Lab2_R.Rmd

Eddie Coda

2024-06-02

STAT550 Applied Probability

Lab 2: R and RStudio basics

Total: 30 points

In this assignment you will need to solve a series of programming questions about R using RStudio. **You are encouraged to look up the official documentation**.

- RMarkdown at its core is a combination of `R` and Markdown used to generate reproducible reports for data analyses.
- RMarkdown Cheatsheet
- Community Forum
- **Search for anything R related**

Your goal is to run all the cells below one by one from top to bottom. Before you run some code chunks, you need to complete the missing code (indicated by the **NA**) in them.

For each **task** chunk that requires your completion, you should run it to check if your code is correct. The cell output should be similar to the "expected output" that follows.

Chunk Options

- `include = FALSE` prevents code and results from appearing in the finished file. R Markdown still runs the code in the chunk, and the results can be used by other chunks.
- `echo = FALSE` prevents code, but not the results from appearing in the finished file. This is a useful way to embed figures.
- `message = FALSE` prevents messages that are generated by code from appearing in the finished file.
- `warning = FALSE` prevents warnings that are generated by code from appearing in the finished.

The ? code pulls up documentation of a function. This will spawn a browser window when knitting, or potentially crash during knitting.

Importing Packages

When using RMarkdown, any time you *knit* your document to its final form, say `.html`, a number of programs run in the background. Your current `R` environment seen in RStudio will be reset. Any objects you created while working interactively inside RStudio will be ignored. Essentially a new `R` session will be spawned in the background and the code in your document is run there from start to finish. For this reason, things such as importing data must be explicitly coded into your document.

```
# Install and Load packages in a single line for each
# Install and Load packages in a single line for each
if (!require("tidyverse")) install.packages("tidyverse");
library("tidyverse")

## Loading required package: tidyverse

## ── Attaching core tidyverse packages ──────────────────────── tidyverse
2.0.0 ──
## ✓ dplyr     1.1.4     ✓ readr     2.1.5
## ✓ forcats   1.0.0     ✓ stringr   1.5.1
## ✓ ggplot2   3.5.1     ✓ tibble    3.2.1
## ✓ lubridate 1.9.3     ✓ tidyr     1.3.1
## ✓ purrr     1.0.2
## ── Conflicts ──────────────────────────────────────────
tidyverse_conflicts() ──
## ✕ dplyr::filter() masks stats::filter()
## ✕ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all
conflicts to become errors

if (!require("ggplot2")) install.packages("ggplot2"); library("ggplot2")
if (!require("pander")) install.packages("pander"); library("pander")

## Loading required package: pander

if (!require("rmarkdown")) install.packages("rmarkdown");
library("rmarkdown")

## Loading required package: rmarkdown

# tinytex is only required if you need to deal with Tex/LaTeX files
if (!require("tinytex")) install.packages("tinytex"); library("tinytex")

## Loading required package: tinytex

if(!require("magrittr")) install.packages("magrittr");
```

```
## Loading required package: magrittr
##
## Attaching package: 'magrittr'
##
## The following object is masked from 'package:purrr':
##
##     set_names
##
## The following object is masked from 'package:tidyr':
##
##     extract

library("magrittr")
library(dplyr)
```

Coin Flip Example

Starting on page 23 in Probability with applications and R by Robert P. Dobrow, we are introduced to flipping a coin:

`CoinFlip.R` simulates 1000 times, the probability of getting three heads in three coin tosses.

We can also write it like this:

```
num_flips <- 1000

coin <- c('heads', 'tails')

# Let's flip the biased coin
flips <- sample(coin, size = num_flips, replace = TRUE)

freqs <- table(flips)

freqs

## flips
## heads tails
##   534   466
```

