Assignment 1 - QBS 177

Code

Spencer Bertsch
Jan. 2022

Some helpful commands:
* Insert a new code chunk: *Cmd+Option+I*
* Run a code cell: *Cmd+Shift+Enter*
* Preview the notebook HTML file: *Cmd+Shift+K*

Imports

Hide

```
library(glue)
```

1. Create three vectors x,y,z with integers and each vector has 3 elements. Combine the three vectors to become a 3×3 matrix A where each column represents a vector. (1 pt)

Hide

```
x <- c(1:3)
y <- c(4:6)
z <- c(7:9)

# Here we can use cbind to horizontally concatenate the three individual
# one dimensional vectors.
m <- cbind(x, y, z)
print(m)
```

```
     x y z
[1,] 1 4 7
[2,] 2 5 8
[3,] 3 6 9
```

2. Create a vector with 12 integers. Convert the vector to a 4x3 matrix B using matrix(). Please also obtain the transpose matrix of B named tB. Now tB is a 3×4 matrix. By the rule of matrix multiplication in algebra, can we perform tB * tB in R language? (Is a 3×4 matrix multiplied by a 3x4 allowed?) What result would we get? (1 pt)

Here we can create a 4x3 matrix using the matrix() method. By flipping the byrow parameter between TRUE and FALSE we can either iterate down columns or across rows to generate the matrix.

```
B <- matrix(data = vec, nrow = 4, ncol = 3, byrow = FALSE, dimnames = NULL)
     [,1] [,2] [,3]
[1,]    1    5    9
[2,]    2    6   10
[3,]    3    7   11
[4,]    4    8   12


B <- matrix(data = vec, nrow = 4, ncol = 3, byrow = TRUE, dimnames = NULL)
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
[4,]   10   11   12
```

Hide

```
vec <- c(1:12)
# create a matrix B from the vector cev
B <- matrix(data = vec, nrow = 4, ncol = 3, byrow = TRUE, dimnames = NULL)

# transpose the matrix B
tB <- t(B)
```

By the rule of matrix multiplication in algebra, can we perform tB x tB in R language?

Yes and no. It depends on whether we are performing matrix multiplication or element-wise multiplication. We can remember from linear algebra that we need the inner dimensions to match when we perform matrix multiplication. For example, we could multiply a 3x5 matrix and a 5x4 matrix together because there are the same number of columns in the first matrix as there are rows in the second.

According the the most fundamental rules of linear algebra, no, a 3×4 matrix multiplied by a 3×4 NOT allowed because the inner dimensions do not match. However, this is where we can make a necessary distinction between matrix multiplication and element-wise multiplication of matrices.

In the R language, we can use matrix multiplication with the syntax: `A %*% B` , and we can implement element-wise multiplication by using: `A * B` . If we're performing element-wise multiplication, then yes of course we can multiply two 3x4 matrices together.

Is a 3×4 matrix multiplied by a 3×4 allowed? Yes if we're performing element-wise multiplication, `A * B` in R, no if we're performing matrix multiplication `A %*% B` in R.

What result would we get?
Here is the result of performing tB * tB:

Hide

```
# Here we can see that element wise multiplication produces a resulting matrix as the
# input matrices.
tB * tB
```

```
     [,1] [,2] [,3] [,4]
[1,]    1   16   49  100
[2,]    4   25   64  121
[3,]    9   36   81  144
```

And here we can see from the 'non-conformable arguments' error that we can't perform matrix multiplication on matrices in which the inner dimensions don't match.

Hide

```
tB %*% tB
```

```
Error in tB %*% tB : non-conformable arguments
```

3. Generate a 10 x 10 matrix (square matrix) A1 with proper number of random numbers, then generate another 10 x 12 matrix A2. If we have `A1*M = A2` (Here * represents the conventional multiplication), please solve for M.Hint: use the runif() and solve() functions. (1 pt)

Hide

```
# define the matrices A1 and A2
A1 <- matrix(rpois(100, 10), nrow=10, ncol=10)
A2 <- matrix(rpois(120, 10), nrow=10, ncol=12)

# solve for M
M <- solve(A1, A2)

# check to make sure we got the correct answer
answer <- A1 %*% M
difference = max(abs(answer - A2))
print(paste0("If A1*M = A2 is correct, then this should be zero: ", round(difference, 3
)))
```

```
[1] "If A1*M = A2 is correct, then this should be zero: 0"
```

