Lab 2: Basic Probability

Spring 2020

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

Packages needed:

knitr, xtable, pander

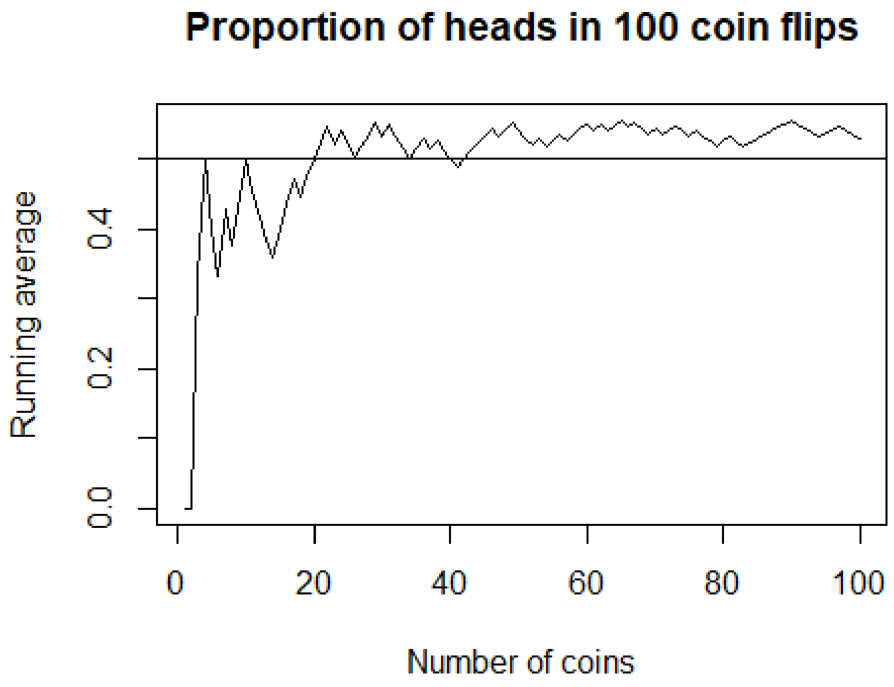
Task 1: Coin flipping

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 24 and 453 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
coinflips = sample(0:1, simnum, replace = TRUE) # flip the coin: heads = 1, t
ails = 0
heads = cumsum(coinflips) # cumulative sum of number of heads after each coin
toss
prop = heads/(1:simnum) # running proportion of heads after each coin toss
head(heads) # report cumulative number of heads after each of the first 6 fli
ps

## [1] 0 0 1 2 2 2

heads[50]

## [1] 27

# 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")
abline(h=0.5) # add a line at 50%
```



The problem

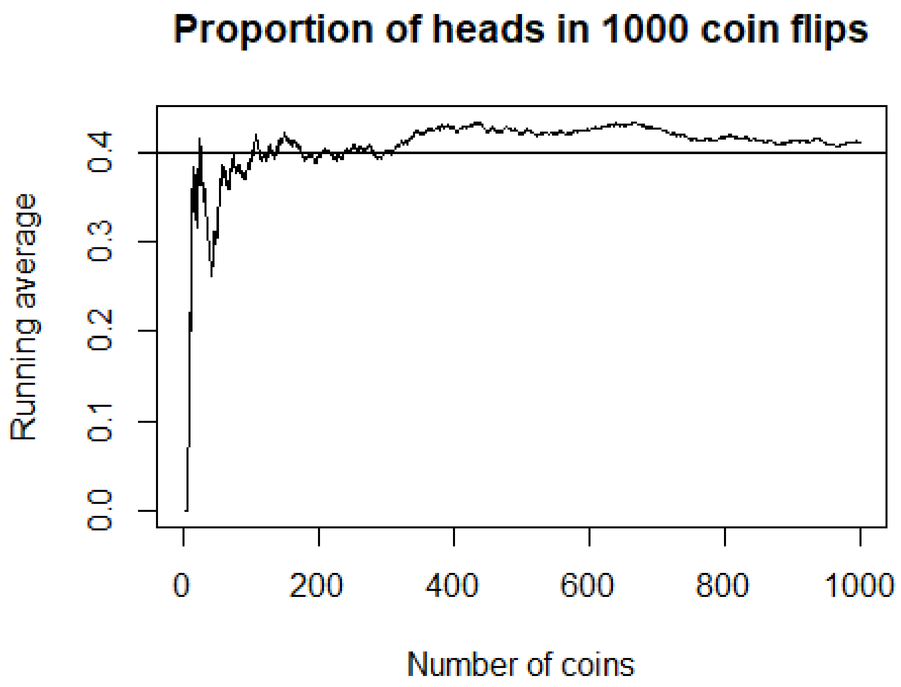
Let us now flip a "biased" coin. Perform the experiment of observing the number of heads after tossing a coin 1000 times, with the probability of getting a heads on any one toss being 40%. To change the probability in the R function `sample` use the parameter `prob=c(0.6,0.4)`; note that we need to specify the probability of a tails (0) and a heads (1) in this parameter. Note that if you do not want the code presented in your html report, use the parameter `echo=FALSE` in the code chunk.

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
coinflips = sample(0:1,simnum,replace = TRUE, prob = c(0.6,0.4)) # flip the c
oin: heads = 1, tails = 0
heads = cumsum(coinflips) # cumulative sum of number of heads after each coin
toss
prop = heads/(1:simnum) # running proportion of heads after each coin toss
```

```
head(heads) # report cumulative number of heads after each of the first 6 fli
ps

## [1] 0 0 0 0 0 0

heads[10]

## [1] 2

heads[50]

## [1] 16

heads[100]

## [1] 39

heads[200]

## [1] 79

heads[500]

## [1] 211

# 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 1000 coin flips")
abline(h=0.4) # add a line at 50%
```



