Module 2: Solutions

Exercise I (Basics)

a) Make the following three vectors without using the `c()` command: $x = (1, 1, 1, 1, 1)$, $y = (1, 3, 5, 7, 9)$, $z = (2, 2^2, 2^3, 2^4, 2^5)$ (Hint: `seq()` and `rep()` may be useful – check the help.)

```
x <- rep(1, 5)
y <- seq(1, 9, by = 2)
z <- 2^(1:5)
```

b) Make a matrix `X` with columns `x`, `y`, and `z`.

```
X <- cbind(x, y, z)
```

c) Try to add two vectors and/or matrices that do not match in dimensions, and see if you can figure out what R does.

```
rep(1, 4) + 1:2 # Recycles the shorter one until vectors are of same length
```

```
## [1] 2 3 2 3
```

```
rep(1, 5) + 1:2 # Warns if the length doesn't fit with integer repetition
```

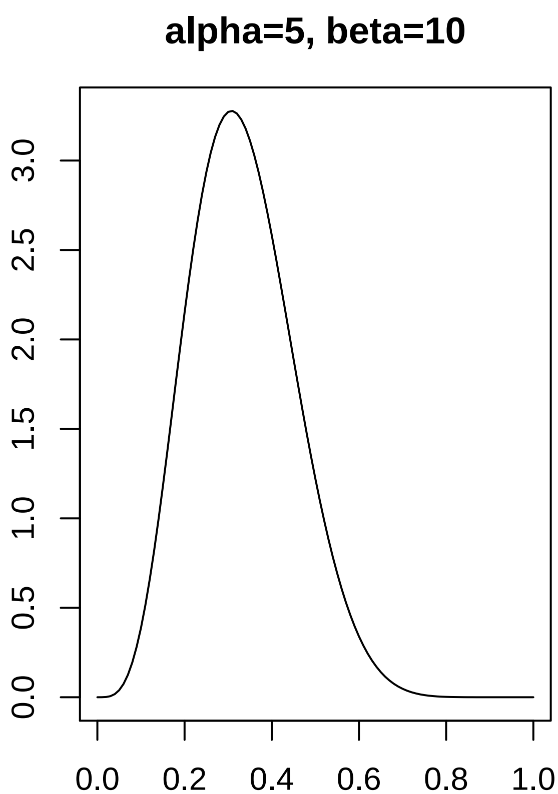
```
## Warning in rep(1, 5) + 1:2: longer object length is not a multiple of
## shorter object length
```

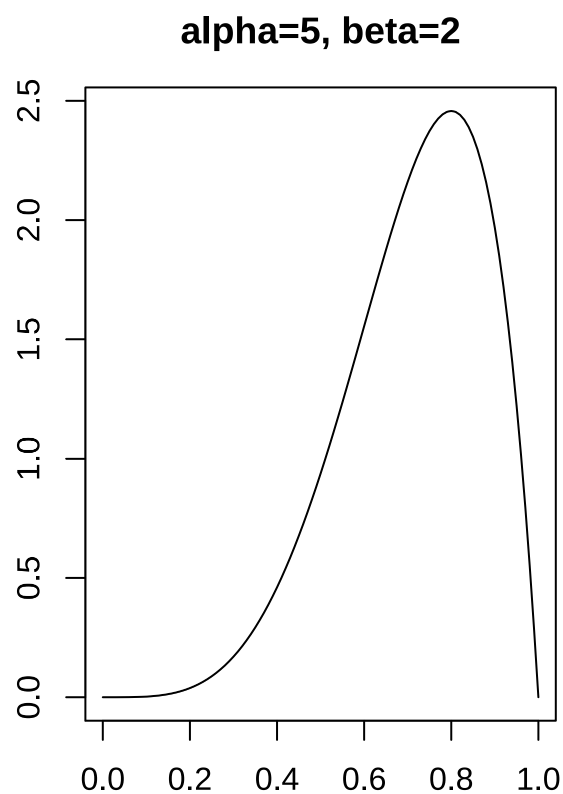
```
## [1] 2 3 2 3 2
```

Exercise II (Distributions)

a) Plot the density function for the beta distribution for a few different values of the two shape parameters (note the support of the density in the help file).

