POLS 396

Problem Set 2 Due: Tuesday Feb. 8 @ 1:20pm

## Task

Create a vector (named `myVector`) containing 1,000,000 normally distributed random numbers with a mean of zero and standard deviation of $\pi$. In other words, your vector should contain numbers that were randomly drawn from a normal (a.k.a Gaussian) distribution. Just prior to creating your vector, issue the following command: `set.seed(seed = 1)`.[1]

To verify that the random numbers in your vector are indeed normally distributed execute the following code and assess if the kernel density estimate looks vaguely like a “bell-curve."

```
> myDensity <- density(x = myVector)
> plot(x = myDensity)
```

With this vector in hand determine the following:

1. How many of the 1,000,000 random numbers are greater that $2\pi$?
2. What percentage of the 1,000,000 random numbers are greater that $2\pi$?
3. How many of the 1,000,000 random numbers are smaller that $-2\pi$?
4. What percentage of the 1,000,000 random numbers are smaller than $-2\pi$?
5. What percentage of the 1,000,000 random numbers are greater than $-\pi$ but smaller than $\pi$?
6. What percentage of the 1,000,000 random numbers are greater than $-2\pi$ but smaller than $2\pi$?
7. What percentage of the 1,000,000 random numbers are greater than $-3\pi$ but smaller than $3\pi$?

## Extra-Credit

I have created a vector for you, saved it and uploaded it to our website. You can download this vector directly into **R** using the following code:

```
> load(file = url("https://unca-pols.org/Files/Data/
   PetersVector.Rdata"))
```

Use the `ls()` function to verify that the object has been downloaded and the vector was created in your **R**-session.

```
> ls()
[1] "PetersVector"
```

[1] The `set.seed()` function ensures that we are all getting exactly the same set of random numbers. You need to excute this function everytime you want to recreate your vector.

With the vector loaded, answer the following:

1. How many square numbers does `PetersVector` contain?
2. Where in the vector is the largest and smallest square number located?

### Tips

Prior to solving the problem, try to produce a solution for a vector for which you know the answers. For example, consider the following vector.

```
> TestVector <- c(7, 16, 17, 64, 11)
```

For the vector above you know that it contains two square numbers (16 and 64). You also know that the smallest square number (i.e., 16) is located in second "slot" of the vector and the largest square number (i.e., 64) is the fourth element of the vector. If you can write a set of instructions that produces the answer for this vector, it should work for any arbitrary vector of natural numbers.

*Email me your code as a properly formated and annotated* `.R` *file!*