ticphasetype

`ticphasetype` is useful for representing classical statiscics in population genetics by means of phase-type theory. This provides additional flexibility and efficiency for computing and understanding these statistics at the finest level. Do not hesitate to run `?ticphasetype` for a quick summary of the available functions, or to open the help files for the individual functions.

```
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
```

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

```r
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:

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

```r
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,] -13.44763  9.000765  1.041592  3.405277
#> [2,]   0.00000 -7.709989  2.402870  5.307120
#> [3,]   0.00000  0.000000 -4.576215  4.576215
#> [4,]   0.00000  0.000000  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,] -13.44763  9.000765  1.041592  3.405277
#> [2,]   0.00000 -7.709989  2.402870  5.307120
#> [3,]   0.00000  0.000000 -4.576215  4.576215
#> [4,]   0.00000  0.000000  0.000000 -1.000000
#>
#> Initial probabilities:
#>      [,1] [,2] [,3] [,4]
#> [1,]    1    0    0    0
#>
#> Defect:
#> [1] 0
#>
#> Mean: 1.223683
```

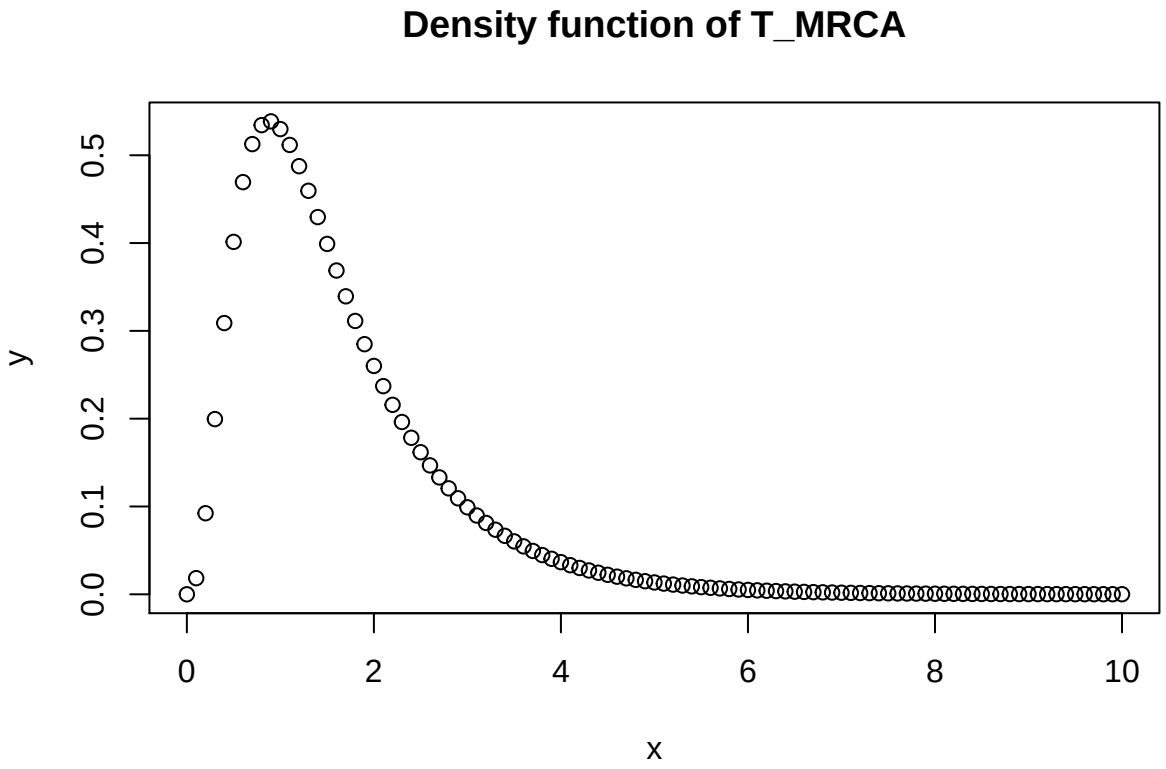
```
#>
#> Variance: 1.044896
```

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:

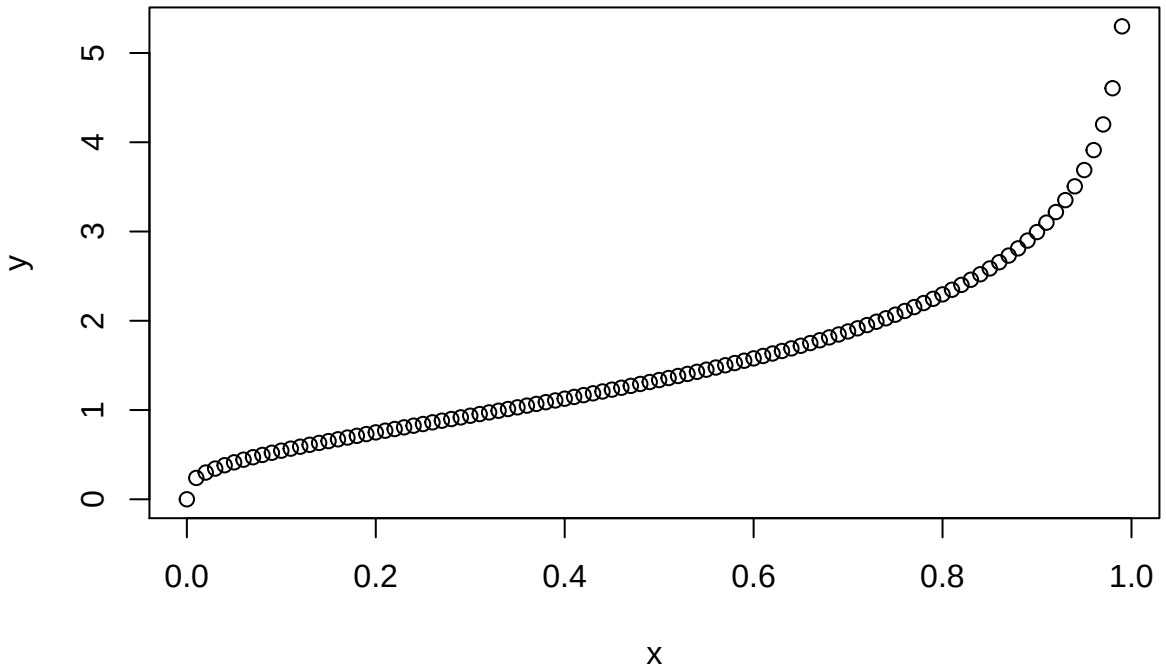
```
dphtype(0.5, T_MRCA_ph)
#> [1] 0.4013376
```

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



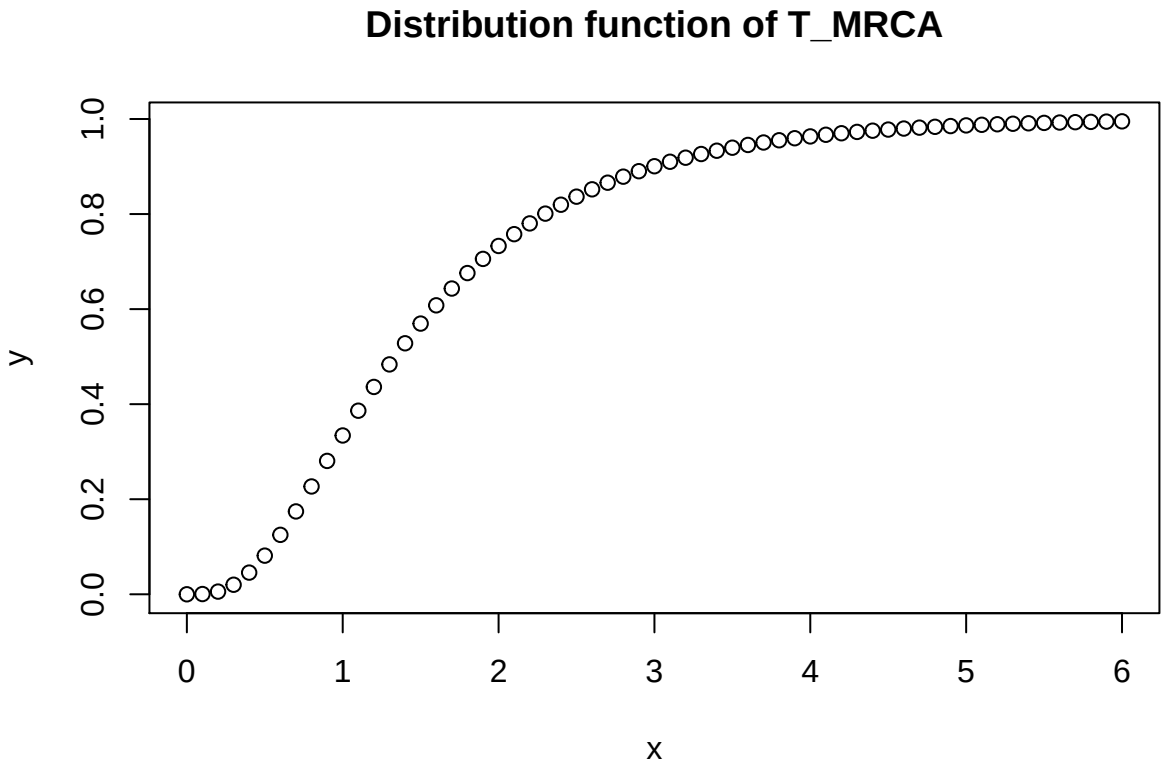

