Paul Intrevado

Exploratory Data Analysis with R

Summer 2018

Table of Contents

1. A Brief Introduction
2. Introduction to R
3. RMarkdown

Section 1

A Brief Introduction

The Warden

Who Am I?

- Ph.D. Operations Management (McGill University)
  - Research focused on service operations
  - Model and solve optimization problems (e.g., MIPs)
- B.Sc., M.Sc. Industrial Engineering (Purdue University)
- B. Commerce (McGill University)
- Assistant Prof @ USF as of August 2014
- I teach or have taught
  - MSDS 593 - Exploratory Data Analysis with R
  - MSDS 601 - Linear Regression Analysis
  - MSDS 605/625/627/632 - Practicum
  - CS 686 - Machine Learning
  - BSDS 100 - Introduction to Data Science with R
- MS Data Science Practicum Director [2014-2017]
- Founding Associate Director of Data Institute [2016-2017]

What I Do

$$\min_{\mathbf{y},\mathbf{z}^+,\mathbf{z}^-} \sum_{j\in\mathbb{J}}\sum_{\nu\in\mathbb{V}}\sum_{t\in\mathbb{T}} r^{(t-1)}\left[\sum_{\omega\in\Omega}\left[f^+_{j\nu\omega}z^+_{j\nu\omega t} - f^-_{j\nu\omega}z^-_{j\nu\omega t}\right] + \gamma_\nu\left(\kappa_{j\nu t} - \rho_{j\nu t} + \sum_{i\in\mathbb{I}} y_{ij\nu t}\right)\right] \quad (4.1)$$

subject to

$$\kappa_{j\nu t} = \kappa_{j\nu(t-1)} + \sum_{\omega\in\Omega} C_{\nu\omega}\left(z^+_{j\nu\omega t} - z^-_{j\nu\omega t}\right) \qquad \forall\ j\in\mathbb{J}, \nu\in\mathbb{V}, t\in\mathbb{T} \quad (4.2)$$

$$\rho_{j\nu t} = \rho_{j\nu(t-1)} + \sum_{\omega\in\Omega} C_{\nu\omega}\left(z^+_{j\nu\omega t} - z^-_{j\nu\omega t}\right) - \sum_{i\in\mathbb{I}} y_{ij\nu(t-1)} + \alpha_{j\nu(t-1)} - \psi^{(\nu+1\to\nu)}_{j(t-1)} + \psi^{(\nu\to\nu-1)}_{j(t-1)}$$
$$\forall\ j\in\mathbb{J}, \nu\in\mathbb{V}, t\in\mathbb{T} \quad (4.3)$$

$$\alpha_{j\nu t} = \mu_{\nu t}\left(\kappa_{j\nu t} - \rho_{j\nu t} + \sum_{i\in\mathbb{I}} y_{ij\nu t}\right) \qquad \forall\ j\in\mathbb{J}, \nu\in\mathbb{V}, t\in\mathbb{T}\setminus|\mathbb{T}| \quad (4.4)$$

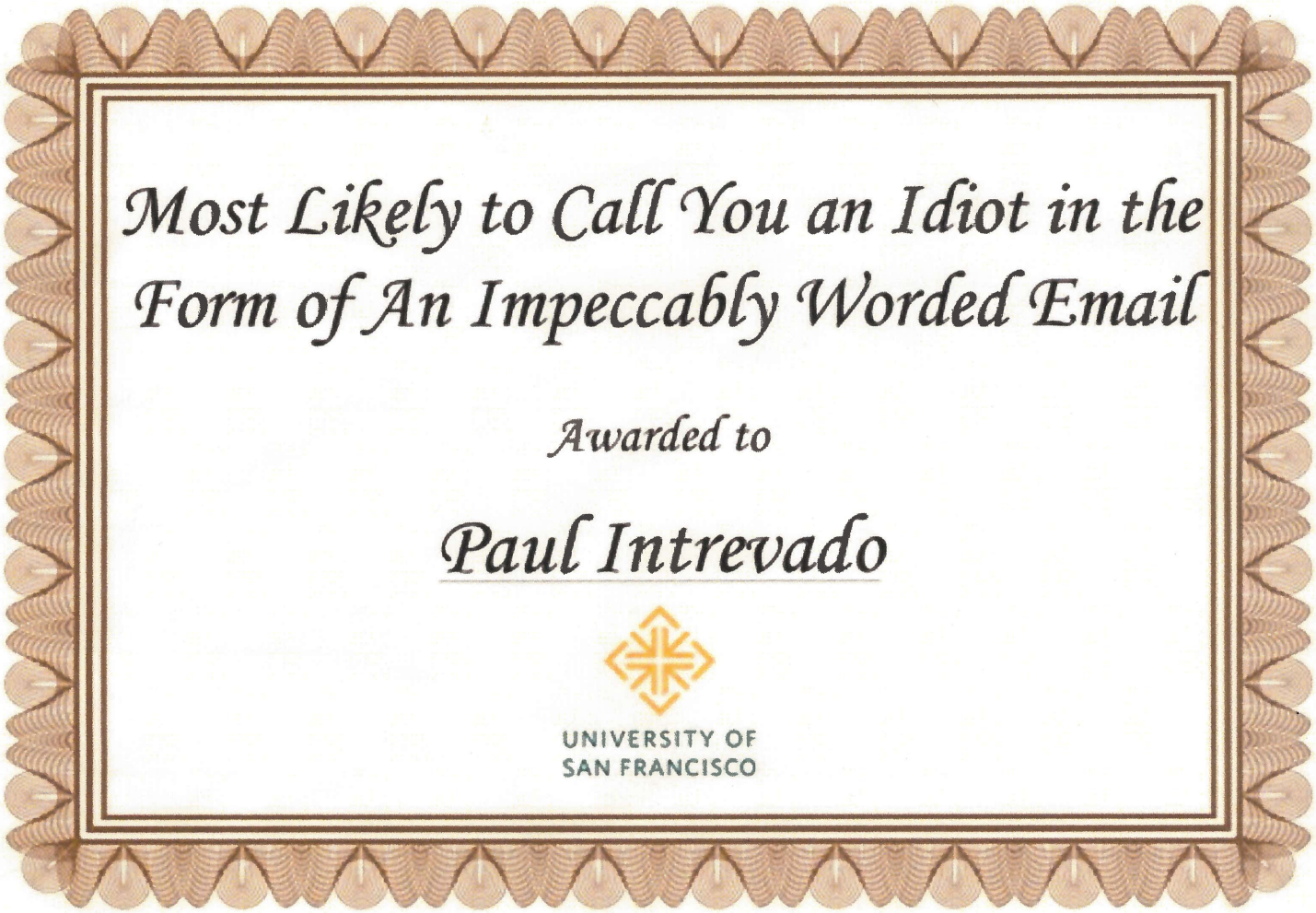
$$\psi^{(\nu\to\nu-1)}_{jt} = \tau^{(\nu\to\nu-1)}_t\left(\kappa_{j\nu t} - \rho_{j\nu t} + \sum_{i\in\mathbb{I}} y_{ij\nu t} - \alpha_{j\nu t}\right) \qquad \forall\ j\in\mathbb{J}, \nu\in\mathbb{V}\setminus\{1\}, t\in\mathbb{T}\setminus|\mathbb{T}| \quad (4.5)$$

$$\eta_{\nu t}\lambda_{it} + \psi^{(\nu+1\to\nu)}_{i(t-1)} = \sum_{j\in\mathbb{J}} y_{ij\nu t} \qquad \forall\ i\in\mathbb{I}, \nu\in\mathbb{V}, t\in\mathbb{T} \quad (4.6)$$

$$\eta_{\nu t}\lambda_{it}\pi_{i\nu t} + \psi^{(\nu+1\to\nu)}_{i(t-1)} \le y_{ij\nu t} \qquad \forall\ i=j\in\mathbb{J}, \nu\in\mathbb{V}, t\in\mathbb{T} \quad (4.7)$$

Awards from Alumni

Awards from Alumni



Awards from Alumni

Software Programming Experience

- C / C++
- MATLAB
- CPLEX
- Gurobi
- R
- Python
- SQL
- CSS / HTML (web)
- MS Excel / MS Access / VBA

Subsection 2

About this Class

Course Material

- All digital homework submissions and grades will be submitted/posted through Canvas
- All other course material can be found at the following link

https://goo.gl/xzbwzw

Syllabus

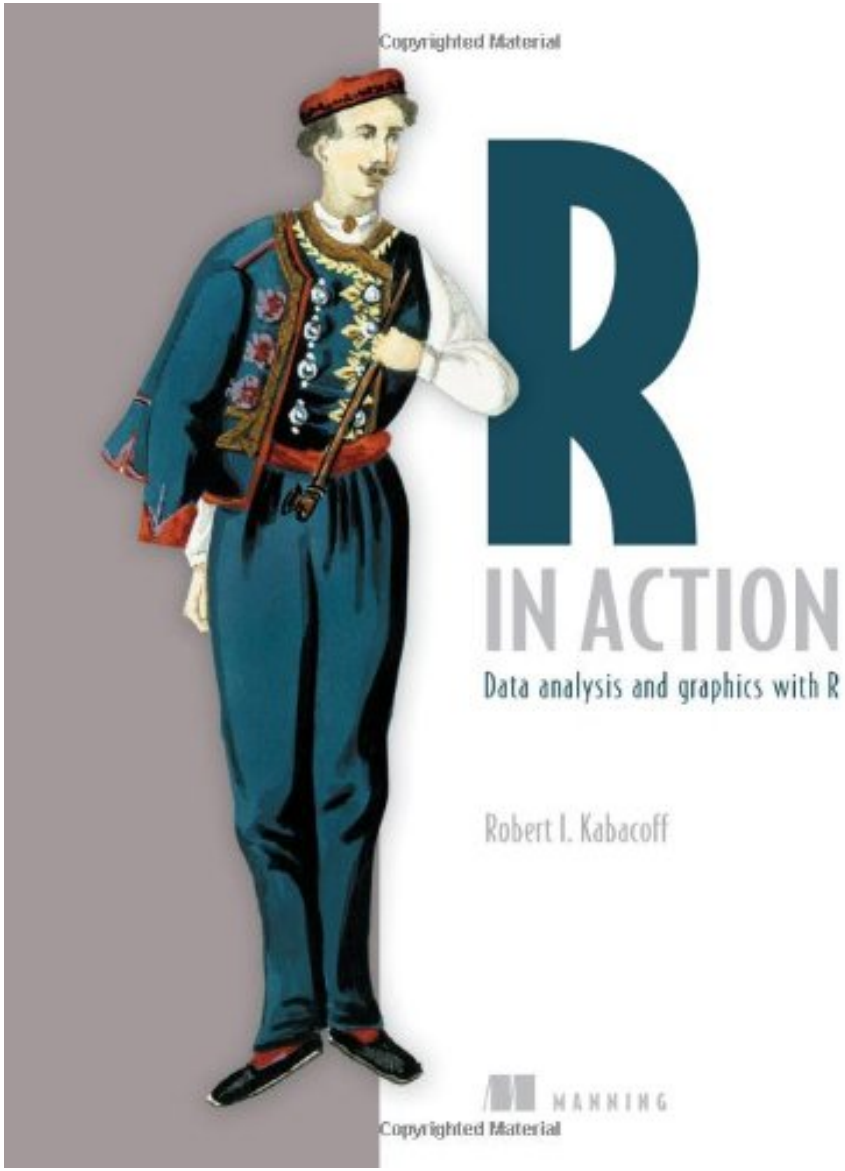
MSDS 593 – Exploratory Data Analysis with R
Instructor: Paul Intrevado
Course Syllabus
Summer 2018

