class "scanoneperm", as output by `scanone` when `n.perm` is specified.
- `lodcolumn`: This indicates the LOD score column to plot. This should be a single number between 1 and the number of LOD columns in the object input.
- `...`: Passed to the function `hist` when it is called.

Details

The function plots a histogram of the permutation results obtained by `scanone` when `n.perm` is specified. If separate permutations were performed for the autosomes and the X chromosome (using `perm.Xsp=TRUE`), separate histograms are given.

Value

None.

Author(s)

Karl W Broman, <broman@wisc.edu>

See Also

`scanone`, `summary.scanoneperm`
Examples

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

operm <- scanone(fake.bc, method="hk", n.perm=100)
plot(operm)

plot.scanPhyloQTL

Plot LOD curves from single-QTL scan to map QTL to a phylogenetic tree

Description

Plot the LOD curves for each partition for a genome scan with a single diallelic QTL (the output of scanPhyloQTL).

Usage

## S3 method for class 'scanPhyloQTL'
plot(x, chr, incl.markers=TRUE, col, xlim, ylim, lwd=2, gap=25, mtick=c("line", "triangle"), show.marker.names=FALSE, alternate.chrid=FALSE, legend=TRUE, ...)

Arguments

x An object of class "scanPhyloQTL", as output by scanPhyloQTL.

chr Optional vector indicating the chromosomes to plot. This should be a vector of character strings referring to chromosomes by name; numeric values are converted to strings. Refer to chromosomes with a preceding - to have all chromosomes but those considered. A logical (TRUE/FALSE) vector may also be used.

incl.markers Indicate whether to plot line segments at the marker locations.

col Optional vector of colors to use for each partition.

xlim Limits for x-axis (optional).

ylim Limits for y-axis (optional).

lwd Line width.

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.

legend Indicates whether to include a legend in the plot.

... Passed to the function plot.scanone when it is called.
Value
None.

Author(s)
Karl W Broman, <broman@wisc.edu>

References

See Also
scanPhyloQTL, max.scanPhyloQTL, summary.scanPhyloQTL, plot.scanone, inferredpartitions, simPhyloQTL, par, colors

Examples
## Not run:
# example map; drop X chromosome
data(map10)
map10 <- map10[1:19]

# simulate data
x <- simPhyloQTL(4, partition="AB|CD", crosses=c("AB", "AC", "AD"),
                   map=map10, n.ind=150,
                   model=c(1, 50, 0.5, 0))

# run calc.genoprob on each cross
x <- lapply(x, calc.genoprob, step=2)

# scan genome, at each position trying all possible partitions
out <- scanPhyloQTL(x, method="hk")

# maximum peak
max(out, format="lod")

# approximate posterior probabilities at peak
max(out, format="postprob")

# all peaks above a threshold for LOD(best) - LOD(2nd best)
summary(out, threshold=1, format="lod")

# all peaks above a threshold for LOD(best), showing approx post'r prob
summary(out, format="postprob", threshold=3)

# plot of results
plot(out)
## End(Not run)
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("viridis", "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 indicating the chromosomes to plot. This should be a vector of character strings referring to chromosomes by name; numeric values are converted to strings. Refer to chromosomes with a preceding - to have all chromosomes but those considered. A logical (TRUE/FALSE) vector may also be used.
- **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 lower-right and upper-left triangles, respectively. If one number is given, the same limits are used for both triangles. If `zlim` is 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.
- **zscale**: If TRUE, a color scale is plotted at the right.
plot.scantwo

point.at.max  If TRUE, plot an X at the maximum LOD.

col.scheme  Name of color pallet. The default is "viridis"; see Option D at https://bids.github.io/colormap

gamma  Parameter affecting range of colors when col.scheme="gray" or="redblue".

allow.neg  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.

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 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 under 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.

Output of addpair

Note that, for output from addpair in which the new loci are indicated explicitly in the formula, the summary provided by plot.scantwo is somewhat special. In particular, the lower and upper arguments are ignored.

In the case that the formula used in addpair was not symmetric in the two new QTL, the x-axis in the plot corresponds to the first of the new QTL and the y-axis corresponds to the second of the new QTL.
Author(s)
Hao Wu; Karl W Broman, <broman@wisc.edu>; Brian Yandell

See Also
scantwo, summary.scantwo, plot.scanone, -.scantwo

Examples

```r
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)
```

plot.scantwoperm  
Plot permutation results for a 2d, 2-QTL genome scan

Description
Plot a histogram of the permutation results from a two-dimensional, two-QTL genome scan.

Usage
```r
## S3 method for class 'scantwoperm'
plot(x, lodcolumn=1, include_rug=TRUE, ...)
```
The function plots a histogram of the permutation results obtained by `scantwo` when `n.perm` is specified. Separate histograms are provided for the five LOD scores, full, f1, int, add, and av1.

Value

None.

Author(s)

Karl W Broman, <broman@wisc.edu>

See Also

`scantwo, summary.scantwoperm`

Examples

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

operm2 <- scantwo(fake.bc, method="hk", n.perm=10)
plot(operm2)
```
Arguments

- **x**: An object of class `cross`. See `read.cross` for details.
- **chr**: Optional vector indicating the chromosomes to be drawn in the plot. This should be a vector of character strings referring to chromosomes by name; numeric values are converted to strings. Refer to chromosomes with a preceding - to have all chromosomes but those considered. A logical (TRUE/FALSE) vector may also be used.
- **ind**: Indicates the individuals for which the error LOD scores should be plotted (passed to `subset.cross`).
- **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, <broman@wisc.edu>

References


See Also

`calc.errorlod`, `top.errorlod`, `image`, `subset.cross`, `plotGeno`
Examples

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
plotErrorlod(hyper)
plotErrorlod(hyper,chr=1)

plotGeno

Plot observed genotypes, flagging likely errors

Description

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

Usage

plotGeno(x, chr, ind, include.xo=TRUE, horizontal=TRUE,
cutoff=4, min.sep=2, cex=1.2, ...)

Arguments

x An object of class cross. See read.cross for details.
chr The chromosome to plot. Only one chromosome is allowed. (This should be a
character string referring to the chromosomes by name.)
ind Vector of individuals to plot (passed to subset.cross). If missing, all individ-
uals 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.
plotInfo

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 the X chromosome in a backcross or intercross, hemizygous males are plotted as if they were homozygous (that is, with white and black circles).

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, <broman@wisc.edu>

See Also

calc.errorlod, top.errorlod, subset.cross

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
plotGeno(hyper, chr=1, ind=160:200, cutoff=7, min.sep=2)

plotInfo

Plot the proportion of missing genotype information

Description

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

plotInfo(x, chr, method=c("entropy","variance","both"), step=1,
        off.end=0, error.prob=0.001,
        map.function=c("haldane","kosambi","c-f","morgan"),
        alternate.chrid=FALSE, fourwaycross=c("all", "AB", "CD"),
        include.genofreq=FALSE, ...)  

Arguments

- **x**: An object of class cross. See read.cross for details.
- **chr**: Optional vector indicating the chromosomes to plot. This should be a vector of character strings referring to chromosomes by name; numeric values are converted to strings. Refer to chromosomes with a preceding - to have all chromosomes but those considered. A logical (TRUE/FALSE) vector may also be used.
- **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 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(observed 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.
- **fourwaycross**: For a phase-known four-way cross, measure missing genotype information overall ("all"), or just for the alleles from the first parent ("AB") or from the second parent ("CD").
- **include.genofreq**: If TRUE, estimated genotype frequencies (from the results of calc.genoprob averaged across the individuals) are included as additional columns in the output.
- **...**: 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 $H = \sum_{g} p_g \log p_g \log n$, 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 \texttt{plot.scanone}.

Note that \texttt{summary.scanone} may be used to display the maximum missing information on each chromosome.

\textbf{Value}

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

\textbf{Author(s)}

Karl W Broman, <broman@wisc.edu>

\textbf{See Also}

\texttt{plot.scanone, plotMissing, calc.genoprob, geno.table}

\textbf{Examples}

\begin{verbatim}
data(hyper)
plotInfo(hyper, chr=c(1,4))

# save the results and view maximum missing info on each chr
info <- plotInfo(hyper)
summary(info)
plotInfo(hyper, bandcol="gray70")
\end{verbatim}

---

\textbf{plotLodProfile}  \hspace{1cm} \textit{Plot 1-d LOD profiles for a multiple QTL model}

\textbf{Description}

Use the results of \texttt{refineqtl} to plot one-dimensional LOD profiles for each QTL.

\textbf{Usage}

\begin{verbatim}
plotLodProfile(qtl, chr, incl.markers=TRUE, gap=25, lwd=2, lty=1, col="black",
              qtl.labels=TRUE, mtick=c("line", "triangle"),
              show.marker.names=FALSE, alternate.chrid=FALSE,
              add=FALSE, showallchr=FALSE, labelsep=5, ...)\end{verbatim}
Arguments

- **qt1**: An object of class "qt1"; must have been produced by `refineqtl` using `keeplodprofiles=TRUE`.
- **chr**: Optional vector indicating the chromosomes to plot. This should be a vector of character strings referring to chromosomes by name; numeric values are converted to strings. Refer to chromosomes with a preceding - to have all chromosomes but those considered. A logical (TRUE/FALSE) vector may also be used.
- **incl.markers**: Indicate whether to plot line segments at the marker locations.
- **gap**: Gap separating chromosomes (in cM).
- **lwd**: Line widths for each QTL trace (length 1 or the number of QTL).
- **lty**: Line types for each QTL trace (length 1 or the number of QTL).
- **col**: Line col for each QTL trace (length 1 or the number of QTL).
- **qt1.labels**: If TRUE, place a label on each QTL trace.
- **mtick**: Tick mark type for markers (line segments or upward-pointing triangels).
- **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.
- **add**: If TRUE, add curves to a current plot.
- **showallchr**: If FALSE (the default), only show the chr with a QTL.
- **labelsep**: If qt1.labels=TRUE, separation between peak LOD and QTL label, as percent of the height of the plot.
- **...**: Passed to the function `plot` when it is called.

Details

The function plots LOD profiles in the context of a multiple QTL model, using a scheme best described in Zeng et al. (2000). The position of each QTL is varied, keeping all other loci fixed. If a QTL is isolated on a chromosome, the entire chromosome is scanned; if there are additional linked QTL, the position of a QTL is scanned over the largest interval possible without allowing the order of QTLs along a chromosome to change. At each position for the QTL being scanned, we calculate a LOD score comparing the full model, with the QTL of interest at that particular position (and all others at their fixed positions) to the model with the QTL of interest (and any interactions that include that QTL) omitted.

Care should be take regarding the arguments lwd, lty, and col; if vectors are given, they should be in the order of the QTL within the object, which may be different than the order in which they are plotted. (The LOD profiles are sorted by chromosome and position.)

Value

None.
plotMap

Author(s)
Karl W Broman, <broman@wisc.edu>

References

See Also
refineqtl, makeqtl, scanqtl

Examples

data(fake.bc)

fake.bc <- calc.genoprob(fake.bc, step=2)
qt1 <- makeqtl(fake.bc, chr=c(2,5), pos=c(32.5, 17.5), what="prob")

out <- scanone(fake.bc, method="hk")

# refine QTL positions and keep LOD profiles
rqtl <- refineqtl(fake.bc, qt1=qt1, method="hk", keeplodprofile=TRUE)

# plot the LOD profiles
plotLodProfile(rqtl)

# add the initial scan results, for comparison
plot(out, add=TRUE, chr=c(2,5), col="red")

---

plotMap

Plot genetic map

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, ...)
plotMap(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 (and names) of chromosomes. 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**: Optional vector indicating the chromosomes to plot. This should be a vector of character strings referring to chromosomes by name; numeric values are converted to strings. Refer to chromosomes with a preceding - to have all chromosomes but those considered. A logical (TRUE/FALSE) vector may also be used.

- **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.

- **...**: Passed to `plot`.

Details

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

For a comparison of two maps, the first map is on the left (or, if `horizontal=TRUE`, on the top). Lines are drawn to connect markers. Markers that exist in just one map and not the other are indicated by short line segments, on one side or the other, that are not connected across.

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, <broman@wisc.edu>

See Also

- `est.map`
- `plot.cross`
**plotMissing**

*Plot grid of missing genotypes*

**Examples**

```r
data(fake.bc)

plotMap(fake.bc)
plotMap(fake.bc, horizontal=TRUE)

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

plotMap(fake.bc, show.marker.names=TRUE)
```

**Description**

Plot a grid showing which genotypes are missing.

**Usage**

```r
plotMissing(x, chr, reorder=FALSE, main="Missing genotypes", alternate.chrid=FALSE, ...)
```

**Arguments**

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

- **chr**
  Optional vector indicating the chromosomes to plot. This should be a vector of character strings referring to chromosomes by name; numeric values are converted to strings. Refer to chromosomes with a preceding `-` to have all chromosomes but those considered. A logical (TRUE/FALSE) vector may also be used.

- **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, <broman@wisc.edu>

See Also

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

Examples

```r
data(fake.f2)
plotMissing(fake.f2)
```

---

**plotModel**  
*Plot a QTL model*

Description

Plot a graphical representation of a QTL model, with nodes representing QTL and line segments representing pairwise interactions.

Usage

```r
plotModel(qtl, formula, circrad.rel=0.25, circrad.abs,  
          cex.name=1, chronly=FALSE, order, ...)
```

Arguments

- `qtl`  
  A QTL object (as created by `makeqtl`) or vector of character strings indicating the names for the QTL. This is also allowed to be a list that contains a component named "chr" (and, optionally, components named "pos" and "formula").

- `formula`  
  Optional formula defining the QTL model. If missing, we look for an attribute "formula" to the input QTL object or a item named "formula" within the QTL object.

- `circrad.rel`  
  Radius of the circles that indicate the QTL, relative to the distance between the circles.

- `circrad.abs`  
  Optional radius of the circles that indicate the QTL; note that the plotting region will have x- and y-axis limits spanning 3 units.
plotModel

- cex.name: Character expansion for the QTL names.
- chronly: If TRUE and a formal QTL object is given, only the chromosome IDs are used to identify the QTL.
- order: Optional vector indicating a permutation of the QTL to define where they are to appear in the plot. QTL are placed around a circle, starting at the top and going clockwise.
- ...: Passed to the function `plot`.

Value
None.

Author(s)
Karl W Broman, <broman@wisc.edu>

See Also
stepwiseqtl, makeqtl

Examples

# plot a QTL model, using a vector of character strings to define the QTL
plotModel(c("1","4","6","15"), formula=y~Q1+Q2+Q3*Q4)

# plot an additive QTL model
hyper <- calc.genoprob(hyper)
qtl <- makeqtl(hyper, chr=c(1,4,6,15), pos=c(68.3,30,60,18), what="prob")
plotModel(qtl)

# include an interaction
plotModel(qtl, formula=y~Q1+Q2+Q3*Q4)

# alternatively, include the formula as an attribute to the QTL object
attr(qtl, "formula") <- y~Q1+Q2+Q3*Q4
plotModel(qtl)

# if formula given, the attribute within the object is ignored
plotModel(qtl, y~Q1+Q2+Q3+Q4)

# NULL formula indicates additive QTL model
plotModel(qtl, NULL)

# reorder the QTL in the figure
plotModel(qtl, order=c(1,3,4,2))

# show just the chromosome numbers
plotModel(qtl, chronly=TRUE)
plotPheno

Plot a phenotype distribution

Description

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

Usage

plotPheno(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. Alternatively, one may give a numeric vector of phenotypes, in which case it must have the length equal to the number of individuals in the cross, and there must be either non-integers or values < 1 or > no. phenotypes; this last case may be useful for studying transformations.

...
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, <broman@wisc.edu>

See Also

plot.cross, plotMap, plotMissing, hist, barplot

Examples

data(fake.bc)
plotPheno(fake.bc, pheno.col=1)
plotPheno(fake.bc, pheno.col=3)
plotPheno(fake.bc, pheno.col="age")
**plotPXG**

**Plot phenotypes versus marker genotypes**

**Description**
Plot the phenotype values versus the genotypes at a marker or markers.

**Usage**

```r
plotPXG(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. One may also give a character string matching a phenotype name. Finally, one may give a numeric vector of phenotypes, in which case it must have the length equal to the number of individuals in the cross, and there must be either non-integers or values < 1 or > no. phenotypes; this last case may be useful for studying transformations.
- `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).
- `...`: Passed to `plot`.

**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, <broman@wisc.edu>; Brian Yandell

See Also

find.marker, effectplot, find.flanking, effectscan

Examples

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

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

# output of the function contains the raw data
output <- plotPXG(listeria, mname)
head(output)

# another example
data(fake.f2)
mname <- find.marker(fake.f2, 1, 37) # marker D1M437
plotPXG(fake.f2, mname)

mname2 <- find.marker(fake.f2, "X", 14) # marker DXM66
plotPXG(fake.f2, mname2)

plotPXG(fake.f2, c(mname,mname2))
plotPXG(fake.f2, c(mname2,mname))

plotRF

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

plotRF(x, chr, what=c("both","lod","rf"), alternate.chrid=FALSE,
zmax=12, mark.diagonal=FALSE,
col.scheme=c("viridis", "redblue"), ...)
Arguments

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

chr Optional vector indicating the chromosomes to plot. This should be a vector of character strings referring to chromosomes by name; numeric values are converted to strings. Refer to chromosomes with a preceding - to have all chromosomes but those considered. A logical (TRUE/FALSE) vector may also be used.