```
qphtype(0.5, T_MRCA_ph)
#> [1] 1.335995
```

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



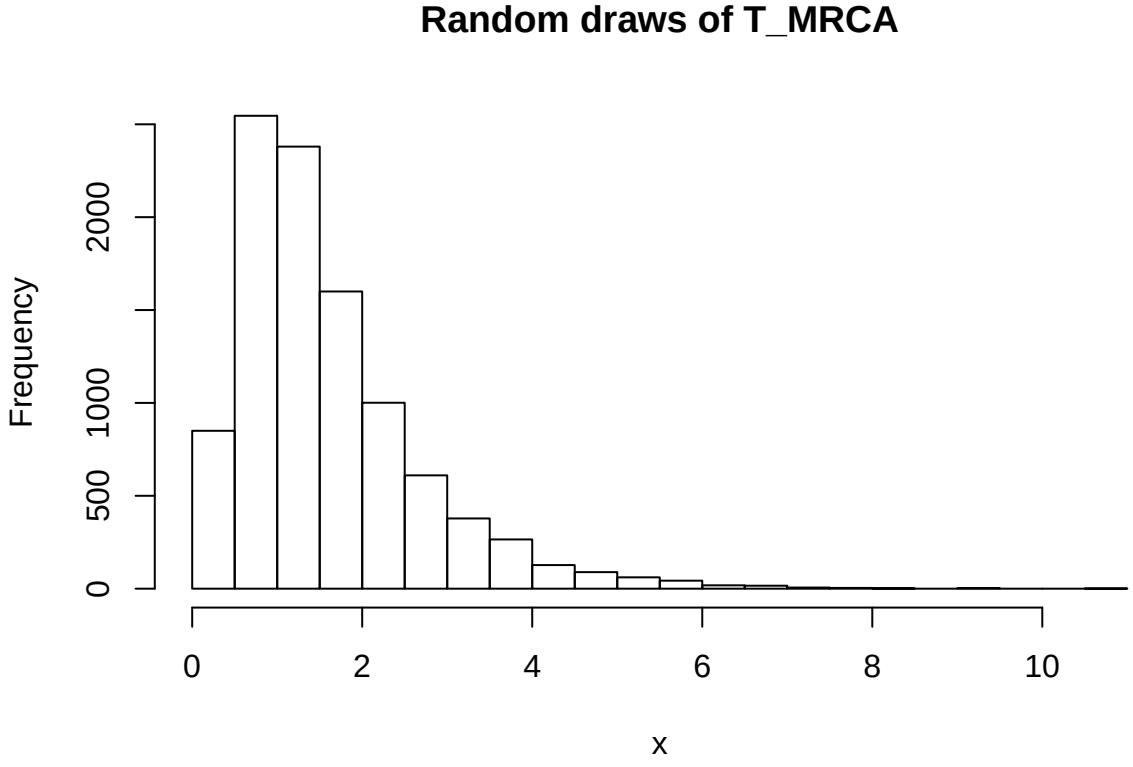

