STA 141 Worksheet 1

Richard McCormick

September 5, 2023

Due Date: Thursday, September 14, 2023 before 11:00am.

Instructions

Worksheets must be turned in as a PDF file through Canvas. The worksheet is worth a total of **15 points**, which is 3 percent of your overall grade.

Exercises

Exercise 1 - Assignment and data types

(a) In the `R` programming language we use a `<-` to assign a value to a variable name. For example, the following assigns a value of 141 to a variable called a. You can run this code by clicking on the arrow in the far right corner of the code box.

```
a <- 141
```

You should now see a variable saved to your working environment in the top right window of RStudio. From now on, any time we use the variable a it will be the same as using the value 141 (unless we save over it with a different value).

Any time you want to see the value of a variable you can just type its name into a code block like below and click the arrow to run the block:

```
a
```

```
## [1] 141
```

In the box below, create a variable called x and assign it a numerical value of your choosing. Type x on the second line and then click run to make sure it all worked and the value printed:

```
x <- 69420
x
```

```
## [1] 69420
```

(b) You can check the variable type by using a function called class. The following will print the data type of the variable a that we created above:

```
class(a)
```

```
## [1] "numeric"
```

Check the data type of the variable x here:

```
class(x)
```

```
## [1] "numeric"
```

(c) You can always change the value saved to a variable (if you couldn't it probably would be called a constant not a variable). Change the value of x to 4.0 and check its type here:

```
x <- 4.0
class(x)
```

```
## [1] "numeric"
```

(d) The other main data type in the R programming language is characters (aka strings). These are simply words saved as variables. When we have a table of data we will be able to create categorical variables by ensuring that only certain characters are present in the column of the table. To assign a character to a variable name we do the same thing as above but wrap the value (the word) in quotes. The following will create a variable called NAU with a value of great:

```
NAU <- "great"
NAU
```

```
## [1] "great"
```

We can check the class of the NAU variable using the same class function as we did above.

In the following code block, create a variable called student and assign your name as its value. Check its data type on the following line.

```
student <- "Student"
class(student)
```

```
## [1] "character"
```

Exercise 2 - Dataframes

Before you begin this question make sure you have downloaded the file called `fish.csv` from canvas and saved it into the same folder as this file (preferably the STA141 folder that you've created).

The following will read in a dataset called fish and save it as fishes:

```
fishes <- read.csv("fish.csv")
```

You should be able to see the dataframe saved in the top right window of RStudio with your other variables.

To view a dataframe in a new tab we can double click on the dataset in the top right window.

To see a summary of a dataframe in RStudio we can use the summary function like this:

```
summary(fishes)
```

```
##   ï..Species            Weight          Length1         Length2
##  Length:159         Min.   :   0.0   Min.   : 7.50   Min.   : 8.40
##  Class :character   1st Qu.: 120.0   1st Qu.:19.05   1st Qu.:21.00
##  Mode  :character   Median : 273.0   Median :25.20   Median :27.30
##                     Mean   : 398.3   Mean   :26.25   Mean   :28.42
##                     3rd Qu.: 650.0   3rd Qu.:32.70   3rd Qu.:35.50
##                     Max.   :1650.0   Max.   :59.00   Max.   :63.40
##     Length3          Height           Width
##  Min.   : 8.80   Min.   : 1.728   Min.   :1.048
##  1st Qu.:23.15   1st Qu.: 5.945   1st Qu.:3.386
##  Median :29.40   Median : 7.786   Median :4.248
```

```
##  Mean   :31.23   Mean   : 8.971   Mean   :4.417
##  3rd Qu.:39.65   3rd Qu.:12.366   3rd Qu.:5.585
##  Max.   :68.00   Max.   :18.957   Max.   :8.142
```

You can see that this provides statistics for the numerical columns and shows the class of character columns.

(a) To find the number of rows or columns in a dataframe, we can use the nrow or ncol functions, respectively. The following shows the number of columns in fishes:

```
ncol(fishes)
```

```
## [1] 7
```

Find the number of rows in fishes in the code block below:

```
nrow(fishes)
```

```
## [1] 159
```

(b) Often we don't want to use all of the data in the dataframe. To remove the rows or columns we don't want we use slicing. When you're doing this, you need to remember that you always need to tell R the rows you want to keep followed by the columns you want to keep, separated by a comma.

