

# R and RStudio

BSDS 100 - Intro to Data Science with R



- The R Programming Language and RStudio
  - R vs. other programming languages
  - Installation
  - Shortcuts



- R is a user-friendly, open source integrated development environment (IDE).
- Known for having a rich set of packages and functions for statistical analysis
- Can run from a terminal window (using BASH scripting)

# What is R ?



“R is a language and environment for statistical computing and graphics. It is a GNU project which is similar to the S language and environment which was developed at Bell Laboratories by John Chambers and colleagues. R can be considered as a different implementation of S. There are some important differences, but much code written for S runs unaltered under R.

R provides a wide variety of statistical (linear and nonlinear modelling, classical statistical tests, time-series analysis, classification, clustering, etc.) and graphical techniques, and is highly extensible. The S language is often the vehicle of choice for research in statistical methodology, and R provides an Open Source” - R Website.



- Open source (free)
- Platform agnostic
- Highly regarded visualization capabilities (`ggplot2`)
- Read/write from/to data sources
- Massive library of data manipulation and statistical packages

# But... what about Excel?



# Excel is great for certain things...



grades

Home Insert Page Layout Formulas Data Review View

Cut Copy Paste Format

Calibri (Body) 16

Wrap Text

General

Conditional Formatting Format as Table

C11

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
		Midterm	Final	Asn #1	Asn #2	Asn #3	Asn #4	Asn #5	Asn #6	V1	V2	Final Points	Letter Grade	
1	Student													
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	Student 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+	



## Sample Data

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

## Results:

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





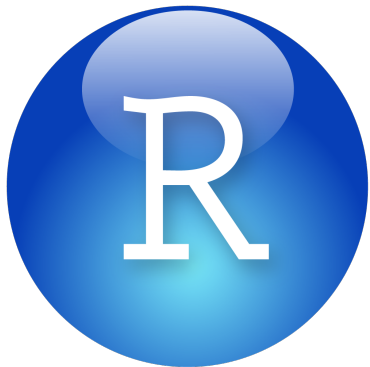
## Sample Data

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

## Results:

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

# What about Python?





- 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 of R functions are written in C (or variants)
- MATLAB, Mathematica and the NumPy package for Python are also vectorized

[https://www.noamross.net/archives/  
2014-04-16-vectorization-in-r-why/](https://www.noamross.net/archives/2014-04-16-vectorization-in-r-why/)



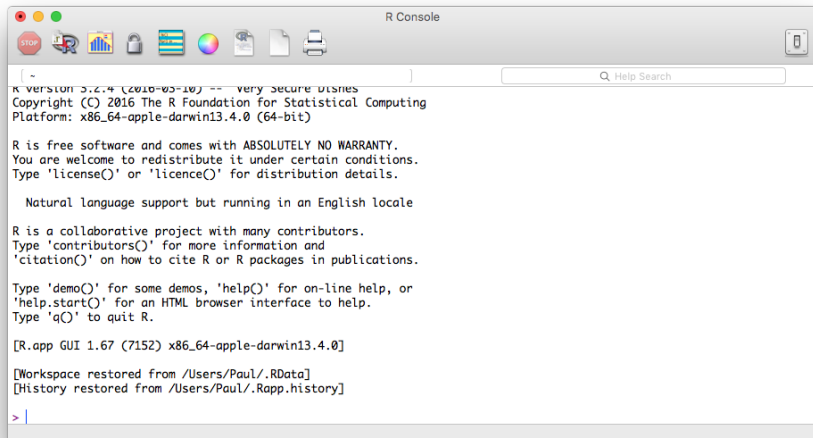
- Download and install at this website:  
<https://www.r-project.org>
- **Important:** You will have to re-install R from time-to-time to maintain the newest version so that code remains compatible! New versions generally come out every 4 - 6 months.



- RStudio has a very nice graphical user interface (GUI) that is easier to use than base R
- We will be using this throughout the course
- Make sure that you have R installed first. Then, download and install at this website:

