Package 'ASMap'

September 30, 2014

Type Package

Title Linkage map construction using the MSTmap algorithm

Version 0.3-1

Date 2014-09-30

Author Julian Taylor <julian.taylor@adelaide.edu.au>, David Butler <david.butler@daff.qld.gov.au>.

Maintainer Julian Taylor <julian.taylor@adelaide.edu.au>

Depends R (>= 2.0.0), qtl, gtools, fields, lattice

Description Functions for (A)ccurate and (S)peedy linkage map construction, manipulation and diagnosis of Doubled Haploid, Backcross and Recombinant Inbred R/qtl objects. This includes extremely fast linkage map clustering and optimal marker ordering using MSTmap (see Wu et al.,2008).

License GPL (>= 2)

NeedsCompilation yes

Repository CRAN

Date/Publication 2014-09-30 14:41:43

R topics documented:

ASMap-package 2
breakCross 3
combineMap 5
fixClones 6
genClones 8
heatMap 9
mapDH 11
mapDHu 11
mapF2 12

mergeCross 12
mstmap.cross 13
mstmap.data.frame 17
pp.init 21
profileGen 22
profileMark 24
pullCross 26
pushCross 28
quickEst 30
statGen 31
statMark 32

Index **34**

`ASMap-package` *Additional functions for linkage map construction and manipulation of R/qtl objects.*

Description

Additional functions for linkage map construction and manipulation of R/qtl objects. This includes extremely fast linkage map clustering and marker ordering using MSTmap (see Wu et al., 2008).

Details

Package:	ASMap
Type:	Package
Version:	0.3
Date:	2014-08-20
License:	GPL 2

Welcome to the first release version of the ASMap package!

One of the fundamental reasons why this package exists was to utilize and implement the source code for the the Minimum Spanning Tree algorithm derived in Wu et al. (2008) (reference below) for linkage map construction. The algorithm is lightning quick at linkage group clustering and optimal marker ordering and can handle large numbers of markers.

The package contains two very efficient functions, `mstmap.data.frame` and `mstmap.cross`, that provide users with a highly flexible set linkage map construction methods using the MSTmap algorithm. `mstmap.data.frame` constructs a linkage map from a data frame of genetic marker data and will use the entire contents of the object to form linkage groups and optimally order markers within each linkage group. `mstmap.cross` is a linkage map construction function for **qtl** package objects and can be used to construct linkage maps in a flexible number of ways. See `?mstmap.cross` for complete details.

To complement the computationally efficient linkage map construction functions, the package also contains functions `pullCross` and `pushCross` that allow the pulling/pushing markers of different

types to and from the linkage map. This system gives users the ability to initially pull markers aside that are not needed for immediate construction and push them back later if required. There are also functions for fast numerical and graphical diagnosis of unconstructed and constructed linkage maps. Specifically, there is an improved `heatMap` that graphically displays pairwise recombination fractions and LOD scores with separate legends for each. `profileGen` can be used to simultaneously profile multiple statistics such as recombination counts and double recombination counts for individual lines across the constructed linkage map. `profileMark` allows simultaneous graphical visualization of marker or interval statistics profiles across the genome or subsetted for a predefined set of linkage groups. Both of these functions utilize the power of the advanced graphics package **lattice** to provide seamless multiple displays.

Other miscellaneous utilities for **qtl** objects include

- `mergeCross`: Merging of linkage groups
- `breakCross`: Breaking of linkage groups
- `combineMap`: Combining linkage maps
- `quickEst`: Very quick estimation of genetic map distances
- `genClones`: Reporting genotype clones
- `fixClones`: Consensus genotypes for clonal groups

A comprehensive vignette showcasing the package will be available shortly!

Author(s)

Julian Taylor, Dave Butler, Timothy Close, Yonghui Wu, Stefano Lonardi Maintainer: Julian Taylor <julian.taylor@adelaide.edu.au>

References

Y. Wu, P. Bhat, T.J. Close, S. Lonardi, Efficient and Accurate Construction of Genetic Linkage Maps from Minimum Spanning Tree of a Graph Plos Genetics, Volume 4, Issue 10, 2008.

See Also

`qtl-package`

`breakCross` *Break linkage groups of an R/qtl cross object*

Description

Breaks linkage groups of an R/qtl cross object from a user specified list.

Usage

```
breakCross(cross, split = NULL, suffix = "numeric", sep = ".")
```

Arguments

cross	An R/qtl cross object with any class structure.
split	A list named by the linkage groups required for splitting and containing marker names immediately preceding where the splits are to be made (see Details).
suffix	This can be a vector of character strings containing "numeric" or "alpha" specifying whether integers or letters are to be appended to the old linkage group names to form new names. This argument may also be list with elements named by the linkage groups that are in split and containing the new names for the split linkage groups (see Examples).
sep	The character separator to be used to separate the linkage group name and the suffix.

Details

The splitting of any linkage group only needs to be defined by the markers immediately preceding where the splits are to be made. Multiple splits in the one linkage group are possible as well as splitting across multiple linkage groups with one call.

Value

The cross object is returned with identical class structure as the inputted cross object. The "geno" element will contain separate linkage groups for the user defined splits.

Author(s)

Julian Taylor

See Also

mergeCross

Examples

```
data(mapDH, package = "ASMap")

mapDH1 <- breakCross(mapDH, split = list("4A" = "4A.m.8"))
pull.map(mapDH1)[["4A.1"]]
pull.map(mapDH1)[["4A.2"]]

## manually choose suffix

mapDH1 <- breakCross(mapDH, split = list("4A" = "4A.m.8"), suffix = list("4A" = c("4AA","4AB")))
```

combineMap *Combine linkage maps from multiple R/qtl cross objects*

Description

Combine map information, marker data and phenotype data from multiple R/qtl cross objects

Usage

```
combineMap(..., id = "Genotype", keep.all = TRUE)
```

Arguments

...	An unlimited set of arguments with each argument defining an R/qtl cross object. All R/qtl objects can have any class structure but it must be identical across objects. (see Details for more information.)
id	The name of the common column in the pheno element of each cross object representing the genotype names. Default is "Genotype".
keep.all	A logical value determining whether all genotypes should be kept in the final linkage map regardless of their absence in some linkage maps (see Details). Default is TRUE.

Details

This function combines linkage maps from multiple R/qtl cross objects by merging marker data and map information as well as phenotypic data if present. The function contains some initial checks before proceeding with the combining. Firstly, all R/qtl cross objects must have the same class structure and have a column in the pheno element of the object named by the argument id. The marker names must also be unique across linkage maps as well as avoid the symbol ";". This symbol is reserved for string manipulation within the function.

The combining of linkage maps occurs in multiple ways. Firstly, it merges the linkage maps sequentially based on genotype names. If keep.all=TRUE then the marker data and phenotypic data are "padded out" when genotype names are not shared between maps. If keep.all=FALSE then the marker data and phenotype data are shrunk to only include genotypes that are shared among all linkage maps. Secondly, if a linkage group name is shared across linkage maps then the marker data from the shared linkage group in each of the maps will be merged.

