Package ‘delmap’

November 22, 2012

Type Package

Title DELETION MAPPING

Version 1.0

Date 2012-11-12

Author Andrew W. George

Maintainer Andrew George `<andrew.george@csiro.au>`

Description
Deletion mapping is a statistical technique for ordering genetic markers. Unlike linkage mapping that is reliant upon recombinations to order loci, deletion mapping depends upon genomic deletions. These genomic deletions are introduced artificially. This package visualizes and analyzes deletion data on inbreds. The data can be read in from file or simulated.

License GPL (>= 2.0)

Depends R (>= 2.15.0), TSP, seriation

R topics documented:

`delmap-package` *Deletion Mapping*

Description

A package for using deletion data to fine map genetic marker loci. It is assumed the marker loci are (almost) completely linked and unable to be ordered through linkage. It is also assumed that a plant will contain, at most, a single deletion that spans multiple marker loci. The deletions are allowed to vary in size and there may be unobserved marker data. The package contains functions to find the best ordering of the marker loci, to visualize the deletion map, and to impute missing marker genotypes.

Details

Package:	delmap
Type:	Package
Version:	1.0
Date:	2012-11-12
License:	GPL (>=2)

~~ An overview of how to use the package, including the most important ~~ ~~ functions ~~

Author(s)

Author: Andrew W. George Maintainer: Andrew W. George <andrew.george@csiro.au>

Examples

```
S <- SimDeletions(numlines = 500, plines = 0.2, nummarkers = 20,
    Enumdel = 8, p.missing = 0.02,  seed = NULL)
```

`CleanData` *Clean Data in Readiness for Analysis*

Description

`CleanData` removes those rows and columns in the deletion data that do not carry any deletions.

Usage

```
CleanData(dmat= NULL, ignoreNA=TRUE)
```

Arguments

`dmat`	An object of class `delmap.data`
`ignoreNA`	if set to TRUE, missing genotypes will be ignored. If set to FALSE, rows/columns with missing genotypes will be retained.

Details

Rows and columns of genotype data that record no deletions can be removed from the analysis. This has the advantage of reducing the dimension of the problem.

Value

An object of class `delmap.data`.

Note

By default, missing genoyptes are ignored, which will lead to rows/columns being deleted if they do not carry any deletions. However, it may be more appropriate to set `ignoreNA=FALSE`. This will ensure rows/columns that have missing genotypes will be retained. Imputation can then be used later to impute the missing genotypes.

Author(s)

Andrew W George

See Also

`ReadData`, `ImputeMissingGeno`

Examples

```
# generate deletion data
Slist <-  SimDeletions( numlines = 50, plines = 0.2, nummarkers = 20,
    Enumdel = 4, p.missing = 0.02,  seed = 1)

# Number of marker loci before beign cleaned
nrow(Slist[["missing"]])

# Clean data but where missing genotypes are ignored.
CleanData(Slist[["missing"]])
nrow(CleanData(Slist[["missing"]]))

# Clean data but where colums/rows with missing genotypes are retained
# for later use.
nrow(CleanData(Slist[["missing"]],ignoreNA=FALSE))
```

`CreateDistMatrix` *Distance Measure Computation*

Description

This function computes a distance matrix from 0/1 deletion data. It is assumed the deletion data does not contain any missing genotypes.

Usage

```
CreateDistMatrix(mat = NULL)
```

Arguments

`mat`	a matrix or an object that can be converted to be a matrix that contains only 0's and 1's.

Details

'CreateDistMatrix' is a function for calculating a distance matrix between all pairs of loci from deletion data. This matrix is foundational to being able to order marker loci based on deletions. Our underlying assumption is that loci that are very close to each other will share a greater proportion of deletions across lines than those loci that are far apart. We expect this assumption to be valid if the genomic window being spanned by the marker loci is very small.

Our distance measure is based on the number of deletions in common between two loci. Suppose the gentoype data are

M1	M2	M3	M4
0	0	1	1
0	0	0	1
1	0	0	0
0	1	0	0
1	1	1	0

where we have data on five inbred plants on four marker loci, and 1 is a deletion.

