Package: isoRelate (via r-universe)

June 27, 2024

Title Identity-by-Descent Inference of Haploid Recombining Organisms

Version 0.1.0

Description Pairwise identity by descent inference of haploid species using single nucleotide polymorphism data. isoRelate can detect IBD in the presence of multi-clonal infections and also provides a function for identifying loci under recent positive selection.

Depends R (>= 3.2.1)

License MIT + file LICENSE

LazyData TRUE

RoxygenNote 6.0.1.9000

LinkingTo Rcpp

Imports Rcpp (>= 0.12.5), data.table, doParallel, foreach, ggplot2 (>= 2.1.0), igraph, intergraph, ggnetwork, grDevices, graphics, utils, stats

Suggests knitr, rmarkdown

VignetteBuilder knitr

Repository https://plasmogenepi.r-universe.dev

RemoteUrl https://github.com/bahlolab/isoRelate

RemoteRef HEAD

RemoteSha 109ee470f1d1e9e4cd131045c23eea7047cff649

Contents

`AmatrixDD` *The matrix A in the equation Ax=b for 2 diploid chromosomes*

Description

The matrix A in the equation Ax=b for 2 diploid chromosomes

Usage

```
AmatrixDD(pop_allele_freqs, genotypes)
```

Arguments

`pop_allele_freqs`	A numeric vector of population allele frequencies for each SNP
`genotypes`	An integer matrix of genotype calls for a pair of isolates. Each coloumn represents and isolate and each row represents a SNP.

`AmatrixHD` *The matrix A in the equation Ax=b for 1 haploid and 1 diploid chromosome*

Description

The matrix A in the equation Ax=b for 1 haploid and 1 diploid chromosome

Usage

```
AmatrixHD(pop_allele_freqs, genotypes)
```

Arguments

`pop_allele_freqs`	A numeric vector of population allele frequencies for each SNP
`genotypes`	An integer matrix of genotype calls for a pair of isolates. Each coloumn represents and isolate and each row represents a SNP.

AmatrixHH *The matrix A in the equation Ax=b for 2 haploid chromosomes*

Description

The matrix A in the equation Ax=b for 2 haploid chromosomes

Usage

```
AmatrixHH(pop_allele_freqs, genotypes)
```

Arguments

`pop_allele_freqs`	A numeric vector of population allele frequencies for each SNP
`genotypes`	An integer matrix of genotype calls for a pair of isolates. Each coloumn represents and isolate and each row represents a SNP.

annotation_genes *Plasmodium Falciparum Gene Annotation Dataset*

Description

Gene annotations for the reference genome 3D7 were downloaded from `http://www.plasmodb.org/common/downloads/Current_Release/Pfalciparum3D7/gff/data/`, release PlasmoDB-28, last modified 23/03/2016.

Usage

```
annotation_genes
```

Format

A data frame with 6 columns of information

chr Chromosomes

start Base-pair positions of start of genes

end Base-pair positions of end of genes

strand The positive or negaitve gene strand

name Gene name, commonly abbreviated

gene_id Gene id

`areColors` *Internal function*

Description

`areColors()` checks if colour names are valid

Usage

```
areColors(x)
```

Arguments

`x`	vector of length 1 or higher containing numeric or character values for colours

`bVectorDD` *The vector b in the equation Ax=b for 2 diploid chromosomes*

Description

The vector b in the equation Ax=b for 2 diploid chromosomes

Usage

```
bVectorDD(genotypes)
```

Arguments

`genotypes`	An integer matrix of genotype calls for a pair of isolates. Each coloumn represents and isolate and each row represents a SNP.

`bVectorHD` *The vector b in the equation Ax=b for 1 haploid chromosome and 1 diploid chromosome*

Description

The vector b in the equation Ax=b for 1 haploid chromosome and 1 diploid chromosome

Usage

```
bVectorHD(genotypes)
```

Arguments

`genotypes`	An integer matrix of genotype calls for a pair of isolates. Each coloumn represents and isolate and each row represents a SNP.

bVectorHH *The vector b in the equation Ax=b for 2 haploid chromosomes*

Description

The vector b in the equation Ax=b for 2 haploid chromosomes

Usage

```
bVectorHH(genotypes)
```

Arguments

`genotypes`	An integer matrix of genotype calls for a pair of isolates. Each coloumn represents and isolate and each row represents a SNP.

calculateAlpha *Calculate alpha*

Description

Calculate alpha

Usage

```
calculateAlpha(number_states, initial_prob, meiosis, number_snps, genotypes,
  pop_allele_freqs, positions_cM, error, gender_1, gender_2)
```

Arguments

`number_states`	Integer. The number of IBD states in the model
`initial_prob`	A numeric vector containing the initial state probabilities
`meiosis`	Integer. The number of meiosis separating the two isolates
`number_snps`	Integer. The number of SNPs
`genotypes`	A integer martix containing the genotype calls for a pair of isolates
`pop_allele_freqs`	A numeric vector of population allele frequencies
`positions_cM`	A numeric vector of SNP genetic map positions in cM
`error`	Numeric. The genotype error rate
`gender_1`	Integer. The MOI estimate of isolate 1
`gender_2`	Integer. The MOI estimate of isolate 2

calculateBeta *Calculate beta*

Description

Calculate beta

Usage

```
calculateBeta(number_states, initial_prob, meiosis, number_snps, genotypes,
  pop_allele_freqs, positions_cM, scale, error, gender_1, gender_2)
```

Arguments

`number_states`	Integer. The number of IBD states in the model
`initial_prob`	A numeric vector containing the initial state probabilities
`meiosis`	Integer. The number of meiosis separating the two isolates
`number_snps`	Integer. The number of SNPs
`genotypes`	A integer martix containing the genotype calls for a pair of isolates
`pop_allele_freqs`	A numeric vector of population allele frequencies
`positions_cM`	A numeric vector of SNP genetic map positions in cM
`scale`	A numeric vector containing the scaling values used to scale alpha
`error`	Numeric. The genotype error rate
`gender_1`	Integer. The MOI estimate of isolate 1
`gender_2`	Integer. The MOI estimate of isolate 2

calculateGamma *Calculate gamma*

Description

Calculate gamma

Usage

```
calculateGamma(number_states, initial_prob, meiosis, number_snps, genotypes,
  pop_allele_freqs, positions_cM, error, gender_1, gender_2)
```

Arguments

`number_states` Integer. The number of IBD states in the model

`initial_prob` A numeric vector containing the initial state probabilities

`meiosis` Integer. The number of meiosis separating the two isolates

`number_snps` Integer. The number of SNPs

`genotypes` A integer martix containing the genotype calls for a pair of isolates

`pop_allele_freqs` A numeric vector of population allele frequencies

`positions_cM` A numeric vector of SNP genetic map positions in cM

`error` Numeric. The genotype error rate

`gender_1` Integer. The MOI estimate of isolate 1

`gender_2` Integer. The MOI estimate of isolate 2

`calculateLogLikelihood` *Calculate the log-likelihood of the data*

Description

Calculate the log-likelihood of the data

Usage

```
calculateLogLikelihood(number_states, initial_prob, meiosis, number_snps,
  genotypes, pop_allele_freqs, positions_cM, error, gender_1, gender_2)
```

Arguments

`number_states` Integer. The number of IBD states in the model

`initial_prob` A numeric vector containing the initial state probabilities

`meiosis` Integer. The number of meiosis separating the two isolates

`number_snps` Integer. The number of SNPs

`genotypes` A integer martix containing the genotype calls for a pair of isolates

`pop_allele_freqs` A numeric vector of population allele frequencies

`positions_cM` A numeric vector of SNP genetic map positions in cM

`error` Numeric. The genotype error rate

`gender_1` Integer. The MOI estimate of isolate 1

`gender_2` Integer. The MOI estimate of isolate 2

`calculateMeiosis` *Estimation of Meiosis*

Description

calculateMeiosis() estimates the number of meiosis separating a pair of isolates given the global IBD pop_allele_freqs estimates. This method is described in Purcell et al (2007).

Usage

```
calculateMeiosis(omega.0, omega.1, omega.2)
```

Arguments

`omega.0` A numeric value between 0 and 1 representing the pop_allele_freqs of sharing 0 alleles IBD. The sum of omega.0, omega.1 and omega.2 should equal 1.

`omega.1` A numeric value between 0 and 1 representing the pop_allele_freqs of sharing 1 alleles IBD.

`omega.2` A numeric value between 0 and 1 representing the pop_allele_freqs of sharing 2 alleles IBD.

Value

The number of meiosis separating the pair of isoaltes.

`calculateMissingness` *Calculate Missingness Proportions*

Description

Calculates the proportion of missing data for each SNPs or each isolate where missing values are denoted by -1. Missing values are calculated for each column of `genotypes` (where columns are isolates and rows are SNPs), however `genotypes` can be transposed to calculate missingness proportions for SNPs.

