Data analysis in R applied to Marine Science

Day 1. Introduction to R
27 January 2019

Dr. Tamara Huete-Stauffer – Dr. Grégoire Michoud – Dr. Daffne López-Sandoval – Dr. Malika Kheireddine

جامعة الملك عبدالله للعلوم والتقنية
King Abdullah University of Science and Technology
RSRC Red Sea Research Center

Schedule and reminders

- Sunday 27 – Thursday 31st
- Morning sessions: 9:00 -12:00
- Afternoon sessions: 13:30 – 17:00
- Coffee at 9:00 and 14:30 (**Bring your mugs!**)

- Monday 28th 14:00 Prof. Manny Aranda defense
- Monday 28th 16:00 President's Town Hall
- Thursday 31st 15:00 RSRC Town Hall

Syllabus

- Day 1: First steps
- Day 2: All you ever wanted to know about plots
- Day 3: Statistics is your friend
- Day 4: Reproduceable workflows (Diversity)
- Day 5: Useful packages + Working with your data

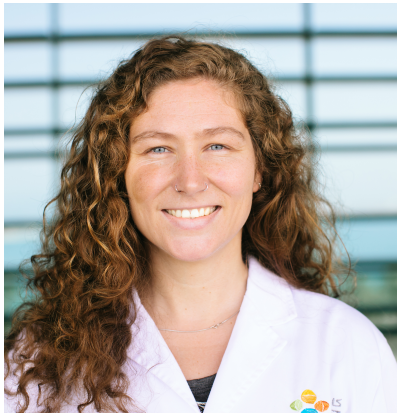
What you will need

- Internet connection
- Google Drive shared folder: https//drive.google.com/drive/folders/1iEWUA7s53lhqz8EEybnUPeUYYiq491LP?usp=sharing
- R: https://cloud.r-project.org/
- Rstudio: https://www.rstudio.com/products/rstudio/download/

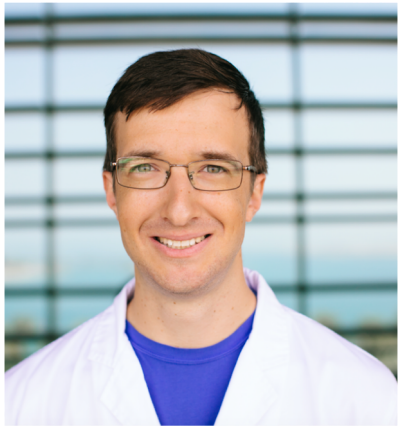
and coffee...

https://commons.wikimedia.org/wiki/File:Coffee_cup_flat.svg

Instructors



Tamara Huete-Stauffer
Postdoctoral fellow
Prof. Xelu Morán
Marine microbial ecology
tamara.huetestauffer@kaust.edu.sa



Grégoire Michoud
Postdoctoral fellow
Prof. Daniele Daffonchio
Microbiology of brine pools
gregoire.michoud@kaust.edu.sa

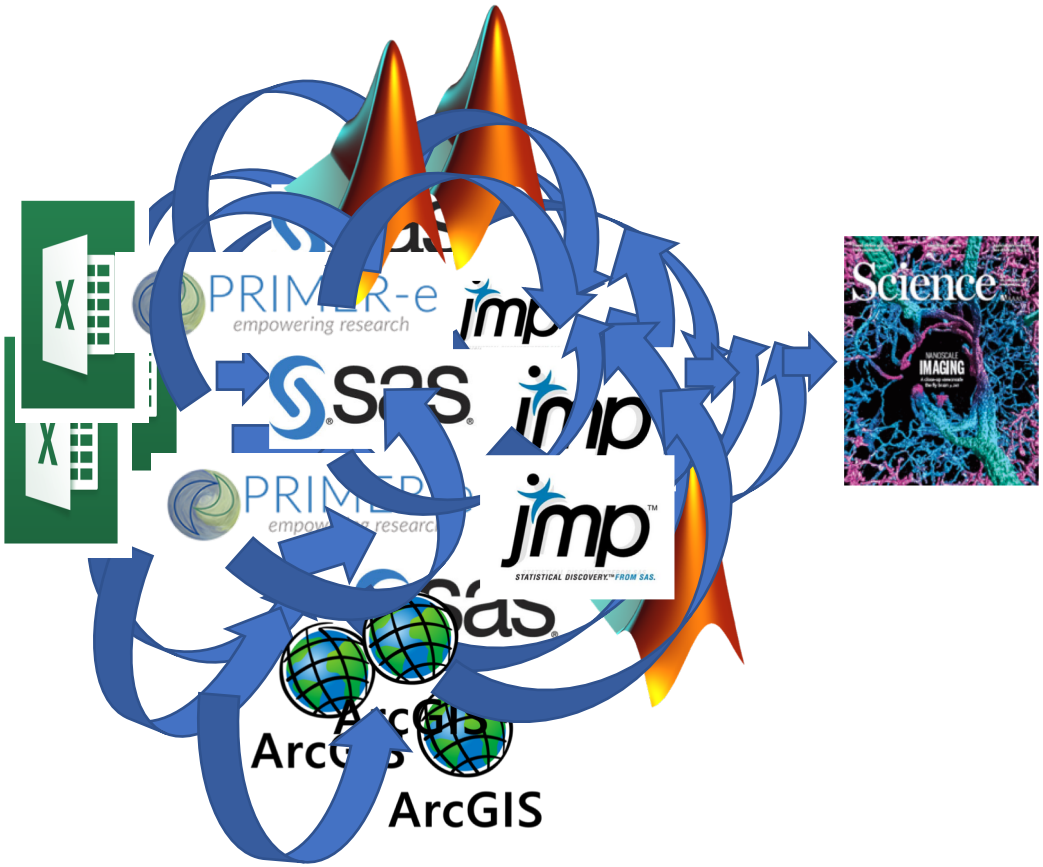
Daffne López-Sandoval
Postdoctoral Fellow
Prof. Susana Agustí
Phytoplankton ecology
daffne.lopezsandoval@kaust.edu.sa

Intro to R

Why R?

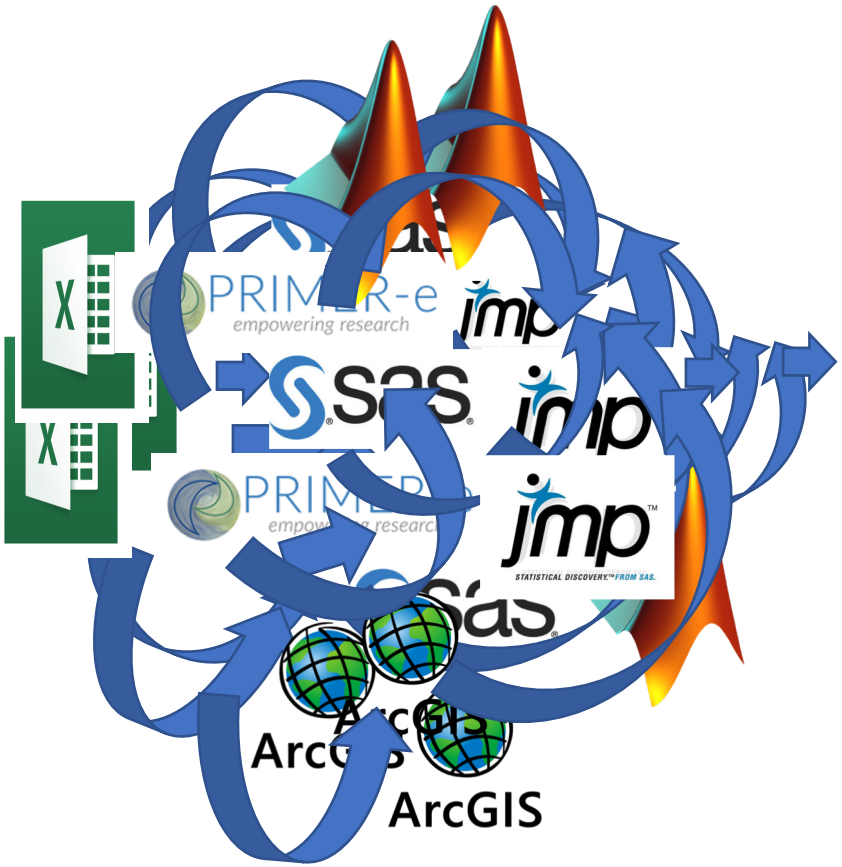

PRIMER-e
empowering research
jmp
STATISTICAL DISCOVERY.™ FROM SAS.
sas
ArcGIS
Science
NANOSCALE
IMAGING

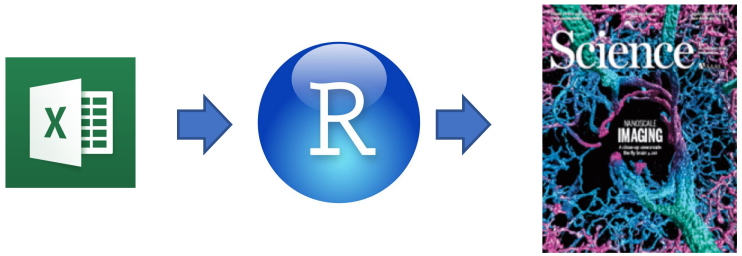
Why R?


PRIMER-e
empowering research
jmp
sas
jmp
jmp
STATISTICAL DISCOVERY™ FROM SAS.
sas
ArcGIS
ArcGIS

Science
NANOSCALE
IMAGING

Why R?





http://www.sciencemag.org/; https://en.m.wikipedia.org/wiki/File:Matlab_Logo.png; https://commons.wikimedia.org/wiki/File:Microsoft_Excel_2013_logo.svg; https://www.primer-e.com/; https://commons.wikimedia.org/wiki/File:ArcGIS_logo.png; https://www.jmp.com/en_us/home.html; https://www.sas.com/en_ae/home.html

Why R?

PROS

- **FREE!**
- Open source
- Very versatile
  - data management
  - stats
  - plots
  - maps
- Everyone is doing it
  - Someone else has had the same or similar problem!
  - The solution is out there
- Reproducibility
  - Know what you did (trace your steps)
    - Research and publishing takes for ever
  - Publish what you did

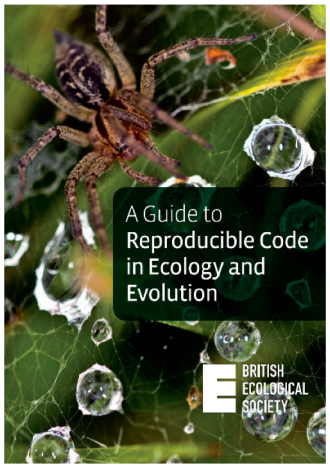
CONS

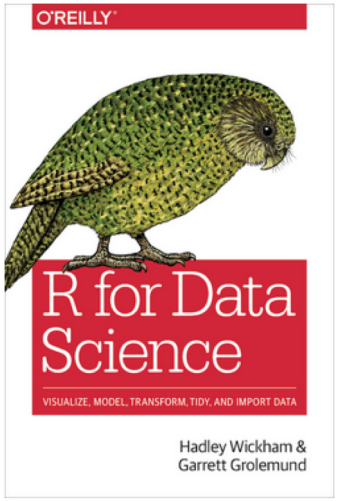
- Steep learning curve
- With power comes responsibility
  - Know what you are doing
- There is no code standard

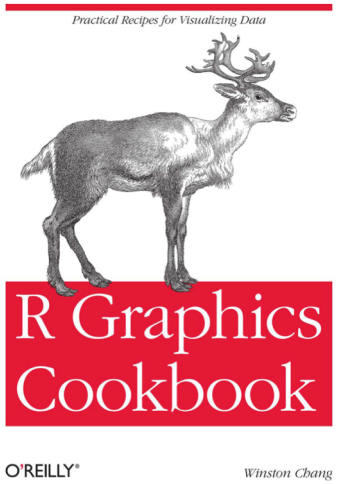
https://zerotorhero.files.wordpress.com/2012/12/rhero.jpg

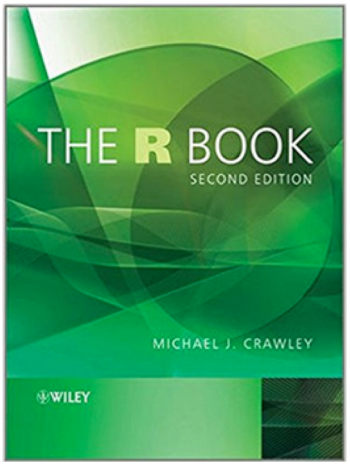
Key references

- The world wide web (www.)!!!!
  - Quick-R
  - Stack overflow
  - https://www.r-bloggers.com/
  - Learn how to search- using the right terms!
- R for Data Science
  - http://r4ds.had.co.nz/
- RStudio help pages and cheatsheats
  - https://www.rstudio.com/online-learning/#R
- Guide to reproducible code
  https://methodsblog.wordpress.com/2017/12/04/reproducible-code/
- The R book (2007)









R is dynamic

- R is constantly evolving and growing
  - Be aware that new packages appear very often
  - Sometimes, some packages disappear if they are old or nobody is keeping them up to date to the new versions
  - Some packages are redundant (they do the same thing)
  - Some functions that do different things have the same name (they come from different packages)

https://all-free-download.com/free-vector/download/three-green-plant-growth-cycle_312598.html

R is dynamic

- R is constantly evolving and growing
  - Be aware that new packages appear very often
  - Sometimes, some packages disappear if they are old or nobody is keeping them up to date to the new versions
  - Some packages are redundant (they do the same thing)
  - Some functions that do different things have the same name (they come from different packages)
- There is always more than one way to reach the **same solution**

https://all-free-download.com/free-vector/download/three-green-plant-growth-cycle_312598.html

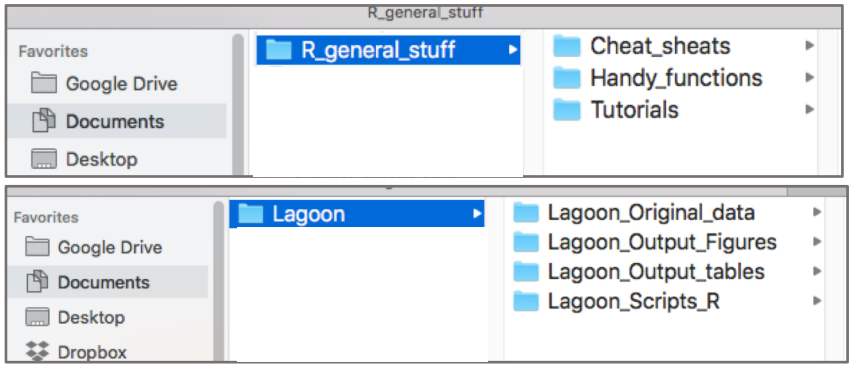
R is dynamic

- R is constantly evolving and growing
  - Be aware that new packages appear very often
  - Sometimes, some packages disappear if they are old or nobody is keeping them up to date to the new versions
  - Some packages are redundant (they do the same thing)
  - Some functions that do different things have the same name (they come from different packages)
- There is always more than one way to reach the **same solution**
- There are different coding styles
  - Base: the classical
  - Tidyverse: the modern. The community is moving towards this way

https://all-free-download.com/free-vector/download/three-green-plant-growth-cycle_312598.html

R can get messy: Helpful hints

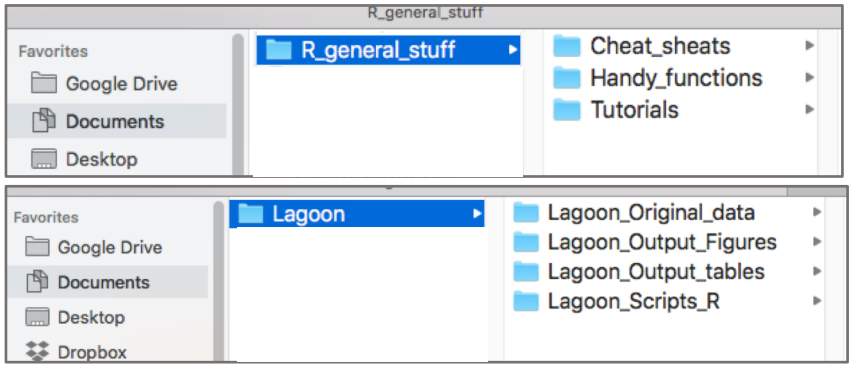
1. Folder setup



- 1 General R folder
- Different folders for each project

R can get messy: Helpful hints

1. Folder setup



- 1 General R folder
- Different folders for each project

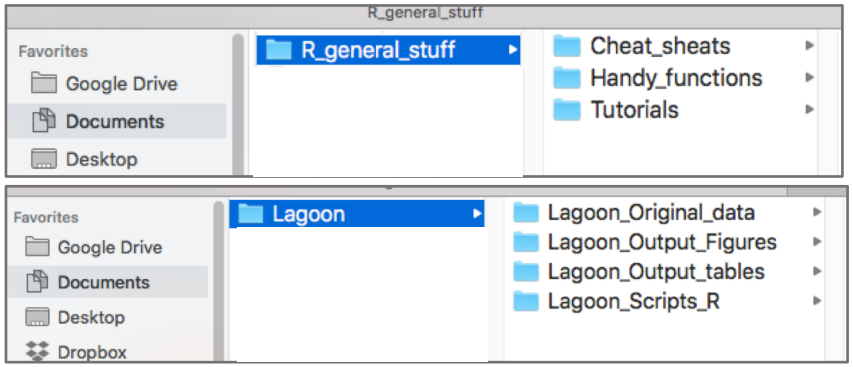
2. Script names

```
01_lagoon-nutrient_data_load.R
02_lagoon-nutrient_clean_data.R
03_lagoon-nutrient_regressions.R
```

Consecutive numbers for sequential steps or for newest versions

R can get messy: Helpful hints

1. Folder setup



- 1 General R folder
- Different folders for each project

2. Script names

```
01_lagoon-nutrient_data_load.R
02_lagoon-nutrient_clean_data.R
03_lagoon-nutrient_regressions.R
```

Consecutive numbers for sequential steps or for newest versions

3. Script structure

1. Start script loading necessary packages
2. Then import data or open workspace
3. Then start coding!

Let's be tidy with the course data...

- Download the folder form the google Drive link

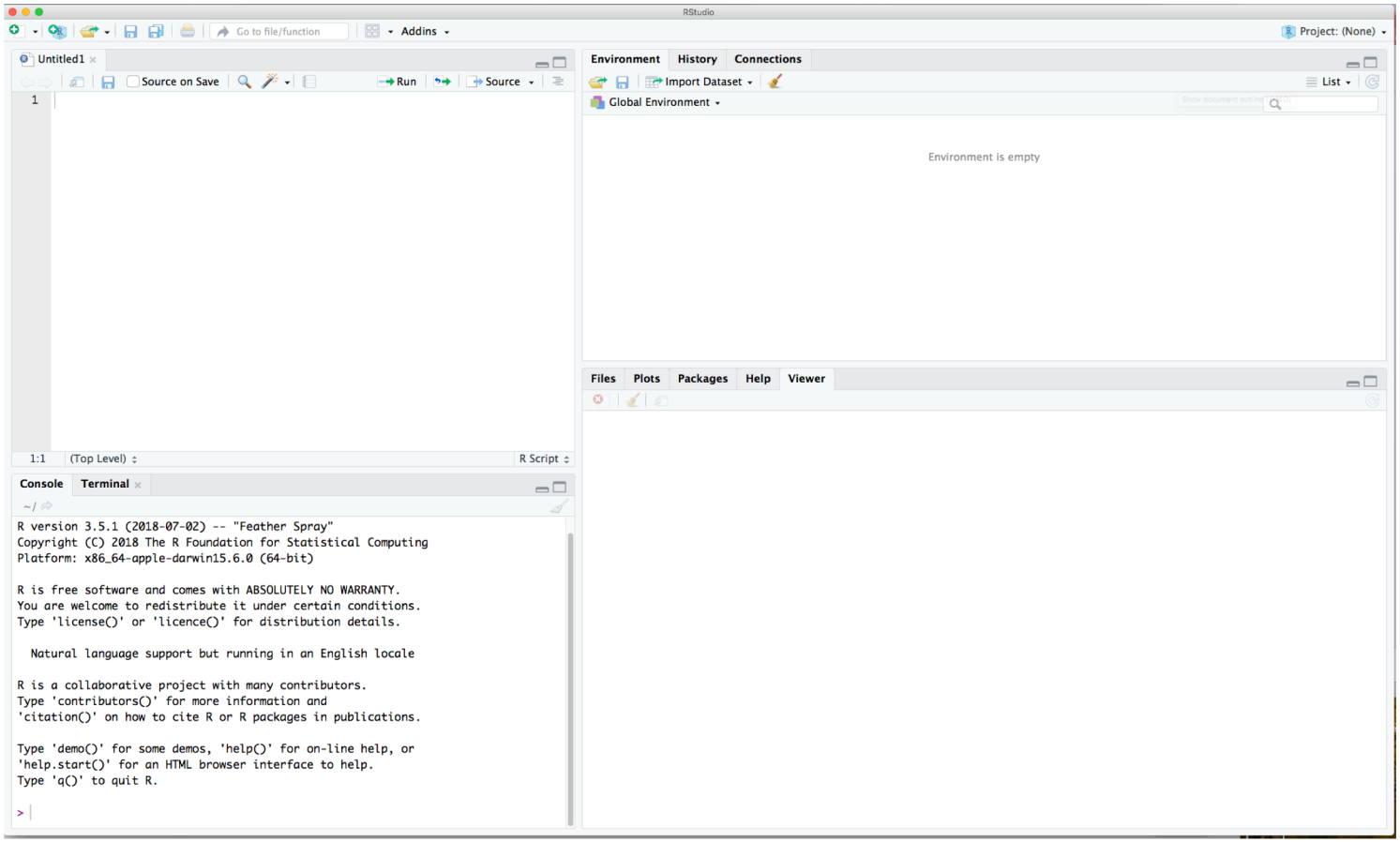
https//drive.google.com/drive/folders/1iEWUA7s53lhqz8EEybnUPeUYYiq491LP?usp=sharing

- Keep the same folder structure
- Follow our instructions for creating new scripts and where to save them
- At the end of each day, we will upload the presentations and the full scripts with all the solutions

