# Package 'mpMap'

February 20, 2015

**Type** Package

**Title** Multi-parent RIL genetic analysis

**Version** 1.14

**Date** 2012-06-06

**Author** Emma Huang

**Maintainer** Emma Huang <Emma.Huang@csiro.au>

**Description** Tools for constructing linkage maps, reconstructing haplotypes, estimating linkage disequilibrium and QTL mapping in multi-parent RIL designs (e.g. MAGIC)

**License** GPL-2

**LazyLoad** yes

**Depends** gdata, seriation, qtl, wgaim

**Suggests** aod, asreml, happy.hbrem, Heatplus, MASS, VPdtw, lattice

**Collate** 'add3pt.R' 'calc.genoprob2.R' 'calcmpprob.R' 'check-ped.R' 'check-QTL.R' 'clean.mpcross.R' 'cleanmap.R' 'cleanrf.R' 'combine-chr.R' 'combine-ibd.R' 'combine-ld.R' 'combine-rf.R' 'compare-orders.R' 'compute-bnld.R' 'compute-bnrf.R' 'computemap.R' 'convertped.R' 'CR-calcLD.R' 'CR-cross.R' 'CR-estrf.R' 'CR-gen-geno.R' 'fill.R' 'fillmiss.R' 'findqtl2.R' 'fit.mpqtl.R' 'gen4ped.R' 'gen8ped.R' 'generate-error.R' 'generate-obs.R' 'generate-pheno.R' 'getpval.R' 'mapcomp.R' 'maporder.R' 'mp-mapdist.R' 'mpadd.R' 'mpcalcld.R' 'mpcross.R' 'mpestrf.R' 'mpgroup.R' 'mpIM.R' 'mpMap-output.R' 'mporder.R' 'mpprob.R' 'mpsegrat.R' 'nai.R' 'plot.mpcross.R' 'plot.mpprob.R' 'plot.mpqtl.R' 'plotlink.map.R' 'print.mapcomp.R' 'print.mpcross.R' 'print.mpprob.R' 'print.mpqtl.R' 'qtlmap.R' 'read.mpcross.R' 'reorgRIgenoprob2.R' 'sim.mpcross.R' 'sim.mpped.R' 'sim.sigthr.R' 'subset.mpcross.R' 'subset.mpprob.R' 'summary.mpcross.R' 'summary.mpprob.R' 'summary.mpqtl.R' 'supportinterval.R' 'waldTests.R'

**Repository** CRAN

**Date/Publication** 2012-06-06 05:59:03

**NeedsCompilation** yes

# R topics documented:

## mpMap-package *Genetic analysis in multi-parent crosses*

### Description

Simulates data from MAGIC designs (4/8 parents), constructs linkage maps, reconstructs haplotypes, and performs QTL mapping

### Details

| | |
|---|---|
| Package: | mpMap |
| Type: | Package |
| Version: | 1.14 |
| Date: | 2012-06-06 |
| License: | GPL-2 |
| LazyLoad: | yes |

Functions for data manipulation: `mpcross`, `read.mpcross`, `write.mpcross`, `cleanmap`, `clean.mpcross`, `mpadd`, `subset.mpcross`
Functions for simulation: `sim.mpcross`, `sim.mpped`
Functions for map construction: `mpestrf`, `mpgroup`, `mporder`, `add3pt`, `compare_orders`
Functions for haplotype reconstruction: `mpprob`
Functions for QTL mapping: `mpIM`, `sim.sigthr`, `findqtl2`

### Author(s)

Emma Huang with contributions from Rohan Shah

Maintainer: Emma Huang <Emma.Huang@csiro.au>

## add3pt *Add markers to a framework map using 3-point likelihoods*

### Description

Based on a framework map and chromosome assignments, markers are inserted at the midpoints of intervals. Positions are chosen by maximizing the 3 point likelihood.

### Usage

```
add3pt(mpcross, newmpcross, newchr,
  mapfx = c("haldane", "kosambi"))
```

### Arguments

| | |
|---|---|
| mpcross | Object of class mpcross, with framework markers |
| newmpcross | Object of class mpcross, with additional markers |
| newchr | Set of chromosome listings for additional markers |
| mapfx | Map function to use - default is haldane |

### Details

Note that the values in newchr need to correspond to the chromosomes already existing in mpcross.

### Value

Combination of the two mpcross objects with new markers inserted at midpoints of intervals with maximum LOD values if LOD > 3. Note that if markers are inserted at the same midpoint, it will be necessary to locally reorder them and then re-estimate the length of the map by summing adjacent recombination fractions.

### See Also

mporder, mpadd

---

## clean.mpcross *Check data format and compute summary statistics for genotypes*

---

### Description

Given an object of class 'mpcross', the function checks that the data is in the correct format, containing founder and final genotypes, ids, and a pedigree. The number of markers genotyped for both founders and finals should coincide. The pedigree should be completely numeric. Markers which are not polymorphic across the founders are removed, as are markers which have missing values in the founders.

### Usage

```
## S3 method for class 'mpcross'
clean(object, ...)
```

### Arguments

| | |
|---|---|
| object | Object of class mpcross |
| ... | Additional arguments |

### Details

Summary statistics for the genotypes are printed, included the number of markers with varying levels of missing data, with varying levels of segregation distortion, and with different numbers of alleles.

### Value

| | |
|---|---|
| alleles | Number of alleles at each marker |
| missing | Percent missing data at each marker |
| seg | Matrix with one row for each marker and columns for the marker name, the chisquare test for segregation distortion, and the p-value of the test |

### See Also

mpcross

### Examples

```
map <- sim.map(len=100, n.mar=11, eq.spacing=TRUE, include.x=FALSE)
sim.ped <- sim.mpped(4, 1, 500, 6, 1)
sim.dat <- sim.mpcross(map=map, pedigree=sim.ped, qtl=matrix(data=c(1, 45, .4, 0, 0, 0), nrow=1, ncol=6, byrow=TR
dat.chk <- clean(sim.dat)
```

---

## cleanmap *Clean map for use in QTL mappin*

---

### Description

Given an mpcross object, this function will remove markers which are clustered too tightly together to be useful for QTL mapping. Markers will be removed from the map, from the data, and from the estimated recombination fractions, effectively subsetting the cross down to a more spaced out grid of markers.

### Usage

```
cleanmap(mpcross, mindist = 1)
```

### Arguments

| | |
|---|---|
| mpcross | Object of class mpcross |
| mindist | Minimum distance between markers in cM |

### Value

An mpcross object is returned which has markers removed which are within the minimum distance specified. These markers do not provide additional information for QTL mapping and increase the computational burden.

### See Also

mpcross

compare_orders *Compare potential orders for linkage groups*

## Description

Compares potential orderings on the basis of likelihood or number of expectedcrossovers. Based off of the ripple function in R/qtl.

