Working with Data in R

Rob Williams

UNC-CH Political Science

January 22, 2019

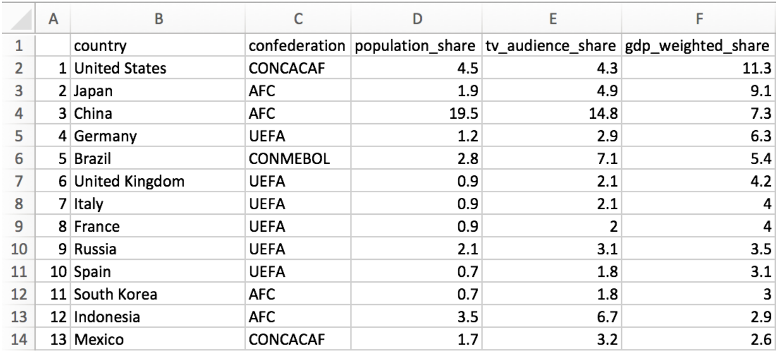
Today

Everything we did last week, we did within R

- We didn't have to load any data from outside of R
- Today we'll learn how to work with data in R
- And how to load real data into R from our computer

Data (not in R)

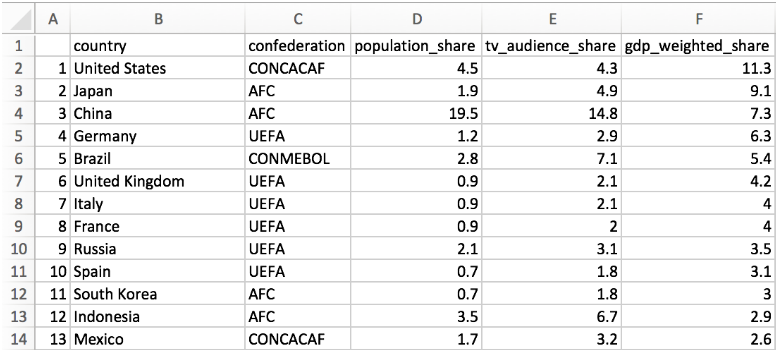
When you open datasets in a spreadsheet program like Excel, you'll see something like the following



	A	B	C	D	E	F
1		country	confederation	population_share	tv_audience_share	gdp_weighted_share
2	1	United States	CONCACAF	4.5	4.3	11.3
3	2	Japan	AFC	1.9	4.9	9.1
4	3	China	AFC	19.5	14.8	7.3
5	4	Germany	UEFA	1.2	2.9	6.3
6	5	Brazil	CONMEBOL	2.8	7.1	5.4
7	6	United Kingdom	UEFA	0.9	2.1	4.2
8	7	Italy	UEFA	0.9	2.1	4
9	8	France	UEFA	0.9	2	4
10	9	Russia	UEFA	2.1	3.1	3.5
11	10	Spain	UEFA	0.7	1.8	3.1
12	11	South Korea	AFC	0.7	1.8	3
13	12	Indonesia	AFC	3.5	6.7	2.9
14	13	Mexico	CONCACAF	1.7	3.2	2.6

Data (not in R)

When you open datasets in a spreadsheet program like Excel, you'll see something like the following



	A	B	C	D	E	F
1		country	confederation	population_share	tv_audience_share	gdp_weighted_share
2	1	United States	CONCACAF	4.5	4.3	11.3
3	2	Japan	AFC	1.9	4.9	9.1
4	3	China	AFC	19.5	14.8	7.3
5	4	Germany	UEFA	1.2	2.9	6.3
6	5	Brazil	CONMEBOL	2.8	7.1	5.4
7	6	United Kingdom	UEFA	0.9	2.1	4.2
8	7	Italy	UEFA	0.9	2.1	4
9	8	France	UEFA	0.9	2	4
10	9	Russia	UEFA	2.1	3.1	3.5
11	10	Spain	UEFA	0.7	1.8	3.1
12	11	South Korea	AFC	0.7	1.8	3
13	12	Indonesia	AFC	3.5	6.7	2.9
14	13	Mexico	CONCACAF	1.7	3.2	2.6

- Each column represents a different **variable**
- Each row represents a different **observation**

Data basics

On Tuesday, we learned how to use the `c()` function to create a **vector** in R

```
x <- c(1, 4, 2, 9)
```

- This vector has a mathematical representation as well: $\mathbf{x} = [1, 4, 2]$
- We can also write this vector as

$$\boldsymbol{x} = \begin{bmatrix} 1 \\ 4 \\ 2 \end{bmatrix}$$

- This is called a **column vector**
- You can think of each column in the World Cup audience data as a column vector

