Package 'sim1000G'

February 15, 2018

Type Package

Title Genotype Simulations for Rare or Common Variants Using Haplotypes from 1000 Genomes

Version 1.38

Date 2018-02-13

Author Apostolos Dimitromanolakis <apostolis@live.ca>, Jingxiong Xu <jingxiong.xu@mail.utoronto.ca>, Agnieszka Krol <krol@lunenfeld.ca>, Laurent Briollais <laurent@lunenfeld.ca>

Maintainer Apostolos Dimitromanolakis <apostolis@live.ca>

Description Generates realistic simulated genetic data in families or unrelated individuals.

License GPL (>= 2)

Depends R (>= 2.15.2), stats, hapsim, MASS, stringr, readr

NeedsCompilation no

VignetteBuilder knitr, prettydoc

RoxygenNote 6.0.1

Suggests knitr, prettydoc, testthat, rmarkdown, gplots

Encoding UTF-8

Repository CRAN

R topics documented:

`sim1000G-package` *Simulations of rare/common variants using haplotype data from 1000 genomes*

Description

Documentation and examples can be found at the package directory folder inst / doc or at our github url: https://adimitromanolakis.github.io/sim1000G/ inst/doc/SimulatingFamilyData.html

Details

See also our github repository page at: https://github.com/adimitromanolakis/sim1000G

computePairIBD1 *Computes pairwise IBD1 for a specific pair of individuals. See function computePairIBD12 for description.*

Description

Computes pairwise IBD1 for a specific pair of individuals. See function computePairIBD12 for description.

Usage

```
computePairIBD1(i, j)
```

Arguments

i	Index of first individual
j	Index of second individual

Value

Mean IBD1 as computed from shared haplotypes

Examples

```
library("sim1000G")

examples_dir = system.file("examples", package = "sim1000G")
vcf_file = file.path(examples_dir, "region.vcf.gz")
vcf = readVCF( vcf_file, maxNumberOfVariants = 100 ,
               min_maf = 0.12 ,max_maf = NA)

# For realistic data use the function downloadGeneticMap
generateUniformGeneticMap()

startSimulation(vcf, totalNumberOfIndividuals = 200)

ped1 = newNuclearFamily(1)

v = computePairIBD1(1, 3)

cat("IBD1 of pair = ", v, "\n");
```

`computePairIBD12` *Computes pairwise IBD1/2 for a specific pair of individuals*

Description

Computes pairwise IBD1/2 for a specific pair of individuals

Usage

```
computePairIBD12(i, j)
```

Arguments

`i`	Index of first individual
`j`	Index of second individual

Value

Mean IBD1 and IBD2 as computed from shared haplotypes

Examples

```
library("sim1000G")

examples_dir = system.file("examples", package = "sim1000G")
vcf_file = file.path(examples_dir, "region.vcf.gz")

vcf = readVCF( vcf_file, maxNumberOfVariants = 100 ,
               min_maf = 0.12 ,max_maf = NA)

generateUniformGeneticMap()

startSimulation(vcf, totalNumberOfIndividuals = 200)

ped1 = newNuclearFamily(1)

v = computePairIBD12(1, 3)

cat("IBD1 of pair = ", v[1], "\n");
cat("IBD2 of pair = ", v[2], "\n");
```

computePairIBD2 *Computes pairwise IBD2 for a specific pair of individuals*

Description

Computes pairwise IBD2 for a specific pair of individuals

Usage

```
computePairIBD2(i, j)
```

Arguments

i	Index of first individual
j	Index of second individual

Value

Mean IBD2 as computed from shared haplotypes

Examples

```
library("sim1000G")

examples_dir = system.file("examples", package = "sim1000G")
vcf_file = file.path(examples_dir, "region.vcf.gz")
vcf = readVCF( vcf_file, maxNumberOfVariants = 100 ,
               min_maf = 0.12 ,max_maf = NA)

# For realistic data use the function downloadGeneticMap
generateUniformGeneticMap()

startSimulation(vcf, totalNumberOfIndividuals = 200)

ped1 = newNuclearFamily(1)

v = computePairIBD2(1, 3)

cat("IBD2 of pair = ", v, "\n");
```

`createVCF` *Creates a regional vcf file using bcftools to extract a region from 1000 genomes vcf files*

Description

Creates a regional vcf file using bcftools to extract a region from 1000 genomes vcf files

Usage

```
createVCF()
```

Value

none

`crossoverCDFvector` *Contains recombination model information.*

Description

This vector contains the density between two recombination events, as a cumulative density function.

