Data Science for Public Policy

Aaron R. Williams - Georgetown University

PPOL 670 | Assignment 01
R Basics

Due Date: Tuesday, January 25th at 11:59 PM

Deliverable: A single, well-documented `.R` script submitted to Canvas. The `.R` script should be called `assignment01_<NetID>.R`. Each exercise should be clearly labeled and the grader should be able to run each segment of R code.

Rubric

- Each exercise is clearly labeled with a comment. Use `#` to create a comments in your R script.
- Each answer runs successfully and provides a correct answer.
- Attempted exercises will receive more than zero points. Unattempted exercises will receive zero points.

Points: 5 points

Learning and data science are both collaborative practices. We encourage you to discuss class topics and homework topics with each other. However, the work you submit must be your own. A student should never see another student's code or receive explicit coding instructions for a homework problem. Please attend office hours or contact one of the instructors if you need help or clarification.

Plagiarism on homework or projects will be dealt with to the full extent allowed by Georgetown policy (see http://honorcouncil.georgetown.edu).

This assignment covers R basics from lecture 1. It should not require any external data. Question 09 is the only question that uses an external R package. The assignment should be completed entirely in R.

Setup

1. Create a new folder on your computer called `assignment01`.
2. Open R Studio.
3. In the top right, click "Project: (None)".
4. Click "New Project".
5. Create a new project in the existing directory `assignment01`.
6. In R Studio, open a `.R` script and save it at `assignment01/assignment01_<NetID>.R`.
7. Add your name and GUID as comments at the top of the .R script.

Exercise 01 (0.5 point)

1. Assign the number `4` to `a`.
2. Assign the number `5` to `b`.
3. Multiply `a` and `b` and assign the result to `c`. Note that the result of the multiplication is saved to the global environment but not printed to the Console.
4. On the fourth line of your code for this exercise, include `c` to print the result of your calculation.

Exercise 02 (0.5 point)

1. Assign the vector `c(1, NA, 20, 45)` to `ex2_vector`.
2. Use the `mean` function on `ex2_vector`. Is the answer surprising?
3. Submit `?mean` in the console to read the documentation for `mean()`.
4. Finally, calculate the mean with the argument that ignores missing values.

Exercise 03 (0.5 point)

`:` can be used to create a sequence of integers. For example, `1:10` creates `1 2 3 4 5 6 7 8 9 10`.

R is vectorized, which means operations can be performed on every element in a vector at the same time. For example, `1:4 + 2` returns `3 4 5 6`.

1. Assign a sequence of integers from 1,432 to -546 to `integer_sequence`.
2. Multiply every element of `integer_sequence` by 2, but do not assign the result.
3. In one line of code, square every element of `integer_sequence` and then take the square root of every element that results from that operation (resulting, in the end, in the absolute value of `integer_sequence`). Note: there is a square root function, or you can raise the sequence to the power of $\frac{1}{2}$.

Note, steps 2., and both operations in step 3. should be applied to the original integer_sequence.

Exercise 04 (0.5 point)

Logical tests are useful in data science. R returns Booleans (TRUE or FALSE) for logical tests. Logical tests are also vectorized.

We can compare each value in a vector to one value. For example:

```
1:10 < 5
```

```
##  [1]  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
```

We can also compare the i^{th} element from one vector to the i^{th} element from another vector.

```
1:9 == 9:1
```

```
## [1] FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
```

Sometimes a vector of Booleans isn't particularly helpful and we simply want a count of `TRUE`s and `FALSE`s. We can count Booleans with `table()`.

```r
animals1 <- c("cat", "dog", "pig")
animals2 <- c("cow", "dog", "pig")

table(animals1 == animals2)
```

```
##
## FALSE  TRUE
##     1     2
```

We can also perform summary operations on Booleans, where `TRUE` equals one and `FALSE` equals zero.

```r
# count of TRUEs
sum(animals1 == animals2)
```

```
## [1] 2
```

```r
# proportion of TRUEs
mean(animals1 == animals2)
```

```
## [1] 0.6666667
```

1. Create a vector of integers from 1 to 100 and assign it `ex4_vector`.
2. Pick your favorite integer from 1 to 100 and assign it to `favorite_number`.
3. Calculate the number of elements in `ex4_vector` that are
   a. Equal to `favorite_number` (==)
   b. Greater than `favorite_number` (>)
   c. Greater than or equal to `favorite_number` (>=)
   d. Less than `favorite_number` (<)
   e. Less than or equal to `favorite_number` (<=)
4. Pick your three favorite integers from 1 to 100 and assign them to a vector called `favorite_numbers`.
5. Calculate the number of elements in `ex4_vector` that are in `favorite_numbers` using `%in%`.

replacement就是在原vector中取样的时候是否放回。即使是样本的size < length(vector) 的时候，如果是可放回的情况（replacement = TRUE），那也可能出现抽样的结果 sample() 的答案里有一样的元素。的情况。

Exercise 05 (0.5 point)

This exercise will demonstrate how to calculate the number of missing values in a vector.