The matrix

We can think of the entire World Cup audience dataset as a **matrix**

- A matrix is a mathematical construct made up of **rows** and **columns**

$$\mathbf{A} = \begin{bmatrix} 1 & 3 \\ 4 & 5 \\ 2 & 6 \end{bmatrix}$$

- We describe the **dimensionality** (shape) of a matrix in terms of **rows** and **columns**
  - $\mathbf{A}$ is a 3×2 matrix
- We refer to elements of a matrix by their row and column positions
  - $\mathbf{A}_{1,1} = 1$
  - $\mathbf{A}_{2,2} =$

The matrix

We can think of the entire World Cup audience dataset as a **matrix**

- A matrix is a mathematical construct made up of **rows** and **columns**

$$\mathbf{A} = \begin{bmatrix} 1 & 3 \\ 4 & 5 \\ 2 & 6 \end{bmatrix}$$

- We describe the **dimensionality** (shape) of a matrix in terms of **rows** and **columns**
  - $\mathbf{A}$ is a 3×2 matrix
- We refer to elements of a matrix by their row and column positions
  - $\mathbf{A}_{1,1} = 1$
  - $\mathbf{A}_{2,2} = 5$
  - $\mathbf{A}_{3,1} =$

The matrix

We can think of the entire World Cup audience dataset as a **matrix**

- A matrix is a mathematical construct made up of **rows** and **columns**

$$\mathbf{A} = \begin{bmatrix} 1 & 3 \\ 4 & 5 \\ 2 & 6 \end{bmatrix}$$

- We describe the **dimensionality** (shape) of a matrix in terms of **rows** and **columns**
  - $\mathbf{A}$ is a 3×2 matrix
- We refer to elements of a matrix by their row and column positions
  - $\mathbf{A}_{1,1} = 1$
  - $\mathbf{A}_{2,2} = 5$
  - $\mathbf{A}_{3,1} = 2$

The matrix

We can think of the entire World Cup audience dataset as a **matrix**

- A matrix is a mathematical construct made up of **rows** and **columns**

$$\mathbf{A} = \begin{bmatrix} 1 & 3 \\ 4 & 5 \\ 2 & 6 \end{bmatrix}$$

- We describe the **dimensionality** (shape) of a matrix in terms of **rows** and **columns**
  - $\mathbf{A}$ is a 3×2 matrix
- We refer to elements of a matrix by their row and column positions
  - $\mathbf{A}_{1,1} = 1$
  - $\mathbf{A}_{2,2} = 5$
  - $\mathbf{A}_{3,1} = 2$

Unlike in Excel, it's important to remember the dimensionality of our data in R!

Matrices in R

We can create a matrix very easily in R

- First, create two vectors

```
a1 <- c(1, 4, 2)
a2 <- c(3, 5, 6)
```

- Next, use the `cbind()` function to combine by column

```
A <- cbind(a1, a2)
```

- And let's check out our handiwork

```
A
```

```
##      [,1] [,2]
## [1,]    1    3
## [2,]    4    5
## [3,]    2    6
```

Matrices in R

Just like we can refer to the elements of a matrix by row and column number mathematially, we do can the same thing in R

- To do this, we use the square bracket operator [] after the name of our matrix object
  - To get $\mathbf{A}_{1,1}$, we do

    ```
    A[1,1]
    ```

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

  - To get $\mathbf{A}_{2,2}$, we do

    ```
    A[2,2]
    ```

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

  - To get $\mathbf{A}_{3,1}$, we do

    ```
    A[3,1]
    ```

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

Matrices in R

But what if we want to get more than just one element out of our matrix?

- We can give R more than one number for either rows

```
A[1:2, 1] # rows 1 and 2, colum 1
```

```
## [1] 1 4
```

- or columns

```
A[2, 1:2] # row 2, columns 1 and 2
```

```
## [1] 4 5
```

- If we want to get an entire row or column, we can just leave that side of the [] blank

```
A[, 2] # rows 1-3, column 2
```

```
## [1] 3 5 6
```

Folders and files

Now that we know about matrices, we can actually get our hands on some data!

