Introduction to Data Science Using R

Jason Renn

Required Software

Welcome to the first Korbel Skills Workshop[1]. These materials are available online at the following url:

Github Repository

http://bit.ly/2p7PYAg

You will need to install the following software for this workshop. Please install both (in order) in the next several minutes as we begin with a brief introduction of Data Science and the R statistical programming language.[2]

- R Statistical Programming Language
- RStudio IDE

What is Data Science?

Beyond its status as a buzzword[3], data science is simply a combination of a number of quantitatively-oriented skills, including statistics, mathematics, and computer science. There are many other terms and specific skills, but these are the core.

I would add that on top of the computer/quantitative skills, you will also have to learn how to write about and discuss data science techniques if you plan on using them in your career. Even for those of you that do not plan on persuing research, you will find yourself in a professional setting where you will have to engage with data. Why learn these data science skills? They should complement your knowledge about the content (domain expertise) and rhetorical skills *to help make you heard*.

These workshops will only serve as an introduction to these tools, primarily the software. If you want more training in these skills, know that it will require work. We will discuss more advanced uses of these tools and additional training at the end of the workshop.

Why `R`?

`R` is a computer programming language. It is open-source, popular, and strikes a balance between having more flexibility than many other software packages (e.g. Stata, SPSS) and being a little easier to use (e.g. Python, SAS, S). If you encounter data science in your careers, it will probably be in `R`.[4]

[1]All workshop materials are covered under a GPL 3.0 license. If you'd like more experience with the tools covered here, consult the instructors.

[2]If you would like to setup your computer for future workshops, consider installing a TeX Distribution, either MiKTeX for Windows or MacTeX for Mac. If you're on a Linux, some distributions include TeX Live but you should make sure to update. You can also install git for version control and create an account on github. If you use your du.edu email address, you'll qualify for a free student account.

[3]See Davenport and Patil (2012)

[4]Also note that because R is open-source, it is free. A commercial license for SPSS can run as high as $8,000 with similarly high costs for Stata as well

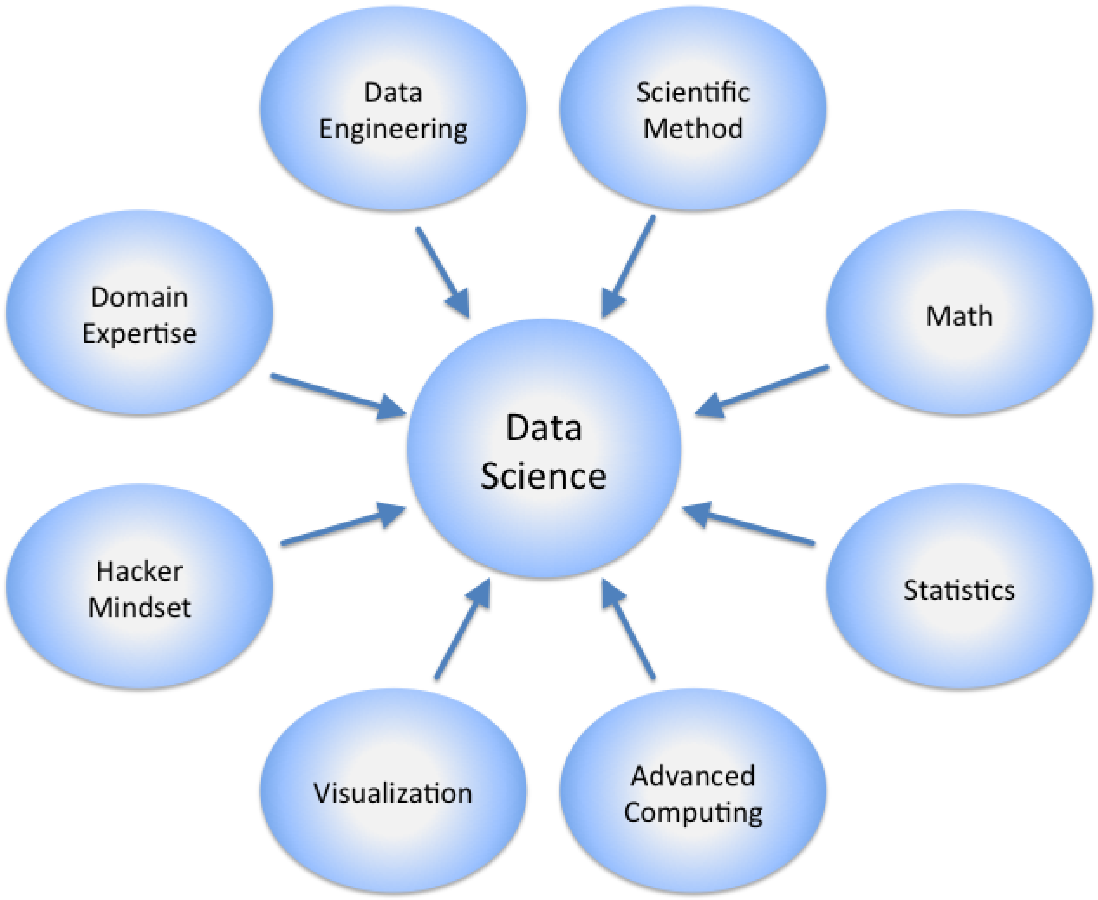

Data Engineering
Scientific Method
Domain Expertise
Math
Data Science
Hacker Mindset
Statistics
Visualization
Advanced Computing

