Package ‘SimRVSequences’

March 8, 2020

Type Package
Title Simulate Genetic Sequence Data for Pedigrees
Version 0.2.4
Maintainer Christina Nieuwoudt <cnieuwou@sfu.ca>
Depends R (>= 3.5.0)
Imports SimRVPedigree (>= 0.1.0), kinship2 (>= 1.6.4), intervals (>= 0.15.1), reshape2 (>= 1.4.1), stats (>= 3.3.0), Matrix (>= 1.2-12), dplyr (>= 0.7.5), magrittr (>= 1.5), rlang (>= 0.2.0), methods (>= 3.5.0)
Suggests knitr (>= 1.13), rmarkdown (>= 0.9.6), vcfR (>= 1.8.0), pryr (>= 0.1.4), testthat
License GNU General Public License
Encoding UTF-8
LazyData true
RoxygenNote 6.1.1
VignetteBuilder knitr
NeedsCompilation no
Author Christina Nieuwoudt [aut, cre], Jinko Graham [aut]
Repository CRAN
Date/Publication 2020-03-07 23:20:05 UTC

R topics documented:

combine_exons ..................................................... 2
create_slimMap .................................................... 3
### Description
Combine overlapping exons into a single observation

### Usage
```r
combine_exons(exon_data)
```

### Arguments
- **exon_data** `data.frame`. This data frame must include named variables: `chrom`, a chromosome identifier; `exonStart`, the first position of the exon in base pairs; and `exonEnd`, the last position of the exon in base pairs.

### Value
A data frame of combined exon segments. This data frame includes the variables: `chrom`, a chromosome identifier; `exonStart`, the first position of the combined exon segment in base pairs; and `exonEnd`, the last position of the combined exon segment in base pairs.

### Examples
```r
# create a data frame that contains the
# the variables: chrom, exonStart, and exonEnd
exDat <- data.frame(chrom = c(1, 1, 1, 2, 2, 2),
                    exonStart = c(1, 2, 5, 1, 3, 3),
                    exonEnd   = c(3, 4, 7, 4, 5, 6))

exDat

# supply exDat to combine_exons to combine
# overlapping exon segments
create_slimMap

create_slimMap(exon_df, mutation_rate = 1e-08, recomb_rate = 1e-08)

Arguments

exon_df Data frame. A data frame that contains the positions of each exon to simulate. This data frame must contain the variables chrom, exonStart, and exonEnd. See details.

mutation_rate Numeric. The per-site per-generation mutation rate, assumed to be constant across the genome. By default, mutation_rate = 1E-8, as in Harris and Nielson (2016).

recomb_rate Numeric. The per-site per-generation mutation rate, assumed to be constant across the genome. By default, mutation_rate = 1E-8, as in Harris and Nielson (2016).

Details

The Eidos program SLiM (Haller and Messer 2017) is a versatile forwards-in-time evolutionary simulator. SLiM simulates recombination hotspots by way of a user-specified recombination map. This recombination map may be utilized to simulate mutations over unlinked regions (i.e. in different chromosomes) or in linked but non-contiguous regions (i.e. in exon-only data). The create_slimMap function may be used to generate the recombination map required by SLiM to simulate exon-only SNV data.

We expect that exon_df does not contain any overlapping segments. Prior to supplying the exon data to create_slimMap users must combine overlapping exons into a single observation. The combine_exons function may be used to accomplish this task.

The argument exon_df must contain the following variables:

<table>
<thead>
<tr>
<th>name</th>
<th>type</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>chrom</td>
<td>numeric</td>
<td>chromosome identification number</td>
</tr>
<tr>
<td>exonStart</td>
<td>numeric</td>
<td>the position of the first base pair in the exon</td>
</tr>
<tr>
<td>exonStop</td>
<td>numeric</td>
<td>the position of the last base pair in the exon</td>
</tr>
</tbody>
</table>

The data frame returned by create_slimMap contains variables required by SLiM to simulate exon-
only data. Additionally, the returned data frame also includes variables that are required to re-map mutations to their correct positions when importing SLiM data to R. The variables contained in the returned data frame are described as follows.

chrom The chromosome number.

segLength The length of the segment in base pairs. We assume that segments contain the positions listed in exonStart and exonEnd. Therefore, for a combined exon segment, segLength is calculated as exonEnd - exonStart + 1.

recRate The per-site per-generation recombination rate. Following Harris and Nielsen (2016), segments between exons on the same chromosome are simulated as a single base pair with rec_rate equal to recombination rate multiplied by the number of base pairs in the segment. For each chromosome, a single site is created between the last exon on the previous chromosome and the first exon of the current chromosome. This site will have recombination rate 0.5 to accommodate unlinked chromosomes.

mutRate The per-site per-generation mutation rate. Since we are interested in exon-only data, the mutation rate outside exons is set to zero.

exon A logical variable that is TRUE if the segment is an exon and FALSE otherwise.

simDist The simulated exon length, in base pairs. When exon = TRUE, simDist = segLength; however, when exon = FALSE, simDist = 1 since segments between exons on the same chromosome are simulated as a single base pair.

endPos The simulated end position, in base pairs, of the segment.

Only three of the variables returned by create_slimMap are required by SLiM to simulate exon-only data: recRate, mutRate, and endPos. The other variables seen in the output above are used by the read_slim function to re-map mutations to their correct positions when importing SLiM data to R.

Please note: SLiM is written in a scripting language called Eidos. Unlike an R array, the first position in an Eidos array is 0. Therefore, users must shift the variable endPos forward 1 unit before supplying this variable to SLiM. See example.

Value

A recombination map that may be used in conjunction with SLiM (Haller and Messer 2017). See details and example.

References


See Also

combine_exons
**Examples**

