Course Introduction

Guy J. Abel

Data Analysis and Visualization in R for Beginners

- Course Title: Data Analysis and Visualization in R for Beginners
- Aim: Introduce the R programming language for importing, handling and visualising data.
- Main messages
  - R has many packages that provide endless potential ways to collect, organise, model and visualise data.
  - Writing code in scripts is a much more efficient way to conduct data related research than point and click based approaches.
  - The tidyverse set of packages provide many useful tools for importing, handling and visualising data in R, using a relatively simple syntax.
  - Getting comfortable with R usually requires a lots of practise, but opens up a great number of new tools beyond the reach of most other statistical software.

Data Analysis and Visualization in R for Beginners

- Course Outline
  - Part 0: Course Introduction
  - Part 1: Introduction to R
  - Part 2: Visualizing Data
  - Part 3: Handling Data - Data Importing
  - Part 4: Handling Data - Data Wrangling
  - Part 5: Wrap Up - R Resources
- Handout folder with slides, code in slides, exercises and exercise solutions.

R History

- R evolved from the 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.
- R is continually developed by a group of statisticians called 'the R core team', see `http://www.r-project.org`.

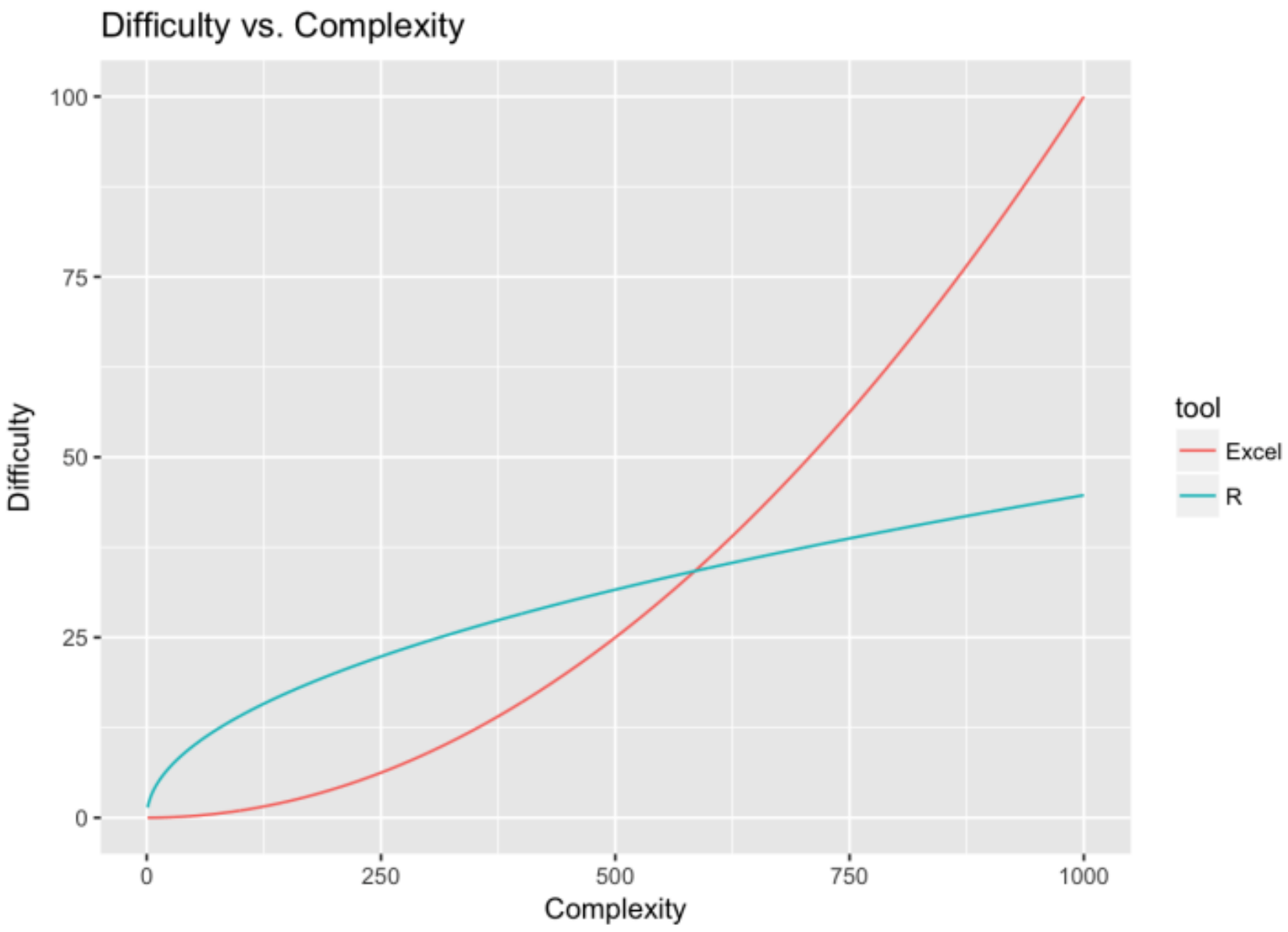
Why use R?

