Introduction

Guy J. Abel

Data Science for Population Studies

- This course is not Population Economics.
  - It does have the course code 3XS371026
  - This course should have been called *Data Science for Population Studies*
  - It is going to be very useful for *3XS371022 - Statistical Methods for Population Studies*
- The course book is *R for Data Science* by Garrett Grolemund and Hadley Wickham.
  - Available for free online: `http://r4ds.had.co.nz/`
  - The webpage and book were written using R and RStudio!
    - These slides were written using R and and RStudio!
- Hard copies of the book are on their way.

Data Science for Population Studies

- Course Outline
  - Part 0: Introduction
  - Part 1: R Basics
  - Part 2: Visualizing Demographic Data
  - Part 3: Handling Demographic Data
  - Part 4: Simple Demographic Projections
  - Part 5: Mapping Demographic Data
- Folder with slides, R code, exercises and solutions
  - `https://pan.baidu.com/s/1bpKx2TT`
  - Password: `a3gf`

Data Science for Population Studies

- (Approximate) Teaching Schedule
  - Week 1-2: Part 0 and 1
  - Week 3-4: Part 2
  - Week 5-7: Part 3
  - Week 7: Part 4
  - Week 8: Part 5
- Exercises
  - In-class exercises in Part 1-4.
- Assessment
  - Assignments for Parts 1-4 (40% each)
  - Final Coursework (60%)

Why use R.

- Free. Open source.
- Leading software in statistics
- Cross-platform (Windows, Mac, Linux).
- Expanding number of packages making R programming more flexible, fast and simpler.
- Large, active, helpful and friendly user base.
- Many different data structures
- Many different graphics options with complete control.
- Reproduce analyses.