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.

zmax Maximum LOD score plotted; values above this are all thresholded at this value.

mark.diagonal If TRUE, include black line segments around the pixels along the diagonal, to better separate the upper left triangle from the lower right triangle.

col.scheme The color palette. The default is "viridis"; see Option D at https://bids.github.io/colormap

... Generally ignored, but you can include main to change or omit the title of the figure.

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.

With col.scheme="viridis" (the default), purple corresponds to a large LOD score or a small recombination fraction, while yellow is the reverse. With col.scheme="redblue", 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.

Author(s)

Karl W Broman, <broman@wisc.edu>

See Also

est.rf, pull.rf, plot.rfmatrix, image, badorder, ripple
Examples

```r
data(badorder)
badorder <- est.rf(badorder)
plotRF(badorder)

# plot just chr 1
plotRF(badorder, chr=1)

# plot just the recombination fractions
plotRF(badorder, what="rf")

# plot just the LOD scores, and just for chr 2 and 3
plotRF(badorder, chr=2:3, what="lod")
```

---

**pull.argmaxgeno**

Pull out the results of the Viterbi algorithm from a cross

---

**Description**

Pull out the results of `argmax.geno` from a cross as a matrix.

**Usage**

```r
pull.argmaxgeno(cross, chr, include.pos.info=FALSE, rotate=FALSE)
```

**Arguments**

- `cross` An object of class `cross`. See `read.cross` for details.
- `chr` Optional vector indicating the chromosomes to consider. This should be a vector of character strings referring to chromosomes by name; numeric values are converted to strings. Refer to chromosomes with a preceding - to have all chromosomes but those considered. A logical (TRUE/FALSE) vector may also be used.
- `include.pos.info` If TRUE, include columns with marker name, chromosmoe ID, and cM position. (If `include.pos.info=TRUE`, we take `rotate=TRUE`.)
- `rotate` If TRUE, return matrix with individuals as columns and positions as rows. If FALSE, rows correspond to individuals.

**Value**

A matrix containing numeric indicators of the inferred genotypes. Multiple chromosomes are pasted together.

**Author(s)**

Karl W Broman, <broman@wisc.edu>
pull.draws

See Also

pull.geno, pull.genoprob, pull.draws, argmax.geno

Examples

data(listeria)
listeria <- argmax.geno(listeria, step=1, stepwidth="max")
amg <- pull.argmaxgeno(listeria, chr=c(5,13), include.pos.info=TRUE, rotate=TRUE)
amg[1:5,1:10]

data(listeria)
listeria <- sim.geno(listeria, step=5, stepwidth="max", n.draws=8)
dr <- pull.draws(listeria, chr=c(5,13))
dr[1:20,1:10,1]
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 Optional vector indicating the chromosomes to consider. This should be a vector of character strings referring to chromosomes by name; numeric values are converted to strings. Refer to chromosomes with a preceding - to have all chromosomes but those considered. A logical (TRUE/FALSE) vector may also be used.

Value

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

Author(s)

Karl W Broman, <broman@wisc.edu>

See Also

pull.pheno, pull.map pull.draws, pull.genoprob, pull.argmaxgeno

Examples

data(listeria)
dat <- pull.geno(listeria)

# image of the genotype data
image(1:ncol(dat),1:nrow(dat),t(dat),ylab="Individuals",xlab="Markers",
     col=c("red","yellow","blue","green","violet"))
abline(v=cumsum(c(0,nmar(listeria)))+0.5)
abline(h=nrow(dat)+0.5)
pull.genoprob

Pull out the genotype probabilities from a cross

Description

Pull out the results of `calc.genoprob` from a cross as a matrix.

Usage

```r
pull.genoprob(cross, chr, omit.first.prob=FALSE,
              include.pos.info=FALSE, rotate=FALSE)
```

Arguments

- **cross**: An object of class `cross`. See `read.cross` for details.
- **chr**: Optional vector indicating the chromosomes to consider. This should be a vector of character strings referring to chromosomes by name; numeric values are converted to strings. Refer to chromosomes with a preceding `-` to have all chromosomes but those considered. A logical (TRUE/FALSE) vector may also be used.
- **omit.first.prob**: If TRUE, omit the probabilities for the first genotype at each position (since they sum to 1).
- **include.pos.info**: If TRUE, include columns with marker name, genotype, chromosome ID, and cM position. (If `include.pos.info=TRUE`, we take `rotate=TRUE`.)
- **rotate**: If TRUE, return matrix with individuals as columns and positions/genotypes as rows. If FALSE, rows correspond to individuals.

Value

A matrix containing genotype probabilities. Multiple chromosomes and the multiple genotypes at each position are pasted together.

Author(s)

Karl W Broman, <broman@wisc.edu>

See Also

`pull.geno`, `pull.argmaxgeno`, `pull.draws`, `calc.genoprob`

Examples

```r
data(listeria)
listeria <- calc.genoprob(listeria, step=1, stepwidth="max")
pr <- pull.genoprob(listeria, chr=c(5,13), omit.first.prob=TRUE, include.pos.info=TRUE, rotate=TRUE)
pr[1:5,1:10]
```
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, as.table=FALSE)
```

Arguments

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

chr Optional vector indicating the chromosomes to consider. This should be a vector of character strings referring to chromosomes by name; numeric values are converted to strings. Refer to chromosomes with a preceding - to have all chromosomes but those considered. A logical (TRUE/FALSE) vector may also be used.

as.table If TRUE, return the genetic map as a table with chromosome assignments and marker names. If FALSE, return the map as a "map" object.

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, <broman@wisc.edu>

See Also

```
replace.map, plotMap, map2table
```

Examples

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

*Drop all but a selected set of markers*

**Description**

Drop all but a selected set of markers from the data matrices and genetic maps.

**Usage**

```r
pull.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 not specified 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, <broman@wisc.edu>

**See Also**

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

**Examples**

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

---

**pull.pheno**

*Pull out phenotype data from a cross*

**Description**

Pull out selected phenotype data from a cross object, as a data frame or vector.

**Usage**

```r
pull.pheno(cross, pheno.col)
```
Arguments

cross       An object of class cross. See read.cross for details.
pheno.col   A vector specifying which phenotypes to keep or discard. This may be a logical vector, a numeric vector, or a vector of character strings (for the phenotype names). If missing, the entire set of phenotypes is output.

Value

A data.frame with columns specifying phenotypes and rows specifying individuals. If there is just one phenotype, a vector (rather than a data.frame) is returned.

Author(s)

Karl W Broman, <broman@wisc.edu>

See Also

pull.geno, pull.map

Examples

data(listeria)
pull.pheno(listeria, "sex")

Description

Pull out either the pairwise recombination fractions or the LOD scores, as calculated by est.rf, from a cross object.

Usage

pull.rf(cross, what=c("rf", "lod"), chr)

Arguments

cross       An object of class cross. See read.cross for details.
what        Indicates whether to pull out a matrix of estimated recombination fractions or a matrix of LOD scores.
chr         Optional vector indicating the chromosomes to consider. This should be a vector of character strings referring to chromosomes by name; numeric values are converted to strings. Refer to chromosomes with a preceding - to have all chromosomes but those considered. A logical (TRUE/FALSE) vector may also be used.
Value

An object of class "rfmatrix", which is a matrix of either estimated recombination fractions between all marker pairs or of LOD scores (for the test of rf=1/2) for all marker pairs. The genetic map is included as an attribute.

Author(s)

Karl W Broman, <broman@wisc.edu>

See Also

est.rf, plot.rfmatrix, plotRF

Examples

data(fake.f2)

fake.f2 <- est.rf(fake.f2)
rf <- pull.rf(fake.f2)
lod <- pull.rf(fake.f2, "lod")
plot(rf[,1], lod[,1], xlab="rec frac", ylab="LOD score")
marker <- markernames(fake.f2, chr=5)[6]
par(mfrow=c(2,1))
plot(rf, marker, bandcol="gray70")
plot(lod, marker, bandcol="gray70")

qtlversion

Installed version of R/qtl

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, <broman@wisc.edu>

Examples

qtlversion()
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", "mapqtl", "tidy"),
  dir="", file, genfile, mapfile, phefile, chrfile,
  mnamesfile, pnamesfile, na.strings=c("-","NA"),
  genotypes=c("A","H","B","D","C"), alleles=c("A","B"),
  estimate.map=FALSE, convertXdata=TRUE, error.prob=0.0001,
  map.function=c("haldane","kosambi","c-f","morgan"),
  BC.gen=0, F.gen=0, crosstype, ...)
```

**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, gary and mapqtl only).
- **mapfile** File with marker position information (all except the csv formats).
- **phefile** File with phenotype data (formats csvs, csvsr, karl, gary and mapqtl only).
- **chrfile** 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.
read.cross

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, mapqtl, 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.

error.prob In the case that the marker map must be estimated: Assumed genotyping error rate used in the calculation of the penetrance Pr(observed genotype | true genotype).

map.function In the case that the marker map must be estimated: Indicates whether to use the Haldane, Kosambi, Carter-Falconer, or Morgan map function when converting genetic distances into recombination fractions. (Ignored if m > 0.)

BC.gen Used only for cross type "bcsft".

F.gen Used only for cross type "bcsft".

crosstype Optional character string to force a particular cross type.

... 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), dec to specify the character used for the decimal point (the default is a period), and comment.char to specify a character to indicate comment lines.

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), Karl Broman’s format (karl) and MapQTL/JoinMap (mapqtl). 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 multiple formats. Also see https://rqtl.org/sampledata.

The ... argument enables additional arguments to be 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), dec to specify the character used for the decimal point (the default is a period), and comment.char to specify a character to indicate comment lines.

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 gets converted into numeric codes, as follows.

The genotype data for a backcross is coded as 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, 11 = not AC, 12 = not BC, 13 = not AD, 14 = not BD.

The pheno data.frame of size (n.ind x n.phe) containing the phenotypes. If a phenotype with the name id or ID is included, these identifiers will be used in top.errorlod, plotErrorlod, and plotGeno as identifiers for the individual.

While the data format is complicated, there are a number of functions, such as subset.cross, to assist in pulling out portions of the data.

X chromosome

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 so 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 genotype codes must be the same across all markers. For example, you can’t have one marker coded AA/AB/BB and another coded A/H/B. This includes genotypes for the X chromosome, for which hemizygous individuals should be coded as if they were homozygous.

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 with the same name). 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, plotErrorlod, and plotGeno as identifiers for the individual.

The first row in each file contains the column names. For the phenotype file, these are the names of the phenotypes. For the genotype file, the first cell will be the name of the identifier column (id or ID) and the subsequent fields will be the marker names.

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.

Tidy format

This format requires three simple CSV files, separating the genotype, phenotype, and marker map information so that each file may be of a simple form.

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:

```
nen.chr
nnen.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.

MapQTL format

This format requires three files, described in the manual of the MapQTL program (same as JoinMap).

genfile corresponds to the loc file containing the genotype data. Each marker and its genotypes should be on a single line.

mapfile corresponds to the map file containing the linkage group assignment, marker names and their map positions.

phefile corresponds to the qua file containing the phenotypes.

For the moment, only 4-way crosses are supported (CP population type in MapQTL).
Author(s)
Karl W Broman, <broman@wisc.edu>; Brian S. Yandell; Aaron Wolen

References

See Also
subset.cross, summary.cross, plot.cross, c.cross, clean.cross, write.cross, sim.cross, read.table. The sampledata directory in the package distribution contains sample data files in multiple formats. Also see https://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")

# you can read files directly from the internet
datweb <- read.cross("csv", "https://rqtl.org/sampledata", "listeria.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="chr.dat", mnamesfile="mnames.txt", pnamesfile="pnames.txt")

# Karl format
dat7 <- read.cross("karl", dir="Mydata", genfile="gen.txt", phefile="phe.txt", mapfile="map.txt")

## End(Not run)
readMWril  

Read data for 4- or 8-way RIL

Description

Data for a set of 4- or 8-way recombinant inbred lines (RIL) is read from a pair of comma-delimited files and converted into an object of class cross. We require chromosome assignments for the genetic markers, and assume that markers are in their correct order.

Usage

readMWril(dir='', rilfile, founderfile, 
  type=c("ri4self", "ri4sib", "ri8self", "ri8sibIRIP1", "ri8sib", "bgmagic16"), 
  na.strings=c("-","NA"), rotate=FALSE, ...)

Arguments

dir Directory in which the data files will be found. In Windows, use forward slashes ("/") or double backslashes ("\") to specify directory trees.

rilfile Comma-delimited file for the RIL, in the "csv" format described in the help file for read.cross.

founderfile File with founder strains' genotypes, in the same orientation as the rilfile, but with just marker names and the founders' marker genotypes.

type The type of RIL.

na.strings A vector of strings which are to be interpreted as missing values. 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.

rotate If TRUE, the rilfile and founderfile are rotated (really transposed), with rows corresponding to markers and columns corresponding to individuals.

... 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 rilfile should include a phenotype cross containing character strings of the form ABCDEFGH, indicating the cross used to generate each RIL. The genotypes should be coded as integers (e.g., 1 and 2).

The founder strains in the founderfile should be the strains A, B, C, ..., as indicated in the cross phenotype.

The default arrangement of the files is to have markers as columns and individuals/founders as rows. If rotate=TRUE, do the opposite: markers as rows and individuals/founders as columns.
reduce2grid

Reduce to a grid of pseudomarkers.

Description

For high-density marker data, rather than run scanone at both the markers and at a set of pseudomarkers, we reduce to just a set of evenly-spaced pseudomarkers.

Usage

reduce2grid(cross)

Arguments

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

Details

Genotype probabilities (from calc.genoprob) and/or imputations (from sim.geno) are subset to a grid of pseudomarkers.

This is so that, in the case of high-density markers, we can do the genome scan calculations at a smaller set of points (on an evenly-spaced grid, but not at the markers) to save computation time.

You need to first have run calc.genoprob and/or sim.geno, and you must use stepwidth="fixed".

When plotting results with plot.scanone, use incl.markers=FALSE, as the output of scanone won’t include information about the marker locations and so will plot tick marks only at the first marker on each chromosome.
refineqtl

Value

The input cross object with included genotype probabilities or imputations subset to an evenly-spaced grid.

Author(s)

Karl W Broman, <broman@wisc.edu>

See Also

calc.genoprob, sim.geno, scanone, plot.scanone

Examples

data(hyper)
hyper <- calc.genoprob(hyper, step=2)
hypersub <- reduce2grid(hyper)

## Not run: out <- scanone(hypersub)
plot(out, incl.markers=FALSE)
## End(Not run)

refineqtl

Refine the positions of QTL

Description

Iteratively scan the positions for QTL in the context of a multiple QTL model, to try to identify the positions with maximum likelihood, for a fixed QTL model.

Usage

refineqtl(cross, pheno.col=1, qtl, chr, pos, qtl.name, covar=NULL, formula, method=c("imp","hk"), model=c("normal","binary"), verbose=TRUE, maxit=10, incl.markers=TRUE, keeplodprofile=TRUE, tol=1e-4, maxit.fitqtl=1000, forceXcovar=FALSE)

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. One may also give a character string matching the phenotype name. Finally, one may give a numeric vector of phenotypes, in which case it must have the length equal to the number of individuals in the cross, and there must be either non-integers or values < 1 or > no. phenotypes; this last case may be useful for studying transformations.
qt1  A QTL object, as produced by `makeqtl`, containing the positions of the QTL. Provide either `qt1` or the pair `chr` and `pos`.

`chr`  Vector indicating the chromosome for each QTL; if `qt1` is provided, this should not be.

`pos`  Vector indicating the positions for each QTL; if `qt1` is provided, this should not be.

`qt1.name`  Optional user-specified name for each QTL. If `qt1` is provided, this should not be.

`covar`  A matrix or data.frame of covariates. These must be strictly numeric.

`formula`  An object of class `formula` indicating the model to be fitted. (It can also be the character string representation of a formula.) QTLs are indicated as `Q1`, `Q2`, etc. Covariates are indicated by their names in `covar`.

`method`  Indicates whether to use multiple imputation or Haley-Knott regression.

`model`  The phenotype model: the usual model or a model for binary traits

`verbose`  If TRUE, give feedback about progress. If `verbose` is an integer > 1, further messages from `scanqtl` are also displayed.

`maxit`  Maximum number of iterations.

`incl.markers`  If FALSE, do calculations only at points on an evenly spaced grid.

`keeplodprofile`  If TRUE, keep the LOD profiles from the last iteration as attributes to the output.

`tol`  Tolerance for convergence for the binary trait model.

`maxit.fitqtl`  Maximum number of iterations for fitting the binary trait model.

`forceXcovar`  If TRUE, force inclusion of X-chr-related covariates (like sex and cross direction).

**Details**

QTL positions are optimized, within the context of a fixed QTL model, by a scheme described in Zeng et al. (1999). Each QTL is considered one at a time (in a random order), and a scan is performed, allowing the QTL to vary across its chromosome, keeping the positions of all other QTL fixed. If there is another QTL on the chromosome, the position of the floating QTL is scanned from the end of the chromosome to the position of the flanking QTL. If the floating QTL is between two QTL on a chromosome, its position is scanned between those two QTL positions. Each QTL is moved to the position giving the highest likelihood, and the entire process is repeated until no further improvement in likelihood can be obtained.