## Usage

```
compare_orders(cross, chr, orders, window = 2,
  method = c("countxo", "likelihood"),
  error.prob = 1e-04,
  map.function = c("haldane", "kosambi", "c-f", "morgan"),
  maxit = 4000, tol = 1e-06, sex.sp = TRUE,
  verbose = TRUE)
```

## Arguments

| | |
|---|---|
| cross | Object of class mpcross or cross |
| chr | Selected chromosomes |
| orders | Orders to be compared |
| window | Window size for order comparison |
| method | Method for comparison, either counting the number of crossovers or likelihood value. See ripple for further details. |
| error.prob | See ripple for details. |
| map.function | See ripple for details. |
| maxit | See ripple for details. |
| tol | See ripple for details. |
| sex.sp | See ripple for details. |
| verbose | See ripple for details. |

## Details

Uses the core of the ripple function from R/qtl in order to compare orderings output from mporder. Note that method="likelihood" is substantially slower than method="countxo" and should not be used for a large number of orderings or a large window size.

## Value

The matrix of orders with crossover counts or likelihood values appended. Similar output to ripple.

## References

R/qtl

## See Also

mporder, ripple

## Examples

```
map <- sim.map(len=100, n.mar=11, eq.spacing=TRUE, include.x=FALSE)
sim.ped <- sim.mpped(4, 1, 500, 6, 1)
sim.dat <- sim.mpcross(map=map, pedigree=sim.ped, seed=1)
compare_orders(sim.dat, chr=1, orders=rbind(1:11, c(1:3, 6:4, 7:11)))
```

---

# computemap *Computes map distances*

---

## Description

Given an mpcross object with a map order and matrix of recombination fractions, this function will estimate map positions

## Usage

```
computemap(object, mapfx = c("haldane", "kosambi"),
  missfx = 2)
```

## Arguments

| | |
|---|---|
| object | Object of class mpcross |
| mapfx | Map function to convert recombination fractions to cM |
| missfx | Function to impute missing recombination fraction values |

## Value

An mpcross object is returned whose map component has been estimated based on the map order and matrix of recombination fractions. Missing recombination fractions are imputed either by filling in the closest non-missing value (missfx=1) or by averaging the distance between other nearby markers (missfx=2).

## See Also

mpcross

`fillmiss` *Fill in missing values for an mpcross object*

## Description

Use multi-point founder probabilities to fill in the most likely value for a missing genotype

## Usage

```
fillmiss(object, threshold = 0.7, dart = FALSE)
```

## Arguments

| | |
|---|---|
| `object` | Object of class `mpcross` |
| `threshold` | Threshold for probability to call an allele |
| `dart` | Flag for whether genotypes are coded as 0/1; shortcut for computation if true |

## Value

An mpcross object with a new component $missfinals for the original set of genotypes. The component $finals is replaced by the imputed values.

## See Also

`mpprob`, `mpcross`

`findqtl2` *Detect a second QTL in a QTL profile from (composite) interval mapping*

## Description

Given the output from an initial scan of chromosomes with significant genetic variation, locates the second peak in a QTL profile.

## Usage

```
findqtl2(mpqtl, window, drop, dwindow = 5)
```

## Arguments

| | |
|---|---|
| `mpqtl` | Object of class `mpqtl` |
| `window` | cM on each side of initial QTL from which to exclude second peak |
| `drop` | Drop from original peak required in order to detect a second peak which is above the significance threshold |
| `dwindow` | Window over which to smooth p-values - default is five markers |

## Value

The original input object with additional entries for newly detected QTL.

## See Also

mpIM, plot.mpqtl, summary.mpqtl

## Examples

```
sim.map <- sim.map(len=rep(100, 2), n.mar=11, include.x=FALSE, eq.spacing=TRUE)
sim.ped <- sim.mpped(4, 1, 500, 6, 1)
sim.dat <- sim.mpcross(map=sim.map, pedigree=sim.ped, qtl=matrix(data=c(1, 10, .4, 0, 0, 0, 1, 70, 0, .35, 0, 0),
mpp.dat <- mpprob(sim.dat, program="qtl", step=2)
mpq.dat <- mpIM(object=mpp.dat, ncov=0, responsename="pheno")
mpq2 <- findqtl2(mpq.dat, drop=2)
plot(mpq2)
summary(mpq2)
```

---

# fit *Fit a full model including all QTL and effects from base model*

---

## Description

Given the output from a scan of chromosomes with significant genetic variation, fits a full mixed model containing all effects in base model and all QTL effects.

## Usage

```
fit (object, ...)
## S3 method for class 'mpqtl'
 fit(object, ...)
```

## Arguments

| | |
|---|---|
| object | Object of class mpqtl |
| ... | Additional arguments to be used in asreml |

## Value

An asreml model and summary table of QTL effects, p-values and Wald statistics from fitting the full model.

## See Also

mpIM, summary.mpqtl

## Examples

```
sim.map <- sim.map(len=rep(100, 2), n.mar=11, include.x=FALSE, eq.spacing=TRUE)
sim.ped <- sim.mpped(4, 1, 500, 6, 1)
sim.dat <- sim.mpcross(map=sim.map, pedigree=sim.ped, qtl=matrix(data=c(1, 10, .4, 0, 0, 0, 1, 70, 0, .35, 0, 0),
mpp.dat <- mpprob(sim.dat, program="qtl", step=2)
mpq.dat <- mpIM(object=mpp.dat, ncov=0, responsename="pheno")
fit(mpq.dat)
```

---

`haldaneR2X` *Conversion between recombination fractions (R) and map distance (X)*

---

## Description

Haldane and Kosambi map functions and inverses for converting between recombination fractions and map distance (in cM)

## Usage

```
haldaneR2X(r)
haldaneX2R(x)
kosambiR2X(r)
kosambiX2R(x)
```

## Arguments

| | |
|---|---|
| `x` | Map distance (measured in centiMorgans) |
| `r` | Recombination fraction |

## References

Jurg Ott, Analysis of Human Genetic Linkage, Johns Hopkins University Press, Baltimore 1999

---

`mapcomp` *Functions for comparison of two map orders*

---

## Description

Takes in two maps with the aim of comparing the position of common markers. Creates a mapcomp object for plotting and summary.