Figure 1:

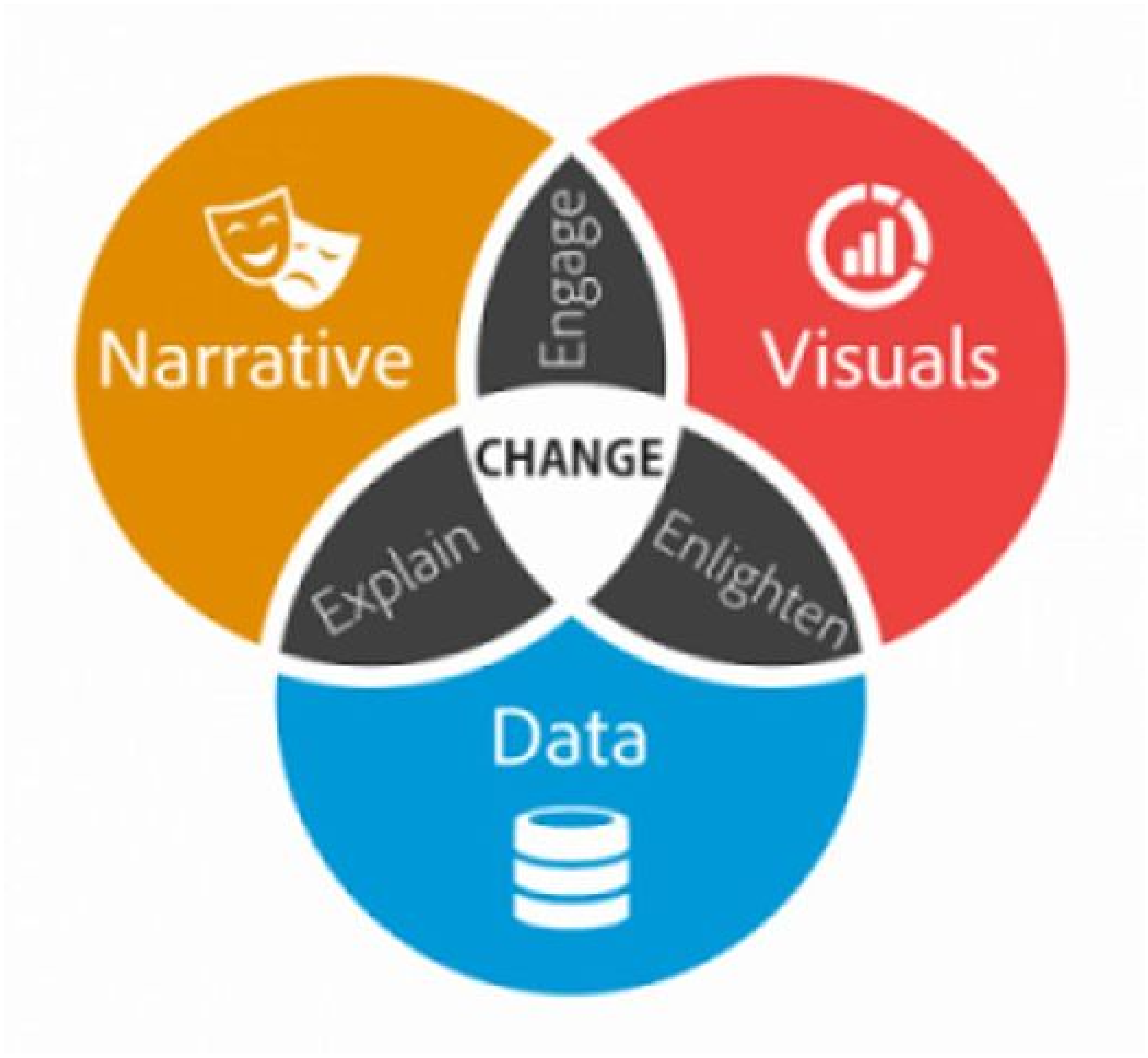
