Exercises in R – with solutions

Exercise 1 – vectors

Produce the following vectors in R:

- $(1, 2, 3, \ldots, 10)$
- $(1, 2, \ldots, 10, 8, 7, \ldots., 1)$
- $(2, 4, 6, \ldots, 20)$
- $(1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3)$

Solution

```
x1 = seq(1, 10)
print(x1)
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
c(x1, seq(8, 1, by=-1))
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10  8  7  6  5  4  3  2  1
```

```
x1 * 2
```

```
##  [1]  2  4  6  8 10 12 14 16 18 20
```

```
x2 = seq(1, 3)
c(x2, x2, x2, x2)
```

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

End of Solution

Exercise 2 – sampling random variables

Set $\lambda = 2$ and $n = 20$. Take a random sample of size n from Exponential(λ) and name it x. Then for this sample calculate the vector of values

$$f(x) = \lambda e^{-\lambda\ x}.$$

Hint: You should google "r exponential random number" or something similar and find the R command that generates exponential random variables.

Solution By googling, we use the function `rexp()`

```
lam = 2
n = 20
x = rexp(n, rate=lam)
print(x)
```

```
##  [1] 0.2293756 0.2771044 0.1998351 0.4670808 0.7280048 1.6373509 2.8692490
##  [8] 0.2109508 0.2514512 0.8381356 0.1643639 0.1428593 0.3518570 1.7462932
## [15] 0.5464407 1.1067469 0.5186806 0.5294515 1.4185572 0.1899214
```

```
y = lam * exp(-lam * x)
print(y)
```

```
##  [1] 1.26414501 1.14905318 1.34108227 0.78583036 0.46632964 0.07565629
##  [7] 0.00643920 1.31159713 1.20954553 0.37414044 1.43967776 1.50294807
## [13] 0.98948889 0.06084417 0.67049819 0.21863607 0.70877722 0.69367215
## [19] 0.11718900 1.36793773
```

End of Solution

Exercise 3 – vector operation

Produce two distinct samples from $N(0,1)$ of size 10 and name them x and y.
Produce the vector of differences

$$(x_2 - y_1, x_3 - y_2, \ldots, x_{10} - y_9),$$

where $x_i,\ y_j$ are the i^{th} and j^{th} components of x and y respectively.

Solution By googling, we use the function `rnorm()` to generate normal random variables.

```
n = 10
x = rnorm(n)
y = rnorm(n)
z = x[2:n] - y[1:(n-1)]
```

A further exploration of properties of `z`:

```
cat(mean(z), sd(z))
```

```
## 0.3187368 1.73231
```

Does this make sense? Try this with $n = 100, 10000, and 1000000$. **End of Solution**

Exercise 4 – matrix operation

Produce the following matrix, A:

$$\begin{pmatrix} 2 & 3 & 4 \\ -1 & 0 & -2 \\ 2 & 7 & 1 \end{pmatrix}$$

and then compute A^2 and A^{-1}.

Solution

```
A = matrix(c(2, -1, 2, 3, 0, 7, 4, -2, 1), nrow=3)
print(A)
```

```
##      [,1] [,2] [,3]
## [1,]    2    3    4
## [2,]   -1    0   -2
## [3,]    2    7    1
```

```
A %*% A
```

```
##      [,1] [,2] [,3]
## [1,]    9   34    6
## [2,]   -6  -17   -6
## [3,]   -1   13   -5
```

```
solve(A)
```

```
##             [,1]       [,2]       [,3]
## [1,] -1.5555556 -2.7777778  0.6666667
## [2,]  0.3333333  0.6666667  0.0000000
## [3,]  0.7777778  0.8888889 -0.3333333
```

End of Solution

Exercise 5 – function with for loop

Write a function that for given a and n it will return the following sum:

$$a + 2a^2 + 3a^3 + \ldots + na^n.$$

Hence, verify that $1 + 2 + 3 + ... + 100 = 5050$.

Solution

```r
func1 = function(a, n) {
  y = 0
  for (i in 1:n) {
    y = y + i * a^i
  }
  return(y)
}
func1(1, 100)
```

```
## [1] 5050
```

Or

```r
func2 = function(a, n) {
  x = seq(1, n)
  return(sum( x * (a^x) ))
}
func2(1, 100)
```

```
## [1] 5050
```

End of Solution

Exercise 6 – for loop with break

Write a function that simulates the experiment of rolling a dice until one gets a 6. The outcome of the experiment is the number of rolls it requires to get a 6. Repeat your experiment a large number of times and report the average number of rolls it requires to get a 6. Hint: you may need to use `break`, which breaks the execution of a `for`. You can google “r for loop break” to read about its usage. You can also assume that there is a maximum number of rolls we will try, i.e. we give up after say, 1000, rolls.