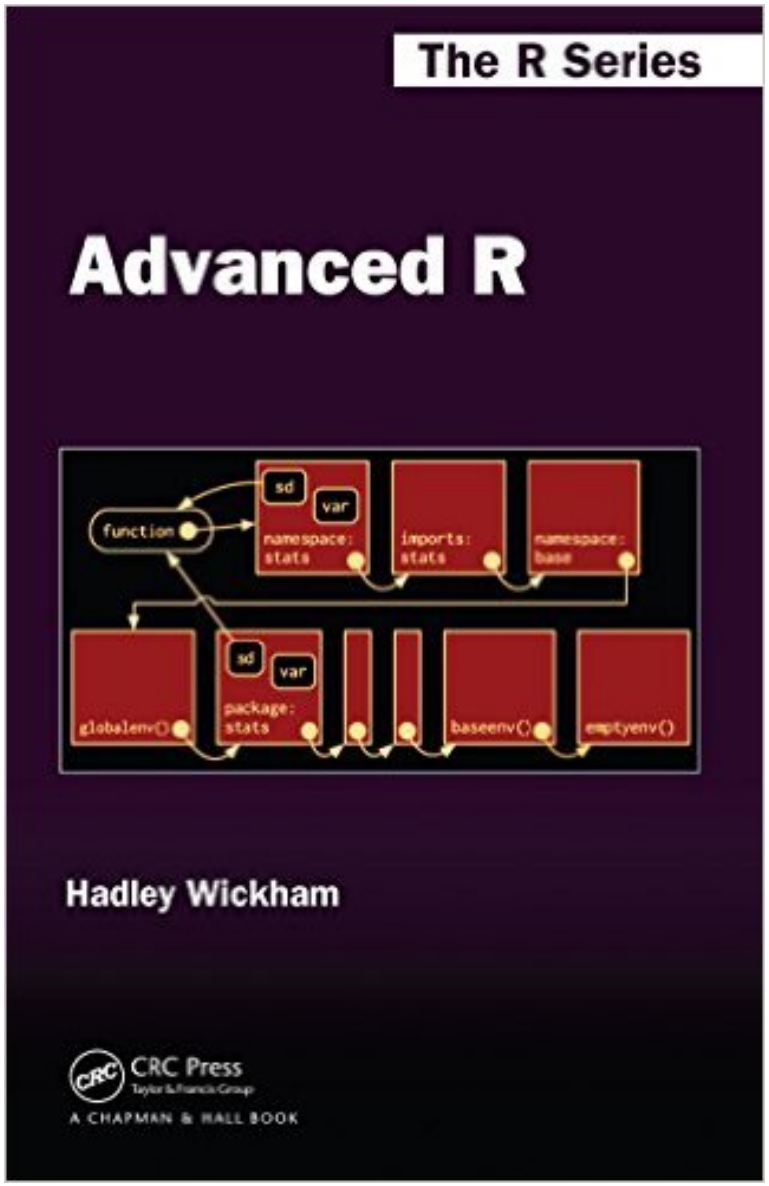
SUMMARY INFORMATION
Offices: Shared Office (Downtown) / McLaren Hall, Room 103 (Main Campus)
Office Hours: Wednesdays, 10.00 - 11.30 and 13.45 - 15.15 in Shared Office
Office Phone: 415/422.2527
Email: pintrevado@usfca.edu

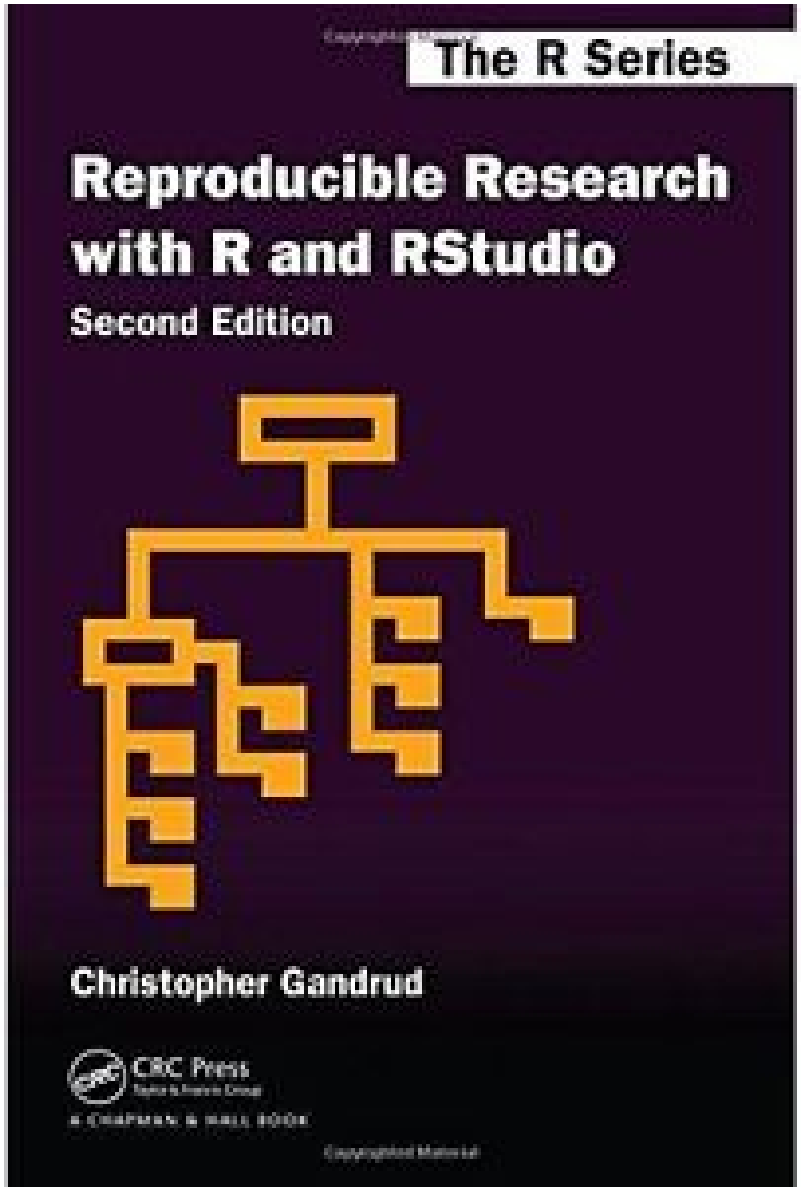
Class Time: 10:00 - 12:00 / 13:00 - 15:00 Thursdays and Fridays
Final Exam: 13.00 - 15.00 Friday, August 10^{th}, 2018
Quizzes: 09:15 - 10:00 Thursdays