<https://www.rstudio.com/products/RStudio/>

# The R Graphical User Interface (GUI)

A screenshot of the R Console window. The window has a title bar with standard macOS window controls (red, yellow, green buttons) and a toolbar with icons for stop, undo, redo, zoom in, zoom out, print, and a mobile device icon. The main text area contains the following output:

```
[ ~ ]
R version 3.2.4 (2016-05-10) -- "very secure wishes"
Copyright (C) 2016 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin13.4.0 (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.

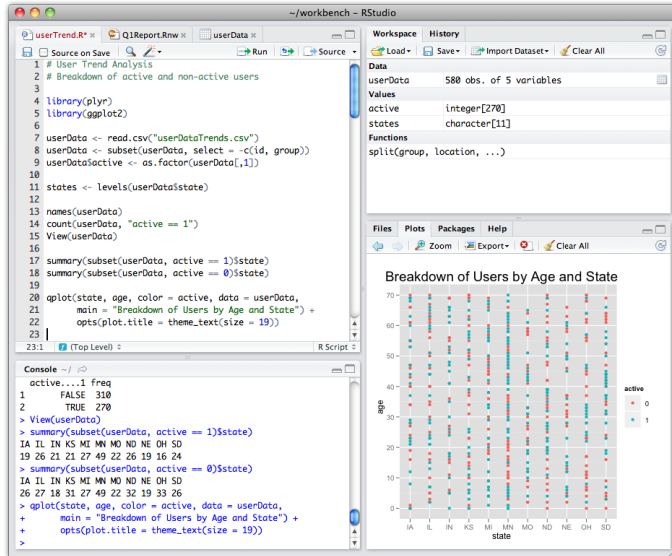
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[R.app GUI 1.67 (7152) x86_64-apple-darwin13.4.0]

[Workspace restored from /Users/Paul/.RData]
[History restored from /Users/Paul/.Rapp.history]

> |
```

# The RStudio GUI





## RStudio has 4 panels

- Console (bottom left) - where all calculations are performed
- Scripting (top left) - where writing of new scripts should be done
- Files/Packages/Help/Plots (bottom right) - for easy analysis
- Workspace & History (top right) - what is stored in your current Rsession (Variables/Data/Functions)





At their core, R and RStudio are just calculators! Try the following

$$3 + 2 = ? \quad \log(10) = ? \quad \sqrt{32} = ?$$

```
> 3 + 2
```

```
> log(10)
```

```
> sqrt(32)
```



---

Function	Action
----------	--------

---

<code>?foo</code>	Help on the function <code>foo</code>
-------------------	---------------------------------------

<code>??foo</code>	Search the help system for instances of the function <code>foo</code>
--------------------	---

<code>data()</code>	List all available example datasets contained in currently loaded packages
---------------------	--

<code>getwd()</code>	List the current working directory
----------------------	------------------------------------

<code>ls()</code>	List the objects in the current directory
-------------------	---

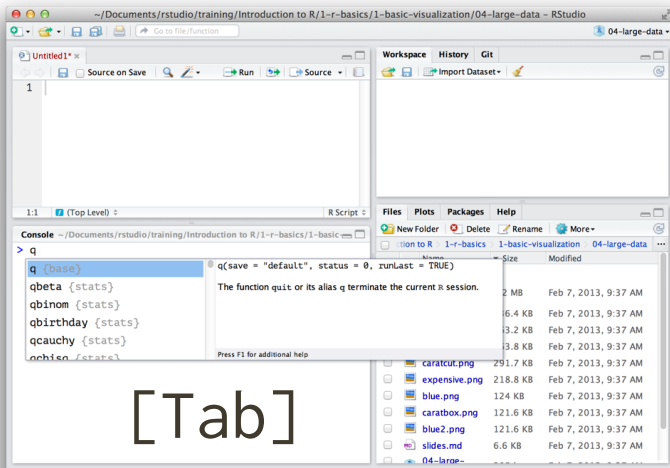
---

# Basic R Workspace Functions



Function	Action
<code>getwd()</code>	List the current working directory.
<code>setwd("mydirectory")</code>	Change the current working directory to <i>mydirectory</i> .
<code>ls()</code>	List the objects in the current workspace.
<code>rm(objectlist)</code>	Remove (delete) one or more objects.
<code>help(options)</code>	Learn about available options.
<code>options()</code>	View or set current options.
<code>history(#)</code>	Display your last # commands (default = 25).
<code>savehistory("myfile")</code>	Save the commands history to <i>myfile</i> ( default = <i>.Rhistory</i> ).
<code>loadhistory("myfile")</code>	Reload a command's history (default = <i>.Rhistory</i> ).
<code>save.image("myfile")</code>	Save the workspace to <i>myfile</i> (default = <i>.RData</i> ).
<code>save(objectlist, file="myfile")</code>	Save specific objects to a file.
<code>load("myfile")</code>	Load a workspace into the current session (default = <i>.RData</i> ).
<code>q()</code>	Quit R. You'll be prompted to save the workspace.