```r
#load hg_exons data
data(hg_exons)

#since the exons in hg_exons have already been combined into
#overlapping exons, we supply hg_exons to create_slimMap
slimMap <- create_slimMap(hg_exons)
head(slimMap)

# restrict output to the variables required by SLiM
slimMap <- slimMap[, c("recRate", "mutRate", "endPos")]

# shift endPos up by one unit
slimMap$endPos <- slimMap$endPos - 1

# print first four rows of slimMap
head(slimMap, n = 4)
```

---

**Description**

This data set contains 20,000 haplotypes (i.e. rows) spanning 500 single-nucleotide variants (SNVs). This dataset is intended to accompany the EXmuts dataset; each row of EXmuts describes a column (i.e. SNV) in EXhaps.

**Usage**

EXhaps

**Format**

A sparseMatrix of class dgCMatrix with 20000 rows and 500 variables. Each row represents an observed haplotype and each column represents an SNV locus.

**Details**

This dataset is intended to accompany the EXmuts dataset. Together, the EXmuts and EXhaps datasets represent example output of the SNVdata object returned by the read_slim function. The EXhaps data set represents the sparse matrix Haplotypes returned by read_slim, and the EXmuts data set represents the Mutations data frame returned by read_slim. This toy data set, used primarily for demonstration, contains 50 SNVs which were randomly sampled from genes in the apoptosis sub-pathway, and 450 SNVs sampled from outside the pathway.

**See Also**

EXmuts, read_slim
**EXmuts**

*Example Mutations dataset*

### Description

This data set catalogs the 500 single-nucleotide variants contained in the EXhaps dataset. This dataset is intended to accompany the EXhaps dataset; each row of EXmuts describes a column (i.e. SNV) in EXhaps.

### Usage

EXmuts

### Format

A data set with 500 rows and 6 variables:

- **colID** Numeric. The corresponding column number of the SVN in the EXhaps dataset.
- **chrom** Numeric. The chromosome number.
- **position** Numeric. The location of the SNV on the chromosome, in base pairs.
- **afreq** Numeric. The derived allele frequency of the SNV.
- **marker** Character. The names of the genes contained in the combined exon.
- **pathwaySNV** Logical. Indicates if the SNV is located within the pathway.

### Details

Together, the EXmuts and EXhaps datasets represent example output of the SNVdata object returned by the read_slim function. The EXhaps data set represents the sparse matrix Haplotypes returned by read_slim, and the EXmuts data set represents the Mutations data frame returned by read_slim. This toy data set, used primarily for demonstration, contains 50 SNVs which were randomly sampled from genes in the apoptosis sub-pathway, and 450 SNVs sampled from outside the pathway.

### See Also

EXhaps, read_slim
**genos2sparseMatrix**  
*Convert genotypes to haplotypes.*

**Description**

This function may be used to convert phased genotype data for diploid organisms into a sparse matrix.

**Usage**