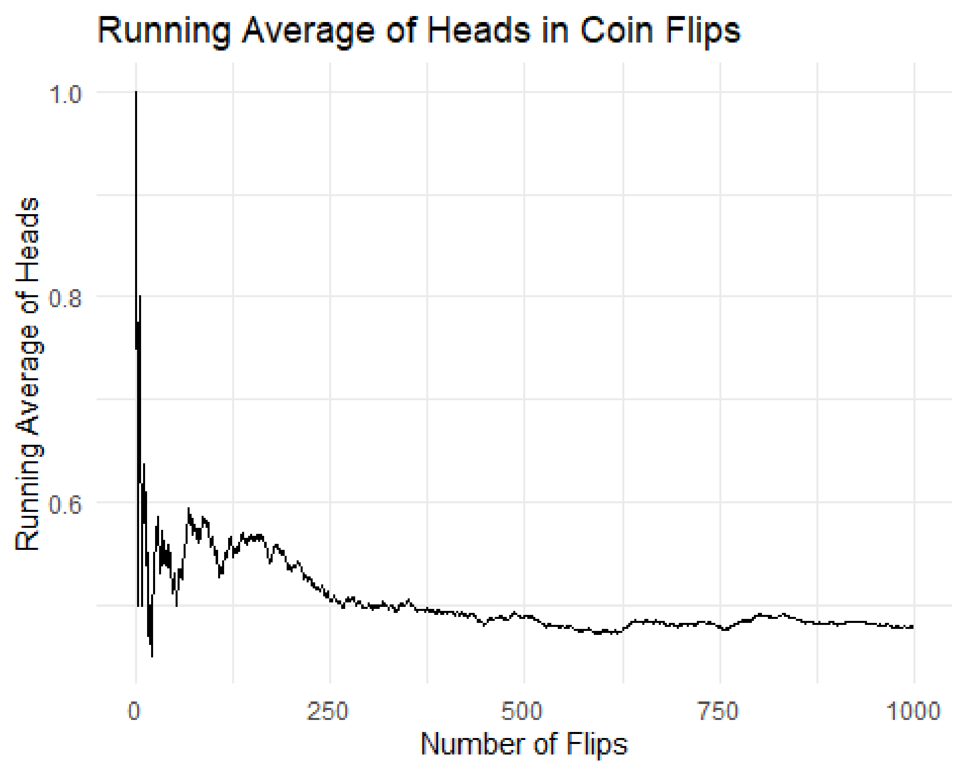
Plot the flips & Display the table - Setup

1. We'll create a logical vector indicating whether each flip is a head. Then, we'll use cumsum to get the cumulative sum.
2. We'll divide the cumulative sum of heads by the sequence of tosses.
3. We'll plot the running average of heads, which is just the running proportion of heads.
4. Create a dataframe and construct the table:

    - Number of coin tosses
    - Number of heads
    - Proportion of heads
5. Transpose the data frame.
6. Use pander to display the table.

I'll add some comments so you can pretend you understand what's happening.

```r
num_flips <- 1000 # number of flips

coin <- c('heads', 'tails') # flips simulation

flips <- sample(coin, size = num_flips, replace = TRUE)

is_head <- as.numeric(flips == 'heads') # 1 for heads, 0 for tails

cum_heads <- cumsum(is_head) # Cumulative sum of heads

run_prop <- cum_heads / (1:num_flips) # Running proportion of heads

# Plotting the running average of heads
ggplot(data = data.frame(x = 1:num_flips, y = run_prop), aes(x = x, y = y)) +
  geom_line() +
  labs(title = 'Running Average of Heads in Coin Flips',
       x = 'Number of Flips',
       y = 'Running Average of Heads') +
  theme_minimal()
```



```r
# Specific intervals
intervals <- c(10, 50, 100)

# Extracting data for those intervals
heads_at_intervals <- cum_heads[intervals]
prop_at_intervals <- run_prop[intervals]

# Creating a transposed table
table_data <- data.frame(
  "Number of coin tosses" = intervals,
  "Number of heads" = heads_at_intervals,
  "Proportion of heads" = prop_at_intervals
)
# Rename columns to avoid dots
colnames(table_data) <- c("Number of Coin Tosses", "Number of Heads",
"Proportion of Heads")

# because you suddenly love rows
transposed_table <- t(table_data)

# Display the table with centered text
panderOptions('table.alignment.default', 'center')
pander(transposed_table)
```

Number of Coin Tosses	10	50	100
Number of Heads	6	26	56

Proportion of Heads	0.6	0.52	0.56

Here’s how you mess with probability:

1. Adjust the **`prob`** parameter in the **`sample`** function. This parameter defines the probability of selecting each element in your vector.

