PSC 290 MC Simulations Homework 1

Marwin Carmo

2024-09-26

Intro

Answer all the questions below. This homework is due on Oct 16 before class. If you don't like the idea of a homework due before 9am, then it is due on Oct 15 at midnight. Upload both this Rmd file (completed) and the PDF output. If you are having troubles knitting to pdf, try knitting to html instead. Be sure your troubles aren't due to. Now plot a histogram of the values missing objects in the Rmd file (e.g., all code should run without producing errors).

You may work on this assignment with others, and you may ask for help (e.g., on Slack) while working. The work that you submit, though, must be your own.

Part 1: distribution functions in R

This part has you use the R functions `dnorm`, `qnorm`, `pnorm`, etc. to find likelihoods, quantiles, and probabilities.

Question 1

Let x be a variable that follows a normal distribution with mean = 3 and standard deviation = 17, i.e., $x \sim N(3, 17)$. What proportion of the distribution of x falls below 20?

```
pnorm(20, mean = 3, sd = 17)
```

```
## [1] 0.8413447
```

Question 2

Let y be a variable that follows a normal distribution with mean = 24 and standard deviation = 3, i.e., $x \sim N(24, 3)$. What value of y cuts off the upper 5% of the distribution?

```
qnorm(p = .95, mean = 24, sd = 3)
```

```
## [1] 28.93456
```

Question 3

Let $p \sim N(10, 6)$. How much more likely is it that you would observe a value of 12 compared to a value of 14?

```r
p <- dnorm(c(12, 14), mean = 10, sd = 6)
p[1]/p[2]
```

```
## [1] 1.18136
```

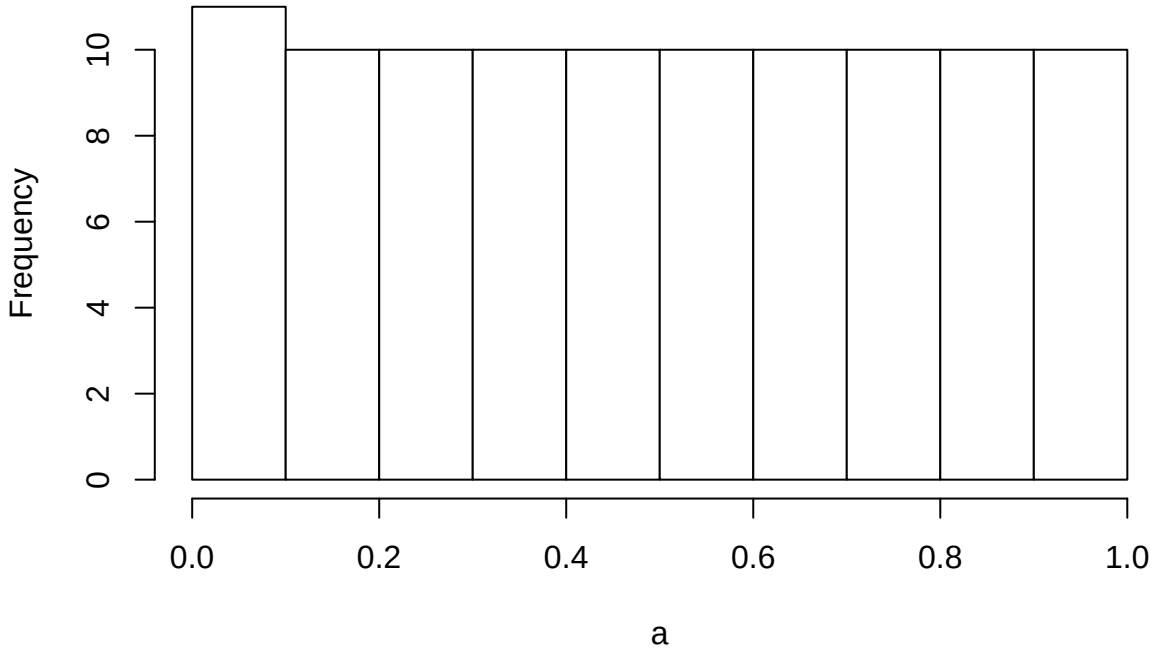
Observing a value of 12 is 1.18 times more likely than observing a value of 14.