- R has become very popular over recent years for a number of reasons:
  - Free
  - Open source; many packages
  - Many different data structures
  - Many different graphics options with complete control.
  - Leading software in statistics
  - Reproduce analyses
  - Large, active, helpful and friendly user base.

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

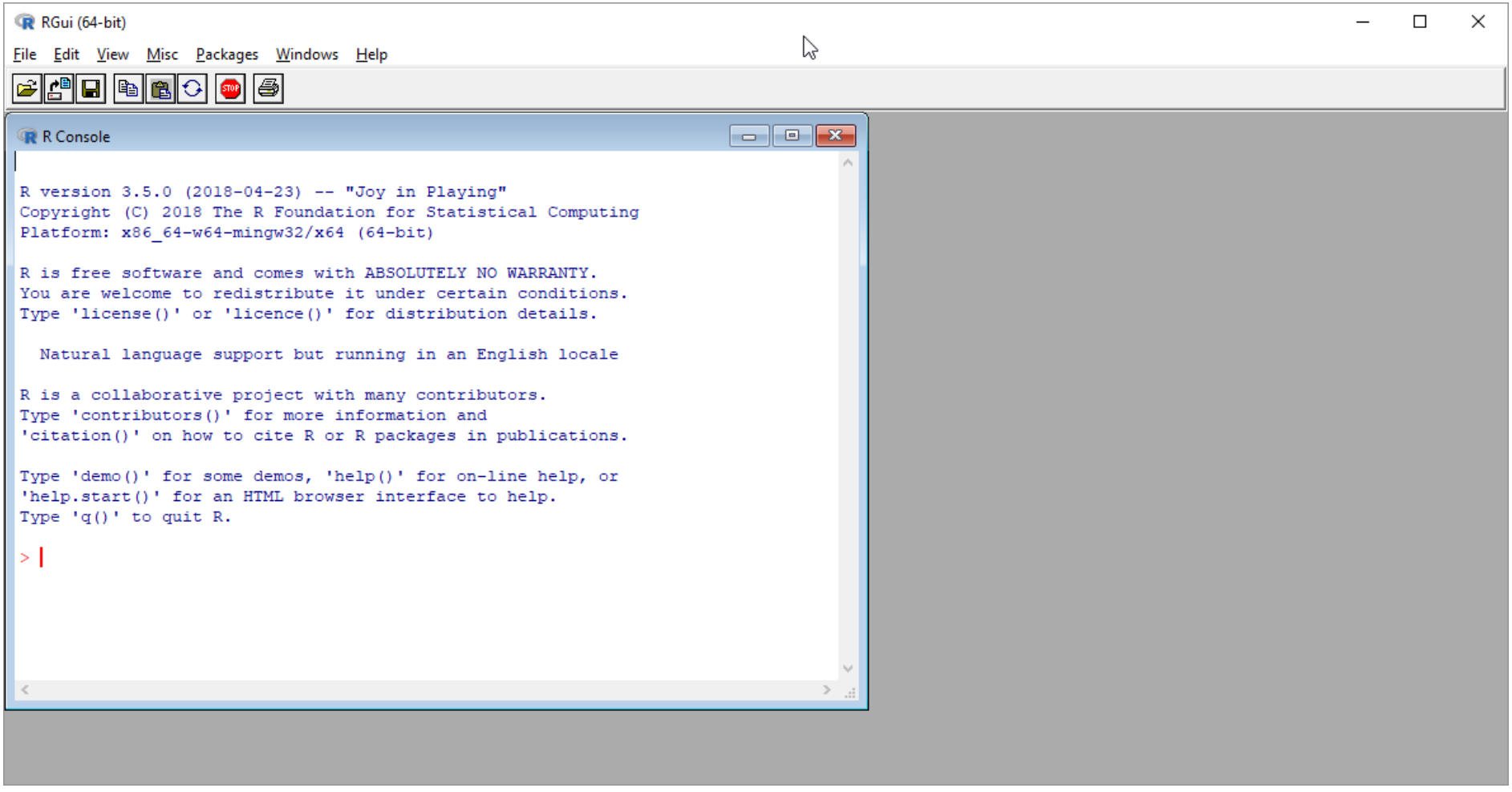


Installing R

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. Run the downloaded `.exe`.
   - Default setup is usually fine - click `OK` and `Agree` lots