## Usage

```
mapcomp(object1, object2)
## S3 method for class 'mapcomp'
plot(x, ...)
## S3 method for class 'mapcomp'
summary(object, ...)
```

## Arguments

| | |
|---|---|
| object1 | Object inheriting class mpcross or class map |
| object2 | Object inheriting class mpcross or class map |
| object | Object of class mapcomp for summarizing |
| x | Object of class mapcomp |
| ... | Additional arguments |

## Value

An object of class mapcomp with components:

| | |
|---|---|
| commonmrk | A matrix containing 5 columns with the names of all common markers for the two maps (mname), the chromosome mapped to in the first map (chr1), the position mapped to in the first map (pos1), the chromosome mapped to in the second map (chr2) and the position mapped to in the second map (pos2) |
| samechr | A matrix containing 5 columns as above. Differs from commonmrk in that duplicated markers in either map will have been removed, so all markers map to exactly one chromosome |
| map1 | The first map - either the originally input object1, or object1$map if it inherits class mpcross |
| map2 | The first map - either the originally input object2, or object2$map if it inherits class mpcross |
| correlations | The correlation between positions in map1 and map2 for each chromosome |
| dup1 | The names of markers duplicated in map1 |
| dup2 | The names of markers duplicated in map2 |

Plot produces for a comparison for each chromosome of positions of markers which are mapped to that chromosome in both maps

Summary function returns printed summary including - number of markers in each map; number of markers common to both maps; number of duplicated markers in each map; number of markers mapped to different chromosomes; correlations between positions on each chromosome.

## Examples

```
map1 <- sim.map(len=rep(100, 4), n.mar=51, include.x=FALSE)
map2 <- sim.map(len=rep(100, 4), n.mar=51, include.x=FALSE)
mc <- mapcomp(map1, map2)
summary(mc)
plot(mc)
```

## maporder *Helper function to ensure that mpcross objects are in map order*

### Description

Orders markers within all components of mpcross object (founders, finals, etc.) to match the map order

### Usage

```
maporder(object)
```

### Arguments

| | |
|---|---|
| object | Object of class mpcross |

### Value

The original object, with all components reordered to match the map. Additional unmapped markers will appear after all mapped markers.

## mpadd *Add markers onto an existing 'mpcross' object*

### Description

If the 'mpcross' object contains only genotypes and pedigree, the markers are added to the genotypes after some format checking. If the 'mpcross' object contains estimated recombination fractions and LOD scores, the markers are added in after estimating recombination fractions with the existing markers. If the 'mpcross' object contains a map, the markers are added onto the map using 3-point mapping.

### Usage

```
mpadd(mpcross1, mpcross2, r, theta = 0.15, LOD = 5,
  mapfx = c("haldane", "kosambi"))
```

### Arguments

| | |
|---|---|
| mpcross1 | Original object of class mpcross |
| mpcross2 | Additional object of class mpcross |
| r | Grid of recombination fractions over which to maximize likelihood |
| theta | Threshold for recombination fractions used in constructing linkage groups |
| LOD | Threshold for LOD scores used in constructing linkage groups |
| mapfx | Map function for converting recombination distances to map distances |

## Value

Object of class 'mpcross' which is of the same stage as the first object. Recombination fractions and linkage groups will be recomputed after adding in the markers in the second object if necessary.

## See Also

mpestrf, mpgroup, mpcross

---

`mpcalcld` *Calculate linkage disequilibrium between all pairs of markers*

---

## Description

Calculate linkage disequilibrium for multiple multi-allelic measures, including Lewontin's D, W, and correlation.

## Usage

```
mpcalcld(object)
```

## Arguments

| | |
|---|---|
| `object` | Object of class `mpcross` |

## Value

Original object with additional component ld consisting of separate matrices for each different measure.

| | |
|---|---|
| `W` | Related to Chi-square test for independence (measure of non-centrality) |
| `LewontinD` | Lewontin's D' statistic |
| `delta2` | Delta-squared statistic |
| `r2` | Correlation between markers |

## References

Zhao H, Nettleton D, Soller M and JCM Dekkers (2005) Evaluation of linkage disequilibrium measures between multi-allelic markers as predictors of linkage disequilibrium between markers and QTL. Genet Res Camb 86:77-87.

## See Also

mpestrf

mpcross *Multi-parent cross object*

## Description

The class of object generated from `sim.mpcross` and the format input to functions to construct linkage maps for multi-parent crosses. Basic constructor - takes required R objects and formats as an mpcross object.

## Usage

```
mpcross(founders, finals, pedigree, id, fid)
```

## Arguments

| | |
|---|---|
| `founders` | Matrix of founder genotypes |
| `finals` | Matrix of final genotypes |
| `pedigree` | Cross pedigree |
| `id` | Vector of numeric IDs for lines (which rows in pedigree are observed) |
| `fid` | Vector of founder IDs |

## Value

Let n.founders be the number of founders (either 4 or 8); n.finals be the number of final RILs genotyped; and n.mrk be the number of markers genotyped. Then

| | |
|---|---|
| `founders` | Founder genotypes - a matrix with dimensions n.founders x n.mrk. Row names of the matrix should be lines names; Column names should be marker names |
| `finals` | Final genotypes - a matrix with dimensions n.finals x n.mrk. Row names of the matrix should be line name; column names should be marker names. |
| `id` | Vector of final IDs with length n.finals |
| `fid` | Vector of founder IDs with length n.founders |
| `pedigree` | Numeric pedigree with 3 columns - mother, father, ID |
| `pheno` | Phenotypic data |

Optional components:

| | |
|---|---|
| `map` | Linkage map - either from which data was generated in `sim.mpcross` or estimated at some point |
| `ibd` | IBD genotypes from founders if data has been generated from `sim.mpcross` |
| `qtlgeno` | Genotypes at QTL if data has been generated from `sim.mpcross` |
| `rf` | Recombination fractions if data has been analyzed with `mpestrf` |
| `lg` | Linkage groups if data has been analyzed with `mpgroup` |

## See Also

sim.mpcross, mpestrf, mpgroup

## Examples

```
map <- sim.map(len=rep(100,2), n.mar=11, eq.spacing=TRUE, include.x=FALSE)
sim.ped <- sim.mpped(4, 1, 500, 6, 1)
sim.dat <- sim.mpcross(map=map, pedigree=sim.ped, qtl=matrix(data=c(1, 10, .4, 0, 0, 0, 1, 70, 0, .35, 0, 0), nrow
```

---

# mpestrf *Estimate pairwise recombination fractions between markers*

---

## Description

Estimates pairwise recombination fractions by maximizing the likelihood for a multi-parent cross over a grid of possible values. Theta values and corresponding LOD scores are returned for each pair of markers in the object.

