

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



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

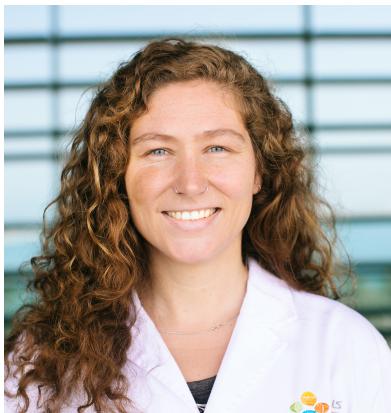
What you will need

- 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...

Instructors



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



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



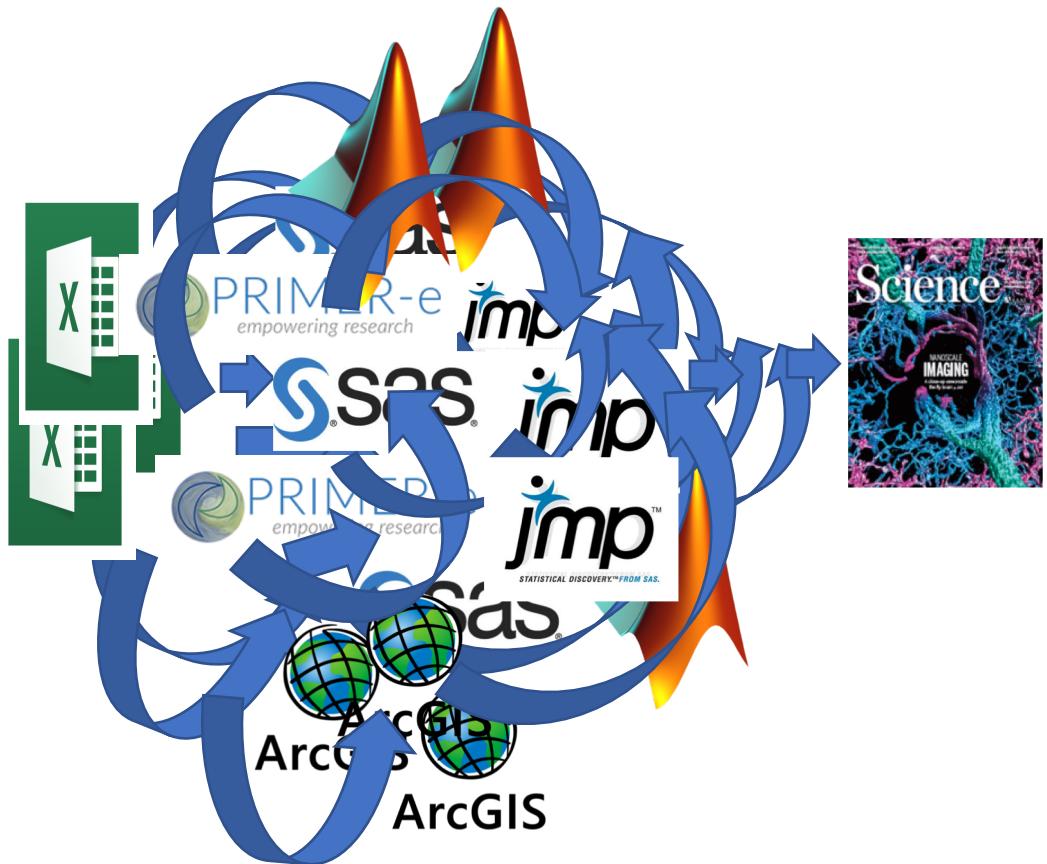
Daffne López-Sandoval
Postdoctoral Fellow
Prof. Susana Agustí
Phytoplankton ecology
daffne.lopezsandoval@kaust.edu.sa

Intro to R

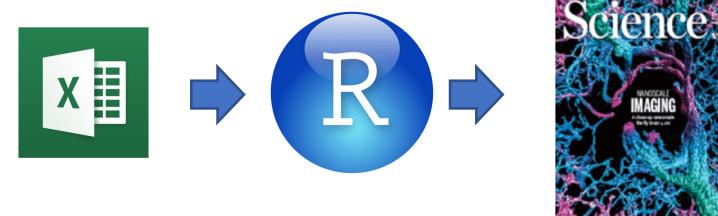
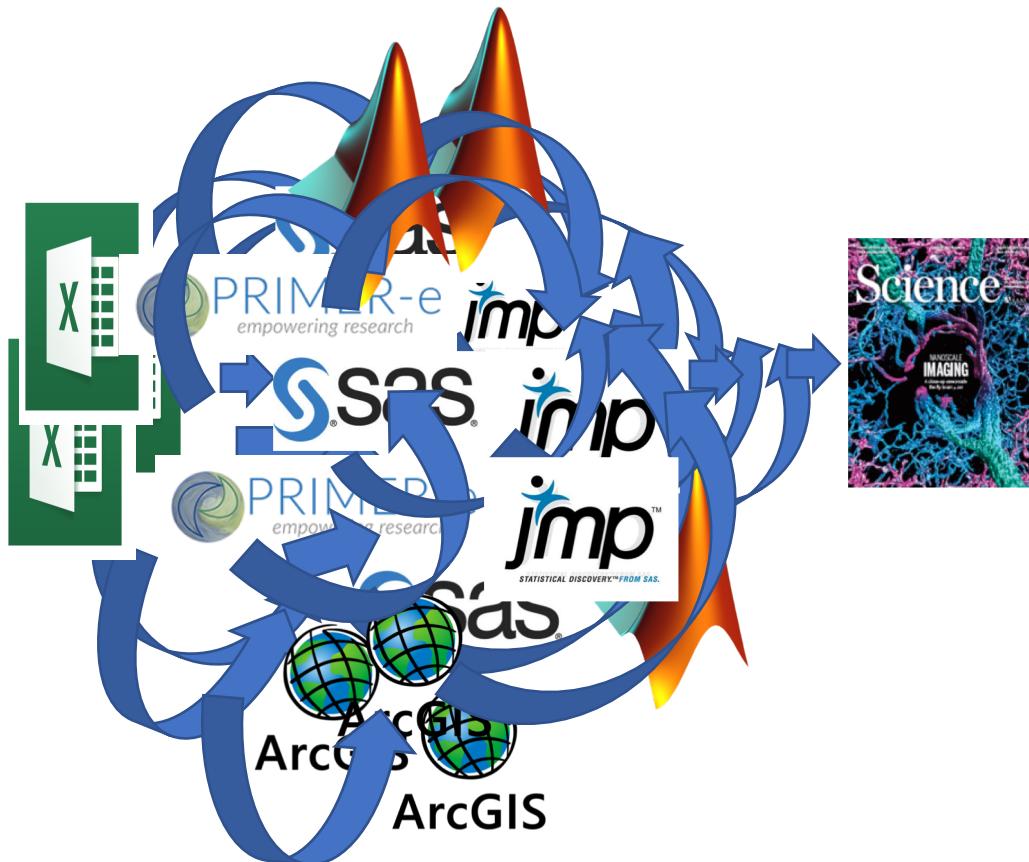
Why R?



