Package 'hapFabia'

November 13, 2026

Title hapFabia: Identification of very short segments of identity by descent (IBD) characterized by rare variants in large sequencing data

Version 1.55.0

Date 2017-03-11

Author Sepp Hochreiter <hochreit@bioinf.jku.at>

Maintainer Andreas Mitterecker <mitterecker@bioinf.jku.at>

Depends R (>= 3.6.0), Biobase, fabia (>= 2.3.1)

Imports methods, graphics, grDevices, stats, utils

Description A package to identify very short IBD segments in large sequencing data by FABIA biclustering. Two haplotypes are identical by descent (IBD) if they share a segment that both inherited from a common ancestor. Current IBD methods reliably detect long IBD segments because many minor alleles in the segment are concordant between the two haplotypes. However, many cohort studies contain unrelated individuals which share only short IBD segments. This package provides software to identify short IBD segments in sequencing data. Knowledge of short IBD segments are relevant for phasing of genotyping data, association studies, and for population genetics, where they shed light on the evolutionary history of humans. The package supports VCF formats, is based on sparse matrix operations, and provides visualization of haplotype clusters in different formats.

License LGPL (>= 2.1)

Collate AllClasses.R AllGenerics.R hapFabia.R methods-Factorization-class.R methods-IBDsegment-class.R methods-IBDsegmentList-class.R zzz.R

URL http://www.bioinf.jku.at/software/hapFabia/hapFabia.html

biocViews Genetics, GeneticVariability, SNP, Sequencing, Sequencing, Visualization, Clustering, SequenceMatching, Software

git_url https://git.bioconductor.org/packages/hapFabia

git_branch devel

git_last_commit 067e34d

git_last_commit_date 2026-04-28

Repository Bioconductor 3.24
Date/Publication 2026-11-12

Contents

analyzeIBDsegments . . . 2
chr1ASW1000G . . . 7
compareIBDsegmentLists . . . 8
extractIBDsegments . . . 9
findDenseRegions . . . 11
hapFabia . . . 13
hapFabiaVersion . . . 18
hapRes . . . 19
IBDsegment-class . . . 20
IBDsegmentList-class . . . 25
IBDsegmentList2excel . . . 28
identifyDuplicates . . . 30
iterateIntervals . . . 34
makePipelineFile . . . 40
matrixPlot . . . 41
mergedIBDsegmentList . . . 42
mergeIBDsegmentLists . . . 43
plotIBDsegment . . . 44
res . . . 47
setAnnotation . . . 48
setStatistics . . . 50
sim . . . 51
simu . . . 52
simulateIBDsegments . . . 54
simulateIBDsegmentsFabia . . . 56
split_sparse_matrix . . . 57
toolsFactorizationClass . . . 61
vcftoFABIA . . . 63

Index **68**

`analyzeIBDsegments` *Loop over extracted IBD segments to supply a descriptive statistics*

Description

`analyzeIBDsegments`: R implementation of `analyzeIBDsegments`.

The functions provides a loop over all detected IBD segments in order to compute descriptive statistics.

Usage

```
analyzeIBDsegments(fileName,runIndex="",
  annotPostfix="_annot.txt",startRun=1,endRun,shift=5000,
  intervalSize=10000)
```

Arguments

fileName	file name prefix without type of the result of hapFabia; Attention no type!
runIndex	a string that marks the output of this run if splitting the analysis into subsets of intervals defined by startRun and endRun.
annotPostfix	postfix string for the SNV annotation file.
startRun	first interval.
endRun	last interval.
shift	distance between start of adjacent intervals.
intervalSize	number of SNVs in an interval.

Details

The functions provides a loop over all detected IBD segments in order to compute descriptive statistics. The loop goes over the intervals that have been analyzed for IBD segments by iterateIntervals. Duplicates are ignored at this analysis and must be identified in a preceding step via identifyDuplicates. Other statistics and annotations can be computed if the code is changed accordingly.

Implementation in R.

Value

list containing:

startRun	first interval.
endRun	last interval.
noIBDsegments	number of IBD segments.
avIBDsegmentPos	vector of physical locations of IBD segments.
avIBDsegmentLengthSNV	vector of lengths of IBD segments given in number of SNVs.
avIBDsegmentLength	vector of lengths of IBD segments in bp.
avnoIndividp	vector of number of individuals that belong to the IBD segment.
avnoTagSNVs	vector of number of tagSNVs that mark the IBD segment.
avnoFreq	vector of minor allele frequencies of tagSNVs.
avnoGroupFreq	vector of minor allele frequencies within the considered subpopulation.
avnotagSNVChange	vector indicating a switch between minor and major alleles of tagSNVs (1=switched,0=not switched).
avnotagSNVsPerIndividual	vector of number of tagSNVs per individual.
avnoindividualPerTagSNV	vector of number of individuals that possess the minor allele per tagSNV .
avIBDsegmentPosS	summary statistics of physical locations of IBD segments.
avIBDsegmentLengthS	summary statistics of lengths of IBD segments.
avnoIndividS	summary statistics of number of individuals that belong to the IBD segment.

`avnoTagSNVsS` summary statistics of number of tagSNVs that mark the IBD segment.

`avnoFreqS` summary statistics of minor allele frequencies of tagSNVs.

`avnoGroupFreqs` summary statistics of minor allele frequencies within the considered subpopulation.

`avnotagSNVChangeS` summary statistics of vector that indicates a switch between minor and major alleles of tagSNVs (1=switched,0=not switched).

`avnotagSNVsPerIndividualS` summary statistics of number of tagSNVs per individual.

`avnoindividualPerTagSNVS` summary statistics of number of individuals that possess the minor allele per tagSNV.

Author(s)

Sepp Hochreiter

References

S. Hochreiter et al., 'FABIA: Factor Analysis for Bicluster Acquisition', Bioinformatics 26(12):1520-1527, 2010.

See Also

IBDsegment-class, IBDsegmentList-class, analyzeIBDsegments, compareIBDsegmentLists, extractIBDsegments, findDenseRegions, hapFabia, hapFabiaVersion, hapRes, chr1ASW1000G, IBDsegmentList2excel, identifyDuplicates, iterateIntervals, makePipelineFile, matrixPlot, mergeIBDsegmentLists, mergedIBDsegmentList, plotIBDsegment, res, setAnnotation, setStatistics, sim, simu, simulateIBDsegmentsFabia, simulateIBDsegments, split_sparse_matrix, toolsFactorizationClas
vcftoFABIA

Examples

```
print("Loop over extracted IBD segments to supply a descriptive statistics")
## Not run:
########################################
## Already run in "iterateIntervals.Rd" ##
########################################

#Work in a temporary directory.

old_dir <- getwd()
setwd(tempdir())


# Load data and write to vcf file.
data(chr1ASW1000G)
write(chr1ASW1000G,file="chr1ASW1000G.vcf")

#Create the analysis pipeline for IBD segment detection
makePipelineFile(fileName="chr1ASW1000G",shiftSize=500,intervalSize=1000,haplotypes=TRUE)

source("pipeline.R")
```

```
# Following files are produced:
list.files(pattern="chr1")



# Next we load interval 5 and there the first and second IBD segment
posAll <- 5
start <- (posAll-1)*shiftSize
end <- start + intervalSize
pRange <- paste("_",format(start,scientific=FALSE),"_",format(end,scientific=FALSE),sep="")
load(file=paste(fileName,pRange,"_resAnno",".Rda",sep=""))
IBDsegmentList <- resHapFabia$mergedIBDsegmentList
summary(IBDsegmentList)
IBDsegment1 <- IBDsegmentList[[1]]
summary(IBDsegment1)
IBDsegment2 <- IBDsegmentList[[2]]
summary(IBDsegment2)




#Plot the first IBD segment in interval 5
plot(IBDsegment1,filename=paste(fileName,pRange,"_mat",sep=""))


#Plot the second IBD segment in interval 5
plot(IBDsegment2,filename=paste(fileName,pRange,"_mat",sep=""))

setwd(old_dir)


## End(Not run)

## Not run:
###here an example of the the automatically generated pipeline
### with: shiftSize=5000,intervalSize=10000,fileName="filename"

#####define intervals, overlap, filename #######
shiftSize <- 5000
intervalSize <- 10000
fileName="filename" # without type
haplotypes <- TRUE
dosage <- FALSE

#####load library#######
library(hapFabia)

#####convert from .vcf to _mat.txt#######
vcftoFABIA(fileName=fileName)

#####copy haplotype, genotype, or dosage matrix to matrix#######
if (haplotypes) {
    file.copy(paste(fileName,"_matH.txt",sep=""), paste(fileName,"_mat.txt",sep=""))
} else {
    if (dosage) {
        file.copy(paste(fileName,"_matD.txt",sep=""), paste(fileName,"_mat.txt",sep=""))
    } else {
```

```
        file.copy(paste(fileName,"_matG.txt",sep=""), paste(fileName,"_mat.txt",sep=""))
    }
}

#####split/ generate intervals#######
split_sparse_matrix(fileName=fileName,intervalSize=intervalSize,
shiftSize=shiftSize,annotation=TRUE)

#####compute how many intervals we have#######
ina <- as.numeric(readLines(paste(fileName,"_mat.txt",sep=""),n=2))
noSNVs <- ina[2]
over <- intervalSize%/%shiftSize
N1 <- noSNVs%/%shiftSize
endRunA <- (N1-over+2)


#####analyze each interval#######
#####may be done by parallel runs#######
iterateIntervals(startRun=1,endRun=endRunA,shift=shiftSize,
intervalSize=intervalSize,fileName=fileName,individuals=0,
upperBP=0.05,p=10,iter=40,alpha=0.03,cyc=50,IBDsegmentLength=50,
Lt = 0.1,Zt = 0.2,thresCount=1e-5,mintagSNVsFactor=3/4,
pMAF=0.03,haplotypes=haplotypes,cut=0.8,procMinIndivids=0.1,thresPrune=1e-3,
simv="minD",minTagSNVs=6,minIndivid=2,avSNVsDist=100,SNVclusterLength=100)

#####identify duplicates#######
identifyDuplicates(fileName=fileName,startRun=1,endRun=endRunA,
shift=shiftSize,intervalSize=intervalSize)

#####analyze results; parallel#######
anaRes <- analyzeIBDsegments(fileName=fileName,startRun=1,endRun=endRunA,
shift=shiftSize,intervalSize=intervalSize)
print("Number IBD segments:")
print(anaRes$noIBDsegments)
print("Statistics on IBD segment length in SNVs (all SNVs in the IBD segment):")
print(anaRes$avIBDsegmentLengthSNVS)
print("Statistics on IBD segment length in bps:")
print(anaRes$avIBDsegmentLengthS)
print("Statistics on number of individuals belonging to IBD segments:")
print(anaRes$avnoIndividS)
print("Statistics on number of tagSNVs of IBD segments:")
print(anaRes$avnoTagSNVsS)
print("Statistics on MAF of tagSNVs of IBD segments:")
print(anaRes$avnoFreqS)
print("Statistics on MAF within the group of tagSNVs of IBD segments:")
print(anaRes$avnoGroupFreqS)
print("Statistics on number of changes between major and minor allele frequency:")
print(anaRes$avnotagSNVChangeS)
print("Statistics on number of tagSNVs per individual of an IBD segment:")
print(anaRes$avnotagSNVsPerIndividualS)
print("Statistics on number of individuals that have the minor allele of tagSNVs:")
print(anaRes$avnoindividualPerTagSNVS)

#####load result for interval 50#######
posAll <- 50 # (50-1)*5000 = 245000: interval 245000 to 255000
start <- (posAll-1)*shiftSize
end <- start + intervalSize
```

```
pRange <- paste("_",format(start,scientific=FALSE),"_",
format(end,scientific=FALSE),sep="")
load(file=paste(fileName,pRange,"_resAnno",".Rda",sep=""))
IBDsegmentList <- resHapFabia$mergedIBDsegmentList # $

summary(IBDsegmentList)
#####plot IBD segments in interval 50#######
plot(IBDsegmentList,filename=paste(fileName,pRange,"_mat",sep=""))
   ##attention: filename without type ".txt"

#####plot the first IBD segment in interval 50#######

IBDsegment <- IBDsegmentList[[1]]
plot(IBDsegment,filename=paste(fileName,pRange,"_mat",sep=""))
   ##attention: filename without type ".txt"


## End(Not run)
```

---

chr1ASW1000G *Example genotype data in* vcf *format*

Description

Genotype data in vcf format. Contains chr1ASW1000G of class character.

A vcf file is produced by write(chr1ASW1000G,file="chr1ASW1000G.vcf").

Usage

```
chr1ASW1000G
```

Format

String chr1ASW1000G of class character.

Source

1000Genomes phase 1 release v3; chromosome 1, only ASW and only few SNVs.

References

S. Hochreiter et al., 'FABIA: Factor Analysis for Bicluster Acquisition', Bioinformatics 26(12):1520-1527, 2010.

See Also

IBDsegment-class, IBDsegmentList-class, analyzeIBDsegments, compareIBDsegmentLists, extractIBDsegments, findDenseRegions, hapFabia, hapFabiaVersion, hapRes, chr1ASW1000G, IBDsegmentList2excel, identifyDuplicates, iterateIntervals, makePipelineFile, matrixPlot, mergeIBDsegmentLists, mergedIBDsegmentList, plotIBDsegment, res, setAnnotation, setStatistics, sim, simu, simulateIBDsegmentsFabia, simulateIBDsegments, split_sparse_matrix, toolsFactorizationClas
vcftoFABIA

compareIBDsegmentLists *Hierarchical clustering of IBD segments stored in IBD segment list(s)*

Description

compareIBDsegmentLists: R implementation of compareIBDsegmentLists.

The IBD segments in one or two list(s) are compared by hierarchical clustering. Different similarity measures are available. Called by hapFabia.

Usage

```
## S4 method for signature
## 'IBDsegmentList,ANY,character,ANY,ANY,numeric,numeric'
compareIBDsegmentLists(IBDsegmentList1,IBDsegmentList2=NULL,simv="minD",pTagSNVs=NULL,pIndivid=N
```

Arguments

IBDsegmentList1	list of IBD segments given as IBDsegmentList object.
IBDsegmentList2	optional: second list of IBD segments given as IBDsegmentList object.
simv	similarity measure: minD (percentage of the smaller set explained by the larger set), jaccard (Jaccard index), dice (Dice index), or maxD (percentage of the larger set explained by the smaller set); default minD.
pTagSNVs	optional: exponent for tagSNV similarity.
pIndivid	optional: exponent for individuals similarity.
minTagSNVs	required minimal number of overlapping SNVs to assign similarity different from zero; default 6.
minIndivid	required minimal number of overlapping individuals to assign similarity different from zero; default 2.

Details

Similarities are separately computed for SNVs and for individuals using one of the similarity measure: "minD" (percentage of the smaller set explained by the larger set), "jaccard" (Jaccard index), "dice" (Dice index), or "maxD" (percentage of the larger set explained by the smaller set). One minus the product between SNV similarity and individuals similarity is the final value used for clustering.

The final similarity measure is not a distance but is symmetric, one for similarity of an IBD segment with itself, zero if either CNVs or individuals have no overlap, and between zero and one. Called by hapFabia.

Implementation in R.

Value

clust	object of class hclust which describes the tree produced by the hierarchical clustering method.

Author(s)

Sepp Hochreiter

References

S. Hochreiter et al., 'FABIA: Factor Analysis for Bicluster Acquisition', Bioinformatics 26(12):1520-1527, 2010.

See Also

IBDsegment-class, IBDsegmentList-class, analyzeIBDsegments, compareIBDsegmentLists, extractIBDsegments, findDenseRegions, hapFabia, hapFabiaVersion, hapRes, chr1ASW1000G, IBDsegmentList2excel, identifyDuplicates, iterateIntervals, makePipelineFile, matrixPlot, mergeIBDsegmentLists, mergedIBDsegmentList, plotIBDsegment, res, setAnnotation, setStatistics, sim, simu, simulateIBDsegmentsFabia, simulateIBDsegments, split_sparse_matrix, toolsFactorizationClas
vcftoFABIA

Examples

```
data(hapRes)
IBDsegmentList1 <- hapRes$IBDsegmentList1
IBDsegmentList2 <- hapRes$IBDsegmentList2
comp <-
compareIBDsegmentLists(IBDsegmentList1,
   IBDsegmentList2,simv="minD",pTagSNVs=NULL,
   pIndivid=NULL,minTagSNVs=6,minIndivid=2)

show(comp)
```

---

extractIBDsegments *Extract IBD segments from a* fabia *result*

Description

extractIBDsegments: R implementation of extractIBDsegments.

IBD segments are identified in FABIA Factorization objects. First accumulations of correlated SNVs are found. Then IBD segments in these accumulations are disentangled. Finally IBD segments are pruned off spurious correlated SNVs.

Usage

```
## S4 method for signature
## 'Factorization,
##   list,
##   data.frame,
##   character,
##   matrix,
##   numeric,
##   numeric,
```

```
##   numeric,
##   numeric,
##   numeric,
##   numeric,
##   numeric,
##   numeric'
extractIBDsegments(res,sPF,annot=NULL,chrom="",labelsA=NULL,ps=0.9,psZ=0.8,inteA=500,thresA=11,m
```

Arguments

res	result of fabia given as Factorization object.
sPF	genotype data obtained by fabia procedure samplesPerFeature; it gives for each SNV the individuals/chromosomes that possess the minor allele.
annot	annotation for the tagSNVs as an object of the class data.frame; if it is NULL then a dummy annotation is generated.
chrom	the chromosome the genotyping data stems from.
labelsA	labels for the individuals; if it is NULL then dummy labels by enumerating individuals are generated.
ps	quantile above which the L values are considered for IBD segment extraction.
psZ	quantile above which the largest Z values are considered for IBD segment extraction.
inteA	number of SNVs in a histogram bin which correspond to the desired IBD segment length.
thresA	threshold for histogram counts above which SNVs are viewed to be locally accumulated in a histogram bin.
mintagSNVs	threshold for minimal tagSNV overlap of intervals in a IBD segment.
off	offset of the histogram.
procMinIndivids	percent of cluster individuals that must have the minor allele to consider an SNV as IBD segment tagSNV.
thresPrune	threshold on the probability of having minimal distance to neighboring tagSNVs; used to prune off SNVs at the border of IBD segments.

Details

The threshold thresA for counts in a bin, which indicates SNV accumulations, is computed and provided by hapFabia when calling this method. Distance probabilities for pruning are based on an exponential distribution with the median distance between tagCNVs as parameter (one over the rate). Thus, the counts are assumed to be Poisson distributed. At the IBD segment border, SNVs that have a large distance to the closest tagSNV are pruned off. thresPrune gives the pruning threshold via a p-value for observing this distance or a larger based on the exponential distribution.

Implementation in R.

Value

An instance of the class IBDsegmentList containing the extracted IBD segments.