RStudio

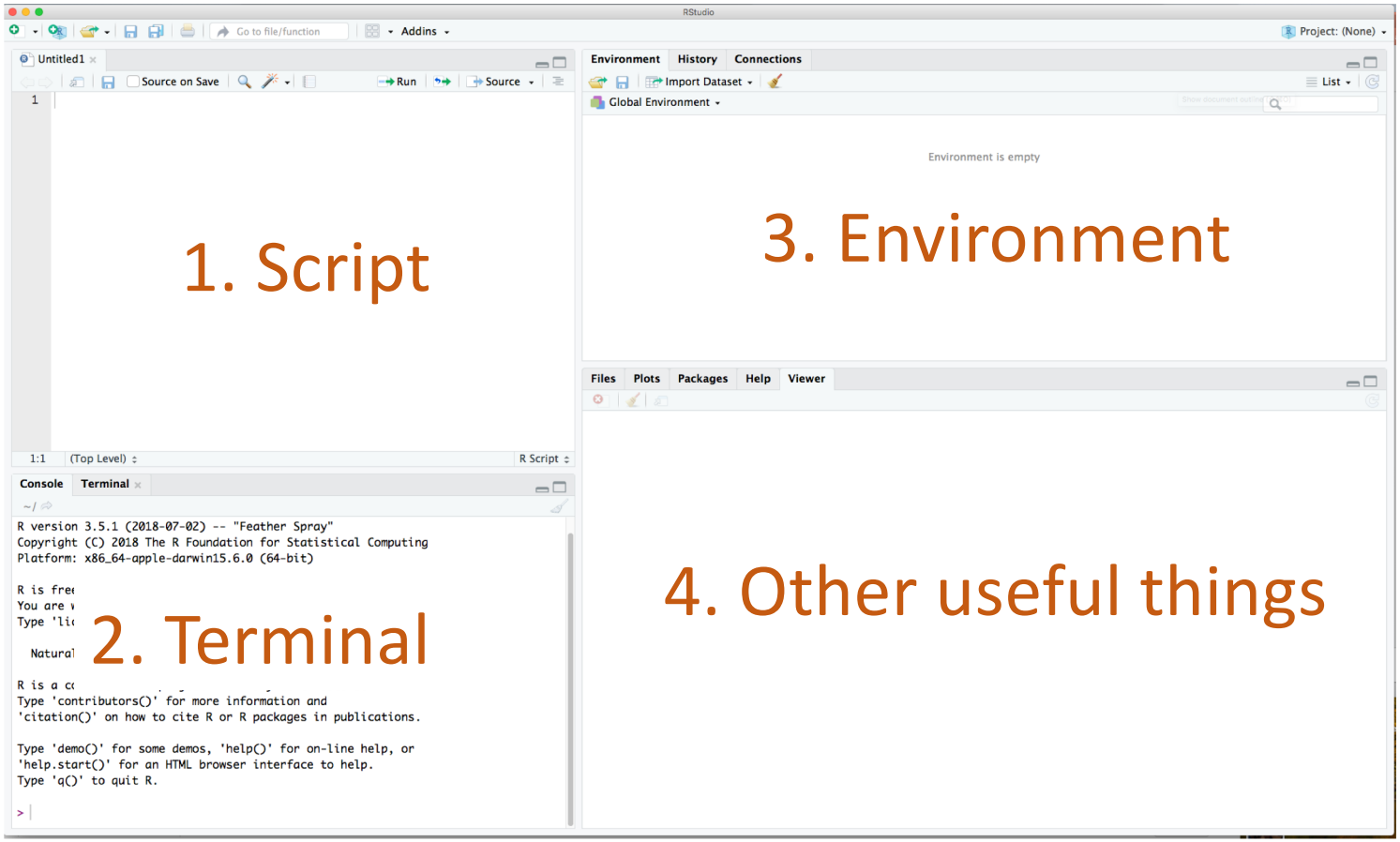
RStudio basics

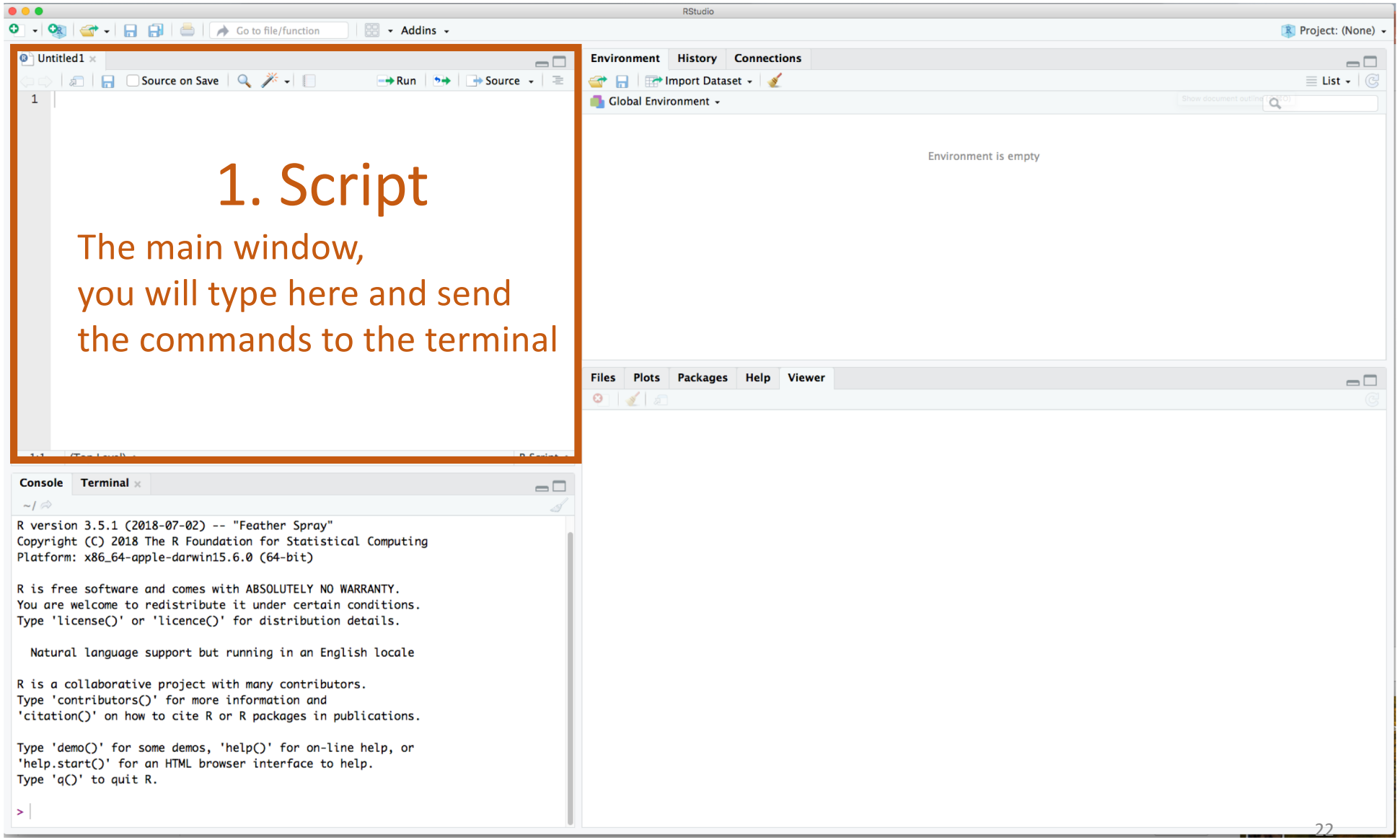
- More user friendly than plain R

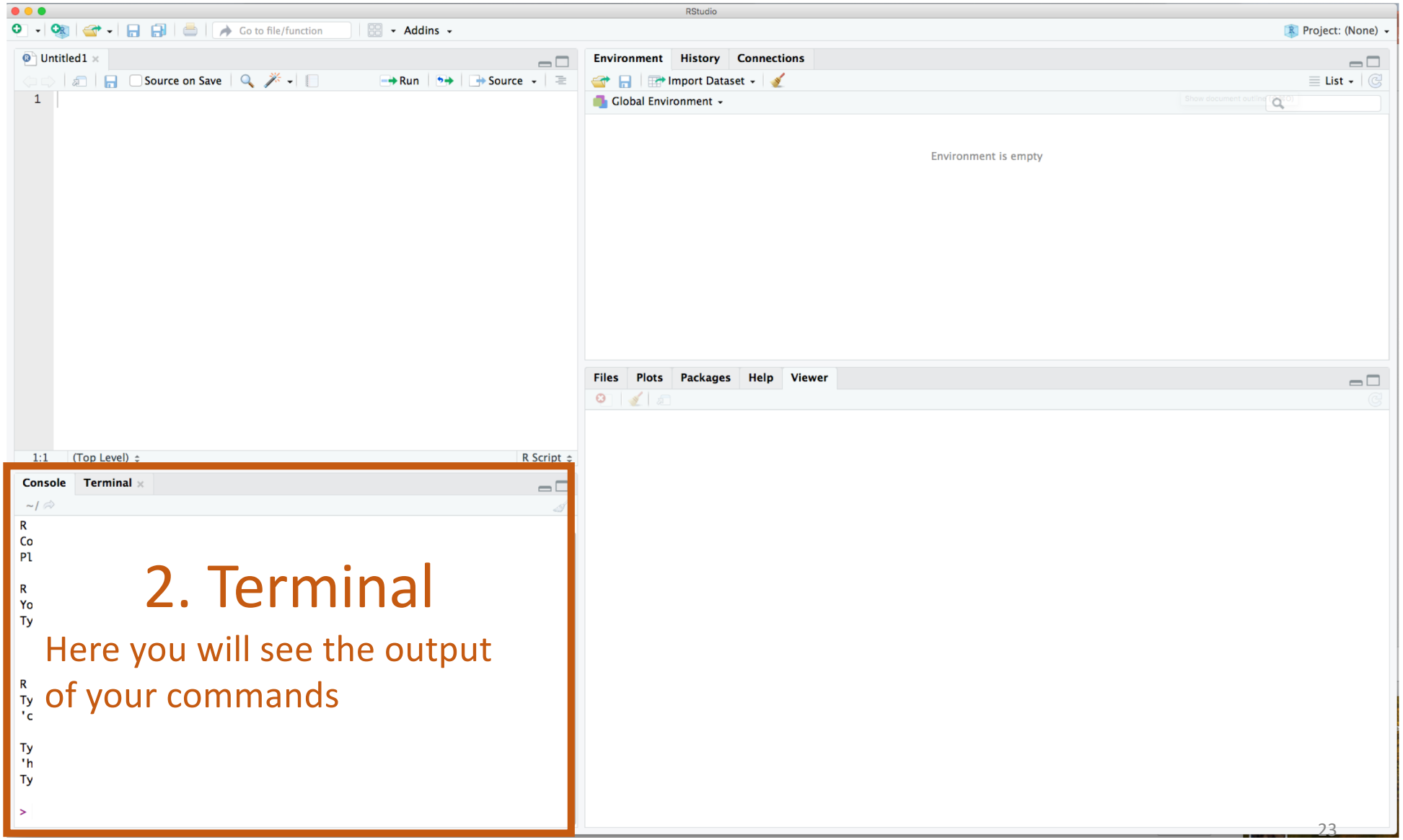


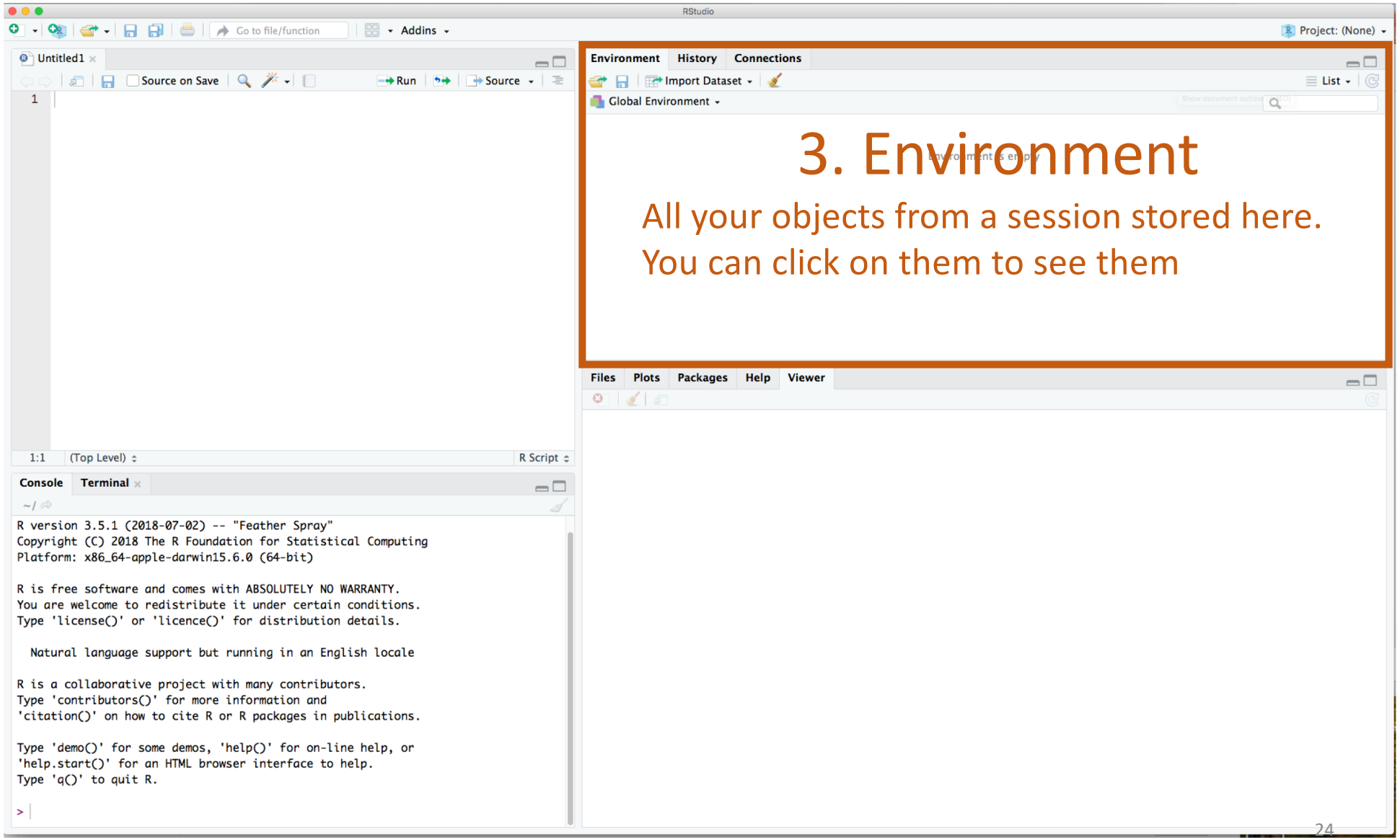
RStudio basics

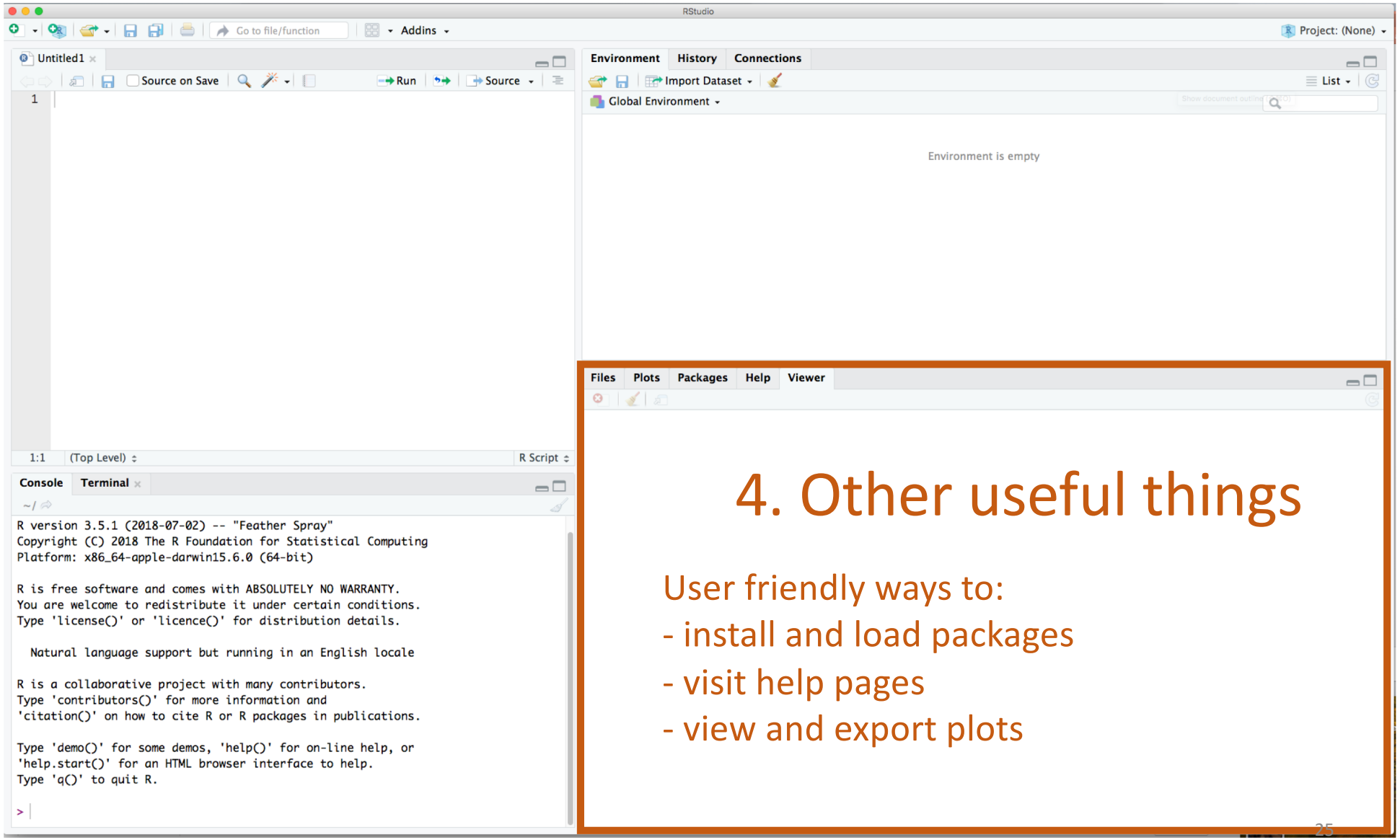
- More user friendly than plain R

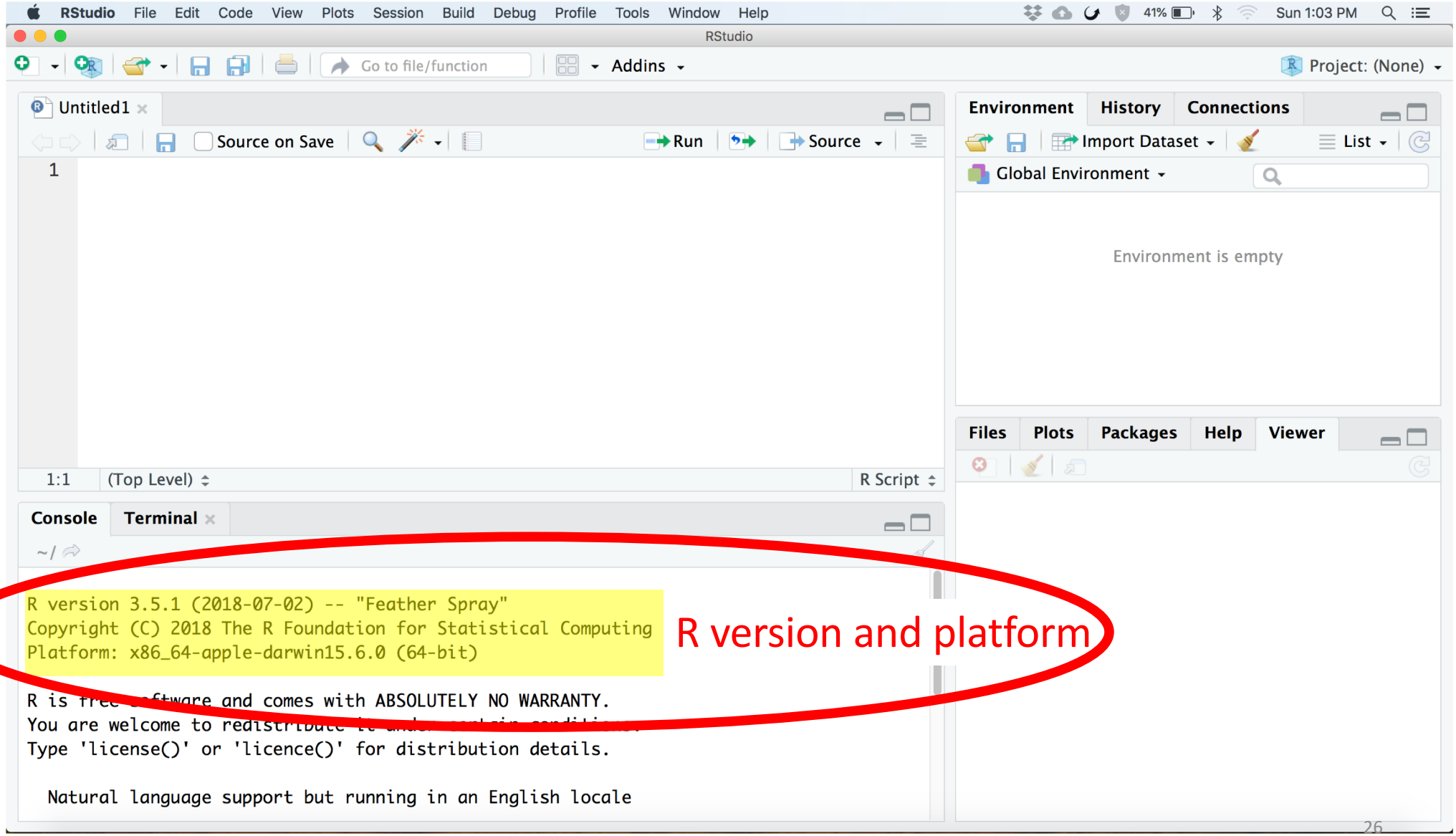



RStudio
Addins
Project: (None)
Untitled1
Source on Save
Run
Source
1
1. Script
The main window,
you will type here and send
the commands to the terminal
Environment
History
Connections
Import Dataset
List
Global Environment
Environment is empty
Files
Plots
Packages
Help
Viewer
Console
Terminal
R version 3.5.1 (2018-07-02) -- "Feather Spray"
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin15.6.0 (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.
>

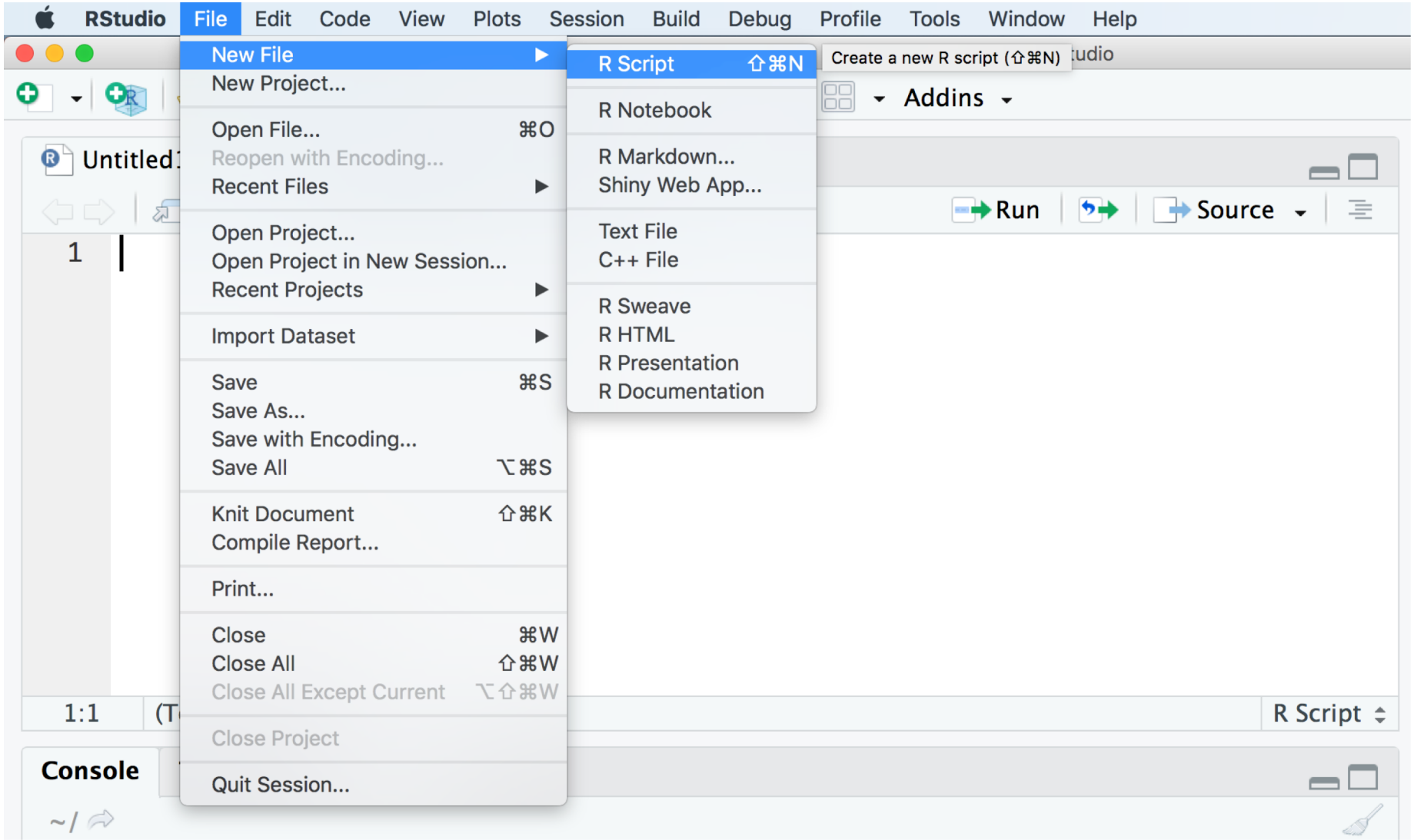

RStudio
Go to file/function
Addins
Project: (None)
Untitled1
Source on Save
Run
Source
1
1:1
(Top Level)
R Script
Environment
History
Connections
Import Dataset
List
Global Environment
Environment is empty
Files
Plots
Packages
Help
Viewer
Console
Terminal
~/
2. Terminal
Here you will see the output of your commands


RStudio
Go to file/function
Addins
Project: (None)
Untitled1
Source on Save
Run
Source
1
1:1
(Top Level)
R Script
Console
Terminal
R version 3.5.1 (2018-07-02) -- "Feather Spray"
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin15.6.0 (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
Connections
Import Dataset
List
Global Environment
3. Environment
All your objects from a session stored here.
You can click on them to see them
Files
Plots
Packages
Help
Viewer

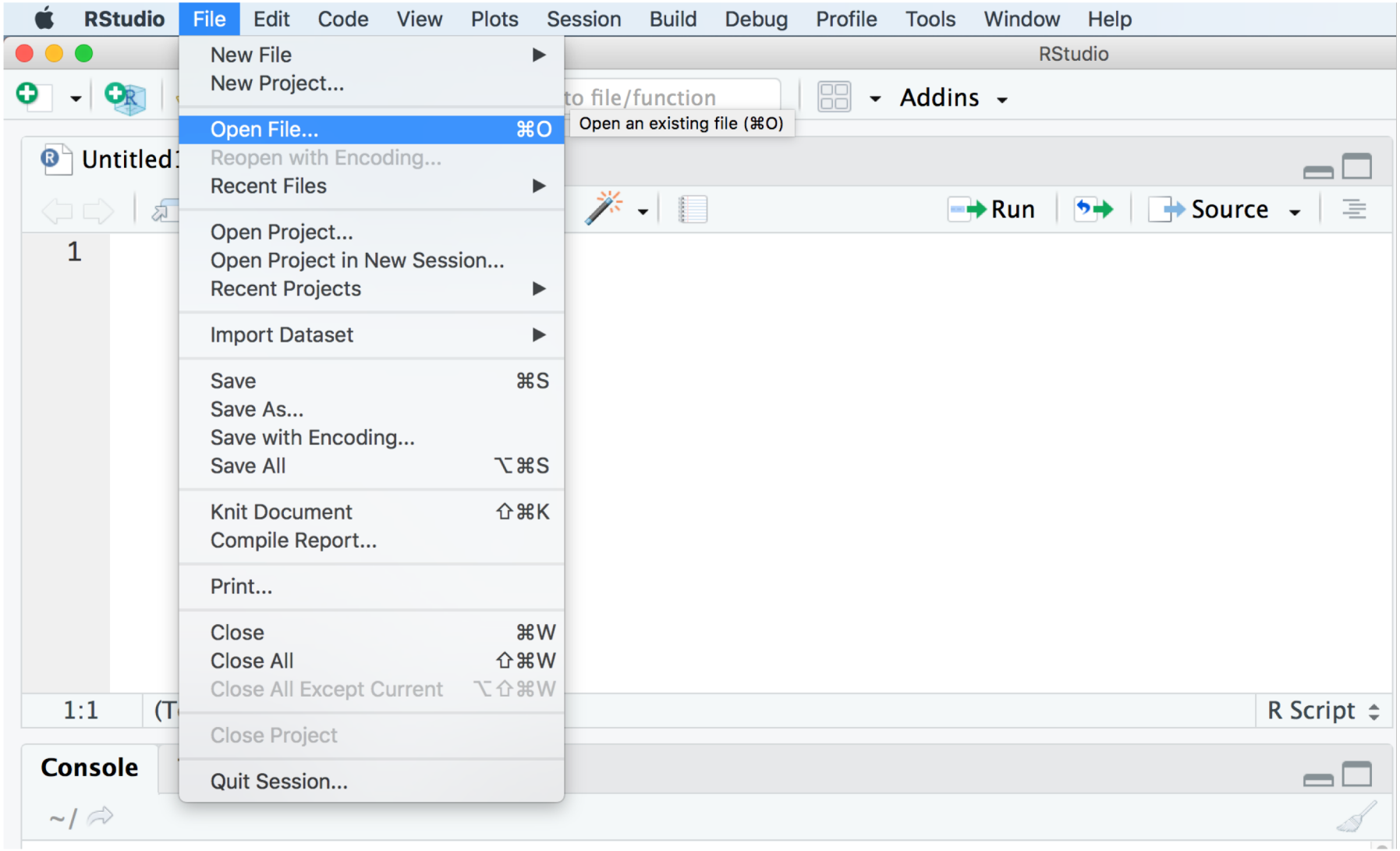

RStudio
Go to file/function
Addins
Project: (None)
Untitled1
Source on Save
Run
Source
1
1:1
(Top Level)
R Script
Console
Terminal
~/
R version 3.5.1 (2018-07-02) -- "Feather Spray"
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin15.6.0 (64-bit)
R is free software 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
Connections
Import Dataset
List
Global Environment
Environment is empty
Files
Plots
Packages
Help
Viewer
4. Other useful things
User friendly ways to:
- install and load packages
- visit help pages
- view and export plots


RStudio File Edit Code View Plots Session Build Debug Profile Tools Window Help
Sun 1:03 PM
RStudio
Go to file/function
Addins
Project: (None)
Untitled1
Source on Save
Run
Source
1
1:1
(Top Level)
R Script
Console
Terminal
R version 3.5.1 (2018-07-02) -- "Feather Spray"
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin15.6.0 (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
Environment
History
Connections
Import Dataset
List
Global Environment
Environment is empty
Files
Plots
Packages
Help
Viewer
R version and platform

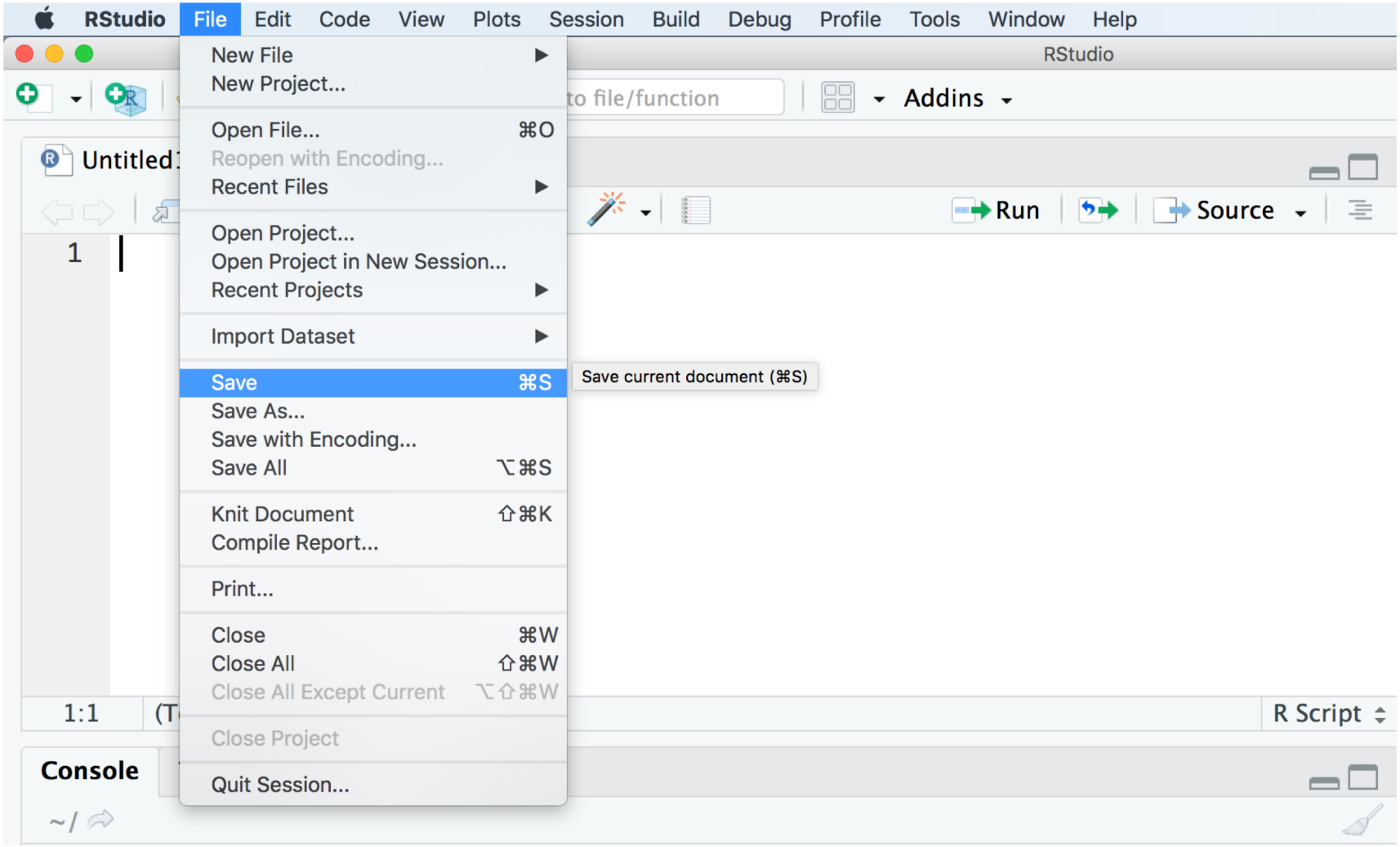
- Create new script



- Open an existing script

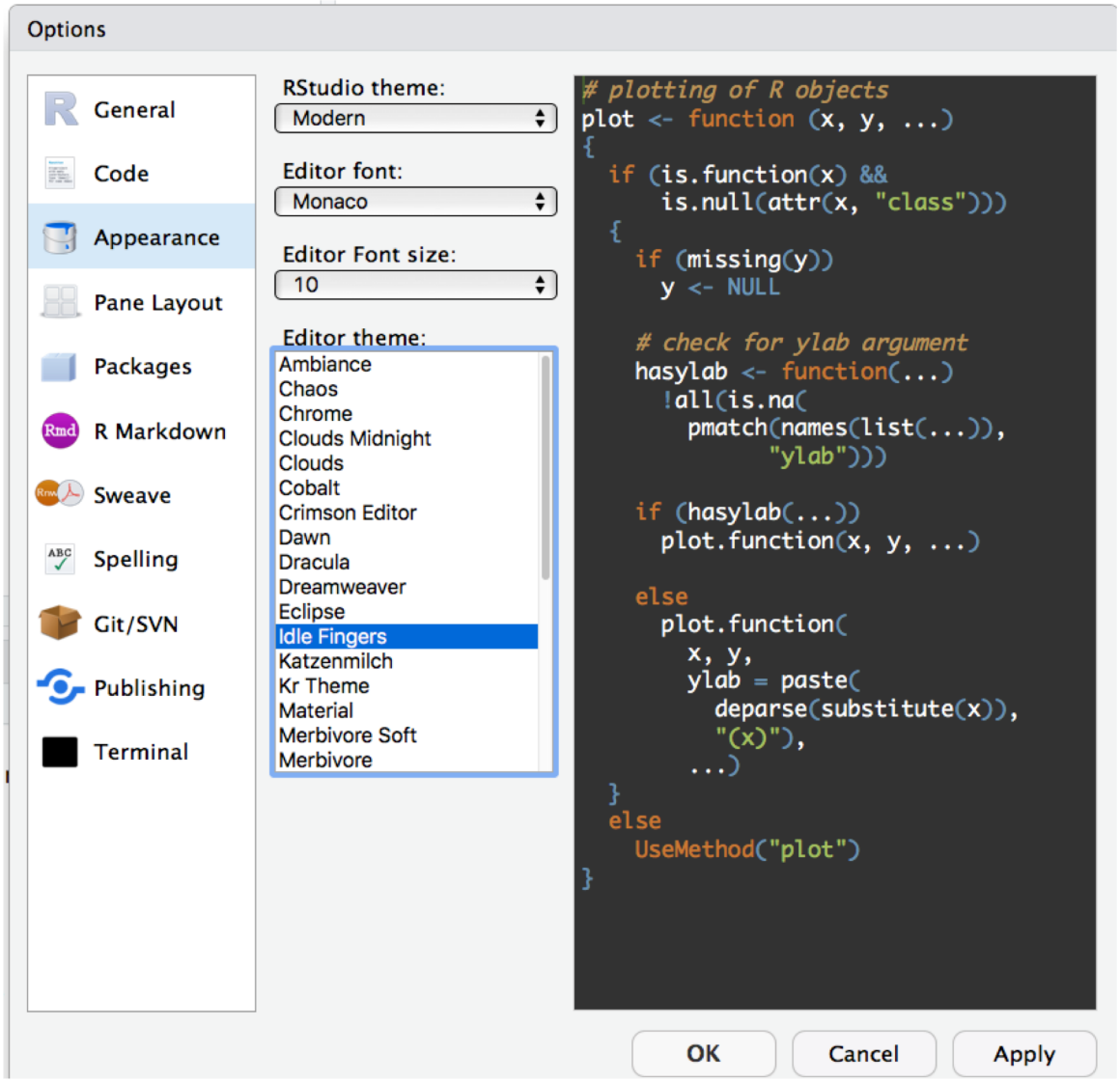


- Save a script

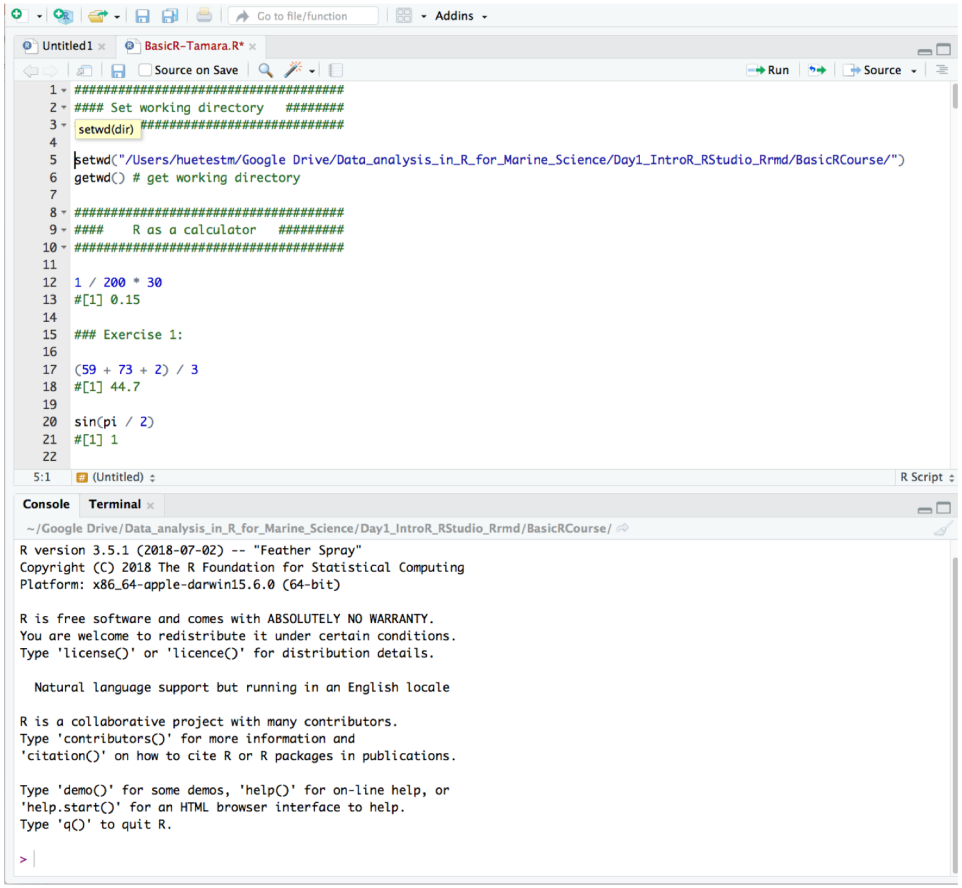


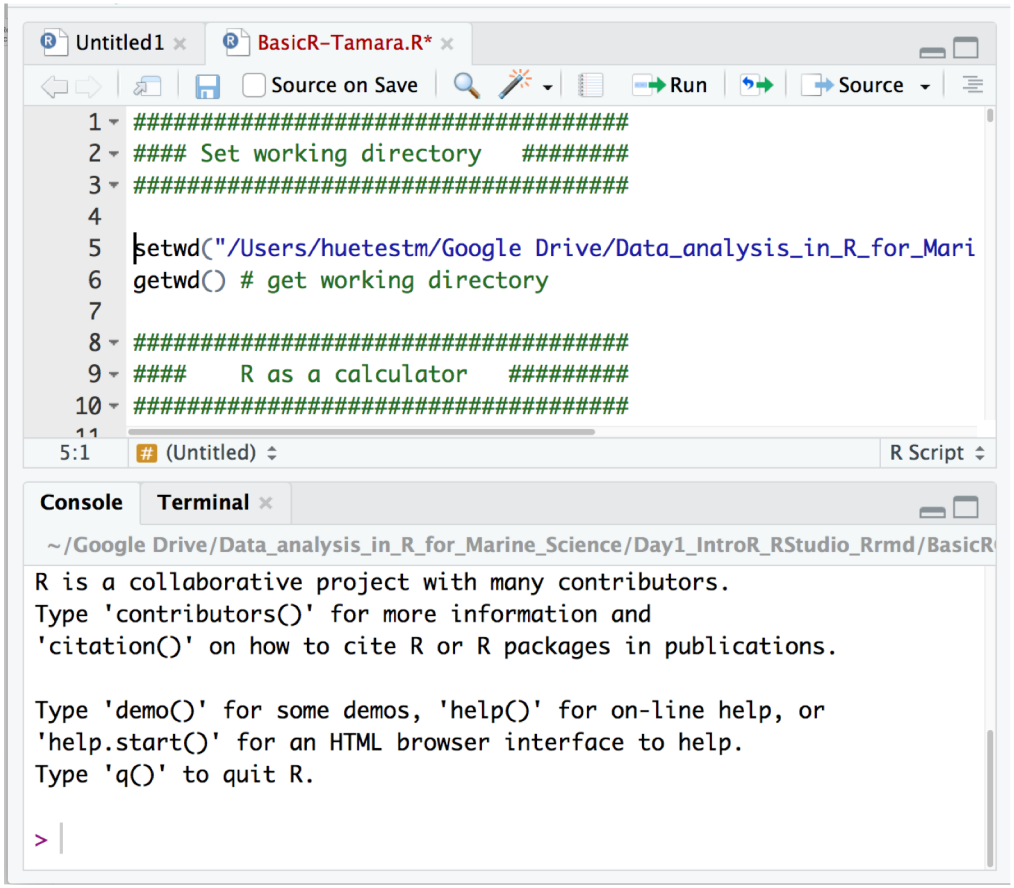
Change RStudio appearance
(*or how to look cool*)