2. Set **`prob = c(0.7, 0.3)`** for heads and tails respectively. That means there’s a 70% chance of getting heads and a 30% chance for tails.

```
num_flips <- 1000

coin.biased <- c('heads', 'tails')

# Let's flip the biased coin
flips.biased <- sample(coin.biased, size = num_flips, replace = TRUE, prob =
c(0.7, 0.3))

freqs.biased <- table(flips.biased)

freqs.biased

## flips.biased
## heads tails
##   705   295
```

Task 1: Plot the biased coin (10pts)

We want to calculate the running average of **tails** this time.

1. logical vector indicating whether each flip is a tail

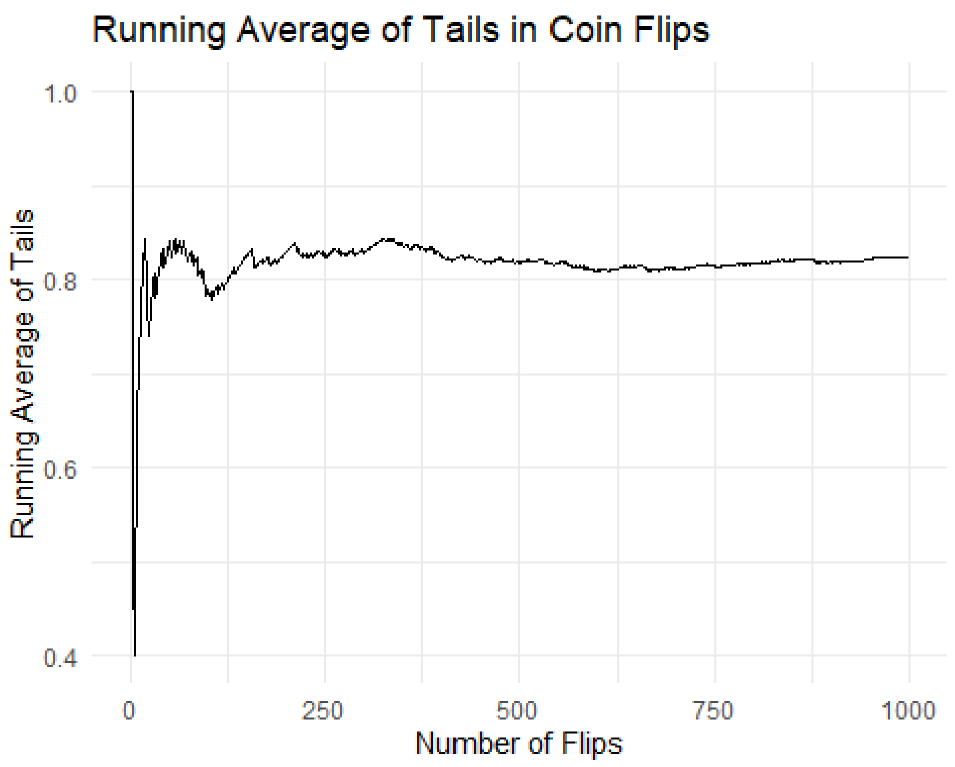
2. use `cumsum()` to get the cumulative sum

3. divide the cumulative sum of heads by the sequence of tosses

4. Change intervals to show 25, 50, 75, 100 flips

```
num_flips <- 1000 # number of flips

coin.biased <- c('heads', 'tails') # flips simulation

flips.biased <- sample(coin.biased, size = num_flips, replace = TRUE, prob =
c(0.2, 0.8))

################START CODE HERE##############
is_tails.biased <- as.numeric(flips.biased == 'tails') # 1 for tails, 0 for
heads
```

```r
tails.biased <- cumsum(is_tails.biased) # Cumulative sum of tails

run_prop.biased <- tails.biased / (1:num_flips) # Running proportion of tails

intervals <- c(25, 50, 75, 100) # table flip values
################END CODE HERE################

# Plotting the running average of tails
ggplot(data = data.frame(x = 1:num_flips, y = run_prop.biased), aes(x = x, y
= y)) +
  geom_line() +
  labs(title = "Running Average of Tails in Coin Flips",
       x = "Number of Flips",
       y = "Running Average of Tails") +
  theme_minimal()
```



```r
tails_at_intervals.biased <- tails.biased[intervals]
prop_at_intervals.biased <- run_prop.biased[intervals]

table_data_biased <- data.frame(
  "Number of Coin Tosses" = intervals,
  "Number of Tails" = tails_at_intervals.biased,
  "Proportion of Tails" = prop_at_intervals.biased
)

# Transposing it, because why present data in a readable way
```

```r
transposed_table_biased <- t(table_data_biased)

# Pretending we care about presentation
panderOptions('table.alignment.default', 'center')
pander(transposed_table_biased)
```

