Package 'AbSim'

June 18, 2017

Type Package

Title Time Resolved Simulations of Antibody Repertoires

Version 0.2.2

Date 2017-6-18

Author Alexander Yermanos

Maintainer Alexander Yermanos <ayermano@ethz.ch>

Depends R(>= 3.1.0), ape, poweRlaw

Description Simulation methods for the evolution of antibody repertoires. The heavy and light chain variable region of both human and C57BL/6 mice can be simulated in a time-dependent fashion. Both single lineages using one set of V-, D-, and J-genes or full repertoires can be simulated. The algorithm begins with an initial V-D-J recombination event, starting the first phylogenetic tree. Upon completion, the main loop of the algorithm begins, with each iteration representing one simulated time step. Various mutation events are possible at each time step, contributing to a diverse final repertoire.

License GPL-2

Encoding UTF-8

RoxygenNote 5.0.1

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2017-06-18 18:14:37 UTC

R topics documented:

`clonalExpansion` *Clonally expands the simulated repertoire generated from fullRepertoire function.*

Description

Clonally expands the simulated repertoire generated from fullRepertoire function.

Usage

```
clonalExpansion(ab.repertoire, rep.size, distribution, with.germline,
  dist.parameters)
```

Arguments

`ab.repertoire`	The output from the fullRepertoire function. This takes should be a nested list, with the first element containing a list of sequence arrays, the second element containing the corresponding list of names, and the third element containing a list of trees. The following list elements (containing the sampling information) will be disregarded for repertoire expansion.
`rep.size`	Controls the total size of the final repertoire. This number should be greater than the total number of sequences that are provided as input. Note that currently using the "powerlaw" distribution will return less than the exact rep.size parameter due to integer values of clonal frequencies.
`distribution`	This parameter controls how the clonal frequency of the repertoire is distributed. Options include "powerlaw" ("pl"), "identical" ("id")
`with.germline`	Logical - If false, the germline from each lineage will be removed.
`dist.parameters`	Supplies the parameters for how the clonal frequencies should be distributed.

Value

Returns a list with three elements. The first list element contains a character array, with the sequences composing the repertoire. The second element is a character array with the names of the sequences, and the third element in the list corresponds to the original phylogenetic trees that served as the basis for expansion

See Also

fullRepertoire

Examples

```
## Not run:
clonalExpansion(ab.repertoire=fullRepertoire.output,
 rep.size=3*length(unlist(fullRepertoire.output[[1]])),
 distribution="identical",
 with.germline="FALSE")

## End(Not run)
```

`fullRepertoire` *Simulates full heavy chain antibody repertoires for either human or mice.*

Description

Simulates full heavy chain antibody repertoires for either human or mice.

Usage

```
fullRepertoire(max.seq.num, max.timer, SHM.method, baseline.mut,
  SHM.branch.prob, SHM.branch.param, SHM.nuc.prob, species, VDJ.branch.prob,
  proportion.sampled, sample.time, max.tree.num, chain.type, vdj.model,
  vdj.insertion.mean, vdj.insertion.stdv)
```

Arguments

`max.seq.num`	The maximum number of tips allowed at the end of the simulation. The simulation will end when either this or the max.timer is reached. Note - this function does not take clonal frequency into account. This parameter resembles the species richness, or the measure of unique sequences in the repertoire.
`max.timer`	The maximum number of time steps allowed during the simulation. The simulation will end when either this or the max.seq.num is reached.
`SHM.method`	The mode of SHM speciation events. Options are either: "poisson","data","motif","wrc", and "all". Specifying "poisson" will result in mutations that can occur anywhere in the heavy chain region, with each nucleotide having an equal probability for a mutation event. Specifying "data" focuses mutation events during SHM in the

CDR regions (based on IMGT), and there will be an increased probability for transitions (and decreased probability for transversions). Specifying "motif" will cause neighbor dependent mutations based on a mutational matrix from high throughput sequencing data sets (Yaari et al., Frontiers in Immunology, 2013). "wrc" allows for only the WRC mutational hotspots to be included (where W equals A or T and R equals A or G). Specifying "all" will use all four types of mutations during SHM branching events, where the weights for each can be specified in the "SHM.nuc.prob" parameter.