## Usage

```
mpestrf(object, r, grid = FALSE)
```

## Arguments

| | |
|---|---|
| object | Object of class mpcross |
| r | Grid of potential recombination values. If missing the function will maximize over (0, .005, .01, .015, ... , .095, .1, .11, .12, ... .49, .5). |
| grid | Flag for whether to output the entire grid of likelihoods at each potential recombination value |

## Value

Returned object is of the class 'mpcross' with the additional component rf. If n.mrk is the number of markers genotypes, this is a list with components:

| | |
|---|---|
| rf$theta | n.mrk x n.mrk matrix of estimated recombination fractions between each pair of loci |
| rf$lod | n.mrk x n.mrk matrix of LOD scores at the estimated recombination values |
| rf$lkhd | n.mrk x n.mrk matrix of likelihood values at the estimated recombination values |

## See Also

mpcross, plot.mpcross

## Examples

```
map <- sim.map(len=100, n.mar=11, eq.spacing=TRUE, include.x=FALSE)
sim.ped <- sim.mpped(4, 1, 500, 6, 1)
sim.dat <- sim.mpcross(map=map, pedigree=sim.ped, qtl=matrix(data=c(1, 50, .4, 0, 0, 0), nrow=1, ncol=6, byrow=TR
dat.rf <- mpestrf(sim.dat)
plot(dat.rf)
```

---

`mpgroup` *Group markers into linkage groups given 2-pt recombination fraction estimates*

---

## Description

Groups markers based on estimated pairwise recombination fractions. Linkage groups are built up by adding a marker if it satisfies the criteria for linkage with at least one other marker in the group.

## Usage

```
mpgroup(mpcross, theta = 0.15, LOD = 5)
```

## Arguments

| | |
|---|---|
| `mpcross` | an object of class `mpcross` which includes the component `rf` output by `mp.est.rf`. See `mpcross.object` for further details. |
| `theta` | Threshold for grouping based on recombination fraction value |
| `LOD` | Threshold for grouping based on LOD score value |

## Value

The original object, with the added component `lg` which is a list with the following components:

| | |
|---|---|
| `n.groups` | The number of linkage groups formed by the function |
| `groups` | Vector labelling each marker by assigned linkage group. Missing values mean that the marker was linked to more than one group and could not be assigned with confidence |
| `LODthresh` | The LOD threshold value used to determine linkage |
| `thetathresh` | The theta threshold value used to determine linkage |
| `order` | A list with a component for each constructed linkage group which contains the order of the markers within the linkage group |

## See Also

`mpestrf`

map <- sim.map(len=100, n.mar=11, eq.spacing=TRUE, include.x=FALSE) sim.ped <- sim.mpped(4, 1, 500, 6, 1) sim.dat <- sim.mpcross(map=map, pedigree=sim.ped, qtl=matrix(data=c(1, 50, .4, 0, 0, 0), nrow=1, ncol=6, byrow=TRUE), seed=1) dat.rf <- mpestrf(sim.dat) dat.lg <- mpgroup(dat.rf)

mpIM *(Composite) Interval Mapping for QTL detection in multi-parent crosses*

## Description

Interval mapping in multi-parent crosses with options for single-stage mixed model approach; multi-stage approach using predicted means; multi-stage approach including cofactors (CIM)

## Usage

```
mpIM(baseModel, object, pheno, idname = "id",
  threshold = 0.001, chr, step = 0,
  responsename = "predmn", ncov = 1000, window = 10,
  mrkpos = FALSE, ...)
```

## Arguments

| | |
|---|---|
| baseModel | Base phenotypic model for analysis |
| object | Object of class mpcross |
| pheno | Phenotypic object |
| idname | The idname in phenotypic data for which to output predicted means. Should match rownames of the object$finals |
| threshold | Significance threshold for QTL p-values |
| chr | Subset of chromosomes for which to compute QTL profile |
| step | Step size at which to compute the QTL profile. See mpprob for further description of default values |
| responsename | Optional input of response name to look for in object$pheno |
| ncov | Number of marker covariates to search for - default is to search for as many as possible using stepAIC (forward/backward selection) |
| window | Window of cM on each side of markers where we exclude covariates in CIM |
| mrkpos | Flag for whether to consider both marker positions and step positions or just steps. Is overridden if step=0 |
| ... | Additional arguments |

## Details

Depending on the options selected, different models will be fit for QTL detection. If the baseModel input does not include a term matching the idname input, it will be assumed that a single-stage QTL mapping approach is desired. In this case, no covariates will be added (ncov will be set to 0); all models will be fitted in asreml; and all phenotypic covariates and design factors specified in the baseModel will be fitted along with genetic covariates in mixed model interval mapping.

If the baseModel input does include a term matching the idname, then it will be assumed that a two-stage QTL mapping approach is desired. In this case, the baseModel will be fit using asreml

and predicted means will be output to be used as a response in linear model interval mapping. If ncov>0 additional marker cofactors will be fit; otherwise simple interval mapping will be run. All phenotypic covariates and design factors specified in the baseModel will be fit in the first stage.

Note that no weights are used in the second stage of analysis which may result in a loss of efficiency compared to a one-stage approach.

If no baseModel is input, it will be assumed that predicted means have been included in object as a phenotypic variable named predmn. In this case pheno is not required and asreml does not need to be used. (composite) Interval mapping will proceed as in the two-stage case depending on the value of ncov.

## Value

The original input object with additional component QTLresults containing the following elements:

| | |
|---|---|
| pheno | Input phenotype data |
| pvalue | Each component contains estimated p-values at each position on a given chromosome |
| wald | Each component contains Wald statistics at each position on a given chromosome |
| fndrfx | Each component contains founder effects estimated at each position on a given chromosome |
| qtl | Each component contains the position and effects of a detected QTL |
| call | Input arguments to function |

and with attributes describing the number of QTL detected, and the threshold used for detection. Will only return one QTL per chromosome; to find more QTL see findqtl2

## See Also

plot.mpqtl, summary.mpqtl

## Examples

```
sim.map <- sim.map(len=rep(100, 2), n.mar=11, include.x=FALSE, eq.spacing=TRUE)
sim.ped <- sim.mpped(4, 1, 500, 6, 1)
sim.dat <- sim.mpcross(map=sim.map, pedigree=sim.ped, qtl=matrix(data=c(1, 10, .4, 0, 0, 0, 1, 70, 0, .35, 0, 0),
mpp.dat <- mpprob(sim.dat, program="qtl")
## Two-stage simple interval mapping
mpq.dat <- mpIM(object=mpp.dat, ncov=0, responsename="pheno")
```

mporder *Order markers within linkage groups*

## Description

Orders markers within linkage groups using two-point or multipoint probabilities. Two-point ordering is based on estimated recombination fractions; multi-point ordering is based on R/qtl ripple function.

## Usage

```
mporder(object, chr, type = c("2", "m"),
  mapfx = c("haldane", "kosambi"), rm.rf = TRUE,
  window = 3, repeats = 1,
 criterion = c("Path_length", "AR_events", "AR_deviations", "Gradient_raw", "Inertia", "Least_squa
  missfx = 2, ...)
```

## Arguments

| | |
|---|---|
| object | Object of class mpcross |
| chr | Selected chromosomes or linkage groups to order |
| type | Which type of ordering to perform - two-point or multipoint |
| rm.rf | Flag for whether to remove recombination fraction values from 2-pt ordering which have missing values |
| window | Window size for multipoint ordering |
| repeats | Number of times to repeat multipoint ordering |
| mapfx | Map function to use to compute final cM positions |
| criterion | Criterion used in 2-pt ordering to determine best order |
| missfx | Function to use to fill missing recombination fractions. See fill |
| ... | Additional arguments |

## Details

*Two-point ordering*
To use the two-point ordering, the recombination fractions between all pairs of markers must first be estimated. If there are missing values in this matrix, the markers with the largest number of missing values will be removed until there are no missing values left. These markers will not be used in the ordering and are recommended to be inserted into the resulting framework map using add3pt later.

