# First assignment

*Andrej Balaz*

In this assignment, we should use phase-type distributions to calculate descriptive statistics of $S_{total}$.

First, we imported libraries…

```r
library(tidyverse)
library(zeallot)
library(expm)
library(gmp)
```

…, and functions from the previous exercises.

```r
source('~/Projects/phase_type/1_BlockCountingProcess.R')
source('~/Projects/phase_type/2_rewardtransformation.R')
```

Then we defined input parameters. These are: $n$ using the variable n and $\theta$ using the variable theta. The parameter $n$ represents the number of sequences, we are considering and the parameter $\theta$ represents the mutation rate scaled by population size.

```r
n = 25
theta = 2
```

Next, we obtained the rate matrix and the state space matrix from the block counting process. These matrices were then used to calculate expected site frequency spectrum, probability mass function of variable $S_{total}$ and summary statistics of the variable $S_{total}$.

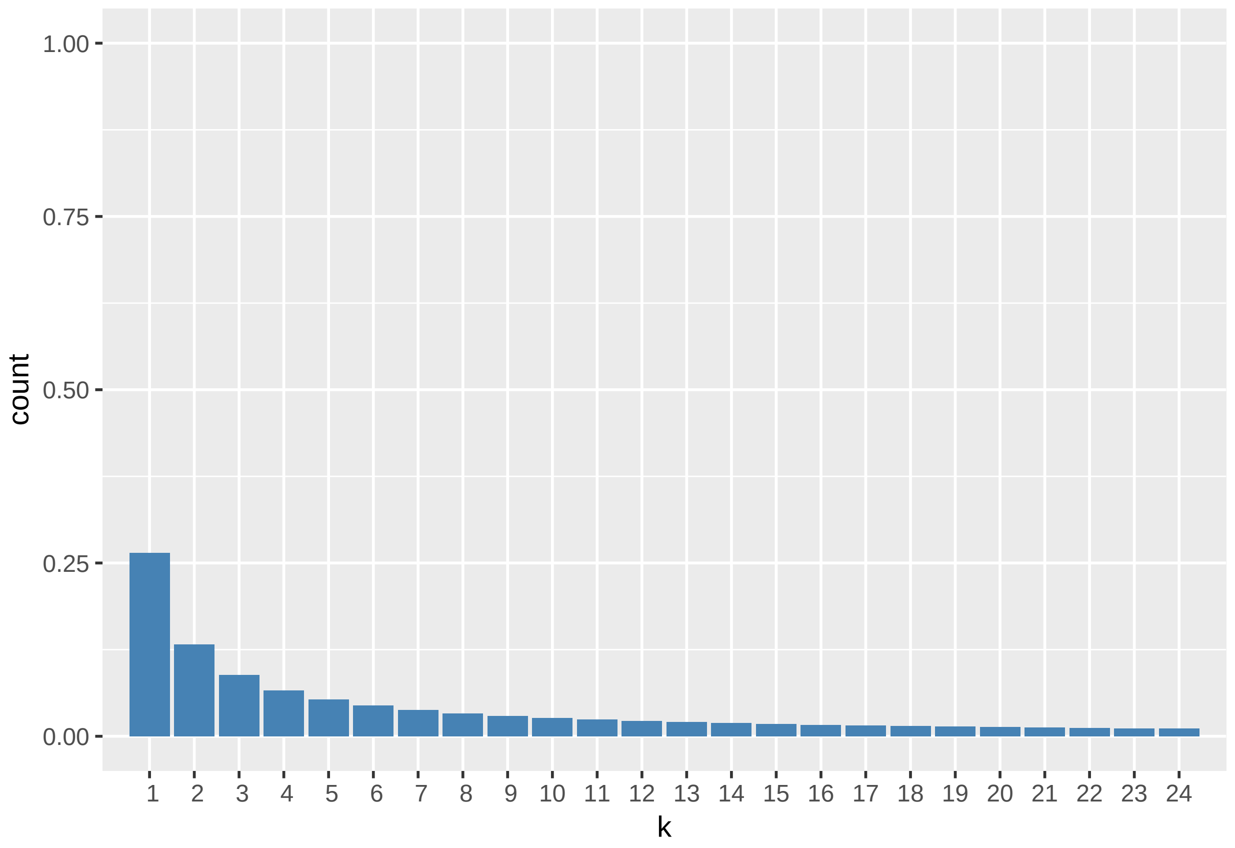
```r
c(RateM, StSpM) %<-% RateMAndStateSpace(n)
```