`baseline.mut` Specifies the probability (gamma) for each nucleotide to be mutated inbetween speciation events. These mutations do not cause any branching events. This parameter gives each site a probability to be mutated (in all current sequences) at each time step. Currently these are only Poisson distributed but future releases will change it to allow for other mutation methods.

`SHM.branch.prob`

Specifies the probability for a given sequence to undergo SHM events (thus, branching events) This parameter corresponds to the distribution specified in "SHM.branch.prob". For "identical" only one value should be supplied. For "uniform", a vector of length 3 should be specified corresponding to n,min,max respectively (stats::runif(n, min = 0, max = 1)). For "exponential", a single value controlling the rate parameter (from stats::rexp()) should be supplied. For "lognorm" a vector of length two should be supplied, with the first value corresponding to meanlog and the second corresponding to sdlog (from stats::rlnorm). Similarly, for "normal" distribution, two values corresponding to the mean and standard deviation (respectively) should be supplied.

`SHM.branch.param`

Describes the probability of undergoing SHM events. This parameter is responsible for describing how likely each sequence will undergo branching events in the phylogeny. The following options are possible: "identical", "uniform", "exponential" ("exp"), "lognormal" ("lognorm"), "normal" ("norm").

`SHM.nuc.prob` Specifies the rate at which nucleotides change during speciation (SHM) events. This parameter depends on the type of mutation specified by SHM.method. For both "poisson" and "data", the input value determines the probability for each site to mutate (the whole sequence for "poisson" and the CDRs for "data"). For either "motif" or "wrc", the number of mutations per speciation event should be specified. Note that these are not probabilities, but the number of mutations that can occur (if the mutation is present in the sequence). If "all" is specified, the input should be a vector where the first element controls the poisson style mutations, second controls the "data", third controls the "motif" and fourth controls the "wrc".

`species` Either "mus" for C57BL/6 germline genes or "hum" for human germline genes. These genes were taking from IMGT. When more than one allele was present for a given gene, the first was used.

`VDJ.branch.prob`

The probabilty of a new VDJ recombination event of occuring. For the singleLineage function this will result in a branching event at the site of the unmutated germline. For fullRepertoire function, this will cause a new tree to begin.

`proportion.sampled`
: Value ranging from 0 and 1 specifying the proportion of sequences to be sampled at each time point. Specifiying 1 indicates that all sequences will be recovered at each time point, whereas 0.5 will sample half of the sequences.

`sample.time`
: Integer array indicating the time points at which sampling events should occur.

`max.tree.num`
: Integer value describing maximum number of trees allowed to generate the core sequences of the repertoire. Each of these trees is started by an independent VDJ recombination event.

`chain.type`
: String determining whether heavy or light chain should be simulated. Either "heavy" for heavy chains or "light" for light chains. Heavy chains will have V-D-J recombination, whereas light chain will just have V-J recombination.

`vdj.model`
: Specifies the model used to simulate V-D-J recombination. Can be either "naive" or "data". "naive" is chain independent and does not differentiate between different species. To rely on the default "experimental" options, this should be "data" and the parameter vdj.insertion.mean should be "default". This will allow for different mean additions for either the VD and JD junctions and will differ depending on species.

`vdj.insertion.mean`
: Integer value describing the mean number of nucleotides to be inserted during simulated V-D-J recombination events. If "default" is entered, the mean will be normally distribut

`vdj.insertion.stdv`
: Integer value describing the standard deviation corresponding to insertions of V-D-J recombination. No "default" parameter currently supported but will be updated with future experimental data. This should be a number if using a custom distribution for V-D-J recombination events, but can be "default" if using the "naive" vdj.model or the "data", with vdj.insertion.mean set to "default".