Multiple methods are used to investigate optimal two-point orderings. These are taken from the package seriation and include simulated annealing, hierarchical clustering, and traveling salesman solver. The orders are compared on the basis of the argument criterion. Thus the total path length, or sum of adjacent recombination fractions can be minimized; or the number of Anti-Robinson events/deviations; or the number of crossovers; or the sum of the adjacent two-point LOD scores.

*Multi-point ordering*
The multi-point ordering assumes that there is a pre-existing map, and then repeatedly applies the ripple function in R/qtl to investigate local permutations of the order. These orderings are constrained by the arguments `window` and `repeats`, which determine how large the perturbations are and how many are considered. Large values of `window` are very time consuming; recommended values are 5 or less, due to the number of permutations which must be considered. Large values of `repeats` will eventually converge to an ordering in which all local rearrangements of size `window` have been optimized with respect to the number of crossovers.

## Value

The original object with a new map component. Any pre-existing map will be retained as component $oldmap.

## See Also

`mpestrf`, `mpgroup`, `add3pt`, `seriate`, `ripple`

---

`mpprob` *Compute founder probabilities for multi-parent crosses*

---

## Description

Using haplotype probabilities, computes the probability that each location on a genome was inherited from each founder. Locations are run either at markers only, at the midpoints of all intervals or at step sizes of x cM. Probabilities can be computed using internally, or with R/happy.hbrem or R/qtl.

## Usage

```
mpprob(object, chr, step = 0, mrkpos = TRUE,
  mapfx = c("haldane", "kosambi"), ibd = FALSE,
  threshold = 0.7, program = c("mpMap", "qtl", "happy"),
  tempfiledirectory = "", generations = 5)
```

## Arguments

| | |
|---|---|
| `object` | Object of class `mpcross` |
| `chr` | Subset of chromosomes |
| `step` | Step size (in cM) to create grid of positions at which to compute probabilities. At default value of 0, probabilities are calculated at marker positions only |
| `threshold` | Threshold for calling founder probabilities |
| `mapfx` | Map function used to convert map to recombination fractions |
| `ibd` | Flag to indicate whether to compute probabilities using IBD genotypes |
| `mrkpos` | Flag to indicate whether to compute probabilities at both marker positions and step size or just step size. Is overridden for step size of 0 |

`program` R package to use to compute probabilities

`tempfiledirectory` Directory in which to output temporary files. Default is current working directory

`generations` Number of generations to assume in HAPPY. see `happy`

## Details

If `program=="mpMap"` then probabilities are computed using flanking markers at positions across the genome and represent 3-point haplotype probabilities. If `program=="happy"` then probabilities are computed using default values in R/happy.hbrem, which calculates ancestral haplotypes without using pedigree information. This only allows for probabilities to be computed at midpoints of intervals. If `program=="qtl"` then probabilities are computed from multipoint founder probabilities in R/qtl.

If `step<0` for R/mpMap or R/qtl or `step==0` for R/happy.hbrem, then probabilities are computed at the midpoints of marker intervals. However, if `step==0` for R/qtl or R/mpMap, probabilities are computed only at marker locations.

## Value

The input mpcross object is returned with two additional components:

`prob` A list with founder probabilities for each chromosome. Format is a matrix with n.founders * n.markers columns and n.lines rows. Each group of n.founders columns will add up to 1. Founder probabilities are in the order of founders in the input founder matrix.

`estfnd` A list with estimated founders for each chromosome. Format is a matrix with n.markers columns and n.lines rows. Missing values indicate where no founder probability exceeded the input threshold. Numeric values for founders indicate the row in the input founder matrix corresponding to the estimated founder.

## See Also

`plot.mpprob`, `summary.mpprob`

## Examples

```
sim.map <- sim.map(len=rep(100, 2), n.mar=11, include.x=FALSE, eq.spacing=TRUE)
sim.ped <- sim.mpped(4, 1, 500, 6, 1)
sim.dat <- sim.mpcross(map=sim.map, pedigree=sim.ped, qtl=matrix(data=c(1, 50, .4, 0, 0, 0), nrow=1, ncol=6, byro
mpp.dat <- mpprob(sim.dat, program="qtl")
plot(mpp.dat)
summary(mpp.dat)
```

## nai *Count how many generations of advanced intercross are in a pedigree*

### Description

Counts the number of generations of breeding preceding selfing and subtracts off the number necessary to minimally mix the founders' genomes

### Usage

```
nai(pedigree)
```

### Arguments

| | |
|---|---|
| pedigree | Pedigree for a multi-parent cross. Can be generated using sim.mpped |

### Value

Integer - number of generations of advanced intercrossing after mixing stage but before selfing.

### See Also

sim.mpped

### Examples

```
sim.map <- list(Chr1=seq(0,100,10))
sim.ped <- sim.mpped(4, 1, 500, 6, 1)
nai(sim.ped)
sim.ped <- sim.mpped(4, 1, 500, 6, 1, 5)
nai(sim.ped)
```

## plot.mpcross *Plot summary of mpcross object*

### Description

Plots summary of phenotypes and genetic map for mpcross object. If calculated, plots a heatmap of recombination fraction estimates and transformed LOD scores with legend.

### Usage

```
## S3 method for class 'mpcross'
plot(x, chr, ...)
```

### Arguments

| | |
|---|---|
| x | Object of class mpcross |
| chr | Selected chromosomes. Default is all |
| ... | Additional arguments to plot functions |

### Value

Phenotype distributions are plotted as histograms or barplots depending on the variable type.

Genetic maps are plotted using a version of plotlink.map for all genetic maps included in the object (may be both a simulated version and an estimated version).

If recombination fractions have been estimated, a heatmap with recombination fraction estimates below the diagonal and scaled LOD scores above the diagonal is also plotted. LOD scores are transformed to 2^(-(LOD/4+1)) in order to be on the same scale as the theta values.