First, the number of lines that share a deletion between two loci is calculated, giving

$$\begin{matrix} 2 & 1 & 1 & 0 \\ 1 & 2 & 1 & 0 \\ 1 & 1 & 2 & 1 \\ 0 & 0 & 1 & 2 \end{matrix}$$

Second, we subtract this matrix from its maximum value giving

$$\begin{matrix} 0 & 1 & 1 & 2 \\ 1 & 0 & 1 & 2 \\ 1 & 1 & 0 & 1 \\ 2 & 2 & 1 & 0 \end{matrix}$$

Third, the diagonal elements are set to a large value (say 100) giving the distance matrix

$$\begin{matrix} 100 & 1 & 1 & 2 \\ 1 & 100 & 1 & 2 \\ 1 & 1 & 100 & 1 \\ 2 & 2 & 1 & 100 \end{matrix}$$

We can then use this distance matrix in the Travelling Salesman Problem to find the "best" ordering of the marker loci.

Value

Returns an object of class 'dist'.

Author(s)

Andrew W. George

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index) for the standard data sets.

# deletion data where 1 is a deletion
dd <- matrix(data=c(0,0,1,1,
                        0,0,0,1,
                        1,0,0,0,
                        0,1,0,0,
                        1,1,1,0), nrow=5)
# calculate distance matrix for deletion data
CreateDistMatrix(mat=dd)
```

`delmap.data` *Data Class for Deletion Data*

Description

`delmap.data` is an delmap object class.

Value

An object of class `delmap.data` contains a matrix of 0,1 values where 1's denote deletions. It also contains the line names and marker loci names.

Author(s)

Andrew W. George

See Also

`ReadData`

`IdentifyMarkerBlocks` *Identify Marker Blocks*

Description

This function identifies all marker blocks from the deletion data. The deletion data cannot contain any missing genotypes.

Usage

```
IdentifyMarkerBlocks(dmat = NULL)
```

Arguments

`dmat`	an object of class delmap.data which contains the genotype data in 0/1 form and has no missing genotypes.

Details

A marker block is a group of marker loci that can be ordered relative to each other. They can be ordered because there are overlapping deletions that span the block of marker loci.

There may be multiple marker blocks for a set of deletion data. The markers within the blocks can be ordered but the marker loci across blocks cannot be ordered relative to each other.

Missing genotypes are not allowed and must first be imputed with `ImputeMissingGeno`

Value

Returns an integer vector of length the number of marker loci in `dmat`. Each unique block of markers recieves its own integer values. For example, a vector of 1,1,1,2,2,3,3 means that markers 1,2, and 3 belong to a block, markers 4 and 5 belong to a seperate marker block, and markers 6 and 7 belong to another block of markers.

Note

The marker blocks may change if marker genotypes are being imputed.

Author(s)

Andrew W. George

See Also

`IdentifyMarkerOrd`

Examples

```
# generate deletion data
Slist <-  SimDeletions( numlines = 50, plines = 0.2, nummarkers = 20,
    Enumdel = 4, p.missing = 0.02,  seed = 1)

# class of object that does not contain any missing data
class(Slist[["nomissing"]])

## Clean data i.e. remove rows/columns without deletion data
Slist[["nomissing"]] <- CleanData(Slist[["nomissing"]])

# print the block and possible marker orderings with each block for
# the simulated deletion data
IdentifyMarkerBlocks(Slist[["nomissing"]])
```

`IdentifyMarkerOrd` *Identify Marker Orderings*

Description

This function identifies the different possible marker orders that are possible with a set of deletion data and an assumed marker order. The assumed marker order is derived from the order of the marker columns in the object `dmat`.

Usage

