Package 'jackalope'

July 11, 2019

Type Package

Title A Swift, Versatile Phylogenomic and High-Throughput Sequencing Simulator

Version 0.1.2

Description Simply and efficiently
simulates (i) variants from reference genomes and (ii) reads from both Illumina <https://www.illumina.com/>
and Pacific Biosciences (PacBio) <https://www.pacb.com/> platforms.
It can either read reference genomes from FASTA files or simulate new ones.
Genomic variants can be simulated using summary statistics, phylogenies,
Variant Call Format (VCF) files, and coalescent simulations—the latter of which
can include selection, recombination, and demographic fluctuations.
'jackalope' can simulate single, paired-end, or mate-pair Illumina reads,
as well as PacBio reads.
These simulations include sequencing errors, mapping qualities, multiplexing,
and optical/polymerase chain reaction (PCR) duplicates.
Simulating Illumina sequencing is based on ART
by Huang et al. (2012) <doi:10.1093/bioinformatics/btr708>.
PacBio sequencing simulation is based on
SimLoRD by Stöcker et al. (2016) <doi:10.1093/bioinformatics/btw286>.
All outputs can be written to standard file formats.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Depends R (>= 2.10)

biocViews

Imports ape, R6, Rcpp (>= 0.12.11), RcppProgress (>= 0.1), zlibbioc

LinkingTo Rcpp, RcppArmadillo, RcppProgress, Rhtslib, zlibbioc

SystemRequirements C++11

RoxygenNote 6.1.1

Suggests coala, knitr, scrm, testthat, vcfR

VignetteBuilder knitr

URL https://github.com/lucasnell/jackalope

BugReports https://github.com/lucasnell/jackalope/issues

NeedsCompilation yes

Author Lucas A. Nell [cph, aut, cre] (<https://orcid.org/0000-0003-3209-0517>)

Maintainer Lucas A. Nell <lucas@lucasnell.com>

Repository CRAN

Date/Publication 2019-07-11 12:12:44 UTC

R topics documented:

`create_genome` *Create a reference genome.*

Description

Random sequences are generated to create a new `ref_genome` object. Note that this function will never generate empty sequences.

Usage

```
create_genome(n_seqs, len_mean, len_sd = 0, pi_tcag = rep(0.25, 4),
  n_threads = 1)
```

Arguments

n_seqs	Number of sequences.
len_mean	Mean for the gamma distribution of sequence sizes.
len_sd	Standard deviation for the gamma distribution of sequence sizes. If set to <= 0, all sequences will be the same length. Defaults to 0.
pi_tcag	Vector of length 4 containing the nucleotide equilibrium frequencies for "T", "C", "A", and "G", respectively. Defaults to rep(0.25, 4).
n_threads	Number of threads to use for parallel processing. This argument is ignored if OpenMP is not enabled. Defaults to 1.

Value

A ref_genome object.

Examples

```
genome <- create_genome(10, 100e3, 100, pi_tcag = c(0.1, 0.2, 0.3, 0.4))
```

---

create_variants *Create variants from a reference genome.*

Description

Uses one of multiple methods to create haploid variants from a reference genome. See vars_functions for the methods available.

Usage

```
create_variants(reference, vars_info, sub, ins = NULL, del = NULL,
  gamma_mats = NULL, region_size = 100, n_threads = 1,
  show_progress = FALSE)
```

Arguments

reference	A ref_genome object from which to generate variants. This argument is required.
vars_info	Output from one of the vars_functions. These functions organize higher-level information for use here. See vars_functions for brief descriptions and links to each method. If this argument is NULL, all arguments other than reference are ignored, and an empty variants object with no variants is returned. This is designed for use when you'd like to add mutations manually. If you create a blank variants object, you can= use its add_vars method to add variants manually.

sub	Output from one of the sub_models functions that organizes information for the substitution models. See sub_models for more information on these models and their required parameters. This argument is only allowed to be missing if you are using a VCF file to create variants. Defaults to NULL.
ins	Output from the indels function that specifies rates of insertions by length. Passing NULL to this argument results in no insertions. Defaults to NULL.
del	Output from the indels function that specifies rates of deletions by length. Passing NULL to this argument results in no deletions. Defaults to NULL.
gamma_mats	Output from the site_var function that specifies variability in mutation rates among sites (for both substitutions and indels). Passing NULL to this argument results in no variability among sites. Defaults to NULL.
region_size	Size of regions to break genome into for sampling mutation locations. This causes Gamma regions to be split into smaller sections (obviously the Gamma values themselves are not changed). Doing this splitting is useful because sampling within a region is more computationally costly than sampling among regions. Higher numbers will result in lower memory usage but slower speed. Defaults to 100.
n_threads	Number of threads to use for parallel processing. This argument is ignored if OpenMP is not enabled. Threads are spread across sequences, so it doesn't make sense to supply more threads than sequences in the reference genome. Defaults to 1.
show_progress	Boolean for whether to show a progress bar during processing. Defaults to FALSE.

