Package 'MPR.genotyping'

January 24, 2018

Type Package

Title Maximum Parsimony of Recombination to Infer Parental Genotypes

Version 0.8

Date 2018-01-22

Author
Weitong Guo[aut] <464984199@qq.com> and Weibo Xie[aut,cre] <xwbcn@webmail.hzau.edu.cn>

Maintainer Weitong Guo <464984199@qq.com>

Description Infer parental genotypes based on low-coverage population sequencing data and thus can genotype mapping populations and construct ultra-high density linkage map in a parent-independent manner. Weibo Xie et al. (2010) <doi:10.1073/pnas.1005931107>.

License GPL (>= 2)

Depends R (>= 2.10), qtl, stats

NeedsCompilation yes

Repository CRAN

Date/Publication 2018-01-24 18:24:42 UTC

R topics documented:

`MPR.genotyping-package`

Maximum Parsimony of Recombination to Infer Parental Genotypes

---

Description

Infer parental genotypes based on low-coverage population sequencing data and thus can genotype mapping populations and construct ultra-high density linkage map in a parent-independent manner. Weibo Xie et al. (2010) <doi:10.1073/pnas.1005931107>.

Details

The DESCRIPTION file:

Package:	MPR.genotyping
Type:	Package
Title:	Maximum Parsimony of Recombination to Infer Parental Genotypes
Version:	0.8
Date:	2018-01-22
Author:	Weitong Guo[aut] <464984199@qq.com> and Weibo Xie[aut,cre] <xwbcn@webmail.hzau.edu.cn>
Maintainer:	Weitong Guo <464984199@qq.com>
Description:	Infer parental genotypes based on low-coverage population sequencing data and thus can genotype mapping po
License:	GPL (>=2)
Depends:	qtl,stats

Index of help topics:

```
MPR.genotyping-package
                        Maximum Parsimony of Recombination to Infer
                        Parental Genotypes
NumRecomEvents          calculate the number of recombination events
base2Allele             Get Initial Allele from SNP Matrix
```

base2Geno	Formatting the Matrix of SNP to Int Type
base2Geno_ori	Formatting the Matrix of SNP to Int Type(NA)
checkData	Data for check
correctFUNHMM	correct FUN HMM
correctGeno	Correct Geno
fSNP	Example data of negative SNP
findBlockAndFilter	find Block And Filter
geno2Cross	geno to Cross
genoToBin	geno To Bin
genotypeCallsBayes	genotype Calls by Bayes
globalMPRByMarkers	MPR inference in whole chromosome
globalMPRRefine	refine SNPs by resampling
globalMPRwithoutMarkers	MPR without Markers
hmm.vitFUN.rils	Genotyping by HMM
localMPR	infer parental genotypes by minimizing the number of recombination events
makeEmissionFUN	emission of HMM
markerData	Data of Markers
mergeBinMap	merge Bin Map
mergeBlocks	merge Blocks
myBaseData	Example SNP data(smaller)
phenoData	data of phenotype
phy2get.haldane.rils	transition of HMM
snpData	Example SNP data

For implementing algorithms based on the principle of maximum parsimony of recombination (MPR) in a mapping population to infer parental genotypes and genotype the population in a parent-independent manner.

Author(s)

Weitong Guo[aut] <464984199@qq.com> and Weibo Xie[aut,cre] <xwbcn@webmail.hzau.edu.cn>

Maintainer: Weitong Guo <464984199@qq.com>

References

Weibo Xie etal. Parent-independent genotyping for constructing an ultrahigh-density linkage map based on population sequencing. <PNAS>. 2010.