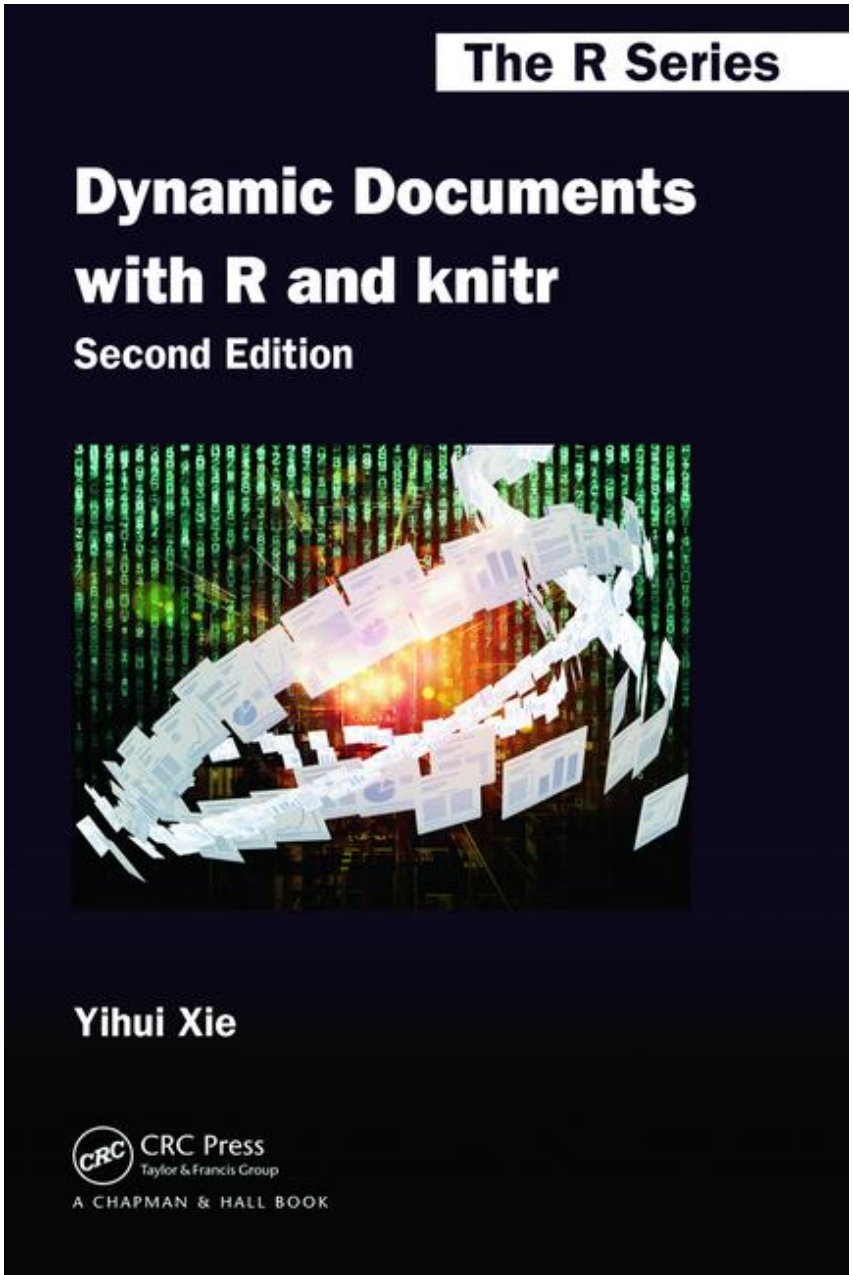
Subsection 3

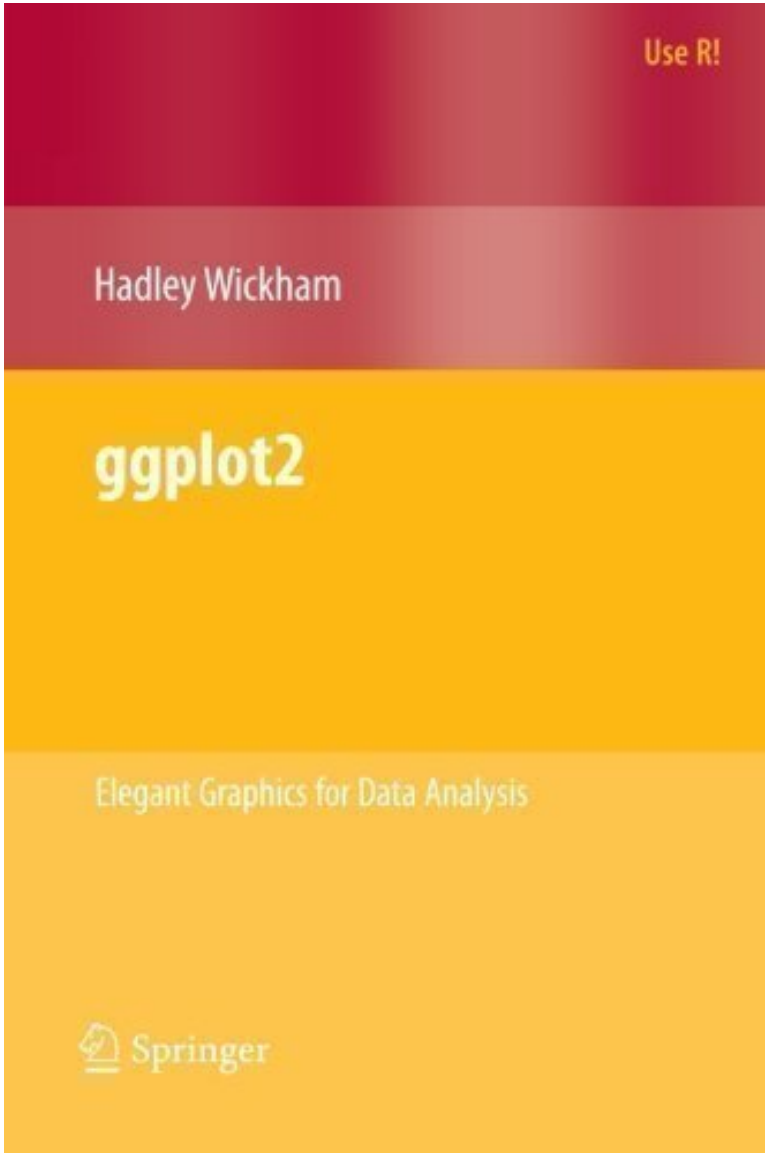
Reference Textbooks

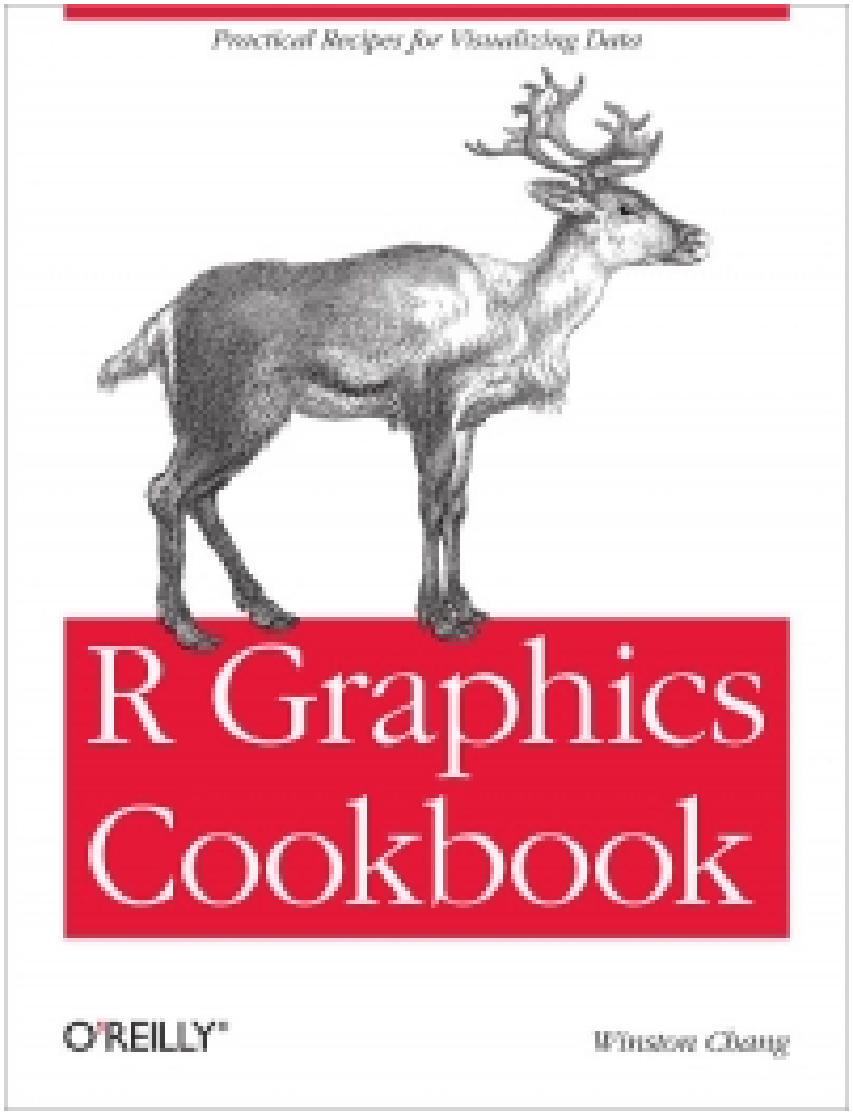

Copyrighted Material
R
IN ACTION
Data analysis and graphics with R
Robert I. Kabacoff
MANNING
Copyrighted Material

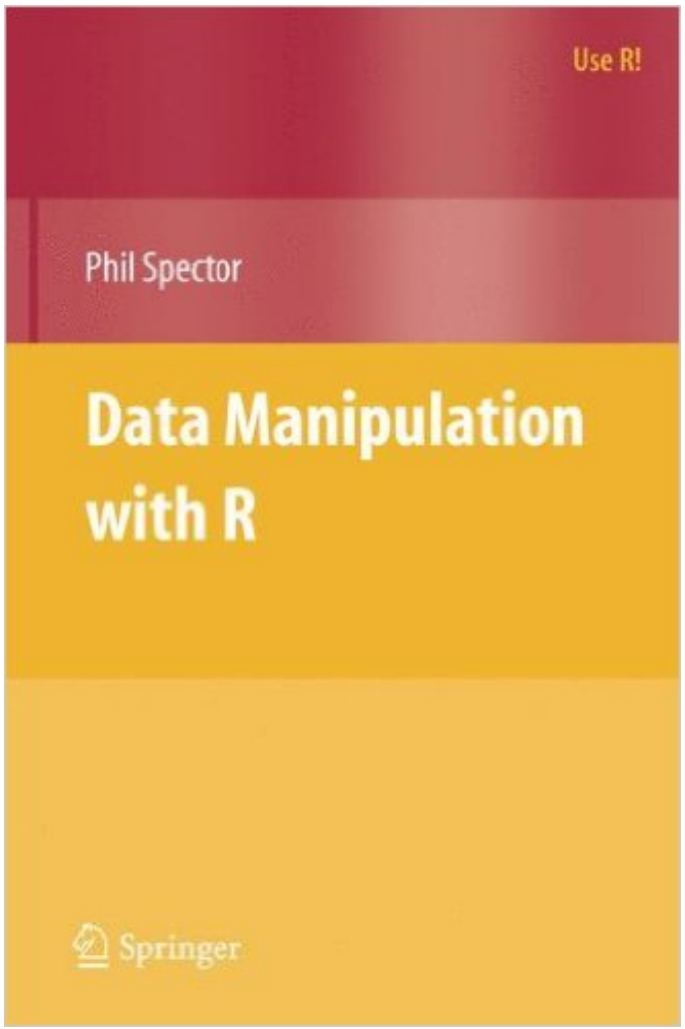

The R Series
Advanced R
function
sd
var
namespace:
stats
imports:
stats
namespace:
base
globalenv()
sd
var
package:
stats
baseenv()
emptyenv()
Hadley Wickham
CRC Press
A CHAPMAN & HALL BOOK

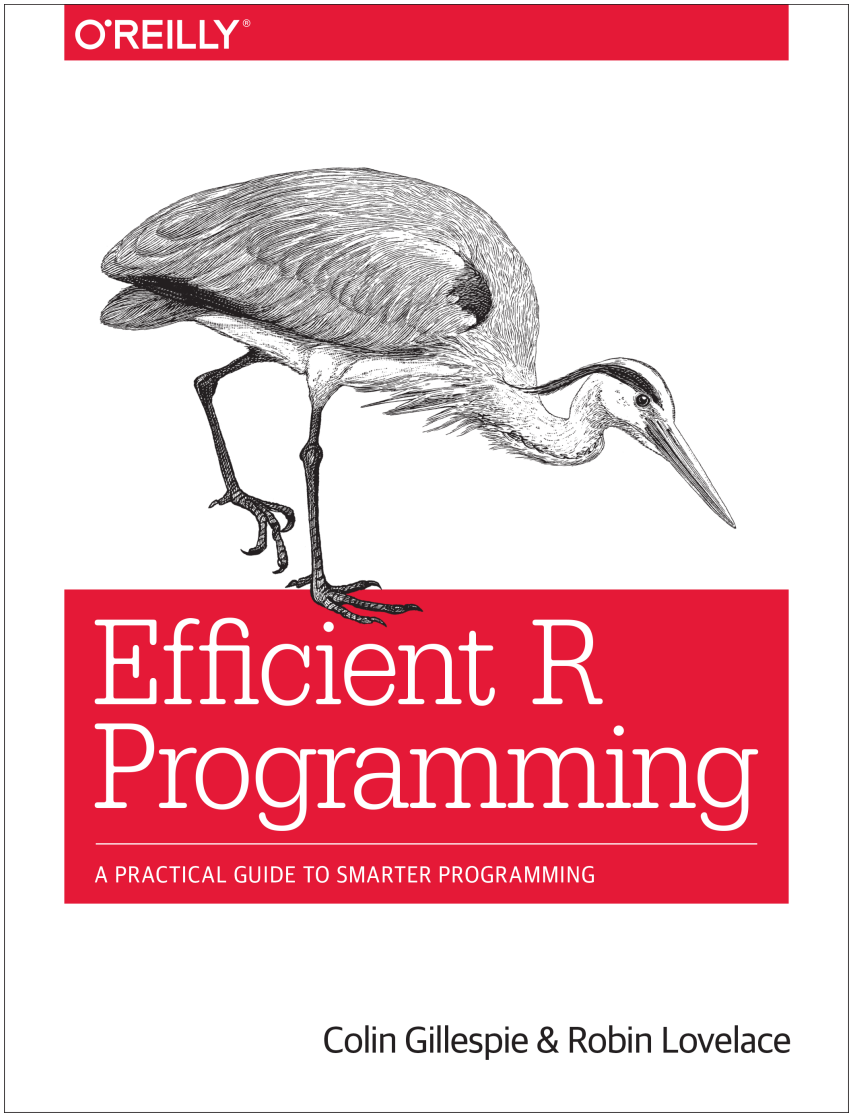

The R Series
Reproducible Research with R and RStudio
Second Edition
Christopher Gandrud
CRC Press

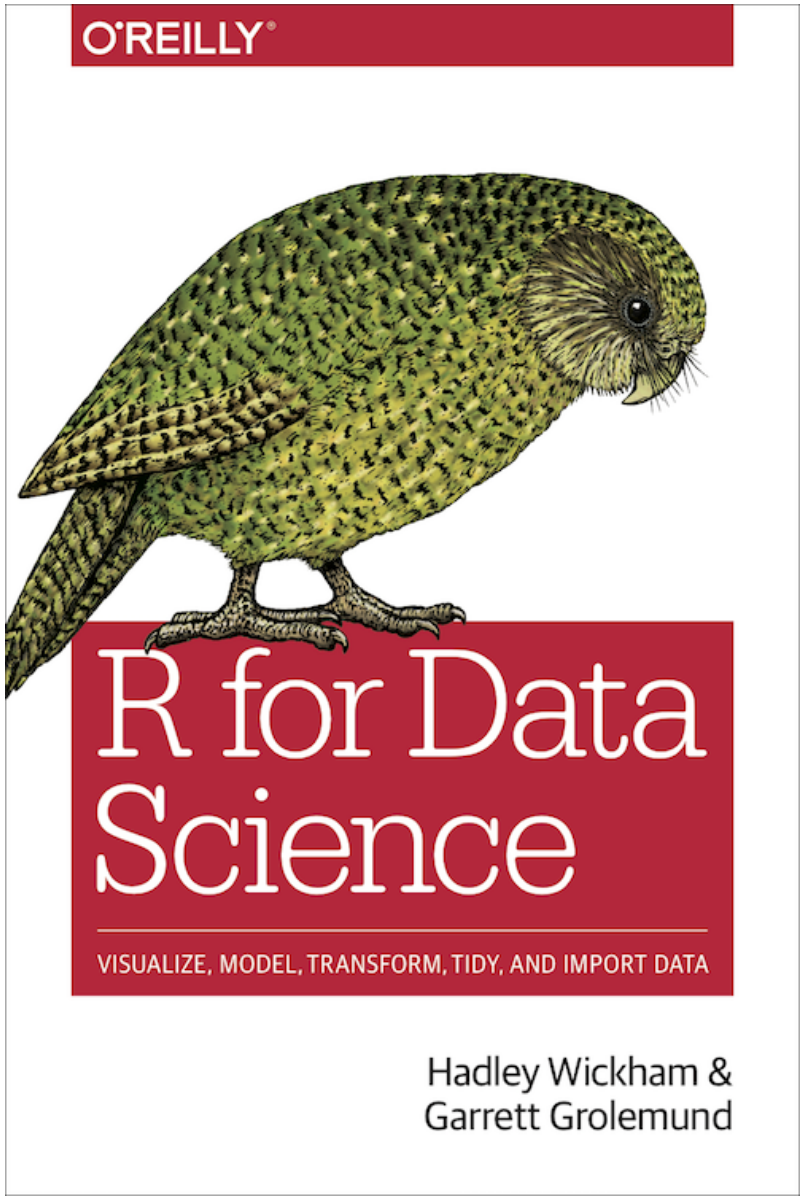

The R Series
Dynamic Documents with R and knitr
Second Edition
Yihui Xie
CRC Press
Taylor & Francis Group
A CHAPMAN & HALL BOOK


