Package ‘junctions’

April 9, 2025

Type Package

Title The Breakdown of Genomic Ancestry Blocks in Hybrid Lineages

Version 2.1.3

Description Individual based simulations of hybridizing populations, where the accumulation of junctions is tracked. Furthermore, mathematical equations are provided to verify simulation outcomes. Both simulations and mathematical equations are based on Janzen (2018, <doi:10.1101/058107>) and Janzen (2022, <doi:10.1111/1755-0998.13519>).

License GPL (>= 2)

URL `https://thijsjanzen.github.io/junctions/`

BugReports `https://github.com/thijsjanzen/junctions/issues`

Depends RcppParallel (>= 5.0.0)

Imports nloptr, Rcpp, tibble

Suggests dplyr, ggplot2, knitr, magrittr, rmarkdown, testthat, tidyr

LinkingTo nloptr, Rcpp, RcppParallel

VignetteBuilder knitr

Encoding UTF-8

RoxygenNote 7.3.2

SystemRequirements C++17

NeedsCompilation yes

Author Thijs Janzen [aut, cre]

Maintainer Thijs Janzen <`thijsjanzen@gmail.com`>

Repository CRAN

Date/Publication 2025-04-09 09:40:06 UTC

Contents

`junctions-package` *Extending The Theory of Junctions*

Description

The theory of junctions is extended by this package by including the effect of a finite number of recombination sites along the chromosome. The package provides functions to calculate the estimated number of junctions, depending on the time since the onset of hybridization, population size, number of recombination sites, initial heterozygosity and the number of crossovers per meiosis.

Details

This package provides individual based simulations in order to simulate the accumulation of junctions over time, both for chromosomes with a finite and an infinite number of recombination sites. Furthermore, the package provides mathematical tools to verify the outcomes of the individual based simulations.

Update version 2.1.1 : fixed typo in one of the examples

Update version 2.1.0 : updated tbb::task_scheduler_init to tbb::global_control

Update version 2.0.2 : simplified some tests

Update version 2.0 : merged many functions with similar functionality, added vignette that provides overview of all functionality.

Update version 1.9 : added c++ versions of the unphased and phased likelihoods.

Update version 1.8 : added multithreading using the TBB library.

Update version 1.7 : further improved the recombination function following Hanno Hildenbrandt's suggestions

Update version 1.6 : improved the recombination function to run twice as fast

Update version 1.5.1: added option to track the true number of junctions

Update version 1.5: added support for inferring the time since admixture based on phased and unphased data. Also included are simulation functions to simulate appropriate data (e.g. phased and unphased).

Update version 1.4: added support for estimating the number of junctions, and simulating the number of junctions, under a backcrossing scheme, using the code supplied in Lavretsky et al. 2019.

Update version 1.3: added support for estimating the time since admixture using unphased data.
Update version 1.3: added individual based simulations returning phased and unphased data.
Update version 1.3: Updated entire package to Roxygen.

Update version 1.2: added support for estimating the expected number of junctions for arbitrarily distributed markers.

Update version 1.1: updated underlying random number generator for picking recombination sites. The previous generator had limited precision, which could generate duplicate recombination sites. This update fixes that

Author(s)

Maintainer: Thijs Janzen <thijsjanzen@gmail.com>

References

Janzen, T. , Nolte, A. W. and Traulsen, A. (2018), The breakdown of genomic ancestry blocks in hybrid lineages given a finite number of recombination sites. Evolution, 72: 735-750. doi:10.1111/evo.13436

Lavretsky, P, Janzen, T. and McCracken, KG. (2019) Identifying hybrids & the genomics of hybridization: Mallards & American black ducks of Eastern North America. Ecology and Evolution 9: 3470-3490. doi:10.1002/ece3.4981

`calculate_mat` *Function to calculate the maximum accurate time*

Description

Function that calculates the maximum time after hybridization after which the number of junctions can still be reliably used to estimate the onset of hybridization. This is following equation 15 in Janzen et al. 2018.

