Assignment 2: Intro to R

Ellen Bledsoe

2024-02-13

Assignment Description

Purpose

The goal of this assignment is to get comfortable using R and to work with 1-dimensional data in R.

Task

Write R code to successfully answer each question below.

Criteria for Success

- Code is within the provided code chunks or new code chunks are created where necessary
- Code chunks run without errors
- Code chunks have brief comments indicating which code is answering which part of the question
- Code will be assessed as follows:
  - Produces the correct answer using the requested approach: 100%
  - Generally uses the right approach, but a minor mistake results in an incorrect answer: 90%
  - Attempts to solve the problem and makes some progress using the core concept, but returns the wrong answer and does not demonstrate comfort with the core concept: 50%
  - Answer demonstrates a lack of understanding of the core concept: 0%
- Any questions requiring written answers are answered with sufficient detail

Due Date

Jan 29 at midnight MST

Exercises

1. Basic Expressions (10 pts)

In the code chunk below, type out the following calculations.

a. `2 - 10`
b. `3 * 5`
c. `9 / 2`

d. `5 - 3 * 2`
e. `(5 - 3) * 2`
f. `4 ^ 2`
g. `8 / 2 ^ 2`

Run each expression by either clicking the `Run` button in the top-right corner of the editor or press `Ctrl+Enter` (Windows & Linux) or `Command+Enter` (Mac) to run code and print the results in the console.

If no code is highlighted/selected, this will run the line the cursor is on. If you highlighted/selected a block of code it will run that entire group of lines.

You can also run the entire code chunk by clicking the small green arrow on the righthand side of the code chunk.

In your code chunk, be sure to comment the code. Before each expression, add a comment that indicates which exercise it is (e.g., `#1a`, `#1b`).

```
2 - 10
```

```
## [1] -8
```

```
3 * 5
```

```
## [1] 15
```

```
9 / 2
```

```
## [1] 4.5
```

```
5 - 3 * 2
```

```
## [1] -1
```

```
(5 - 3) * 2
```

```
## [1] 4
```

```
4 ^ 2
```

```
## [1] 16
```

```
8 / 2 ^ 2
```

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

2. Basic Variables (10 pts)

Here is a small bit of code that converts a mass in kilograms to a mass in grams and then prints out the resulting value.

```r
mass_kg <- 2.62
mass_g <- mass_kg * 1000
mass_g
```

```
## [1] 2620
```

In the code chunk below, create similar code to convert a mass in pounds to a mass kilograms. Remember to comment your code!

a. Create a variable to store a body mass in pounds. Assign this variable a value of 3.5 (an appropriate mass for a *Sylvilagus audubonii*).
b. Convert the variable from body mass in pounds to body mass in kilograms (by dividing it by 2.2046), and assign it to a new variable.
c. Print the value of the new variable to the screen by typing the name of the variable and then running that line of code.

```r
mass_lbs <- 3.5
mass_kg <- mass_lbs / 2.2046
mass_kg
```

```
## [1] 1.58759
```

3. More Variables (10 pts)

Calculate a total biomass in grams for 3 white-throated woodrats (*Neotoma albigula*) and then convert it to kilograms.

The total biomass is three times the average size of a single individual. An average individual weighs 250 grams.

a. Create a variable `grams` and assign it the mass of a single *Neotoma albigula* (250).
b. Create a variable `number` and assign it the number of individuals (3).
c. Create a variable `biomass` and assign it a value by multiplying the `grams` and `number` variables together.
d. Convert the value of `biomass` into kilograms (there are 1000 grams in a kilogram so divide by 1000) and assign this value to a new variable.
e. Print the final answer to the screen.

```r
grams <- 250
number <- 3
biomass <- grams * number
biomass_kg <- biomass / 1000
print(biomass_kg)
```

```
## [1] 0.75
```

4. Built-in Functions (10 pts)

A built-in function is one that you don't need to install and load a package to use. Some examples include:

- `abs()` returns the absolute value of a number (e.g., `abs(-2)`)

- `round()`, rounds a number (the first argument) to a given number of decimal places (the second argument) (e.g., `round(12.1123, 2)`)
- `sqrt()`, takes the square root of a number (e.g., `sqrt(4)`)
- `tolower()`, makes a string all lower case (e.g., `tolower("HELLO")`)
- `toupper()`, makes a string all upper case (e.g., `toupper("hello")`)

