Introduction to Machine Learning for Social Scientists

Class 2: Intro to R

Edgar Franco Vivanco

Stanford University
Department of Political Science

edgarf1@stanford.edu

Summer 2018

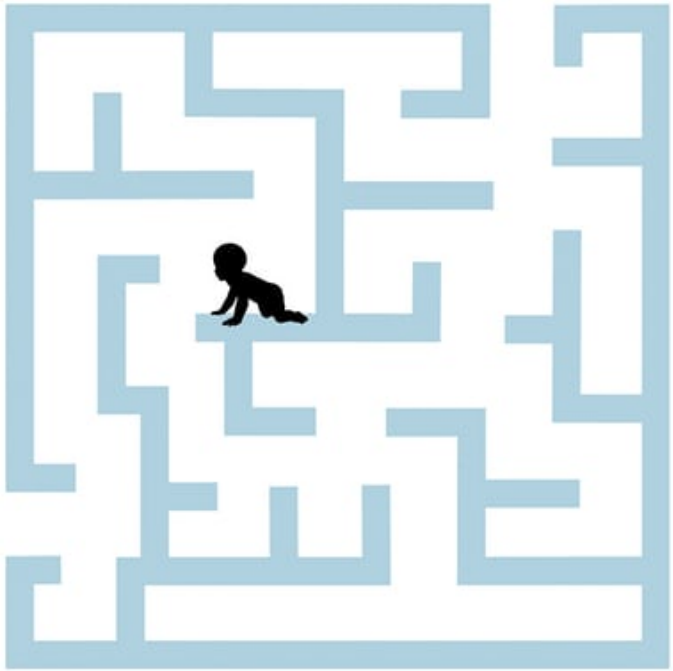
Learning Machines

- An example of AI "learning" how to solve problems.
- It is doing the kinds of things that animals do and that is to take direct routes wherever possible and shortcuts when they are available.
- A deep neural network, a computer program that uses multiple layers of artificial neurons to process information

Source: The Guardian, Google DeepMind's AI program learns human navigation skills

Learning Humans

- Can machines really "learn" like humans?
- "The learning process wasnt decoding, as he had originally thought, but something infinitely more continuous, complex and social."
- Human learning was communal and interactive. For a robot, the acquisition of language was abstract and formulaic.



Source: The Guardian, How babies learn and why robots cant compete

Today's Goals

- Get started with R

Today's Goals

- Get started with R
- Review some basic concepts of programming and data structures

Today's Goals

- Get started with R
- Review some basic concepts of programming and data structures
- Test the distribution of the group

Today's Goals

- Get started with R
- Review some basic concepts of programming and data structures
- Test the distribution of the group
- Get familiar with the class dynamics

Today's Goals

- Get started with R
- Review some basic concepts of programming and data structures
- Test the distribution of the group
- Get familiar with the class dynamics
- Get ready to start next week with ML

Download:

http://cran.cnr.berkeley.edu/

https://www.rstudio.com/products/rstudio/download/

What is R?

- R is a free software environment for statistical computing and graphics.
- R is a programing language.
- Lots of packages (More than 10,000!!!)for which allow specialized statistical techniques, graphical devices (ggplot2), import/export capabilities, reporting tools (knitr, Sweave), etc.
- Some programs are better for specific purposes but R is intended to do all. Like a Swiss Army Knife.

Why R?

- R does not involve lots of pointing and clicking, and thats a good thing
- R code is great for reproducibility
- R is interdisciplinary and extensible
- R works on data of all shapes and sizes
- R produces high-quality graphics
- R has a large and welcoming community (Many of them are willing to help you through mailing lists and websites such as Stack Overflow)