Why R?



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?

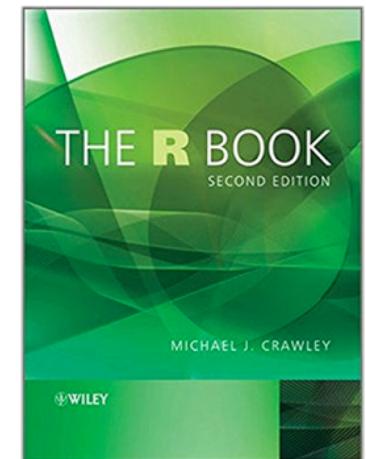
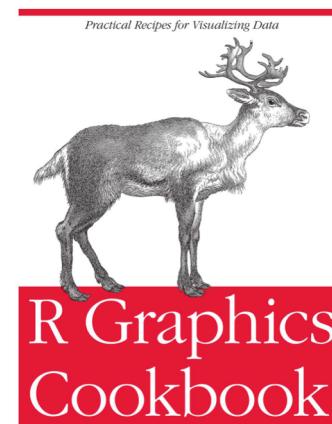
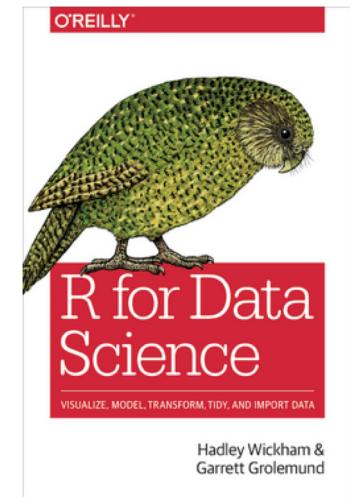
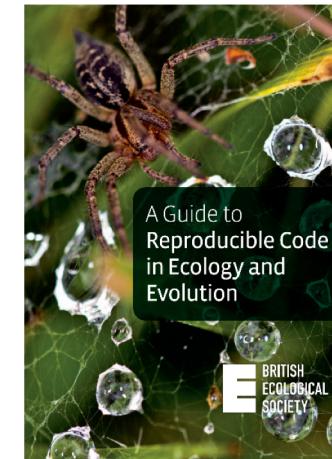
PROS	CONS
<ul style="list-style-type: none">• FREE!• Open source• Very versatile<ul style="list-style-type: none">• data management• stats• plots• maps• Everyone is doing it<ul style="list-style-type: none">• Someone else has had the same or similar problem!• The solution is out there• Reproducibility<ul style="list-style-type: none">• Know what you did (trace your steps)<ul style="list-style-type: none">• Research and publishing takes forever• Publish what you did	<ul style="list-style-type: none">• Steep learning curve• With power comes responsibility<ul style="list-style-type: none">• Know what you are doing• There is no code standard



10

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 cheatsheets
 - <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)