Usage

```
calculate_mat(N = Inf, R = Inf, H_0 = 0.5, C = 1)
```

Arguments

`N`	Population Size
`R`	Number of genetic markers
`H_0`	Frequency of heterozygosity at t = 0
`C`	Mean number of crossovers per meiosis (e.g. size in Morgan of the chromosome)

Value

The maximum accurate time

Examples

```
calculate_mat(N = Inf, R = 1000, H_0 = 0.5, C = 1)
```

`calc_k` *Calculate the limit of the number of junctions*

Description

Calculate the average number of junctions after an infinite number of generations, provided information on the initial heterozygosity, population size and the number of generations.

Usage

```
calc_k(N = Inf, R = Inf, H_0 = 0.5, C = 1)
```

Arguments

`N`	population size
`R`	number of markers
`H_0`	initial heterozygosity (at the time of admixture)
`C`	Mean number of crossovers per meiosis (e.g. size in Morgan of the chromosome)

Value

The number of junctions for at time = infinity

Examples

```
k <-  calc_k(N = 100, R = 1000, H_0 = 0.5, C = 1)
```

`estimate_time` *Estimate the time since the onset of hybridization, using the number of junctions*

Description

Estimate the time since the onset of hybridization, following equation 14 in Janzen et al. 2018

Usage

```
estimate_time(J = NA, N = Inf, R = Inf, H_0 = 0.5, C = 1)
```

Arguments

`J`	The observed number of junctions
`N`	Population Size
`R`	Number of genetic markers
`H_0`	Frequency of heterozygosity at t = 0
`C`	Mean number of crossovers per meiosis (e.g. size in Morgan of the chromosome)

Value

The number of generations passed since the onset of hybridization

Examples

```
J <- number_of_junctions(N = 100, R = 1000, H_0 = 0.5, C = 1, t = 200)
estimate_time(J = J, N = 100, R = 1000, H_0 = 0.5, C = 1)
# should be 200 again
```

`estimate_time_diploid` *estimates the time since admixture, given diploid ancestry data.*

Description

Calculates the time since admixture, given unphased ancestry data.

Usage

```
estimate_time_diploid(
  ancestry_information,
  analysis_type = "individuals",
  phased = FALSE,
  pop_size = 1000,
  freq_ancestor_1 = 0.5,
  lower_lim = 2,
  upper_lim = 2000,
  num_threads = 1,
  verbose = FALSE
)
```

Arguments

`ancestry_information`	a matrix with five columns: column 1) indicator of individual, column 2) indicator of chromosome, 3) location of marker in Morgan, 4) ancestry at chromosome 5) ancestry at chromosome 2.
`analysis_type`	how should the data be broken down? there are multiple options: "individuals" - time is inferred for each individual separately, grouping all chromosomes together that belong to the same individual. "chromosomes" - time is inferred for each chromosome separately, grouping chromosomes together belonging from separate individuals. "separate" - time is inferred for each chromosome from each individual separately, "all" - time is inferred jointly for all chromosomes and individuals, grouping all chromosomes and individuals together.
`phased`	is the data phased?
`pop_size`	population size
`freq_ancestor_1`	Frequency of ancestor 1 at t = 0
`lower_lim`	lower limit of the optimization algorithm. Increase if the expected admixture time is relatively ancient
`upper_lim`	upper limit of hte optimization algorithm. If set too large, recent admixture events can be overlooked - best to set as low as possible.
`num_threads`	num_threads, default is all threads. 5 threads is recommended.
`verbose`	display intermediate output? Default = FALSE

`estimate_time_haploid` *estimate time using likelihood for a single chromosome*

Description

Estimate the time since the onset of hybridization, for a haploid genome

Usage

```
estimate_time_haploid(
  ancestry_matrix,
  N = 1000,
  freq_ancestor_1 = 0.5,
  lower_lim = 2,
  upper_lim = 1000,
  verbose = FALSE
)
```

Arguments

`ancestry_matrix`	matrix with 3 columns, column 1 = chromosome, column 2 = location in Morgan, column 3 = ancestry.
`N`	Population Size
`freq_ancestor_1`	Frequency of ancestor 1 at t = 0
`lower_lim`	lower limit of the optimization algorithm. Increase if the expected admixture time is relatively ancient
`upper_lim`	upper limit of the optimization algorithm. If set too large, recent admixture events can be overlooked - best to set as low as possible.
`verbose`	return verbose output