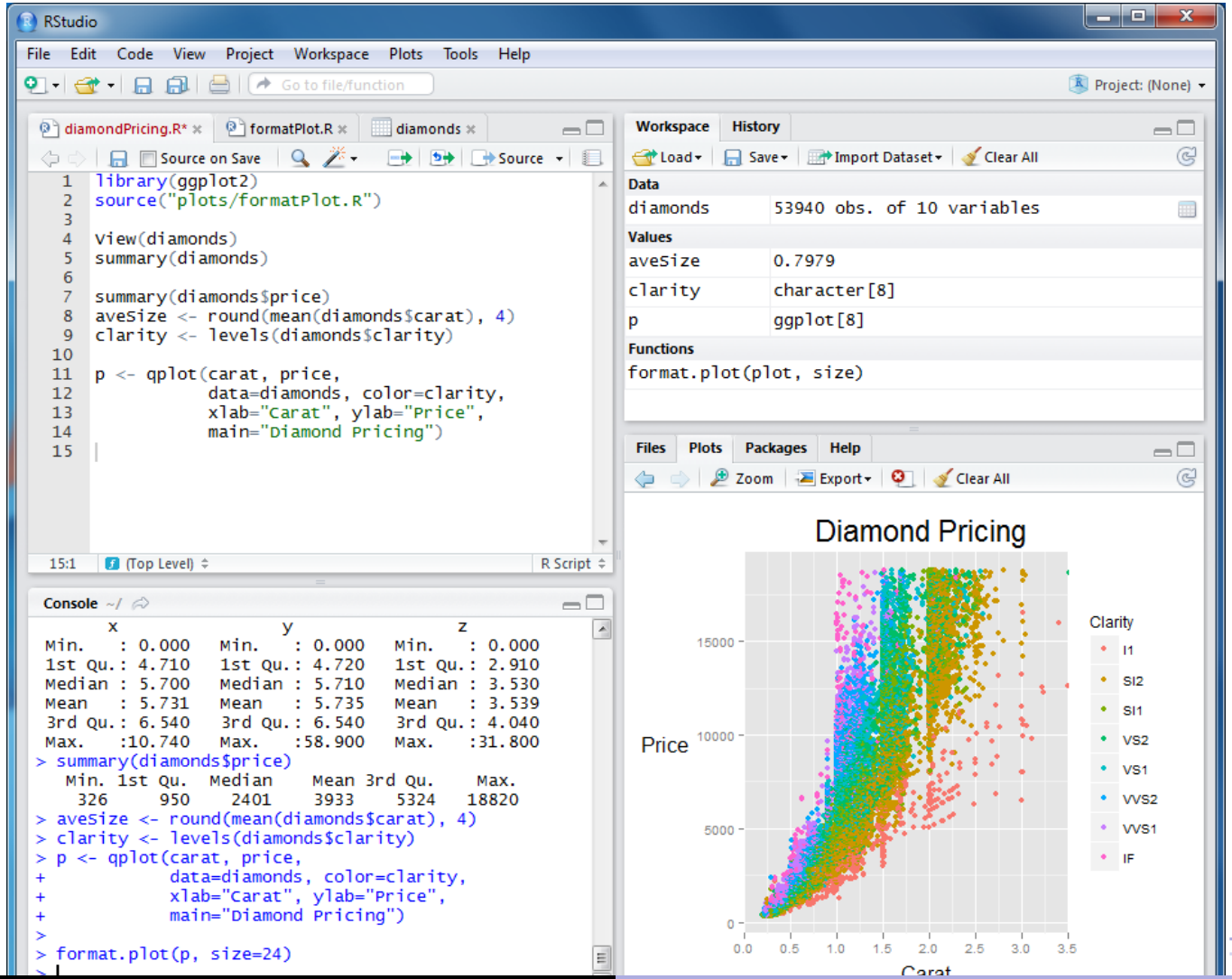
R Studio

What is R studio?

RStudio is a set of integrated tools designed to help you be more productive with R. It includes a console, syntax-highlighting editor that supports direct code execution, as well as tools for plotting, history, debugging and workspace management.

. . . just an easier way to work with R.