Use R!
Hadley Wickham
ggplot2
Elegant Graphics for Data Analysis
Springer


Practical Recipes for Visualizing Data
R Graphics Cookbook
O'REILLY
Winston Chang


Use R!
Phil Spector
Data Manipulation with R
Springer


O'REILLY
Efficient R Programming
A PRACTICAL GUIDE TO SMARTER PROGRAMMING
Colin Gillespie & Robin Lovelace

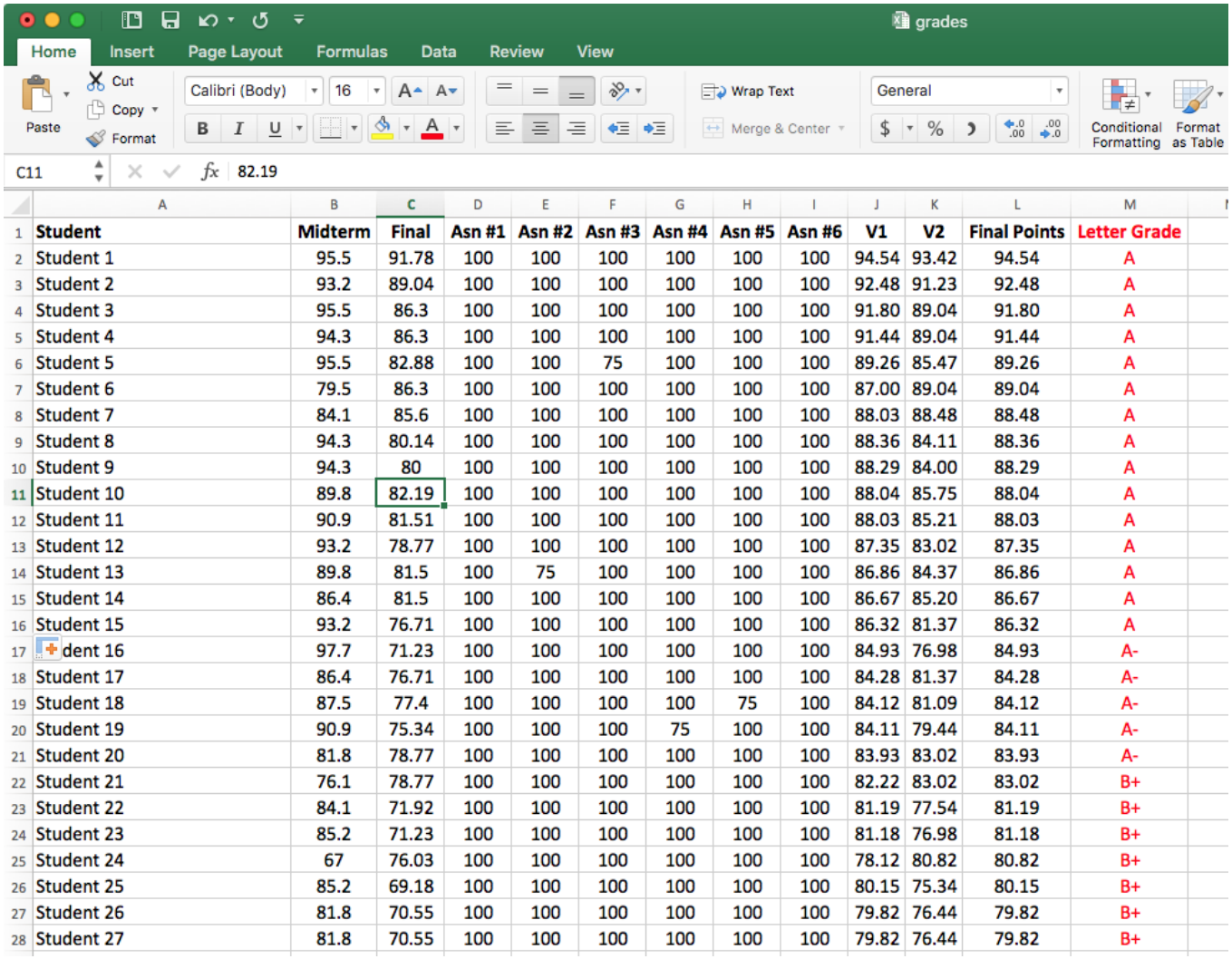

O'REILLY
R for Data Science
VISUALIZE, MODEL, TRANSFORM, TIDY, AND IMPORT DATA
Hadley Wickham &
Garrett Grolemund

Section 2

Introduction to R

What About Excel?

Excel is Great for Certain Things...



grades

Home | Insert | Page Layout | Formulas | Data | Review | View

Paste | Cut | Copy | Format | Calibri (Body) | 16 | Wrap Text | General | Merge & Center | Conditional Formatting | Format as Table

C11 | 82.19

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	**Student**	**Midterm**	**Final**	**Asn #1**	**Asn #2**	**Asn #3**	**Asn #4**	**Asn #5**	**Asn #6**	**V1**	**V2**	**Final Points**	**Letter Grade**
2	Student 1	95.5	91.78	100	100	100	100	100	100	94.54	93.42	94.54	A
3	Student 2	93.2	89.04	100	100	100	100	100	100	92.48	91.23	92.48	A
4	Student 3	95.5	86.3	100	100	100	100	100	100	91.80	89.04	91.80	A
5	Student 4	94.3	86.3	100	100	100	100	100	100	91.44	89.04	91.44	A
6	Student 5	95.5	82.88	100	100	75	100	100	100	89.26	85.47	89.26	A
7	Student 6	79.5	86.3	100	100	100	100	100	100	87.00	89.04	89.04	A
8	Student 7	84.1	85.6	100	100	100	100	100	100	88.03	88.48	88.48	A
9	Student 8	94.3	80.14	100	100	100	100	100	100	88.36	84.11	88.36	A
10	Student 9	94.3	80	100	100	100	100	100	100	88.29	84.00	88.29	A
11	Student 10	89.8	82.19	100	100	100	100	100	100	88.04	85.75	88.04	A
12	Student 11	90.9	81.51	100	100	100	100	100	100	88.03	85.21	88.03	A
13	Student 12	93.2	78.77	100	100	100	100	100	100	87.35	83.02	87.35	A
14	Student 13	89.8	81.5	100	75	100	100	100	100	86.86	84.37	86.86	A
15	Student 14	86.4	81.5	100	100	100	100	100	100	86.67	85.20	86.67	A
16	Student 15	93.2	76.71	100	100	100	100	100	100	86.32	81.37	86.32	A
17	dent 16	97.7	71.23	100	100	100	100	100	100	84.93	76.98	84.93	A-
18	Student 17	86.4	76.71	100	100	100	100	100	100	84.28	81.37	84.28	A-
19	Student 18	87.5	77.4	100	100	100	100	75	100	84.12	81.09	84.12	A-
20	Student 19	90.9	75.34	100	100	100	75	100	100	84.11	79.44	84.11	A-
21	Student 20	81.8	78.77	100	100	100	100	100	100	83.93	83.02	83.93	A-
22	Student 21	76.1	78.77	100	100	100	100	100	100	82.22	83.02	83.02	B+
23	Student 22	84.1	71.92	100	100	100	100	100	100	81.19	77.54	81.19	B+
24	Student 23	85.2	71.23	100	100	100	100	100	100	81.18	76.98	81.18	B+
25	Student 24	67	76.03	100	100	100	100	100	100	78.12	80.82	80.82	B+
26	Student 25	85.2	69.18	100	100	100	100	100	100	80.15	75.34	80.15	B+
27	Student 26	81.8	70.55	100	100	100	100	100	100	79.82	76.44	79.82	B+
28	Student 27	81.8	70.55	100	100	100	100	100	100	79.82	76.44	79.82	B+

...but Not Everything

Sample Data

- Six columns of data with $\sim$ 1.05 million rows
- Column 5: `startDate`
- Column 6: `endDate`
- **Objective**: test to see if `endDate` $<$ `startDate`

...but Not Everything

Sample Data

- Six columns of data with $\sim$ 1.05 million rows
- Column 5: `startDate`
- Column 6: `endDate`
- **Objective**: test to see if `endDate` $<$ `startDate`

RESULTS

- **Excel**: good luck...
- R: 33 min (poor coding technique)
- R: 58.5 sec (improved coding technique)

R or Python?

Vectorization in R

- Vectorized code saves time asking `type` questions
- There is an optimized engine—a basic linear algebra system (BLAS)—that is highly efficient at solving linear algebra problems
- A lot or R functions are written in C (or variants)
- MATLAB, Mathematica and the NumPy package for Python are also vectorized