Examples

```
r <- create_genome(10, 1000)
v_phylo <- create_variants(r, vars_phylo(ape::rcoal(5)), sub_JC69(0.1))
v_theta <- create_variants(r, vars_theta(0.001, 5), sub_K80(0.1, 0.2))
```

---

evo_rates *Table of evolutionary rates.*

Description

From Table 1 in Sung et al. (2016).

Usage

```
evo_rates
```

Format

A data frame with 15 rows and 8 variables:

`domain` Either `Bacteria` or `Eukarya` for what type of organism the species is.

`species` Species name.

`Ge` Effective genome size using only coding DNA.

`Gc_Gnc` Effective genome size using coding DNA and non-coding DNA that is under purifying selection.

`indels` Rate of insertions and deletions (10^{-10} events per site per generation).

`subs` Base-substitution mutation rate (10^{-10} events per site per generation).

`Ne` Effective population size ($\times 10^6$).

`theta_s` Population mutation rate estimated using θ_s.

`pi_s` Population mutation rate estimated using π_s.

Source

http://dx.doi.org/10.1534/g3.116.030890

References

Sung, W., M. S. Ackerman, M. M. Dillon, T. G. Platt, C. Fuqua, V. S. Cooper, and M. Lynch. 2016. Evolution of the insertion-deletion mutation rate across the tree of life. *G3: Genes | Genomes | Genetics* **6**:2583–2591.

`illumina` *Create and write Illumina reads to FASTQ file(s).*

Description

From either a reference genome or set of haploid variants, create Illumina reads from error profiles and write them to FASTQ output file(s). I encourage you to cite the reference below in addition to `jackalope` if you use this function.

Usage

```
illumina(seq_object, out_prefix, n_reads, read_length, paired,
  frag_mean = 400, frag_sd = 100, matepair = FALSE, seq_sys = NULL,
  profile1 = NULL, profile2 = NULL, ins_prob1 = 9e-05,
  del_prob1 = 0.00011, ins_prob2 = 0.00015, del_prob2 = 0.00023,
  frag_len_min = NULL, frag_len_max = NULL, variant_probs = NULL,
  barcodes = NULL, prob_dup = 0.02, sep_files = FALSE,
  compress = FALSE, comp_method = "bgzip", n_threads = 1L,
  read_pool_size = 1000L, show_progress = FALSE, overwrite = FALSE)
```

Arguments

seq_object	Sequencing object of class ref_genome or variants.
out_prefix	Prefix for the output file(s), including entire path except for the file extension.
n_reads	Number of reads you want to create.
read_length	Length of reads.
paired	Logical for whether to use paired-end reads. This argument is changed to TRUE if matepair is TRUE.
frag_mean	Mean of the Gamma distribution that generates fragment sizes. Defaults to 400.
frag_sd	Standard deviation of the Gamma distribution that generates fragment sizes. Defaults to 100.
matepair	Logical for whether to simulate mate-pair reads. Defaults to FALSE.
seq_sys	Full or abbreviated name of sequencing system to use. See "Sequencing systems" section for options. See "Sequencing profiles" section for more information on how this argument, profile1, and profile2 are used to specify profiles. Defaults to NULL.
profile1	Custom profile file for read 1. See "Sequencing profiles" section for more information on how this argument, profile2, and seq_sys are used to specify profiles. Defaults to NULL.
profile2	Custom profile file for read 2. See "Sequencing profiles" section for more information on how this argument, profile1, and seq_sys are used to specify profiles. Defaults to NULL.
ins_prob1	Insertion probability for read 1. Defaults to 0.00009.
del_prob1	Deletion probability for read 1. Defaults to 0.00011.
ins_prob2	Insertion probability for read 2. Defaults to 0.00015.
del_prob2	Deletion probability for read 2. Defaults to 0.00023.
frag_len_min	Minimum fragment size. A NULL value results in the read length. Defaults to NULL.
frag_len_max	Maximum fragment size. A NULL value results in 2^32-1, the maximum allowed value. Defaults to NULL
variant_probs	Relative probability of sampling each variant. This is ignored if sequencing a reference genome. NULL results in all having the same probability. Defaults to NULL.
barcodes	Character vector of barcodes for each variant, or a single barcode if sequencing a reference genome. NULL results in no barcodes. Defaults to NULL.
prob_dup	A single number indicating the probability of duplicates. Defaults to 0.02.
sep_files	Logical indicating whether to make separate files for each variant. This argument is coerced to FALSE if the seq_object argument is not a variants object. Defaults to FALSE.
compress	Logical specifying whether or not to compress output file, or an integer specifying the level of compression, from 1 to 9. If TRUE, a compression level of 6 is used. Defaults to FALSE.

comp_method	Character specifying which type of compression to use if any is desired. Options include "gzip" and "bgzip". This is ignored if compress is FALSE, and it throws an error if it's set to "gzip" when n_threads > 1 (since I don't have a method to do gzip compression in parallel). Defaults to "bgzip".
n_threads	The number of threads to use in processing. If compress is TRUE or > 0 (indicating compressed output), setting n_threads to 2 or more makes this function first create an uncompressed file/files using n_threads threads, then compress that/those file/files also using n_threads threads. There is no speed increase if you try to use multiple threads to create compressed output on the fly, so that option is not included. If you want to be conservative with disk space (by not having an uncompressed file present even temporarily), set n_threads to 1. This argument is ignored if the package was not compiled with OpenMP. Defaults to 1.
read_pool_size	The number of reads to store before writing to disk. Increasing this number should improve speed but take up more memory. Defaults to 1000.
show_progress	Logical for whether to show a progress bar. Defaults to FALSE.
overwrite	Logical for whether to overwrite existing FASTQ file(s) of the same name, if they exist.