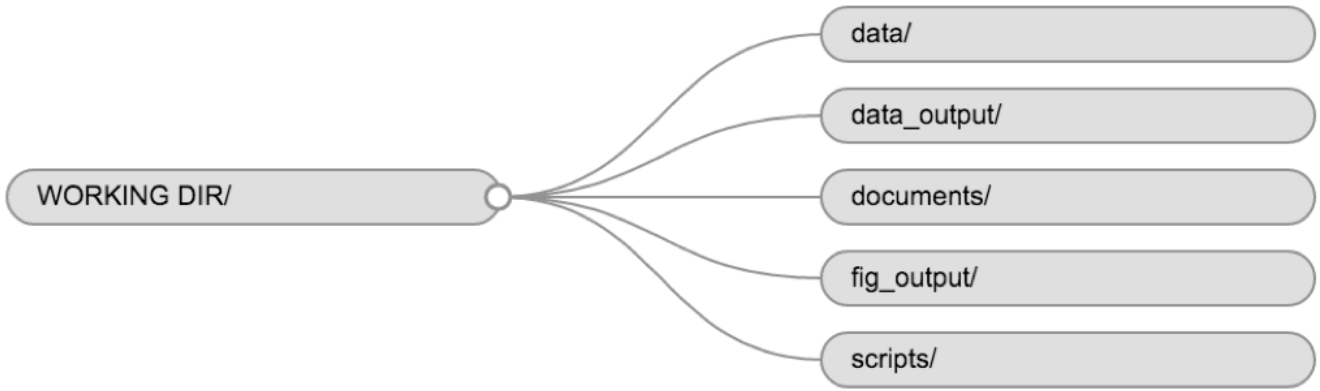
R Studio


RStudio
File Edit Code View Project Workspace Plots Tools Help
diamondPricing.R* formatPlot.R diamonds
Source on Save Source
1 library(ggplot2)
2 source("plots/formatPlot.R")
3
4 View(diamonds)
5 summary(diamonds)
6
7 summary(diamonds$price)
8 aveSize <- round(mean(diamonds$carat), 4)
9 clarity <- levels(diamonds$clarity)
10
11 p <- qplot(carat, price,
12 data=diamonds, color=clarity,
13 xlab="Carat", ylab="Price",
14 main="Diamond Pricing")
15
15:1 (Top Level) R Script
Console ~/
x y z
Min. : 0.000 Min. : 0.000 Min. : 0.000
1st Qu.: 4.710 1st Qu.: 4.720 1st Qu.: 2.910
Median : 5.700 Median : 5.710 Median : 3.530
Mean : 5.731 Mean : 5.735 Mean : 3.539
3rd Qu.: 6.540 3rd Qu.: 6.540 3rd Qu.: 4.040
Max. :10.740 Max. :58.900 Max. :31.800
> summary(diamonds$price)
Min. 1st Qu. Median Mean 3rd Qu. Max.
326 950 2401 3933 5324 18820
> aveSize <- round(mean(diamonds$carat), 4)
> clarity <- levels(diamonds$clarity)
> p <- qplot(carat, price,
+ data=diamonds, color=clarity,
+ xlab="Carat", ylab="Price",
+ main="Diamond Pricing")
>
> format.plot(p, size=24)
Workspace History
Load Save Import Dataset Clear All
Data
diamonds 53940 obs. of 10 variables
Values
aveSize 0.7979
clarity character[8]
p ggplot[8]
Functions
format.plot(plot, size)
Files Plots Packages Help
Zoom Export Clear All
Diamond Pricing
Price
15000
10000
5000
0
0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5
Carat
Clarity
I1
SI2
SI1
VS2
VS1
VVS2
VVS1
IF

Getting set up

1. Start RStudio.
2. Under the File menu, click on New project. Choose New directory, then New project.
3. Enter a name for this new folder (or directory), and choose a convenient location for it. This will be your working directory for the rest of the day (e.g., /polisci241).
4. Click on Create project.
5. Download the code handout, place it in your working directory and rename it

Working directory



Creating objects

- `<-` is the assignment operator. It assigns values on the right to objects on the left. So, after executing `<- 3` , the value of x is 3. The arrow can be read as 3 goes into x.

Creating objects

- `<-` is the assignment operator. It assigns values on the right to objects on the left. So, after executing `<- 3` , the value of x is 3. The arrow can be read as 3 goes into x.
- In RStudio, typing Alt + - (push Alt at the same time as the - key) will write `<-` in a single keystroke in a PC, while typing Option + - (push Option at the same time as the - key) does the same in a Mac.

Creating objects

- `<-` is the assignment operator. It assigns values on the right to objects on the left. So, after executing `<- 3` , the value of x is 3. The arrow can be read as 3 goes into x.
- In RStudio, typing Alt + - (push Alt at the same time as the - key) will write `<-` in a single keystroke in a PC, while typing Option + - (push Option at the same time as the - key) does the same in a Mac.
- Objects can be given any name such as x, current_temperature, or subject_id. You want your object names to be explicit and not too long. They cannot start with a number (2x is not valid, but x2 is).

Functions and their arguments

Functions are canned scripts that automate more complicated sets of commands including operations assignments, etc. Many functions are predefined, or can be made available by importing R packages (more on that later).