RStudio →
Preferences →
Appearance

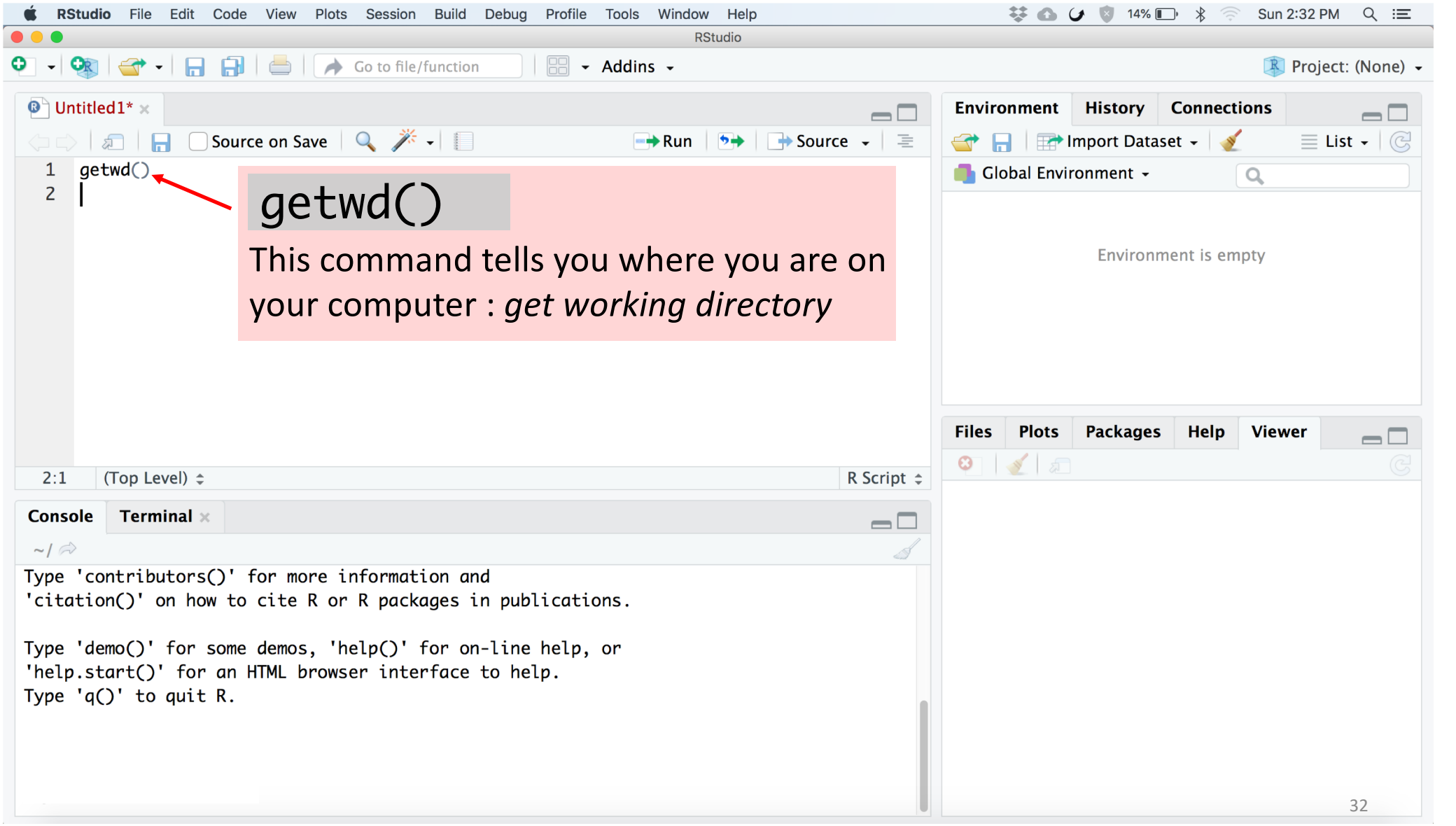


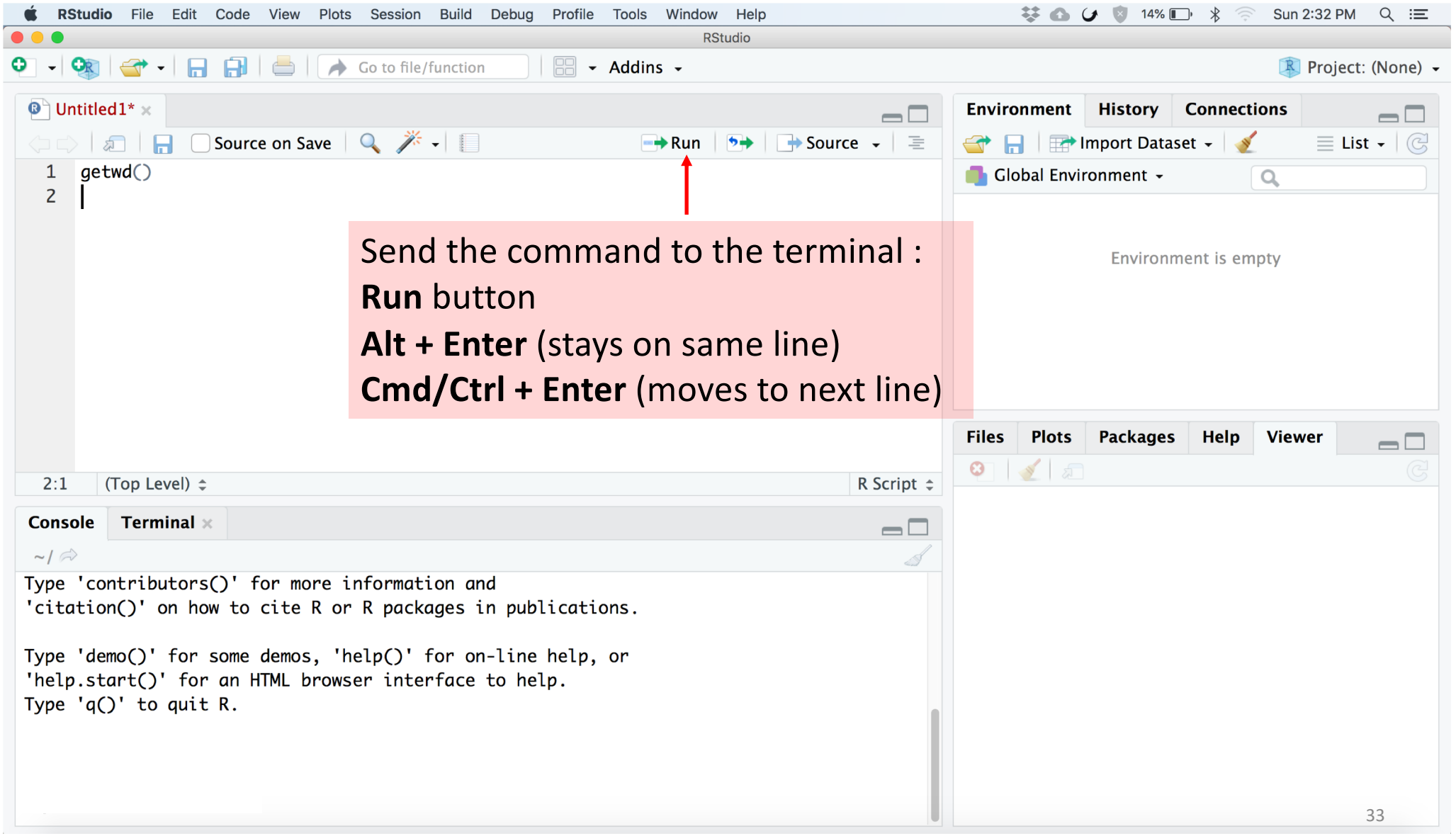
Increase font size → **Cmd +** *Ctrl for Windows

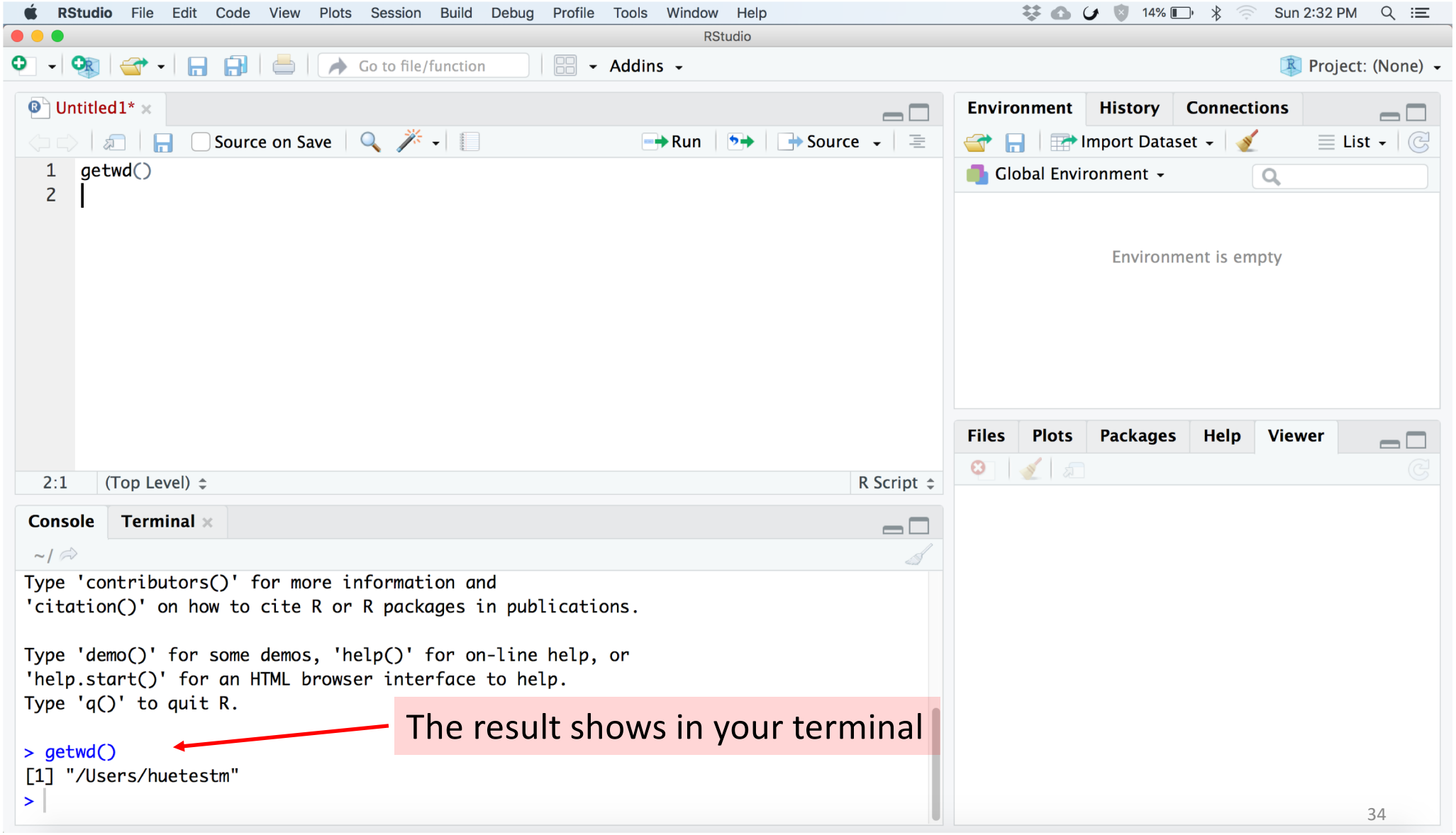


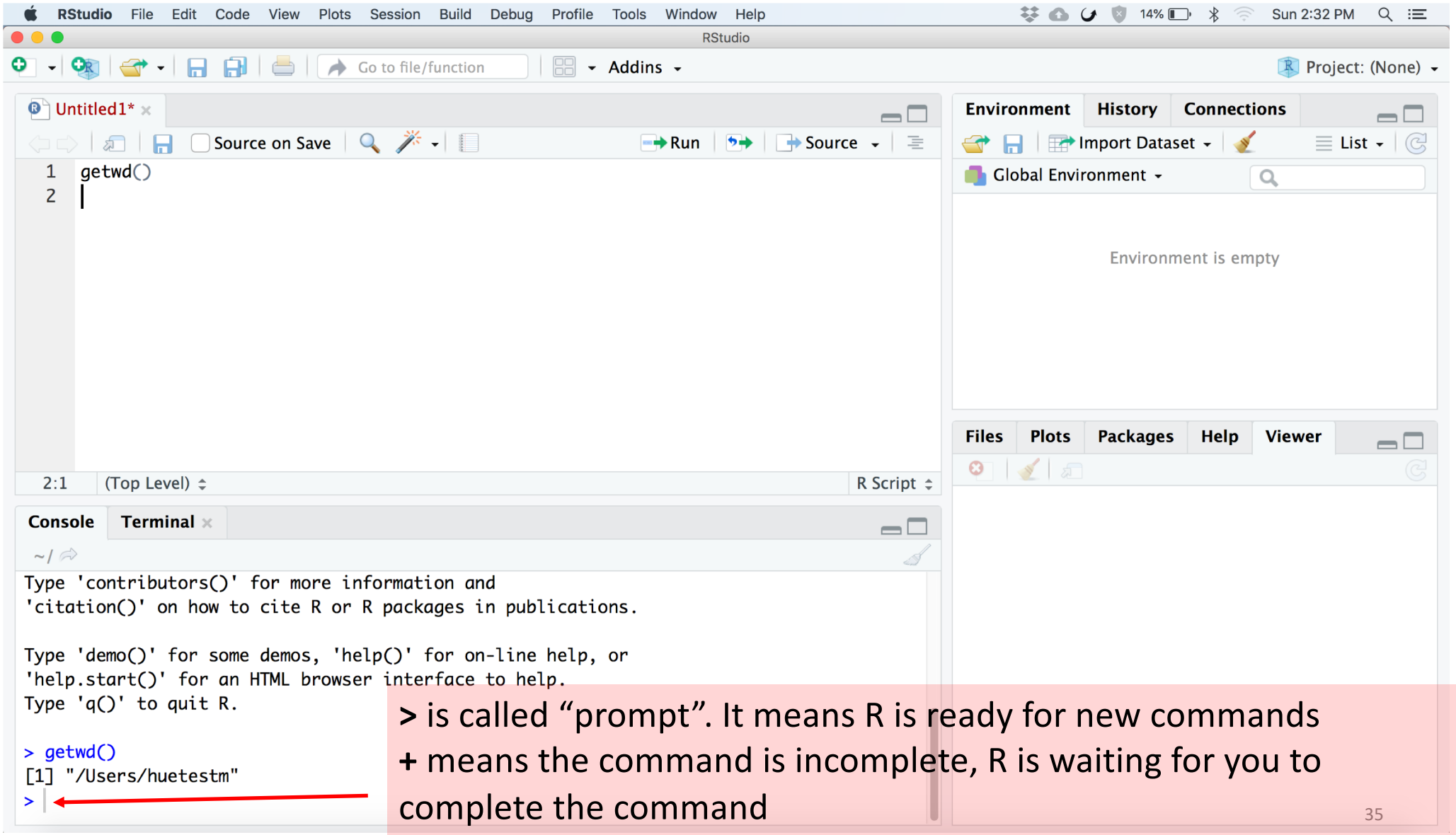


Cmd - ← Decrease font size


RStudio File Edit Code View Plots Session Build Debug Profile Tools Window Help
Sun 2:32 PM
RStudio
Go to file/function
Addins
Project: (None)
Untitled1*
Source on Save
Run
Source
1 getwd()
2
getwd()
This command tells you where you are on your computer : get working directory
Environment
History
Connections
Import Dataset
List
Global Environment
Environment is empty
Files
Plots
Packages
Help
Viewer
2:1
(Top Level)
R Script
Console
Terminal
~/
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.
32

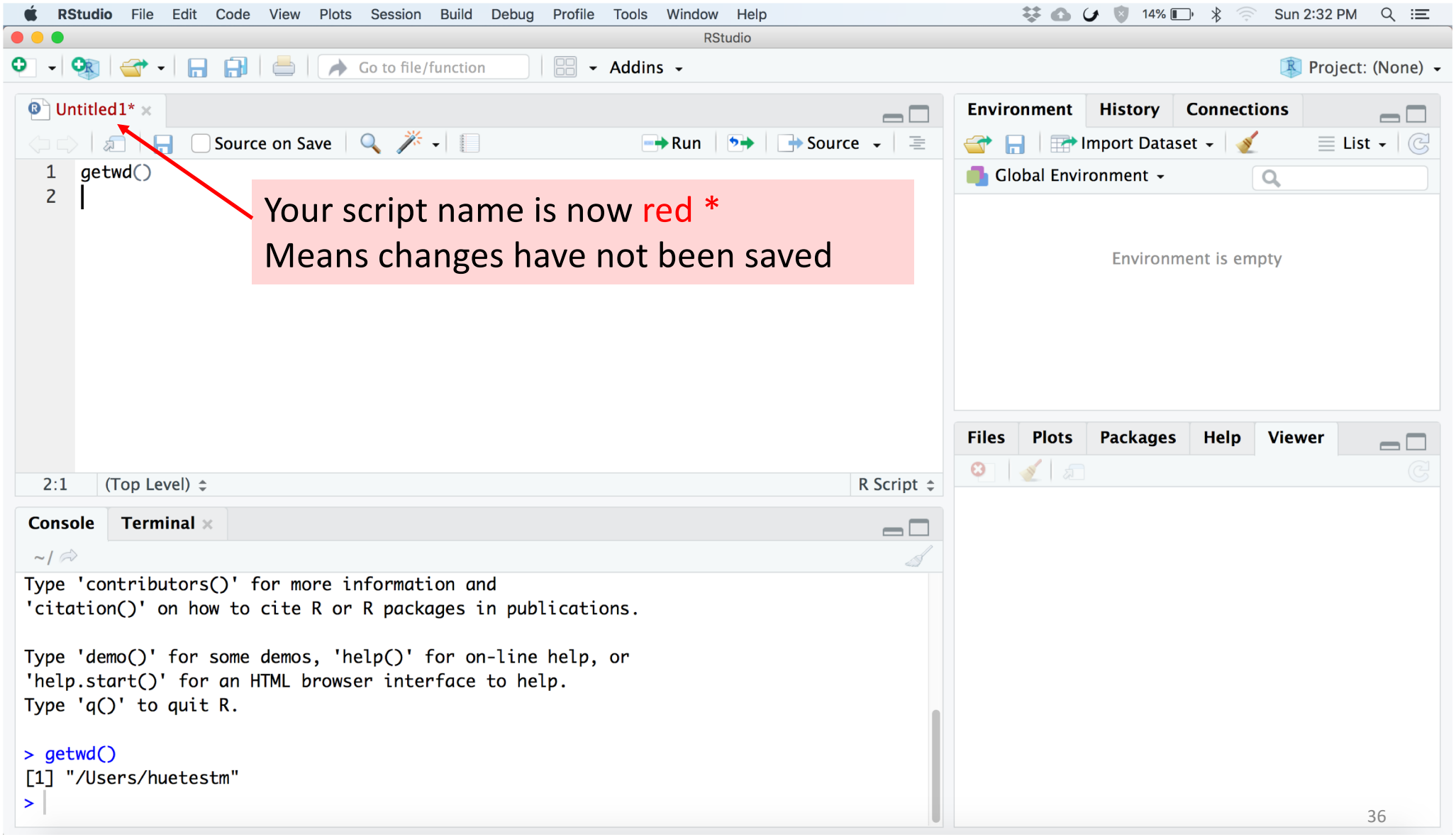

RStudio File Edit Code View Plots Session Build Debug Profile Tools Window Help
Sun 2:32 PM
RStudio
Go to file/function
Addins
Project: (None)
Untitled1*
Source on Save
Run
Source
1 getwd()
2
Send the command to the terminal :
Run button
Alt + Enter (stays on same line)
Cmd/Ctrl + Enter (moves to next line)
Environment
History
Connections
Import Dataset
List
Global Environment
Environment is empty
Files
Plots
Packages
Help
Viewer
2:1
(Top Level)
R Script
Console
Terminal
~/
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.
33

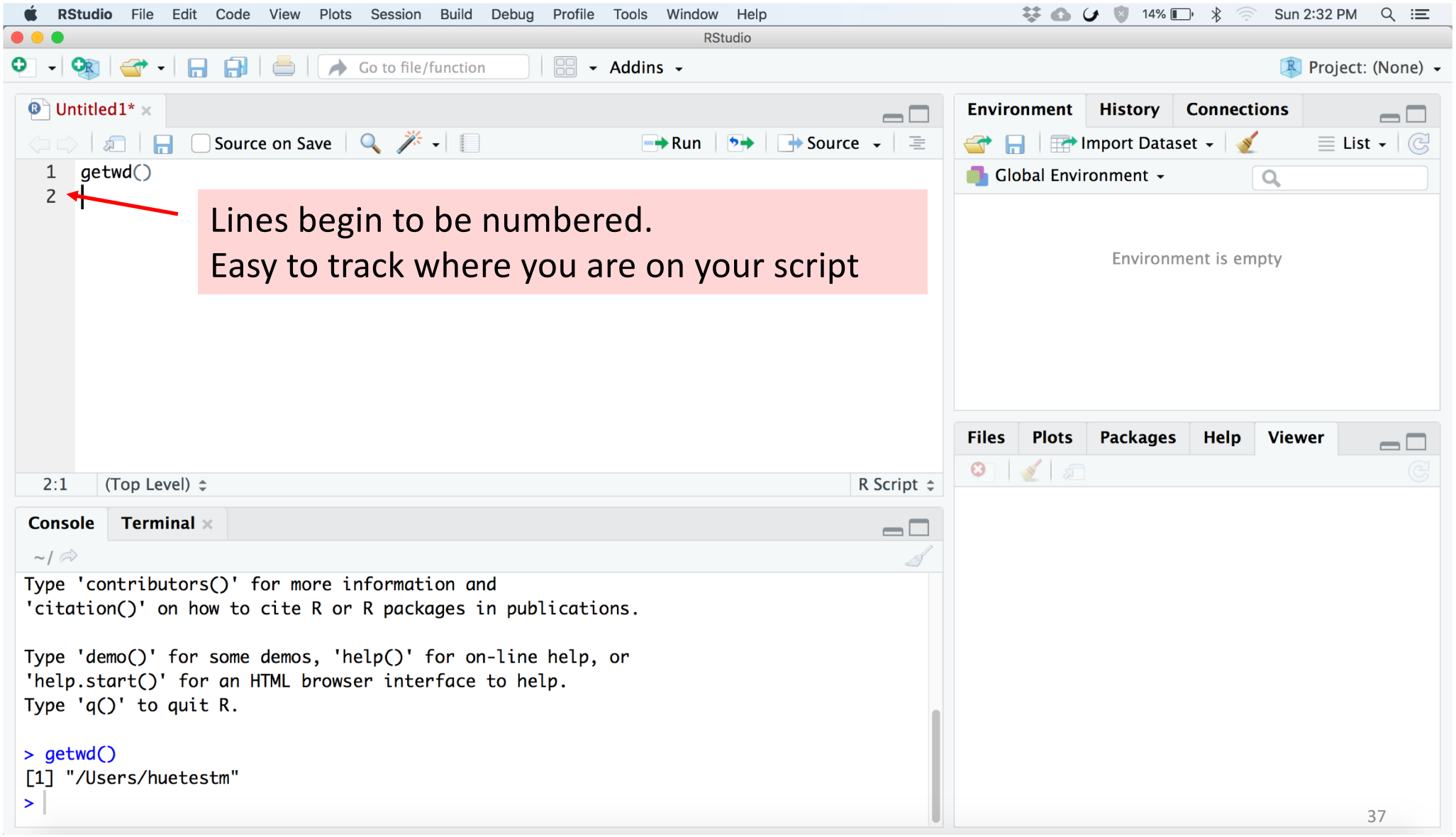

RStudio File Edit Code View Plots Session Build Debug Profile Tools Window Help
Sun 2:32 PM
RStudio
Go to file/function
Addins
Project: (None)
Untitled1*
Source on Save
Run
Source
1 getwd()
2
2:1 (Top Level)
R Script
Console
Terminal
~/
Type 'contributors()' for more information and
'citation()' on how 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.
> getwd()
[1] "/Users/huetestm"
>
The result shows in your terminal
Environment
History
Connections
Import Dataset
List
Global Environment
Environment is empty
Files
Plots
Packages
Help
Viewer

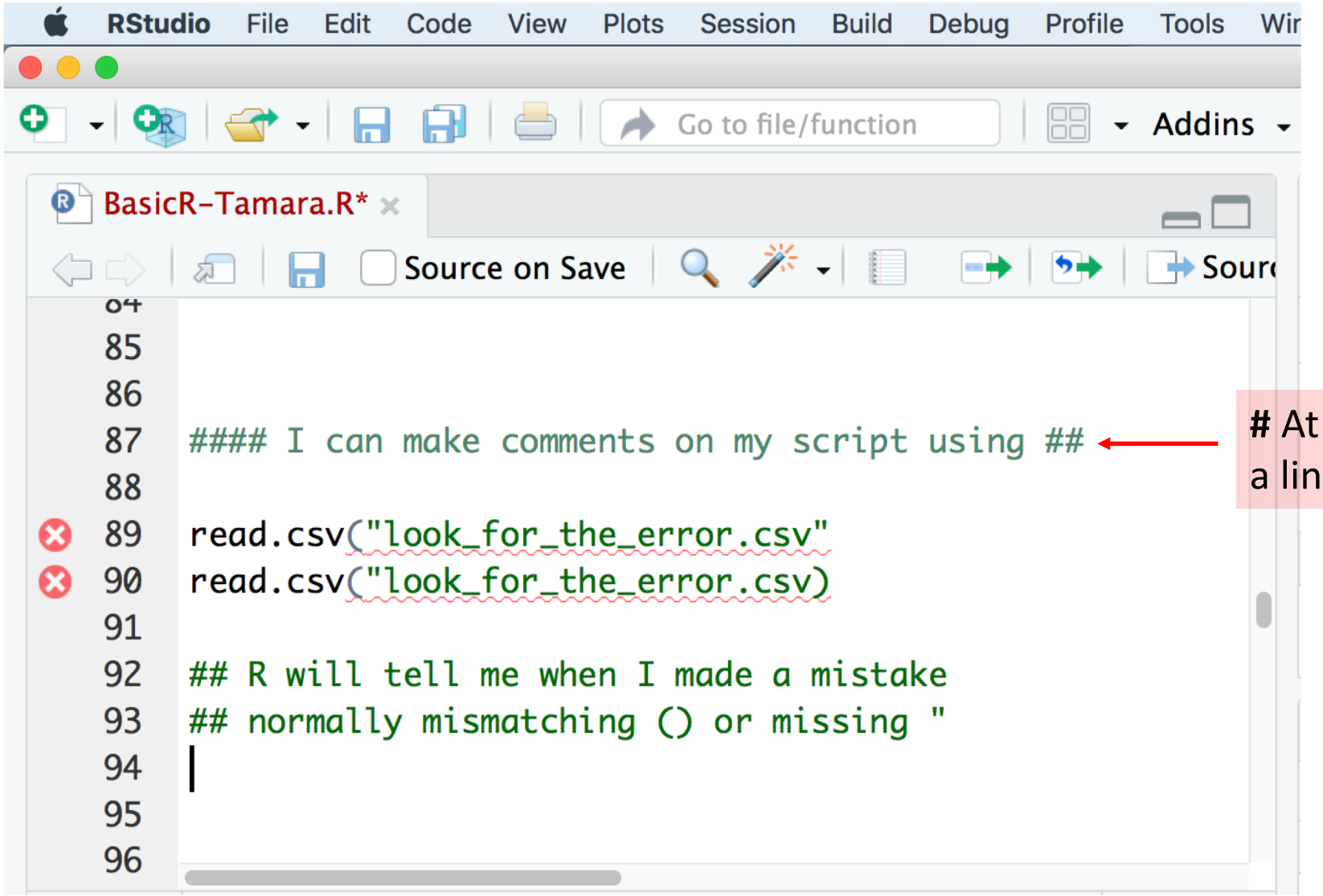


> is called “prompt”. It means R is ready for new commands

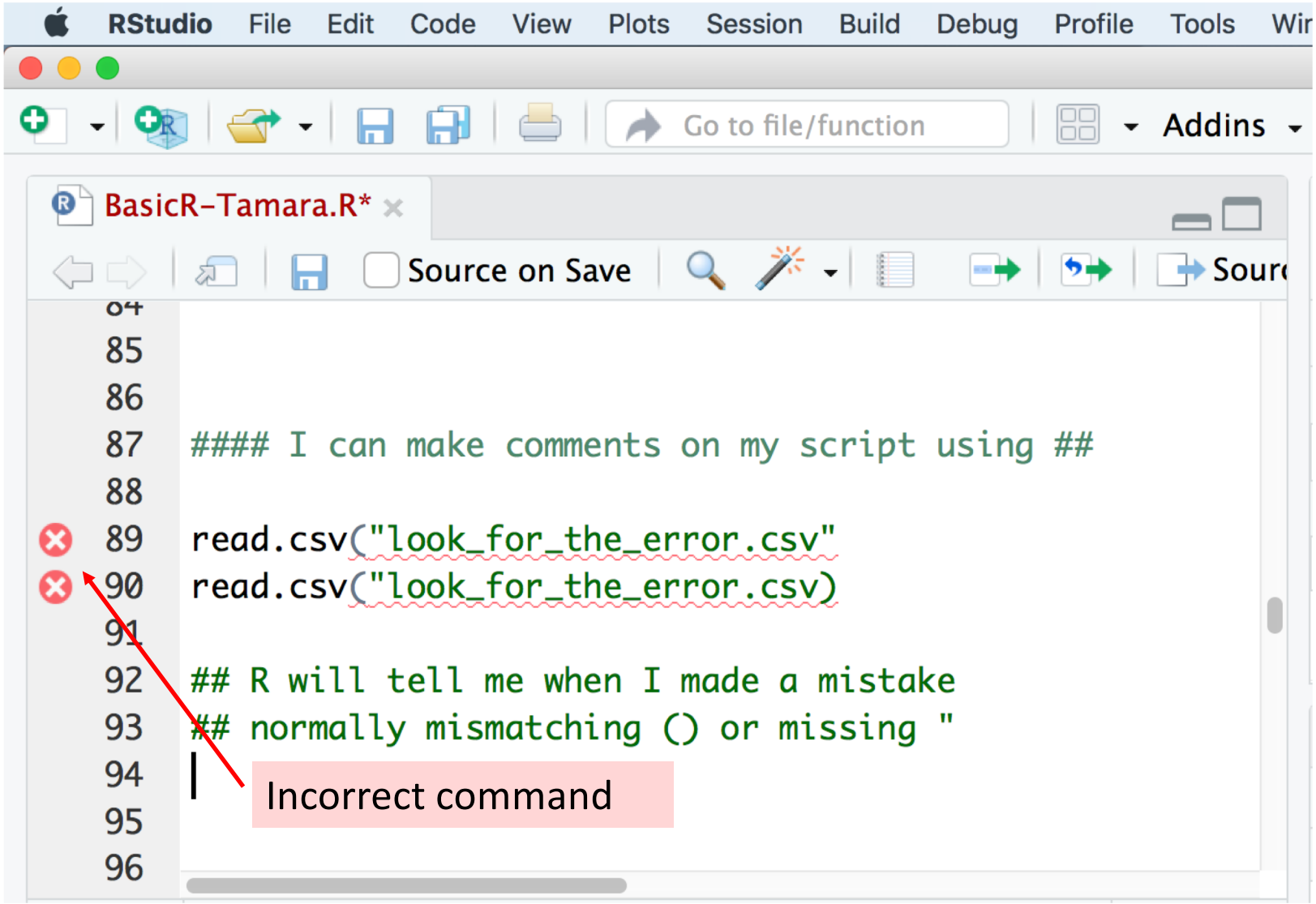
+ means the command is incomplete, R is waiting for you to complete the command


