Package: polySimIBD (via r-universe)

June 8, 2024

Type Package

Title Structured Wright Fisher Simulator for Malaria Genetics

Version 1.1.0

Description Unique features of the malaria life-cycle and transmission dynamics requires extensions of typical population genetic simulators. Using a spatial discrete-loci, discrete-time structured Wright Fisher framework, we simulate malaria population genetics forwards in time. Users are then able to capture the full Ancestral Recombination Graph.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

URL https://github.com/nickbrazeau/polySimIBD

BugReports https://github.com/nickbrazeau/polySimIBD/issues

Imports ggplot2, magrittr, purrr, Rcpp, methods, RColorBrewer, grDevices, goodegg

Suggests tibble, tidygraph, dplyr, knitr, rmarkdown, covr, testthat

Remotes nickbrazeau/goodegg,

VignetteBuilder knitr

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

LinkingTo Rcpp

SystemRequirements C++11

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

RemoteUrl https://github.com/nickbrazeau/polySimIBD

RemoteRef HEAD

RemoteSha 2bf75b73bcf9170a718f2766ad0b3aeb7df7f47f

Contents

`argraph` *The Ancestral Recombination Graph This S3 class represents the ARG from the realized simulation*

Description

A list of `bv_tree` for each discrete-loci, which constitutes the ARG

`bvtree` *A Simplified Tree: bvtree This S3 class represents the bvtree which is a simple tree marginal representation*

Description

The `bv_tree` class is a lightweight representation of a marginal tree

Fields

`c` vector; the node connection for each haplotype (each haplotype is an element in a vector)

`t` vector; the timing of the node connection (time to MRCA)

`z` vector; the order of coalescence for each set of haplotypes

bvtreeToNewick *Convert* bvtree *to Newick Tree Format*

Description

Recursively converts bvtree to Newick tree format for compatibility with other downstream packages. The output format is for all leafs to be named and distances, which are the coalescent times, are preserved. Additonally, clades have the total tree length explicitly stated versus the traditional iterative shorthand of some Newick examples.

Usage

```
bvtreeToNewick(bvtree, tlim = 10)
```

Arguments

bvtree	S3 class; internal class for the polySimIBD: package bvtree
tlim	numeric; the maximum number of generations to consider before exiting gracefully if all samples have not coalesced

Details

Note, the overall TMRCA is set to the tlim, which in Newick format looks inappropriately rooted.

Note, the function does not "know" the original tlim that was specified in the forward-simulation, and thus must be re-stated by the user.

Value

Newick String

get_arg *Get ancestral recombination graph from forward simulations*

Description

Given an object swf, which is the result of forward simulation using the function sim_swf(), walks backwards through the ancestry and calculates the coalescent tree at every locus for the specified hosts and/or haplotypes.

Usage

```
get_arg(swf, host_index = NULL, haplo_index = NULL)
```

Arguments

swf	result of forwards simulation using the function sim_swf()
host_index	a vector of target hosts. Defaults to all hosts
haplo_index	a list of target haplotypes within the hosts specified by host_index. Defaults to all haplotypes within the specified hosts

---

get_bvibd *Get Between-Host Identity by Descent from forward simulations*

Description

Given an object swf, which is the result of forward simulation using the function sim_swf(), walks backwards through the ancestry and calculates the between host identity by descent. Calculation is based on *Verity et. al 2020, Nat Comms, PMC7192906.*

Usage

```
get_bvibd(swf, host_index = NULL, haplo_index = NULL)
```

Arguments

swf	result of forwards simulation using the function sim_swf()
host_index	a vector of target hosts. Defaults to all hosts
haplo_index	a list of target haplotypes within the hosts specified by host_index. Defaults to all haplotypes within the specified hosts

---

get_effective_coi *Extract Effective COI by Loci from SWF Simulation for a Single Host*

Description

From a single host in a SWF Simulation, extract the effective COI for loci within the ARG. Effective COI is defined as the number of non-coalesced genomes at the end of tlim. For example, if a host contains five strains (i.e. five parasites), then at loci 1 we can determine "mini" coalescent trees have occurred. If the effective COI is 2, then three strains have formed one coalescent tree and two strains have formed a separate coalescent tree. Notably, this framework is independent for each recombination event (versus loci which are just genetic markers). This means, if loci 10 is located within a different recombination block than loci 1 - and therefore is a separate entry in the ARG - we expect the effective COI to be (potentially) different.