Value

Nothing is returned.

Sequencing profiles

This section outlines how to use the seq_sys, profile1, and profile2 arguments. If all arguments are NULL (their defaults), a sequencing system is chosen based on the read length. If, however, one or more arguments has been provided, then how they're provided should depend on whether you want single- or paired-end reads.

For single-end reads

- profile2 should be NULL.
- Only seq_sys or profile1 should be provided, not both.

For paired-end reads

- If providing seq_sys, don't provide either profile1 or profile2.
- If providing profile1, you must also provide profile2 (they can be the same if you want) and you cannot provide seq_sys.

Sequencing systems

Sequencing system options are the following, where, for each system, "name" is the full name, "abbrev" is the abbreviated name, "max_len" indicates the maximum allowed read length, and "paired" indicates whether paired-end sequencing is allowed.

name	abbrev	max_len	paired
Genome Analyzer I	GA1	44	Yes

Genome Analyzer II	GA2	75	Yes
HiSeq 1000	HS10	100	Yes
HiSeq 2000	HS20	100	Yes
HiSeq 2500	HS25	150	Yes
HiSeqX v2.5 PCR free	HSXn	150	Yes
HiSeqX v2.5 TruSeq	HSXt	150	Yes
MiniSeq TruSeq	MinS	50	No
MiSeq v1	MSv1	250	Yes
MiSeq v3	MSv3	250	Yes
NextSeq 500 v2	NS50	75	Yes

ID lines

The ID lines for FASTQ files are formatted as such:

`@<genome name>-<sequence name>-<starting position>-<strand>[/<read#>]`

where the part in `[]` is only for paired-end Illumina reads, and where `genome name` is always `REF` for reference genomes (as opposed to variants).

References

Huang, W., L. Li, J. R. Myers, and G. T. Marth. 2012. ART: a next-generation sequencing read simulator. *Bioinformatics* **28**:593–594.

Examples

```
rg <- create_genome(10, 100e3, 100)
illumina(rg, "illumina_reads", n_reads = 100,
         read_length = 100, paired = FALSE)
```

---

indels *Insertions and deletions (indels) specification*

Description

Construct necessary information for insertions and deletions (indels) that will be used in `create_variants`.

Usage

```
indels(rate, max_length = 10, a = NULL, rel_rates = NULL)
```

Arguments

rate	Single number specifying the overall indel rate among all lengths.
max_length	Maximum length of indels. Defaults to 10.
a	Extra parameter necessary for generating rates from a Lavalette distribution. See Details for more info. Defaults to NULL.
rel_rates	A numeric vector of relative rates for each indel length from 1 to the maximum length. If provided, all arguments other than rate are ignored. Defaults to NULL.

Details

Both insertions and deletions require the rate parameter, which specifies the overall insertion/deletion rate among all lengths. The rate parameter is ultimately combined with a vector of relative rates among the different lengths of insertions/deletions from 1 to the maximum possible length. There are three different ways to specify/generate relative-rate values.

1. Assume that rates are proportional to exp(-L) for indel length L from 1 to the maximum length (Albers et al. 2011). This method will be used if the following arguments are provided:
   - rate
   - max_length
2. Generate relative rates from a Lavalette distribution (Fletcher and Yang 2009), where the rate for length L is proportional to {L * max_length / (max_length - L + 1)}^(-a). This method will be used if the following arguments are provided:
   - rate
   - max_length
   - a
3. Directly specify values by providing a numeric vector of relative rates for each insertion/deletion length from 1 to the maximum length. This method will be used if the following arguments are provided:
   - rate
   - rel_rates

Value

An indel_rates object, which is just a wrapper around a numeric vector. You can access the rates vector for indel_rates object x by running as.numeric(x).

References

Albers, C. A., G. Lunter, D. G. MacArthur, G. McVean, W. H. Ouwehand, and R. Durbin. 2011. Dindel: accurate indel calls from short-read data. Genome Research 21:961–973.

Fletcher, W., and Z. Yang. 2009. INDELible: a flexible simulator of biological sequence evolution. Molecular Biology and Evolution 26:1879–1888.