```
pphtype(0.5, T_MRCA_ph)
#> [1] 0.0812838
```

```
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] 1.27 1.34 2.13
```

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



Discrete Phase-Type Distributions

The class `disc_phase_type` contains:

- reward-transformed matrix (denoted as `P` in [HSB 2019]).
- vector of initial probabilities (α).
- defect (probability of not entering any transient state prior to absorption).

Total number of segregating sites S_{total}

The total number of segregating sites can be represented using a `disc_phase_type` class, this is, by means of a discrete phase-type distribution. For example, when `n`=4:

```
stotal_ph = disc_phase_type(n = 4)
stotal_ph
#> $subint_mat
#>      [,1] [,2]      [,3]
#> [1,]  0.4  0.3 0.2000000
#> [2,]  0.0  0.5 0.3333333
#> [3,]  0.0  0.0 0.6666667
#>
#> $init_probs
#>      [,1] [,2] [,3]
#> [1,]    1    0    0
#>
#> $defect
#> [1] 0
#>
#> attr(,"class")
#> [1] "disc_phase_type"
```

i-tons ξ_i

The i-tons ξ_i can also be represented using a discrete phase-type distribution. For example, the tripletons ξ_3 when `n`=6:

```
itons_ph = disc_phase_type(n = 6, itons = 3)
itons_ph
#> $subint_mat
#>           [,1]      [,2]       [,3]
#> [1,] 0.1428571 0.1071429 0.07142857
#> [2,] 0.0000000 0.2500000 0.16666667
#> [3,] 0.0000000 0.0000000 0.66666667
#>
#> $init_probs
#>      [,1] [,2] [,3]
#> [1,]  0.4  0.4    0
#>
#> $defect
#> [1] 0.2
#>
#> attr(,"class")
#> [1] "disc_phase_type"
```

User-defined sub-intensity matrix

As with the class `phase_type`, using the `disc_phase_type()` generator the user can specify the sub-intensity matrix, with optional initial probabilities (see `?disc_phase_type` for further information).

Mean and Variance

Factorial moments in general [Bladt M., Nielsen B.F. 2017]:

$$\boldsymbol{E}[\tau(\tau-1)...(\tau-k+1)] = k!\boldsymbol{\pi}\boldsymbol{T}^{k-1}(\boldsymbol{I}-\boldsymbol{T})^{-k}\boldsymbol{e}$$

It should be kept in memory, that the number of segregating sites is $S+1 \sim DPH_p(\boldsymbol{\alpha}, \boldsymbol{P})$, because we are not allowing the model to start in the absorbing state right away. For cases where the defect is not zero, the probability is not 1 but rather $1 - defect$.