Author(s)

Sepp Hochreiter

References

S. Hochreiter et al., 'FABIA: Factor Analysis for Bicluster Acquisition', Bioinformatics 26(12):1520-1527, 2010.

See Also

IBDsegment-class, IBDsegmentList-class, analyzeIBDsegments, compareIBDsegmentLists, extractIBDsegments, findDenseRegions, hapFabia, hapFabiaVersion, hapRes, chr1ASW1000G, IBDsegmentList2excel, identifyDuplicates, iterateIntervals, makePipelineFile, matrixPlot, mergeIBDsegmentLists, mergedIBDsegmentList, plotIBDsegment, res, setAnnotation, setStatistics, sim, simu, simulateIBDsegmentsFabia, simulateIBDsegments, split_sparse_matrix, toolsFactorizationClas vcftoFABIA

Examples

```
data(hapRes)
res <- hapRes$res
sPF <- hapRes$sPF
annot <- hapRes$annot
nnL <- length(Z(res)[1,])
labelsA <- cbind(as.character(1:nnL),
   as.character(1:nnL),as.character(1:nnL),
   as.character(1:nnL))
resIBDsegmentList <- extractIBDsegments(res=res,
   sPF=sPF,annot=annot,chrom="1",labelsA=labelsA,
   ps=0.9,psZ=0.8,inteA=50,thresA=6,mintagSNVs=6,
   off=0,procMinIndivids=0.1,thresPrune=1e-3)

summary(resIBDsegmentList)

print("Position of the first IBD segment:")
print(IBDsegmentPos(resIBDsegmentList[[1]]))

print("Length of the first IBD segment:")
print(IBDsegmentLength(resIBDsegmentList[[1]]))
```

---

findDenseRegions *Find accumulations of values via histogram counts*

Description

findDenseRegions: R implementation of findDenseRegions.

Extracts histogram bins which counts are larger than a threshold. Only values larger than a given quantile are considered.

Usage

```
findDenseRegions(obs,p=0.9,inte=500,thres=11,off=0)
```

Arguments

obs	values for constructing the histogram.
p	quantile above which the values of obs are included into the histogram.
inte	size of the histogram bins.
thres	threshold for histogram counts: bin with counts larger equal the threshold are selected.
off	offset of the histogram bin positions.

Details

Extracts histogram bins which counts are larger than a threshold. The threshold is supplied and can be computed according to some assumptions on expected bin counts. Only values larger than a given quantile are considered.

Implementation in R.

Value

list with

m	vector of locations of the selected bin (middle of bins).
l	vector of lengths of the bins.
pos	list where each element is a vector of locations of values that contributed to the counts (SNV positions).
len	vector of counts or equivalently vector of the number of SNVs.

Author(s)

Sepp Hochreiter

References

S. Hochreiter et al., 'FABIA: Factor Analysis for Bicluster Acquisition', Bioinformatics 26(12):1520-1527, 2010.

See Also

IBDsegment-class, IBDsegmentList-class, analyzeIBDsegments, compareIBDsegmentLists, extractIBDsegments, findDenseRegions, hapFabia, hapFabiaVersion, hapRes, chr1ASW1000G, IBDsegmentList2excel, identifyDuplicates, iterateIntervals, makePipelineFile, matrixPlot, mergeIBDsegmentLists, mergedIBDsegmentList, plotIBDsegment, res, setAnnotation, setStatistics, sim, simu, simulateIBDsegmentsFabia, simulateIBDsegments, split_sparse_matrix, toolsFactorizationClas
vcftoFABIA

Examples

```
data(res)
ib <- findDenseRegions(L(res)[,1],p=0.9,
   inte=50,thres=6,off=0)
print(ib$len)
```

---

hapFabia *IBD segment extraction by FABIA*

Description

hapFabia: R implementation of the *hapFabia* method.

hapFabia extracts **short IBD segments** tagged by rare variants from phased or unphased genotypes. hapFabia is designed for rare variants in very large sequencing data. The method is based on FABIA biclustering and utilizes the package **fabia**.

Usage

```
hapFabia(fileName,prefixPath="",sparseMatrixPostfix="_mat",
   annotPostfix="_annot.txt",individualsPostfix="_individuals.txt",
   labelsA=NULL,pRange="",individuals=0,lowerBP=0,upperBP=0.05,
   p=10,iter=40,quant=0.01,eps=1e-5,alpha=0.03,cyc=50,non_negative=1,
   write_file=0,norm=0,lap=100.0,IBDsegmentLength=50,Lt = 0.1,
   Zt = 0.2,thresCount=1e-5,mintagSNVsFactor=3/4,pMAF=0.03,
   haplotypes=FALSE,cut=0.8,procMinIndivids=0.1,thresPrune=1e-3,
   simv="minD",minTagSNVs=6,minIndivid=2,avSNVsDist=100,SNVclusterLength=100)
```

Arguments

fileName	name of the file that contains the genotype matrix in sparse format.
prefixPath	path to the genotype file.
sparseMatrixPostfix	postfix string for the sparse matrix.
annotPostfix	postfix string for the SNV annotation file.
individualsPostfix	postfix string for the file containing the names of the individuals.
labelsA	annotation of the individuals as matrix individuals x 4 (individual ID, subpopulation, population, genotyping platform).
pRange	indicates which DNA interval is processed.
individuals	vector of individuals that are included into the analysis; default = 0 (all individuals).
lowerBP	lower bound on minor allele frequencies (MAF); however at least two occurrences are required to remove private SNVs.
upperBP	upper bound on minor allele frequencies (MAF) to extract rare variants.
p	number of biclusters per fabia iteration.
iter	number of fabia iterations.

quant	percentage of fabia loadings L that are removed after each iteration.
eps	lower bound on fabia variational parameter lapla; default 1e-5.
alpha	fabia sparseness of the loadings; default = 0.03.
cyc	number of cycles per fabia iteration; default 50.
non_negative	non-negative fabia factors and loadings if non_negative = 1; default = 1 (yes).
write_file	fabia results are written to files (L in sparse format), default = 0 (not written).
norm	fabia data normalization; default 1 (no normalization).
lap	minimal value of fabia's variational parameter; default 100.0.
IBDsegmentLength	expected typical IBD segment length in kbp.
Lt	percentage of largest fabia L values to consider for IBD segment extraction.
Zt	percentage of largest fabia Z values to consider for IBD segment extraction.
thresCount	p-value of random histogram hit; default 1e-5.
mintagSNVsFactor	percentage of the histogram tagSNVs count threshold (mintagSNVs in extractIBDsegments); used to define minimal overlap of individual intervals in an IBD segment; default 3/4.
pMAF	averaged and corrected (for non-uniform distributions) minor allele frequency.
haplotypes	haplotypes = TRUE indicates phased genotypes that is two chromosomes per individual otherwise unphased genotypes.
cut	cutoff for merging IBD segments after a hierarchical clustering; default 0.8.
procMinIndivids	Percentage of cluster individuals a tagSNV must tag to be considered as tagSNV for the IBD segment.
thresPrune	Threshold for pruning border tagSNVs based on an exponential distribution where border tagSNVs with large distances to the next tagSNV are pruned.
simv	Similarity measure for merging clusters: "minD" (percentage of smaller explained by larger set), "jaccard" (Jaccard index), "dice" (Dice index), or "maxD"; default "minD".
minTagSNVs	Minimum matching tagSNVs for cluster similarity; otherwise the similarity is set to zero.
minIndivid	Minimum matching individuals for cluster similarity; otherwise the similarity is set to zero.
avSNVsDist	average distance between SNVs in base pairs - used together with IBDsegmentLength to compute the number of SNVs in the histogram bins; default=100.
SNVclusterLength	if IBDsegmentLength=0 then the number of SNVs in the histogram bins can be given directly; default 100.

Details

This function uses a genotype matrix in sparse matrix format and extracts IBD segments by biclustering. First, it performs Fabia biclustering and then extracts local accumulations of loadings. Then it disentangles IBD segments and prunes off spurious correlated SNVs. Finally, it merges similar IBD segments to account for larger IBD segments that were broken during the analysis.

Annotation file ..._annot.txt for SNVs:

1. first line: number individuals;
2. second line: number SNVs;
3. for each SNV a line containing following field that are blank separated: "chromosome", "physical position", "snvNames", "snvMajor", "snvMinor", "quality", "pass", "info of vcf file", "fields in vcf file", "frequency", "0/1: 1 is changed if major allele is actually minor allele".

`labelsA` is a matrix ("number individuals" x 4), where for each individual following characteristics are given:

1. id;
2. subPopulation;
3. population;
4. platform.

The probability of observing k or more correlated SNVs in a histogram bin is computed. The minimal k which pushes the probability below the threshold `thresCount` is used to find accumulation of correlated SNVs that are assumed to belong to a IBD segment.

Let p be the probability of a random minor allele match between t individuals. The probability of observing k or more matches for n SNVs in a histogram bin is given by one minus the binomial distribution $F(k; n, p)$:

$$1 \; - \; F(k-1; n, p) \; = \; \Pr(K \geq k) \; = \; \sum_{i=k}^{\infty} \binom{n}{i} \, p^i \, (1-p)^{n-i}$$

If q is the minor allele frequency (MAF) for one SNV, the probability p of observing the minor allele of this SNV in all t individuals is $p = q^t$. The value q is given by the parameter `pMAF`.

Implementation in R.

Value

List containing

`mergedIBDsegmentList`
: an object of the class `IBDsegmentList` that contains the extracted IBD segments that were extracted from two histograms with different offset.

`res`
: the result of FABIA.

`sPF`
: individuals per loading of this FABIA result.

`annot`
: annotation for the genotype data.

`IBDsegmentList1`
: an object of the class `IBDsegmentList` that contains the result of IBD segment extraction from the first histogram.

`IBDsegmentList2`
: an object of the class `IBDsegmentList` that contains the result of IBD segment extraction from the second histogram.

`mergedIBDsegmentList1`
: an object of the class `IBDsegmentList` that contains the merged result of the first IBD segment extraction (redundancies removed).

`mergedIBDsegmentList2`
: an object of the class `IBDsegmentList` that contains the merged result of the second IBD segment extraction (redundancies removed).

Author(s)

Sepp Hochreiter

References

S. Hochreiter et al., 'FABIA: Factor Analysis for Bicluster Acquisition', Bioinformatics 26(12):1520-1527, 2010.

See Also

IBDsegment-class, IBDsegmentList-class, analyzeIBDsegments, compareIBDsegmentLists, extractIBDsegments, findDenseRegions, hapFabia, hapFabiaVersion, hapRes, chr1ASW1000G, IBDsegmentList2excel, identifyDuplicates, iterateIntervals, makePipelineFile, matrixPlot, mergeIBDsegmentLists, mergedIBDsegmentList, plotIBDsegment, res, setAnnotation, setStatistics, sim, simu, simulateIBDsegmentsFabia, simulateIBDsegments, split_sparse_matrix, toolsFactorizationClas
vcftoFABIA

Examples

```
old_dir <- getwd()
setwd(tempdir())

data(simu)

namesL <- simu[["namesL"]]
haploN <- simu[["haploN"]]
snvs <- simu[["snvs"]]
annot <- simu[["annot"]]
alleleIimp <- simu[["alleleIimp"]]
write.table(namesL,file="dataSim1fabia_individuals.txt",
   quote = FALSE,row.names = FALSE,col.names = FALSE)
write(as.integer(haploN),file="dataSim1fabia_annot.txt",
   ncolumns=100)
write(as.integer(snvs),file="dataSim1fabia_annot.txt",
   append=TRUE,ncolumns=100)
write.table(annot,file="dataSim1fabia_annot.txt",
   sep = " ",quote = FALSE,row.names = FALSE,
   col.names = FALSE,append=TRUE)
write(as.integer(haploN),file="dataSim1fabia_mat.txt",
   ncolumns=100)
write(as.integer(snvs),file="dataSim1fabia_mat.txt",
   append=TRUE,ncolumns=100)

for (i in 1:haploN) {

  a1 <- which(alleleIimp[i,]>0.01)

  al <- length(a1)
  b1 <- alleleIimp[i,a1]

  a1 <- a1 - 1
  dim(a1) <- c(1,al)
  b1 <- format(as.double(b1),nsmall=1)
  dim(b1) <- c(1,al)

  write.table(al,file="dataSim1fabia_mat.txt",
```

```
      sep = " ", quote = FALSE,row.names = FALSE,
      col.names = FALSE,append=TRUE)
  write.table(a1,file="dataSim1fabia_mat.txt",
      sep = " ", quote = FALSE,row.names = FALSE,
      col.names = FALSE,append=TRUE)
  write.table(b1,file="dataSim1fabia_mat.txt",
      sep = " ", quote = FALSE,row.names = FALSE,
      col.names = FALSE,append=TRUE)

}


hapRes <- hapFabia(fileName="dataSim1fabia",prefixPath="",
   sparseMatrixPostfix="_mat",
   annotPostfix="_annot.txt",individualsPostfix="_individuals.txt",
   labelsA=NULL,pRange="",individuals=0,lowerBP=0,upperBP=0.15,
   p=10,iter=1,quant=0.01,eps=1e-5,alpha=0.03,cyc=50,non_negative=1,
   write_file=0,norm=0,lap=100.0,IBDsegmentLength=10,Lt = 0.1,
   Zt = 0.2,thresCount=1e-5,mintagSNVsFactor=3/4,pMAF=0.1,
   haplotypes=FALSE,cut=0.8,procMinIndivids=0.1,thresPrune=1e-3,
   simv="minD",minTagSNVs=6,minIndivid=2,avSNVsDist=100,SNVclusterLength=100)

summary(hapRes$mergedIBDsegmentList)

plot(hapRes$mergedIBDsegmentList[[1]],filename="dataSim1fabia_mat")


### Another Example
simulateIBDsegmentsFabia(fileprefix="dataSim",
   minruns=2,maxruns=2,snvs=1000,individualsN=100,
   avDistSnvs=100,avDistMinor=10,noImplanted=1,
   implanted=10,length=50,minors=30,mismatches=0,
   mismatchImplanted=0.5,overlap=50)

hapRes <- hapFabia(fileName="dataSim2fabia",prefixPath="",
   sparseMatrixPostfix="_mat",
   annotPostfix="_annot.txt",individualsPostfix="_individuals.txt",
   labelsA=NULL,pRange="",individuals=0,lowerBP=0,upperBP=0.15,
   p=10,iter=1,quant=0.01,eps=1e-5,alpha=0.03,cyc=50,non_negative=1,
   write_file=0,norm=0,lap=100.0,IBDsegmentLength=10,Lt = 0.1,
   Zt = 0.2,thresCount=1e-5,mintagSNVsFactor=3/4,pMAF=0.1,
   haplotypes=FALSE,cut=0.8,procMinIndivids=0.1,thresPrune=1e-3,
   simv="minD",minTagSNVs=6,minIndivid=2,avSNVsDist=100,SNVclusterLength=100)

## Summary of the IBD segment list
summary(hapRes$mergedIBDsegmentList)

## Summary of the IBD segment
summary(hapRes$mergedIBDsegmentList[[1]])


## Plot an IBD segment
plot(hapRes$mergedIBDsegmentList[[1]],filename="dataSim2fabia_mat")

## Not run:
## It is interactive, thus dontrun!
```

```
## Plot an IBD segment list
plot(hapRes$mergedIBDsegmentList,filename="dataSim2fabia_mat")


## End(Not run)

setwd(old_dir)
```

---

hapFabiaVersion *Display version info for package* **hapFabia**

Description

hapFabiaVersion displays version information about the package.

Usage

```
hapFabiaVersion()
```

Value

Displays version information about the package

Author(s)

Sepp Hochreiter

See Also

IBDsegment-class, IBDsegmentList-class, analyzeIBDsegments, compareIBDsegmentLists, extractIBDsegments, findDenseRegions, hapFabia, hapFabiaVersion, hapRes, chr1ASW1000G, IBDsegmentList2excel, identifyDuplicates, iterateIntervals, makePipelineFile, matrixPlot, mergeIBDsegmentLists, mergedIBDsegmentList, plotIBDsegment, res, setAnnotation, setStatistics, sim, simu, simulateIBDsegmentsFabia, simulateIBDsegments, split_sparse_matrix, toolsFactorizationClas vcftoFABIA

Examples

```
hapFabiaVersion()
```

hapRes *Example result of* hapFabia

Description

Results of a hapFabia call. List containing

1. mergedIBDsegmentList: an object of the class IBDsegmentList that contains the extracted IBD segments that were extracted from two histograms with different offset.
2. res: the result of FABIA.
3. sPF: samples per loading of this FABIA result.
4. annot: annotation for the genotype data.
5. IBDsegmentList1: an object of the class IBDsegmentList that contains the result of IBD segment extraction from the first histogram.
6. IBDsegmentList2: an object of the class IBDsegmentList that contains the result of IBD segment extraction from the second histogram.
7. mergedIBDsegmentList1: an object of the class IBDsegmentList that contains the merged result of the first IBD segment extraction (redundancies removed).
8. mergedIBDsegmentList2: an object of the class IBDsegmentList that contains the merged result of the second IBD segment extraction (redundancies removed).

Usage

```
hapRes
```

Format

List containing

1. mergedIBDsegmentList: an object of the class IBDsegmentList that contains the extracted IBD segments that were extracted from two histograms with different offset.
2. res: the result of FABIA.
3. sPF: samples per loading of this FABIA result.
4. annot: annotation for the genotype data.
5. IBDsegmentList1: an object of the class IBDsegmentList that contains the result of IBD segment extraction from the first histogram.
6. IBDsegmentList2: an object of the class IBDsegmentList that contains the result of IBD segment extraction from the second histogram.
7. mergedIBDsegmentList1: an object of the class IBDsegmentList that contains the merged result of the first IBD segment extraction (redundancies removed).
8. mergedIBDsegmentList2: an object of the class IBDsegmentList that contains the merged result of the second IBD segment extraction (redundancies removed).

Source

result of a call of hapFabia

References

S. Hochreiter et al., 'FABIA: Factor Analysis for Bicluster Acquisition', Bioinformatics 26(12):1520-1527, 2010.

See Also

IBDsegment-class, IBDsegmentList-class, analyzeIBDsegments, compareIBDsegmentLists, extractIBDsegments, findDenseRegions, hapFabia, hapFabiaVersion, hapRes, chr1ASW1000G, IBDsegmentList2excel, identifyDuplicates, iterateIntervals, makePipelineFile, matrixPlot, mergeIBDsegmentLists, mergedIBDsegmentList, plotIBDsegment, res, setAnnotation, setStatistics, sim, simu, simulateIBDsegmentsFabia, simulateIBDsegments, split_sparse_matrix, toolsFactorizationClas vcftoFABIA

IBDsegment-class *IBDsegment instances and methods*

Description

IBDsegment is a class to store characteristics of an IBD segment in one of its instances. Characteristics of an IBD segment include its genomic position, its length, the individuals/chromosomes that belong to it, the tagSNVs that tag/mark it, etc.