Examples

```
# relative rates are proportional to `exp(-L)` for indel
# length `L` from 1 to 5:
indel_rates1 <- indels(0.1, max_length = 5)

# relative rates are proportional to Lavalette distribution
# for length from 1 to 10:
indel_rates2 <- indels(0.2, max_length = 10, a = 1.1)

# relative rates are all the same for lengths from 1 to 100:
indel_rates3 <- indels(0.2, rel_rates = rep(1, 100))
```

`jackalope` *jackalope: An efficient, flexible molecular evolution and sequencing simulator.*

Description

`jackalope` simply and efficiently simulates (i) variants from reference genomes and (ii) reads from both Illumina and Pacific Biosciences (PacBio) platforms. It can either read reference genomes from FASTA files or simulate new ones. Genomic variants can be simulated using summary statistics, phylogenies, Variant Call Format (VCF) files, and coalescent simulations—the latter of which can include selection, recombination, and demographic fluctuations. `jackalope` can simulate single, paired-end, or mate-pair Illumina reads, as well as reads from Pacific Biosciences These simulations include sequencing errors, mapping qualities, multiplexing, and optical/PCR duplicates. All outputs can be written to standard file formats.

`pacbio` *Create and write PacBio reads to FASTQ file(s).*

Description

From either a reference genome or set of haploid variants, create PacBio reads and write them to FASTQ output file(s). I encourage you to cite the reference below in addition to `jackalope` if you use this function.

Usage

```
pacbio(seq_object, out_prefix, n_reads, chi2_params_s = c(0.01214, -5.12,
  675, 48303.0732881, 1.46910512123303), chi2_params_n = c(0.00189237136,
  2.5394497, 5500), max_passes = 40, sqrt_params = c(0.5, 0.2247),
  norm_params = c(0, 0.2), prob_thresh = 0.2, ins_prob = 0.11,
  del_prob = 0.04, sub_prob = 0.01, min_read_length = 50,
```

```
  lognorm_read_length = c(0.200110276521, -10075.4363813, 17922.611306),
  custom_read_lengths = NULL, prob_dup = 0, variant_probs = NULL,
  sep_files = FALSE, compress = FALSE, comp_method = "bgzip",
  n_threads = 1L, read_pool_size = 100L, show_progress = FALSE,
  overwrite = FALSE)
```

Arguments

seq_object	Sequencing object of class ref_genome or variants.
out_prefix	Prefix for the output file(s), including entire path except for the file extension.
n_reads	Number of reads you want to create.
chi2_params_s	Vector containing the 5 parameters for the curve determining the scale parameter for the chi^2 distribution. Defaults to c(0.01214, -5.12, 675, 48303.0732881, 1.469105121233026
chi2_params_n	Vector containing the 3 parameters for the function determining the n parameter for the chi^2 distribution. Defaults to c(0.00189237136, 2.53944970, 5500).
max_passes	Maximal number of passes for one molecule. Defaults to 40.
sqrt_params	Vector containing the 2 parameters for the square root function for the quality increase. Defaults to c(0.5, 0.2247).
norm_params	Vector containing the 2 parameters for normal distributed noise added to quality increase square root function Defaults to c(0, 0.2).
prob_thresh	Upper bound for the modified total error probability. Defaults to 0.2.
ins_prob	Probability for insertions for reads with one pass. Defaults to 0.11.
del_prob	Probability for deletions for reads with one pass. Defaults to 0.04.
sub_prob	Probability for substitutions for reads with one pass. Defaults to 0.01.
min_read_length	Minium read length for lognormal distribution. Defaults to 50.
lognorm_read_length	Vector containing the 3 parameters for lognormal read length distribution. Defaults to c(0.200110276521, -10075.4363813, 17922.611306).
custom_read_lengths	Sample read lengths from a vector or column in a matrix; if a matrix, the second column specifies the sampling weights. If NULL, it samples read lengths from the lognormal distribution using parameters in lognorm_read_length. Defaults to NULL.
prob_dup	A single number indicating the probability of duplicates. Defaults to 0.0.
variant_probs	Relative probability of sampling each variant. This is ignored if sequencing a reference genome. NULL results in all having the same probability. Defaults to NULL.
sep_files	Logical indicating whether to make separate files for each variant. This argument is coerced to FALSE if the seq_object argument is not a variants object. Defaults to FALSE.
compress	Logical specifying whether or not to compress output file, or an integer specifying the level of compression, from 1 to 9. If TRUE, a compression level of 6 is used. Defaults to FALSE.

comp_method — Character specifying which type of compression to use if any is desired. Options include "gzip" and "bgzip". This is ignored if compress is FALSE, and it throws an error if it's set to "gzip" when n_threads > 1 (since I don't have a method to do gzip compression in parallel). Defaults to "bgzip".

n_threads — The number of threads to use in processing. If compress is TRUE or > 0 (indicating compressed output), setting n_threads to 2 or more makes this function first create an uncompressed file/files using n_threads threads, then compress that/those file/files also using n_threads threads. There is no speed increase if you try to use multiple threads to create compressed output on the fly, so that option is not included. If you want to be conservative with disk space (by not having an uncompressed file present even temporarily), set n_threads to 1. This argument is ignored if the package was not compiled with OpenMP. Defaults to 1.

read_pool_size — The number of reads to store before writing to disk. Increasing this number should improve speed but take up more memory. Defaults to 100.

show_progress — Logical for whether to show a progress bar. Defaults to FALSE.

overwrite — Logical for whether to overwrite existing FASTQ file(s) of the same name, if they exist.

Value

Nothing is returned.

ID lines

The ID lines for FASTQ files are formatted as such:

@<genome name>-<sequence name>-<starting position>-<strand>

where genome name is always REF for reference genomes (as opposed to variants).