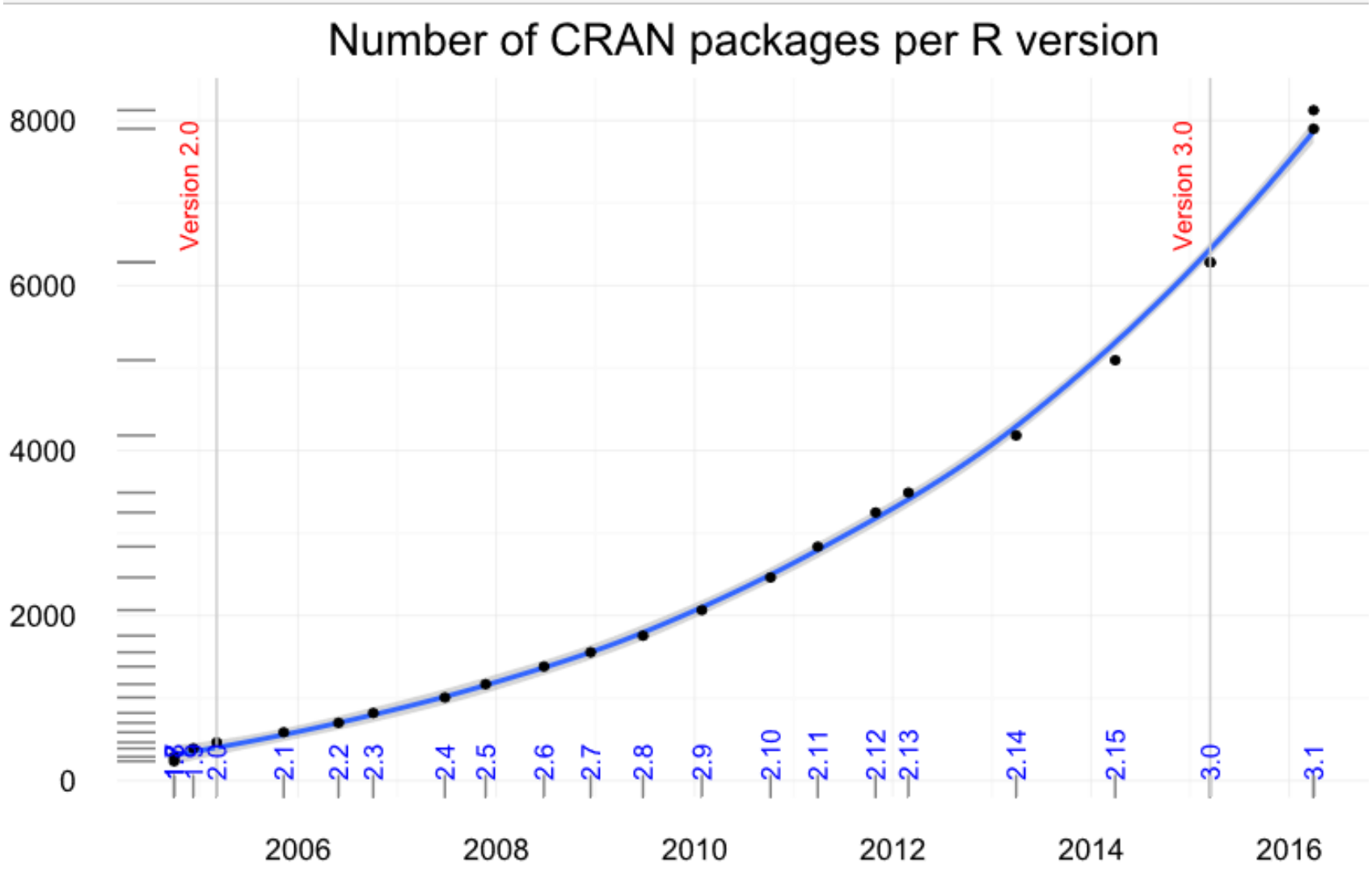
R History

- S language developed by John Chambers and others at Bell Labs.
- A commercial version of S with additional features was developed and marketed as S-Plus
- R was first written as a research project by Ross Ihaka and Robert Gentleman to be 'not unlike' S.
- R and S-Plus can best be viewed as two implementations of the S language.
- It is now under active development by a group of statisticians called 'the R core team', with a home page at www.r-project.org.

R History

- 1993 Research project in Auckland, NZ
- 1995 R Released as open-source software
- 1997 R core group formed
- 2000 R 1.0.0 released
- 2003 R Foundation founded
- 2004 First international user conference in Vienna
- 2015 R Consortium founded