Part 2: Understanding PDFs and CDFs

Question 4

Using the `seq()` function, generate a sequence of numbers from 0 to 1, in increments of .01. Name this object `a`. Plot a histogram of the values in the vector you just created.

```r
a <- seq(0, 1, by = .01)
hist(a, col = "white", main=NULL)
```

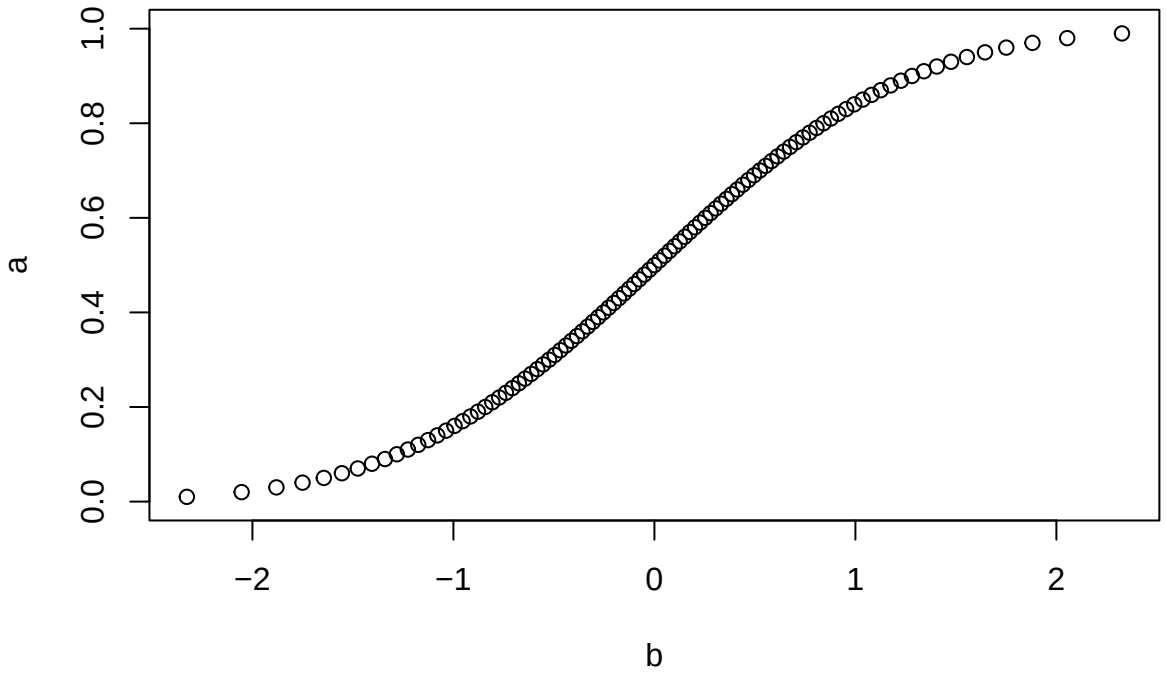


Question 5

Apply the `qnorm` function to the vector of numbers you just created, and call this new object `b`. Make a scatterplot (or line graphs with points) with `b` on the x-axis and `a` on the y-axis. What is this plot called, and what does it show?

```r
b <- qnorm(a)

plot(b, a)
```