Number.of.Coin.Tosses	25	50	75	100
Number.of.Tails	19	42	62	79
Proportion.of.Tails	0.76	0.84	0.8267	0.79

Birthday Problem- Setup

Assume this is a randomly selected group of 15 people, what is the chance that at least two people have the same birthday?

```r
nPeople <- 15 # group size
nTrials <- 10000 # trials

simlist <- vector(length = nTrials)
for (i in 1:nTrials) {
  # now, let's roll that 365-sided die
  trial <- sample(1:365, nPeople, replace = TRUE)
  success <- if (2 %in% table(trial)) 1 else 0
  simlist[i] <- success
}

print(paste(nPeople, 'people gives us a', mean(simlist),'probability of at
least two people having the same birthday.'))

## [1] "15 people gives us a 0.2503 probability of at least two people having
the same birthday."
```

What do you think the chances are that you share a birthday with someone in your class?

Assume this is a randomly selected group of 50 people, what is the chance that at least two people have the same birthday?

```r
# let's make a function this time
same_birthday <- function(n){
  bdays <- sample(1:365, n, replace = TRUE)
  any(duplicated(bdays)) # checking for a match
}

B <- 10000
results <- replicate(B, same_birthday(50))
print(paste("50 people... Probability at least two have the same birthday:
",mean(results)))
```

```
## [1] "50 people... Probability at least two have the same birthday:
0.9686"
```

Here we use `replicate()` call `same_birthday` 10000 times with a group size of 50

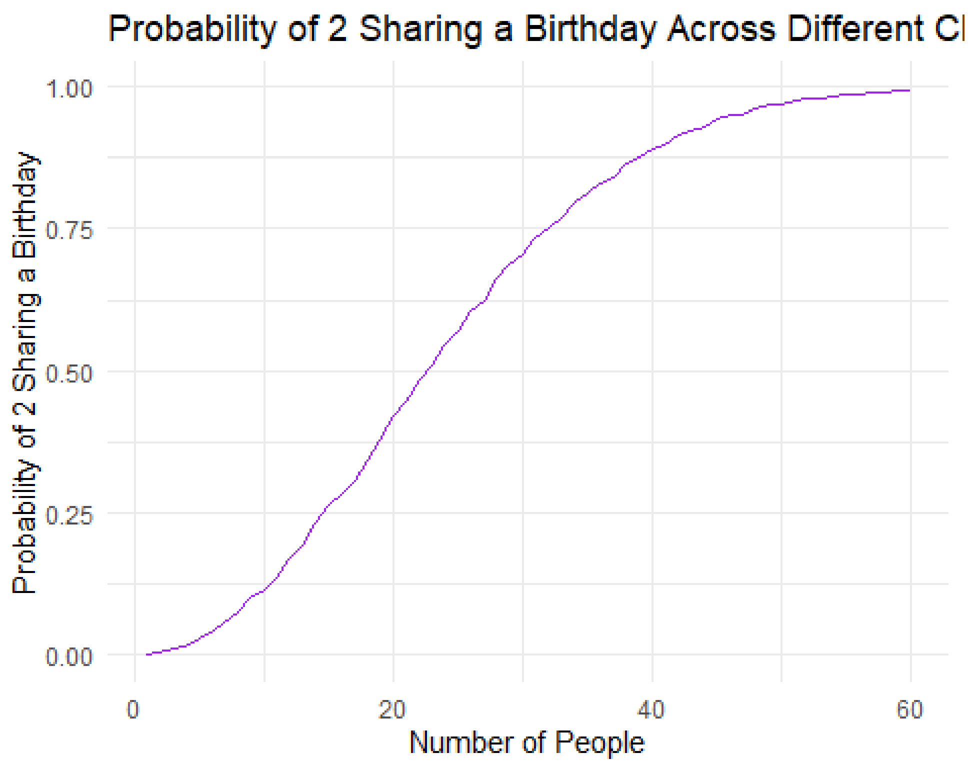
- Don't know what `replicate()` is? Run this in the console: `?replicate`

Listen Up.

What we have here is not just a simple coding exercise; it's a test of patience and understanding of computational limits. In our quest to uncover the mysteries of the birthday paradox, we're running a staggering **600,000 simulations**.

