ticphasetype

```
library(ticphasetype)
```

Continuous phase-type distributions: T_{MRCA} and T_{Total}

In an evolutionary tree the time until two sequences coalesce T_i can be measured in number of generations R_i divided by the population size N, this is, $T_i = R_i/N$. T_i can easily be proven to approximate to an exponential distribution with rate $\binom{i}{2}$.

In order to understand the evolutionary history of sequences two additional quantities can be defined –namely the time until the most recent common ancestor T_{MRCA} and the total tree length T_{Total}. T_{MRCA} will simply be the sum of all times until two sequences coalesce, in other words $T_{MRCA} = T_n + T_{n-1} + ... + T_2$, where $T_i \sim \binom{i}{2}$. T_{Total}, on the other hand, takes into account the length of all possible branches, so $T_{Total} = nT_n + (n-1)T_{n-1} + ... + 2T_2$ and, thus, $iT_i \sim \exp(\binom{i-1}{2})$.

The mean and variance of these two quantities can be derived relatively easy. Defining their distribution, however, has proven to be more challenging since both T_{MRCA} and T_{Total} are sums of independent exponentially distributed variables with different rates. Their distribution can be computed as a series of convolutions, but their formulation, application and interpretation might be challenging for the average population geneticist.

Instead, we can think of the sum of exponential distributions as a continuous-time Markov chain, where coalescent events are represented as Markov jumps with rate T_i for T_{MRCA} and iT_i for T_{Total}. The Markov chain will end with an absorbing state, which in both cases will be the MRCA.

The Markov chain can be represented using phase-type theory, where the jump rates are defined with a sub-intensity matrix T and the initial distribution will be defined as a row vector π. If we define τ as the smallest time (or length) for which we are in the absorbing state, then $\tau \sim PH(\pi, T)$. This continuous phase-type distribution has well-documented and easy-to-implement formulas for the expectation, the variance, the survival function, the distribution function and the density function. Moreover, since both π and T can easily be speficied, we can easily represent evolutionary histories that do not follow the standard coalescent model and still use the same phase-type formulas.

`ticphasetype` contains an efficient implementation of continuous phase-type distributions. It has a user-friendly interface for creating phase-type representations of T_{MRCA} and T_{Total} under the standard coalescent model, but it also allows the user to specify their own sub-intensity matrix and initial probabilities for a more flexible implementation.

T_{MRCA}

A `phase_type` class representing a continuous phase-type distribution for T_{MRCA} can be generated using `phase_type()`. For example, when `n`=5:

```
T_MRCA_ph <- phase_type(type = 'T_MRCA', n = 5)
```

There are a number of methods associated with the `phase_type`, such as:

```
mean(T_MRCA_ph)
#> [1] 1.6
```

```
var(T_MRCA_ph)
#> [1] 1.148889
```

```
summary(T_MRCA_ph)
#>
#> Subintensity matrix:
#>      [,1] [,2] [,3] [,4]
#> [1,]  -10   10    0    0
#> [2,]    0   -6    6    0
#> [3,]    0    0   -3    3
#> [4,]    0    0    0   -1
#>
#> Initial probabilities:
#>      [,1] [,2] [,3] [,4]
#> [1,]    1    0    0    0
#>
#> Defect:
#> [1] 0
#>
#> Mean: 1.6
#>
#> Variance: 1.148889
```

T_{Total}

T_{Total} can also be represented using the `phase_type` class:

```
T_Total_ph <- phase_type(type = 'T_Total', n = 5)
```

```
summary(T_Total_ph)
#>
#> Subintensity matrix:
#>      [,1] [,2] [,3] [,4]
#> [1,]   -2  2.0  0.0  0.0
#> [2,]    0 -1.5  1.5  0.0
#> [3,]    0  0.0 -1.0  1.0
#> [4,]    0  0.0  0.0 -0.5
#>
#> Initial probabilities:
#>      [,1] [,2] [,3] [,4]
#> [1,]    1    0    0    0
#>
#> Defect:
#> [1] 0
#>
#> Mean: 4.166667
#>
#> Variance: 5.694444
```