See Also

localMPR,globalMPRByMarkers,globalMPRRefine,genotypeCallsBayes,correctGeno

Examples

```
data(myBaseData)
allele.random <- base2Allele(myBaseData)
allele.MPR <- localMPR(baseData=myBaseData,maxNStep=5,verbose = TRUE)
```

```
## The workflow of parent-independent genotyping
## load SNP alleles at putative SNP sites
data(snpData)
## load overlapping SNP alleles from low-depth sequences of one parent
data(markerData)

set.seed(123)

## select 30 markers randomly
markers <- sample(names(markerData)[10:50],20)

## select SNP sites which contain the 30 markers
ids <- match(markers,rownames(snpData))
str(myBaseData <- snpData[min(ids):max(ids),])

## global MPR aiding with marker data
allele.MPR <- globalMPRByMarkers(myBaseData,markers=markerData,numTry=3,numBaseStep=50,
             numBaseCandidateStep=100,numMarkerStep=10,useMedianToFindKnown=TRUE,
             maxIterate=150,maxNStep=3,scoreMin=0.8,verbose=TRUE)

## genotypes at some SNP site may be missing due to concordence less than 80% (scoreMin)
table(is.na(alleleA <- allele.MPR[,1]))

## check with positive dataset. looks good but contains false SNPs and may be with some errors

## then you need to refine the MPR results
set.seed(123);system.time(all.res <- globalMPRRefine(myBaseData,alleleA=na.omit(
             allele.MPR[,1]),numGroup=238,groupSort=TRUE,numPerm=10,numTry=3,
             numBaseStep=50,numBaseCandidateStep=100,numKnownStep=30,
             numKnownCandidateStep=50,useMedianToFindKnown=TRUE,maxIterate=150,
             maxNStep=3,scoreMin=0.8,saveMidData=TRUE,verbose=TRUE))

## summarize results using Bayesian inference
perm <- 10
res <- all.res$midData[[perm]]
table(geno.res <- genotypeCallsBayes(res$call,errorRate=1e-11,eps=1e-10,maxIterate=100,
             verbose=FALSE)$type)

## reconstruct parental genotypes
allele.MPR <- res$allele
allele.MPR[geno.res==2|apply(res$call,1,min)>=perm,] <- NA
allele.MPR[geno.res==3,] <- allele.MPR[geno.res==3,c(2,1)]

## the power of the MPR algorithm. lost 1 high quality SNP but removed >90% of inferior SNPs

## correct genotypes using HMM
snpSet <- sort(rownames(na.omit(allele.MPR)))
ids <- match(snpSet,rownames(myBaseData))
table(is.na(geno.data <- base2Geno_ori(myBaseData[ids,],allele.MPR[ids,])))

geno.data.cr <- geno.data
SNPbyChr <- split(1:nrow(geno.data),substr(rownames(geno.data),1,2))
```

```
for(chr in names(SNPbyChr)){
  cat("\r",chr)
  ids <- SNPbyChr[[chr]]
  geno.data.chr <- correctGeno(geno.data[ids,],correct.FUN=correctFUNHMM,
hmmFUN=hmm.vitFUN.rils,geno.probability=c(0.4975, 0.4975,0.005),
transitionFUN=phy2get.haldane.rils,
emissionFUN=makeEmissionFUN(errorRate=0.0106))
  geno.data.cr[ids,] <- geno.data.chr
}
```

`base2Allele` *Get Initial Allele from SNP Matrix*

Description

Get initial parental genotypes from SNP matrix randomly.

Usage

```
base2Allele(baseData = NULL)
```

Arguments

`baseData`	matrix of SNP

Details

base : matrix of SNP

Value

matrix :2 columns and nrows of SNP data

See Also

`globalMPRByMarkers`

Examples

```
base=matrix(c("A",NA,"G",
NA,"T","A",
"C","G",NA),3,3,byrow=TRUE)
base2Allele(base)
```

base2Geno *Formatting the Matrix of SNP to Int Type*

Description

formatting the matrix of snp to int type

Usage

```
base2Geno(baseData = NULL, allele.matrix = NULL)
```

Arguments

baseData	the matrix of snp
allele.matrix	the allele of each snp site

Details

the snp from parent1 will become 1, the snp is not detected by sequencing will become 0, the snp from parent2 will become -1.

Value

returned a matrix which have same size of matrix of snp, but "A","G","C","T" become "1","0","-1".