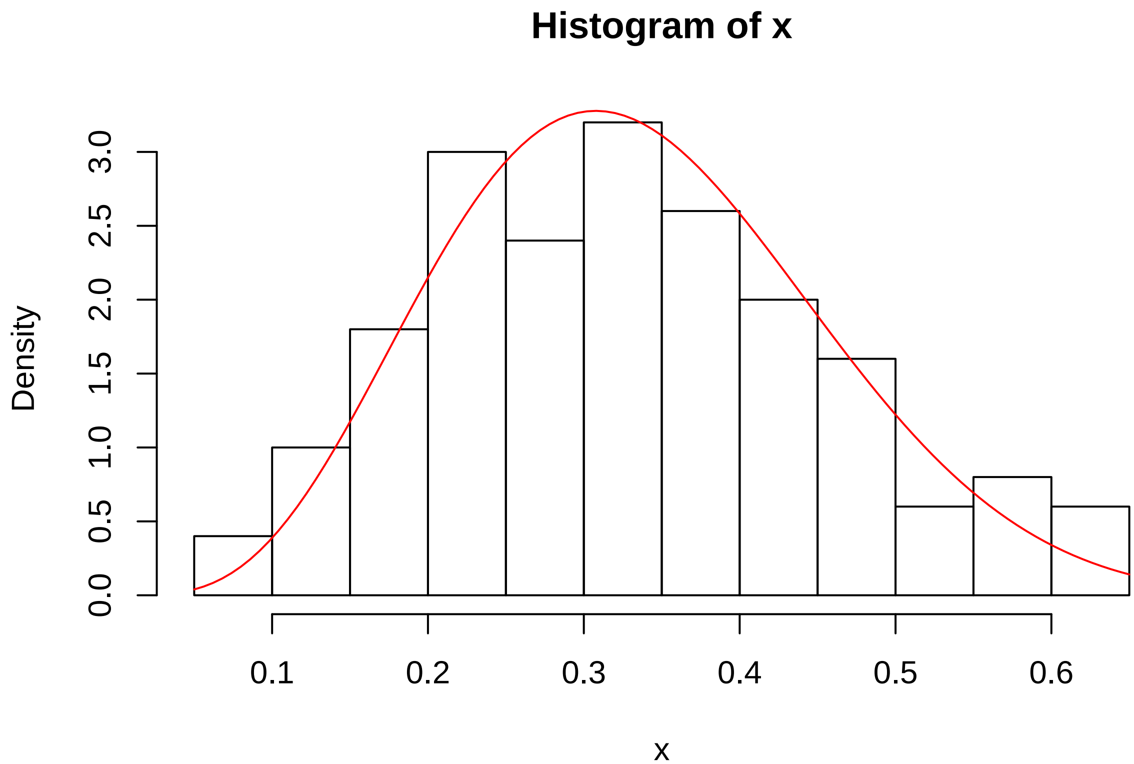
```
par(mfrow=c(1,2),mar=c(3,3,3,1))
curve(dbeta(x, 5, 10), main = "alpha=5, beta=10")
curve(dbeta(x, 5, 2), main = "alpha=5, beta=2")
```





b) Generate 100 realizations from a beta distribution, and make a histogram. Add the theoretical density to the histogram. (Hint: the `curve` command has an argument `add=TRUE` that allows you to add a plot on top of the histogram; note that this is only useful if the histogram is normalized to integrate to one, which can be achieved by including the argument `probability=TRUE` to the `hist` command)

```r
x <- rbeta(100, 5, 10)
hist(x, prob = TRUE)
curve(dbeta(x, 5, 10), add = TRUE, col = "red")
```



c) Calculate the average of the 100 realizations. Can you guess what the theoretical mean is for your parameter values (feel free to repeat the experiment and/or increase the number of realizations)? Using other parameter values can you guess the general formula for the mean in terms of the parameters (without looking it up somewhere)?

```r
mean(x) # ~ 5/(5+10) = 0.33333
```

```
## [1] 0.3285543
```

```r
x <- rbeta(100, 2, 2)
mean(x) # ~ 2/(2+2) = 0.5
```

```
## [1] 0.5009329
```

```r
x <- rbeta(100, 6, 2)
mean(x) # ~ 6/(6+2) = 0.75
```

```
## [1] 0.7604999
```