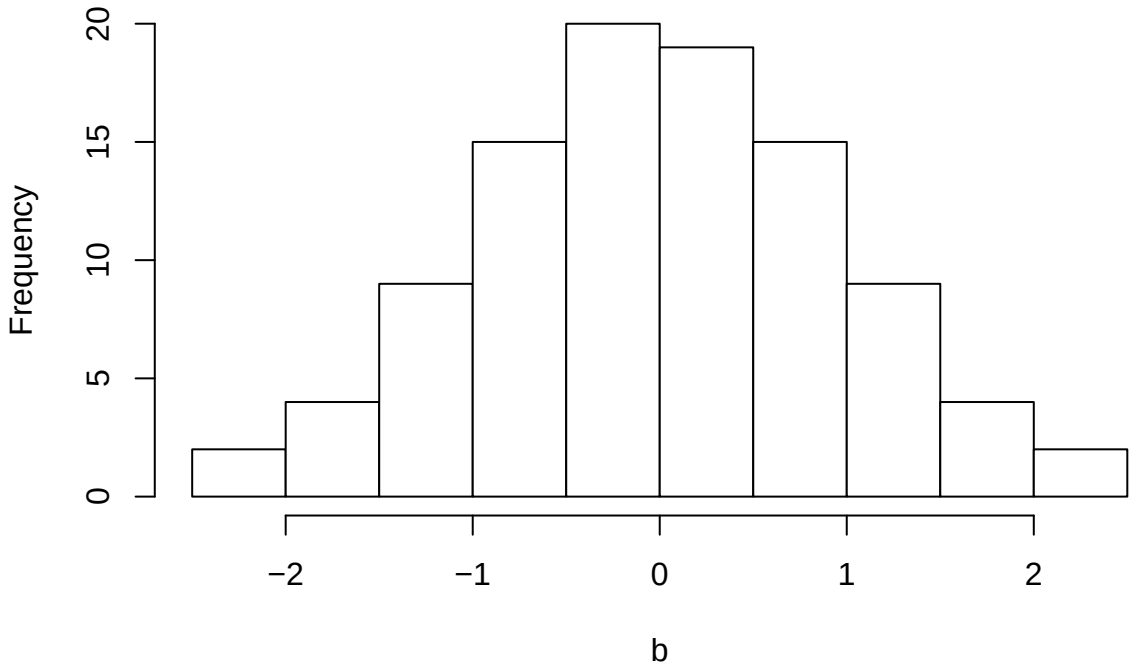


The plot is called Q-Q plot, and shows the the correspondence between the quantiles of a standard normal distribution and the quantiles of a uniform distribution.

Question 6

Now plot a histogram of the values in `b`. What does this distribution look like? Explain what happened.

```
hist(b, col = "white", main = NULL)
```



The plot behaves like a normal distribution centered around 0. ‘a‘ is a vector of quantiles ranging from 0 to 1. To create ‘b‘ we map the quantiles defined in ‘a‘ to a standard normal distribution. The histogram of ‘b‘ reflects the shape of this standard normal distribution.

Part 3: Generating Random Data with different distributional forms

Question 7

The previous question showed us how to transform a sequence of probabilities, uniformly distributed from 0-1, into a vector of quantiles that follow a normal distribution. We can use that same method to generate a *random* variable that follows a normal distribution, by first generating random uniform data.

Generate a random variable of length 10000 (i.e., 10000 random values) from a uniform distribution with range (0, 1). Use `set.seed` to produce replicable results. Give your variable a name.

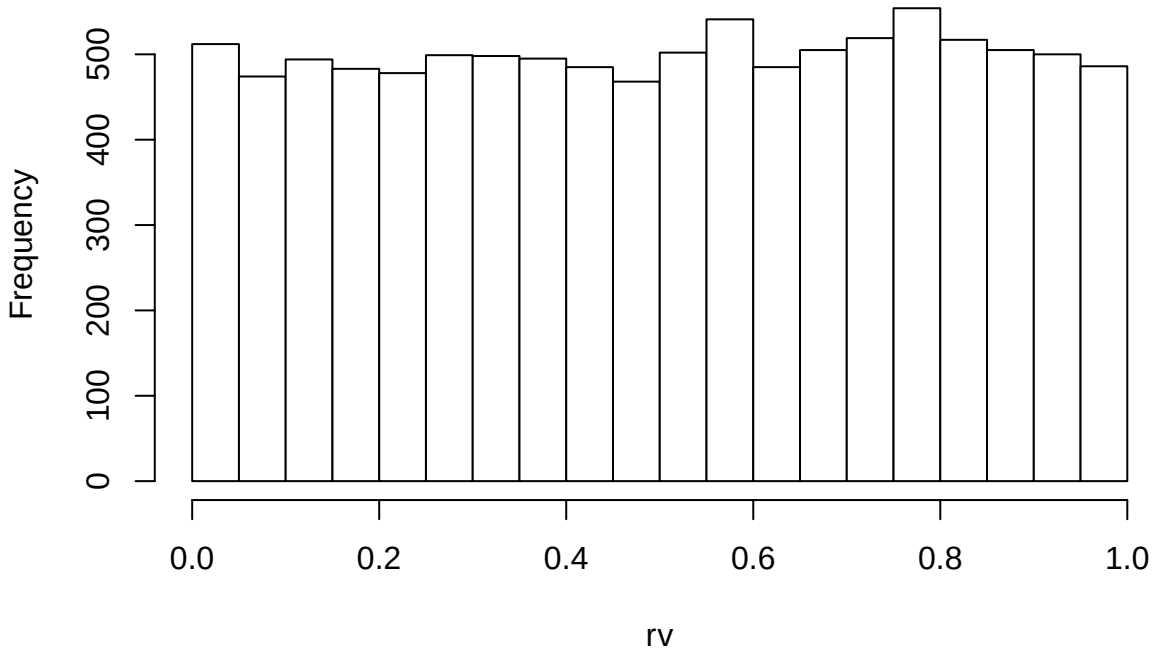
```
set.seed(890)

rv <- runif(10000, 0, 1)
```