Usage

```
crossoverCDFvector
```

Format

An object of class `logical` of length 1.

`downloadGeneticMap` *Downloads a genetic map for a particular chromosome under GRCh37 coordinates for use with sim1000G.*

Description

Downloads a genetic map for a particular chromosome under GRCh37 coordinates for use with sim1000G.

Usage

```
downloadGeneticMap(chromosome, dir = NA)
```

Arguments

`chromosome`	Chromosome number to download recombination distances from.
`dir`	Directory to save the genetic map to (default: extdata)

Examples

```
downloadGeneticMap(22, dir=tempdir() )
```

`generateChromosomeRecombinationPositions` *Generates a recombination vector arising from one meiotic event. The origin of segments is coded as (0 - haplotype1 , 1 - haplotype2)*

Description

Generates a recombination vector arising from one meiotic event. The origin of segments is coded as (0 - haplotype1 , 1 - haplotype2)

Usage

```
generateChromosomeRecombinationPositions(chromosomeLength = 500)
```

Arguments

`chromosomeLength`	The length of the region in cm.

Examples

```
library("sim1000G")

# generate a recombination events for chromosome 4
readGeneticMap(4)
generateChromosomeRecombinationPositions(500)
```

`generateFakeWholeGenomeGeneticMap` *Generates a fake genetic map that spans the whole genome.*

Description

Generates a fake genetic map that spans the whole genome.

Usage

```
generateFakeWholeGenomeGeneticMap(vcf)
```

Arguments

`vcf`	A vcf file read by function readVCF.

Examples

```
library("sim1000G")

examples_dir = system.file("examples", package = "sim1000G")
vcf_file = sprintf("%s/region.vcf.gz", examples_dir)
vcf = readVCF( vcf_file, maxNumberOfVariants = 100 ,
               min_maf = 0.12 ,max_maf = NA)

# For realistic data use the function
# downloadGeneticMap
generateFakeWholeGenomeGeneticMap(vcf)

pdf(file=tempfile())
plotRegionalGeneticMap(seq(1e6,100e6,by=1e6/2))
dev.off()
```

`generateRecombinationDistances`
Generate inter-recombination distances using a chi-square model. Note this are the distances between two succesive recombination events and not the absolute positions of the events. To generate the locations of the recombination events see the example below.

Description

Generate inter-recombination distances using a chi-square model. Note this are the distances between two succesive recombination events and not the absolute positions of the events. To generate the locations of the recombination events see the example below.

Usage

```
generateRecombinationDistances(n)
```

Arguments

`n`	Number of distances to generate

Value

vector of distances between two recombination events.

Examples

```
library("sim1000G")

distances = generateRecombinationDistances(20)


positions_of_recombination = cumsum(distances)

if(0) hist(generateRecombinationDistances(20000),n=100)
```

`generateRecombinationDistances_noInterference`
Generate recombination distances using a no-interference model.

Description

Generate recombination distances using a no-interference model.

Usage

```
generateRecombinationDistances_noInterference(n)
```

Arguments

n	Number of distances to generate

Value

recombination distances in centimorgan

Examples

```
library("sim1000G")
mean ( generateRecombinationDistances_noInterference ( 200 ) )
```

---

generateSingleRecombinationVector

Genetates a recombination vector arising from one meiotic event. The origin of segments is coded as (0 - haplotype1 , 1 - haplotype2)

Description

Genetates a recombination vector arising from one meiotic event. The origin of segments is coded as (0 - haplotype1 , 1 - haplotype2)

Usage

```
generateSingleRecombinationVector(cm)
```

Arguments

cm	The length of the region that we want to generate recombination distances.

