Getting Started with R

STAT 133

Gaston Sanchez

`github.com/ucb-stat133/stat133-fall-2016`

R

We will use R as our main computational-analytical tool for this course

About R

R entails both:

- Environment for Statistical Computing
- Programming Language

Why R?

- Allows custom analysis
- High-level scripting language
- Statistical programming language
- Interactive exploratory data analysis

Why R?

- Easy to replicate analysis
- Sound numerical methods
- Large community of contributors
- It's Free!

Why R?

As the Spanish say

- Bueno
- Bonito
- Barato

(Good, Beautiful and Inexpensive)

Some Notes

- R is a free implementation of a dialect of the **S** language
- S is the statistics and graphics environment created by John Chambers
- S was designed to blur the distinction between users and programmers
- S is a system for **interactive** data analysis

Interactive Use

- R also follows the idea of **interactive** data analysis
- *interactive*: as having a dialogue with the computer
- You type one or more commands, execute them, and get the results
- i.e. ask questions, get answers

In summary

- Environment for Statistical Computing
- Programming Language
- Free Software
- Open Source
- Extensible with packages

Learning a Programming Language

Old Chinese Proverb

- I hear and I forget
- I see and I remember
- I do and I understand

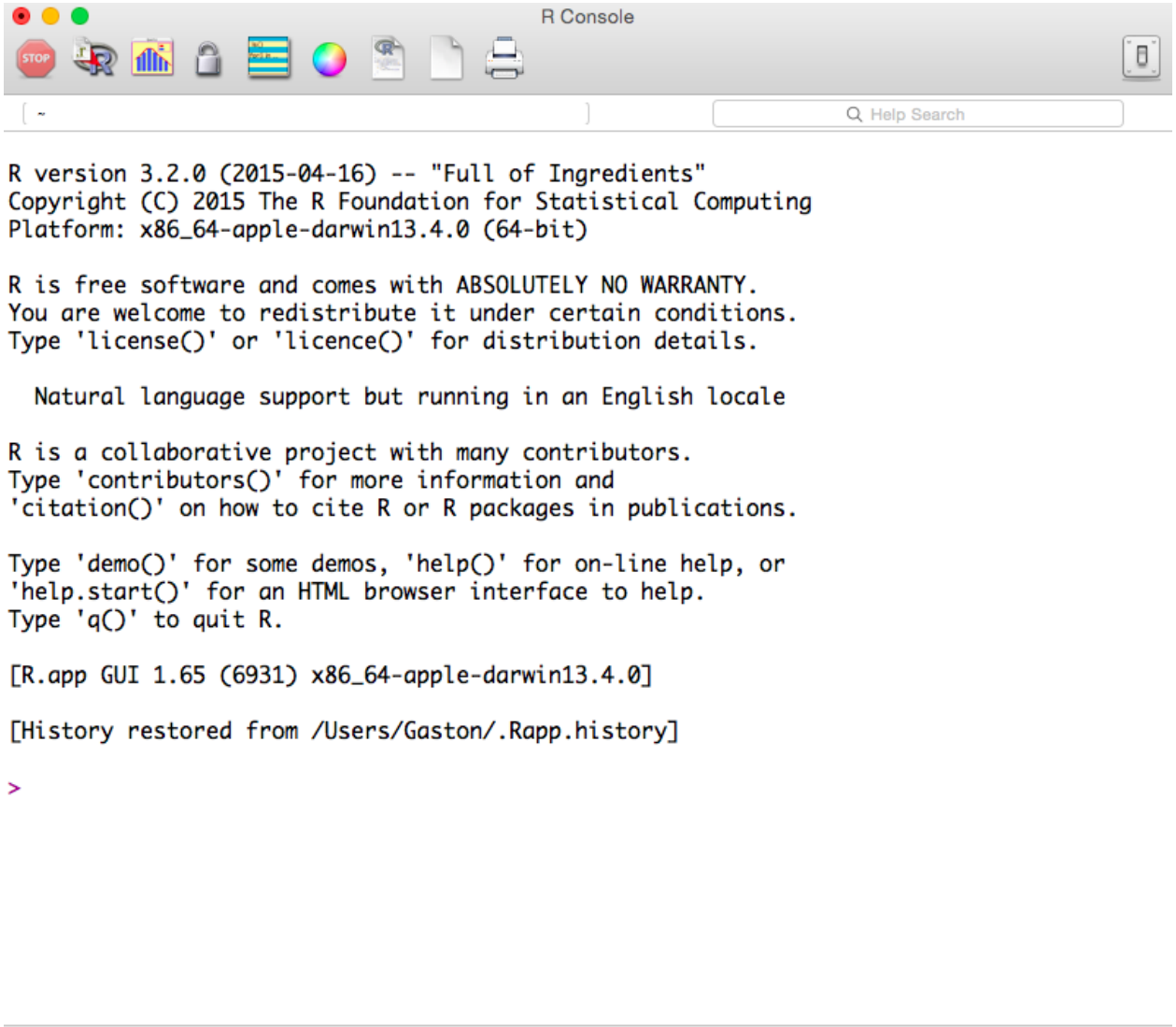
Learning a Programming Language

Learning R

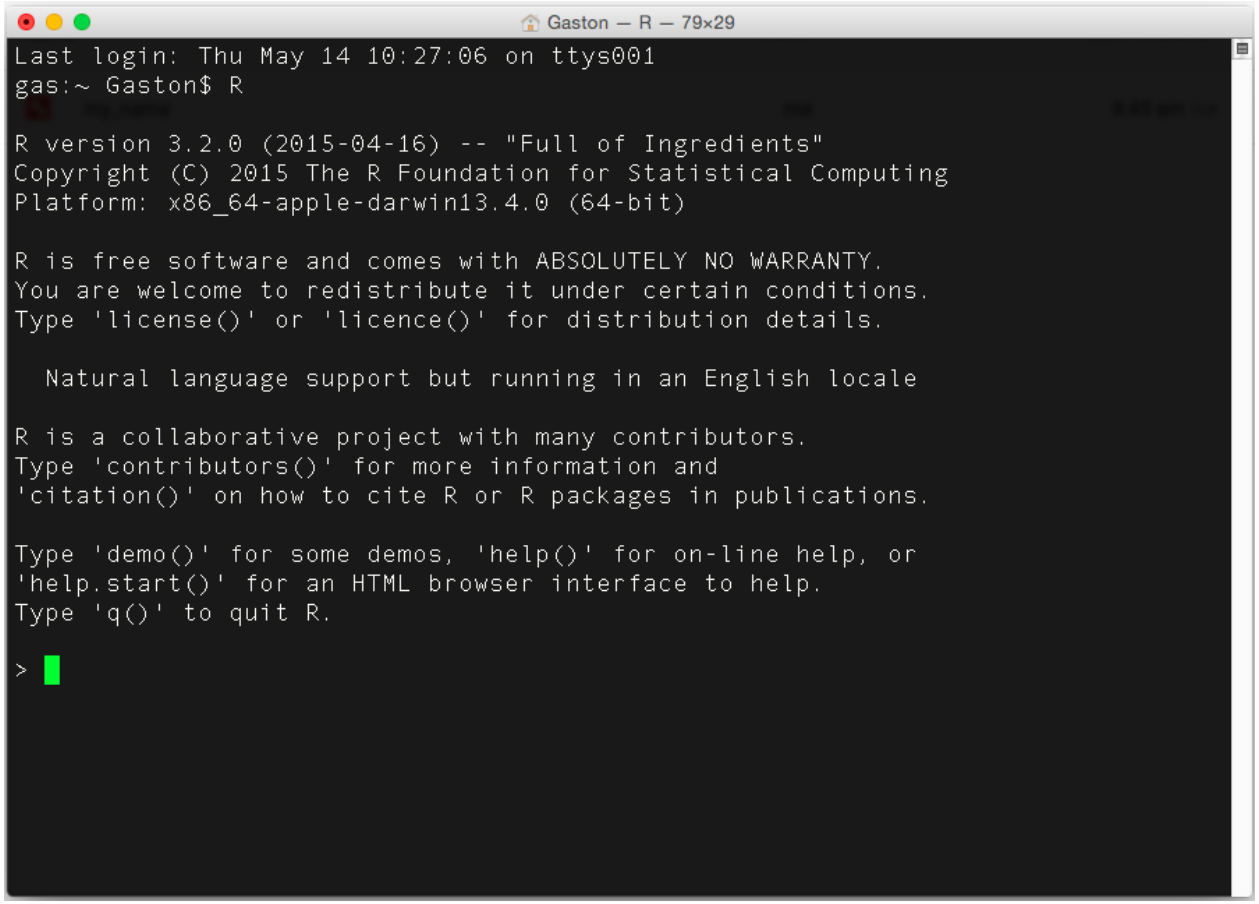
Learning R (or any programming language)

