Package ‘TimeMachine’

August 29, 2016

Version 1.2
Date 2014-01-13
Title Time Machine
Depends R (>= 2.15), methods
Description Implements the Time Machine, a simulation approach for stochastic trees.
License GPL (>= 2)
Author Gianluca Campanella [aut, cre],
Marc Chadeau-Hyam [aut],
Maria De Iorio [ctb],
Ajay Jasra [ctb]
Maintainer Gianluca Campanella <gianluca@campanella.org>
NeedsCompilation yes
Repository CRAN
Date/Publication 2014-01-13 12:59:44

R topics documented:

full.transitions .......................................................... 2
mu.mle ................................................................. 2
pdm ................................................................. 3
pophistory ......................................................... 4
sample.pop ......................................................... 5
stationary.dist .................................................. 5
time-dist ......................................................... 6
tm ................................................................. 7

Index 10
**full.transitions**

*Full Transition Matrix Computation*

**Description**
Computes the full transition matrix from a given unitary transition matrix.

**Usage**
```
full.transitions(unitary.transitions, loci)
```

**Arguments**
- `unitary.transitions`  
  Square matrix of order 2 specifying per-locus transition probabilities
- `loci`  
  Number of loci

**Value**
A square matrix of order 2^loci specifying all transition probabilities.

---

**mu.mle**

*Maximum-Likelihood Estimation of the Mutation Rate*

**Description**
Estimates the mutation rate using a maximum-likelihood approach.

**Usage**
```
mu.mle(transitions, pi=NULL, population, n, mu.int, samples, 
  threads=NULL, ...)```

**Arguments**
- `transitions`  
  Transition matrix between types
- `pi`  
  Stationary distribution associated with the transition matrix, or NULL to compute it
- `population`  
  Vector representing the initial population
- `n`  
  Target population size
- `mu.int`  
  Vector specifying the end-points of the interval to be searched for the optimal mutation rate
- `samples`  
  Number of samples to simulate
- `threads`  
  Number of threads used for simulations, or NULL to use the default OpenMP setting
- `...`  
  Further arguments passed to optimize
**Value**

- **mu.hat**: The maximum-likelihood estimate of the mutation rate
- **loglik**: The log-likelihood value associated with the estimated mutation rate

**Examples**

```r
# Load example dataset
data(pdm)
transitions <- full.transitions(pdm$unitary.transitions, pdm$loci)
pi <- stationary.dist(transitions)

mu.int <- c(0.1, 10)
samples <- 10

# MLE of the mutation rate for a single target population size
n <- 10
mle.res <- mu.mle(transitions, pi, pdm$population, n, mu.int, samples)
print(mle.res)

# MLE of the mutation rate for 10 different target population sizes, including
# up to the MRCA (n = 1)
ns <- 1:10
mle.res <- sapply(ns, mu.mle, transitions=transitions, pi=pi,
                   population=pdm$population, mu.int=mu.int, samples=samples)
print(mle.res)
```

---

**pdm**  
*Example Dataset*

**Description**

Dataset describing an initial population of 100 individuals subdivided into 1024 types, corresponding to each possible combination of genotypes over 10 loci.

**Usage**

```r
pdm
```

**Format**

- **unitary.transitions**: Unitary (per-locus) transition matrix
- **loci**: Number of loci
- **population**: Population
Description

Estimates the sequential event number (SEN) quantiles for each simulated population size.

Usage

```r
## S3 method for class 'tm'
pophistory(x, probs=seq(0, 1, 0.25))
```

Arguments

- `x` An object of class `tm` as returned by a call to `tm`
- `probs` Probabilities passed to `quantile`

Value

A matrix of column vectors containing the estimated SEN quantiles, with row names specifying the population size.

Examples

```r
# Load example dataset
data(pdm)

transitions <- full.transitions(pdm$unitary.transitions, pdm$loci)
pi <- stationary.dist(transitions)
n <- 10
mu <- 1
samples <- 10
est.res <- tm(transitions, pi, pdm$population, n, mu, samples)

# Plot minimum, median and maximum SEN as a function of population size
h <- pophistory(est.res, probs=c(0, 0.5, 1))
matplot(h, rownames(h), type="s", lty=c(3, 1, 3), col="black",
xlab="Population size", ylab="SEN")
```
sample.pop

Sample Populations from the Wright-Fisher Model

Description

Samples a population of the desired size from the Wright-Fisher model.

Usage

sample.pop(transitions, pi=NULL, pop.size, mu)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>transitions</td>
<td>Transition matrix between types</td>
</tr>
<tr>
<td>pi</td>
<td>Stationary distribution associated with the transition matrix, or NULL to compute it</td>
</tr>
<tr>
<td>pop.size</td>
<td>Desired population size</td>
</tr>
<tr>
<td>mu</td>
<td>Mutation rate</td>
</tr>
</tbody>
</table>

Value

A vector representing the sampled population.

stationary.dist

Compute Stationary Distribution of a Markov Chain

Description

Computes the stationary distribution of a Markov chain specified by its transition matrix.

Usage

stationary.dist(transitions)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>transitions</td>
<td>Square matrix specifying the transition probabilities of the Markov chain</td>
</tr>
</tbody>
</table>

Value

A vector representing the stationary distribution.
Description

Estimates the distribution of the times (expressed in SEN) between two coalescent events or two mutations.

Usage