```r
genos2sparseMatrix(genotypes)
```

**Arguments**

- `genotypes`: A dataframe or matrix of genotypes. The columns of genotypes are assumed to be individuals (i.e. a diploid human) and the rows are assumed to be mutations. See details.

**Details**

The columns of `genotypes` are assumed to be individuals (i.e. a diploid human) and the rows are assumed to be mutations. Thus, the (i,j)th entry of `genotypes` is the genotype of the jth person at the ith SNV site. Please note that genotypes should not contain missing values. Additionally, genotypes may take one of the following three forms:

- "000" if the individual is homozygous for the reference allele,
- "001" or "011" if the individual is heterozygous for the alternate allele,
- "111" if the individual is homozygous for the alternate allele.

**Value**

A sparseMatrix. Note that the rows and columns of the returned matrix have been transposed so that individual haplotypes are rows, and each column represents an SNV.

---

**hg_apopPath**  
*Apoptosis sub-pathway dataset*

**Description**

This data set catalogs combined exon segments from the 25 genes that have the highest interaction with the `TNFSF10` gene, a known member of the human apoptosis pathway.

**Usage**

```r
gg_apopPath
```
Format

A data set with 253 rows and 5 variables:

- `chrom`: Numeric. The chromosome number.
- `exonStart`: Numeric. The position of the first base pair.
- `exonStop`: Numeric. The position of the last base pair.
- `NCBIref`: Character. The NCBI reference sequence accession number of the gene(s) in which the exon(s) reside.
- `gene`: Character. The name of the gene.

Details

The `hg_exons` data set catalogs combined exon segments from the 22 human autosomes.

References


---

**hg_exons**

*Human exon data*

Description

This data set catalogs combined exon segments from the 22 human autosomes.

Usage

`hg_exons`
load_1KG

Format
A data set with 223565 rows and 4 variables:

- **chrom** Numeric. The chromosome number.
- **exonStart** Numeric. The position of the first base pair in the combined exon segment.
- **exonStop** Numeric. The position of the last base pair in the combined exon segment.
- **NCBIref** Character. The NCBI reference sequence accession number of the gene(s) in which the exon(s) reside.

Details
The hg_exons data set catalogs the positions of exons residing in the 22 human autosomes. The data contained in hg_exons was collected from the hg 38 reference genome with the UCSC Genome Browser’s Table Browser Tool. In hg_exons overlapping exons have been combined into a single observation. When exons from genes with different NCBI accession numbers have been combined the variable NCBIref will contain multiple accession numbers separated by commas. We note that different accession numbers may exist for transcript variants of the same gene.

References


load_1KG

Description
Load pre-formatted 1000 Genomes Project exon data

Usage

```r
load_1KG(chrom, pathway_df = NULL)
```

Arguments

- **chrom** Numeric. The chromosome number(s). A numeric list of chromosome numbers representing the 1000 Genomes Project exon-data to load.
- **pathway_df** Data frame. (Optional) A data frame that contains the positions for each exon in a pathway of interest. This data frame must contain the variables chrom, exonStart, and exonEnd. See Details.
Details

The `load_1KG` is used to load pre-formatted, exon-only SNV data from any of the 22 human autosomes. The original data was obtained from:

http://ftp.1000genomes.ebi.ac.uk/vol1/ftp/data_collections/1000_genomes_project/release/.

The data was reduced to remove any related individuals, to accomplish this we randomly sampled one relative from each set of related individuals. This resulted in the removal of 22 individuals. Additional information regarding the formatting of the 1000 Genomes Project data may be found at https://github.com/simrvprojects/1000-Genomes-Exon-Data/ in the pdf file entitled "Documentation for Creating Exon Data_090319.pdf".

We expect that `pathwayDF` does not contain any overlapping segments. Users may combine overlapping exons into a single observation with the `combine_exons` function.

Value

An object of class `SNVdata` containing the imported exon data.

References


See Also

`combine_exons`

Examples