The following will keep the 5th row and the 3rd column:

```
fishes[5,3]
```

```
## [1] 26.5
```

If we don't include a number before or after the comma then we will keep all of the rows/columns. The following will keep the 5th row and all the columns:

```
fishes[5,]
```

```
##   ï..Species Weight Length1 Length2 Length3 Height Width
## 5      Bream    430    26.5      29      34 12.444 5.134
```

And this will keep all the rows and only the 3rd column:

```
fishes[,3]
```

```
##   [1] 23.2 24.0 23.9 26.3 26.5 26.8 26.8 27.6 27.6 28.5 28.4 28.7 29.1 29.5 29.4
##  [16] 29.4 30.4 30.4 30.9 31.0 31.3 31.4 31.5 31.8 31.9 31.8 32.0 32.7 32.8 33.5
##  [31] 35.0 35.0 36.2 37.4 38.0 12.9 16.5 17.5 18.2 18.6 19.0 19.1 19.4 20.4 20.5
##  [46] 20.5 21.0 21.1 22.0 22.0 22.1 23.6 24.0 25.0 29.5 23.6 24.1 25.6 28.5 33.7
##  [61] 37.3 13.5 14.3 16.3 17.5 18.4 19.0 19.0 19.8 21.2 23.0 24.0  7.5 12.5 13.8
##  [76] 15.0 15.7 16.2 16.8 17.2 17.8 18.2 19.0 19.0 19.0 19.3 20.0 20.0 20.0 20.0
##  [91] 20.0 20.5 20.5 20.7 21.0 21.5 22.0 22.0 22.6 23.0 23.5 25.0 25.2 25.4 25.4
## [106] 25.4 25.9 26.9 27.8 30.5 32.0 32.5 34.0 34.0 34.5 34.6 36.5 36.5 36.6 36.9
## [121] 37.0 37.0 37.1 39.0 39.8 40.1 40.2 41.1 30.0 31.7 32.7 34.8 35.5 36.0 40.0
## [136] 40.0 40.1 42.0 43.2 44.8 48.3 52.0 56.0 56.0 59.0  9.3 10.0 10.1 10.4 10.7
## [151] 10.8 11.3 11.3 11.4 11.5 11.7 12.1 13.2 13.8
```

If we want to keep rows a to b, we can inlcude them separated by a colon. The following will keep rows 3 to 5 and all the columns.

```
fishes[3:5,]
```

```
##   ï..Species Weight Length1 Length2 Length3  Height  Width
```

```
## 3      Bream    340    23.9    26.5    31.1 12.3778 4.6961
## 4      Bream    363    26.3    29.0    33.5 12.7300 4.4555
## 5      Bream    430    26.5    29.0    34.0 12.4440 5.1340
```

In the code block below, slice the dataframe to keep only the first 35 rows and the first 2 columns.

```
fishes[1:35, 1:2]
```

```
##    ï..Species Weight
## 1       Bream    242
## 2       Bream    290
## 3       Bream    340
## 4       Bream    363
## 5       Bream    430
## 6       Bream    450
## 7       Bream    500
## 8       Bream    390
## 9       Bream    450
## 10      Bream    500
## 11      Bream    475
## 12      Bream    500
## 13      Bream    500
## 14      Bream    340
## 15      Bream    600
## 16      Bream    600
## 17      Bream    700
## 18      Bream    700
## 19      Bream    610
## 20      Bream    650
## 21      Bream    575
## 22      Bream    685
## 23      Bream    620
## 24      Bream    680
## 25      Bream    700
## 26      Bream    725
## 27      Bream    720
## 28      Bream    714
## 29      Bream    850
## 30      Bream   1000
## 31      Bream    920
## 32      Bream    955
## 33      Bream    925
## 34      Bream    975
## 35      Bream    950
```

(c) You can also access a column of a dataframe using the $ and its name. For example the following will access the Species column of our dataframe:

```
fishes$Species
```

```
## NULL
```

In the following code block, print out the Weight column of the fishes dataframe.

```
fishes$Weight
```

```
##   [1]  242.0  290.0  340.0  363.0  430.0  450.0  500.0  390.0  450.0  500.0
##  [11]  475.0  500.0  500.0  340.0  600.0  600.0  700.0  700.0  610.0  650.0
##  [21]  575.0  685.0  620.0  680.0  700.0  725.0  720.0  714.0  850.0 1000.0
##  [31]  920.0  955.0  925.0  975.0  950.0   40.0   69.0   78.0   87.0  120.0
##  [41]    0.0  110.0  120.0  150.0  145.0  160.0  140.0  160.0  169.0  161.0
##  [51]  200.0  180.0  290.0  272.0  390.0  270.0  270.0  306.0  540.0  800.0
##  [61] 1000.0   55.0   60.0   90.0  120.0  150.0  140.0  170.0  145.0  200.0
##  [71]  273.0  300.0    5.9   32.0   40.0   51.5   70.0  100.0   78.0   80.0
##  [81]   85.0   85.0  110.0  115.0  125.0  130.0  120.0  120.0  130.0  135.0
##  [91]  110.0  130.0  150.0  145.0  150.0  170.0  225.0  145.0  188.0  180.0
## [101]  197.0  218.0  300.0  260.0  265.0  250.0  250.0  300.0  320.0  514.0
## [111]  556.0  840.0  685.0  700.0  700.0  690.0  900.0  650.0  820.0  850.0
## [121]  900.0 1015.0  820.0 1100.0 1000.0 1100.0 1000.0 1000.0  200.0  300.0
## [131]  300.0  300.0  430.0  345.0  456.0  510.0  540.0  500.0  567.0  770.0
## [141]  950.0 1250.0 1600.0 1550.0 1650.0    6.7    7.5    7.0    9.7    9.8
## [151]    8.7   10.0    9.9    9.8   12.2   13.4   12.2   19.7   19.9
```