Question 8

Use a histogram to plot the values of the variable you just generated. You can use the `hist()` function in base R, or use `ggplot` and make it pretty. Set the width of each histogram bun to be .05 units wide.

```
hist(rv, breaks = seq(0, 1, by = 0.05), col = "white", main = NULL)
```



Question 9

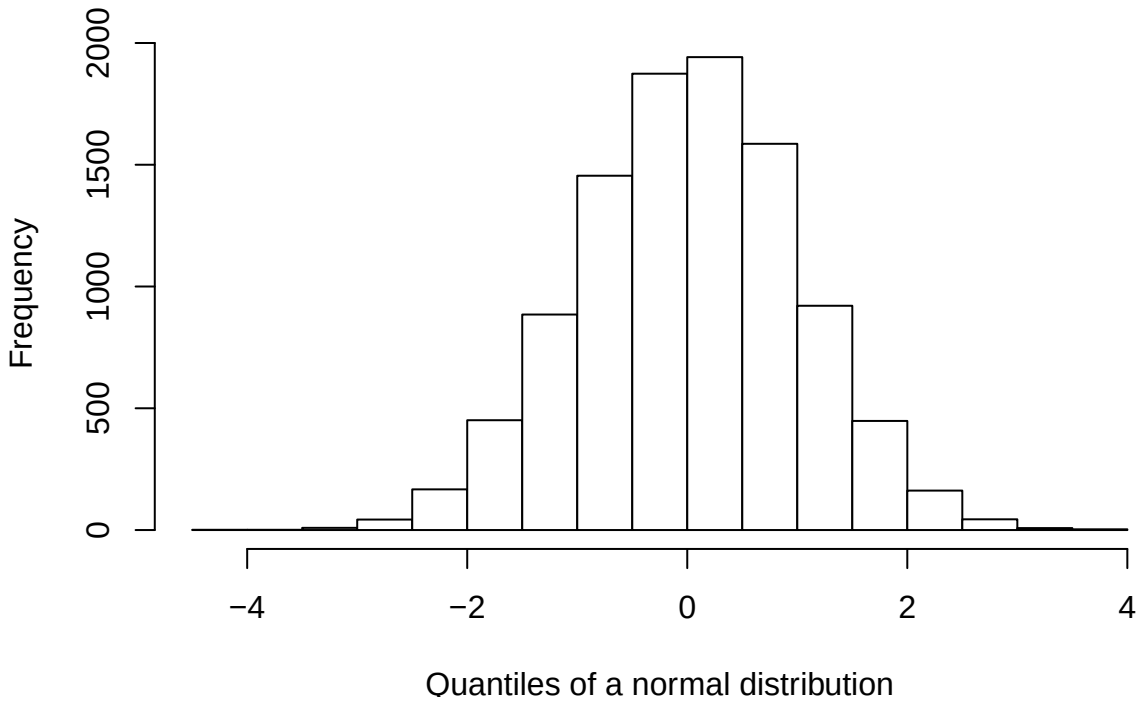
Use the `qnorm` function to find the quantiles of the normal distribution corresponding to each value that you generated. Save these quantiles in a new named object.

```
q <- qnorm(rv)
```

Question 10

Use a histogram to plot the quantiles that you found in the previous question. Again, you can use the `hist()` function in base R, or use `ggplot` and make it pretty. Whatever you use, give your axes interpretable labels.