Value

Returns a nested list. output[[1]][[1]] is an array of the simulated sequences output[[2]][[1]] is an array names corresponding to each sequence. For example, output[[2]][[1]][1] is the name of the sequence corresponding to output[[1]][[1]][1]. The simulated tree of this is found in output[[3]][[1]]. The length of the output list is determined by the number of sampling points Thus if you have two sampling points, output[[4]][[1]] would be a character array holding the sequences with output[[5]][[1]] as a character array holding the corresponding names. Then the sequences recovered second sampling point would be stored at output[[6]][[1]], with the names at output[[7]][[1]]. This nested list was designed for full antibody repertoire simulations, and thus, may seem unintuitive for the single lineage function. The first sequence and name corresponds to the germline sequence that served as the root of the tree. See vignette for comprehensive example

See Also

singleLineage

hotspot_df *hotspot_df*

Description

Hotspot mutations taken from Yaari et al., Frontiers in Immunology, 2013. This contains transition probabilities for all 5mer combinations based on high throughput sequencing data. The transition probabilities are for the middle nucleotide in each 5mer set. This can be customized by changing the genes and sequences. Custom mutation hotspots can be supplied by modifying this dataframe. Repeating particular hotspot entries allows for the hotspot to mutate more than one time per SHM event.

Usage

```
hotspot_df
```

Format

An object of class `data.frame` with 1024 rows and 6 columns.

Details

@format A data frame with 1024 rows and 6 variables:

pattern Character array where each entry corresponds to a 5 base motif. The mutation probabilities correspond to the middle nucleotide in each 5mer.

toA The probability for the middle nucleotide in "pattern" to mutate to an adenine

toC The probability for the middle nucleotide in "pattern" to mutate to an cytosine

toG The probability for the middle nucleotide in "pattern" to mutate to an guanine

toT The probability for the middle nucleotide in "pattern" to mutate to an thymine

Source The origin of how this motif was discovered. Either Inferred or Experimental

Source

Yaari et al., Frontiers in Immunology, 2013

ighd_hum_df *ighd_hum_df*

Description

human germline IgH-D (heavy chain d-gene segments). When multiple alleles were present, the first one was included. These names and sequences can be changed by customized by changing this dataframe. Additionally, repeating elements can give certain germline gene elements a larger probability of being used during repertoire evolution.

Usage

```
ighd_hum_df
```

Format

A data frame with 37 rows and 2 variables:

gene The gene name

seq The corresponding sequence

Source

IMGT

ighd_mus_df *ighd_mus_df*

Description

C57BL/6 germline IgH-D (heavy chain d-gene segments). When multiple alleles were present, the first one was included. These names and sequences can be changed by customized by changing this dataframe. Additionally, repeating elements can give certain germline gene elements a larger probability of being used during repertoire evolution.

Usage

```
ighd_mus_df
```

Format

A data frame with 27 rows and 2 variables:

gene The gene name

seq The corresponding sequence

Source

IMGT

`ighj_hum_df` *ighj_hum_df*

Description

human germline IgH-V (heavy chain v-gene segments). When multiple alleles were present, the first one was included. These names and sequences can be changed by customized by changing this dataframe. Additionally, repeating elements can give certain germline gene elements a larger probability of being used during repertoire evolution.

Usage

```
ighj_hum_df
```

Format

A data frame with 6 rows and 2 variables:

gene The gene name

seq The corresponding sequence

Source

IMGT

`ighj_mus_df` *ighj_mus_df*

Description

C57BL/6 germline IgH-J (heavy chain j-gene segments). When multiple alleles were present, the first one was included. These names and sequences can be changed by customized by changing this dataframe. Additionally, repeating elements can give certain germline gene elements a larger probability of being used during repertoire evolution.

Usage

```
ighj_mus_df
```

Format

A data frame with 4 rows and 2 variables:

gene The gene name

seq The corresponding sequence

Source

IMGT

`ighv_hum_df` *ighv_hum_df*

Description

human germline IgH-V (heavy chain v gene segments). When multiple alleles were present, the first one was included. These names and sequences can be changed by customized by changing this dataframe. Additionally, repeating elements can give certain germline gene elements a larger probability of being used during repertoire evolution.

Usage

```
ighv_hum_df
```

Format

A data frame with 119 rows and 2 variables:

gene The gene name

seq The corresponding sequence

Source

IMGT

ighv_mus_df *ighv_mus_df*

Description

C57BL/6 germline IgH-V (heavy chain v-gene segments). When multiple alleles were present, the first one was included. These names and sequences can be changed by customized by changing this dataframe. Additionally, repeating elements can give certain germline gene elements a larger probability of being used during repertoire evolution.