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



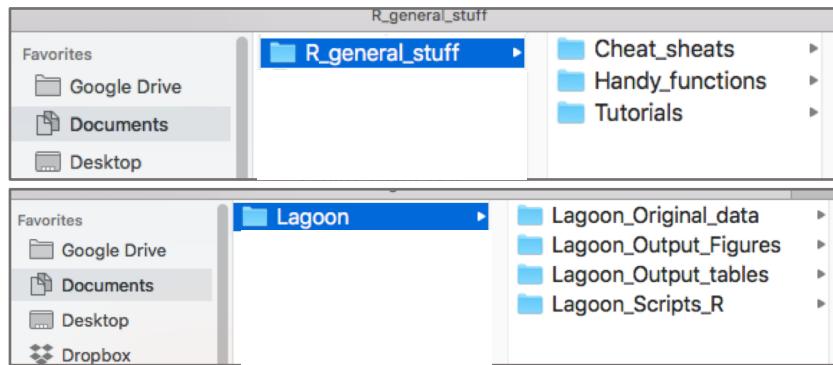
R is dynamic

- 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



R can get messy: Helpful hints

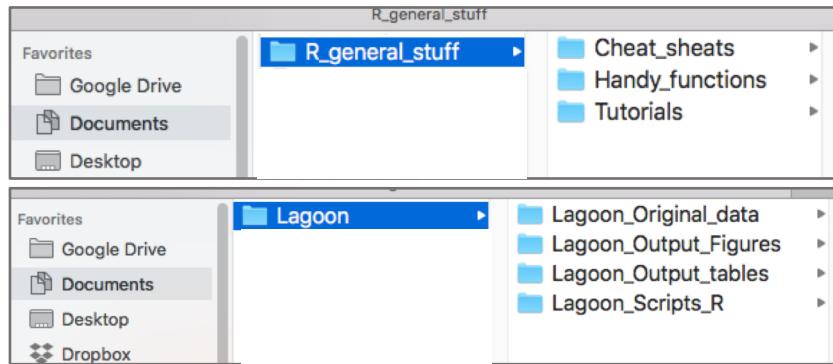
1. Folder setup



- 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

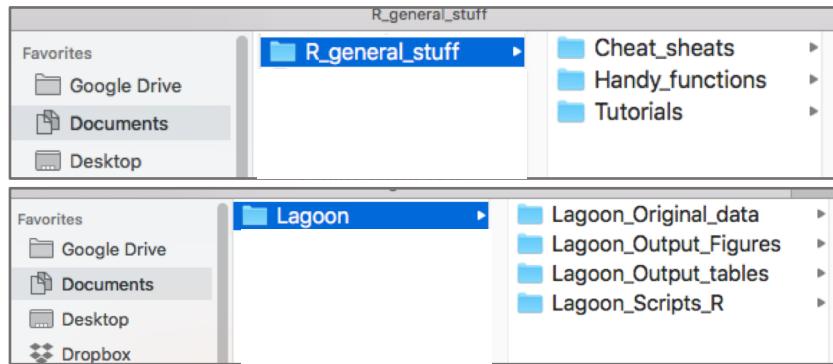
2. Script names

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 from the google Drive link

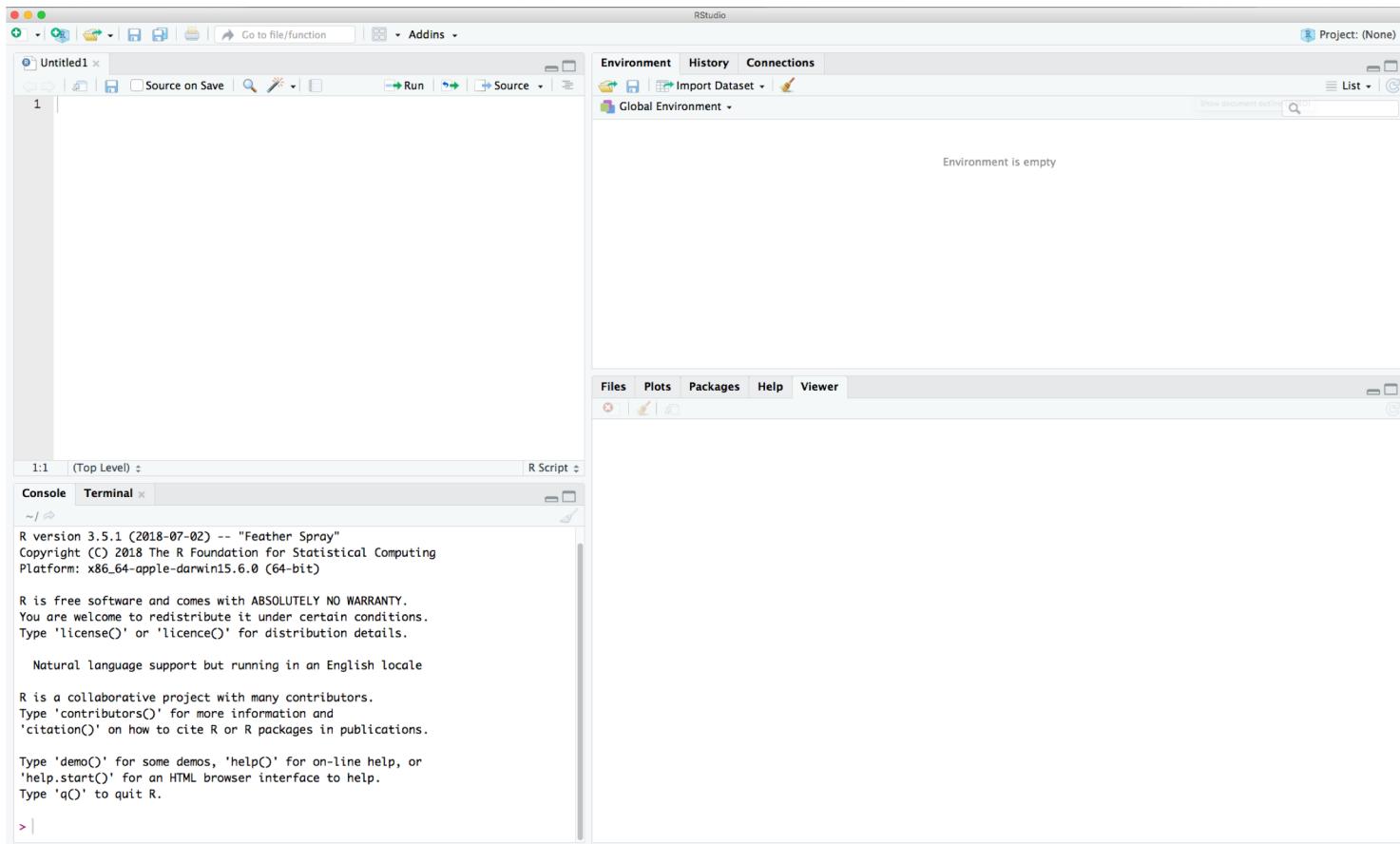
[https://drive.google.com/drive/folders/1iEWUA7s53lhqz8EEybnUPeUYYiq491LP
?usp=sharing](https://drive.google.com/drive/folders/1iEWUA7s53lhqz8EEybnUPeUYYiq491LP?usp=sharing)

- 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

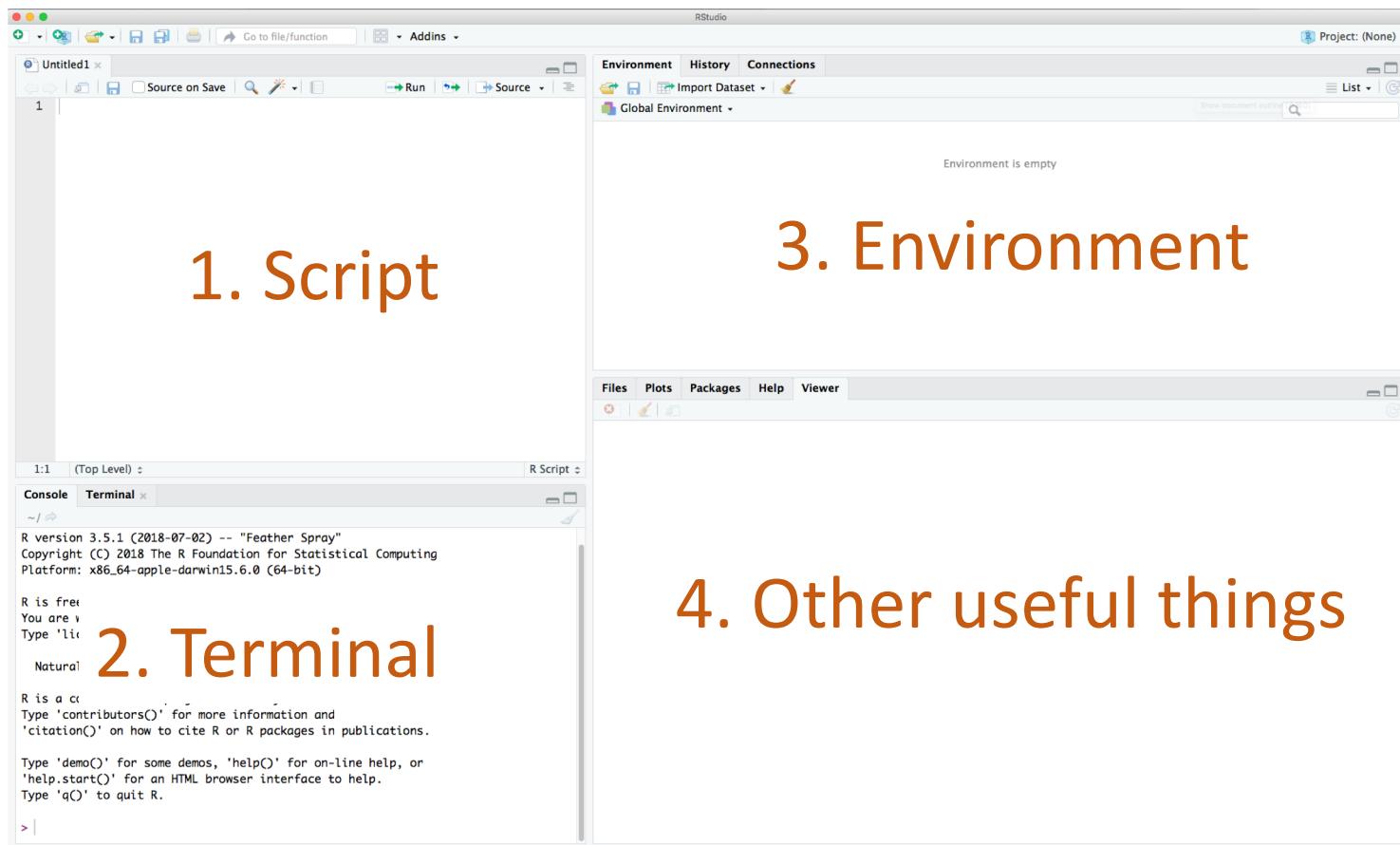
RStudio basics

- More user friendly than plain R



RStudio basics

- More user friendly than plain R



The screenshot shows the RStudio interface. The main window is highlighted with a red border. Inside the main window, the title bar says "Untitled1 x". Below the title bar is a toolbar with icons for file operations like Open, Save, and Print, along with "Go to file/function" and "Addins". The main area contains the text "1" and the heading "1. Script". To the right of the main window is the "Environment" tab of the global environment panel, which displays the message "Environment is empty". Below the main window is the "Console" tab of the terminal panel, showing the R startup message and a prompt "> |".