Several things are important to note with this function. Non-matching genotype names between linkage maps will expand the final marker data and phenotypic data so it is prudent to check genotype names are correct in each of the linkage maps before combining. If maps share the same linkage group names and do not require merging the duplicate linkage group names in one of the linkage maps will need to be altered before combining. As a final process, markers are ordered within linkage groups according to distances supplied in each of the linkage maps.

It should also be noted that this function does not re-construct the final linkage map after combining the set of linkage maps. For efficient linkage map reconstruction of the combined R/qtl object see mstmap.cross().

Value

A single R/qtl cross object is returned with identical class structure as the inputted cross objects.

Author(s)

Julian Taylor

See Also

breakCross and mergeCross

Examples

```
data(mapDH, package = "ASMap")

## create copy of mapDH with some different linkage groups
## and change marker names so they are unique

mapDH1 <- mapDH
names(mapDH1$geno)[5:14] <- paste("L",1:10, sep = "")
mapDH1$geno <- lapply(mapDH1$geno, function(el){
    names(el$map) <- dimnames(el$data)[[2]] <- paste(names(el$map), "A", sep = "")
    el})

mapDHc <- combineMap(mapDH, mapDH1)
nmar(mapDHc)
```

---

fixClones *Consensus genotypes for clonal genotype groups*

Description

Consensus genotypes for clonal genotype groups of an R/qtl object.

Usage

```
fixClones(object, gc, id = "Genotype", consensus = TRUE)
```

Arguments

object	An R/qtl object object with any class structure.
gc	A data frame of genotype clone infomation usually from a call to genClones (see Details).
id	Character string defining the column of object$pheno containing the genotype names.

`consensus` A logical value. If `TRUE` then consensus genotypes will be calculated for each clonal group by intelligently collapsing alleles for each marker (see Details). If `FALSE` then for each clonal group the genotype with the least missing alleles across the genome will be retained and the remaining genotypes from each group will be removed.

Details

This function provides a very efficient way of dealing with genotype clones in a genetic marker set. This function can be used at any stage of the map construction process as it retains linkage group and marker position information.

The `gc` argument needs to be a data frame of clone information and is easily obtained from a call to `genClones`. If this function is not used then the data frame must contain at least three columns with the first two columns named `"G1"` and `"G2"` containing the pairs of genotypes that are clones and a `"group"` column that indicates the clonal group the pairs of genotypes belongs to.

If `consensus = TRUE` then the function will intelligently collapse the alleles for each marker to form a consensus genotype. Specifically, the allele value will remain unchanged when there are observed allele values across all genotypes in the clone group. For cases where there are missing alleles for some but not all of the genotypes, the consensus genotype will be given the common allele value from the genotypes that contained observed allele values. If there is more than one unique allele value across the genotypes for any marker then it is set to missing.

Value

The cross object is returned with identical class structure as the imputted cross object.

Author(s)

Julian Taylor

See Also

`comparegeno` and `genClones`

Examples

```
data(mapDH, package = "ASMap")

gc <- genClones(mapDH)
mapDHf <- fixClones(mapDH, gc$cgd, consensus = TRUE)
```

genClones *Find and report genotype clones*

Description

Find and report genotype clones for R/qtl objects.

Usage

```
genClones(object, chr, tol = 0.9, id = "Genotype")
```

Arguments

object	An R/qtl object object with any class structure.
chr	A character string of linkage group names.
tol	Pairs of genotypes with a proporion of matching alleles above this tolerance will be returned.
id	Character string defining the column of object$pheno containing the genotype names.

Details

This function extends the functionality of comparegeno in the **qtl** package by providing breakdown statistics for the pairs of genotypes that have a proportion of matching alleles above tol.

Value

A list is returned with the matrix from comparegeno as an element cgm and the breakdown statistics for returned genotype pairs in cgd. Specifically, the statistics contain a "group" column which determines the clonal group the pair of genotypes belongs to.

Author(s)

Julian Taylor

See Also

comparegeno

Examples

```
data(mapDH, package = "ASMap")

gc <- genClones(mapDH)
```

heatMap *Heat map of the estimated pairwise recombination fractions and LOD linkage between markers.*

Description

Heat map of the estimated pairwise recombination fractions and LOD linkage between markers that provides extended functionality of Bromans R/qtl `plot.rf` function.

Usage

```
heatMap(x, chr, mark, what = c("both", "lod", "rf"), lmax = 12,
        rmin = 0, markDiagonal = FALSE, color = rev(rainbow(256, start =
        0, end = 2/3)), ...)
```

Arguments

`x`	A `"cross"` object generated from the R/qtl package.
`chr`	A character string of linkage group names to subset the cross object.
`mark`	An argument to subset linkage groups further into marker subsets. This can be a single numerical vector of markers positions which will subset all linkage groups in the same manner. Or it may be a list of numerical vectors named by the linkage group names with which to subset the linkage groups separately.
`what`	A character string of either `"lod"`, `"rf"` or `"both"`. If `"lod"` only pairwise LOD scores between markers are plotted. If `"rf"` then only pairwise recombination fractions between markers are plotted. If `"both"` then both are plotted with LOD on the lower triangle and recombination in the upper triangle. This is the default (see Details).
`lmax`	The threshold LOD score to implemented. Scores above this threshold will be plotted at the same colour.
`rmin`	The threshold recombination fraction to be implemented. Recombination fractions below this threshold will be plotted at the same colour.
`markDiagonal`	Logical value. If `TRUE` then borders are added around the diagonal elements of the heat map.
`color`	The colour spectrum used to display the heat map. The default is the first two thirds of `rainbow()` (see Details).
`...`	There are additional features available through this argument that can be used to customize the heatmap (see Details).

Details

This function is a rewrite of Bromans **qtl** package function `plot.rf` that provides extended functionality. When `what = "lod"` is chosen the pairwise LOD linkage between markers is displayed on the heat map with a legend on the right hand side spanning zero to `lmax` across the `color` spectrum. If `what = "rf"` the pairwise estimated recombination fractions are displayed on the heat map

with a legend on the right hand side spanning `rmin` to 0.5 across the `color` spectrum. The legend also extends past 0.5 to display estimated recombination fractions between 0.5 and one through a colour spectrum of the maximum `color` value to white. This functionality now gives users the ability to detect markers that may be problematic or possibly out of phase. For `what = "both"` the pairwise LOD linkage is displayed on the lower triangle of the heat map and the pairwise estimated recombination fractions are displayed on the upper triangle. If this option is chosen, legends are displayed for both components of the heat map.

The default colour spectrum is the first two thirds of `rainbow()`. This colour spectrum matches the one in `plot.rf` and provides an aesthetically pleasing palette for the diagnosis of pairwise linkage between markers. Specifically, the color spectrum displays weak linkage and/or low recombination between markers as blue or "cool" areas and strong linkage and/or recombination between markers are shown as red or "hot" areas.