Usage

```
## S4 method for signature 'IBDsegment'
plot(x,filename, ...)

## S4 method for signature 'IBDsegment'
plotLarger(x,filename,fact=1.0,addSamp=c(), ...)

## S4 method for signature 'IBDsegment'
summary(object, ...)
```

Arguments

x	object of the class IBDsegment.
object	object of the class IBDsegment.
filename	filename of the file that contains the genotyping data. Call of readSamplesSpfabia from the package **fabia** to read genotype data for the plot. ATTENTION: without file type ".txt"!
fact	factor by which the IBD segment is extended. The extension is done both at the left and at the right hand side.
addSamp	vector giving additional individuals, e.g. for adding 10 individuals out of N the code sample(N,10) may be used.
...	further arguments.

Details

plot Plots an IBD segment where the SNVs within the cluster are shown. In the plot the y-axis gives the individuals or the chromosomes and the x-axis consecutive SNVs. Minor alleles of tagSNVs are marked by a particular color. Minor alleles of other SNVs (private, common, from other clusters, etc) are marked by a different color. The model from **fabia** is also shown. The default color coding uses yellow for major alleles, violet for minor alleles of tagSNVs, and blue for minor alleles of other SNVs. `model L` indicates tagSNVs identified by hapFabia in violet.

plotLarger Plots an IBD segment where the SNVs within the cluster are shown, however additional individuals that do not possess the IBD segment can be added. Further the range can be increased by a certain factor. Additional arguments are `fact` and `addSamp`. `fact` gives the factor by which the IBD segment should be extended to both the left and the right. `addSamp` is a vector of individuals/haplotypes which are added to the individuals that possess the IBD segment. The plot allows to view the IBD segment in the context of its DNA location and additional individuals.

summary Prints the ID of the IBD segment, the bicluster it is derived from, the chromosome and position where it is located, its length, the number of individuals/chromosomes that belong to it, the number of tagSNVs that mark it.

Implementation in R.

Value

no value.

Slots

Objects of class `IBDsegment` have the following slots:

`ID` number of the IBD segment in the current extraction.

`bicluster_id` ID of the bicluster the IBD segment was found in.

`chromosome` the chromosome.

`IBDsegmentPos` genomic location of the IBD segment.

`IBDsegmentLength` length of the IBD segment in the number of SNVs. For the length in bp: `max(tagSNVPositions(x))-min(tagSNVPositions(x))`.

`numberIndividuals` number of samples belonging to the IBD segment.

`numbertagSNVs` number tagSNVs marking the IBD segment.

`individuals` IDs of individuals or chromosomes belonging to the IBD segment.

`tagSNVs` IDs of SNVs that mark the IBD segment (tagSNVs).

`populationIndividuals` the population each individual belongs to.

`idIndividuals` IDs of the individuals or chromosomes.

`labelIndividuals` label of the individuals.

`platformIndividuals` for each individual the technology/platform that was used to genotype it.

`coreClusterIndividuals` IDs of individuals that constitute the core of the IBD segment.

`tagSNVPositions` physical positions of the tagSNVs on the chromosome in base pairs.

`tagSNVAlleles` alleles of the tagSNVs in the form `Ref:Alt` where Ref denotes reference allele and Alt the alternative allele.

`tagSNVNames` name of the tagSNVs according to a given annotation.

`tagSNVFreq` frequency of tagSNVs in the whole data set.

`tagSNVGroupFreq` frequency of tagSNVs in the population that is considered.

`tagSNVChange` if the minor allele was more frequent than the major, then both were switched. Switching is marked by a 1 and otherwise it is 0.

`tagSNVsPerIndividual` for each sample: tagSNVs are counted for which the sample has the minor allele.

`individualPerTagSNV` for each tagSNV: samples are counted for which the SNV has its minor allele.

`tagSNVAnno` the functional annotation of tagSNVs for each tagSNV: like stop-loss, stop-gain, non-synonymous, synonymous, promoter, exonic, intronic, intergenic, etc.

Constructor

Constructor of class IBDsegment.

`IBDsegment(ID=0,bicluster_id=0,chromosome="",IBDsegmentPos=0,IBDsegmentLength=0,numberIndividual`

Accessors

In the following `x` denotes an IBDsegment object.

`ID(x)`, `ID(x) <- value`: Returns or sets `ID`, where the return value and `value` are both numeric.

`bicluster_id(x)`, `bicluster_id(x) <- value`: Returns or sets `bicluster_id`, where the return value and `value` are both numeric.

`chromosome(x)`, `chromosome(x) <- value`: Returns or sets `chromosome`, where the return value and `value` are both strings.

`IBDsegmentPos(x)`, `IBDsegmentPos(x) <- value`: Returns or sets `IBDsegmentPos`, where the return value and `value` are both numeric.

`IBDsegmentLength(x)`, `IBDsegmentLength(x) <- value`: Returns or sets `IBDsegmentLength`, where the return value and `value` are both numeric.

`numberIndividuals(x)`, `numberIndividuals(x) <- value`: Returns or sets `numberIndividuals`, where the return value and `value` are both numeric.

`numbertagSNVs(x)`, `numbertagSNVs(x) <- value`: Returns or sets `numbertagSNVs`, where the return value and `value` are both numeric.

`individuals(x)`, `individuals(x) <- value`: Returns or sets `individuals`, where the return value and `value` are both vectors.

`tagSNVs(x)`, `tagSNVs(x) <- value`: Returns or sets `tagSNVs`, where the return value and `value` are both vectors.

`populationIndividuals(x)`, `populationIndividuals(x) <- value`: Returns or sets `populationIndividuals`, where the return value and `value` are both vectors.

`idIndividuals(x)`, `idIndividuals(x) <- value`: Returns or sets `idIndividuals`, where the return value and `value` are both vectors.

`labelIndividuals(x)`, `labelIndividuals(x) <- value`: Returns or sets `labelIndividuals`, where the return value and `value` are both vectors.

`platformIndividuals(x)`, `platformIndividuals(x) <- value`: Returns or sets `platformIndividuals`, where the return value and `value` are both vectors.

`coreClusterIndividuals(x)`, `coreClusterIndividuals(x) <- value`: Returns or sets `coreClusterIndividuals`, where the return value and `value` are both vectors.

`tagSNVPositions(x)`, `tagSNVPositions(x) <- value`: Returns or sets `tagSNVPositions`, where the return value and `value` are both vectors.

`tagSNVAlleles(x)`, `tagSNVAlleles(x) <- value`: Returns or sets `tagSNVAlleles`, where the return value and `value` are both vectors.

`tagSNVNames(x)`, `tagSNVNames(x) <- value`: Returns or sets `tagSNVNames`, where the return value and `value` are both vectors.

`tagSNVFreq(x)`, `tagSNVFreq(x) <- value`: Returns or sets `tagSNVFreq`, where the return value and `value` are both vectors.

`tagSNVGroupFreq(x)`, `tagSNVGroupFreq(x) <- value`: Returns or sets `tagSNVGroupFreq`, where the return value and `value` are both vectors.

`tagSNVChange(x)`, `tagSNVChange(x) <- value`: Returns or sets `tagSNVChange`, where the return value and `value` are both vectors.

`tagSNVsPerIndividual(x)`, `tagSNVsPerIndividual(x) <- value`: Returns or sets `tagSNVsPerIndividual`, where the return value and `value` are both vectors.

`individualPerTagSNV(x)`, `individualPerTagSNV(x) <- value`: Returns or sets `individualPerTagSNV`, where the return value and `value` are both vectors.

`tagSNVAnno(x)`, `tagSNVAnno(x) <- value`: Returns or sets `tagSNVAnno`, where the return value and `value` are both vectors.

Signatures

plot `signature(x = "IBDsegment", y = "missing")` Plot of an IBD segment, where tagSNVs, minor and major alleles are plotted.

plotLarger `signature(x="IBDsegment", filename="character",fact="numeric",addSamp="ANY")` Plot of an IBD segment with additional individuals and the IBD segment extended to the left and to the right.

summary `signature(object = "IBDsegment")` Summary of IBD segment object.

Author(s)

Sepp Hochreiter

References

S. Hochreiter et al., 'FABIA: Factor Analysis for Bicluster Acquisition', Bioinformatics 26(12):1520-1527, 2010.

See Also

`IBDsegment-class`, `IBDsegmentList-class`, `analyzeIBDsegments`, `compareIBDsegmentLists`, `extractIBDsegments`, `findDenseRegions`, `hapFabia`, `hapFabiaVersion`, `hapRes`, `chr1ASW1000G`, `IBDsegmentList2excel`, `identifyDuplicates`, `iterateIntervals`, `makePipelineFile`, `matrixPlot`, `mergeIBDsegmentLists`, `mergedIBDsegmentList`, `plotIBDsegment`, `res`, `setAnnotation`, `setStatistics`, `sim`, `simu`, `simulateIBDsegmentsFabia`, `simulateIBDsegments`, `split_sparse_matrix`, `toolsFactorizationClas`
`vcftoFABIA`

Examples

```
old_dir <- getwd()
setwd(tempdir())

data(hapRes)
data(simu)
namesL <- simu[["namesL"]]
haploN <- simu[["haploN"]]
snvs <- simu[["snvs"]]
annot <- simu[["annot"]]
alleleIimp <- simu[["alleleIimp"]]
write.table(namesL,file="dataSim1fabia_individuals.txt",
   quote = FALSE,row.names = FALSE,col.names = FALSE)
write(as.integer(haploN),file="dataSim1fabia_annot.txt",
   ncolumns=100)
write(as.integer(snvs),file="dataSim1fabia_annot.txt",
   append=TRUE,ncolumns=100)
write.table(annot,file="dataSim1fabia_annot.txt",
   sep = " ", quote = FALSE,row.names = FALSE,
   col.names = FALSE,append=TRUE)
write(as.integer(haploN),file="dataSim1fabia_mat.txt",
   ncolumns=100)
write(as.integer(snvs),file="dataSim1fabia_mat.txt",
   append=TRUE,ncolumns=100)

for (i in 1:haploN) {

  a1 <- which(alleleIimp[i,]>0.01)

  al <- length(a1)
  b1 <- alleleIimp[i,a1]

  a1 <- a1 - 1
  dim(a1) <- c(1,al)
  b1 <- format(as.double(b1),nsmall=1)
  dim(b1) <- c(1,al)

  write.table(al,file="dataSim1fabia_mat.txt",
    sep = " ", quote = FALSE,row.names = FALSE,
    col.names = FALSE,append=TRUE)
  write.table(a1,file="dataSim1fabia_mat.txt",
    sep = " ", quote = FALSE,row.names = FALSE,
    col.names = FALSE,append=TRUE)
  write.table(b1,file="dataSim1fabia_mat.txt",
    sep = " ", quote = FALSE,row.names = FALSE,
    col.names = FALSE,append=TRUE)

}

mergedIBDsegmentList <- hapRes$mergedIBDsegmentList

IBDsegment <- mergedIBDsegmentList[[1]]


# Summary method
summary(IBDsegment)
```

```
# Plot method
plot(IBDsegment,filename="dataSim1fabia_mat")

# Extended plot: more examples and borders
plotLarger(IBDsegment,filename="dataSim1fabia_mat",3,sample(100,10))


# ACCESSORS

# IDs of the IBD segment
   ID(IBDsegment)
   bicluster_id(IBDsegment)

# General Information
    IBDsegmentPos(IBDsegment)
    IBDsegmentLength(IBDsegment)
    numberIndividuals(IBDsegment)
    numbertagSNVs(IBDsegment)
    coreClusterIndividuals(IBDsegment)

# Information on individuals / chromosomes
    individuals(IBDsegment)
    populationIndividuals(IBDsegment)
    idIndividuals(IBDsegment)
    labelIndividuals(IBDsegment)
    platformIndividuals(IBDsegment)
    tagSNVsPerIndividual(IBDsegment)

# Information on tagSNVs
    tagSNVs(IBDsegment)
    tagSNVPositions(IBDsegment)
    tagSNVAlleles(IBDsegment)
    tagSNVNames(IBDsegment)
    tagSNVFreq(IBDsegment)
    tagSNVGroupFreq(IBDsegment)
    tagSNVChange(IBDsegment)
    individualPerTagSNV(IBDsegment)
    tagSNVAnno(IBDsegment)


setwd(old_dir)
```

`IBDsegmentList-class` *IBDsegmentList instances and methods*

Description

`IBDsegmentList` is a class to store a list of IBD segments with its statistics. Lists can be merged or analyzed in subsequent steps.

Usage

```
## S4 method for signature 'IBDsegmentList'
plot(x, ... )


## S4 method for signature 'IBDsegmentList'
summary(object, ...)
```

Arguments

x	object of the class IBDsegment.
object	object of the class IBDsegment.
...	further arguments. PLOT: filename is the name of the file that contains the genotyping data used for plotting. The name is required for the call of readSamplesSpfabia from the package **fabia** to read genotype data for the plot. ATTENTION: filename without file type ".txt"!

Details

Plot Plots all IBD segments of an IBD segment list, where the SNVs within the cluster are shown. In the plot the y-axis gives the individuals or the chromosomes and the x-axis consecutive SNVs. Minor alleles of tagSNVs are marked by a particular color. Minor alleles of other SNVs (private, common, from other clusters, etc) are marked by a different color. The model from **fabia** is also shown. The default color coding uses yellow for major alleles, violet for minor alleles of tagSNVs, and blue for minor alleles of other SNVs. model L indicates tagSNVs identified by hapFabia in violet. Asks for the next plot by pushing a key.

Summary Prints the number of IBD segments in the list and some statistics if available.

Implementation in R.

Value

no value.

Slots

Objects of class IBDsegmentList have the following slots:

IBDsegments: List of IBD segments.

lengthList: Number of IBD segments in the list.

statistics: Statistics of IBD segments like average length, average number of individuals belonging to an IBD segment, average number of tagSNVs of an IBD segment, etc.

Constructor

Constructor of class IBDsegmentList.

```
IBDsegmentList(IBDsegments=list(),lengthList=0,statistics=list())
```

Accessors

In the following x denotes an IBDsegmentList object.

`IBDsegments(x)`, `IBDsegments(x) <- value`: Returns or sets `IBDsegments`, where the return value and `value` are both a list.

`lengthList(x)`, `lengthList(x) <- value`: Returns or sets `lengthList`, where the return value and `value` are both a number.

`statistics(x)`, `statistics(x) <- value`: Returns or sets `statistics`, where the return value and `value` are both a list.

`x[[i]]`, `x[[i]] <- value`: Returns or sets an entry in the list `x`, where the return value and `value` are both an instance of the class `IBDsegment`.

`x[i]`, `x[i] <- value`: Returns or sets a sublist of the list `x`, where the return value and `value` are both an instance of the class `IBDsegmentList`.

Signatures

plot `signature(x = "IBDsegmentList", y = "missing")` Plotting all IBD segments of the list using an interactive command.

summary `signature(object = "IBDsegmentList")` Summary of a list of IBD segments, where the number of clusters and a statistics are given.

Functions that return objects of this class

IBDsegmentList objects are returned by `fabia`, `fabias`, `fabiap`, `fabiasp`, `mfsc`, `nmfsc`, `nmfdiv`, and `nmfeu`.

Extension to store results of other methods

The class `IBDsegmentList` may contain the result of different matrix factorization methods. The methods may be generative or not.

Methods my be "singular value decomposition" (M contains singular values as well as avini, L and Z are orthonormal matrices), "independent component analysis" (Z contains the projection/sources, L is the mixing matrix, M is unity), "factor analysis" (Z contains factors, L the loadings, M is unity, U the noise, Psi the noise covariance, lapla is a variational parameter for non-Gaussian factors, avini and ini are the information the factors convey about the observations).

Author(s)

Sepp Hochreiter

References

S. Hochreiter et al., 'FABIA: Factor Analysis for Bicluster Acquisition', Bioinformatics 26(12):1520-1527, 2010.

See Also

`IBDsegment-class`, `IBDsegmentList-class`, `analyzeIBDsegments`, `compareIBDsegmentLists`, `extractIBDsegments`, `findDenseRegions`, `hapFabia`, `hapFabiaVersion`, `hapRes`, `chr1ASW1000G`, `IBDsegmentList2excel`, `identifyDuplicates`, `iterateIntervals`, `makePipelineFile`, `matrixPlot`, `mergeIBDsegmentLists`, `mergedIBDsegmentList`, `plotIBDsegment`, `res`, `setAnnotation`, `setStatistics`, `sim`, `simu`, `simulateIBDsegmentsFabia`, `simulateIBDsegments`, `split_sparse_matrix`, `toolsFactorizationClas`
`vcftoFABIA`

Examples

```
data(hapRes)

mergedIBDsegmentList <-
   hapRes$mergedIBDsegmentList

# Summary method
summary(mergedIBDsegmentList)

# Accessors
lengthList(mergedIBDsegmentList)

statistics(mergedIBDsegmentList)

summary(IBDsegments(mergedIBDsegmentList))


# Subsets

summary(mergedIBDsegmentList[[1]])

summary(mergedIBDsegmentList[1])

mergedIBDsegmentList[[2]] <-
   mergedIBDsegmentList[[1]]
mergedIBDsegmentList[[3]] <-
   hapRes$mergedIBDsegmentList2[[1]]

summary(mergedIBDsegmentList)

# mergedIBDsegmentList[c(3,4)] <-
#    mergedIBDsegmentList[c(1,2)]
# summary(mergedIBDsegmentList)
```

IBDsegmentList2excel *Store an IBD segment list in EXCEL / csv format*

Description

IBDsegmentList2excel: R implementation of IBDsegmentList2excel.

IBD segment list is stored in a file in EXCEL format, more precise in comma separated format (.csv).

Usage

```
## S4 method for signature 'IBDsegmentList,character'
IBDsegmentList2excel(IBDsegmentList,filename)
```

Arguments

IBDsegmentList	list of IBD segments given as an object of the class IBDsegmentList.
filename	name of the file where the IBD segment list is stored in EXCEL format.

Details

IBD segment list is stored in comma separate format (`.csv`) which can readily be read by EXCEL.

The EXCEL (`.csv`) file contains following columns:

1. `ID`: number of the IBD segment in the current extraction.
2. `bicluster_id`: ID of the bicluster the IBD segment was found in.
3. `chromosome`: the chromosome.
4. `IBDsegmentPos`: genomic location of the IBD segment.
5. `IBDsegmentLength`: length of the IBD segment.
6. `numberIndividuals`: number of samples belonging to the IBD segment.
7. `numbertagSNVs`: number tagSNVs marking the IBD segment.
8. `individuals`: IDs of individuals or chromosomes belonging to the IBD segment.
9. `tagSNVs`: IDs of SNVs that mark the IBD segment (tagSNVs).
10. `populationIndividuals`: the population each individual belongs to.
11. `idIndividuals`: IDs of the individuals or chromosomes.
12. `labelIndividuals`: label of the individuals.
13. `platformIndividuals`: for each individual the technology/platform that was used to genotype it.
14. `coreClusterIndividuals`: IDs of individuals that constitute the core of the IBD segment.
15. `tagSNVPositions`: physical positions of the tagSNVs on the chromosome in base pairs.
16. `tagSNVAlleles`: alleles of the tagSNVs in the form `Ref:Alt` where Ref denotes reference allele and Alt the alternative allele.
17. `tagSNVNames`: name of the tagSNVs according to a given annotation.
18. `tagSNVFreq`: frequency of tagSNVs in the whole data set.
19. `tagSNVGroupFreq`: frequency of tagSNVs in the population that is considered.
20. `tagSNVChange`: if the minor allele was more frequent than the major, then both were switched. Switching is marked by a 1 and otherwise it is 0.
21. `tagSNVsPerIndividual`: for each sample: tagSNVs are counted for which the sample has the minor allele.
22. `individualPerTagSNV`: for each tagSNV: samples are counted for which the SNV has its minor allele.
23. `tagSNVAnno`: the functional annotation of tagSNVs for each tagSNV: like stop-loss, stop-gain, non-synonymous, synonymous, promoter, exonic, intronic, intergenic, etc.