Examples

```
library("sim1000G")

examples_dir = system.file("examples", package = "sim1000G")
vcf_file = file.path(examples_dir, "region.vcf.gz")
vcf = readVCF( vcf_file, maxNumberOfVariants = 100 ,
               min_maf = 0.12 ,max_maf = NA)

# For realistic data use the function downloadGeneticMap
generateUniformGeneticMap()
generateSingleRecombinationVector( 1:100 )
```

generateUniformGeneticMap

Generates a uniform genetic map.

Description

Generates a uniform genetic map by approximating 1 cm / Mbp. Only used for examples.

Usage

```
generateUniformGeneticMap()
```

Examples

```
library("sim1000G")

examples_dir = system.file("examples", package = "sim1000G")
vcf_file = sprintf("%s/region.vcf.gz", examples_dir)
vcf = readVCF( vcf_file, maxNumberOfVariants = 100 ,
               min_maf = 0.12 ,max_maf = NA)

# For realistic data use the function readGeneticMap
generateUniformGeneticMap()

pdf(file=tempfile())
plotRegionalGeneticMap(seq(1e6,100e6,by=1e6/2))
dev.off()
```

generateUnrelatedIndividuals

Generates variant data for n unrelated individuals

Description

Generates variant data for n unrelated individuals

Usage

```
generateUnrelatedIndividuals(N = 1)
```

Arguments

N	how many individuals to generate

Value

IDs of the generated individuals

Examples

```
library("sim1000G")

examples_dir = system.file("examples", package = "sim1000G")
vcf_file = file.path(examples_dir, "region.vcf.gz")
vcf = readVCF( vcf_file, maxNumberOfVariants = 100 , min_maf = 0.12)

genetic_map_of_region =
   system.file("examples",
     "chr4-geneticmap.txt",
     package = "sim1000G")

readGeneticMapFromFile(genetic_map_of_region)

startSimulation(vcf, totalNumberOfIndividuals = 1200)
ids = generateUnrelatedIndividuals(20)

# See also the documentation on our github page
```

---

geneticMap *Holds the genetic map information that is used for simulations.*

Description

Holds the genetic map information that is used for simulations.

Usage

```
geneticMap
```

Format

An object of class `environment` of length 0.

`getCMfromBP` *Converts centimorgan position to base-pair. Return a list of centimorgan positions that correspond to the bp vector (in basepairs).*

Description

Converts centimorgan position to base-pair. Return a list of centimorgan positions that correspond to the bp vector (in basepairs).

Usage

```
getCMfromBP(bp)
```

Arguments

`bp`	vector of base-pair positions

Examples

```
library("sim1000G")

examples_dir = system.file("examples", package = "sim1000G")
vcf_file = sprintf("%s/region.vcf.gz", examples_dir)
vcf = readVCF( vcf_file, maxNumberOfVariants = 100,
  min_maf = 0.12)

# For realistic data use the function downloadGeneticMap
generateUniformGeneticMap()
getCMfromBP(seq(1e6,100e6,by=1e6))
```

`loadSimulation` *Load some previously saved simulation data by function saveSimulation*

Description

Load some previously saved simulation data by function saveSimulation

Usage

```
loadSimulation(id)
```

Arguments

`id`	Name the simulation to load which was previously saved by saveSimulation

Examples

```
examples_dir = system.file("examples", package = "sim1000G")
vcf_file = file.path(examples_dir, "region.vcf.gz")

vcf = readVCF( vcf_file, maxNumberOfVariants = 100 ,
          min_maf = 0.12 ,max_maf = NA)

# For a realistic genetic map
# use the function readGeneticMap
generateUniformGeneticMap()

startSimulation(vcf, totalNumberOfIndividuals = 200)

ped1 = newNuclearFamily(1)

saveSimulation("sim1")

vcf = readVCF( vcf_file, maxNumberOfVariants = 100 ,
               min_maf = 0.02 ,max_maf = 0.5)

startSimulation(vcf, totalNumberOfIndividuals = 200)
saveSimulation("sim2")

loadSimulation("sim1")
```