In this step, we calculated the expected site frequency spectrum given our parameters $n$ and $\theta$.

```r
# calculate xi_i
xi_vec = c()
for (i in 1:(n-1)) {
  tmp = rewardtransformparm(rewards = StSpM[1:(dim(StSpM)[1]-1), i],
                            initprob = c(1, rep(0, (dim(StSpM)[1]-1)-1)),
                            subintensemat = RateM[1:(dim(StSpM)[1]-1), 1:(dim(StSpM)[1]-1)])

  # xi_i + 1 ~ DPH(pi, P)
  pi = tmp$newinitprob
  P = solve(diag(length(pi)) - 2/theta * tmp$newsubintensitymatrix)

  # E[xi_i + 1]
  xi = pi %*% solve((diag(length(pi)) - P)) %*% rep(1, length(pi))
  xi_vec = c(xi_vec, xi+tmp$defect-1)

  # clear envirnoment
}
rm(list = c('tmp', 'pi', 'P', 'xi', 'i'))
```

The expected site frequency spectrum was visualized.

```r
# visualize SFS
ggplot() +
  geom_bar(mapping = aes(x = 1:(n-1), y = xi_vec/sum(xi_vec)),
           stat = 'identity', fill="steelblue") +
  scale_x_discrete(name = 'k', limits=1:(n-1)) +
  scale_y_continuous(name = 'count', limits = c(0, 1))
```



The number of summary statistics was calculated using the site frequency spectrum.

The expected number of segregating sites:

```
sum(xi_vec)
```

```
## [1] 7.551916
```

The expected number of singletons, doubletons, tripletons…:

```
xi_vec
```

```
##  [1] 2.00000000 1.00000000 0.66666667 0.50000000 0.40000000 0.33333333
##  [7] 0.28571429 0.25000000 0.22222222 0.20000000 0.18181818 0.16666667
## [13] 0.15384615 0.14285714 0.13333333 0.12500000 0.11764706 0.11111111
## [19] 0.10526316 0.10000000 0.09523810 0.09090909 0.08695652 0.08333333
```

The expected tail statistics:

```
k = 2                                           # tail parameter
sum(xi_vec[k:length(xi_vec)])
```

```
## [1] 5.551916
```

```
rm(list = c('k'))
```

The expected pairwise difference:

```
summands = c()
for (i in 1:n-1) {
  summands = c(summands, (i*(n-i)*xi_vec[i]))
}

(1/choose(n, 2)) * sum(summands)
```

```
## [1] 2
```