Much of the extra functionality of this function comes from the use of `image.plot` in the **fields** package. This function allows the partitioning of the plotting region into a `bigplot` region for the heat map and a `smallplot` region for the legend. This is called twice when `what = "both"`. The size of the regions can be manipulated by passing the `bigplot` or `smallplot` arguments to the function but it is advised to use the defaults. Further manipulation of the heat map can achieved by passing other arguments of the function `image.plot`. Users should consult the help file for `image.plot` for more details. It should be noted that the argument `legend.args` needs to be avoided as it used in this function.

Value

A heat map is displayed on the current plotting device.

Author(s)

Julian Taylor

See Also

`plot.rf`

Examples

```
data(mapDH, package = "ASMap")

## bulking linkage groups and reconstructing entire linkage map

test1 <- mstmap(mapDH, bychr = FALSE, dist.fun = "kosambi", trace = FALSE)

## plot heat map of result

heatMap(test1, lmax = 30)
```

mapDH *A constructed linkage map for a doubled haploid wheat population*

Description

A constructed linkage map for a doubled haploid wheat population in the form of a constructed R/qtl object.

Usage

```
data(mapDH)
```

Format

This data relates to a fully constructed linkage map of 599 markers genotyped on 218 individuals. The linkage map consists of 23 linkage groups spanning the whole genome. Map distances were estimated using the Hidden Markov algorithm in `read.cross` with the `"kosambi"` mapping function. The map was originally constructed with MultiPoint and curated with MapManager and R/qtl. The data is in R/qtl format with a class structure `c("bc","cross")`. See `read.cross` documentation for more details on the format of this object.

Examples

```
data(mapDH, package = "ASMap")
```

mapDHu *An unconstructed linkage map for a doubled haploid wheat population*

Description

An unconstructed linkage map for a doubled haploid wheat population in the form of a constructed R/qtl object.

Usage

```
data(mapDHu)
```

Format

This data is the unconstructed version of mapDH and consists of 620 markers spanning one large linkage group genotyped on 218 individuals. 584 markers are from the original map and have been randomized within the linkage group. In addition, the unconstructed map contains 12 markers that co-locate with other markers, 12 markers with excessive segregation distortion and 12 markers with excessive missing values. These extra markers have been artificially placed in the object for the purpose of highlighting features of the package.

Examples

```
data(mapDHu, package = "ASMap")
```

`mapF2` *Simulated constructed linkage map for a self pollinated F2 barley population*

Description

Simulated constructed linkage map for a self pollinated F2 barley population in the form of an R/qtl object.

Usage

```
data(mapF2)
```

Format

This data relates to a fully constructed linkage map of 700 simulated markers genotyped on 250 individuals. The map consists of 7 linkage groups, each contaning 100 markers spanning an approximate linkage group length of 200cM. The map was constructed using `mstmap.cross` from the **ASMap** package and map distances were estimated using the `"kosambi"` mapping function. The data is in R/qtl format with a class structure `c("bcsft","cross")`.

Examples

```
data(mapF2, package = "ASMap")
```

`mergeCross` *Merge linkage groups of an R/qtl cross object*

Description

Merges linkage groups of an R/qtl cross object from a user specified list.

Usage

```
mergeCross(cross, merge = NULL, gap = 5)
```

Arguments

cross	An R/qtl cross object with any class structure.
merge	A list with elements containing the linkage groups to be merged with each element named by the proposed linkage group name (see Examples).
gap	The cM gap to put between the merged map elements in the complete linkage group.

Details

This merging function allows you to perform multiple merges of two or more linkage groups in one call. User should ensure linkage group names are correct and that proposed linkage group names do not already exist.

Value

The cross object is returned with identical class structure as the inputted cross object. The "geno" element should now contain merged linkage groups for the user defined merges.

Author(s)

Julian Taylor

See Also

breakCross

Examples

```
data(mapDH, package = "ASMap")

mapDH1 <- breakCross(mapDH, split = list("4A" = "4A.m.8"))
pull.map(mapDH1)[["4A.1"]]
pull.map(mapDH1)[["4A.2"]]

mapDH2 <- mergeCross(mapDH1, merge = list("4A" = c("4A.1","4A.2")))
pull.map(mapDH2)[["4A"]]
```

---

mstmap.cross *Extremely fast linkage map construction for R/qtl objects using MSTmap.*

Description

Extremely fast linkage map construction for R/qtl objects utilizing the source code for MSTmap (see Wu et al., 2008). The construction includes linkage group clustering, marker ordering and genetic distance calculations.

Usage

```
## S3 method for class 'cross'
mstmap(object, chr, id = "Genotype", bychr = TRUE,
       suffix = "numeric", anchor = FALSE, dist.fun = "kosambi",
       objective.fun = "COUNT", p.value = 1e-06, noMap.dist = 15,
       noMap.size = 0, miss.thresh = 1, mvest.bc = FALSE, detectBadData =
       FALSE, return.imputed = FALSE, trace = FALSE, ...)
```

Arguments

object	A "cross" object generated from the R/qtl package. Specifically the object needs to inherit from one of the following classes "bc", "dh","riself","bcsft" (see Details).
chr	A character string of linkage group names that require re-construction and/or optimal ordering of the markers they contain. (see Details).
id	The name of the column in object$pheno that uniquely identifies the genotype names. Default is "Genotype".
bychr	Logical value. For a given set of linkage groups defined by chr, if TRUE then split linkage groups (only if required, see p.value) and order markers within linkage groups. If FALSE then combine linkage groups and reconstruct. Default is TRUE.
suffix	Character string either "numeric" or "alpha" determining whether numeric or alphabetic ascending values are post-fixed to linkage group names when splitting linkage groups.
anchor	Logical value. The MSTmap algorithm does not respect the inputted marker order of the linkage map required for construction. For a given set of linkage groups defined by chr, if TRUE the order of the inputted markers is respected regardless of the choices of chr and bychr. Default is FALSE.
dist.fun	Character string defining the distance function used for calculation of genetic distances. Options are "kosambi" and "haldane". Default is "kosambi".
objective.fun	Character string defining the objective function to be used when constructing the map. Options are "COUNT" for minimising the sum of recombination events between markers and "ML" for maximising the likelihood objective function. Default is "COUNT".
p.value	Numerical value to specify the threshold to use when constructing linkage groups. Defaults to 1e-06. If a value greater than one is given this feature is turned off and it is assumed that all marker data inputted belong to the same linkage group (see Details).
noMap.dist	Numerical value to specify the smallest genetic distance a set of isolated markers can appear distinct from other linked markers. Isolated markers will appear in their own linkage groups and will be of size specified by noMap.size.
noMap.size	Numerical value to specify the maximum size of isolated marker linkage groups that have been identified using noMap.dist. This feature can be turned off by setting it to 0. Default is 0.

miss.thresh	Numerical value to specify the threshold proportion of missing marker scores allowable in each of the markers. Markers above this threshold will not be included in the linkage map. Default is 1.
mvest.bc	Logical value. If TRUE missing markers will be imputed before clustering the markers into linkage groups. This is restricted to "bc","dh","riself" populations only (see Details). Default is FALSE.
detectBadData	Logical value. If TRUE possible genotyping errors are detected, set to missing and then imputed as part of the marker ordering algorithm. Genotyping errors will also be printed in the file specified by trace. This is restricted to "bc","dh","riself" populations only. (see Details). Default is FALSE.
return.imputed	Logical value. If TRUE then the imputed marker probability matrix is returned for the linkage groups that are constructed (see Details). Default is FALSE.
trace	An automatic tracing facility. If trace = FALSE then minimal MSTmap output is piped to the screen during the algorithm. If trace = TRUE, then detailed output from MSTmap is piped to "MSToutput.txt". This file is equivalent to the output that would be obtained from running the MSTmap executable from the command line.
...	Currently ignored.