References

Stöcker, B. K., J. Köster, and S. Rahmann. 2016. SimLoRD: simulation of long read data. *Bioinformatics* **32**:2704–2706.

Examples

```
rg <- create_genome(10, 100e3, 100)
pacbio(rg, "pacbio_reads", n_reads = 100)
```

read_fasta *Read a fasta file.*

Description

Accepts uncompressed and gzipped fasta files.

Usage

```
read_fasta(fasta_files, fai_files = NULL, cut_names = FALSE)
```

Arguments

fasta_files	File name(s) of the fasta file(s).
fai_files	File name(s) of the fasta index file(s). Providing this argument speeds up the reading process significantly. If this argument is provided, it must be the same length as the fasta_files argument. Defaults to NULL, which indicates the fasta file(s) is/are not indexed.
cut_names	Boolean for whether to cut sequence names at the first space. This argument is ignored if fai_file is not NULL. Defaults to FALSE.

Value

A ref_genome object.

ref_genome *An R6 class representing a reference genome.*

Description

Note: This class wraps a pointer to a C++ object, so do NOT change fields in this class directly. It will cause your R session to do bad things. (Ever seen the bomb popup on RStudio? Manually mess with these fields and you surely will.) For safe ways of manipulating the reference genome, see the "Methods" section.

Usage

```
ref_genome
```

Format

An R6Class generator object

Value

An object of class ref_genome.

Fields

`genome` An `externalptr` to a C++ object storing the sequences representing the genome.

Methods

Viewing information:

`n_seqs()` View the number of sequences.

`sizes()` View vector of sequence sizes.

`names()` View vector of sequence names.

`sequence(seq_ind)` View a sequence string based on an index, `seq_ind`.

`gc_prop(seq_ind, start, end)` View the GC proportion for a range within a reference sequence.

`nt_prop(nt, seq_ind, start, end)` View the proportion of a range within a reference sequence that is of nucleotide `nt`.

Editing information:

`set_names(new_names)` Set names for all sequences. `new_names` is a character vector of what to change names to, and it must be the same length as the # sequences.

`clean_names()` Clean sequence names, converting `" :;=%,\|/\"\'"` to `"_"`.

`add_seqs(new_seqs, new_names = NULL)` Add one or more sequences directly. They can optionally be named (using `new_names`). Otherwise, their names are auto-generated.

`rm_seqs(seq_names)` Remove one or more sequences based on names in the `seq_names` vector.

`merge_seqs()` Merge all sequences into one after first shuffling their order.

`filter_seqs(threshold, method)` Filter sequences by size (`method = "size"`) or for a proportion of total bases (`method = "prop"`). For the latter, sequences are first size-sorted, then the largest `N` sequences are retained that allow at least `threshold * sum(<all sequence sizes>)` base pairs remaining after filtering.

`replace_Ns(pi_tcag, n_threads = 1, show_progress = FALSE)` Replace `Ns` in reference sequence with nucleotides sampled with probabilities given in `pi_tcag`. You can optionally use multiple threads (`n_threads` argument) and/or show a progress bar (`show_progress`).

See Also

`read_fasta` `create_genome`

site_var *Specify variation in mutation rates among sites*

Description

Construct necessary information for among-site variation in mutation rates that will be used in `create_variants`.

Usage

```
site_var(reference, shape = NULL, region_size = NULL, invariant = 0,
  mats = NULL, out_prefix = NULL, compress = FALSE,
  comp_method = "bgzip")
```

Arguments

`reference`	A `ref_genome` object from which you will eventually generate variants.
`shape`	Shape parameter for the Gamma distribution that generates gamma distances, The variance of the distribution is `1 / shape`, and its mean is fixed to 1. Values `<= 0` are not allowed. Defaults to `NULL`.
`region_size`	Size of regions to break the genome into, where all sites within a region have the same gamma distance. Defaults to `NULL`.
`invariant`	Proportion of regions that are invariant. Must be in range `[0,1)`. Defaults to `0`.
`mats`	List of matrices, one for each sequence in the genome. Each matrix should have two columns. The first should contain the end points for each region. The second should contain the gamma distances for each region. Note that if gamma distances don't have a mean (weighted by sequence length for each gamma-distance value) equal to 1, you're essentially changing the overall mutation rate. If this argument is provided, `shape` and `region_size` are ignored. Defaults to `NULL`.
`out_prefix`	String specifying the file name prefix for an output BED file that will be generated by this function and that will specify the gamma distances for each region. If `NULL`, no output file is produced. Defaults to `NULL`.
`compress`	Logical specifying whether or not to compress output file, or an integer specifying the level of compression, from 1 to 9. If `TRUE`, a compression level of `6` is used. Defaults to `FALSE`.
`comp_method`	Character specifying which type of compression to use if any is desired. Options include `"gzip"` and `"bgzip"`. This is ignored if `compress` is `FALSE`. Defaults to `"bgzip"`.