CRAN

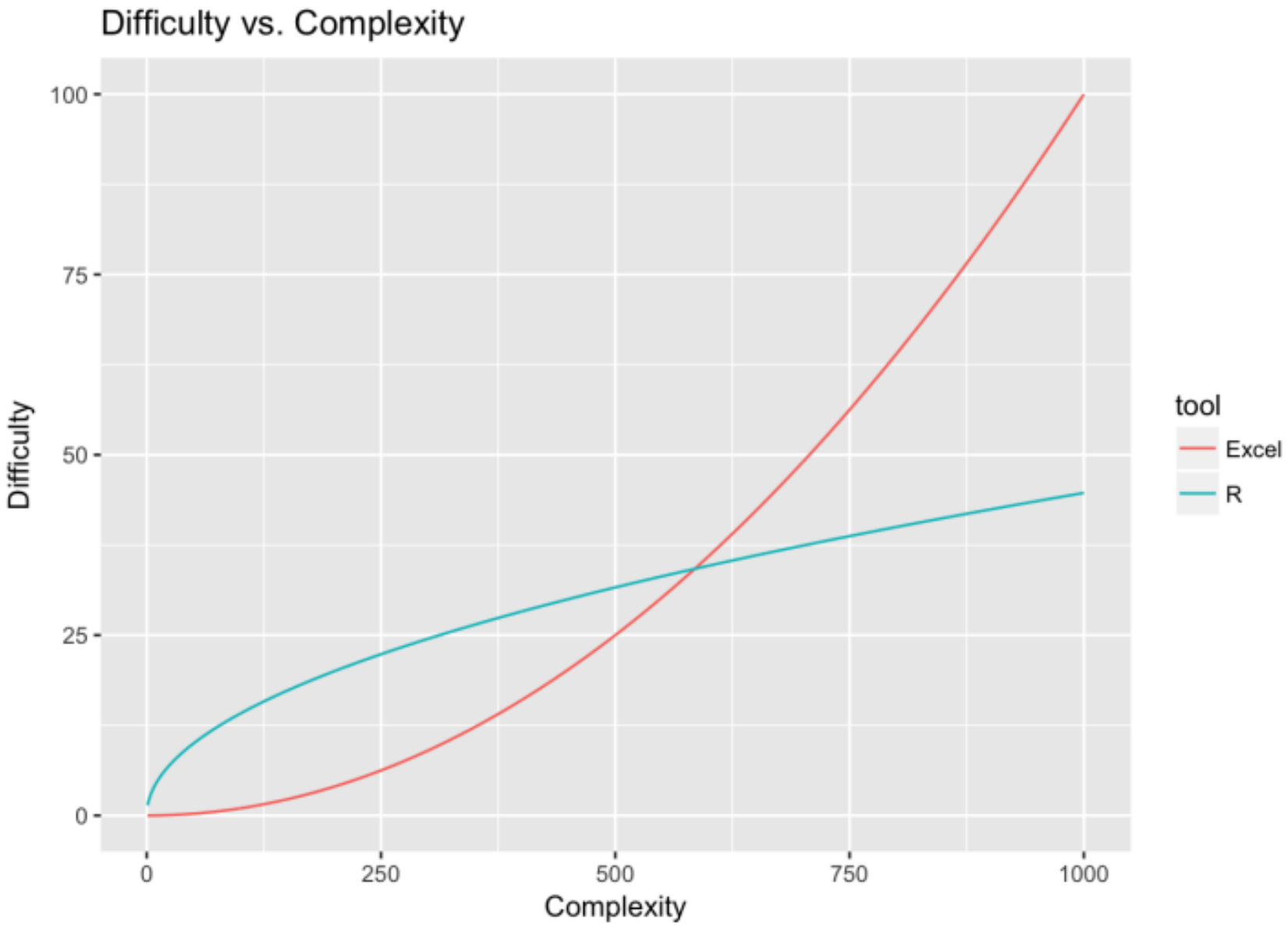


Source: Andrie de Vries

What is the catch?

- Not a spreadsheet (e.g. Excel). So you do not 'see' what's going on.
- There is no real GUI (i.e. no point-and-click interface).
  - If you want to fit a model in R or create a plot you will need to write R code.
- Somewhat steep learning curve.
- Comes with ABSOLUTELY NO WARRANTY.

Learning Curve



Installing R

1. Visit the Comprehensive R Archive Network (CRAN) website at http://cran.r-project.org/.
2. At the top of the page click the appropriate link for your operating system
   - Windows: Click on the link to the base sub-directory and then on Download R [version xxx] for Windows.
   - Mac OS X: Click the first link under the 'Files' heading and download the file: [R-xxx.pkg].
3. Unless you have chosen to compile R from source, run the executable you just downloaded.

- R is updated regularly (roughly 3 times a year).
  - These updates, contain improvements, bug fixes, and new features.
- Newly developed or updated packages also often are not backward compatible.
  - Make sure you keep R up-to-date.

