# Lab 2: Basic Probability

San Diego State University - STAT550

Aiden Jajo

In this lab we will work through two basic probability problems, and in the process practice more with RMarkdown.

**Packages needed:** knitr, xtable, pander

**R Markdown: Automated intro text when creating a new .Rmd file in RStudio** This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

## Task 1: Coin flipping

This task illustrates the interpretation of a probability as the long run relative frequency of an event after a large number of trials.

Dobrow presents R code on pages 29 for simulating coin tosses. We perform the experiment of observing the number of heads after tossing a fair coin 100 times (probability of a heads on any one toss is 50%). Just like rolling a die, we can use the R function `sample` to flip a coin. Though recognize there are only two outcomes: heads (1) and tails (0). We will report the number of heads after 50 tosses and intermediary output. We will also graphically display the cumulative proportion of heads again the coin toss (1 to 100). The `type="l"` parameter in the R function `plot` will draw a solid line. *Always label your axes! In RMarkdown, a graph title is useful too.*

### Code set-up

```
simnum = 100 # number of coin flips

# Simulate fair coin flips: heads = 1, tails = 0
coinflips = sample(0:1, simnum, replace = TRUE)

# Calculate cumulative sum of heads after each toss
heads = cumsum(coinflips)

# Calculate running proportion of heads
prop = heads/(1:simnum)

# Report cumulative number of heads after first 6 flips
head(heads)
```

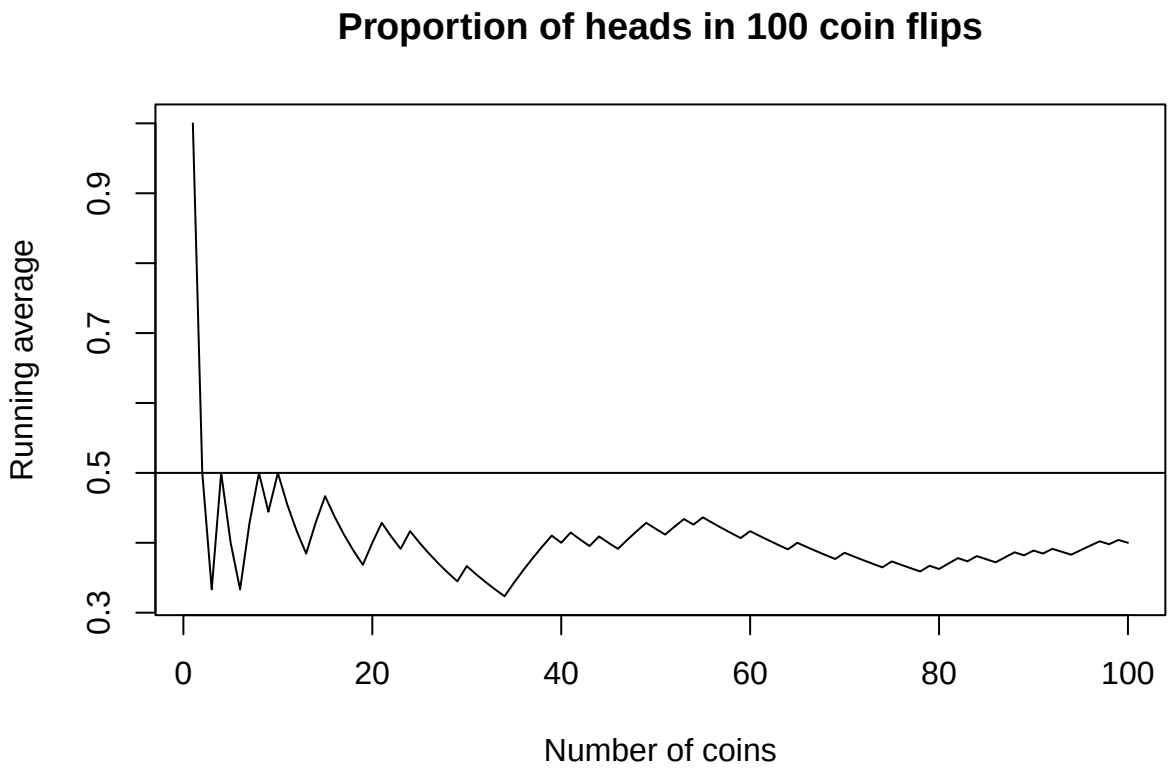
```
## [1] 1 1 1 2 2 2
```

```
# Report heads after 50 tosses
heads[50]
```

```
## [1] 21
```

```
# Create running mean plot for proportion of heads
plot(1:simnum, prop, type="l",
     xlab="Number of coins",
     ylab="Running average",
     main="Proportion of heads in 100 coin flips")

# Add reference line at 50% (true probability)
abline(h=0.5)
```