Value

The number of generations passed since the onset of hybridization

---

`estimate_time_one_chrom` *Estimate the time since the onset of hybridization, using the observed number of junctions, taking into account the distribution of markers on a single chromosome*

Description

Estimate the time since the onset of hybridization, following equation 1 in Janzen et al. unpublished

Usage

```
estimate_time_one_chrom(
  J = NA,
  N = Inf,
  H_0 = 0.5,
  marker_distribution = NA,
  lower_lim = 2,
  upper_lim = 1000
)
```

Arguments

`J`	The observed number of junctions
`N`	Population Size
`H_0`	Frequency of heterozygosity at t = 0
`marker_distribution`	A vector containing the position of all markers in Morgan.
`lower_lim`	lower limit of the optimization algorithm. Increase if the expected admixture time is relatively ancient
`upper_lim`	upper limit of the optimization algorithm. If set too large, recent admixture events can be overlooked - best to set as low as possible.

Value

The number of generations passed since the onset of hybridization

Examples

```
markers <- seq(from = 0, to = 1, length.out = 100)
J <- number_of_junctions_markers(N = 100, H_0 = 0.5, t = 200,
marker_distribution = markers)
estimate_time_one_chrom(J = J,
                        N = 100,
                        H_0 = 0.5,
                        marker_distribution = markers) #should be 200 again
```

`log_likelihood_diploid`

calculate the log likelihood of observing diploid ancestry data.

Description

Calculates the log likelihood of observing the phased data, given the population size, initial heterozygosity and time since admixture

Usage

```
log_likelihood_diploid(
  local_anc_matrix,
  pop_size,
  freq_ancestor_1 = 0.5,
  t,
  phased = FALSE,
  num_threads = 1
)
```

Arguments

`local_anc_matrix`	a matrix with four columns: column 1) chromosome indicator, 2) location of marker in Morgan on respective chromosome 3) ancestry at chromosome 4) ancestry at chromosome 2.
`pop_size`	population size
`freq_ancestor_1`	Frequency of ancestor 1 at t = 0
`t`	time since admixture
`phased`	is the data phased or not? default is false.
`num_threads`	number of threads, default is one thread. Set to -1 to use all available threads.

Value

log likelihood

`log_likelihood_haploid` *log likelihood of the time since admixture for a haploid genome*

Description

log likelihood of the time since admixture for a set of single chromosomes (for ex. in Yeast).

Usage

```
log_likelihood_haploid(ancestry_matrix, N = 1000, freq_ancestor_1 = 0.5, t = 2)
```

Arguments

`ancestry_matrix`	matrix with 3 columns, column 1 = chromosome, column 2 = location in Morgan, column 3 = ancestry.
`N`	Population Size
`freq_ancestor_1`	Frequency of ancestor 1 at t = 0
`t`	time since admixture

Value

loglikelihood

`number_of_junctions` *Calculate the average number of junctions*

Description

Calculate the average number of junctions in a single chromosome after t generations, provided information on the initial heterozygosity, population size and the number of generations.

Usage

```
number_of_junctions(N = Inf, R = Inf, H_0 = 0.5, C = 1, t = 100)
```

Arguments

`N`	Population Size
`R`	Number of genetic markers
`H_0`	Frequency of heterozygosity at t = 0
`C`	Mean number of crossovers per meiosis (e.g. size in Morgan of the chromosome)
`t`	Time since admixture

Value

Estimated number of junctions at time t

Examples

```
jt <-  number_of_junctions(N = 100, R = 1000, H_0 = 0.5, C = 1, t = 1000)
jt2 <- number_of_junctions(N = 100, R = 1000, H_0 = 0.5, C = 1, t = 0:1000)
```

`number_of_junctions_backcross` *Calculate the average number of junctions during backcrossing*

Description

Calculate the expected number of junctions after t generations, in a backcrossing mating scheme.

Usage

```
number_of_junctions_backcross(H_0 = 0.5, C = 1, t = 100)
```

Arguments

`H_0`	Frequency of heterozygosity at t = 0
`C`	Mean number of crossovers per meiosis (e.g. size in Morgan of the chromosome)
`t`	Time since admixture

Value

Estimated number of junctions at time t

Examples

```
jt <-  number_of_junctions_backcross(H_0 = 0.1, C = 1, t = 5)
```

`number_of_junctions_di` *Calculate the expected number of junctions between two markers separated by a given amount of recombination*

Description

Calculate the expected number of junctions after t generations, provided information on the initial heterozygosity, population size, the number of generations since the onset of admixture and the distance between two markers.