---

newFamily3generations

Generates genotype data for a family of 3 generations

Description

Generates genotype data for a family of 3 generations

Usage

```
newFamily3generations(familyid, noffspring2 = 2, noffspring3 = c(1, 1))
```

Arguments

familyid	What will be the family_id (for example: 100)
noffspring2	Number of offspring in generation 2
noffspring3	Number of offspring in generation 3 (vector of length noffspring2)

Value

family structure object

Examples

```
library("sim1000G")

examples_dir = system.file("examples", package = "sim1000G")
vcf_file = file.path(examples_dir, "region.vcf.gz")
vcf = readVCF( vcf_file, maxNumberOfVariants = 100 ,
               min_maf = 0.12 ,max_maf = NA)

generateUniformGeneticMap()

startSimulation(vcf, totalNumberOfIndividuals = 200)

ped_line = newFamily3generations(12, 3, c(3,3,2) )
```

---

newFamilyWithOffspring

Simulates genotypes for 1 family with n offspring

Description

Simulates genotypes for 1 family with n offspring

Usage

```
newFamilyWithOffspring(family_id, noffspring = 2)
```

Arguments

family_id	What will be the family_id (for example: 100)
noffspring	Number of offsprings that this family will have

Value

family structure object

Examples

```
ped_line = newFamilyWithOffspring(10,3)
```

`newNuclearFamily` *Simulates genotypes for 1 family with 1 offspring*

Description

Simulates genotypes for 1 family with 1 offspring

Usage

```
newNuclearFamily(family_id)
```

Arguments

`family_id`	What will be the family_id (for example: 100)

Value

family structure object

Examples

```
library("sim1000G")

examples_dir = system.file("examples", package = "sim1000G")
vcf_file = file.path(examples_dir, "region.vcf.gz")
vcf = readVCF( vcf_file, maxNumberOfVariants = 100 ,
   min_maf = 0.12 ,max_maf = NA)

genetic_map_of_region = system.file("examples","chr4-geneticmap.txt",
   package = "sim1000G")
readGeneticMapFromFile(genetic_map_of_region)

startSimulation(vcf, totalNumberOfIndividuals = 1200)
fam1 = newNuclearFamily(1)
fam2 = newNuclearFamily(2)

# See also the documentation on our github page
```

pkg.opts *Holds general package options*

Description

Holds general package options

Usage

```
pkg.opts
```

Format

An object of class `environment` of length 1.

plotRegionalGeneticMap *Generates a plot of the genetic map for a specified region.*

Description

The plot shows the centimorgan vs base-pair positions. The position of markers that have been read is also depicted as vertical lines

Usage

```
plotRegionalGeneticMap(bp)
```

Arguments

bp	Vector of base-pair positions to generate a plot for library("sim1000G") examples_dir = system.file("examples", package = "sim1000G") vcf_file = sprintf(" vcf = readVCF(vcf_file, maxNumberOfVariants = 100, min_maf = 0.12) # For realistic data use the function readGeneticMap generateUniformGenetic-Map() pdf(file=tempfile()) plotRegionalGeneticMap(seq(1e6,100e6,by=1e6/2)) dev.off()

`printMatrix` *Utility function that prints a matrix. Useful for IBD12 matrices.*

Description

Utility function that prints a matrix. Useful for IBD12 matrices.

Usage

```
printMatrix(m)
```

Arguments

`m`	Matrix to be printed

Examples

```
printMatrix (  matrix(runif(16), nrow=4) )
```

`readGeneticMap` *Reads a genetic map downloaded from the function downloadGeneticMap or reads a genetic map from a specified file. If the argument filename is used then the genetic map is read from the corresponding file. Otherwise, if a chromosome is specified, the genetic map is downloaded for human chromosome using grch37 coordinates.*

Description

The map must contains a complete chromosome or enough markers to cover the area that will be simulated.