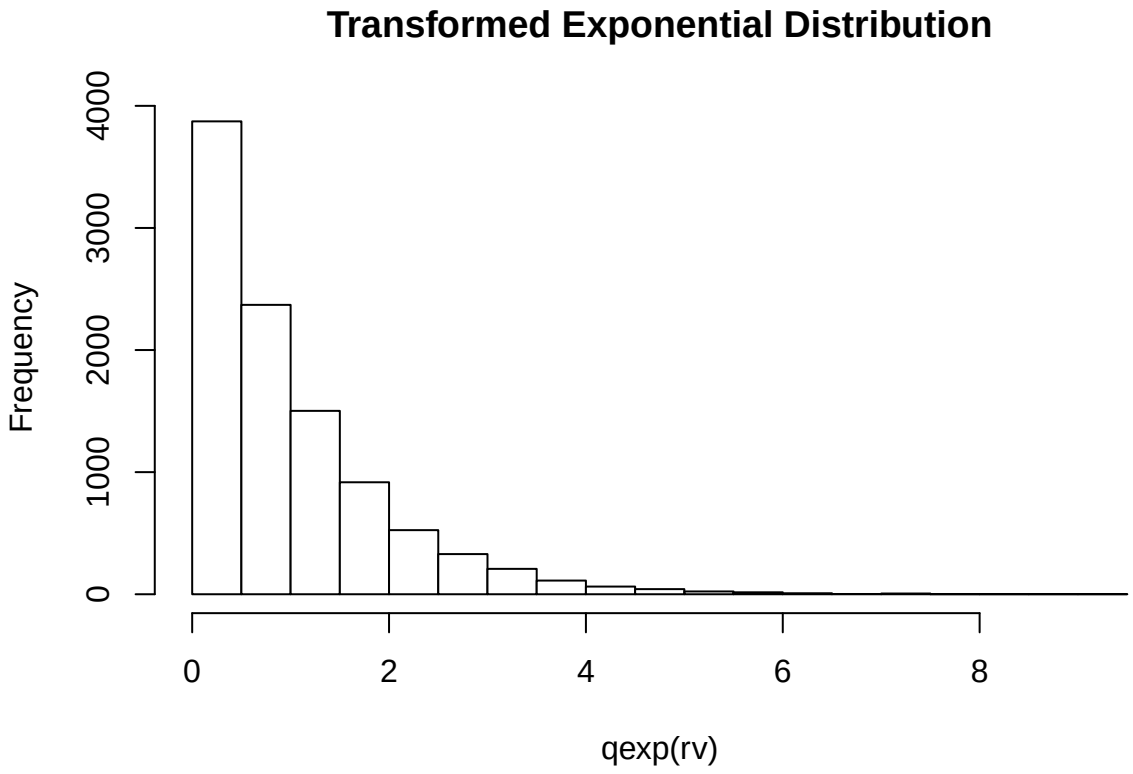
```
hist(q, col = "white", main = NULL, xlab = "Quantiles of a normal distribution")
```



Question 11

That was cool. Take your same original random uniform variable, and give it a different distribution. Plot a histogram of your transformed values to check that your method worked.

```
hist(qexp(rv), main = "Transformed Exponential Distribution", col = "white")
```



Part 4: some mini-simulations to answer questions

Question 12

In class you simulated data to sum up a number of die rolls. Now you're going to do the same thing but write a function to do it. Your function should have 2 arguments: N is the size of the sample (i.e., the number of dice that are rolled), and w is the number of sides on the die (i.e., the number of possible outcomes the die can take). Your function should return the sum or average of the N rolls. Run your function to demonstrate that it works.

```
die_roll <- function(N = 1, w = 6) {

  sides <- seq(1, w, by = 1)
  res <- sample(sides, N, replace = TRUE)
  return(mean(res)) # returning the average
}

die_roll(N = 100, w = 6)
```

```
## [1] 3.62
```

Question 13

How many rolls of a 6-sided die do you need to add together (or average) before you get a distribution that behaves more or less like a normal distribution? You could use many different criteria to define "behaves more or less like a normal distribution". Pick one that can be quantified in some way ("I eyeballed the histogram and it looks normal to me" is not a quantifiable criterion) and justify your choice. Then run

the simulation and report your result. Use the function that you write in the previous question in your simulation.