Usage

```
number_of_junctions_di(N = Inf, H_0 = 0.5, t = 100, di = 1e-06)
```

Arguments

`N`	Population Size
`H_0`	Frequency of heterozygosity at t = 0
`t`	Time since admixture
`di`	Distance between two markers in Morgan

Value

Estimated number of junctions at time t

Examples

```
number_of_junctions_di(N = 100, H_0 = 0.5, t = 1000, di = 0.01)
```

`number_of_junctions_markers`

Calculate the expected total number of junctions in a chromosome, given the distribution of markers

Description

Calculate the expected number of junctions after t generations, provided information on the initial heterozygosity, population size, the number of generations since the onset of admixture and the distribution of markers.

Usage

```
number_of_junctions_markers(
  N = Inf,
  H_0 = 0.5,
  t = 100,
  marker_distribution = NA
)
```

Arguments

`N`	Population Size
`H_0`	Frequency of heterozygosity at t = 0
`t`	Time since admixture
`marker_distribution`	A vector containing the position of all markers in Morgan.

Value

Estimated number of observed junctions at time t

Examples

```
markers <- seq(from = 0, to = 1, length.out = 1000)
jt <-  number_of_junctions_markers(N = 100,
                                   H_0 = 0.5,
                                   t = 1000,
                                   marker_distribution = markers)
random_markers <- sort(runif(1000, 0, 1))
jt2 <- number_of_junctions_markers(N = 100,
                                   H_0 = 0.5,
                                   t = 1000,
                                   marker_distribution = random_markers)
```

`sim_backcrossing` *Function to simulate data using a back crossing scheme*

Description

Individual based simulation of the accumulation of junctions, under a back crossing scheme

Usage

```
sim_backcrossing(
  population_size = 100,
  freq_ancestor_1 = 0.5,
  total_runtime = 5,
  size_in_morgan = 1,
  number_of_markers = 100,
  seed = 6,
  time_points = -1
)
```

Arguments

`population_size`	Population size
`freq_ancestor_1`	Frequency of ancestor 1 at t = 0
`total_runtime`	Number of generations to simulate
`size_in_morgan`	Mean number of crossovers per meiosis (e.g. size in Morgan of the chromosome)
`number_of_markers`	number of molecular markers
`seed`	Seed of the pseudo-random number generator
`time_points`	vector with time points at which local ancestry has to be recorded to be returned at the end of the simulation. If left at -1, ancestry is recorded at every generation (computationally heavy).

Value

List with five entries: average_junctions: average number of junctions over time, detected_junctions: average number of detected junctions, given the markers. markers: vector with the locations of the molecular markers, junction_distribution: distribution of junctions per time step average_heterozygosity: average heterozygosity.

Examples

```
sim_backcrossing(population_size = 100,
                      total_runtime = 5,
                      size_in_morgan = 1,
                      number_of_markers = 100,
                      seed = 6,
                      time_points = 1:5)
```

`sim_fin_chrom` *Individual Based Simulation of the accumulation of junctions*

Description

Individual based simulation of the accumulation of junctions for a chromosome with regularly distributed markers.

Usage

```
sim_fin_chrom(
  pop_size = 100,
  freq_ancestor_1 = 0.5,
  total_runtime = 100,
  morgan = 1,
  seed = 42,
  R = 100
)
```

Arguments

`pop_size`	Population Size
`freq_ancestor_1`	Frequency of ancestor 1 at t = 0
`total_runtime`	Maximum time after which the simulation is to be stopped
`morgan`	Mean number of crossovers per meiosis (e.g. size in Morgan of the chromosome)
`seed`	Seed of the pseudo-random number generator
`R`	Number of regularly distributed markers

Value

`avgJunctions`	vector of the average number of junctions at time = [0, total_runtime]

Examples

```
sim_fin_chrom(pop_size = 100, freq_ancestor_1 = 0.5,
                 total_runtime = 10, morgan = 1, seed = 42,
                 R = 100)
```

`sim_inf_chrom` *Individual Based Simulation of the accumulation of junctions*

Description

Individual based simulation of the accumulation of junctions for a chromosome with an infinite number of recombination sites.