RStudio File Edit Code View Plots Session Build Debug Profile Tools Window Help
Sun 2:32 PM
RStudio
Go to file/function
Addins
Project: (None)
Untitled1*
Source on Save
Run
Source
1 getwd()
2
Your script name is now red *
Means changes have not been saved
2:1
(Top Level)
R Script
Console
Terminal
~/
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.
> getwd()
[1] "/Users/huetestm"
>
Environment
History
Connections
Import Dataset
List
Global Environment
Environment is empty
Files
Plots
Packages
Help
Viewer


RStudio File Edit Code View Plots Session Build Debug Profile Tools Window Help
Sun 2:32 PM
RStudio
Go to file/function
Addins
Project: (None)
Untitled1*
Source on Save
Run
Source
1 getwd()
2
Lines begin to be numbered.
Easy to track where you are on your script
2:1
(Top Level)
R Script
Environment
History
Connections
Import Dataset
List
Global Environment
Environment is empty
Files
Plots
Packages
Help
Viewer
Console
Terminal
~/
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.
> getwd()
[1] "/Users/huetestm"
>
37



At the beginning of a line for comments


RStudio File Edit Code View Plots Session Build Debug Profile Tools Wir
Go to file/function
Addins
BasicR-Tamara.R*
Source on Save
85
86
87 #### I can make comments on my script using ##
88
89 read.csv("look_for_the_error.csv"
90 read.csv("look_for_the_error.csv)
91
92 ## R will tell me when I made a mistake
93 ## normally mismatching () or missing "
94
95
96
Incorrect command

First steps in R

Create new script: Day1_IntroR.R
Save in folder: ~/R_Course_2019/Day1_Intro/IntroR/

1. Set working directory

- get working directory

```
getwd()
```

- set working directory

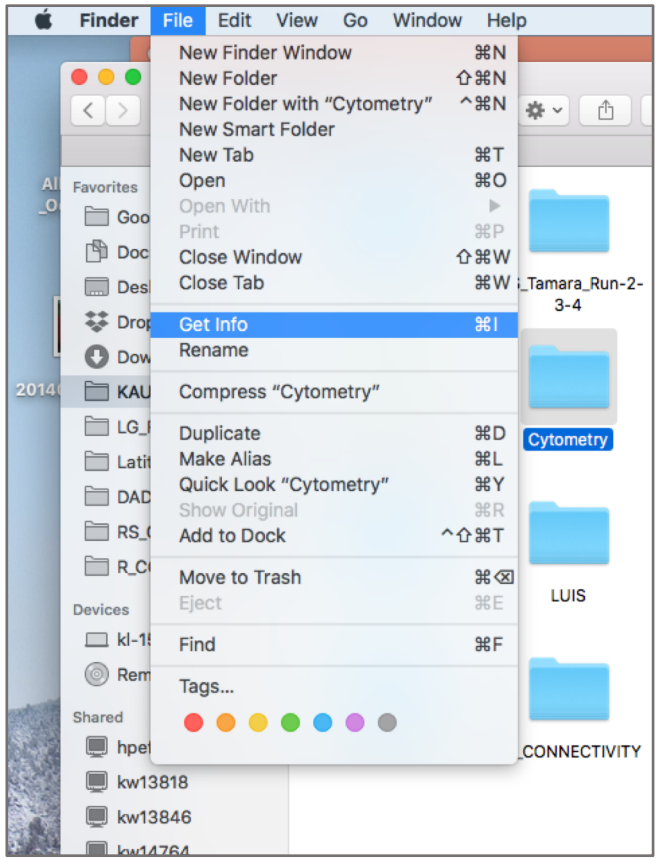
```
setwd("my/path/to/my/folder/")
```

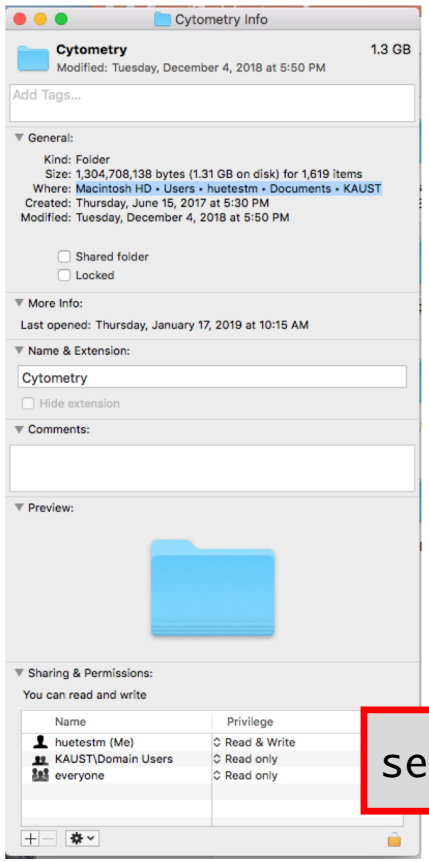
For now, all the files you are working with should be in the directory that you call working directory.

You can always retrieve files or send outputs to other folders, but for now we will keep it simple.

Finding your path

File → Get Info → Copy path → Paste in R






```
setwd("/Users/huetestm/Documents/KAUST/Cytometry")
```

Add " " at the beginning and end
Check that the path is correct and complete

Find your path to the R_Course_2019 folder
(where you saved the drive folder)

```
setwd("path/R_Course_2019/Day1_IntroR/")
```


https://pixabay.com/en/yoga-silhouette-meditation-relax-3163283/

2. R as a calculator

- We can do any mathematical operation in R

```
1/200 * 30
```

Type this in your script and send it to the terminal (Cmd + Enter)

Look at the output in your terminal.

```
> 1/200 * 30
```

You get the command echoed but with > in front
> is called prompt, it indicates that R is ready to receive commands

```
[1] 0.15
```

Below that you see your result, preceded by [1]
This means there is 1 result, if you have more, you would get [2] [3] ...

Exercise 1: Make calculations

```
(59 + 73 + 2) / 3
```

```
sin(pi / 2)
```

```
2^3
```

```
9^1/3
```

```
9^(1/3)
```

```
log10(20)
```

```
log(20)
```

Exercise 1: Make calculations

`(59 + 73 + 2) / 3`		`[1] 44.7`
`sin(pi / 2)`	$\sin(\prod/2)$	`[1] 1`
`2^3`	2^3	`[1] 8`
`9^1/3`	$9^1/3$	`[1] 3`
`9^(1/3)`	$9^{1/3} = \sqrt[3]{9}$	`[1] 2.080084`
`log10(20)`	$\log_{10}(20)$	`[1] 1.30103`
`log(20)`	Ln(20)	`[1] 2.995732`

3. Objects

Assign/create objects using **<-** or **=**
(Alt – does <- symbol in R)

```
x <- 3
x =  3
```

```
> x <- 3
> 2 * x
[1] 6
```

```
> potato <- 3
> 2 * potato
[1] 6
```

- Choosing **names** for objects
  - Combination of letters, numbers and the symbols: _ - .
  - never start with a number
  - R is case sensitive
  - should be easy but meaningful
  - avoid function names (mean, sd, plot)

  - Examples:
    hist_abundance
    bars_growth
    counts
    meanAb
    seDepth

but R is much more than a calculator...

We can do any kind of analysis!

We use **functions** that are inside **packages**:

- package `vegan`
  - function `vegdist()`
  - function `metaMDS()`
  - function `rda()`
  - function `cca()`
  - `.....`

To know what a function does and how it works we type:

```
?vegdist
help(vegdist)
```

Very Useful !!!

4. Packages

1. **Install packages:** Download packages of functions to your computer. They will be stored in a R library (created automatically when you download R)

2. **Load packages:** For each session you load the packages you will need.

   That way you save memory on R

4. Packages

1. **Install packages:** Download packages of functions to your computer. They will be stored in a R library (created automatically when you download R)

There is not one main repository of packages, and each one uses it's own commands to install:

- **R CRAN** (the main source) → `install.packages("package")`
- from **developmental sites** → `library(devtools); install_github("package")`
- **Bioconductor** → `library(BiocManager); BiocManager::install("package", version ="3.8")`
- From a **local tar.gz file** → `install.packages("path/to/package.tar.gz", repos= NULL, type ="source")`

This looks complicated, but you can very easily find instructions on internet

2. **Load packages:** For each session you load the packages you will need.

   That way you save memory on R

4. Packages

1. **Install packages:** Download packages of functions to your computer. They will be stored in a R library (created automatically when you download R)

There is not one main repository of packages, and each one uses it's own commands to install:

- **R CRAN** (the main source) → `install.packages("package")`
- from **developmental sites** → `library(devtools); install_github("package")`
- **Bioconductor** → `library(BiocManager); BiocManager::install("package", version ="3.8")`
- From a **local tar.gz file** → `install.packages("path/to/package.tar.gz", repos= NULL, type ="source")`

This looks complicated, but you can very easily find instructions on internet

2. **Load packages:** For each session you load the packages you will need.

   That way you save memory on R

```
library("package")
```

Exercise 2: Install packages

- Make sure you have internet connection
- Install packages one at a time

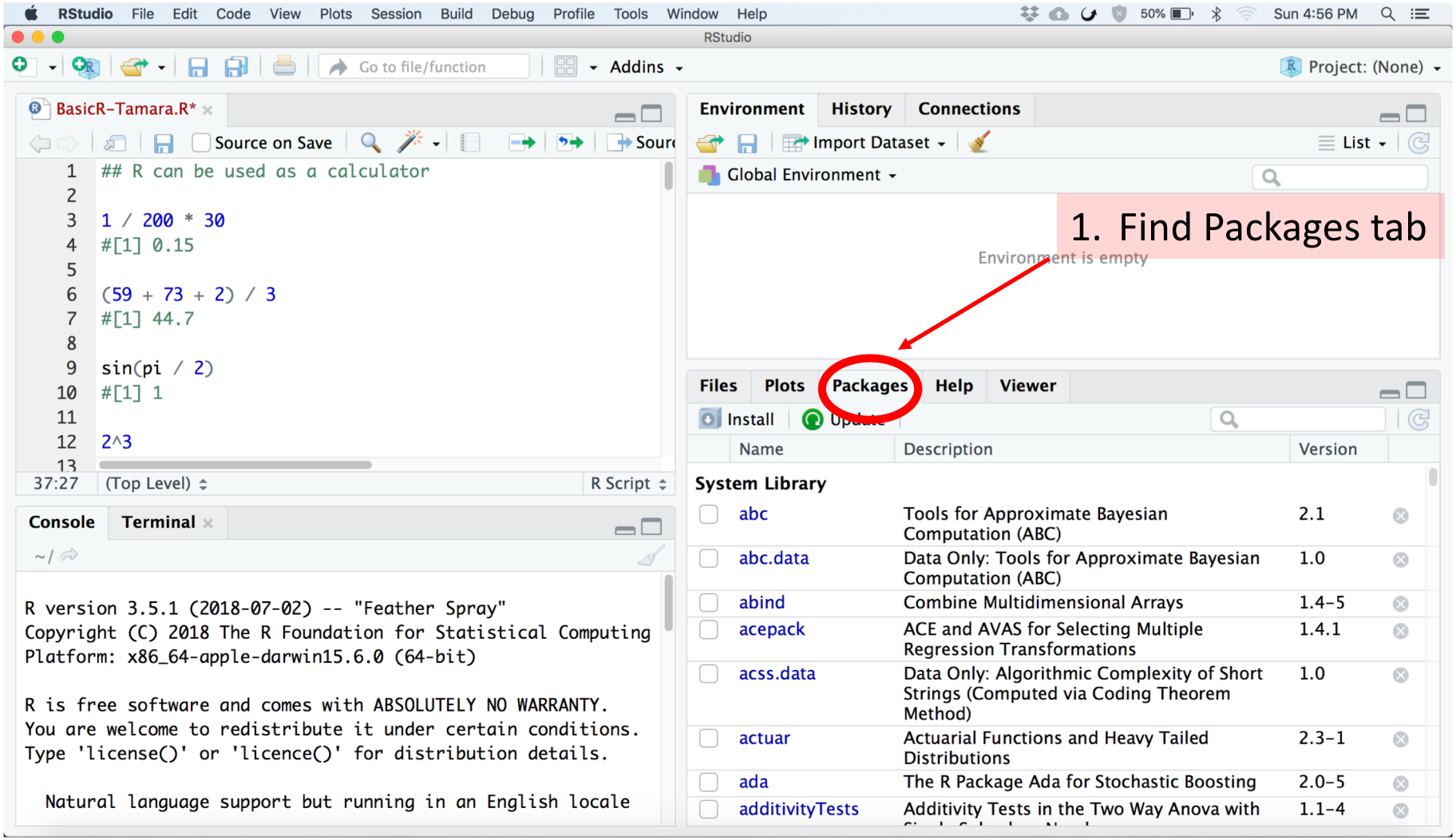
1. **Packages from CRAN**

   - The first time you will be asked to select a CRAN mirror, select a location/country close to you (should be faster)

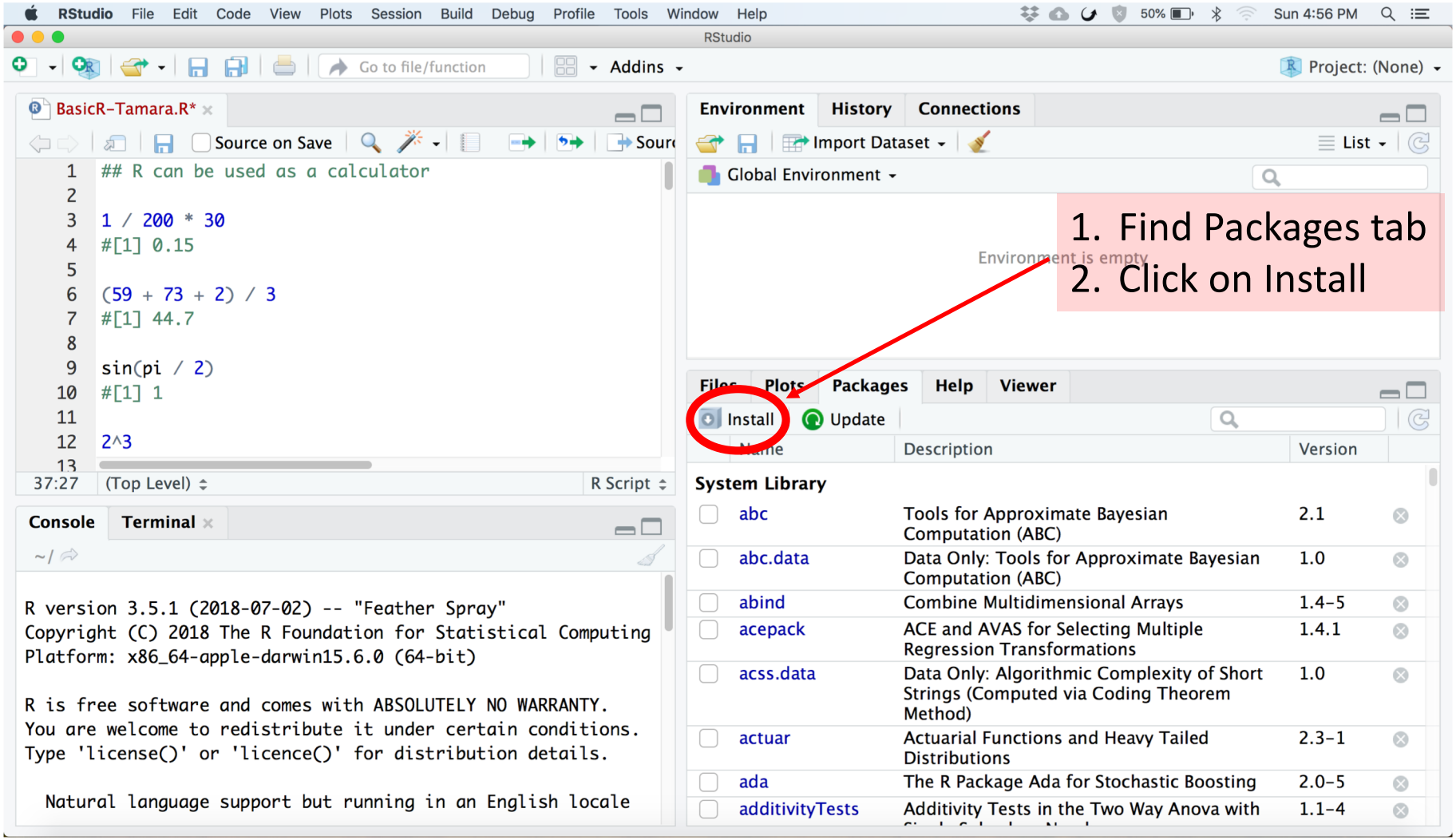
```
install.packages("ggplot2")
```

```
install.packages("vegan")
```

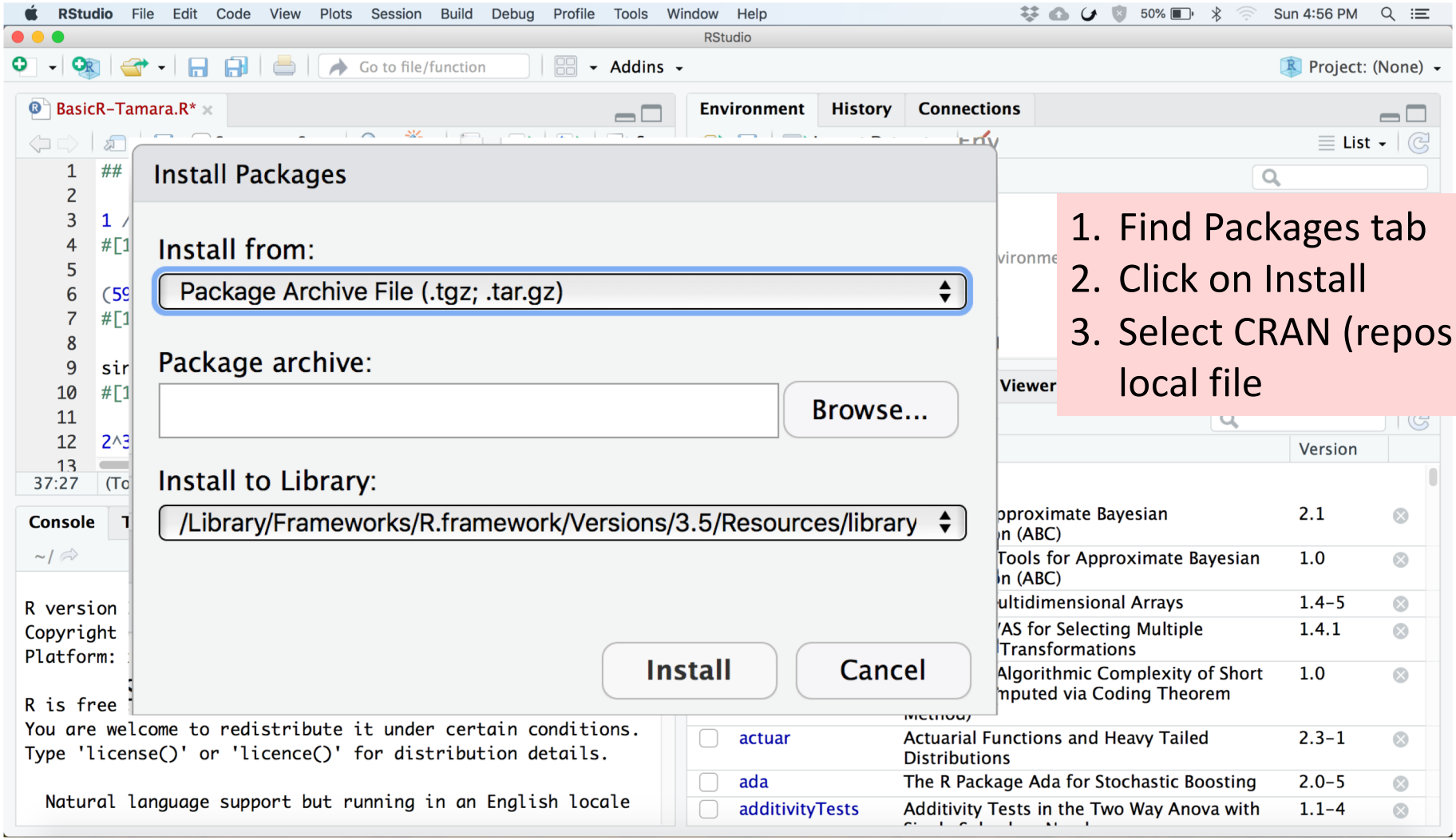
*Alternative for CRAN with R Studio



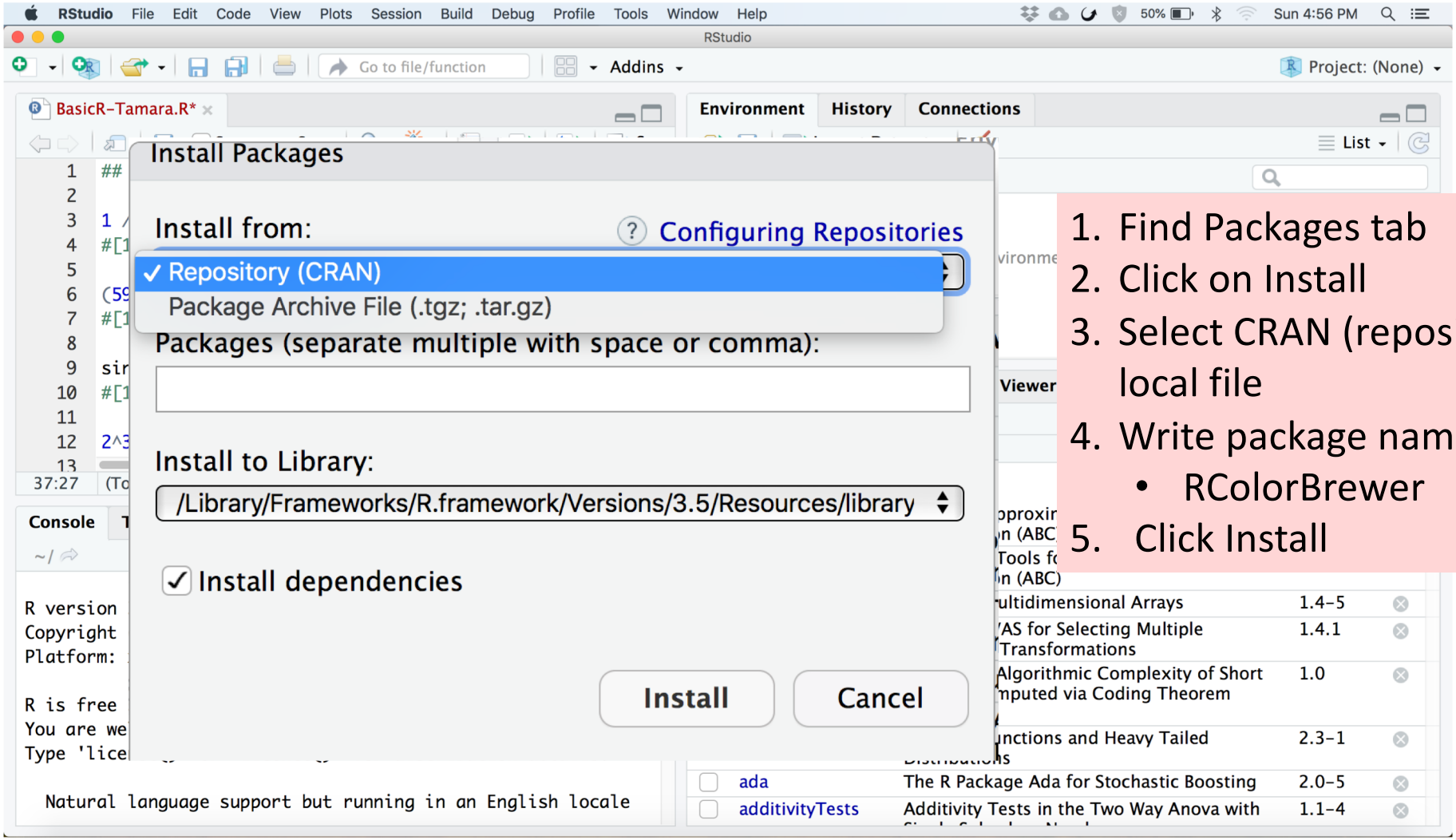
*Alternative for CRAN with R Studio



*Alternative for CRAN with R Studio



*Alternative for CRAN with R Studio



1. Find Packages tab
2. Click on Install
3. Select CRAN (repository) or local file
4. Write package name
   - RColorBrewer
5. Click Install

Exercise 2: Install packages

- Make sure you have internet connection
- Install packages one at a time

2. Packages from developmental sites (*e.g.* github)

- Extra step: Install first a helping package called devtools

```
install.packages("devtools")
```

```
library("devtools")
```

```
install_github("gavinsimpson/ggvegan")
```

Exercise 2: Install packages

- Make sure you have internet connection
- Install packages one at a time

3. Packages from Bioconductor

- Extra step: Install first a helping package called BiocManager

```
install.packages ("BiocManager")
```

```
library(BiocManager)
```

```
BiocManager::install("phyloseq", version ="3.8")
```

- You may still see this online (R versions before 3.5)

```
source("https://bioconductor.org/biocLite.R")
```

```
biocLite("phyloseq")
```

Some packages that we will use during the course:

- ade4
- BiocManager
- corrplot
- devtools
- ggplot2
- ggvegan
- ggpubr
- marmap
- phyloseq
- plot3D
- RColorBrewer
- reshape2
- tidyverse
- Hmisc
- Others…

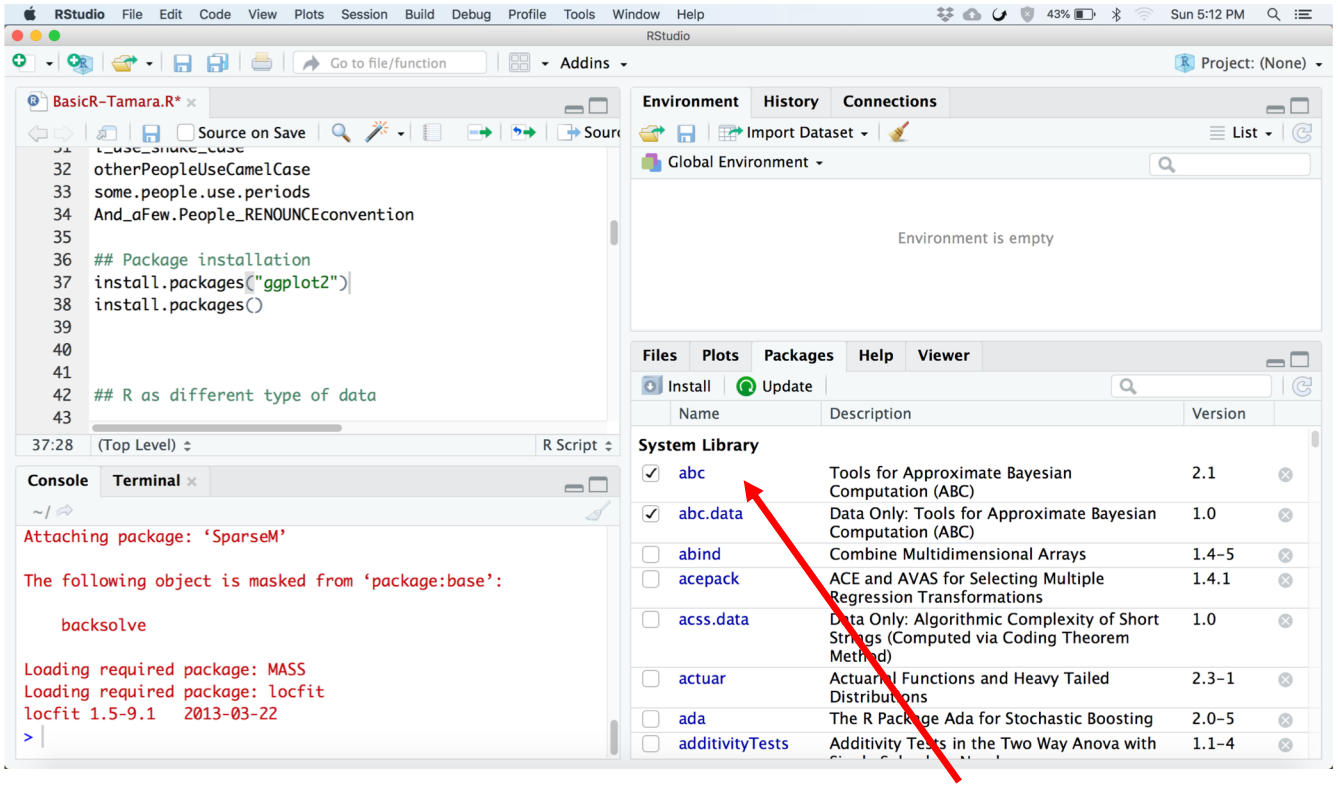
- CRAN
- from developmental site (need devtools)
- from Bioconductor (need BiocManager)

Loading packages

1. Command line

```
library("ggplot2")
```

2. RStudio easy



Just click on the package

Installing vs. loading

- `install.packages()`

- `library()`

`Installing vs. loading`

- `install.packages()`
- Install the package to your computer
- Only once
- Package will be stored forever in your R library (except if you update R)
- Sometimes you will be asked to update the packages
- If you update R, you may need to reinstall the packages

- `library()`

`Installing vs. loading`

- `install.packages()`
- Install the package to your computer
- Only once
- Package will be stored forever in your R library (except if you update R)
- Sometimes you will be asked to update the packages
- If you update R, you may need to reinstall the packages

- `library()`
- Loads the package to your current R session
- You will need to load the package every time you start R and you want to use it

5. Data formats for input

- csv
- txt
- Rdata
- Excel
- Plenty of other formats through specific packages (ctd, netcdf)….

