

ENTMLGY 6702 Entomological Techniques and Data Analysis

Loading data into R

8/29/2023

1 Introduction

We are going to practice a few things using the **Iris** data from:

Anderson. 1935. The irises of the Gaspe Peninsula, *Bulletin of the American Iris Society* (59) 2–5.

All the data files for this tutorial and activity are provided on the course GitHub page: (https://github.com/kiperry/ENT6702_DataAnalysis).

1.1 Why should I learn how to load in different types of data?

We will learn how to load in a few different types of files because (i) many of you might currently or one day use a program/software that returns data in only one file type, (ii) a collaborator might send you data stored in an unfamiliar file type, and/or (iii) third party data (e.g., from the USDA) might only be available in certain file types. So, learning to be flexible now, or least knowing there are multiple options, can save you headaches in the future.

1.2 Different types of files

A quick way (on a Windows machine) to determine what type of file you have and the directory in which it is located - two pieces of information that are necessary for loading data into R - is to right click on the file and select **Properties**. In the examples below, you will need to replace the file location with the directory in which you placed the file you downloaded (e.g., “C:/Users/ENT_6702/Stuff/Iris_data_csv_file.csv”) AND make sure the file suffix (e.g., .csv vs. .txt) is correct. Note the use of forward slashes, “/”, and not back slashes “\”, to separate levels of a directory.

1.2.1 Delimiters

You need to know which type of file you have because different file types use different characters (most commonly a comma or a tab) to separate the cells of data (= rows and columns of information). You won't actually see the commas, however, in comma separated files, so again looking at the **Properties** of the file can be useful. R needs to know the delimiting or separating character to correctly load data.

2 Example using `read.table()`

Load in the .csv (csv = comma separated values) version of the **Iris** data using the `read.table()` command. This command can handle a few different file types. The `file` argument is where you put your directory and file information. The `header=TRUE` argument tells R the first row of your data contains column names (which is almost always the case) and NOT actual data values. The `sep` argument tells R how values are separated. Again, for R to recognize rows and columns correctly, the `sep` argument needs to be specified correctly. Also note the use of quotation marks to input information for some arguments.

```
iris_df_csv <-
  read.table(file="DIRECTORY INFO HERE/USE FORWARD SLASHES/Iris_data_csv_file.csv",
            header=TRUE, sep=",")
```

Provide a `summary()`, `head()`, `tail()`, and `str()` of the data after loading it into R. This is a useful step to ensure that the data were loaded correctly.

```
iris_df_csv <- read.table(file = "C:/Users/perry.1864/OneDrive - The Ohio State University/Desktop/AU2020/Iris_data_csv_file.csv",
                        header = TRUE, sep = ",")
summary(iris_df_csv)
```

```
##   Sepal.Length   Sepal.Width   Petal.Length   Petal.Width
##   Min.    :4.300   Min.    :2.000   Min.    :1.000   Min.    :0.100
##   1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
##   Median :5.800   Median :3.000   Median :4.350   Median :1.300
##   Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
##   3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##   Max.    :7.900   Max.    :4.400   Max.    :6.900   Max.    :2.500
##   Species
##   Length:150
##   Class :character
##   Mode  :character
##
##
##
```

```
head(iris_df_csv) # top 6 rows of data
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1         5.1         3.5         1.4         0.2  setosa
## 2         4.9         3.0         1.4         0.2  setosa
## 3         4.7         3.2         1.3         0.2  setosa
## 4         4.6         3.1         1.5         0.2  setosa
## 5         5.0         3.6         1.4         0.2  setosa
## 6         5.4         3.9         1.7         0.4  setosa
```

```
tail(iris_df_csv) # bottom 6 rows of data
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 145         6.7         3.3         5.7         2.5 virginica
## 146         6.7         3.0         5.2         2.3 virginica
## 147         6.3         2.5         5.0         1.9 virginica
## 148         6.5         3.0         5.2         2.0 virginica
## 149         6.2         3.4         5.4         2.3 virginica
## 150         5.9         3.0         5.1         1.8 virginica
```

```
str(iris_df_csv) # column type, summary info
```

```
## '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 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : chr  "setosa" "setosa" "setosa" "setosa" ...
```

If a different number than the default (6) rows is of interest when using `head()` or `tail()` use:

```
tail(iris_df_csv, n=2)
```

```
##      Sepal.Length Sepal.Width Petal.Length Petal.Width  Species
## 149           6.2         3.4         5.4         2.3 virginica
## 150           5.9         3.0         5.1         1.8 virginica
```