**The problem**

Let us now flip a "biased" coin. Perform the experiment of observing the number of heads after tossing a coin 1000 times, with the probability of getting a heads on any one toss being 40%. To change the probability in the R function `sample` use the parameter `prob=c(0.6,0.4)`; note that we need to specify the probability of a tails (0) and a heads (1) in this parameter.

**Report the following:**

- Proportion of heads after 10, 50, 100, 200, and 500 tosses (see table code chunk below under "RMarkdown presenting output"!)
- Plot of the cumulative proportion of heads vs. coin toss number (1 to 1000); label the axes and title the graphic appropriately!
- On the plot, draw a horizontal line at *y=0.40*, the probability of tossing a head for this coin

```
simnum = 1000 # number of coin flips

# Flip biased coin: P(heads) = 0.4, P(tails) = 0.6
coinflips = sample(0:1, simnum, replace = TRUE, prob = c(0.6, 0.4))

# Calculate cumulative sum of heads after each coin toss
heads = cumsum(coinflips)

# Calculate running proportion of heads after each coin toss
prop = heads/(1:simnum)

# Display values needed for the table
heads[10]
```

```
## [1] 5
```

```
heads[50]
```

```
## [1] 21
```

```
heads[100]
```

```
## [1] 39
```

```
heads[200]
```

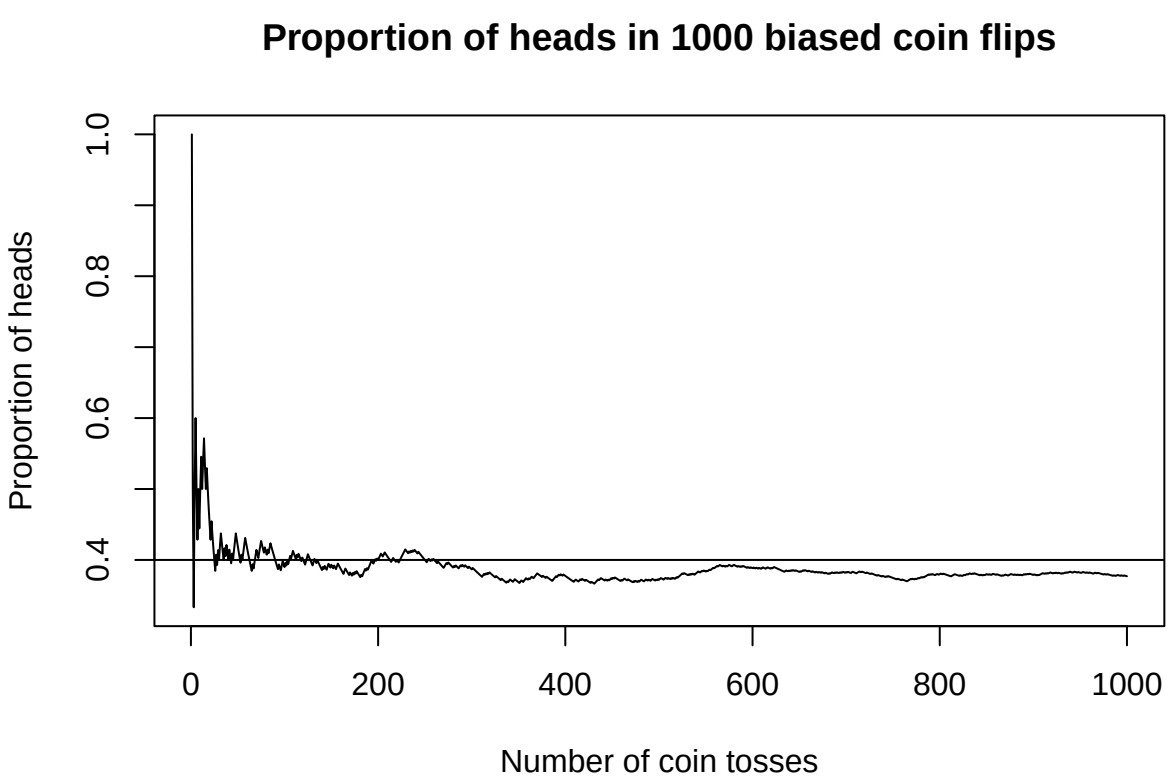
```
## [1] 80
```

```
heads[500]
```

```
## [1] 186
```

```
# Create running mean plot for proportion of heads
plot(1:simnum, prop, type="l",
     xlab="Number of coin tosses",
     ylab="Proportion of heads",
     main="Proportion of heads in 1000 biased coin flips")

# Add reference line at 40% (true probability for biased coin)
abline(h=0.4)
```