Details

A site's deviance from the average mutation rate is determined by its "gamma distance". A site's overall mutation rate is the mutation rate for that nucleotide (substitution + indel) multiplied by the site's gamma distance. There are two options for specifying gamma distances:

1. Generate gamma distances from a Gamma distribution. This method will be used if the shape and region_size arguments are provided. If the mats argument is also provided, this method will NOT be used. See argument descriptions for more info.
2. Manually input matrices that specify the gamma distance and end points for regions each gamma distance refers to. This method will be used if the mats argument is provided. See argument descriptions for more info.

Value

A site_var_mats object, which is a wrapper around a list of matrices, one for each sequence in the reference genome. Although the print method is different, you can otherwise treat these objects the same as you would a list (e.g., x[[1]], x[1:2], length(x)).

Examples

```
ref <- create_genome(3, 100)
# generating from Gamma distribution
gamma_mats <- site_var(ref, shape = 0.5,
                       region_size = 5)
# with custom matrices
gamma_mats <- site_var(ref,
                       mats = replicate(3,
                           cbind(seq(10, 100, 10),
                           rgamma(10, 0.9)),
                           simplify = FALSE))
```

---

sub_models *Construct necessary information for substitution models.*

Description

For a more detailed explanation, see vignette("sub-models").

Usage

```
sub_TN93(pi_tcag, alpha_1, alpha_2, beta)

sub_JC69(lambda)

sub_K80(alpha, beta)

sub_F81(pi_tcag)

sub_HKY85(pi_tcag, alpha, beta)

sub_F84(pi_tcag, beta, kappa)
```

```
sub_GTR(pi_tcag, abcdef)

sub_UNREST(Q)
```

Arguments

pi_tcag	Vector of length 4 indicating the equilibrium distributions of T, C, A, and G respectively. Values must be >= 0, and they are forced to sum to 1.
alpha_1	Substitution rate for T <-> C transition.
alpha_2	Substitution rate for A <-> G transition.
beta	Substitution rate for transversions.
lambda	Substitution rate for all possible substitutions.
alpha	Substitution rate for transitions.
kappa	The transition/transversion rate ratio.
abcdef	A vector of length 6 that contains the off-diagonal elements for the substitution rate matrix. See `vignette("sub-models")` for how the values are ordered in the matrix.
Q	Matrix of substitution rates for "T", "C", "A", and "G", respectively. Item `Q[i,j]` is the rate of substitution from nucleotide `i` to nucleotide `j`. Do not include indel rates here! Values on the diagonal are calculated inside the function so are ignored.

Value

A `sub_model_info` object, which is just a wrapper around a list with fields `Q` and `pi_tcag`. The former has the rate matrix, and the latter has the equilibrium nucleotide densities for "T", "C", "A", and "G", respectively. Access the rate matrix for a `sub_model_info` object named `x` via `x$Q` and densities via `x$pi_tcag`.

Functions

- `sub_TN93`: TN93 model.
- `sub_JC69`: JC69 model.
- `sub_K80`: K80 model.
- `sub_F81`: F81 model.
- `sub_HKY85`: HKY85 model.
- `sub_F84`: F84 model.
- `sub_GTR`: GTR model.
- `sub_UNREST`: UNREST model.

See Also

`create_variants`

Examples

```
# Same substitution rate for all types:
Q_JC69 <- sub_JC69(lambda = 0.1)

# Transitions 2x more likely than transversions:
Q_K80 <- sub_K80(alpha = 0.2, beta = 0.1)

# Same as above, but incorporating equilibrium frequencies
sub_HKY85(pi_tcag = c(0.1, 0.2, 0.3, 0.4),
          alpha = 0.2, beta = 0.1)
```

---

variants *An R6 class representing haploid variants from a reference genome.*

Description

Note: This class wraps a pointer to a C++ object, so do NOT change fields in this class directly. It will cause your R session to do bad things. (Ever seen the bomb popup on RStudio? Manually mess with these fields and you surely will.) For safe ways of manipulating the variants' information, see the "Methods" section.

Usage

```
variants
```

Format

An R6Class generator object

Value

An object of class `variants`.

Fields

`genome` An `externalptr` to a C++ object storing the sequences representing the genome.

`reference` An `externalptr` to a C++ object storing the sequences representing the genome. This field is private, so you can't view it, but I'm listing it here so that I can provide a few extra notes about it:

- **This point is the most important.** Since it's a pointer, if you make any changes to the reference genome that it points to, those changes will also show up in the `variants` object. For example, if you make a `variants` object named `V` based on an existing `ref_genome` object named `R`, then you merge sequences in `R`, `V` will now have merged sequences. If you've already started adding mutations to `V`, then all the indexes used to store those mutations will be inaccurate. So when you do anything with `V` later, your R session will crash or have errors.

- If a ref_genome object is used to create a variants object, deleting the ref_genome object won't cause issues with the variants object. However, the variants class doesn't provide methods to edit sequences, so only remove the ref_genome object when you're done editing the reference genome.

Methods

Viewing information:

`n_seqs()` View the number of sequences.

`n_vars()` View the number of variants.

`sizes(var_ind)` View vector of sequence sizes for a given variant.

`seq_names()` View vector of sequence names.

`var_names()` View vector of variant names.