http://www.noamross.net/blog/2014/4/16/vectorization-in-r–why.html

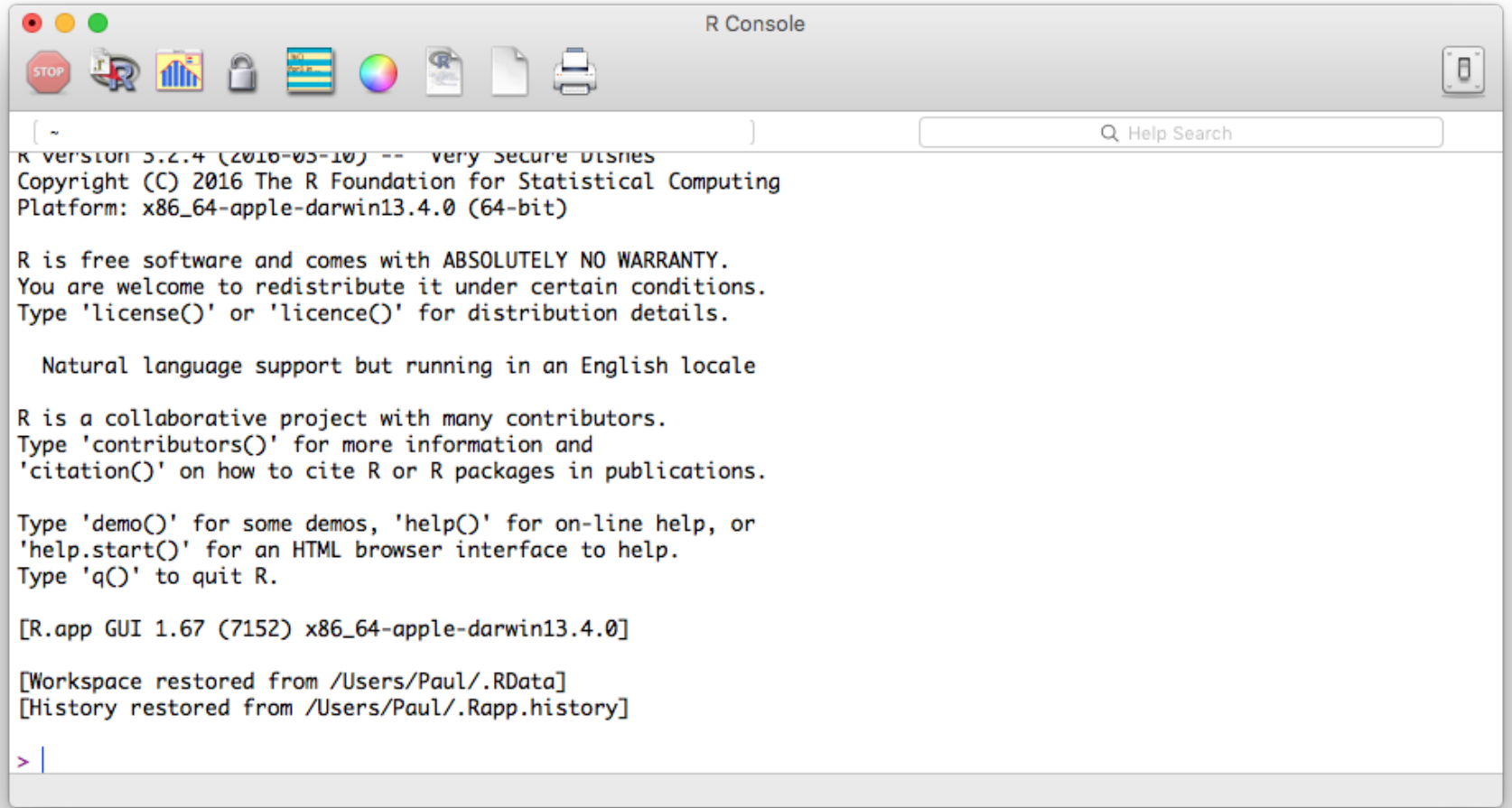
Why Use R?

- Open source (free)
- Runs on just about any platform
- Great visualization capabilities (ggplot2)
- Read/write from/to various data sources
- Scripting language (interpreted)
- Deep library of advanced data manipulation and statistical packages

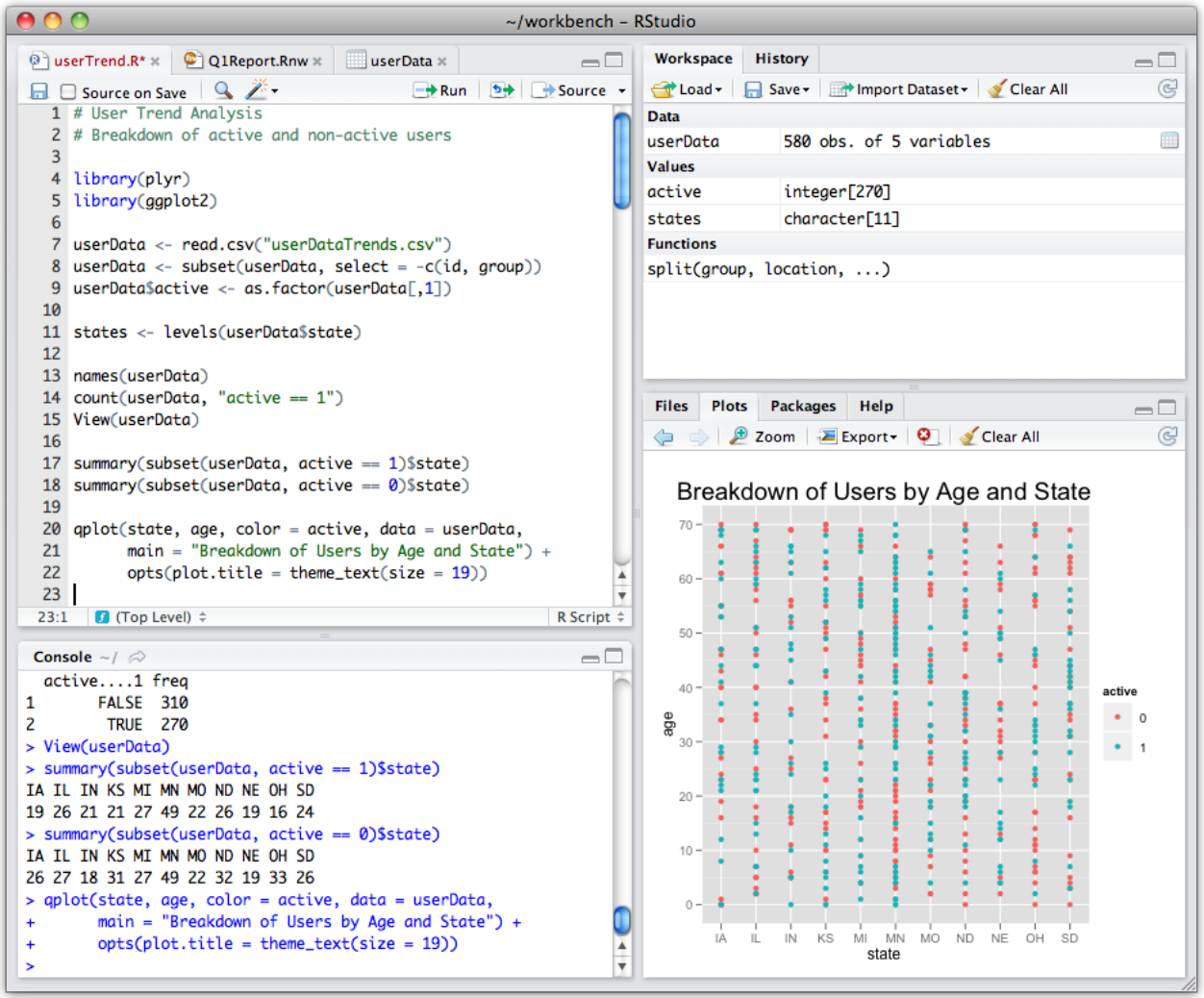
Installing R

- RStudio is a nice, user-friendly integrated development environment (IDE), but can be quirky at times — still highly recommended and what I will use in class
- Required to install R regardless
- You can even run R from a terminal window if you wish
- If you are curious to see some basic R demos, type `demo()` to see the list of demonstration code available and then run one if you like, e.g., `demo(persp)`

This is what R Looks Like



This is what RStudio Looks Like



RStudio

RStudio has Four Panels

- Console
- Scripting/Viewing
- Files/Packages/Help/Viewer
- Environment/History/Plots

Open up RStudio for a guided tour `class01intro.R`

Notes on R

- R is case-sensitive
- It is best practice to use the assignment operator '<-' instead of the equality operator '=' for all code, even though both work, e.g.,

Syntax	*Comments*
`x <- 5`	standard syntax
`x = 5`	poor syntax, not permitted
`5 -> x`	awkward syntax, not permitted (but it works)

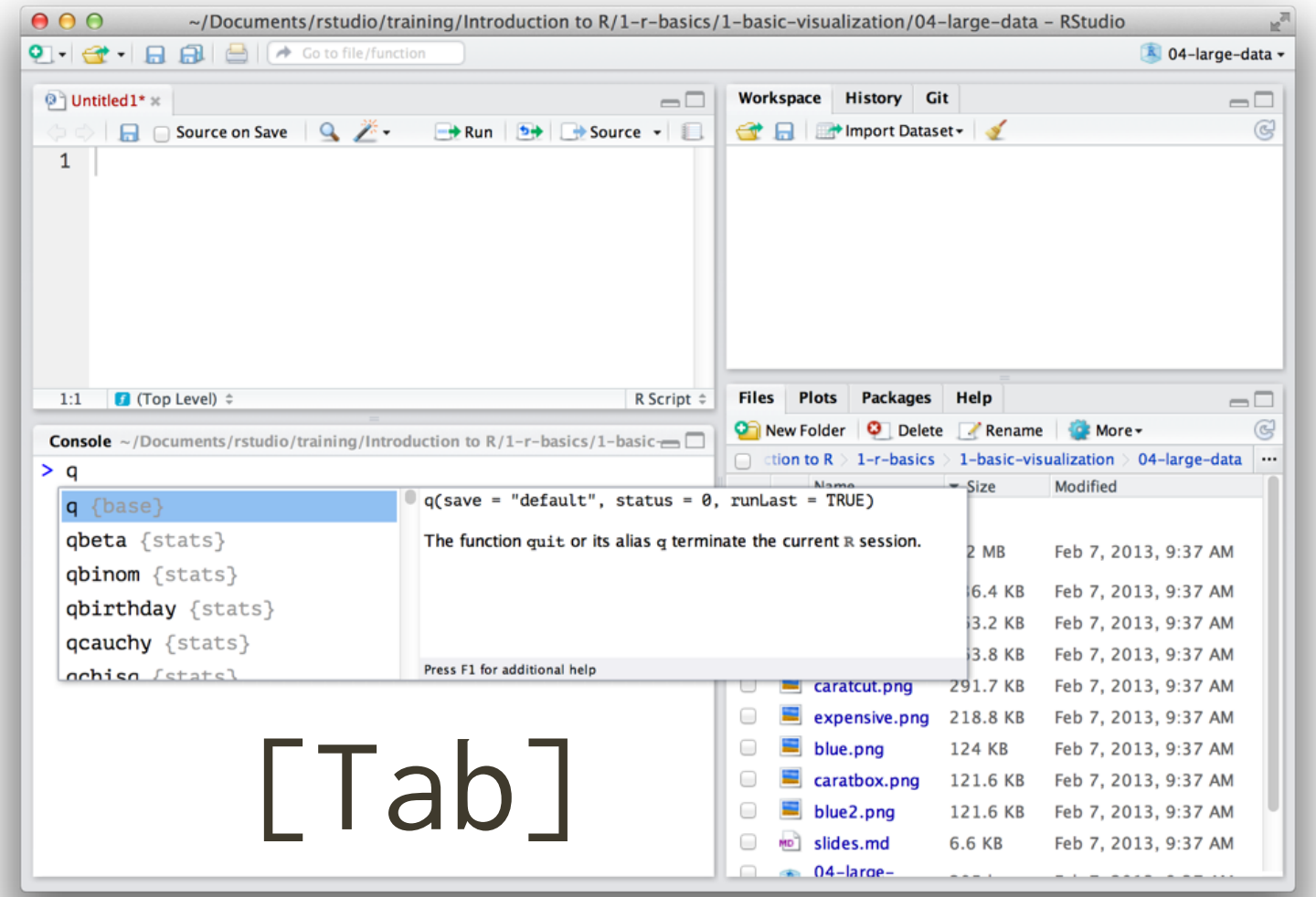
- Keyboard shortcut for assignment operator
  - `Option (alt)` + `-`