```
IdentifyMarkerOrd(dmat = NULL)
```

Arguments

`dmat`	An object of class `delmap.data` which contains the genotype data in 0/1 form and has no missing genotypes.

Details

In some cases, it is possible to change the ordering of the marker loci (ie the order of the columns of genotype data) without changing the overall pattern of deletions. With `IdentifyMarkerOrd` we are able to find all possible orderings that are consistent with an assumed marker ordering.

The different marker orderings may occur in marker blocks. These blocks are identified with the function `IdentifyMarkerBlocks`. Marker blocks are groupings of marker loci that, according to the deletion data, can occur in any order. `IdentifyMarkerOrd` orders marker loci within marker blocks.

Missing genotypes are not allowed and must first be imputed with `ImputeMissingGeno`

Value

A list is returned with entries

`blocks`	is an integer vector which gives which block each marker loci belongs
`orders`	is a list where the number of entries corresponds to the number of marker blocks. Each list entry is an integer vector where the number of integer elements corresponds to the number of marker loci in the marker block. The integer elements indicate which marker loci can be permuted without chaning the overall pattern of deletions (given the assumed marker ordering). For example, a vector of 1,1,2,3,3,3 means that markers 1 and 2 can be swapped without affecting the pattern of deletions, marker 3 cannot be swapped with any other marker, and markers 4, 5, and 6 can be permuted.

Note

The marker blocks may change if marker genotypes are being imputed.

Author(s)

Andrew W. George

See Also

ImputeMissingGeno; IdentifyMarkerBlocks

Examples

```
# generate deletion data
Slist <-  SimDeletions( numlines = 50, plines = 0.2, nummarkers = 20,
    Enumdel = 4, p.missing = 0.02,  seed = 1)

# class of object that does not contain any missing data
class(Slist[["nomissing"]])

## Clean data i.e. remove rows/columns without deletion data
Slist[["nomissing"]] <- CleanData(Slist[["nomissing"]])

# print the block and possible marker orderings with each block for
# the simulated deletion data
IdentifyMarkerOrd(Slist[["nomissing"]])
```

ImputeMissingGeno *Impute Missing Marker Genotypes*

Description

This function imputes, uniformly, the missing marker genotypes.

Usage

```
ImputeMissingGeno(dmat = NULL)
```

Arguments

dmat	An object of class delmap.data which contains the genotype data in 0/1 form and contains missing genotypes.

Details

Missing genotypes are imputed, assuming 0,1 occurs with equal frequency.

Value

An object of class delmap.data is returned that has the missing genotyeps imputed.

Author(s)

Andrew W. George

Examples

```
# example to be done
# generate deletion data
Slist <-  SimDeletions( numlines = 50, plines = 0.2, nummarkers = 20,
    Enumdel = 4, p.missing = 0.02,  seed = 1)

# class of object that does not contain any missing data
class(Slist[["missing"]])


# impute missing genotypes
ImputeMissingGeno(Slist[["missing"]])
```

PermuteCols *Change Order of Marker Columns*

Description

`PermuteCols` column permutes the data in the object deldata.mat.

Usage

```
PermuteCols(dmat= NULL)
```

Arguments

`dmat`	An object of class `delmap.data`

Details

`PermuteCols` permutes, randomly, the order of the columns of genotype data.

Value

An object of class `delmap.data`.

Author(s)

Andrew W George

Examples

```
# generate deletion data
Slist <-  SimDeletions( numlines = 50, plines = 0.2, nummarkers = 20,
    Enumdel = 4, p.missing = 0.02,  seed = 1)

# print the data, including the marker ordering
print(Slist[["nomissing"]])

# permute the data and print the data, including the new marker ordering
print(PermuteCols(Slist[["nomissing"]]))
```

`print.delmap.data` *Print Deletion Data*

Description

'print' prints the deletion data.

Usage

```
## S3 method for class 'delmap.data'
print(x, ...)
```

Arguments

`x`	an object of class 'delmap.dat' that has either been read in from file or simulated.
`...`	further arguments passed to or from other methods.

Details

'print.delmap.data' is a customized print function for printing the contents of the 'delmap.data' object. The first five lines of data, plant names, and marker names are displayed.

Author(s)

Andrew W. George

See Also

`summary`

Examples