- You'll get frustrated
- It takes time to become fluent
- Lots of trials and errors
- Be patient
- Practice, practice, practice

R Console (in Mac)



R from Mac terminal



Open an R session

Entering Input

R prompt

At the R prompt, >, we type *expressions*.

```
> 5 + 3
>
> "some text"
>
> 3^2
```

(I won't show the prompt in the slides)

CalculatoR

You can use R as a calculator

```
2 + 3
4 - 1
3 * 4
10 / 2
3^3
```

CalculatoR

Using functions

```
sqrt(9)

log(5)

exp(1)

(1.3 - 5)^2 + (log(5) / 3.14)
```

Assignments

You can assign values to objects using the assignment operator or the equal sign:

```
# assignment with 'arrow'
a <- 2 + 3

# assignment with 'equal'
b = 2 * 3
```

Comments

The hash symbol # (or number sign) indicates a comment. Anything to the right of # is ignored.

```
# this is a comment
txt <- 'this is some text'

sqrt(9)  # example of square root

# -------------
# more comments
# -------------
```

R basics

R is case sensitive

```
# Z different from z
Z <- 1
z <- 2
Z + z


## [1] 3
```

R basics

Case sensitive: this means that `"hello"` is not the same as `"Hello"` or `"HELLO"`

```
hello <- "hello"
Hello <- "Hello"

# are they equal?
hello == Hello

## [1] FALSE
```

R basics

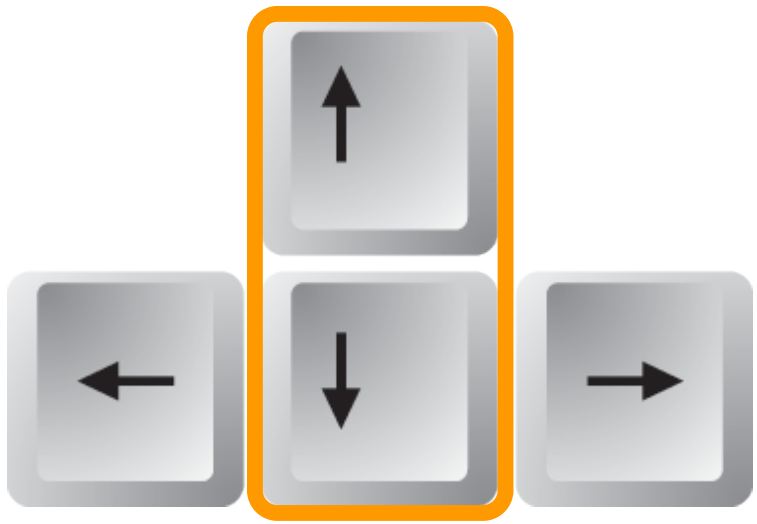
A simple plot:

```
# some coordinates
x <- 1:10
y <- x^2

plot(x, y)
```

R basics

Use the up and down arrows to navigate through previous commands or instructions:



R basics

To list all objects in your current session, you can use either `objects()` or `ls()`

```
# current objects
objects()

## [1] "a"     "b"     "hello" "Hello" "z"     "Z"

# alternatively
ls()

## [1] "a"     "b"     "hello" "Hello" "z"     "Z"
```

To clear the screen console type: `control + l`

Reserved letters, words, and commands

R has a number of reserved letters (e.g. `c`, `q`, `t`, `T`, `F`), and words (e.g. `vector`, `list`, `matrix`, `plot`, `sum`), that it uses for commands and functions.