Basic R Functions

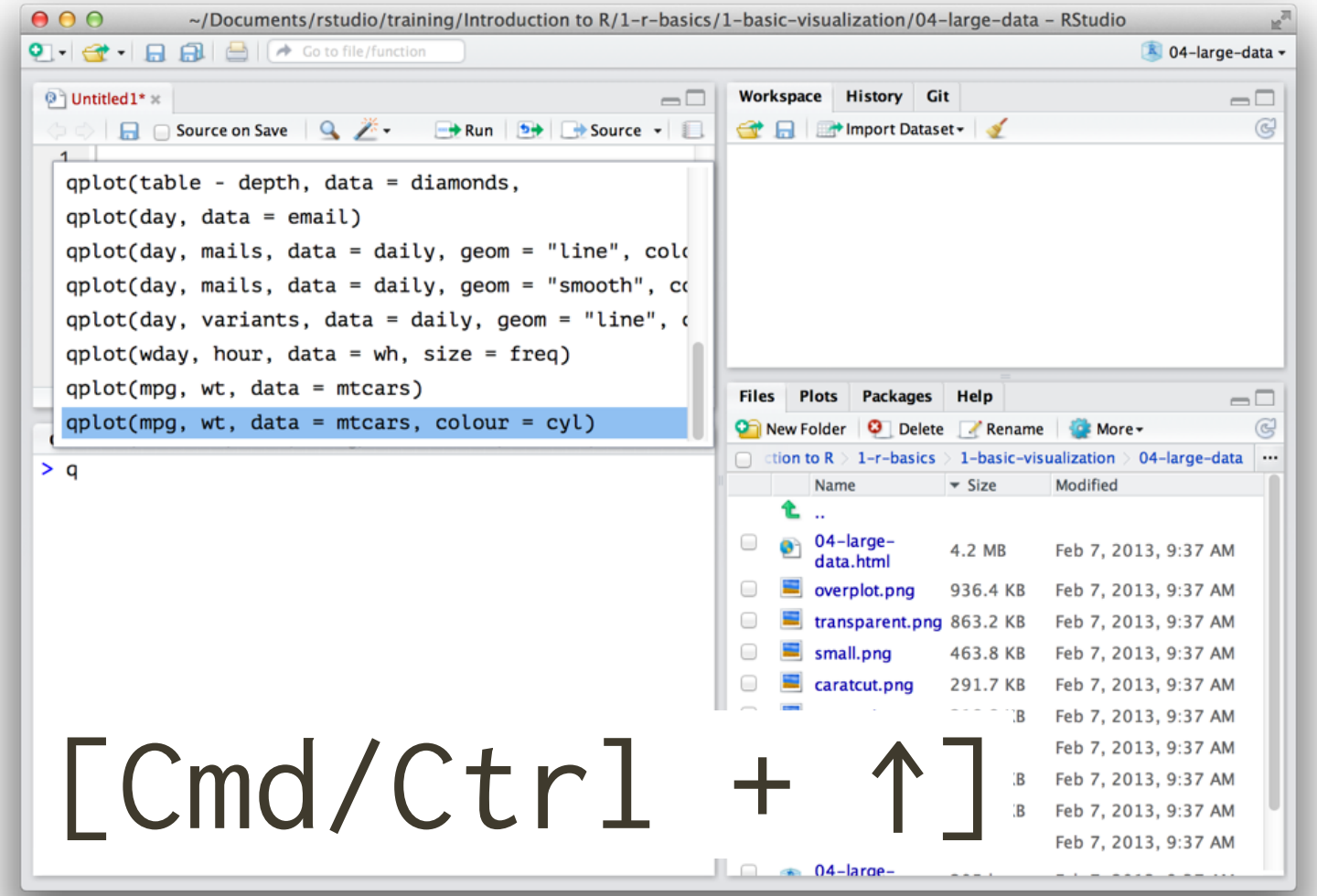
Function	*Action*
`?foo`	Help on the function `foo`
`??foo`	Search the help system for instances of the function `foo`
`RSiteSearch("foo")`	Search for the string `foo` in online help manuals and archived mailing lists
`data()`	List all available example datasets contained in currently loaded packages
`getwd()`	List the current working directory
`setwd("~/Desktop")`	Change working directory to `Desktop`
`rm()`	Remove (delete) one or more objects
`ls()`	List the objects in the current directory

The Best Place for Answers to R Questions?

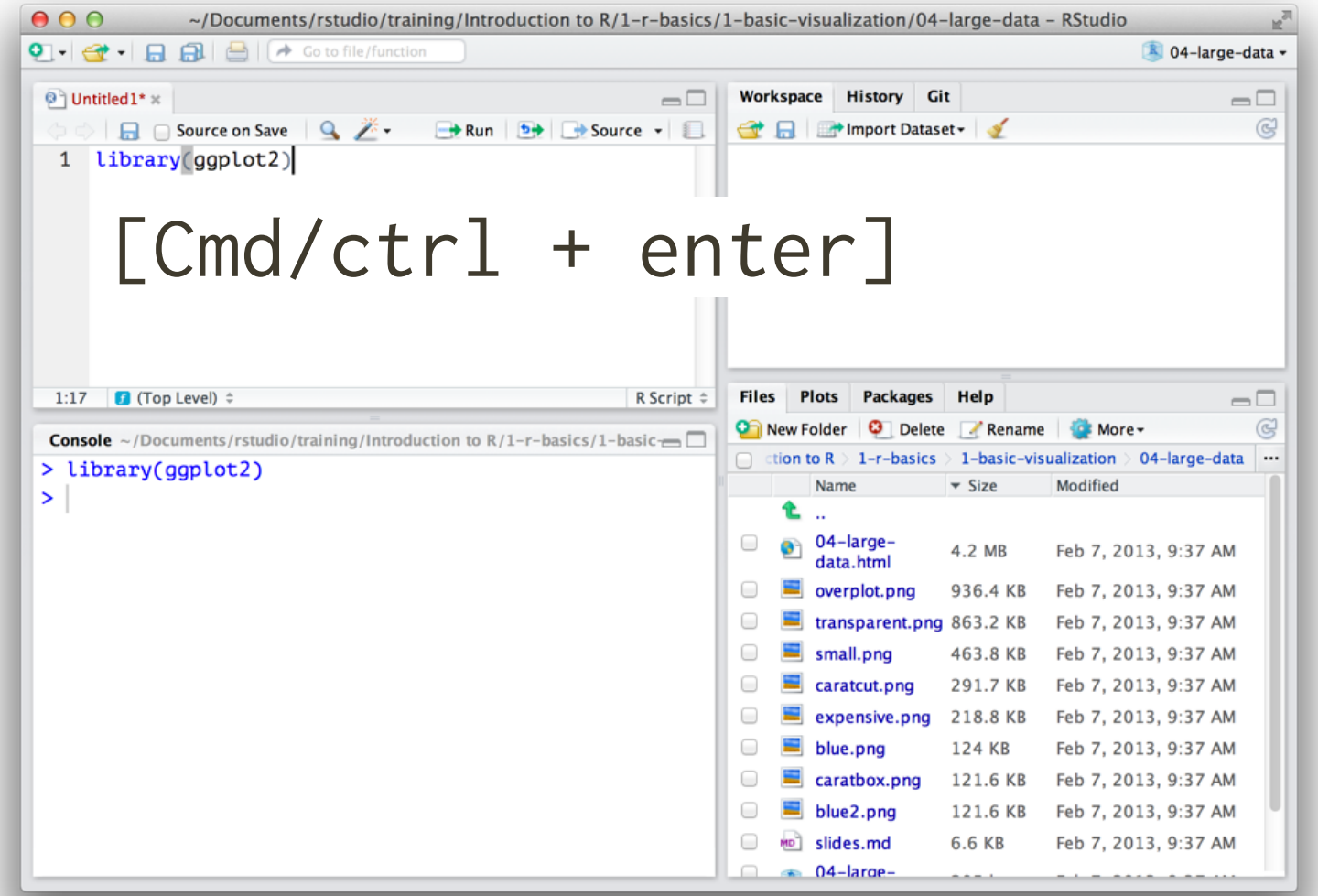
Useful R Keyboard Shortcuts: Autocomplete