User-defined sub-intensity matrix

Moreover, the height of an evolutionary tree can also be represented by a user-defined sub-intensity matrix. This is specially useful if the model does not follow Kingman's n–coalescent, but other coalescent models such as the psi-coalescent or the beta-coalescent. As an example, an arbitrary sub-intensity matrix can be generated:

```
n=5
subint_arbit <- matrix(0, nrow = n-1, ncol = n-1)

for (i in 1:(n-1)) {
  for (j in 1:(n-1)) {
    if (j>i) {
      subint_arbit[i,j] <- runif(1, 1, 10)
    }
  }
  subint_arbit[i,i] <- -sum(subint_arbit[i,])
}

subint_arbit[n-1,n-1] <- -1
```

```
subint_arbit
#>           [,1]      [,2]      [,3]      [,4]
#> [1,] -22.05029   9.23274  8.505234  4.312321
#> [2,]   0.00000 -16.24583  8.961016  7.284811
#> [3,]   0.00000   0.00000 -6.783488  6.783488
#> [4,]   0.00000   0.00000  0.000000 -1.000000
```

This matrix can be supplied to the `phase_type()` generator function, together with optional initial probabilities:

```
T_arbit_ph <- phase_type(subint_mat = subint_arbit)
#> Warning in phase_type(subint_mat = subint_arbit): The initial probability
#> vector is automatically generated.
```

If the initial probabilities are not supplied (as is the case above), then `phase_type()` automatically generates a vector of initial probabilities as $\pi = (1, 0, ..., 0)$ and raises a warning message.

We can apply the methods `phase_type` class methods for this new user-tailored phase-type distribution:

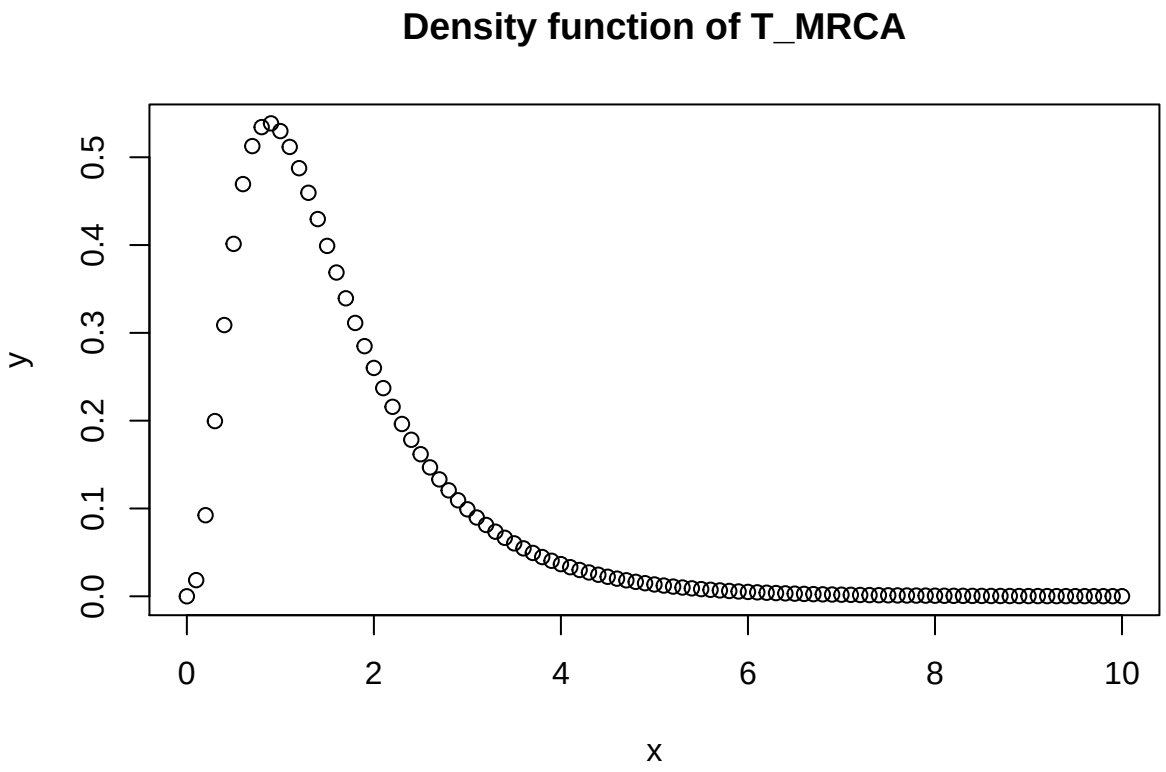
```
summary(T_arbit_ph)
#>
#> Subintensity matrix:
#>           [,1]      [,2]      [,3]      [,4]
#> [1,] -22.05029   9.23274  8.505234  4.312321
#> [2,]   0.00000 -16.24583  8.961016  7.284811
#> [3,]   0.00000   0.00000 -6.783488  6.783488
#> [4,]   0.00000   0.00000  0.000000 -1.000000
#>
#> Initial probabilities:
#>      [,1] [,2] [,3] [,4]
#> [1,]    1    0    0    0
#>
#> Defect:
#> [1] 0
#>
#> Mean: 1.162033
#>
#> Variance: 1.022609
```