Usage

```
calculateMissingness(genotypes)
```

Arguments

`genotypes` An integer matrix of genotype data of the form -1, 0, 1 and 2 representing missing genotypes, homozygous reference, heterozygous and homozygous alternative respectively.

Value

A vector of length `n` where `n` is the number of columns in `genotypes`.

`calculatePopAlleleFreq` *Calculate Allele Frequencies for SNPs from Genotype Data*

Description

`calculatePopAlleleFreq` calculates reference allele frequencies for each SNP given genotype data.

Usage

```
calculatePopAlleleFreq(genotypes, moi)
```

Arguments

`genotypes`	An integer matrix of genotype data of the form -1, 0, 1 and 2 representing missing genotypes, homozygous reference, heterozygous and homozygous alternative respectively. Each column of `genotypes` represents a unique isolate and each row of `genotypes` represents a unique SNP.
`moi`	An integer vector of multiplicity of infection (MOI) estimates for each isoalte. Isolate MOI estimates should be ordered such that value `n` of `moi` corresponds to column `n` of `genotypes`.

`calculateScale` *Calculate scale*

Description

Calculate scale

Usage

```
calculateScale(number_states, initial_prob, meiosis, number_snps, genotypes,
  pop_allele_freqs, positions_cM, error, gender_1, gender_2)
```

Arguments

`number_states` Integer. The number of IBD states in the model

`initial_prob` A numeric vector containing the initial state probabilities

`meiosis` Integer. The number of meiosis separating the two isolates

`number_snps` Integer. The number of SNPs

`genotypes` A integer martix containing the genotype calls for a pair of isolates

`pop_allele_freqs` A numeric vector of population allele frequencies

`positions_cM` A numeric vector of SNP genetic map positions in cM

`error` Numeric. The genotype error rate

`gender_1` Integer. The MOI estimate of isolate 1

`gender_2` Integer. The MOI estimate of isolate 2

`calculateViterbi` *Calculate the Viterbi sequence*

Description

Calculate the Viterbi sequence

Usage

```
calculateViterbi(number_states, initial_prob, meiosis, number_snps, genotypes,
  pop_allele_freqs, positions_cM, error, gender_1, gender_2)
```

Arguments

`number_states` Integer. The number of IBD states in the model

`initial_prob` A numeric vector containing the initial state probabilities

`meiosis` Integer. The number of meiosis separating the two isolates

`number_snps` Integer. The number of SNPs

`genotypes` A integer martix containing the genotype calls for a pair of isolates

`pop_allele_freqs` A numeric vector of population allele frequencies

`positions_cM` A numeric vector of SNP genetic map positions in cM

`error` Numeric. The genotype error rate

`gender_1` Integer. The MOI estimate of isolate 1

`gender_2` Integer. The MOI estimate of isolate 2

clusterSummary *A function that prints summary information of clusters*

Description

A function that prints summary information of clusters

Usage

```
clusterSummary(cluster.list)
```

Arguments

cluster.list	a list of isolate groups where unique elements correspond to isolates in unique clusters

emissionProbDD *The emission probabilities for 2 diploid chromosomes*

Description

The emission probabilities for 2 diploid chromosomes

Usage

```
emissionProbDD(pop_allele_freq, genotype_1, genotype_2, ibd)
```

Arguments

pop_allele_freq	The population allele frequency for SNP i. This corresponds to the reference allele
genotype_1	The genotype for isolate 1 from the pair for SNP i
genotype_2	The genotype for isolate 2 from the pair for SNP i
ibd	The IBD state

`emissionProbHD` *The emission probabilities for 1 haploid chromosome and 1 diploid chromosome*

Description

The emission probabilities for 1 haploid chromosome and 1 diploid chromosome

Usage

```
emissionProbHD(pop_allele_freq, genotype_1, genotype_2, ibd, male_column,
  female_column)
```

Arguments

`pop_allele_freq`	The population allele frequency for SNP i. This corresponds to the reference allele
`genotype_1`	The genotype for isolate 1 from the pair for SNP i
`genotype_2`	The genotype for isolate 2 from the pair for SNP i
`ibd`	The IBD state
`male_column`	The haploid isolate from the pair. Either 1 or 2
`female_column`	The diploid isolate from the pair. Either 1 or 2

`emissionProbHH` *The emission probabilities for 2 haploid chromosomes*

Description

The emission probabilities for 2 haploid chromosomes

Usage

```
emissionProbHH(pop_allele_freq, genotype_1, genotype_2, ibd)
```

Arguments

`pop_allele_freq`	The population allele frequency for SNP i. This corresponds to the reference allele
`genotype_1`	The genotype for isolate 1 from the pair for SNP i
`genotype_2`	The genotype for isolate 2 from the pair for SNP i
`ibd`	The IBD state

`emissionProbMissingGeno` *Calculating the emission probability sumation when missing genotype calls present*

Description

Calculating the emission probability sumation when missing genotype calls present

Usage

```
emissionProbMissingGeno(pop_allele_freq, genotype_1, genotype_2, error,
  gender_1, gender_2, ibd_j)
```

Arguments

`pop_allele_freq`	The population allele frequency of SNP t
`genotype_1`	The genotype of isolate 1 at SNP t
`genotype_2`	The genotype of isolate 2 at SNP t
`error`	The genotype error rate
`gender_1`	The MOI estimate of isolate 1
`gender_2`	The MOI estimate of isolate 2
`ibd_j`	The IBD state

`genotypeErrorD` *The genotyping error probability for 1 diploid chromosome*

Description

The genotyping error probability for 1 diploid chromosome

Usage

```
genotypeErrorD(truth, observed, error)
```

Arguments

`truth`	The true genotype
`observed`	The observed genotype
`error`	The genotype error rate

`genotypeErrorH` *The genotyping error probability for 1 haploid chromosome*

Description

The genotyping error probability for 1 haploid chromosome

Usage

```
genotypeErrorH(truth, observed, error)
```

Arguments

`truth`	The true genotype
`observed`	The observed genotype
`error`	The genotype error rate

`getColourPaletteMajor` *IsoRelate Colour Palette for groups*

Description

`getColourPaletteMajor()` generates a spectrum colour palette with a specified number of colours.

Usage

```
getColourPaletteMajor(number.colours)
```

Arguments

`number.colours`	numeric. The number of colours to return.

Value

A character vector of length=`number.colours` containing a colour specturm.

getColourPaletteMinor *getColourPaletteMinor() generates a specified number of shades of a given colour.*

Description

getColourPaletteMinor() generates a specified number of shades of a given colour.

Usage

```
getColourPaletteMinor(major.colour, number.colours)
```

Arguments

major.colour	character. The colour name or code.
number.colours	numeric. The number of colours to return.

Value

A character vector of length=number.colours containing colour shades, excluding white.

getGenotypes *Pre-Analysis Data Processing*

Description

getGenotypes() performs pre-analysis data processing of PLINK formatted unphased genotype data, including removal of SNPs and isolates with high proportions of missing data and SNPs with low minor allele frequencies. It also calculates SNP allele frequencies from either the input dataset or a specified reference dataset.

Usage

```
getGenotypes(ped.map, reference.ped.map = NULL, maf = 0.01,
  isolate.max.missing = 0.1, snp.max.missing = 0.1, chromosomes = NULL,
  input.map.distance = "cM", reference.map.distance = "cM")
```

Arguments

ped.map A list with 2 objects:

1. A data frame which contains the PLINK PED information. The first six columns of this data frame are:
   (a) Family ID (type "character", "numeric" or "integer")
   (b) Isolate ID (type "character", "numeric" or "integer")
   (c) Paternal ID (type "character", "numeric" or "integer")

   (d) Maternal ID (type "character", "numeric" or "integer")
   (e) Multiplicity of infection (MOI) (1 = single infection or haploid, 2 = multiple infections or diploid)
   (f) Phenotype (type "character", "numeric" or "integer")

   where each row describes a single isolate. The IDs are alphanumeric: the combination of family and isolate ID should uniquely identify a sample. The paternal, maternal and phenotype columns are not used by isoRelate, however they are required for completeness of a standard pedigree and are typically filled with the numeric value zero. Columns 7 onwards are the isolate genotypes where the A and B alleles are coded as 1 and 2 respectively and missing data is coded as 0. All SNPs must have two alleles specified and each allele should be in a separate column. For example, the alleles in columns 7 and 8 correspond to the unphased genotypes of SNP 1 in the map file. For single infections, genotypes should be specified as homozygous. Either both alleles should be missing (i.e. 0) or neither. Column names are not required.

2. A data frame which contains the PLINK MAP information. This data frame contains exactly four columns of information:
   (a) Chromosome (type "character", "numeric" or "integer")
   (b) SNP identifier (type "character")
   (c) Genetic map distance (centi morgans, cM, or morgans, M) (type "numeric")
   (d) Base-pair position (type "numeric" or "integer")

   where each row describes a single SNP Genetic map distance and base-pair positions are expected to be positive values. The MAP file must be ordered by increasing genetic map distance. SNP identifiers can contain any characters expect spaces or tabs; also you should avoid * symbols in the names. The MAP file must contain as many markers as are in the PED file. Column names are not required.