Details

The **qtl** cross object needs to inherit one of the allowable classes "bc","dh","riself","bcsft". This provides a safeguard against attempts to construct a map for more complex populations that can exist in **qtl**. Users should be aware when doubled haploid populations are read in using read.cross() from the **qtl** package they inherit the class "bc". Users can apply the class "dh" by simply changing the class of the object. For the purpose of linkage map construction the classes "bc" and "dh" will provide equivalent results.

MSTmap supports "RILn" populations, where n is the number of generations of selfing. Markers in these populations are required to be fully informative i.e. contain 3 distinct allele types such as AA, BB for parental homozygotes and AB for phase unknown heterozygotes. If read.cross is used to import the "RILn" population the resultant object will initially be given a class "f2". The level of selfing would then have to be encoded into the object by applying one of the two conversion functions available in the **qtl** package. For a population that has been generated by selfing n times the conversion function convert2bcsft can be used by setting the arguments F.gen = n and BC.gen = 0. Populations that are genuine advanced RILs can be converted using the convert2riself function.

This method function is designed to be an "all-in-one" function that will allow you to construct linkage maps extremely fast in multiple different ways from the supplied cross object. Initially, the map can be kept complete or a subset of selected linkage groups can be chosen using the chr argument. Setting bychr = FALSE will bulk the marker information for the selected linkage groups and, if necessary, form new linkage groups and optimise the marker order within each. Setting bychr = TRUE will ensure that markers are optimally ordered within each linkage group. This will also break linkage groups depending on the p-value given in the call (see below for details of the use of p.value). If the linkage map was initially subsetted, the linkage groups not involved in the subset are returned to ensure the map is complete.

The algorithm allows an adjustment of the p.value threshold for clustering of markers to distinct linkage groups (see Wu et al., 2008) and is highly dependent on the number of individuals in the

population. As the number of individuals increases the p.value threshold should be decreased accordingly. This may require some trial and error to achieve desired results. When bychr = TRUE, established linkage groups may also split depending on the p.value given. To prevent this the p.value threshold may be increased to a desired value or the splitting may be prevented altogether by supplying a value greater than one to this argument.

If mvest.bc = TRUE and the population type is "bc","dh","riself" then missing values are imputed before markers are clustered into linkage groups. This is only a simple imputation that places a 0.5 probability of the missing observation being one allele or the other and is used to assist the clustering algorithm when there is known to be high numbers of missing observations between pairs of markers.

It should be highlighted that for population types "bc","dh","riself", imputation of missing values occurs regardless of the value of mv.est.bc. This is achieved using an EM algorithm that is tightly coupled with marker ordering (see Wu et al., 2008). Initially a marker order is obtained omitting missing marker scores and then imputation is performed based on the underlying recombinant probabilities of the flanking markers with the markers containing the missing value. The recombinant probabilities are then recomputed and an update of the pairwise distances are calculated. The ordering algorithm is then run again and the complete process is repeated until convergence. Note, the imputed probability matrix for the linkage map being constructed is returned if return.imputed = TRUE.

For populations "bc","dh","riself", if detectBadData = TRUE the marker ordering algorithm also includes the detection of genotyping errors. For any individual genotype, the detection method is based on a weighted Euclidean metric (see Wu et al., 2008) that is a function of the recombination probabilities of all the markers with the marker containing the suspicious observation. Any genotyping errors detected are set to missing and the missing values are then imputed as part of the marker ordering algorithm. Note, the detection of these errors and their amendment can be returned in the imputed probability matrix if return.imputed = TRUE.

If return.imputed = TRUE and the object has class "bc","dh","riself" then the marker probability matrix is returned for the linkage groups that have been constructed using the algorithm. Each linkage group is named identically to the linkage groups of the map and contains an ordered "map" element and a "data" element consisting of marker probabilities of the A allele being present (i.e. P(A) = 1, P(B) = 0). Both elements contain a possibly reduced version of the marker set that includes all non-colocating markers as well as the first marker of any set of co-locating markers.

Value

The function returns a cross object with an identical class structure to the cross object inputted. The object is a list with usual components "pheno" and "geno". If markers were omitted for any reason during the construction, the object will have an "omit" component with all omitted markers in a collated matrix. If return.imputed = TRUE then the object will also contain an "imputed.geno" element.

Author(s)

Julian Taylor, Dave Butler, Timothy Close, Yonghui Wu, Stefano Lonardi

References

Y. Wu, P. Bhat, T.J. Close, S. Lonardi, Efficient and Accurate Construction of Genetic Linkage Maps from Minimum Spanning Tree of a Graph Plos Genetics, Volume 4, Issue 10, 2008.

See Also

mstmap.data.frame and breakCross

Examples

```
data(mapDH, package = "ASMap")

## bulking linkage groups and reconstructing entire linkage map

test1 <- mstmap(mapDH, bychr = FALSE, dist.fun = "kosambi", trace = FALSE)
pull.map(test1)

## one linkage group at a time (possibly break established linkage
## groups)

test2 <- mstmap(mapDH, bychr = TRUE, dist.fun = "kosambi", trace = FALSE)
pull.map(test2)

## one linkage group at a time (do not break established linkage groups)

test3 <- mstmap(mapDH, bychr = TRUE, dist.fun = "kosambi", p.value = 2, trace = FALSE)
pull.map(test3)

## impute before clustering and detect genotyping errors, pipe output to file

test4 <- mstmap(mapDH, bychr = FALSE, dist.fun = "kosambi", trace = TRUE,
         mvest.bc = TRUE, detectBadData = TRUE)
pull.map(test4)
```

---

mstmap.data.frame *Extremely fast linkage map construction for data frame objects using MSTmap.*

Description

Extremely fast linkage map construction for data frame objects utilizing the source code for MSTmap (see Wu et al., 2008). The construction includes linkage group clustering, marker ordering and genetic distance calculations.