RStudio

Untitled1 x

Source on Save | Import Dataset | Global Environment

Environment History Connections

Environment is empty

1

1. Script

The main window,
you will type here and send
the commands to the terminal

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)

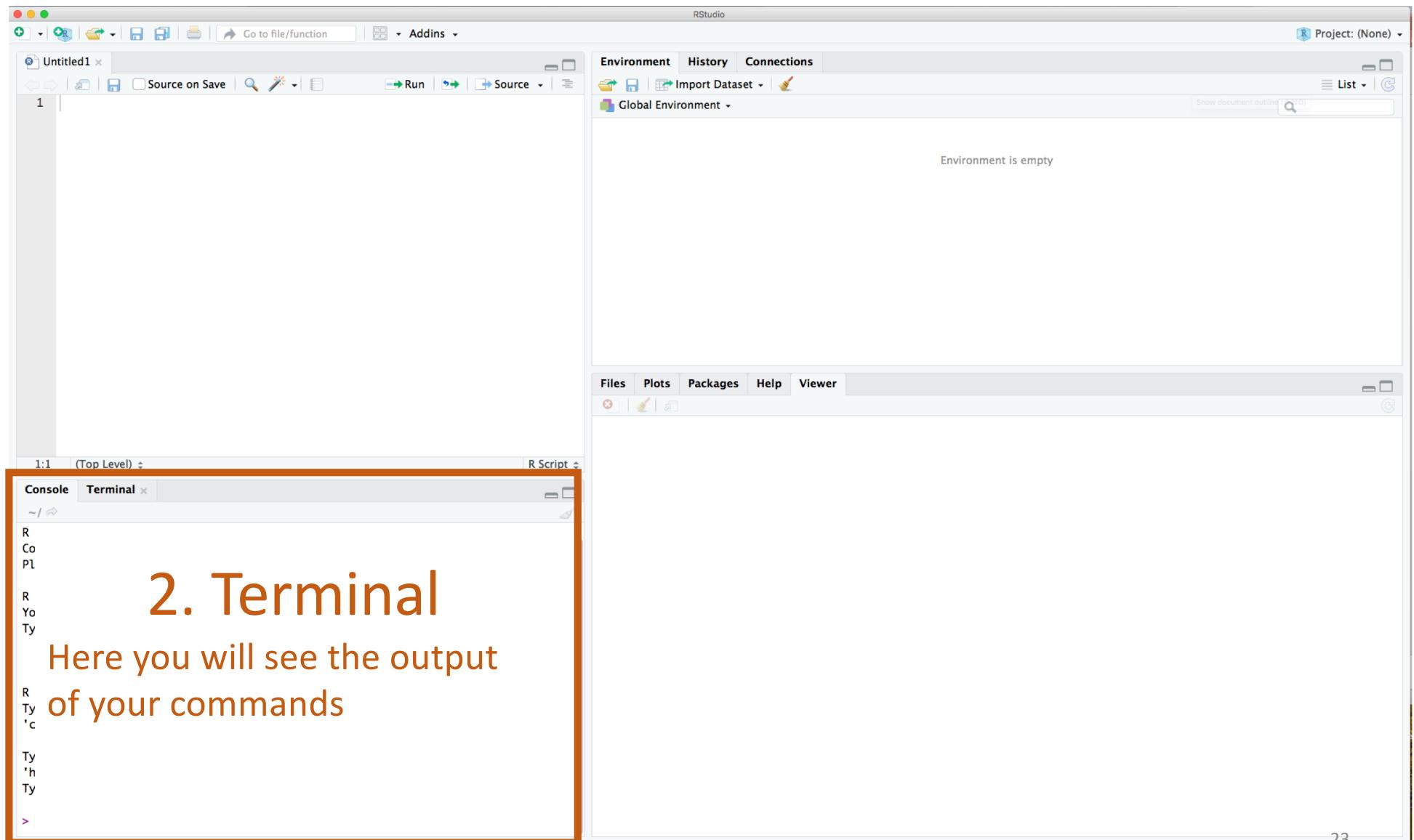
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

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.
```

> |



3. Environment

All your objects from a session stored here.
You can click on them to see them

RStudio

Untitled1

Environment History Connections

Import Dataset

Global Environment

Files Plots Packages Help Viewer

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.

> |

The screenshot shows the RStudio interface. The top menu bar includes 'File', 'Edit', 'View', 'Code', 'Tools', 'Help', and 'Project: (None)'. The left pane contains a file browser with 'Untitled1' selected, and a 'Console' tab showing R startup information. The right pane has tabs for 'Environment', 'History', and 'Connections', with 'Environment' active. The 'Global Environment' dropdown is open, showing the message 'Environment is empty'. A red box highlights the 'Viewer' tab in the bottom right corner.

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 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

The screenshot shows the RStudio interface on a Mac OS X system. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Window, and Help. The status bar at the top right shows the date and time as Sun 1:03 PM and battery level as 41%. The main workspace consists of several panes:

- Code Pane:** A script editor window titled "Untitled1" containing the number "1".
- Environment Pane:** Shows tabs for Environment, History, and Connections. The Environment tab displays the message "Environment is empty".
- Console Pane:** Displays the R startup message:

```
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)
```

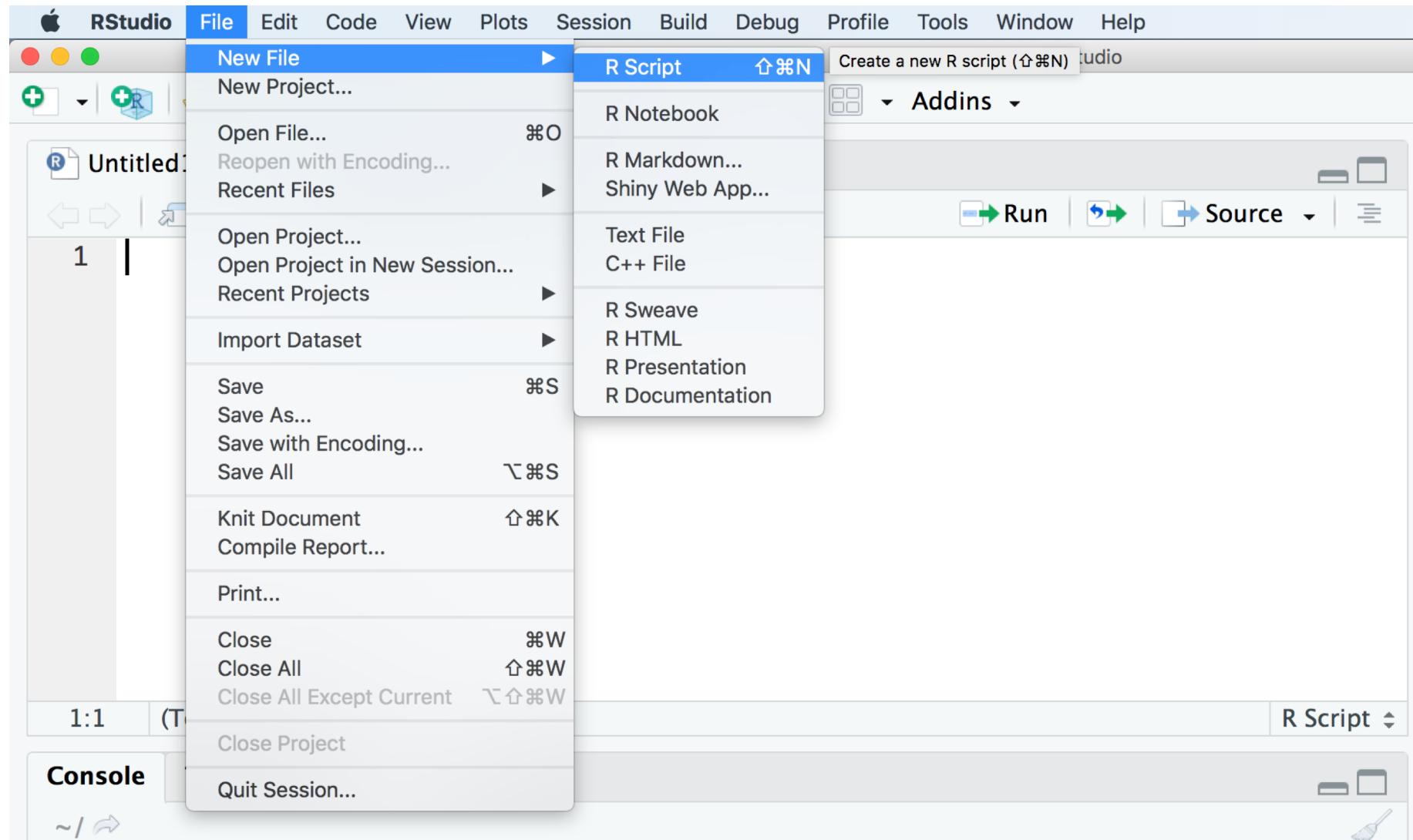
Below this, the standard R license notice is shown:

```
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.
```

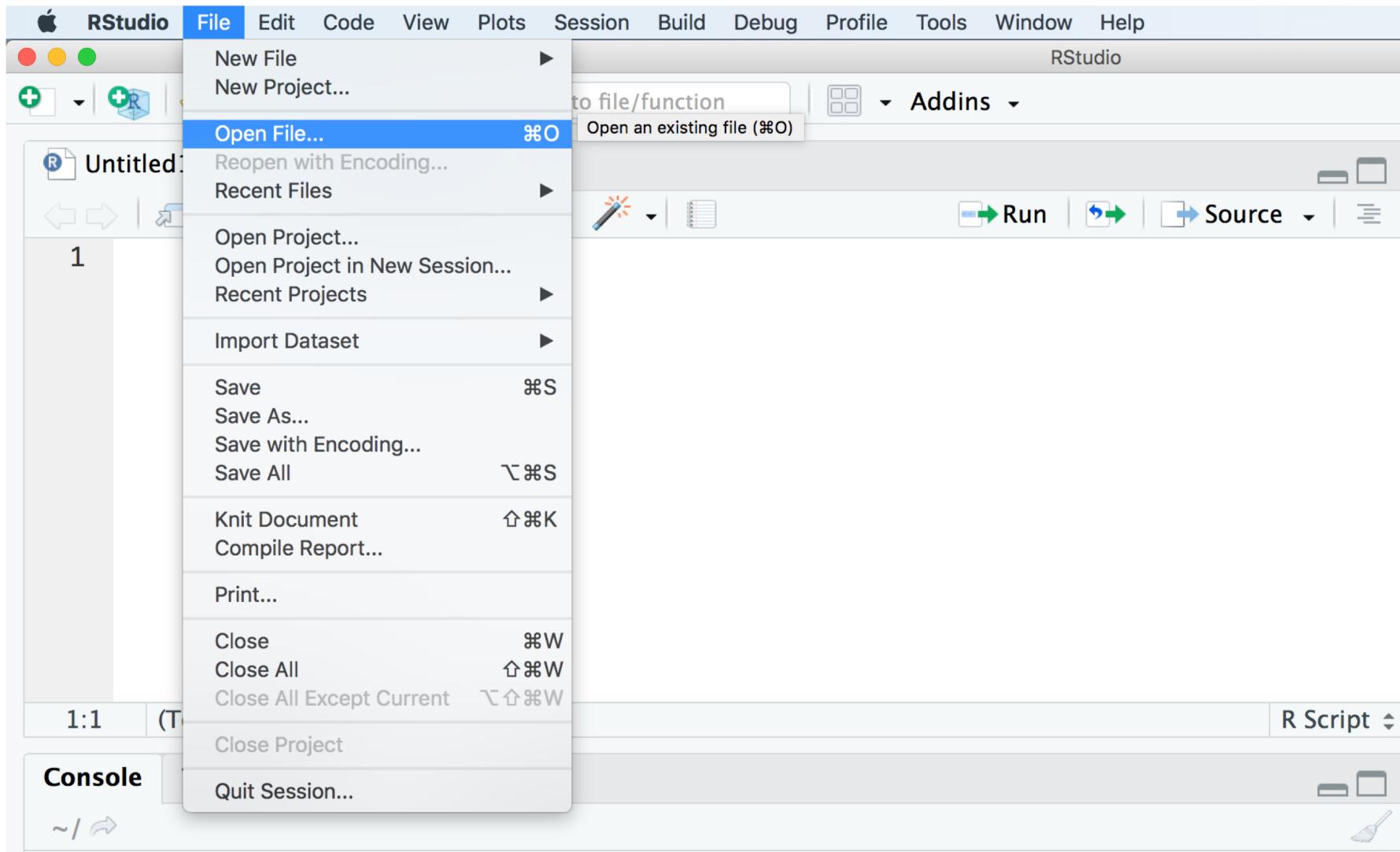
A red oval highlights the entire console output area, and the text "R version and platform" is overlaid in red on the right side of the oval.

- Plots, Packages, and Help Panes:** These are located at the bottom right of the workspace.

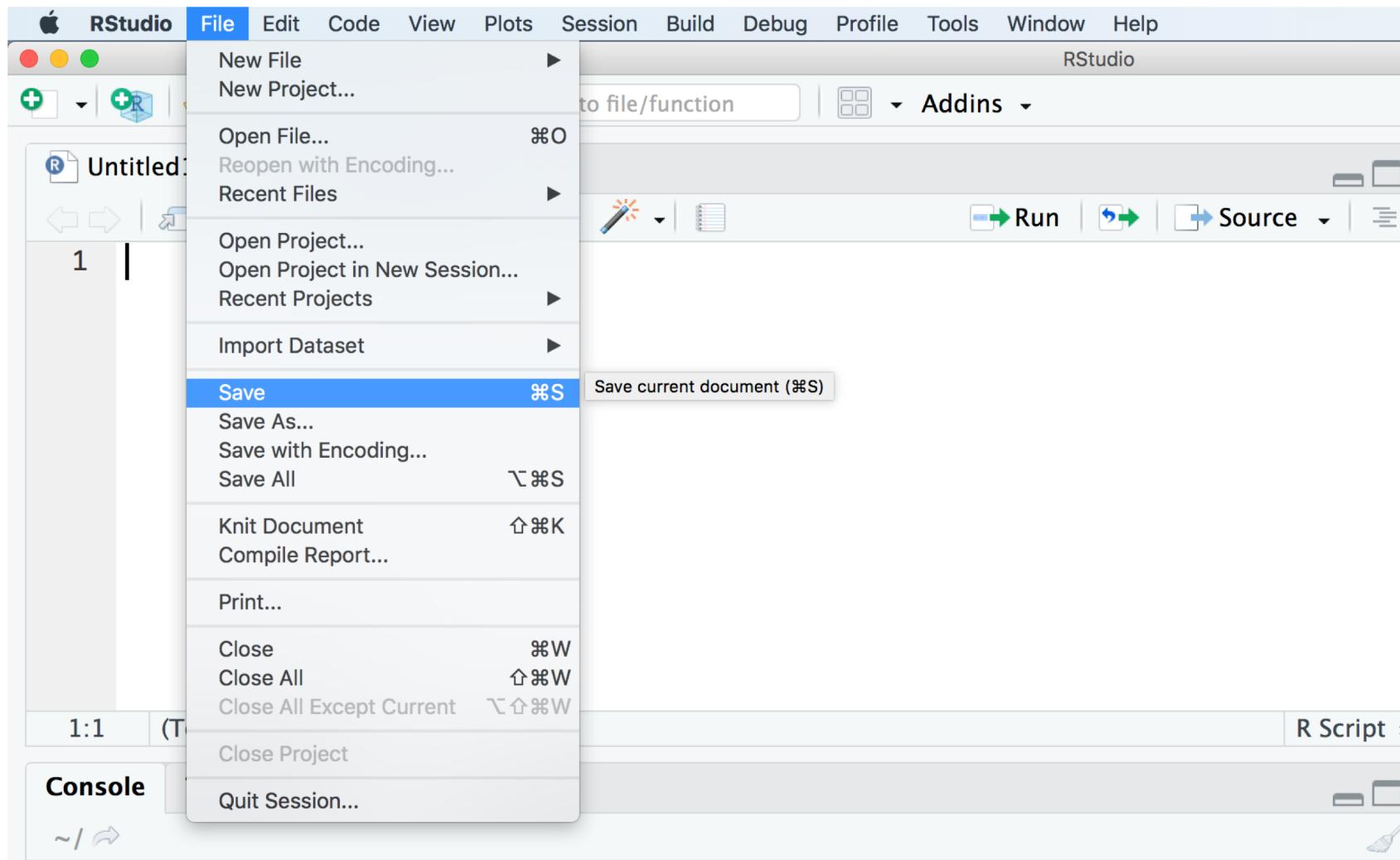
- Create new script



- Open an existing script

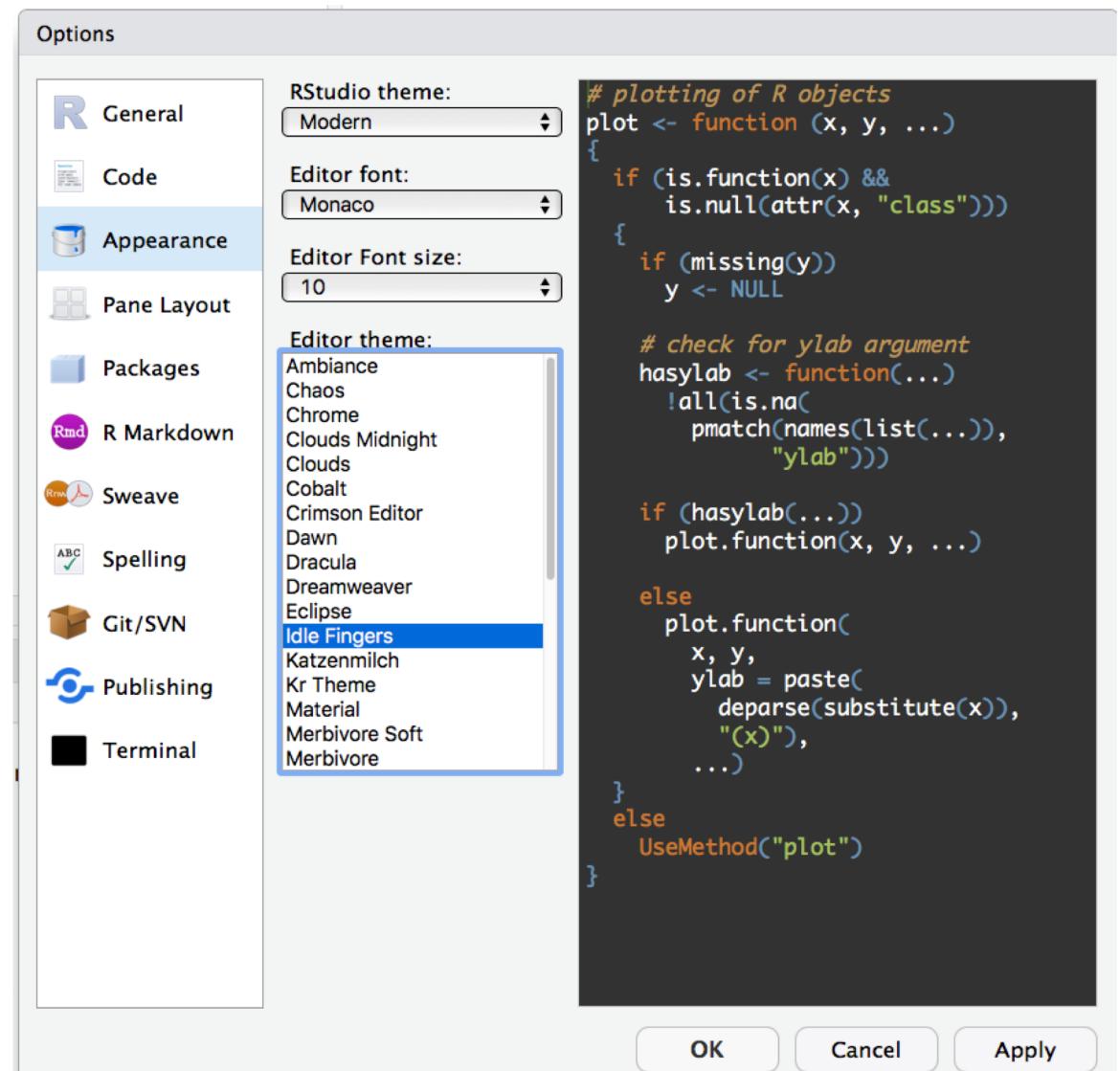


- Save a script



Change RStudio appearance (or how to look cool)

RStudio →
Preferences →
Appearance



Increase font size

Cmd + *Ctrl for Windows

The screenshot shows the RStudio interface. The top menu bar includes 'File', 'Edit', 'Source', 'Run', 'View', 'Tools', 'Help', and 'Addins'. Below the menu is a toolbar with icons for file operations like Open, Save, and Run. The main area has two tabs: 'Untitled1' and 'BasicR-Tamara.R*'. The 'BasicR-Tamara.R*' tab contains R code. The 'Console' tab shows the R startup message and the command-line interface.