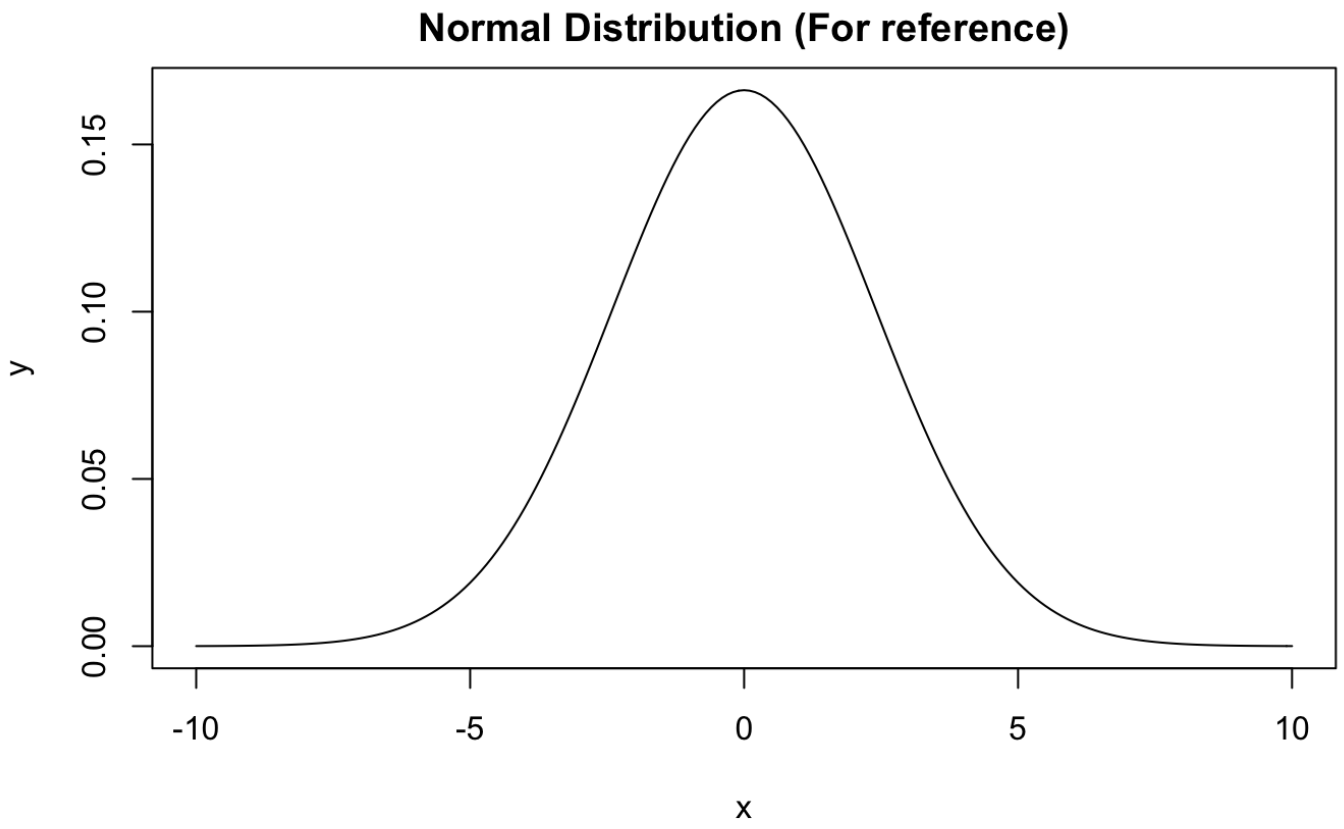
4. Demonstrate Central Limit Theorem using the following pseudo code:

```
n <-5000 # number of random trials
K <- 1 # number of tosses in each batch
L <- 6 # number of sides of of the dice

CLT_results <- c()
#initialize a vector
for(i in 1:n){

  first_batch_toss <- sample(1:L, size=K, replace=T)
  second_batch_toss <- sample(1:L, size=K, replace=T)
  result_i <- sum(first_batch_toss) + sum(second_batch_toss)

  # append to vector
  CLT_results <- c(CLT_results, result_i)

}

#visualize
hist(CLT_results, xlab="")
plot(density(CLT_results))
```

Sequentially increase the value in n, K and L. Determine which parameter can make the density plot reasonably bell shaped. Explain why and relate to Central Limit Theorem. (2 pt)