NOTE: If you ever want to check the names of the columns in your data you can use the colnames function like this:

```
colnames(fishes)
```

```
## [1] "ï..Species" "Weight"     "Length1"    "Length2"    "Length3"
## [6] "Height"     "Width"
```

Exercise 3 - Data types inside dataframes

There are also a number of datasets that are installed in R. We can load a dataset on Orange Trees by running:

```
data(Orange)
```

You can learn more about the data by running:

```
?Orange
```

```
## starting httpd help server ... done
```

(a) View the dataset using the function we learned above.

```
summary(Orange)
```

```
##  Tree       age         circumference
##  3:7   Min.   : 118.0   Min.   : 30.0
##  1:7   1st Qu.: 484.0   1st Qu.: 65.5
##  5:7   Median :1004.0   Median :115.0
##  2:7   Mean   : 922.1   Mean   :115.9
##  4:7   3rd Qu.:1372.0   3rd Qu.:161.5
##        Max.   :1582.0   Max.   :214.0
```

(b) It looks like the first column should be categorical as the values represent 5 different trees and the age and circumference are numerical. You can check the data type of the column using the class function we

used above. For example, the age column:

```
class(Orange$age)
```

```
## [1] "numeric"
```

Check whether the `Tree` column is a factor (the name for a categorical variable in R):

```
class(Orange$Tree)
```

```
## [1] "ordered" "factor"
```

To check the levels of a factor we have a levels function. We can see all the different levels for Tree by running:

```
levels(Orange$Tree)
```

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

(c) Changing the data type of a variable is called casting. To cast the Tree column to a numeric data type we would run:

```
Orange$Tree <- as.numeric(Orange$Tree)
```

You can see that we assign the new data to the old column name so that it overwrites it.

To cast it back to a factor we would run:

```
Orange$Tree <- as.factor(Orange$Tree)
class(Orange$Tree)
```

```
## [1] "factor"
```

Add a second line to the code above to verify the class is now a factor again.

Exercise 4

Load the Titanic dataset (built-in to R) by running:

```
data(Titanic)
Titanic <- data.frame(Titanic)
```

(a) Write 2 lines of code to show how many rows and columns are in the dataset.

```
nrow(Titanic)
```

```
## [1] 32
```

```
ncol(Titanic)
```

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

(b) Write a line of code that shows the names of all the variables (columns) in the dataset.

```
colnames(Titanic)
```

```
## [1] "Class"    "Sex"      "Age"      "Survived" "Freq"
```

(c) What is the data type of the `Age` column?

```
class(Titanic$Age)
```

```
## [1] "factor"
```

(d) Slice the dataset so that the only remaining rows are adult females (retaining all the columns). Assign this to a variable called `adultfemale`. (Hint: You might want to use the View function to work out which rows to keep)

```
adultfemale = Titanic[Titanic$Sex == c('Female'),]
```

Show that your answer is correct by printing the adultfemale variable.

```
adultfemale
```

```
##    Class    Sex   Age Survived Freq
## 5    1st Female Child       No    0
## 6    2nd Female Child       No    0
## 7    3rd Female Child       No   17
## 8   Crew Female Child       No    0
## 13   1st Female Adult       No    4
## 14   2nd Female Adult       No   13
## 15   3rd Female Adult       No   89
## 16  Crew Female Adult       No    3
## 21   1st Female Child      Yes    1
## 22   2nd Female Child      Yes   13
## 23   3rd Female Child      Yes   14
## 24  Crew Female Child      Yes    0
## 29   1st Female Adult      Yes  140
## 30   2nd Female Adult      Yes   80
## 31   3rd Female Adult      Yes   76
## 32  Crew Female Adult      Yes   20
```