One may provide either a `qt1` object (as produced by `makeqtl`), or vectors `chr` and `pos` (and, optionally, `qt1.name`) indicating the positions of the QTL.

If a `qt1` object is provided, QTL that do not appear in the model `formula` are ignored, but they remain part of the QTL object that is output.

**Value**

An object of class `qt1`, with QTL placed in their new positions.

If `keeplodprofile=TRUE`, LOD profiles from the last pass through the refinement algorithm are retained as an attribute, "lodprofile", to the object. These may be plotted with `plotLodProfile`. 
reorderqtl

Author(s)
Karl W Broman, <broman@wisc.edu>

References

See Also
fitqtl, makeqtl, scanqtl, addtoqtl, dropfromqtl, replaceqtl, plotLodProfile

Examples

data(fake.bc)

fake.bc <- calc.genoprob(fake.bc, step=2)
qtl <- makeqtl(fake.bc, chr=c(2,5), pos=c(32.5, 17.5), what="prob")
rqtl <- refineqtl(fake.bc, qtl=qtl, method="hk")

reorderqtl

Reorder the QTL in a qtl object

Description
This function changes the order of the QTL in a QTL object.

Usage
reorderqtl(qtl, neworder)

Arguments

qtl A qtl object, as created by makeqtl.
neworder A vector containing the positive integers up to the number of QTL in qtl, indicating the new order for the QTL. If missing, the QTL are ordered by chromosome and then by their position within a chromosome.

Details
Everything in the input qtl is reordered except the altname component, which contains names of the form Q1, Q2, etc.
Value

The input qtl object, with the loci reordered.

Author(s)

Karl W Broman, <broman@wisc.edu>

See Also

makeqtl, fitqtl, dropfromqtl, addtoqtl, replaceqtl

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 <- calc.genoprob(fake.f2)
qtl <- makeqtl(fake.f2, qc, qp, what="prob")

qtl <- reorderqtl(qtl, c(2,3,1))
qtl

qtl <- reorderqtl(qtl)
qtl

replace.map

Replace the genetic map of a cross

Description

Replace the map portion of a cross object.

Usage

replace.map(cross, map)

## S3 method for class 'cross'
replacemap(object, map)

Arguments

cross An object of class cross. See read.cross for details.
object Same as cross.
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. Maps for results from calc.genoprob, sim.geno and argmax.geno are also replaced, using interpolation if necessary.

Author(s)
Karl W Broman, <broman@wisc.edu>

See Also
pull.map, est.map

Examples

```r
data(fake.f2)
newmap <- est.map(fake.f2)
plotMap(fake.f2, newmap)
fake.f2 <- replace.map(fake.f2, newmap)
```

Description
Replace the positions of LOD scores in output from scanone with values based on an alternative map (such as a physical map), with pseudomarker locations determined by linear interpolation.

Usage
```r
## S3 method for class 'scanone'
replacemap(object, map)
```

Arguments
- **object**: An object of class "scanone", as output by the function scanone.
- **map**: A list containing the alternative genetic map. All chromosomes in object should have corresponding chromosomes in map, and markers must be in the same order in the two maps. There must be at least two markers on each chromosome in map that appear in object.

Details
The positions of pseudomarkers are determined by linear interpolation between markers. In the case of pseudomarkers beyond the ends of the terminal markers on chromosomes, we use the overall lengths of the chromosome in object and map to determine the new spacing.
Value

The input object with the positions of LOD scores revised to match those in the input map.

Author(s)

Karl W Broman, <broman@wisc.edu>

See Also

replacemap.cross, est.map, replacemap.scantwo

Examples

data(fake.f2)
origmap <- pull.map(fake.f2)
newmap <- est.map(fake.f2)
fake.f2 <- replacemap(fake.f2, newmap)
fake.f2 <- calc.genoprob(fake.f2, step=2.5)
out <- scanone(fake.f2, method="hk")
out.rev <- replacemap(out, origmap)

---

replacemap.scantwo  Replace the genetic map in QTL mapping results with an alternate map

Description

Replace the positions of LOD scores in output from scantwo with values based on an alternative map (such as a physical map), with pseudomarker locations determined by linear interpolation.

Usage

## S3 method for class 'scantwo'
replacemap(object, map)

Arguments

object  An object of class "scantwo", as output by the function scantwo.
map  A list containing the alternative genetic map. All chromosomes in object should have corresponding chromosomes in map, and markers must be in the same order in the two maps. There must be at least two markers on each chromosome in map that appear in object.

Details

The positions of pseudomarkers are determined by linear interpolation between markers. In the case of pseudomarkers beyond the ends of the terminal markers on chromosomes, we use the overall lengths of the chromosome in object and map to determine the new spacing.
Value

The input object with the positions of LOD scores revised to match those in the input map.

Author(s)

Karl W Broman, <broman@wisc.edu>

See Also

replacemap.cross, est.map, replacemap.scanone

Examples

data(hyper)
origmap <- pull.map(hyper)
newmap <- est.map(hyper)
hyper <- replacemap(hyper, newmap)
hyper <- calc.genoprob(hyper, step=0)
out <- scantwo(hyper, method="hk")
out.rev <- replacemap(out, origmap)

replaceqtl

Replace a QTL in a qtl object with a different position

Description

This function replaces a QTL or QTLs in a qtl object with a different position.

Usage

replaceqtl(cross, qtl, index, chr, pos, qtl.name, drop.lod.profile=TRUE)

Arguments

cross An object of class cross. See read.cross for details.
qtl A qtl object, as created by makeqtl.
index Numeric index indicating the QTL to be replaced.
chr Vector (of same length as index) indicating the chromosomes for the new QTL.
pos Vector (of same length as index) indicating the positions for the new QTL. If there is no marker or pseudomarker at a position, the nearest position is used.
qtl.name Optional vector (of same length as index) of user-specified names for each new 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 Chromsome 1 at 10 cM.
drop.lod.profile If TRUE, remove any LOD profiles from the object.
Value

The input qtl object, but with some QTL replaced by new ones. See makeqtl for details on the format.

Author(s)

Karl W Broman, <broman@wisc.edu>

See Also

makeqtl, fitqtl, dropfromqtl, addtoqtl, reorderqtl

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 <- calc.genoprob(fake.f2, step=2, err=0.001)
qtl <- makeqtl(fake.f2, qc, qp, what="prob")
qtl <- replaceqtl(fake.f2, qtl, 2, 6, 48.1)

rescalemap

Rescale genetic maps

Description

Rescale a genetic map by multiplying all positions by a constant

Usage

rescalemap(object, scale=1e-6)

Arguments

object An object of class cross (see read.cross for details) or map (see sim.map for details).
scale Scale factor by which all positions will be multiplied.

Details

This function is included particularly for the case that map positions in a cross object were provided in basepairs and one wishes to quickly convert them to Mbp or some other approximation of cM distances. (In the mouse, 1 cM is approximation 2 Mbp, so one might use scale=5e-7 in this function.)
Value

If the input is a map object, a map object is returned; if the input is a cross object, a cross object is returned. In either case, the positions of markers are simply multiplied by scale.

Author(s)

Karl W Broman, <broman@wisc.edu>

See Also

replace.map, est.map

Examples

data(hyper)
rescaled <- rescalemap(hyper, scale=2)
plotMap(hyper, rescaled)

ripple Compare marker orders

Description

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

Usage

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-6, sex.sp=TRUE, verbose=TRUE, n.cluster=1)

Arguments

cross An object of class cross. See read.cross for details.
chr The chromosome to investigate. Only one chromosome is allowed. (This should be a character string referring to the chromosomes by name.)
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 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.
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.
n.cluster
If the package snow is available and n.perm > 0, permutations are run in parallel using this number of nodes. This is really only useful with method = "likelihood".

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, <broman@wisc.edu>

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

Examples

data(badorder)
rip1 <- ripple(badorder, chr=1, window=3)
summary(rip1)

## Not run:
rip2 <- ripple(badorder, chr=1, window=2, method="likelihood")
summary(rip2)

## End(Not run)
badorder <- switch.order(badorder, 1, rip1[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,
batchsize=250, n.cluster=1, ind.noqtl)

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. This should be a vector of character strings referring to chromosomes by name; numeric values are converted to strings. Refer to chromosomes with a preceding - to have all chromosomes but those considered. A logical (TRUE/FALSE) vector may also be used.
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. One may also give a character strings matching the phenotype names. Finally, one may give a numeric vector of phenotypes, in which case it must have the length equal to the number of individuals in the cross, and there must be either non-integers or values < 1 or > no. phenotypes; this last case may be useful for studying transformations.
model The phenotype 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 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.

batchsize
The number of phenotypes (or permutations) to be run as a batch; used only for methods "hk" and "imp".

n.cluster
If the package snow is available and n.perm > 0, permutations are run in parallel using this number of nodes.

ind.noqtl
Indicates individuals who should not be allowed a QTL effect (used rarely, if at all); this is a logical vector of same length as there are individuals in the cross.

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

Individuals with missing phenotypes are dropped.

In the case that n.perm > 0, so that a permutation test is performed, the R function scanone is called repeatedly. If 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, n.perm specifies the number of permutations used for the autosomes; for the X chromosome, n.perm
\( \times \frac{L_A}{L_X} \) 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 \( n \text{. perm} \) 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 "scanone" and has attributes "model", "method", and "type" (the latter is the type of cross analyzed).

If \( n \text{. perm} \) is specified, the function returns the results of a permutation test and the output has class "scanoneperm". If \( \text{perm.Xsp}=\text{FALSE} \), the function returns a matrix with \( n \text{. perm} \) 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 \( \text{perm.Xsp}=\text{TRUE} \), 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 **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 **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 em, hk, and mr are available for this model. See Xu and Atchley (1996) and Broman (2003).

The **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 \( \text{ties.random} \) is \text{TRUE}, ties in the phenotypes are assigned random ranks; if it is \text{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. (2006).
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>both sexes</td>
<td>sex</td>
<td>AA/AB/AY/BY</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>all female</td>
<td>grand mean</td>
<td>AA/AB</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>all male</td>
<td>grand mean</td>
<td>AY/BY</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>F2 Direction</td>
<td>Sexes</td>
<td>Null</td>
<td>Alternative</td>
<td>df</td>
</tr>
<tr>
<td>Both</td>
<td>sex</td>
<td>AA/ABf/ABr/BB/AY/BY</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>all female</td>
<td>pgm</td>
<td>AA/ABf/ABr/BB</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>all male</td>
<td>grand mean</td>
<td>AY/BY</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Forward</td>
<td>sex</td>
<td>AA/AB/AY/BY</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>all female</td>
<td>grand mean</td>
<td>AA/AB</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>all male</td>
<td>grand mean</td>
<td>AY/BY</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Backward</td>
<td>sex</td>
<td>AB/BB/AY/BY</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>all female</td>
<td>grand mean</td>
<td>AB/BB</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>all male</td>
<td>grand mean</td>
<td>AY/BY</td>
<td>1</td>
<td></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. Autosome- 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, <broman@wisc.edu>; Hao Wu

References


See Also

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

###################
# Normal Model
###################
data(hyper)

# Genotype probabilities for EM and H-K
## Not run: 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
## Not run: hyper.c1n4 <- sim.geno(subset(hyper, chr=c(1,4)),
## step=2.5, n.draws=8)

## End(Not run)
out.imp <- scanone(hyper.c1n4, 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)

# 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)
# scan with square-root of phenotype
# (Note that pheno.col can be a vector of phenotype values)
out.sqrt <- scanone(hyper, pheno.col=sqrt(pull.pheno(hyper, 1)))
plot(out.em - out.sqrt, ylim=c(-0.1,0.1),
     ylab="Difference in LOD")
abline(h=0, lty=2, col="gray")

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

# X-specific permutations
# Not run: 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)
# 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)
## Not run: 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
##########################
binphe <- as.numeric(pull.pheno(listeria,1)==264)
out.bin <- scanone(listeria, pheno.col=binphe, model="binary")
summary(out.bin, thr=3)
# Plot LOD for binary model with LOD(p) from 2-part model
plot(out.bin, out.2p, lodcolumn=c(1,2), lty=1, col=c("black","red"),
     chr=c(1,5,13))

# Permutation test
## Not run: permo <- scanone(listeria, pheno.col=binphe, model="binary",
## n.perm=1000)
## End(Not run)

# Thresholds
summary(permo)

###########################
# Covariates
###########################
data(fake.bc)
## Not run: fake.bc <- calc.genoprob(fake.bc, step=2.5)

# genome scans without covariates
out.nocovar <- scanone(fake.bc)

# genome scans with covariates
ac <- pull.pheno(fake.bc, c("sex","age"))
ic <- pull.pheno(fake.bc, "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**: The chromosome to investigate. Only one chromosome is allowed. (This should be a character string referring to the chromosomes by name.)
- **pheno.col**: Column number in the phenotype matrix which should be used as the phenotype. One may also give a character string matching a phenotype name. Finally, one may give a numeric vector of phenotypes, in which case it must have the length equal to the number of individuals in the cross, and there must be either non-integers or values < 1 or > no. phenotypes; this last case may be useful for studying transformations.
- **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.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".
scanonevar

Author(s)

Karl W Broman, <broman@wisc.edu>

References


See Also

scanone, summary.scanoneboot, plot.scanoneboot, lodint, bayesint

Examples

data(fake.f2)
fake.f2 <- calc.genoprob(fake.f2, step=1, err=0.001)
## Not run: bootoutput <- scanoneboot(fake.f2, chr=13, method="hk")

plot(bootoutput)
summary(bootoutput)


gene scan for QTL affecting mean and/or variance

Description

Genome scan with a single QTL model for loci that can affect the variance as well as the mean.

Usage

scanonevar(cross, pheno.col=1, mean_covar=NULL, var_covar=NULL,
maxit=25, tol=1e-6, quiet=TRUE)

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. This must be a single value (integer index or phenotype name) or a numeric vector of phenotype values, in which case it must have the length equal to the number of individuals in the cross, and there must be either non-integers or values < 1 or > no. phenotypes; this last case may be useful for studying transformations.
mean_covar Numeric matrix with covariates affecting the mean.
var_covar Numeric matrix with covariates affecting the variances.
maxit  Maximum number of iterations in the algorithm to fit the model at a given position.

tol  Tolerance for convergence.

quiet If FALSE, print some information about the course of the calculations.

Value

A data frame (with class "scanone", in the form output by scanone), with four columns: chromosome, position, the -log P-value for the mean effect, and the -log P-value for the effect on the variance. The result is given class "scanone"

Author(s)

Lars Ronnegard and Karl Broman

References


See Also

scanone, summary.scanone, calc.genoprob, summary.scanoneperm

Examples

data(fake.bc)

fake.bc <- calc.genoprob(fake.bc, step=2.5)
out <- scanonevar(fake.bc)
color <- c("slateblue", "violetred")
plot(out, lod=1:2, col=color, bandcol="gray80")
legend("topright", lwd=2, col=color)

# use format="allpeaks" to get summary for each of mean and variance
# also consider format="tabByCol" or format="tabByChr"
summary(out, format="allpeaks")

# with sex and age as covariates
covar <- fake.bc$pheno[,c("sex", "age")]
out.cov <- scanonevar(fake.bc, mean_covar=covar, var_covar=covar)
Permutation test for mean effect in scanonevar

Description

Executes permutations of the genotypes in the mean-effect part of scanonevar

Usage

`scanonevar.meanperm(cross, pheno.col=1, mean_covar=NULL, var_covar=NULL, maxit=25, tol=1e-6, n.mean.perm = 2, seed = 27517, quiet=TRUE)`

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. This must be a single value (integer index or phenotype name) or a numeric vector of phenotype values, in which case it must have the length equal to the number of individuals in the cross, and there must be either non-integers or values < 1 or > no. phenotypes; this last case may be useful for studying transformations.

mean_covar: Numeric matrix with covariates affecting the mean.

var_covar: Numeric matrix with covariates affecting the variances.

maxit: Maximum number of iterations in the algorithm to fit the model at a given position.

tol: Tolerance for convergence.

n.mean.perm: Numeric vector of length one indicates the number of permutations to execute.

seed: Numeric vector of length one indicates the random seed to start the permutations.

quiet: If FALSE, print some information about the course of the calculations.

Value

A vector of length `n.mean.perm` of the maximum negative log10 p-value that resulted from each permutation.
scanonevar.varperm

Permutation test for variance effect in scanonevar

Description

Executes permutations of the genotypes in the variance-effect part of scanonevar

Usage

scanonevar.varperm(cross, pheno.col=1, mean_covar=NULL, var_covar=NULL, maxit=25, tol=1e-6, n.var.perm = 2, seed = 27517, quiet=TRUE)

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 pheno-
type. This must be a single value (integer index or phenotype name) or a nu-
meric vector of phenotype values, in which case it must have the length equal
to the number of individuals in the cross, and there must be either non-integers
or values < 1 or > no. phenotypes; this last case may be useful for studying
transformations.
mean_covar Numeric matrix with covariates affecting the mean.
var_covar Numeric matrix with covariates affecting the variances.
maxit Maximum number of iterations in the algorithm to fit the model at a given posi-
tion.
tol Tolerance for convergence.
n.var.perm Numeric vector of length one indicates the number of permutations to execute.
seed Numeric vector of length one indicates the random seed to start the permuta-
tions.
quiet If FALSE, print some information about the course of the calculations.