Solution

```r
get6 = function() {
  for (i in 1:1000000) { # try for a maximum of 1000000 rolls of the die
    if (sample(1:6, 1) == 6) { # if we get a 6 on this roll, stop the floor look
      break
    }
  }
  # report the last index of the for loop, which is the number of rolls so far
  return(i)
```

```
}

n = 1000
x = rep(NA, n) # we create an empty vector of length n to store n outcomes
for (i in 1:n) {
  x[i] = get6() # we store each outcome of the experiment
}
mean(x)
```

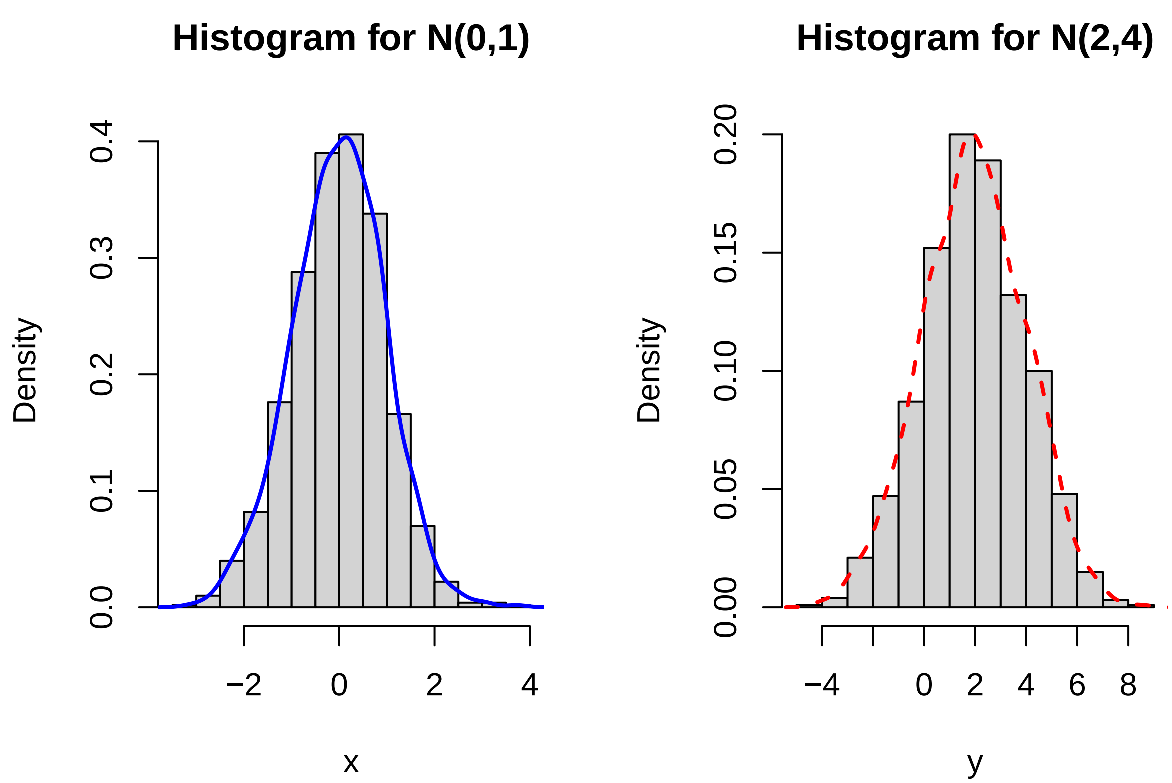
```
## [1] 6.378
```

End of Solution

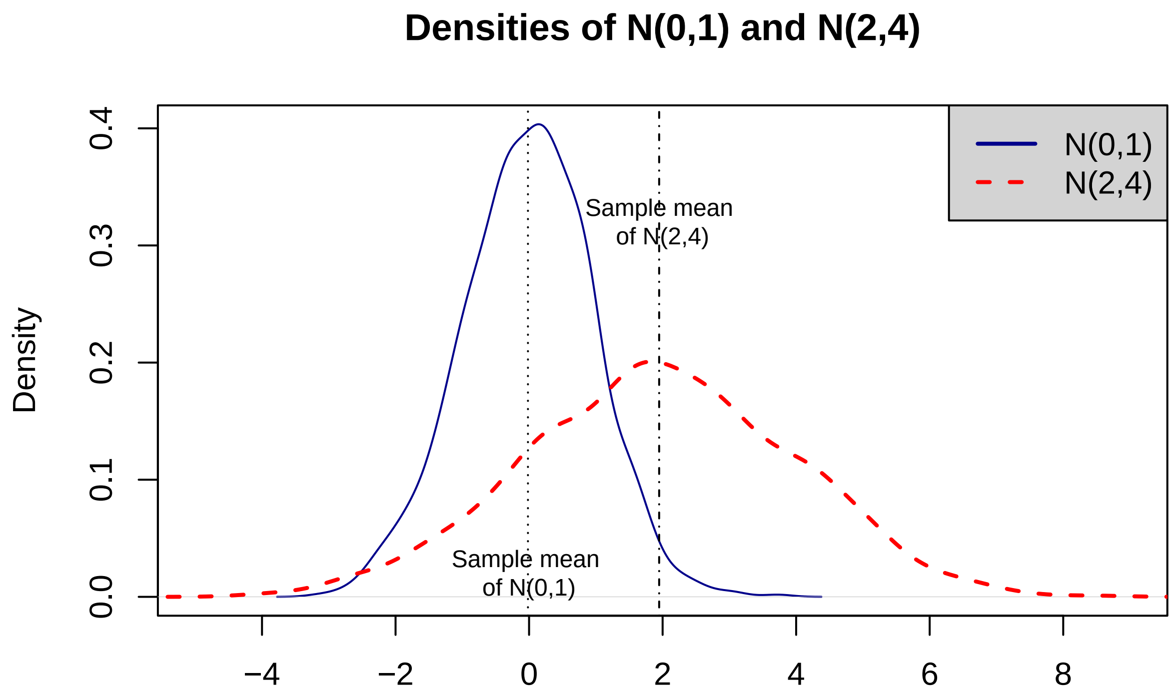
Exercise 7 – customising graphs

Produce two random random vectors, distributed according to $N(0,1)$ and $N(2,2^2)$, respectively. Store these two vectors in `x` and `y` respectively. In a single figure, produce two histograms of `x` and `y` respectively side by side. The command `par(mfrow=c(1,2))` will produce arrange subfigures using 1 row and 2 columns. You can experiment with different configuration of subfigures.

on top of each histogram, use the `R` function `density()` to estimate the density function from a random sample. The `R` command `lines()` add a line plot to an existing graph, which you have produced using `hist()` or `plot()`. Try to get your plot to look like this:



Produce a single plot with the density estimates of both `x` and `y`. Add a vertical line using the `R` function `abline()` to the sample mean of each dataset. Add proper legend and text description to the plot. You plot should look like this:



Solution

```
# not required, but can be used to keep the same sample every time the program is run
set.seed(19621)
# random sample of size 1000 from standard normal
x = rnorm(1000)