Exercise III (Functions and loops)

a) Make a function with a for loop that can calculate the product of all the entries of an input vector. Compare with the built-in function `prod` (don't call your function `prod`, or you won't able to use the built-in function easily).

```r
myprod <- function(x){
  p <- 1
  for(xi in x){
    p <- p*xi
  }
  return(p)
}
x <- c(1, 3.2, -1)
myprod(x) == prod(x)
```

```
## [1] TRUE
```

b) Make a function that will calculate the Fibonacci numbers up to n (an input parameter). Does it handle n=1 and 2 correctly? (Hint: An `if` statement may be useful here.)

```r
fib <- function(n){
  if(length(n)!=1 || n<1){
    stop("n must be a single number >= 1.")
  }
  n <- floor(n)
  rslt <- rep(1, n)
  if(n >= 3){
    for(i in 3:n){
      rslt[i] <- rslt[i-2] + rslt[i-1]
    }
  }
  return(rslt)
}
fib(1)
```

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

```r
fib(2)
```

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

```r
fib(7)
```

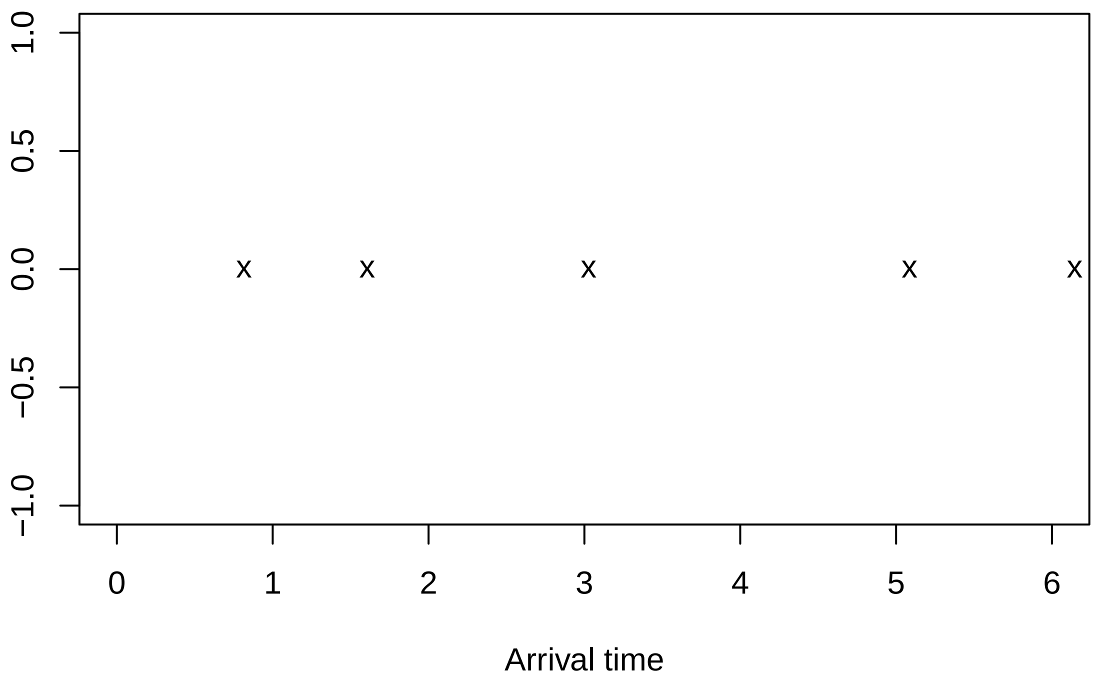
```
## [1]  1  1  2  3  5  8 13
```

Exercise IV (Arrival times)

Consider a process of arrivals on the real line where the inter-arrival time is exponentially distributed with rate 1. E.g.

```r
set.seed(54321) # For reproducibility
t1 <- rexp(1)
t2 <- t1 + rexp(1)
t3 <- t2 + rexp(1)
t4 <- t3 + rexp(1)
```

```r
t5 <- t4 + rexp(1)
plot(c(t1, t2, t3, t4, t5), rep(0, 5), pch = "x",
     xlim = c(0,6), ylab = "", xlab = "Arrival time")
```