Implementation in R.

Value

writes to comma separated `.csv` file

Author(s)

Sepp Hochreiter

References

S. Hochreiter et al., 'FABIA: Factor Analysis for Bicluster Acquisition', Bioinformatics 26(12):1520-1527, 2010.

See Also

IBDsegment-class, IBDsegmentList-class, analyzeIBDsegments, compareIBDsegmentLists, extractIBDsegments, findDenseRegions, hapFabia, hapFabiaVersion, hapRes, chr1ASW1000G, IBDsegmentList2excel, identifyDuplicates, iterateIntervals, makePipelineFile, matrixPlot, mergeIBDsegmentLists, mergedIBDsegmentList, plotIBDsegment, res, setAnnotation, setStatistics, sim, simu, simulateIBDsegmentsFabia, simulateIBDsegments, split_sparse_matrix, toolsFactorizationClas
vcftoFABIA

Examples

```
old_dir <- getwd()
setwd(tempdir())

data(hapRes)
mergedIBDsegmentList <- hapRes$mergedIBDsegmentList
IBDsegmentList2excel(IBDsegmentList=mergedIBDsegmentList,
   filename="testResult.csv")

setwd(old_dir)
```

---

identifyDuplicates *Identify duplicates of IBD segments*

Description

identifyDuplicates: R implementation of identifyDuplicates.

IBD segments that are similar to each other are identified. This function is in combination with split_sparse_matrix whith splits a chromosome in overlapping intervals. These intervals are analyzed by iterateIntervals for IBD segments. Now, these IBD segments are checked for duplicates by identifyDuplicates. Results are written to the file "dups.Rda".

Usage

```
identifyDuplicates(fileName,startRun=1,endRun,
   shift=5000,intervalSize=10000)
```

Arguments

fileName	file name prefix without type of the result of hapFabia; Attention no type!
startRun	first interval.
endRun	last interval.
shift	distance between start of adjacent intervals.
intervalSize	number of SNVs in a interval.

Details

IBD segments that are similar to each other are identified and the result is written to the file "dups.Rda". For analysis across a whole chromosome this information is important in order to avoid multiple counting of features from the same IBD segment. Used subsequently to `iterateIntervals` which analyzes intervals of a chromosome for IBD segments. The information on duplicates (or similar IBD segments) is important for a subsequent run of `analyzeIBDsegments` to avoid redundancies.

Results are saved in "dups.Rda" which contains

1. `dups` (the index of duplicates),
2. `un` (the index of non-duplicates),
3. `countsA1` (the counts and mapping to intervals for non-duplicates), and
4. `countsA2` (the counts and mapping to intervals for all IBD segments).

Implementation in R.

Value

IBD segments that are similar to each other are identified

Author(s)

Sepp Hochreiter

References

S. Hochreiter et al., 'FABIA: Factor Analysis for Bicluster Acquisition', Bioinformatics 26(12):1520-1527, 2010.

See Also

`IBDsegment-class`, `IBDsegmentList-class`, `analyzeIBDsegments`, `compareIBDsegmentLists`, `extractIBDsegments`, `findDenseRegions`, `hapFabia`, `hapFabiaVersion`, `hapRes`, `chr1ASW1000G`, `IBDsegmentList2excel`, `identifyDuplicates`, `iterateIntervals`, `makePipelineFile`, `matrixPlot`, `mergeIBDsegmentLists`, `mergedIBDsegmentList`, `plotIBDsegment`, `res`, `setAnnotation`, `setStatistics`, `sim`, `simu`, `simulateIBDsegmentsFabia`, `simulateIBDsegments`, `split_sparse_matrix`, `toolsFactorizationClass`, `vcftoFABIA`

Examples

```
print("Identify duplicates of IBD segments")
## Not run:
########################################
## Already run in "iterateIntervals.Rd" ##
########################################

#Work in a temporary directory.

old_dir <- getwd()
setwd(tempdir())


# Load data and write to vcf file.
data(chr1ASW1000G)
```

```
write(chr1ASW1000G,file="chr1ASW1000G.vcf")

#Create the analysis pipeline for extracting IBD segments
makePipelineFile(fileName="chr1ASW1000G",shiftSize=500,intervalSize=1000,haplotypes=TRUE)

source("pipeline.R")

# Following files are produced:
list.files(pattern="chr1")



# Next we load interval 5 and there the first and second IBD segment
posAll <- 5
start <- (posAll-1)*shiftSize
end <- start + intervalSize
pRange <- paste("_",format(start,scientific=FALSE),"_",format(end,scientific=FALSE),sep="")
load(file=paste(fileName,pRange,"_resAnno",".Rda",sep=""))
IBDsegmentList <- resHapFabia$mergedIBDsegmentList
summary(IBDsegmentList)
IBDsegment1 <- IBDsegmentList[[1]]
summary(IBDsegment1)
IBDsegment2 <- IBDsegmentList[[2]]
summary(IBDsegment2)



#Plot the first IBD segment in interval 5
plot(IBDsegment1,filename=paste(fileName,pRange,"_mat",sep=""))


#Plot the second IBD segment in interval 5
plot(IBDsegment2,filename=paste(fileName,pRange,"_mat",sep=""))

setwd(old_dir)


## End(Not run)


## Not run:
###here an example of the the automatically generated pipeline
### with: shiftSize=5000,intervalSize=10000,fileName="filename"

#####define intervals, overlap, filename #######
shiftSize <- 5000
intervalSize <- 10000
fileName="filename" # without type
haplotypes <- TRUE
dosage <- FALSE

#####load library#######
library(hapFabia)

#####convert from .vcf to _mat.txt#######
vcftoFABIA(fileName=fileName)
```

```
#####copy haplotype, genotype, or dosage matrix to matrix#######
if (haplotypes) {
    file.copy(paste(fileName,"_matH.txt",sep=""), paste(fileName,"_mat.txt",sep=""))
} else {
    if (dosage) {
        file.copy(paste(fileName,"_matD.txt",sep=""), paste(fileName,"_mat.txt",sep=""))
    } else {
        file.copy(paste(fileName,"_matG.txt",sep=""), paste(fileName,"_mat.txt",sep=""))
    }
}

#####split/ generate intervals#######
split_sparse_matrix(fileName=fileName,intervalSize=intervalSize,
shiftSize=shiftSize,annotation=TRUE)

#####compute how many intervals we have#######
ina <- as.numeric(readLines(paste(fileName,"_mat.txt",sep=""),n=2))
noSNVs <- ina[2]
over <- intervalSize%/%shiftSize
N1 <- noSNVs%/%shiftSize
endRunA <- (N1-over+2)


#####analyze each interval#######
#####may be done by parallel runs#######
iterateIntervals(startRun=1,endRun=endRunA,shift=shiftSize,
intervalSize=intervalSize,fileName=fileName,individuals=0,
upperBP=0.05,p=10,iter=40,alpha=0.03,cyc=50,IBDsegmentLength=50,
Lt = 0.1,Zt = 0.2,thresCount=1e-5,mintagSNVsFactor=3/4,
pMAF=0.03,haplotypes=haplotypes,cut=0.8,procMinIndivids=0.1,thresPrune=1e-3,
simv="minD",minTagSNVs=6,minIndivid=2,avSNVsDist=100,SNVclusterLength=100)

#####identify duplicates#######
identifyDuplicates(fileName=fileName,startRun=1,endRun=endRunA,
shift=shiftSize,intervalSize=intervalSize)

#####analyze results; parallel#######
anaRes <- analyzeIBDsegments(fileName=fileName,startRun=1,endRun=endRunA,
shift=shiftSize,intervalSize=intervalSize)
print("Number IBD segments:")
print(anaRes$noIBDsegments)
print("Statistics on IBD segment length in SNVs (all SNVs in the IBD segment):")
print(anaRes$avIBDsegmentLengthSNVS)
print("Statistics on IBD segment length in bp:")
print(anaRes$avIBDsegmentLengthS)
print("Statistics on number of individuals belonging to IBD segments:")
print(anaRes$avnoIndividS)
print("Statistics on number of tagSNVs of IBD segments:")
print(anaRes$avnoTagSNVsS)
print("Statistics on MAF of tagSNVs of IBD segments:")
print(anaRes$avnoFreqS)
print("Statistics on MAF within the group of tagSNVs of IBD segments:")
print(anaRes$avnoGroupFreqS)
print("Statistics on number of changes between major and minor allele frequency:")
print(anaRes$avnotagSNVChangeS)
print("Statistics on number of tagSNVs per individual of an IBD segment:")
```

```
print(anaRes$avnotagSNVsPerIndividualS)
print("Statistics on number of individuals that have the minor allele of tagSNVs:")
print(anaRes$avnoindividualPerTagSNVS)

#####load result for interval 50#######
posAll <- 50 # (50-1)*5000 = 245000: interval 245000 to 255000
start <- (posAll-1)*shiftSize
end <- start + intervalSize
pRange <- paste("_",format(start,scientific=FALSE),"_",
format(end,scientific=FALSE),sep="")
load(file=paste(fileName,pRange,"_resAnno",".Rda",sep=""))
IBDsegmentList <- resHapFabia$mergedIBDsegmentList # $

summary(IBDsegmentList)
#####plot IBD segments in interval 50#######
plot(IBDsegmentList,filename=paste(fileName,pRange,"_mat",sep=""))
   ##attention: filename without type ".txt"

#####plot the first IBD segment in interval 50#######

IBDsegment <- IBDsegmentList[[1]]
plot(IBDsegment,filename=paste(fileName,pRange,"_mat",sep=""))
   ##attention: filename without type ".txt"


## End(Not run)
```

`iterateIntervals` *Loop over DNA intervals with a call of* `hapFabia`

Description

`iterateIntervals`: R implementation of `iterateIntervals`.

Loops over all intervals and calls `hapFabia` and then stores the results. Intervals have been generated by `split_sparse_matrix`.

Usage

```
iterateIntervals(startRun=1,endRun,shift=5000,intervalSize=10000,
   annotationFile=NULL,fileName,prefixPath="",
   sparseMatrixPostfix="_mat",annotPostfix="_annot.txt",
   individualsPostfix="_individuals.txt",individuals=0,
   lowerBP=0,upperBP=0.05,p=10,iter=40,quant=0.01,eps=1e-5,
   alpha=0.03,cyc=50,non_negative=1,write_file=0,norm=0,
   lap=100.0,IBDsegmentLength=50,Lt = 0.1,Zt = 0.2,
   thresCount=1e-5,mintagSNVsFactor=3/4,pMAF=0.03,
   haplotypes=FALSE,cut=0.8,procMinIndivids=0.1,thresPrune=1e-3,
   simv="minD",minTagSNVs=6,minIndivid=2,avSNVsDist=100,SNVclusterLength=100)
```

Arguments

startRun	first interval.
endRun	last interval.
shift	distance between starts of adjacent intervals.
intervalSize	number of SNVs in a interval.
annotationFile	file name of the annotation file for the individuals.
fileName	passed to hapFabia: file name of the genotype matrix in sparse format.
prefixPath	passed to hapFabia: path to the genotype file.
sparseMatrixPostfix	passed to hapFabia: postfix string for the sparse matrix.
annotPostfix	passed to hapFabia: postfix string for the SNV annotation file.
individualsPostfix	passed to hapFabia: postfix string for the file containing the names of the individuals.
individuals	passed to hapFabia: vector of individuals which are included into the analysis; default = 0 (all individuals).
lowerBP	passed to hapFabia: lower bound on minor allele frequencies (MAF); however at least two occurrences are required to remove private SNVs.
upperBP	passed to hapFabia: upper bound on minor allele frequencies (MAF) to extract rare variants.
p	passed to hapFabia: number of biclusters per iteration.
iter	passed to hapFabia: number of iterations.
quant	passed to hapFabia: percentage of loadings L to remove in each iteration.
eps	passed to hapFabia: lower bound for variational parameter lapla; default 1e-5.
alpha	passed to hapFabia: sparseness of the loadings; default = 0.03.
cyc	passed to hapFabia: number of cycles per iterations; default 50.
non_negative	passed to hapFabia: non-negative factors and loadings if non_negative = 1; default = 1 (yes).
write_file	passed to hapFabia: results are written to files (L in sparse format), default = 0 (not written).
norm	passed to hapFabia: data normalization; default 0 (no normalization).
lap	passed to hapFabia: minimal value of the variational parameter; default 100.0.
IBDsegmentLength	passed to hapFabia: typical IBD segment length in kbp.
Lt	passed to hapFabia: percentage of largest Ls to consider for IBD segment extraction.
Zt	passed to hapFabia: percentage of largest Zs to consider for IBD segment extraction.
thresCount	passed to hapFabia: p-value of random histogram hit; default 1e-5.
mintagSNVsFactor	passed to hapFabia: percentage of IBD segment overlap; default 3/4.
pMAF	passed to hapFabia: averaged and corrected (for non-uniform distributions) minor allele frequency.

`haplotypes` passed to hapFabia: haplotypes = TRUE then phased genotypes meaning two chromosomes per individual otherwise unphased genotypes.

`cut` passed to hapFabia: cutoff for merging IBD segments after a hierarchical clustering; default 0.8.

`procMinIndivids` passed to hapFabia: percentage of cluster individuals a tagSNV must tag to be considered as tagSNV for the IBD segment.

`thresPrune` passed to hapFabia: threshold for pruning border tagSNVs based on an exponential distribution where border tagSNVs with large distances to the next tagSNV are pruned.

`simv` passed to hapFabia: similarity measure for merging clusters: `"minD"` (percentage of smaller explained by larger set), `"jaccard"` (Jaccard index), `"dice"` (Dice index), or `"maxD"`; default `"minD"`.

`minTagSNVs` passed to hapFabia: minimum matching tagSNVs for cluster similarity; otherwise the similarity is set to zero.

`minIndivid` passed to hapFabia: minimum matching individuals for cluster similarity; otherwise the similarity is set to zero.

`avSNVsDist` passed to hapFabia: average distance between SNVs in base pairs - used together with `IBDsegmentLength` to compute the number of SNVs in the histogram bins; default=100.

`SNVclusterLength` passed to hapFabia: if `IBDsegmentLength=0` then the number of SNVs in the histogram bins can be given directly; default 100.

Details

Implementation in R. Reads annotation of the individuals if available, then calls `hapFabia` and stores its results. Results are saved in EXCEL format and as R binaries.

`iterateIntervals` loops over all intervals and calls `hapFabia` and then stores the results. Intervals have been generated by `split_sparse_matrix`. The results are the indentified IBD segments which are stored separately per interval. A subsequent analysis first calls `identifyDuplicates` to identify IBD segments that are found more than one time and then analyzes the IBD segments by `analyzeIBDsegments`.

The SNV annotation file `..._annot.txt` contains:

1. first line: number individuals;
2. second line: number SNVs;
3. for each SNV a line containing following field that are blank separated: "chromosome", "physical position", "snvNames", "snvMajor", "snvMinor", "quality", "pass", "info of vcf file", "fields in vcf file", "frequency", "0/1: 1 is changed if major allele is actually minor allele".

The individuals annotation file, which name is give to `annotationFile`, contains per individual a tab separated line with

1. id;
2. subPopulation;
3. population;
4. platform.

Value

Loop over DNA intervals with a call of hapFabia

Author(s)

Sepp Hochreiter

References

S. Hochreiter et al., 'FABIA: Factor Analysis for Bicluster Acquisition', Bioinformatics 26(12):1520-1527, 2010.

See Also

IBDsegment-class, IBDsegmentList-class, analyzeIBDsegments, compareIBDsegmentLists, extractIBDsegments, findDenseRegions, hapFabia, hapFabiaVersion, hapRes, chr1ASW1000G, IBDsegmentList2excel, identifyDuplicates, iterateIntervals, makePipelineFile, matrixPlot, mergeIBDsegmentLists, mergedIBDsegmentList, plotIBDsegment, res, setAnnotation, setStatistics, sim, simu, simulateIBDsegmentsFabia, simulateIBDsegments, split_sparse_matrix, toolsFactorizationClas
vcftoFABIA