Usage

```
readGeneticMap(chromosome, filename = NA, dir = NA)
```

Arguments

`chromosome`	Chromosome number to download a genetic map for , or
`filename`	A filename of an existing genetic map to read from (default NA).
`dir`	Directory the map file will be saved (only if chromosome is specified).

Examples

```
readGeneticMap(chromosome = 22)
```

`readGeneticMapFromFile`

Reads a genetic map to be used for simulations. The genetic map should be of a single chromosome and covering the extent of the region to be simulated. Whole chromosome genetic maps can also be used.

Description

The file must be contain the following columns in the same order: chromosome, basepaire, rate(not used), centimorgan

Usage

```
readGeneticMapFromFile(filelocation)
```

Arguments

`filelocation` Filename containing the genetic map

Examples

```
## Not run:

fname = downloadGeneticMap(10)

cat("genetic map downloaded at :", fname, "\n")
readGeneticMapFromFile(fname)


## End(Not run)
```

`readVCF` *Read a vcf file, with options to filter out low or high frequency markers.*

Description

Read a vcf file, with options to filter out low or high frequency markers.

Usage

```
readVCF(filename = "data.vcf", thin = NA, maxNumberOfVariants = 400,
  min_maf = 0.02, max_maf = NA, region_start = NA, region_end = NA)
```

Arguments

`filename`	Input VCF file
`thin`	How much to thin markers
`maxNumberOfVariants`	Maximum number of variants to keep from region
`min_maf`	Minimum allele frequency of markers to keep. If NA skip min_maf filtering.
`max_maf`	Maximum allele frequency of markers to keep. If NA skip max_maf filtering.
`region_start`	Extract a region from a vcf files with this starting basepair position
`region_end`	Extract a region from a vcf files with this ending basepair position

Value

VCF object to be used by startSimulation function.

Examples

```
examples_dir = system.file("examples", package = "sim1000G")
vcf_file = file.path(examples_dir,
  "region-chr4-93-TMEM156.vcf.gz")

vcf = readVCF( vcf_file, maxNumberOfVariants = 500 ,
               min_maf = 0.02 ,max_maf = NA)

str(as.list(vcf))
```

`resetSimulation` *Removes all individuals that have been simulated and resets the simulator.*

Description

Removes all individuals that have been simulated and resets the simulator.

Usage

```
resetSimulation()
```

Value

nothing

Examples

```
resetSimulation()
```

`retrieveGenotypes` *Retrieve a matrix of simulated genotypes for a specific set of individual IDs*

Description

Retrieve a matrix of simulated genotypes for a specific set of individual IDs

Usage

```
retrieveGenotypes(ids)
```

Arguments

`ids`	Vector of ids of individuals to retrieve.

Examples

```
library("sim1000G")

examples_dir = system.file("examples", package = "sim1000G")
vcf_file = file.path(examples_dir, "region.vcf.gz")
vcf = readVCF( vcf_file, maxNumberOfVariants = 100 ,
               min_maf = 0.12 ,max_maf = NA)

# For realistic data use the function downloadGeneticMap
generateUniformGeneticMap()

startSimulation(vcf, totalNumberOfIndividuals = 200)

ped1 = newNuclearFamily(1)

retrieveGenotypes(ped1$gtindex)
```

`saveSimulation` *Save the data for a simulation for later use. When simulating multiple populations it allows saving and restoring of simulation data for each population.*

Description

Save the data for a simulation for later use. When simulating multiple populations it allows saving and restoring of simulation data for each population.

Usage

```
saveSimulation(id)
```

Arguments

`id`	Name the simulation will be saved as.