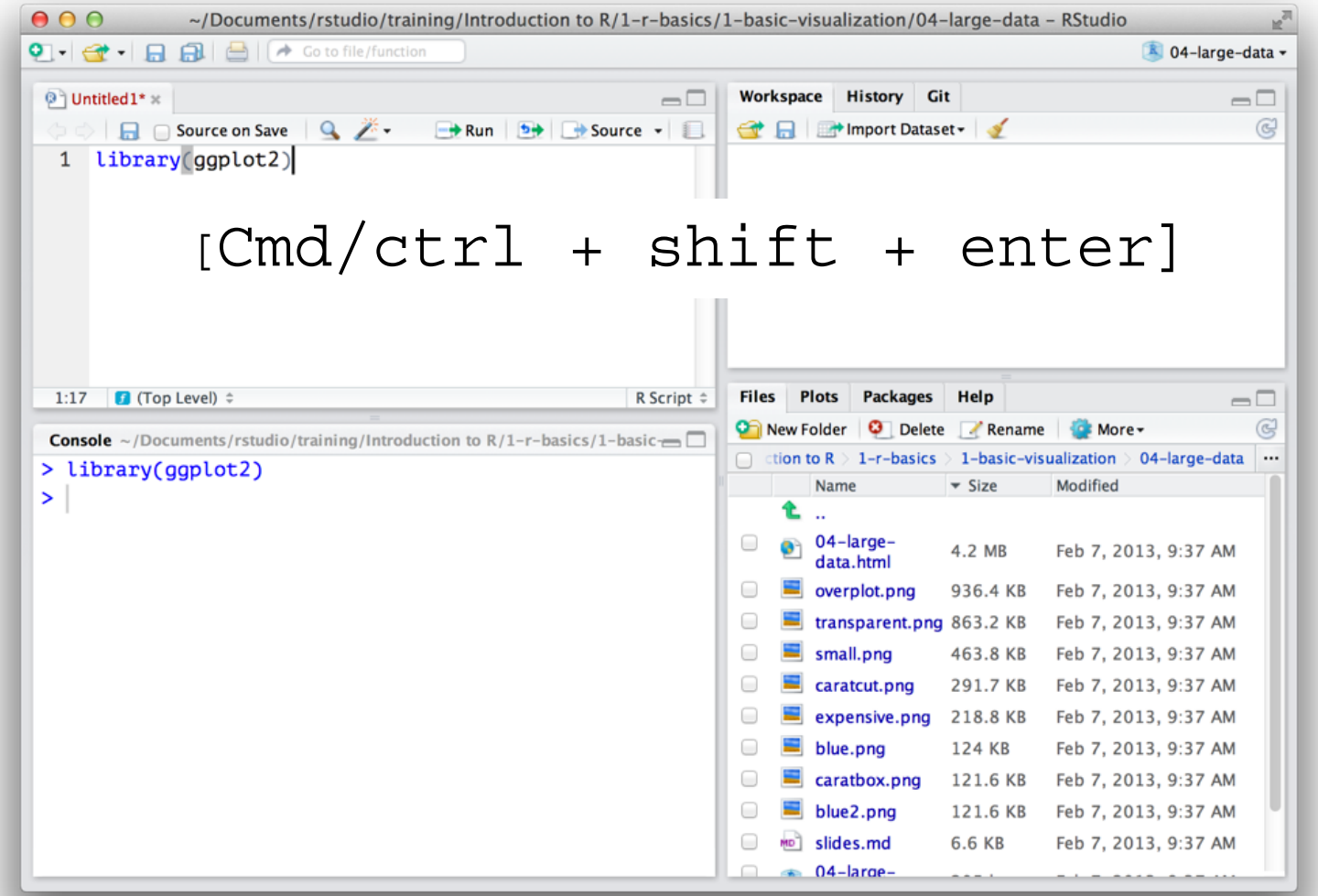
Useful R Keyboard Shortcuts: History



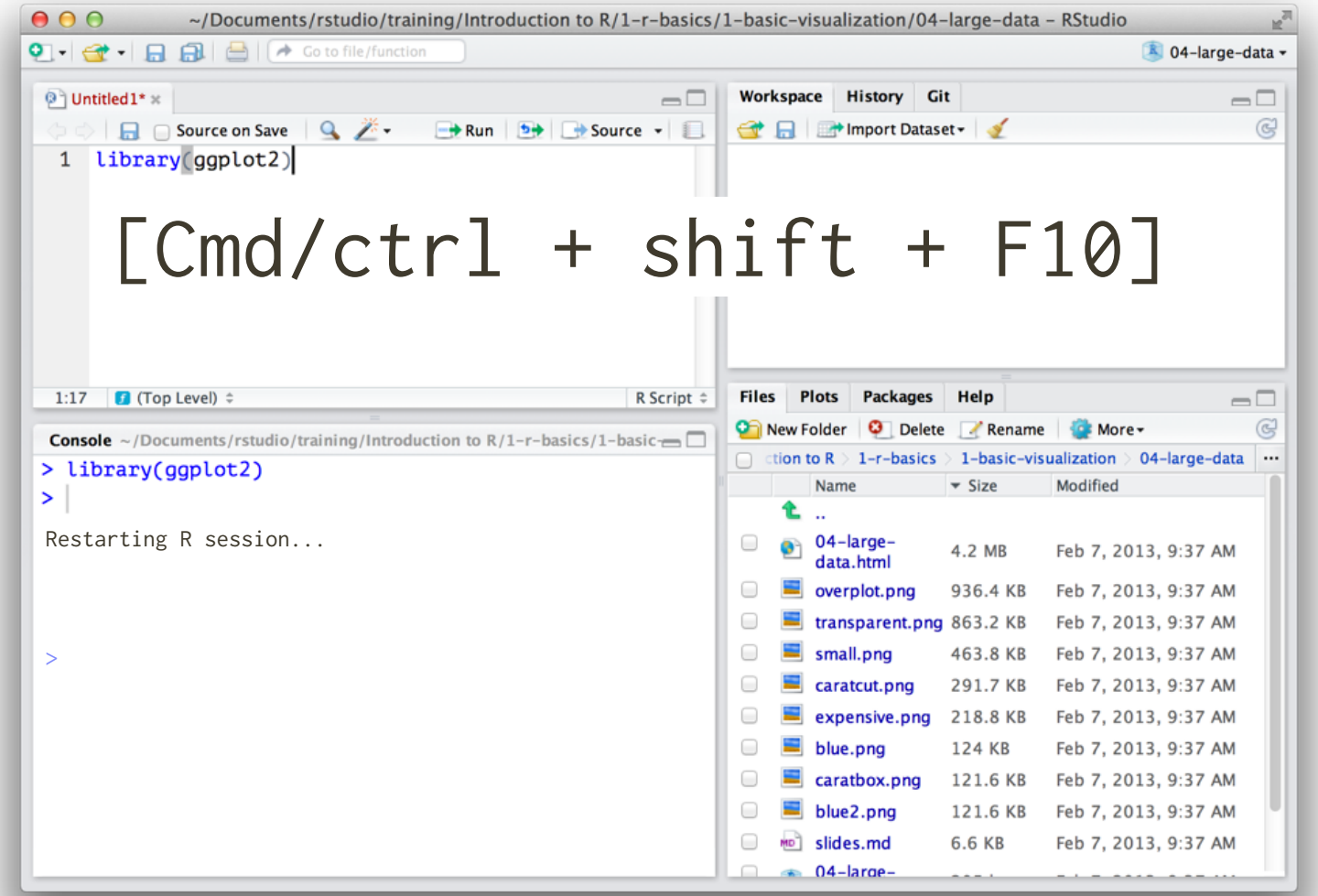
Useful R Keyboard Shortcuts: Execute Subset of Code



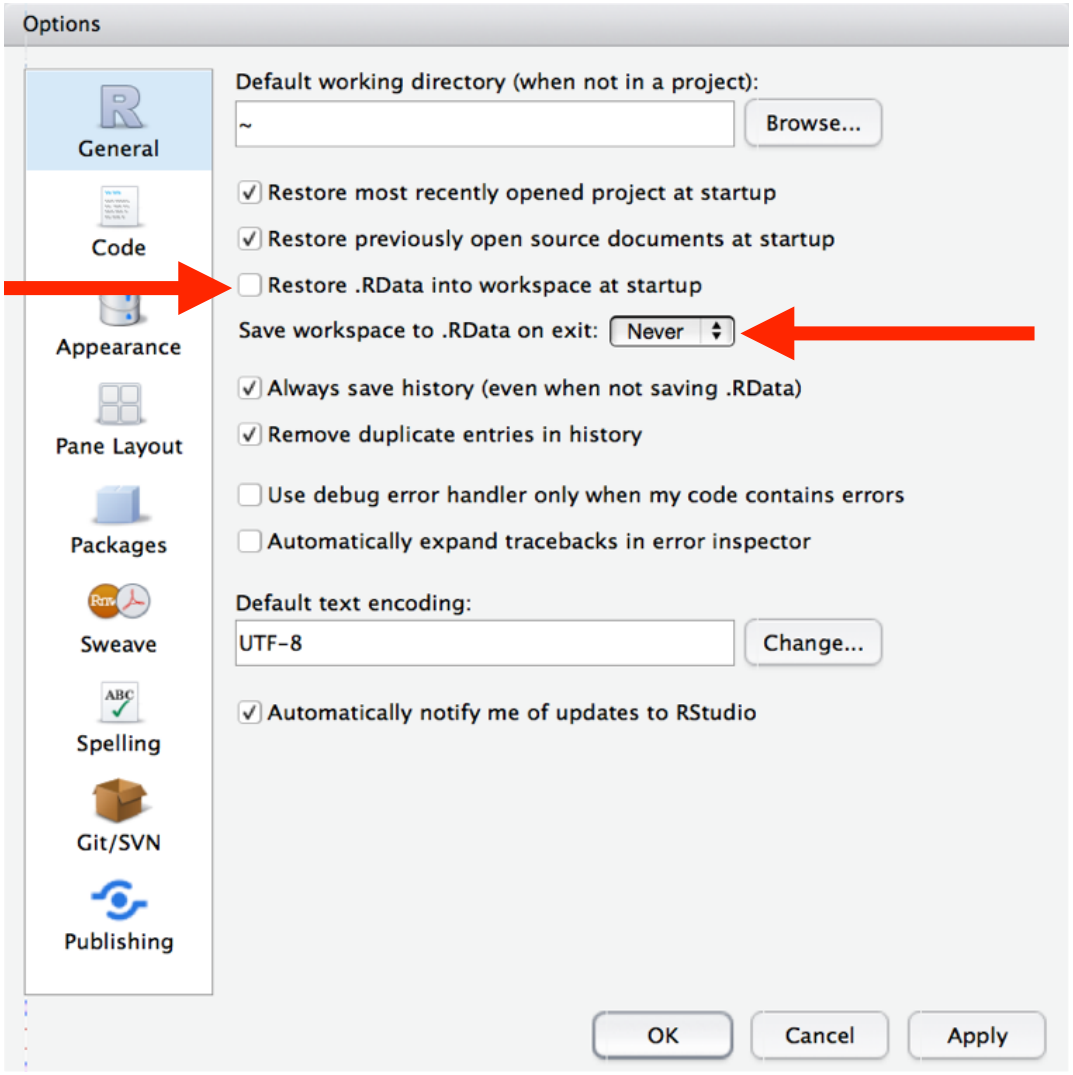
Useful R Keyboard Shortcuts: Execute All Code



Useful R Keyboard Shortcuts: Restarting an R Session



IMPORTANT R Setting



TRY THIS [IN A SCRIPTING WINDOW]

1. 10,352 + 987,653
2. 10,352 / 987,653
3. $2^6 \times 99$
4. 91,4545 modulus 33

TRY THIS [IN A SCRIPTING WINDOW]

1. 10,352 + 987,653
2. 10,352 / 987,653
3. $2^6 \times 99$
4. 91,4545 modulus 33

SOLUTIONS

1.
```
> 10352 + 987653
```
2.
```
> 10352 / 987653
```
3.
```
> > 2**6 * 99    OR    2^6 * 99
```
4.
```
> 914545 %% 33
```

A Brief Digression

- Whenever writing code, you want to be sure to clear your environment to ensure the fidelity of your results
- In each and every R script file I write, I always include the following two lines of code

```
rm(list=ls())
cat("\014")
```

1. `rm(list=ls())` removes all objects in the current environment
2. For Mac, `cat("\014")` clears the console window
    - same as keystroke `ctrl` + `l`

Packages in R

- Vanilla R comes with extensive capabilities
- ...BUT...
- some of the most exciting features in R are available as optional modules called Packages that you can download and install
- Packages are—like R—free, user-contributed modules that you can download and install
- There are $\sim 10,000+$ R packages [July 2018]

Packages in R

- Vanilla R comes with extensive capabilities
- ...BUT...
- some of the most exciting features in R are available as optional modules called Packages that you can download and install
- Packages are—like R—free, user-contributed modules that you can download and install
- There are $\sim 10,000+$ R packages [July 2018]
- Given R packages are user-contributed, some contain errors (some of which we have found right here in MSAN), so feel free to use R for analysis, but maybe double-check your output before you buy or sell billions of dollars of stock based on R package calculations

What are R Packages?

- Packages are collections of functions, data and compiled code in a well-defined format
- The `library()` function shows you which packages are **saved** in your library (i.e., which packages have been downloaded)

n.b. `library()` **does not** tell you which packages are loaded, it only tells you which packages are downloaded

- `search()` tells you which packages are loaded and ready to use
- `install.packages("myPackageName")` is used to install packages, and `library("myPackageName")` loads the package into your current working session
- You only ever need to download a package once, but you always need to load packages when restarting an R session

Loading, Using & Maintaining R Packages [EXAMPLE]

```
> install.packages("dplyr")
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  ...
                                 Dload  Upload   Total   Spent    Left  ...
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:-- ...

The downloaded binary packages are in
  /var/folders/jm/3w7pqfms0nvg_ypvnvkkk83h0000gn/...

> summarise(iris, meanVal = mean(Sepal.Length))

Error: could not find function "summarise"

> dplyr::summarise(iris, meanVal = mean(Sepal.Length))
   meanVal
1 5.843333
```

Loading, Using & Maintaining R Packages [EXAMPLE] [CONT'D]