- 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.

R GUI


RGui (64-bit)
File Edit View Misc Packages Windows Help
R Console
R version 3.5.0 (2018-04-23) -- "Joy in Playing"
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
Natural language support but running in an English locale
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
>

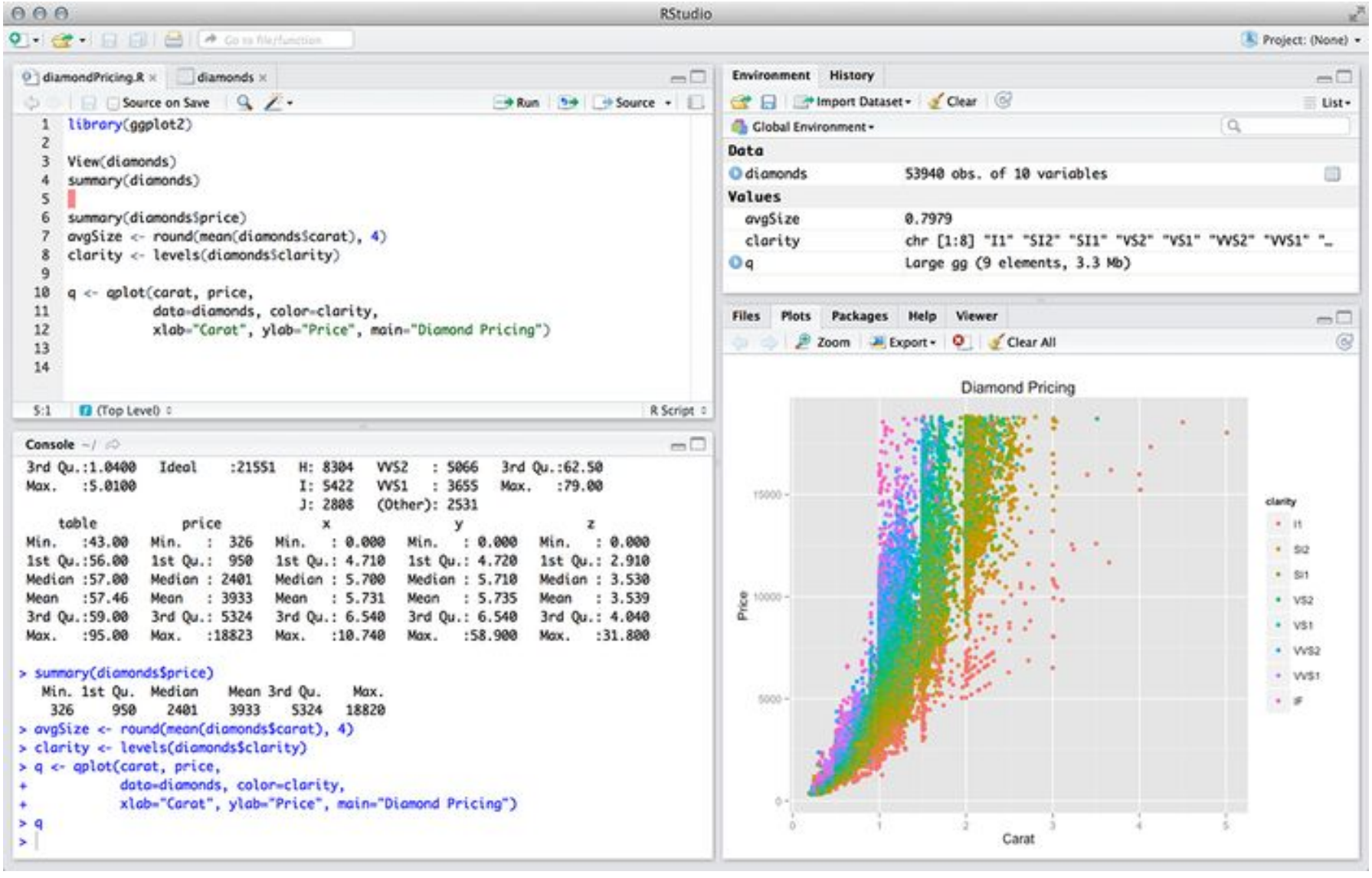
Text Editors

- 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.
  - It's a language to interpret your inputs.
- R 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
    `http://www.rstudio.com/products/rstudio/download/`