```r
exdata = load_1KG(21:22)
unique(exdata$Mutations$chrom)

head(exdata$Mutations)
exdata$Haplotypes[1:20, 1:10]
head(exdata$Samples)
```

Description

To import SLiM data into R, we provide the `read_slim` function, which has been tested for SLiM versions 2.0-3.1. The `read_slim` function is only appropriate for single-nucleotide variant (SNV) data produced by SLiM's `outputFull()` method. We do not support output in MS or VCF data format, i.e. produced by `outputVCFsample()` or `outputMSSample()` in SLiM.

Usage

```r
read_slim(file_path, keep_maf = 0.01, recomb_map = NULL,
          pathway_df = NULL, recode_recurrent = TRUE)
```
**read_slim**

**Arguments**

- **file_path** character. The file path or URL of the .txt output file created by the outputFull() method in SLiM.

- **keep_maf** numeric. The largest allele frequency for retained SNVs, by default keep_maf = 0.01. All variants with allele frequency greater than keep_maf will be removed. Please note, removing common variants is recommended for large data sets due to the limitations of data allocation in R. See details.

- **recomb_map** data frame. (Optional) A recombination map of the same format as the data frame returned by create_slimMap. See details.

- **pathway_df** data frame. (Optional) A data frame that contains the positions for each exon in a pathway of interest. See details.

- **recode_recurrent** logical. When TRUE recurrent SNVs are cataloged a single observation; by default, recode_recurrent = TRUE. See details.

**Details**

In addition to reducing the size of the data, the argument keep_maf has practicable applicability. In family-based studies, common SNVs are generally filtered out prior to analysis. Users who intend to study common variants in addition to rare variants may need to run chromosome specific analyses to allow for allocation of large data sets in R.

The argument recomb_map is used to remap mutations to their actual locations and chromosomes. This is necessary when data has been simulated over non-contiguous regions such as exon-only data. If create_slimMap was used to create the recombination map for SLiM, simply supply the output of create_slimMap to recomb_map. If recomb_map is not provided we assume that the SNV data has been simulated over a contiguous segment starting with the first base pair on chromosome 1.

The data frame pathway_df allows users to identify SNVs located within a pathway of interest. When supplied, we expect that pathway_df does not contain any overlapping segments. All overlapping exons in pathway_df MUST be combined into a single observation. Users may combine overlapping exons with the combine_exons function.

When TRUE, the logical argument recode_recurrent indicates that recurrent SNVs should be recorded as a single observation. SLiM can model many types of mutations; e.g. neutral, beneficial, and deleterious mutations. When different types of mutations occur at the same location carriers will experience different fitness effects depending on the carried mutation. However, when mutations at the same location have the same fitness effects, they represent a recurrent mutation. Even so, SLiM stores recurrent mutations separately and calculates their prevalence independently. When the argument recode_recurrent = TRUE we store recurrent mutations as a single observation and calculate the derived allele frequency based on their combined prevalence. This convention allows for both reduction in storage and correct estimation of the derived allele frequency of the mutation. Users who prefer to store recurrent mutations from independent lineages as unique entries should set recode_recurrent = FALSE.

An object of class **SNVdata**, which inherits from a list and contains: The read_slim function returns an object of class **SNVdata**, which inherits from a list and contains the following two items:
1. Haplotypes A sparse matrix of class dgCMatrix (see \texttt{dgCMatrix-class}). The columns in Haplotypes represent distinct SNVs, while the rows represent individual haplotypes. We note that this matrix contains two rows of data for each diploid individual in the population: one row for the maternally inherited haplotype and the other for the paternally inherited haplotype.

2. Mutations A data frame cataloging SNVs in Haplotypes. The variables in the Mutations data set are described as follows:

- \texttt{colID} Associates the rows, i.e. SNVs, in Mutations to the columns of Haplotypes.
- \texttt{chrom} The chromosome that the SNV resides on.
- \texttt{position} The position of the SNV in base pairs.
- \texttt{afreq} The derived allele frequency of the SNV.
- \texttt{marker} A unique character identifier for the SNV.
- \texttt{type} The mutation type, as specified in the user’s slim simulation.
- \texttt{pathwaySNV} Identifies SNVs located within the pathway of interest as \texttt{TRUE}.

Please note: the variable \texttt{pathwaySNV} will be omitted when \texttt{pathway_df} is not supplied to \texttt{read_slim}.

**Value**

An object of class \texttt{SNVdata}, which inherits from a list and contains:

- \texttt{Haplotypes} A sparse matrix of haplotypes. See details.
- \texttt{Mutations} A data frame cataloging SNVs in Haplotypes. See details.

**References**


**See Also**

\texttt{create_slimMap, combine_exons, dgCMatrix-class}

**Examples**

