Reading, Writing and Exploring Data

Exercise files

Please download all exercise files for this exercise at http://georgejmount.com/hillsdaler.

Loading our libraries

It's a good idea to call the libraries you're going to use in your script at the beginning of the program.

We will be using two libraries,`tidyverse` and `readxl`. Load them now:

```
library(tidyverse)
library(readxl)
```

If you see an error message, check your packages are installed: **`install.packages()`**.

Reading in Files from R

While R has its own file extension for storing data, it's more common to reading and writing to outside file types. We will cover how to read and write to `.csv`, `.txt` and `.xlsx` files.

we will be importing in the famous `iris` dataset which is popular in data science training.

The data set consists of 50 samples from each of three species of Iris (Iris setosa, Iris virginica and Iris versicolor). Four features were measured from each sample: the length and the width of the sepals and petals, in centimetres.

Working Directories in R

An active R session always has a working directory associated with it. This is where, by default, all files are read and write to.

Check the location of your current directory with the `getwd` function:

```
getwd()
```

getwd

You can set a working directory with `setwd()`.

File paths in R are always enclosed in double quotation marks, and R uses forward slashes, not backslashes, when specifying folder locations. This will take some getting used to, especially for Windows users!

```
setwd("C:/RFiles/")
```

Relative and absolute file paths

There are two ways to refer to outside files in R: using relative or absolute file paths.

Which to use? It all depends on whether you have set a **working directory** with `setwd()` and whether the file you want to read/write to is in that folder.

For example, let's say I have the iris dataset saved in the folder `C:/RFiles/` **and** I have set the working directory to the same folder.

In this case I will use the **relative** file path...

```
setwd("C:/RFiles")
iris<-read_csv("iris.csv")
```

Let's say that instead I had set my working directory to another folder, `"C:/Iris"`.

In this case a relative reference will not work.

```
setwd("C:/Iris")
iris <- read_csv("iris.csv")
```

Instead, we need to use the *absolute* file path because the file is not in the working directory.

For the remainder of this unit, I will save all files to `C:/RFiles` which will also be my working directory:

```
setwd("C:/RFiles")
```

Reading and Writing to CSV Files

To *read* files we will use `read_csv()`

Make sure to assign this function to an object! If not, you will simply have read the file in R without any way to call it!

To *write* files we will use (you guessed it!) `write_csv()`.

This function will take one extra argument in that we first type the object where the dataset is pointed to and then write to the file we want.

However, we will not need to assign it to an object, since we are exporting it anyway.

```
#write the iris dataframe to iriswrite.csv
write_csv(iris,"iriswrite.csv")
```

This file will now appear in our *working directory* (`C:/RFiles`).

Reading and writing to .txt files

This will be very similar to the `read_csv` and `write_csv` functions: We use `read_tsv` and `write_tsv`.

```
#assign the iris.txt dataset to iris variable
iris <- read_tsv("iris.txt")
```

```
#write the iris dataset to Iriswrite.txt
iris <- write_tsv("iriswrite.txt")
```

Reading and Writing Excel Files

We will now read in the Iris.xlsx file using the `read_excel` function:

```
#read file to excel
iris <- read_excel("iris.xlsx")
```

To write files to Excel, we will. . . **write to csv**.

Unfortunately, the `readxl` package only reads files from Excel, as the name suggests. There are packages to write to Excel, but I find that writing to CSV and then opening in Excel is the easiest way.

```
#write to csv and then open in Excel
write_csv(iris,"C:/RFiles/iriswrite2.csv")
```

A sneak peek at your data

If you are used to exploring your data visually, as you might when scrolling down a spreadsheet, you can use `View()`.

```
#open up spreadsheet-type environment
View(iris)
```

Because you're often dealing with such huge datasets, it may not be practical to try viewing the entire table.

head and tail

Instead, use `head` and `tail` to print the first and last X rows, respectively.

```
#view first 3 observations
head(iris,3)
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Spec
## 1          5.1         3.5          1.4         0.2  set
## 2          4.9         3.0          1.4         0.2  set
## 3          4.7         3.2          1.3         0.2  set
```

```
#view last 3 observations
tail(iris,3)
##     Sepal.Length Sepal.Width Petal.Length Petal.Width
## 148          6.5         3.0          5.2         2.0 vi
## 149          6.2         3.4          5.4         2.3 vi
## 150          5.9         3.0          5.1         1.8 vi
```

A brief detour of tibbles

You may have noticed your output for the previous functions began with: `A tibble: 3 x 5`.

What exactly is a tibble?

Put simply, it is a 'modern' data frame used by the libraries of the `tidyverse`. They are nearly identical in concept - tibbles are just programmed to work better for `tidyverse` libraries.

Back to our scheduled programming (heh, heh)...

str

The `str()` function gives us an idea of the structure and type of the object.

```
str(iris)
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.
##  $ Species     : Factor w/ 3 levels "setosa","versicolor
```

names

If you just want a list of the names in the dataset, you can run the `names()` function.

```
names(iris)
## [1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.
## [5] "Species"
```

summary

Finally, you can get summary statistics of your variables with the `summary()` function:

```
summary(iris)
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.
##  Median :5.800   Median :3.000   Median :4.350   Median
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.
##        Species
##  setosa    :50
##  versicolor:50
##  virginica :50
##
##
##
##
```

DRILLS

1. Install and load the `caret` package to your machine.
2. What is your current directory?
3. `R` comes loaded with a dataframe called `mtcars`. 4a. Return the first and last ten rows of this dataframe. 4b. What are the column names of this dataframe? 4c. Get summary statistics on the variables of this dataframe.
4. Write the `mtcars` dataframe to your machine as a .csv file.

Questions?