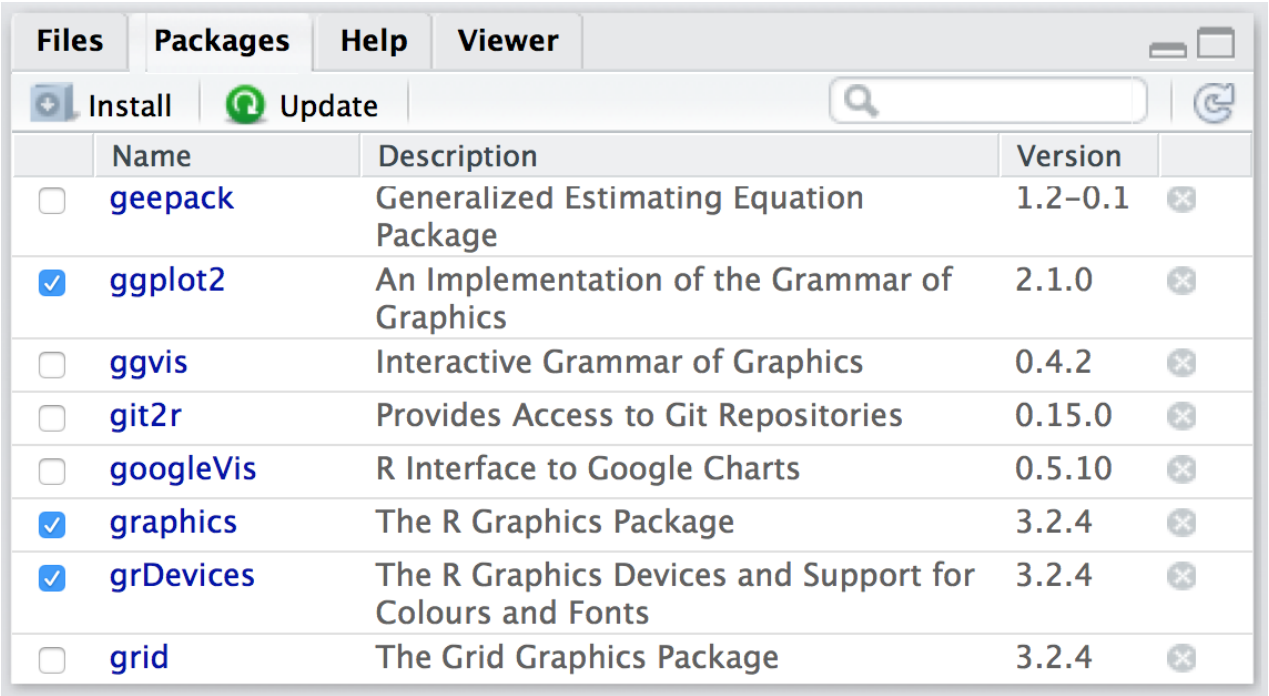
```
# this loads the package
> library(dplyr)

# it can now be called without the dplyr:: prefix
> summarise(iris, meanVal = mean(Sepal.Length))
   meanVal
1 5.843333
```

- Packages are often updated by their authors, so be sure to keep your packages up to date
  - `update.packages()` will update all packages tab
  - `installed.packages()` will list all packages you have downloaded, along with their version numbers, dependencies and other information

Loading, Using & Maintaining R Packages [CONT'D]

A far less cumbersome way to install, load and update packages in RStudio is using the appropriate icons in the "Packages" window



Notes on R Packages

Packages sometimes (often?) have dependencies on other packages, e.g.,

```
#load the package MatchIt
> library(MatchIt)
Loading required package: MASS
```

- Without asking, when loading the MatchIt package, the MASS package is automatically loaded
- This is helpful in one respect, since MatchIt leverages certain functionality from MASS; if MASS wasn't automatically loaded, then calling certain functions from MatchIt might throw an error

n.b. Be careful of function masking: because there are so many R packages available from so many different authors, it often happens that different packages have identically named functions

Notes on R Packages [CONT'D]

When calling a function, R searches the Global Environment first, then iterates through all packages for the function, beginning from the most recently added

```
> search()
 [1] ".GlobalEnv"        "package:reshape2"  "package:plyr"
 [4] "package:MatchIt"   "package:MASS"      "package:ggplot2"
 [7] "tools:rstudio"     "package:stats"     "package:graphics"
[10] "package:grDevices" "package:utils"     "package:datasets"
[13] "package:methods"   "Autoloads"         "package:base"

> (.packages())
 [1] "reshape2"  "plyr"      "MatchIt"   "MASS"      "ggplot2"   "stats"
 [7] "graphics"  "grDevices" "utils"     "datasets"  "methods"   "base"
```

Notes on R Packages [CONT'D]

When calling a function, R searches the Global Environment first, then iterates through all packages for the function, beginning from the most recently added

```
> search()
 [1] ".GlobalEnv"        "package:reshape2"  "package:plyr"
 [4] "package:MatchIt"   "package:MASS"      "package:ggplot2"
 [7] "tools:rstudio"     "package:stats"     "package:graphics"
[10] "package:grDevices" "package:utils"     "package:datasets"
[13] "package:methods"   "Autoloads"         "package:base"

> (.packages())
 [1] "reshape2"  "plyr"      "MatchIt"   "MASS"      "ggplot2"   "stats"
 [7] "graphics"  "grDevices" "utils"     "datasets"  "methods"   "base"
```

- If you are using a lot of packages and want to be certain you are calling a function from a specific package, use the double colon operator, e.g., `plyr::rename()`

Notes on R Packages [EXAMPLE]

TRY THIS

```
> library(magrittr)
> library(dplyr)

> iris %>% select(Petal.Length, Species) %>% group_by(Species) %>%
    summarize(meanVal = mean(Petal.Length))

> library(Hmisc)

> iris %>% select(Petal.Length, Species) %>% group_by(Species) %>%
    summarize(meanVal = mean(Petal.Length))

Error in summarize(., meanVal = mean(Petal.Length)) :
  argument "by" is missing, with no default
```

Notes on R Packages [EXAMPLE]

TRY THIS

```
> library(magrittr)
> library(dplyr)

> iris %>% select(Petal.Length, Species) %>% group_by(Species) %>%
    summarize(meanVal = mean(Petal.Length))

> library(Hmisc)

> iris %>% select(Petal.Length, Species) %>% group_by(Species) %>%
    summarize(meanVal = mean(Petal.Length))

Error in summarize(., meanVal = mean(Petal.Length)) :
  argument "by" is missing, with no default
```

SOLUTION

```
> iris %>% select(Petal.Length, Species) %>% group_by(Species) %>%
    dplyr::summarize(meanVal = mean(Petal.Length))
```

Notes on R Packages [CONT'D]

- If you feel you have too many packages loaded and want to unload (detach) one, you can use the following command: `detach("package:MatchIt", unload = TRUE)`

  n.b. Be aware that when unloading `MatchIt` you do not automatically unload the dependency that was loaded when you loaded `MatchIt`, namely, `MASS`

- You may choose to use `require("myPackageName")` in lieu of `library("myPackageName")` to load a package, but be cautious when doing so
  - `require()` attempts to load a package and returns a logical to indicate whether or not the could not be loaded; if the package could not be loaded, a **warning** is thrown
  - `library()` attempts to load a package and throws an **error** if the package could not be loaded

How Big is Big Data in R?

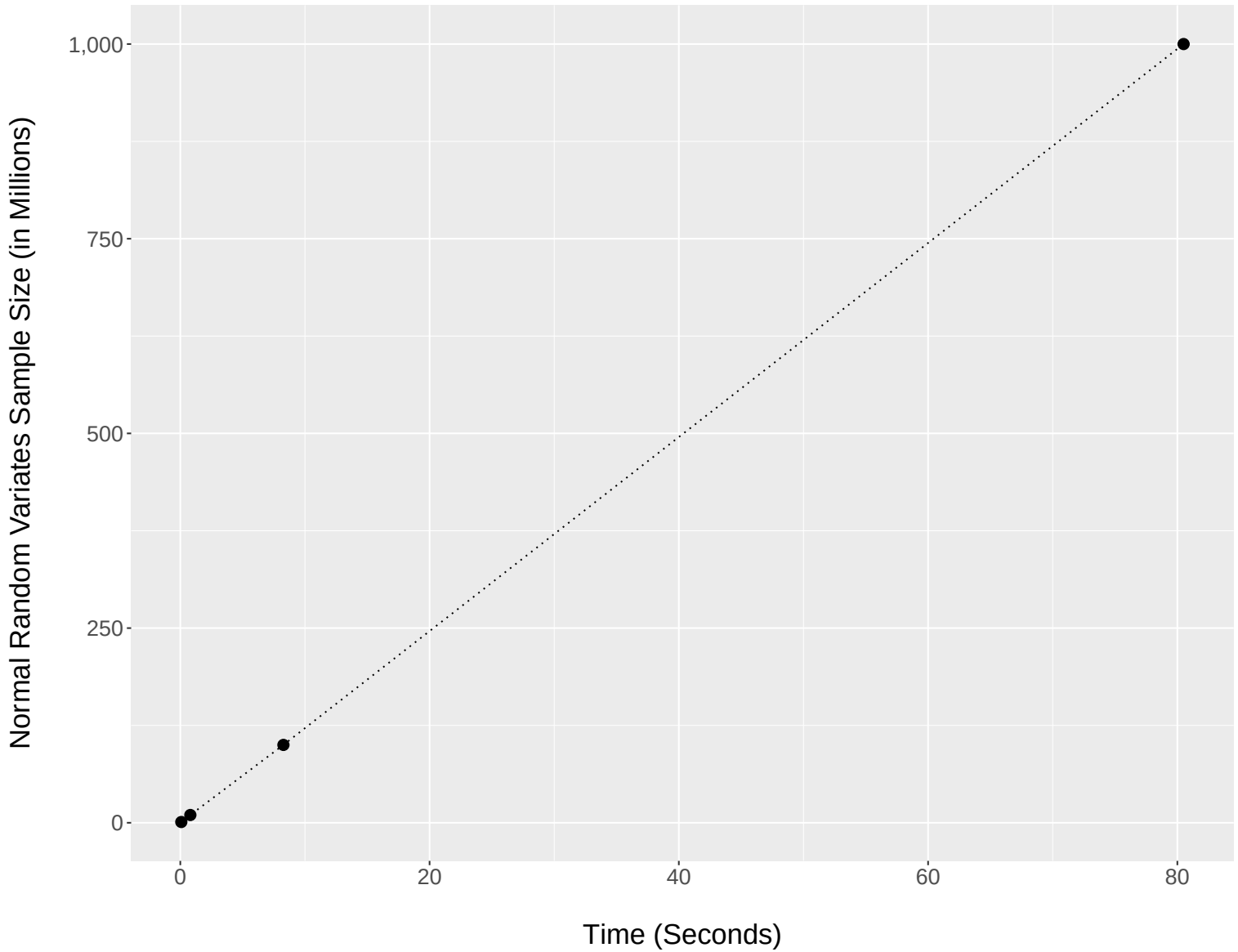
- R holds data in memory, effectively limiting data to the amount of RAM a computer has access to
- It is not uncommon to work with a data set containing 100,000,000 elements (e.g., 100,000 observations of 1,000 variables or 1,000,000 observations of 100 variables) without difficulty
- Of course all of the above approximations depend on what type of data is contained in each variable, e.g., I am currently working on a data set with 2.2 million records and twenty variables, which takes approximately one minute to load into memory
- The other issue to consider is what techniques and/or functions will be applied to the data

How Big is Big Data in R? [CONT'D]

- The more complex and memory intensive the task, the smaller the data will be required to be
- Basic plotting would likely require far less computational exertion that a complex statistical learning model

TRY THIS

1. Create a vector named `myVec` with 1,000,000 random $\sim \mathcal{N}(10, 5)$ variates. How much time does this take?
2. Repeat and time with 10,000,000 random variates
3. Repeat and time with 100,000,000 random variates
4. Repeat and time with 1,000,000,000 random variates
5. Generate a graph with *Sample Size* on the y axis and *Time* on the x axis


1,000
750
500
250
0
Normal Random Variates Sample Size (in Millions)
0
20
40
60
80
Time (Seconds)

SOLUTION

```
# create vector and time execution
system.time(
  myVec <- rnorm(1000000, mean = 10, sd = 5)
)

# load tidyverse to access tibble and ggplot packages
library(tidyverse)

myDF <- tibble(
  sampleSize = c(1, 10, 100, 1000),
  time = c(0.079, 0.805, 8.274, 80.5)
)

library(magrittr) # for piping
library(scales)   # for commas in y axis

# create graph in ggplot
myDF %>%
  ggplot(aes(x = time, y = sampleSize)) +
    geom_line(linetype = "dotted") +
    geom_point(size = 3) +
    xlab("\nTime (Seconds)") +
    ylab("Normal Random Variates Sample Size (in Millions)\n") +
    scale_y_continuous(labels = comma)
```

Data Sets in R?

- R comes built in with multiple data sets you can play with
- Many (most?) packages also have data sets
- `data()` will bring up a list of all data sets available across all loaded packages
- `help(<nameOfDataSet>)` will provide you a detailed description of the data set in question