- For every number of people from 1 to 60, we simulate 10,000 iterations of birthday match
- To calculate the probability of at least two people sharing a birthday.
- We're not just asking, 'hey, do you two share a birthday?'
- We're meticulously combing through every possible combination of birthdays.
- So when you hit 'run' and it seems like time stands still, remember, your computer is doing the heavy lifting of thousands of birthday parties.
- In short, when you see that run time creeping up, don't panic. just wait a minute (or 57 seconds, to be precise).
- Why 57 seconds? Because that's how long it took on my machine. Yours may be longer.
- Your computer is working hard to reveal the fascinating probabilities behind the birthday paradox.

```r
# Start the timer
start.time <- Sys.time()
# Similar function to same_birhtday()
compute_prob_two <- function(n, B = 10000) {
  results <- replicate(B, {
    # roll those 365-sided dice again
    birthdays <- sample(1:365, n, replace = TRUE)
    max(table(birthdays)) >= 2 # same same but different
  })
  mean(results)
}

n <- seq(1, 60) # expand the group size to 60
prob_two <- sapply(n, compute_prob_two) # sapply might take longer to run

qplot(n, prob_two, geom="line") +
  geom_line(aes(y = prob_two), color = "purple") +
  xlab("Number of People") +
  ylab("Probability of 2 Sharing a Birthday") +
  ggtitle("Probability of 2 Sharing a Birthday Across Different Class Sizes")
+
  theme_minimal()
```

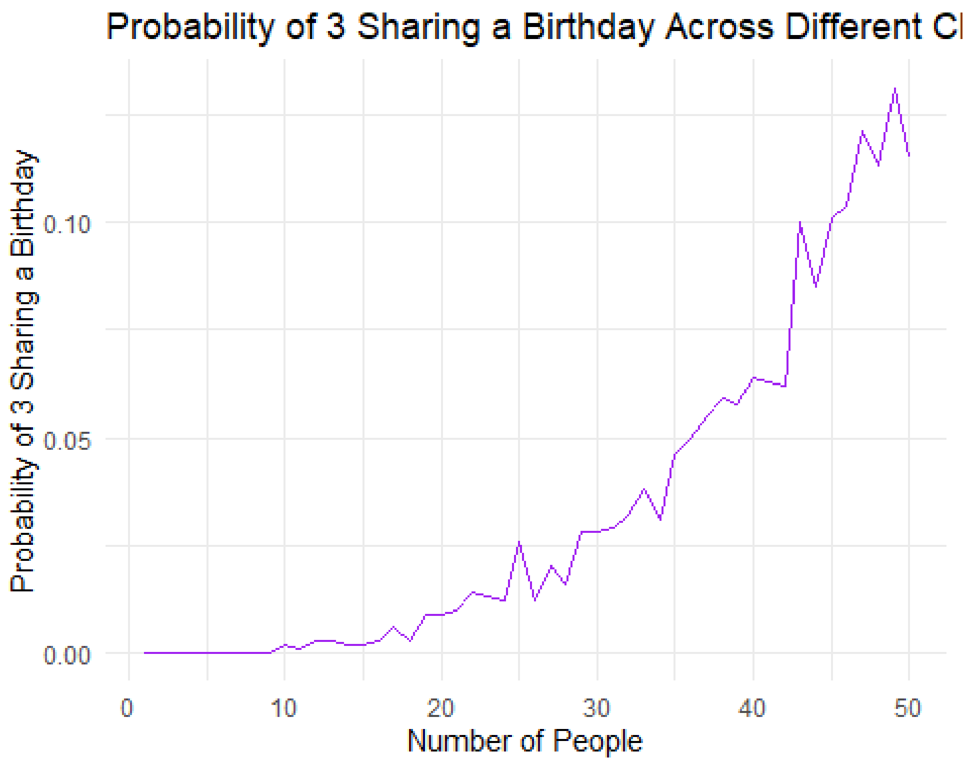


```
# Calculate the elapsed time
elapsed.time <- round(Sys.time() - start.time, 2)
paste("Elapsed Time: ", elapsed.time, " seconds")

## [1] "Elapsed Time:  52.43  seconds"
```

Task 2.1 (5pts): Simulate & plot the probability that 3 people have the same birthday in a class of 50 students. (This wont take long to run)