Usage

```
ighv_mus_df
```

Format

A data frame with 164 rows and 2 variables:

gene The gene name

seq The corresponding sequence

Source

IMGT

igkj_hum_df *igkj_hum_df*

Description

human germline IgK-J (light chain kappa j-gene segments). When multiple alleles were present, the first one was included. These names and sequences can be changed by customized by changing this dataframe. Additionally, repeating elements can give certain germline gene elements a larger probability of being used during repertoire evolution.

Usage

```
igkj_hum_df
```

Format

A data frame with 5 rows and 2 variables:

gene The gene name

seq The corresponding sequence

Source

IMGT

`igkj_mus_df` *igkj_mus_df*

Description

mouse germline IgK-J (light chain kappa J-gene segments). When multiple alleles were present, the first one was included. These names and sequences can be changed by customized by changing this dataframe. Additionally, repeating elements can give certain germline gene elements a larger probability of being used during repertoire evolution.

Usage

```
igkj_mus_df
```

Format

A data frame with 5 rows and 2 variables:

gene The gene name

seq The corresponding sequence

Source

IMGT

`igkv_hum_df` *igkv_hum_df*

Description

human germline IgK-V (light chain kappa v-gene segments). When multiple alleles were present, the first one was included. These names and sequences can be changed by customized by changing this dataframe. Additionally, repeating elements can give certain germline gene elements a larger probability of being used during repertoire evolution.

Usage

```
igkv_hum_df
```

Format

A data frame with 74 rows and 2 variables:

gene The gene name

seq The corresponding sequence

Source

IMGT

`igkv_mus_df` *igkv_mus_df*

Description

mouse germline IgK-V (light chain kappa V-gene segments). When multiple alleles were present, the first one was included. These names and sequences can be changed by customized by changing this dataframe. Additionally, repeating elements can give certain germline gene elements a larger probability of being used during repertoire evolution.

Usage

```
igkv_mus_df
```

Format

A data frame with 76 rows and 2 variables:

gene The gene name

seq The corresponding sequence

Source

IMGT

iglj_hum_df *iglj_hum_df*

Description

human germline IgL-J (light chain lambda j-gene segments). When multiple alleles were present, the first one was included. These names and sequences can be changed by customized by changing this dataframe. Additionally, repeating elements can give certain germline gene elements a larger probability of being used during repertoire evolution.

Usage

```
iglj_hum_df
```

Format

A data frame with 9 rows and 2 variables:

gene The gene name

seq The corresponding sequence

Source

IMGT

iglj_mus_df *iglj_mus_df*

Description

mouse germline IgL-J (light chain lambda J-gene segments). When multiple alleles were present, the first one was included. These names and sequences can be changed by customized by changing this dataframe. Additionally, repeating elements can give certain germline gene elements a larger probability of being used during repertoire evolution.

Usage

```
iglj_mus_df
```

Format

A data frame with 7 rows and 2 variables:

gene The gene name

seq The corresponding sequence

Source

IMGT

`iglv_hum_df` *iglv_hum_df*

Description

human germline IgL-V (light chain lambda v-gene segments). When multiple alleles were present, the first one was included. These names and sequences can be changed by customized by changing this dataframe. Additionally, repeating elements can give certain germline gene elements a larger probability of being used during repertoire evolution.

Usage

```
iglv_hum_df
```

Format

A data frame with 49 rows and 2 variables:

gene The gene name

seq The corresponding sequence

Source

IMGT

`iglv_mus_df` *iglv_mus_df*

Description

mouse germline IgL-V (light chain lambda V-gene segments). When multiple alleles were present, the first one was included. These names and sequences can be changed by customized by changing this dataframe. Additionally, repeating elements can give certain germline gene elements a larger probability of being used during repertoire evolution.

Usage

```
iglv_mus_df
```

Format

A data frame with 12 rows and 2 variables:

gene The gene name

seq The corresponding sequence

Source

IMGT

`one_spot_df` *one_spot_df*

Description

WRC hotspot mutations taken from Yaari et al., Frontiers in Immunology, 2013. These include only the mutations following the WRC pattern, where W equals A or T and R equals A or G). Custom mutation hotspots can be supplied by modifying this dataframe. Repeating particular hotspot entries allows for the hotspot to mutate more than one time per SHM event.

