Assignment 3

SOSC 13200-2

Vincent Zhang

1/31/22 @ 11:59pm

```r
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.1.1     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

1.

Consider the random process of flipping a fair coin three times.

1.1 (5pts)

Create a new R object: specifically, a vector whose elements describe the sample space in terms of heads and tails. E.g., three heads could be described as "HHH". Name your vector `omega`.

```r
omega <- c("HHH", "HHT", "HTH", "HTT", "THH", "THT", "TTH", "TTT")
```

1.2 (6pts)

We are interested in a random variable X, which is the number of heads that we get from the random process above. Generate a new data.frame object with two columns. One column, `X`, should contain the total number of heads we could obtain via our random process. The second column, `probs`, should contain the probability that these different events occur.

Be sure to print your data.frame so that it shows in your final PDF.

HINT: The coin is fair, so each of the outcomes in the sample space in 1.1 occurs with equal probability. Note how many heads we get in each outcome. Then note the proportion of times we get no heads, one head, etc. These proportions are equal to the probability.

```
# your code here
n <- length(omega)
X <- c(0, 1, 2, 3)
probs <- c(1, 3, 3, 1) / n
X_tbl <- data.frame(X, probs)
X_tbl
```

```
##   X probs
## 1 0 0.125
## 2 1 0.375
## 3 2 0.375
## 4 3 0.125
```

1.3 (6pts)

Calculate the mean of X. You may not use the `mean()` function. And, your calculation must draw on elements in both columns of your data.frame above.

```
# your code here
mean <- 0
for (i in seq_along(X_tbl$X)) {
  mean <- mean + (X_tbl$X[i] * X_tbl$probs[i])
}

mean
```

```
## [1] 1.5
```

1.4 (8pts)

Simulate our random process above. Your output should be a single realization of the random variable `X` (i.e., a number that denotes the number of heads in our coin tosses.)

Note: Below, I have set a random seed, so that any time you knit your assignment, you will get the same number. For analyses that involve sampling or random processes, it is important to set a random seed so that you can get reproducible results. Feel free to change the seed number to anything you want. In general, you should set your seed just ONCE per RMD.

```
# I have set a random seed here
set.seed(60637)
# your code here
sample(X_tbl$X, prob = X_tbl$probs, size = 1)
```

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

1.5 (12pts)

Now run your random process so that you sample from it 10,000 times. [PLEASE: DO NOT PRINT ALL 10,000 OBSERVATIONS IN YOUR PDF. Just assign them to a new R object.] What is the average number of heads across all 10K observations? NOTE: This is called the “sample mean” for a given sample.

```r
# your code here
observations <- sample(X_tbl$X, prob = X_tbl$probs, replace = TRUE, size = 10000)
mean(observations)
```

```
## [1] 1.5062
```

1.6 (8pts)

Write your own simple function called `mymean`. Your function should calculate the sample mean from a vector. Apply your new function to your sample of size 10K from 1.5.

*NOTE: Don't use `mean()` inside your function, and don't call the specific object you created in the last question inside your function. Your `mymean()` function should work when applied to any vector.

*ALSO NOTE: R allows us to generate our own functions and apply them to data. The function above is relatively simple, but you may still need to do some research on writing functions in R.

```r
# your code here
mymean <- function(vec) {
  n <- length(vec)
  sum <- 0
  for (i in seq_along(vec)) {
    sum <- sum + vec[i]
  }

  return(sum / n)
}

mymean(observations)
```

```
## [1] 1.5062
```

1.7 (15pts)

Re-run the code from 1.5 to get another length 10k sample from the same random process. [DON'T PRINT THIS WHOLE OBJECT.] Apply your `my_mean()` function to it.

```r
# your code here
observations <- sample(X_tbl$X, prob = X_tbl$probs, replace = TRUE, size = 10000)
mymean(observations)
```

```
## [1] 1.4985
```

2.

Using the same random process of flipping three fair coins, create a new dummy variable `Y`. The variable should equal 1 if we get three heads, and 0 otherwise.

2.1 (6pts)

Write a data.frame object with two columns. One column, `Y`, describes all of the possible values of Y we could get. The second column, `probs`, describes the probability that each of these events occurs.

Be sure to print your data.frame so that it shows in your PDF.

```
# your code here
Y <- c(0, 1)
probs <- c(7/8, 1/8)
Y_tbl <- data.frame(Y, probs)
Y_tbl
```

```
##   Y probs
## 1 0 0.875
## 2 1 0.125
```

2.2 (10pts)

Write a new data.frame object that has three columns. Two columns, `X` and `Y`, jointly describe the values that `X` and `Y` can take on together. The third column, `probs`, describes the probability that each of these pairs of events occurs jointly.

Be sure to print your data.frame so that it shows in your PDF.

```
# your code here
XY_tbl <- expand.grid(X,Y)
colnames(XY_tbl) <- c("X", "Y")
joint_probs <- c(1/8, 3/8, 3/8, 0, 0, 0, 0, 1/8)
XY_tbl$joint_probs <- joint_probs
XY_tbl <- XY_tbl %>% filter(joint_probs > 0)
XY_tbl
```

```
##   X Y joint_probs
## 1 0 0       0.125
## 2 1 0       0.375
## 3 2 0       0.375
## 4 3 1       0.125
```

2.3 (12pts)

Report the conditional mean of X given that Y equals 0.

Recall that conditional probability can be written as:

$$\text{P}[A|B] = \frac{\text{P}[AB]}{\text{P}[B]}$$

```
# your code here
Y0_probs <- XY_tbl %>% filter(Y == 0)
conditional_probs <- Y0_probs$joint_probs / (7/8) # 7/8 is the prob of y==0
```

```r
# conditional_probs
mean_prob <- 0
for (i in seq_along(Y0_probs$X)) {
  mean_prob <- mean_prob + (conditional_probs[i] * Y0_probs$X[i])
}

mean_prob
```

```
## [1] 1.285714
```

2.4 (12pts)

Are the events that $X = 3$ and that $Y = 1$ independent?

We basically want to show that

$$\mathrm{P}[XY] = \mathrm{P}[X]P[Y]$$

for the given X and Y.

```r
# your code here

PX <- (X_tbl %>% filter(X == 3))$probs
PY <- (Y_tbl %>% filter(Y == 1))$probs

PXY <- (XY_tbl %>% filter(X == 3 && Y == 1))$probs

is_independent <- (PX * PY == PXY)
is_independent
```

```
## logical(0)
```

```r
#They are not independent
```

they are not logically independent because for X = 3 and Y = 1, the probabily of PXPY != PXY