5. Data formats for input

- csv
- txt
- Rdata
- Excel
- Plenty of other formats through specific packages (ctd, netcdf)….

Preferred plain text formats for tables

→ csv and txt are the same, only changes the separator between values

- csv uses **comma**
- txt uses **tab**

5. Data formats for input

- csv
- txt
- Rdata

Rdata is a specific R format
You can save any R object as .Rdata and when you load it again, it will look exactly like it did when you saved it (plot, table, test)

- Excel
- Plenty of other formats through specific packages (ctd, netcdf)….

5. Data formats for input

- csv
- txt
- Rdata
- Excel

Not recommended
→ Convert to csv
→ Be aware:
  - Only the active sheet will be saved
  - No formulas or plots will be saved
  - Column names without spaces or weird characters (%, $,*)

If unavoidable, there are packages to read an excel sheet

- Plenty of other formats through specific packages (ctd, netcdf)….

5. Data formats for input

- csv → `read.csv("table.csv")`
- txt → `read.table("table.txt")`
- Rdata → `load("table.Rdata")`
- Excel → `read_excel("table.xlsx", sheet=1)`

Sometimes you have weird text files:
Use **sep** to change the separator

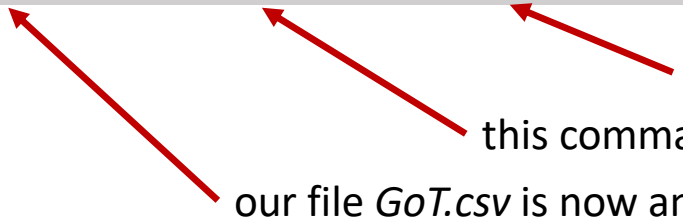
```
read.csv("table.csv",sep="\t")
read.csv("table.csv",sep=",")
read.csv("table.csv",sep=";")
read.csv("table.csv",sep=".")
```

* from tidyverse package

- Plenty of other formats through specific packages (ctd, netcdf)….

Exercise 3. Read csv file

```
got <- read.csv("GoT.csv")
```



this is the name of the file we want to open

this command opens a csv file

our file *GoT.csv* is now an object inside R called *got*

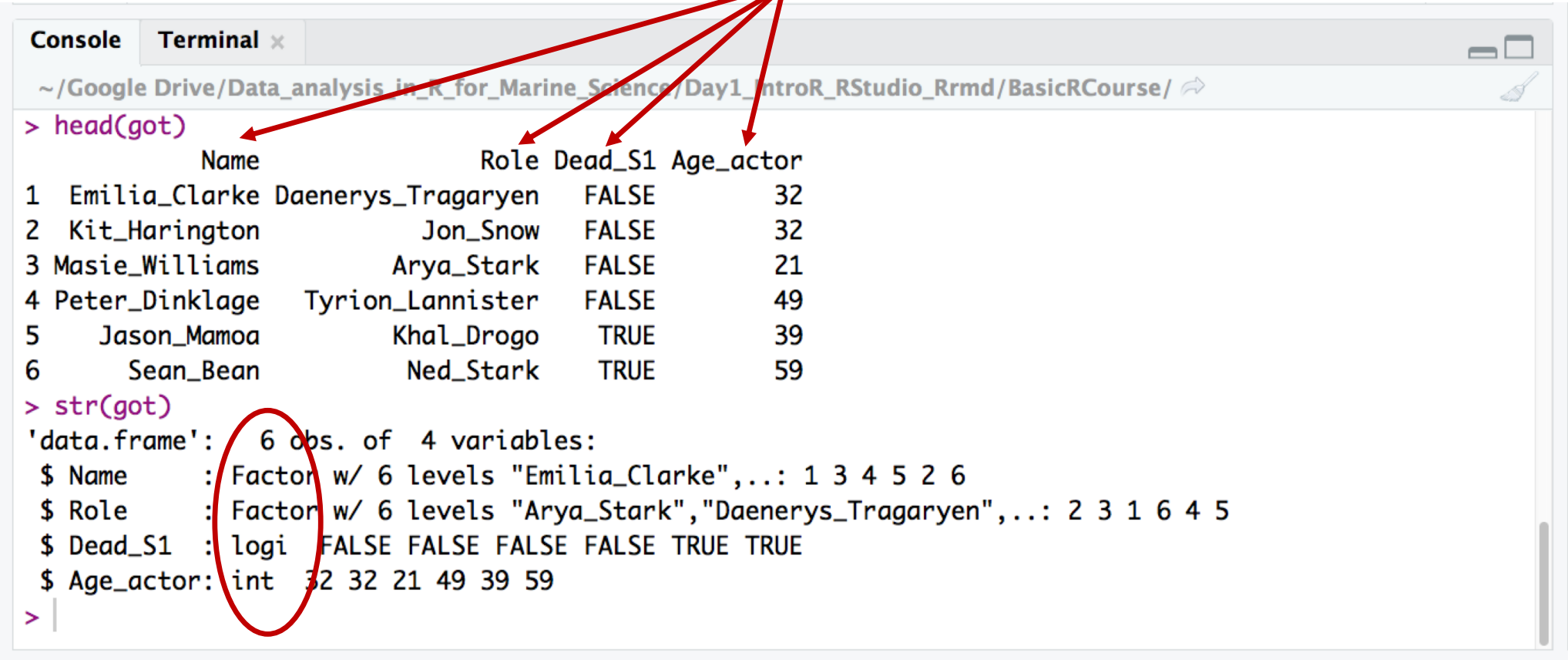
```
head(got)
```

Check the first 6 lines of the object *got*

```
str(got)
```

Check the structure of the object *got*

4 Columns



Data class

6.Data classes

- **numeric** (decimal and whole numbers)
- **character** (anything with letters)
- **logical** (TRUE, FALSE)
- **factor** (numeric or characters that have levels)

6.Data classes

- **numeric** (decimal and whole numbers)

```
a <- c(1.2, -4.5, 3.2, 5, 0.3)
```

- **character** (anything with letters)

- **logical** (TRUE, FALSE)

- **factor** (numeric or characters that have levels)

c() means **concatenate**, it puts all the elements separated by comma in the same unit

6.Data classes

- **numeric** (decimal and whole numbers)

```
a <- c(1.2, -4.5, 3.2, 5, 0.3)
```

- **character** (anything with letters)

```
b <- c("a","b","c","gold","katy_perry")
```

- **logical** (TRUE, FALSE)

- **factor** (numeric or characters that have levels)

c() means **concatenate**, it puts all the elements separated by comma in the same unit

Character/name vectors always go with **" "**

6.Data classes

- **numeric** (decimal and whole numbers)

```
a <- c(1.2, -4.5, 3.2, 5, 0.3)
```

- **character** (anything with letters)

```
b <- c("a","b","c","gold","katy_perry")
```

- **logical** (TRUE, FALSE)

```
d <- c(TRUE,TRUE,FALSE,TRUE,FALSE)
```

- **factor** (numeric or characters that have levels)

c() means **concatenate**, it puts all the elements separated by comma in the same unit

Character/name vectors always go with **""**

6.Data classes

- **numeric** (decimal and whole numbers)

```
a <- c(1.2, -4.5, 3.2, 5, 0.3)
```

- **character** (anything with letters)

```
b <- c("a","b","c","gold","katy_perry")
```

- **logical** (TRUE, FALSE)

```
d <- c(TRUE,TRUE,FALSE,TRUE,FALSE)
```

- **factor** (numeric or characters that have levels)

```
e <- factor(c("summer","winter","summer","fall"))
```

```
levels(e)
```

```
> fall summer winter
```

c() means **concatenate**, it puts all the elements separated by comma in the same unit

Character/name vectors always go with **""**

6.Data classes

- **numeric** (decimal and whole numbers)

```
a <- c(1.2, -4.5, 3.2, 5, 0.3)
```

- **character** (anything with letters)

```
b <- c("a","b","c","gold","katy_perry")
```

- **logical** (TRUE, FALSE)

```
d <- c(TRUE,TRUE,FALSE,TRUE,FALSE)
```

- **factor** (numeric or characters that have levels)

```
e <- factor(c("summer","winter","summer","fall"))
levels(e)
> fall summer winter
```

c() means **concatenate**, it puts all the elements separated by comma in the same unit

Character/name vectors always go with **""**

class() will tell you the class of your data

Change class formats

- to character

```
as.character()
```

- to numeric

```
as.numeric()
```

- to factor

```
as.factor()
```

Some conversions are tricky:

- You can transform a number to character, but you can't convert a character to number

Exercise 4. Data classes

```
a <- c(1.2, -4.5, 3.2, 5, 0.3)
```

```
a
```

```
class(a)
```

```
a.ch <- as.character(a)
```

```
class(a.ch)
```

```
a.ch
```

Exercise 4. Data classes

```
a <- c(1.2, -4.5, 3.2, 5, 0.3)
```

```
a
```

```
[1]  1.2 -4.5  3.2  5.0  0.3
```

```
class(a)
```

```
[1] "numeric"
```

```
a.ch <- as.character(a)
```

```
class(a.ch)
```

```
[1] "character"
```

```
a.ch
```

```
[1] "1.2"  "-4.5" "3.2"  "5"    "0.3"
```

Exercise 4. Data classes

```
a <- c(1.2, -4.5, 3.2, 5, 0.3)
```

```
a
```

```
[1]  1.2 -4.5  3.2  5.0  0.3
```

```
class(a)
```

```
[1] "numeric"
```

```
a.ch <- as.character(a)
```

```
class(a.ch)
```

```
[1] "character"
```

```
a.ch
```

```
[1] "1.2"  "-4.5" "3.2"  "5"    "0.3"
```

```
a <- as.character(a)
```

You can overwrite an object

You will loose the information stored in the previous one

Exercise 4. Data classes

```
a <- c(1.2, -4.5, 3.2, 5, 0.3)
```

```
a
```

```
[1]  1.2 -4.5  3.2  5.0  0.3
```

```
class(a)
```

```
[1] "numeric"
```

```
a.ch <- as.character(a)
```

```
class(a.ch)
```

```
[1] "character"
```

```
a.ch
```

```
[1] "1.2"  "-4.5" "3.2"  "5"    "0.3"
```

What happens if you try to convert a character string to numbers?

```
b <- c("a","b","c","gold","katy_perry")
```

```
as.numeric(b)
```

Exercise 4. Data classes

```
a <- c(1.2, -4.5, 3.2, 5, 0.3)
```

```
a
```

```
[1]  1.2 -4.5  3.2  5.0  0.3
```

```
class(a)
```

```
[1] "numeric"
```

```
a.ch <- as.character(a)
```

```
class(a.ch)
```

```
[1] "character"
```

```
a.ch
```

```
[1] "1.2"  "-4.5" "3.2"  "5"    "0.3"
```

What happens if you try to convert a character string to numbers?

```
b <- c("a","b","c","gold","katy_perry")
```

```
as.numeric(b)
```

```
[1] NA NA NA NA NA
```

7. Types of data objects

- So far we have been working with 2 types of objects:

7. Types of data objects

- So far we have been working with 2 types of objects:

Vector: 1 dimension data objects (also called **strings**)

```
a <- c(1.2, -4.5, 3.2, 5, 0.3)
```

```
a
```

```
[1]  1.2 -4.5  3.2  5.0  0.3
```

7. Types of data objects

- So far we have been working with 2 types of objects:

Vector: 1 dimension data objects (also called **strings**)

```
a <- c(1.2, -4.5, 3.2, 5, 0.3)
```

```
a
```

```
[1]  1.2 -4.5  3.2  5.0  0.3
```

Data frame: 2 dimension data objects (rows and columns)

```
got
```

```
             Name                Role  Dead_S1 Age_actor
1  Emilia_Clarke Daenerys_Tragaryen   FALSE         32
2  Kit_Harington           Jon_Snow   FALSE         32
3 Masie_Williams         Arya_Stark   FALSE         21
4 Peter_Dinklage   Tyrion_Lannister   FALSE         49
5    Jason_Mamoa         Khal_Drogo    TRUE         39
6      Sean_Bean          Ned_Stark    TRUE         59
```

7. Types of data objects

- So far we have been working with 2 types of objects:

Vector: 1 dimension data objects (also called **strings**)

```
a <- c(1.2, -4.5, 3.2, 5, 0.3)
```

```
a
```

```
[1]  1.2 -4.5  3.2  5.0  0.3
```

This is a numeric vector
All the elements are the same

Data frame: 2 dimension data objects (rows and columns)

```
got
```

	Name	Role	Dead_S1	
1	Emilia_Clarke	Daenerys_Tragaryen	FALSE	32
2	Kit_Harington	Jon_Snow	FALSE	32
3	Masie_Williams	Arya_Stark	FALSE	21
4	Peter_Dinklage	Tyrion_Lannister	FALSE	49
5	Jason_Mamoa	Khal_Drogo	TRUE	39
6	Sean_Bean	Ned_Stark	TRUE	59

In a dataframe each column can have a different format

7. Types of data objects

- So far we have been working with 2 types of objects:

Vector: 1 dimension data objects (also called **strings**)

```
a <- c(1.2, -4.5, 3.2, 5, 0.3)
```

```
a
```

```
[1]  1.2 -4.5  3.2  5.0  0.3
```

This is a numeric vector
All the elements are the same

Data frame: 2 dimension data objects (rows and columns)

```
got
```

	Name	Role	Dead_S1	
1	Emilia_Clarke	Daenerys_Tragaryen	FALSE	32
2	Kit[illegible]n	[illegible]	[illegible]	[illegible]
3	Masie_Williams	Arya_Stark	[illegible]	[illegible]
4	Peter_Dinklage	Tyrion_Lannister	FALSE	49
5	Jason_Mamoa	Khal_Drogo	TRUE	39
6	Sean_Bean	Ned_Stark	TRUE	59

Character | Character | Logical | Numeric

In a dataframe each column can have a different format

Data Objects in R

Vector	1 dimension	All elements have the same data types
Matrix	2 dimensions	Data types: numeric, character logic, factor
Array	2 or more dimensions	
Data frame	2 dimensions	table-like data object allowing different data types for different columns
List	Collection of data objects, each element of a list is a data object	

Data Objects in R

Vector	1 dimension	All elements have the same data types Data types: numeric, character logic, factor
Matrix	2 dimensions	
Array	2 or more dimensions	
Data frame	2 dimensions	table-like data object allowing different data types for different columns
List	Collection of data objects, each element of a list is a data object	

got

	Name	Role	Dead_S1	Age_actor
1	Emilia_Clarke	Daenerys_Tragaryen	FALSE	32
2	Kit_Harington	Jon_Snow	FALSE	32
3	Masie_Williams	Arya_Stark	FALSE	21
4	Peter_Dinklage	Tyrion_Lannister	FALSE	49
5	Jason_Mamoa	Khal_Drogo	TRUE	39
6	Sean_Bean	Ned_Stark	TRUE	59

got =

	C1	C2	C3	C4
	Name	Role	Dead_S1	Age_actor
R1	Emilia_Clarke	Daenerys_Targaryen	FALSE	32
R2	Kit_Williams	Jon_Snow	FALSE	32
R3	Masie_Williams	Arya_Stark	FALSE	21
R4	Peter_Dinklage	Tyrion_Lannister	FALSE	49
R5	Jason_Mamoa	Khal_Drogo	TRUE	39
R6	Sean_Bean	Ned_Stark	TRUE	59

This is a **dataframe:** data in rows and columns, with headers

got

	Name	Role	Dead_S1	Age_actor
1	Emilia_Clarke	Daenerys_Tragaryen	FALSE	32
2	Kit_Harington	Jon_Snow	FALSE	32
3	Masie_Williams	Arya_Stark	FALSE	21
4	Peter_Dinklage	Tyrion_Lannister	FALSE	49
5	Jason_Mamoa	Khal_Drogo	TRUE	39
6	Sean_Bean	Ned_Stark	TRUE	59

got =

	C1	C2	C3	C4
	Name	Role	Dead_S1	Age_actor
R1	Emilia_Clarke	Daenerys_Targaryen	FALSE	32
R2	Kit_Williams	Jon_Snow	FALSE	32
R3	Masie_Williams	Arya_Stark	FALSE	21
R4	Peter_Dinklage	Tyrion_Lannister	FALSE	49
R5	Jason_Mamoa	Khal_Drogo	TRUE	39
R6	Sean_Bean	Ned_Stark	TRUE	59

Each column is a separate **vector**

Exercise 5. Create objects

- Vector

```
f <- c(1,2,3,4)
g <- c("GA","GA","GB","GB")
h <- c(TRUE,FALSE,TRUE, TRUE)
```

1. Create 3 different of vectors

Exercise 5. Create objects

- Vector

```
f <- c(1,2,3,4)
g <- c("GA","GA","GB","GB")
h <- c(TRUE,FALSE,TRUE, TRUE)
```

1. Create 3 different of vectors

- Dataframe

```
mydata <- data.frame(f,g,h)
names(mydata) <- c("ID","Team","Passed")
View(mydata)
head(mydata)
```

2. Create a dataframe from the 3 vectors
   - they should have the same number of elements
   - name the columns

Exercise 5. Create objects

- Vector

```
f <- c(1,2,3,4)
g <- c("GA","GA","GB","GB")
h <- c(TRUE,FALSE,TRUE, TRUE)
```

1. Create 3 different of vectors

- Dataframe

```
mydata <- data.frame(f,g,h)
names(mydata) <- c("ID","Team","Passed")
View(mydata)
head(mydata)
```

2. Create a dataframe from the 3 vectors
   - they should have the same number of elements
   - name the columns

Alternative way:

```
mydata <- data.frame(ID = c(1,2,3,4),
                     Team = c("GA","GA","GB","GB"),
                     Passed = c(TRUE,FALSE,TRUE, TRUE))
```

Exercise 5. Create objects

- Vector

```
f <- c(1,2,3,4)
g <- c("GA","GA","GB","GB")
h <- c(TRUE,FALSE,TRUE, TRUE)
```

1. Create 3 different of vectors

- Dataframe

```
mydata <- data.frame(f,g,h)
names(mydata) <- c("ID","Team","Passed")
View(mydata)
head(mydata)
```

2. Create a dataframe from the 3 vectors
   - they should have the same number of elements
   - name the columns

- Matrix

```
matrix_example <- matrix(1:18, ncol=6, nrow=3)
matrix_example
```

3. Create a matrix

```
mydata <- as.matrix(mydata)
```

Convert a data.frame to matrix

Exercise 6:Create your dataframe

- Create a data frame called **experiment** that looks like this:

Subject	Weight_kg	Height_cm	Age
S1	57	160	25
S2	45	135	15
S3	68	167	33

Exercise 6:Create your dataframe

- Create a data frame called **experiment** that looks like this:

Subject	Weight_kg	Height_cm	Age
S1	57	160	25
S2	45	135	15
S3	68	167	33

```
experiment <- data.frame(Subject    = c("S1","S2","S3"),
                         Weight_kg  = c(57,45,68),
                         Height_cm  = c(160,135,167),
                         Age        = c(25,15,33))
```

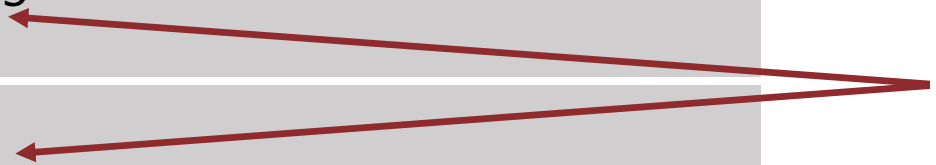
8. Functions and arguments

- Functions make computing fast

```
x <- c(2,3,5)
```

```
> (2+3+5)/3
[1] 3.333
```

```
> mean(x)
[1] 3.333
```



It's the same result but it saves time and avoids errors

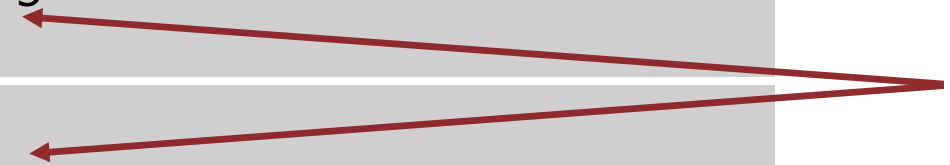
8. Functions and arguments

- Functions make computing fast

```
x <- c(2,3,5)
```

```
> (2+3+5)/3
[1] 3.333
```

```
> mean(x)
[1] 3.333
```



It's the same result but it saves time and avoids errors

- Each function has **arguments**: the parameters needed to compute a value.

8. Functions and arguments

- Functions make computing fast

```
x <- c(2,3,5)
```

```
> (2+3+5)/3
[1] 3.333
```

```
> mean(x)
[1] 3.333
```

It's the same result but it saves time and avoids errors

- Each function has **arguments**: the parameters needed to compute a value.

```
?mean
```

```
mean(x, na.rm, trim)
```

Check help pages with **?**

- What does the function do?
- What are the arguments?
- Examples

8. Functions and arguments

- Functions make computing fast

```
x <- c(2,3,5)
```

```
> (2+3+5)/3
[1] 3.333
```

```
> mean(x)
[1] 3.333
```

It's the same result but it saves time and avoids errors

- Each function has **arguments**: the parameters needed to compute a value.

```
?mean
```

```
mean(x, na.rm, trim)
```

x: vector of numeric values
na.rm: logical, remove NAs
trim: round values before computing mean

Check help pages with **?**

- What does the function do?
- What are the arguments?
- Examples

Some arguments have default values

Some arguments are optional

Exercise 7: built in functions

```
?seq()
seq(from = 1, to = 10)
seq(1, 10)
seq(1, 10, by = 2)
seq(1, 10, length.out = 3)
```

Exercise 7: built in functions

```
?seq()
seq(from = 1, to = 10)
seq(1, 10)
seq(1, 10, by = 2)
seq(1, 10, length.out = 3)
```

arguments are taken by name or by order

Exercise 7: built in functions

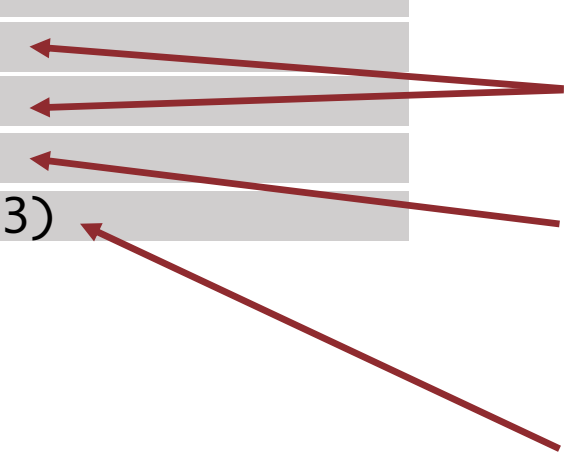
```
?seq()
seq(from = 1, to = 10)
seq(1, 10)
seq(1, 10, by = 2)
seq(1, 10, length.out = 3)
```

arguments are taken
by name or by order

default by=1
if you do not write anything
it will do a step of 1

Exercise 7: built in functions

```
?seq()
seq(from = 1, to = 10)
seq(1, 10)
seq(1, 10, by = 2)
seq(1, 10, length.out = 3)
```



arguments are taken
by name or by order

default by=1
if you do not write anything
it will do a step of 1

Adjusts to how many
output numbers you want

Exercise 7: built in functions

```
?seq()
seq(from = 1, to = 10)
seq(1, 10)
seq(1, 10, by = 2)
seq(1, 10, length.out = 3)
```

```
?rep()
rep(20, times=10)
rep(c(20,10), times=10)
rep(c(20,10), each=10)
c(rep("male",times=20), rep("female",times=20))
```

Exercise 8: Use sample()

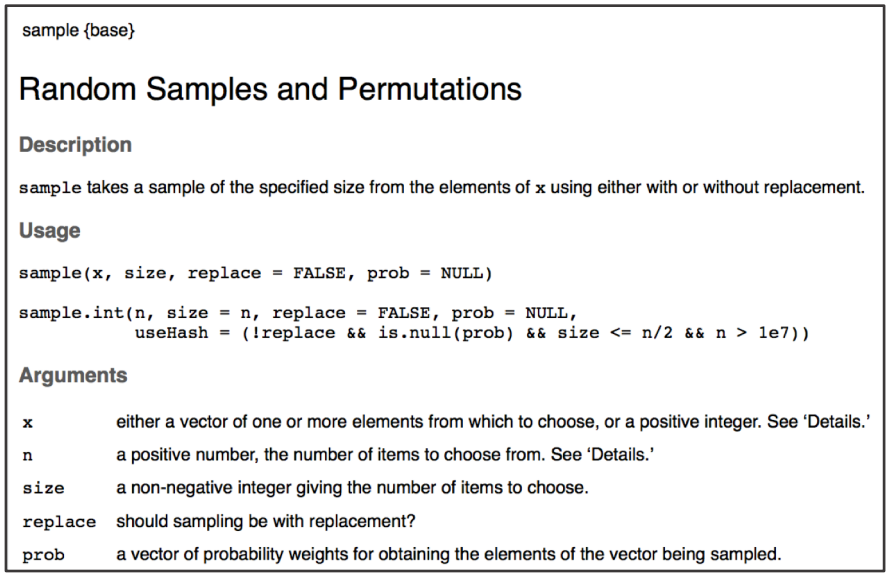
- sample()
- sample(x=, size=, replace=)
  - get a random number between 1 and 50, without replacement
  - get 6 random numbers between 3 and 40, with replacement

?sample

description of function

how to use it

description of arguments



sample {base}

Random Samples and Permutations

Description

`sample` takes a sample of the specified size from the elements of `x` using either with or without replacement.

Usage

```
sample(x, size, replace = FALSE, prob = NULL)

sample.int(n, size = n, replace = FALSE, prob = NULL,
           useHash = (!replace && is.null(prob) && size <= n/2 && n > 1e7))
```

Arguments

`x`	either a vector of one or more elements from which to choose, or a positive integer. See 'Details.'
`n`	a positive number, the number of items to choose from. See 'Details.'
`size`	a non-negative integer giving the number of items to choose.
`replace`	should sampling be with replacement?
`prob`	a vector of probability weights for obtaining the elements of the vector being sampled.

Exercise 8: Use sample()

- sample()
  - get a random number between 1 and 50, without replacement

```
sample(x=1:50,size=1,replace=FALSE)
```

  - get 6 random numbers between 3 and 40, with replacement

```
sample(x=3:40,size=6,replace=TRUE)
```

Create your own functions

```
function(arg1, arg2) {
            do something with arg1 and arg2
            return(result)}
```

Note the use of **{}**

Create your own functions

```
function(arg1, arg2) {
            do something with arg1 and arg2
            return(result)}
```

Note the use of **{}**

```
fahrenheit_to_celsius <- function(temp_F) {
      temp_C <- ((temp_F - 32) * (5 / 9))
      return(temp_C)
}
```

Create your own functions

```
function(arg1, arg2) {
            do something with arg1 and arg2
            return(result)}
```

Note the use of **{}**

name of function


```
fahrenheit_to_celsius <- function(temp_F) {
      temp_C <- ((temp_F - 32) * (5 / 9))
      return(temp_C)
}
```

Create your own functions

```
function(arg1, arg2) {
            do something with arg1 and arg2
            return(result)}
```

Note the use of **{}**

name of function

argument: the temperature in Fahrenheit

```
fahrenheit_to_celsius <- function(temp_F) {
      temp_C <- ((temp_F - 32) * (5 / 9))
      return(temp_C)
}
```

Create your own functions

```
function(arg1, arg2) {
            do something with arg1 and arg2
            return(result)}
```

Note the use of **{}**

name of function

argument: the temperature in Fahrenheit

```
fahrenheit_to_celsius <- function(temp_F) {
      temp_C <- ((temp_F - 32) * (5 / 9))
      return(temp_C)
}
```

calculate the temperature in Celsius

Create your own functions

```
function(arg1, arg2) {
            do something with arg1 and arg2
            return(result)}
```

Note the use of **{}**

name of function

argument: the temperature in Fahrenheit

```
fahrenheit_to_celsius <- function(temp_F) {
      temp_C <- ((temp_F - 32) * (5 / 9))
      return(temp_C)
}
```

calculate the temperature in Celsius

return the temperature in Celsius

Create your own functions

```
function(arg1, arg2) {
            do something with arg1 and arg2
            return(result)}
```

Note the use of **{}**

name of function

argument: the temperature in Fahrenheit

```
fahrenheit_to_celsius <- function(temp_F) {
      temp_C <- ((temp_F - 32) * (5 / 9))
      return(temp_C)
}
```

calculate the temperature in Celsius

return the temperature in Celsius

```
fahrenheit_to_celsius(300)
```

```
[1] 148.8889
```

Create your own functions

```
function(arg1, arg2) {
            do something with arg1 and arg2
            return(result)}
```

Note the use of **{}**

name of function

argument: the temperature in Fahrenheit

```
fahrenheit_to_celsius <- function(temp_F) {
      temp_C <- ((temp_F - 32) * (5 / 9))
      return(temp_C)
}
```

calculate the temperature in Celsius

return the temperature in Celsius

```
fahrenheit_to_celsius(300)
```

```
[1] 148.8889
```

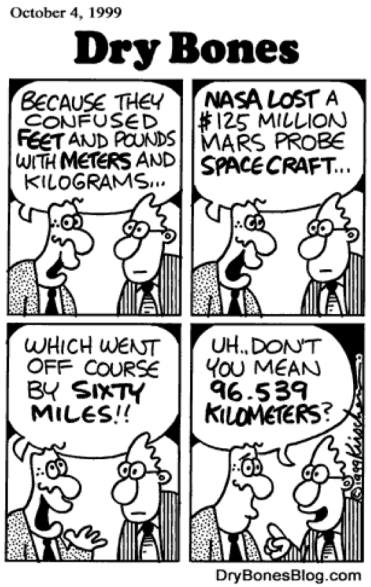
```
fahrenheit_to_celsius(c(300,20,100,800))
```

```
[1] 148.888889  -6.666667  37.777778 426.666667
```

Exercise 9: make a function

In 1999, NASA's Mars Climate Orbiter mission failed because of a mix-up with the units between the English system and the Metric system. Instead of using Watt (N m/s) they used pound-force-second (Lb f/s) and the Orbiter went off course and burned up in the atmosphere of Mars.

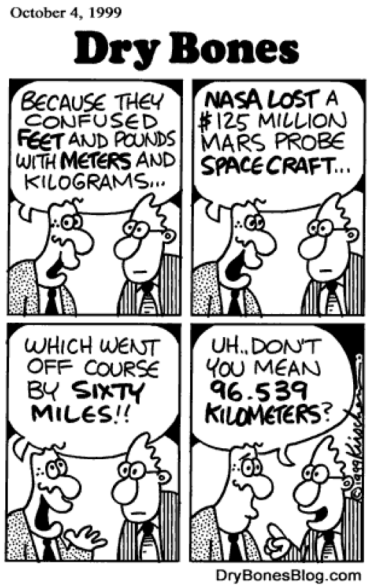
Prove that you could have saved the NASA mission by making a function that converts any Lb f/s values to N m/s
(1 Lb f/s = 1.36 N m/s)



https://drybonesblog.blogspot.com/2008/10/mars-miscalculation-1999.html

Exercise 9: make a function

In 1999, NASA's Mars Climate Orbiter mission failed because of a mix-up with the units between the English system and the Metric system. Instead of using Watt (N m/s) they used pound-force-second (Lb f/s) and the Orbiter went off course and burned up in the atmosphere of Mars.