`reference.ped.map`

An optional list containing reference data used to calculate SNP allele frequencies. The list has 2 objects in the same format as `ped.map`. The default value is `reference.ped.map=NULL` and isoRelate will calculate the SNP allele frequencies from the input data. This is not recommended for small datasets or datasets of mixed populations.

`maf` A numeric value denoting the smallest minor allele frequency allowed in the analysis. The default value is 0.01.

`isolate.max.missing`

A numeric value denoting the maximum proportion of missing data allowed for each isolate. The default value is 0.1.

`snp.max.missing`

A numeric value denoting the maximum proportion of missing data allowed for each SNP. The default value is 0.1.

`chromosomes` A vector containing a subset of chromosomes to perform formatting on. The default value is `chromosomes=NULL` which will reformat all genotypes for all chromosomes in the MAP data frame.

`input.map.distance`
: A character string of either "M" or "cM" denoting whether the genetic map distances in the input MAP data frame are in Morgans (M) or centi-Morgans (cM). The default is cM.

`reference.map.distance`
: A character string of either "M" or "cM" denoting whether the genetic map distances in the reference MAP data frame are in Morgans (M) or centi-Morgans (cM). The default is cM.

Value

A list of two objects named `pedigree` and `genotypes`:

1. A pedigree containing the isolates that remain after filtering. The pedigree is the first six columns of the PED file and these columns are headed `fid`, `iid`, `pid`, `mid`, `moi` and `aff` respectively.
2. A data frame with the first five columns:
   (a) Chromosome (type "`character`", "`numeric`" or "`integer`")
   (b) SNP identifiers (type "`character`")
   (c) Genetic map distance (Morgans, M) (type "`numeric`")
   (d) Base-pair position (type "`integer`")
   (e) Population allele frequency (type "`integer`")

   where each row describes a single SNP. These columns are headed `chr`, `snp_id`, `pos_M`, `pos_bp` and `freq` respectively. Columns 6 onwards contain the genotype data for each isolate, where a single column corresponds to a single isolate. These columns are labeled with merged family IDs and isolate IDs separated by a slash symbol (/).

See Also

`getIBDparameters` and `getIBDsegments`.

Examples

```
# take a look at the data
str(png_pedmap)

# reformat and filter to call genotypes
my_genotypes <- getGenotypes(ped.map = png_pedmap,
                             reference.ped.map = NULL,
                             maf = 0.01,
                             isolate.max.missing = 0.1,
                             snp.max.missing = 0.1,
                             chromosomes = NULL,
                             input.map.distance = "cM",
                             reference.map.distance = "cM")
```

getIBDiclusters *Interval Cluster Networks*

Description

getIBDiclusters() produces a network of clusters of isolates that have been inferred IBD over a specified interval. Isolates that are not IBD over the interval are not included in the network or output. The networks are created using the R package igraph.

Usage

```
getIBDiclusters(ped.genotypes, ibd.segments, interval = NULL, prop = 0,
  hi.clust = FALSE)
```

Arguments

ped.genotypes	A list containing 2 objects. See the Value description in getGenotypes for more details on this input.
ibd.segments	A data frame containing the IBD segments detected by isoRelate. See the Value description in getIBDsegments for more details on this input.
interval	A vector of length 3 containing the region to identify clusters over. This vector should contain the chromosome ID, the start of the interval in base-pairs and the end of the interval in base-pairs; in this order respectively.
prop	Numeric value between 0 and 1 (inclusive). The minimum proportion of an interval (in base-pairs) shared IBD between a pair of isolates in order for the pair to be included in the network. For example, if prop=1 then two isolates will be included if they are IBD over the entire interval, whereas prop=0.5 will include isolates that are IBD over at least 50% of the interval. The default is prop=0, which includes isolates if they have an IBD segment that overlaps the interval in any way.
hi.clust	Logical. Whether to perform hierarchical clustering using the fastgreedy.community approach in the igraph package.

Value

A list of three objects named clusters, i.network and hi.clust:

1. A list where each object contains the names of isolates that form a disjoint cluster in the network. If hierarchical clustering has been performed then the clusters may not be disjoint.
2. An igraph network used in the construction of network plots. See http://igraph.org/r/ doe more details.
3. Logical. Whether or not hierarchical clustering has been performed.

See Also

getGenotypes, getIBDsegments and getIBDpclusters.

Examples

```
# generate the isolates who are IBD over the Plasmodium falciparum CRT gene
my_i_clusters <- getIBDiclusters(ped.genotypes = png_genotypes,
                                 ibd.segments = png_ibd,
                                 interval = c("Pf3D7_07_v3", 403222, 406317),
                                 prop=0,
                                 hi.clust = FALSE)

str(my_i_clusters)
```

---

getIBDiR *Selection Significance Statistic*

Description

getIBDiR() calculates a summary statistic for each SNP that can be used to assess the significance of excess IBD sharing at genomic loci, thus identifying regions under positive selection. First relatedness between isolates and SNP allele frequencies are accounted for, then normalization procedures are applied where we assume our transformed summary statistic follows a chi-squared distribution with 1 degree of freedom. This allows the calculation of -log10 (P-values) which we denote as the iR statistic. SNPs with iR values greater than some threshold (i.e. -log10 (P-values) > -log10 (0.05)) provide evidence of positive selection. getIBDiR can return NA iR statistics for a number of reasons, including trying to generate iR statistics when there are no IBD pairs or when all pairs are IBD, or when only several isolates are analyzed.

Usage

```
getIBDiR(ped.genotypes, ibd.matrix, groups = NULL)
```

Arguments

ped.genotypes A list containing 2 objects. See the Value description in getGenotypes for more details on this input.

ibd.matrix A data frame containing the binary IBD information for each SNP and each pair. See the returned Value in getIBDmatrix for more details.

groups A data frame with 3 columns of information:

1. Family ID
2. Isolate ID
3. Group ID

where IBD proportions are calculated for

1. all pairs of isolates within the same group
2. all pairwise-group comparisons where isolates belong to different groups

Group ID, for example, can be the geographic regions where the isolates were collected. The default is groups=NULL and IBD proportions will be calculated over all pairs.

Value

A data frame the following 7 columns:

1. Chromosome (type "character", "numeric" or "integer")
2. SNP identifiers (type "character")
3. Genetic map distance (centi morgans, cM) (type "numeric")
4. Base-pair position (type "integer")
5. Population (type "character" or "numeric")
6. Subpopulation (type "character" or "numeric")
7. iR statistic (type "numeric")
8. -log10 p vlaue (type "numeric")

where each row describes a unique SNP. The column `Population` is filled with 1s by default, while `Subpopulation` contains the group IDs from `groups`, where the proportion of pairs IBD has been calculated for all pairs of isolates belonging to the same group as well as all pairs of isolates where each isolate belongs to a different group. If `groups=NULL` then `Subpopulation` will be filled with 0s also. The population columns have been included for plotting purposes. The data frame is headed `chr`, `snp_id`, `pos_M`, `pos_bp`, `pop`, `subpop`, `iR` and `log10_pvalue` respectively.

See Also

`getGenotypes`, `getIBDmatrix` and `getIBDproportion`.

Examples

```
# generate a binary IBD matrix
my_matrix <- getIBDmatrix(ped.genotypes = png_genotypes,
                          ibd.segments = png_ibd)

# calculate the significance of IBD sharing
my_iR <- getIBDiR(ped.genotypes = png_genotypes,
                  ibd.matrix = my_matrix,
                  groups = NULL)
```

`getIBDmatrix` *Binary IBD Matrix*

Description

`getIBDmatrix()` produces a binary matrix of IBD (1) and non-IBD (0) results for each SNP and isolate pair combination. Each row identifies a unique SNP while each column identifies a unique isolate pair.

Usage

```
getIBDmatrix(ped.genotypes, ibd.segments)
```

Arguments

`ped.genotypes` A list containing 2 objects. See the `Value` description in `getGenotypes` for more details on this input.

`ibd.segments` A data frame containing the IBD segments detected by isoRelate. See the `Value` description in `getIBDsegments` for more details on this input.

Value

A data frame with the first four columns:

1. Chromosome (type `"character"`, `"numeric"` or `"integer"`)
2. SNP identifiers (type `"character"`)
3. Genetic map distance (centi morgans, cM) (type `"numeric"`)
4. Base-pair position (type `"integer"`)

where each row describes a unique SNP. Columns 1-4 are headed `chr`, `snp_id`, `pos_M` and `pos_bp` respectively. Columns 5 onwards contain the binary IBD information for each isolate pair, where a single column corresponds to a single pair. These columns are labeled with merged family IDs and isolate IDs separated by a slash symbol (/). For example fid1/iid1/fid2/iid2.

See Also

`getGenotypes`, `getIBDsegments`, `getIBDproportion`, `getIBDiR`.

