Math 20 – Lab 2 (due August 10)

Each experiment asks you to perform some simulations and describe the results. When you're asked to explain your findings, I'm not looking for long paragraphs; just a couple of sentences will do. Please type your responses (use LaTeX, if you know how, or else you can use Word or another text editor, though it surely won't look as nice!); your responses should be written in the same document and titled `Math 20 Lab 2 - [your first and last name]`.

Each experiment also asks you to submit code. You must name the files as instructed and submit them with your completed assignment. Please *comment your code* so that I know what you're trying to do.

Please send everything to me (one .docx/.doc/.pdf and two .R files) **by email** by the beginning of class on Friday, August 10.

1. One way to estimate the area or volume of a geometric figure is to select points uniformly at random from a simple region containing the figure (such as a box or a cube) whose area or volume you know, then calculate the proportion of those points that fell within the figure whose area or volume is to be estimated. In this experiment, we will use Monte Carlo simulations to estimate the volume of the unit n-dimensional sphere for several values of n.

   (a) First, let's estimate the area of the unit circle. The unit circle is a circle of radius 1, centered at $(0,0)$. Pick points uniformly at random from the square with opposite corners $(-1,-1)$ and $(1,1)$ (use the `runif` function for this). A randomly chosen point (x,y) falls within the circle if and only if $x^2+y^2 \leq 1$. Pick points until the ratio
   $$\frac{\text{number of points in the circle}}{\text{number of points in the box}}$$
   starts to converge to a few decimal places, then use this to estimate the area of the circle. (This make take 10s or 100s of millions of points–convergence is slow.) Check you answer against what you know the true area of the unit circle is. (It will help to have your code print out the updated ratio every so often, but if you print too often it will become the bottleneck for your program. I'd recommend something like every 100,000 or 1 million inerations. Ask me if you need help figuring out how to do this.)

   (b) Repeat the same experiment to estimate the volume of the unit sphere. Now, you need to randomly pick points within the cube with opposite corners $(-1,-1,-1)$ and $(1,1,1)$, and a point (x,y,z) is in the sphere if and only if $x^2+y^2+z^2 \leq 1$.

   (c) Repeat the same experiment to estimate the volume of the n-dimensional sphere for all n between 4 and 10, inclusive. Present a table that summarizes your findings for all n between 2 and 10, and describe your results. Form a conjecture about what happens as n grows.

(d) Gather some more evidence for your conjecture by testing higher values of n.

(e) Submit your code for this problem in a file named `spheres-yourlastname.R`. Make sure you've commented your code so I understand what it does and how to run it.

2. You decide to start investing in bitcoin on a day when the price is exactly \$100 per bitcoin. Bitcoins can be bought or sold in any fractional quantity–for example, if the price of bitcoin is \$30 one day, and you only have \$10, you can buy exactly 1/3 of a bitcoin.

   In this experiment, we will simulate several buying/selling techniques to see if we can beat the system and become millionaires. For simplicity, we will assume that the price of bitcoin changes exactly once per day, and that at the end of each day you can decide whether to buy more bitcoin (convert cash to bitcoin) or sell bitcoin (convert bitcoin to cash). To simulate the fluctuation in bitcoin price, use the following R function:

```
price <- function(days, initial_price) {
  return(cumprod(c(initial_price, exp(0.02*rnorm(days)))))
}
```