**Questions:**

- Describe the behavior of the graphic (cumulative proportion of heads) during the first 150 tosses (1-150), next 150 tosses (151-300), and then later tosses.

*During the first 150 tosses, the proportion of heads shows a large fluctuation around the true probability of 0.40. During the next 150 tosses, the curve begins to stabilize closer to the true probability of 0.40 and the variability begins to decrease. During the later tosses, the curve becomes very stable. It gets closer to the true probability of 0.40 with very little fluctuations.*

- What do you notice about the limiting value of the curve in your plot?

*The limiting value of the curve in the plot approaches 0.40. This is the exact true probability of getting heads for a coin flip with a biased coin.*

- Why would you expect the behavior you discuss in the previous two bullets?

*I would expect the behavior discussed in the previous bullets because the more trials ran, the observed proportion gets closer to the theoretical probability. During early experimentation, random variation has a great impact on the proportion shown. While the coin continues to flip, the proportion stabilizes.*

**RMarkdown presenting output:** Below is code to present a table for the proportion of heads after 10, 50, and 100 tosses.

Reminders:

The `echo=FALSE` parameter prevents printing of code from a code chunk. The `include=FALSE` parameter prevents printing of output from a code chunk. The `results=asis` allows the LaTeX code produced by xtable to be compiled and output.

```
# we will create a table using xtable and pander
library(knitr)
library(xtable)
library(pander)

# output desired summary statistics
# formatC used so integer coin tosses do not have a decimal place in the figure!
numtoss = formatC(c(10, 50, 100, 200, 500), digits=0, format="d", flag="#")

# Extract number of heads at specified toss counts
num.heads = c(heads[10], heads[50], heads[100], heads[200], heads[500])
num.heads = formatC(num.heads, digits=0, format="d", flag="#")

# formatC used here so proportions have exactly two decimal places (including zeros at the end)!
prop.heads = c(prop[10], prop[50], prop[100], prop[200], prop[500])
prop.heads = formatC(signif(prop.heads,digits=6), digits=2, format="f", flag="#")

# Build table structure
table.elts = rbind(numtoss, num.heads, prop.heads)
row.names(table.elts) = cbind("# coin tosses", "Number of heads", "Proportion of heads")

# Create and display formatted table
lab1.table = xtable(table.elts,
                    caption = "Proportion of heads for a given number of tosses of a biased coin.",
                    label="cointoss",
                    align = "|l|rrrrr|")
pander(lab1.table)
```

Table 1: Proportion of heads for a given number of tosses of a biased coin.

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| **# coin tosses** | 10 | 50 | 100 | 200 | 500 |
| **Number of heads** | 5 | 21 | 39 | 80 | 186 |
| **Proportion of heads** | 0.50 | 0.42 | 0.39 | 0.40 | 0.37 |

The table shows the proportion of heads after a certain number of tosses in a single simulation experiment. These proportions provide empirical estimates of the probability of a head for specified simulation sample sizes.

### The problem

Directly in the code chunk above, add columns for 200 tosses and 500 tosses. Hint: you will need to append these two elements to `numtoss`, `num.heads`, and `prop.heads`. Also, note that in the `xtable` function, the align is only for 3 right-justified columns; need to augment that to 5 right-justified columns.

## Task 2: Divisibility probability

This task provides a probability problem to explore if-then statements and functions in R. Consider an integer drawn uniformly at random from the numbers {1, 2, ..., 1000} such that each number is equally likely. We wish to simulate the probability that the number drawn is divisible by 3, 5, or 6.