Examples

```
## Not run:

###here an example of the the automatically generated pipeline
### with: shiftSize=5000,intervalSize=10000,fileName="filename"


#####define intervals, overlap, filename #######
shiftSize <- 5000
intervalSize <- 10000
fileName="filename" # without type
haplotypes <- TRUE
dosage <- FALSE

#####load library#######
library(hapFabia)

#####convert from .vcf to _mat.txt#######
vcftoFABIA(fileName=fileName)

#####copy haplotype, genotype, or dosage matrix to matrix#######
if (haplotypes) {
    file.copy(paste(fileName,"_matH.txt",sep=""), paste(fileName,"_mat.txt",sep=""))
} else {
    if (dosage) {
        file.copy(paste(fileName,"_matD.txt",sep=""), paste(fileName,"_mat.txt",sep=""))
    } else {
        file.copy(paste(fileName,"_matG.txt",sep=""), paste(fileName,"_mat.txt",sep=""))
    }
}

#####split/ generate intervals#######
split_sparse_matrix(fileName=fileName,intervalSize=intervalSize,
```

```
shiftSize=shiftSize,annotation=TRUE)

#####compute how many intervals we have#######
ina <- as.numeric(readLines(paste(fileName,"_mat.txt",sep=""),n=2))
noSNVs <- ina[2]
over <- intervalSize%/%shiftSize
N1 <- noSNVs%/%shiftSize
endRunA <- (N1-over+2)

#####analyze each interval#######
#####may be done by parallel runs#######
iterateIntervals(startRun=1,endRun=endRunA,shift=shiftSize,
intervalSize=intervalSize,fileName=fileName,individuals=0,
upperBP=0.05,p=10,iter=40,alpha=0.03,cyc=50,IBDsegmentLength=50,
Lt = 0.1,Zt = 0.2,thresCount=1e-5,mintagSNVsFactor=3/4,
pMAF=0.035,haplotypes=haplotypes,cut=0.8,procMinIndivids=0.1,thresPrune=1e-3,
simv="minD",minTagSNVs=6,minIndivid=2,avSNVsDist=100,SNVclusterLength=100)

#####identify duplicates#######
identifyDuplicates(fileName=fileName,startRun=1,endRun=endRunA,
shift=shiftSize,intervalSize=intervalSize)

#####analyze results; parallel#######
anaRes <- analyzeIBDsegments(fileName=fileName,startRun=1,endRun=endRunA,
shift=shiftSize,intervalSize=intervalSize)
print("Number IBD segments:")
print(anaRes$noIBDsegments)
print("Statistics on IBD segment length in SNVs (all SNVs in the IBD segment):")
print(anaRes$avIBDsegmentLengthSNVS)
print("Statistics on IBD segment length in bp:")
print(anaRes$avIBDsegmentLengthS)
print("Statistics on number of individuals belonging to IBD segments:")
print(anaRes$avnoIndividS)
print("Statistics on number of tagSNVs of IBD segments:")
print(anaRes$avnoTagSNVsS)
print("Statistics on MAF of tagSNVs of IBD segments:")
print(anaRes$avnoFreqS)
print("Statistics on MAF within the group of tagSNVs of IBD segments:")
print(anaRes$avnoGroupFreqS)
print("Statistics on number of changes between major and minor allele frequency:")
print(anaRes$avnotagSNVChangeS)
print("Statistics on number of tagSNVs per individual of an IBD segment:")
print(anaRes$avnotagSNVsPerIndividualS)
print("Statistics on number of individuals that have the minor allele of tagSNVs:")
print(anaRes$avnoindividualPerTagSNVS)

#####load result for interval 50#######
posAll <- 50 # (50-1)*5000 = 245000: interval 245000 to 255000
start <- (posAll-1)*shiftSize
end <- start + intervalSize
pRange <- paste("_",format(start,scientific=FALSE),"_",
format(end,scientific=FALSE),sep="")
load(file=paste(fileName,pRange,"_resAnno",".Rda",sep=""))
IBDsegmentList <- resHapFabia$mergedIBDsegmentList # $

summary(IBDsegmentList)
#####plot IBD segments in interval 50#######
```

```
plot(IBDsegmentList,filename=paste(fileName,pRange,"_mat",sep=""))
   ##attention: filename without type ".txt"

#####plot the first IBD segment in interval 50#######

IBDsegment <- IBDsegmentList[[1]]
plot(IBDsegment,filename=paste(fileName,pRange,"_mat",sep=""))
   ##attention: filename without type ".txt"


## End(Not run)

#Work in a temporary directory.

old_dir <- getwd()
setwd(tempdir())


# Load data and write to vcf file.
data(chr1ASW1000G)
write(chr1ASW1000G,file="chr1ASW1000G.vcf")

#Create the analysis pipeline for haplotype data (1000Genomes)
makePipelineFile(fileName="chr1ASW1000G",shiftSize=500,intervalSize=1000,haplotypes=TRUE)

source("pipeline.R")

# Following files are produced:
list.files(pattern="chr1")



# Next we load interval 5 and there the first and second IBD segment
posAll <- 5
start <- (posAll-1)*shiftSize
end <- start + intervalSize
pRange <- paste("_",format(start,scientific=FALSE),"_",format(end,scientific=FALSE),sep="")
load(file=paste(fileName,pRange,"_resAnno",".Rda",sep=""))
IBDsegmentList <- resHapFabia$mergedIBDsegmentList
summary(IBDsegmentList)
IBDsegment1 <- IBDsegmentList[[1]]
summary(IBDsegment1)
IBDsegment2 <- IBDsegmentList[[2]]
summary(IBDsegment2)




#Plot the first IBD segment in interval 5
plot(IBDsegment1,filename=paste(fileName,pRange,"_mat",sep=""))


#Plot the second IBD segment in interval 5
plot(IBDsegment2,filename=paste(fileName,pRange,"_mat",sep=""))

setwd(old_dir)
```

makePipelineFile *Generate* pipleline.R

Description

makePipelineFile creates pipleline.R for sourcing with source("pipleline.R") to run a whole IBD segment extraction pipeline.

Usage

```
makePipelineFile(fileName,shiftSize=5000,intervalSize=10000,haplotypes=FALSE,dosage=FALSE)
```

Arguments

fileName	file name of the genotype file in vcf format
shiftSize	distance between start of adjacent intervals.
intervalSize	number of SNVs in a interval.
haplotypes	should haplotypes (phased genotypes) be used; default FALSE.
dosage	should dosages be used if haplotypes is FALSE; default FALSE.

Details

makePipelineFile creates Pipleline.R for sourcing with source("pipleline.R") to run a whole IBD segment extraction pipeline.

Attention: this code may run a while for large data sets.

Value

Run a whole IBD segment extraction pipeline

Author(s)

Sepp Hochreiter

References

S. Hochreiter et al., 'FABIA: Factor Analysis for Bicluster Acquisition', Bioinformatics 26(12):1520-1527, 2010.

See Also

IBDsegment-class, IBDsegmentList-class, analyzeIBDsegments, compareIBDsegmentLists, extractIBDsegments, findDenseRegions, hapFabia, hapFabiaVersion, hapRes, chr1ASW1000G, IBDsegmentList2excel, identifyDuplicates, iterateIntervals, makePipelineFile, matrixPlot, mergeIBDsegmentLists, mergedIBDsegmentList, plotIBDsegment, res, setAnnotation, setStatistics, sim, simu, simulateIBDsegmentsFabia, simulateIBDsegments, split_sparse_matrix, toolsFactorizationClas
vcftoFABIA

Examples

```
old_dir <- getwd()
setwd(tempdir())

makePipelineFile(fileName="genotypeData",
   shiftSize=500,intervalSize=1000)
a <- scan(file = "pipeline.R",
   what = "character")
cat(a)

setwd(old_dir)
```

`matrixPlot` *Basic plot function for IBD segments*

Description

`matrixPlot`: R implementation of `matrixPlot`.

Plots a matrix where different values are coded by different colors. Basically the image plot function `image` with a particular scaling, color coding, and axis.

Usage

```
matrixPlot(x,range=NULL,yLabels=NULL,zlim=NULL,title=NULL,colRamp=12,grid=FALSE,pairs=FALSE,padj
```

Arguments

`x`	matrix that codes alleles and annotations of an IBD segment.
`range`	optional: physical range of the IBD segment.
`yLabels`	optional: labels of the individuals.
`zlim`	optional: limits imposed onto the matrix values.
`title`	title of the plot.
`colRamp`	color representation.
`grid`	does the plot have a grid?; default FALSE (no).
`pairs`	for pairwise groups, e.g. case-control, twins, etc.; default FALSE (no).
`padj`	adjustment for each tick label perpendicular to the reading direction.
`...`	other graphical parameters may also be passed as arguments to this function.

Details

Implementation in R.

Value

Plots a matrix where different values are coded by different colors

Author(s)

Sepp Hochreiter

References

S. Hochreiter et al., 'FABIA: Factor Analysis for Bicluster Acquisition', Bioinformatics 26(12):1520-1527, 2010.

See Also

IBDsegment-class, IBDsegmentList-class, analyzeIBDsegments, compareIBDsegmentLists, extractIBDsegments, findDenseRegions, hapFabia, hapFabiaVersion, hapRes, chr1ASW1000G, IBDsegmentList2excel, identifyDuplicates, iterateIntervals, makePipelineFile, matrixPlot, mergeIBDsegmentLists, mergedIBDsegmentList, plotIBDsegment, res, setAnnotation, setStatistics, sim, simu, simulateIBDsegmentsFabia, simulateIBDsegments, split_sparse_matrix, toolsFactorizationClas
vcftoFABIA

Examples

```
mat <- matrix(0,nrow=10,ncol=40)
v1 <- sample(1:10,5)
v21 <- sample(1:40,4)
v22 <- sample(1:40,4)
w1 <- rep(0,10)
w2 <- rep(0,40)
w1[v1] <- 1
w2[v21] <- 1
w2[v22] <- 2
mat <- mat + tcrossprod(w1,w2)

matrixPlot(mat)
```

---

mergedIBDsegmentList *Example IBD segment list as a result of* hapFabia

---

Description

Result of a hapFabia call given as instance of the class IBDsegmentList. The object mergedIBDsegmentList contains the extracted IBD segments.

Usage

```
mergedIBDsegmentList
```

Format

object mergedIBDsegmentList of the class IBDsegmentList

Source

result of a call of hapFabia

References

S. Hochreiter et al., 'FABIA: Factor Analysis for Bicluster Acquisition', Bioinformatics 26(12):1520-1527, 2010.

See Also

IBDsegment-class, IBDsegmentList-class, analyzeIBDsegments, compareIBDsegmentLists, extractIBDsegments, findDenseRegions, hapFabia, hapFabiaVersion, hapRes, chr1ASW1000G, IBDsegmentList2excel, identifyDuplicates, iterateIntervals, makePipelineFile, matrixPlot, mergeIBDsegmentLists, mergedIBDsegmentList, plotIBDsegment, res, setAnnotation, setStatistics, sim, simu, simulateIBDsegmentsFabia, simulateIBDsegments, split_sparse_matrix, toolsFactorizationClas vcftoFABIA

---

mergeIBDsegmentLists *Merging IBD segments*

Description

mergeIBDsegmentLists: R implementation of mergeIBDsegmentLists.

Merges and combines the IBD segments of one or two list(s). A vector gives for each cluster in the IBD segment list its new cluster membership as an integer number. Called by hapFabia where new membership is determined by hierarchical clustering.

Usage

```
## S4 method for signature 'IBDsegmentList,ANY,vector'
mergeIBDsegmentLists(IBDsegmentList1,IBDsegmentList2=NULL,clustIBDsegmentList)
```

Arguments

IBDsegmentList1	object of class IBDsegmentList.
IBDsegmentList2	optional: second object of class IBDsegmentList.
clustIBDsegmentList	vector giving for each cluster in the IBD segment list its new cluster membership as an integer number.

Details

A vector gives for each IBD segment its new membership as an integer number. IBD segments that belong to the same new cluster are merged.

Implementation in R.

Value

IBDsegmentListmerge	object of IBDsegmentList containing the merged IBD segments.

Author(s)

Sepp Hochreiter

References

S. Hochreiter et al., 'FABIA: Factor Analysis for Bicluster Acquisition', Bioinformatics 26(12):1520-1527, 2010.

See Also

IBDsegment-class, IBDsegmentList-class, analyzeIBDsegments, compareIBDsegmentLists, extractIBDsegments, findDenseRegions, hapFabia, hapFabiaVersion, hapRes, chr1ASW1000G, IBDsegmentList2excel, identifyDuplicates, iterateIntervals, makePipelineFile, matrixPlot, mergeIBDsegmentLists, mergedIBDsegmentList, plotIBDsegment, res, setAnnotation, setStatistics, sim, simu, simulateIBDsegmentsFabia, simulateIBDsegments, split_sparse_matrix, toolsFactorizationClas
vcftoFABIA

Examples

```
data(hapRes)
IBDsegmentList1 <- hapRes$IBDsegmentList1
IBDsegmentList2 <- hapRes$IBDsegmentList2
comp <-
compareIBDsegmentLists(IBDsegmentList1,
   IBDsegmentList2,simv="minD",pTagSNVs=NULL,
   pIndivid=NULL,minTagSNVs=6,minIndivid=2)
if (!is.null(comp)) {
  clustIBDsegmentList <- cutree(comp,h=0.8)
  mergedIBDsegmentList <-
     mergeIBDsegmentLists(IBDsegmentList1=
     IBDsegmentList1,IBDsegmentList2=
     IBDsegmentList2,clustIBDsegmentList=
     clustIBDsegmentList)
}
summary(IBDsegmentList1)
summary(IBDsegmentList2)

summary(mergedIBDsegmentList)
print(IBDsegmentPos(mergedIBDsegmentList[[1]]))
print(IBDsegmentLength(mergedIBDsegmentList[[1]]))
```

`plotIBDsegment` *Plots an IBD segment given genotype data and tagSNVs*

Description

`plotIBDsegment`: R implementation of `plotIBDsegment`.

A IBD segment is plotted where individuals that are plotted must be provided together with tagSNVs and their physical positions. The individuals are provided as genotyping matrix. The model, i.e. the tagSNVs, is shown, too. Annotations of tagSNVs like match with another genome (Neandertal, Denisova) can be visualized.

Usage

```
plotIBDsegment(Lout,tagSNV,physPos=NULL,colRamp=12,val=c(0.0,2.0,1.0),chrom="",count=0,labelsNA=
```

Arguments

Lout	the alleles of the individuals are provided; typically via a previous call of the **fabia** function readSamplesSpfabia.
tagSNV	tagSNVs given as numbers if all SNVs are enumerated.
physPos	physical position of tagSNVs in bp.
colRamp	passed to matrixPlot: color representation.
val	vector of 3 components containing values for representation of reference allele, minor allele that is not tagSNV, minor allele that is tagSNV, respectively.
chrom	chromosome as string.
count	counter which is shown in the title if larger than 0 (for viewing a series of IBD segments).
labelsNA	labels for the individuals.
prange	vector of two integer values giving the begin and end of a subinterval of the interval for zooming in.
labelsNA1	labels for tagSNVs by the model obtained by **fabia**; other tagSNV annotations like match with another genome (Neandertal, Denisova) can be visualized.
grid	does the plot have a grid?; default FALSE (no).
pairs	for pairwise groups, e.g. case-control, twins, etc.; default FALSE (no).
...	other graphical parameters may also be passed as arguments to this function.

Details

A IBD segment is plotted showing tagSNVs and minor alleles of other SNVs. Provided are individuals to plot together with tagSNVs and their physical positions. Other annotations of tagSNVs can be visualized.

In the plot the y-axis gives the individuals or the chromosomes and the x-axis consecutive SNVs. The default color coding uses yellow for major alleles, violet for minor alleles of tagSNVs, and blue for minor alleles of other SNVs. model L indicates tagSNVs identified by hapFabia in violet.

Implementation in R.

Value

Plots an IBD segment given genotype data and tagSNVs

Author(s)

Sepp Hochreiter

References

S. Hochreiter et al., 'FABIA: Factor Analysis for Bicluster Acquisition', Bioinformatics 26(12):1520-1527, 2010.

See Also

IBDsegment-class, IBDsegmentList-class, analyzeIBDsegments, compareIBDsegmentLists, extractIBDsegments, findDenseRegions, hapFabia, hapFabiaVersion, hapRes, chr1ASW1000G, IBDsegmentList2excel, identifyDuplicates, iterateIntervals, makePipelineFile, matrixPlot, mergeIBDsegmentLists, mergedIBDsegmentList, plotIBDsegment, res, setAnnotation, setStatistics, sim, simu, simulateIBDsegmentsFabia, simulateIBDsegments, split_sparse_matrix, toolsFactorizationClas vcftoFABIA

Examples

```
old_dir <- getwd()
setwd(tempdir())

data(hapRes)

data(simu)

namesL <- simu[["namesL"]]
haploN <- simu[["haploN"]]
snvs <- simu[["snvs"]]
annot <- simu[["annot"]]
alleleImp <- simu[["alleleImp"]]
write.table(namesL,file="dataSim1fabia_individuals.txt",
   quote = FALSE,row.names = FALSE,col.names = FALSE)
write(as.integer(haploN),file="dataSim1fabia_annot.txt",
   ncolumns=100)
write(as.integer(snvs),file="dataSim1fabia_annot.txt",
   append=TRUE,ncolumns=100)
write.table(annot,file="dataSim1fabia_annot.txt",
   sep = " ", quote = FALSE,row.names = FALSE,
   col.names = FALSE,append=TRUE)
write(as.integer(haploN),file="dataSim1fabia_mat.txt",
   ncolumns=100)
write(as.integer(snvs),file="dataSim1fabia_mat.txt",
   append=TRUE,ncolumns=100)

for (i in 1:haploN) {

  a1 <- which(alleleImp[i,]>0.01)

  al <- length(a1)
  b1 <- alleleImp[i,a1]

  a1 <- a1 - 1
  dim(a1) <- c(1,al)
  b1 <- format(as.double(b1),nsmall=1)
  dim(b1) <- c(1,al)

  write.table(al,file="dataSim1fabia_mat.txt",
     sep = " ", quote = FALSE,row.names = FALSE,
     col.names = FALSE,append=TRUE)
  write.table(a1,file="dataSim1fabia_mat.txt",
     sep = " ", quote = FALSE,row.names = FALSE,
     col.names = FALSE,append=TRUE)
  write.table(b1,file="dataSim1fabia_mat.txt",
     sep = " ", quote = FALSE,row.names = FALSE,
```

```
     col.names = FALSE,append=TRUE)

}



mergedIBDsegmentList <- hapRes$mergedIBDsegmentList
individuals <- individuals(mergedIBDsegmentList[[1]])
tagSNVs <- tagSNVs(mergedIBDsegmentList[[1]])
tagSNVs <-
   as.integer(sort.int(as.integer(unique(tagSNVs))))
tagSNVPositions <-
   tagSNVPositions(mergedIBDsegmentList[[1]])
labelIndividuals <-
   labelIndividuals(mergedIBDsegmentList[[1]])
Lout <- readSamplesSpfabia(X="dataSim1fabia_mat",
   samples=individuals,lowerB=0,upperB=1000.0)
tagSNVsL <- list(tagSNVs)
labelsK <- c("model L")
plotIBDsegment(Lout=Lout,tagSNV=tagSNVsL,
   physPos=tagSNVPositions,colRamp=12,val=c(0.0,2.0,1.0),
   chrom="1",count=0,labelsNA=labelIndividuals,
   labelsNA1=labelsK)

setwd(old_dir)
```

`res` *Example result of* `spfabia`

Description

Result of a **fabia** call.

Usage

```
res
```

Format

class "Factorization"

Source

result of a call of `spfabia`

References

S. Hochreiter et al., 'FABIA: Factor Analysis for Bicluster Acquisition', Bioinformatics 26(12):1520-1527, 2010.

See Also

IBDsegment-class, IBDsegmentList-class, analyzeIBDsegments, compareIBDsegmentLists, extractIBDsegments, findDenseRegions, hapFabia, hapFabiaVersion, hapRes, chr1ASW1000G, IBDsegmentList2excel, identifyDuplicates, iterateIntervals, makePipelineFile, matrixPlot, mergeIBDsegmentLists, mergedIBDsegmentList, plotIBDsegment, res, setAnnotation, setStatistics, sim, simu, simulateIBDsegmentsFabia, simulateIBDsegments, split_sparse_matrix, toolsFactorizationClas vcftoFABIA

`setAnnotation` *Fills in annotations of tagSNVs of a list of IBD segments*

Description

`setAnnotation`: R implementation of `setAnnotation`.

Fills in the tagSNV annotation of IBD segments given in an object of the class `IBDsegmentList`. The annotation must be given in a file where the first column contains the position of the SNV and the second the chromosome. The other columns give the annotation like "stop gain", "stop loss", "synonymous", "non-synonymous", "exonic", "intronic", "intergenic", "promotor", etc. However other annotations like whether the minor allele is identical to the Denisova or Neandertal base can be included.