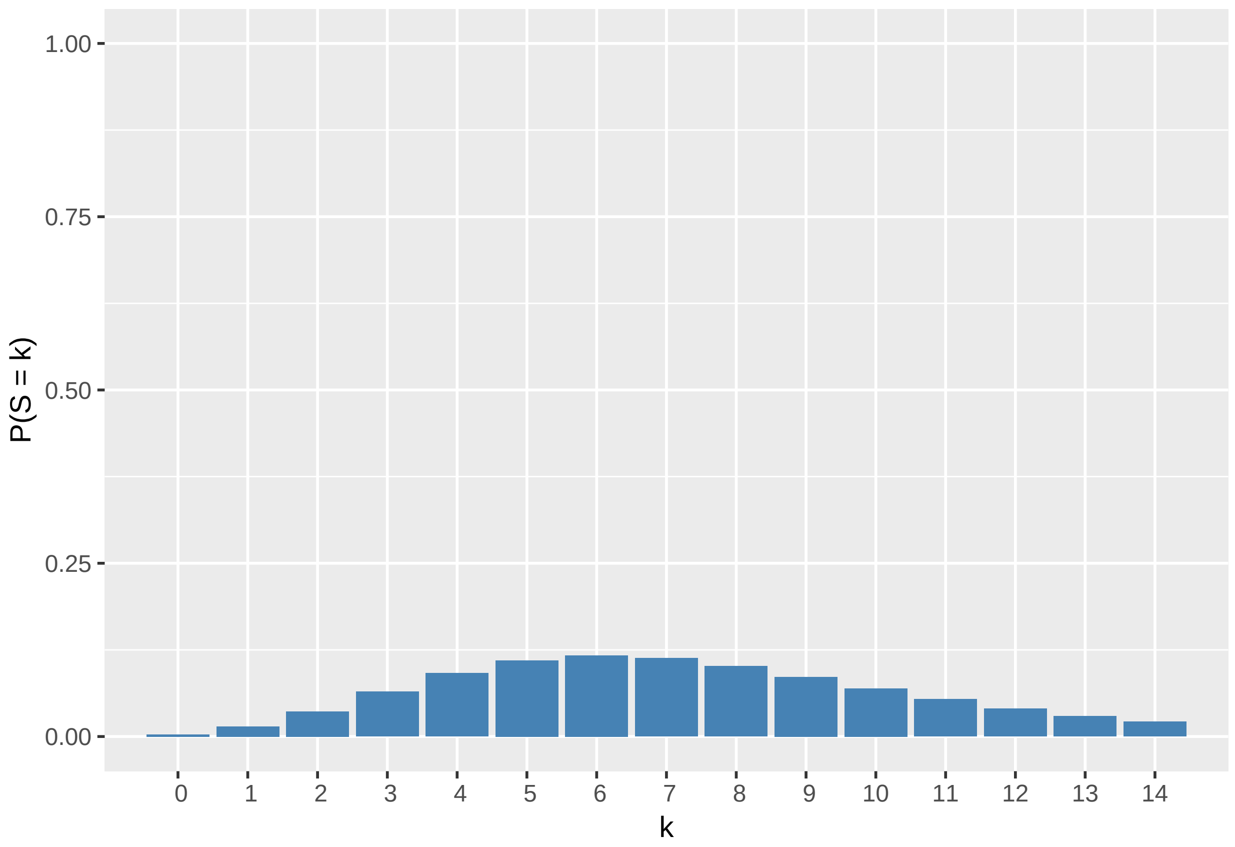
```
rm(list = c('summands', 'i'))
```

Next, we also calculated the probability mass function of random variable $S_{total}$…

```
c(pi, T, defect) %<-% rewardtransformparm(rewards = rowSums(StSpM[1:(dim(StSpM)[1]-1),
                                                              1:(dim(StSpM)[2]-1)]),
                                          initprob = c(1, rep(0, (dim(StSpM)[1]-1)-1)),
                                          subintensemat = RateM[1:(dim(StSpM)[1]-1),
                                                                1:(dim(StSpM)[1]-1)])

P = solve(diag(length(pi)) - 2/theta * T)

p = rep(1, (dim(StSpM)[1]-1)) - rowSums(P)
probs = c()
for (i in 1:15) {
  prob = pi %*% (P %^% (i-1)) %*% p
  probs = c(probs, prob)
}
rm(list = c('prob', 'i', 'p', 'defect'))
```

…and visualized the results. Since we were calculating the distribution random variable $S_{total}+1$, we needed to shift $k$ by one to the left.

```
# P(S+1 = k) => P(S = k-1)
ggplot() +
  geom_bar(mapping = aes(x = 0:(length(probs)-1), y = probs),
           stat = 'identity', fill="steelblue") +
  scale_x_discrete(name = 'k', limits=0:14) +
  scale_y_continuous(name = 'P(S = k)', limits = c(0, 1))
```



Next, we calculated summary statistics of the distribution for $S_{total} + 1$. For this we needed to calculate moments of the distribution.

First, we calculated the factorial moments using the Theorem 1.2.69 (BN p. 35).

```r
factorial_moments = c()
for (k in 1:4) {
  fct_moment = factorial(k) %*% pi %*% (P %^% (k-1)) %*%
    (solve((diag(dim(P)[1]) - P)) %^% k) %*% rep(1, dim(P)[1])
  factorial_moments = c(factorial_moments, fct_moment)
}
rm(list = c('fct_moment', 'k'))
```