Use these built-in functions to print the following items:

a. The absolute value of -15.5.
b. 4.483847 rounded to one decimal place.
c. 3.8 rounded to the nearest integer. *(You don't have to specify the number of decimal places in this case if you don't want to, because* ***`round()`*** *will default to using* `0` *if the second argument is not provided. Look at* ***`help(round)`*** *or* ***`?round`*** *to see how this is indicated.)*
d. `"species"` in all capital letters.
e. `"SPECIES"` in all lower case letters.
f. Assign the value of the square root of 2.6 to a variable. Then round the variable you've created to 2 decimal places and assign it to another variable. Print out the rounded value.

```
abs(-15.5)
```

```
## [1] 15.5
```

```
round(4.483847, 1)
```

```
## [1] 4.5
```

```
round(3.8, 0)
```

```
## [1] 4
```

```
toupper("species")
```

```
## [1] "SPECIES"
```

```
tolower("SPECIES")
```

```
## [1] "species"
```

```
value <- sqrt(2.6)
value <- round(value, 2)
value
```

```
## [1] 1.61
```

```
round(sqrt(2.6), 2)
```

```
## [1] 1.61
```

Optional Challenge: Do the same thing as task 6 (immediately above), but instead of creating the intermediate variable, perform both the square root and the round on a single line by putting the `sqrt()` call inside the `round()` call. Putting functions inside of other functions is called *nesting*.

5. Modify the Code (15 pts)

The following code estimates the total net primary productivity (NPP) per day for two sites. It does this by multiplying the grams of carbon produced in a single square meter per day by the total area of the site. It then prints the daily NPP for each site.

```
# grams of carbon per sq. meter
site1_g_carbon_m2_day <- 5
site2_g_carbon_m2_day <- 2.3

# number of sq. meters
site1_area_m2 <- 200
site2_area_m2 <- 450

# daily NPP per site
site1_npp_day <- site1_g_carbon_m2_day * site1_area_m2
site2_npp_day <- site2_g_carbon_m2_day * site2_area_m2

site1_npp_day
```

```
## [1] 1000
```

```
site2_npp_day
```

```
## [1] 1035
```

Be sure to run the code chunk above in order to answer the following questions.

At the end of this question, add a new code chunk (`Ctrl + Shift + I` or `Cmd + Shift + I`). In that code chunk, add additional lines of code come accomplish the following:

a. The sum of the total daily NPP for the two sites.
b. The difference between the daily NPP for the two sites. We only want an absolute difference, so use abs() function to make sure the number is positive.
c. The total NPP over a year for the two sites combined (the sum of the total daily NPP values from part (1) multiplied by 365).

Add code chunk here:

```
# grams of carbon per sq. meter
site1_g_carbon_m2_day <- 5
site2_g_carbon_m2_day <- 2.3

# number of sq. meters
site1_area_m2 <- 200
site2_area_m2 <- 450

# daily NPP per site
site1_npp_day <- site1_g_carbon_m2_day * site1_area_m2
site2_npp_day <- site2_g_carbon_m2_day * site2_area_m2

# a.  The sum of the total daily NPP for the two sites combined.
total_npp_day <- site1_npp_day + site2_npp_day
total_npp_day
```

```
## [1] 2035
```

```
# b.  The difference between the total daily NPP for the two sites.
daily_diff <- abs(site1_npp_day - site2_npp_day)
daily_diff
```

```
## [1] 35
```

```
# c.  The total NPP over a year for the two sites combined.
total_npp_year <- total_npp_day * 365
total_npp_year
```

```
## [1] 742775
```

6. Basic Vectors (15 pts)

Run the following code chunk to use for answering Question 6.

```
numbers <- c(5, 2, 26, 8, 16)
```

Now, add a code chunk and write code that accomplishes the following:

a. The number of items in the `numbers` vector (using the `length` function)
b. The third item in the numbers vector (using `[]`)
c. The smallest number in the `numbers` vector (using the `min` function)
d. The largest number in the `numbers` vector (using the `max` function)
e. The average of the numbers in the numbers vector (using the `mean` function)
f. The first, second and third `numbers` in the numbers vector (using `[]`)
g. The sum of the values in the `numbers` vector (using the `sum` function)

Add code chunk here:

```
numbers <- c(5, 2, 26, 8, 16)

length(numbers)
```

```
## [1] 5
```

```
numbers[3]
```

```
## [1] 26
```

```
min(numbers)
```

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

```
max(numbers)
```

```
## [1] 26
```

```r
mean(numbers)
```

```
## [1] 11.4
```

```r
numbers[1:3]
```

```
## [1]  5  2 26
```

```r
sum(numbers)
```

```
## [1] 57
```

7. Nulls in Vectors (15 pts)

Run the following code chunk to use for answering Question 7.