```r
# Specify the URL of the example output data simulated by SLiM.
file_url <- 'https://raw.githubusercontent.com/cnieuwoudt/Example-SLiMSim/master/example_SLIMout.txt'
s_out <- read_slim(file_url)

class(s_out)
str(s_out)

# As seen above, read_slim returns an object of class SNVdata,
# which contains two items. The first is a sparse matrix
# named Haplotypes, which contains the haplotypes for each individual in the
# simulation. The second item is a data set named Mutations, which catalogs
```
# the mutations in the Haplotypes matrix.

# View the first 5 lines of the mutation data
head(s_out$Mutations, n = 5)

# view the first 20 mutations on the first 10 haplotypes
s_out$Haplotypes[1:10, 1:20]

---

**Simulate sequence data for a sample of pedigrees**

**Description**

Simulate single-nucleotide variant (SNV) data for a sample of pedigrees.

**Usage**

```r
sim_RVstudy(ped_files, SNV_data, affected_only = TRUE, remove_wild = TRUE, pos_in_bp = TRUE, gamma_params = c(2.63, 2.63/0.5), burn_in = 1000, SNV_map = NULL, haplos = NULL)
```

**Arguments**

- `ped_files` Data frame. A data frame of pedigrees for which to simulate sequence data, see details.
- `SNV_data` SNVdata. An object of class SNVdata created by `SNVdata`.
- `affected_only` Logical. When `affected_only = TRUE`, we only simulate SNV data for the disease-affected individuals and the family members that connect them along a line of descent. When `affected_only = FALSE`, SNV data is simulated for the entire study. By default, `affected_only = TRUE`.
- `remove_wild` Logical. When `remove_wild = TRUE` the data is reduced by removing SNVs which are not observed in any of the study participants; otherwise if `remove_wild = FALSE` no data reduction occurs. By default, `remove_wild = TRUE`.
- `pos_in_bp` Logical. This argument indicates if the positions in `SNV_map` are listed in base pairs. By default, `pos_in_bp = TRUE`. If the positions in `SNV_map` are listed in centiMorgan please set `pos_in_bp = FALSE` instead.
- `gamma_params` Numeric list of length 2. The respective shape and rate parameters of the gamma distribution used to simulate distance between chiasmata. By default, `gamma_params = c(2.63, 2*2.63)`, as discussed in Voorrips and Maliepaard (2012).
- `burn_in` Numeric. The "burn-in" distance in centiMorgan, as defined by Voorrips and Maliepaard (2012), which is required before simulating the location of the first chiasmata with interference. By default, `burn_in = 1000`. The burn in distance in cM. By default, `burn_in = 1000`. 
This argument has been deprecated. Users now supply objects of class SNVdata to argument SNV_data.

This argument has been deprecated. Users now supply objects of class SNVdata to argument SNV_data.

**Details**

The `sim_RVstudy` function is used to simulate single-nucleotide variant (SNV) data for a sample of pedigrees. Please note: this function is NOT appropriate for users who wish to simulate genotype conditional on phenotype. Instead, `sim_RVstudy` employs the following algorithm.

1. For each pedigree, we sample a single **causal rare variant (cRV)** from a pool of SNVs specified by the user.
2. Upon identifying the familial cRV we sample founder haplotypes from haplotype data conditional on the founder's cRV status at the familial cRV locus.
3. Proceeding forward in time, from founders to more recent generations, for each parent/offspring pair we:
   a. simulate recombination and formation of gametes, according to the model proposed by Voorrips and Maliepaard (2012), and then
   b. perform a conditional gene drop to model inheritance of the cRV.

It is important to note that due to the forwards-in-time algorithm used by `sim_RVstudy`, certain types of inbreeding and/or loops cannot be accommodated. Please see examples.

For a detailed description of the model employed by `sim_RVstudy`, please refer to section 6 of the vignette.

The data frame of pedigrees, `ped_files`, supplied to `sim_RVstudy` must contain the variables:

<table>
<thead>
<tr>
<th>name</th>
<th>type</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FamID</td>
<td>numeric</td>
<td>family identification number</td>
</tr>
<tr>
<td>ID</td>
<td>numeric</td>
<td>individual identification number</td>
</tr>
<tr>
<td>sex</td>
<td>numeric</td>
<td>sex identification variable: sex = 0 for males, and sex = 1 females.</td>
</tr>
<tr>
<td>dadID</td>
<td>numeric</td>
<td>identification number of father</td>
</tr>
<tr>
<td>momID</td>
<td>numeric</td>
<td>identification number of mother</td>
</tr>
<tr>
<td>affected</td>
<td>logical</td>
<td>disease status indicator: set affected = TRUE if individual has disease.</td>
</tr>
<tr>
<td>DA1</td>
<td>numeric</td>
<td>paternally inherited allele at the cRV locus:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DA1 = 1 if the cRV is inherited, and 0 otherwise.</td>
</tr>
<tr>
<td>DA2</td>
<td>numeric</td>
<td>maternally inherited allele at the cRV locus:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DA2 = 1 if the cRV is inherited, and 0 otherwise.</td>
</tr>
</tbody>
</table>

If `ped_files` does not contain the variables DA1 and DA2 the pedigrees are assumed to be fully sporadic. Hence, the supplied pedigrees will not segregate any of the SNVs in the user-specified pool of cRVS.

Pedigrees simulated by the `sim_RVped` and `sim_ped` functions of the SimRVPedigree package are properly formatted for the `sim_RVstudy` function. That is, the pedigrees generated by these functions contain all of the variables required for `ped_files` (including DA1 and DA2).
The data frame `SNV_map` catalogs the SNVs in `haplos`. The variables in `SNV_map` must be formatted as follows:

<table>
<thead>
<tr>
<th>name</th>
<th>type</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>colID</td>
<td>numeric</td>
