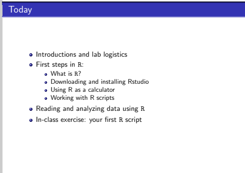
Today

- Introductions and lab logistics
- First steps in R:
  - What is R?
  - Downloading and installing Rstudio
  - Using R as a calculator
  - Working with R scripts
- Reading and analyzing data using R
- In-class exercise: your first R script



Today

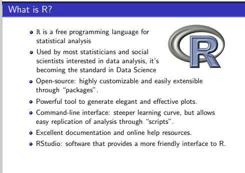
- Introductions and lab logistics
- First steps in R:
  - What is R?
  - Downloading and installing Rstudio
  - Using R as a calculator
  - Working with R scripts
- Reading and analyzing data using R
- In-class exercise: your first R script

Approximate timing of Lab 1

- Introductions and lab logistics = 15 minutes
- First steps in R = 45 minutes (20 minutes for slides, 25 minutes working interactively with R)
- BREAK (10 minutes)
- Reading and analyzing data = 30 minutes
- In-class exercise = 20 minutes

What is R?

- R is a free programming language for statistical analysis
- Used by most statisticians and social scientists interested in data analysis, it's becoming the standard in Data Science
- Open-source: highly customizable and easily extensible through "packages".
- Powerful tool to generate elegant and effective plots.
- Command-line interface: steeper learning curve, but allows easy replication of analysis through "scripts".
- Excellent documentation and online help resources.
- RStudio: software that provides a more friendly interface to R.



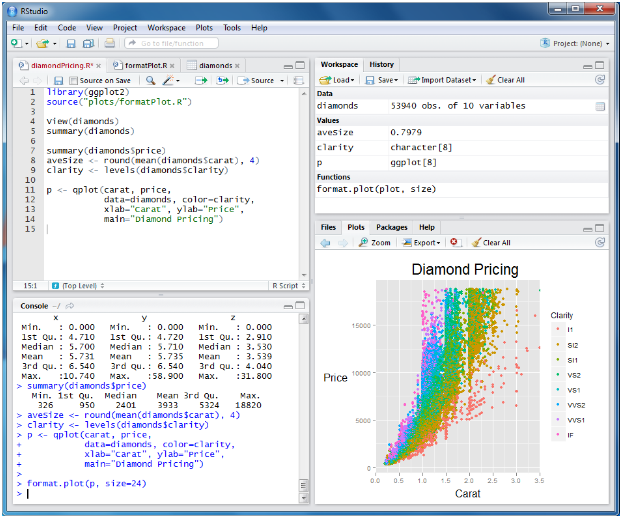
What is R?

- R is a free programming language for statistical analysis
- Used by most statisticians and social scientists interested in data analysis, it's becoming the standard in Data Science
- Open-source: highly customizable and easily extensible through "packages".
- Powerful tool to generate elegant and effective plots.
- Command-line interface: steeper learning curve, but allows easy replication of analysis through "scripts".
- Excellent documentation and online help resources.
- RStudio: software that provides a more friendly interface to R.

A few points to emphasize:

- Initially developed by statisticians, but now language of choice for many researchers in the social and natural sciences
- $> 5K$ packages expand the functionalities of R, allowing us to prepare plots (e.g. maps), capture social media data, fit advanced statistical models... you can even send emails, do Google searches, and read the NY Times.
- We will see some examples of plots today. The NY Times initially prepares all their plots in R before using other tools.
- Command line: very different from other statistical packages students may have learned (SPSS, for example). Emphasize importance of replicability (more on this later).
- Documentation: we will see later how to get help for any command.

RStudio



```r
library(ggplot2)
source("plots/formatPlot.R")

View(diamonds)
summary(diamonds)

summary(diamonds$price)
aveSize <- round(mean(diamonds$carat), 4)
clarity <- levels(diamonds$clarity)

p <- qplot(carat, price,
           data=diamonds, color=clarity,
           xlab="Carat", ylab="Price",
           main="Diamond Pricing")

```

Data

diamonds	53940 obs. of 10 variables

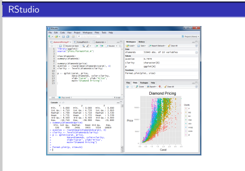
Values

aveSize	0.7979
clarity	character[8]
p	ggplot[8]

Functions

```
format.plot(plot, size)
```

```
       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)
> 
```


RStudio
File Edit Code View Project Workspace Plots Tools Help
diamondPricing.R formatPlot.R diamonds
Source on Save
Source
Workspace History
Import Dataset
Clear All
Files Plots Packages Help
Zoom Export
Diamond Pricing
Price
Carat
Clarity
Console

```
library(ggplot2)
source("plots/formatPlot.R")

View(diamonds)
summary(diamonds)

summary(diamonds$price)
aveSize <- round(mean(diamonds$carat), 4)
clarity <- levels(diamonds$clarity)

p <- qplot(carat, price,
           data=diamonds, color=clarity,
           xlab="Carat", ylab="Price",
           main="Diamond Pricing")
```

```
Data
diamonds      53940 obs. of 10 variables
Values
aveSize       0.7979
clarity       character[8]
p             ggplot[8]
Functions
format.plot(plot, size)
```

```
       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)
>
```

This is what an RStudio session usually looks like.
Four panels:

1. Console: where you type R commands interactively and see the output and error messages. For example, last command was typed to change format of plot.
2. Workspace: shows all objects (data) in memory. In this case, “diamonds” is a dataset with characteristics of diamonds (one of the default datasets in R).
3. Viewer: shows plots produced by R, helps you find files, displays help menus, etc. Here it shows a plot comparing quality and price of diamonds.
4. Script: where you write R code. This is a list of commands that whoever wrote this script typed to create the plot you see on the viewer. We will prepare our own examples later.