Examples

```
# generate a binary IBD matrix
my_matrix <- getIBDmatrix(ped.genotypes = png_genotypes,
                          ibd.segments = png_ibd)
```

`getIBDparameters` *Parameter Estimation*

Description

`getIBDparameters()` estimates the number of meiosis and the probabilities of sharing 0, 1 and 2 alleles IBD between all pairwise combinations of isolates.

Usage

```
getIBDparameters(ped.genotypes, number.cores = 1)
```

Arguments

`ped.genotypes` A list containing 2 objects. See the `Value` description in `getGenotypes` for more details on this input. Note the family IDs and isolate IDs in object 1 of this list must match the family IDs and isolate IDs in the header of object 2 of this list.

`number.cores` Positive integer. The number of cores used for parallel execution.

Value

A data frame with the following eight columns:

1. Family 1 ID
2. Isolate 1 ID
3. Family 2 ID
4. Isolate 2 ID
5. The number of meiosis separating the pair
6. Probability of sharing 0 alleles IBD
7. Probability of sharing 1 allele IBD
8. Probability of sharing 2 alleles IBD

where each row describes a unique pair of isolates. The data frame is headed `fid1`, `iid1`, `fid2`, `iid2`, `m`, `ibd0`, `ibd1` and `ibd2` respectively.

See Also

`getGenotypes` and `getIBDsegments`.

Examples

```
# following processing and filtering of genotype data,
# we estimate the proportion of genome shared IBD
my_parameters <- getIBDparameters(ped.genotypes = png_genotypes,
                                  number.cores = 1)

head(my_parameters)
```

`getIBDpclusters` *Genome Cluster Networks*

Description

`getIBDpclusters()` produces a network of clusters of isolates that share a minimum proportion of genome IBD. Isolates that do not share a minimum proportion IBD are not included in the network or output. The networks are created using R package `igraph`.

Usage

```
getIBDpclusters(ped.genotypes, ibd.segments, prop = 1, hi.clust = FALSE)
```

Arguments

`ped.genotypes` A list containing 2 objects. See the `Value` description in `getGenotypes` for more details on this input.

`ibd.segments` A data frame containing the IBD segments detected by isoRelate. See the `Value` description in `getIBDsegments` for more details on this input.

`prop` Numeric value between (0,1]. The minimum proportion of genome shared IBD between a pair of isolates in order for the pair to be included in the network. For example, if `prop=1` then two isolates will be included if they are IBD over the entire genome, i.e. identical isolates, whereas `prop=0.5` will include isolates that share at least 50% of their genome IBD. The default is `prop=1`.

`hi.clust` Logical. Whether to perform hierarchical clustering using the `fastgreedy.community` approach in the `igraph` package.

Value

A list of three objects named `clusters`, `i.network` and `hi.clust`:

1. A list where each object contains the names of isolates that form a disjoint cluster in the network. If hierarchical clustering has been performed then the clusters may not be disjoint.
2. An `igraph` network used in the construction of network plots. See `http://igraph.org/r/` doe more details.
3. Logical. Whether or not hierarchical clustering has been performed.

See Also

`getGenotypes`, `getIBDsegments` and `getIBDiclusters`.

`getIBDposterior` *IBD Posterior Probabilities*

Description

`getIBDposterior()` calculates the posterior probabilities of IBD sharing between pairs of isolates.

Usage

```
getIBDposterior(ped.genotypes, parameters, number.cores = 1, error = 0.001)
```

Arguments

`ped.genotypes` A list containing 2 objects. See the `Value` description in `getGenotypes` for more details on this input.

`parameters` A data frame containing meioses and IBD probability estimates for all pairwise combinations of isolates. See the `Value` description in `getIBDparameters` for more details on this input.

`number.cores` Positive integer. The number of cores used for parallel execution.

`error` The genotyping error rate. The default value is 0.001.

Value

A data frame with the first four columns:

1. Chromosome
2. SNP identifiers
3. Genetic map distance
4. Base-pair position

where each row describes a single SNP. These columns are headed `chr`, `snp_id`, `pos_M` and `pos_bp` respectively. Columns 5 onwards contain the posterior probabilities for each pair of isolates, where a single column corresponds to one pair of isolates. These columns are labeled with merged family IDs and isolate IDs separated by a slash symbol (/).

Examples

```
## Not run:
# calculate the posterior probability of IBD sharing
# note: this can take a while to run if there are many pairs
my_posterior <- getIBDposterior(ped.genotypes = png_genotypes,
                                parameters = png_parameters,
                                number.cores = 1,
                                error = 0.001)

head(my_posterior[,1:10])

## End(Not run)
```

`getIBDproportion` *Proportion of Pairs IBD*

Description

`getIBDproportion()` calculates the proportion of pairs inferred IBD at each SNP.

Usage

```
getIBDproportion(ped.genotypes, ibd.matrix, groups = NULL)
```

Arguments

`ped.genotypes` A list containing 2 objects. See the `Value` description in `getGenotypes` for more details on this input.

`ibd.matrix` A data frame containing the binary IBD information for each SNP and each pair. See the returned `Value` in `getIBDmatrix` for more details.

`groups` A data frame with 3 columns of information:

1. Family ID

2. Isolate ID
3. Group ID

where, if specified, IBD proportions are calculated for

1. all pairs of isolates within the same group
2. all pairwise-group comparisons where isolates belong to different groups

Group ID, for example, can be the geographic regions where the isolates were collected. The default is groups=NULL and IBD proportions will be calculated over all pairs.

Value

A data frame the following 7 columns:

1. Chromosome (type "character", "numeric" or "integer")
2. SNP identifiers (type "character")
3. Genetic map distance (centi morgans, cM) (type "numeric")
4. Base-pair position (type "integer")
5. Population (type "character" or "numeric")
6. Subpopulation (type "character" or "numeric")
7. Proportion of pairs IBD (type "integer")

where each row describes a unique SNP. The column Population is filled with 1s by default, while Subpopulation contains the group IDs from groups, where the proportion of pairs IBD has been calculated for all isolates belonging to the same group as well as all isolates from different groups. If groups=NULL then Subpopulation will be filled with 1s also. The population columns have been included for plotting purposes. The data frame is headed chr, snp_id, pos_M, pos_bp, pop, subpop and prop_ibd respectively.

See Also

getGenotypes, getIBDmatrix and getIBDiR.

Examples

```
# generate a binary IBD matrix
my_matrix <- getIBDmatrix(ped.genotypes = png_genotypes,
                          ibd.segments = png_ibd)

# calculate the proportion of pairs IBD at each SNP
my_proportion <- getIBDproportion(ped.genotypes = png_genotypes,
                                  ibd.matrix = my_matrix,
                                  groups = NULL)

# creating a stratification dataset
my_groups <- png_genotypes[[1]][,1:3]
my_groups[1:10,"pid"] <- "a"
my_groups[11:25,"pid"] <- "b"
my_groups[26:38,"pid"] <- "c"
```

```
my_proportion <- getIBDproportion(ped.genotypes = png_genotypes,
                                  ibd.matrix = my_matrix,
                                  groups = my_groups)

head(my_proportion)
```

---

getIBDsegments *IBD Segment Detection*

Description

getIBDsegments() detects genomic regions shared IBD between all pairwise combinations of isolates.

Usage

```
getIBDsegments(ped.genotypes, parameters, number.cores = 1,
  minimum.snps = 20, minimum.length.bp = 50000, error = 0.001)
```

Arguments

ped.genotypes	A list containing 2 objects. See the Value description in getGenotypes for more details on this input.
parameters	A data frame containing meioses and IBD probability estimates for all pairwise combinations of isolates. See the Value description in getIBDparameters for more details on this input.
number.cores	Positive integer. The number of cores used for parallel execution.
minimum.snps	An integer value denoting the minimum number of SNPs in an IBD segment for it to be reported. The default value is 20 SNPs.
minimum.length.bp	The minimum length of a reported IBD segment. The default value is 50,000 bp.
error	The genotyping error rate. The default value is 0.001.

Value

A data frame with the following columns

1. Family 1 ID
2. Isolate 1 ID
3. Family 2 ID
4. Isolate 2 ID
5. Chromosome
6. Start SNP

7. End SNP
8. Start position bp
9. End position bp
10. Start position M
11. End position M
12. Number of SNPs
13. Length bp
14. Length M
15. IBD status (1 = one allele shared IBD, 2 = two alleles shared IBD)

where each row describes a unique IBD segment. The data frame is headed `fid1`, `iid1`, `fid2`, `iid2`, `chr`, `start_snp`, `end_snp`,`start_position_bp`, `end_position_bp`, `start_position_M`, `end_position_M`, `number_snps`, `length_bp`, `length_M` and `ibd_status` respectively.

See Also

`getGenotypes` and `getIBDparameters`.