Density, distribution and quantile functions

`ticphasetype` also includes the density function (`dphtype()`), quantile function (`qphtype()`), distribution function (`pphtype()`) and random draw generator (`rphtype()`) for the continuous phase-type distribution:

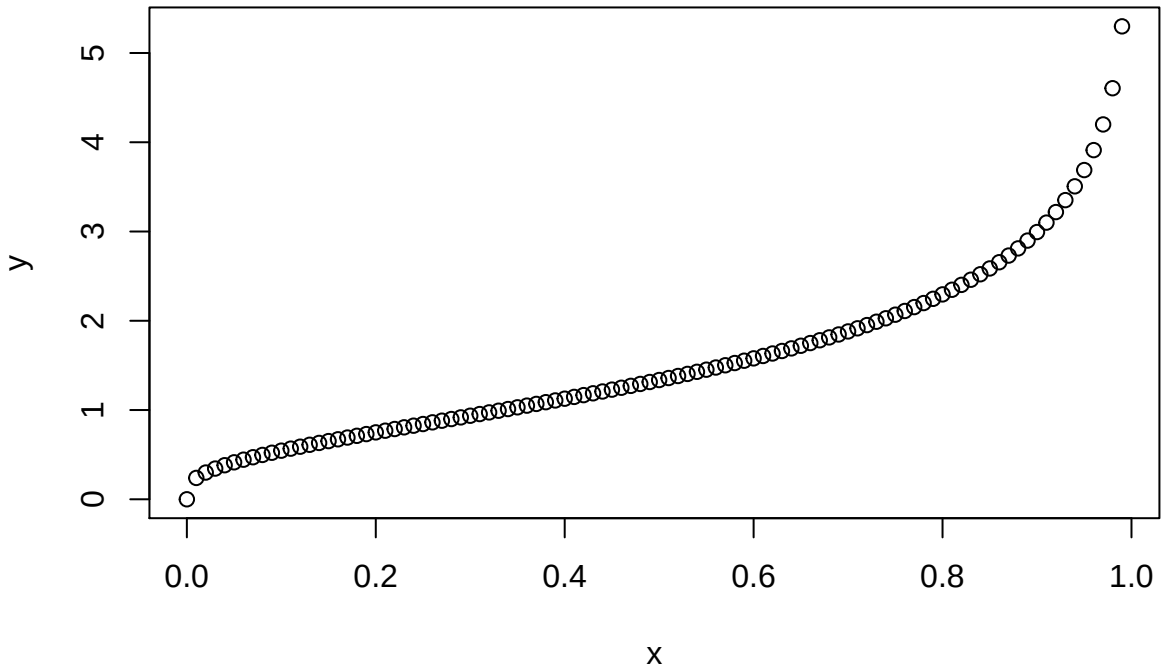
```
dphtype(0.5, T_MRCA_ph)
#> [1] 0.4013376
```

```
x <- seq(0,10,0.1)
y <- dphtype(x, T_MRCA_ph)
plot(x, y)
title('Density function of T_MRCA')
```