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. *The first 150 tosses can go either way, either head or tails. The next 150 tosses however seem to level out at 40% and it stays around that range.*

- What do you notice about the limiting value of the curve in your plot? *The limiting value is the value we set the probability of heads for our unfair coin.*

- Why would you expect the behavior you discuss in the previous two bullets? *We would expect this behavior because after repeated simulations, the proportion of heads should be around 40%.*

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.

```r
# 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 figu
re!
numtoss = formatC(c(10, 50, 100,200,500), digits=0, format="d", flag="#")
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=
"#")
table.elts = rbind(numtoss, num.heads, prop.heads)
row.names(table.elts) = cbind("# coin tosses", "Number of heads", "Proportion
of heads")
lab1.table = xtable(table.elts, caption = "Proportion of heads for a given nu
mber of tosses of a biased coin.", label="cointoss", align = "|l|rrrrr|")
pander(lab1.table)
```

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	2	16	39	79	211
Proportion of heads	0.20	0.32	0.39	0.40	0.42

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 interger 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 25 for simulating this experiment, and provides the exact probability calculation in Example 1.20. 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.

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

```
# simdivis() simulates one trial
simdivis = function(){
  num = sample(1:1000, 1) # draw a number at random from the integers 1 to 10
00
  # determine if the number is divisible by 3, 5 or 6 by checking if the rema
inder is 0
  if (num%%3==0 || num%%5==0 | num%%6==0) 1 else 0
}
simlist = replicate(1000, simdivis()) # replicate the experiment 1000 times
mean(simlist) # compute the estimated probability as the proportion of times
the number is divisible by 3, 5, or 6

## [1] 0.448
```

The problem

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

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

```r
# simdivis() simulates one trial
simdivis = function(){
  num = sample(1:5000, 1) # draw a number at random from the integers 1 to 10
00
  # determine if the number is divisible by 3, 5 or 6 by checking if the rema
inder is 0
  if (num%%4==0 || num%%7==0 | num%%10==0) 1 else 0
}
simlist = replicate(100000, simdivis()) # replicate the experiment 1000 times

div=cumsum(simlist) # compute the estimated probability as the proportion of
times the number is divisible by 4, 7, or 10
prop = div / (1:100000)
# 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 figu
re!

numdiv = formatC(c(100,1000,10000,100000), digits=0, format="d", flag="#")
num.div = c(div[100], div[1000], div[10000],div[100000])
num.div = formatC(num.div, digits=0, format="d", flag="#")
prop.div = c(prop[100], prop[1000], prop[10000],prop[100000])
prop.div = formatC(signif(prop.div,digits=6), digits=2, format="f", flag="#")
# formatC used here so proportions have exactly two decimal places (including
zeros at the end)!

table.elts = rbind(numdiv, num.div, prop.div)
row.names(table.elts) = cbind("# numbers", "Divisible by 4/7/10", "Proportion
")
lab1.table = xtable(table.elts, caption = "Proportion of numbers divisible by
4,7, or 10.", label="cointoss", align = "|l|rrrr|")
pander(lab1.table)
```

Proportion of numbers divisible by 4,7, or 10.

	1	2	3	4
# numbers	100	1000	10000	100000
Divisible by 4/7/10	44	400	3995	39911

Proportion	0.44	0.40	0.40	0.40

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.

See table directly under code.

- How do these values compare to the truth?

They are very close to the true probability of 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%?

[Put your answer here, in between the asterisks]