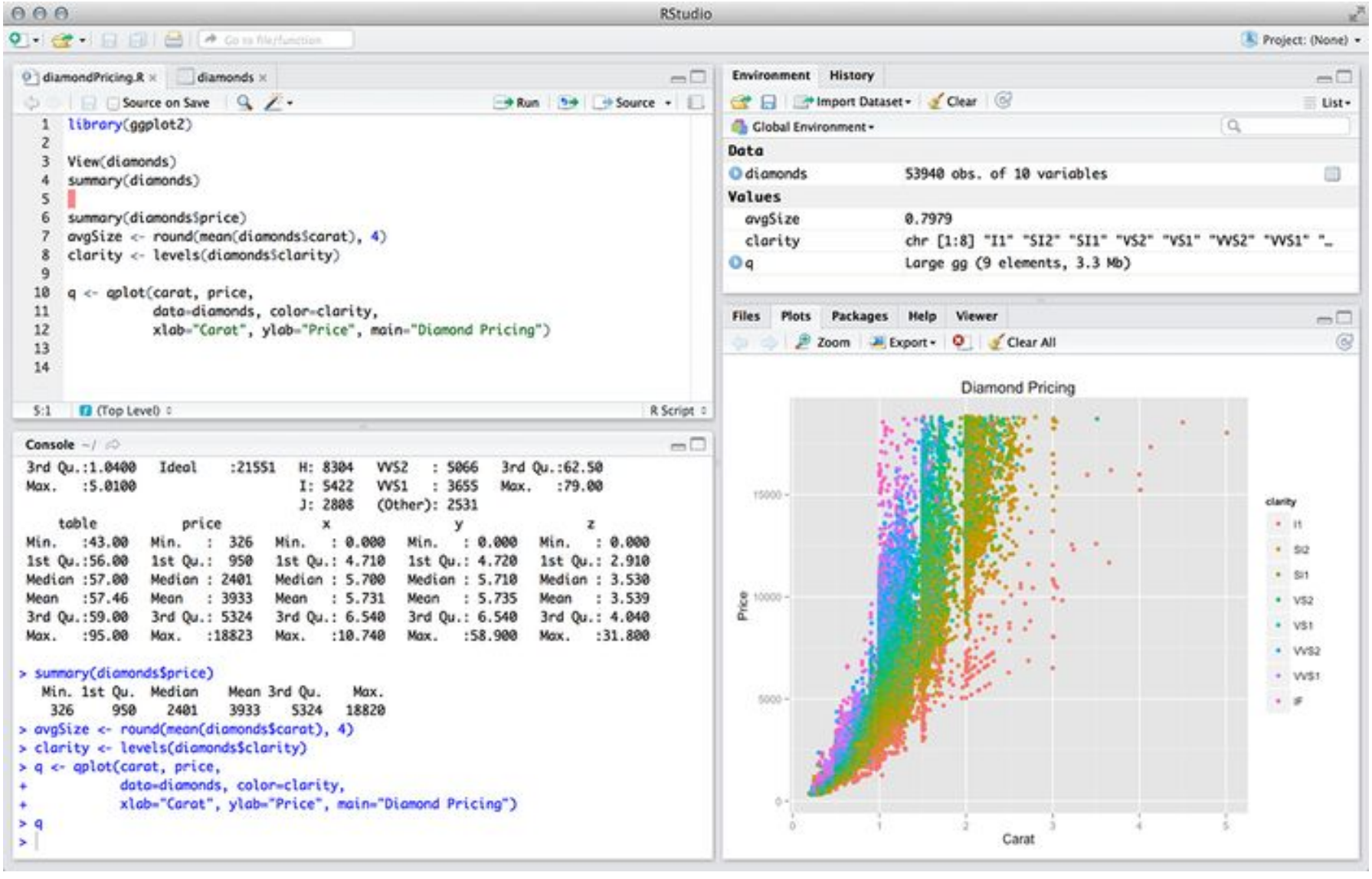
Text Editors

- The R GUI is limited when it comes to doing actual work, writing programs, and maintaining your code.
- R is a command line interpreter
  - Not a text editor
  - Not full-featured statistical software.
  - Is a language to interpret your inputs.
- It does not care where those inputs come from, how you entered them, or if you saved them.
  - We can use R code scripts write our R code in.
  - We can send the R code scripts to run at the command line
  - We can save the R code scripts for future use.
- Using R code scripts protects us if something happens (R crashes, power goes out, or you close your R-session without saving)
- There are a many front end programs that allow you to write your code and feed it to R
  - The most popular is RStudio https://www.rstudio.com/products/rstudio/download/

RStudio

- RStudio does what the original version of R does. It also has a menu system with some extra wizards to
  - Installing and updating packages;
  - Monitoring, saving and loading work-spaces;
  - Importing and exporting data;
  - Browsing and exporting graphics;
  - Browsing files and documentation
- RStudio can also interface with Git(Hub), shinapps.io, RPubs and more.
- RStudio divides its world into four panels.
  - Several of the panels are further subdivided into multiple tabs.
  - Which tabs appear in which panels can be customized by the user.