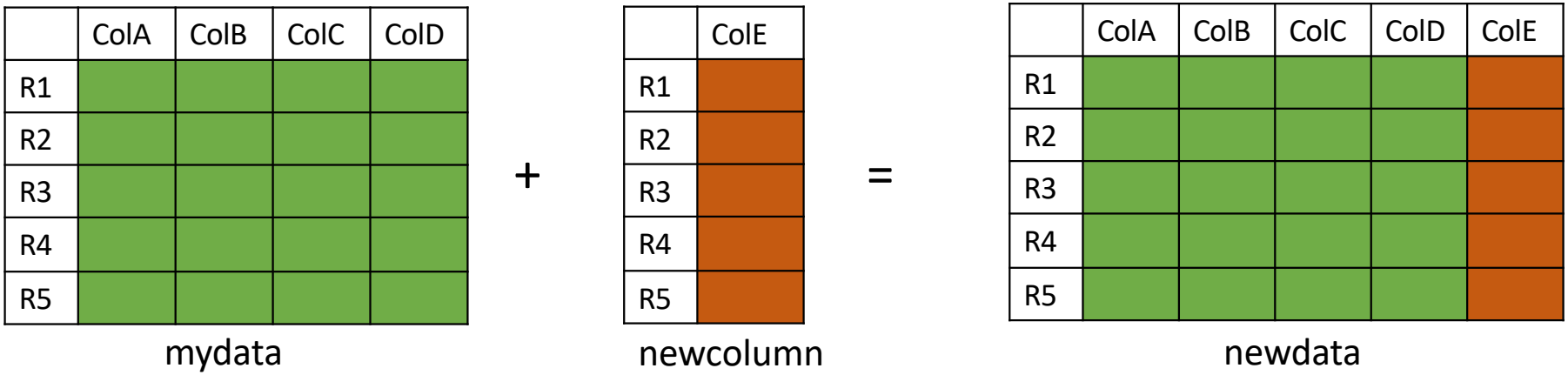


Prove that you could have saved the NASA mission by making a function that converts any Lb f/s values to N m/s
(1 Lb f/s = 1.36 N m/s)

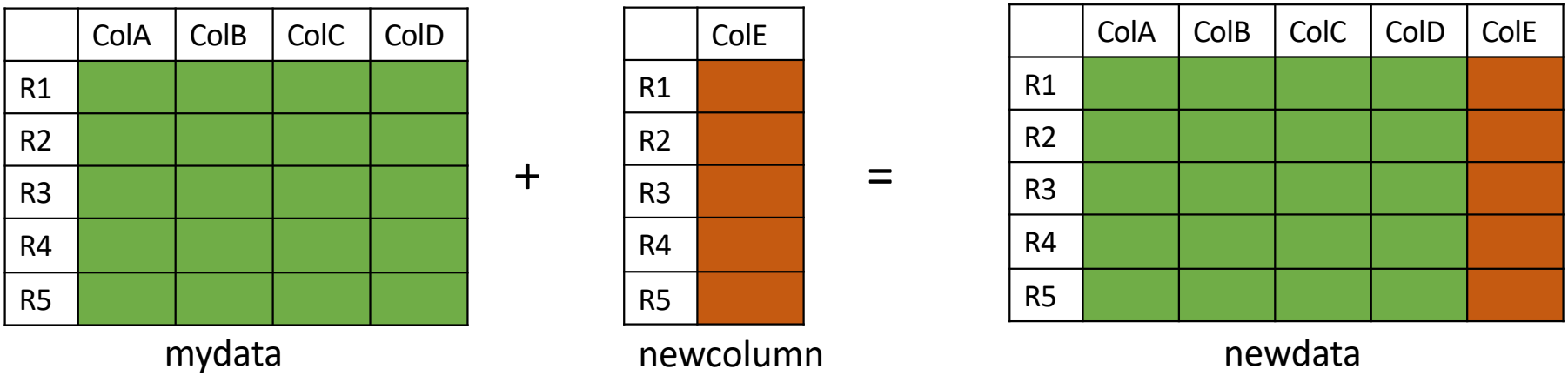
```
lbfs_to_Nms <- function(lbfs){
  Ns <- 1.36*lbfs
  return(Ns)
}
```

https://drybonesblog.blogspot.com/2008/10/mars-miscalculation-1999.html

9. Add columns to a dataframe



9. Add columns to a dataframe




```
newdata <- cbind(mydata, newcolumn)
```

name we give our new dataframe object

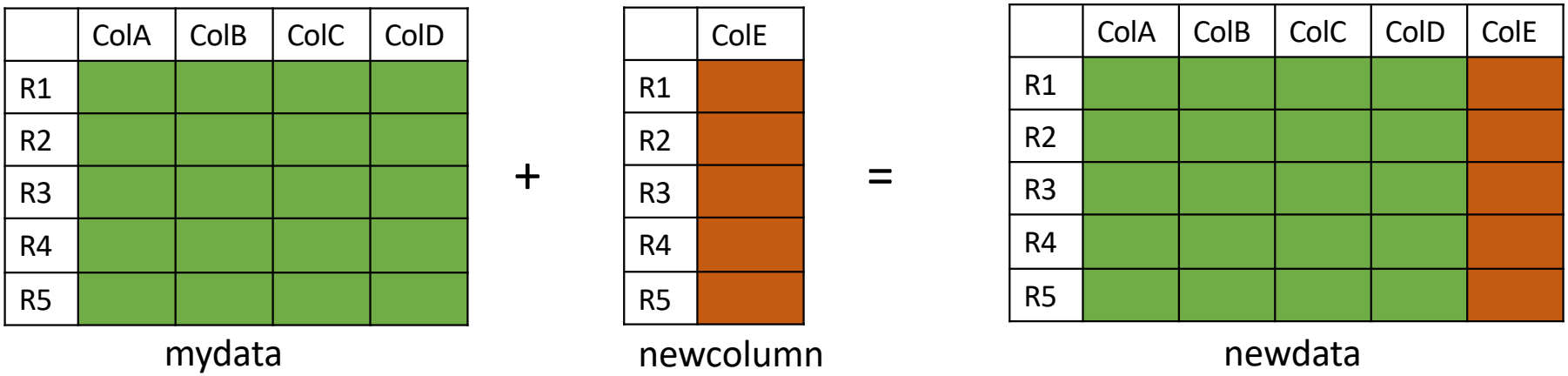
cbind = column bind

dataframe to which we want to add the column

new column we want to add

The new column should have the same number of rows as the dataframe

9. Add columns to a dataframe




```
newdata <- cbind(mydata, newcolumn)
```

new column we want to add

dataframe to which we want to add the column

cbind = column bind

name we give our new dataframe object

The new column should have the same number of rows as the dataframe

```
Gender <- c("Female", "Male", "Female", "Male", "Male", "Male")
gotGender <- cbind(got,Gender)
```

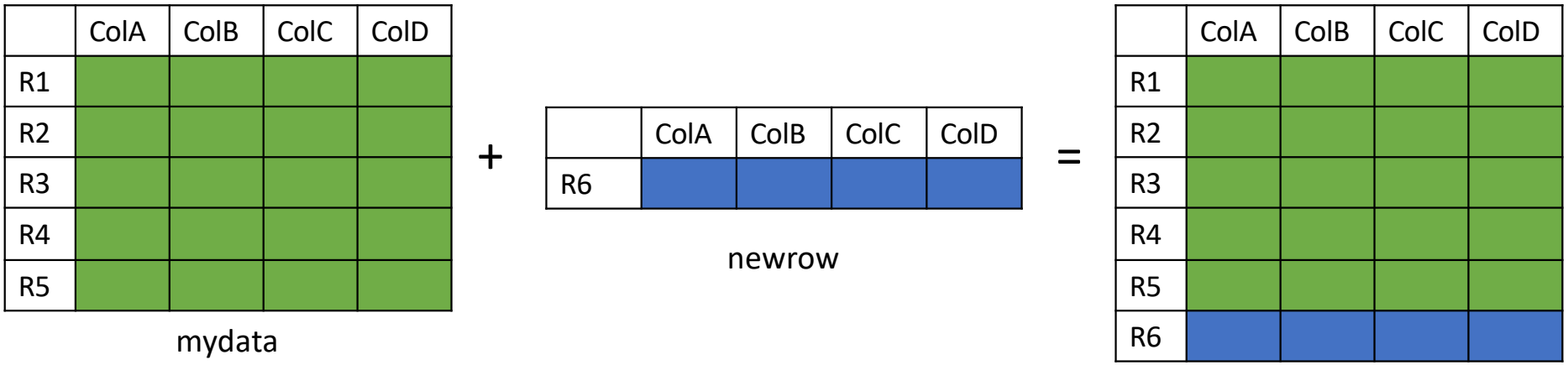
got =

	C1	C2	C3	C4
	Name	Role	Dead_S1	Age_actor
R1	Emilia_Clarke	Daenerys_Targaryen	FALSE	32
R2	Kit_Williams	Jon_Snow	FALSE	32
R3	Masie_Williams	Arya_Stark	FALSE	21
R4	Peter_Dinklage	Tyrion_Lannister	FALSE	49
R5	Jason_Mamoa	Khal_Drogo	TRUE	39
R6	Sean_Bean	Ned_Stark	TRUE	59

gotGender =

	C1	C2	C3	C4	C5
	Name	Role	Dead_S1	Age_actor	Gender
R1	Emilia_Clarke	Daenerys_Targaryen	FALSE	32	Female
R2	Kit_Williams	Jon_Snow	FALSE	32	Male
R3	Masie_Williams	Arya_Stark	FALSE	21	Female
R4	Peter_Dinklage	Tyrion_Lannister	FALSE	49	Male
R5	Jason_Mamoa	Khal_Drogo	TRUE	39	Male
R6	Sean_Bean	Ned_Stark	TRUE	59	Male

10. Add rows to a dataframe




```
newdata <- rbind(mydata, newrow)
```

new row we want to add

dataframe to which we want to add the row

rbind = row bind

name we give our new dataframe object

The new row should have the same number of columns as the dataframe

```
cersei <- c("Lena_Headey","Cersei_Lannister","FALSE","45","Female")
gotAddCersei <- cbind(gotGender, cersei)
```

gotGender =

	C1	C2	C3	C4	C5
	Name	Role	Dead_S1	Age_actor	Gender
R1	Emilia_Clarke	Daenerys_Targaryen	FALSE	32	Female
R2	Kit_Williams	Jon_Snow	FALSE	32	Male
R3	Masie_Williams	Arya_Stark	FALSE	21	Female
R4	Peter_Dinklage	Tyrion_Lannister	FALSE	49	Male
R5	Jason_Mamoa	Khal_Drogo	TRUE	39	Male
R6	Sean_Bean	Ned_Stark	TRUE	59	Male

gotAddCersei =

	C1	C2	C3	C4	C5
	Name	Role	Dead_S1	Age_actor	Gender
R1	Emilia_Clarke	Daenerys_Targaryen	FALSE	32	Female
R2	Kit_Williams	Jon_Snow	FALSE	32	Male
R3	Masie_Williams	Arya_Stark	FALSE	21	Female
R4	Peter_Dinklage	Tyrion_Lannister	FALSE	49	Male
R5	Jason_Mamoa	Khal_Drogo	TRUE	39	Male
R6	Sean_Bean	Ned_Stark	TRUE	59	Male
R7	Lena_Headey	Cersei_Lannister	FALSE	45	Female

```
> gotAddCersei <- rbind(gotGender,cersei)
Warning messages:
1: In `[<-.factor`(`*tmp*`, ri, value = "Lena_Headey") :
  invalid factor level, NA generated
2: In `[<-.factor`(`*tmp*`, ri, value = "Cersei_Lannister") :
  invalid factor level, NA generated
```

Red warnings = Panic Attack!!!!

```
> gotAddCersei <- rbind(gotGender,cersei)
Warning messages:
1: In `[<-.factor`(`*tmp*`, ri, value = "Lena_Headey") :
  invalid factor level, NA generated
2: In `[<-.factor`(`*tmp*`, ri, value = "Cersei_Lannister") :
  invalid factor level, NA generated
```

Red warnings = Panic Attack!!!!

http://www.keepcalmandposters.com/poster/137746_keep_calm_and_love_r

```
> gotAddCersei <- rbind(gotGender,cersei)
Warning messages:
1: In `[<-.factor`(`*tmp*`, ri, value = "Lena_Headey") :
  invalid factor level, NA generated
2: In `[<-.factor`(`*tmp*`, ri, value = "Cersei_Lannister") :
  invalid factor level, NA generated
> gotAddCersei
            Name                Role Dead_S1 Age_actor Gender
1  Emilia_Clarke Daenerys_Tragaryen   FALSE        32 Female
2  Kit_Harington           Jon_Snow   FALSE        32   Male
3 Masie_Williams         Arya_Stark   FALSE        21 Female
4 Peter_Dinklage   Tyrion_Lannister   FALSE        49   Male
5    Jason_Mamoa         Khal_Drogo    TRUE        39   Male
6      Sean_Bean           Ned_Stark    TRUE        59   Male
7           <NA>                <NA>   FALSE        45 Female
```

Red warnings = Panic Attack!!!!

What happened???

```
> gotAddCersei <- rbind(gotGender,cersei)
Warning messages:
1: In `[<-.factor`(`*tmp*`, ri, value = "Lena_Headey") :
  invalid factor level, NA generated
2: In `[<-.factor`(`*tmp*`, ri, value = "Cersei_Lannister") :
  invalid factor level, NA generated
> gotAddCersei
            Name                Role Dead_S1 Age_actor Gender
1  Emilia_Clarke Daenerys_Tragaryen   FALSE        32 Female
2  Kit_Harington           Jon_Snow   FALSE        32   Male
3 Masie_Williams         Arya_Stark   FALSE        21 Female
4 Peter_Dinklage   Tyrion_Lannister   FALSE        49   Male
5    Jason_Mamoa         Khal_Drogo    TRUE        39   Male
6      Sean_Bean          Ned_Stark    TRUE        59   Male
7           <NA>               <NA>   FALSE        45 Female
> str(gotAddCersei)
'data.frame':   7 obs. of  5 variables:
 $ Name     : Factor w/ 6 levels "Emilia_Clarke",..: 1 3 4 5 2 6 NA
 $ Role     : Factor w/ 6 levels "Arya_Stark","Daenerys_Tragaryen",..: 2 3 1 6 4 5 NA
 $ Dead_S1  : chr  "FALSE" "FALSE" "FALSE" "FALSE" ...
 $ Age_actor: chr  "32" "32" "21" "49" ...
 $ Gender   : Factor w/ 2 levels "Female","Male": 1 2 1 2 2 2 1
>
```

Red warnings = Panic Attack!!!!

What happened???

Factors!! phew!

1. Convert column *Role* to character

```
class(gotGender$Role)
```

To access a column in a dataframe we use $

```
gotGender$Role <- as.character(gotGender$Role)
```

Overwrite the column for the change to apply

```
class(gotGender$Role)
```

2. Do the same with column *Name*

3. Now you can add the new row

```
gotAddCersei <- rbind(gotGender, cersei)
```

4. Overwrite the old object *got* with the new object with an added column and row

```
got <- gotAddCersei
```

```
            Name                Role Dead_S1 Age_actor Gender
1  Emilia_Clarke Daenerys_Targaryen   FALSE        32 Female
2  Kit_Harington           Jon_Snow   FALSE        32   Male
3 Masie_Williams         Arya_Stark   FALSE        21 Female
4 Peter_Dinklage   Tyrion_Lannister   FALSE        49   Male
5    Jason_Mamoa         Khal_Drogo    TRUE        39   Male
6      Sean_Bean          Ned_Stark    TRUE        59   Male
7    Lena_Headey   Cersei_Lannister   FALSE        45 Female
```

11. Subset a dataframe

	C1	C2	C3	C4	C5
	Name	Role	Dead_S1	Age_actor	Gender
R1	Emilia_Clarke	Daenerys_Targaryen	FALSE	32	Female
R2	Kit_Williams	Jon_Snow	FALSE	32	Male
R3	Masie_Williams	Arya_Stark	FALSE	21	Female
R4	Peter_Dinklage	Tyrion_Lannister	FALSE	49	Male
R5	Jason_Mamoa	Khal_Drogo	TRUE	39	Male
R6	Sean_Bean	Ned_Stark	TRUE	59	Male
R7	Lena_Headey	Cersei_Lannister	FALSE	45	Female

Access data in a dataframe:

```
dataframe[row, column]
```

11. Subset a dataframe

	C1	C2	C3	C4	C5
	Name	Role	Dead_S1	Age_actor	Gender
R1	Emilia_Clarke	Daenerys_Targaryen	FALSE	32	Female
R2	Kit_Williams	Jon_Snow	FALSE	32	Male
R3	Masie_Williams	Arya_Stark	FALSE	21	Female
R4	Peter_Dinklage	Tyrion_Lannister	FALSE	49	Male
R5	Jason_Mamoa	Khal_Drogo	TRUE	39	Male
R6	Sean_Bean	Ned_Stark	TRUE	59	Male
R7	Lena_Headey	Cersei_Lannister	FALSE	45	Female

Access data in a dataframe:

```
dataframe[row, column]
```

```
got[1,1]  =  [1] Emilia_Clarke
```

11. Subset a dataframe

	C1	C2	C3	C4	C5
	Name	Role	Dead_S1	Age_actor	Gender
R1	Emilia_Clarke	Daenerys_Targaryen	FALSE	32	Female
R2	Kit_Williams	Jon_Snow	FALSE	32	Male
R3	Masie_Williams	Arya_Stark	FALSE	21	Female
R4	Peter_Dinklage	Tyrion_Lannister	FALSE	49	Male
R5	Jason_Mamoa	Khal_Drogo	TRUE	39	Male
R6	Sean_Bean	Ned_Stark	TRUE	59	Male
R7	Lena_Headey	Cersei_Lannister	FALSE	45	Female

Access data in a dataframe:

```
dataframe[row, column]
```

```
got[1,1]  =  [1] Emilia_Clarke
got[1,2]  =  [1] Daenerys_Targaryen
```

11. Subset a dataframe

	C1	C2	C3	C4	C5
	Name	Role	Dead_S1	Age_actor	Gender
R1	Emilia_Clarke	Daenerys_Targaryen	FALSE	32	Female
R2	Kit_Williams	Jon_Snow	FALSE	32	Male
R3	Masie_Williams	Arya_Stark	FALSE	21	Female
R4	Peter_Dinklage	Tyrion_Lannister	FALSE	49	Male
R5	Jason_Mamoa	Khal_Drogo	TRUE	39	Male
R6	Sean_Bean	Ned_Stark	TRUE	59	Male
R7	Lena_Headey	Cersei_Lannister	FALSE	45	Female

Access data in a dataframe:

```
dataframe[row, column]
```

```
got[1,1]  =  [1] Emilia_Clarke
got[1,2]  =  [1] Daenerys_Targaryen
got[1,]   =  All the columns in row 1
```

11. Subset a dataframe

	C1	C2	C3	C4	C5
	Name	Role	Dead_S1	Age_actor	Gender
R1	Emilia_Clarke	Daenerys_Targaryen	FALSE	32	Female
R2	Kit_Williams	Jon_Snow	FALSE	32	Male
R3	Masie_Williams	Arya_Stark	FALSE	21	Female
R4	Peter_Dinklage	Tyrion_Lannister	FALSE	49	Male
R5	Jason_Mamoa	Khal_Drogo	TRUE	39	Male
R6	Sean_Bean	Ned_Stark	TRUE	59	Male
R7	Lena_Headey	Cersei_Lannister	FALSE	45	Female

Access data in a dataframe:

```
dataframe[row, column]
```

```
got[1,1]  = [1] Emilia_Clarke
got[1,2]  = [1] Daenerys_Targaryen
got[1,]   = All the columns in row 1
got[,1]   = All the rows in column 1
```

11. Subset a dataframe

	C1	C2	C3	C4	C5
	Name	Role	Dead_S1	Age_actor	Gender
R1	Emilia_Clarke	Daenerys_Targaryen	FALSE	32	Female
R2	Kit_Williams	Jon_Snow	FALSE	32	Male
R3	Masie_Williams	Arya_Stark	FALSE	21	Female
R4	Peter_Dinklage	Tyrion_Lannister	FALSE	49	Male
R5	Jason_Mamoa	Khal_Drogo	TRUE	39	Male
R6	Sean_Bean	Ned_Stark	TRUE	59	Male
R7	Lena_Headey	Cersei_Lannister	FALSE	45	Female

Access data in a dataframe:

```
dataframe[row, column]
```

```
got[1,1]  = [1] Emilia_Clarke
got[1,2]  = [1] Daenerys_Targaryen
got[1,]   = All the columns in row 1
got[,1]   = All the rows in column 1
got[1]    = Column 1
```

11. Subset a dataframe

	C1	C2	C3	C4	C5
	Name	Role	Dead_S1	Age_actor	Gender
R1	Emilia_Clarke	Daenerys_Targaryen	FALSE	32	Female
R2	Kit_Williams	Jon_Snow	FALSE	32	Male
R3	Masie_Williams	Arya_Stark	FALSE	21	Female
R4	Peter_Dinklage	Tyrion_Lannister	FALSE	49	Male
R5	Jason_Mamoa	Khal_Drogo	TRUE	39	Male
R6	Sean_Bean	Ned_Stark	TRUE	59	Male
R7	Lena_Headey	Cersei_Lannister	FALSE	45	Female

Access data in a dataframe:

```
dataframe[row, column]
```

```
got[1,1]  = [1] Emilia_Clarke
got[1,2]  = [1] Daenerys_Targaryen
got[1,]   = All the columns in row 1
got[,1]   = All the rows in column 1
got[1]    = Column 1
```

by index = `got[,2]`

by name = `got[,"Role"]` = `got$Role` = `select(got,"Role")` *

All the rows in column 2

* tidyverse

11. Subset a dataframe

	C1	C2	C3	C4	C5
	Name	Role	Dead_S1	Age_actor	Gender
R1	Emilia_Clarke	Daenerys_Targaryen	FALSE	32	Female
R2	Kit_Williams	Jon_Snow	FALSE	32	Male
R3	Masie_Williams	Arya_Stark	FALSE	21	Female
R4	Peter_Dinklage	Tyrion_Lannister	FALSE	49	Male
R5	Jason_Mamoa	Khal_Drogo	TRUE	39	Male
R6	Sean_Bean	Ned_Stark	TRUE	59	Male
R7	Lena_Headey	Cersei_Lannister	FALSE	45	Female

Access data in a dataframe:

```
dataframe[row, column]
```

```
got[1,1]  = [1] Emilia_Clarke
got[1,2]  = [1] Daenerys_Targaryen
got[1,]   = All the columns in row 1
got[,1]   = All the rows in column 1
got[1]    = Column 1
```

by index = `got[,2]`

by name = `got[,"Role"]` = `got$Role` = `select(got,"Role")` *

All the rows in column 2

by value in a column =

```
subset(got, Gender =="Female")
got[got$Gender =="Female",]
filter(got,Gender =="Female")
```

*

All rows with the females

* tidyverse

Exercise 10. Table handling

1. get all rows in column 4
2. get columns 2 and 3
3. get columns 2 and 4
4. select male actors
5. select all columns except column 1
6. select actors over 40 years old
7. select male actors over 40 years old

Exercise 10. Table handling

1. get all rows in column 4 `got[,4]` = `got[,"Age_actor"]` = `got$Age_actor`
2. get columns 2 and 3 `got[,2:3]` = `got[,c(2,3)]`
3. get columns 2 and 4 `got[,c(2,4)]`
4. select male actors `subset(got, Gender =="Male")` = `got[got$Gender =="Male",]`
5. select all columns except column 1 `got[,-1]` = `got[-1]` = `got[2:5]`
6. select actors over 40 years old `subset(got, Age > 40)` = `got[got$Age > 40,]`
7. select male actors over 40 years old `subset(got, Age > 40 & Gender =="Male")`

   `got[got$Age > 40 & got$Gender =="Male",]`

12. If-else statements

- if (condition is TRUE) {do something}

```
x <- 3
if (x == 3) {x * 2}
[1] 6
```

Note the use of **{}**

12. If-else statements

- **if** (condition is TRUE) {do something}

```
x <- 3
if (x == 3) {x * 2}
[1] 6
```

Note the use of **{}**

- **if** (condition is TRUE){do something} **else**{do something else}

```
x <- 4
if (x == 3) {x * 2} else {x * 3}
[1] 12
```

13.For loops

- Repeat the same operation over a list of elements
- each repetition is called an iteration. Tipically **i**

```
some_numbers <- seq(1,10)
```

```
empty_vector <- vector(mode="numeric", length=10)
```

```
for(i in 1:10) {
empty_vector[i] <- some_numbers[i]^2
  print(empty_vector[i])
}
```

```
empty_vector
```

Calculate the square of the numbers 1 to 10

Again we have **{}**

14. Save and export

- export tables

```
write.csv(got,"Name_of_file.csv", row.names = FALSE)
```

14. Save and export

- export tables

```
write.csv(got,"Name_of_file.csv", row.names = FALSE)
```

- export figures

```
pdf("Name_of_image.pdf”, height = 6, width = 7)
plot(x = 1:10, y = 1:10)
dev.off()
```

1. Open new window
2. Write image
3. Close window

14. Save and export

- export tables

```
write.csv(got,"Name_of_file.csv", row.names = FALSE)
```

- export figures

```
pdf("Name_of_image.pdf", height = 6, width = 7)
plot(x = 1:10, y = 1:10)
dev.off()
```

1. Open new window
2. Write image
3. Close window

- Figure formats:
- Vectorial : `postscript()` `pdf()`
- Raster: `png()` `tiff()` `jpeg()`

Vectorial images are better for publication figures, they are scalable and have much better quality

14. Save and export

- export tables

```
write.csv(got,"Name_of_file.csv", row.names = FALSE)
```

- export figures

```
pdf("Name_of_image.pdf", height = 6, width = 7)
plot(x = 1:10, y = 1:10)
dev.off()
```

1. Open new window
2. Write image
3. Close window

  - Figure formats:
  - Vectorial : `postscript()` `pdf()`
  - Raster: `png()` `tiff()` `jpeg()`

Vectorial images are better for publication figures, they are scalable and have much better quality

- export a workspace

```
save.image("Name_of_workspace.Rdata")
```

14. Save and export

- export tables

```
write.csv(got,"Name_of_file.csv", row.names = FALSE)
```

- export figures

```
pdf("Name_of_image.pdf", height = 6, width = 7)
plot(x = 1:10, y = 1:10)
dev.off()
```

1. Open new window
2. Write image
3. Close window

  - Figure formats:
  - Vectorial : `postscript()` `pdf()`
  - Raster: `png()` `tiff()` `jpeg()`

Vectorial images are better for publication figures, they are scalable and have much better quality

- export a workspace

```
save.image("Name_of_workspace.Rdata")
```

***Save to a different folder: Add the whole path when you write the name of the file

```
write.csv(got,"Path/to/folder/Name_of_file.csv", row.names = F)
```

Open/Save pairs

Open	Save	File class
`read.csv()`	`write.csv()`	csv file
`read.table()`	`write.table()`	txt file
`load()`	`save()`	1 or several objects as Rdata
`load()`	`save.image()`	Workspace as Rdata
`readRDS()`	`saveRDS`	1 object as Rdata

R Markdown

R markdown

- **file.Rmd** = Upgrade from the traditional script (.R).
- It combines text, code, plots, references.
- Allows reproducible research
- Super useful to send reports to collaborators or make tutorials.
- It also looks very professional

16S_vegan_tutorial

Tamara

12/11/2018

```
knitr::opts_chunk$set(include = TRUE)
library(vegan)
```

```
## Loading required package: permute
```

```
## Loading required package: lattice
```

```
## This is vegan 2.5-3
```

```
library(ggplot2)
library(ade4)
library(ggvegan)
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
library(reshape2)
library(RColorBrewer)
```

16S diversity tutorial

To work with diversity data you need 3 files:

- OTU table (or if you are doing metagenomes/transcriptomes, gene counts). The first column will be the OTU name, the next columns will be each of your samples. The data will be the counts of an OTU in each sample
- Taxonomic assigment table (or function annotation for metagenomes/trascriptomes). The first column will also be the OTU name (this has to match the OTU table) and the columns will be the taxonomic assignmnents
- metadata table. The first column here should be the name of your samples (the same as in the OTU table) and each column is a variable of interest or your sample (both continuous and categorical)

The metadata table is typically a separate file, but depending on the pipeline you use for the annotation, you may get the OTU table together or separate from your Taxonomic table. We will make an example with the OTU and Taxonomic files merged, since this will require an extra step to separate them.