See Also

globalMPRByMarkers

Examples

```
base=matrix(c("A",NA,"G",
NA,"T","A",
"C","G",NA),3,3,byrow=TRUE)
allele=matrix(c("A","G",
"T","A",
"C","G"),3,2,byrow=TRUE)
base2Geno(base,allele)
```

base2Geno_ori *Formatting the Matrix of SNP to Int Type(NA)*

Description

formatting the matrix of SNP to int type(NA)

Usage

```
base2Geno_ori(baseData = NULL, allele.matrix = NULL)
```

Arguments

baseData	the matrix of snp
allele.matrix	the allele of each snp site

Details

the snp from parent1 will become 0, the snp is not detected by sequencing will become NA, the snp from parent2 will become 1.

Value

returned a matrix which have same size of matrix of snp, but "A","G","C","T" become "0","NA","1".

See Also

base2Geno

Examples

```
base=matrix(c("A",NA,"G",
NA,"T","A",
"C","G",NA),3,3,byrow=TRUE)
allele=matrix(c("A","G",
"T","A",
"C","G"),3,2,byrow=TRUE)
base2Geno_ori(base,allele)
```

checkData *Data for check*

Description

this is used to check up the genotype results in my example.

Usage

```
data("checkData")
```

Format

The format is: chr [1:11948, 1:2] "A" "A" "C" "T" "A" ... - attr(*, "dimnames")=List of 2 ..$: chr [1:11948] "0500000526A" "0500000556A" "0500000559G" "0500000591G" ... ..$: chr [1:2] "ZS97" "MH63"

Details

you can use "table(checkData[ids,1]==alleleA)"

Source

http://www.ncpgr.cn/supplements/MPR_genotyping/MPR_genotyping.tar.gz

Examples

```
#load data
data(checkData)
```

correctFUNHMM *correct FUN HMM*

Description

correct FUN HMM hmm.vitFUN.rils

correctGeno *Correct Geno*

Description

correct Geno

See Also

hmm.vitFUN.rils

findBlockAndFilter *find Block And Filter*

Description

find Block And Filter

fSNP *Example data of negative SNP*

Description

low quality snp data

Usage

```
data("fSNP")
```

Format

The format is: chr [1:1031] "0500021626C" "0500115741G" ...

Source

http://www.ncpgr.cn/supplements/MPR_genotyping/MPR_genotyping.tar.gz

Examples

```
#load data
data(fSNP)
```

geno2Cross *geno to Cross*

Description

geno to Cross

genoToBin *geno To Bin*

Description

geno To Bin

genotypeCallsBayes *genotype Calls by Bayes*

Description

After globalMPRRefine, you should use Bayes to call final genotype.

Usage

```
genotypeCallsBayes(ALLELE.num, errorRate = 5e-04, eps = 1e-10,
maxIterate = 100, verbose = FALSE)
```

Arguments

ALLELE.num	one result of globalMPRRefine
errorRate	errorRate
eps	eps
maxIterate	maxIterate to control running time
verbose	choose to show details

See Also

globalMPRRefine

Examples

```
ALLELE.num=matrix(c(100,1,
50,55,
89,2,
1,101,
0,78),5,3,byrow=TRUE)
genotypeCallsBayes(ALLELE.num)
```

globalMPRByMarkers *MPR inference in whole chromosome*

Description

The function to do MPR inference in whole chromosome by using localMPR to infer parental genotypes in hundreds of local regions and assemble them aiding with low-coverage sequences of one parent or known markers.

Usage

```
globalMPRByMarkers(baseData, markers = NULL, alleleA = NULL, numTry = 3,
numBaseStep = 50, numBaseCandidateStep = numBaseStep * 2,
numKnownStep = pmax(numBaseStep/5, 10),
numKnownCandidateStep = numKnownStep * 1.5,
useMedianToFindKnown = TRUE, maxNStep = 3, scoreMin = 0.8, verbose = FALSE,
strSTART = "\r", strEND = "", ...)
```

Arguments

baseData	(Necessary input) character matrix of SNP dataset
markers	character vector of markers data with SNP position names
alleleA	(Necessary input) character vector of one parent allele
numTry	maximum number of the times of using one SNP from one group (or RIL).
numBaseStep	number of SNP to run localMPR().
numBaseCandidateStep	number of SNP candidate in one step
numKnownStep	number of makers to run localMPR().
numKnownCandidateStep	number of makers candidate in one step
useMedianToFindKnown	In one local genomic region (window), we will choose the nearest some makers to this region. Median of the region will be the center.
maxNStep	parameter of localMPR()
scoreMin	be used to control the accuracy of MPR in one local genomic region (one window).
verbose	report verbose progress
strSTART	part of displaying format (verbose)
strEND	part of displaying format (verbose)
...	arguments to be passed to other methods.