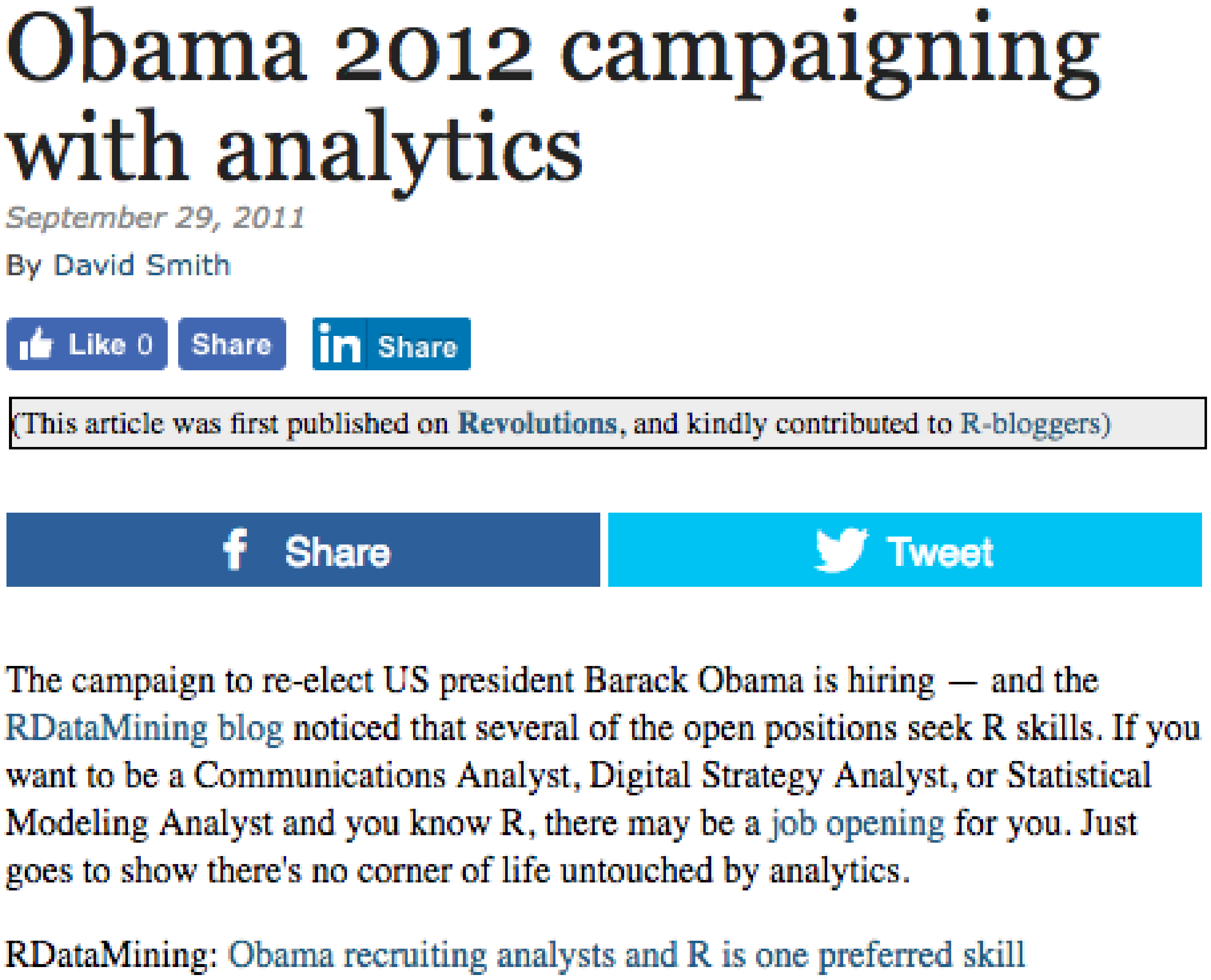


