CS100: Studio 4

Programming Practice and Measures of Dispersion

October 2 and 3, 2019

Instructions

During today’s studio, you will be practicing with some programming fundamentals, and reviewing measures of dispersion. Please write all of your code, and answers to the questions, in an R markdown document.

Upon completion of all tasks, a TA will give you credit for today’s studio. If you do not manage to complete all the assigned work during the studio period, do not worry. You can continue to work on this assignment until the following Wednesday at 6 PM. Come by TA hours any time before then to show us your completed work and get credit for today’s studio.

Objectives

By the end of this studio, you will be able to: * program with for loops * program with functions * merge datasets in R

By the end of this studio, you will better understand measures of dispersion, including variance, covariance, and correlation.

Part 1: FizzBuzz

The FizzBuzz problem is a short programming task, often asked during software engineering interviews. Here is one variant:

Write a function called FizzBuzz that takes as input a number, and prints “Fizz!” if the number is a multiple of 3, “Buzz!” if the number is a multiple of 5, and “FIZZ BUZZ!!!” if the number is a multiple of both 3 and 5. If the number is neither a multiple of 3 nor a multiple of 5, it should just print a sad face.

Here are some sample inputs and their corresponding outputs:

2: :(

3: Fizz!

5: Buzz!

6: Fizz!

15: FIZZ BUZZ!!!

19: :(
The first task in today’s studio is to solve the Fizzbuzz problem.

To write Fizzbuzz, you will need the \texttt{mod} operator, or \%\% in R. Modular arithmetic gives the remainder when dividing some number \(x\) by some number \(y\). For example, \(7 \%\% 3\), read as “7 mod 3”, gives 1 because 7 divided by 3 is 2 with a remainder of 1. Solve the following examples by hand, and then run them in R to verify your understanding.

\[
\begin{array}{l}
8 \%\% 6 \\
7 \%\% 4 \\
10 \%\% 5 \\
3 \%\% 3 \\
\end{array}
\]

Write a function \texttt{fizzbuzz} that takes as input a number \(n\) and prints out the number as well as Fizz!, Buzz!, FIZZ BUZZ!!!, or :, as required. Note that printing both Fizz! and Buzz! if you encounter a multiple of both 3 and 5 instead of FIZZ BUZZ!!! is incorrect.

**Hint:** You may find \texttt{if else} statements helpful!

**Another hint:** Recall the syntax for functions:

\[
\text{function \_name} <- \text{function (argument) } \{
\text{function body}
\}
\]

**Yet another hint:** Feel free to use the \texttt{cat} function, which print a combination of variables and strings, with spaces in between them. For example,

```r
hello <- "Hello"
world <- "world"
year <- 2019
cat(hello, world, year, "!")
```

This code will display Hello world 2019 !.

Once you have a working version of \texttt{FizzBuzz}, write a \texttt{for} loop that calls your \texttt{fizzbuzz} function on a vector comprising your favorite numbers.

Once your \texttt{for} loop is working, please call over a TA to review your work. This is an important checkpoint; we want to make sure that you understand these programming concepts, because you will need them for future assignments.

**Part 2: Measures of Dispersion**

In this part of studio, you’ll be revisiting the international development data that we used in the first Friday section. To refresh your memory about the data sets, please take another look at the Section 1 slides.
First, let’s read in the data. Specifically, insert the following code chunk into your RMarkdown file. Be careful copying and pasting, because the quotation marks may give you trouble!

```
\{r setup, include = FALSE\}
hd1 <- read.csv("http://cs.brown.edu/courses/cs100/lectures/scripts/section1/2014hdi.csv")
gini <- read.csv("http://cs.brown.edu/courses/cs100/lectures/scripts/section1/2014gini.csv")
worlddev <- read.csv("http://cs.brown.edu/courses/cs100/lectures/scripts/section1/2014worlddev.csv")
continents <- read.csv("http://cs.brown.edu/courses/cs100/lectures/scripts/section1/continents.csv")
```

In order to effectively analyze the data, we are again going to merge these four data sets together. In Section 1, we joined the data in Google Sheets using VLOOKUP. To do the same in R, we use the `merge` function. You may recall from lecture that the syntax for `merge` looks something like the following:

```
combined <- merge(first_data_frame, second_data_frame, by = "variable")
```

The `by` argument tells R how to join the data in the two data frames together. For example, if we join `hdi` and `gini` on the `Country` column, each country’s `hdi` data is joined with that same country’s `gini` data in the resulting data frame.

Go ahead and use `merge` to join these two data frames.

Once again, we would like to join all four data frames. However, `continents` and `worlddev` do not have a `Country` column. Instead, `continents` has a column called “name”, and `worlddev` has a column called `Country.Name`. Not to worry! R’s `merge` function allows you to specify which columns in the data frames that you want to join by, using the arguments `by.x` and `by.y`. So, to join `continents` and `worlddev`, you can write the following:

```
combined <- merge(continents, worlddev, by.x = "name", by.y = "Country.Name")
```

Please merge all four of the data frames together into a new one called `merged_data`.

For further information on the `merge` command, feel free to take a look at the R documentation here.

Joining data is a very important skill, and one that data scientists encounter regularly. Please call a TA over make sure you’ve merged the data correctly.

**Standard Deviation**

In addition to measures of central tendency (mean, median, and mode), measures of variability are also helpful when trying to make sense of data. Let’s take a look at some of the common measures of variability!