See Also

localMPR

Examples

```
## load sample dataset
data(snpData)
data(markerData)

## select 30 markers randomly
set.seed(123);markers <- sample(names(markerData)[10:50],20)
```

```
## select SNP sites which contain the 30 markers
ids <- match(markers,rownames(snpData))
str(myBaseData <- snpData[min(ids):max(ids),])

## global MPR aiding with marker data
allele.MPR <- globalMPRByMarkers(myBaseData,markers=markerData,numTry=3,
numBaseStep=50,numBaseCandidateStep=100,
            numMarkerStep=10,useMedianToFindKnown=TRUE,
            maxNStep=3,scoreMin=0.8,verbose=TRUE)
```

`globalMPRRefine` *refine SNPs by resampling*

Description

The function to refine SNPs by resampling and Bayesian inference based on results of globalMPRByMarkers.

Usage

```
globalMPRRefine(baseData, markers = NULL, alleleA = NULL, numGroup = ncol(baseData),
groupSort = FALSE, numPerm = 10, numTry = 3, numBaseStep = 50,
numBaseCandidateStep = numBaseStep * 2, numKnownStep = numBaseStep/2,
numKnownCandidateStep = numBaseStep * 2, useMedianToFindKnown = TRUE,
maxNStep = 3, scoreMin = 0.8, useOnlyKnownToType = FALSE, useBayes = FALSE,
errorRate = 5e-04, saveMidData = FALSE, verbose = FALSE, strSTART = "\r",
strEND = "", ...)
```

Arguments

`baseData`	Your SNP matrix
`markers`	Your markers data
`alleleA`	We can use the results of globalMPRByMarkers() to be something like markers
`numGroup`	The number of group to run in one permutation.
`groupSort`	Your groups (or RILs) will sort by coverage.(higher coverage will make the result in local region better)
`numPerm`	The number of permutations
`numTry`	The maximum number of the times of using one SNP from one group (or RIL).
`numBaseStep`	The number of SNP to run localMPR().
`numBaseCandidateStep`	The number of SNP candidate in one step
`numKnownStep`	The number of makers to run localMPR().
`numKnownCandidateStep`	The number of makers candidate in one step

useMedianToFindKnown
: In one local genomic region (window), we will choose the nearest some makers to this region. Median of the region will be the center.

maxNStep
: The parameter of localMPR()

scoreMin
: The score is used to control the accuracy of MPR in one local genomic region (window).

useOnlyKnownToType
: Using different scoring system

useBayes
: "useBayes=TRUE" will make this routine integrate with Bayesian inference

errorRate
: the parameter of genotypeCallsBayes() if you set "useBayes=TRUE".

saveMidData
: If you want to check the effect of the number of permutations, "saveMidData = TRUE" will help you.

verbose
: Report verbose progress.

strSTART
: The part of displaying format (verbose)

strEND
: The part of displaying format (verbose)

...
: Arguments to be passed to other methods.

See Also

globalMPRByMarkers

Examples

```
data(snpData)
data(markerData)

## select 30 markers randomly
set.seed(123);markers <- sample(names(markerData)[10:50],20)

## select SNP sites which contain the 30 markers
ids <- match(markers,rownames(snpData))
str(myBaseData <- snpData[min(ids):max(ids),])

## global MPR aiding with marker data
allele.MPR <- globalMPRByMarkers(myBaseData,markers=markerData,numTry=3,
numBaseStep=50,numBaseCandidateStep=100,
            numMarkerStep=10,useMedianToFindKnown=TRUE,
maxNStep=3,scoreMin=0.8,verbose=TRUE)

## then you need to refine the MPR results
set.seed(123);system.time(all.res <- globalMPRRefine(myBaseData,alleleA=na.omit(
            allele.MPR[,1]),numGroup=238,groupSort=TRUE,numPerm=1,numTry=3,
            numBaseStep=50,numBaseCandidateStep=100,numKnownStep=30,
            numKnownCandidateStep=50,useMedianToFindKnown=TRUE,
            maxNStep=3,scoreMin=0.8,saveMidData=TRUE,verbose=TRUE))
```