```
# generate deletion data
S <- SimDeletions( numlines = 500, plines = 0.2, nummarkers = 20,
    Enumdel = 8, p.missing = 0.02,  seed = NULL)

# print contents of data where all genotypes are observed
print(S$nomissing)

#print contents of data where there are missing genotypes
print(S$missing)
```

`ReadData` *Read Deletion Data*

Description

Reads a file in table format and creates a delmap.data object from it, with plants corresponding to rows and genetic markers corresponding to columns in the file

Usage

```
ReadData(datafile, line.names = FALSE, na.strings = "NA", marker.names = FALSE,
```

Arguments

`datafile`	the name of the file which the data are to be read from. Each row of the table appears as one line of the file. If it does not contain an absolute path, the file name is relative to the current working directory, 'getwd()'.
`line.names`	a logical value indicating whether the file contains plant names as the first column.
`na.strings`	a character vector of strings which are to be interpreted as 'NA' values.
`marker.names`	a logical value indicating whether the file contains the names of the marker loci as its first line.
`csv`	a logical value indicating if the file is comma seperated.

Details

Deletion data are read into R with this function. The data can be integer or character where any integer/character value is valid. The integer/character value with the lowest frequency is assumed to denote a deletion. The values are converted into 0/1 data where 1 denotes a deletion. If row/column names are not specified in this file, plant names L1, L2, . . . and marker names M1, M2, . . . will be assigned to the rows and columns, respectively.

Value

An object of class delmap.data. It contains the data in matrix form that has 0/1 elements. It also contains the plant names and marker loci names.

Author(s)

Andrew W. George

See Also

See Also `SimDeletions`

Examples

```
## Not run:
# read in deletion data
deldat <- ReadData(datafile="./deldata.txt", marker.names=TRUE, line.names=TRUE)

## End(Not run)
```

`SimDeletions` *Simulation*

Description

'SimDeletions' returns a matrix of genotype data where the rows correspond to plants and the columns correspond to tightly linked marker loci. A column (i.e., marker locus) can contain genotypes 0/1/NA where 0 is no deletion, 1 is a deletion, and NA is missing.

Usage

```
SimDeletions(numlines = NULL, plines = 0.5, nummarkers = NULL, labels = NULL, En
```

Arguments

`numlines`	number of inbred lines/plants.
`plines`	the probability of a plant carrying a deletion.
`nummarkers`	the number of marker loci spanning the genomic region of interest.
`labels`	a character vector containing the names of the marker loci.
`Enumdel`	the average length (in terms of number of marker loci) for a deletion.
`p.missing`	the probability of a genotype being unobserved.
`seed`	an integer number that if set, will produce reproducable replicates.

Details

Data are generated assuming the marker loci are in the true order and that a plant can only contain a single deletion (but that deletion may span several marker loci). The length of a deletion is random and follows a poisson distribution with mean Enumdel. If p.missing is set, then the data will contain missing genotypes (coded NA).

Value

a list is returned that contains only a single element if p.missing is NULL or two elements if p.missing is set. The elements are:

`nomissing`	is a matrix of genotype data 0/1 where 0 is no deletion and 1 is a deletion. The number of rows corresponds to the number of plants in the simulated population and the number of columns is the number of genetic marker loci. This will be the only element in output list if p.missing is NULL.
`missing`	is a matrix of genotype data 0/1 that is equivalent to "nomissing" but where marker genotypes have been replaced, randomly, and set to NA.

Author(s)

Andrew W. George

See Also

See Also `plot`

Examples

```
# To generate a inbred population of 500 plants
# where genotype data are collected on 20 marker loci,
# the probability of a plant having a genomic deletion is 0.2,
# the average length of a deletion is 8 marker loci, and
# the probability of a marker genotype being missing is 0.02,
# use:
S <- SimDeletions( numlines = 500, plines = 0.2, nummarkers = 20,
    Enumdel = 8, p.missing = 0.02,  seed = NULL)

# print contents of matrix
print(S$nomissing)

# print contents of matrix
print(S$missing)
```

Index