R basics

Try typing the following:

```
c
F
T
t
sum
letters
```

R basics

Many languages use semicolons after each line. But in R there's almost no need to use semicolons

```
# no need for semicolons
2 + 4
2 + 4;

# except in this case (NOT recommended)
# (various statements in the same line)
2 + 4; A <- 2 * 5; B <- 'abc'
```

About R

- Interactive language
- You type in commands and instructions
- Invoke a computation with an expression
- Expressions are evaluated
- Returns a value or output

R's Starting Message

```
R version 3.2.0 (2015-04-16) -- "Full of Ingredients"
Copyright (C) 2015 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 Version

- Usually 2 versions of R released per year
- Each version has its own name
- e.g. R version 3.2.0 `"Full of Ingredients"`

About R (con't)

```
# GNU GPL2 license
license()

# humans behind R
contributors()

# citing R
citation()

# some demos
demo()
```

Terminate R sessions

To quit a session simply type `quit()` or `q()`

```
# saves your workspace
quit(save = "yes")
```

```
# doesn't save your workspace
quit(save = "no")
```

Terminate R sessions

- If you use `quit("yes")` or `q("yes")` R will save your workspace (the created objects and variables).
- The workspace is saved in an `.RData` file.
- Next time you open R, the saved workspace should be available.

Saved Workspace

If you previously typed `q("yes")`, open a new R session and inspect what objects do you have:

```
# list objectst in your workspace
ls()
```

Recording your work

- In addition to `quit(save = "yes")`, there's also the function `savehistory()`
- You can use `savehistory()` to save everything you did
- It may be useful to call `savehistory()` at the end of a session
- By default, the commands-history will be saved in a file called `.Rhistory` (you can use other extension)
- You can open this file in any text editor

Recording your work

Type some expressions, save your commands-history, and then quit R (without saving workspace)