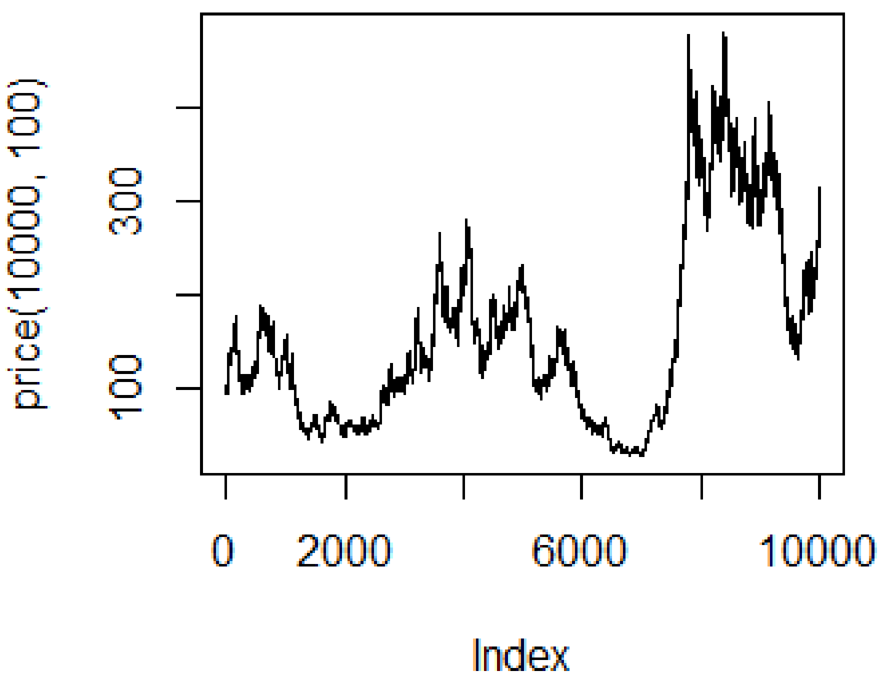
   Call this function with two variables: the first is the number of days you want to simulate, and the second is the initial price. The function will return a vector V of prices such that $V[i]$ is the price on day i.

   For example,

```
> daily_prices <- price(10, 100)
> daily_prices
 [1] 100.00000 102.23581 98.19005 100.10508 98.43542 101.79972
 [7] 103.20257 102.63872 99.83685 100.63760 101.12234
> daily_prices[3]
[1] 98.19005
```

   You can also look at plots using the command

```
> plot(price(10000, 100), type="l")
```



Your code will produce ("know") the prices for all future days before you start trading, but no cheating! You can only use the current day's information (ie. prices from today and previous days) to make buy/sell decisions.

(a) First we'll try Strategy A. You start with \$1000, and on Day 1, when the price is \$100 per bitcoin, you convert it all to bitcoin (in other words, buy 10 bitcoins). Then, you wait. If the price of bitcoin ever reaches $\geq$ 20% more than what you paid for it, you sell all of your bitcoin for the new price. After selling, you wait for the price to go down at least 10% from your sell point, then you use all of your cash to buy again, repeating the process (waiting until the price increases to at least 20% more than this to buy, then waiting until it decreases at least 10% to sell, etc.). Suppose you follow this procedure for 10,000 days. At the end, your net worth is equal to your cash holdings plus the cash value of your bitcoin holdings on that day. Run many simulations to find your expected net worth on day 10,000. Perhaps a much better metric is the following: how did you do compared to bitcoin as a whole? Would you be better off if you had just bought bitcoin on the first day and held it? Include plots in your write-up!

(b) Next we'll try Strategy B, which is a less extreme version of Strategy A. You start on Day 1 with \$1000, and you spend \$200 on bitcoin (in other words, buy 2 bitcoins). Every day, you do the following: if the price goes up from the previous day, you sell 20% of your current bitcoin holdings (convert to cash). If the price goes down from the previous day, you use 20% of your current cash holdings to buy more bitcoin. Run many simulations to find your expected net worth after 10,000 days. How does the strategy do compared to just buying and holding bitcoin? Include plots!

(c) Come up with another trading strategy (no cheating/time travel to look at future prices!), and test it against Strategies A and B. Describe your findings. How good of a strategy can you come up with? How well can you outperform the market?

(d) Submit your code for this problem in a file named `bitcoin-yourlastname.R`. Make sure you've commented your code so I understand what it does and how to run it.