Usage

```
sim_inf_chrom(
  pop_size = 100,
  freq_ancestor_1 = 0.5,
  total_runtime = 100,
  morgan = 1,
  markers = -1,
  seed = 42
)
```

Arguments

`pop_size`	Population Size
`freq_ancestor_1`	Frequency of ancestor 1 at t = 0
`total_runtime`	Maximum time after which the simulation is to be stopped
`morgan`	Mean number of crossovers per meiosis (e.g. size in Morgan of the chromosome)
`markers`	The number of genetic markers superimposed on the chromosome. If markers is set to -1, no markers are superimposed (faster simulation)
`seed`	Seed of the pseudo-random number generator

Value

`avgJunctions`	vector of the average number of junctions at time = [0, total_runtime]

Examples

```
v <- sim_inf_chrom(pop_size = 100, freq_ancestor_1 = 0.5,
                   total_runtime = 10, morgan = 1, markers = 100,
                   seed = 42)
plot(v$avgJunctions, type = "l", xlab = "Generations",
ylab = "Number of Junctions", main = "Example Infinite Chromosome")
lines(v$detectedJunctions, col = "blue")
legend("bottomright", c("Real number","Number detected"),
       lty = 1, col = c("black", "blue"))
```

`sim_phased_unphased` *Individual Based Simulation of the accumulation of junctions*

Description

Individual based simulation of the accumulation of junctions, returning phased and unphased data. Ancestry on both chromosomes of 10 randomly sampled individuals per generations is returned.

Usage

```
sim_phased_unphased(
  pop_size = 100,
  freq_ancestor_1 = 0.5,
  total_runtime = 100,
  size_in_morgan = 1,
  markers = 100,
  time_points = -1,
  num_threads = 1,
  verbose = FALSE,
  record_true_junctions = FALSE,
  num_indiv_sampled = 10,
  coverage = 1,
  error_rate = 0
)
```

Arguments

`pop_size`	Population Size
`freq_ancestor_1`	Frequency of ancestor 1 at t = 0
`total_runtime`	Maximum time after which the simulation is to be stopped
`size_in_morgan`	Mean number of crossovers per meiosis (e.g. size in Morgan of the chromosome)
`markers`	If a single number is provided, the number is used as the total number of markers generated either randomly, or using a regular distribution (a regular distribution is chosen if the number is negative). If a vector is provided, that vector is used.

`time_points`	vector with time points at which local ancestry has to be recorded to be returned at the end of the simulation. If left at -1, ancestry is recorded at every generation (computationally heavy).
`num_threads`	default is 1. -1 takes all available threads.
`verbose`	displays a progress bar
`record_true_junctions`	keep track of the true number of junctions?
`num_indiv_sampled`	the number of individuals sampled at each time point to be genotyped
`coverage`	fraction of markers that can be succesfully phased
`error_rate`	fraction of markers that are erroneously phased (e.g. swapped)

Value

a tibble with five columns: [time, individual, marker location, ancestry chromosome 1, ancestry chromosome 2]

Examples

```
## Not run:
sim_phased_unphased(pop_size = 100, freq_ancestor_1 = 0.5,
                    total_runtime = 10, size_in_morgan = 1,
                    markers = 10, time_points = c(0, 5, 10),
                    num_threads = 1)

## End(Not run)
```

`time_error` *Estimate the error in the time estimate*

Description

Calculate the error in the estimate of the onset of hybridization, following Equations 3 & 4 in the Supplementary information of Janzen et al. 2018.

Usage

```
time_error(t = NA, N = Inf, R = Inf, H_0 = 0.5, C = 1, relative = TRUE)
```

Arguments

`t`	Inferred time
`N`	Population Size
`R`	Number of genetic markers
`H_0`	Frequency of heterozygosity at t = 0

`C`	Mean number of crossovers per meiosis (e.g. size in Morgan of the chromosome)
`relative`	Boolean flag, if TRUE: return the relative error, if FALSE: return error in generations

Value

Expected error in the time estimate

Index