1. Sample for a birthday is like throwing a 365-sided dice 'n' times
2. check if any birthday is celebrated by at least 3 people
3. generating a sequence from 1 to 50 to simulate different class sizes
4. using sapply to apply 'compute_prob_three' across all class sizes

```
compute_prob_three <- function(n, B=1000) {
  results <- replicate(B, {

    ################START CODE HERE##############
    birthdays <- sample(1:365, n, replace = TRUE) # 365-sided dice again
    any(table(birthdays) >= 3) # at least 3
    ################END CODE HERE################

  })
  mean(results)
}

    ################START CODE HERE##############
n <- 1:50 # vector for range of class sizes
prob_three <- sapply(n, compute_prob_three) # look at you, using sapply like
a pro...
    ################END CODE HERE################


ggplot(data.frame(n, prob_three), aes(x = n, y = prob_three)) +
  geom_line(color = "purple") +
  xlab("Number of People") +
  ylab("Probability of 3 Sharing a Birthday") +
  ggtitle("Probability of 3 Sharing a Birthday Across Different Class Sizes")
+
  theme_minimal()
```



Task 2.2 (5pts): Answer question 1-3

Examine table 2.1 (Dobrow p.48) below:

```
--------------------------
k    Exact   Approximate
---- ------- -------------
15   0.253      0.25
23   0.507       0.5
30   0.706      0.696
40   0.891      0.882
50   0.97       0.965
60   0.994      0.992
--------------------------
```

1. Considering the dramatic increase in probabilities as 'k' values rise in table 2.1, ***why do you think we stopped at k=60?*** *Discuss the implications of further increases in 'k' on the probability values, and consider both practical and theoretical limitations.* ***Why might extending this table beyond k=60 offer diminishing returns*** *for our understanding of the birthday paradox?*

Answer: The table stops at k=60 likely due to the diminishing returns on further increasing k. As k increases, the probability of at least two people sharing a birthday approaches 1 very rapidly. Beyond k=60, the increase in probability becomes less noticeable, practically reaching a near certainty. This is due to the nature of the birthday problem, where the probability increases non-linearly with k. Further extending the table beyond k=60 would offer diminishing returns because the additional computational effort required would provide little additional insight into the problem's dynamics.

2. The formula for calculating the probability of at least two people sharing a birthday in a group of 23 people is given by `1 - prod(seq(343,365))/(365)^23`. ***Explain the significance of subtracting this product from 1. What does the product represent, and why is it important to consider the complement of this probability when solving the birthday problem?***

Answer: Subtracting the product from 1 in the formula `1 - prod(seq(343,365))/(365)^23` represents calculating the complement of the probability of no two people sharing a birthday. The product represents the probability of no two people in the group of 23 having the same birthday. It is essential to consider the complement because it provides a simpler approach to calculating the probability of at least two people sharing a birthday. By subtracting the probability of no matches from 1, we obtain the probability of at least one match, which is the desired outcome in the birthday problem.

3. Looking at Graphs of Task 2 you completed above:

Why doesn't the probability of 3 people sharing the birthday have an S-curve like the probability of 2 people sharing a birthday?

Answer: The probability of 3 people sharing a birthday does not exhibit an S-curve like the probability of 2 people sharing a birthday because the dynamics of the birthday problem change with the number of people involved. As the number of people increases, the probability of at least two people sharing a birthday increases rapidly, resulting in the characteristic S-curve shape. However, when considering three people, the dynamics become more complex, leading to a different probability distribution. The probability of three people sharing a birthday may exhibit different patterns, influenced by factors such as group size and the distribution of birthdays.