<td>associates the rows in <code>SNV_map</code> to the columns of <code>haplos</code></td>
</tr>
<tr>
<td>chrom</td>
<td>numeric</td>
<td>the chromosome that the SNV resides on</td>
</tr>
<tr>
<td>position</td>
<td>numeric</td>
<td>is the position of the SNV in base pairs when argument</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>pos_in_bp = TRUE</code> or centiMorgan when <code>pos_in_bp = FALSE</code></td>
</tr>
<tr>
<td>marker</td>
<td>character</td>
<td>(Optional) a unique character identifier for the SNV.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If missing this variable will be created from <code>chrom</code> and <code>position</code>.</td>
</tr>
<tr>
<td>pathwaySNV</td>
<td>logical</td>
<td>(Optional) identifies SNVs located within the pathway of interest as <code>TRUE</code></td>
</tr>
<tr>
<td>is_CRV</td>
<td>logical</td>
<td>identifies causal rare variants (cRVs) as <code>TRUE</code>.</td>
</tr>
</tbody>
</table>

Please note that when the variable `is_CRV` is missing from `SNV_map`, we sample a single SNV to be the causal rare variant for all pedigrees in the study, which is identified in the returned `famStudy` object.

**Value**

A object of class `famStudy`. Objects of class `famStudy` are lists that include the following named items:

- `ped_files` A data frame containing the sample of pedigrees for which sequence data was simulated.
- `ped_haplos` A sparse matrix that contains the simulated haplotypes for each pedigree member in `ped_files`.
- `haplo_map` A data frame that maps the haplotypes (i.e. rows) in `ped_haplos` to the individuals in `ped_files`.
- `SNV_map` A data frame cataloging the SNVs in `ped_haplos`.

Objects of class `famStudy` are discussed in detail in section 5.2 of the vignette.

**References**


**See Also**

`sim_RVped, read_slim, summary.famStudy`
Examples

library(SimRVSequences)

# load pedigree, haplotype, and mutation data
data(study_peds)
data(EXmuts)
data(EXhaps)

# create variable 'is_CRV' in EXmuts. This variable identifies the pool of
# causal rare variants from which to sample familial cRVs.
EXmuts$is_CRV = FALSE
EXmuts$is_CRV[c(26, 139, 223, 228, 472)] = TRUE

# create object of class SNVdata
my_SNVdata <- SNVdata(Haplotypes = EXhaps,
                      Mutations = EXmuts)

# supply required inputs to the sim_RVstudy function
seqDat = sim_RVstudy(ped_files = study_peds,
                      SNV_data = my_SNVdata)

# Inbreeding examples
# Due to the forward-in-time model used by sim_RVstudy certain types of
# inbreeding and/or loops *may* cause fatal errors when using sim_RVstudy.
# The following examples demonstrate: (1) inbreeding that can be accommodated
# under this model, and (2) when this limitation is problematic.

# Create inbreeding in family 1 of study_peds
imb_ped1 <- study_peds[study_peds$FamID == 3,]
imb_ped1[imb_ped1$ID == 18, c("momID") ] = 7
plot(imb_ped1)

# Notice that this instance of inbreeding can be accommodated by our model.
seqDat = sim_RVstudy(ped_files = imb_ped1,
                      SNV_data = my_SNVdata)

# Create different type of inbreeding in family 1 of study_peds
imb_ped2 <- study_peds[study_peds$FamID == 3,]
imb_ped2[imb_ped2$ID == 8, c("momID") ] = 18
plot(imb_ped2)

# Notice that inbreeding in imb_ped2 will cause a fatal
# error when the sim_RVstudy function is executed
## Not run:
seqDat = sim_RVstudy(ped_files = imb_ped2,
                      SNV_data = my_SNVdata)

## End(Not run)
SNVdata

Constructor function for an object of class SNVdata

Description

Constructor function for an object of class SNVdata

Usage

SNVdata(Haplotypes, Mutations, Samples = NULL)

Arguments

Haplotypes sparseMatrix. A sparse matrix of haplotype data, which contains the haplotypes for unrelated individuals representing the founder population. Rows are assumed to be haplotypes, while columns represent SNVs.

Mutations Data frame. A data frame that catalogs the SNVs in Haplotypes.

Samples An optional dataframe or matrix describing the individuals in Haplotypes.

Value

an object of class SNVdata.

study_peds

Example pedigrees

Description

This data set contains ped data for five ascertained pedigrees. The ascertained pedigrees were simulated by the sim_RVped function, which is included with the R package SimRVPedigree.

Usage

data(study_peds)

Format

A data frame with 77 rows and 15 variables:

- **FamID** Family identification number.
- **ID** Individual identification number.
- **sex** Sex identification variable: `sex = 0` for males, and `sex = 1` females.
- **dadID** Identification number of father
- **momID** Identification number of mother
affected  disease-affection status: affected = TRUE if individual has developed disease, and FALSE otherwise.

DA1  Paternally inherited allele at the familial disease locus: DA1 = 1 if the casual variant is inherited, and 0 otherwise.

DA2  Maternally inherited allele at the familial disease locus: DA2 = 1 if the casual variant is inherited, and 0 otherwise.

birthYr  The individual’s birth year.

onsetYr  The individual’s year of disease onset, when applicable, and NA otherwise.

deadYr  The individual’s year of death, when applicable, and NA otherwise.

RR  The subject’s relative-risk of disease

available  Availability status: available = TRUE if individual is recalled by the proband, and FALSE if not recalled or a marry-in.

Gen  The individual’s generation number relative to the eldest pedigree founder. That is, the seed founder will have Gen = 1, his or her offspring will have Gen = 2, etc.

proband  Proband identification variable: proband = TRUE if the individual is the proband, and FALSE otherwise.

References


---

summary.famStudy  Summary function for objects of class famStudy

Description

Summary function for objects of class famStudy, i.e. objects returned by the sim_RVstudy function.

Usage