```
1 #####  
2 #### Set working directory #####  
3 setwd(dir)  
4 #####  
5 setwd("/Users/huetestm/Google Drive/Data_analysis_in_R_for_Marine_Science/Day1_IntroR_RStudio_Rrmd/BasicRCourse/")  
6 getwd() # get working directory  
7 #####  
8 ##### R as a calculator #####  
9 ##### Exercise 1:  
10 #####  
11 1 / 200 * 30  
12 #[1] 0.15  
13  
14  
15 (59 + 73 + 2) / 3  
16 #[1] 44.7  
17  
18 sin(pi / 2)  
19 #[1] 1  
20  
21  
22  
5:1 (Untitled) :  
R Script
```

Console Terminal
~/Google Drive/Data_analysis_in_R_for_Marine_Science/Day1_IntroR_RStudio_Rrmd/BasicRCourse/
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.

This screenshot shows the same RStudio interface as the left one, but with a larger font size for the code and console output. The code in the 'BasicR-Tamara.R*' tab and the text in the 'Console' tab are both larger and easier to read.

```
1 #####  
2 #### Set working directory #####  
3 #####  
4 setwd("/Users/huetestm/Google Drive/Data_analysis_in_R_for_Marine_Science/Day1_IntroR_RStudio_Rrmd/BasicRCourse")  
5 getwd() # get working directory  
6 #####  
7 ##### R as a calculator #####  
8 #####  
9 ##### Exercise 1:  
10 #####  
11 # (Untitled) :
```

Console Terminal
~/Google Drive/Data_analysis_in_R_for_Marine_Science/Day1_IntroR_RStudio_Rrmd/BasicRCourse/
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.

Cmd -



Decrease font size

A screenshot of the RStudio interface. In the top-left pane, there is an 'Untitled1*' script file containing the code 'getwd()' on line 1. A red arrow points from the word 'getwd()' in the code to a callout box. The callout box has a pink background and contains the text: 'This command tells you where you are on your computer : *get working directory*'. The rest of the RStudio interface is visible, including the toolbar, the Environment tab in the top-right, and the Console tab in the bottom-left.

Untitled1*

```
1 getwd()
```

getwd()

This command tells you where you are on your computer : *get working directory*

Environment is empty

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.

A screenshot of the RStudio interface. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Window, and Help. The status bar shows battery level at 14%, time as Sun 2:32 PM, and a search icon. The main workspace shows an 'Untitled1*' script with the code 'getwd()' and a red arrow pointing to the 'Run' button in the toolbar. A pink callout box contains instructions: 'Send the command to the terminal : Run button Alt + Enter (stays on same line) Cmd/Ctrl + Enter (moves to next line)'. The Environment panel shows 'Environment', 'History', and 'Connections' tabs, with 'Global Environment' selected and a note 'Environment is empty'. The bottom panel shows the 'Console' tab active, displaying R's startup message and help information. The page number 33 is in the bottom right corner.

Send the command to the terminal :

Run button

Alt + Enter (stays on same line)

Cmd/Ctrl + Enter (moves to next line)

Environment is empty

33

The screenshot shows the RStudio interface with the following components:

- Top Bar:** RStudio, File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Window, Help.
- Toolbar:** Go to file/function, Addins.
- Project:** Project: (None)
- Code Editor:** Untitled1* contains the code

```
1 getwd()  
2
```

.
- Environment Tab:** Shows the Global Environment pane with the message "Environment is empty".
- Global Environment:** Shows the Global Environment pane with the message "Environment is empty".
- Console Tab:** Displays the R environment information and the command history:

```
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"  
>
```
- Terminal Tab:** Displays the R environment information and the command history:

```
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"  
>
```
- Message:** A red callout box points to the terminal output with the text "The result shows in your terminal".
- Bottom Right:** Page number 34.

The screenshot shows the RStudio IDE interface. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Window, and Help. The status bar at the top right shows battery level (14%), signal strength, and the time (Sun 2:32 PM). The main window contains several panes:

- Script Editor (Left):** An untitled R script with the following code:

```
1 getwd()  
2
```
- Environment Viewer (Top Right):** Shows tabs for Environment, History, and Connections. The Environment tab displays "Global Environment" with a message: "Environment is empty".
- Console (Bottom Left):** Displays R help messages and a command history:

```
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"  
>
```

A red arrow points from the explanatory text below to the command prompt (>) in the console.
- Help/Viewer (Bottom Right):** Tabs for Files, Plots, Packages, Help, and Viewer.

Explanatory Text (Red Box):

> 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

35

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Window Help

Go to file/function Addins Project: (None)

Untitled1* x

Source on Save Run Source

1 getwd()
2

Your script name is now red *
Means changes have not been saved

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"
>

A screenshot of the RStudio interface. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Window, and Help. The status bar shows battery level at 14%, time as Sun 2:32 PM, and a search icon. The main window has tabs for Untitled1* and Addins. The script editor (Untitled1*) shows two lines of code: 1 getwd() and 2. A red arrow points to the line number 2. A pink callout box contains the text: "Lines begin to be numbered. Easy to track where you are on your script". The environment pane shows "Environment is empty" and the global environment. The files pane shows tabs for Files, Plots, Packages, Help, and Viewer. The bottom console pane shows the R environment and a terminal tab.

1 getwd()
2

Lines begin to be numbered.
Easy to track where you are on your script

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 screenshot shows the RStudio interface with a script file open. The file is named 'BasicR-Tamara.R'. The code in the script is as follows:

```
84  
85  
86  
87 ##### I can make comments on my script using ## ← # At the beginning of  
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
```

Annotations highlight the use of `##` for comments and point to the syntax errors at lines 89 and 90. A red callout box with an arrow points to the first `##` comment at line 87, containing the text: '# At the beginning of a line for comments'.

The screenshot shows the RStudio interface with a script file named "BasicR-Tamara.R". The code contains several syntax errors, indicated by red 'X' icons and wavy red underlines. A red arrow points from the bottom of the "incorrect command" message to one of the error icons.