RStudio


RStudio
diamondPricing.R
diamonds
Source on Save
Run
Source
1 library(ggplot2)
2
3 View(diamonds)
4 summary(diamonds)
5
6 summary(diamonds$price)
7 avgSize <- round(mean(diamonds$carat), 4)
8 clarity <- levels(diamonds$clarity)
9
10 q <- qplot(carat, price,
11 data=diamonds, color=clarity,
12 xlab="Carat", ylab="Price", main="Diamond Pricing")
13
14
5:1 (Top Level) R Script
Console ~/
3rd Qu.:1.0400 Ideal :21551 H: 8304 VVS2 : 5066 3rd Qu.:62.50
Max. :5.0100 I: 5422 VVS1 : 3655 Max. :79.00
J: 2808 (Other): 2531
table price x y z
Min. :43.00 Min. : 326 Min. : 0.000 Min. : 0.000 Min. : 0.000
1st Qu.:56.00 1st Qu.: 950 1st Qu.: 4.710 1st Qu.: 4.720 1st Qu.: 2.910
Median :57.00 Median : 2401 Median : 5.700 Median : 5.710 Median : 3.530
Mean :57.46 Mean : 3933 Mean : 5.731 Mean : 5.735 Mean : 3.539
3rd Qu.:59.00 3rd Qu.: 5324 3rd Qu.: 6.540 3rd Qu.: 6.540 3rd Qu.: 4.040
Max. :95.00 Max. :18823 Max. :10.740 Max. :58.900 Max. :31.800
> summary(diamonds$price)
Min. 1st Qu. Median Mean 3rd Qu. Max.
326 950 2401 3933 5324 18820
> avgSize <- round(mean(diamonds$carat), 4)
> clarity <- levels(diamonds$clarity)
> q <- qplot(carat, price,
+ data=diamonds, color=clarity,
+ xlab="Carat", ylab="Price", main="Diamond Pricing")
> q
Environment History
Import Dataset Clear List
Global Environment
Data
diamonds 53940 obs. of 10 variables
Values
avgSize 0.7979
clarity chr [1:8] "I1" "SI2" "SI1" "VS2" "VS1" "VVS2" "VVS1" "_
q Large gg (9 elements, 3.3 Mb)
Files Plots Packages Help Viewer
Zoom Export Clear All
Diamond Pricing
Price
Carat
clarity
I1
SI2
SI1
VS2
VS1
VVS2
VVS1
IF

Bottom Left (Console)

- Commands entered in the Console tab are immediately executed by R.
- Very similar to the original version of R.
- R can work as a simple calculator (more on these later), for example we can type

```
> 5 + 3
[1] 8
> 15.3 * 23.4
[1] 358.02
> sqrt(16)
[1] 4
```

- The console, like the standard R program, does not care where those inputs come from, how you entered them, or if you saved them.

Top Left (Editor)

- R commands can be stored in a file.
  - RStudio provides an editor for editing R scripts running parts of or the whole code.
- To create a R script file select `File | New File | R Script` from the RStudio menu.
  - Then type the following into the R script (`Untitled1`)

```
> 5 + 3
[1] 8
> 15.3 * 23.4
[1] 358.02
> sqrt(16)
[1] 4
```

Top Left (Editor)

- To run a single line of code:
  - Place cursor
  - Press the `Run` button, `Ctrl + R` or `Ctrl + Enter`
- To run multiple lines of code:
  - Highlight lines using cursor or using `Shift` + $\rightarrow/\leftarrow/\uparrow/\downarrow$/`Page Up`/`Page Down`
  - Press the `Run` button, `Ctrl + R` or `Ctrl + Enter`
- To run all the code
  - Highlight all lines using cursor or using `Ctrl + a`
  - Press the `Run` button, `Ctrl + R` or `Ctrl + Enter`