Value

A vector of length n.var.perm of the maximum negative log10 p-value that resulted from each permutation.
**scanPhyloQTL**

*Single-QTL genome scan to map QTL to a phylogenetic tree*

**Description**

Jointly consider multiple intercrosses with a single diallelic QTL model, considering all possible partitions of the strains into the two QTL allele groups.

**Usage**

```r
scanPhyloQTL(crosses, partitions, chr, pheno.col=1,
model=c("normal", "binary"), method=c("em", "imp", "hk"),
addcovar, maxit=4000, tol=0.0001, useAllCrosses=TRUE,
verbose=FALSE)
```

**Arguments**

- `crosses`: A list with each component being an intercross, as an object of class `cross` (see `read.cross` for details). The names (of the form "AB") indicate the strains in the cross.
- `partitions`: A vector of character strings of the form "ABCD" or "ABCD" indicating the set of partitions of the strains into two allele groups. If missing, all partitions should be considered.
- `chr`: Optional vector indicating the chromosomes for which LOD scores should be calculated. This should be a vector of character strings referring to chromosomes by name; numeric values are converted to strings. Refer to chromosomes with a preceding `-` to have all chromosomes but those considered. A logical (TRUE/FALSE) vector may also be used.
- `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. One may also give a character strings matching the phenotype names. Finally, one may give a numeric vector of phenotypes, in which case it must have the length equal to the number of individuals in the cross, and there must be either non-integers or values < 1 or > no. phenotypes; this last case may be useful for studying transformations.
- `model`: The phenotype model: the usual normal model or a model for binary traits
- `method`: Indicates whether to use the EM algorithm, imputation, or Haley-Knott regression.
- `addcovar`: Optional set of additive covariates to include in the analysis, as a list with the same length as `crosses`. They must be numeric vectors or matrices, as for `scanone`.
- `maxit`: Maximum number of iterations for method "em".
- `tol`: Tolerance value for determining convergence for method "em".
- `useAllCrosses`: If TRUE, use all crosses in the analysis of all partitions, with crosses not segregating the QTL included in the estimation of the residual variance.
- `verbose`: If TRUE, print information about progress.
Details

The aim is to jointly consider multiple intercrosses to not just map QTL but to also, under the assumption of a single diallelic QTL, identify the set of strains with each QTL allele.

For each partition (of the strains into two groups) that is under consideration, we pull out the set of crosses that are segregating the QTL, re-code the alleles, and combine the crosses into one large cross. Crosses not segregating the QTL are also used, though with no QTL effects.

Additive covariate indicators for the crosses are included in the analysis, to allow for the possibility that there are overall shifts in the phenotypes between crosses.

Value

A data frame, as for the output of scanone, though with LOD score columns for each partition that is considered. The result is given class "scanPhyloQTL".

Author(s)

Karl W Broman, <broman@wisc.edu>

References


See Also

plot.scanPhyloQTL, summary.scanPhyloQTL, max.scanPhyloQTL, inferredpartitions, simPhyloQTL

Examples

# example map; drop X chromosome
data(map10)
map10 <- map10[1:19]

# simulate data
x <- simPhyloQTL(4, partition="AB|CD", crosses=c("AB", "AC", "AD"),
                 map=map10, n.ind=150,
                 model=c(1, 50, 0.5, 0))

# run calc.genoprob on each cross
## Not run: x <- lapply(x, calc.genoprob, step=2)

# scan genome, at each position trying all possible partitions
out <- scanPhyloQTL(x, method="hk")

# maximum peak
max(out, format="lod")
# approximate posterior probabilities at peak
max(out, format="postprob")

# all peaks above a threshold for LOD(best) - LOD(2nd best)
summary(out, threshold=1, format="lod")

# all peaks above a threshold for LOD(best), showing approx post'\textquoteleft r prob
summary(out, format="postprob", threshold=3)

# plot results
plot(out)

---

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=c("imp","hk"), model=c("normal", "binary"),
incl.markers=FALSE, verbose=TRUE, tol=1e-4, maxit=1000,
forceXcovar=FALSE)
```

**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. One may also give a character string matching a phenotype name. Finally, one may give a numeric vector of phenotypes, in which case it must have the length equal to the number of individuals in the cross, and there must be either non-integers or values $< 1$ or $> 0$. phenotypes; this last case may be useful for studying transformations.

- `chr`  
  Vector indicating the chromosome for each QTL. (These should be character strings referring to the chromosomes by name.)

- `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 matrix or data.frame of covariates. These must be strictly numeric.

- `formula`  
  An object of class `formula` indicating the model to be fitted. (It can also be the character string representation of a formula.) QTLs are indicated as $Q1, Q2$, etc. Covariates are indicated by their names in `covar`.

- `method`  
  Indicates whether to use multiple imputation or Haley-Knott regression.
model
incl.markers
verbose
tol
maxit
forceXcovar

The phenotype model: the usual model or a model for binary traits.
If FALSE, do calculations only at points on an evenly spaced grid. If `calc.genoprob`
or `sim.geno` were run with `stepwidth="variable"` or `stepwidth="max"`, we force `incl.markers=TRUE`.

If TRUE, give feedback about progress.
Tolerance for convergence for the binary trait model.
Maximum number of iterations for fitting the binary trait model.
If TRUE, force inclusion of X-chr-related covariates (like sex and cross direction).

Details

The formula is used to specified 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. We enforce a hierarchical structure on the model formula: if a QTL or covariate is in involved in an interaction, its main effect are also be included.

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 $y \sim Q1*Q2*Sex$, 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 $y \sim Q1 + Q2 + Q3 + Sex + Q1:Q2 + Q1:Sex + Q2:Sex + Q1:Q2:Sex$.

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`, `refineqtl`
Examples

data(fake.f2)

# take out several QTLs
cq <- c(1, 8, 13)
fake.f2 <- subset(fake.f2, chr=cq)

# impute genotypes
fake.f2 <- calc.genoprob(fake.f2, step=5, err=0.001)

# 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=cq, pos=pos,
                  formula=y~Q1+Q2+Q3, method="hk")

# 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=cq, pos=pos,
                    formula=y~Q1+Q2+Q3+Q1:Q3, method="hk")

# 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

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,
        clean.nmar=1, clean.distance=0,
        maxit=4000, tol=1e-4,
        verbose=TRUE, n.perm, perm.Xsp=FALSE, perm.strata=NULL,
        assumeCondIndep=FALSE, batchsize=250, n.cluster=1)
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. This should be a vector of character strings referring to chromosomes by name; numeric values are converted to strings. Refer to chromosomes with a preceding - to have all chromosomes but those considered. A logical (TRUE/FALSE) vector may also be used.

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. One may also give character strings matching the phenotype names. Finally, one may give a numeric vector of phenotypes, in which case it must have the length equal to the number of individuals in the cross, and there must be either non-integers or values < 1 or > no. phenotypes; this last case may be useful for studying transformations.

model The phenotype model: the usual normal model or a model for binary traits.

method Indicates whether to use the 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" or stepwidth="max", we force incl.markers=TRUE.

clean.output If TRUE, clean the output with clean.scantwo, replacing LOD scores for pairs of positions that are not well separated 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.

clean.nmar If clean.output=TRUE, this is the number of markers that must separate two positions.

clean.distance If clean.output=TRUE, this is the cM distance that must separate two positions.

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.
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.

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.

If TRUE, assume conditional independence of QTL genotypes given marker genotypes. This is an approximation, but it may speed things up.

The number of phenotypes (or permutations) to be run as a batch; used only for methods "hk" and "imp".

If the package snow is available and \( n \_perm > 0 \), permutations are run in parallel using this number of nodes.

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.

For model="binary", a logistic regression model is used.
Value

If \( \text{n.perm} \) is missing, the function returns a list with class "scantwo" and containing three components. The first component is a matrix of dimension \([\text{tot.pos} \times \text{tot.pos}]\); 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 \text{scanone}. 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 \text{scanone} including sex and/or cross direction as additive covariates, which is needed for a proper calculation of conditional LOD scores.

If \( \text{n.perm} \) is specified, the function returns a list with six 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. Fifth, 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. Finally, the maximum single-QTL LOD score (that is, from a single-QTL scan). The latter is not used in \text{summary.scantwo}, but does get calculated at each permutation, so we include it for the sake of completeness.

If \( \text{n.perm} \) is specified and \( \text{perm.Xsp} = \text{TRUE} \), the result is a list with the permutation results for the regions A:A, A:X, and X:X, each of which is a list with the six different LOD scores. Independent permutations are performed in each region, \( \text{n.perm} \) is the number of permutations for the A:A region; additional permutations are are used for the A:X and X:X parts, as estimates of quantiles farther out into the tails are needed.

X chromosome

The X chromosome must be treated specially in QTL mapping.

As in \text{scanone}, 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 \text{scanone}.)

If \( \text{n.perm} \) is specified and \( \text{perm.Xsp} = \text{TRUE} \), X-chromosome-specific permutations are performed, to obtain separate thresholds for the regions A:A, A:X, and X:X.

Author(s)

Karl W Broman, <broman@wisc.edu>; Hao Wu

References


See Also

- `plot.scantwo`
- `summary.scantwo`
- `scanone`
- `max.scantwo`
- `summary.scantwoperm`
- `c.scantwoperm`

Examples

```r
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 <- pull.pheno(fake.bc, c("sex","age"))
ic <- pull.pheno(fake.bc, "sex")

out <- scantwo(fake.bc, method="hk", pheno.col=1,
              addcovar=ac, intcovar=ic)
plot(out)
```
Permutation test for 2d genome scan by Haley-Knott regression

**Description**

Perform a permutation test with a two-dimensional genome scan with a two-QTL model, with possible allowance for additive covariates, by Haley-Knott regression.

**Usage**

```r
scantwopermhk(cross, chr, pheno.col=1,
               addcovar=NULL, weights=NULL, n.perm=1,
               batchsize=1000,
               perm.strata=NULL, perm.Xsp=NULL,
               verbose=FALSE, assumeCondIndep=FALSE)
```

**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. This should be a vector of character strings referring to chromosomes by name; numeric values are converted to strings. Refer to chromosomes with a preceding `-` to have all chromosomes but those considered. A logical (TRUE/FALSE) vector may also be used.
- `pheno.col`: Column number in the phenotype matrix which should be used as the phenotype. This should be a single value (numeric index or character string for a phenotype name), but it may also be a vector of numeric values with length equal to the number of individuals in the cross, in which case it is taken to be a vector of individuals' phenotypes.
- `addcovar`: Additive covariates.
- `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"`.
- `n.perm`: Number of permutation replicates.
- `batchsize`: If `n.perm > batchsize`, permutations will be run in batches of no more than `batchsize` permutations.
- `perm.strata`: 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.
- `perm.Xsp`: If TRUE, run separate permutations for A:A, A:X, and X:X. In this case, `n.perm` refers to the number of permutations for the A:A part; more permutations are used for the A:X and X:X parts, as estimates of quantiles farther out into the tails are needed.
- `verbose`: If TRUE, display information about the progress of calculations.
- `assumeCondIndep`: If TRUE, assume conditional independence of QTL genotypes given marker genotypes. This is an approximation, but it may speed things up.
Details

This is a scaled-back version of the permutation test provided by scantwo: only for a normal model with Haley-Knott regression, and not allowing interactive covariates.

This is an attempt to speed things up and attenuate the memory usage problems in scantwo.

In the case of perm.Xsp=TRUE (X-chr-specific thresholds), we use a stratified permutation test, stratified by sex and cross-direction.

Value

A list with six 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. Fifth, 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. Finally, the maximum single-QTL LOD score (that is, from a single-QTL scan). The latter is not used in summary.scantwoperm, but does get calculated at each permutation, so we include it for the sake of completeness.

If perm.Xsp=TRUE, this is a list of lists, for the A:A, A:X, and X:X sections, each being a list as described above.

Author(s)

Karl W Broman, <broman@wisc.edu>; Hao Wu

References


See Also

scantwo, plot.scantwoperm, summary.scantwoperm, c.scantwoperm

Examples

data(fake.f2)

fake.f2 <- calc.genoprob(fake.f2, step=5)
operm <- scantwopermhk(fake.f2, n.perm=2)
summary(operm, alpha=0.05)
shiftmap

Shift starting points in genetic maps

Description

Shift starting points in a genetic map to a set of defined positions

Usage

shiftmap(object, offset=0)

Arguments

object
An object of class cross (see read.cross for details) or map (see sim.map for details).

offset
Defines the starting position for each chromosome. This should be a single value (to be used for all chromosomes) or a vector with length equal to the number of chromosomes, defining individual starting positions for each chromosome. For a sex-specific map (as in a 4-way cross), we use the same offset for both the male and female maps.

Value

If the input is a map object, a map object is returned; if the input is a cross object, a cross object is returned. In either case, the positions of markers are shifted so that the starting positions are as in offset.

Author(s)

Karl W Broman, <broman@wisc.edu>

See Also

replace.map, est.map

Examples

data(hyper)
shiftedhyper <- shiftmap(hyper, offset=0)
par(mfrow=c(1,2))
plotMap(hyper, shift=FALSE, alternate.chrid=TRUE)
plotMap(shiftedhyper, shift=FALSE, alternate.chrid=TRUE)
Simulate a QTL experiment

Description

Simulates data for a QTL experiment using a model in which QTLs act additively.

Usage

sim.cross(map, model=NULL, n.ind=100,
  type=c("f2", "bc", "4way", "risib", "riself",
        "ri4sib", "ri4self", "ri8sib", "ri8self", "bcsft"),
  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"),
  founderGeno, random.cross=TRUE, ...)

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 chro-
  mosome 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), a phase-
  known 4-way cross (4way), or recombinant inbred lines (by selfing or by sib-
  mating, and with the usual 2 founder strains or with 4 or 8 founder strains).
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 mark-
  ers 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 in-
  cluded 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.
founderGeno For 4- or 8-way RIL, the genotype data of the founder strains, as a list whose
  components are numeric matrices (no. markers x no. founders), one for each
  chromosome.
For 4- or 8-way RIL, indicates whether the order of the founder strains should be randomized, independently for each RIL, or whether all RIL be derived from a common cross. In the latter case, for a 4-way RIL, the cross would be \((AxB)\times(CxD)\).

For \texttt{type = "bcsft"}, additional arguments passed to \texttt{sim.cross.bcsft}.

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 chromosome 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 \texttt{cross}. See \texttt{read.cross} for details.

If \texttt{keep.qtlgeno} is TRUE, the cross object will contain a component \texttt{qtlgeno} which is a matrix containing the QTL genotypes (with complete data and no errors), coded as in the genotype data.

If \texttt{keep.errorind} is TRUE and errors were simulated, each component of \texttt{geno} will each contain a matrix \texttt{errors}, with 1's indicating simulated genotyping errors.

Recombinant inbred lines

In the simulation of recombinant inbred lines (RIL), we simulate a single individual from each line, and no phenotypes are simulated (so the argument \texttt{model} is ignored).

The types \texttt{riself} and \texttt{risib} are the usual two-way RIL.

The types \texttt{ri4self}, \texttt{ri4sib}, \texttt{ri8self}, and \texttt{ri8sib} are RIL by selfing or sib-mating derived from four or eight founding parental strains.

For the 4- and 8-way RIL, one must include the genotypes of the founding individuals; these may be simulated with \texttt{simFounderSnps}. Also, the output cross will contain a component \texttt{cross}, which
is a matrix with rows corresponding to RIL and columns corresponding to the founders, indicating order of the founder strains in the crosses used to generate the RIL.

The coding of genotypes in 4- and 8-way RIL is rather complicated. It is a binary encoding of which founder strains’ genotypes match the RIL’s genotype at a marker, and not that this is specific to the order of the founders in the crosses used to generate the RIL. For example, if an RIL generated from 4 founders has the 1 allele at a SNP, and the four founders have SNP alleles 0, 1, 0, 1, then the RIL allele matches that of founders B and D. If the RIL was derived by the cross (AxB)x(CxD), then the RIL genotype would be encoded \(2^2-1 + 2^3-1 = 6\). If the cross was derived by the cross (DxA)x(CxB), then the RIL genotype would be encoded \(2^1-1 + 2^4-1 = 9\). These get reorganized after calls to `calc.genoprob`, `sim.geno`, or `argmax.geno`, and this approach simplifies the hidden Markov model (HMM) code.

For the 4- and 8-way RIL, genotyping errors are simulated only if the founder genotypes are 0/1 SNPs.

**Author(s)**

Karl W Broman, <broman@wisc.edu>

**References**


**See Also**

`saimap`, `read.cross`, `fake.f2`, `fake.bc` `fake.4way`, `simFounderSnps`

**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)))

### simulate 100 backcross individuals with 3 QTL
# a 10-cM map model after the mouse
data(map10)
```
fakebc <- sim.cross(map10, type="bc", n.ind=100,
                     model=rbind(c(1,45,1), c(5,20,1), c(5,50,1)))

### simulate 8-way RIL by sibling mating
# get lengths from the above 10-cM map
L <- ceiling(sapply(map10, max))

# simulate a 1 cM map
themap <- sim.map(L, n.mar=L+1, eq.spacing=TRUE)

# simulate founder genotypes
pg <- simFounderSnps(themap, "8")

# simulate the 8-way RIL by sib mating (256 lines)
ril <- sim.cross(themap, n.ind=256, type="ri8sib", founderGeno=pg)

---

sim.geno

**Simulate genotypes given observed marker data**

**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**

```
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", "max"))
```

**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 (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.)
**stepwidth** Indicates whether the intermediate points should with fixed or variable step sizes. We recommend using "fixed"; "variable" was included for the qtlbim package (https://cran.r-project.org/src/contrib/Archive/qtlbim). The "max" option inserts the minimal number of intermediate points so that the maximum distance between points is step.