```r
## S3 method for class 'famStudy'
summary(object, ...)
```

Arguments

- `object`  An object of class famStudy, returned by the sim_RVstudy function.
- `...`  additional arguments passed to other methods.
summary.famStudy

Details

The `summary.famStudy` function returns a list containing two items. The first item, `fam_allele_count`, is a matrix that contains counts of the SNVs shared by the disease-affected relatives in each pedigree. This matrix will contain a row of counts for each pedigree in the supplied `famSustudy` object. The first column in `fam_allele_count` is named `FamID` and identifies each pedigree by their family identification number. The remaining columns in `fam_allele_count` are named according to the respective marker names of the shared SNVs.

The second item returned by `summary.famStudy` is a data frame named `pathway_count`, which catalogs the SNVs shared among disease-affected study participants. This data frame contains the following variables:

<table>
<thead>
<tr>
<th>name</th>
<th>type</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>chrom</td>
<td>numeric</td>
<td>chromosome identification number</td>
</tr>
<tr>
<td>position</td>
<td>numeric</td>
<td>the position of the SNV</td>
</tr>
<tr>
<td>marker</td>
<td>character</td>
<td>a unique character identifier for the SNV</td>
</tr>
<tr>
<td>total</td>
<td>numeric</td>
<td>the number of SNV copies observed in disease-affected study participants.</td>
</tr>
<tr>
<td>is_crv</td>
<td>logical</td>
<td>identifies causal rare variants (cRVs) as TRUE</td>
</tr>
<tr>
<td>pathwaySNV</td>
<td>logical</td>
<td>identifies SNVs located within the pathway of interest as TRUE.</td>
</tr>
</tbody>
</table>

Please note, the variable `pathwaySNV` is omitted when missing from the `SNV_map` data frame in the `famStudy` object. See `sim_RVstudy` for more details.

Value

- `fam_allele_count`: A matrix that contains counts of the SNVs shared by the disease-affected relatives in each pedigree.
- `pathway_count`: A data frame that catalogs the SNVs shared among disease-affected study participants. See details.

See Also

`sim_RVstudy`

Examples