```
mean(itons_ph)
#> [1] 1.666667
```

```
var(itons_ph)
#> [1] 2.311111
```

```
summary(itons_ph)
#>
#> Subintensity matrix:
#>           [,1]      [,2]       [,3]
#> [1,] 0.1428571 0.1071429 0.07142857
#> [2,] 0.0000000 0.2500000 0.16666667
#> [3,] 0.0000000 0.0000000 0.66666667
#>
#> Initial probabilities:
#>      [,1] [,2] [,3]
#> [1,]  0.4  0.4    0
#>
#> Defect:
#> [1] 0.2
#>
#> Mean: 1.666667
#>
#> Variance: 2.311111
```

Density and Distribution function

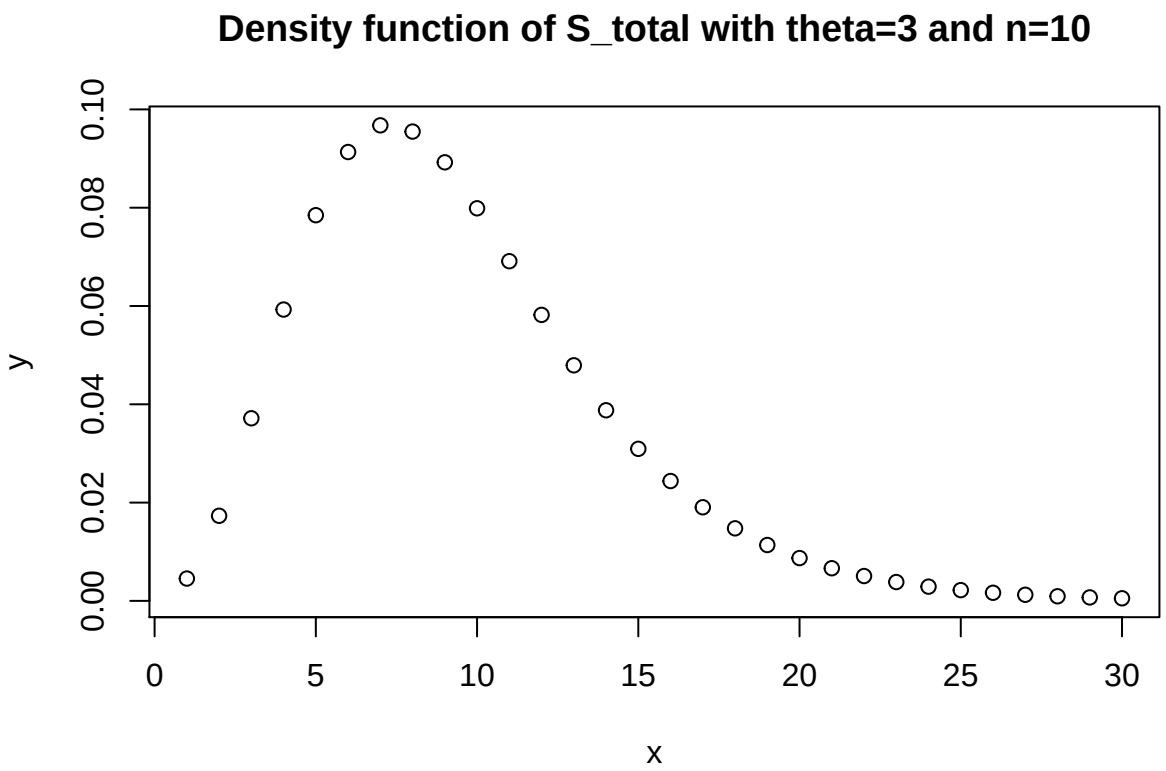
The `dphtype()` and `pphtype()` functions of `ticphasetype` can also accommodate the `disc_phase_type` class. For example, for S_{total} when $\theta = 3$:

```
S_tot_ph <- disc_phase_type(10, theta = 3)
summary(S_tot_ph)
#>
#> Subintensity matrix:
#>       [,1]      [,2]      [,3]      [,4]       [,5]       [,6]       [,7]
#>  [1,] 0.25 0.2045455 0.1636364 0.1272727 0.09545455 0.06818182 0.04545455
#>  [2,] 0.00 0.2727273 0.2181818 0.1696970 0.12727273 0.09090909 0.06060606
#>  [3,] 0.00 0.0000000 0.3000000 0.2333333 0.17500000 0.12500000 0.08333333
```