Usage

```
get_effective_coi(swf, host_index = NULL)
```

Arguments

`swf`	result of forwards simulation using the function `sim_swf()`
`host_index`	a vector of target hosts. Defaults to all hosts

Details

Function limited to a single host per "realization"

Value

vector of effective COI by loci

`get_within_ibd` *Calculate Within-Host IBD*

Description

The within-host IBD is calculated as the number of strains that have coalesced within the `tlim` at each loci divided by the original (i.e. not effective) COI. As an example, consider that there are three strains (i.e. parasites) within a host and that the parasite genome has ten equidistant loci with a single recombination breakpoint at loci 5 (i.e.). Within this framework, we consider at loci 1:5 if 2/3 strains have coalesced, the within-host IBD for this section is 2/3. Next, for loci 6:10 if no strains have coalesced the within-host IBD is 0. Combining these results with-weighting for respective length/portion of the genome (weights here equal and therefore negligible) the overall within-host IBD is:

$$\frac{2+0}{Host_{COI}-1}$$

, where one is subtracted from the Host-COI for self-comparison, wwhich gives (3-1) + (3-1) (for each loci). Note, because we consider self comparisons, the denominator is always less than the true COI.

Usage

```
get_within_ibd(swf, host_index = NULL)
```

Arguments

`swf`	result of forwards simulation using the function `sim_swf()`
`host_index`	a vector of target hosts. Defaults to all hosts

Details

Function limited to a single host per "realization"

Value

double of within-host IBD

polySimIBD *Structured Wright Fisher Simulator for Malarai Genetics*

Description

Unique features of the malaria life-cycle and transmission dynamics requires extensions of typical population genetic simulators. Using a discrete-loci, discrete-time structured Wright Fisher framework, we simulate malaria population genetics forwards in time. Users are then able to capture the full Ancestral Recombination Graph.

sim_swf *The Structured Wright Fisher Model for IBD*

Description

Simulate a population forwards with recombination that approximates the Structured Wright Fisher Process and tracks haplotype identity by descent where individuals represent demes, such that within a deme individual-level COI is considered.

Usage

```
sim_swf(pos, N, m, rho, mean_coi, tlim, migr_mat = 1, verbose = FALSE)
```

Arguments

pos	vector; the genomic coordinates for chromosome and position of the sites
N	integer vector; The number of individuals to consider in each deme
m	numeric numeric; Probability of internal migration where m represents the probability of moving from host_origin to host_new by m*(1-1/N) of each deme
rho	numeric; expected recombination rate
mean_coi	numeric vector; The lambda of a right-shifted Poisson process, 1 + Pos(lambda) representing the average COI of each deme
tlim	numeric; the maximum number of generations to consider before exiting gracefully if all samples have not coalesced
migr_mat	numeric matrix; Migrations rates or probabilities between destination and origin. Note, because this is a Wright-Fisher model, we are drawing parents and therefore migration matrix is parameterized towards "where one came from" versus "where one is headed": origin specified as columns and destination in rows Default value of 1 indicates non-spatial model. Note, if probability matrix, rows must sum to 1 (valid marginal probability); otherwise, values will be assumed to be rates and converted to probabilities
verbose	boolean

Details

Demes are assumed to be ordered throughout (i.e. the order needs to be consistent between N, m, mean_coi, and the rows and columns of the migration matrix).

Migration matrix is assumed to be a distance matrix that is either a rate or a probability. The program will coerce the matrix into a probability distribution between origin and destination based on the row-sums.

This function is intended to be fed into the get_arg function for interpretability.

Value

Returns a list of length six that contains the COI of each individual. A recombination list of length of tlim where each element contains the recombination block – as a boolean – of the two parental haplotypes. (the number of generations it took for all lineages to coalesce). Finally, there are lists for the parental host and parental haplotype assignments for the "paternal" and "maternal" haplotypes (1 and 2), respectively.

---

subset_bvtree *Subset an object of class bvtree*

Description

Given a bvtree and a vector of indices s, creates a new tree which is a subset of the original tree focusing only on the elements s.

Usage

```
subset_bvtree(bvtree, s)
```

Arguments

bvtree	an object of class "bvtree"
s	a vector specifying which elements in the bvtree to focus on

---

swfsim *The Structured Wright Fisher Model Output This S3 class represents realization of the Discrete-Time Discrete-Loci Spatial Wright Fisher Malaria Model.*

Description

The realization of the Discrete-Time Discrete-Loci Spatial Wright Fisher Malaria Model contains all of the information to create the ARG: each generation's parents, the resulting recombination events between parents, and the offspring haplotypes

Fields

`pos` vector; the genomic coordinates for chromosome and position of the sites

`coi` vector; the COI of each host

`recomb` list; recombination blocks for each ancestral host for each generation

`parent_host1` list; parent for host 1 for each generation

`parent_host2` list; parent for host 2 for each generation

`parent_haplo1` list; haplotypes for parent 1 for each generation

`parent_haplo2` list; haplotypes for parent 2 for each generation

Index