```r
## S3 method for class 'tm'
ttc(x)
## S3 method for class 'tm'
ttm(x)
```

Arguments

- `x` An object of class `tm` as returned by a call to `tm`

Value

A vector representing the distribution of the time to coalescent event for `ttc`, or the time to mutation for `ttm`.

Examples

```r
# Load example dataset
data(pdm)

transitions <- full.transitions(pdm$unitary.transitions, pdm$loci)
pi <- stationary.dist(transitions)
n <- 10
mu <- 1
samples <- 10
est.res <- tm(transitions, pi, pdm$population, n, mu, samples)

# Plot distributions of time to coalescent event and time to mutation
par(mfrow=c(1, 2))
plot(ttc(est.res), type="b", pch=20, lty=3, xlab="SEN", ylab="",
     main="Time to coalescent event")
plot(ttm(est.res), type="b", pch=20, lty=3, xlab="SEN", ylab="",
     main="Time to mutation")
```
Description

Estimates the log-likelihood for the given parameter values using the Time Machine.

Usage

```r
tm(transitions, pi=NULL, population, n, mu, samples, threads=NULL)
```

## S3 method for class 'tm'
```r
density(x, ...)
```

## S3 method for class 'tm'
```r
mean(x, ...)
```

## S3 method for class 'tm'
```r
print(x, ...)
```

## S3 method for class 'tm'
```r
quantile(x, ...)
```

## S3 method for class 'tm'
```r
summary(object, ..., digits=max(3, getOption("digits")-3))
```

Arguments

- `transitions`: Transition matrix between types
- `pi`: Stationary distribution associated with the transition matrix, or NULL to compute it
- `population`: Vector representing the initial population
- `n`: Target population size
- `mu`: Mutation rate
- `samples`: Number of samples to simulate
- `threads`: Number of threads used for simulations, or NULL to use the default OpenMP setting
- `x, object`: An object of class `tm` as returned by a call to `tm`
- `...`: Further arguments passed to the corresponding generic function
- `digits`: Minimum number of significant digits for number formatting
Value
An object of class `tm` representing the result of the simulations.

- **population**: Vector representing the initial population
- **n**: Target population size
- **mu**: Mutation rate
- **logliks**: Vector of simulated log-likelihoods
- **corrections**: Vector of correction terms
- **coalescent.events**: Matrix of column vectors (one per simulation) containing SENs when coalescent events were simulated
- **final.populations**: Matrix of columns vectors (one per simulation) containing final population sizes for each type
- **simulation.times**: Computation time for each simulation (in seconds)
- **total.time**: Total computation time (in seconds)

Examples

```
# Load example dataset
data(pdm)

transitions <- full.transitions(pdm$unitary.transitions, pdm$loci)
pi <- stationary.dist(transitions)
n <- 10
samples <- 10

# Estimate log-likelihood for a fixed value of the mutation rate
mu <- 1
est.res <- tm(transitions, pi, pdm$population, n, mu, samples)
print(est.res)

# Estimate log-likelihood for different values of the mutation rate
mus <- seq(0.1, 10, 0.1)
estimates <- sapply(mus, function(mu) {
  tm(transitions, pi, pdm$population, n, mu, samples)
}, simplify=FALSE)

# Compute mean log-likelihood for each value of the mutation rate mu
mean.logliks <- do.call(rbind, lapply(estimates, function(x) {
  c(x$mu, mean(x))
}))

# Plot mean log-likelihoods
plot(mean.logliks, pch=20, xlab=expression(mu), ylab="Mean log-likelihood",
main=paste("Mean log-likelihoods - Sample size:", samples))

# Estimate log-likelihood for different target population sizes and fixed
# mutation rate
ns <- 2:50
mu <- 1
estimates <- sapply(ns, function(n) {
    tm(transitions, pi, pdm$population, n, mu, samples)
}, simplify=FALSE)

# Compute mean correction term/log-likelihood ratios for each target population size
mean.corrections <- do.call(rbind, lapply(estimates, function(x) {
    c(x$n, mean(x$correction / x$logliks))
}))

# Plot mean correction term/log-likelihood ratios
plot(mean.corrections[,1], rev(mean.corrections[,2]), pch=20, 
     xlab="Target population size", ylab="Mean correction term/log-likelihood", 
     axes=FALSE)
axis(1, at=mean.corrections[,1], labels=rev(mean.corrections[,1]))
axis(2)
Index

*Topic datasets
   pdm, 3
   density.tm(tm), 7
   full.transitions, 2
   mean.tm(tm), 7
   mu.mle, 2
   pdm, 3
   pophistory, 4
   print.tm(tm), 7
   quantile.tm(tm), 7
   sample.pop, 5
   stationary.dist, 5
   summary.tm(tm), 7
   time-dist, 6
   tm, 7
   ttc(time-dist), 6
   ttm(time-dist), 6