- Unfortunately, before we can, we have to take a little detour to learn about how our computers work
- If we don't we're going to have a hard time getting our data *into* R

Computers store files in **directories**, which we often refer to as folders

- Every word document you write is a file
- Every photo you download is a file
- Every file lives in a folder
- Folders can live inside other folders
- All of these files and folders live inside your computer

Boxes and Cats

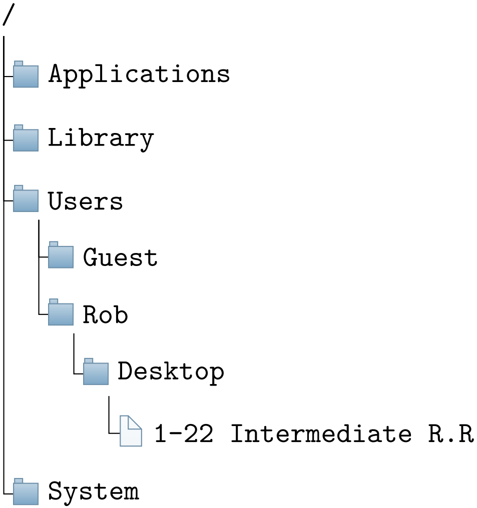
- You can put a box in a box
- You can put a cat in a box
- You can put a cat in a box in a box
- You **can't** put a box in a cat
- You **can't** put a cat in a cat

Boxes and Cats

- You can put a box in a box
- You can put a cat in a box
- You can put a cat in a box in a box
- You **can't** put a box in a cat
- You **can't** put a cat in a cat

You can think of your computer as the room all these boxes and cats are in

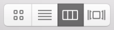
Directory structures

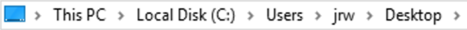


- /
  - Applications
  - Library
  - Users
    - Guest
    - Rob
      - Desktop
        - 1-22 Intermediate R.R
  - System

Making directories easier

The default settings on Finder (Mac) and File Explorer (Windows), obscure how directory structures work, and where your files actually *are*

- You can make things easier on a Mac by choosing to display your files in columns by clicking the columns button in Finder 

- The closest thing Windows has in File Explorer is the toolbar at the top of a window 

  This PC › Local Disk (C:) › Users › jrw › Desktop ›

Directories in R

Directories are important because we need to tell R where to look for files. R has a **working directory**, which is where R will look for any files you tell it to. The working directory is a directory (folder) on your computer.

- To check R's current working directory, use the `getwd()` command

```
getwd()
```

```
## [1] "/Users/Rob/Dropbox/UNC/Teaching/281 Spring 2019/Slides"
```

- To change R's working directory, use the `setwd()` command

```
setwd('~/Dropbox/UNC/Teaching/281 Spring 2019/Slides')
```

This won't actually change my working directory, because I'm setting the new working directory to the current working directory.

Directories in R

Whenever you first start up R, your working directory will be your home directory.

- On my Mac this is `/Users/Rob`
- On a Windows desktop on campus this is `C:/Users/jrw/Documents`
- You can use the tilde (`~`) as a shortcut for your home directory when setting your working directory in R
- On my Mac, these two commands do the same thing
  - `setwd('~/Desktop/R')`
  - `setwd('/Users/Rob/Desktop/R')`
- One way to make this process a little easier, is to make sure that RStudio *isn't* open, and then open it by clicking on the R script you want to edit. After you do this, RStudio will open and the working directory will be the folder where that R script is located.

Loading data in R

Now that we've told R where it should be looking, we need to actually read our data into R.

- We do this with the `read.csv()` command
- `.csv` files are like Excel spreadsheets, but much simpler
    - You can look at and edit them with a regular text editor (TextEdit, Notepad, etc.)
    - Almost any type of statistical software can open them (Stata, SPSS, Python, etc.)
    - You can't put things like formulas or links to other sheets in them

Remember, we need to **assign** the contents of the `.csv` file to an **object**, otherwise R will just print them out in the console

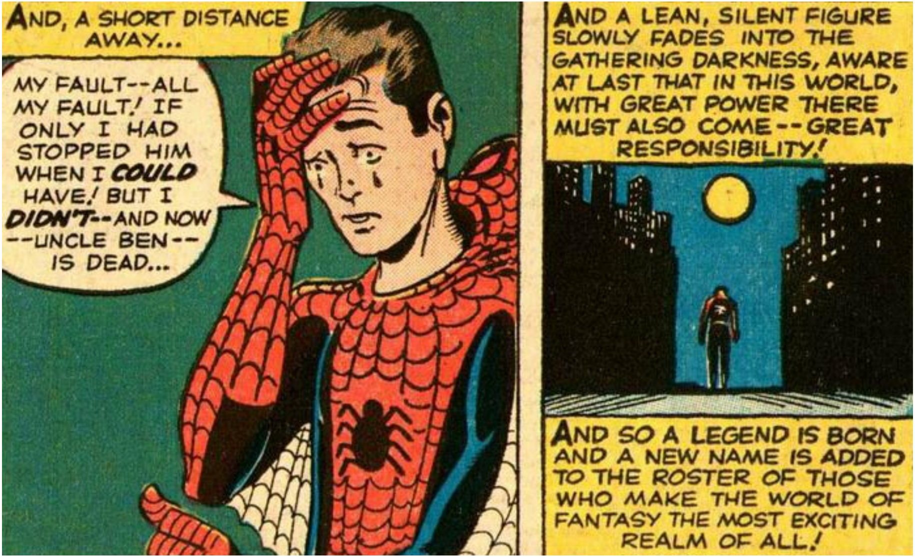
```
dat <- read.csv('Data/FIFA Audiences.csv')
```

Another R caveat

You can name an R object **anything**.