3 A word of caution

R will recognize your columns as either categorical (which R calls a **factor** or a **character**) OR numeric. In this case, **Species** is a character and the other columns/variables are numeric. The **str()** command helps you confirm R is doing this step correctly. A single, non-numeric character in a column will make R load in that column as a factor. You can feed data to **as.factor()** to convert, for example, numbers or characters to factors. Make sure you have a good reason for doing so (e.g., you recorded treatments, technically a category, as 1, 2, and 3 in your spreadsheet).

```
iris_df_csv$Species_factor <- as.factor(iris_df_csv$Species)
iris_df_csv$Sepal.Length_factor <- as.factor(iris_df_csv$Sepal.Length)
str(iris_df_csv)
```

```
## 'data.frame':   150 obs. of  7 variables:
## $ Sepal.Length      : num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
## $ Sepal.Width       : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
## $ Petal.Length      : num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
## $ Petal.Width       : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
## $ Species           : chr  "setosa" "setosa" "setosa" "setosa" ...
## $ Species_factor    : Factor w/ 3 levels "setosa","versicolor",...: 1 1 1 1 1 1 1 1 1 1 ...
## $ Sepal.Length_factor: Factor w/ 35 levels "4.3","4.4","4.5",...: 9 7 5 4 8 12 4 8 2 7 ...

summary(iris_df_csv)
```

```
##      Sepal.Length      Sepal.Width      Petal.Length      Petal.Width
## Min.   :4.300      Min.   :2.000      Min.   :1.000      Min.   :0.100
## 1st Qu.:5.100      1st Qu.:2.800      1st Qu.:1.600      1st Qu.:0.300
## Median :5.800      Median :3.000      Median :4.350      Median :1.300
## Mean   :5.843      Mean   :3.057      Mean   :3.758      Mean   :1.199
## 3rd Qu.:6.400      3rd Qu.:3.300      3rd Qu.:5.100      3rd Qu.:1.800
## Max.   :7.900      Max.   :4.400      Max.   :6.900      Max.   :2.500
##
##      Species           Species_factor Sepal.Length_factor
## Length:150      setosa      :50      5           :10
## Class :character versicolor:50      5.1         : 9
## Mode  :character virginica :50      6.3         : 9
##                                     5.7         : 8
##                                     6.7         : 8
##                                     5.5         : 7
##                                     (Other):99
```

4 Example using readxl

Download the **readxl** package to your hard drive using R. To achieve this, you can use the **Packages** tab in RStudio or using an R function:

```
install.packages("readxl")
```

After you have downloaded the package to your hard drive, you will not have to complete that step again

(each package need only be installed once) unless you get a new computer or update R. Next, load the package into your current R session. Whenever you close R and start a new session, you will have to repeat this step.

```
library("readxl")
```

Load the data into R using the `read_excel()` command. This command will require a file type that ends in `.xlsx`. The `sheet` argument tells R which sheet in Excel to look for the data and `range` tells R where the top left and bottom right of your data frame occur in Excel. So, `range=B5:D10` would load in a rectangular data frame and the top left corner would be the cell B5 in Excel and the bottom right corner would be D10 (i.e., a data frame with three columns and 6 rows). Remember: in practice, your data frame will need to be a complete rectangle with no blank cells.

```
iris_df_excel <- read_excel(path="YOUR DIRECTORY INFO/FILE NAME WITH SUFFIX HERE",
                             sheet="INSERT CORRECT SHEET NAME",
                             range="TOP LEFT CELL:BOTTOM RIGHT CELL")
```

Provide a `summary()` of the data after loading it into R.

```
iris_df_excel <- read_excel(path="C:/Users/perry.1864/OneDrive - The Ohio State University/Desktop/AU2020/
                             sheet="some_famous_data",range="A1:E151")
summary(iris_df_excel) # summary of data
```

```
##   Sepal.Length   Sepal.Width   Petal.Length   Petal.Width
##   Min.    :4.300   Min.    :2.000   Min.    :1.000   Min.    :0.100
##   1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
##   Median :5.800   Median :3.000   Median :4.350   Median :1.300
##   Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
##   3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##   Max.    :7.900   Max.    :4.400   Max.    :6.900   Max.    :2.500
##   Species
##   Length:150
##   Class :character
##   Mode  :character
##
##
##
```

5 R Activity 2

1. Load in the .txt version of the **Iris** data using the `read.table()` command. Text files are tab delimited, so you will need to edit the code appropriately. Use `?read.table` and/or a Google search for something like “tab delimitation r read.table.” Also, I have replaced some of the values in this .txt version of the data with a “.”, which is how I record missing values (i.e., I have created some missing values in the data). You can use different identifiers (e.g., “-”, “Missing”) in your own work, but whatever you choose, be perfectly consistent across all missing values. Hint: you will need to tell R how to recognize those missing values using an additional argument in the `read.table()`.
2. How many rows are in the data?
3. How many **NA** values are there per column?
4. Provide the first 10 rows of the data.
5. Provide the last 3 rows of the data.