Examples

```
## Not run:
# prior to IBD detection, parameter estimates must be estimated.
# Assuming this has been done, IBD inference is performed
my_ibd <- getIBDsegments(ped.genotypes = png_genotypes,
                         parameters = png_parameters,
                         number.cores = 1,
                         minimum.snps = 20,
                         minimum.length.bp = 50000,
                         error = 0.001)

head(my_ibd)

## End(Not run)
```

`getIBDsummary` *IBD Segment Summary*

Description

`getIBDsummary()` prints a brief summary of the detected IBD segments to the console.

Usage

```
getIBDsummary(ped.genotypes, ibd.segments)
```

Arguments

ped.genotypes A list containing 2 objects. See the Value description in getGenotypes for more details on this input.

ibd.segments A data frame containing the IBD segments detected by isoRelate. See the Value description in link{getIBDsegments} for more details on this input.

getOverlap *function to find IBD that overlap interval*

Description

function to find IBD that overlap interval

Usage

```
getOverlap(region.1, region.2)
```

Arguments

region.1 IBD segment boundary

region.2 interest interval

groupPairs *Group Combinations for Analysis*

Description

Creates a data frame containing family IDs and isolate IDs for each pair to be analysed. Each row corresponds to a unique pair.

Usage

```
groupPairs(group)
```

Arguments

group A character vector of all family IDs

haplotypeToGenotype *Call Genotypes from Haplotype Data*

Description

haplotypeToGenotype transforms PLINK haplotype data into genotype data of the form -1, 0, 1 and 2 representing missing genotypes, homozygous reference, heterozygous and homozygous alternative respectively. Haploid isolates are coded as diploid although will not have heterozygous genotypes.

Usage

```
haplotypeToGenotype(haplotypes, moi)
```

Arguments

haplotypes	An integer matrix of haplotype data in PLINK format. A allele is denoted 1, B allele is denoted 2 and missing data is denoted 0
moi	An integer vector of multiplicity of infection (MOI) estimates for each isoalte. Isolate MOI estimates should be ordered such that value n of moi corresponds to column n of haplotypes.

Value

A matrix with genotype calls where columns correspond to isolates and rows correspond to SNPs

highlight_genes *Plasmodium Falciparum Gene Highlight Dataset*

Description

Gene annotations for 31 commonly studied Plasmodium falciparum genes, including antimalarial drug resistance genes, vaccine candidates and var genes. Gene annotations are for the reference genome 3D7 were downloaded from http://www.plasmodb.org/common/downloads/Current_Release/Pfalciparum3D7/gff/data/, release PlasmoDB-28, last modified 23/03/2016.

Usage

```
highlight_genes
```

Format

A data frame with 5 columns of information

chr Chromosomes

start Base-pair positions of start of genes

end Base-pair positions of end of genes

name Gene name, commonly abbreviated

gene_id Gene id

IBDLabel *Internal Function*

Description

`IBDLabel` is a function used to label unique IBD segments in a pair of isolates for a particular chromosome by determining breakpoints in IBD vs non-IBD regions. IBD segments are labelled in sequential order genome wide.

Usage

```
IBDLabel(snp_id, number_snps)
```

Arguments

`snp_id`	A numeric vector of SNP identifiers for IBD segments on the chromosome of interest.
`number_snps`	integer. The number of IBD SNPs for the chromosome of interest.

IBDMatrix *Binary Matrix of IBD*

Description

Creates a binary matrix of IBD (1) and non-IBD (0) with each row representing a single SNP and each column representing a unique pair. The number of rows is equal to the total number of SNPs and the number of columns is equal to the number of pairs.

Usage

```
IBDMatrix(chromosomes, positions_bp, number_pairs, ibd_pairs_colnumbers,
  ibd_chromosomes, ibd_start_bp, ibd_stop_bp)
```

Arguments

`chromosomes` A character vector containing the corresponding chromosome for each SNP

`positions_bp` A numeric vector containing the corresponding bp position for each SNP

`number_pairs` Numeric. The total number of pairs analysed

`ibd_pairs_colnumbers` A numeric vector corresponding to column numbers in the output matrix where each unique number refers to a unique pair with IBD inferred

`ibd_chromosomes` A character vector containing the chromosome for each detected IBD segment

`ibd_start_bp` A numeric vector containing the base-pair position for the start of each detected IBD segment

`ibd_stop_bp` A numeric vector containing the base-pair position for the end of each detected IBD segment

IBDparameters *IBDparameters() calculates IBD probabilities then estimates the number of meiosis for an isolate pair.*

Description

IBDparameters() calculates IBD probabilities then estimates the number of meiosis for an isolate pair.

Usage

```
IBDparameters(genotypes, pop_allele_freqs, gender_1, gender_2)
```

Arguments

`genotypes` An integer matrix of genotype calls for a pair of isolates. Each coloumn represents and isolate and each row represents a SNP.

`pop_allele_freqs` A numeric vector of population allele frequencies for each SNP.

`gender_1` An integer denoting the MOI value of isoalte 1.

`gender_2` An integer denoting the MOI value of isoalte 2.

Value

A vector of 4 values representing the number of meiosis and the probabilities of sharing 0, 1 and 2 alleles IBD respectively.

IBDTable *Internal Function*

Description

IBDTable() produces summaries of detected IBD segments for a single pair of isolates. These summaries include the genetic map start and end of IBD segments in bp, cM and SNP identifiers; and lengths of IBD segments in bp, cM and SNPs.

Usage

```
IBDTable(ibd.results)
```

Arguments

ibd.results A data frame containing family ID and isolate ID for isolate 1, family ID and isolate ID for isolate 2, numeric SNP identifiers, chromosome identifiers, genetic map positions of SNPs in Morgans (M) and base-pairs (bp) and the Viterbi results respectively, for all SNPs.

Value

A data frame containing a summary of all IBD segments inferred for this pair of isolates. The data frame contains the following columns:

1. Family 1 ID
2. Isolate 1 ID
3. Family 2 ID
4. Isolate 2 ID
5. Chromosome
6. SNP identifier
7. Start SNP
8. End SNP
9. Start position bp
10. End position bp
11. Start position M
12. End position M
13. Number of SNPs
14. Length bp
15. Length M
16. IBD status (1 = 1 allele shared IBD, 2 = 2 alleles shared IBD)

`iRfunction` *Internal Function*

Description

`iRfunction()` calculates the iR statistic used to assess the significance of excess IBD sharing at loci in the genome. The final statistic, -log10 (P-values), is returned for each SNP.

Usage

```
iRfunction(locus.matrix, frequency)
```

Arguments

`locus.matrix`	A data frame containing binary IBD values. See `getIBDmatrix` for more details.
`frequency`	A vector of population allele frequencies for each SNP.

`isolatePairs` *Pair Combinations for Analysis*

Description

Creates a data frame containing family IDs and isolate IDs for each pair to be analysed. Each row corresponds to a unique pair.

Usage

```
isolatePairs(fid, iid)
```

Arguments

`fid`	A character vector of all family IDs
`iid`	A character vector of all individual ID

`merge_lists` *Internal Function*

Description

`merge_lists()` is a function used to merge summary IBD results for multiple pairs when running the IBD analysis on multiple cores

Usage

```
merge_lists(A, B)
```

Arguments

`A`	List with 2 objects for one pair. Object 1 is the IBD summaries for a pair and object 2 is the IBD posterior probabilities.
`B`	List with 2 objects for another pair. Object 1 is the IBD summaries for a pair and object 2 is the IBD posterior probabilities.

Value

A list with 2 objects containing merged lists from `A` and `B` above.

`my_ibd` *IBD Segments For The Papua New Guinea Dataset*

Description

The IBD segments inferred using isoRelate with the parameter settings as in the Vignette.

Usage

```
my_ibd
```

Format

A data frame with

fid1 Family 1 ID

iid1 Isolate 1 ID

fid2 Family 2 ID

iid2 Isolate 2 ID

chr Chromosome

start_snp IBD segment start SNP ID

end_snp IBD segment end SNP ID

start_position_bp IBD segment start SNP base-pair position

end_position_bp IBD segment end SNP base-pair position

start_position_M IBD segment start SNP morgan position

end_position_M IBD segment end SNP morgan position

number_snps Number of SNPs in IBD segment

length_bp Length of IBD segment in base-pairs

length_M Length of IBD segment in morgans

ibd_status The number of alleles shared IBD (either 1 or 2)

`plotIBDclusters` *Plot Cluster Networks*

Description

`plotIBDclusters()` Produces a figure of an isoRelate cluster network, where unique isolates are represented by vertices and a line is drawn between two vertices if the isolates have been inferred IBD via the criteria specified in either `getIBDiclusters` or `getIBDpclusters`.The networks are created using the R package `igraph`.

Usage

```
plotIBDclusters(ped.genotypes, clusters, groups = NULL, vertex.color = NULL,
  vertex.frame.color = "white", vertex.size = 4, vertex.name = FALSE,
  edge.color = "gray60", edge.width = 0.8, mark.border = "white",
  mark.col = "gray94", add.legend = TRUE, legend.x = -1.5,
  legend.y = -0.25, layout = NULL, return.layout = FALSE)
```

Arguments

`ped.genotypes` A list containing 2 objects. See the `Value` description in `getGenotypes` for more details on this input.