Figure 2:

Figure 3:


Obama 2012 campaigning with analytics

September 29, 2011

By David Smith

(This article was first published on Revolutions, and kindly contributed to R-bloggers)

The campaign to re-elect US president Barack Obama is hiring — and the RDataMining blog noticed that several of the open positions seek R skills. If you want to be a Communications Analyst, Digital Strategy Analyst, or Statistical Modeling Analyst and you know R, there may be a job opening for you. Just goes to show there's no corner of life untouched by analytics.

RDataMining: Obama recruiting analysts and R is one preferred skill

Figure 4:

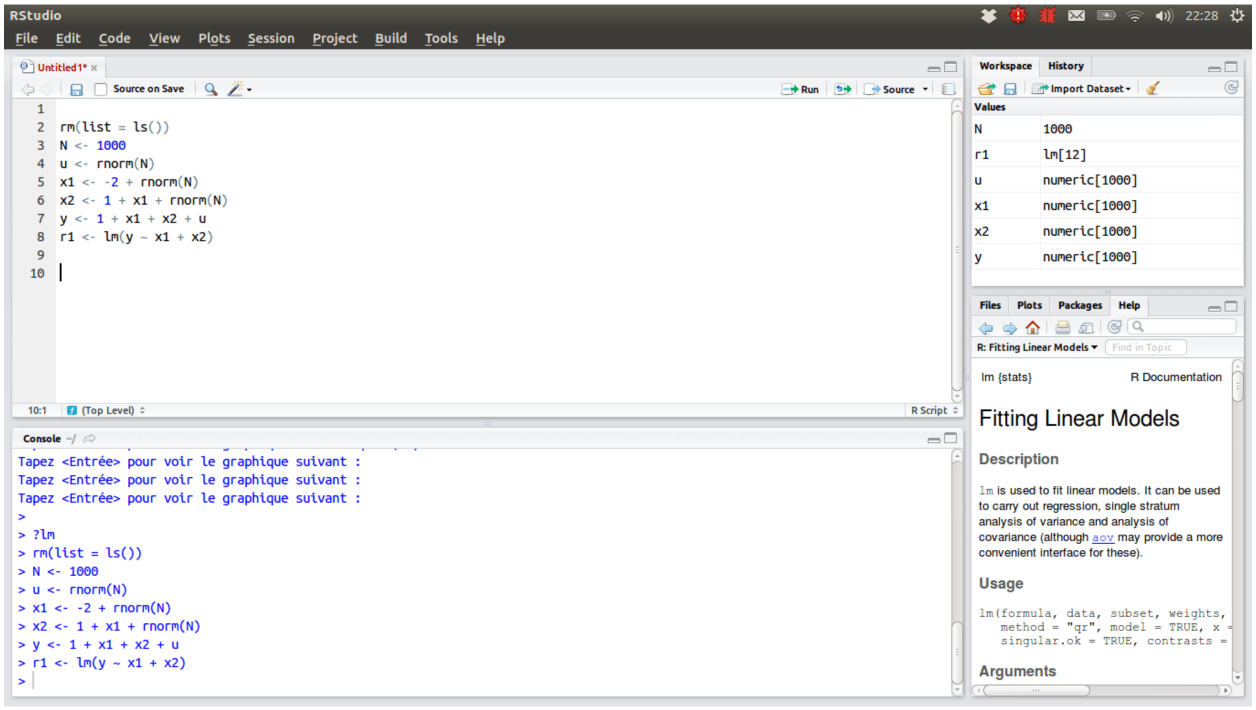
Consider the example of the 2012 Presidential Campaign.[5] The Obama campaign used data scientists, to collect data and create statistical models to help register voters and their craft outreach strategy. Rayid Ghani, a fellow at the University of Chicago and the Chief Scientist for the 2012 Obama for America campaign, led the effort. One of tools that he and his team used? `R`.

Will `R` always be around? Probably not. But it will prepare you to learn new languages as well. If you go further with data science, you will probably have to learn some other software. This is a mindset as much as it is formal training.

Exploring RStudio

RStudio is an Integrated Development Environment (IDE) that makes R much easier to use than in days past. There is a lot of functionality built into RStudio, including dynamic documents, version control, and

[5] See (https://venturebeat.com/2013/05/19/obama-campaigns-chief-data-guy-gets-candid-about-the-data-strategy-that-won-the-election/) and (https://www.technologyreview.com/s/508836/how-obama-used-big-data-to-rally-voters-part-1/) for more details.



```
rm(list = ls())
N <- 1000
u <- rnorm(N)
x1 <- -2 + rnorm(N)
x2 <- 1 + x1 + rnorm(N)
y <- 1 + x1 + x2 + u
r1 <- lm(y ~ x1 + x2)
```

```
N       1000
r1      lm[12]
u       numeric[1000]
x1      numeric[1000]
x2      numeric[1000]
y       numeric[1000]
```

```
Tapez <Entrée> pour voir le graphique suivant :
Tapez <Entrée> pour voir le graphique suivant :
Tapez <Entrée> pour voir le graphique suivant :
>
> ?lm
> rm(list = ls())
> N <- 1000
> u <- rnorm(N)
> x1 <- -2 + rnorm(N)
> x2 <- 1 + x1 + rnorm(N)
> y <- 1 + x1 + x2 + u
> r1 <- lm(y ~ x1 + x2)
>
```

lm {stats} R Documentation

Fitting Linear Models

Description

`lm` is used to fit linear models. It can be used to carry out regression, single stratum analysis of variance and analysis of covariance (although `aov` may provide a more convenient interface for these).

Usage

```
lm(formula, data, subset, weights,
   method = "qr", model = TRUE, x =
   singular.ok = TRUE, contrasts =
```

Arguments

Figure 5:

linkages to cloud-based resources. We won't cover those today – just the basics. For now, open up RStudio to make sure that the install worked.

The Interface

You should see something that looks like the image below:

By default there is space for four panels, though the top-left panel may be minimized the first time that you open RStudio.[6] Don't worry. We'll cover that in a moment.

Console

The first place that you should look is the console. This is where you enter code and commands into R for your computer to process. In the old days, we'd have to open up a terminal/command prompt.

Try to enter the following code into the **Console** panel to make sure that RStudio can correctly locate `R`. If successful, you the console should print out a friendly little message.

```
print("hello world")
```

Environment

In the top-right corner is a panel that has multiple tabs. Select the **Environment** tab if it isn't already selected. Note that it should currently say "Environment is empty."

[6]This is similar to Stata's layout.