Standard deviation is one such measure. Given a set of

\[ n \]
data points
\[ X = \{ x_i \mid i = 1, \ldots, n \} \],
the formula for (population) standard deviation 

\[ s_X \]
is:

\[ s_X = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (x_i - \mu)^2}, \]

where

\[ \mu \]
is the (population) mean.

Looking at this formula, we see that what is being computed first here is the average of the squared differences between each data point and the mean. This quantity is called \textbf{variance}. After taking square roots, we arrive at the \textbf{standard deviation}. The standard deviation can be easier to interpret than the variance, since it is of the same magnitude as the data.

To be sure you fully understand the formula for standard deviation, let’s first compute the standard deviation of a few variables in a spreadsheet. Refer back to your spreadsheet of merged international development data from Section 1; or if you can’t easily access that, you can find a clean version here.

Here are the necessary steps:

1. Compute the average of your measurements
2. Calculate the distance between each measurement and this average
3. Square these distances
4. Compute their average
5. At this point, you would have calculated the variance. If you are interested in the standard deviation, rather than the variance, take the square root.

Follow the steps above and calculate the standard deviation of one or two variables.

Once you have calculated the standard deviation of a few variables in your spreadsheet, we’d like for you to do the same thing in R! First, you might want to read in our version of the merged data. Otherwise, you will likely encounter a bunch of NAs and missing values, which R will complain about when you try to apply arithmetic operations to them. Next week, you’ll be learning how to clean data yourselves. YAY!

Here’s one way to load our data set:

```r
merged_data <- read.csv("http://cs.brown.edu/courses/cs100/studios/data/4/merged_data.csv")
```
After loading the data, use the \texttt{sd} function to calculate the standard deviation of one or two variables. If you don’t know how to use the \texttt{sd} function yet (and there is no reason why you should!), type \texttt{help(sd)}, and then consult the information that pops up in the Help window.

Does this formula yield the same result as your manual calculation? \textbf{Hint:} It should be close, but not identical.

The reason it is not identical is, the formula above is the formula for \textit{population} standard deviation, but the \texttt{sd} formula calculates something called \textit{sample} standard deviation. The difference, as the names suggest, is whether you are calculating the standard deviation of an entire population (as we are here; all countries are represented), or of only a representative sample. We do not actually have measurements for \textit{all} individuals—we are missing fertility rates for a few countries—but we have enough measurements, that population standard deviation still makes sense.

Finally, write a \texttt{for} loop that iterates over a variable’s measurements and calculates the standard deviation of that variable.

\textbf{Covariance}

\textbf{Covariance} is a measure of how two variables vary with one another. It is calculated as follows: Given two sets of

\[
\begin{align*}
X &= \{x_i \mid i = 1, \ldots, n\} \\
Y &= \{y_i \mid i = 1, \ldots, n\}
\end{align*}
\]

\[
\text{cov}_{X,Y} = \frac{1}{n} \sum_{i=1}^{n} (x_i - \mu_x) \cdot (y_i - \mu_y)
\]

where \(\mu_x\) is the mean of \(X\), and \(\mu_y\) is the mean of \(Y\), respectively. Note that the formula for covariance generalizes that of variance, by which we mean that

\[
\text{cov}_{X,X}
\]
is precisely the variance of $X$.

**Correlation** is covariance, normalized. In other words, correlation is covariance divided by the product of the standard deviation of $X$ and the standard deviation of $Y$. Correlation always falls between $-1$ and $1$ (inclusive), so it is easier than covariance to interpret. In particular, it is much easier to compare different correlations across different pairs of variables.

Use the `cov` function to calculate the covariance between “Life Expectancy” and “Gross National Income”. This result would be difficult to interpret relative to the covariance between two other variables, especially when they differ in magnitude.

Next use the `cor` function to calculate the correlation between these same two variables. This gives the normalized strength of the relationship between the two variables.

Look at the correlations between “Life Expectancy” and some other variables, and “Gross National Income” and some other variables. Report whether the results are as you expected.

**Quartiles**

Another way to investigate variability in data is to look at generalizations of the median. The median is a value that splits a data set in half; half the data lie above the median, and half lie below. But other points of interest might be the point below which only 25% of the data fall, or the point above which only 25% of the data fall. These particular values are called **quartiles**.

There are 120 countries in our data set. One quarter of 120 is 30.

Sort the data in descending order by “GNI per capita”. Which country is ranked 30th in this list, and what is its income? Similarly, which countries are ranked 60th and 90th, and what are their incomes? And what are the minimum and maximum incomes?

Pick another variable of your choice, and repeat this exercise. Find the minimum, the maximum, and the values at the 25th, 50th, and 75th percentiles.
You might be wondering if it is really necessary to sort and then count to find quartiles. Well, it isn’t. R provides a `quantile` function, which takes in a column of a dataframe (or a list) and returns the different quartiles.

Using “GNI per capita” again, find all five quartile points using the `quantile` function. Do the values calculated by `quantile` match the values you calculated a few minutes ago? Why not? (Hint: It is because what we told you to do was a little off. What was wrong with our instructions?)

Compute the Interquartile Range (IQR) of “GNI per capita” by subtracting the 1st quartile value from the 3rd quartile value and multiplying by 1.5. Add the IQR to the median. The IQR rule of thumb would deem any values above this sum to be outliers. Likewise, it would also deem as outliers any values below the median less the IQR. Do you think this rule of thumb is appropriate for these measurements? Eyeballing the data, which values might you deem as outliers? Qatar probably? Any others?

**End of Studio**

When you are done please call over a TA to review your work, and check you off for this studio. If you do not finish within the two hour studio period, remember to come to TA office hours to get checked off.