- Make a function f that sequentially generates inter-arrival times and counts the number of arrivals in the interval $[0, T]$ where T is the only parameter of the function. E.g. for the example above $f(1) = 1$, $f(2) = f(3) = 2$, $f(4) = f(5) = 3$, $f(6) = 4$. (Hint: Either use `while`-loop instead of `for`-loop or simply use a conservatively long `for`-loop – maybe `1:100` – and break out of the loop once T is exceeded.)

```r
f <- function(T){
  t <- 0
  i <- 0
  while(t < T){
    i <- i + 1
    t <- t + rexp(1)
  }
  return(i-1)
}
```

- Generate 1000 realizations of the random number of arrivals in $[0, 5]$. Calculate the empirical mean and variance. Compare with a Poisson distributed random variable with rate parameter 5.

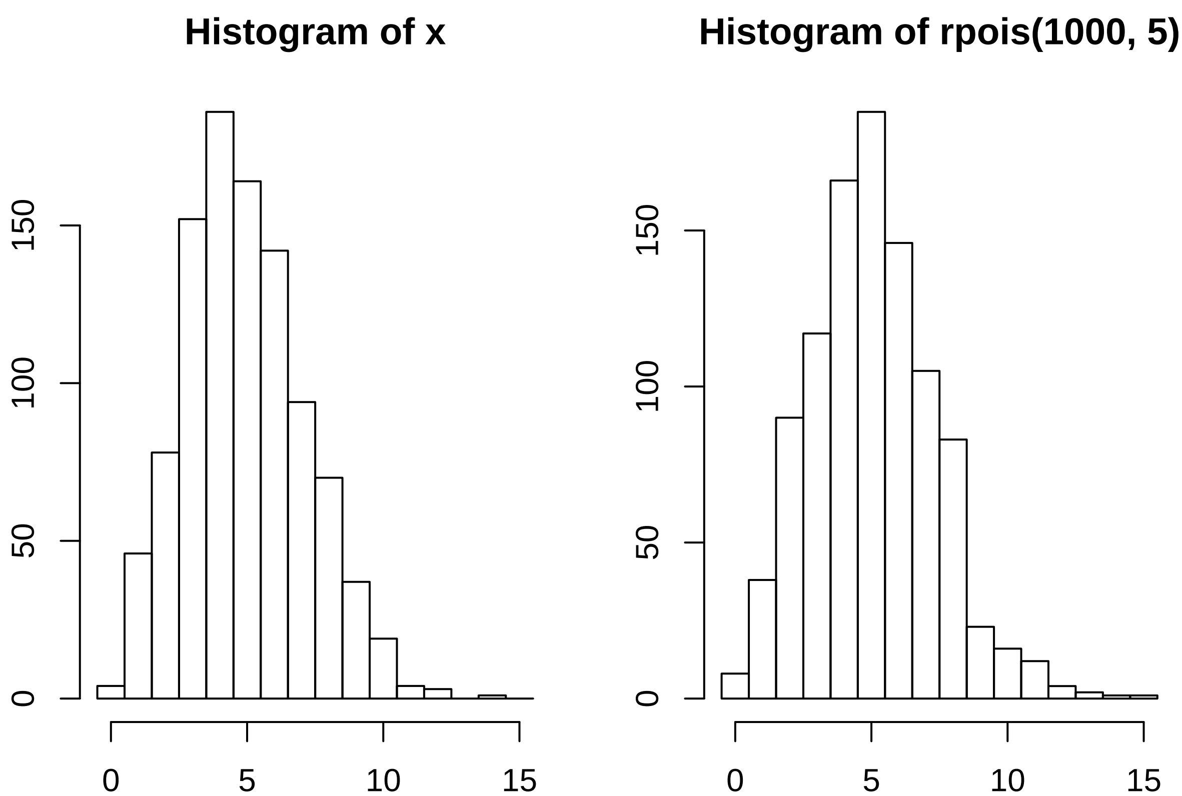
```r
set.seed(42)
x <- replicate(1000, f(5))
mean(x) # Close to 5 = Poisson mean
```

```
## [1] 4.909
```

```r
var(x) # Close to 5 = Poisson var
```

```
## [1] 4.915635
```

```r
par(mfrow=c(1,2), mar=c(3,3,3,1))
hist(x, breaks = (0:16)-0.5)
hist(rpois(1000, 5), breaks = (0:16)-0.5)
```