Usage

```
## S3 method for class 'data.frame'
mstmap(object, pop.type = "DH", dist.fun = "kosambi",
      objective.fun = "COUNT", p.value = 1e-06, noMap.dist = 15,
      noMap.size = 0, miss.thresh = 1, mvest.bc = FALSE, detectBadData = FALSE,
      as.cross = TRUE, return.imputed = TRUE, trace = FALSE, ...)
```

Arguments

object	A "data.frame" object containing marker information. The data.frame must explicitly be arranged with markers in rows and genotypes in columns. Marker names are obtained from the rownames of the object and genotype names are obtained from the names component of the object (see Details).
pop.type	Character string specifying the population type of the data frame object. Accepted values are "DH" (doubled haploid), "BC" (backcross), "RILn" (non-advanced RIL population with n generations of selfing) and "ARIL" (advanced RIL) (see Details). Default is "DH".
dist.fun	Character string defining the distance function used for calculation of genetic distances. Options are "kosambi" and "haldane". Default is "kosambi".
objective.fun	Character string defining the objective function to be used when constructing the map. Options are "COUNT" for minimising the sum of recombination events between markers and "ML" for maximising the likelihood objective function. Default is "COUNT".
p.value	Numerical value to specify the threshold to use when constructing linkage groups. Defaults to 1e-06. If a value greater than one is given this feature is turned off and it is assumed that all marker data inputted belong to the same linkage group (see Details).
noMap.dist	Numerical value to specify the smallest genetic distance a set of isolated markers can appear distinct from other linked markers. Isolated markers will appear in their own linkage groups ad will be of size specified by noMap.size.
noMap.size	Numerical value to specify the maximum size of isolated marker linkage groups that have been identified using noMap.dist. This feature can be turned off by setting it to 0. Default is 0.
miss.thresh	Numerical value to specify the threshold proportion of missing marker scores allowable in each of the markers. Markers above this threshold will not be included in the linkage map. Default is 1.
mvest.bc	Logical value. If TRUE missing markers will be imputed before clustering the markers into linkage groups. This is restricted to "BC","DH","ARIL" populations only (see Details).
detectBadData	Logical value. If TRUE possible genotyping errors are detected, set to missing and then imputed as part of the marker ordering algorithm. Genotyping errors will also be printed in the file specified by trace. This is restricted to "BC","DH","ARIL" populations only. (see Details). Default is FALSE.
as.cross	Logical value. If TRUE the constructed linkage map is returned as a R/qtl cross object (see Details). If FALSE then the constructed linkage map is returned as a

	data.frame with extra columns indicating the linkage group, marker name/position and genetic distance. Default is TRUE.
return.imputed	Logical value. If TRUE then the imputed marker probability matrix is returned for the linkage groups that are constructed (see Details). Default is FALSE.
trace	An automatic tracing facility. If trace = FALSE then minimal MSTmap output is piped to the screen during the algorithm. If trace = TRUE, then detailed output from MSTmap is piped to "MSToutput.txt". This file is equivalent to the output that would be obtained from running the MSTmap executable from the command line.
...	Currently ignored.

Details

The data frame object must have an explicit format with markers in rows and genotypes in columns. The marker names are required to be in the rownames component and the genotype names are required to be in the names component of the object. In each set of names there must be no spaces. If spaces are detected they are exchanged for a "-". Each of the columns of the data frame must be of class "character" (not factors). If converting from a matrix, this can easily be achieved by using the stringAsFactors = FALSE argument for any data.frame method.

It is important to know what population type the data frame object is and to correctly input this into pop.type. If pop.type = "ARIL" then it is assumed that the minimal number of heterozygotes have been set to missing before proceeding. The advanced RIL population is then treated like a backcross population for the purpose of linkage map construction. Genetic distances are adjusted post construction. For non-advanced RIL populations pop.type = "RILn", the number of generations of selfing is limited to 20 to ensure sensible input.

The content of the markers in object can either be all numeric (see below) or all character. If markers are of type character then the following allelic content must be explicitly adhered to. For pop.type "BC", "DH" or "ARIL" the two allele types should be represented as ("A" or "a") and ("B" or "b"). For non-advanced RIL populations (pop.type = "RILn") phase unknown heterozygotes should be represented as "X". For all populations, missing marker scores should be represented as ("U" or "-").

This function also extends the functionality of the MSTmap algorithm by allowing users to input a complete numeric data frame of marker probabilities for pop.type "BC", "DH" or "ARIL". The values must be inclusively between 1 (A) and 0 (B) and be representative of the probability that the A allele is present. No missing values are allowed.

The algorithm allows an adjustment of the p.value threshold for clustering of markers to distinct linkage groups (see Wu et al., 2008) and is highly dependent on the number of individuals in the population. As the number of individuals increases the p.value threshold should be decreased accordingly. This may require some trial and error to achieve desired results.

If mvest.bc = TRUE and the population type is "BC","DH","ARIL" then missing values are imputed before markers are clustered into linkage groups. This is only a simple imputation that places a 0.5 probability of the missing observation being one allele or the other and is used to assist the clustering algorithm when there is known to be high numbers of missing observations between pairs of markers.

It should be highlighted that for population types "BC","DH","ARIL", imputation of missing values occurs regardless of the value of mv.est.bc. This is achieved using an EM algorithm that is

tightly coupled with marker ordering (see Wu et al., 2008). Initially a marker order is obtained omitting missing marker scores and then imputation is performed based on the underlying recombinant probabilities of the flanking markers with the markers containing the missing value. The recombinant probabilities are then recomputed and an update of the pairwise distances are calculated. The ordering algorithm is then run again and the complete process is repeated until convergence. Note, the imputed probability matrix for the linkage map being constructed is returned if `return.imputed = TRUE`.

For populations `"BC"`,`"DH"`,`"ARIL"`, if `detectBadData = TRUE`, the marker ordering algorithm also includes the detection of genotyping errors. For any individual genotype, the detection method is based on a weighted Euclidean metric (see Wu et al., 2008) that is a function of the recombination probabilities of all the markers with the marker containing the suspicious observation. Any genotyping errors detected are set to missing and the missing values are then imputed if `mv.est = TRUE`. Note, the detection of these errors and their amendment is returned in the imputed probability matrix if `return.imputed = TRUE`

If `as.cross = TRUE` then the constructed object is returned as a R/qtl cross object with the appropriate class structure. For `"RILn"` populations the constructed object is given the class `"bcsft"` by using the **qtl** package conversion function `convert2bcsft` with arguments `F.gen = n` and `BC.gen = 0`. For `"ARIL"` populations the constructed object is given the class `"riself"`.

If `return.imputed = TRUE` and `pop.type` is one of `"BC"`,`"DH"`,`"ARIL"`, then the marker probability matrix is returned for the linkage groups that have been constructed using the algorithm. Each linkage group is named identically to the linkage groups of the map and, if `as.cross = TRUE`, contains an ordered `"map"` element and a `"data"` element consisting of marker probabilities of the A allele being present (i.e. P(A) = 1, P(B) = 0). Both elements contain a possibly reduced version of the marker set that includes all non-colocating markers as well as the first marker of any set of co-locating markers. If `as.cross = FALSE` then an ordered data frame of matrix probabilities is returned.

Value

If `as.cross = TRUE` the function returns an R/qtl cross object with the appropriate class structure. The object is a list with usual components `"pheno"` and `"geno"`. If `as.cross = FALSE` the function returns an ordered data frame object with additional columns that indicate the linkage group, the position and marker names and genetic distance of the markers within in each linkage group. If markers were omitted for any reason during the construction, the object will have an `"omit"` component with all omitted markers in a collated matrix. If `return.imputed = TRUE` then the object will also contain an `"imputed.geno"` element.