```
2 * 2
2^10

# first comment
course <- "stat133"

# converting units
height_ft <- 5.9
height_in <- height_ft * 12
height_m <- height_ft * 0.3048

savehistory(file = 'test-session.R')
quit(save = "no")
```

Open the file `"test-session.R"` and see what's in it

R Console

- Minimal GUI
- The console is OK for short expressions
- The console is good as a calculator
- But very limited for longer expressions
- It's better to alternate with **source scripts**

Coding Scripts

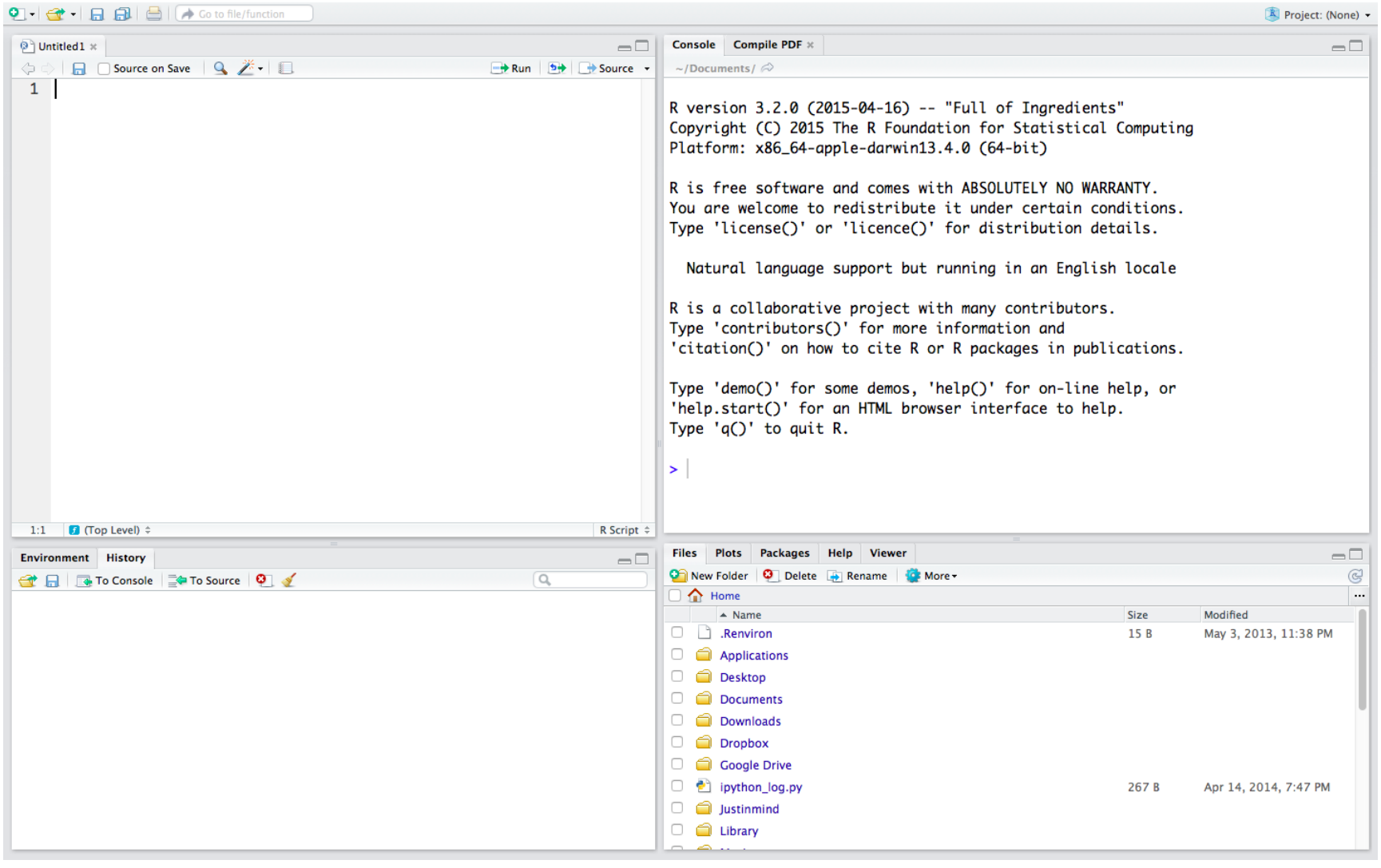
Working with R

While you can use R's console and `savehistory()`, it is better to write all your commands in a separate file. You can use an R script file (`.R` extension) or any other text file.

- you can use the R script window
- you can interact with a text editor
- you can use an IDE (e.g. RStudio)

Open RStudio

RStudio


Go to file/function
Project: (None)
Untitled1
Source on Save
Run
Source
1
1:1
(Top Level)
R Script
Console
Compile PDF
~/Documents/
R version 3.2.0 (2015-04-16) -- "Full of Ingredients"
Copyright (C) 2015 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.
>
Environment
History
To Console
To Source
Files
Plots
Packages
Help
Viewer
New Folder
Delete
Rename
More
Home
Name
Size
Modified
.Renviron
15 B
May 3, 2013, 11:38 PM
Applications
Desktop
Documents
Downloads
Dropbox
Google Drive
ipython_log.py
267 B
Apr 14, 2014, 7:47 PM
Justinmind
Library

R $\neq$ RStudio

Working with RStudio

RStudio provides an IDE that makes it really easy to work with R (everything in a single window)

- console pane
- source pane
- environment & history pane
- files, plots, and help pane

Working with RStudio

Create a new R script and type the following commands:

```
# amount
units <- 3
price <- 2.57
amount <- units * price

paste("The total amount is", amount)
```

Find out how to run the commands from the script (don't type in the console!)

Working with RStudio

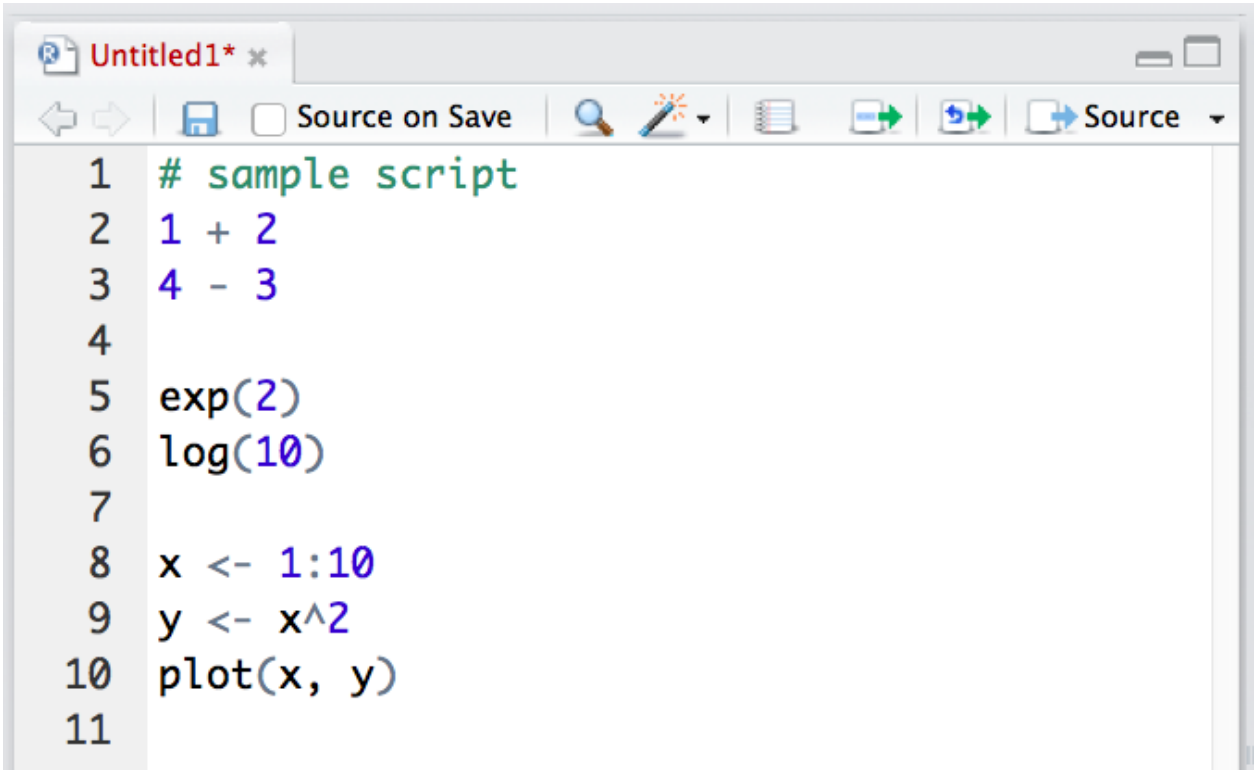
A simple bar-chart

```
# amount
fruits <- c('apple', 'orange', 'peach')
units <- c(5, 7, 3)
price <- c(0.8, 0.5, 0.6)
amount <- units * price
barplot(amount, names.arg = fruits)
```

What things can you do in RStudio with a plot?

Working with RStudio

Menu buttons of an RStudio script



Customize RStudio

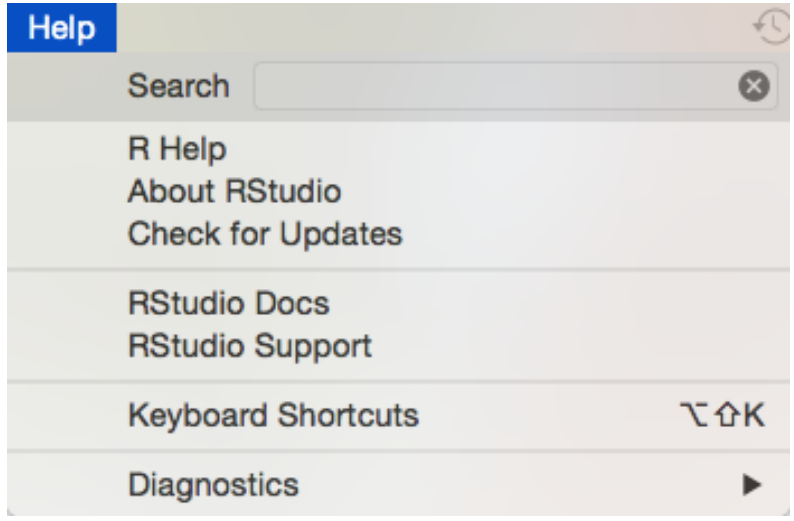
On the menu bar go to *RStudio* and then go to *Preferences*. Customize the following:

- Pane Layout
- Appearance: font type, size, and theme
- R General
- Code Editing

Also check the RStudio keyboard shortcuts (see `Help`)

RStudio Documentation

In the manu bar click on `Help` and then go to `RStudio Docs`



More at https://support.rstudio.com/hc/en-us

RStudio Docs

RStudio

Keyboard Shortcuts

Console

Description	Windows & Linux	Mac
Move cursor to Console	Ctrl+2	Ctrl+2
Clear console	Ctrl+L	Command+L
Move cursor to beginning of line	Home	Command+Left
Move cursor to end of line	End	Command+Right
Navigate command history	Up/Down	Up/Down
Popup command history	Ctrl+Up	Command+Up
Interrupt currently executing command	Esc	Esc
Change working directory	Ctrl+Shift+H	Ctrl+Shift+H

and many more

R and RStudio

- We use RStudio because it makes many things easier
- However, nothing we produce requires RStudio to be created
- RStudio is the means, NOT the end
- The important thing is R (and its packages)

R Packages

- Functions in R are structured in packages (a.k.a. *libraries**)
- The basic distribution of R comes with a number of default (pre-installed) packages
- e.g. `"stats"`, `"utils"`, `"graphics"`, etc.
- To be used, most packages need to be loaded via the function `library()`
- In addition to the default distributed packages, there is an extensive (and growing) list of contributed packages

Installing Packages

R comes with some pre-installed packages. However, one of the benefits of R is the availability of external packages. To install a package use the function `install.packages()`