1. LOADING AND PREPARING DATA

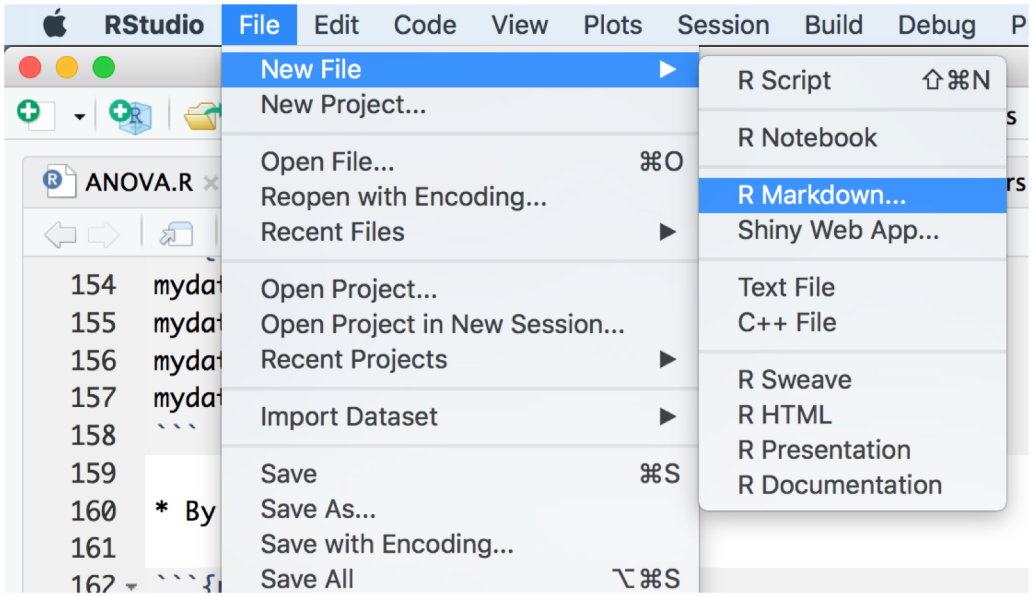
1.1 Start by loading your data and taking a look at it:

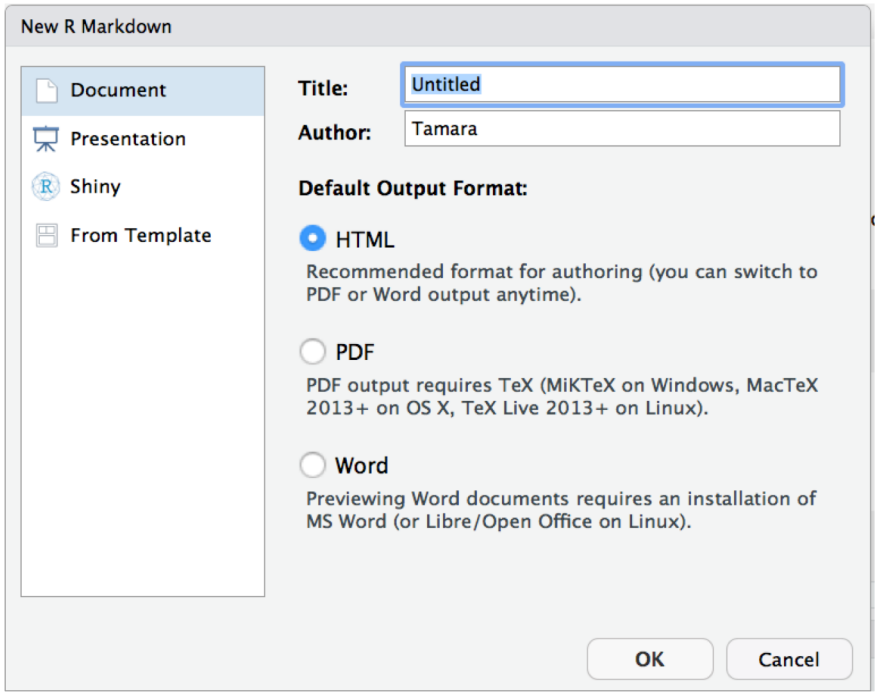
```
#setwd("Day4_16S_diversity_MDS/")
microbial16S <- read.csv("16S_OTU_TAX.csv")
dim(microbial16S) ## 994 rows x 45 columns
```

```
## [1] 994  45
```

```
names(microbial16S)
```

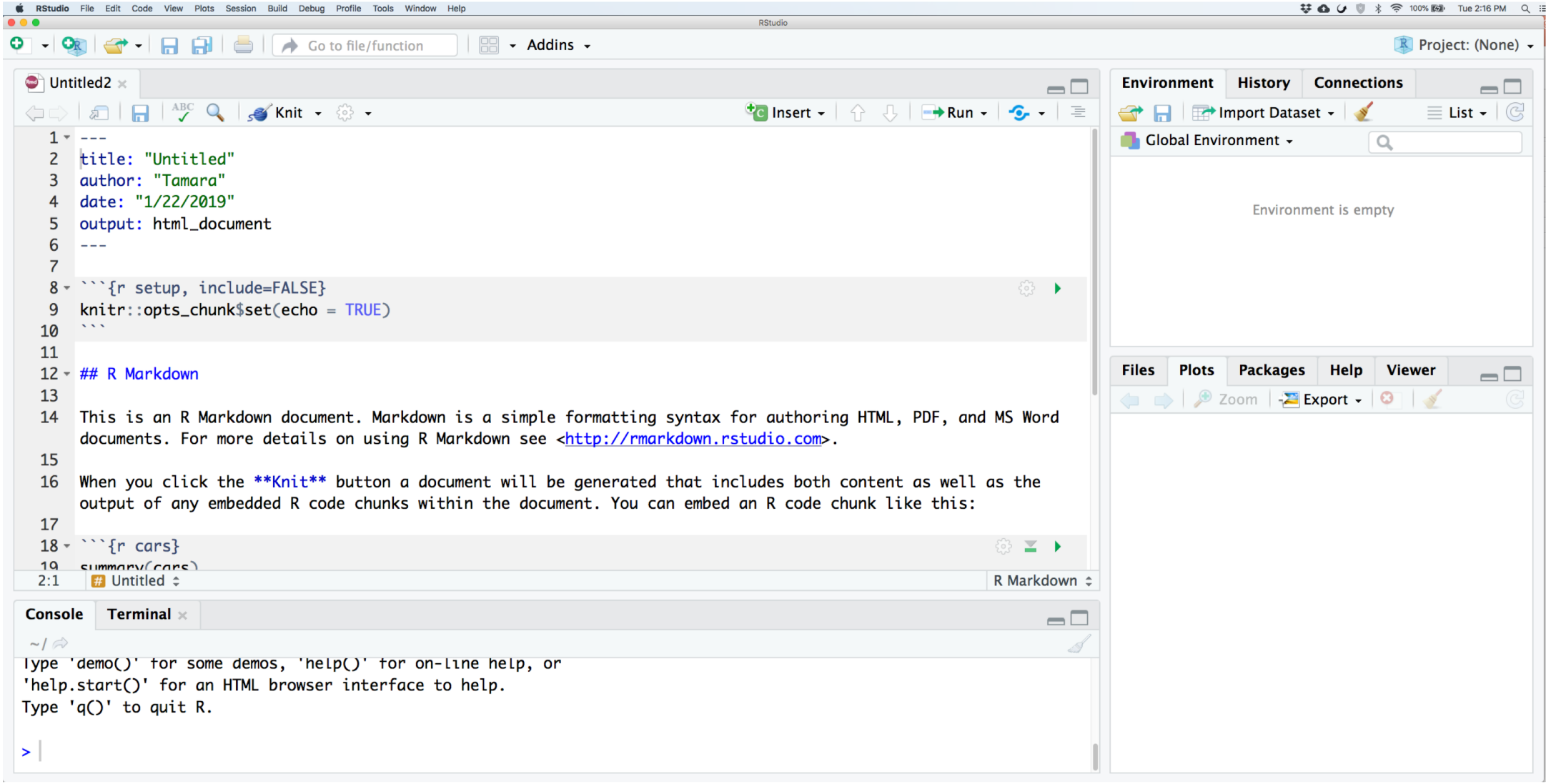
R markdown example



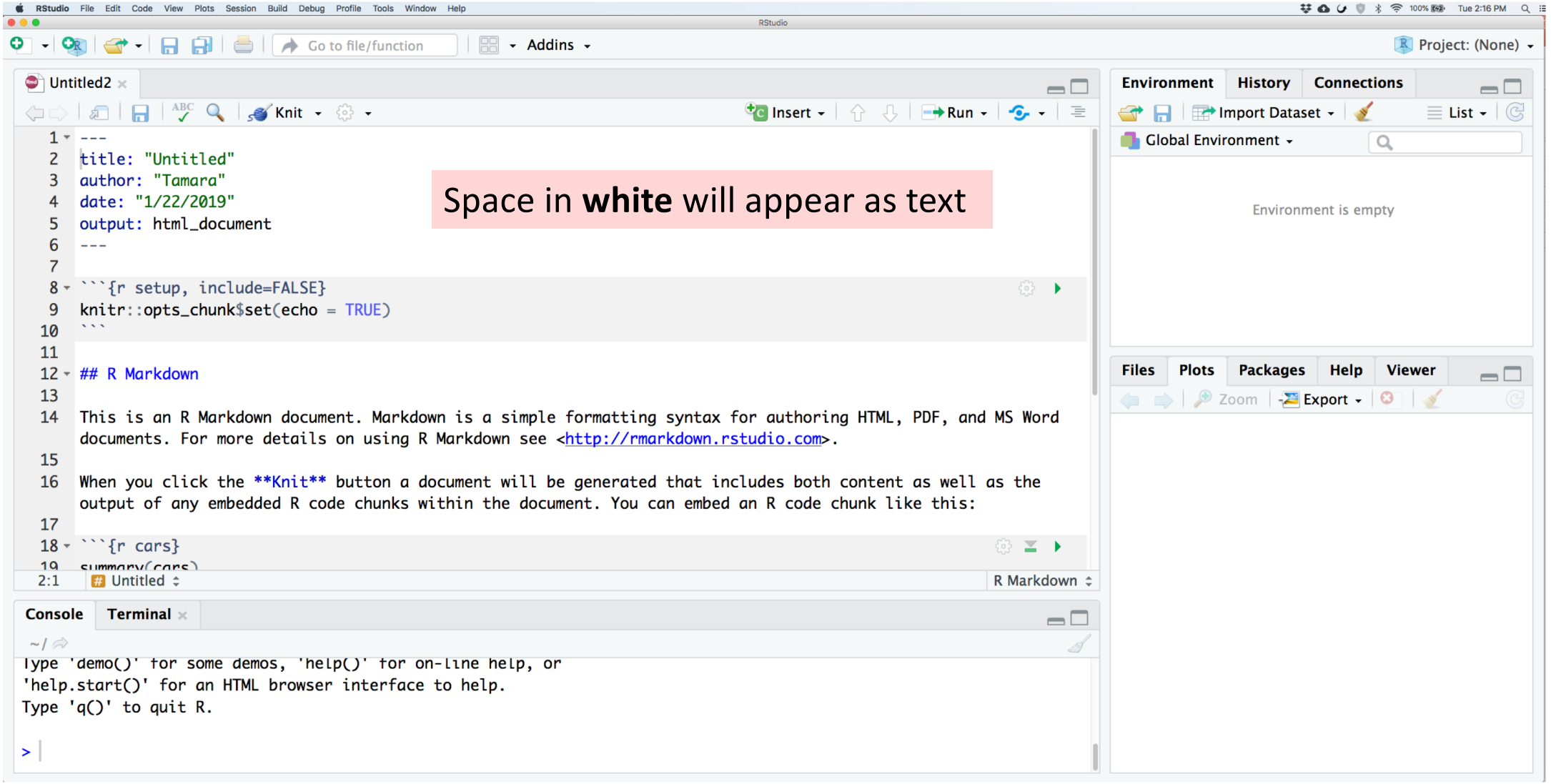


File → New File → R Markdown

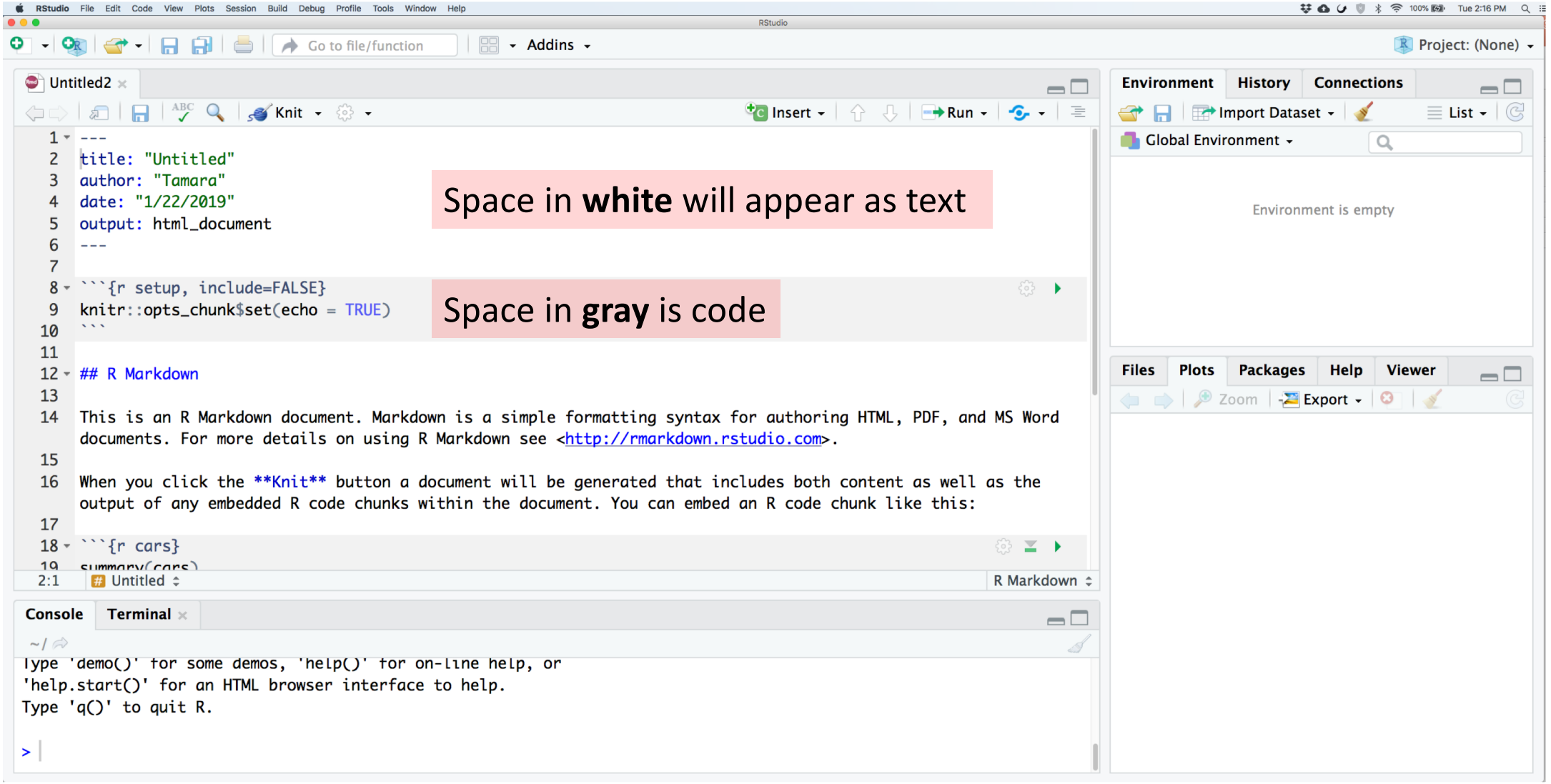
Name file and select HTML
You need to install LaTeX for pdf
http://www.tug.org/mactex/

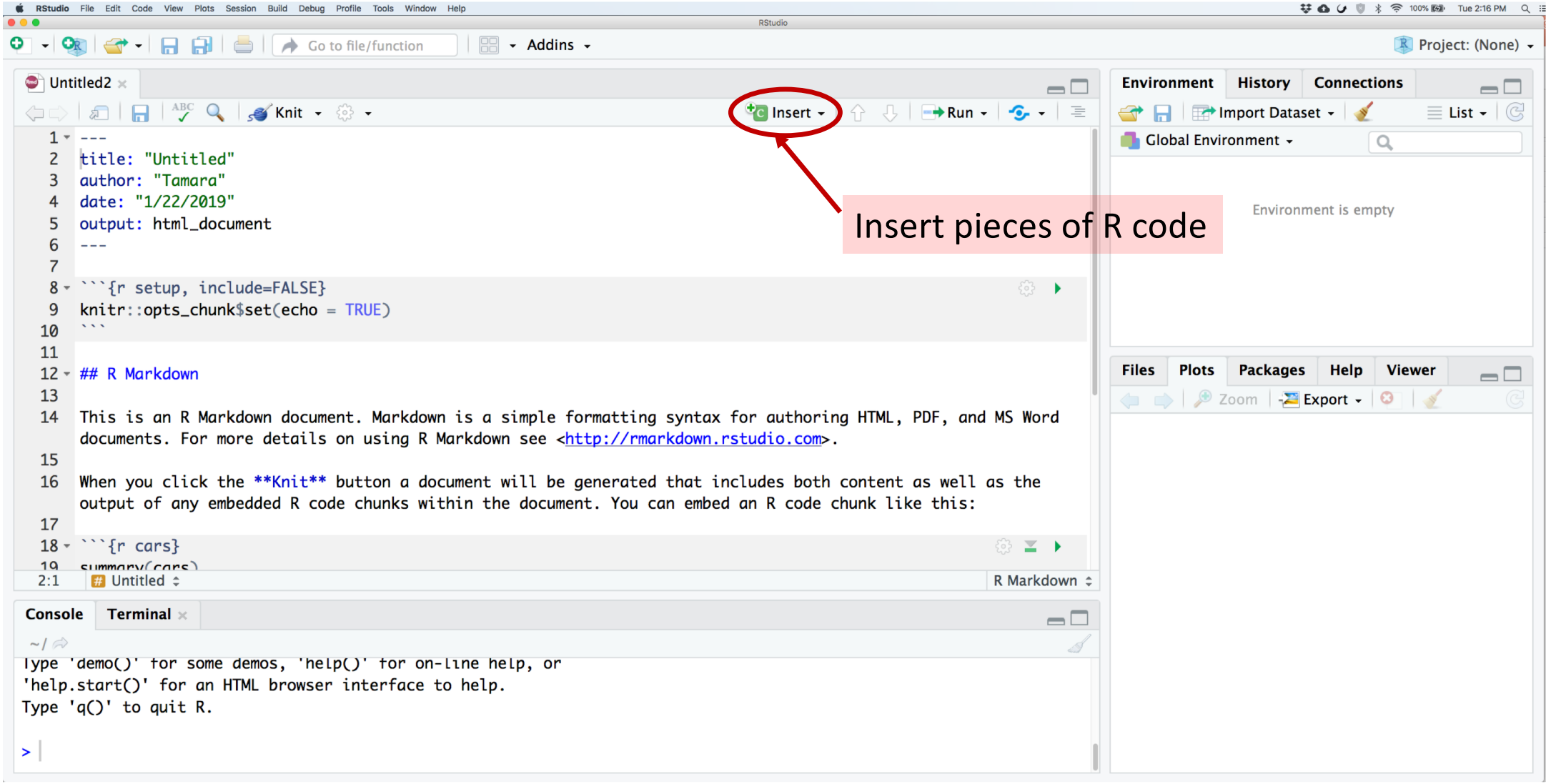

RStudio File Edit Code View Plots Session Build Debug Profile Tools Window Help
Tue 2:16 PM
RStudio
Go to file/function
Addins
Project: (None)
Untitled2
Knit
Insert
Run
---
title: "Untitled"
author: "Tamara"
date: "1/22/2019"
output: html_document

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```
R Markdown
This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.
When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:
```{r cars}
summary(cars)
2:1
Untitled
R Markdown
Console
Terminal
~/
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
Connections
Import Dataset
List
Global Environment
Environment is empty
Files
Plots
Packages
Help
Viewer
Zoom
Export
```

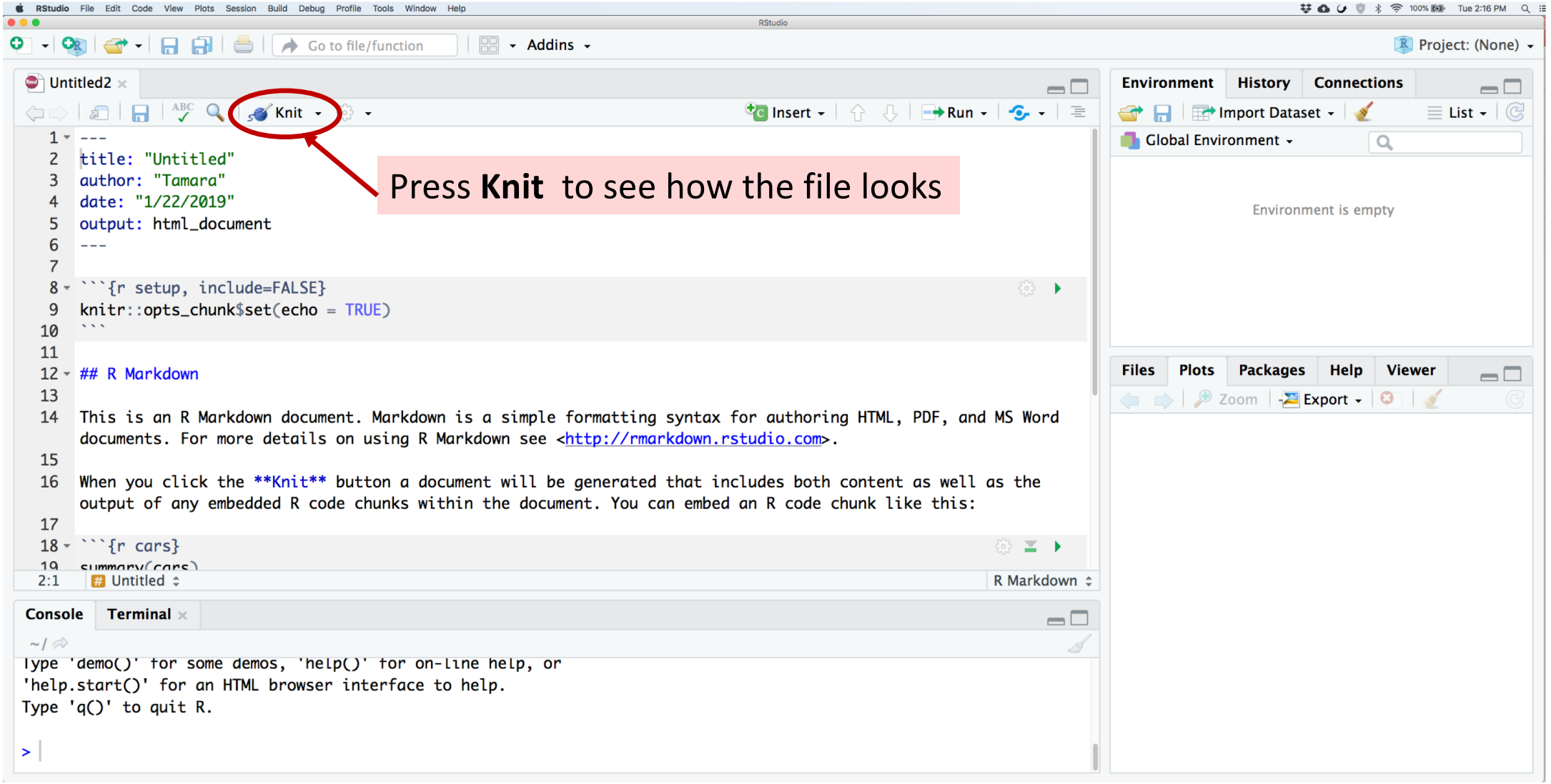

Space in **white** will appear as text
Untitled2
Knit
Insert
Run
---
title: "Untitled"
author: "Tamara"
date: "1/22/2019"
output: html_document

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```
R Markdown
This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.
When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:
```{r cars}
summary(cars)
2:1
Untitled
R Markdown
Console
Terminal
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
Connections
Import Dataset
List
Global Environment
Environment is empty
Files
Plots
Packages
Help
Viewer
Zoom
Export
```

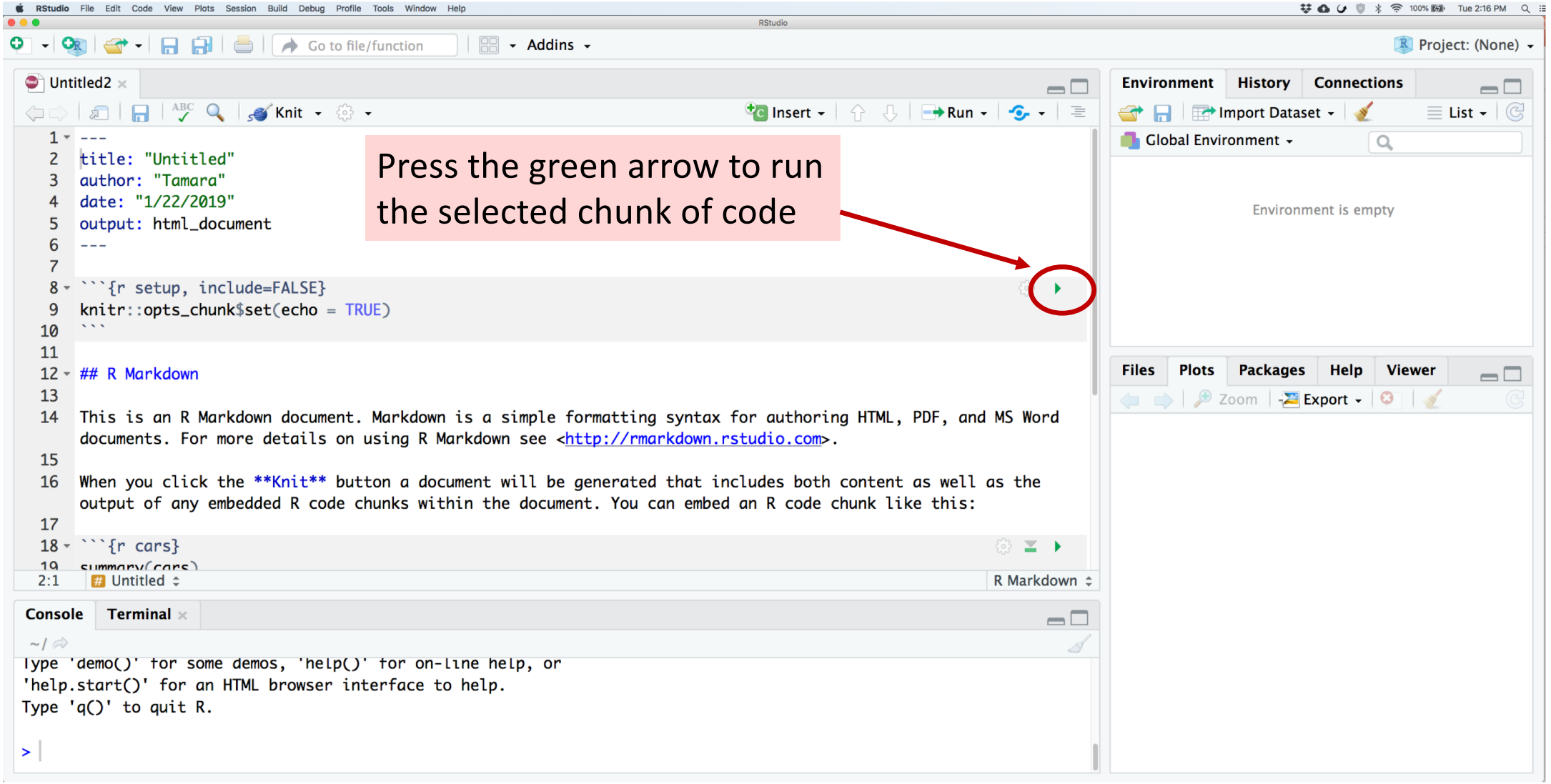

Space in **white** will appear as text
Space in **gray** is code
Untitled2
Knit
Insert
Run
Go to file/function
Addins
Project: (None)
Environment
History
Connections
Import Dataset
List
Global Environment
Environment is empty
Files
Plots
Packages
Help
Viewer
Zoom
Export
Console
Terminal
Untitled
R Markdown


Insert pieces of R code
Untitled2
Knit
Insert
Run
---
title: "Untitled"
author: "Tamara"
date: "1/22/2019"
output: html_document

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```
R Markdown
This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.
When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:
```{r cars}
summary(cars)
2:1
Untitled
R Markdown
Console
Terminal
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
>
Environment
History
Connections
Import Dataset
List
Global Environment
Environment is empty
Files
Plots
Packages
Help
Viewer
Zoom
Export
```


Press **Knit** to see how the file looks
Untitled2
Knit
Insert
Run
---
title: "Untitled"
author: "Tamara"
date: "1/22/2019"
output: html_document

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```
R Markdown
This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.
When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:
```{r cars}
summary(cars)
2:1
Untitled
R Markdown
Console
Terminal
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
Connections
Import Dataset
List
Global Environment
Environment is empty
Files
Plots
Packages
Help
Viewer
Zoom
Export
```


Press the green arrow to run the selected chunk of code
Untitled2
Knit
Insert
Run
Environment
History
Connections
Import Dataset
List
Global Environment
Environment is empty
Files
Plots
Packages
Help
Viewer
Zoom
Export
Console
Terminal
R Markdown

Let’s check out more options with the tutorial

folder: ~/R_Course_2019/Day1_IntroR/Rmarkdown/
script: Rmarkdown_tutorial.Rmd

Base plots

folder: ~/R_Course_2019/Day1_IntroR/BasePlot/

script: BasePlot.Rmd

Plotting systems in R

- **Base plotting system** ("fast and easy" but ugly)
- Lattice package
- ggplot2 package (not so fast and easy but highly customable and effective effective for creating publication quality graphics.)

Base Plot

- One of the most powerful functions of R is it's ability to produce a wide range of graphics to quickly and easily visualise data.
- Plots can be replicated, modified and even publishable with just a handful of commands.

Open the file **BasePlot.Rmd**

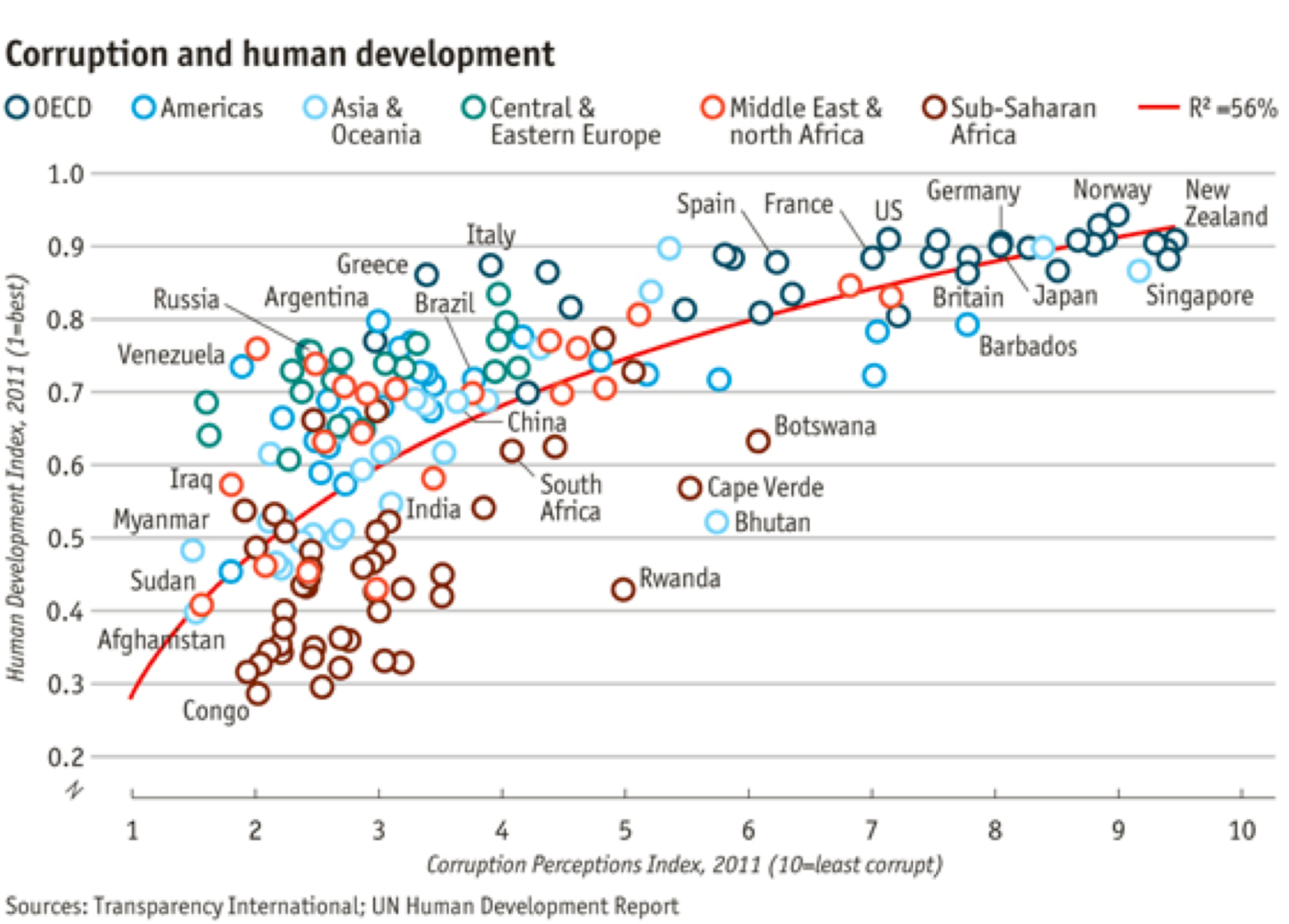
Main options

- Type of plot
  - Scatterplots : plot()
  - Bar charts : barplot()
  - Box plots : boxplot()
  - Histograms : hist()
- Text
  - Size : cex
  - Font family : family
- Lines
  - Line style : lty
  - Line width : lwd
- Points
  - Symbol : pch()
  - Color : col()
  - Fill color : bg