Author(s)

Julian Taylor, Dave Butler, Timothy Close, Yonghui Wu, Stefano Lonardi

References

Y. Wu, P. Bhat, T.J. Close, S. Lonardi, Efficient and Accurate Construction of Genetic Linkage Maps from Minimum Spanning Tree of a Graph Plos Genetics, Volume 4, Issue 10, 2008.

See Also

`mstmap.cross`

Examples

```
data(mapDH, package = "ASMap")

## forming data frame object from R/qtl object

dfg <- t(do.call("cbind", lapply(mapDH$geno, function(el) el$data)))
dimnames(dfg)[[2]] <- as.character(mapDH$pheno[["Genotype"]])
dfg <- dfg[sample(1:nrow(dfg), nrow(dfg), replace = FALSE),]
dfg[dfg == 1] <- "A"
dfg[dfg == 2] <- "B"
dfg[is.na(dfg)] <- "U"
dfg <- cbind.data.frame(dfg, stringsAsFactors = FALSE)

## construct map

testd <- mstmap(dfg, dist.fun = "kosambi", trace = FALSE)
pull.map(testd)

## let's get a timing on that ...

system.time(testd <- mstmap(dfg, dist.fun = "kosambi", trace = FALSE))
```

---

pp.init *Parameter initialization function*

Description

Parameter initialization function for `pushCross` and `pullCross`

Usage

```
pp.init(seg.thresh = 0.05, seg.ratio = NULL, miss.thresh = 0.1, max.rf =
        0.25, min.lod = 3)
```

Arguments

`seg.thresh`	Numerical value between zero and one determining the p-value threshold for the test of marker segregation distortion.
`seg.ratio`	A character string of the form "AA:BB" or "AA:AB:BB" describing the ratio of the alleles.
`miss.thresh`	Numerical value between zero and one determining the proportion of missing values.
`max.rf`	The maximum recombination fraction to consider when attempting to cluster pushed markers back into linkage groups.
`min.lod`	The minimum LOD score to consider when attempting to cluster pushed markers back into linkage groups.

Details

This parameter initialization function is used by the function `pullCross` to pull markers from a linkage map and `pushCross` to push markers back into a linkage map. How the arguments `seg.thresh`, `seg.ratio` and `miss.thresh` are used depends on which function is called. See `pushCross` and `pullCross` for more details.

Value

Return user defined parameter values for each of the parameters.

Author(s)

Julian Taylor

See Also

`pushCross`; `pullCross`

Examples

```
data(mapDH, package = "ASMap")

## pull markers from a linkage map with a segregation distortion

pars <- pp.init(seg.thresh = 0.05)
mapDH.s <- pullCross(mapDH, type = "seg.distortion", pars = pars)
mapDH.s$seg.distortion$table
```

`profileGen` *Profile individual genotype statistics for an R/qtl cross object*

Description

Profile individual genotype statistics for the current linkage map order of and R/qtl cross object

Usage

```
profileGen(cross, chr, bychr = TRUE, stat.type = c("xo", "dxo", "miss"),
           id = "Genotype", xo.lambda = NULL, ...)
```

Arguments

`cross`	An R/qtl `cross` object with class structure `"bc"`, `"dh"`,`"riself"`,`"bcsft"`. (see `?mstmap.cross` for more details.)
`chr`	Character vector of linkage group names used for subsetting the linkage map.
`bychr`	Logical vector determining whether statistics should be plotted by chromosome (see Details).

stat.type	Character string of any combination of "xo" or "dxo" or "miss". "xo" calculates the number of crossovers, "dxo" calculates the number of double crossover and "miss" calculates the number of missing values.
id	Character string determining the column of cross$pheno that contains the genotype names.
xo.lambda	A numerical value for the expected rate of recombination. (see Details).
...	Other arguments to be passed to the high level lattice plot.

Details

This function uses statGen to profile statistics for the genotypes for the current order of the linkage map. Any combination of "xo" or "dxo" or "miss" may be given to simultaneous plot. If bychr = TRUE then the plots will be further partitioned by linkage groups given by chr.

If a numerical value is given for xo.lambda then the recombination count for each genotype is tested against the expected recombination rate xo.lambda using a simple one-tailed test of a Poisson mean. Any lines that have a p-value less than than a family wise error rate based on bonferroni adjustment of the usual alpha level of 0.05 are annotated on the profiles being plotted.

Value

A lattice panel plot with panels described by the stat.type given in the call and genotype statistics are returned invisibly. If xo.lambda is not NULL then these statistics also include a logical vector named "xo.lambda" that is returned from testing the individuals for inflated recombination rates (see Details).

Author(s)

Julian Taylor

See Also

statGen

Examples

```
data(mapDH, package = "ASMap")

## profile all genotype crossover and double crossover statistics

profileGen(mapDH, stat.type = c("xo","dxo"), xo.lambda = 25)
```

`profileMark` *Profile individual marker and interval statistics for an R/qtl cross object*

Description

Graphically profile individual marker and interval statistics for an R/qtl cross object

Usage

```
profileMark(cross, chr, stat.type = "marker", use.dist = TRUE,
          map.function = "kosambi", crit.val = NULL, display.markers = FALSE,
          mark.line = FALSE, ...)
```

Arguments

`cross`	An R/qtl `cross` object with class structure `"bc"`, `"dh"`,`"riself"`,`"bcsft"`. (see `?mstmap.cross` for more details.)
`chr`	Character vector of linkage group names used for subsetting the linkage map.
`stat.type`	Character string of either `"marker"` or `"interval"` or both. Also this can be a set of character strings relating to individual marker or interval statistics that want to be viewed simultaneously (see Details).
`use.dist`	Logical value determining whether the actual map distances should be use to represent marker positions. If `FALSE` then markers are placed equidistant from each other.
`map.function`	Character string of either `"koasmbi"`, `"haldane"`, `"morgan"` or `"cf"` defining the map function to be used for interval related statistics.
`crit.val`	The critical value to be used in displaying marker or intervals above a certain threshold (see Details).
`display.markers`	A logical value determining whether marker names should be displayed on the bottom axis.
`mark.line`	A logical value determining whether vertical lines should be drawn at marker positions. This may be useful to line up marker positions across several plots.
`...`	Other arguments to be passed to the high level lattice plot.

Details

This graphical function calls the function `statMark` to retrieve marker and interval statistics. If `"marker"` is given as the `stat.type` then the complete set of marker statistics is plotted simultaneously. If `"interval"` is given as the `stat.type` then the function simultaneously plots the complete set of interval statistics. Both can also be chosen.

This function also allows users to choose any combination of marker or interval statistics they would like to view. The set of available marker statistics that can be profiled are given below

- "seg.dist": Profile the -log10 p-value. results from a test of segregation distortion for each marker.
- "miss": Profile the proportion of missing values for each marker.
- "prop": Profile the allele proportions for each marker.
- "dxo": Profile the number of double crossovers occurring at each marker.