**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, <broman@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 chromo-
            some.
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.

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, <broman@wisc.edu>

See Also

sim.cross, plotMap, replace.map, pull.map

Examples

# simulate 4 autosomes, each with 10 markers
map <- sim.map(c(100, 90, 80, 40), 10, include.x=FALSE)
plotMap(map)

# equally spaced markers
map2 <- sim.map(c(100, 90, 80, 40), 10, include.x=FALSE, eq.spacing=TRUE)
plot(map2)
Simulate founder SNPs for a multiple-strain RIL

Description

Simulate genotype data for the founding strains for a panel of multiple-strain RIL.

Usage

simFounderSnps(map, n.str=c("4","8"), pat.freq)

Arguments

map       A list whose components are vectors containing the marker locations on each of the chromosomes.
n.str     Number of founding strains (4 or 8).
pat.freq  Frequency of SNP genotype patterns in the founder (a vector of length n.str/2 + 1): (monoallelic, SNP unique to one founder, SNP present in 2 founders, [and, for the case of 8 founders, SNP in 3/8 founders, SNP in 4/8 founders].)

Details

The SNPs are simulated to be in linkage equilibrium.

Value

A vector of the same length as there are chromosomes in map, with each component being a matrix of 0's and 1's, of dim n.str x n.mar.

Author(s)

Karl W Broman, <broman@wisc.edu>

See Also

sim.map, sim.cross

Examples

data(map10)
x <- simFounderSnps(map10, "8", c(0, 0.5, 0.2, 0.2, 0.1))
**Description**

Simulate a set of intercrosses with a single diallelic QTL.

**Usage**

```r
simPhyloQTL(n.taxa=3, partition, crosses, map, n.ind=100, model,
  error.prob=0, missing.prob=0, partial.missing.prob=0,
  keep.qtlgeno=FALSE, keep.errorind=TRUE, m=0, p=0,
  map.function=c("haldane","kosambi","c-f","morgan"))
```

**Arguments**

- **n.taxa**: Number of taxa (i.e., strains).
- **partition**: A vector of character strings of the form "AB\text{ICD}" or "A\text{IBCD}" indicating, for each QTL, which taxa have which allele. If missing, simulate under the null hypothesis of no QTL.
- **crosses**: A vector of character strings indicating the crosses to do (for the form "AB", "AC", etc.). These will be sorted and then only unique ones used. If missing, all crosses will be simulated.
- **map**: A list whose components are vectors containing the marker locations on each of the chromosomes.
- **n.ind**: The number of individuals in each cross. If length 1, all crosses will have the same number of individuals; otherwise the length should be the same as crosses.
- **model**: A matrix where each row corresponds to a different QTL, and gives the chromosome number, cM position and effects of the QTL (assumed to be the same in each cross in which the QTL is segregating).
- **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.

QTLs are assumed to act additively, and the residual phenotypic variation is assumed to be normally distributed with variance $1$.

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).

Value

A list with each component being an object of class `cross`. See `read.cross` for details. The names (e.g. "AB", "AC", "BC") indicate the crosses.

If `keep.qtlgeno` is TRUE, each 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` in each cross will each contain a matrix `errors`, with 1's indicating simulated genotyping errors.

Author(s)

Karl W Broman, <broman@wisc.edu>

References


See Also

`scanPhyloQTL`, `inferredpartitions`, `summary.scanPhyloQTL`, `max.scanPhyloQTL`, `plot.scanPhyloQTL`, `sim.cross`, `read.cross`

Examples

```r
## Not run:
# example map; drop X chromosome
data(map10)
map10 <- map10[1:19]

# simulate data
x <- simPhyloQTL(4, partition="AB|CD", crosses=c("AB", "AC", "AD"),
                  map=map10, n.ind=150,
                  model=c(1, 50, 0.5, 0))

# run calc.genoprob on each cross
```
x <- lapply(x, calc.genoprob, step=2)

# scan genome, at each position trying all possible partitions
out <- scanPhyloQTL(x, method="hk")

# maximum peak
max(out, format="lod")

# approximate posterior probabilities at peak
max(out, format="postprob")

# all peaks above a threshold for LOD(best) - LOD(2nd best)
summary(out, threshold=1, format="lod")

# all peaks above a threshold for LOD(best), showing approx post' r prob
summary(out, format="postprob", threshold=3)

# plot of results
plot(out)

## End(Not run)

simulatemissingdata  Simulates missing genotype data

Description

Simulate missing genotype data by removing some genotype data from the cross object

Usage

simulatemissingdata(cross, percentage = 5)

Arguments

cross  An object of class cross. See read.cross for details.
percentage  How much of the genotype data do we need to randomly drop?

Value

An object of class cross with percentage

Author(s)

Danny Arends <danny.arends@gmail.com>
See Also

- The MQM tutorial: https://rqtql.org/tutorials/MQM-tour.pdf
- MQM - MQM description and references
- mqmscan - Main MQM single trait analysis
- mqmscanall - Parallelized traits analysis
- mqmaugment - Augmentation routine for estimating missing data
- mqmautocofactors - Set cofactors using marker density
- mqmsetcofactors - Set cofactors at fixed locations
- mqmpermutation - Estimate significance levels
- scanone - Single QTL scanning

Examples

data(multitrait)
multitrait <- fill.geno(multitrait)
multimissing5 <- simulatemissingdata(multitrait, perc=5)
perc <- (sum(nmissing(multimissing5))/sum(ntyped(multimissing5)))

stepwiseqtl

Stepwise selection for multiple QTL

Description

Performs forward/backward selection to identify a multiple QTL model, with model choice made via a penalized LOD score, with separate penalties on main effects and interactions.

Usage

stepwiseqtl(cross, chr, pheno.col=1, qtl, formula, max.qtl=10, covar=NULL,
            method=c("imp", "hk"), model=c("normal", "binary"),
            incl.markers=TRUE, refine.locations=TRUE,
            additive.only=FALSE, scan.pairs=FALSE, penalties,
            keeplodprofile=TRUE, keeptrace=FALSE, verbose=TRUE,
            tol=1e-4, maxit=1000, require.fullrank=FALSE)

Arguments

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

chr Optional vector indicating the chromosomes to consider in search for QTL. This should be a vector of character strings referring to chromosomes by name; numeric values are converted to strings. Refer to chromosomes with a preceding – to have all chromosomes but those considered. A logical (TRUE/FALSE) vector may also be used.
pheno.col  Column number in the phenotype matrix which should be used as the phenotype. One may also give character strings matching the phenotype names. Finally, one may give a numeric vector of phenotypes, in which case it must have the length equal to the number of individuals in the cross, and there must be either non-integers or values < 1 or > no. phenotypes; this last case may be useful for studying transformations.

qtl  Optional QTL object (of class "qtl", as created by makeqtl) to use as a starting point.

formula  Optional formula to define the QTL model to be used as a starting point.

max.qtl  Maximum number of QTL to which forward selection should proceed.

covar  Data frame of additive covariates.

method  Indicates whether to use multiple imputation or Haley-Knott regression.

model  The phenotype model: the usual model or a model for binary traits

incl.markers  If FALSE, do calculations only at points on an evenly spaced grid.

refine.locations  If TRUE, use refineqtl to refine the QTL locations after each step of forward and backward selection.

additive.only  If TRUE, allow only additive QTL models; if FALSE, consider also pairwise interactions among QTL.

scan.pairs  If TRUE, perform a two-dimensional, two-QTL scan at each step of forward selection.

penalties  Vector of three (or six) values indicating the penalty on the number of QTL terms. If three values, these are the penalties on main effects and heavy and light penalties on interactions. If six values, these include X-chr-specific penalties, and the values are: main effect for autosomes, main effect for X chr, heavy penalty on A:A interactions, light penalty on A:A interactions, penalty on A:X interactions, and penalty on X:X interactions. See the Details below. If missing, default values are used that are based on simulations of backcrosses and intercrosses with genomes modeled after that of the mouse.

keeplodprofile  If TRUE, keep the LOD profiles from the last iteration as attributes to the output.

keeptrace  If TRUE, keep information on the sequence of models visited through the course of forward and backward selection as an attribute to the output.

verbose  If TRUE, give feedback about progress. If verbose is an integer > 1, even more information is printed.

tol  Tolerance for convergence for the binary trait model.

maxit  Maximum number of iterations for fitting the binary trait model.

require.fullrank  If TRUE, give LOD=0 when covariate matrix in the linear regression is not of full rank.

Details

We seek to identify the model with maximal penalized LOD score. The penalized LOD score, defined in Manichaikul et al. (2009), is the LOD score for the model (the log_{10} likelihood ratio
comparing the model to the null model with no QTL) with penalties on the number of QTL and QTL:QTL interactions.

We consider QTL models allowing pairwise interactions among QTL but with an enforced hierarchy in which inclusion of a pairwise interaction requires the inclusion of both of the corresponding main effects. Additive covariates may be included, but currently we do not explore QTL:covariate interactions. Also, the penalized LOD score criterion is currently defined only for autosomal loci, and results with the X chromosome should be considered with caution.

The penalized LOD score is of the form

\[ pLOD(\gamma) = LOD(\gamma) - T_m p_m - T_h p_h - T_l p_l \]

where \( \gamma \) denotes a model, \( p_m \) is the number of QTL in the model ("main effects"), \( p_h \) is the number of pairwise interactions that will be given a heavy interaction penalty, \( p_l \) is the number of pairwise interactions that will be given a light interaction penalty, \( T_m \) is the penalty on main effects, \( T_h \) is the heavy interaction penalty, and \( T_l \) is the light interaction penalty. The penalties argument is the vector \((T_m, T_h, T_l)\). If \( T_l \) is missing (penalties has a vector of length 2), we assume \( T_l = T_h \), and so all pairwise interactions are assigned the same penalty.

The "heavy" and "light" interaction penalties can be a bit confusing. Consider the clusters of QTL that are connected via one or more pairwise interactions. To each such cluster, we assign at most one "light" interaction penalty, and give all other pairwise interactions the heavy interaction penalty. In other words, if \( p_i \) is the total number of pairwise interactions for a QTL model, we let \( p_l \) be the number of clusters of connected QTL with at least one pairwise interaction, and then let \( p_h = p_i - p_l \).

Let us give an explicit example. Consider a model with 6 QTL, and with interactions between QTL 2 and 3, QTL 4 and 5 and QTL 4 and 6 (so we have the model formula \( y \sim Q1 + Q2 + Q3 + Q4 + Q5 + Q6 + Q2:Q3 + Q4:Q5 + Q4:Q6 \)). There are three clusters of connected QTL: (1), (2,3) and (4,5,6). We would assign 6 main effect penalties \((T_m)\), 2 light interaction penalties \((T_l)\), and 1 heavy interaction penalty \((T_h)\).

Manichaikul et al. (2009) described a system for deriving the three penalties on the basis of permutation results from a two-dimensional, two-QTL genome scan (as calculated with scantwo). These may be calculated with the function calc.penalties.

A forward/backward search method is used, with the aim to optimize the penalized LOD score criterion. That is, we seek to identify the model with maximal the penalized LOD score. The search algorithm was based closely on an algorithm described by Zeng et al. (1999).

We use forward selection to a model of moderate size (say 10 QTL), followed by backward elimination all the way to the null model. The chosen model is that which optimizes the penalized LOD score criterion, among all models visited. The detailed algorithm is as follows. Note that if additive.only=TRUE, no pairwise interactions are considered.

1. Start at the null model, and perform a single-QTL genome scan, and choose the position giving the largest LOD score. If scan.pairs=TRUE, start with a two-dimensional, two-QTL genome scan instead. If an initial QTL model were defined through the arguments qtl and formula, start with this model and jump immediately to step 2.

2. With a fixed QTL model in hand:
   (a) Scan for an additional additive QTL.
   (b) For each QTL in the current model, scan for an additional interacting QTL.
   (c) If there are \( \geq 2 \) QTL in the current model, consider adding one of the possible pairwise interactions.
   (d) If scan.pairs=TRUE perform a two-dimensional, two-QTL scan, seeking to add a pair of novel QTL, either additive or interacting.
(e) Step to the model that gives the largest value for the model comparison criterion, among those considered at the current step.

3. Refine the locations of the QTL in the current model (if refine.locations=TRUE).

4. Repeat steps 2 and 3 up to a model with some pre-determined number of loci.

5. Perform backward elimination, all the way back to the null model. At each step, consider dropping one of the current main effects or interactions; move to the model that maximizes the model comparison criterion, among those considered at this step. Follow this with a refinement of the locations of the QTL.

6. Finally, choose the model having the largest model comparison criterion, among all models visited.

In this forward/backward algorithm, it is likely best to build up to an overly large model and then prune it back. Note that there is no "stopping rule"; the chosen model is that which optimizes the model comparison criterion, among all models visited. The search can be time consuming, particularly if a two-dimensional scan is performed at each forward step. Such two-dimensional scans may be useful for identifying QTL linked in repulsion (having effects of opposite sign) or interacting QTL with limited marginal effects, but our limited experience suggests that they are not necessary; important linked or interacting QTL pairs can be picked up in the forward selection to a large model, and will be retained in the backward elimination phase.

Value

The output is a representation of the best model, as measured by the penalized LOD score (see Details), among all models visited. This is QTL object (of class "qtl", as produced by makeqtl), with attributes "formula", indicating the model formula, and "pLOD" indicating the penalized LOD score.

If keeplodprofile=TRUE, LOD profiles from the last pass through the refinement algorithm are retained as an attribute, "lodprofile", to the object. These may be plotted with plotLodProfile.

If keeptrace=TRUE, the output will contain an attribute "trace" containing information on the best model at each step of forward and backward elimination. This is a list of objects of class "compactqtl", which is similar to a QTL object (as produced by makeqtl) but containing just a vector of chromosome IDs and positions for the QTL. Each will also have attributes "formula" (containing the model formula) and "pLOD" (containing the penalized LOD score).

Methods

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).

Author(s)

Karl W Broman, <broman@wisc.edu>
strip.partials

References


See Also

calc.penalties, plotModel, makeqtl, fitqtl, refineqtl, addqtl, addpair

Examples

```r
data(fake.bc)

## Not run: fake.bc <- calc.genoprob(fake.bc, step=2.5)

outsw <- stepwiseqtl(fake.bc, max.qtl=3, method="hk", keeptrace=TRUE)

# best model
outsw
plotModel(outsw)

# path through model space
thetrace <- attr(outsw, "trace")

# plot of these
par(mfrow=c(3,3))
for(i in seq(along=thetrace))
  plotModel(thetrace[[i]], main=paste("pLOD =",round(attr(thetrace[[i]],"pLOD"), 2)))
```

strip.partials

Strip partially informative genotypes

Description

Replace all partially informative genotypes (e.g., dominant markers in an intercross) with missing values.

Usage

```r
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.

Value

The same class `cross` object as in the input, but with partially informative genotypes made missing.

Author(s)

Karl W Broman, <broman@wisc.edu>

See Also

`plotMissing`, `plotInfo`

Examples

```r
data(listeria)
sum(nmissing(listeria))
listeria <- strip.partials(listeria)
sum(nmissing(listeria))
```

---

**subset.cross**

Subsetting data for QTL experiment

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, ...)
## S3 method for class 'cross'
x[chr, ind]
```

Arguments

- `x` An object of class `cross`. See `read.cross` for details.
- `chr` Optional vector specifying which chromosomes to keep or discard. This may be a logical, numeric, or character string vector. See Details, below.
- `ind` Optional vector specifying which individuals to keep discard. This may be a logical, numeric or character string vector. See Details, below.
- `...` Ignored at this point.
Details

The chr argument may be a logical vector with length equal to the number of chromosomes in the input cross x. Alternatively, it should be a vector of character strings referring to chromosomes by name. Numeric values are converted to strings. Refer to chromosomes with a preceding - to have all chromosomes but those considered.

If the ind argument is a logical vector (TRUE/FALSE), it should have length equal to the number of individuals in the input cross x. The individuals with corresponding TRUE values are retained.

If the ind argument is numeric, it should have values either between 1 and the number of individuals in the input cross x (in which case these individuals will be retained), or it should have values between -1 and -n, where n is the number of individuals in the input cross x, in which case all except these individuals will be retained.

If the input cross object x contains individual identifiers (a phenotype column labeled "id" or "ID"), and if the ind argument contains character strings, then these will be matched against the individual identifiers. If all values in ind are preceded by a -, we omit those individuals whose IDs match those in ind. Otherwise, we retain those individuals whose IDs match those in ind.