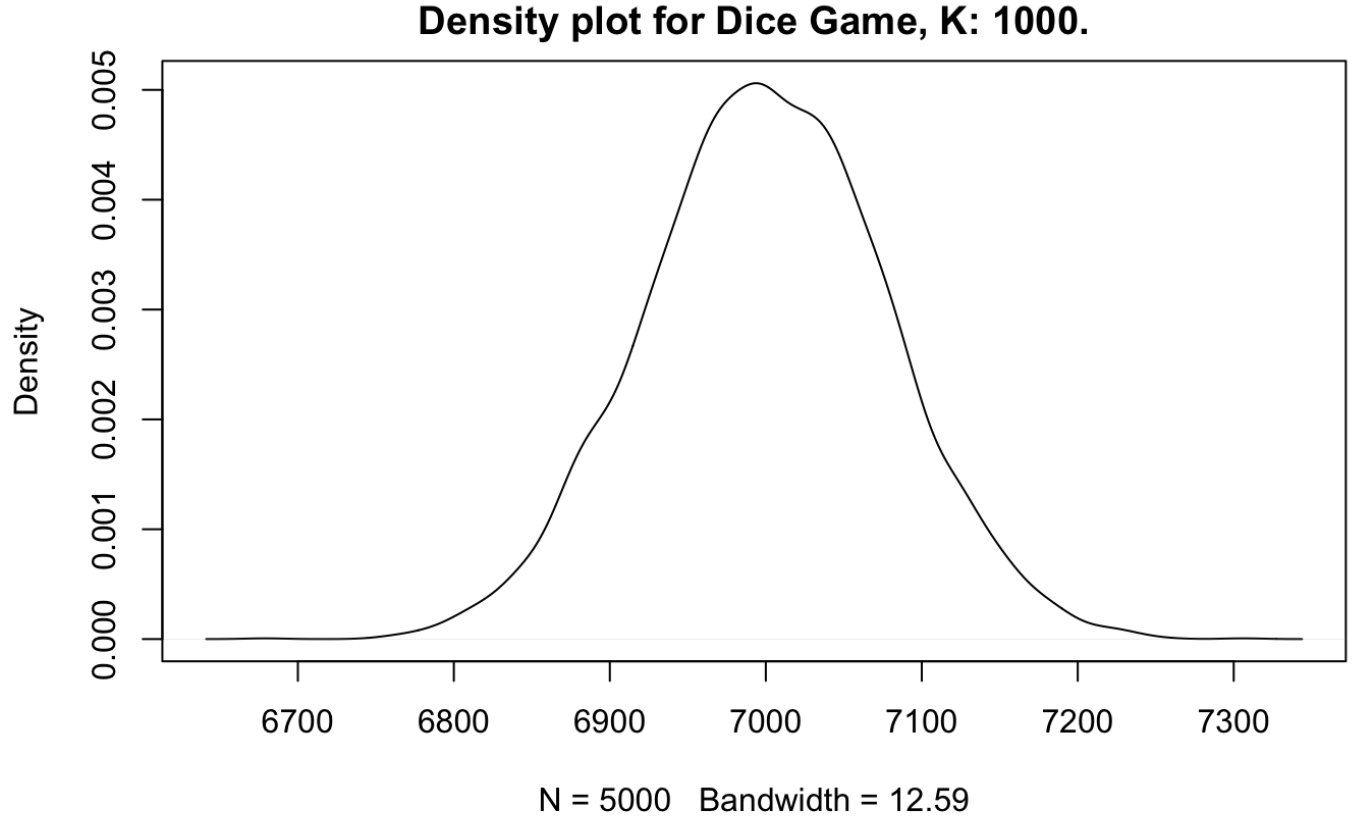
Hide

```
# implement code here
n <-5000 # number of random trials
K <- 1000 # number of tosses in each batch
L <- 6 # number of sides of of the dice

CLT_results <- c()
#initialize a vector
for(i in 1:n){

  first_batch_toss <- sample(1:L, size=K, replace=T)
  second_batch_toss <- sample(1:L, size=K, replace=T)
  result_i <- sum(first_batch_toss) + sum(second_batch_toss)

  # append to vector
  CLT_results <- c(CLT_results, result_i)

}

# also visualize a normal distribution for reference
x <- seq(-10, 10, by = .1)
y <- dnorm(x, mean = 0, sd = 2.4)
plot(x, y, type = "l", lty = 1, main="Normal Distribution (For reference)")
```