Afterwards, we converted the factorial moments to the raw moments (https://en.wikipedia.org/wiki/Factorial_moment#Calculation_of_moments).

```r
raw_moments = c()
for (r in 1:4) {
  stirling2 = Stirling2.all(r)
  sum = 0.0
  for (j in 1:r) {
    sum = sum + (as.integer(stirling2[j]) * factorial_moments[j])
  }
  raw_moments = c(raw_moments, sum)
}
rm(list = c('stirling2', 'sum', 'r', 'j'))
```

The raw moments were then used to calculate central moments (https://www.vosesoftware.com/riskwiki/Rawmomentsversuscentralmoments.php) and standardized moments (https://en.wikipedia.org/wiki/

Standardized_moment).

```
central_moment_1st = 0
central_moment_2nd = raw_moments[2] - (raw_moments[1]^2)
central_moment_3rd = raw_moments[3] - 3*raw_moments[1]*raw_moments[2] +
  2*(raw_moments[1]^3)
central_moment_4th = raw_moments[4] - 4*raw_moments[1]*raw_moments[3] +
  6*(raw_moments[1]^2)*raw_moments[2] - 3*(raw_moments[1]^4)
central_moments = c(central_moment_1st, central_moment_2nd,
                    central_moment_3rd, central_moment_4th)

standardized_moment_1st = 0
standardized_moment_2nd = 1
standardized_moment_3rd = central_moment_3rd/(central_moment_2nd^(3/2))
standardized_moment_4rd = central_moment_4th/(central_moment_2nd^(4/2))
standardized_moments = c(standardized_moment_1st, standardized_moment_2nd,
                         standardized_moment_3rd, standardized_moment_4rd)

rm(list = c('central_moment_1st', 'central_moment_2nd',
            'central_moment_3rd', 'central_moment_4th'))
rm(list = c('standardized_moment_1st', 'standardized_moment_2nd',
            'standardized_moment_3rd', 'standardized_moment_4rd'))
```

The calculated moments directly corresponds to the widely used summary statistics for distributions - mean, variance, skewness and kurtosis. These values holds for the random variable $S_{total} + 1$.

```
cat(paste0('E[S+1] = ', raw_moments[1],
           '\nVar[S+1] = ', central_moments[2],
           '\nSkew[S+1] = ', standardized_moments[3],
           '\nKurt[S+1] = ', standardized_moments[4]))
```

```
## E[S+1] = 8.55191635550701
## Var[S+1] = 13.968409969871
## Skew[S+1] = 0.881526666038412
## Kurt[S+1] = 4.39242857031414
```

Since variance, skewness and kurtosis are based on the central moments, shifting the distribution does not change their corresponding values. This is not true for the mean. Fortunatelly, for the mean we can use the linearity of expectation rule and adjust our result accordingly.

```
cat(paste0('E[S] = ', raw_moments[1] - 1,
           '\nVar[S] = ', central_moments[2],
           '\nSkew[S] = ', standardized_moments[3],
           '\nKurt[S] = ', standardized_moments[4]))
```

```
## E[S] = 7.55191635550701
## Var[S] = 13.968409969871
## Skew[S] = 0.881526666038412
## Kurt[S] = 4.39242857031414
```

## Estimators of $\theta$

There are two widely used estimators of theta, Watterson's estimator $\hat{\theta}_w$ and Tajima's estimator $\hat{\theta}_t$. Watterson's estimator is defined as:

$$\hat{\theta}_w = \frac{S}{\sum_{k=1}^{n-1} 1/k}$$

where $S$ is the number of segregating sites.

Tajima's estimator is defined as:

$$\hat{\theta}_t = \frac{\sum_{i<j} d_{ij}}{n \cdot (n-1)/2}$$

where $d_{ij}$ is the number of differences between sequence $i$ and sequence $j$.

Both of these estimators provide an unbiased estimate of $\theta$ under the assumption of infinite sites model. The difference is that a selective sweeps tend to have stronger effect on Tajima's estimator and the balancing selection tends to have a stronger effect on Watterson's estimator. Therefore, I think we should not prefer one over the another, because using both of them can give us some additional information about the possible selection process in our sample and point out if our sample violates the underlying assumptions of the model.

---

Testing of the code

The code in this file was tested with $n = 4$ and $\theta = 2$. Where it was possible, the results were compared with values produced by Wakeley's equations. All values were checked for consistency and reasonability.