Value

The input cross object, but with only the specified subset of the data.

Author(s)

Karl W Broman, <broman@wisc.edu>

See Also

pull.map, drop.markers, subset.map

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 <- pull.pheno(listeria, 1)
listeriaB <- subset(listeria, ind = (!is.na(y) & y < 264))

# individual identifiers
listeria$pheno$ID <- paste("mouse", 1:nind(listeria), sep="")
listeriaC <- subset(listeria, ind=c("mouse1","mouse11","mouse21"))
listeriaD <- subset(listeria, ind=c("-mouse1","-mouse11","-mouse21"))

# you can also use brackets (like matrix with rows=chromosomes and columns=individuals)
temp <- listeria[c("5","13"),]  # chr 5 and 13
  temp <- listeria[, 1:10]  # first ten individuals
  temp <- listeria[5, 1:10]  # chr 5 for first ten individuals
Subsetting chromosomes for a genetic map

Description

Pull out a specified set of chromosomes from a map object.

Usage

```r
## S3 method for class 'map'
subset(x, ...)
## S3 method for class 'map'
x[...]
```

Arguments

- `x` A list whose components are vectors of marker locations.
- `...` Vector of chromosome indices.

Value

The input map object, but with only the specified subset of chromosomes.

Author(s)

Karl W Broman, <broman@wisc.edu>

See Also

`subset.cross`

Examples

```r
data(map10)
map10 <- subset(map10, chr=1:5)

# you can also use brackets
map10 <- map10[2:3]
```
**subset.scanone**  
*Subsetting the results of a genome scan*

**Description**

Pull out a specified set of chromosomes and/or LOD columns from `scanone` output.

**Usage**

```r
## S3 method for class 'scanone'
subset(x, chr, lodcolumn, ...)
```

**Arguments**

- `x`: An object of class `scanone`, output from `scanone`.
- `chr`: Optional vector specifying which chromosomes to keep. This should be a vector of character strings referring to chromosomes by name; numeric values are converted to strings. Refer to chromosomes with a preceding - to have all chromosomes but those considered. A logical (TRUE/FALSE) vector may also be used.
- `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, <broman@wisc.edu>

**See Also**

`summary.scanone`, `scanone`

**Examples**

```r
data(fake.bc)
fake.bc <- calc.genoprob(fake.bc, step=2.5)
out <- scanone(fake.bc, method="hk", pheno.col=1:2)
summary(subset(out, chr=18:19), format="allpeaks")
```
**subset.scanoneperm**  Subsetting permutation test results

**Description**

Pull out results for a specified set LOD columns from permutation results from `scanone`.

**Usage**

```r
## S3 method for class 'scanoneperm'
subset(x, repl, lodcolumn, ...)
## S3 method for class 'scanoneperm'
x[repl, lodcolumn]
```

**Arguments**

- `x` Permutation results from `scanone`, run with `n.perm>0`.
- `repl` A vector specifying which permutation replicates to keep or (if negative) omit.
- `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` permutation results, but with only the specified subset of the data.

**Author(s)**

Karl W Broman, `<broman@wisc.edu>`

**See Also**

- `summary.scanoneperm`
- `scanone`
- `c.scanoneperm`
- `cbind.scanoneperm`
- `rbind.scanoneperm`

**Examples**

```r
data(fake.bc)
fake.bc <- calc.genoprob(fake.bc, step=5)
operm <- scanone(fake.bc, method="hk", pheno.col=1:2, n.perm=25)
operm2 <- subset(operm, lodcolumn=2)

# alternatively
operm2alt <- operm[,2]
```
**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**
```r
## S3 method for class 'scantwo'
subset(x, chr, lodcolumn, ...)
```

**Arguments**
- `x` An object of class `scantwo`, output from `scantwo`.
- `chr` Optional vector specifying which chromosomes to keep. This should be a vector of character strings referring to chromosomes by name; numeric values are converted to strings. Refer to chromosomes with a preceding `-` to have all chromosomes but those considered. A logical (TRUE/FALSE) vector may also be used.
- `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, <broman@wisc.edu>

**See Also**
`summary.scantwo`, `scantwo`

**Examples**
```r
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))
```
## S3 method for class 'scantwoperm'
subset(x, repl, lodcolumn, ...)
## S3 method for class 'scantwoperm'
x[repl, lodcolumn]

### Arguments

- **x**: Permutation results from scantwo, run with n.perm>0.
- **repl**: A vector specifying which permutation replicates to keep or (if negative) omit. Ignored in case of X-chr specific permutations
- **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 permutation results, but with only the specified subset of the data.

### Examples

data(fake.bc)
fake.bc <- calc.genoprob(fake.bc, step=0)
operm <- scantwo(fake.bc, method="hk", pheno.col=1:2, n.perm=5)
operm2 <- subset(operm, lodcolumn=2)

# alternatively
operm2alt <- operm[,2]
Print pairs of individuals with similar genotype data.

Description

Prints a summary the output from `comparegeno` that includes pairs of individuals whose proportion of matching genotypes is above a chosen threshold.

Usage

```r
## S3 method for class 'comparegeno'
summary(object, thresh=0.9, ...)
```

Arguments

- `object`: An object of class `comparegeno`, the output of the function `comparegeno`.
- `thresh`: Threshold on the proportion of matching genotypes.
- `...`: Ignored at this point.

Value

A data frame with each row being a pair of individuals and columns including the individual identifiers (via `getid`, or just as numeric indexes) along with the proportion of matching genotypes.

Author(s)

Karl W Broman, <broman@wisc.edu>

See Also

`comparegeno`, `plot.comparegeno`

Examples

```r
data(fake.f2)
cg <- comparegeno(fake.f2)
summary(cg, 0.7)
```
summary.cross

Print summary of QTL experiment

Description
Print summary information about a cross object.

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

Arguments

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

... Ignored at this point.

Value
An object of class summary.cross containing a variety of summary information about the cross (this is generally printed automatically).

Author(s)
Karl W Broman, <broman@wisc.edu>

See Also
read.cross, plot.cross, nind, nmar, nchr, totmar, nphe

Examples
data(fake.f2)
summary(fake.f2)

summary.fitqtl

Summary of fit of qtl model

Description
Print summary information about the results of fitqtl.

Usage
## S3 method for class 'fitqtl'
summary(object, pvalues=TRUE, simple=FALSE, ...)

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

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

Arguments

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

... Ignored at this point.

Value
An object of class summary.cross containing a variety of summary information about the cross (this is generally printed automatically).

Author(s)
Karl W Broman, <broman@wisc.edu>

See Also
read.cross, plot.cross, nind, nmar, nchr, totmar, nphe

Examples
data(fake.f2)
summary(fake.f2)
**summary.qtl**

Print summary of a QTL object

**Description**

Print summary information about a qtl object.

**Usage**

```r
## 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, <broman@wisc.edu>

See Also

makeqtl

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 <- calc.genoprob(fake.f2, step=2, err=0.001)
qtl <- makeqtl(fake.f2, qc, qp, what="prob")

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.

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, <broman@wisc.edu>

See Also

`ripple`, `est.map`, `est.rf`

Examples

```r
## 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)
```

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", "tabByCol", "tabByChr"),
    perms, alpha, lodcolumn=1, pvalues=FALSE,
    ci.function=c("lodint", "bayesint"), ...)
```

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. 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, though it can have just one column of permutation results, in which case they are reused for all LOD score columns in the `scanone` output, `object`. (These can also be permutation results from `scantwo`, which permutations for a one-dimensional scan.)
- `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.
- `ci.function`: For formats "tabByCol" and "tabByChr", indicates the function to use to get approximate confidence intervals for QTL location.
- `...`: For formats "tabByCol" and "tabByChr", additional arguments are passed to the function indicated by `ci.function` (for example, `drop` for `lodint` or `prob` for `bayesint`, or `expandtmarkers` for either).

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 `perms` for the significance level `alpha`. If neither `threshold` nor `alpha` are specified, the peak on each chromosome is returned. Again note that with this format, only the LOD score column indicated by `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 `lodcolumn` column, are also returned.
For 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 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 threshold is a single number, it is applied to all of the LOD score columns. If alpha is specified, it must be a single significance level, applied for all LOD score columns, and again perms must be specified, and these are used to calculate the LOD score threshold for the significance level alpha.

For 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 threshold and 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.

For format="tabByCol", there will be a separate table for each LOD score column, with a single peak per chromosome. Included are columns indicating chromosome, peak position, lower and upper limits of the confidence interval calculated via lodint or bayesint, and lod score.

The output for format="tabByChr", is similar to that of format="tabByCol", but with results organized by chromosome rather than by LOD score column.

If pvalues=TRUE, and 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 perm.Xsp=TRUE in scanone), autosome- and X-chromosome specific thresholds and p-values are calculated by the method in Broman et al. (2006).

Value

An object of class summary.scanone, to be printed by print.summary.scanone.

Author(s)

Karl W Broman, <broman@wisc.edu>

References


See Also

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=1000)

# 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)

# 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)

# format="tabByCol"
summary(out2, format="tabByCol", perms=operm2, alpha=0.05, pvalues=TRUE)
# format="tabByChr", but using bayes intervals
summary(out2, format="tabByChr", perms=operm2, alpha=0.05, pvalues=TRUE,
        ci.function="bayesint")

# format="tabByChr", but using 99% bayes intervals
summary(out2, format="tabByChr", perms=operm2, alpha=0.05, pvalues=TRUE,
        ci.function="bayesint", prob=0.99)

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, expandtomarkers=FALSE, ...)
```

Arguments

- `object`: Output from `scanoneboot`.
- `prob`: Desired coverage.
- `expandtomarkers`: If TRUE, the interval is expanded to the nearest flanking markers.
- `...`: 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, `<broman@wisc.edu>`

See Also

`scanoneboot`, `plot.scanoneboot`, `lodint`, `bayesint`

Examples

```r
## Not run: data(fake.f2)
fake.f2 <- calc.genoprob(fake.f2, step=1, err=0.001)
bootoutput <- scanoneboot(fake.f2, chr=13, method="hk")
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),
          controlAcrossCol=FALSE, ...)
```

**Arguments**

- `object` Output from the function *scanone* with *n_perm* > 0.
- `alpha` Genome-wide significance levels.
- `controlAcrossCol` If TRUE, control error rate not just across the genome but also across the columns of LOD scores.
- `...` 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-α 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 Broman et al. (2006). 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.

If `controlAcrossCol=TRUE`, we use a trick to control the error rate not just across the genome but also across the LOD score columns. Namely, we convert each column of permutation results to ranks, and then for each permutation replicate we find the maximum rank across the columns. We then find the appropriate quantile of the maximized ranks, and then backtrack to the corresponding LOD score within each of the columns. See Burrage et al. (2010), right column on page 118.
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.

Author(s)

Karl W Broman, <broman@wisc.edu>

References


See Also

`scanone, summary.scanone, plot.scanoneperm`

Examples

data(fake.f2)

    fake.f2 <- calc.genoprob(fake.f2, step=2.5)

    operm1 <- scanone(fake.f2, n.perm=100, method="hk")
    summary(operm1)

    operm2 <- scanone(fake.f2, n.perm=100, method="hk", perm.Xsp=TRUE)
    summary(operm2)

    # Add noise column
    fake.f2$pheno$noise <- rnorm(nind(fake.f2))
    operm3 <- scanone(fake.f2, pheno.col=c("phenotype", "noise"), n.perm=10, method="hk")
    summary(operm3)
    summary(operm3, controlAcrossCol=TRUE, alpha=c(0.05, 0.36))

summary.scanPhyloQTL

Summarize the results a genome scan to map a QTL to a phylogenetic tree

Description

Print the maximum LOD scores for each partition on each chromosome, from the results of `scanPhyloQTL`. 
Usage

```r
## S3 method for class 'scanPhyloQTL'
summary(object, format=c("postprob", "lod"),
         threshold, ...)
```

Arguments

- `object`: An object output by the function `scanPhyloQTL`.
- `format`: Indicates whether to provide LOD scores or approximate posterior probabilities; see Details below.
- `threshold`: A threshold determining which chromosomes should be output; see Details below.
- `...`: Ignored at this point.

Details

This function is used to report chromosomes deemed interesting from a one-QTL genome scan to map QTL to a phylogenetic tree (by `scanPhyloQTL`).

For `format="lod"`, the output contains the maximum LOD score for each partition on each chromosome (which do not necessarily occur at the same position). The position corresponds to the peak location for the partition with the largest LOD score on that chromosome. The last column is the overall maximum LOD (across partitions) on that chromosome. The second-to-last column is the inferred partition (i.e., that with the largest LOD score. The third-to-last column is the difference between the LOD score for the best partition and that for the second-best.

For `format="postprob"`, the final column contains the maximum LOD score across partitions. But instead of providing the LOD scores for each partition, these are converted to approximate posterior probabilities under the assumption of a single diallelic QTL on that chromosome: on each chromosome, we take $10^{LOD}$ for the partitions and rescale them to sum to 1.

The `threshold` argument is applied to the last column (the maximum LOD score across partitions).

Value

An object of class `summary.scanPhyloQTL`, to be printed by `print.summary.scanPhyloQTL`.

Author(s)

Karl W Broman, <broman@wisc.edu>

References


See Also

`scanPhyloQTL`, `plot.scanPhyloQTL`, `max.scanPhyloQTL`, `summary.scanone`, `inferredpartitions`, `simPhyloQTL`
## Examples

```r
## Not run:
# example map; drop X chromosome
data(map10)
map10 <- map10[1:19]

# simulate data
x <- simPhyloQTL(4, partition="AB|CD", crosses=c("AB", "AC", "AD"),
                   map=map10, n.ind=150,
                   model=c(1, 50, 0.5, 0))

# run calc.genoprob on each cross
x <- lapply(x, calc.genoprob, step=2)

# scan genome, at each position trying all possible partitions
out <- scanPhyloQTL(x, method="hk")

# maximum peak
max(out, format="lod")

# approximate posterior probabilities at peak
max(out, format="postprob")

# all peaks above a threshold for LOD(best) - LOD(2nd best)
summary(out, threshold=1, format="lod")

# all peaks above a threshold for LOD(best), showing approx post'rx prob
summary(out, format="postprob", threshold=3)

# plot of results
plot(out)

## End(Not run)
```

---

### Description

Summarize the results of a two-dimensional genome scan.

### Usage

```r
## S3 method for class 'scantwo'
summary(object, thresholds, 
         what=c("best", "full", "add", "int"),
         perms, alphas, lodcolumn=1, pvalues=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.

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, though it can have just one column of permutation results, in which case they are assumed to apply to any chosen LOD score column.

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.

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.

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;

Output of addpair

Note that, for output from addpair in which the new loci are indicated explicitly in the formula, the summary provided by summary.scantwo is somewhat special. All arguments except allpairs and thresholds (and, of course, the input object) are ignored.

If the formula is symmetric in the two new QTL, the output has just two LOD score columns: lod.2v0 comparing the full model to the model with neither of the new QTL, and lod.2v1 comparing the full model to the model with just one new QTL.

If the formula is not symmetric in the two new QTL, the output has three LOD score columns: lod.2v0 comparing the full model to the model with neither of the new QTL, lod.2v1b comparing the full model to the model in which the first of the new QTL is omitted, and lod.2v1a comparing the full model to the model with the second of the new QTL omitted.

The thresholds argument should have length 1 or 2, rather than the usual 5. Rows will be retained if lod.2v0 is greater than thresholds[1] and lod.2v1 (or either of lod.2v1a or lod.2v1b) is greater than thresholds[2]. (If a single thresholds is given, we assume that thresholds[2]==0.)

The older version

The previous version of this function is still available, though it is now named summaryScantwoOld. 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, <broman@wisc.edu>

See Also

scantwo, plot.scantwo, max.scantwo, condense.scantwo
Examples

```r
data(fake.f2)

fake.f2 <- calc.genoprob(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)
operm.2dim <- c(operm.2dimA, operm.2dimB)
## End(Not run)

# estimated LOD thresholds
summary(operm.2dim)

# Summary, citing significance levels and so estimating thresholds
# from the permutation results
summary(out.2dim, perms=operm.2dim, alpha=rep(0.05, 5))

# Similar, but ignoring the interaction LOD score in the rule
summary(out.2dim, perms=operm.2dim, alpha=c(0.05, 0.05, 0, 0.05, 0.05))

# Similar, but also getting genome-scan-adjusted p-values
summary(out.2dim, perms=operm.2dim, alpha=c(0.05, 0.05, 0, 0.05, 0.05),
        pvalues=TRUE)
```

**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

```r
## S3 method for class 'scantwoperm'
summary(object, alpha=c(0.05, 0.10), ...)
```

Arguments

- `object`: Output from the function `scantwo` with `n.perm > 0`.
- `alpha`: Genome-wide significance levels.
- `...`: Ignored at this point.

Details

We take the $1 - \alpha$ quantiles of the individual LOD scores.

In the case of X-chr-specific permutations, we use the combined length of the autosomes, $L_A$, and the length of the X chromosome, $L_X$, and calculate the area of the A:A, A:X, and X:X regions as $L_A^2/2$, $L_A L_X$, and $L_X^2/2$, and then use the nominal significance levels of $1 - (1 - \alpha)^p$, where $p$ is the proportional area for that region.