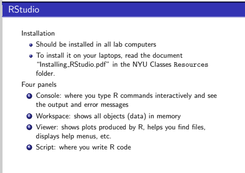
RStudio

Installation

- Should be installed in all lab computers
- To install it on your laptops, read the document "Installing_RStudio.pdf" in the NYU Classes `Resources` folder.

Four panels

1. Console: where you type R commands interactively and see the output and error messages
2. Workspace: shows all objects (data) in memory
3. Viewer: shows plots produced by R, helps you find files, displays help menus, etc.
4. Script: where you write R code



RStudio

Installation

- Should be installed in all lab computers
- To install it on your laptops, read the document “Installing_RStudio.pdf” in the NYU Classes `Resources` folder.

Four panels

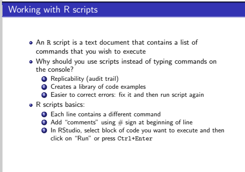
1. Console: where you type R commands interactively and see the output and error messages
2. Workspace: shows all objects (data) in memory
3. Viewer: shows plots produced by R, helps you find files, displays help menus, etc.
4. Script: where you write R code

You can find RStudio by clicking on the “Start” button on the bottom-left corner, and then typing “RStudio”
I will be available after the end of the class in case you need help installing RStudio on your laptops.

Important to make sure that R is installed before RStudio is installed!

Working with R scripts

- An R script is a text document that contains a list of commands that you wish to execute
- Why should you use scripts instead of typing commands on the console?
  1. Replicability (audit trail)
  2. Creates a library of code examples
  3. Easier to correct errors: fix it and then run script again
- R scripts basics:
  1. Each line contains a different command
  2. Add "comments" using # sign at beginning of line
  3. In RStudio, select block of code you want to execute and then click on "Run" or press `Ctrl+Enter`



Working with R scripts

- An R script is a text document that contains a list of commands that you wish to execute
- Why should you use scripts instead of typing commands on the console?
  1. Replicability (audit trail)
  2. Creates a library of code examples
  3. Easier to correct errors: fix it and then run script again
- R scripts basics:
  1. Each line contains a different command
  2. Add "comments" using # sign at beginning of line
  3. In RStudio, select block of code you want to execute and then click on "Run" or press `Ctrl+Enter`

Define scripts. Go back to screen capture of RStudio to show an example. On reasons for using scripts:

1. Replicability: important for academic purposes, but also if you want to run analyses multiple times (e.g. analysis of social media messages mentioning a company, and then do it again for another)
2. Code examples: to avoid forgetting how to do something, you can save it as a script.
3. Correct errors: sometimes we realize we made a mistake 10 minutes ago. We can just go back to that point in the code, fix it, and run the script again.

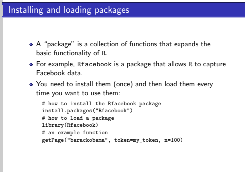
Basics:

1. Commands: one line to open the data, another to summarize it...
2. Comments: very important. Add details about what you do. You'd be surprised how soon you forget what that line of code you wrote does.
3. In Mac, command + enter

Installing and loading packages

- A “package” is a collection of functions that expands the basic functionality of R.
- For example, Rfacebook is a package that allows R to capture Facebook data.
- You need to install them (once) and then load them every time you want to use them:

```
# how to install the Rfacebook package
install.packages("Rfacebook")
# how to load a package
library(Rfacebook)
# an example function
getPage("barackobama", token=my_token, n=100)
```



Installing and loading packages

- A “package” is a collection of functions that expands the basic functionality of R.
- For example, Rfacebook is a package that allows R to capture Facebook data.
- You need to install them (once) and then load them every time you want to use them:

```
# how to install the Rfacebook package
install.packages("Rfacebook")
# how to load a package
library(Rfacebook)
# an example function
getPage("barackobama", token=my_token, n=100)
```

There are more than 5,000 packages, most of them written by the community of R users. They do very different things (mention examples from before).

This code is just an example. We will see more later on in the class.

Note that RStudio allows you to install packages interactively (easier if you only need to install a couple of packages)

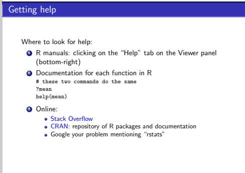
Getting help

Where to look for help:

1. R manuals: clicking on the "Help" tab on the Viewer panel (bottom-right)
2. Documentation for each function in R

```
# these two commands do the same
?mean
help(mean)
```

3. Online:
   - Stack Overflow
   - CRAN: repository of R packages and documentation
   - Google your problem mentioning "rstats"



Getting help

Where to look for help:

1. R manuals: clicking on the “Help” tab on the Viewer panel (bottom-right)
2. Documentation for each function in R

```
# these two commands do the same
?mean
help(mean)
```

3. Online:
   - Stack Overflow
   - CRAN: repository of R packages and documentation
   - Google your problem mentioning “rstats”

TA will be available for any problem you might have, but for quick questions you can use the following resources.

1. R has a built-in manual. You can find it on “help”, then look for “An Introduction to R” and choose the chapter that you're interested in.
2. Each function is documented. Two different ways to find it.
3. Online, Stack Overflow is a repository of questions about R and other programming languages. Do your search with “[r]” to find questions tagged as such. CRAN is where the packages are hosted, and you can find documentation for each them there. Finally, “rstats” is the token by convention to refer to R issues (“r” is too frequent and therefore less google-able).