Example: Function *mean()* will return the mean of a set of numbers:

```
mean(1:5)
[1] 3
```

Vectors and data types

Vectors are one of the many data structures that R uses. Other important ones are lists (list), matrices (matrix), data frames (data.frame), factors (factor) and arrays (array).

Vectors and data types

Vectors are one of the many data structures that R uses. Other important ones are lists (list), matrices (matrix), data frames (data.frame), factors (factor) and arrays (array).

A vector is the most common and basic data type in R, and is pretty much the workhorse of R. A vector is composed by a series of values, which can be either numbers or characters. We can assign a series of values to a vector using the c() function.

Vectors and data types

Vectors are one of the many data structures that R uses. Other important ones are lists (list), matrices (matrix), data frames (data.frame), factors (factor) and arrays (array).

A vector is the most common and basic data type in R, and is pretty much the workhorse of R. A vector is composed by a series of values, which can be either numbers or characters. We can assign a series of values to a vector using the c() function.

For example we can create a vector of weights and assign it to a new object weight:

```
weight <- c(50, 60, 65, 82)
```

R!

Subsetting vectors

If we want to extract one or several values from a vector, we must provide one or several indices in square brackets. For instance:

```
schools <- c("stanford", "harvard", "princeton", "yale")
```

Subsetting vectors

If we want to extract one or several values from a vector, we must provide one or several indices in square brackets. For instance:

```
schools <- c("stanford", "harvard", "princeton", "yale")
schools[1]
```

Subsetting vectors

If we want to extract one or several values from a vector, we must provide one or several indices in square brackets. For instance:

```
schools <- c("stanford", "harvard", "princeton", "yale")
schools[1]
[1] "stanford"
```

What is the result for:

```
schools[c(3, 2)]
```

Subsetting vectors

If we want to extract one or several values from a vector, we must provide one or several indices in square brackets. For instance:

```
schools <- c("stanford", "harvard", "princeton", "yale")
schools[1]
[1] "stanford"
```

What is the result for:

```
schools[c(3, 2)]
[1] "princeton", "harvard"
```

R!

Missing data

As R was designed to analyze datasets, it includes the concept of missing data (which is uncommon in other programming languages). Missing data are represented in vectors as NA.

Missing data

As R was designed to analyze datasets, it includes the concept of missing data (which is uncommon in other programming languages). Missing data are represented in vectors as NA.

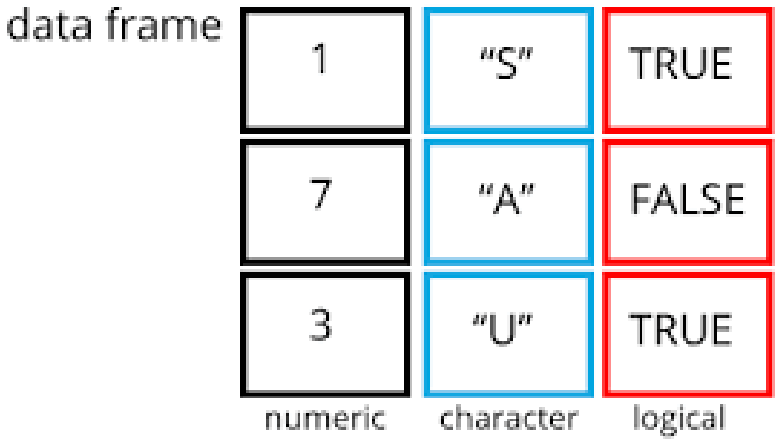
When doing operations on numbers, most functions will return NA if the data you are working with include missing values. This feature makes it harder to overlook the cases where you are dealing with missing data. You can add the argument *na.rm=TRUE* to calculate the result while ignoring the missing values.

```
heights <- c(2, 4, 4, NA, 6)
mean(heights)
```

R!

Data frame

Data frames are the de facto data structure for most tabular data, and what we use for statistics and plotting.



Packages

There are many available packages that provide access to new functions.

- 'install.packages(package-name)' will download a package from one of the CRAN mirrors assuming that a binary is available for your operating system. If you have not set a preferred CRAN mirror in your 'options()', then a menu will pop up asking you to choose a location.
- 'library(package-name)' will load a package so you can use it. It is required at the beginning of each R session.

NEXT

- Analyzing datasets
- Relating variables and creating models
- Homework 1 available today. Due July 4th at 1.30pm.