PS132B Problem Set 2

Due 12:29PM Tuesday February 14, 2023

Please submit this assignment by uploading your R Markdown code file (`.Rmd`) AND either an html (`.html`) or pdf (`.pdf`) output onto bCourses before the due time, with easy-to-recognize file names (e.g., `pset2_KirkBansak.Rmd`). Your homework will be graded based on completeness, accuracy, and readability of both code and written answers.

The point allocation in this problem set is given by:

Q1.1	Q1.2	Q1.3	Q1.4	Q1.5	Q1.6	Q2.1	Q2.2	Q2.3
4	4	4	4	4	4	8	8	8

Q3.1	Q3.2	Q3.3	Q3.4	Q3.5	Q3.6	Q3.7	Q3.8	Q3.9	Q3.10	Total
4	4	4	4	4	10	4	10	4	4	100

Part 1

1. Begin by repeating 1.1, 1.2, and 1.3 from the first problem set. That is: (a) Create a numeric vector that is the sequence of all integers between 1 and 1000 and assign this vector the name `vec1`; (b) Create another vector of the same 1000 integers but whose order is randomized; and (c) Bind these two vectors together in a data frame, and call the data frame `dat`, making sure that the first column of `dat` corresponds to `vec1` and the second column corresponds to `vec2`.

   Note: When you create your second vector, you should set the random number generator seed to 12345 to enable reproducibility. That is, run the line `set.seed(12345)` before creating the second vector.

2. Determine which elements of column 2 of `dat` contain the numbers 2, 47, 290, and 812. Store the indices of these elements (i.e. the row numbers) in a manner of your choosing.

3. Now replace the instances of the numbers 2, 47, 290, and 812 in column 2 of `dat` with missing values (NA).

4. Rename the columns of `dat`. The new names, in order, should be `caseid` and `wage`.

5. Calculate the mean, median, and standard deviation of `wage` and report those values. Since there are NA values, you will need to use the "`na.rm = TRUE`" argument when calculating these values.

6. Create a new data frame, `dat2`, that is `dat` without the missing values. In other words, delete all observations (rows) that have missing wage values.

Part 2

Find the data file named `CAcities.csv` on bCourses, and save it onto your computer in the same folder/directory as the `R Markdown` file you are creating for this assignment. This small dataset contains the names and populations (in 2020) of the 10 largest cities in California. Read the data into `R` as a data frame. You may name this data frame anything you want.

1. Write and run a for-loop that prints the names of the cities in the dataset in the order that they appear in the data.

2. Write and run another for-loop that prints the names of the cities in the order of their population sizes, starting with the largest. If you want, you can write an additional line (or lines) of code before the loop to aid in the procedure. However, you should not do anything to "manually" solve the problem (i.e. the code should still work if the dataset contained completely different cities).

   Hint: The `order()` function may be useful. Remember that you can learn how to use and familiarize yourself with a new function by looking up its documentation in `R`. Enter `?order` into the `R` console.

3. Create a plot of your choosing that has the cities on the x-axis and the population on the y-axis. You are encouraged to use the `ggplot2` package for this, but it is not required.

Part 3

Find the data file named `data_health_synth_small.csv` on bCourses. This is the same dataset you used on the first problem set. Read the data into `R` as a data frame named `hdat`.

1. Remove all observations that have an `NA` for any variable.

2. Create a histogram for the cost variable. You are encouraged to use the `ggplot2` package of this, but it is not required.

3. Create a scatterplot that compare the cost and bps_mean variables, with bps_mean on the x-axis. You are encouraged to use the `ggplot2` package of this, but it is not required.

4. Create a vector that is comprised of a random sample from the cost variable, where the sampling is performed **with replacement** and the sampled vector is the same length as the cost variable. Name that sampled vector `cost_samp`.

5. Compute and compare the means of the original cost variable and `cost_samp`. Comment on whether the two values the exact same, very similar, or very different, as well as why the results make sense.

6. Write and execute a for-loop that repeats the sampling procedure in 3.4 and stores the mean value of the sampled vector each time. Your loop should have 1000 iterations. This means the sampling procedure will occur 1000 times, you will compute the mean of the sampled vector 1000 times, and you will store all 1000 means.

   Set the seed (i.e. `set.seed(12345)`) before the loop to enable reproducibility.

   You may store the means however you wish. A straightforward way would be to create an empty vector or list in advance, and have that vector/list get populated with the results as the loop executes.

7. You should now have stored 1000 means. Compute the standard deviation of those values (i.e. simply use the `sd()` function.

8. Now write a function that performs the procedures in 3.6 and 3.7, but could be applied to any numeric vector of data (i.e. so that we

could plug in the data for some other variable other than cost). The function should be named `my_sampsd_function`. It should have a single argument, the input for which is vector of numeric data. And it should have a single output, which is the standard deviation as computed in 3.7. That is, the code you need to write for your function will look like the following:

```
my_sampsd_function <- function(inputvec){
    ...
    ...
    ...
    return(myoutput)
}
```

Hint: You could do this by copying your code from 3.6 and 3.7, wrapping it in your function, and then making the appropriate minor modifications to achieve the desired functionality.

9. Verify that your function is working by running it with the cost variable as the input (i.e. run `my_sampsd_function(hdat$cost)` and present the answer). Be sure to set the seed (i.e. `set.seed(12345)`) right before running the function.

   The answer should be exactly identical to your answer from 3.7 if you have been properly using the `set.seed()` functionality, or almost identical if there is something wrong with your usage of `set.seed`.

10. Now run your function with hdat$bps_mean as the input. If your function is working properly, the answer should be different from 3.9. Again, set the seed (i.e. `set.seed(12345)`) right before running the function.