```
#>  [4,] 0.00 0.0000000 0.0000000 0.3333333 0.25000000 0.17857143 0.11904762
#>  [5,] 0.00 0.0000000 0.0000000 0.0000000 0.37500000 0.26785714 0.17857143
#>  [6,] 0.00 0.0000000 0.0000000 0.0000000 0.00000000 0.42857143 0.28571429
#>  [7,] 0.00 0.0000000 0.0000000 0.0000000 0.00000000 0.00000000 0.50000000
#>  [8,] 0.00 0.0000000 0.0000000 0.0000000 0.00000000 0.00000000 0.00000000
#>  [9,] 0.00 0.0000000 0.0000000 0.0000000 0.00000000 0.00000000 0.00000000
#>             [,8]       [,9]
#>  [1,] 0.02727273 0.01363636
#>  [2,] 0.03636364 0.01818182
#>  [3,] 0.05000000 0.02500000
#>  [4,] 0.07142857 0.03571429
#>  [5,] 0.10714286 0.05357143
#>  [6,] 0.17142857 0.08571429
#>  [7,] 0.30000000 0.15000000
#>  [8,] 0.60000000 0.30000000
#>  [9,] 0.00000000 0.75000000
#>
#> Initial probabilities:
#>      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
#> [1,]    1    0    0    0    0    0    0    0    0
#>
#> Defect:
#> [1] 0
#>
#> Mean: 9.486905
#>
#> Variance: 22.34481
```

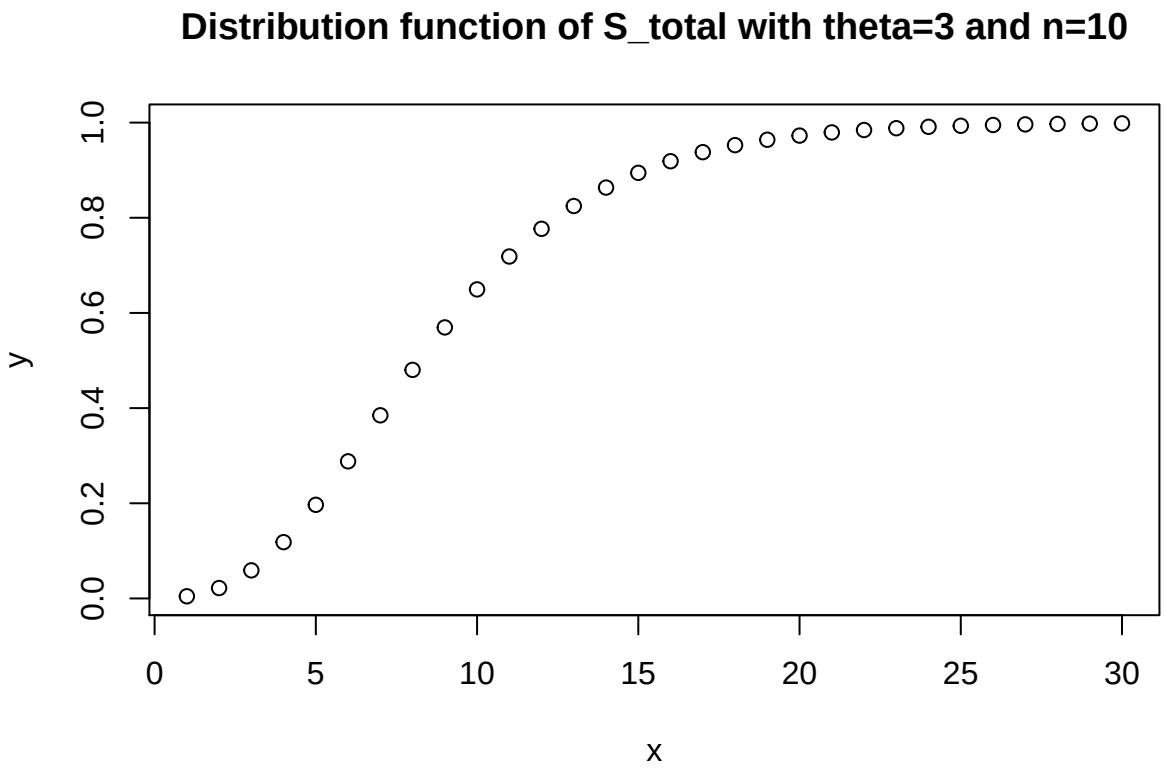
```
dphtype(3, S_tot_ph)
#> [1] 0.03715474
```