Usage

```
one_spot_df
```

Format

A data frame with 32 rows and 6 variables:

pattern Character array where each entry corresponds to a 5 base motif. The mutation probabilities correspond to the middle nucleotide in each 5mer.

toA The probability for the middle nucleotide in "pattern" to mutate to an adenine

toC The probability for the middle nucleotide in "pattern" to mutate to an cytosine

toG The probability for the middle nucleotide in "pattern" to mutate to an guanine

toT The probability for the middle nucleotide in "pattern" to mutate to an thymine

Source The origin of how this motif was discovered. Either Inferred or Experimental

Source

Yaari et al., Frontiers in Immunology, 2013

`repertoireFastas` *Converts AbSim's output sequences and names into fasta file format. Each tree within the repertire will be stored in an individual fasta file. Utilizes sink() and cloneAllConnections() from R base*

Description

Converts AbSim's output sequences and names into fasta file format. Each tree within the repertire will be stored in an individual fasta file. Utilizes sink() and cloneAllConnections() from R base

Usage

```
repertoireFastas(ab.repertoire, fasta.name, with.germline)
```

Arguments

`ab.repertoire`	The output from the either the singleLineage function or the fullRepertoire function. This takes should be a nested list, with the first element containing a list of sequence arrays, the second element containing the corresponding list of names, and the third element containing a list of trees. The following list elements (containing the sampling information) will be disregarded for repertoire expansion.
`fasta.name`	The name of the fasta files. Each tree will be numbered corresponding to the index within the ab.repertoire list. For example, the second tree in an ab.repertoire object will be "fasta.name_2.fasta",
`with.germline`	Logical value specifying if the germline sequences should be included in the fasta file (as the first sequence)

Examples

```
## Not run:
repertoireFastas(ab.repertoire=ab.repertoire.output,
 fasta.name="sample_trees",
 with.germline=TRUE

## End(Not run)
```

`singleLineage` *Antibody lineage simulations using only one set of V(D)J germline genes. The main difference between this function and the fullRepertoire function is that there can be multiple VDJ recombination events within one tree. Each VDJ recombination event in the singleLineage function is a branching event within the existing tree, whereas the VDJ recombination events in the fullRepertoire function start a new tree.*

Description

Antibody lineage simulations using only one set of V(D)J germline genes. The main difference between this function and the fullRepertoire function is that there can be multiple VDJ recombination events within one tree. Each VDJ recombination event in the singleLineage function is a branching event within the existing tree, whereas the VDJ recombination events in the fullRepertoire function start a new tree.

Usage

```
singleLineage(max.seq.num, max.timer, SHM.method, SHM.nuc.prob, baseline.mut,
  SHM.branch.prob, SHM.branch.param, species, max.VDJ, VDJ.branch.prob,
  proportion.sampled, sample.time, chain.type, vdj.model, vdj.insertion.mean,
  vdj.insertion.stdv)
```

Arguments

`max.seq.num`	The maximum number of tips allowed at the end of the simulation. The simulation will end when either this or the max.timer is reached. Note - this function does not take clonal frequency into account. This parameter resembles the species richness, or the measure of unique sequences in the repertoire.
`max.timer`	The maximum number of time steps allowed during the simulation. The simulation will end when either this or the max.seq.num is reached.
`SHM.method`	The mode of SHM speciation events. Options are either: "poisson","data","motif","wrc", and "all". Specifying "poisson" will result in mutations that can occur anywhere in the heavy chain region, with each nucleotide having an equal probability for a mutation event. Specifying "data" focuses mutation events during SHM in the CDR regions (based on IMGT), and there will be an increased probability for transitions (and decreased probability for transversions). Specifying "motif" will cause neighbor dependent mutations based on a mutational matrix from high throughput sequencing data sets (Yaari et al., Frontiers in Immunology, 2013). "wrc" allows for only the WRC mutational hotspots to be included (where W equals A or T and R equals A or G). Specifying "all" will use all four types of mutations during SHM branching events, where the weights for each can be specified in the "SHM.nuc.prob" parameter.
`SHM.nuc.prob`	Specifies the rate at which nucleotides change during speciation (SHM) events. This parameter depends on the type of mutation specified by SHM.method. For both "poisson" and "data", the input value determines the probability for each site to mutate (the whole sequence for "poisson" and the CDRs for "data"). For either "motif" or "wrc", the number of mutations per speciation event should be specified. Note that these are not probabilities, but the number of mutations that can occur (if the mutation is present in the sequence). If "all" is specified, the input should be a vector where the first element controls the poisson style mutations, second controls the "data", third controls the "motif" and fourth controls the "wrc".
`baseline.mut`	Specifies the probability (gamma) for each nucleotide to be mutated inbetween speciation events. These mutations do not cause any branching events. This parameter gives each site a probability to be mutated (in all current sequences)

at each time step. Currently these are only Poisson distributed but future releases will change it to allow for other mutation methods.

