

Lab 3: Importing and working with Data

1 Introduction

The goal of this lab is to introduce you to how R deals with the most common way data is stored for analysis, a rectangular format like an Excel spreadsheet. Ideally each row represents data on a single observation and each column contains data on a single variable, or characteristic, of the observation. This is called **tidy data**. We will start to learn some tools to look at the data, and for getting data from an external file into R for analysis.

1.1 Matrices and Data Frames

Complete the **Matrices and Data Frames** Swirl lesson.

1.1.1 More on Data Frames

R comes pre-loaded with some training data sets you can practice with. One of them is the full **iris** data by Edgar Anderson. You can read more about this data set by typing `?iris`.

Similar to vectors and matrices, you select elements from a data frame with the help of square brackets `[]`. By using a comma, you can indicate what to select from the rows and the columns respectively. For example to select rows 1,2,3 and columns 2,3,4 in **iris** you type

```
iris[1:3,2:4]
```

```
##   Sepal.Width Petal.Length Petal.Width
## 1         3.5         1.4         0.2
## 2         3.0         1.4         0.2
## 3         3.2         1.3         0.2
```

You will often want to select an entire column, namely one specific variable from a data frame. If you want to select all elements of the variable **Sepal.Width**, both `iris[,3]` and `iris$Sepal.Width` do the trick. I call the latter method “dollar sign notation” for obvious reasons.

Often you don’t know what the variable name is or you want to check the spelling. The functions `names()` and `colnames()` is where you’ll turn to. For data frames they have the same result, but they do not each work on all types of R objects.

```
names(iris)
```

```
## [1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"
## [5] "Species"
```

```
colnames(iris)
```

```
## [1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"
## [5] "Species"
```

1.1.2 Practice

1. View the top 5 entries in the `Species` variable using both bracket and dollar sign notation.
2. Observe what the error message is when you have a typo in the variable name. Type `table(iris$species)` in the console to see (note the lower case s in species).

1.2 Looking at Data

Complete the **Looking at Data** Swirl lesson.

1.2.1 Practice

1. Use `summary()` to find out how many different species of flowers were sampled, and how many flowers from each species were sampled.
2. Use `table()` on the `Species` variable to confirm your answer.

1.3 Tidy Data

Review Ramnath Vaidyanathan's [workshop lecture notes](#) on tidy data, and either Hadley Wickham's paper on [Tidy Data](#) (at least through section 3) -or- an informal and code heavy [vignette](#) for the tidy data package.

The following is data collected by the Pew Research Center that examines the relationship between income and religion in the US. In other words, which religions have the wealthiest adherents?

| | religion | <\$10k | \$10-20k | \$20-30k | \$30-40k | \$40-50k | \$50-75k | \$75-100k |
|----|-------------------------|--------|----------|----------|----------|----------|----------|-----------|
| 1 | Agnostic | 27 | 34 | 60 | 81 | 76 | 137 | 122 |
| 2 | Atheist | 12 | 27 | 37 | 52 | 35 | 70 | 73 |
| 3 | Buddhist | 27 | 21 | 30 | 34 | 33 | 58 | 62 |
| 4 | Catholic | 418 | 617 | 732 | 670 | 638 | 1116 | 949 |
| 6 | Evangelical Prot | 575 | 869 | 1064 | 982 | 881 | 1486 | 949 |
| 7 | Hindu | 1 | 9 | 7 | 9 | 11 | 34 | 47 |
| 8 | Historically Black Prot | 228 | 244 | 236 | 238 | 197 | 223 | 131 |
| 9 | Jehovah's Witness | 20 | 27 | 24 | 24 | 21 | 30 | 15 |
| 10 | Jewish | 19 | 19 | 25 | 25 | 30 | 95 | 69 |
| 11 | Mainline Prot | 289 | 495 | 619 | 655 | 651 | 1107 | 939 |
| 12 | Mormon | 29 | 40 | 48 | 51 | 56 | 112 | 85 |
| 13 | Muslim | 6 | 7 | 9 | 10 | 9 | 23 | 16 |
| 14 | Orthodox | 13 | 17 | 23 | 32 | 32 | 47 | 38 |
| 15 | Other Christian | 9 | 7 | 11 | 13 | 13 | 14 | 18 |
| 16 | Other Faiths | 20 | 33 | 40 | 46 | 49 | 63 | 46 |
| 17 | Other World Religions | 5 | 2 | 3 | 4 | 2 | 7 | 3 |
| 18 | Unaffiliated | 217 | 299 | 374 | 365 | 341 | 528 | 407 |

1. What characteristics are being measured here? Hint: There are 3.
2. Is this tidy data?
3. What would a tidy version of this data set look like?

1.4 Importing Data

In this bootcamp we are only going to explore reading flat files that exist on your computer into R from three most commonly used data sources: A text file, A CSV file and an Excel file.

Go to the [Course schedule](#) and download the files for this lab. Be sure to put into your RBootcamp folder. You have to **right click** and select **save as** to correctly save the Email file}. Otherwise it will open in a browser window and you *cannot* save it from there correctly. Once they are in your RBootcamp folder, go ahead and open them just to see what they look like.

1.4.0.1 Text and CSV files Text files are very simple files that have a `.txt` file extension. Columns are separated by a **delimiter**. Common delimiters include a space, a comma (,) or a tab. Uncommon delimiters could include a % or even a semi-colon. Follow along with these examples and make sure you can read in the data correctly and that it matches what is shown below.