One of the virtures of R is that it can load several objects into memory. Other software (Stata in particular) only loads one. While this helps keep the enviroment clean, it can make data management tasks pretty difficult. Let's make a few objects in R and save them. Make sure to pay attention to the **Environment** tab to see how it changes with the following commands. Type or copy/paste the following lines into the **Console**.

```
# This is a comment.
# Any line in R that begins with a "#" will not execute. For example:
# print("hello")
# Use comments to document code. I will use it provide additional explanation for code-related topics.
# Don't worry about typing of copying commented lines. Do read them.

# To assign a value, use the "<-". You can also use an "=" but that is typically reserved. We'll see wh
# You can lie about your age if you want.
yourAge <- 28
favoriteColor <- "blue"
x = 1
y = 3
```

What happened to the environment? You can now call any of the variables that you just created by referring to the by name. Try thte following:

```
x
y
x+y
```

Note that R is case sensitive. Typing the following will give you an error.

```
yourage
favoritecolor
```

You also can't save a variable with a space in the name – at least you shouldn't.

```
# This won't work
your age <- 28
```

File Explorer

In the bottom-right, you'll see the **Files** panel. You can access files saved on your computer here and if it is a file that R can load as a dataset, then you'll also get a prompt with options. You could easily never touch the **Files** panel, instead using the **Console** or options within the "File ->" menu to access things, but it can be useful. There are usually three or four ways to do something in 'R'. Do what works for you.

Scripts, Markdown, Datasets

The final panel contains text files that you can edit and will also display datasets. Most often, this space will have either a `.R` (script) or `.Rmd` (RMarkdown) file where you can save commands and output. Similar to a Stata `.do` file, this allows you to keep a record of everything you've done in case you have to redo something. Unlike the console output, you can easily save these files. The console output will get cleared from time to time. We won't create a script or markdown file today, but they are useful.

Installing and Requiring Packages

One of the best aspects of `R` is that people continue to develop additional features and expand the functionality. Any user can download and attach these software packages to their own installation of `R`. Since this is all open-source, there aren't any warranties or support, so the challenge with these packages is keeping them up-to-date and reading through some rather obtuse documentation.

We're going to install a few packages right now that will allow us to load data into 'R' from several file formats.

```r
# During the installation process, R may ask you to create a personal library or restart R. This should
# You can install one package at a time
install.packages("haven")

# Or multiple
install.packages("readxl", "foreign")
```

With these packages installed, you can now use commands that are not part of "base `R`.

Importing Datasets

One of the first skills that you'll have to learn when dealing with data is opening it. Many file formats are proprietary and used to require that you own the software (or a conversion tool) in order to view and use the data. The packages that you downloaded in the previous section will allow you to open these files within `R`. You can even use RStudio to view the files and export them to easier to access formats.

To begin, load the downloaded packages into memory using the `require()` command. Note that loading a package is a distinct step that you have to do after you download (`install.packages()`) a package.

```r
require(haven)

# Similarly, you can use the library() command
library(readxl)
```

You can open saved files on your computer or load data straight from the internet. Let's try this with two common IR datasets, the Polity Index and the UCDP/PRIO Armed Conflict Dataset.

- Polity Dataset Website
- Political Terror Score Website

```r
# Save and load through file menu
library(readxl)
polity <- read_excel("~/Downloads/p4v2016.xls")
View(p4v2016)

# Straight from the website
pts <- read.csv("http://www.politicalterrorscale.org/Data/Files/PTS-2017.csv")
```

For this example, we will use a dataset provided by the World Bank. This is a small dataset with only a little more than one thousand observations. You could use a spreadsheet program like Excel to open up and edit the data. For larger files, with tens-of-thousands, or even millions of observations are much easier to work with in `R` or similar programs.

We will load this file from the World Bank's website using an R package called `WDI`. This will allow us to do some basic data management before importing everything as well. We'll then export the data into a number of different formats so that you can view it in your desired program or even .

You can find these data files online in the repository's `data` folder:

Data Folder

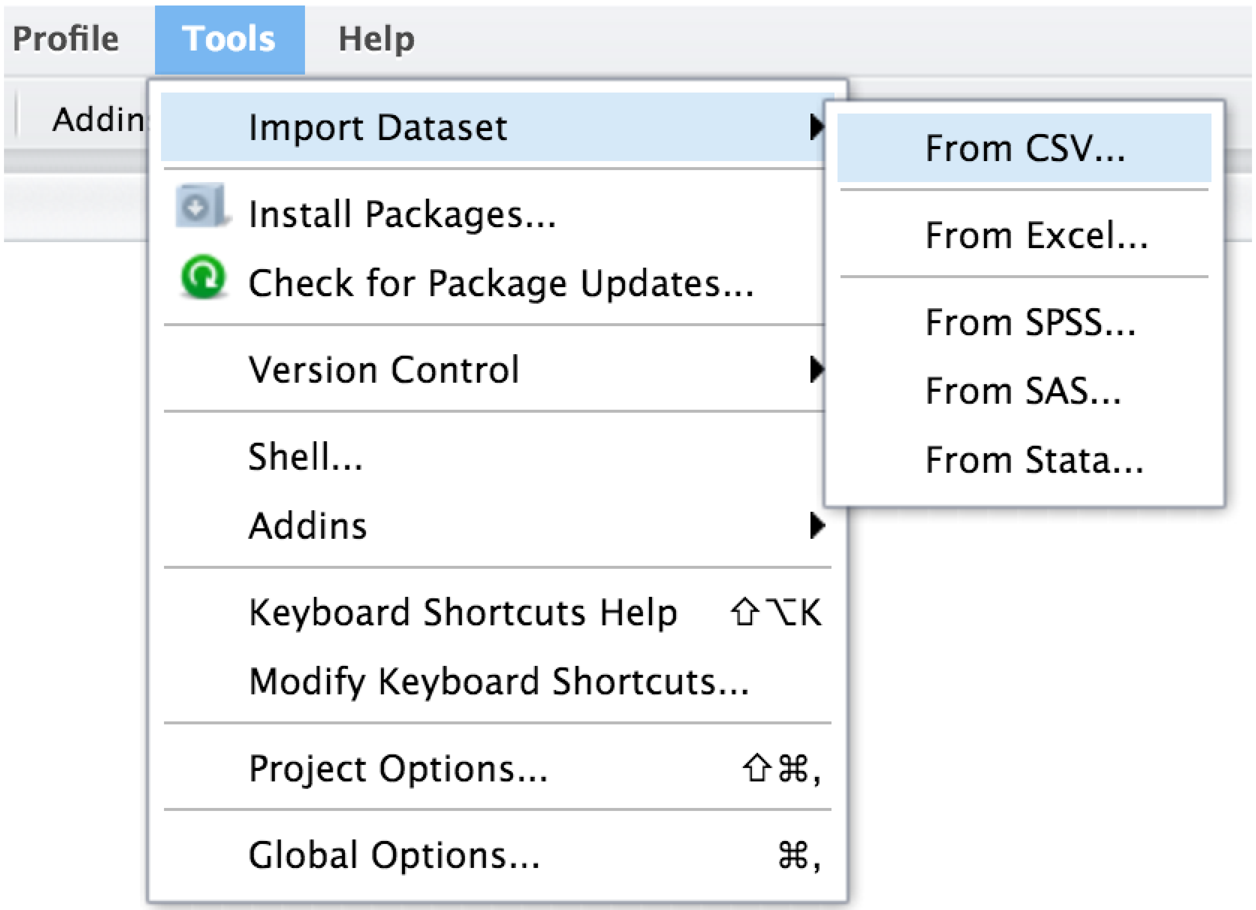


Figure 6:

R Data format

If you find a file with a `.Rdata` or `.rds` file extension, you can load it simply by finding it in the **Files** panel, clicking it, and choosing "yes" when prompted to load the data file into memory.

You can also find the file on your computer using the "File ->"Open File" dialogue menu. Notice that both of these methods simply pass a command to the console which looks like this:

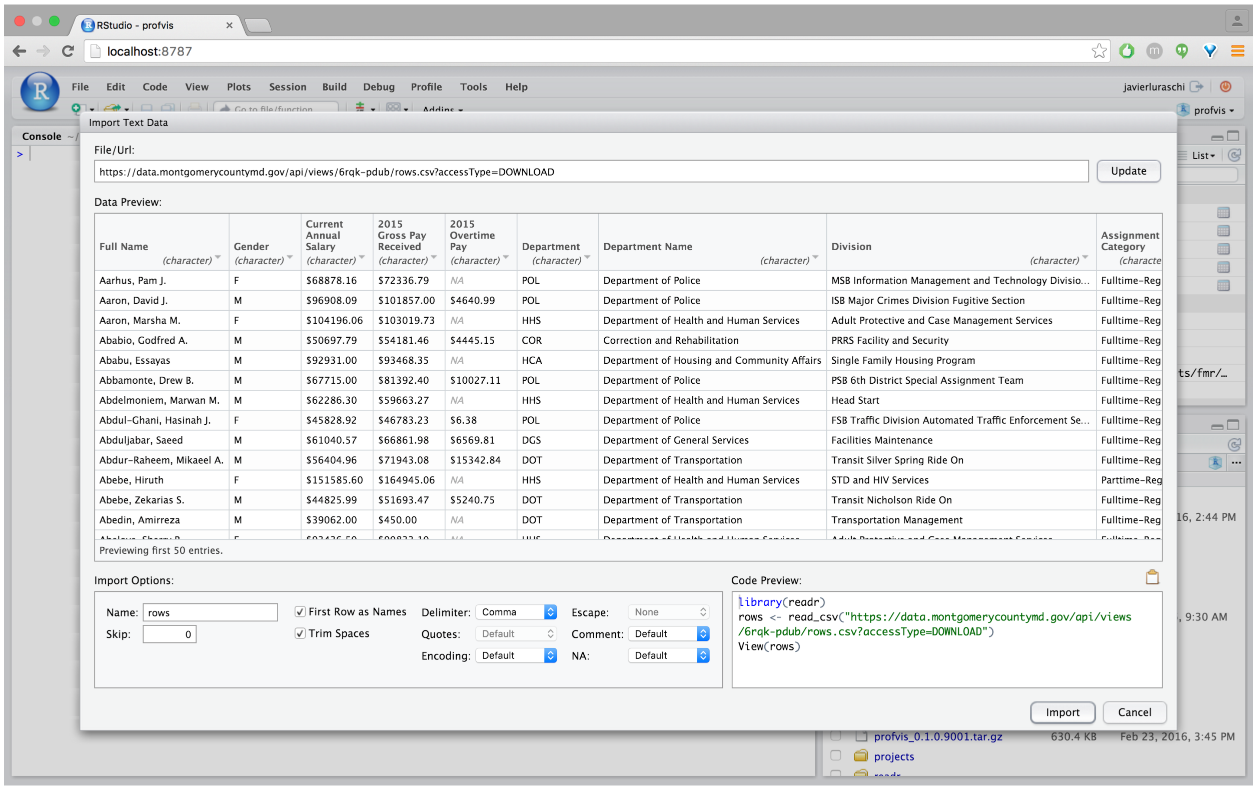
```
# Your specific file path will differ
load("~/user/documents/file/worldBank.Rdata")
```