`clusters` A named list of three objects containing network information. See the `Value` description in either `getIBDiclusters` or `getIBDpclusters` for more details on this input.

`groups` A data frame with 3 columns of information:

1. Family ID
2. Isolate ID
3. Group ID

Group ID, for example, can be the geographic regions where the isolates were collected. If `groups` is specified then each isolate in the pedigree must belong to a group. Vertices in the network will be colored according to group allocation. The default is `groups=NULL` and all vertices will have the same color.

`vertex.color` A vector of characters or numeric values of the vertex colors in the network. If `groups` is specified then `vertex.color` should contain the same number of colors as unique groups.

`vertex.frame.color`
Character string or numeric value. A single color that will be used as the vertex border. Default is `vertex.fame.color="white"`.

`vertex.size` Numeric value indicating the size of the vertices in the network. Default is `vertex.size=4`.

`vertex.name` Logical. Whether to add isolate names to the vertices. Default is `vertex.name=FALSE`.

`edge.color` Character string or numeric value. A single color to be used for all edges. Default is `edge.color="gray60"`.

`edge.width` Numeric. A single value indicating the width of the edges. Default is `edge.width=0.8`.

`mark.border` Character string or numeric value. A single color to be used for all borders in hierarchical clustering groups. Default is `mark.border="white"`.

`mark.col` Character string or numeric value. A single color to be used to fill hierarchical clustering groupings. Default is `mark.col="gray94"`.

`add.legend` Logical. Whether to include a legend in the plot. Default is `add.legend=TRUE`.

`legend.x` Numerical. A single value indicating the x-coordinate of the legend, with default `legend.x=-1.5`.

`legend.y` Numerical. A single value indicating the y-coordinate of the legend, with default `legend.y=-0.25`.

`layout` A matrix containing the x and y coordinates of the vertices, generated using the Fruchterman-Reingold force-directed layout.

`return.layout` Logical. Whether or not to return the layout matrix (vertex positions) in the network. This layout can be used as the input for the parameter `layout` to avoid different network configurations each time `plotClusters()` is run on the same network.

See Also

`getGenotypes`, `getIBDpclusters` and `getIBDiclusters`.

Examples

```
# generate the isolates who are IBD over the Plasmodium falciparum CRT gene
my_i_clusters <- getIBDiclusters(ped.genotypes = png_genotypes,
                                 ibd.segments = png_ibd,
                                 interval = c("Pf3D7_07_v3", 403222, 406317),
                                 prop=0,
                                 hi.clust = FALSE)

str(my_i_clusters)

# creating a stratification dataset
my_groups <- png_genotypes[[1]][,1:3]
my_groups[1:10,"pid"] <- "a"
```

```
my_groups[11:25,"pid"] <- "b"
my_groups[26:38,"pid"] <- "c"

# plot the network of clusters
plotIBDclusters(ped.genotypes = png_genotypes,
                clusters = my_i_clusters,
                groups = my_groups,
                vertex.color = NULL,
                vertex.frame.color = "white",
                vertex.size = 4,
                vertex.name = FALSE,
                edge.color = "gray60",
                edge.width = 0.8,
                mark.border = "white",
                mark.col = "gray94",
                add.legend = TRUE,
                legend.x = -1.5,
                legend.y = -0.25,
                layout = NULL,
                return.layout = FALSE)

# generate the isolates who share at least than 90% of their genome IBD
my_p_clusters <- getIBDpclusters(ped.genotypes = png_genotypes,
                                 ibd.segments = png_ibd,
                                 prop=0.9,
                                 hi.clust = FALSE)

# plot the network of clusters
plotIBDclusters(ped.genotypes = png_genotypes,
                clusters = my_p_clusters,
                groups = my_groups,
                vertex.color = NULL,
                vertex.frame.color = "white",
                vertex.size = 4,
                vertex.name = FALSE,
                edge.color = "gray60",
                edge.width = 0.8,
                mark.border = "white",
                mark.col = "gray94",
                add.legend = TRUE,
                legend.x = -1.5,
                legend.y = -0.25,
                layout = NULL,
                return.layout = FALSE)
```

plotIBDiR *Plot iR Statistics*

Description

plotIBDiR() plots the -log10 (p-values) used to assess the significance of excess IBD sharing.

Usage

```
plotIBDiR(ibd.iR, interval = NULL, annotation.genes = NULL,
  annotation.genes.color = NULL, highlight.genes = NULL,
  highlight.genes.labels = TRUE, highlight.genes.color = NULL,
  highlight.genes.alpha = 0.1, point.size = 1, point.color = NULL,
  add.rug = FALSE, plot.title = NULL, add.legend = FALSE,
  facet.label = TRUE, facet.scales = "fixed")
```

Arguments

`ibd.iR` A data frame containing the iR summary statistics for each SNP. See the returned `Value` in `getIBDiR` for more details. If multiple subpopulations are specified (column name "subpop") then the iR statistics for each subpopulation will be plotted on a separate facet in the figure (see `http://docs.ggplot2.org/current/facet_grid.html` on faceting). If there are many subpopulations (>8) it may be better to plot subsets of the subpopulations as apposed to all subpopulations in a single figure. If multiple populations (column name "pop") are specified then only the first population will be included in the figure. Genomic locations of annotation genes can be included in the figure and specific intervals can be highlighted.

`interval` A vector of length 3 containing the genomic locations of a specific region to plot. This vector should contain the chromosome ID, the start of the interval in base-pairs and the end of the interval in base-pairs; in this order respectively. The default is `interval=NULL` which will plot iR statistics over all chromosomes in `ibd.iR`.

`annotation.genes`

A data frame containing information on annotation genes to be included in the figure. This data frame must have at least 5 columns of information:

1. Chromosome (type `"numeric"` or `"integer"`)
2. Gene name (type `"character"`)
3. Start location of the gene in base-pairs (type `"numeric"` or `"integer"`)
4. End location of the gene in base-pairs (type `"numeric"` or `"integer"`)
5. Gene strand (+ or -) (type `"character"`)

`annotation.genes` must contain the following headers `chr`, `name`, `start`, `end` and `strand`. This data frame does not have to be in a specific order, however it must contain all of the above information with respective labels. The default is annotation.genes=NULL.

`annotation.genes.color`

A vector of characters or numeric values containing the two colors according to gene stand (positive (+) or negative (-))

`highlight.genes`

A data frame containing information of genes or regions to highlight. The data frame must have at least 4 columns of information:

1. Chromosome (type `"numeric"` or `"integer"`)
2. Gene name (type `"character"`)
3. Start location of the gene in base-pairs (type `"numeric"` or `"integer"`)

4. End location of the gene in base-pairs (type "numeric" or "integer")

highlight.genes should contain the following headers chr, name, start and end. This data frame does not have to be in a specific order, however it must contain all of the above information with respective labels. The default is highlight.genes=NULL.

highlight.genes.labels
Logical. Whether to include gene names as labels in the figure. The default is highlight.genes.labels=FALSE.

highlight.genes.color
Character string or numeric value. A single color that will be used to highlight a region/gene. The default is highlight.genes.color=NULL.

highlight.genes.alpha
Numeric. A single value between 0 and 1 indicating the gene color transparency. The default is highlight.genes.alpha=0.1.

point.size
Numeric. The size of the points in the figures. The default is point.size=1.

point.color
A vector of characters or numeric values denoting the color of points to be plotted. If there are multiple subpopulations then the number of colors specified should equal the number of subpopulations. The default is point.color=NULL which will use isoRelate default colors.

add.rug
Logical. Whether to include SNP positions as a rug in the figure. The default is add.rug=FALSE

plot.title
A character string of a title to be added to the figure The default is plot.title=NULL which does not add a title to the plot.

add.legend
Logical. Whether a legend containing subpopulation information should be plotted. The default is add.legend=FALSE.

facet.label
Logical. Whether to include facet labels if multiple subpopulations (column name "subpop") are specified.

facet.scales
A character string of either "fixed", "free", "free_x" or "free_y" specifying the facet axis-scales. The default is facet.scales="fixed"

See Also

getIBDiR

Examples

```
# generate a binary IBD matrix
my_matrix <- getIBDmatrix(ped.genotypes = png_genotypes,
                          ibd.segments = png_ibd)

# calculate the significance of IBD sharing
my_iR <- getIBDiR(ped.genotypes = png_genotypes,
                  ibd.matrix = my_matrix,
                  groups = NULL)

# plot the iR statistics
plotIBDiR(ibd.iR = my_iR,
          interval = NULL,
          annotation.genes = NULL,
          annotation.genes.color = NULL,
          highlight.genes = NULL,
          highlight.genes.labels = FALSE,
          highlight.genes.color = NULL,
          highlight.genes.alpha = 0.1,
          point.size = 1,
          point.color = NULL,
          add.rug = FALSE,
          plot.title = "Significance of IBD sharing",
          add.legend = FALSE,
          facet.label = TRUE,
          facet.scales = "fixed")
```

plotIBDproportions *Plot The Proportion of Pairs IBD*

Description

plotIBDproportions() plots the proportion of pairs IBD for each SNP across the genome.