`globalMPRwithoutMarkers` *MPR without Markers*

Description

MPR without Markers

Usage

```
globalMPRwithoutMarkers(baseData)
```

Arguments

`baseData`	snpData

Details

formatting is important

`hmm.vitFUN.rils` *Genotyping by HMM*

Description

correct genotype by Hidden Markov Model

Usage

```
hmm.vitFUN.rils(geno, position, geno.probability, transitionFUN =
phy2get.haldane.rils, emissionFUN = makeEmissionFUN(errorRate = 0.01), ...)
```

Arguments

`geno`	a numeric matrix of SNP genotype formatted
`position`	vector of SNP positions
`geno.probability`	vector of start probability
`transitionFUN`	a function about transition
`emissionFUN`	a function about emission
`...`	arguments to be passed to other methods.

Examples

```
O <- c(1,1,1,1,1,1,1,2,1,1,1,1,1,1,2,1,1,1,1,1,1,1,2,2,2,2,2,2,2,2,1,2,2,2,2,2,2,2,1,2,2,2,2,
    2,1,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,1,2,1,1,2,2,1,2,1,2,1,2,1,2,1,2,1,1,1,1,1,1,1,2,1,1,1,1,
     1,1,1,1,1,1,1,2,1,1,1,1,1,1,1,2,1,1,1,1,1)
O.pos <- 30e3*1:length(O)

O.cr <- hmm.vitFUN.rils(geno=O,position=O.pos,geno.probability=c(0.4975, 0.4975,0.005),
transitionFUN = phy2get.haldane.rils,
emissionFUN = makeEmissionFUN(errorRate = 0.01))

rbind(raw=paste(O,collapse=''),correct=paste(O.cr,collapse=''))
##         [,1]
## raw     "1111111211111121111111222222221222222221222221222222222
## 2222121122121212121211111112111111111112111111211111"
## correct "11111111111111111111112222222222222222222222222222222222
## 22223333333333333331111111111111111111111111111111"
```

`localMPR` *infer parental genotypes by minimizing the number of recombination events*

Description

The core function to infer parental genotypes (MPR inference) in a local chromosome region by minimizing the number of recombination events in the population.

Usage

```
localMPR(baseData, allele.matrix = NULL, maxNStep = 5,
returnNumRecomEvents = FALSE, verbose = FALSE, strEND = "\n", ...)
```

Arguments

`baseData`	a character matrix of SNP
`allele.matrix`	a character matrix of alleles
`returnNumRecomEvents`	whether report the number of recombination events in the final result
`maxNStep`	the number of SNP in an exchange of the alleles.
`verbose`	report verbose progress.
`strEND`	the part of displaying format (verbose)
`...`	arguments to be passed to other methods.

Details

Several factors may affect the accuracy of MPR inference: (a). The number of SNPs processed each time (window size); (b). The density of putative SNPs (the distance between SNP sites); (c). The maximum step size of the heuristic perturbation (the parameter of maxNStep); (d). The number of RILs. And we wrote core code in C to improve computational speed.

Value

the result of local MPR genotyping allele

See Also

NumRecomEvents,globalMPRByMarkers,globalMPRRefine,

Examples

```
set.seed(123)
## select 50 SNP sites to test inference of parental genotypes from snpData.rda
data(myBaseData)

## random assignments of parental genotypes to alleles will result in
## a big number of recombinations
allele.random <- base2Allele(myBaseData)

## a big number of recombinations
NumRecomEvents(myBaseData,allele.random)
## 162

## MPR inference with maximum step size of 5
allele.MPR <- localMPR(baseData=myBaseData,maxIterate=50,maxNStep=5,showDetail=TRUE)

## should be a small number compared with random assignments above
NumRecomEvents(myBaseData,allele.MPR)
## 33
```

---

makeEmissionFUN *emission of HMM*

Description

make emission matrix for HMM

Usage

```
makeEmissionFUN(errorRate = 0.01)
```

Arguments

errorRate	the error rate of RIL genotype calls at a specific SNP site

See Also

hmm.vitFUN.rils

markerData *Data of Markers*

Description

Load overlapping SNP alleles from low-depth sequences of one parent, you can also use genotype results from traditional genotyping studies, e.g. results of SSR/RFLP markers.