Usage

```
## S4 method for signature 'IBDsegmentList,character'
setAnnotation(IBDsegmentList,filename)
```

Arguments

`IBDsegmentList`	object of class `IBDsegmentList`.
`filename`	File containing the SNV annotations, where the first column contains the SNV position and the second the chromosome.

Details

Implementation in R.

Value

object of class `IBDsegmentList` in which the annotation for the tagSNVs is set.

Author(s)

Sepp Hochreiter

References

S. Hochreiter et al., 'FABIA: Factor Analysis for Bicluster Acquisition', Bioinformatics 26(12):1520-1527, 2010.

See Also

IBDsegment-class, IBDsegmentList-class, analyzeIBDsegments, compareIBDsegmentLists, extractIBDsegments, findDenseRegions, hapFabia, hapFabiaVersion, hapRes, chr1ASW1000G, IBDsegmentList2excel, identifyDuplicates, iterateIntervals, makePipelineFile, matrixPlot, mergeIBDsegmentLists, mergedIBDsegmentList, plotIBDsegment, res, setAnnotation, setStatistics, sim, simu, simulateIBDsegmentsFabia, simulateIBDsegments, split_sparse_matrix, toolsFactorizationClas vcftoFABIA

Examples

```
old_dir <- getwd()
setwd(tempdir())

data(hapRes)

res <- hapRes$res
sPF <- hapRes$sPF
annot <- hapRes$annot
nnL <- length(Z(res)[1,])
labelsA <- cbind(as.character(1:nnL),
   as.character(1:nnL),as.character(1:nnL),
   as.character(1:nnL))
resIBDsegmentList <-
  extractIBDsegments(res=res,sPF=sPF,
  annot=annot,chrom="1",labelsA=labelsA,
  ps=0.9,psZ=0.8,inteA=50,thresA=6,mintagSNVs=6,
  off=0,procMinIndivids=0.1,thresPrune=1e-3)


tagSNVPositions <-
   tagSNVPositions(resIBDsegmentList[[1]])
snvR <- sample(min(tagSNVPositions):max(tagSNVPositions),
   length(tagSNVPositions))
snvA <- sort(unique(c(tagSNVPositions,snvR)))

func = c("stopGain","stopLoss","synonymous",
   "non-synonymous","-","-","-","-","-","-")
for (i in 1:length(snvA)) {

if (i>1) {
write(paste(snvA[i],"1",sample(func,1),sep=" "),
   file="snvAnnotation.txt",ncolumns=100,append=TRUE)
} else {

write(paste(snvA[i],"1",sample(func,1),sep=" "),
   file="snvAnnotation.txt",ncolumns=100,append=FALSE)
}


}

tagSNVAnno(resIBDsegmentList[[1]])

resIBDsegmentList <- setAnnotation(resIBDsegmentList,
   filename="snvAnnotation.txt")
```

```
tagSNVAnno(resIBDsegmentList[[1]])

setwd(old_dir)
```

`setStatistics` *Computes and stores the statistics of an IBD segment list*

Description

`setStatistics`: R implementation of `setStatistics`.

Computes the statistics of an IBD segment list given as an object of the class `IBDsegmentList`. In the slot `statistics` of an object of the class `IBDsegmentList` the summary statistics across the list of IBD segments are stored. Following characteristics are stored in the list `statistics`:

1. "avIBDsegmentPosS": physical position
2. "avIBDsegmentLengthSNVS": length in SNVs
3. "avIBDsegmentLengthS": length in bp
4. "avnoIndividS": number individuals
5. "avnoTagSNVsS": number tagSNVs
6. "avnoFreqS": tagSNV frequency
7. "avnoGroupFreqS": tagSNV group frequency
8. "avnotagSNVChangeS": tagSNV change between minor and major allele
9. "avnotagSNVsPerIndividualS": tagSNVs per individual
10. "avnoindividualPerTagSNVS": individuals per tagSNV.

Usage

```
## S4 method for signature 'IBDsegmentList'
setStatistics(IBDsegmentList)
```

Arguments

`IBDsegmentList`	object of class `IBDsegmentList`.

Details

The list can be extended by the user.

Implementation in R.

Value

object of class `IBDsegmentList` with statistics set

Author(s)

Sepp Hochreiter

References

S. Hochreiter et al., 'FABIA: Factor Analysis for Bicluster Acquisition', Bioinformatics 26(12):1520-1527, 2010.

See Also

IBDsegment-class, IBDsegmentList-class, analyzeIBDsegments, compareIBDsegmentLists, extractIBDsegments, findDenseRegions, hapFabia, hapFabiaVersion, hapRes, chr1ASW1000G, IBDsegmentList2excel, identifyDuplicates, iterateIntervals, makePipelineFile, matrixPlot, mergeIBDsegmentLists, mergedIBDsegmentList, plotIBDsegment, res, setAnnotation, setStatistics, sim, simu, simulateIBDsegmentsFabia, simulateIBDsegments, split_sparse_matrix, toolsFactorizationClas vcftoFABIA

Examples

```
data(hapRes)
res <- hapRes$res
sPF <- hapRes$sPF
annot <- hapRes$annot
nnL <- length(Z(res)[1,])
labelsA <- cbind(as.character(1:nnL),as.character(1:nnL),
   as.character(1:nnL),as.character(1:nnL))
resIBDsegmentList <-
  extractIBDsegments(res=res,sPF=sPF,annot=annot,
  chrom="1",labelsA=labelsA,ps=0.9,psZ=0.8,inteA=50,
  thresA=6,mintagSNVs=6,off=0,procMinIndivids=0.1,
  thresPrune=1e-3)

summary(resIBDsegmentList)

resIBDsegmentList <- setStatistics(resIBDsegmentList)

summary(resIBDsegmentList)
```

---

sim *Similarity measures for IBD segments*

Description

sim: R implementation of sim.

Similarity measure for IBD segments, tagSNVs, and individuals.

Usage

```
sim(x,y,simv="minD",minInter=2)
```

Arguments

x	first vector.
y	second vector.
simv	similarity measure: "minD" (percentage of smaller explained by larger set), "jaccard" (Jaccard index), "dice" (Dice index), or "maxD"; default "minD".
minInter	minimal size of intersection that is required for a non-zero measure; default 2.

Details

Similarity measure for IBD segments, tagSNVs, and individuals.

Implementation in R.

Value

erg	the similarity measure between 0 and 1, where 0 is not similar and 1 is identical.

Author(s)

Sepp Hochreiter

References

S. Hochreiter et al., 'FABIA: Factor Analysis for Bicluster Acquisition', Bioinformatics 26(12):1520-1527, 2010.

See Also

IBDsegment-class, IBDsegmentList-class, analyzeIBDsegments, compareIBDsegmentLists, extractIBDsegments, findDenseRegions, hapFabia, hapFabiaVersion, hapRes, chr1ASW1000G, IBDsegmentList2excel, identifyDuplicates, iterateIntervals, makePipelineFile, matrixPlot, mergeIBDsegmentLists, mergedIBDsegmentList, plotIBDsegment, res, setAnnotation, setStatistics, sim, simu, simulateIBDsegmentsFabia, simulateIBDsegments, split_sparse_matrix, toolsFactorizationClass, vcftoFABIA

Examples

```
x <- sample(1:15,8)
y <- sample(1:15,8)
sim(x,y,simv="minD",minInter=1)
```

---

simu *Example simulation data for* hapFabia

Description

The data obtained by simulateForFabia in binary format.

Usage

```
simu
```

Format

contains a list simu with variables

1. namesL,
2. haploN,
3. snvs,
4. annot,
5. alleleIimp which contains the information on the implanted IBD segment: sample names, number of chromosomes/individuals, number of tagSNVs, annotation of tagSNVs, the genotype data where 0 = reference an 1 = minor allele.

annot is a list with entries:

1. chromosome,
2. position,
3. snvNames,
4. snvMajor,
5. snvMinor,
6. quality,
7. pass,
8. info,
9. fields,
10. frequency, and
11. changed.

Source

from simulateForFabia

References

S. Hochreiter et al., 'FABIA: Factor Analysis for Bicluster Acquisition', Bioinformatics 26(12):1520-1527, 2010.

See Also

IBDsegment-class, IBDsegmentList-class, analyzeIBDsegments, compareIBDsegmentLists, extractIBDsegments, findDenseRegions, hapFabia, hapFabiaVersion, hapRes, chr1ASW1000G, IBDsegmentList2excel, identifyDuplicates, iterateIntervals, makePipelineFile, matrixPlot, mergeIBDsegmentLists, mergedIBDsegmentList, plotIBDsegment, res, setAnnotation, setStatistics, sim, simu, simulateIBDsegmentsFabia, simulateIBDsegments, split_sparse_matrix, toolsFactorizationClas
vcftoFABIA

simulateIBDsegments *Generates simulated genotyping data with IBD segments*

Description

simulateIBDsegments: R implementation of simulateIBDsegments.

Genotype data with rare variants is simulated. Into these datan IBD segments are implanted. All data sets and information are written to files.

Usage

```
simulateIBDsegments(fileprefix="dataSim",minruns=1,
   maxruns=100,snvs=10000,individualsN=100,avDistSnvs=100,
   avDistMinor=25,noImplanted=1,implanted=10,length=100,
   minors=20,mismatches=0,mismatchImplanted=0.5,overlap=50,
   noOverwrite=FALSE)
```

Arguments

fileprefix	prefix of file names containing data generated in this simulation.
minruns	start index for generating multiple data sets.
maxruns	end index for generating multiple data sets.
snvs	number of SNVs in this simulation.
individualsN	number of individuals in this simulation.
avDistSnvs	average genomic distance in bases between SNVs.
avDistMinor	average distance between minor alleles, thus 1/avDistMinor is the average minor allele frequency (MAF).
noImplanted	number of IBD segments that are implanted.
implanted	number of individuals belonging to specific IBD segment.
length	length of the IBD segments in number of SNVs.
minors	number of tagSNVs for each IBD segment.
mismatches	number of minor allele tagSNV mismatches for individuals belonging to the IBD segment.
mismatchImplanted	percentage of individuals of an IBD segment that have mismatches.
overlap	minimal overlap of the founder interval between individuals belonging to a specific IBD segment (the interval may be broken at the ends).
noOverwrite	noOverwrite=TRUE ensures that an IBD segment is not superimposed by another IBD segment.

Details

Data simulations focuses on rare variants but common variants are possible, too. Linkage disequilibrium and haplotype blocks are not simulated except by implanting IBD segments.

Simulated data is written to files. For BEAGLE the data is written to "...beagle.txt". For PLINK the data is written to "...plink.ped", "...plink.map", and "...plink.fam". For the MCMC method the data is written to "...mcmc.genotype", "...mcmc.posmaf", and "...mcmc.initz". For RELATE the data is written to "...relate.geno", "...relate.pos", and "...relate.chr". For **fabia** the data is written to "...fabia_individuals.txt", "...fabia_annot.txt" "...fabia_mat.txt".

Information on parameters for data simulation is written to "...Parameters.txt" while information on implanted IBD segments is written to "...Impl.txt".

Most information is also written in R binary ".Rda" files.

Implementation in R.

Value

Generates simulated genotyping data with IBD segments

Author(s)

Sepp Hochreiter

References

S. Hochreiter et al., 'FABIA: Factor Analysis for Bicluster Acquisition', Bioinformatics 26(12):1520-1527, 2010.

See Also

IBDsegment-class, IBDsegmentList-class, analyzeIBDsegments, compareIBDsegmentLists, extractIBDsegments, findDenseRegions, hapFabia, hapFabiaVersion, hapRes, chr1ASW1000G, IBDsegmentList2excel, identifyDuplicates, iterateIntervals, makePipelineFile, matrixPlot, mergeIBDsegmentLists, mergedIBDsegmentList, plotIBDsegment, res, setAnnotation, setStatistics, sim, simu, simulateIBDsegmentsFabia, simulateIBDsegments, split_sparse_matrix, toolsFactorizationClas vcftoFABIA

Examples

```
## Not run:
old_dir <- getwd()
setwd(tempdir())

simulateIBDsegments(minruns=1,maxruns=1,snvs=1000,individualsN=10,avDistSnvs=100,avDistMinor=15,noImplante

setwd(old_dir)


## End(Not run)
```

simulateIBDsegmentsFabia
Generates simulated genotyping data with IBD segments for **fabia**

Description

simulateIBDsegmentsFabia: R implementation of simulateIBDsegmentsFabia.

Genotype data is simulated which contains rare variants and implanted IBD segments. Output is written for the bicluster algorithm **fabia**.

Usage

```
simulateIBDsegmentsFabia(fileprefix="dataSim",
   minruns=1,maxruns=1,snvs=1000,individualsN=100,
   avDistSnvs=100,avDistMinor=10,noImplanted=1,
   implanted=10,length=50,minors=30,mismatches=0,
   mismatchImplanted=0.5,overlap=50)
```

Arguments

fileprefix	prefix of file names containing data generated in this simulation.
minruns	start index for generating multiple data sets.
maxruns	end index for generating multiple data sets.
snvs	number of SNVs in this simulation.
individualsN	number of individuals in this simulation.
avDistSnvs	average genomic distance in bases between SNVs.
avDistMinor	average distance between minor alleles, thus 1/avDistMinor is the average minor allele frequency (MAF).
noImplanted	number of IBD segments that are implanted.
implanted	number of individuals into which a specific IBD segment is implanted.
length	length of the IBD segments in number of SNVs.
minors	number of tagSNVs for each IBD segment.
mismatches	number of base mismatches of an implanted IBD segment to the original IBD segment.
mismatchImplanted	percentage of IBD segment occurrence that have mismatches.
overlap	minimal IBD segment overlap between implanted IBD segments (they are broken at the ends).

Details

Data simulations for **fabia** focuses on rare variants but common variants are possible. Linkage disequilibrium and haplotype blocks are not simulated except by implanting IBD segments.

Simulated data is written to "...fabia_individuals.txt", "...fabia_annot.txt" "...fabia_mat.txt".

Implementation in R.

Value

Generates simulated genotyping data with IBD segments for fabia

Author(s)

Sepp Hochreiter

References

S. Hochreiter et al., 'FABIA: Factor Analysis for Bicluster Acquisition', Bioinformatics 26(12):1520-1527, 2010.

See Also

IBDsegment-class, IBDsegmentList-class, analyzeIBDsegments, compareIBDsegmentLists, extractIBDsegments, findDenseRegions, hapFabia, hapFabiaVersion, hapRes, chr1ASW1000G, IBDsegmentList2excel, identifyDuplicates, iterateIntervals, makePipelineFile, matrixPlot, mergeIBDsegmentLists, mergedIBDsegmentList, plotIBDsegment, res, setAnnotation, setStatistics, sim, simu, simulateIBDsegmentsFabia, simulateIBDsegments, split_sparse_matrix, toolsFactorizationClas vcftoFABIA

Examples

```
old_dir <- getwd()
setwd(tempdir())

simulateIBDsegmentsFabia()

setwd(old_dir)
```

---

split_sparse_matrix *Splits genotyping data in sparse matrix format into intervals*

Description

split_sparse_matrix: C implementation with an R wrapper of split_sparse_matrix.

Genotype data of a chromosome is split into intervals where each interval leads to a genotype file in sparse matrix format.

Usage

```
split_sparse_matrix(fileName,sparseMatrixPostfix="_mat.txt",
intervalSize=10000,shiftSize=5000,annotation=TRUE)
```

Arguments

fileName	string giving the genotype data in sparse matrix format without type. Attention: no type!
sparseMatrixPostfix	postfix string for sparse matrix format.
intervalSize	number of SNVs in one interval.
shiftSize	number of SNVs between beginning of adjacent intervals that is the number of SNVs the intervals are shifted.
annotation	boolean variable indicating whether a annotation file is available.

Details

Genotype data in split into intervals of size `intervalSize`, where the distance of the start of adjacent intervals is `shiftSize`. Thus, it is possible to generate overlapping intervals to account for IBD segments that are located at the border of an interval.

Implementation in C. Also a command line program is supplied.

Value

Splits genotyping data in sparse matrix format into intervals

Author(s)

Sepp Hochreiter

References

S. Hochreiter et al., 'FABIA: Factor Analysis for Bicluster Acquisition', Bioinformatics 26(12):1520-1527, 2010.

See Also

IBDsegment-class, IBDsegmentList-class, analyzeIBDsegments, compareIBDsegmentLists, extractIBDsegments, findDenseRegions, hapFabia, hapFabiaVersion, hapRes, chr1ASW1000G, IBDsegmentList2excel, identifyDuplicates, iterateIntervals, makePipelineFile, matrixPlot, mergeIBDsegmentLists, mergedIBDsegmentList, plotIBDsegment, res, setAnnotation, setStatistics, sim, simu, simulateIBDsegmentsFabia, simulateIBDsegments, split_sparse_matrix, toolsFactorizationClas
vcftoFABIA