RStudio

- RStudio does what the original version of R does, but has a more advanced layout of windows to help with:
  - Editing code including automatic colour highlights, tab spacing, auto-completion, spell check on text,...
  - Monitoring, saving and loading work-spaces;
  - Viewing history of past commands
  - Browsing files and documentation
  - Viewing and saving plots
  - Installing and updating packages
- Menu system also has many other useful features such as find and replace text, changing the font size, themes and arrangement of windows, building R packages
- Interface with Git(Hub), shinyapps.io, RPubs and more.
- Download from `https://rstudio.com/products/rstudio/download/#download`
  - Default setup is usually fine - click `OK` and `Agree` lots
- 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` + →/←/↑/↓/`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 these line now from the `s00_intro.R` file.

```
> # simple commands
> 1 + 1
[1] 2
> mean(x = c(1,5,6))
[1] 4
```

Top Right

1. Work space 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
   - R has 1000's of packages. Packages contain extra functions.
   - 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 mean function:
> ?mean
```

Help

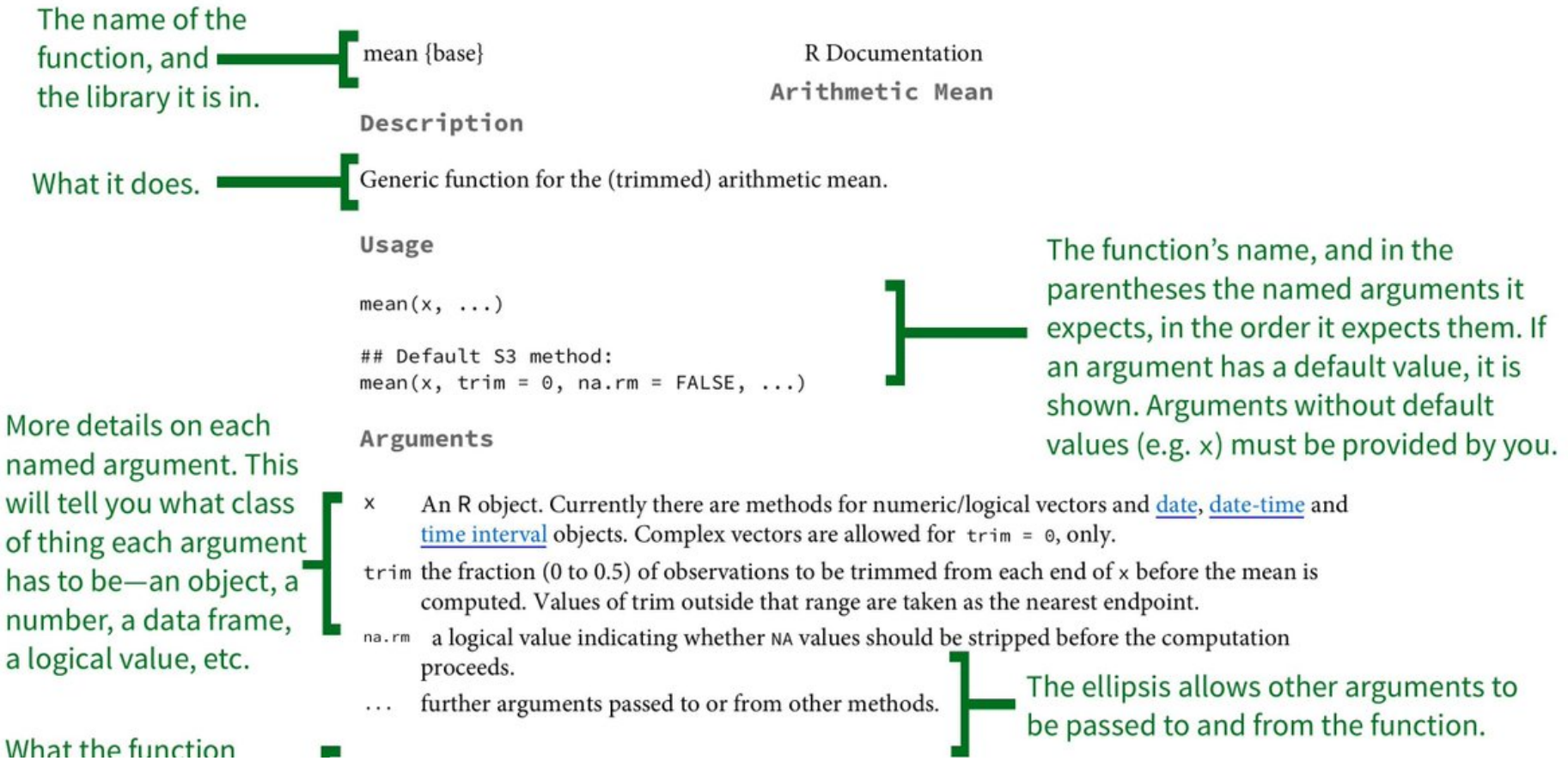
- When you first meet a new function it is useful to look at the help file
  - Understand the possible inputs
  - Understand the default inputs.
- As in the previous slide we can simply pull up a help page for the `log()` function like this:

```
> ?log
```

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

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

Help


The name of the function, and the library it is in.
mean {base}
R Documentation
Arithmetic Mean
Description
What it does.
Generic function for the (trimmed) arithmetic mean.
Usage
mean(x, ...)
Default S3 method:
mean(x, trim = 0, na.rm = FALSE, ...)
The function's name, and in the parentheses the named arguments it expects, in the order it expects them. If an argument has a default value, it is shown. Arguments without default values (e.g. x) must be provided by you.
More details on each named argument. This will tell you what class of thing each argument has to be—an object, a number, a data frame, a logical value, etc.
Arguments
x An R object. Currently there are methods for numeric/logical vectors and date, date-time and time interval objects. Complex vectors are allowed for trim = 0, only.
trim the fraction (0 to 0.5) of observations to be trimmed from each end of x before the mean is computed. Values of trim outside that range are taken as the nearest endpoint.
na.rm a logical value indicating whether NA values should be stripped before the computation proceeds.
... further arguments passed to or from other methods.
The ellipsis allows other arguments to be passed to and from the function.
What the function

Help

What the function returns—i.e., the result of whatever operation or calculation it performs. This can be a single number, as here, or a multi-part object such as a list, a data frame, a plot, or a model.

Value

If `trim` is zero (the default), the arithmetic mean of the values in `x` is computed, as a numeric or complex vector of length one. If `x` is not logical (coerced to numeric), numeric (including integer) or complex, `NA_real_` is returned, with a warning.
If `trim` is non-zero, a symmetrically trimmed mean is computed with a fraction of `trim` observations deleted from each end before the mean is computed.

References

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.

See Also

`weighted.mean`, `mean.POSIXct`, `colMeans` for row and column means.

Other related functions

Examples

```
x <- c(0:10, 50)
xm <- mean(x)
c(xm, mean(x, trim = 0.10))
```

Self-contained examples that you can run at the console. These may use built-in datasets or other R functions.

[Package *base* version 3.4.3 Index]

Visit the package's Index page to look for Demos and Vignettes detailing how it works.

Help

- 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 learned how to use R, so there is lots of help available.

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)
- Please install the tidyverse package as soon as possible (if you have not already done so):
  - We will use it in the next lesson, takes sometime to download if internet connection is slow.

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

Package Teething Problems

- Installing packages might not work for Windows users if your username has non-Latin characters. Either change `R_LIBS_USER` (see below) or
  a) Uninstall RStudio
  b) Create a new Windows user (with Latin characters only)
  c) Log-in to windows with new user
  d) 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.
- Trouble with installing packages that are using the source code as bing updtated on CRAN
  - `install.packages("packages_name", type="binary")`

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`

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 we will work on lots of exercises. You might get stuck. 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.

General Points

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.
   - Let me know if you have a problem with installing packages.
5. If you having trouble installing R, RStudio or packages I have set up an RStudio cloud project where everything is installed for you:
   - `https://rstudio.cloud/project/1593361`
   - Need to create a RStudio account
   - Save changes using button on top right