Examples

```
examples_dir = system.file("examples", package = "sim1000G")

vcf_file = file.path(examples_dir, "region.vcf.gz")
vcf = readVCF( vcf_file, maxNumberOfVariants = 100 ,
               min_maf = 0.12 ,max_maf = NA)
```

```
# For realistic data use the functions downloadGeneticMap
generateUniformGeneticMap()

startSimulation(vcf, totalNumberOfIndividuals = 200)

ped1 = newNuclearFamily(1)

saveSimulation("sim1")
```

`setRecombinationModel` *Set recombination model to either poisson (no interference) or chi-square.*

Description

Set recombination model to either poisson (no interference) or chi-square.

Usage

```
setRecombinationModel(model)
```

Arguments

`model`	Either poisson or chisq

Examples

```
generateUniformGeneticMap()

do_plots = 0

setRecombinationModel("chisq")
if(do_plots == 1)
 hist(generateRecombinationDistances(100000),n=200)

setRecombinationModel("poisson")
if(do_plots == 1)
 hist(generateRecombinationDistances(100000),n=200)
```

SIM — *Holds data necessary for a simulation.*

Description

Holds data necessary for a simulation.

Usage

```
SIM
```

Format

An object of class `environment` of length 7.

startSimulation — *Starts and initializes the data structures required for a simulation. A VCF file should be read beforehand with the function readVCF.*

Description

Starts and initializes the data structures required for a simulation. A VCF file should be read beforehand with the function readVCF.

Usage

```
startSimulation(vcf, totalNumberOfIndividuals = 2000, subset = NA,
  randomdata = 0, typeOfGeneticMap = "download")
```

Arguments

`vcf`	Input vcf file of a region (can be .gz). Must contain phased data.
`totalNumberOfIndividuals`	Maximum Number of individuals to allocate memory for. Set it above the number of individuals you want to simulate.
`subset`	A subset of individual IDs to use for simulation
`randomdata`	If 1, disregards the genotypes in the vcf file and generates independent markers that are not in LD.
`typeOfGeneticMap`	Specify whether to download a genetic map for this chromosome

Examples

```
library("sim1000G")
library(gplots)

examples_dir = system.file("examples", package = "sim1000G")
vcf_file = file.path(examples_dir, "region.vcf.gz")

vcf = readVCF( vcf_file, maxNumberOfVariants = 100)


genetic_map_of_region = system.file(
   "examples",
   "chr4-geneticmap.txt",
   package = "sim1000G"
)

readGeneticMapFromFile(genetic_map_of_region)

pdf(file=tempfile())
plotRegionalGeneticMap(vcf$vcf[,2]+1)
dev.off()

startSimulation(vcf, totalNumberOfIndividuals = 200)
```

`subsetVCF` *Generate a market subset of a vcf file*

Description

Generate a market subset of a vcf file

Usage

```
subsetVCF(vcf, var_index = NA, var_id = NA, individual_id = NA)
```

Arguments

`vcf`	VCF data as created by function readVCF
`var_index`	index of number to subset. Should be in the range 1..length(vcf$varid)
`var_id`	id of markers to subset. Should be a selection from vcf$varid. NA if no filtering on id to be performed.
`individual_id`	IDs of individuals to subset. Should be a selection from vcf$individual_id

Value

VCF object to be used by startSimulation function.

Examples

```
examples_dir = system.file("examples", package = "sim1000G")

vcf_file = file.path(examples_dir, "region-chr4-93-TMEM156.vcf.gz")

vcf = readVCF( vcf_file, maxNumberOfVariants = 500 ,
               min_maf = 0.02 ,max_maf = NA)

vcf2 = subsetVCF(vcf, var_index = 1:50)
```

`writePED` *Writes a plink compatible PED/MAP file from the simulated genotypes*

Description

Writes a plink compatible PED/MAP file from the simulated genotypes

Usage

```
writePED(vcf, fam, filename = "out")
```

Arguments

`vcf`	vcf object used in simulation
`fam`	Individuals / families to be written
`filename`	Basename of output files (.ped/.map will be added automatically)