Another R caveat

You can name an R object **anything**.


AND, A SHORT DISTANCE AWAY...
MY FAULT--ALL MY FAULT! IF ONLY I HAD STOPPED HIM WHEN I COULD HAVE! BUT I DIDN'T--AND NOW --UNCLE BEN-- IS DEAD...
AND A LEAN, SILENT FIGURE SLOWLY FADES INTO THE GATHERING DARKNESS, AWARE AT LAST THAT IN THIS WORLD, WITH GREAT POWER THERE MUST ALSO COME--GREAT RESPONSIBILITY!
AND SO A LEGEND IS BORN AND A NEW NAME IS ADDED TO THE ROSTER OF THOSE WHO MAKE THE WORLD OF FANTASY THE MOST EXCITING REALM OF ALL!

Naming

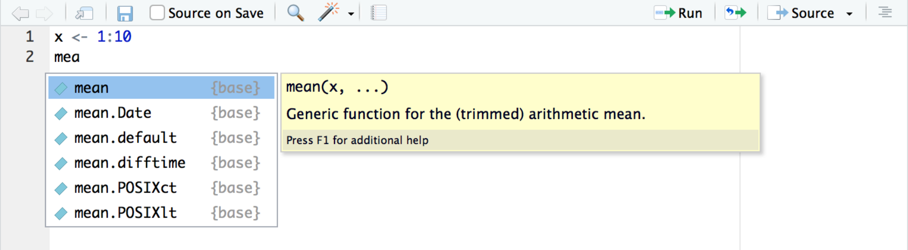
I didn't call our FIFA viewership dataset object `data` because `data()` is a base function in R

- Here's pretty much the most destructive thing you can do when naming ojects in R:

```
T <- F
T
```

```
## [1] FALSE
```

Lucikly, RStudio makes this easy to avoid with **code completion**



```
x <- 1:10
mea
```

mean(x, ...)

Generic function for the (trimmed) arithmetic mean.

Press F1 for additional help

Getting to know your data

Now that we've got our data into R, let's see what class of object they are

```
class(dat)
```

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

- Uh oh. We'be got a **data frame** instead of a matrix; no need to worry
  - Data frames are just fancier matrices
- We can get a quick overview with the `head()` function

```
head(dat)
```

```
##   X        country confederation population_share tv_audience_share
## 1 1  United States      CONCACAF              4.5               4.3
## 2 2          Japan           AFC              1.9               4.9
## 3 3          China           AFC             19.5              14.8
## 4 4        Germany          UEFA              1.2               2.9
## 5 5         Brazil      CONMEBOL              2.8               7.1
## 6 6 United Kingdom          UEFA              0.9               2.1
##   gdp_weighted_share
## 1               11.3
## 2                9.1
## 3                7.3
## 4                6.3
## 5                5.4
## 6                4.2
```

Getting to know your data

- We can pull out the column **names** we just saw with `head()` using the `names()` function

```
names(dat)
```

```
## [1] "X"                  "country"            "confederation"
## [4] "population_share"   "tv_audience_share"  "gdp_weighted_share"
```

- We can use the $ operator to access individual columns in a data frame

```
head(dat$population_share)
```

```
## [1]  4.5  1.9 19.5  1.2  2.8  0.9
```

  - The `head()` function works on vectors, too!

- This is better than typing `dat[, 2]` because it makes your code more **readable**
  - When you see `dat$population_share`, you can tell that line of code uses population share
  - This *isn't* the case for `dat[, 2]`

Getting more in-depth

Let's do some quick summarizing of our data

- How many countries are in each confederation?
- We can use the `table()` function

```
table(dat$confederation)
```

```
##
##      AFC      CAF CONCACAF CONMEBOL      OFC     UEFA
##       43       50       30       10       12       46
```

- The Confederation of African Football has the most members
- The Confederación Sudamericana de Fútbol has the fewest

Hands on with R

- Download today's R script from Sakai and open it up in RStudio