Examples

```
## Not run:
########################################
## Already run in "iterateIntervals.Rd" ##
########################################

#Work in a temporary directory.

old_dir <- getwd()
setwd(tempdir())
```

```
# Load data and write to vcf file.
data(chr1ASW1000G)
write(chr1ASW1000G,file="chr1ASW1000G.vcf")

#Create the analysis pipeline for IBD segment extraction
makePipelineFile(fileName="chr1ASW1000G",shiftSize=500,intervalSize=1000,haplotypes=TRUE)

source("pipeline.R")

# Following files are produced:
list.files(pattern="chr1")



# Next we load interval 5 and there the first and second IBD segment
posAll <- 5
start <- (posAll-1)*shiftSize
end <- start + intervalSize
pRange <- paste("_",format(start,scientific=FALSE),"_",format(end,scientific=FALSE),sep="")
load(file=paste(fileName,pRange,"_resAnno",".Rda",sep=""))
IBDsegmentList <- resHapFabia$mergedIBDsegmentList
summary(IBDsegmentList)
IBDsegment1 <- IBDsegmentList[[1]]
summary(IBDsegment1)
IBDsegment2 <- IBDsegmentList[[2]]
summary(IBDsegment2)




#Plot the first IBD segment in interval 5
plot(IBDsegment1,filename=paste(fileName,pRange,"_mat",sep=""))


#Plot the second IBD segment in interval 5
plot(IBDsegment2,filename=paste(fileName,pRange,"_mat",sep=""))

setwd(old_dir)


## End(Not run)

## Not run:
###here an example of the the automatically generated pipeline
### with: shiftSize=5000,intervalSize=10000,fileName="filename"

#####define intervals, overlap, filename #######
shiftSize <- 5000
intervalSize <- 10000
fileName="filename" # without type
haplotypes <- TRUE
dosage <- FALSE

#####load library#######
library(hapFabia)

#####convert from .vcf to _mat.txt#######
```

```
vcftoFABIA(fileName=fileName)

#####copy haplotype, genotype, or dosage matrix to matrix#######
if (haplotypes) {
    file.copy(paste(fileName,"_matH.txt",sep=""), paste(fileName,"_mat.txt",sep=""))
} else {
    if (dosage) {
        file.copy(paste(fileName,"_matD.txt",sep=""), paste(fileName,"_mat.txt",sep=""))
    } else {
        file.copy(paste(fileName,"_matG.txt",sep=""), paste(fileName,"_mat.txt",sep=""))
    }
}

#####split/ generate intervals#######
split_sparse_matrix(fileName=fileName,intervalSize=intervalSize,
shiftSize=shiftSize,annotation=TRUE)

#####compute how many intervals we have#######
ina <- as.numeric(readLines(paste(fileName,"_mat.txt",sep=""),n=2))
noSNVs <- ina[2]
over <- intervalSize%/%shiftSize
N1 <- noSNVs%/%shiftSize
endRunA <- (N1-over+2)

#####analyze each interval#######
#####may be done by parallel runs#######
iterateIntervals(startRun=1,endRun=endRunA,shift=shiftSize,
intervalSize=intervalSize,fileName=fileName,individuals=0,
upperBP=0.05,p=10,iter=40,alpha=0.03,cyc=50,IBDsegmentLength=50,
Lt = 0.1,Zt = 0.2,thresCount=1e-5,mintagSNVsFactor=3/4,
pMAF=0.03,haplotypes=haplotypes,cut=0.8,procMinIndivids=0.1,thresPrune=1e-3,
simv="minD",minTagSNVs=6,minIndivid=2,avSNVsDist=100,SNVclusterLength=100)

#####identify duplicates#######
identifyDuplicates(fileName=fileName,startRun=1,endRun=endRunA,
shift=shiftSize,intervalSize=intervalSize)

#####analyze results; parallel#######
anaRes <- analyzeIBDsegments(fileName=fileName,startRun=1,endRun=endRunA,
shift=shiftSize,intervalSize=intervalSize)
print("Number IBD segments:")
print(anaRes$noIBDsegments)
print("Statistics on IBD segment length in SNVs (all SNVs in the IBD segment):")
print(anaRes$avIBDsegmentLengthSNVS)
print("Statistics on IBD segment length in bp:")
print(anaRes$avIBDsegmentLengthS)
print("Statistics on number of individuals belonging to IBD segments:")
print(anaRes$avnoIndividS)
print("Statistics on number of tagSNVs of IBD segments:")
print(anaRes$avnoTagSNVsS)
print("Statistics on MAF of tagSNVs of IBD segments:")
print(anaRes$avnoFreqS)
print("Statistics on MAF within the group of tagSNVs of IBD segments:")
print(anaRes$avnoGroupFreqS)
print("Statistics on number of changes between major and minor allele frequency:")
print(anaRes$avnotagSNVChangeS)
print("Statistics on number of tagSNVs per individual of an IBD segment:")
```

```
print(anaRes$avnotagSNVsPerIndividualS)
print("Statistics on number of individuals that have the minor allele of tagSNVs:")
print(anaRes$avnoindividualPerTagSNVS)

#####load result for interval 50#######
posAll <- 50 # (50-1)*5000 = 245000: interval 245000 to 255000
start <- (posAll-1)*shiftSize
end <- start + intervalSize
pRange <- paste("_",format(start,scientific=FALSE),"_",
format(end,scientific=FALSE),sep="")
load(file=paste(fileName,pRange,"_resAnno",".Rda",sep=""))
IBDsegmentList <- resHapFabia$mergedIBDsegmentList # $

summary(IBDsegmentList)
#####plot IBD segments in interval 50#######
plot(IBDsegmentList,filename=paste(fileName,pRange,"_mat",sep=""))
   ##attention: filename without type ".txt"

#####plot the first IBD segment in interval 50#######

IBDsegment <- IBDsegmentList[[1]]
plot(IBDsegment,filename=paste(fileName,pRange,"_mat",sep=""))
   ##attention: filename without type ".txt"


## End(Not run)
```

`toolsFactorizationClass`

Tools to analyze results of **fabia**

Description

These tools allow to analyze results of the package **fabia**. They can be used to identify IBD segment regions and for adjusting the parameters of `extractIBDsegments` and `hapFabia` such as `ps` (top L values for extraction), `psZ` (top Z values for extraction), `inteA` (length of histogram bins).

`plotL` plots the loadings of a **fabia** result that are above a threshold either as points, histogram or by a smooth scatter plot.

`topLZ` returns largest L or Z values of a **fabia** result, where thresholds are given either by a quantile or by a value.

`histL` supplies a histogram of the loadings obtained by **fabia**.

Usage

```
plotL(res,n=1,p=NULL,w=NULL,type="points",
   intervv=500,off=0,t="p",cex=1)

histL(res,n=1,p=NULL,w=NULL,intervv=500,off=0)

topLZ(res,n=1,LZ="L",indices=TRUE,p=NULL,w=NULL)
```

Arguments

res	**fabia** result; instance of the class Factorization.
n	the number of the bicluster to consider.
p	the quantile threshold above which values are returned (p or w must be given).
w	the value threshold above which values are returned (p or w must be given).
type	the type of the plot: type=c("points","histogram","smoothScatter").
intervv	length of the interval bins for histograms.
off	offset of the interval bins from zero for histograms.
t	points type for the plot.
cex	size of the points for the plot.
LZ	"L" for loadings L or "Z" for factors Z.
indices	if TRUE (default) indices are given and otherwise values.

Details

plotL plots the loadings of a **fabia** result that are above a threshold either as points, histogram or by a smooth scatter plot. Thresholds can be given by a quantile or by a value.

topLZ returns largest L or Z indices/values of a **fabia** result. Thresholds are given by quantile or by a value.

histL computes histogram of the loadings obtained by **fabia**.

Implementation in R.

Value

plotL: nothing.

topLZ: vector of indices or values depending on the logical parameter indices.

histL: object of class histogram.

Author(s)

Sepp Hochreiter

References

S. Hochreiter et al., 'FABIA: Factor Analysis for Bicluster Acquisition', Bioinformatics 26(12):1520-1527, 2010.

See Also

IBDsegment-class, IBDsegmentList-class, analyzeIBDsegments, compareIBDsegmentLists, extractIBDsegments, findDenseRegions, hapFabia, hapFabiaVersion, hapRes, chr1ASW1000G, IBDsegmentList2excel, identifyDuplicates, iterateIntervals, makePipelineFile, matrixPlot, mergeIBDsegmentLists, mergedIBDsegmentList, plotIBDsegment, res, setAnnotation, setStatistics, sim, simu, simulateIBDsegmentsFabia, simulateIBDsegments, split_sparse_matrix, toolsFactorizationClass, vcftoFABIA

Examples

```
data(res)

plotL(res,n=1,p=0.95,w=NULL,type="histogram",
   intervv=50,off=0,t="p",cex=1)
plotL(res,n=1,p=0.95,w=NULL,type="points",
   intervv=50,off=0,t="p",cex=1)
plotL(res,n=1,p=NULL,w=0.5,type="points",
   intervv=50,off=0,t="p",cex=1)
plotL(res,n=1,p=0.95,w=NULL,type="smooth",
   intervv=50,off=0,t="p",cex=1)
plotL(res,n=1,p=NULL,w=0.5,type="smooth",
   intervv=50,off=0,t="p",cex=1)

topLZ(res,n=1,LZ="L",indices=TRUE,p=0.95,w=NULL)
topLZ(res,n=1,LZ="L",indices=TRUE,p=NULL,w=0.95)

topLZ(res,n=1,LZ="Z",indices=TRUE,p=0.95,w=NULL)
topLZ(res,n=1,LZ="Z",indices=TRUE,p=NULL,w=0.4)

topLZ(res,n=1,LZ="L",indices=FALSE,p=0.95,w=NULL)
topLZ(res,n=1,LZ="L",indices=FALSE,p=NULL,w=0.95)

topLZ(res,n=1,LZ="Z",indices=FALSE,p=0.95,w=NULL)
topLZ(res,n=1,LZ="Z",indices=FALSE,p=NULL,w=0.4)

h1 <- histL(res,n=1,p=0.9,w=NULL,intervv=50,off=0)
print(h1$counts)
h1 <- histL(res,n=1,p=NULL,w=0.5,intervv=50,off=0)
print(h1$counts)
```

`vcftoFABIA` *Converting genotyping data from* `vcf` *to sparse matrix format*

Description

`vcftoFABIA`: C implementation with an R wrapper of `vcftoFABIA`.

Converts files giving the genotype in vcf format to sparse matrix formats as used by FABIA. Phased and unphased genotypes as well as dosages or likelihoods are written to files in sparse matrix formats. Haplotype data is stored in `fileName_matH.txt`, genotype data in `fileName_matG.txt`, and dosage data in `fileName_matD.txt`.

SNV annotations are stored in `fileName_annot.txt` and the names of the individuals in `fileName_individuals.txt`. These files serve as input for `split_sparse_matrix`.

Usage

```
vcftoFABIA(fileName,prefixPath="",noSnvs=NULL,outputFile=NULL)
```

Arguments

fileName	string giving the file name without the file type '.vcf'. Attention: remove file type!
prefixPath	path to the file.
noSnvs	optional: the number of SNVs; needed for memory allocation; if not known it is determined by first reading all lines of the file.
outputFile	optional: prefix for the output files, if not given then the input file prefix is used.

Details

The function vcftoFABIA converts fileName.vcf to sparse matrix format giving (if not outputFile is given then Outfilename=fileName):

1. Outfilename_matH.txt (haplotype data),
2. Outfilename_matG.txt (genotype data),
3. Outfilename_matD.txt (dosage data),

together with the SNV annotation file and individual's label file:

1. Outfilename_annot.txt and
2. Outfilename_individuals.txt.

In a subsequent step these files can be split into intervals by split_sparse_matrix and thereafter IBD segments extracted by iterateIntervals.

Implementation in C. Also command line programs are supplied.

Value

Converting genotyping data from vcf to sparse matrix format

Author(s)

Sepp Hochreiter

References

S. Hochreiter et al., 'FABIA: Factor Analysis for Bicluster Acquisition', Bioinformatics 26(12):1520-1527, 2010.

See Also

IBDsegment-class, IBDsegmentList-class, analyzeIBDsegments, compareIBDsegmentLists, extractIBDsegments, findDenseRegions, hapFabia, hapFabiaVersion, hapRes, chr1ASW1000G, IBDsegmentList2excel, identifyDuplicates, iterateIntervals, makePipelineFile, matrixPlot, mergeIBDsegmentLists, mergedIBDsegmentList, plotIBDsegment, res, setAnnotation, setStatistics, sim, simu, simulateIBDsegmentsFabia, simulateIBDsegments, split_sparse_matrix, toolsFactorizationClas
vcftoFABIA

Examples

```
## Not run:
########################################
## Already run in "iterateIntervals.Rd" ##
########################################

#Work in a temporary directory.

old_dir <- getwd()
setwd(tempdir())


# Load data and write to vcf file.
data(chr1ASW1000G)
write(chr1ASW1000G,file="chr1ASW1000G.vcf")

#Create the analysis pipeline for IBD segment extraction
makePipelineFile(fileName="chr1ASW1000G",shiftSize=500,intervalSize=1000,haplotypes=TRUE)

source("pipeline.R")

# Following files are produced:
list.files(pattern="chr1")



# Next we load interval 5 and there the first and second IBD segment
posAll <- 5
start <- (posAll-1)*shiftSize
end <- start + intervalSize
pRange <- paste("_",format(start,scientific=FALSE),"_",format(end,scientific=FALSE),sep="")
load(file=paste(fileName,pRange,"_resAnno",".Rda",sep=""))
IBDsegmentList <- resHapFabia$mergedIBDsegmentList
summary(IBDsegmentList)
IBDsegment1 <- IBDsegmentList[[1]]
summary(IBDsegment1)
IBDsegment2 <- IBDsegmentList[[2]]
summary(IBDsegment2)




#Plot the first IBD segment in interval 5
plot(IBDsegment1,filename=paste(fileName,pRange,"_mat",sep=""))


#Plot the second IBD segment in interval 5
plot(IBDsegment2,filename=paste(fileName,pRange,"_mat",sep=""))

setwd(old_dir)


## End(Not run)

## Not run:
```

```
###here an example of the the automatically generated pipeline
### with: shiftSize=5000,intervalSize=10000,fileName="filename"

#####define intervals, overlap, filename #######
shiftSize <- 5000
intervalSize <- 10000
fileName="filename" # without type
haplotypes <- TRUE
dosage <- FALSE

#####load library#######
library(hapFabia)

#####convert from .vcf to _mat.txt#######
vcftoFABIA(fileName=fileName)

#####copy haplotype, genotype, or dosage matrix to matrix#######
if (haplotypes) {
    file.copy(paste(fileName,"_matH.txt",sep=""), paste(fileName,"_mat.txt",sep=""))
} else {
    if (dosage) {
        file.copy(paste(fileName,"_matD.txt",sep=""), paste(fileName,"_mat.txt",sep=""))
    } else {
        file.copy(paste(fileName,"_matG.txt",sep=""), paste(fileName,"_mat.txt",sep=""))
    }
}

#####split/ generate intervals#######
split_sparse_matrix(fileName=fileName,intervalSize=intervalSize,
shiftSize=shiftSize,annotation=TRUE)

#####compute how many intervals we have#######
ina <- as.numeric(readLines(paste(fileName,"_mat.txt",sep=""),n=2))
noSNVs <- ina[2]
over <- intervalSize%/%shiftSize
N1 <- noSNVs%/%shiftSize
endRunA <- (N1-over+2)

#####analyze each interval#######
#####may be done by parallel runs#######
iterateIntervals(startRun=1,endRun=endRunA,shift=shiftSize,
intervalSize=intervalSize,fileName=fileName,individuals=0,
upperBP=0.05,p=10,iter=40,alpha=0.03,cyc=50,IBDsegmentLength=50,
Lt = 0.1,Zt = 0.2,thresCount=1e-5,mintagSNVsFactor=3/4,
pMAF=0.03,haplotypes=haplotypes,cut=0.8,procMinIndivids=0.1,thresPrune=1e-3,
simv="minD",minTagSNVs=6,minIndivid=2,avSNVsDist=100,SNVclusterLength=100)

#####identify duplicates#######
identifyDuplicates(fileName=fileName,startRun=1,endRun=endRunA,
shift=shiftSize,intervalSize=intervalSize)

#####analyze results; parallel#######
anaRes <- analyzeIBDsegments(fileName=fileName,startRun=1,endRun=endRunA,
shift=shiftSize,intervalSize=intervalSize)
print("Number IBD segments:")
print(anaRes$noIBDsegments)
print("Statistics on IBD segment length in SNVs (all SNVs in the IBD segment):")
```

```
print(anaRes$avIBDsegmentLengthSNVS)
print("Statistics on IBD segment length in bp:")
print(anaRes$avIBDsegmentLengthS)
print("Statistics on number of individuals belonging to IBD segments:")
print(anaRes$avnoIndividS)
print("Statistics on number of tagSNVs of IBD segments:")
print(anaRes$avnoTagSNVsS)
print("Statistics on MAF of tagSNVs of IBD segments:")
print(anaRes$avnoFreqS)
print("Statistics on MAF within the group of tagSNVs of IBD segments:")
print(anaRes$avnoGroupFreqS)
print("Statistics on number of changes between major and minor allele frequency:")
print(anaRes$avnotagSNVChangeS)
print("Statistics on number of tagSNVs per individual of an IBD segment:")
print(anaRes$avnotagSNVsPerIndividualS)
print("Statistics on number of individuals that have the minor allele of tagSNVs:")
print(anaRes$avnoindividualPerTagSNVS)

#####load result for interval 50#######
posAll <- 50 # (50-1)*5000 = 245000: interval 245000 to 255000
start <- (posAll-1)*shiftSize
end <- start + intervalSize
pRange <- paste("_",format(start,scientific=FALSE),"_",
format(end,scientific=FALSE),sep="")
load(file=paste(fileName,pRange,"_resAnno",".Rda",sep=""))
IBDsegmentList <- resHapFabia$mergedIBDsegmentList # $

summary(IBDsegmentList)
#####plot IBD segments in interval 50#######
plot(IBDsegmentList,filename=paste(fileName,pRange,"_mat",sep=""))
   ##attention: filename without type ".txt"

#####plot the first IBD segment in interval 50#######

IBDsegment <- IBDsegmentList[[1]]
plot(IBDsegment,filename=paste(fileName,pRange,"_mat",sep=""))
   ##attention: filename without type ".txt"


## End(Not run)
```