Usage

```
data("markerData")
```

Format

The format is: Named chr [1:272] "G" "T" "A" "C" "G" "C" "A" "T" "A" "C" "T" "A" ... - attr(*, "names")= chr [1:272] "0500033821G" "0500059721T" "0500125700G" "0500169119C" ...

Source

http://www.ncpgr.cn/supplements/MPR_genotyping/MPR_genotyping.tar.gz

Examples

```
data(markerData)
```

mergeBinMap *merge Bin Map*

Description

make final Bin Map

mergeBlocks *merge Blocks*

Description

a routine to merge Blocks

myBaseData *Example SNP data(smaller)*

Description

set.seed(123);myBaseData <- snpData[sample(200,50),]

Usage

```
data("myBaseData")
```

Format

The format is: chr [1:50, 1:238] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA "G" NA NA NA NA NA NA NA NA NA NA NA NA NA ... - attr(*, "dimnames")=List of 2 ..$: chr [1:50] "0500069986C" "0500170322G" "0500097599A" "0500174545T" ... ..$: chr [1:238] "MZ001" "MZ002" "MZ003" "MZ004" ...

Details

smaller

Source

snpData

Examples

```
## load data
data(myBaseData)
```

NumRecomEvents *calculate the number of recombination events*

Description

As the parental genotypes we assumed, this function can calculate how many recombination events would be needed to produce genotypes of RILs.

Usage

```
NumRecomEvents(baseData, allele.matrix, genoData = NULL)
```

Arguments

baseData	matirx of SNP
allele.matrix	matrix: alleles of parental genotypes
genoData	matirx of SNP which has been formatted.

Details

We wrote core code for calculating the number of recombination events in C to improve computational speed.

Value

R	the number of recombination events

See Also

localMPR

Examples

```
set.seed(123)
data(myBaseData)
allele.random <- base2Allele(myBaseData)
NumRecomEvents(myBaseData,allele.random)
# 162
```

---

phenoData *data of phenotype*

Description

grain width of RILs

Usage

```
data("phenoData")
```

Format

The format is: num [1:241, 1] 2.85 2.7 2.8 2.9 3.2 2.8 2.75 2.75 2.9 3.35 ... - attr(*, "dimnames")=List of 2 ..$: chr [1:241] "MZ001" "MZ002" "MZ003" "MZ004" ... ..$: chr "Grain_width"

Details

Grain width is quantitative phenotype

Source

http://www.ncpgr.cn/supplements/MPR_genotyping/MPR_genotyping.tar.gz

Examples

```
#load data
data(phenoData)
```

`phy2get.haldane.rils` *transition of HMM*

Description

make matrix of transition for HMM

Usage

```
phy2get.haldane.rils(a, b, l)
```

Arguments

a	state now
b	state next
l	the distance between two positions

`snpData` *Example SNP data*

Description

A total of 238 RILs developed from the cross be- tween Zhenshan 97 and Minghui 63 were re-sequenced with an Illumina Genome Analyzer II using the bar-coded multiplexed sequencing approach

Usage

```
data("snpData")
```

Format

The format is: chr [1:15795, 1:238] "A" NA NA NA NA NA NA "G" NA NA NA NA NA ... - attr(*, "dimnames")=List of 2 ..$: chr [1:15795] "0500000526A" "0500000556A" "0500000559G" "0500000591G" ... ..$: chr [1:238] "MZ001" "MZ002" "MZ003" "MZ004" ...

Details

SNP data format as it

Source

http://www.ncpgr.cn/supplements/MPR_genotyping/MPR_genotyping.tar.gz

Examples

```
# load data
data(snpData)
```

Index