Hint: remember ***`na.rm = TRUE`****?*

```r
numbers <- c(7, 6, 22, 5, NA, 42)
```

In a new code chunk below, calculate the following:

a. The smallest number in the `numbers` vector
b. The largest number in the `numbers` vector
c. The average of the numbers in the `numbers`
d. The sum of the values in the `numbers` vector

Add code chunk here:

```r
numbers <- c(7, 6, 22, 5, NA, 42)

min(numbers, na.rm = TRUE)
```

```
## [1] 5
```

```r
max(numbers, na.rm = TRUE)
```

```
## [1] 42
```

```r
mean(numbers, na.rm = TRUE)
```

```
## [1] 16.4
```

```r
sum(numbers, na.rm = TRUE)
```

```
## [1] 82
```

8. Shrub Volume Vectors (15 pts)

You have data on the length, width, and height of 10 individuals of the yew (*Taxus baccata*) stored in the following vectors:

```
length <- c(2.2, 2.1, 2.7, 3.0, 3.1, 2.5, 1.9, 1.1, 3.5, 2.9)
width <- c(1.3, 2.2, 1.5, 4.5, 3.1, NA, 1.8, 0.5, 2.0, 2.7)
height <- c(9.6, 7.6, 2.2, 1.5, 4.0, 3.0, 4.5, 2.3, 7.5, 3.2)
```

Add a code chunk and determine the following:

a. The volume of each shrub (length × width × height). *Storing this in a variable will make some of the next problems easier.*
b. The sum of the volume of all of the shrubs (using the `sum` function).
c. A vector of the height of shrubs with lengths > 2.5.
d. A vector of the height of shrubs with heights > 5.
e. A vector of the heights of the first 5 shrubs (using `[]`).
f. A vector of the volumes of the first 3 shrubs (using `[]`).

Add code chunk here:

```
#1
volume <- length * width * height
volume
```

```
##  [1] 27.456 35.112  8.910 20.250 38.440     NA 15.390  1.265 52.500 25.056
```

```
#2
sum(volume, na.rm = TRUE)
```

```
## [1] 224.379
```

```
#3

height[length > 2.5]
```

```
## [1] 2.2 1.5 4.0 7.5 3.2
```

```
#4
height[height > 5]
```

```
## [1] 9.6 7.6 7.5
```

```
#5
height[1:5]
```

```
## [1] 9.6 7.6 2.2 1.5 4.0
```

```
#6
volume[1:3]
```

```
## [1] 27.456 35.112  8.910
```

Optional Challenge: A vector of the volumes of the last 5 shrubs with the code written so that it will return the last 5 values regardless of the length of the vector (i.e., it will give the last 5 values if there are 10, 20, or 50 individuals).

Add *optional* code chunk here:

```
num_shrubs <- length(volume)
volume[(num_shrubs - 4):num_shrubs]
```

```
## [1]     NA 15.390  1.265 52.500 25.056
```

9. Variable Names (*optional*)

In Q3, we used the variable names `grams`, `number`, and `biomass` to describe the individual mass, number of individuals, and total biomass of some white-throated woodrats. But are these variable names the best choice?

If we came back to the code for this assignment in two weeks would we be able to remember what these variables were referring to and therefore what was going on in the code? The variable name `biomass` is also kind of long. If we had to type it many times it would be faster just to type `b`. We could also use really descriptive alternatives like `individual_mass_in_grams`. Or we would compromise and abbreviate this or leave out some of the words to make it shorter (e.g., `indiv_mass_g`).

Think about good variable names and then create a new version of this code with the variables renamed to be most useful. Make sure your code still runs properly with the name changes.

Turning in Your Assignment

Follow these steps to successfully turn in your assignment on D2L.

1. Click the `Knit` button up near the top of this document. This should produce a PDF file that shows up in the `Files` panel on the bottom-right of your screen. (*If you can't get your file to Knit to PDF, you can submit the .Rmd file instead.*)
2. Click the empty box to the left of the PDF file.
3. Click on the blue gear near the top of the `Files` panel and choose Export.
4. Put your last name at the front of the file name when prompted, then click the Download button. The PDF file of your assignment is now in your "Downloads" folder on your device.
5. Head over to D2L and navigate to the correct Assignment dropbox. Submit the PDF file that you just downloaded.