Index

∗ **IBD segments,short IBD segments, IBD**
  hapFabia, 13
  hapFabiaVersion, 18
∗ **classes**
  IBDsegment-class, 20
  IBDsegmentList-class, 25
∗ **datagen**
  simulateIBDsegments, 54
  simulateIBDsegmentsFabia, 56
∗ **datasets**
  chr1ASW1000G, 7
  hapRes, 19
  mergedIBDsegmentList, 42
  res, 47
  simu, 52
∗ **file,connection**
  split_sparse_matrix, 57
  vcftoFABIA, 63
∗ **file**
  IBDsegmentList2excel, 28
∗ **generation sequencing,genotype,single nucleotide polymorphism,single**
  hapFabia, 13
  hapFabiaVersion, 18
∗ **genetics,haplotype,identity by descent,bicluster,next generation sequencing,genotype,single nucleotide polymorphism,single nucleotide variation,rare variants,rare SNPs, rare SNVs,rare IBD segments,short IBD segments**
  analyzeIBDsegments, 2
  compareIBDsegmentLists, 8
  findDenseRegions, 11
  IBDsegment-class, 20
  IBDsegmentList-class, 25
  IBDsegmentList2excel, 28
  identifyDuplicates, 30
  iterateIntervals, 34
  makePipelineFile, 40
  matrixPlot, 41
  mergeIBDsegmentLists, 43
  plotIBDsegment, 44
  setAnnotation, 48
  setStatistics, 50
  sim, 51
  simulateIBDsegments, 54
  simulateIBDsegmentsFabia, 56
  toolsFactorizationClass, 61
∗ **genetics,haplotype,identity by descent,bicluster,next generation sequencing,genotype,single nucleotide polymorphism,single nucleotide variation,rare variants,rare SNVs,rare IBD segments,short IBD segments**
  extractIBDsegments, 9
∗ **genetics,haplotype,identity by descent,bicluster,next generation sequencing,genotype,single nucleotide polymorphism,single nucleotide variation**
  split_sparse_matrix, 57
  vcftoFABIA, 63
∗ **genetics,haplotype,identity by descent,bicluster,next**
  hapFabia, 13
  hapFabiaVersion, 18
∗ **hplot**
  IBDsegment-class, 20
  IBDsegmentList-class, 25
  matrixPlot, 41
  plotIBDsegment, 44
∗ **methods**
  IBDsegment-class, 20
  IBDsegmentList-class, 25
∗ **models,multivariate,cluster**
  analyzeIBDsegments, 2
  compareIBDsegmentLists, 8
  extractIBDsegments, 9
  findDenseRegions, 11
  hapFabia, 13
  hapFabiaVersion, 18
  identifyDuplicates, 30
  iterateIntervals, 34

makePipelineFile, 40
mergeIBDsegmentLists, 43
setAnnotation, 48
setStatistics, 50
sim, 51
toolsFactorizationClass, 61
∗ **nucleotide variation,rare variants,rare SNPs, rare SNVs,rare**
hapFabiaVersion, 18
∗ **nucleotide variation,rare variants,rare SNVs, rare**
hapFabia, 13
[*(*IBDsegmentList-class*)*, 25
[,IBDsegmentList,numeric,missing-method *(*IBDsegmentList-class*)*, 25
[,IBDsegmentList-method *(*IBDsegmentList-class*)*, 25
[<- *(*IBDsegmentList-class*)*, 25
[<-,IBDsegmentList,numeric,missing,IBDsegmentList-method *(*IBDsegmentList-class*)*, 25
[<-,IBDsegmentList-method *(*IBDsegmentList-class*)*, 25
[[*(*IBDsegmentList-class*)*, 25
[[,IBDsegmentList,numeric,missing-method *(*IBDsegmentList-class*)*, 25
[[,IBDsegmentList-method *(*IBDsegmentList-class*)*, 25
[[<- *(*IBDsegmentList-class*)*, 25
[[<-,IBDsegmentList,numeric,missing,IBDsegment-method *(*IBDsegmentList-class*)*, 25
[[<-,IBDsegmentList-method *(*IBDsegmentList-class*)*, 25

analyzeIBDsegments, 2, *4*, *7*, *9*, *11, 12*, *16*, *18*, *20*, *23*, *27*, *30, 31*, *37*, *40*, *42–44*, *46*, *48, 49*, *51–53*, *55*, *57, 58*, *62*, *64*

bicluster_id *(*IBDsegment-class*)*, 20
bicluster_id,IBDsegment-method *(*IBDsegment-class*)*, 20
bicluster_id<- *(*IBDsegment-class*)*, 20
bicluster_id<-,IBDsegment,numeric-method *(*IBDsegment-class*)*, 20

chr1ASW1000G, *4*, *7*, 7, *9*, *11, 12*, *16*, *18*, *20*, *23*, *27*, *30, 31*, *37*, *40*, *42–44*, *46*, *48, 49*, *51–53*, *55*, *57, 58*, *62*, *64*
chromosome *(*IBDsegment-class*)*, 20
chromosome,IBDsegment-method *(*IBDsegment-class*)*, 20
chromosome<- *(*IBDsegment-class*)*, 20
chromosome<-,IBDsegment,character-method *(*IBDsegment-class*)*, 20

compareIBDsegmentLists, *4*, *7*, 8, *9*, *11, 12*, *16*, *18*, *20*, *23*, *27*, *30, 31*, *37*, *40*, *42–44*, *46*, *48, 49*, *51–53*, *55*, *57, 58*, *62*, *64*
compareIBDsegmentLists,IBDsegmentList,ANY,character,ANY *(*compareIBDsegmentLists*)*, 8
compareIBDsegmentLists,IBDsegmentList-method *(*compareIBDsegmentLists*)*, 8
coreClusterIndividuals *(*IBDsegment-class*)*, 20
coreClusterIndividuals,IBDsegment-method *(*IBDsegment-class*)*, 20
coreClusterIndividuals<- *(*IBDsegment-class*)*, 20
coreClusterIndividuals<-,IBDsegment,vector-method *(*IBDsegment-class*)*, 20

extractIBDsegments, *4*, *7*, *9*, 9, *11, 12*, *16*, *18*, *20*, *23*, *27*, *30, 31*, *37*, *40*, *42–44*, *46*, *48, 49*, *51–53*, *55*, *57, 58*, *62*, *64*
extractIBDsegments,Factorization,list,data.frame,charac *(*extractIBDsegments*)*, 9
extractIBDsegments,Factorization-method *(*extractIBDsegments*)*, 9

findDenseRegions, *4*, *7*, *9*, *11*, 11, *12*, *16*, *18*, *20*, *23*, *27*, *30, 31*, *37*, *40*, *42–44*, *46*, *48, 49*, *51–53*, *55*, *57, 58*, *62*, *64*
hapFabia, *4*, *7*, *9*, *11, 12*, 13, *16*, *18*, *20*, *23*, *27*, *30, 31*, *37*, *40*, *42–44*, *46*, *48, 49*, *51–53*, *55*, *57, 58*, *62*, *64*
hapFabiaVersion, *4*, *7*, *9*, *11, 12*, *16*, *18*, 18, *20*, *23*, *27*, *30, 31*, *37*, *40*, *42–44*, *46*, *48, 49*, *51–53*, *55*, *57, 58*, *62*, *64*
hapRes, *4*, *7*, *9*, *11, 12*, *16*, *18*, 19, *20*, *23*, *27*, *30, 31*, *37*, *40*, *42–44*, *46*, *48, 49*, *51–53*, *55*, *57, 58*, *62*, *64*
histL *(*toolsFactorizationClass*)*, 61
histL,Factorization,numeric,ANY,ANY,numeric,numeric-met *(*toolsFactorizationClass*)*, 61
histL,Factorization-method *(*toolsFactorizationClass*)*, 61

IBDsegment *(*IBDsegment-class*)*, 20
IBDsegment,ANY-method *(*IBDsegment-class*)*, 20
IBDsegment,IBDsegment-method *(*IBDsegment-class*)*, 20
IBDsegment,numeric,numeric,character,numeric,numeric,nu *(*IBDsegment-class*)*, 20
IBDsegment-class, 20
IBDsegment-method *(*IBDsegment-class*)*, 20

IBDsegmentLength *(IBDsegment-class)*, 20
IBDsegmentLength,IBDsegment-method *(IBDsegment-class)*, 20
IBDsegmentLength<- *(IBDsegment-class)*, 20
IBDsegmentLength<-,IBDsegment,numeric-method *(IBDsegment-class)*, 20
IBDsegmentList *(IBDsegmentList-class)*, 25
IBDsegmentList,ANY-method *(IBDsegmentList-class)*, 25
IBDsegmentList,list,numeric,list-method *(IBDsegmentList-class)*, 25
IBDsegmentList-class, 25
IBDsegmentList-method *(IBDsegmentList-class)*, 25
IBDsegmentList2excel, *4*, *7*, *9*, *11*, *12*, *16*, *18*, *20*, *23*, *27*, 28, *30*, *31*, *37*, *40*, *42–44*, *46*, *48*, *49*, *51–53*, *55*, *57*, *58*, *62*, *64*
IBDsegmentList2excel,IBDsegmentList,character-method *(IBDsegmentList2excel)*, 28
IBDsegmentList2excel,IBDsegmentList-method *(IBDsegmentList2excel)*, 28
IBDsegmentPos *(IBDsegment-class)*, 20
IBDsegmentPos,IBDsegment-method *(IBDsegment-class)*, 20
IBDsegmentPos<- *(IBDsegment-class)*, 20
IBDsegmentPos<-,IBDsegment,numeric-method *(IBDsegment-class)*, 20
IBDsegments *(IBDsegmentList-class)*, 25
IBDsegments,IBDsegmentList-method *(IBDsegmentList-class)*, 25
IBDsegments<- *(IBDsegmentList-class)*, 25
IBDsegments<-,IBDsegmentList,list-method *(IBDsegmentList-class)*, 25
ID *(IBDsegment-class)*, 20
ID,IBDsegment-method *(IBDsegment-class)*, 20
ID<- *(IBDsegment-class)*, 20
ID<-,IBDsegment,numeric-method *(IBDsegment-class)*, 20
identifyDuplicates, *4*, *7*, *9*, *11*, *12*, *16*, *18*, *20*, *23*, *27*, *30*, 30, *31*, *37*, *40*, *42–44*, *46*, *48*, *49*, *51–53*, *55*, *57*, *58*, *62*, *64*
idIndividuals *(IBDsegment-class)*, 20
idIndividuals,IBDsegment-method *(IBDsegment-class)*, 20
idIndividuals<- *(IBDsegment-class)*, 20
idIndividuals<-,IBDsegment,vector-method *(IBDsegment-class)*, 20
individualPerTagSNV *(IBDsegment-class)*, 20
individualPerTagSNV,IBDsegment-method *(IBDsegment-class)*, 20
individualPerTagSNV<- *(IBDsegment-class)*, 20
individualPerTagSNV<-,IBDsegment,vector-method *(IBDsegment-class)*, 20
individuals *(IBDsegment-class)*, 20
individuals,IBDsegment-method *(IBDsegment-class)*, 20
individuals<- *(IBDsegment-class)*, 20
individuals<-,IBDsegment,vector-method *(IBDsegment-class)*, 20
iterateIntervals, *4*, *7*, *9*, *11*, *12*, *16*, *18*, *20*, *23*, *27*, *30*, *31*, 34, *37*, *40*, *42–44*, *46*, *48*, *49*, *51–53*, *55*, *57*, *58*, *62*, *64*

labelIndividuals *(IBDsegment-class)*, 20
labelIndividuals,IBDsegment-method *(IBDsegment-class)*, 20
labelIndividuals<- *(IBDsegment-class)*, 20
labelIndividuals<-,IBDsegment,vector-method *(IBDsegment-class)*, 20
lengthList *(IBDsegmentList-class)*, 25
lengthList,IBDsegmentList-method *(IBDsegmentList-class)*, 25
lengthList<- *(IBDsegmentList-class)*, 25
lengthList<-,IBDsegmentList,numeric-method *(IBDsegmentList-class)*, 25

makePipelineFile, *4*, *7*, *9*, *11*, *12*, *16*, *18*, *20*, *23*, *27*, *30*, *31*, *37*, *40*, 40, *42–44*, *46*, *48*, *49*, *51–53*, *55*, *57*, *58*, *62*, *64*
matrixPlot, *4*, *7*, *9*, *11*, *12*, *16*, *18*, *20*, *23*, *27*, *30*, *31*, *37*, *40*, 41, *42–44*, *46*, *48*, *49*, *51–53*, *55*, *57*, *58*, *62*, *64*
mergedIBDsegmentList, *4*, *7*, *9*, *11*, *12*, *16*, *18*, *20*, *23*, *27*, *30*, *31*, *37*, *40*, *42*, 42, *43*, *44*, *46*, *48*, *49*, *51–53*, *55*, *57*, *58*, *62*, *64*
mergeIBDsegmentLists, *4*, *7*, *9*, *11*, *12*, *16*, *18*, *20*, *23*, *27*, *30*, *31*, *37*, *40*, *42*, *43*, 43, *44*, *46*, *48*, *49*, *51–53*, *55*, *57*, *58*, *62*, *64*
mergeIBDsegmentLists,IBDsegmentList,ANY,vector-method *(mergeIBDsegmentLists)*, 43
mergeIBDsegmentLists,IBDsegmentList-method *(mergeIBDsegmentLists)*, 43

numberIndividuals *(IBDsegment-class)*, 20
numberIndividuals,IBDsegment-method *(IBDsegment-class)*, 20

numberIndividuals<- (IBDsegment-class), 20
numberIndividuals<-,IBDsegment,numeric-method (IBDsegment-class), 20
numbertagSNVs (IBDsegment-class), 20
numbertagSNVs,IBDsegment-method (IBDsegment-class), 20
numbertagSNVs<- (IBDsegment-class), 20
numbertagSNVs<-,IBDsegment,numeric-method (IBDsegment-class), 20

platformIndividuals (IBDsegment-class), 20
platformIndividuals,IBDsegment-method (IBDsegment-class), 20
platformIndividuals<- (IBDsegment-class), 20
platformIndividuals<-,IBDsegment,vector-method (IBDsegment-class), 20
plot,IBDsegment,missing-method (IBDsegment-class), 20
plot,IBDsegment-method (IBDsegment-class), 20
plot,IBDsegmentList,missing-method (IBDsegmentList-class), 25
plot,IBDsegmentList-method (IBDsegmentList-class), 25
plotIBDsegment, *4*, *7*, *9*, *11, 12*, *16*, *18*, *20*, *23*, *27*, *30, 31*, *37*, *40*, *42–44*, 44, *46*, *48, 49*, *51–53*, *55*, *57, 58*, *62*, *64*
plotL (toolsFactorizationClass), 61
plotL,Factorization,numeric,ANY,ANY,characters,numeric-method (toolsFactorizationClass), 61
plotL,Factorization-method (toolsFactorizationClass), 61
plotLarger (IBDsegment-class), 20
plotLarger,IBDsegment,character,numeric-method (IBDsegment-class), 20
plotLarger,IBDsegment,missing-method (IBDsegment-class), 20
plotLarger,IBDsegment-method (IBDsegment-class), 20
populationIndividuals (IBDsegment-class), 20
populationIndividuals,IBDsegment-method (IBDsegment-class), 20
populationIndividuals<- (IBDsegment-class), 20
populationIndividuals<-,IBDsegment,vector-method (IBDsegment-class), 20

res, *4*, *7*, *9*, *11, 12*, *16*, *18*, *20*, *23*, *27*, *30, 31*, *37*, *40*, *42–44*, *46*, 47, *48, 49*, *51–53*, *55*, *57, 58*, *62*, *64*

setAnnotation, *4*, *7*, *9*, *11, 12*, *16*, *18*, *20*, *23*, *27*, *30, 31*, *37*, *40*, *42–44*, *46*, *48*, 48, *49*, *51–53*, *55*, *57, 58*, *62*, *64*
setAnnotation,IBDsegmentList,character-method (setAnnotation), 48
setAnnotation,IBDsegmentList-method (setAnnotation), 48
setStatistics, *4*, *7*, *9*, *11, 12*, *16*, *18*, *20*, *23*, *27*, *30, 31*, *37*, *40*, *42–44*, *46*, *48, 49*, 50, *51–53*, *55*, *57, 58*, *62*, *64*
setStatistics,IBDsegmentList-method (setStatistics), 50
sim, *4*, *7*, *9*, *11, 12*, *16*, *18*, *20*, *23*, *27*, *30, 31*, *37*, *40*, *42–44*, *46*, *48, 49*, *51*, 51, *52, 53*, *55*, *57, 58*, *62*, *64*
simu, *4*, *7*, *9*, *11, 12*, *16*, *18*, *20*, *23*, *27*, *30, 31*, *37*, *40*, *42–44*, *46*, *48, 49*, *51, 52*, 52, *53*, *55*, *57, 58*, *62*, *64*
simulateIBDsegments, *4*, *7*, *9*, *11, 12*, *16*, *18*, *20*, *23*, *27*, *30, 31*, *37*, *40*, *42–44*, *46*, *48, 49*, *51–53*, 54, *55*, *57, 58*, *62*, *64*
simulateIBDsegmentsFabia, *4*, *7*, *9*, *11, 12*, *16*, *18*, *20*, *23*, *27*, *30, 31*, *37*, *40*, *42–44*, *46*, *48, 49*, *51–53*, *55*, 56, *57, 58*, *62*, *64*
split_sparse_matrix, *4*, *7*, *9*, *11, 12*, *16*, *18*, *20*, *23*, *27*, *30, 31*, *37*, *40*, *42–44*, *46*, *48, 49*, *51–53*, *55*, *57*, 57, *58*, *62*, *64*
statistics (IBDsegmentList-class), 25
statistics,IBDsegmentList,character-method (IBDsegmentList-class), 25
statistics<- (IBDsegmentList-class), 25
statistics<-,IBDsegmentList,list-method (IBDsegmentList-class), 25
summary,IBDsegment-method (IBDsegment-class), 20
summary,IBDsegmentList-method (IBDsegmentList-class), 25

tagSNVAlleles (IBDsegment-class), 20
tagSNVAlleles,IBDsegment-method (IBDsegment-class), 20
tagSNVAlleles<- (IBDsegment-class), 20
tagSNVAlleles<-,IBDsegment,vector-method (IBDsegment-class), 20
tagSNVAnno (IBDsegment-class), 20
tagSNVAnno,IBDsegment-method (IBDsegment-class), 20
tagSNVAnno<- (IBDsegment-class), 20
tagSNVAnno<-,IBDsegment,vector-method (IBDsegment-class), 20

tagSNVChange *(IBDsegment-class)*, 20
tagSNVChange,IBDsegment-method *(IBDsegment-class)*, 20
tagSNVChange<- *(IBDsegment-class)*, 20
tagSNVChange<-,IBDsegment,vector-method *(IBDsegment-class)*, 20
tagSNVFreq *(IBDsegment-class)*, 20
tagSNVFreq,IBDsegment-method *(IBDsegment-class)*, 20
tagSNVFreq<- *(IBDsegment-class)*, 20
tagSNVFreq<-,IBDsegment,vector-method *(IBDsegment-class)*, 20
tagSNVGroupFreq *(IBDsegment-class)*, 20
tagSNVGroupFreq,IBDsegment-method *(IBDsegment-class)*, 20
tagSNVGroupFreq<- *(IBDsegment-class)*, 20
tagSNVGroupFreq<-,IBDsegment,vector-method *(IBDsegment-class)*, 20
tagSNVNames *(IBDsegment-class)*, 20
tagSNVNames,IBDsegment-method *(IBDsegment-class)*, 20
tagSNVNames<- *(IBDsegment-class)*, 20
tagSNVNames<-,IBDsegment,vector-method *(IBDsegment-class)*, 20
tagSNVPositions *(IBDsegment-class)*, 20
tagSNVPositions,IBDsegment-method *(IBDsegment-class)*, 20
tagSNVPositions<- *(IBDsegment-class)*, 20
tagSNVPositions<-,IBDsegment,vector-method *(IBDsegment-class)*, 20
tagSNVs *(IBDsegment-class)*, 20
tagSNVs,IBDsegment-method *(IBDsegment-class)*, 20
tagSNVs<- *(IBDsegment-class)*, 20
tagSNVs<-,IBDsegment,vector-method *(IBDsegment-class)*, 20
tagSNVsPerIndividual *(IBDsegment-class)*, 20
tagSNVsPerIndividual,IBDsegment-method *(IBDsegment-class)*, 20
tagSNVsPerIndividual<- *(IBDsegment-class)*, 20
tagSNVsPerIndividual<-,IBDsegment,vector-method *(IBDsegment-class)*, 20
toolsFactorizationClass, *4*, *7*, *9*, *11*, *12*, *16*, *18*, *20*, *23*, *27*, *30*, *31*, *37*, *40*, *42–44*, *46*, *48*, *49*, *51–53*, *55*, *57*, *58*, 61, *62*, *64*
topLZ *(toolsFactorizationClass)*, 61
topLZ,Factorization,numeric,character,logical,ANY,ANY-method *(toolsFactorizationClass)*, 61
topLZ,Factorization,numeric,character,logical,ANY-method *(toolsFactorizationClass)*, 61
topLZ,Factorization,numeric,character,logical-method *(toolsFactorizationClass)*, 61
topLZ,Factorization-method *(toolsFactorizationClass)*, 61

vcftoFABIA, *4*, *7*, *9*, *11*, *12*, *16*, *18*, *20*, *23*, *27*, *30*, *31*, *37*, *40*, *42–44*, *46*, *48*, *49*, *51–53*, *55*, *57*, *58*, *62*, 63, *64*