`SHM.branch.prob`

Specifies the probability for a given sequence to undergo SHM events (thus, branching events) This parameter corresponds to the distribution specified in "SHM.branch.prob". For "identical" only one value should be supplied. For "uniform", a vector of length 3 should be specified corresponding to n,min,max respectively (stats::runif(n, min = 0, max = 1)). For "exponential", a single value controlling the rate parameter (from stats::rexp()) should be supplied. For "lognorm" a vector of length two should be supplied, with the first value corresponding to meanlog and the second corresponding to sdlog (from stats::rlnorm). Similarly, for "normal" distribution, two values corresponding to the mean and standard deviation (respectively) should be supplied.

`SHM.branch.param`

Describes the probability of undergoing SHM events. This parameter is responsible for describing how likely each sequence will undergo branching events in the phylogeny. The following options are possible: "identical", "uniform", "exponential" ("exp"), "lognormal" ("lognorm"), "normal" ("norm").

`species` Either "mus" for C57BL/6 germline genes or "hum" for human germline genes. These genes were taking from IMGT. When more than one allele was present for a given gene, the first was used.

`max.VDJ` The maximum number of VDJ events allowed. These VDJ events are independent of each other but use the same VDJ segments to create a new branching event in the tree at the unmutated germline.

`VDJ.branch.prob`

The probabilty of a new VDJ recombination event of occuring. For the singleLineage function this will result in a branching event at the site of the unmutated germline. For fullRepertoire function, this will cause a new tree to begin.

`proportion.sampled`

Value ranging from 0 and 1 specifying the proportion of sequences to be sampled at each time point. Specifiying 1 indicates that all sequences will be recovered at each time point, whereas 0.5 will sample half of the sequences.

`sample.time` Integer array indicating the time points at which sampling events should occur.

`chain.type` String determining whether heavy or light chain should be simulated. Either "heavy" for heavy chains or "light" for light chains. Heavy chains will have V-D-J recombination, whereas light chain will just have V-J recombination.

`vdj.model` Specifies the model used to simulate V-D-J recombination. Can be either "naive" or "data". "naive" is chain independent and does not differentiate between different species. To rely on the default "experimental" options, this should be "data" and the parameter vdj.insertion.mean should be "default". This will allow for different mean additions for either the VD and JD junctions and will differ depending on species.

`vdj.insertion.mean`

Integer value describing the mean number of nucleotides to be inserted during simulated V-D-J recombination events. If "default" is entered, the mean will be normally distribut

`vdj.insertion.stdv`
: Integer value describing the standard deviation corresponding to insertions of V-D-J recombination. No "default" parameter currently supported but will be updated with future experimental data. This should be a number if using a custom distribution for V-D-J recombination events, but can be "default" if using the "naive" vdj.model or the "data", with vdj.insertion.mean set to "default".

Value

Returns a nested list containing both sequence information and phylogenetic trees. If "output" is the returned object, then output[[1]][[1]] is an array of the simulated sequences output[[2]][[1]] is an array names corresponding to each sequence. For example, output[[2]][[1]][1] is the name of the sequence corresponding to output[[1]][[1]][1]. The simulated tree of this is found in output[[3]][[1]]. The length of the output list is determined by the number of sampling points Thus if you have two sampling points, output[[4]][[1]] would be a character array holding the sequences with output[[5]][[1]] as a character array holding the corresponding names. Then the sequences recovered second sampling point would be stored at output[[6]][[1]], with the names at output[[7]][[1]]. This nested list was designed for full antibody repertoire simulations, and thus, may seem unintuitive for the single lineage function. The first sequence and name corresponds to the germline sequence that served as the root of the tree.

See Also

fullRepertoire

Index