If you know the file path and name of the dataset, you can load it directly in the **Console**.

Comma Delimited

A more common file type is a comma delimited or `.csv` file. These are text files with breaks between values (ascii and .txt files are similar). As the name suggests, a `.csv` has commas separating invidual values. This type of file is easy to open in a spreadsheet program, as long as it is not too large to load into your computer's memory.

You can import a `.csv` file into `R` in a similar way, by either finding it in thne **Files** pane or using the menu. Note that as this is not the native file format for `R` you will be "importing" the dataset and not simply "opening" it.



File/Url:

```
https://data.montgomerycountymd.gov/api/views/6rqk-pdub/rows.csv?accessType=DOWNLOAD
```

Data Preview:

Full Name (character)	Gender (character)	Current Annual Salary (character)	2015 Gross Pay Received (character)	2015 Overtime Pay (character)	Department (character)	Department Name (character)	Division (character)	Assignment Category (characte
Aarhus, Pam J.	F	$68878.16	$72336.79	NA	POL	Department of Police	MSB Information Management and Technology Divisio...	Fulltime-Reg
Aaron, David J.	M	$96908.09	$101857.00	$4640.99	POL	Department of Police	ISB Major Crimes Division Fugitive Section	Fulltime-Reg
Aaron, Marsha M.	F	$104196.06	$103019.73	NA	HHS	Department of Health and Human Services	Adult Protective and Case Management Services	Fulltime-Reg
Ababio, Godfred A.	M	$50697.79	$54181.46	$4445.15	COR	Correction and Rehabilitation	PRRS Facility and Security	Fulltime-Reg
Ababu, Essayas	M	$92931.00	$93468.35	NA	HCA	Department of Housing and Community Affairs	Single Family Housing Program	Fulltime-Reg
Abbamonte, Drew B.	M	$67715.00	$81392.40	$10027.11	POL	Department of Police	PSB 6th District Special Assignment Team	Fulltime-Reg
Abdelmoniem, Marwan M.	M	$62286.30	$59663.27	NA	HHS	Department of Health and Human Services	Head Start	Fulltime-Reg
Abdul-Ghani, Hasinah J.	F	$45828.92	$46783.23	$6.38	POL	Department of Police	FSB Traffic Division Automated Traffic Enforcement Se...	Fulltime-Reg
Abduljabar, Saeed	M	$61040.57	$66861.98	$6569.81	DGS	Department of General Services	Facilities Maintenance	Fulltime-Reg
Abdur-Raheem, Mikaeel A.	M	$56404.96	$71943.08	$15342.84	DOT	Department of Transportation	Transit Silver Spring Ride On	Fulltime-Reg
Abebe, Hiruth	F	$151585.60	$164945.06	NA	HHS	Department of Health and Human Services	STD and HIV Services	Parttime-Reg
Abebe, Zekarias S.	M	$44825.99	$51693.47	$5240.75	DOT	Department of Transportation	Transit Nicholson Ride On	Fulltime-Reg
Abedin, Amirreza	M	$39062.00	$450.00	NA	DOT	Department of Transportation	Transportation Management	Fulltime-Reg

Previewing first 50 entries.

Import Options:

Name: rows
Skip: 0
[✓] First Row as Names
[✓] Trim Spaces
Delimiter: Comma
Quotes: Default
Encoding: Default
Escape: None
Comment: Default
NA: Default

Code Preview:

```
library(readr)
rows <- read_csv("https://data.montgomerycountymd.gov/api/views
/6rqk-pdub/rows.csv?accessType=DOWNLOAD")
View(rows)
```

Figure 7:

When you choose the import option, you'll see a preview of the data as well as a number of additional options that you can choose from to make sure that the data load properly. Notice again that RStudio is actually just passing more commands to the console once you press "import."

