BIOS 42411: Lab 1

Kevin Buck

1/18/2023

In Class

To get you up to speed in basic R programming, we are going to use an interactive teaching program within R called **swirl**. After you install and load the package **swirl** you will be prompted to write code in the console interactively. Today, you will only be doing the short course titled "R programming > Basic Building Blocks", but note there are many other swirl courses (https://swirlstats.com/) that you can take to get you up to speed on more advanced topics.

1. If you haven't already, install the package **swirl** by using the function **install.packages()** in your console. Make sure to put the package name in quotes to install the package.
2. Load the package by using the function **library()** in your console.
3. You may be prompted with a message in your console that says "Welcome to swirl". If not, call the function **swirl()** with no arguments (*i.e.* empty parentheses). Follow the prompts and choose the course "1: R programming" and "1: Basic Building Blocks" when asked. Work through this course during tutorial period.

Assignment Instructions

You are going to apply what you learned in **swirl** today, but instead of working in the console, you are going to write and execute code in this RMarkdown document (.Rmd file in Sakai > Resources > Lab 1) which can be saved. RMarkdown documents allow you to save text, code, and graphs all in one document that is edited whenever you edit your code.

All code (and comments) is placed within the gray sections of the document and text can be put in the white part of the document.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
x <- 1+1
x
```

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

where word after "r" within the brackets in the R Markdown document is just a title for that particular chunk of code: it can be whatever you want, but there should be unique names for each chunk. You can run lines of code within the gray sections using CMD+ENTER (Mac) or CTRL+ENTER (Windows). Or you can run the current chunk of code by pushing the green arrow on the top right of the chunk of code you are working on. When you **Knit** an R Markdown document, it automatically runs all of your code.

Example Question

Megan wants to calculate the number of seconds in a year.

a. Calculate the number of seconds in a year and store this as an object.
b. How many seconds are in a year?

```r
# Multiply the # of days/year by # hours/day by # min/hour by # sec/min
seconds <- 365 * 24 * 60 * 60
# Print result
seconds
```

```
## [1] 31536000
```

Answer to Example Question

There are 3.1536×10^7 seconds in a year. You could also write this out by hand as: There are 31536000 seconds in a year.

Question 1: Blue whales

The longest blue whale ever measured was approximately 98 feet long.

a. Save this measurement as an object with the name of your choice (see the section "Names" **here** on how to choose a good name for an object).
b. Using that object, convert this measurement to meters (1ft = 0.3048m) and store it as a new object. **How many meters long was the whale?**

```r
#storing the length of the longest ever whale in feet
longestWhale <- 98
#converting longeset whale measurement to meters
longestWhaleMeters <- longestWhale * 0.3048
```

Answer to Q1

The longest whale ever measured was 29.8704 meters long. ## Question 2: Campus trees Imagine you wanted to know how big a group of trees on campus were. One way that ecologists quantify tree size is by measuring the "diameter at breast height" using a special measuring tape that is wrapped around a tree. Using the measuring tape, you measure five trees and measure diameters of 10.2cm, 15.3cm, 14.5cm, 13.1cm, and 10.4cm.

a. Store this information as an object using vector notation (hint: use the c() function).
b. Calculate the radius of each of the trees and store as a new object. Using this new object, calculate the area of the tree at breast height as if you were to take a slice through the tree (reminder: $A = \pi r^2$). In R π can be written as the letters "pi" to get the numeric value.
c. **What are the areas of the slices for each tree?**

```r
#storing a vector of tree DBHs in cm
treeDBH <- c(10.2,15.3,14.5,13.1,10.4)
# calculating radius for each tree and storing it in a new object
tree1Radius <- treeDBH[1]/(2*pi)
tree2Radius <- treeDBH[2]/(2*pi)
tree3Radius <- treeDBH[3]/(2*pi)
tree4Radius <- treeDBH[4]/(2*pi)
tree5Radius <- treeDBH[5]/(2*pi)
# calculating area of each tree
tree1Area <- pi*(tree1Radius**2)
tree2Area <- pi*(tree2Radius**2)
tree3Area <- pi*(tree3Radius**2)
tree4Area <- pi*(tree4Radius**2)
tree5Area <- pi*(tree5Radius**2)
```

Answer to Q2

The tree areas are $8.2792401cm^2$, $18.6282903cm^2$, $16.7311634cm^2$, $13.6562899cm^2$, and $8.6070993cm^2$.

Question 3: Matrices

A common way to store data in R is using a 2-dimensional matrix. To create a matrix in R, you need to know the number of rows, the number of columns, and the numbers that will populate each cell of the matrix. Create a matrix in R that looks like the following and store as an object:

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

Hint: use the function **matrix()** and use the help feature **?matrix()** (or Google!) to figure out what arguments are needed to execute the function.

a. What does the argument "byrow" within the function **matrix()** mean?
b. What happens when you add an integer to the matrix?

```
#making a matrix like the above one
myMatrix <- matrix(nrow=2, ncol=2, data=c(1,2,3,4),byrow=T)
#testing adding an integer to the myMatrix
1 + myMatrix
```

```
##      [,1] [,2]
## [1,]    2    3
## [2,]    4    5
```

Answer to Q3

a.) The argument 'byrow' fills in the matrix with the supplied data going left to right by row instead of by column. Each element in the data vector is placed into the matrix left to right with each row containing the number of elements specified in 'ncol'.

b.) Adding an integer to the stored matrix object (here myMatrix) adds that value to each matrix element and outputs a matrix result.