The set of available interval statistics that can be profiled are given below

- "erf": Profile the recombination fractions for the intervals.
- "lod": Profile the LOD score for the test of no linkage between markers in an interval.
- "dist": Profile the interval map distance taken from the map component of each linkage group.
- "mrf": Profile the map recombination fraction for the intervals.
- "recomb": Profile the actual number of recombinations within each of the intervals.

If crit.val="bonf" and marker statistics are plotted then any markers that have p-value for the test of segregation distortion less than the family wise error rate based on a bonferroni adjustment of the usual 0.05 alpha level, are annotated on each of the marker plots. If any interval statistics are being plotted then any intervals that have a p-value for the test of no linkage that is less than a bonferroni adjustment of the usual 0.05 alpha level are annotated on each of the interval statistics plots.

Value

A lattice panel plot is displayed with panels described by the stat.type given in the call and the complete marker/interval statistics are returned invisibly. If crit.val is not NULL then both the marker/interval statistics are returned with an extra logical column called "crit.val" from testing markers for segregation distortion and intervals for weak linkage (see Details).

Author(s)

Julian Taylor

See Also

profileMark

Examples

```
data(mapDH, package = "ASMap")

## profile chosen statistics

profileMark(mapDH, stat.type = c("seg.dist","prop","erf"))
```

`pullCross` *Pull markers from a linkage map.*

Description

Pull markers of a certain type from a linkage map and place them aside in the R/qtl object and, if appropriate, keeping their connections with the reduced linkage map.

Usage

```
pullCross(object, chr, type = c("co.located","seg.distortion","missing"),
          pars = NULL, replace = FALSE, ...)
```

Arguments

`object`	An R/qtl `cross` object with class structure `"bc"`, `"dh"`,`"riself"`,`"bcsft"`. (see `?mstmap.cross` for more details.)
`chr`	A character vector of linkage group names with which to subset the linkage map before pulling any markers.
`type`	A character string determining the type of markers to be pulled from the map (see Details).
`pars`	A list of parameters that are used by `pullCross` to pull markers of certain type. The default NULL calls the parameter initialization function `pp.init` with defaults (see Details and Examples).
`replace`	A logical value determining whether the markers and summary of marker information that is pulled from the map replaces information that is already residing in the `type` element of the object.
`...`	Currently ignored.

Details

This function gives users the ability to "pull" markers of several different types from the linkage map and place them in appropriately named elements of the cross object. These elements can be examined by the user and can even be "pushed" back using the complementary command `pushCross`.

Currently supported types are:

- `type = "co.located"`. This type gives the user the ability to reduce a linkage map to a unique set of markers for the purpose of efficient map construction. Co-located markers are pulled from the linkage map using the technology of `findDupMarkers` from the **qtl** package and places them aside in a separate list element called `"co.located"`. This element contains the removed marker data as well as a table that displays the connections between the co-located markers with markers that remain in the linkage map. If required, this table is used by `pushCross` to "push" the co-located markers back into the linkage map.

- `type = "seg.distortion"`. Users can pull markers with segregation distortion from a linkage map with two different thresholding mechanisms called using `pars`. If the list argument `pars` is used with an element called `seg.thresh` then markers are pulled from the map if the p-value from the test for segregation distortion is LESS than `seg.thresh`. Values of `seg.thresh` must be between 0 and 1. If `pars` contains an element `seg.ratio` then markers are pulled from the map based on the ratio provided. The ratio must be in character format and of the type "AA:BB" for two allele populations and "AA:AB:BB" for three allele populations (see Examples for more details). Markers are pulled if their allele proportions are GREATER than the largest proportional ratio or LESS than the smallest proportional ratio given in `seg.thresh`. If neither thresholding mechanisms are given then the default is to use `seg.thresh = 0.05`. If markers are found matching the above criteria they are pulled from the linkage map and placed aside in an element called `"seg.distortion"`. This element contains the removed distorted marker data as well as a table summarizing each of the markers. See examples below for more detail.
- `type = "missing"`. Users can pull markers with a proportional amount of missing allele scores. If `pars` contains an element `miss.thresh` then markers are pulled from the linkage map that have a proportion of missing values GREATER than `miss.thresh`. If no value is given for `miss.thresh` then it defaults to 0.1 or 10% missing values. If markers are found matching the above criteria they are pulled from the map and are placed aside in an separate list element called `"missing"`. This element contains the removed marker data as well as a table summarizing each of the markers. See examples below for more detail.

Value

The cross object is returned with identical class structure as the inputted cross object and an additional elements corresponding to the marker types being pulled from the map.

Author(s)

Julian Taylor

See Also

pushCross

Examples

```
data(mapDH, package = "ASMap")

## pull co-located markers from linkage map

mapDH.c <- pullCross(mapDH, type = "co.located")
mapDH.c$co.located$table
```

pushCross *Push markers into an established R/qtl linkage map.*

Description

Push unlinked markers or markers that were originally placed aside by pullCross back into linkage groups of an established R/qtl linkage map.

Usage

```
pushCross(object, type = c("co.located","seg.distortion","missing","unlinked"),
       unlinked.chr = NULL, pars = NULL, ...)
```

Arguments

object	An R/qtl cross object with class structure "bc", "dh","riself","bcsft". (see ?mstmap.cross for more details.)
type	A character string determining the type of markers to be pushed into the linkage map (see Details).
unlinked.chr	A character string of linkage group names containing markers that require pushing into the remaining linkage groups of the object. This is only useful when type="unlinked". Default is NULL.
pars	A list of parameters that are used by pushCross to push markers a certain type back into the linkage group. The default NULL calls the parameter initialization function pp.init with defaults (see Details and Examples).
...	Currently ignored.

Details

This function was written explicitly to complement pullCross by "pushing" markers of certain types back into linkage groups of an established linkage map.

Currently supported marker types are:

- type = "co.located". Users can push co-located markers back into the linkage map that have been set aside in the cross object element co.located. To ensure this can be used at any stage of the linkage map construction process the function disregards the linkage group information provided in the table formed by using pullCross. Instead it uses the current positions of the markers in the reduced linkage map to determine where to push the co-located markers back to.
- type = "seg.distortion". Users can push markers from the object element "seg.distortion" back into a linkage map using the thresholding mechanisms seg.thresh and seg.ratio called using pars. If seg.thresh is given then the markers are pushed back that have p-values that are GREATER than seg.thresh. If pars contains an element seg.ratio then markers are pushed back based on the ratio provided. The ratio must be in character format and of the type "AA:BB" for two allele populations and "AA:AB:BB" for three allele populations (see Examples for more details). Markers are pushed back if their allele proportions

are LESS than the largest proportional ratio or GREATER than the smallest proportional ratio given in `seg.thresh`. If neither thresholding mechanisms are given then the default is to use `seg.thresh = 0.05`.