Exercise

The goal is using R to remake a basic version of this plot using the file

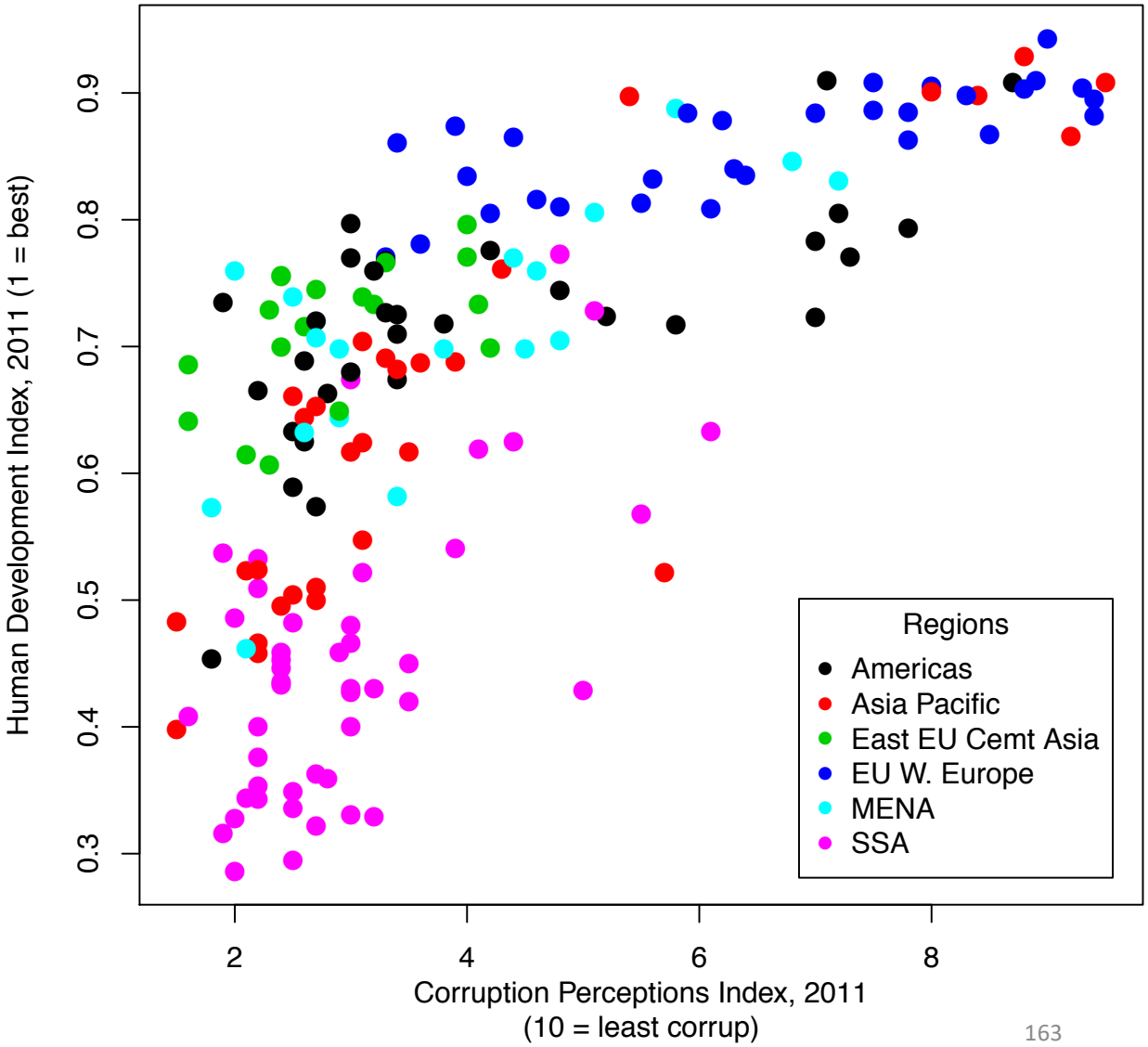
EconomistData.csv



Exercise

The goal is using R to remake a basic version of this plot using the file

EconomistData.csv



Exercise Solution

```
data <- read.csv("EconomistData.csv", row.names = 1, header = T)
plot(… ,
     …,
     col = …,
     pch = …,
     cex = …,
     xlab = "…”,
     ylab = "…"
)
```

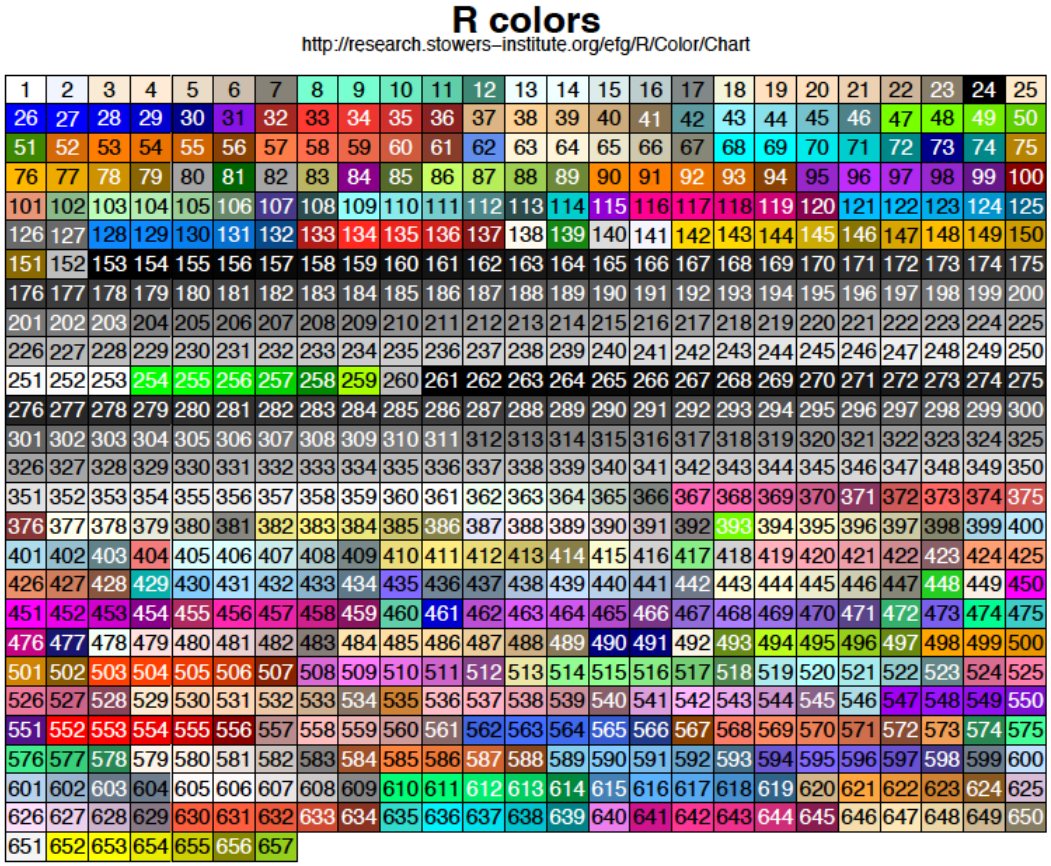
Exercise Solution

```
data <- read.csv("EconomistData.csv", row.names = 1, header = T)
plot(data$CPI,
     data$HDI,
     col = data$Region,
     pch = 16,
     cex = 1.5,
     xlab = "Corruption Perceptions Index, 2011\n(10 = least corrup)",
     ylab = "Human Development Index, 2011 (1 = best)”
)
```

Color in R

folder: ~/R_Course_2019/Day1_IntroR/Color/
script: Color.Rmd

R has 657 unique colors



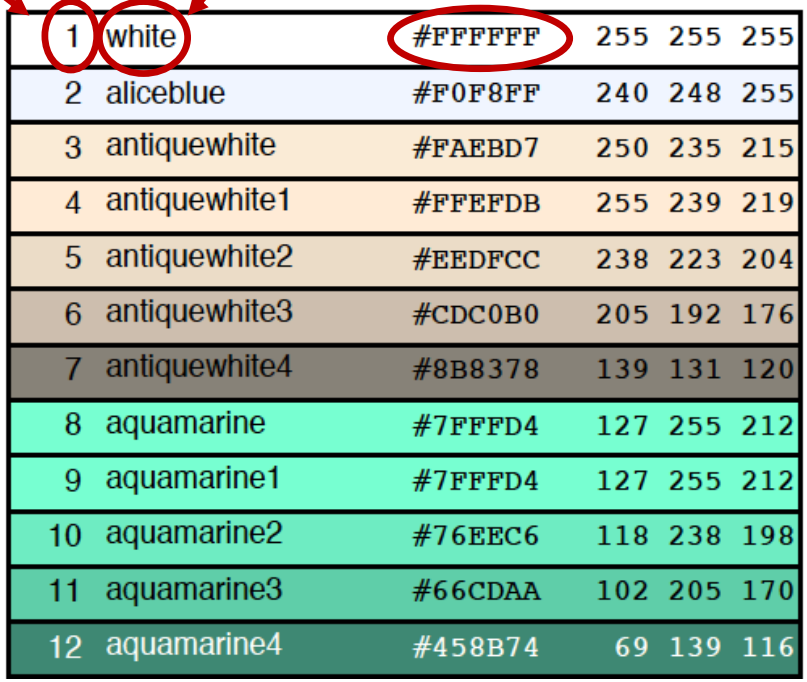
Day1_Intro/Color/ColorChart.pdf

How to access individual colors

by number: `colors()[1]`

by name: `"white"`

by hexadecimal code: `"#FFFFFF"`



1	white	#FFFFFF	255	255	255
2	aliceblue	#F0F8FF	240	248	255
3	antiquewhite	#FAEBD7	250	235	215
4	antiquewhite1	#FFEFDB	255	239	219
5	antiquewhite2	#EEDFCC	238	223	204
6	antiquewhite3	#CDC0B0	205	192	176
7	antiquewhite4	#8B8378	139	131	120
8	aquamarine	#7FFFD4	127	255	212
9	aquamarine1	#7FFFD4	127	255	212
10	aquamarine2	#76EEC6	118	238	198
11	aquamarine3	#66CDAA	102	205	170
12	aquamarine4	#458B74	69	139	116

Exercise. Access individual colors

- Look for color 12 (aquamarine4) in the ColorChart.pdf file
- Complete the color argument in the following plot:

```
plot(x=1:10, y=1:10, col= name , pch=15,cex=1.2) ## name
points(x=1:10,y=2:11, col= number , pch=16,cex=1.2) ## number
points(x=1:10, y=3:12, col= hex code , pch=17,cex=1.2) ## hex code
```

`points()` adds a new layer of points over the existing plot

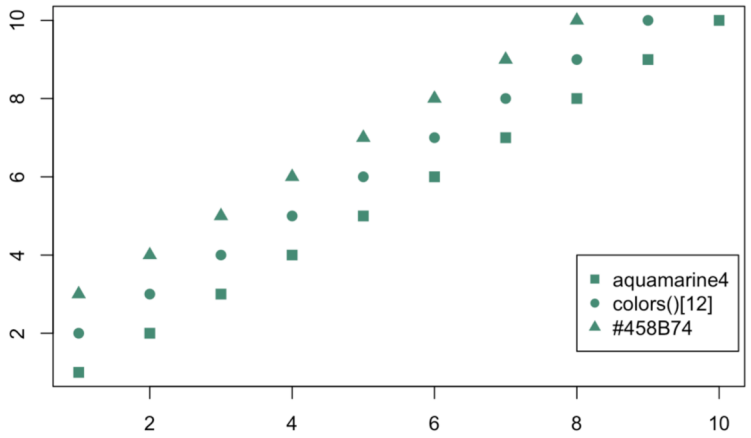
Exercise. Access individual colors

- Look for color 12 (aquamarine4) in the ColorChart.pdf file
- Complete the color argument in the following plot:

```
plot(x=1:10, y=1:10, col="aquamarine4", pch=15,cex=1.2) ## name
```

```
points(x=1:10,y=2:11, col=colors()[12], pch=16,cex=1.2) ## number
```

```
points(x=1:10, y=3:12, col="#458B74", pch=17,cex=1.2) ## hex code
```



Exercise. Choose colors

- Choose any 3 colors in the ColorChart.pdf file
- Complete the color argument in the following plot (write the colors by name, index or hex code, whichever you prefer)

```
plot(x=1:3, y=1:3, col=c(col1,col2,col3) ,pch=15,cex=1.2)
```

Exercise. Choose colors

- Choose any 3 colors in the ColorChart.pdf file
- Complete the color argument in the following plot (write the colors by name, index or hex code, whichever you prefer)

```
plot(x=1:3, y=1:3, col=c(col1,col2,col3) ,pch=15,cex=1.2)
```

```
plot(x=1:3, y=1:3, col=c(colors()[91],"darkturquoise",colors()[641]),
pch=15,cex=1.2)
```

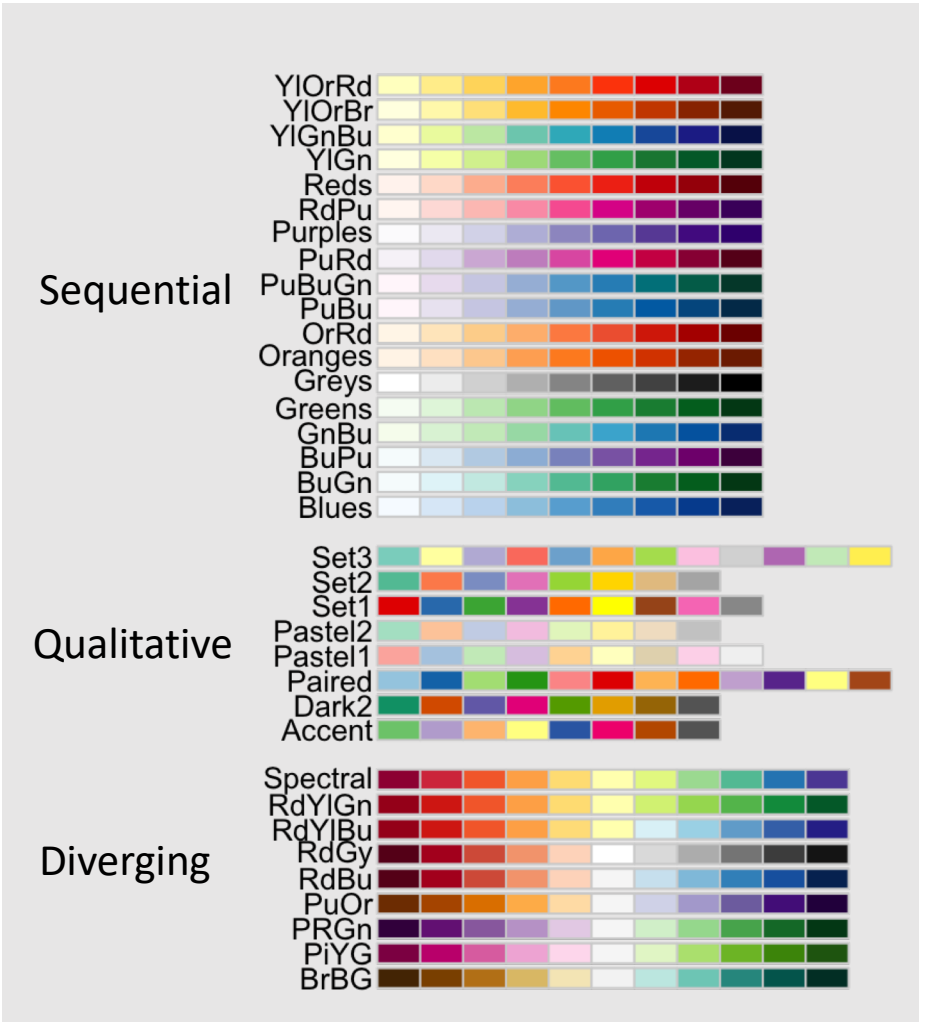
```
plot(x=1:3, y=1:3, col=colors()[c(91,12,641)], pch=15,cex=1.2)
```

Package RColorBrewer

- Finding the right combination of colors can take forever

Package RColorBrewer

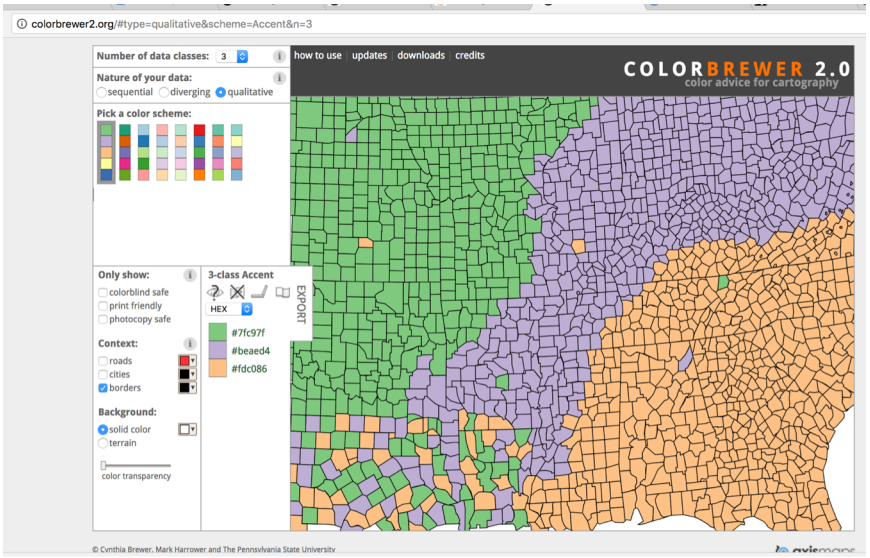
- Finding the right combination of colors can take forever
- RColorBrewer has ready made palettes

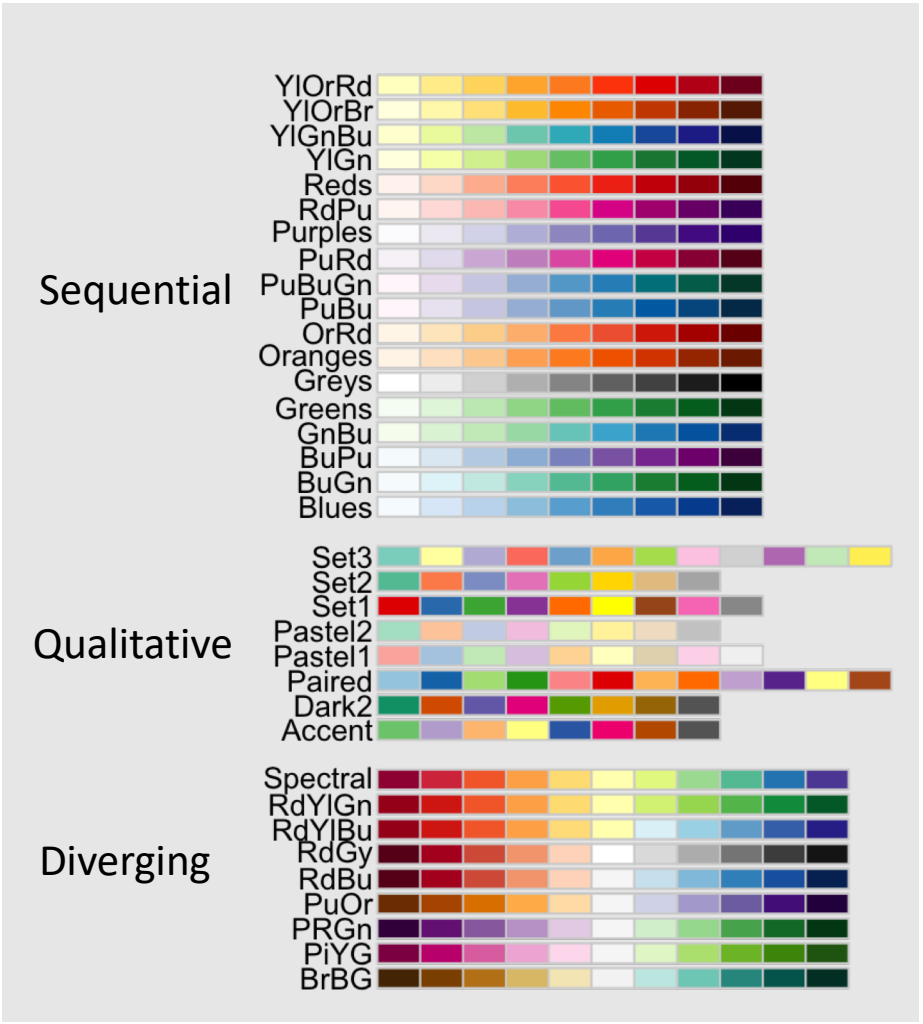


Package RColorBrewer

- Finding the right combination of colors can take forever
- RColorBrewer has ready made palettes
- Check how the colors look (also shows color blind safe options!)

http://colorbrewer2.org/#type=qualitative&scheme=Accent&n=3





Package RColorBrewer

1. Choose from ready made palettes:

```
library(RColorBrewer)
```
→ load package

Package RColorBrewer

1. Choose from ready made palettes:

```
library(RColorBrewer)
```
→ load package

```
display.brewer.all()
```
→ show built in palettes

Package RColorBrewer

1. Choose from ready made palettes:

```
library(RColorBrewer)
```
→ load package

```
display.brewer.all()
```
→ show built in palettes

```
display.brewer.all(colorblindFriendly=TRUE)
```
→ show only color blind friendly

Package RColorBrewer

1. Choose from ready made palettes:

```
library(RColorBrewer)
```
→ load package

```
display.brewer.all()
```
→ show built in palettes

```
display.brewer.all(colorblindFriendly=TRUE)
```
→ show only color blind friendly

```
brewer.pal(n=9,name ="Spectral")
```
→ access colors in the palette

```
[1] "#D53E4F" "#F46D43" "#FDAE61" "#FEE08B" "#FFFFBF" "#E6F598" "#ABDDA4" "#66C2A5" "#3288BD"
```

Package RColorBrewer

1. Choose from ready made palettes:

```
library(RColorBrewer)
```
→ load package

```
display.brewer.all()
```
→ show built in palettes

```
display.brewer.all(colorblindFriendly=TRUE)
```
→ show only color blind friendly

```
brewer.pal(n=9,name ="Spectral")
```
→ access colors in the palette

```
[1] "#D53E4F" "#F46D43" "#FDAE61" "#FEE08B" "#FFFFBF" "#E6F598" "#ABDDA4" "#66C2A5" "#3288BD"
```

How to use it:

```
plot(1:20, col=brewer.pal(n=9,name ="Spectral") ,pch=19)
```

Package RColorBrewer

1. Choose from ready made palettes:

```
library(RColorBrewer)
```
→ load package

```
display.brewer.all()
```
→ show built in palettes

```
display.brewer.all(colorblindFriendly=TRUE)
```
→ show only color blind friendly

```
brewer.pal(n=9,name ="Spectral")
```
→ access colors in the palette

```
[1] "#D53E4F" "#F46D43" "#FDAE61" "#FEE08B" "#FFFFBF" "#E6F598" "#ABDDA4" "#66C2A5" "#3288BD"
```

How to use it:

```
plot(1:20, col=brewer.pal(n=9,name ="Spectral"),pch=19)
```

Package RColorBrewer

1. Choose from ready made palettes:

```
library(RColorBrewer)
```
→ load package

```
display.brewer.all()
```
→ show built in palettes

```
display.brewer.all(colorblindFriendly=TRUE)
```
→ show only color blind friendly

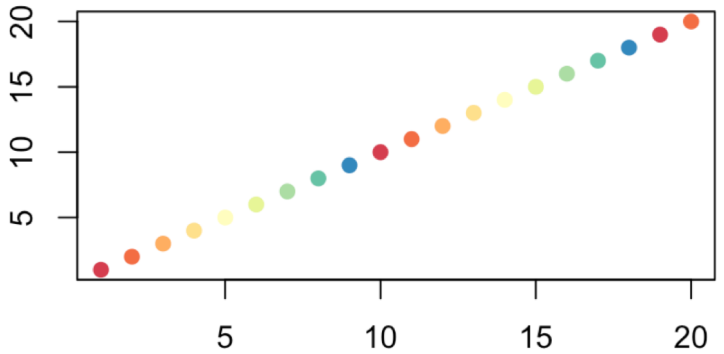
```
brewer.pal(n=9,name ="Spectral")
```
→ access colors in the palette

```
[1] "#D53E4F" "#F46D43" "#FDAE61" "#FEE08B" "#FFFFBF" "#E6F598" "#ABDDA4" "#66C2A5" "#3288BD"
```

How to use it:

```
plot(1:20, col=brewer.pal(n=9,name ="Spectral"),pch=19)
```

We have less colors
than points!
the colors are repeated



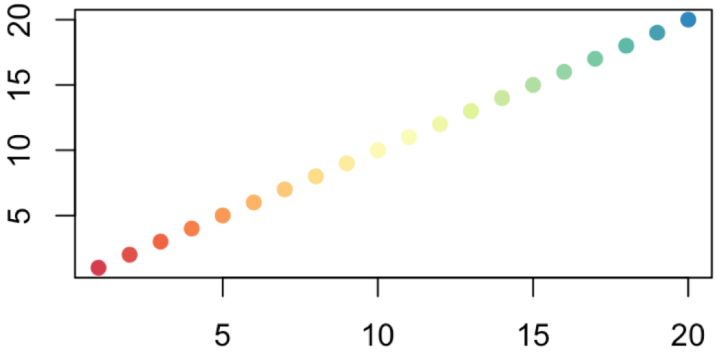
Package RColorBrewer

colorRampPalette()

2. Extend/Reduce colors in ready-made palettes:

```
myPal <- colorRampPalette(brewer.pal(9,"Spectral"))
plot(1:20, col= myPal(20),pch=19)
```

Extend



Package RColorBrewer

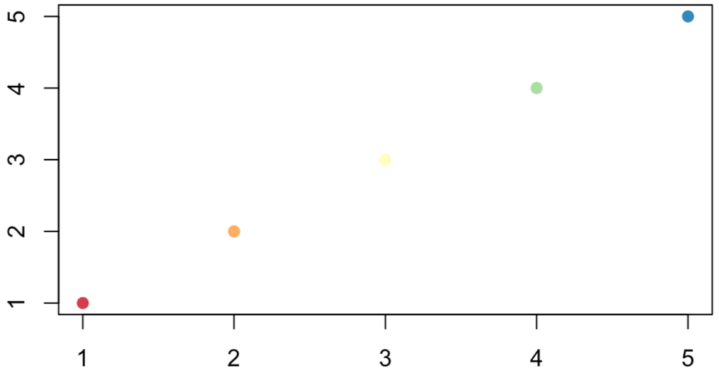
`colorRampPalette()`

2. Extend/Reduce colors in ready-made palettes:

```
myPal <- colorRampPalette(brewer.pal(9,"Spectral"))
plot(1:20, col= myPal(20),pch=19)
plot(1:5, col= myPal(5),pch=19)
```

Extend

Reduce



Package RColorBrewer

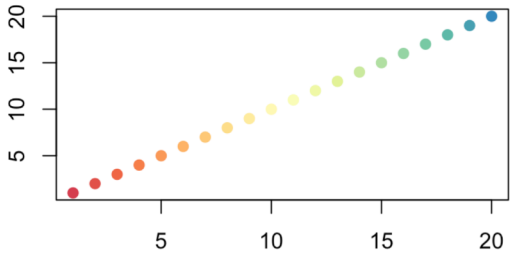
colorRampPalette()

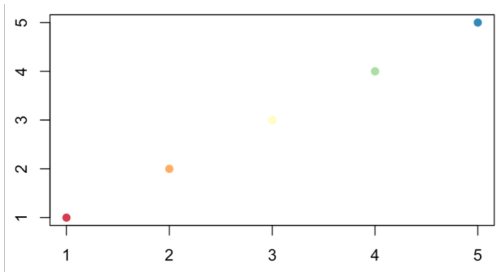
2. Extend/Reduce colors in ready-made palettes:

```
myPal <- colorRampPalette(brewer.pal(9,"Spectral"))
plot(1:20, col= myPal(20),pch=19)
plot(1:5, col= myPal(5),pch=19)
```

Extend

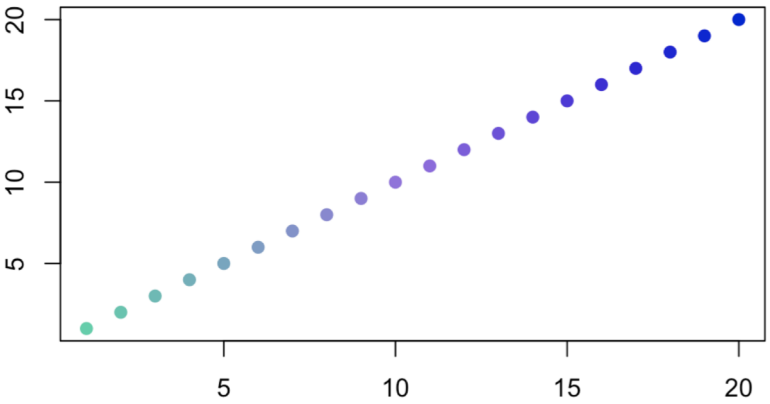
Reduce





3. Or make your own palettes!

```
myPal2 <- colorRampPalette(c("mediumaquamarine","mediumpurple","mediumblue"))
plot(1:20, col= myPal2(20),pch=19)
```



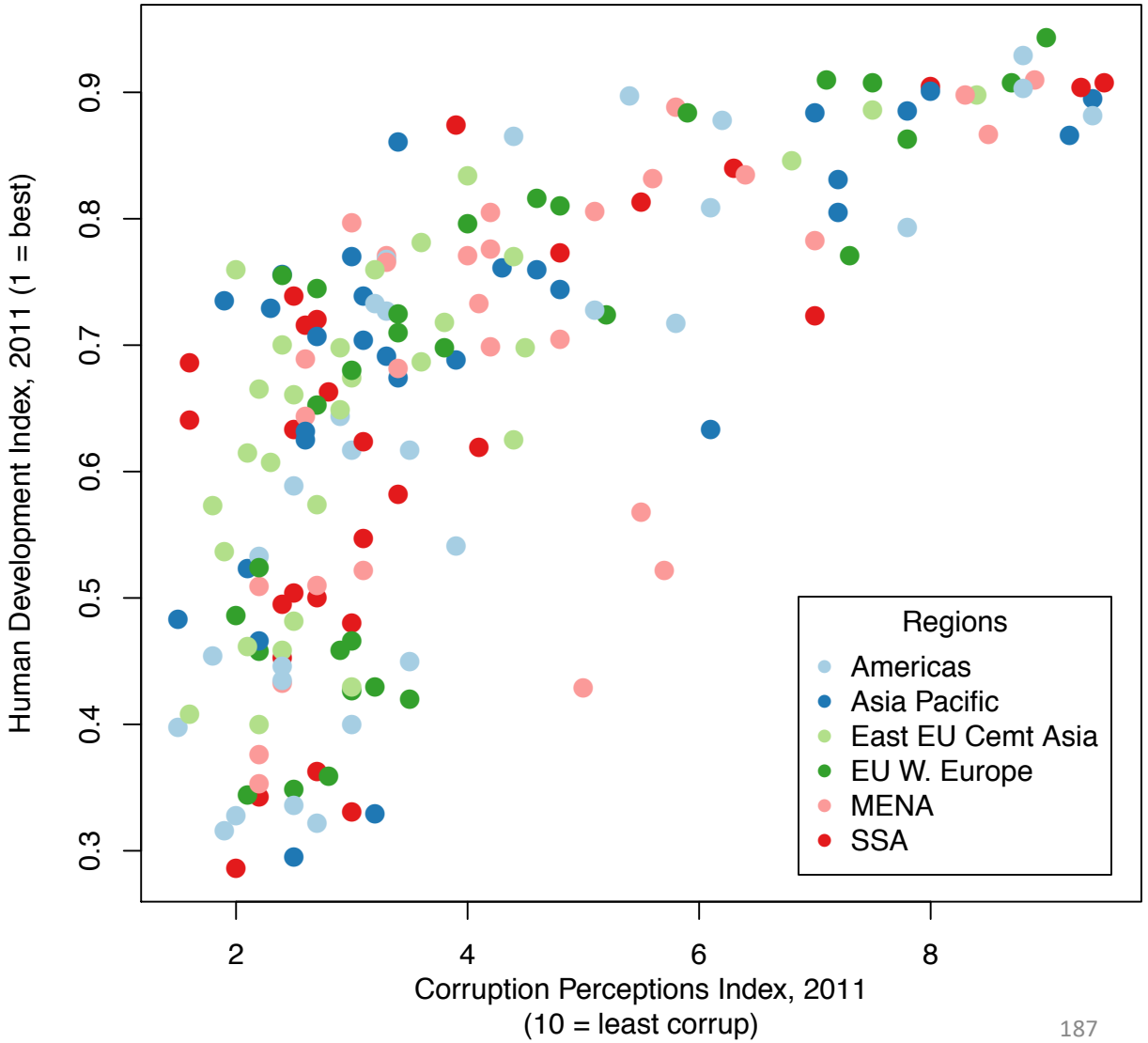
Check out all these options in the tutorial

folder: ~/R_Course_2019/Day1_IntroR/Color/
script: Color.Rmd

Exercise

The goal is using R and RcolorBrewer (choose an appropriate palette) to make this plot using the file

EconomistData.csv



Exercise Solution

```
library(RColorBrewer)
data <- read.csv("EconomistData.csv", row.names = 1, header = T)

plot(data$CPI,
     data$HDI,
     col = brewer.pal(6,"Paired"),
     pch = 16,
     cex = 1.5,
     xlab = "Corruption Perceptions Index, 2011\n(10 = least corrup)",
     ylab = "Human Development Index, 2011 (1 = best)"
)
legend("bottomright", title = "Regions", legend = levels(data$Region),
col = brewer.pal(6,"Paired"), pch = 16)
```