# random sample of size 1000 from N(2,2^2)
y = rnorm(1000, 2, 2)

# 2 graphs placed in 1 row
par(mfrow=c(1,2))
hist(x, main="Histogram for N(0,1)", prob=T)
# lines() is used to add a line plot to an existing graph
# lwd=2 specifies a line width 2, default is 1
lines(density(x), lwd=2, col="blue")
# lty=2 specifies a dashed line, default is 1, which is a solid line
hist(y, main="Histogram for N(2,4)", prob=T)
lines(density(y), lwd=2, lty=2, col="red")
```

```
# to cancel previous 1*2 setting for plots
par(mfrow=c(1,1))
plot(density(x), col="darkblue", xlim=c(-5,9), xlab="",
  main="Densities of N(0,1) and N(2,4)")
lines(density(y), col="red", lwd=2, lty=2)
abline(v=mean(x), lty=3)
abline(v=mean(y), lty=4)
legend("topright", c("N(0,1)","N(2,4)"), lwd=2, lty=1:2,
```

```
  col=c("darkblue","red"), bg="lightgrey")
# cex=0.75 specifies the size of the text, default is 1
# \n adds a line break to the text
text(0, 0.05, "Sample mean \nof N(0,1)", pos=1, cex=0.75)
text(2, 0.35, "Sample mean \nof N(2,4)", pos=1, cex=0.75)
```

End of Solution

Exercise 8 – MOM estimates of $U(0, \theta)$

This has to do with Chapter 2.3 of the notes. Set $\theta = 1$, generate a sample of size 10 from a $U(0, \theta)$ distribution and compute the MOM estimate for the unknown θ. Do this experiment 1000 times to see how many times $\hat{\theta}_{mom} < \max\{x_1, \dots, x_{10}\}$. Repeat the experiment with different values of n. Do you see a trend?

Solution

```
m = 1000
n = 10
count = 0
for (i in 1:m) {
  x = runif(n)
  theta_mom = 2 * mean(x)
  if (theta_mom < max(x)) {
    count = count + 1
  }
}
cat("Out of", m, "runs,", count,
  "number of runs produces non-feasible MOM esimates")
```

```
## Out of 1000 runs, 292 number of runs produces non-feasible MOM esimates
```

End of Solution