### Code set-up

Dobrow presents R code on page 31 for simulating this experiment, and provides the exact probability calculation in Example 1.33. The code presents a slick application of the `replicate` R function, one we used in the R Introduction lab. In particular, a function is written which draws the number at random from the integers 1 to 1000 and then checks if it is divisible by 3, 5, or 6. It uses modular arithmetic, `x%%n` being $x$ *mod* $n$ in R. For example, if the remainder of the number divided by 3 (modulus) is 0, then the number is divisible by 3! We will then repeat the function (experiment) 1000 times to get the empirical probability.

```
# simdivis() simulates one trial
simdivis = function(){
  # Draw a number at random from the integers 1 to 1000
  num = sample(1:1000, 1)

  # Determine if the number is divisible by 3, 5 or 6 by checking if the remainder is 0
  if (num%%3==0 || num%%5==0 | num%%6==0) 1 else 0
}

# Replicate the experiment 1000 times
simlist = replicate(1000, simdivis())

# Compute the estimated probability as the proportion of successes
mean(simlist)
```

**The true probability that a randomly drawn integer between 1 and 1000 is divisible by 3, 5, or 6 is 0.467.**

```
## [1] 0.471
```

### The problem

Simulate the probability that a random integer between 1 and 5000 is divisible by 4, 7, or 10.

```
# Function to simulate one trial
simdivis4710 = function() {
  # Draw random number from 1 to 5000
  num = sample(1:5000, 1)

  # Check if divisible by 4, 7, or 10
  if (num%%4==0 || num%%7==0 || num%%10==0) 1 else 0
```

```
}

# Run simulation with 100 experiments
sim100 = replicate(100, simdivis4710())
prob100 = mean(sim100)
prob100
```

**The true probability that a randomly drawn integer between 1 and 5000 is divisible by 4, 7, or 10 is 0.40.**

```
## [1] 0.42
```

```
# Run simulation with 1000 experiments
sim1000 = replicate(1000, simdivis4710())
prob1000 = mean(sim1000)
prob1000
```

```
## [1] 0.403
```

```
# Run simulation with 10000 experiments
sim10000 = replicate(10000, simdivis4710())
prob10000 = mean(sim10000)
prob10000
```

```
## [1] 0.4053
```

```
# Run simulation with 100000 experiments
sim100000 = replicate(100000, simdivis4710())
prob100000 = mean(sim100000)
prob100000
```

```
## [1] 0.3993
```

**Questions:**

- Present the empirical probability based on repeating the experiment 100, 1000, 10000, and 100000 times. Consider using the `xtable` code chunk from the first task to build a table for these values.

```
library(knitr)
library(xtable)
library(pander)

# Format number of experiments
num.experiments = formatC(c(100, 1000, 10000, 100000), digits=0, format="d", flag="#")

# Format empirical probabilities with 4 decimal places
empirical.probability = c(prob100, prob1000, prob10000, prob100000)
empirical.probability = formatC(empirical.probability, digits=4, format="f", flag="#")

# Build table structure
```

```
table.elts2 = rbind(num.experiments, empirical.probability)
row.names(table.elts2) = cbind("Number of experiments", "Empirical probability")

# Create and display formatted table
lab2.table = xtable(table.elts2,
                    caption = "Empirical probabilities for divisibility by 4, 7, or 10.",
                    label="divtable",
                    align = "|l|rrrr|")
pander(lab2.table)
```

Table 2: Empirical probabilities for divisibility by 4, 7, or 10.

| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| **Number of experiments** | 100 | 1000 | 10000 | 100000 |
| **Empirical probability** | 0.4200 | 0.4030 | 0.4053 | 0.3993 |

- How do these values compare to the truth?

*As the number of experiments grows from 100 to 100,000, the empirical probabilities get closer to the true probability of 0.40. With fewer experiments, like 100, there is more variability and it's likely that the estimate would be further from 0.40. With a greater amount of experiments, like 100,000, the empirical probability is likely to be closer to 0.40.*

- Extra credit: show that the true probability that a random integer between 1 and 5000 is divisible by 4, 7, or 10 is 40%?

*floor(5000/4)=1250 floor(5000/7)=714 floor(5000/10)=500 floor(5000/28)=178 floor(5000/20)=250 floor(5000/70)=71 floor(5000/140)=35 1250+714+500-178-250-71+35=2000 2000/5000=0.40*