### See Also

mpestrf, mpcross.object, heatmap_2, plotlink.map

### Examples

```
sim.map <- sim.map(len=rep(100, 2), n.mar=11, include.x=FALSE, eq.spacing=TRUE)
sim.ped <- sim.mpped(4, 1, 500, 6, 1)
sim.dat <- sim.mpcross(map=sim.map, pedigree=sim.ped, qtl=matrix(data=c(1, 10, .4, 0, 0, 0, 1, 70, 0, .35, 0, 0),
plot(sim.dat)
```

---

## plot.mpprob *Plot summary of founder probabilities and haplotype blocks*

---

### Description

Plot the percentage of each chromosome inherited from each founder

### Usage

```
## S3 method for class 'mpprob'
plot(x, chr, nlines, ...)
```

### Arguments

| | |
|---|---|
| x | Object of class mpprob |
| chr | Chromosomes to plot. Default is all |
| nlines | Number of lines to select to show founder ancestry. Default is all |
| ... | Additional arguments to plot function |

## Value

Barplot of the percentage of each founder on each chromosome; individual heatmaps of which chunks of each chromosome are inherited from each founder.#'

## See Also

mpprob, summary.mpprob, heatmap_2

## Examples

```
sim.map <- sim.map(len=rep(100, 2), n.mar=11, include.x=FALSE, eq.spacing=TRUE)
sim.ped <- sim.mpped(4, 1, 500, 6, 1)
sim.dat <- sim.mpcross(map=sim.map, pedigree=sim.ped, qtl=matrix(data=c(1, 10, .4, 0, 0, 0, 1, 70, 0, .35, 0, 0),
mpp.dat <- mpprob(sim.dat, program="qtl")
plot(mpp.dat)
```

---

# plot.mpqtl *Plot output from interval mapping with detected QTL*

---

## Description

Plot -log10(p-value) or test statistic against cM position for (composite) interval mapping in multi-parent crosses. QTL support intervals are indicated with rectangles surrounding peaks.

## Usage

```
  ## S3 method for class 'mpqtl'
plot(x, wald = FALSE, chr,
   lodsupport = 1, ...)
```

## Arguments

| | |
|---|---|
| x | Object of class mpqtl |
| wald | Flag for whether to plot the Wald statistic or -log10(p) |
| chr | Set of chromosomes to plot |
| lodsupport | x-LOD support interval plotted in green |
| ... | Additional arguments to plotting function |

## Value

Plots the -log10(p) or Wald statistic for all chromosomes against the total genome in cM. QTL support intervals are indicated with shaded rectangles surrounding peaks

## See Also

mpIM, summary.mpqtl

## Examples

```
sim.map <- sim.map(len=rep(100, 2), n.mar=11, include.x=FALSE, eq.spacing=TRUE)
sim.ped <- sim.mpped(4, 1, 500, 6, 1)
sim.dat <- sim.mpcross(map=sim.map, pedigree=sim.ped, qtl=matrix(data=c(1, 10, .4, 0, 0, 0, 1, 70, 0, .35, 0, 0),
mpp.dat <- mpprob(sim.dat, program="qtl", step=2)
mpq.dat <- mpIM(object=mpp.dat, ncov=0, responsename="pheno")
plot(mpq.dat)
```

---

`plotlink.map` *Plots linkage maps*

---

## Description

Plot linkage map (either as input object or as stored in mpcross object). Can also highlight QTL regions when used with qtlmap function.

## Usage

```
plotlink.map(object, chr, max.dist, marker.names = TRUE,
  tick = FALSE, squash = TRUE, colqtl = "red", ...)
```

## Arguments

| | |
|---|---|
| `object` | Either `mpcross` or `map` object |
| `chr` | Chromosomes to plot |
| `max.dist` | Plotting paramters. See `link.map.cross` |
| `marker.names` | Whether to plot marker names |
| `tick` | Plotting parameters. See `link.map.cross` |
| `squash` | Plotting parameters. See `link.map.cross` |
| `colqtl` | Color to plot QTL regions. See `qtlmap` |
| `...` | Additional arguments |

## Value

Modification of link.map.cross function from wgaim to allow more general input objects and to highlight regions around QTL. If any markers are labelled "QTLx" then they will be plotted in a different color.

## See Also

`link.map.cross`, `qtlmap`

qtlmap *Select markers in a region around QTL*

## Description

Outputs a list of markers in regions around QTL. Region is defined by the window parameter of x cM to either side of the QTL positions.

## Usage

```
qtlmap(qtlpos, qtlchr, map, window = 10, qtlnam)
```

## Arguments

| | |
|---|---|
| qtlpos | Vector of QTL positions |
| qtlchr | Vector of QTL chromosomes (one for each position) |
| map | Linkage map to determine where QTL are relative to other markers |
| window | Number of cM to each side of QTL in which to find nearby markers |
| qtlnam | Optional vector of names for the QTL |

## Value

Returns a map selecting out regions +- window around the QTL positions.

## See Also

plotlink.map

## Examples

```
sim.map <- sim.map(len=rep(100, 2), n.mar=11, include.x=FALSE, eq.spacing=TRUE)
sim.ped <- sim.mpped(4, 1, 500, 6, 1)
sim.dat <- sim.mpcross(map=sim.map, pedigree=sim.ped, qtl=matrix(data=c(1, 10, .4, 0, 0, 0, 1, 70, 0, .35, 0, 0),
mpp.dat <- mpprob(sim.dat, program="qtl")
mpq.dat <- mpIM(object=mpp.dat, ncov=0, responsename="pheno")
qmap <- qtlmap(summary(mpq.dat)[,2], summary(mpq.dat)[,1], mpq.dat$map)
plotlink.map(qmap)
```

## read.mpcross *Construct mpcross objects from datafiles*

### Description

Generate an mpcross object by reading in components from files - requires founders marker data, finals marker data, pedigree, and IDs for all lines. Marker map and phenotypic data are optional.

### Usage

```
read.mpcross(founderfile, finalfile, pedfile, mapfile,
  phenofile)
```

### Arguments

| | |
|---|---|
| `founderfile` | File containing founder genotypes - should have 1+(number of founders) columns. The first column contains the marker names - first space left blank. The first row for the other columns contains the founder name. Additional rows contain observed marker data for all founders. |
| `finalfile` | File containing final genotypes - should have 1+(number of lines) columns. The first column contains the marker names - first space left blank. The first row for the other columns contains line names. Additional rows contain observed marker data for all lines. |
| `pedfile` | File containing pedigree data - should have three or four columns - first three columns indicate id, mother and father; fourth column is a flag for whether the lines was observed. |
| `mapfile` | File containing linkage map - should contain three columns - one for marker names, one for chromosome assignment and one for map position in cM |
| `phenofile` | File containing phenotypic data - should contain one column for each phenotype, with rows indexing lines. |

### See Also

`sim.mpcross`, `mpcross`

## sim.mpcross *Simulate data from multi-parent designs*

### Description

Data is simulated according to a pedigree, map and QTL model

## Usage

```
sim.mpcross(map, pedigree, qtl = NULL, vare = 1,
  error.prob = 0, missing.prob = 0, full.prob = 0,
  keep.qtlgeno = TRUE, transpos = 1, transval = 0,
  map.function = c("haldane", "kosambi"), seed = 1,
  fg = NULL, founderld = FALSE)
```