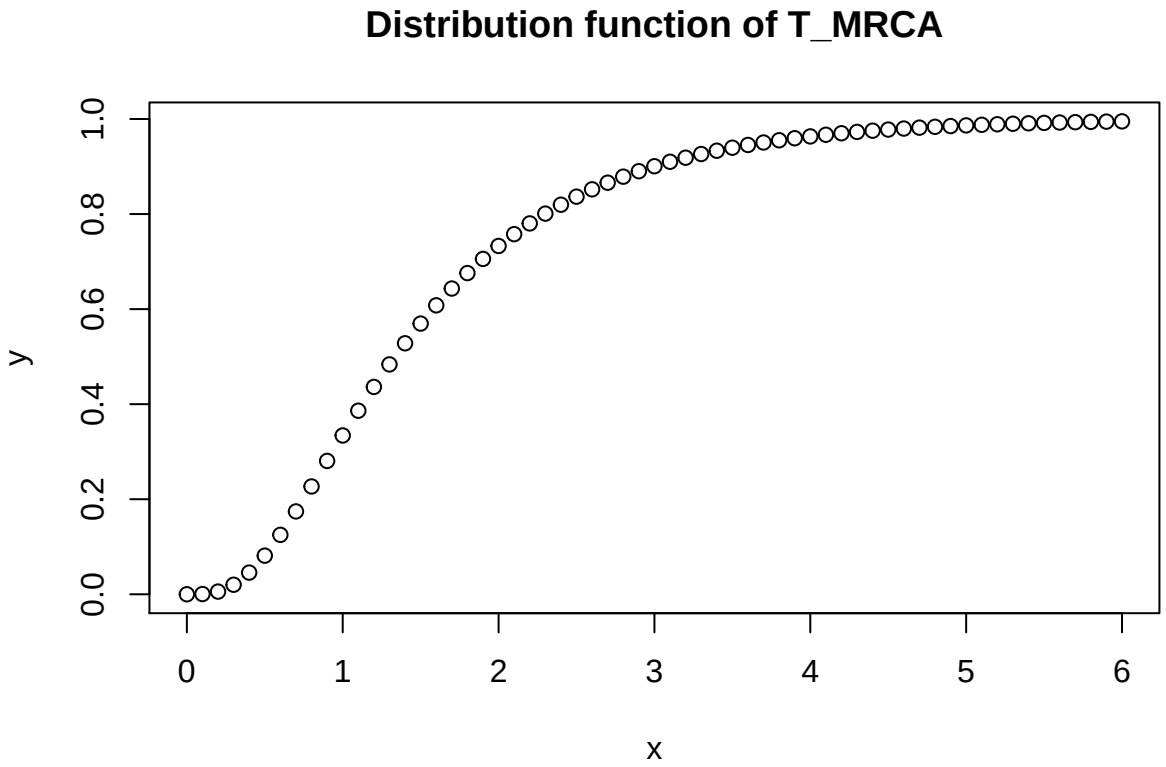

```
qphtype(0.5, T_MRCA_ph)
#> [1] 1.335995
```

```
x <- seq(0,0.99,0.01)
y <- qphtype(x, T_MRCA_ph)
plot(x, y)
title('Quantile function of T_MRCA')
```




```
pphtype(0.5, T_MRCA_ph)
#> [1] 0.0812838
```

```
x <- seq(0,6,0.1)
y <- pphtype(x, T_MRCA_ph)
plot(x, y)
title('Distribution function of T_MRCA')
```




```
rphtype(3, T_MRCA_ph)
#> [1] 2.25 1.19 2.54
```

```
x <- rphtype(10000, T_MRCA_ph)
hist(x, main = 'Random draws of T_MRCA', breaks=20)
```