```
x <- 1:30
y <- dphtype(x, S_tot_ph)
plot(x, y)
title('Density function of S_total with theta=3 and n=10')
```




```
pphtype(15, S_tot_ph)
#> [1] 0.8944268
```

```
x <- 1:30
y <- pphtype(x, S_tot_ph)
plot(x, y)
title('Distribution function of S_total with theta=3 and n=10')
```

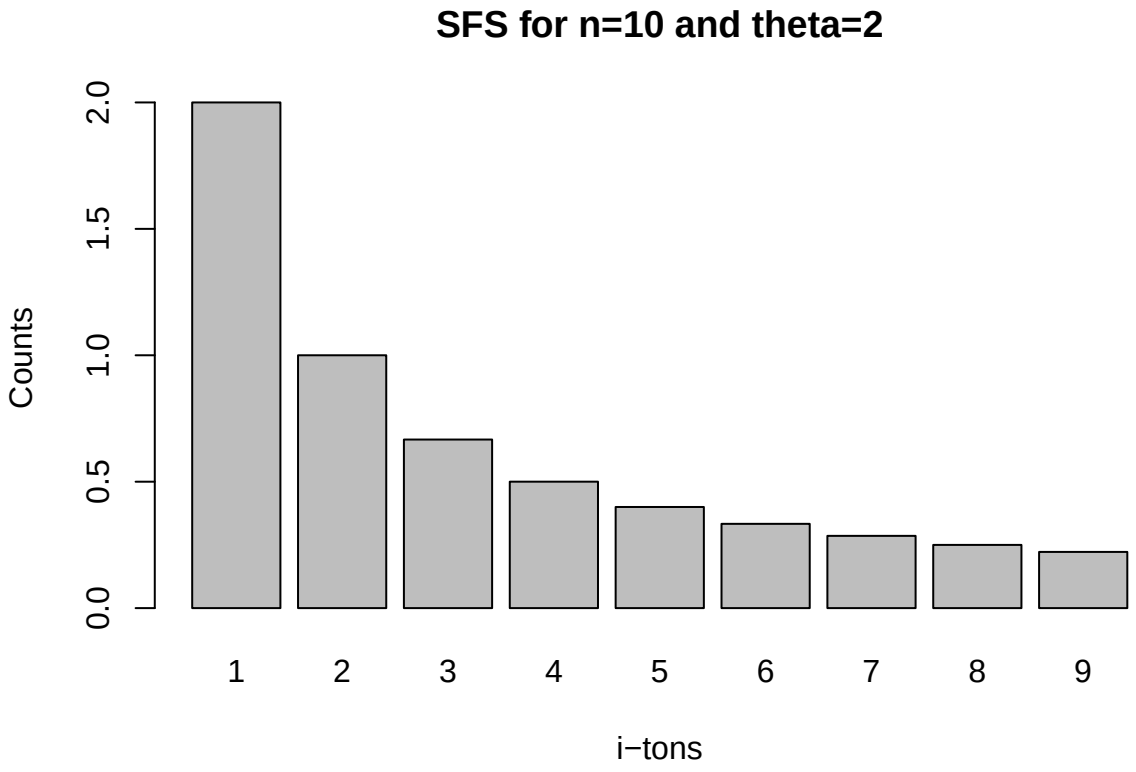


SFS

`ticphasetype` can also be used to calculate and plot the site-frequency spectrum (SFS) given a particular `n`. For example, when `n`=10:

```
(sfs_10 <- sfs(10))
#> $E_ksi
#> [1] 2.0000000 1.0000000 0.6666667 0.5000000 0.4000000 0.3333333 0.2857143
#> [8] 0.2500000 0.2222222
#>
#> $Var_ksi
#> [1] 3.1432981 1.8985261 1.4202570 1.1550265 1.4330159 0.8047619 0.7305367
#> [8] 0.6689815 0.6172840
```

```
sfs(10, plot = TRUE)
```



Tail-Statistic

Koskela 2018 showed that normalized singletons together with a so called `tail-statistic` can help distinguish between traditional Kingsman coalescent and multiple merger coalescent. It is defined as:

$$S_{k+} = \sum_{i=k}^{n-1} \xi_i$$

And this can be received from Phase type distribution very easily by summing all rewards for $itons >= k$ and using them to transform the subintensity matrix.

In `ticphasetype` this is implemented by setting the `tail_stat` argument to `TRUE` in the `disc_phase_type()` function. For example, the mean of the tail statistic can be calculated using:

```
# Mean of S_10+ when n=15
mean(disc_phase_type(15, tail_stat = T, itons = 10)) - 1
#> [1] 0.8451881
# Mean of S_10+ when n=10
mean(disc_phase_type(20, tail_stat = T, itons = 10)) - 1
#> [1] 1.437543
```

Theory behind Discrete Phase-Type distributions

When computing T_{MRCA} or T_{Total} using phase-type representation, in order to reach the absorbing state the model has to first go through all the transient states. This is not neccesarily true if we are interested in the number of segregating sites and the frequency spectrum of alleles in the sample.

There are scenarios where in order to reach the absorbing state, some transient states might be skipped. In some cases there is even a probability (defect α_d) of not entering any 'nonzero' state prior to absorption.

The rate matrix has to cover up all the different transitions with corresponding rates. Then, in order to find the number of certain type of polymorphism (eg. singletons, doubletons, ...) the transition matrix has to be transformed with a reward vector to obtain $\boldsymbol{T^*} = \{t^*_{ij}\}$

$$t^*_{ij} = -\frac{t_{ii}}{r(i)} p_{ij} \; i \neq j \; , \; and \; t^*_i = -\frac{t_{ii}}{r(i)} p_{ij}$$

The diagonal (t_{ii}) contains values such that rows of the intensity matrix $\Lambda^* = \begin{pmatrix} \boldsymbol{T^*} & \boldsymbol{t^*} \\ \boldsymbol{0} & 0 \end{pmatrix}$ sum to zero.

Elements p_{ij} in equation (1) come from transition matrix $\boldsymbol{P}$:

$$\boldsymbol{P} = \boldsymbol{Q}^{++} + \boldsymbol{Q}^{+0}(\boldsymbol{I} - \boldsymbol{Q}^{00})^{-1}\boldsymbol{Q}^{0+}$$

where $\boldsymbol{Q}^{++}, \boldsymbol{Q}^{+0}, \boldsymbol{Q}^{0+}, \boldsymbol{Q}^{00}$ are components of

$$\boldsymbol{Q} = \begin{pmatrix} \boldsymbol{Q}^{++} & \boldsymbol{Q}^{+0} \\ \boldsymbol{Q}^{0+} & \boldsymbol{Q}^{00} \end{pmatrix}$$

which is a matrix consisting of transition probabilities

$$q_{ij} = -\frac{t_{ij}}{t_{ii}} \; i \neq j, \; q_{ii} = 0$$

The superscipts (+ or 0) in (2) and (3) correspond to the indices of nonzero and zero states respectively.

Finally, the probability distribution $P(S = k)$ is in general:

$$P(S = k) = \boldsymbol{\pi} \boldsymbol{P}^k \boldsymbol{p}$$

where

$$\boldsymbol{P} = (\boldsymbol{I} - \frac{2}{\theta}\boldsymbol{T}), \; \boldsymbol{p} = \boldsymbol{e} - \boldsymbol{P}\boldsymbol{e}$$

However in this case $\boldsymbol{T}$ should be substituted by the new reward transformed matrix $\boldsymbol{T^*}$.