HairEyeColors - Setup

Let's take a look at the `HairEyeColor` dataset. This dataset contains the hair and eye color of 592 statistics students. We'll use this dataset to explore conditional probabilities.

```r
show(HairEyeColor)
```

```
## , , Sex = Male
## 
##        Eye
## Hair    Brown Blue Hazel Green
##   Black    32   11    10     3
##   Brown    53   50    25    15
##   Red      10   10     7     7
##   Blond     3   30     5     8
## 
## , , Sex = Female
## 
##        Eye
## Hair    Brown Blue Hazel Green
##   Black    36    9     5     2
##   Brown    66   34    29    14
##   Red      16    7     7     7
##   Blond     4   64     5     8
```

```r
# lets get an easier table to work with
AllHairEye <- as.data.frame(HairEyeColor)

# Aggregate data across gender because we're inclusive like that
AllHairEye <- aggregate(Freq ~ Hair + Eye, data = AllHairEye, sum)
total_students <- sum(AllHairEye$Freq) # avoid recalculation, really selling
you the basics here

# rename for clarity
names(AllHairEye) <- c("HairColor", "EyeColor", "Freq")

# How many students have Brown eye color?
brown.eyes <- AllHairEye %>%
  filter(EyeColor == "Brown") %>%
  pull(Freq) %>% sum()
print(paste("Students with brown eyes:", brown.eyes))
```

```
## [1] "Students with brown eyes: 220"
```

```r
# How many students have Blonde hair?
blonde.hair <- AllHairEye %>%
  filter(HairColor == "Blond") %>%
  pull(Freq) %>% sum()
print(paste("Students with blonde hair:", blonde.hair))
```

```
## [1] "Students with blonde hair: 127"
```

```r
# How many black haired students have brown eyes?
black.hair.brown.eyes <- AllHairEye %>%
  filter(HairColor == "Black", EyeColor == "Brown") %>%
  pull(Freq)
print(paste("Black haired Students with brown eyes:", black.hair.brown.eyes))
```

```
## [1] "Black haired Students with brown eyes: 68"
```

```r
# What is the percentage of students with Green eyes?
green.eyes.freq <- AllHairEye %>%
  filter(EyeColor == "Green") %>%
  pull(Freq) %>% sum()
green.eyes.count <- green.eyes.freq / total_students * 100
print(paste("Percentage of Students with green eyes:", green.eyes.count))
```

```
## [1] "Percentage of Students with green eyes: 10.8108108108108"
```

```r
# What percentage of students have red hair and Blue eyes?
red.hair.blue.eyes <- AllHairEye %>%
  filter(HairColor == "Red", EyeColor == "Blue") %>%
  pull(Freq) %>% sum()
red.hair.blue.eyes.freq <- red.hair.blue.eyes / total_students * 100
print(paste("Percentage of Students with red hair and blue eyes:",
red.hair.blue.eyes.freq))
```

```
## [1] "Percentage of Students with red hair and blue eyes: 2.87162162162162"
```

Let’s get the percentage of students with brown eyes **given** they have red hair.

```r
# How many people have red hair?
red.hair <- AllHairEye %>%
  filter(HairColor == "Red") %>%
  pull(Freq) %>% sum()

# How many people have red hair and brown eyes?
red.hair.brown.eyes <- AllHairEye %>%
  filter(HairColor == "Red", EyeColor == "Brown") %>%
  pull(Freq) %>% sum()

# What is the percentage of people with brown eyes given they have red hair?
browneyes.given.redhair <- red.hair.brown.eyes / red.hair * 100
```

```
print(paste("Percentage of people with brown eyes given they have red hair:",
browneyes.given.redhair))

## [1] "Percentage of people with brown eyes given they have red hair:
36.6197183098592"
```

Let’s isolate male students and calculate the probability of males with brown hair given they have Brown Eyes:

```
# Split data by gender
MaleHairEye <- as.data.frame(HairEyeColor[,, "Male"])
names(MaleHairEye) <- c("HairColor", "EyeColor", "Freq")

# What percentage of Males have Brown Hair given they have Brown Eyes
male.brownhair.given.browneyes <- MaleHairEye %>%
  filter(EyeColor == "Brown") %>%
  summarise(Percentage = sum(Freq[HairColor == "Brown"]) / sum(Freq) * 100)

print(paste("probability of males with Brown Hair given they have Brown
Eyes:", male.brownhair.given.browneyes))

## [1] "probability of males with Brown Hair given they have Brown Eyes:
54.0816326530612"
```