Value

An object of class `summary.scantwoperm`, to be printed by `print.summary.scantwoperm`.

Author(s)

Karl W Broman, <broman@wisc.edu>

References


See Also

- `scantwo`, `summary.scantwo`, `plot.scantwoperm`

Examples

```r
data(fake.f2)
fake.f2 <- calc.genoprob(fake.f2, step=0)

## Not run: operm <- scantwo(fake.f2, n.perm=100, method="hk")
summary(operm)
```
summaryMap \hspace{1cm} \textit{Print summary of a genetic map}

**Description**

Print summary information about a map object.

**Usage**

```r
## S3 method for class 'map'
summary(object, ...)
summaryMap(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, the average inter-marker spacing, and the maximum distance between markers, for each chromosome and overall. An attribute `sexsp` indicates whether the map was sex-specific.

**Author(s)**

Karl W Broman, <broman@wisc.edu>

**See Also**

`chrlen`, `pull.map`, `summary.cross`

**Examples**

```r
data(map10)
summary(map10)
```
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
summaryScantwoOld(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.

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 Šaunak 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, <broman@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
summaryScantwoOld(out.2dim)

# Pairs meeting specified criteria
summaryScantwoOld(out.2dim, c(7, 3, 3))

# Pairs with both conditional LODs > 2
summaryScantwoOld(out.2dim, c(0, 1000, 2))

# Pairs with interaction LOD is above 3
summaryScantwoOld(out.2dim, c(0, 3, 1000))

---

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-6, 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. Only one chromosome is allowed. (This should be a character string referring to the chromosomes by name.)

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.
switchAlleles

Description

Switch alleles at selected markers in a cross object.

Usage

switchAlleles(cross, markers, switch=c("AB", "CD", "ABCD", "parents"))

Arguments

cross An object of class cross. See read.cross for details.
markers Names of markers whose alleles are to be switched.
switch For a 4-way cross, indicates how to switch the alleles (A for B, C for D, both A for B and C for D), or both A for C and B for D (parents).

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, <broman@wisc.edu>

See Also

flip.order, ripple, clean.cross

Examples

data(fake.f2)
fake.f2 <- switchAlleles(fake.f2, 1, c(1,3,2,4:7))
Details

For a backcross, we exchange homozygotes (AA) and heterozygotes (AB).
For doubled haploids and recombinant inbred lines, we exchange the two homozygotes.
For an intercross, we exchange the two homozygotes, and exchange C (i.e., not AA) and D (i.e., not BB). (The heterozygotes in an intercross are left unchanged.)
For a 4-way cross, we consider the argument switch, and the exchanges among the genotypes are more complicated.

Value

The input cross object, with alleles at selected markers switched.

Author(s)

Karl W Broman, <broman@wisc.edu>

See Also

checkAlleles, est.rf, geno.crosstab

Examples

data(fake.f2)
geno.crosstab(fake.f2, "D5M391", "D5M81")

# switch homozygotes at marker D5M391
fake.f2 <- switchAlleles(fake.f2, "D5M391")
geno.crosstab(fake.f2, "D5M391", "D5M81")

## Not run: fake.f2 <- est.rf(fake.f2)
checkAlleles(fake.f2)

## End(Not run)
Arguments

tab A data frame with two columns: chromosome and position. The row names are the marker names.

Value

A map object: a list whose components (corresponding to chromosomes) are vectors of marker positions.

Author(s)

Karl W Broman, <broman@wisc.edu>

See Also

map2table, pull.map, est.map

Examples

```r
tab <- data.frame(chr=c(1,1,1,1,2,2,2,2,3,3,3,3),
                   pos=c(0,2,4,8,0,2,4,8,0,2,4,8))
rownames(tab) <- paste0("marker", 1:nrow(tab))
map <- table2map(tab)
```

Description

List genotypes with large error LOD scores

Usage

top.errorlod(cross, chr, cutoff=4, msg=TRUE)

Arguments

cross An object of class cross. See read.cross for details.
chr Optional vector indicating the chromosomes to consider. This should be a vector of character strings referring to chromosomes by name; numeric values are converted to strings. Refer to chromosomes with a preceding - to have all chromosomes but those considered. A logical (TRUE/FALSE) vector may also be used.
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, <broman@wisc.edu>

See Also

calc.errorlod, plotGeno, plotErrorlod

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)

totmar

Determine the total number of markers

Description

Determine the total number of markers in a cross or map object.

Usage

totmar(object)

Arguments

object An object of class cross (see read.cross for details) or map (see sim.map for details).

Value

The total number of markers in the input.

Author(s)

Karl W Broman, <broman@wisc.edu>
**transformPheno**

**Transformation of the phenotypes in a cross object**

**Description**

Transform phenotypes in a cross object; by default use a logarithmic transformation, though any function may be used.

**Usage**

```r
transformPheno(cross, pheno.col=1, transf=log, ...)
```

**Arguments**

- `cross` An object of class `cross`. See `read.cross` for details.
- `pheno.col` A vector of numeric indices or character strings (indicating phenotypes by name) of phenotypes to be transformed.
- `transf` The function to use in the transformation.
- `...` Additional arguments, to be passed to `transf`.

**Value**

The input cross object with the transformed phenotypes

**Author(s)**

Danny Arends <danny.arends@gmail.com>

**See Also**

`read.cross, plot.cross, summary.cross, nind, nchr, nmar, nphe`

**Examples**

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

Examples

```r
data(multitrait)

# Log transformation of all phenotypes
multitrait.log <- transformPheno(multitrait, pheno.col=1:nphe(multitrait))

# Square-root transformation of all phenotypes
multitrait.sqrt <- transformPheno(multitrait, pheno.col=1:nphe(multitrait),
        transf=sqrt)
```

**Description**

Try all possible positions for a marker, keeping all other markers fixed, and evaluate the log likelihood and estimate the chromosome length.

**Usage**

```r
tryallpositions(cross, marker, chr, error.prob=0.0001,
        map.function=c("haldane","kosambi","c-f","morgan"),
        m=0, p=0, maxit=4000, tol=1e-6, sex.sp=TRUE, verbose=TRUE)
```

**Arguments**

- `cross` An object of class `cross`. See `read.cross` for details.
- `marker` Character string with name of the marker to move about.
- `chr` A vector specifying which chromosomes to test for the position of the marker. This should be a vector of character strings referring to chromosomes by name; numeric values are converted to strings. Refer to chromosomes with a preceding - to have all chromosomes but those considered. A logical (TRUE/FALSE) vector may also be used.
- `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. (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.
typingGap

 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, print information on progress.

Value

A data frame (actually, an object of class "scanone", so that one may use plot.scanone, summary.scanone, etc.) with each row being a possible position for the marker. The first two columns are the chromosome ID and position. The third column is a LOD score comparing the hypotheses that the marker is in that position versus the hypothesis that it is not linked to that chromosome.

In the case of a 4-way cross, with sex.sp=TRUE, there are two additional columns with the estimated female and male genetic lengths of the respective chromosome, when the marker is in that position. With sex.sp=FALSE, or for other types of crosses, there is one additional column, with the estimated genetic length of the respective chromosome, when the marker is in that position.

The row names indicate the nearest flanking markers for each interval.

Author(s)

Karl W Broman, <broman@wisc.edu>

See Also

droponemarker, est.map, ripple, est.rf, switch.order, movemarker

Examples

data(fake.bc)
tryallpositions(fake.bc, "D7M301", 7, error.prob=0, verbose=FALSE)

<table>
<thead>
<tr>
<th>typingGap</th>
<th>Maximum distance between genotyped markers</th>
</tr>
</thead>
</table>

Description

Calculates, for each individual on each chromosome, the maximum distance between genotyped markers.

Usage

typingGap(cross, chr, terminal=FALSE)
typingGap

Arguments

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

chr Optional vector indicating the chromosomes to consider. This should be a vector of character strings referring to chromosomes by name; numeric values are converted to strings. Refer to chromosomes with a preceding - to have all chromosomes but those considered. A logical (TRUE/FALSE) vector may also be used.

terminal If TRUE, just look at terminal typing gaps (from the terminal markers to the first typed marker).

Details

We consider not just the distances between internal genotypes, but also distances from the beginning of the chromosome to the first typed marker, and similarly for the end of the chromosome. (The start and end of a chromosome are taken to be the locations of the initial and final markers.) If terminal=TRUE, we look only at those beginning and end distances.

Value

A matrix with rows corresponding to individuals and columns corresponding to chromosomes. (If there is just one chromosome, it is a numeric vector rather than a matrix.)

Author(s)

Karl W Broman, <broman@wisc.edu>

See Also

ntyped, nmissing, locateX0

Examples

data(hyper)
plot(typingGap(hyper, chr=5),
     ylab="Maximum gap between typed markers (cM)",
     ylim=c(0, diff(range(pull.map(hyper,chr=5)[[1]])))))

plot(typingGap(hyper, chr=4),
     ylab="Maximum gap between typed markers (cM)",
     ylim=c(0, diff(range(pull.map(hyper,chr=4)[[1]])))))

plot(typingGap(hyper, chr=4, terminal=TRUE),
     ylab="Maximum gap between chr end and typed marker (cM)",
     ylim=c(0, diff(range(pull.map(hyper,chr=4)[[1]])))))
write.cross

Write data for a QTL experiment to a file

Description

Data for a QTL experiment is written to a file (or files).

Usage

```r
write.cross(cross, format=c("csv", "csvr", "csvs", "csvsr",
                            "mm", "qtlcart", "gary", "qtab",
                            "mapqtl", "tidy"),
               filestem="data", chr, digits=NULL, descr)
```

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, Gary Churchill’s, QTAB, MapQTL 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. This should be a vector of character strings referring to chromosomes by name; numeric values are converted to strings. Refer to chromosomes with a preceding - to have all chromosomes but those considered. A logical (TRUE/FALSE) vector may also be used.
- **digits**: Number of digits to which phenotype values and genetic map positions should be rounded. If NULL (the default), they are not rounded.
- **descr**: Character string description; used only with format="qtab".

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

QTAB format: See documentation.

MapQTL format: See documentation.

Tidy format: Data is written to three files, "stem_gen.csv", "stem_phe.csv", and "stem_map.csv" (where stem is taken from the filestem argument).

Author(s)

Karl W Broman, <broman@wisc.edu>; Hao Wu; Brian S. Yandell; Danny Arends; Aaron Wolen

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)
**Description**

Get x-axis locations for given cM positions on given chromosomes in a plot from `plot.scanone`.

**Usage**

```r
xaxisloc.scanone(out, thechr, thepos, chr, 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`.
- `thechr`: Chromosome IDs at which x-axis locations are to be determined.
- `thepos`: Chromosome positions at which x-axis locations are to be determined.
- `chr`: Optional vector specifying which chromosomes were plotted. This must be identical to what was used in the call to `plot.scanone`.
- `gap`: Gap separating chromosomes (in cM). This must be identical to what was used in the call to `plot.scanone`.

**Details**

This function allows you to identify the x-axis locations in a plot of genome scan results, produced by `plot.scanone`. This is useful for adding annotations, such as text or arrows.

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

The arguments `thechr` and `thepos` indicate the genomic positions for which x-axis locations are desired. If they both have length > 1, they must have the same length. If one has length > 1 and one has length 1, the one with length 1 is expanded to match.

**Value**

A numeric vector of x-axis locations.

**Author(s)**

Karl W Broman, <brroman@wisc.edu>

**See Also**

`plot.scanone`, `add.threshold`
Examples

data(hyper)

hyper <- calc.genoprob(hyper)
out <- scanone(hyper, method="hk")
plot(out, chr=c(1, 4, 6, 15))

# add arrow and text to indicate peak LOD score
mxout <- max(out)
x <- xaxisloc.scanone(out, mxout$chr, mxout$pos, chr=c(1,4,6,15))
arrows(x+30, mxout$lod, x+5, mxout$lod, len=0.1, col="blue")
text(x+35, mxout$lod, "the peak", col="blue", adj=c(0, 0.5))
Indexed

* Topic **io**
  - read.cross, 206
  - readMWRil, 213
  - write.cross, 301

* Topic **arith**
  - arithscan, 29
  - arithscanperm, 30

* Topic **datagen**
  - sim.cross, 251
  - sim.map, 255
  - simFounderSnps, 257
  - simPhyloQTL, 258

* Topic **datasets**
  - badorder, 31
  - bristle3, 33
  - bristleX, 34
  - fake.4way, 78
  - fake.bc, 79
  - fake.f2, 80
  - hyper, 102
  - listeria, 108
  - locations, 110
  - map10, 114
  - maphis, 116
  - multitrait, 156

* Topic **hplot**
  - add.cim.covar, 10
  - add Threshold, 12
  - effectplot, 70
  - effectscan, 72
  - geno.image, 98
  - mqmplot.cistrans, 137
  - mqmplot.clusteredheatmap, 138
  - mqmplot.cofactors, 139
  - mqmplot.directedqtl, 140
  - mqmplot.heatmap, 142
  - mqmplot.multitrait, 143
  - mqmplot.permutations, 144
  - mqmplot.singletrait, 145
  - plot.comparegeno, 168
  - plot.cross, 169
  - plot.qtl, 170
  - plot.rfmatrix, 172
  - plot.scanone, 173
  - plot.scanoneboot, 175
  - plot.scanoneperm, 176
  - plot.scanPhyloQTL, 177
  - plot.scanTwo, 179
  - plot.scanTwoPerm, 181
  - plotErrorlod, 182
  - plotGeno, 184
  - plotInfo, 185
  - plotLodProfile, 187
  - plotMap, 189
  - plotMissing, 191
  - plotModel, 192
  - plotPheno, 194
  - plotPXG, 195
  - plotRF, 196
  - xaxisloc.scanone, 303

* Topic **manip**
  - c.cross, 35
  - c.scanone, 36
  - c.scanonePerm, 37
  - c.scanTwo, 38
  - c.scanTwoPerm, 39
  - cbind.scanonePerm, 45
  - cbind.scanTwoPerm, 46
  - clean.cross, 51
  - clean.scanTwo, 52
  - convert.map, 58
  - convert.scanone, 59
  - convert.scanTwo, 60
  - convert2riself, 61
  - convert2risib, 62
  - convert2sa, 63
  - drop.dupmarkers, 65
  - drop.markers, 66
drop.nullmarkers, 66
findDupMarkers, 87
flip.order, 94
interpPositions, 106
jittermap, 107
movemarker, 123
pickMarkerSubset, 167
pull.markers, 203
replace.map, 218
replacemap.scanone, 219
replacemap.scantwo, 220
strippartials, 265
subset.cross, 266
subset.map, 268
subset.scanone, 269
subset.scanoneperm, 270
subset.scantwo, 271
subset.scantwoperm, 272
switch.order, 292

*Topic models
A starting point, 6
addcovarint, 13
addpair, 19
addqtl, 22
cim, 49
fitqtl, 89
fitstahl, 92
MQM, 124
mqmautocofactors, 128
mqmfind.marker, 130
mqmpermutation, 133
mqmprocesspermutation, 146
mqmscan, 148
mqmscanall, 150
mqmscanfrd, 152
mqmsetcofactors, 154
scanone, 225
scanonevar, 235
scanPhyloQTL, 239
scanqtl, 241
scantwo, 243
scantwopermhk, 248
stepwiseqtl, 261

*Topic print
chrlen, 48
condense.scantwo, 57
inferredpartitions, 104

max.scanone, 118
max.scanPhyloQTL, 119
max.scantwo, 121
nchr, 157
nind, 158
nmar, 159
nphe, 160
nqtl, 162
qtlversion, 205
summary.comparegeno, 273
summary.cross, 274
summary.fitqtl, 274
summary.qtl, 275
summary.ripple, 276
summary.scanone, 277
summary.scanoneperm, 282
summary.scanPhyloQTL, 283
summary.scantwo, 285
summary.scantwoperm, 288
summaryMap, 290
summaryScantwoOld, 291
top.errorlod, 295
totmar, 296