## Arguments

| | |
|---|---|
| map | Linkage map with which to generate data. See sim.map |
| pedigree | Pedigree for a multi-parent cross. Can be generated using sim.mpped |
| qtl | QTL model, defined by a matrix with one row per QTL and 6 columns: the chromosome of the QTL, the position in cM on that chromosome, and the four founder effects |
| vare | Phenotypic error variance |
| error.prob | Probability of genotyping errors - data will be changed with this probability to one of the other founder values |
| missing.prob | Probability of missing data in final genotypes |
| full.prob | Probability of fully informative markers. Markers will be assigned with this probability to retain IBD genotypes from founders rather than being recoded into binary values. See details below for more information |
| keep.qtlgeno | Flag for whether to retain the QTL genotypes as a component in the output mpcross object |
| transpos | Positions of potential translocation (vector) |
| transval | Which founder carries the translocation |
| map.function | Map function for conversion of linkage map into recombination fractions. Default is "haldane" |
| seed | Random seed for generation of data |
| fg | Input founder genotypes (optional) - otherwise generated randomly |
| founderld | Flag for whether to generate founder genotypes in linkage equilibrium (FALSE=default) or according to recombination map (TRUE) |

## Details

Data are initially generated by transmitting founder genotypes down through the pedigree to the finals. Errors, missing data, and binary alleles are then overlaid on this data (stored in $ibd). If founderld==FALSE, binary alleles are generated at each locus with probability 0.25 that one founder will have the allele; 0.50 that two founders will have the allele; and 0.25 that three founders will have the allele. The founders with the allele are randomly selected after the number of founders with the allele has been simulated. If founderld==TRUE then some markers may be monomorphic and will need to be removed from the resulting object using clean.mpcross.

Note that if founder genotypes are input they should be coded as follows: DArT markers take values in 0,1 SNP markers take values in 0,2 All other markers take some other set of values.

## Value

Object of class mpcross. See mpcross for further details. Additional components are:

| | |
|---|---|
| ibd | Fully informative founder genotypes for all markers |
| qtlgeno | If argument keep.qtlgeno is TRUE then QTL genotypes will be retained |

## Note

Translocations can only be generated when founderld==FALSE and no founder genotypes are input. Note that founder effects in the QTL model are per allele; thus, the phenotypic difference between a line carrying the founder and one that is not will be twice the input founder effect (because all lines are inbred).

## See Also

sim.mpped, sim.map

## Examples

```
map <- sim.map(len=100, n.mar=11, eq.spacing=TRUE, include.x=FALSE)
sim.ped <- sim.mpped(4, 1, 500, 6, 1)
sim.dat <- sim.mpcross(map=map, pedigree=sim.ped, qtl=matrix(data=c(1, 50, .4, 0, 0, 0), nrow=1, ncol=6, byrow=TR
```

---

# sim.mpped *Generate pedigrees from multi-parent designs*

---

## Description

Generates pedigrees for 4-way or 8-way MAGIC designs, with specifications given by the user. These pedigrees can be used to simulate datasets with other functions in the mpMap package.

## Usage

```
sim.mpped(nfounders, nfunnels = 1, nperfam = 50,
  nssdgen = 6, nseeds = 1, iripgen = 0, seed = 1, ...)
```

## Arguments

| | |
|---|---|
| nfounders | Number of founders in the pedgiree |
| nfunnels | Number of different permutations of founder crosses (funnels) to generate. |
| nperfam | Number of plants per funnel (before selfing begins) |
| nssdgen | Number of generations of single-seed descent (selfing) |
| nseeds | Number of seeds propagated from each plant through SSD |
| iripgen | Number of generations of advanced intercrossing between mixing and selfing |
| seed | Random seed for selecting funnels to generate |
| ... | Additional arguments |

## Details

The 4-way pedigree is relatively simple compared to the 8-way pedigree, as there are many fewer possible combinations of lines to intercross. For a 4-way cross there is a maximum of 3 possible funnels (ABCD, ACBD and ADBC), while there are 315 for an 8-way cross. This is computed under the assumption that cross AB is equivalent to cross BA.

For a value of 1, the funnel will be of the form ABCD or ABCDEFGH. For values less than the maximum possible number of funnels the crosses will be selected randomly, with each possibly funnel having equal probability of being generated.

## Value

Matrix with four columns: ID, mother, father, and observed status

## See Also

sim.mpcross

## Examples

```
ped4way <- sim.mpped(4, 1, 500, 6, 1)
ped8way <- sim.mpped(8, 1, 500, 6, 1)
```

---

# sim.sigthr *Simulate a significance threshold for (composite) interval mapping*

---

## Description

Generate a significance threshold by simulating from the null hypothesis. Phenotypic values for the observed genetic data are simulated to represent data with no QTL, and a genomewide p-value threshold is calculated.

## Usage

```
sim.sigthr(mpcross, nsim = 100, alpha = 0.05, pindex = 1,
  step = 0, ncov = 0, ...)
```

## Arguments

| | |
|---|---|
| mpcross | Object of class mpcross |
| nsim | Number of null datasets to simulate; default is 100 |
| alpha | Significance threshold for QTL p-values |
| pindex | Index of phenotype (if more than one in dataset) |
| step | Step size at which to compute the QTL profile. Default is 0, at midpoints of marker intervals |
| ncov | Number of marker covariates to search for - default is 0 for interval mapping |
| ... | Additional arguments |

### Value

List with components:

| | |
|---|---|
| alpha | Input significance threshold |
| nsim | Input number of simulated datasets |
| minp | Genomewide minimum p-value for each simulated dataset |
| thr | Empirical p-value threshold |

### See Also

mpIM

---

## subset.mpcross *Subset mpcross object*

---

### Description

Reduces an mpcross object down to a specified set of chromosomes, markers and/or lines

### Usage

```
## S3 method for class 'mpcross'
subset(x, chr = NULL, markers = NULL,
   lines = NULL, ...)
```

### Arguments

| | |
|---|---|
| x | Object of class mpcross |
| chr | Selected chromosomes TO KEEP. Default is all |
| markers | Selected markers TO KEEP. Default is all |
| lines | Selected lines TO KEEP. Default is all |
| ... | Additional arguments |

### Value

The original object with chromosomes/lines/markers removed which are not listed in the arguments.

### Note

Chromosomes can be input either as the character names of chromosomes or the index of the chromosomes in the map. Markers can be input as character names or the index in the matrix x$finals. Lines can be input as either character values (matching the rownames of x$finals) or indices of rows in that matrix.