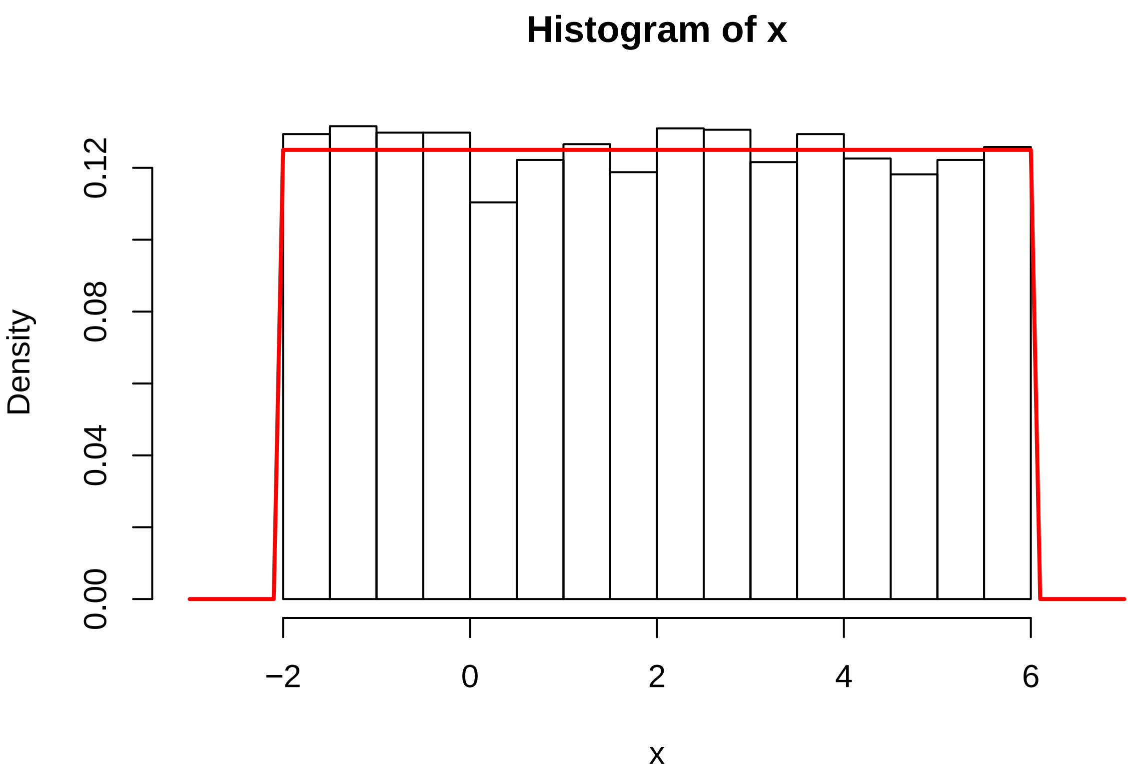


Exercise V (Uniform distribution)

- Make a short report in Rmarkdown about the uniform distribution on $[A, B]$ where $A < B$ should be variables definied in the very beginning of the document. The report should at least include:
  - A histogram of a large sample from the distribution.
  - A plot of the density (preferably overlayed on the histogram).
  - The difference between the sample mean and the theoretical mean as well as the difference between the sample variance and the theoretical variance. (Hint: To calculate the theoretical mean and variance think about the relation between Unif(A,B) and Unif(0,1) and use the results from module 1.)
- Rerun your report with a different choice of A, B and check that everything is still correct.

```r
A <- -2
B <- 6
theo_mean <- (A+B)/2 # Mean
l <- B-A # Length
theo_var <- 1/12*l^2 # Var
```

```r
x <- runif(1e4, A, B) # 10,000 samples
hist(x, prob=TRUE, xlim = c(A-1, B+1))
curve(dunif(x, A, B), from = A-1, to = B+1, add = TRUE, col = "red", lwd = 2)
```




```r
mean(x) - theo_mean
```

```
## [1] -0.02633177
```

```r
var(x) - theo_var
```

```
## [1] 0.03574004
```