*Topic univar
plotInfo, 185

*Topic utilities
addloctocross, 17
addmarker, 18
addtoqtl, 24
allchrsplits, 26
argmax.geno, 27
bayesint, 32
calc.errorlod, 40
calc.genoprob, 42
calc.penalties, 43
checkAlleles, 47
chrnames, 49
cleanGeno, 53
comparecrosses, 54
compareqtl, 55
compareorder, 56
countXO, 64
dropfromqtl, 67
droponemarker, 68
est.map, 74
est.rf, 77
fill.geno, 81
find.flanking, 82
<table>
<thead>
<tr>
<th>Function Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>find.marker</td>
<td>83</td>
</tr>
<tr>
<td>find.markerindex</td>
<td>84</td>
</tr>
<tr>
<td>find.markerpos</td>
<td>85</td>
</tr>
<tr>
<td>find.pheno</td>
<td>86</td>
</tr>
<tr>
<td>find.pseudomarker</td>
<td>86</td>
</tr>
<tr>
<td>formLinkageGroups</td>
<td>95</td>
</tr>
<tr>
<td>formMarkerCovar</td>
<td>96</td>
</tr>
<tr>
<td>geno.crosstab</td>
<td>97</td>
</tr>
<tr>
<td>geno.table</td>
<td>99</td>
</tr>
<tr>
<td>getid</td>
<td>100</td>
</tr>
<tr>
<td>groupclusteredheatmap</td>
<td>101</td>
</tr>
<tr>
<td>inferFounderHap</td>
<td>103</td>
</tr>
<tr>
<td>locateXO</td>
<td>109</td>
</tr>
<tr>
<td>lodint</td>
<td>111</td>
</tr>
<tr>
<td>makeqtl</td>
<td>112</td>
</tr>
<tr>
<td>map2table</td>
<td>115</td>
</tr>
<tr>
<td>markerlrt</td>
<td>117</td>
</tr>
<tr>
<td>markerlnames</td>
<td>117</td>
</tr>
<tr>
<td>mqaugment</td>
<td>126</td>
</tr>
<tr>
<td>mqmextractmarkers</td>
<td>129</td>
</tr>
<tr>
<td>mqmgetmodel</td>
<td>131</td>
</tr>
<tr>
<td>mqmlotcircle</td>
<td>135</td>
</tr>
<tr>
<td>nmissing</td>
<td>159</td>
</tr>
<tr>
<td>nqrank</td>
<td>161</td>
</tr>
<tr>
<td>ntyped</td>
<td>163</td>
</tr>
<tr>
<td>nullmarkers</td>
<td>164</td>
</tr>
<tr>
<td>orderMarkers</td>
<td>165</td>
</tr>
<tr>
<td>phenames</td>
<td>166</td>
</tr>
<tr>
<td>pull.argmaxgeno</td>
<td>198</td>
</tr>
<tr>
<td>pull.draws</td>
<td>199</td>
</tr>
<tr>
<td>pull.geno</td>
<td>200</td>
</tr>
<tr>
<td>pull.genoprob</td>
<td>201</td>
</tr>
<tr>
<td>pull.map</td>
<td>202</td>
</tr>
<tr>
<td>pull.pheno</td>
<td>203</td>
</tr>
<tr>
<td>pull.rf</td>
<td>204</td>
</tr>
<tr>
<td>reduce2grid</td>
<td>214</td>
</tr>
<tr>
<td>refineqtl</td>
<td>215</td>
</tr>
<tr>
<td>reorderqtl</td>
<td>217</td>
</tr>
<tr>
<td>replaceqtl</td>
<td>221</td>
</tr>
<tr>
<td>rescalemap</td>
<td>222</td>
</tr>
<tr>
<td>ripple</td>
<td>223</td>
</tr>
<tr>
<td>scanoneboot</td>
<td>233</td>
</tr>
<tr>
<td>shiftmap</td>
<td>250</td>
</tr>
<tr>
<td>sim.geno</td>
<td>254</td>
</tr>
<tr>
<td>simulateMissingData</td>
<td>260</td>
</tr>
<tr>
<td>summary.scanoneboot</td>
<td>281</td>
</tr>
<tr>
<td>switchAlleles</td>
<td>293</td>
</tr>
<tr>
<td>table2map</td>
<td>294</td>
</tr>
<tr>
<td>transformPheno</td>
<td>297</td>
</tr>
<tr>
<td>tryallpositions</td>
<td>298</td>
</tr>
<tr>
<td>typingGap</td>
<td>299</td>
</tr>
<tr>
<td>+.scanone</td>
<td>229</td>
</tr>
<tr>
<td>+.scanone(arithscan)</td>
<td>29</td>
</tr>
<tr>
<td>+.scanoneperm(arithscanperm)</td>
<td>30</td>
</tr>
<tr>
<td>+.scantwo</td>
<td>29</td>
</tr>
<tr>
<td>-.scanone</td>
<td>29</td>
</tr>
<tr>
<td>-.scanoneperm(arithscanperm)</td>
<td>30</td>
</tr>
<tr>
<td>-.scantwo</td>
<td>29</td>
</tr>
<tr>
<td>-.scantwoperm(arithscanperm)</td>
<td>30</td>
</tr>
<tr>
<td>.Rprofile</td>
<td>6</td>
</tr>
<tr>
<td>[.cross(subset.cross)</td>
<td>266</td>
</tr>
<tr>
<td>[.map(subset.map)</td>
<td>268</td>
</tr>
<tr>
<td>[.scanoneperm(subset.scanoneperm)</td>
<td>270</td>
</tr>
<tr>
<td>[.scantwoperm(subset.scantwoperm)</td>
<td>272</td>
</tr>
</tbody>
</table>

A starting point, 6
abline, 12
add.cim.covar, 10, 51
add.threshold, 12, 174, 303
addcovarint, 13, 16
addint, 14, 15, 21, 24
addloctocross, 17, 135
addmarker, 18
addpair, 14, 16, 19, 24, 122, 180, 265, 287
addqtl, 14, 16, 21, 22, 265
adddtoqtl, 21, 24, 24, 68, 91, 113, 162, 217, 218, 222
allchrsplits, 26
argmax.geno, 27, 43, 51, 81, 82, 94, 104, 198, 199, 219, 253, 255
arthisclan, 29
arthisclanperm, 30
attr, 91, 149
badorder, 31, 77, 117, 197
barplot, 194
bayesint, 32, 112, 234, 235, 278, 279, 281
bristle3, 33, 78, 79, 81, 103, 108
bristleX, 34, 54, 35, 78, 79, 81, 103, 108

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>c.cross</td>
<td>35</td>
</tr>
<tr>
<td>c.scanone</td>
<td>36</td>
</tr>
<tr>
<td>c.scanoneperm</td>
<td>37</td>
</tr>
<tr>
<td>c.scantwo</td>
<td>38</td>
</tr>
<tr>
<td>c.scantwoperm</td>
<td>38</td>
</tr>
</tbody>
</table>
calc.errorlod, 8, 40, 43, 54, 94, 183, 185, 296
calc.penalties, 43, 263, 265
cbind.scanone (c.scanone), 36
cbind.scanoneperm, 37, 38, 45, 270
cbind.scantwo (c.scantwo), 38
cbind.scantwoperm, 40, 46
checkAlleles, 47, 77, 294
chrlen, 48, 290
chrnames, 49, 118, 167
cim, 11, 49, 133, 152
clean, 107
clean.cross, 51, 65–67, 203, 212, 293
clean.scantwo, 52, 52, 244
cleanGeno, 41, 53, 64, 110
colors, 174, 178
comparecrosses, 54
comparegeno, 55, 168, 169, 273
compareorder, 56
condense.scantwo, 57, 287
countXO, 54, 64, 110, 166
data, 7
drop.dupmarkers, 65
drop.markers, 19, 52, 65, 66, 67, 69, 88, 100, 168, 203, 267
drop.nullmarkers, 7, 52, 65, 66, 68, 88, 100, 164, 203
dropfromqtl, 25, 67, 91, 113, 162, 217, 218, 222
droponemarker, 68, 299
effectplot, 70, 74, 84, 87, 196
effectscan, 71, 72, 196
est.map, 7, 27, 31, 59, 63, 69, 74, 77, 93, 115, 116, 149, 166, 190, 207, 219–221, 223, 224, 250, 277, 293, 295, 299
est.rf, 7, 27, 31, 47, 69, 76, 77, 94–96, 116, 117, 123, 172, 197, 204, 205, 224, 277, 294, 299
example, 6
fake.4way, 34, 35, 78, 79, 81, 103, 108, 253
fake.bc, 34, 35, 78, 79, 81, 103, 108, 253
fake.f2, 34, 35, 78, 79, 80, 103, 108, 116, 253
fill.geno, 28, 50, 51, 81, 96, 97, 104, 127
find.flanking, 82, 84, 85, 87, 196
find.marker, 71, 83, 85, 87, 97, 196
find.markers, 83–85, 85, 87
find.pheno, 86
find.pseudomarker, 71, 83–85, 86
findDupMarkers, 87, 168
fitqtl, 14, 16, 21, 24, 25, 68, 89, 112, 113, 162, 217, 218, 221, 222, 242, 265, 274, 275
fitstahl, 76, 92
flip.order, 94, 293
formLinkageGroups, 95, 116, 166
formMarkerCovar, 96
formula, 13, 15, 20, 23, 89, 216, 241
geno.crosstab, 47, 97, 294
geno.image, 98, 192
geno.table, 66, 67, 97, 99, 187, 203
getid, 100
groupclusteredheatmap, 101
heatmap, 139
help.start, 6
hist, 168, 175, 176, 182, 194
hyper, 7, 34, 35, 78, 79, 81, 102, 108
image, 98, 99, 142, 180, 183, 192, 197
inferFounderHap, 103
inferredpartitions, 104, 120, 178, 240, 259, 284
interpPositions, 106
jittermap, 107
listeria, 34, 35, 78, 79, 81, 103, 108
locateXO, 54, 64, 109, 300
locations, 110
lodint, 33, 111, 234, 235, 278, 279, 281
INDEX

map10, 114
map2table, 76, 115, 202, 295
mapthis, 116
markerlrt, 94, 117
markernames, 49, 117, 167
max.scanone, 9, 118, 120, 229, 279
max.scanPhyloQTL, 105, 119, 178, 240, 259, 284
max.scantwo, 10, 21, 57, 121, 247, 287, 292
movemarker, 27, 57, 69, 123, 299
MQM, 18, 102, 124, 125–130, 132, 134, 136, 138–143, 145–147, 150, 151, 153, 154, 156, 261
mqmautocofactors, 18, 102, 125, 127, 128, 129, 131–134, 136, 138–142, 144–148, 150, 151, 153, 154, 156, 261
mqmextractmarkers, 129
mqmfind.marker, 130
mqmgetmodel, 131, 149
mqmpermutation, 18, 102, 125, 127–129, 131, 132, 133, 134, 136, 138–142, 144–147, 150, 151, 153, 154, 156, 261
mqmplot.circle, 135
mqmplot.cistrans, 18, 137
mqmplot.clusteredheatmap, 101, 138
mqmplot.cofactors, 139
mqmplot.directedqtl, 140
mqmplot.heatmap, 142
mqmplot.multitrait, 143
mqmplot.permutations, 144
mqmplot.singletrait, 145, 149
mqmprocesspermutation, 130, 146
mqmscanall, 18, 102, 125, 127–130, 132, 134–143, 145–147, 150, 150, 151, 153, 154, 156, 261
mqmscanfdr, 152
mqmsetcofactors, 18, 102, 125, 127–129, 131, 132, 134, 136, 138–142, 144–148, 150, 151, 153, 154, 156, 261
mqmtestnormal, 149, 155
multitrait, 110, 111, 156
nchr, 157, 158, 159, 161, 274, 297
nind, 158, 159, 159–161, 163, 274, 297
nmar, 158, 159, 161, 274, 297
nmissing, 56, 159, 163, 300
nphe, 158, 159, 160, 274, 297
nqrank, 161
nqtl, 162
ntyped, 160, 163, 300
nullmarkers, 67, 164
orderMarkers, 96, 116, 165
par, 174, 178, 184
phenames, 49, 118, 166
pickMarkerSubset, 88, 167
plot, 73, 143, 174, 188, 190, 193, 195
plot.comparegeno, 56, 168, 273
plot.cross, 7, 99, 158, 159, 161, 169, 190, 192, 194, 212, 274, 297
plot.map (plotMap), 189
plot.qtl, 140, 170
plot.rfmatrix, 77, 172, 197, 205
plot.scanoneboot, 175, 235, 281
plot.scanoneperm, 176, 283
plot.scanPhyloQTL, 105, 120, 177, 240, 259, 284
plot.scantwo, 9, 21, 122, 179, 247, 287, 292
plot.scantwoperm, 181, 249, 289
plotErrorlod, 41, 182, 208, 209, 296
plotGeno, 8, 41, 99, 183, 184, 208, 209, 296
plotInfo, 185, 266
plotLogProfile, 187, 216, 217, 264
plotMap, 7, 63, 76, 114, 170, 171, 189, 194, 202, 256
plotMissing, 7, 99, 170, 187, 191, 194, 266
plotModel, 192, 265
plotPheno, 170, 194
plotPXG, 71, 74, 83, 84, 87, 195
plotRF, 77, 117, 172, 196, 205
points, 11, 174
polyplot, 144
pull.argmaxgeno, 198, 199–201
pull.draws, 199, 199, 200, 201
pull.geno, 97, 199, 200, 201, 204
pull.genoprob, 97, 199, 200, 201
pull.map, 48, 107, 114, 115, 118, 200, 202,
204, 219, 256, 267, 290, 295
pull.markers, 19, 65, 66, 168, 203
pull.pheno, 200, 203
pull.rf, 77, 172, 197, 204
qnorm, 161
qtl-package (A starting point), 6
qtlversion, 205
rank, 161
rbind.scantwoperm, 270
rbind.scantwoperm(c.scantwoperm), 37
rbind.scantwoperm, 272
rbind.scantwoperm(c.scantwoperm), 39
read.cross, 7, 13, 15, 17–20, 22, 25–27, 31,
33–35, 40, 42, 47, 49–51, 53–56, 61,
62, 64–67, 69, 75, 77–81, 83–89, 92,
94–103, 107–109, 113, 116–118,
123, 126, 128, 130, 133, 135, 137,
138, 140–142, 148, 151, 152,
154–161, 163–166, 169, 183, 184,
186, 191, 194, 195, 197–204, 206,
213–215, 218, 221–223, 225, 233,
235, 237–239, 241, 244, 248, 250,
252–254, 259–261, 266, 274, 292,
293, 295–298, 300–302
read.csv, 19
read.table, 207, 209, 212, 213
readMWril, 213
reduce2grid, 214
refineqtl, 14, 16, 21, 24, 32, 91, 111,
187–189, 215, 242, 262, 265
reorderqtl, 23, 68, 91, 113, 162, 217, 222
replace.map, 8, 59, 76, 107, 202, 218, 223,
250, 256
replacemap.cross, 220, 221
replacemap.cross(replace.map), 218
replacemap.scanone, 219, 221
replacemap.scantwo, 220, 220
replacqtl, 23, 68, 91, 113, 162, 217, 218,
221
rescalemap, 107, 222
ripple, 27, 31, 57, 64, 69, 165, 166, 197, 223,
277, 292, 293, 299
rug, 168, 182
save.image, 9
scanone, 8, 9, 12, 18, 23, 24, 29, 30, 32, 33,
36–38, 44, 45, 50, 51, 59, 74, 87, 96,
97, 99, 100, 102, 111, 112, 118, 119,
125, 127–136, 138–142, 144–147,
149–154, 156, 172–176, 180, 214,
215, 219, 225, 234–236, 239, 240,
246, 247, 261, 269, 270, 277–279,
282, 283, 291, 297, 303
scanoneboot, 175, 233, 281
scanonevar, 235
scanonevar.meanperm, 237
scanonevar.varperm, 238
scanPhyloQTL, 105, 119, 120, 177, 178, 239,
259, 283, 284
scanqtl, 14, 16, 20, 21, 23, 24, 91, 189, 216,
217, 241, 275
scantwo, 9, 10, 20, 21, 29, 30, 38–40, 43, 44,
46, 52, 53, 57, 60, 121, 122,
179–182, 220, 229, 243, 249, 263,
271, 272, 278, 285–289, 291, 292
scantwoperm,h, 248
set.seed, 30
shapiro.test, 156
shiftmap, 107, 250
sim.cross, 31, 78–81, 93, 116, 212, 214, 251,
256, 257, 259
sim.geno, 8, 9, 20, 23, 28, 36, 38, 43, 51, 73,
74, 81, 82, 94, 104, 113, 195, 199,
214, 215, 219, 226, 229, 242, 244,
245, 253, 254
sim.map, 106, 114, 157, 159, 222, 250, 253,
255, 257, 296
simFounderSnps, 252, 253, 257
simPhyloQTL, 105, 120, 178, 240, 258, 284
simulate.missing.data, 260
stepwiseqtl, 43, 44, 193, 261
strip.partials, 265
subset.cross, 36, 101, 183–185, 208, 212,
266, 268
subset.map, 267, 268
subset.scanone, 269, 279
subset.scantwoperm, 270
subset.scantwo, 271
INDEX

subset.scantwoperm, 272
summary.comparegeno, 56, 169, 273
summary.cross, 7, 48, 55, 65, 100, 107, 158–161, 163, 208, 212, 274, 290, 297
summary.fitqtl, 91, 274
summary.map (summaryMap), 290
summary.qtl, 113, 162, 275
summary.ripple, 224, 276
summary.scanone, 8, 9, 27, 36, 37, 51, 69, 119, 172, 174, 187, 229, 236, 269, 277, 283, 284, 299
summary.scanoneboot, 175, 234, 235, 281
summary.scanoneperm, 9, 12, 38, 45, 147, 176, 229, 236, 270, 279, 282
summary.scanPhyloQTL, 105, 120, 178, 240, 259, 283
summary.scantwoperm, 40, 46, 182, 247, 249, 272, 288
summaryMap, 48, 290
summaryScantwoOld, 287, 291
switch.order, 27, 57, 69, 95, 123, 224, 292, 299
switchAlleles, 47, 293
table2map, 115, 294
top.errorlod, 8, 41, 101, 183, 185, 208, 209, 295
totmar, 158–161, 163, 274, 296
transformPheno, 297
tryallpositions, 298
typingGap, 299
write.cross, 212, 301
xaxisloc.scanone, 12, 174, 303