```
install.packages("knitr")
```

Loading Packages

Once a package is installed, we use the function `library()` to actually load the package and be able to use its functions:

```
library("knitr")
```

Installing Packages

- `install.packages()` will install R packages in the specified directory (argument `lib`).
- If no `lib` is specified, R will download packages in your **Rlibs** directory
- In RStudio, you can use the pane with the tab **packages** to install packages

Getting Help

Command `help()`

All packages in R provide technical documentation on how to use the available functions. To have access to such documentation we use the function `help()`

```
help(vector)
```

Alternatively, we can use the question mark `?` to ask for the same type of help documentation:

```
?vector
?"+"
?"["
```

Understanding help documentation

- Function name (and package)
- Description
- Usage
- Arguments
- Details, Notes, References
- See Also
- Examples

Searching for help

The use of `help()` requires you to know exactly the name of the object you're looking help for. For a more generic search we can use `help.search()`, or the shorthand version ??

```
help.search("matrix")
??matrix
```

`help.search()` returns a list of functions and packages related with the searched term

Function apropos()

A related function is `apropos` which returns a list of functions containing the searched term

```
apropos("mean")
```

`help.search()` returns a list of functions and packages related with the searched term

Functions in packages

To find out about all the functions in a given package, use the `help()` function specifying the name of the package you are interested in:

```
help(package = "graphics")
```

Functions in packages

To get a list of packages contained in your R version, use `library()` and specify the `lib.loc` argument:

```
# deafult packages in your R version
library(lib.loc = .Library)
```

```
# installed R packages in my computer
library(lib.loc = "/Users/Gaston/Rlibs")
```

In RStudio, you can also use the pane with the tab **packages** to inspect installed packages

Function in what package?

To find which package a function belongs, we use the command `find()`

```
find("t.test")

find("mean")

find("boxplot")
```

More resources

- R website http://www.r-project.org/
- technical manuals
  http://cran.r-project.org/manuals.html
- Contributed documentation
  http://www.r-project.org/other-docs.html
- Task Views http://cran.r-project.org/web/views/
- R journal http://journal.r-project.org/

More resources

Some blogs and groups

- http://www.r-bloggers.com
- http://www.inside-r.org/
- http://www.scoop.it/t/r-for-journalists

More resources

- lots of youtube videos
- http://stackoverflow.com/questions/tagged/r
- http://stats.stackexchange.com/questions/tagged/r
- R programming wikibook
  http://en.wikibooks.org/wiki/R_Programming
- Quick-R http://www.statmethods.net/index.html
- Google

Getting Started

- **R for Beginners** by Emmanuel Paradis
  http://cran.r-project.org/doc/contrib/Paradis-rdebuts_en.pdf
- http://stats.stackexchange.com/questions/tagged/r
- R programming wikibook
  http://en.wikibooks.org/wiki/R_Programming
- Quick-R http://www.statmethods.net/index.html
- Google