- `type = "missing"`. Users can push markers from the object element `"missing"` back into the linkage map using the thresholding parameter `miss.thresh` called using `pars`. Markers will be pushed back that have a proportion of missing values LESS than `miss.thresh`. If no value is given for this parameter it defaults to 0.1 or 10% missing values.
- `type = "unlinked"`. Users can push unlinked markers that reside in linkage groups of the established linkage map. If this type is chosen `unlinked.chr` must be a character string of linkage group names in the object.

For types `"seg.distortion"`, `"missing"` and `"unlinked"` a fast clustering method is used to allocate markers to established linkage groups. This is done very efficiently by reducing the constructed linkage map to a skeleton set of markers before checking linkages. How these linkages are formed can be tweaked by setting `max.rf` and `min.lod` when calling `pars`. These currently default to `max.rf = 0.25` and `min.lod = 3`.

User should explicitly avoid the use of "UL" as part of a linkage group name as this is used internally to name unlinked groups of markers if required. It should also be noted that this function does not re-construct the object after allocating markers to linkage groups. For efficient linkage map reconstruction of an R/qtl object see `mstmap.cross()`.

Value

The cross object is returned with an identical class structure as the inputted cross object with additional markers from the marker types pushed into linkage groups of the established linkage map. If all markers of an element type are pushed back then the element type is removed from the object.

Author(s)

Julian Taylor

See Also

`pullCross`

Examples

```
data(mapDH, package = "ASMap")

## pull co-located markers from map

mapDH.c <- pullCross(mapDH, type = "co.located")
mapDH.c$co.located$table

## push co-located markers back into linkage map

mapDH.z <- pushCross(mapDH.c, type = "co.located")
pull.map(mapDH.z)
```

quickEst *Very quick estimation of genetic map distances.*

Description

Very quick estimation of genetic map distances for a constructed R/qtl object

Usage

```
quickEst(object, chr, map.function = "kosambi", ...)
```

Arguments

object	An R/qtl object object with any class structure.
chr	A character string of linkage group names that require (re)estimation of their genetic map distances.
map.function	Character string of either "koasmbi", "haldane", "morgan" or "cf" defining the mapping function to be used.
...	Other arguments passed to argmax.geno.

Details

For linkage groups with large numbers of markers, the Hidden Markov algorithm in est.map can be extremely slow. The computational burden for this algorithm increases as the number of missing values and genotyping errors increase. quickEst circumvents this by using the Viterbi algorithm computationally implemented in argmax.geno of the **qtl** package. Initial conservative estimates of the map distances are calculated from inverting recombination fractions outputted from est.rf. These are then passed to argmax.geno and imputation of missing allele scores is performed along with re-estimation of map distances.

Value

The cross object is returned with identical class structure as the inputted cross object.

Author(s)

Julian Taylor

See Also

est.map

Examples

```
data(mapDH, package = "ASMap")

mapDH1 <- quickEst(mapDH, map.function = "kosambi")
```

---

statGen *Individual genotype statistics for an R/qtl cross object*

Description

Individual genotype statistics for the current linkage map order of and R/qtl cross object

Usage

```
statGen(cross, chr, bychr = TRUE, stat.type = c("xo","dxo","miss"), id = "Genotype")
```

Arguments

cross	An R/qtl cross object with class structure "bc", "dh","riself","bcsft". (see ?mstmap.cross for more details.)
chr	Character vector of linkage group names used for subsetting the linkage map.
bychr	Logical vector determining whether statistics should be plotted by chromosome (see Details).
stat.type	Character string of any combination of "xo" or "dxo" or both. "miss". "xo" calculates the number of crossovers, "dxo" calculates the number of double crossover and "miss" calculates the number of missing values.
id	Character string determining the column of cross$pheno that contains the genotype names.

Details

This function is used in profileGen to plot any combination of returned linkage map statistics on a single graphical display.

Value

A list with elements named by the stat.type used in the call. If bychr = TRUE then each element is a data frame of statistics with columns named by the linkage groups. If bychr = FALSE then each element is a vector of statistics named by the stat.type.

Author(s)

Julian Taylor

See Also

profileGen

Examples

```
data(mapDH, package = "ASMap")

## produce all genotype crossover and double crossover statistics

statGen(mapDH, stat.type = c("xo","dxo"))
```

---

statMark *Individual marker and interval statistics for an R/qtl cross object*

Description

Individual marker and interval statistics for an R/qtl cross object

Usage

```
statMark(cross, chr, stat.type = c("marker","interval"), map.function = "kosambi")
```

Arguments

cross	An R/qtl cross object with class structure "bc", "dh","riself","bcsft". (see ?mstmap.cross for more details.)
chr	Character vector of linkage group names used for subsetting the linkage map.
stat.type	Character string of either "marker" or "interval" or both. "marker" produces individual marker related statistics and "interval" produces interval related statistics for the current map order (see Details).
map.function	Character string of either "koasmbi", "haldane", "morgan" or "cf" defining the map function to be used for interval related statistics.

Details

If "marker" is chosen then a call to geno.table from **qtl** is used to return individual marker statistics for segregation distortion, as well as allele and missing value proportions. For the current map order the number of double crossovers at each marker are also returned.

If "interval" is chosen then interval statistics are returned for the current map order. These include the estimated recombination fraction and LOD score between adjacent markers, calculated from est.rf in **qtl**. Also returned are the map interval distances and converted map recombination fractions extracted from the "map" component of each linkage group as well as the actual number of recombinations between markers.

This function is used in profileMark to plot any combination of returned linkage map statistics on a single graphical display.

Value

A list named by the `stat.type` used in the call. Each element is a data frame of statistics with columns named by the statistic.

Author(s)

Julian Taylor

See Also

`profileMark`

Examples

```
data(mapDH, package = "ASMap")

## produce all statistics

statMark(mapDH, stat.type = c("marker","interval"))
```

Index

∗Topic **datasets**
  mapDH, 11
  mapDHu, 11
  mapF2, 12
∗Topic **graphics**
  heatMap, 9
∗Topic **misc**
  breakCross, 3
  combineMap, 5
  fixClones, 6
  genClones, 8
  mergeCross, 12
  mstmap.cross, 13
  mstmap.data.frame, 17
  pp.init, 21
  profileGen, 22
  profileMark, 24
  pullCross, 26
  pushCross, 28
  quickEst, 30
  statGen, 31
  statMark, 32
∗Topic **package**
  ASMap-package, 2

ASMap *(*ASMap-package*)*, 2
ASMap-package, 2

breakCross, 3, *6*, *13*, *17*

combineMap, 5
comparegeno, *7*, *8*

est.map, *30*

fixClones, 6

genClones, *7*, 8

heatMap, 9

mapDH, *11*, 11
mapDHu, 11
mapF2, 12
mergeCross, *4*, *6*, 12
mstmap *(*mstmap.cross*)*, 13
mstmap.cross, 13, *20*
mstmap.data.frame, *17*, 17

plot.rf, *10*
pp.init, 21
profileGen, 22, *32*
profileMark, 24, *25*, *33*
pull *(*pullCross*)*, 26
pullCross, *22*, 26, *29*
push *(*pushCross*)*, 28
pushCross, *22*, *27*, 28

quickEst, 30

statGen, *23*, 31
statMark, 32