Usage

```
plotIBDproportions(ibd.proportions, interval = NULL,
  annotation.genes = NULL, annotation.genes.color = NULL,
  highlight.genes = NULL, highlight.genes.labels = TRUE,
  highlight.genes.color = NULL, highlight.genes.alpha = 0.1,
  line.color = NULL, add.rug = TRUE, plot.title = NULL,
  add.legend = TRUE, facet.label = TRUE, facet.scales = "fixed",
  subpop.facet = FALSE)
```

Arguments

ibd.proportions

A data frame containing the proportion of pairs IBD at each SNP. See the returned Value in getIBDproportion for more details. If multiple subpopulations are specified (column name "subpop") then the proportions for each subpopulation will be plotted, either in a single facet or over multiple facets. See http://docs.ggplot2.org/current/facet_grid.html on faceting. If multiple populations (column name "pop") are specified then the proportions for each population will be plotted on a separate facet, with all subpopulations in a single facet. If there are many populations or subpopulations (>8) it may be better to subset the populations to those of interest before plotting. Genomic locations of annotation genes can be included in the figure and specific genes or regions can be highlighted.

`interval` A vector of length 3 containing the genomic locations of a specific region to plot. This vector should contain the chromosome ID, the start of the interval in base-pairs and the end of the interval in base-pairs; in this order respectively. The default is `interval=NULL` which will plot the proportions over all chromosomes in `ibd.proportions`.

`annotation.genes`

A data frame containing information on annotation genes to be included in the figure. This data frame must have at least 5 columns of information:

1. Chromosome (type `"numeric"` or `"integer"`)
2. Gene name (type `"character"`)
3. Start location of the gene in base-pairs (type `"numeric"` or `"integer"`)
4. End location of the gene in base-pairs (type `"numeric"` or `"integer"`)
5. Gene strand (+ or -) (type `"character"`)

`annotation.genes` must contain the following headers `chr`, `name`, `start`, `end` and `strand`. This data frame does not have to be in a specific order, however it must contain all of the above information with respective labels. The default is annotation.genes=NULL.

`annotation.genes.color`

A vector of characters or numeric values containing the two colors representing gene stand (positive (+) or negative (-))

`highlight.genes`

A data frame containing information of genes or regions to highlight. The data frame must have at least 4 columns of information:

1. Chromosome (type `"numeric"` or `"integer"`)
2. Gene name (type `"character"`)
3. Start location of the gene in base-pairs (type `"numeric"` or `"integer"`)
4. End location of the gene in base-pairs (type `"numeric"` or `"integer"`)

`highlight.genes` should contain the following headers `chr`, `name`, `start` and `end`. This data frame does not have to be in a specific order, however it must contain all of the above information with respective labels. The default is highlight.genes=NULL.

`highlight.genes.labels`

Logical. Whether to include gene names as labels in the figure. The default is `highlight.genes.labels=FALSE`.

`highlight.genes.color`

Character string or numeric value. A single color that will be used to highlight a region/gene. The default is `highlight.genes.color=NULL`.

`highlight.genes.alpha`

Numeric. A single value between 0 and 1 indicating the gene color transparency. The default is `highlight.genes.alpha=0.1`.

`line.color` A vector of characters or numeric values denoting the color of lines to be plotted. If there are multiple populations/subpopulations then the number of colors specified should equal the number of unique populations/subpopulations combinations. The default is `line.color=NULL` which will use isoRelate default colors.

`add.rug` Logical. Whether to include SNP positions as a rug in the figure. The default is `add.rug=FALSE`

`plot.title` A character string of a title to be added to the figure The default is `plot.title=NULL` which does not add a title to the plot.

`add.legend` Logical. Whether a legend containing subpopulation information should be plotted. The default is `add.legend=FALSE`.

`facet.label` Logical. Whether to include facet labels if multiple populations/subpopulations (column names "pop" and "subpop") are specified.

`facet.scales` A character string of either `"fixed"`, `"free"`, `"free_x"` or `"free_y"` specifying the facet axis-scales. The default is `facet.scales="fixed"`

`subpop.facet` Logical. Whether to plot subpopulations in separate facets. The default is `subpop.facet=FALSE`. If `subpop.facet=TRUE` and there are multiple populations then subpopulations will **not** be drawn in separate facets.

See Also

getIBDproportion

Examples

```
# generate a binary IBD matrix
my_matrix <- getIBDmatrix(ped.genotypes = png_genotypes,
                          ibd.segments = png_ibd)

# calculate the proportion of pairs IBD at each SNP
my_proportion <- getIBDproportion(ped.genotypes = png_genotypes,
                                  ibd.matrix = my_matrix,
                                  groups = NULL)

# plot the proportion of pairs IBD
plotIBDproportions(ibd.proportions = my_proportion,
                   interval = NULL,
                   annotation.genes = NULL,
                   annotation.genes.color = NULL,
                   highlight.genes = NULL,
                   highlight.genes.labels = TRUE,
                   highlight.genes.color = NULL,
                   highlight.genes.alpha = 0.1,
                   add.rug = FALSE,
                   plot.title = "Proportion of pairs IBD in PNG",
                   add.legend = FALSE,
                   line.color = NULL,
                   facet.label = TRUE,
                   facet.scales = "fixed",
                   subpop.facet = FALSE)

# creating a stratification dataset
my_groups <- png_genotypes[[1]][,1:3]
my_groups[1:10,"pid"] <- "a"
my_groups[11:25,"pid"] <- "b"
```

```
my_groups[26:38,"pid"] <- "c"

my_proportion <- getIBDproportion(ped.genotypes = png_genotypes,
                                  ibd.matrix = my_matrix,
                                  groups = my_groups)

# plot the proportion of pairs IBD
plotIBDproportions(ibd.proportions = my_proportion,
                   interval = NULL,
                   annotation.genes = NULL,
                   annotation.genes.color = NULL,
                   highlight.genes = NULL,
                   highlight.genes.labels = FALSE,
                   highlight.genes.color = NULL,
                   highlight.genes.alpha = 0.1,
                   line.color = NULL,
                   add.rug = FALSE,
                   plot.title = "Proportion of pairs IBD in PNG - with stratification",
                   add.legend = FALSE,
                   facet.label = TRUE,
                   facet.scales = "fixed",
                   subpop.facet = TRUE)
```

plotIBDsegments *Plot IBD Segments*

Description

plotIBDsegments() plots IBD segments for pairs across the genome. IBD segments are depicted by colored blocks.

Usage

```
plotIBDsegments(ped.genotypes, ibd.segments, interval = NULL,
  annotation.genes = NULL, annotation.genes.color = NULL,
  highlight.genes = NULL, highlight.genes.labels = TRUE,
  highlight.genes.color = NULL, highlight.genes.alpha = 0.1,
  segment.height = 0.5, segment.color = NULL, number.per.page = NULL,
  fid.label = TRUE, iid.label = TRUE, ylabel.size = 9, add.rug = FALSE,
  plot.title = NULL, add.legend = TRUE)
```

Arguments

ped.genotypes	A list containing 3 objects. See the Value description in getGenotypes for more details on this input.
ibd.segments	A data frame containing the IBD segments inferred from pairs of isolates See the returned value in getIBDsegments for more details.

`interval` A vector of length 3 containing the genomic locations of a specific region to plot. This vector should contain the chromosome ID, the start of the interval in base-pairs and the end of the interval in base-pairs; in this order respectively. The default is `interval=NULL` which will plot the segments over all chromosomes.

`annotation.genes`

A data frame containing information on annotation genes to be included in the figure. This data frame must have at least 5 columns of information:

1. Chromosome (type `"numeric"` or `"integer"`)
2. Gene name (type `"character"`)
3. Start location of the gene in base-pairs (type `"numeric"` or `"integer"`)
4. End location of the gene in base-pairs (type `"numeric"` or `"integer"`)
5. Gene strand (+ or -) (type `"character"`)

`annotation.genes` must contain the following headers `chr`, `name`, `start`, `end` and `strand`. This data frame does not have to be in a specific order, however it must contain all of the above information with respective labels. The default is annotation.genes=NULL.

`annotation.genes.color`

A vector of characters or numeric values containing the two colors representing gene stand (positive (+) or negative (-))

`highlight.genes`

A data frame containing information of genes or regions to highlight. The data frame must have at least 4 columns of information:

1. Chromosome (type `"numeric"` or `"integer"`)
2. Gene name (type `"character"`)
3. Start location of the gene in base-pairs (type `"numeric"` or `"integer"`)
4. End location of the gene in base-pairs (type `"numeric"` or `"integer"`)

`highlight.genes` should contain the following headers `chr`, `name`, `start` and `end`. This data frame does not have to be in a specific order, however it must contain all of the above information with respective labels. The default is highlight.genes=NULL.