## See Also

mpcross.object

## Examples

```
map <- sim.map(len=rep(100, 2), n.mar=11, include.x=FALSE, eq.spacing=TRUE)
ped <- sim.mpped(4, 1, 500, 6, 1)
sim.dat <- sim.mpcross(map=map, pedigree=ped, qtl=matrix(data=c(1, 10, .4, 0, 0, 0, 1, 70, 0, .35, 0, 0), nrow=2,
sim.dat
red.dat <- subset(sim.dat, chr=1, lines=1:50)
red.dat
```

---

# subset.mpprob *Subset mpprob object*

---

## Description

Reduces an mpprob object down to a specified set of chromosomes, markers and/or lines

## Usage

```
 ## S3 method for class 'mpprob'
subset(x, chr = NULL, markers = NULL,
   lines = NULL, ...)
```

## Arguments

| | |
|---|---|
| x | Object of class mpprob |
| chr | Selected chromosomes TO KEEP. Default is all |
| markers | Selected markers TO KEEP. Default is all |
| lines | Selected lines TO KEEP. Default is all |
| ... | Additional arguments |

## Value

The original object with chromosomes/lines removed which are not listed in the arguments.

## Note

Chromosomes can be input either as the character names of chromosomes or the index of the chromosomes in the map. Markers can be input as either character values matching the colnames of x$finals, or indices of columns in that matrix. Note that if markers are removed, the founder probabilities will be recomputed for the new map with previous settings for mpprob. Lines can be input as either character values (matching the rownames of x$finals) or indices of rows in that matrix.

## See Also

mpprob

## summary.mpcross *Summary of mpcross object*

### Description

Summarizes mpcross object in terms of number of markers, lines, type of markers, and quality of markers

### Usage

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

### Arguments

| | |
|---|---|
| object | Object of class mpcross |
| ... | Additional arguments |

### Value

Printed summary including - markers which had to be removed due to monomorphic or missing founders; numbers of biallelic/multiallelic markers; percent of markers with missing data; percent of markers with high segregation distortion.

### See Also

mpcross

### Examples

```
sim.map <- sim.map(len=rep(100, 2), n.mar=11, include.x=FALSE, eq.spacing=TRUE)
sim.ped <- sim.mpped(4, 1, 500, 6, 1)
sim.dat <- sim.mpcross(map=sim.map, pedigree=sim.ped, qtl=matrix(data=c(1, 10, .4, 0, 0, 0, 1, 70, 0, .35, 0, 0),
summary(sim.dat)
```

## summary.mpprob *Summary of mpprob object*

### Description

Summarizes details about underlying mpcross object as well as descriptive statistics about estimated founder haplotypes

### Usage

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

### Arguments

| | |
|---|---|
| object | Object of class mpprob |
| ... | Additional arguments |

### Value

Output to screen of percentage of each chromosome inherited from founders, average number of recombinations per chromosome and genomewide, and number of finals/founders/chromosomes/markers per chromosome.

### See Also

plot.mpprob, mpprob

### Examples

```
sim.map <- sim.map(len=rep(100, 2), n.mar=11, include.x=FALSE, eq.spacing=TRUE)
sim.ped <- sim.mpped(4, 1, 500, 6, 1)
sim.dat <- sim.mpcross(map=sim.map, pedigree=sim.ped, qtl=matrix(data=c(1, 10, .4, 0, 0, 0, 1, 70, 0, .35, 0, 0),
mpp.dat <- mpprob(sim.dat, program="qtl")
summary(mpp.dat)
```

---

## summary.mpqtl *Summary of mpqtl object*

---

### Description

Prints a summary of the detected QTL

### Usage

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

### Arguments

| | |
|---|---|
| object | Object of class mpqtl |
| ... | Additional arguments |

### Value

Table with rows for each QTL detected: Column 1 is the chromosome where the QTL was detected Column 2 is the position where the QTL was detected on the chromosome Columns 3 and 4 are the flanking markers for the QTL Columns 5, 6, 7 and 8 are the effect estimates for the founders Column 9 is the Wald test statistic for the overall test at that position Column 10 is the p-value for the test statistic

## See Also

mpIM, plot.mpqtl

## Examples

```
sim.map <- sim.map(len=rep(100, 2), n.mar=11, include.x=FALSE, eq.spacing=TRUE)
sim.ped <- sim.mpped(4, 1, 500, 6, 1)
sim.dat <- sim.mpcross(map=sim.map, pedigree=sim.ped, qtl=matrix(data=c(1, 10, .4, 0, 0, 0, 1, 70, 0, .35, 0, 0),
mpp.dat <- mpprob(sim.dat, program="qtl", step=2)
mpq.dat <- mpIM(object=mpp.dat, ncov=0, thr=1, responsename="pheno")
summary(mpq.dat)
```

---

# supportinterval *Calculate support interval for detected QTL*

---

## Description

Calculates support interval for QTL based on Wald profile and QTL position

## Usage

```
supportinterval(x, chr, lodsupport = 1)
```

## Arguments

| | |
|---|---|
| x | Object of class mpqtl |
| chr | Selected chromosomes |
| lodsupport | Size of support interval; default is 1 LOD |

## Details

Computes the x-LOD support interval as the region surrounding a QTL peak in which the Wald profile exceeds the equivalent of x LOD less than the peak value.

## Value

A list with two components: a matrix containing lower and upper bounds for the support intervals for each QTL, and the positions of QTL on each chromosome.

## See Also

plot.mpqtl

## Examples

```
map <- sim.map(len=100, n.mar=11, eq.spacing=TRUE, include.x=FALSE)
sim.ped <- sim.mpped(4, 1, 500, 6, 1)
sim.dat <- sim.mpcross(map=map, pedigree=sim.ped, qtl=matrix(data=c(1, 10, .4, 0, 0, 0, 1, 70, 0, .35, 0, 0), nrow
mpp.dat <- mpprob(sim.dat, program="qtl", step=2)
mpq.dat <- mpIM(object=mpp.dat, ncov=0, responsename="pheno")
si <- supportinterval(mpq.dat)
```

---

`write.mpcross` *Output mpcross objects to other file formats*

---

## Description

Outputs the genotype information from an 'mpcross' object to files which can be read in to either R/qtl cross format or R/happy.hbrem format

## Usage

```
write2cross(object, filestem, chr, ...)
  write2happy(object, filestem, chr, ...)
  write.mpcross(object, filestem="mp", chr,
  format=c("qtl", "happy"), ...)
```

## Arguments

| | |
|---|---|
| `object` | Object of class `mpcross` |
| `filestem` | Filestem for all files which are output; may include directory |
| `chr` | Subset of chromosomes to be output; default is all |
| `format` | Output format - R/qtl or happy.hbrem |
| `...` | Additional arguments |

## Value

For R/qtl cross format, two files are output: filestem.ril.csv and filestem.founder.csv which can be then read into the R/qtl package using `readMWril`. For R/happy.hbrem format, two files are output for each chromosome: e.g. filestem.chr1.alleles and filestem.chr1.data. These can be directly input to happy.

## See Also

`readMWril`, happy

# Index