# Useful R Keyboard Shortcuts: Autocomplete



# Shortcuts: History

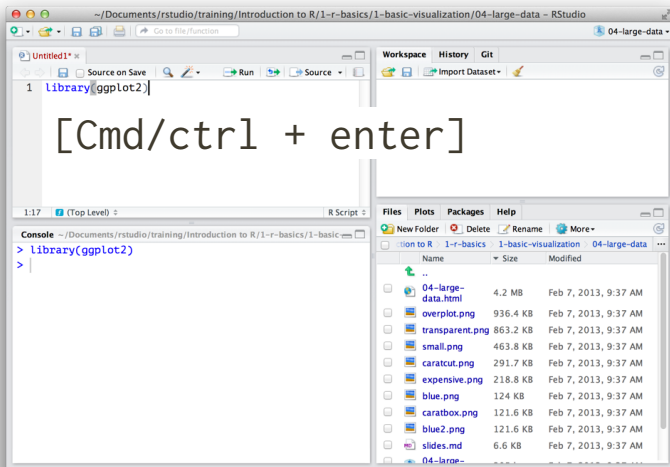


The screenshot shows the RStudio interface with the History pane open. The History pane lists the following commands:

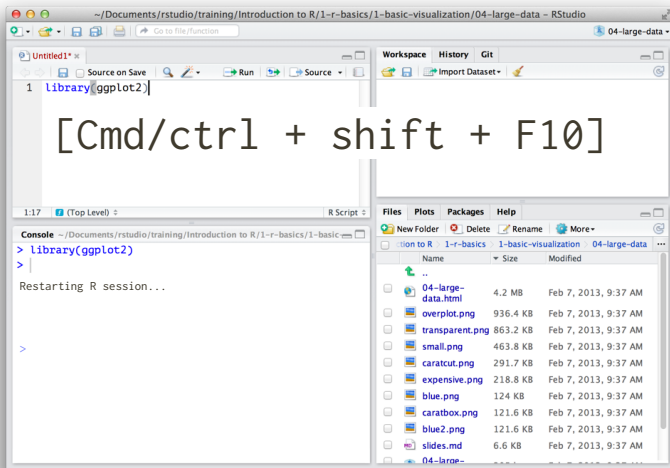
- qplot(table ~ depth, data = diamonds,
- qplot(day, data = email)
- qplot(day, mails, data = daily, geom = "line", col
- qplot(day, mails, data = daily, geom = "smooth", co
- qplot(day, variants, data = daily, geom = "line", c
- qplot(wday, hour, data = wh, size = freq)
- qplot(mpg, wt, data = mtcars)
- qplot(mpg, wt, data = mtcars, colour = cyl)

The last command is highlighted. Below the screenshot, the text **[Cmd/Ctrl + ↑]** is displayed, indicating the keyboard shortcut to navigate to the previous command in the history.

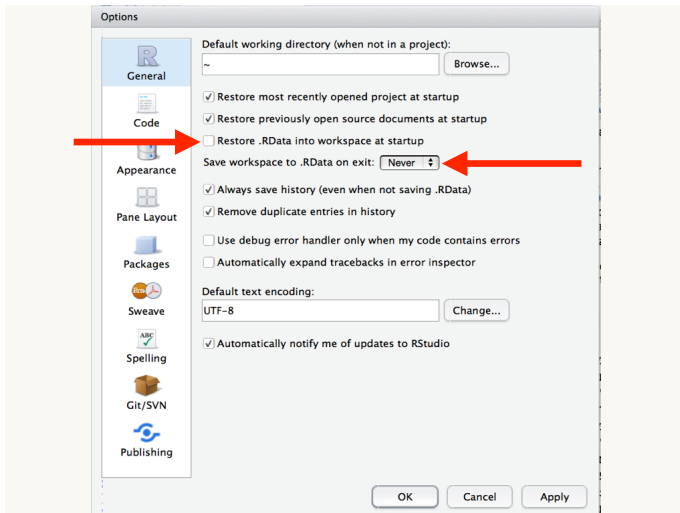
# Shortcuts: Execute Code



# Shortcuts: Restarting an R Session



# Important R Setting







- Whenever writing code, you want to be aware of your environment to ensure the fidelity of your results
- If you want to start with a clean environment and console every time, you can use:

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

- 1 `rm(list=ls())` removes all objects in the current environment
- 2 On a Mac, `cat("\014")` clear the console windows (same as `ctrl + l`)

- Pros/cons to doing this. `rm(list=ls())` can be annoying for another if they source your .R script in an environment they were working in.



- R comes built in with multiple data sets you can play with
- Many 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



- 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
- Computation depends on what type of data is contained in each variable, e.g., a data set with 2.2 million records and twenty variables, which takes approximately one minute to load into memory



- Also depends on what techniques and/or functions will be applied to the data
- The more complex and memory intensive the task, the smaller the data will be required to be
- Ex: basic plotting requires far less computational exertion than a complex statistical learning model
- **Common Definition:** *Big Data* refers to any data set that cannot be loaded into working memory on your personal computer