To quantify when our empirical distribution of n die rolls starts to behave like a normal distribution, we can calculate the Mean Square Error (MSE) of the sample quantiles compared to the theoretical quantiles:

```
set.seed(890)
reps <-  10000
die_values <- 1:6

# To compute the theoretical quantiles we need to find the
# expected value of a 6-sided die roll and its standard deviation
expected_value <- sum(die_values * rep(1/6, 6))
std_dev <- sqrt(sum((1/6*((die_values-mean(die_values))^2))))
# We can define a sequence of rolls from 2 to 100 as it is very likely
# that we'll reach a good approximation with way less rolls than 100.
rolls_list <- 2:100

# In this function we first simulate 10000 times `n` die rolls,
# then, we compute 10000 theoretical quantiles of a distribution
# with a mean equal to the expected value of a 6-sided die roll
# and the standard deviation as the standard error of the mean of `n`
# die rolls.
mse_qq <- sapply(rolls_list, function(n) {
  sim_data <- replicate(reps, die_roll(N = n))
  theoretical_quantiles <- qnorm(seq(1/(reps+1), 1 - 1/(reps+1), length.out = reps),
                                 mean = expected_value, sd= std_dev/sqrt(n))
  sample_quantiles <- quantile(sim_data, probs = seq(1/(reps+1), 1 - 1/(reps+1), length.out = reps))
  return(mean((sample_quantiles - theoretical_quantiles)^2))
})

# Now we can print the MSE by the number of rolls to have an idea of
# when it starts to approach zero.

mse_df <- data.frame(rolls_list,mse_qq)

knitr::kable(mse_df,
    col.names = c("Die rolls", "MSE")
  )
```

Die rolls	MSE
2	0.0361041
3	0.0121763
4	0.0066607
5	0.0040271
6	0.0026464
7	0.0021479
8	0.0014592
9	0.0014451
10	0.0009323
11	0.0007517
12	0.0007201
13	0.0005878

Die rolls	MSE
14	0.0005298
15	0.0004261
16	0.0004127
17	0.0003770
18	0.0002941
19	0.0002494
20	0.0002333
21	0.0002614
22	0.0002000
23	0.0002591
24	0.0001891
25	0.0001615
26	0.0001374
27	0.0001760
28	0.0001546
29	0.0001227
30	0.0001306
31	0.0001319
32	0.0001027
33	0.0001319
34	0.0000975
35	0.0000980
36	0.0001163
37	0.0000744
38	0.0000802
39	0.0000760
40	0.0000644
41	0.0000853
42	0.0000779
43	0.0000960
44	0.0000780
45	0.0000723
46	0.0000547
47	0.0000711
48	0.0000486
49	0.0000503
50	0.0000445
51	0.0000501
52	0.0000606
53	0.0000450
54	0.0000426
55	0.0000467
56	0.0000340
57	0.0000330
58	0.0000417
59	0.0000340
60	0.0000376
61	0.0000418
62	0.0000478
63	0.0000602
64	0.0000329
65	0.0000277

Die rolls	MSE
66	0.0000358
67	0.0000274
68	0.0000247
69	0.0000281
70	0.0000382
71	0.0000308
72	0.0000332
73	0.0000231
74	0.0000204
75	0.0000219
76	0.0000245
77	0.0000277
78	0.0000312
79	0.0000214
80	0.0000238
81	0.0000201
82	0.0000194
83	0.0000210
84	0.0000260
85	0.0000223
86	0.0000188
87	0.0000191
88	0.0000158
89	0.0000259
90	0.0000179
91	0.0000261
92	0.0000252
93	0.0000170
94	0.0000379
95	0.0000239
96	0.0000165
97	0.0000186
98	0.0000152
99	0.0000217
100	0.0000177