`sequence(var_ind, seq_ind)` View a sequence string based on indices for the sequence (`seq_ind`) and variant (`var_ind`).

`gc_prop(var_ind, seq_ind, start, end)` View the GC proportion for a range within a variant sequence.

`nt_prop(nt, var_ind, seq_ind, start, end)` View the proportion of a range within a variant sequence that is of nucleotide `nt`.

Editing information:

`set_names(new_names)` Set names for all variants. `new_names` is a character vector of what to change names to, and it must be the same length as the # variants.

`add_vars(new_names)` Add new, named variant(s) to the object. These variants will have no mutations. If you want to add new variants with mutations, either re-run `create_variants` or use the `dup_vars` method to duplicate existing variants.

`dup_vars(var_names, new_names = NULL)` Duplicate existing variant(s) based on their name(s). You can optionally specify the names of the duplicates (using `new_names`). Otherwise, their names are auto-generated.

`rm_vars(var_names)` Remove one or more variants based on names in the `var_names` vector.

`add_sub(var_ind, seq_ind, pos, nt)` Manually add a substitution for a given variant (`var_ind`), sequence (`seq_ind`), and position (`pos`). The reference nucleotide will be changed to `nt`, which should be a single character.

`add_ins(var_ind, seq_ind, pos, nts)` Manually add an insertion for a given variant (`var_ind`), sequence (`seq_ind`), and position (`pos`). The nucleotide(s) `nts` will be inserted after the designated position.

`add_del(var_ind, seq_ind, pos, n_nts)` Manually add a deletion for a given variant (`var_ind`), sequence (`seq_ind`), and position (`pos`). The designated number of nucleotides to delete (`n_nts`) will be deleted starting at `pos`, unless `pos` is near the sequence end and doesn't have `n_nts` nucleotides to remove; it simply stops at the sequence end in this case.

See Also

`create_variants`

vars_functions *Organize higher-level information for creating variants.*

Description

The following functions organize information that gets passed to `create_variants` to generate variants from a reference genome. Each function represents a method of generation and starts with `"vars_"`. The first three are phylogenomic methods, and all functions but `vars_vcf` will use molecular evolution information when passed to `create_variants`.

Details

`vars_theta` Uses an estimate for theta, the population-scaled mutation rate, and a desired number of variants.

`vars_phylo` Uses phylogenetic tree(s) from `phylo` object(s) or NEWICK file(s), one tree per sequence or one for all sequences.

`vars_gtrees` Uses gene trees, either in the form of an object from the `scrm` or `coala` package or a file containing output in the style of the `ms` program.

`vars_ssites` Uses matrices of segregating sites, either in the form of `scrm` or `coala` coalescent-simulator object(s), or (2) a `ms`-style output file.

`vars_vcf` Uses a variant call format (VCF) file that directly specifies variants. This method does not work if the `vcfR` package isn't installed.

See Also

`create_variants`

---

vars_gtrees *Create necessary information to create variants using gene trees*

Description

This function organizes higher-level information for creating variants from gene trees output from coalescent simulations.

Usage

```
vars_gtrees(obj = NULL, fn = NULL)
```

Arguments

obj	Object containing gene trees. This can be one of the following: (1) A single list with a trees field inside. This field must contain a set of gene trees for each sequence. (2) A list of lists, each sub-list containing a trees field of length 1. The top-level list must be of the same length as the number of sequences. Defaults to NULL.
fn	A single string specifying the name of the file containing the ms-style coalescent output with gene trees. Defaults to NULL.

Details

Using the obj argument is designed after the trees fields in the output from the scrm and coala packages. (These packages are not required to be installed when installing jackalope.) To get gene trees, make sure to add + sumstat_trees() to the coalmodel for coala, or make sure that "-T" is present in args for scrm.

If using an output file from a command-line program like ms/msms, add the -T option.

Value

A vars_gtrees_info object containing information used in create_variants to create haploid variants. This class is just a wrapper around a list of NEWICK tree strings, one for each gene tree.

---

vars_phylo *Create necessary information to create variants using phylogenetic tree(s)*

Description

This function organizes higher-level information for creating variants from phylogenetic tree(s) output as phylo or multiPhylo objects (both from the ape package) or NEWICK files.

Usage

```
vars_phylo(obj = NULL, fn = NULL)
```

Arguments

obj	Object containing phylogenetic tree(s). This can be (1) a single phylo object that represents all sequences in the genome or (2) a list or multiPhylo object containing a phylo object for each reference sequence. In the latter case, phylogenies will be assigned to sequences in the order provided. Defaults to NULL.
fn	One or more string(s), each of which specifies the file name of a NEWICK file containing a phylogeny. If one name is provided, that phylogeny will be used for all sequences. If more than one is provided, there must be a phylogeny for each reference genome sequence, and phylogenies will be assigned to sequences in the order provided. Defaults to NULL.

Value

A `vars_phylo_info` object containing information used in `create_variants` to create haploid variants. This class is just a wrapper around a list containing phylogenetic tree information for each reference sequence.

`vars_ssites` *Create necessary information to create variants using segregating sites matrices*

Description

This function organizes higher-level information for creating variants from matrices of segregating sites output from coalescent simulations.