`sample()` can be used to randomly sample elements from a vector without replacement. The argument `replace = TRUE` can be used to sample with replacement, which is necessary if `size` exceeds `length(x)`.

When using pseudo-random processes in R like `sample()`, it is important to set the seed using `set.seed()` so results are reproducible.

1. Copy the following code to your `.R` and run it:

```r
universities <- c("Georgetown University",
                  "American University",
                  "George Washington University",
                  NA)

set.seed(20200112)
universities_sample <- sample(x = universities, size = 10000, replace = TRUE)
```

2. Use `is.na()` to return a vector of Booleans where TRUE indicates an element in `universities_sample` is NA and false if where an element is not NA.
3. On a new line, wrap the code from step 2. in `sum()` or `table()` to count the number of missing values.
4. On a new line, wrap the code from step 2. in `mean()` to count the proportion of values that are missing.

Exercise 06 (0.5 point)

R contains a bevy of functions for calculating summary statistics on numeric vectors. `rivers` is a built-in vector in R of the lengths of major North American rivers in miles. Submit `rivers` in the console to see the vector. Submit `?rivers` in the console to see more information about `rivers`.

For the following questions, find the correct function for each summary statistic. You can use a search engine like Google and you can search key terms in R using `??`. For example, `??median` will find functions related to median. Most times, the function name should be intuitive.

1. Find the total length of major rivers in North America.
2. Find the average length of major rivers in North America.
3. Find the median major river length in North America.
4. Find the variance of major river lengths in North America.
5. Find the standard deviation of major river lengths in North America. (use the standard deviation function instead of finding the square root of the variance)
6. Find the minimum major river length in North America.
7. Find the length of the `rivers` vector.
8. Use `quantile()` to find the 25th percentile river length.
9. Use `quantile()` to find the 10th, 20th, and 30th percentile river lengths in one function call. Use `?quantile` to see examples.
10. Use `summary()` to find the six-number summary of `rivers`.

Exercise 07 (0.5 point)

Vectors hold a series of values that are all of the same type - like characters or numeric values. Sometimes, those types can be explicitly converted to other types using `as.*()` functions.

```
character_vector <- c("1", "2", "3", "4", "5")
numeric_vector <- c(1, 2, 3, 4, 5)
```

1. Copy `character_vector` and `numeric_vector` to your script.
2. Convert `character_vector` to a numeric vector with `as.numeric()` and test the equivalence of each element to `numeric_vector` using the `%in%` operator. Your code should return five `TRUE`s.
3. Convert `numeric_vector` to a character vector with `as.character()` and test the equivalence of each element to `character_vector` using the `%in%` operator. Your code should return five `TRUE`s.
4. Try comparing `character_vector` and `numeric_vector` without converting the types? Is the result surprising?

Exercise 08 (0.5 point)

1. Create a vector of the decimals 0.44, 0.45, and 0.46 using `c()` or `seq()` and assign it to `decimals`.
2. Write out how you would round these three numbers to one decimal place as a comment.
3. Wrap `decimals` in the `round()` function and include the argument `digits = 1`. What did you expect? How is the result different from your answer in 2.? Include this code in your script.

R "rounds to even". This is slightly different than what most people expect, but rounding to even leads to less biased estimation and results.

Exercise 09 (0.5 point)

CRAN contains more than 12,000 packages that can be easily installed and used in R. Even more packages are available for installation from sources like GitHub (with less quality control). In practice, a handful of common packages are commonly used. The tidyverse is a package of common data science packages that are widely used. The power of the tidyverse, which began being developed around 2007, is one reason R has grown in popularity over the last decade despite R existing for almost thirty years.

1. Install the tidyverse using `install.packages()`. Even if the tidyverse is already installed, `install.packages()` will check for and install updates.
2. Load the tidyverse using `library()`. Packages are typically loaded at the top of R scripts. This way, collaborators know which packages are used in a script. *Please keep `library()` under Exercise 09 in this case.*

If the tidyverse loads correctly, you should see output in the Console similar to the following:

```
Loading required package: tidyverse
  Attaching packages                                           tidyverse 1.2.1
 ggplot2 3.2.1      purrr   0.3.3
 tibble  2.1.3      dplyr   0.8.3
 tidyr   1.0.0      stringr 1.4.0
 readr   1.3.1      forcats 0.4.0
  Conflicts                                          tidyverse_conflicts()
 dplyr::filter() masks stats::filter()
 dplyr::lag()    masks stats::lag()
```

Exercise 10 (0.5 point)

A contractor has a series of pipes, each one foot longer than than the last, ranging from one foot in length to 10 feet in length with radii of 6 inches. The objective is to calculate the total volume of the ten pipes.

I tried writing code but faltered because it was past my bedtime. The following code has two bugs and while it stores the answer in `answer`, it does not print `answer` to the Console.

```
lengths < - 1:10
radius <- 0.5
answer <- sum(pi * Radius ^ 2 * lengths)
```

1. Copy the above code to your R script.
2. Fix the two bugs.
3. Change the code so it prints the correct answer to the console after assigning the result to `answer`.