The actual command to import a `.csv` file should look something like this:

```
# Again, your file path may look different.
# If you didn't run the require(haven) command earlier, this won't work.
# read_csv() is a command within haven, so if it isn't loaded the computer won't recognize it.
worldBank <- read_csv("worldBank.csv")
```

Other Formats (stata, spss, excel)

I have included the `worldBank` data in a few other file formats. The process to import them is similar. I've put the command line/console-based approach here. See if you can get one of the other file formats to load using any method that you please.

```
# A Stata file
worldBank <- read_dta("worldBank.dta")
# SPSS
worldBank <- read_sav("worldBank.sav")
# Excel. Note that this command comes from the "readxl" package, not "haven"
worldBank <- read_excel("worldBank.xlsx")
```

Accessing Data

Now that you have a dataset loaded into `R`, let's go over a few basics that will help you access the data.

Data Structures

Objects in the environment are of a certain type. For example, the `favoriteColor` value that you saved earlier is a set of letters. `R` saved this as a `character`. You can double-check this using the `class()` command. What kind of value is stored inside `yourAge`?

```
class(favoriteColor)
```

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

```
class(yourAge)
```

Datasets are collections of values. Instead of returning a specific variable type when you use the `class()` command on a dataset, the command will return its structure.

```
# the worldBank dataset is a "data.frame."
# You can think of this type of data structure as something with columns and rows or two-dimensional ob
class(worldBank)
```

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

If you look at the `worldBank` dataset, you'll see that it's a collection of many different types of values with values that are numeric and others that are characters. There are other variable types, but this is a useful, basic distinction.

Try looking at the underlying `worldBank` dataset using any of the commands below.

```
head(worldBank)
summary(worldBank)
View(worldBank)
```

Vectors

Each of the columns inside of the `worldBank` dataset is a vector. Others will refer to a vector as an array. A vector is a one-dimensional object. You can also think about it as a list of values. We'll make a small vector here.

```
myVector <- c("look", "over", "here")
```

This a vector that has three values in it. Of course you know this because you can see the code that generated the vector. However, if you didn't make the vector yourself you could check the length with the `length()` command.

```
length(myVector)
```

```
## [1] 3
```

You could access individual values in the array by stating the index or the position for a particular value. For example, if we wanted the first value in the `myVector` array, we would type the following.

```
myVector[1]
```

```
## [1] "look"
```

How would we use this within a dataset? Remember that our `worldBank` dataset is a two-dimensional object, but each column is just a vector. If you want to know what value is in a particular row, you can simply type the index or even a range of indicies. Try the following.

```
worldBank$country[1]
```

```
## [1] "Afghanistan"
```

```
worldBank$country[1:10]
```

```
##  [1] "Afghanistan" "Afghanistan" "Afghanistan" "Afghanistan" "Afghanistan" "Albania"     "Albania"
```

Data Frames

You can refer to an individual column in a data.frame by its name using the syntax in the example above or simply by its position. Think of this like a game of battleship. You can call out a specific position within a data.frame by typing out two numbers: the row and the column.

```
worldBank[1,2]
```

You can also look at the entire row or column by leaving on of the values blank. Try the following.

```
worldBank[,1]
worldBank[3,]
```

Boolean Operations

You can ask `R` to evaluate boolean or TRUE/FALSE statements by using the following syntax.

```
# "==" Asks whether something is exactly equivalent.
# The double equals is sometimes confusing. That's why we usually use "<-" to assign values and avoid t
1 == 1
x == 1
T == F
"bob" == "Bob"

# You can also do greater and less than. Of course this only makes sense for numerical values.
3 > 4
3 < 4

# Are both statements true? Use the '&' or AND command.
T & F
1 < 2 & 2 < 3
```

There are many other operators and commands that return TRUE and FALSE. These basic boolean operations are useful in isolating values within a dataset that meet certain requirements. Consider the following command which returns

```
# The United State's values for 2012
subset(worldBank, country=="United States" & year==2012)
```

```
##            country year iso3c greenhouseGasEmissions infantMortality homicideRate militaryExpenditur
## 1002 United States 2012   USA                6343841               6          4.7
```

```
# The US in any year after 2012
subset(worldBank, country=="United States" & year>2012)
```

```
##              country year iso3c greenhouseGasEmissions infantMortality homicideRate militaryExpenditur
## 1003 United States 2013   USA                     NA             5.9          4.5                   3
## 1004 United States 2014   USA                     NA             5.8          4.4                   3
## 1005 United States 2015   USA                     NA             5.7          4.9                   3
```

Try it Out

Use the example above to find Ecuador's infant mortality in 2012.

Next Steps

This workshop was an introduction to basic functions of the `R` statistical programming language. More training is available that will allow you to work with data and produce summaray statistics, visualizations, statistical models, and merge datasets. For more information and resources, talk with one of the facilitors before you leave. You can email me at duu.renn@du.edu.

If we have time, I'm happy to answer questions and get ideas for future workshops. Some cool links below:

- Shiny Apps
- Mapping Data with ggplot
- Mapping Data with leaflet
- Basic Visualization with ggplot
- Text Mining with R
- Web Scrapping with Rvest
- Web Scrapping with R Selenium
- Regression - OLS
- Regression - GLM
- Free R Book from those with Econ Backgrounds
- Free R book for those with no programing and limited math background
- Swirl Package
- Reproducible Documents
- Where to look for answers (and ask questions)