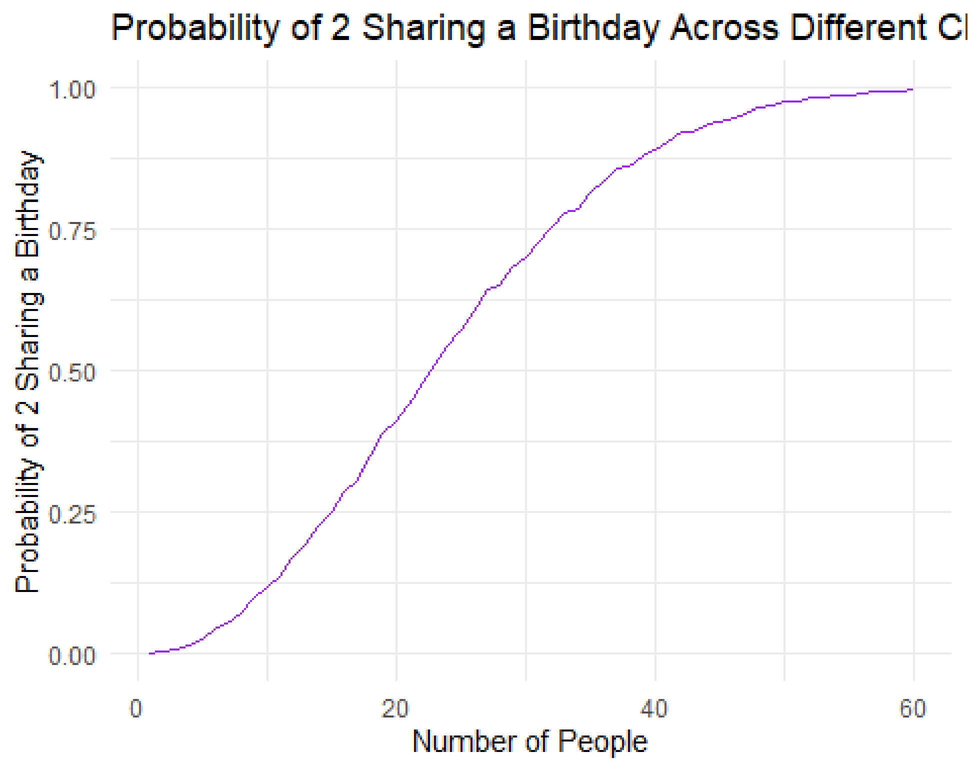
Task 3 Conditions (10pts)

Given that a person is Female, what’s the probability of having Red Hair and Hazel Eyes?

```
################START CODE HERE##############

# Split data by gender
FemaleHairEye <- as.data.frame(HairEyeColor[,, "Female"])

# You probably forgot that you need to add the column names again
names(FemaleHairEye) <- c("HairColor", "EyeColor", "Freq")

female.redhair.hazeleyes <- sum(FemaleHairEye$Freq[FemaleHairEye$HairColor ==
"Red" & FemaleHairEye$EyeColor == "Hazel"]) / sum(FemaleHairEye$Freq) * 100

################END CODE HERE################

print(paste("probability of having Red Hair and Hazel Eyes, given a female
student:", female.redhair.hazeleyes))

## [1] "probability of having Red Hair and Hazel Eyes, given a female
student: 2.23642172523962"
```

Refer back to the sim_birthday_setup chunk in Task2:

```
# Similar function to same_birhtday()
compute_prob_two <- function(n, B = 10000) {
  results <- replicate(B, {
    # roll those 365-sided dice again
    birthdays <- sample(1:365, n, replace = TRUE)
    max(table(birthdays)) >= 2 # same same but different
  })
  mean(results)
}

n <- seq(1, 60) # expand the group size to 60
prob_two <- sapply(n, compute_prob_two) # sapply might take longer to run

qplot(n, prob_two, geom="line") +
  geom_line(aes(y = prob_two), color = "purple") +
  xlab("Number of People") +
  ylab("Probability of 2 Sharing a Birthday") +
  ggtitle("Probability of 2 Sharing a Birthday Across Different Class Sizes")
+
  theme_minimal()
```



Using the pbirthday function we can simplify things:

```
pbirthday(
  n,             #The number of people
  classes = 365, #How many distinct categories the people could fall into
  coincident = 2 #The number of people to fall in the same category
  )
pbirthday(23)
```

[1] 0.5072972

Extra Credit (5pts):

In one line of code, generate the same s-curve plot, using `pbirthday()` and `sapply()`

```
# one line - 36 characters is enough
```

qplot(1:60, sapply(1:60, pbirthday, classes = 365), geom="line") + geom_line(aes(y = sapply(1:60, pbirthday, classes = 365)), color = "purple") + xlab("Number of People") + ylab("Probability of 2 Sharing a Birthday") + ggtitle("Probability of 2 Sharing a Birthday Across Different Class Sizes") + theme_minimal()