```r
library(SimRVSequences)

# load pedigree, haplotype, and mutation data
data(study_peds)
data(EXmuts)
data(EXhaps)

# create variable is_CRV in EXmuts to identify the causal
# rare variants from which to sample familial cRVs.
EXmuts$is_CRV = FALSE
EXmuts$is_CRV[c(26, 139, 223, 228, 472)] = TRUE
```
# supply required inputs to the sim_RVstudy function
seqDat = sim_RVstudy(ped_files = study_peds,
                       SNV_data = SNVdata(Haplotypes = EXhaps,
                                           Mutations = EXmuts))

# to count the number of SNVs shared by the disease-affected relatives in each pedigree, supply the output returned by # sim_RVstudy to the summary function
summary(seqDat)
Index

*Topic datasets
  EXhaps, 5
  EXmuts, 6
  hg_apopPath, 7
  hg_exons, 8
  study_peds, 17

  combine_exons, 2, 3, 4, 10–12
  create.slimMap, 3, 11, 12

  EXhaps, 5, 6
  EXmuts, 5, 6
  genos2sparseMatrix, 7

  hg_apopPath, 7
  hg_exons, 8

  load_1KG, 9

  read.slim, 4–6, 10, 15

  sim_ped, 14
  sim_RVped, 14, 15
  sim_RVstudy, 13, 18, 19
  SNVdata, 11–13, 17
  study_peds, 17
  summary.famStudy, 15, 18