Lab 3

Updated Friday, October 27th at 11:00p

Submit via CCLE by Friday, November 4th, 10pm

Directions: Create an R Markdown document to complete the tasks below. Include all necessary lines of code and explain your work using complete sentences. Both the code you write and the outputs from `R` should be included in the compiled/knitted HTML document. Submit both the `.Rmd` and `.html` files to CCLE. Name the files `#########-lab03.Rmd` and `#########-lab03.html` where the `#########` are replaced by your Bruin ID.

1 Tidy Data

1.1 *Lord of the Rings*

The *Lord of the Rings* is a movie (and book) fantasy trilogy about a hobbit's journey to Mordor in hopes of destroying the ring of power. Having re-read the books recently, I highly recommend checking them out when you're not working on your Stats 20 assignments. The movies are good too, but make sure you find copies of the extended editions if you watch them, ya know ... when you're not working on your Stats 20 assignments.

You've been given three data files containing the number of words spoken by different races and sexes in each of the three *Lord of the Rings* films. They are `The_Fellowship_Of_The_Ring.csv`, `The_Two_Towers.csv` and `The_Return_Of_The_King.csv`.

1. Read the individual data files into `R` using the `readr` package.
2. Find a function in `dplyr` that combines the data together by rows and create a single data file with 9 observations and 4 varibles.
3. Use the `gather()` function from `tidyr` to create a variable called `Sex` which combines the `Male` and `Female` variables. The `Sex` variable should contain the values `Female` and `Male`. The number of words spoken in each `Film`, for each `Race` and each `Sex` should be contained in a variable called `Words`. This data should end up containing 18 observations and 4 variables.
4. Use this tidy dataset to answer the following questions:
   - What are the means, standard deviations and total number of number words spoken by each race in the films? Create a 3 row by 4 column table displaying this information.
   - For each film, describe the distribution of words spoken by each race. That is, which races spoke more, less or about the same number of words in each film. Include a plot to reference as evidence for your answer.

1.2 Comic books

1. Read in the `superheroes.csv` and the `publishers.csv` data sets. These data files contain information about comic book superheroes, whether they are good (heroes) or bad (villains), the character's sex, the publisher of the comic books and the year the comic book publishers were founded.
2. Join these two data files together to include only the characters whose publishers are contained in both datasets. Which character is dropped?
3. Join these two data files together to include only the publishers that are NOT contained in superheroes data.
4. Join these two data files together to create a data frame that keeps all of the publishers and has the superheroes' information appended to each publisher. What are the dimensions of this new data frame?

5. Join these two data files together too create a data frame containing all of the observations from both datasets. What are the dimensions of this new data frame?

2 Simulations using loops

Simulations are incredibly useful in statistics. Statisticians/data scientists use simulations to answer questions that are too complex to solve mathematically or recreate real-world phenomenon (climate change for example).

2.1 Central limit theorem

According to Wikipedia, the central limit theorem states that the mean of a sufficiently large number of independent random sameples will be approximately normally distributed, regardless of the underlying distribution. I.e. if we take many samples independently from one another, even if the distribution those samples come from is skewed, the distribution of the sampled means will be normally distributed.

In this problem, we'll use a `for` loop and `if/else` to demonstrate the central limit theorem.

1. Use the `rbinom()` function to simulate, one time, the number of "successes" from flipping a coin 10 times where the probability of success is 0.2.
2. Next, use `set.seed(2016)` and then use the `rbinom()` function to draw 100,000 random samples from a binomial distribution of size 10 and probability of success equal to 0.2. That is, we flip 10 coins that landed on "Heads" 20% of the time and then repeat the flipping of 10 coins 99,999 more times, each time counting the number of "Heads". Assign these 100,000 random samples the name `draws`.
3. Create a histogram of the 100,000 `draws`. Make sure to specify the `binwidth` so that you can get a good idea of the shape of the distribution. Based on your plot, describe the center, shape and spread of the number of successes. Pay particular attention to the shape of the distribution.
4. Use the `matrix` function to turn your 100,000 draws into a matrix with 40 rows and 2,500 columns (filling the matrix by column … i.e `byrow = FALSE`).
5. Use a `for` loop to calculate the column means of your matrix.
   - Start by using the `vector()` function to create a vector of length 2,500 called `means`.
   - Then use a `for` loop to replace each element in the vector `means` with the column mean of your matrix.
6. Create a histogram of the `means` and describe the center, shape and spread. Use the `mean()` and `sd()` functions to describe the center and spread of `means` and then interpret these values. Describe how the center, shape and spread is different than the distribution of the `draws` from the binomial distribution.

2.2 Random walk

Random walks are used to describe all sorts of real-life phenomenon (such as the stock market or an intoxicated person walking down the sidewalk). In this problem, we'll simulate a random walk and see how long it takes for us to return to our starting point.

NOTE: In this problem, we'll use a `while()` loop which, if not coded properly, can run for an infinitely long time. SAVE YOUR WORK BEFORE AND WHILE WORKING ON THIS PROBLEM. If your code seems to be taking a VERY long time, hit the *stop* sign symbol in the upper right hand corner of your *console* to stop your loop. THIS CAN POTENTIALLY CRASH YOUR `R` SESSION … **SAVE YOUR WORK**

1. Create the following objects in `R`.

```r
place <- 0
zeros <- 0
path <- c()
iteration <- c()
```

- `place` describes our current position. We'll randomly add `+1` or `-1` to place during each iteration of the loop.
- `zeros` describes how many times we're returned to our starting position, i.e. `place = 0`.
- `path` is an empty vector we'll use to describe what position we were at for each iteration.
- `iteration` is an empty vector we'll use to keep track of the set of iterations.

2. Next, fill in the blanks below using the comments as a guide. In your assignment, feel free to remove the comments and replace them with your own.

```
# Set the seed to 2016 so your code matches everyone elses
set.seed(2016)

# We want to loop through the code in the curly braces until we return to our
# starting point.Place a condition on the zeros object that will be TRUE until
# we return to our starting point.
while (____) {
  # Overwrite the iteration vector by (1) keeping all of the values currently
  # in our iteration vector and (2) adds the current length of the iteration
  # object to our iteration vector
  ____ <- c(____, ____(____))

  # Overwrite the path vector by (1) keeping all of the values currently stored
  # in our path vector and (2) adding our current place to the vector.
  ____ <- c(____, ____)

  # Use the runif() function to draw a single number, uniformally, between the
  # values 0 and 1.
  draw <- runif(____)

  # Write an if/else statement that does the following:
  # If draw >= 0.5, add +1 to our current place.
  # else, add -1 to our current place.
  if (____) {
    place <- ____ + 1
  } else {
    ____ <- ____ - 1
  }

  # Write an if statement so that once we reach our starting place
  # (i.e. place == 0):
  # (1) We include our current iteration to the iteration vector (just like you
  # did before)
  # (2) We include our current place to the path vector (just like you did
  # before)
  # (3) overwrite the zeros object to be zeros + 1
  #
  # Your while loop should stop once you're place object is 0.

}
```

3. Create a line plot that shows that `path` of our random walk as the number of `iterations` increased. Be sure to give your plot a descriptive title.
4. How long did it take for our random walk to return to our starting place?