We will use the `read.table()` function that is in base R to read in any type of delimited file. Don't forget that you need to update the path shown (currently displaying `../data/`) to **YOUR** path on **YOUR** machine that points to your RBootcamp folder. Review the lecture notes on how to find your path if you are still struggling with this.

A tab-delimited text file can be read in using `"\t"` as the delimitator character. In this class you **ALWAYS** want to include `header=TRUE` to signify that the data in the first row contains our column names.

```
email50 <- read.table("../data/email50.txt", header=TRUE, sep="\t")
dim(email50)
```

```
## [1] 50 21
```

```
class(email50)
```

```
## [1] "data.frame"
```

```
email50[1:6,1:6]
```

```
##   spam to_multiple from cc sent_email      time
## 1    0             0    1 0          1 2012-01-04 05:19:16
## 2    0             0    1 0          0 2012-02-16 12:10:06
## 3    1             0    1 4          0 2012-01-04 07:36:23
## 4    0             0    1 0          0 2012-01-04 09:49:52
## 5    0             0    1 0          0 2012-01-27 01:34:45
## 6    0             0    1 0          0 2012-01-17 09:31:57
```

CSV is a fancy way of saying a text file with comma-separated values (i.e. CSV). We could use `read.table()` but `read.csv()` is optimized to read in CSV files.

```
NCbirths <- read.csv("../data/NCbirths.csv", header=TRUE)
dim(NCbirths)
```

```
## [1] 1000  13
```

```
class(NCbirths)
```

```
## [1] "data.frame"
```

```
head(NCbirths)
```

```
##   fage mage      mature weeks   premie visits marital gained weight
## 1   NA  13 younger mom    39 full term    10 married    38   7.63
## 2   NA  14 younger mom    42 full term    15 married    20   7.88
## 3  19  15 younger mom    37 full term    11 married    38   6.63
## 4  21  15 younger mom    41 full term     6 married    34   8.00
## 5   NA  15 younger mom    39 full term     9 married    27   6.38
## 6   NA  15 younger mom    38 full term    19 married    22   5.38
##   lowbirthweight gender      habit whitemom
## 1           not low   male nonsmoker not white
## 2           not low   male nonsmoker not white
## 3           not low female nonsmoker    white
## 4           not low   male nonsmoker    white
## 5           not low female nonsmoker not white
## 6              low   male nonsmoker not white
```

```
str(NCbirths)
```

```
## 'data.frame':   1000 obs. of  13 variables:
##  $ fage          : int  NA NA 19 21 NA NA 18 17 NA 20 ...
##  $ mage          : int  13 14 15 15 15 15 15 16 16 ...
##  $ mature        : Factor w/ 2 levels "mature mom","younger mom": 2 2 2 2 2 2 2 2 2 ...
##  $ weeks         : int  39 42 37 41 39 38 37 35 38 37 ...
##  $ premie        : Factor w/ 2 levels "full term","premie": 1 1 1 1 1 1 1 2 1 1 ...
##  $ visits        : int  10 15 11 6 9 19 12 5 9 13 ...
##  $ marital       : Factor w/ 2 levels "married","not married": 1 1 1 1 1 1 1 1 1 1 ...
##  $ gained        : int  38 20 38 34 27 22 76 15 NA 52 ...
##  $ weight        : num  7.63 7.88 6.63 8 6.38 5.38 8.44 4.69 8.81 6.94 ...
##  $ lowbirthweight: Factor w/ 2 levels "low","not low": 2 2 2 2 2 1 2 1 2 2 ...
##  $ gender        : Factor w/ 2 levels "female","male": 2 2 1 2 1 2 2 2 2 1 ...
##  $ habit         : Factor w/ 2 levels "nonsmoker","smoker": 1 1 1 1 1 1 1 1 1 1 ...
##  $ whitemom      : Factor w/ 2 levels "not white","white": 1 1 2 2 1 1 1 1 2 2 ...
```

1.4.0.2 CSV and Excel files The best method I have found so far to read in Excel files is from the `readxl` package by Hadley Wickham. This package need to be installed first, and then can be simply loaded each time you start an R session where you will be reading in this type of data. Go ahead and install it now.

The `read_excel()` function is what we are going to use. Note the use of the underscore `_` instead of a period `.` between read and the file type.

```
library(readxl)
OSCounty <- read_excel("../data/OSCounty.xlsx", sheet=1, col_names=TRUE)
dim(OSCounty)
```

```
## [1] 3116 56
```

```
class(OSCounty)
```

```
## [1] "tbl_df"      "tbl"        "data.frame"
```

```
OSCounty[1:6,1:6]
```

```
##           name    state FIPS pop2010 pop2000 age_under_5
## 1 Autauga County Alabama 1001   54571   43671         6.6
## 2 Baldwin County Alabama 1003  182265  140415         6.1
## 3 Barbour County Alabama 1005   27457   29038         6.2
## 4  Bibb County Alabama 1007   22915   20826         6.0
## 5 Blount County Alabama 1009   57322   51024         6.3
## 6 Bullock County Alabama 1011   10914   11714         6.8
```

```
## str(OSCounty) #Not run due to the length of output.
```

Notice that `OSCounty` isn't just a data frame, but also of class `tbl_df`. It's just another format of data storage. You are welcome to look up the differences and any advantages of using `tbl_df` over `data.frame` on your own time, we will not cover those differences in this class.