Top Left (Editor)

- We can open pre-saved scripts. There are some scripts for each of the slides.
- Open the `s00_intro.R` file I gave you.
  - Some of the R script is commented out (with the #, more on that later) so that I can make the slides.
  - The first few lines are also to do with making the slides (please ignore)
  - Most of the time you can run the commented out code too.
  - Try running this line now from the `s00_intro.R` file.

```
> print("Hello  world!")
[1] "Hello  world!"
```

Top Right

1. Workspace Tab
   - Shows the objects available to the console.
   - These are subdivided into data, values (non-data frame, non-function objects) and functions.
   - Clicking on a data frame will open it a data viewer.
   - Clicking on other objects will open them in a small editor so you can 'fix' them.

```
> # create some objects to view
> x <- 1:10
> d <- cars
```

2. History Tab
   - Commands entered to the console are saved in the History tab.
   - Histories can be saved and loaded, there is a search feature to locate previous commands, and individual lines or sections can be transferred back to the console.
   - Messy version of scripts.

Bottom Right

1. Files Tab
   - Simple file manager to open, move, rename, and delete files.
2. Plots Tab
   - Displays plots created in the console.
   - Navigate to previous plots and also export plots in various formats after interactively re-sizing them.

```
> # example plot of the cars data set (speed vs stopping distances)
> plot(cars)
```

Bottom Right

3. Packages Tab
   - Already mentioned that R has many 1000's of packages.
   - Only 7 are loaded automatically when you open R.
     - You can view these using the `Global Enviroment` drop-down at the top of the Environment tab
   - The Packages tab facilitates installing and loading packages. (Bit more on this later).
   - It will also allow you to search for packages that have been updated since you installed them.
4. Help Tab
   - Displays R help files.
   - These can be searched and navigated
   - You can also open a help file using the ? operator in the console. (Bit more on the next few slides)

```
> # view the help file for the logarithm function:
> ?log
```

Help

- When you first meet a new function it is useful to look at the help file
  - Understand the possible inputs
  - Understand the default inputs.
- As in the previous slide we can simply pull up a help page for the `log()` function like this:

```
> ?log
```

- For some functions, especially basic operators, `?` may not work. In those cases you can use the `help()` function:

```
> help("+")
```

Help

A help page will appear with the following sections (there may be others)

- `Description`: purpose of the function
- `Usage`: an example of a typical implementation
- `Arguments`: a list of the arguments you can supply to the function and what each does
- `Details`: more detailed information about the function and its arguments
- `Value`: information about the likely output of a function (e.g. does the function return an `integer`, or a `list`, or a `matrix`, or something different)
- `See Also`: a list of useful related functions
- `References`: citations which can often be very useful
- `Examples`: example code

Help

- However, sometimes R programmers are not always good at explaining things in plain English.
- The `example()` function runs examples from the help page in the R console:

```
> example("log")

log> log(exp(3))
[1] 3

log> log10(1e7) # = 7
[1] 7

log> x <- 10^-(1+2*1:9)

log> cbind(x, log(1+x), log1p(x), exp(x)-1, expm1(x))
           x
 [1,] 1e-03 9.995003e-04 9.995003e-04 1.000500e-03 1.000500e-03
 [2,] 1e-05 9.999950e-06 9.999950e-06 1.000005e-05 1.000005e-05
 [3,] 1e-07 1.000000e-07 1.000000e-07 1.000000e-07 1.000000e-07
 [4,] 1e-09 1.000000e-09 1.000000e-09 1.000000e-09 1.000000e-09
 [5,] 1e-11 1.000000e-11 1.000000e-11 1.000000e-11 1.000000e-11
 [6,] 1e-13 9.992007e-14 1.000000e-13 9.992007e-14 1.000000e-13
 [7,] 1e-15 1.110223e-15 1.000000e-15 1.110223e-15 1.000000e-15
 [8,] 1e-17 0.000000e+00 1.000000e-17 0.000000e+00 1.000000e-17
```