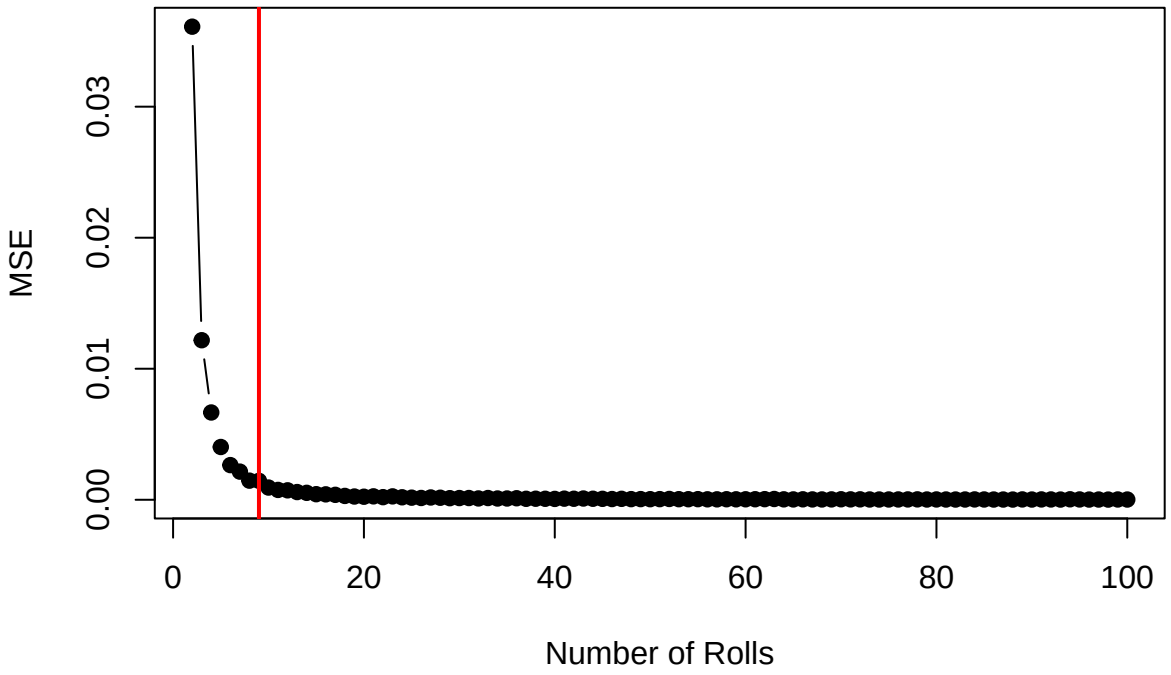
```r
# Given the MSE values observed in the table above, we can define an arbitrary
# threshold for stabilization. The difference in MSE values looks less
# accentuated for MSE < 0.001. Therefore,

threshold <- 0.001

# Now we can find the first point where all subsequent MSE values are below the threshold

stabilization_point <- which(cummin(mse_df$mse_qq) < threshold)[1]

# Plot. Red line marks the stabilization point
plot(rolls_list, mse_qq, type = "b", pch = 19,
     xlab = "Number of Rolls", ylab = "MSE")
abline(v = stabilization_point, col = "red", lwd = 2)
```



For this somewhat arbitrary MSE value, we could say that the distribution of the means of 6-sided die rolls approximates to a normal distribution when $n = 9$.

Question 14

At what sample size is it ok to use the standard normal (z) distribution rather than the t-distribution for hypothesis testing? Suppose that by "ok" we mean that the type-I error rate (rate of false positives) is within half a percent of the nominal .05 rate (assuming an alpha level of .05). Answer this question in two ways: First, do it analytically, using the `qnorm` and `pt` functions to find the proportion of various t-distributions beyond the critical values of the z-distirbution. Assume that the degrees of freedom of the t-distribution are $N - 1$. Second, once you have found an answer, verify it by simulating a large number of samples from the t-distribution at the sample size you identified as being "ok": what proportion of your simulated values fall in the tails past the critical z-values? Remember to set a random number seed to produce a replicable result. How much `monte carlo` error do you have (i.e., how accurate is your simulation result relative to your analytical result)?

```
## Analytical solution

## critical z values
lower_critical <- qnorm(0.05/2)
upper_critical <- qnorm(1 - 0.05/2)

# starting sample size
n <- 2

while (TRUE) {

  df <- n - 1
  # corresponding t values
  t_lower <- pt(lower_critical, df)
  t_upper <- 1 - pt(upper_critical, df)
```

```
  # error rate
  t_error <- t_lower + t_upper

  if (abs(t_error - 0.05) <= 0.005) {
      return(n)
    }
  n <- n + 1
}

#
print(n)
```

```
## [1] 57
```

```
## Simulation
set.seed(890)
reps <- 10000
tvals <- rt(reps, n - 1)

prop_tails <- mean(abs(tvals) > upper_critical)
prop_tails
```

```
## [1] 0.0596
```

```
mc_error <- abs(prop_tails - 0.05)
mc_error
```

```
## [1] 0.0096
```

The required sample size is 57 according to the analytical solution. By simulating 10^4 random t-values from a t-distribution with 56 degrees of freedom, we observe that 5.96% of the values are larger than the critical z-value of 1.959964. The monte carlo error was found to be 0.0096.