`highlight.genes.labels`

Logical. Whether to include gene names as labels in the figure. The default is `highlight.genes.labels=FALSE`.

`highlight.genes.color`

Character string or numeric value. A single color that will be used to highlight a region/gene. The default is `highlight.genes.color=NULL`.

`highlight.genes.alpha`

Numeric. A single value between 0 and 1 indicating the gene color transparency. The default is `highlight.genes.alpha=0.1`.

`segment.height` A numeric value giving the hight of IBD segment blocks, such that 0 < segment.height <= 1. The default is `segment.height=0.5`.

`segment.color` A vector of characters or numeric values denoting the color of the segments to be plotted. Two colors must be specified, one for segments with 1 allele IBD and one for segments with 2 alleles IBD.

`number.per.page`
A numeric value indicating the maximum number of IBD pairs to plot in a single graphics window. The default is `number.per.page=NULL` which will plot all IBD pairs in a single window. This may not be ideal when there are many IBD pairs. If `number.per.page` is set, it is recommended to plot the output to a file as opposed to the R plotting window.

`fid.label` Logical. If `fid.label=TRUE`, family IDs will be included in the y-axis labels; otherwise family IDs will be omitted. The default is `add.fid.name=TRUE`.

`iid.label` Logical. If `iid.label=TRUE`, isolate IDs will be included in the y-axis labels; otherwise isolate IDs will be omitted. The default is `add.iid.name=TRUE`.

`ylabel.size` A numeric value indicating the size of the y-axis labels if drawn. The default is `ylabel.size=9`.

`add.rug` Logical. Whether to include SNP positions as a rug in the figure. The default is `add.rug=FALSE`

`plot.title` A character string of a title to be added to the figure The default is `plot.title=NULL` which does not add a title to the plot.

`add.legend` Logical. If `add.legend=TRUE`, a legend specifying the IBD status (1 allele IBD or 2 alleles IBD) will be included. The default is `add.legend=TRUE`.

See Also

`getGenotypes` and `getIBDsegments`.

Examples

```
# plot IBD segments
plotIBDsegments(ped.genotypes = png_genotypes,
                ibd.segments = png_ibd,
                interval = NULL,
                annotation.genes = NULL,
                annotation.genes.color = NULL,
                highlight.genes = NULL,
                highlight.genes.labels = FALSE,
                highlight.genes.color = NULL,
                highlight.genes.alpha = 0.1,
                segment.height = 0.6,
                number.per.page = NULL,
                fid.label = FALSE,
                iid.label = FALSE,
                ylabel.size = 9,
                add.rug = FALSE,
                plot.title = "Distribution of IBD segments in PNG",
                add.legend = TRUE,
                segment.color = NULL)

# plot IBD segments over an interval: chromosome 7: 350000 - 550000
plotIBDsegments(ped.genotypes = png_genotypes,
                ibd.segments = png_ibd,
                interval = c("Pf3D7_07_v3",350000,550000),
                annotation.genes = annotation_genes,
                annotation.genes.color = NULL,
                highlight.genes = highlight_genes,
                highlight.genes.labels = FALSE,
                highlight.genes.color = NULL,
                highlight.genes.alpha = 0.1,
                segment.height = 0.8,
                number.per.page = NULL,
                fid.label = FALSE,
                iid.label = FALSE,
                ylabel.size = 9,
                add.rug = TRUE,
                plot.title = "Distribution of IBD segments in PNG",
                add.legend = TRUE,
                segment.color = c("purple","green"))
```

png_genotypes *Filtered Genotypes For The Papua New Guinea Dataset*

Description

Processed raw genotype data with the parameter settings as in the Vignette.

Usage

```
png_genotypes
```

Format

A list of two objects named `pedigree` and `genotypes`:

1. `pedigree` is a data frame with the following information:
   (a) Family ID
   (b) Isolate ID
   (c) Paternal ID. This is not used by isoRelate and is set to zero.
   (d) Maternal ID. This is not used by isoRelate and is set to zero.
   (e) Multiplicity of infection - 1 for single infection and 2 for multiple infection
   (f) Affection status of isolate. This is set to 2 and is ignored by isoRelate.
2. `genotypes` is a data frame with the first 5 columns:
   (a) Chromosome
   (b) SNP identifier
   (c) Genetic map distance
   (d) Base-pair position
   (e) Population allele frequency

   where each row describes a single SNP. Columns 6 onwards contain the genotype data for each isolate, where a single column corresponds to a single isolate. These columns are labeled with merged family IDs and isolate IDs separated by a slash symbol (/).

png_ibd *IBD Segments For The Papua New Guinea Dataset*

Description

The IBD segments inferred using isoRelate with the parameter settings as in the Vignette.

Usage

```
png_ibd
```

Format

A data frame with

fid1 Family 1 ID

iid1 Isolate 1 ID

fid2 Family 2 ID

iid2 Isolate 2 ID

chr Chromosome

start_snp IBD segment start SNP ID

end_snp IBD segment end SNP ID

start_position_bp IBD segment start SNP base-pair position

end_position_bp IBD segment end SNP base-pair position

start_position_M IBD segment start SNP morgan position

end_position_M IBD segment end SNP morgan position

number_snps Number of SNPs in IBD segment

length_bp Length of IBD segment in base-pairs

length_M Length of IBD segment in morgans

ibd_status The number of alleles shared IBD (either 1 or 2)

png_parameters *IBD Parameter Estimates For The Papua New Guinea Dataset*

Description

The parameter estimates inferred using isoRelate with the parameter settings as in the Vignette.

Usage

```
png_parameters
```

Format

A data frame with

fid1 Family 1 ID

iid1 Isolate 1 ID

fid2 Family 2 ID

iid2 Isolate 2 ID

m The estimated number of meiosis separating each pair of isolates

ibd0 The estimated proportion of genome with 0 alleles IBD

ibd1 The estimated proportion of genome with 1 allele IBD

ibd2 The estimated proportion of genome with 2 alleles IBD

png_pedmap *Papua New Guinea Plasmodium Falciparum Dataset*

Description

A PED/MAP file containing gneotype information for 38 isolates from Madang, Papua New Guinea. This dataset was released as part of the MalariaGEN Pf3k consortium in VCF file format, which underwent extensive data processing to result in the final SNP set in the PED/MAP file. This data, and more, is available from `https://www.malariagen.net/projects/pf3k`.

Usage

```
png_pedmap
```

Format

List with two objects

PED PLINK PED data frame. See `getGenotypes` for details.

MAP PLINK MAP data frame. See `getGenotypes` for details.

See `getGenotypes` for details.

roundDecimal *Round digits to specified decimal places*

Description

Round digits to specified decimal places

Usage

```
roundDecimal(number, digits)
```

Arguments

`number`	A number to round
`digits`	The number of digits to round to

transitionProbDD *The transition probabilities for 2 diploid chromosomes*

Description

The transition probabilities for 2 diploid chromosomes

Usage

```
transitionProbDD(omega_0, omega_1, omega_2, meiosis, dist_cM, ibd_current,
  ibd_previous)
```

Arguments

`omega_0`	The probability of sharing 0 alleles IBD
`omega_1`	The probability of sharing 1 allele IBD
`omega_2`	The probability of sharing 2 alleles IBD
`meiosis`	The number of meiosis separating the two isoaltes
`dist_cM`	The genetic map distance (cM) between SNP i and SNP j
`ibd_current`	The IBD state of SNP j
`ibd_previous`	The IBD state of SNP i

transitionProbHD *The transition probabilities for 1 haploid and 1 diploid chromosome*

Description

The transition probabilities for 1 haploid and 1 diploid chromosome

Usage

```
transitionProbHD(omega_0, meiosis, dist_cM, ibd_current, ibd_previous)
```

Arguments

`omega_0`	The probability of sharing 0 alleles IBD
`meiosis`	The number of meiosis separating the two isoaltes
`dist_cM`	The genetic map distance (cM) between SNP i and SNP j
`ibd_current`	The IBD state of SNP j
`ibd_previous`	The IBD state of SNP i

transitionProbHH *The transition probabilities for 2 haploid chromosomes*

Description

The transition probabilities for 2 haploid chromosomes

Usage

```
transitionProbHH(omega_0, meiosis, dist_cM, ibd_current, ibd_previous)
```

Arguments

`omega_0`	The probability of sharing 0 alleles IBD
`meiosis`	The number of meiosis separating the two isoaltes
`dist_cM`	The genetic map distance (cM) between SNP i and SNP j
`ibd_current`	The IBD state of SNP j
`ibd_previous`	The IBD state of SNP i

trueGenotypes *Matrices of all possible genotype combinations between pairs, given MOI*

Description

Matrices of all possible genotype combinations between pairs, given MOI

Usage

```
trueGenotypes(gender_1, gender_2)
```

Arguments

`gender_1`	The MOI estimate of isolate 1
`gender_2`	The MOI estimate of isolate 2

Index