Help

- If you know what function you want you can search all the help files of the installed R packages using `??`.

```
> # is there a median function in R?
> ??median
```

- The official manual for R is pre-installed. You can open it using the `help.start()` function.

```
> help.start()
```

- It is very big and not always in plain English
- The web is usually the best resource after the help file.
  - R is very popular. A lot of people have learnt how to use R, so there is lots of help available.
  - More often than not the `[r]` tag on Stack Overflow is the best place to start. `http://stackoverflow.com/questions/tagged/r`

Packages and Libraries

- Packages can be installed using the `install.packages()` function
  - There are over 10,000 R packages, constantly expanding the functionality of R.
- For example the `readxl` package to import data from excel, you run the following command:

```
> install.packages("readxl")
```

- R will connect to a CRAN mirror and download and install the package to your computer.
  - This will be a **one** time only operation.
  - Packages sometimes depend on other packages. These other packages will be downloaded automatically.
  - In RStudio you can select your mirror from the menu via `Tools | Global Options | Packages` (generally the closer the faster)

Package Teething Problems

- Installing packages might not work for Windows users if your username has non-latin characters
  1. Uninstall RStudio
  2. Create a new Windows user (with latin characters only)
  3. Login to windows with new user
  4. Install RStudio with access for all users.
- Installing packages might not work for Windows users if user account does not have administrative rights.
  1. Get administrative rights, or...
  2. `Control Panel -> User Accounts -> User Accounts -> Change my environment variables -> New`
     Enter:
     `Variable name: R_LIBS_USER`
     `Variable value: C:/software/Rpackages` (an example, should be where you want (and have permission) to store packages.

Installing and Loading Packages

- Once installed, R will still be clueless.
  - Package libraries can be loaded to your R session using the the `library()` function.
  - You will need to do this **each** time you start an R session.
  - This keeps R slim (only 7 packages are loaded by default)
- For example to use the `read_excel()` function:

```
> # i heard about read_excel on the web.. what does it do...
> ?read_excel
>
> # oh.. i need to load the library of the read_excel function first
> library(readxl)
> ?read_excel
```

Maintaining Libraries

- Packages are frequently updated. Depending on the developer this could happen very often.
- To keep your packages updated enter this every once in a while:

```
> update.packages()
```

- Also possible via the RStudio menu `Tools | Update Packages`

R News

- R is growing fast. Plenty of sources online to
  - Keep track of developments and new packages
  - Learn about the many other possible things you can do with R not covered in this course
- Some of my favorite places that I follow are:
  - R-Bloggers collects blog post of R users - `https://www.r-bloggers.com/`
    - Has an email list that can send you updates.
  - Twitter has a number of good accounts if you can get access.
    - `#rstats` for R community members
    - `@RLangTip` account for one tip per day
  - Reddit rstats - `https://www.reddit.com/r/rstats/`
  - The R Journal - `https://journal.r-project.org/`

General Points

- R itself will be updated frequently.
- The `installr` package is very useful when you do have to update to a new version of R.
  - Works outside RStudio, in the original R version.
  - Brings up a series of pop-up boxes on what to download and copy across to the new version.

```
> # install.packages("installr")
> library(installr)
> installr()
```

General Points

1. Please be patient. Teaching classes using R is never smooth.
   - Everyone has different computers, R versions, package versions, etc, etc,...
2. Throughout the course I will stop the slides and ask you to carry out some exercises. The pain and frustration that you feel when you start with R is natural. Be patient and embrace the frustration. Learning is not easy, especially a new language that does not accept typos.
3. The exercise solutions are in the folder. Try not to peak at them unless you are really desperate. If you fall behind, spend some extra time to keep working on the exercises. Making mistakes, looking for solutions on the web and experimenting is the best way to learn R.
4. If you have not already, try and install the latest versions of R and RStudio on your laptop.
   - Make sure you can install an R package (e.g. `readxl`)
   - Make sure that when you re-open R you do not need to re-install a package you previously installed, i.e. `library(readxl)` works as soon as you open R.
   - If you have troubles come and see me.