Usage

```
vars_ssites(obj = NULL, fn = NULL)
```

Arguments

`obj`	Object containing segregating sites information. This can be one of the following: (1) A single `list` with a `seg_sites` field inside. This field must contain a matrix for segregating sites for each sequence. The matrix itself should contain the haplotype information, coded using 0s and 1s: 0s indicate the ancestral state and 1s indicate mutant. The matrix column names should indicate the positions of the polymorphisms on the chromosome. If positions are in the range `(0,1)`, they're assumed to come from an infinite- sites model and are relative positions. If positions are integers in the range `[0, sequence length - 1]` or `[1, sequence length]`, they're assumed to come from an finite-sites model and are absolute positions. Defaults to `NULL`.
`fn`	A single string specifying the name of the file containing the `ms`-style coalescent output with segregating site info. Defaults to `NULL`.

Details

For what the `seg_sites` field should look like in a list, see output from the `scrm` or `coala` package. (These packages are not required to be installed when installing `jackalope`.)

Value

A `vars_ssites_info` object containing information used in `create_variants` to create haploid variants. This class is just a wrapper around a list of matrices of segregating site info.

vars_theta *Create necessary information to create variants using theta parameter*

Description

This function organizes higher-level information for creating variants from the population-scaled mutation rate and a desired number of variants.

Usage

```
vars_theta(theta, n_vars)
```

Arguments

theta	Population-scaled mutation rate.
n_vars	Mumber of desired variants.

Value

A vars_theta_info object containing information used in create_variants to create haploid variants. This class is just a wrapper around a list containing the phylogenetic tree and theta parameter.

vars_vcf *Create necessary information to create variants using a VCF file*

Description

This function organizes higher-level information for creating variants from Variant Call Format (VCF) files.

Usage

```
vars_vcf(fn, print_names = FALSE, ...)
```

Arguments

fn	A single string specifying the name of the VCF file
print_names	Logical for whether to print all unique sequence names from the VCF file when VCF sequence names don't match those from the reference genome. This printing doesn't happen until this object is passed to create_variants. This can be useful for troubleshooting. Defaults to FALSE.
...	Arguments to pass to vcfR::read.vcfR, excluding the file argument that will be overridden with the fn argument to this function.

Details

This function won't work if the package vcfR isn't installed.

Value

A vars_vcf_info object containing information used in create_variants to create haploid variants. This class is just a wrapper around a list containing relevant output from vcfR::read.vcfR: haplotypes, reference sequences, positions, sequence names, and variant names.

---

write_fasta *Write a sequence object to a FASTA file.*

Description

This file produces 1 FASTA file for a ref_genome object and one file for each variant in a variants object.

Usage

```
write_fasta(seq_obj, out_prefix, compress = FALSE,
  comp_method = "bgzip", text_width = 80, show_progress = FALSE,
  n_threads = 1, overwrite = FALSE)
```

Arguments

seq_obj	A ref_genome or variants object.
out_prefix	Prefix for the output file.
compress	Logical specifying whether or not to compress output file, or an integer specifying the level of compression, from 1 to 9. If TRUE, a compression level of 6 is used. Defaults to FALSE.
comp_method	Character specifying which type of compression to use if any is desired. Options include "gzip" and "bgzip". This is ignored if compress is FALSE. Defaults to "bgzip".
text_width	The number of characters per line in the output fasta file. Defaults to 80.
show_progress	Logical for whether to show a progress bar. Defaults to FALSE.
n_threads	Number of threads to use if writing from a variants object. Threads are split among variants, so it's not useful to provide more threads than variants. This argument is ignored if seq_obj is a ref_genome object, or if OpenMP is not enabled. Defaults to 1.
overwrite	Logical for whether to overwrite existing file(s) of the same name, if they exist. Defaults to FALSE.

Value

NULL

write_vcf *Write variant info from a* variants *object to a VCF file.*

Description

Compression in this function always uses "bgzip" for compatibility with "tabix".

Usage

```
write_vcf(vars, out_prefix, compress = FALSE, sample_matrix = NULL,
  show_progress = FALSE, overwrite = FALSE)
```

Arguments

vars	A variants object.
out_prefix	Prefix for the output file.
compress	Logical specifying whether or not to compress output file, or an integer specifying the level of compression, from 1 to 9. If TRUE, a compression level of 6 is used. Defaults to FALSE.
sample_matrix	Matrix to specify how haploid variants are grouped into samples if samples are not haploid. There should be one row for each sample, and each row should contain indices or names for the variants present in that sample. Indices/names for variants cannot be repeated. Instead of repeating indices here, you should use the dup_vars method of the variants class to duplicate the necessary variant(s). The number of columns indicates the ploidy level: 2 columns for diploid, 3 for triploid, 4 for tetraploid, and so on; there is no limit to the ploidy level. If this argument is NULL, it's assumed that each variant is its own separate sample. Defaults to NULL.
show_progress	Logical for whether to show a progress bar. Defaults to FALSE.
overwrite	Logical for whether to overwrite existing file(s) of the same name, if they exist. Defaults to FALSE.

Value

NULL

Index