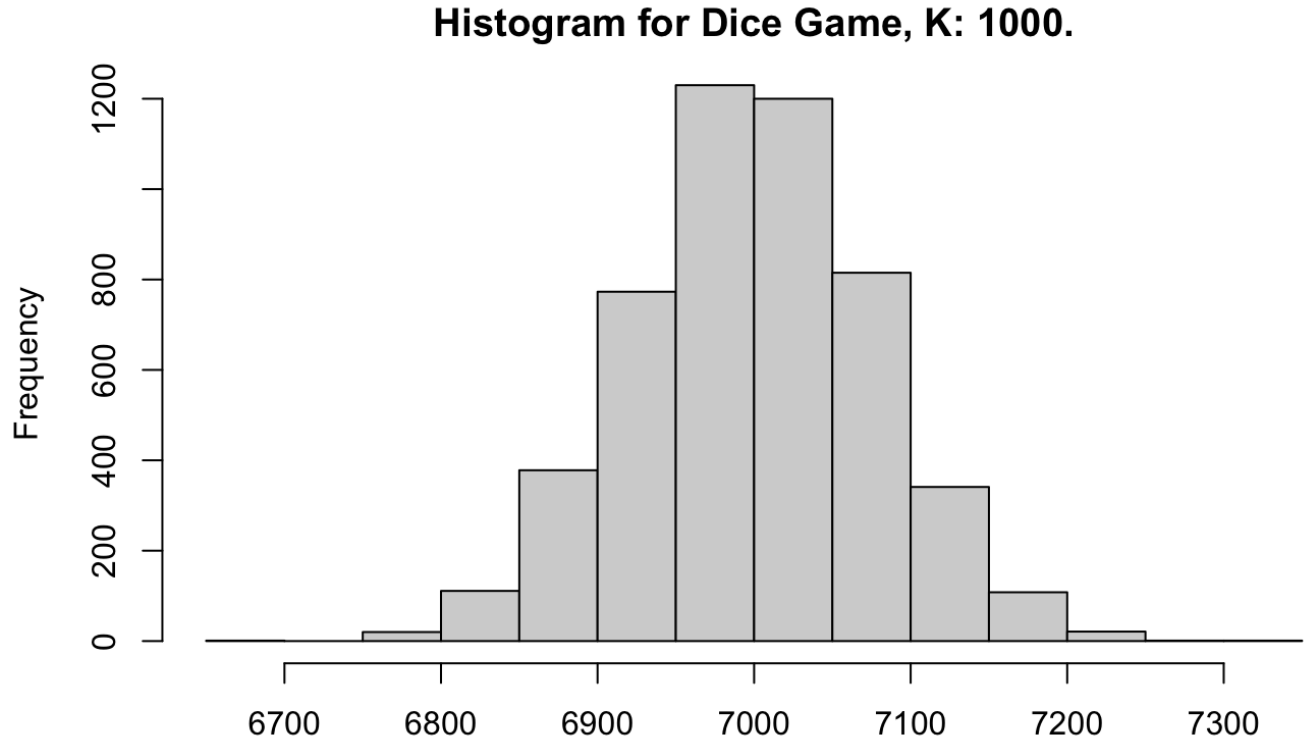
Hide

```
#visualize
plot(density(CLT_results), main=glue('Density plot for Dice Game, K: {K}.'))
```



Hide

```
hist(CLT_results, xlab="", main=glue('Histogram for Dice Game, K: {K}.'))
```



We can see from the above plots that by increasing the **K** parameter (number of tosses in each batch), we can make the resulting density plot reasonably bell shaped. If instead we decrease **K** and take very *few* samples during each trial, then the resulting density plot resembles a triangular distribution. Let's see why a large **K** or `size` parameter in the `sample` R method is needed for the resulting distribution to be reasonably bell shaped.

According to Boston University, "*The central limit theorem states that if you have a population with mean μ and standard deviation σ and take sufficiently large random samples from the population with replacement, then the distribution of the sample means will be approximately normally distributed.*" Source: Central Limit Theorem - BU (https://sphweb.bumc.bu.edu/otlt/mph-modules/bs/bs704_probability/BS704_Probability12.html)

An important takeaway here is that the number of random samples taken needs to be "sufficiently large" in order for the resulting distribution to approach normality. In R, we can sample an array of elements by using the `sample` method. This method takes a parameter called `size` which designates the number of elements that will be sampled from the array (with replacement) during each trial. Therefore we can see the central limit theoram proven visually by varying the `size` parameter which we call **K** in the above code to see the resulting density plot either approach a normal distribution for large **K** values, or approach a triangular distribution for small **K** values.

After reading the BU article mentioned above (https://sphweb.bumc.bu.edu/otlt/mph-modules/bs/bs704_probability/BS704_Probability12.html), I found out that the central limit theorem works even on non-normally distributed data sets as long as the number of samples per trial is high enough. This makes sense because even for a very non-normal distribution such as a Poisson distribution, if we sample that distribution randomly say 100 times, then perform that trial of 100 samples again, we will likely get very similar results. The number of samples in each trial is the important thing here. If we sample only once per trial, then we will just get the original distribution back again. If, however, we increase the number of samples per trial to a high enough number then the results from each trial will begin to resemble a normal distribution.

Sources: https://sphweb.bumc.bu.edu/otlt/mph-modules/bs/bs704_probability/BS704_Probability12.html (https://sphweb.bumc.bu.edu/otlt/mph-modules/bs/bs704_probability/BS704_Probability12.html)