```
84  
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 | Incorrect command  
95  
96
```

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

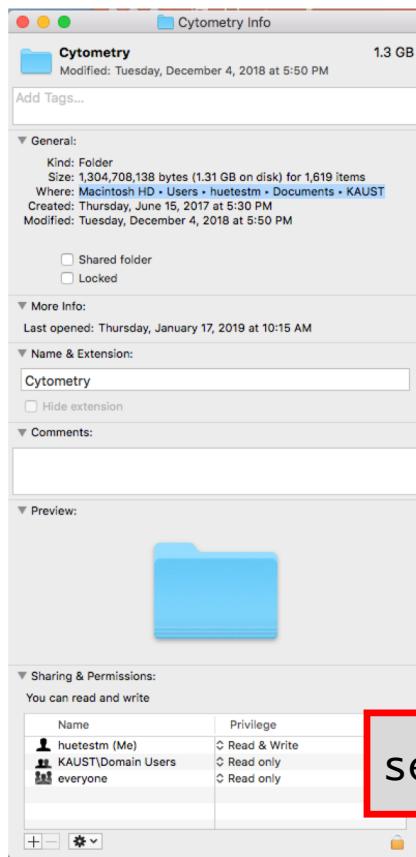
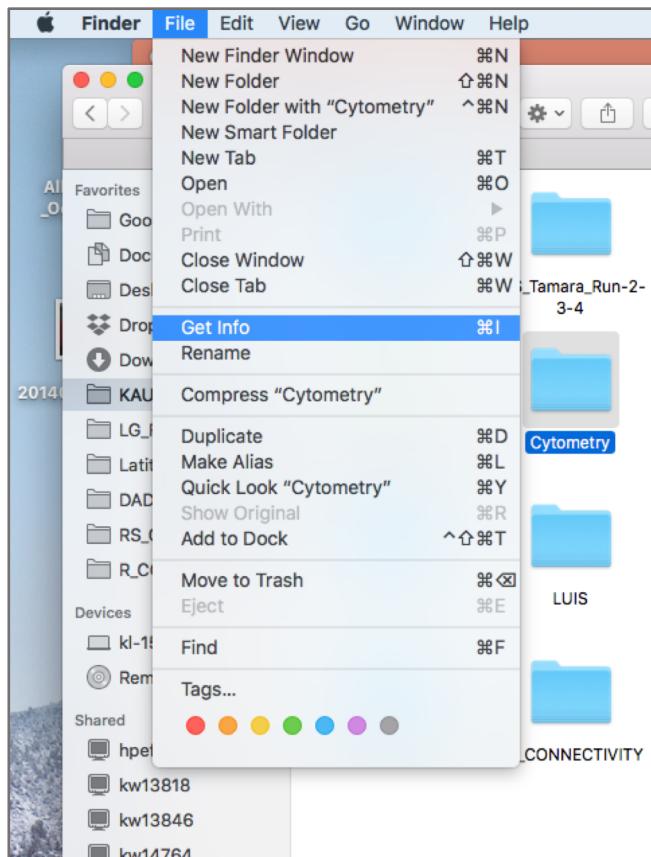
```
setwd("my/path/to/my/folder/")
```

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/")
```

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($\pi/2$)

[1] 1

2^3

2³

[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 its 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 its 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

The screenshot shows the RStudio interface. On the left, there's a script editor window titled "BasicR-Tamara.R*" containing R code. Below it is the "Console" window displaying the R startup message and license information. On the right, the "Environment" tab is active in the top navigation bar, showing an "Empty Environment". A red arrow points from the text "1. Find Packages tab" to the "Packages" tab in the navigation bar of the main panel, which is also highlighted with a red circle. The main panel displays a table of packages in the "System Library".

1. Find Packages tab

Name	Description	Version
abc	Tools for Approximate Bayesian Computation (ABC)	2.1
abc.data	Data Only: Tools for Approximate Bayesian Computation (ABC)	1.0
abind	Combine Multidimensional Arrays	1.4-5
acepack	ACE and AVAS for Selecting Multiple Regression Transformations	1.4.1
acss.data	Data Only: Algorithmic Complexity of Short Strings (Computed via Coding Theorem Method)	1.0
actuar	Actuarial Functions and Heavy Tailed Distributions	2.3-1
ada	The R Package Ada for Stochastic Boosting	2.0-5
additivityTests	Additivity Tests in the Two Way Anova with Simultaneous Inference	1.1-4

*Alternative for CRAN with R Studio

The screenshot shows the RStudio interface. On the left, the Script Editor contains a script named 'BasicR-Tamara.R' with some R code. The Console window displays the R startup message and license information. In the center, the Environment tab of the Global Environment pane shows an empty environment. A red box highlights the 'Packages' tab in the bottom navigation bar, and a red arrow points from this box to the 'Install' button in the Packages pane. The Packages pane lists several packages in the System Library, with the 'Install' button circled in red.

1. Find Packages tab
2. Click on Install

Name	Description	Version
abc	Tools for Approximate Bayesian Computation (ABC)	2.1
abc.data	Data Only: Tools for Approximate Bayesian Computation (ABC)	1.0
abind	Combine Multidimensional Arrays	1.4-5
acepack	ACE and AVAS for Selecting Multiple Regression Transformations	1.4.1
acss.data	Data Only: Algorithmic Complexity of Short Strings (Computed via Coding Theorem Method)	1.0
actuar	Actuarial Functions and Heavy Tailed Distributions	2.3-1
ada	The R Package Ada for Stochastic Boosting	2.0-5
additivityTests	Additivity Tests in the Two Way Anova with Simultaneous Inference	1.1-4

*Alternative for CRAN with R Studio

The screenshot shows the RStudio interface with the 'BasicR-Tamara.R*' file open in the editor. A modal dialog box titled 'Install Packages' is displayed. The 'Install from:' dropdown is set to 'Package Archive File (.tgz; .tar.gz)'. The 'Package archive:' field contains an empty text input and a 'Browse...' button. The 'Install to Library:' field is set to '/Library/Frameworks/R.framework/Versions/3.5/Resources/library'. At the bottom of the dialog are 'Install' and 'Cancel' buttons. The background shows the RStudio environment tab bar with 'Environment', 'History', and 'Connections' selected.

1. Find Packages tab
2. Click on Install
3. Select CRAN (repository) or local file

Package	Version	Action
approximateBayesian	2.1	X
Tools for Approximate Bayesian	1.0	X
multidimensionalArrays	1.4-5	X
AS for Selecting Multiple	1.4.1	X
Transformations		
Algorithmic Complexity of Short	1.0	X
Computations via Coding Theorem		
actuar	2.3-1	X
ada	2.0-5	X
additivityTests	1.1-4	X

*Alternative for CRAN with R Studio

The screenshot shows the RStudio interface with the 'BasicR-Tamara.R*' file open in the editor. A modal dialog box titled 'Install Packages' is displayed. The 'Install from:' dropdown menu has 'Repository (CRAN)' selected. The 'Install to Library:' dropdown shows the path '/Library/Frameworks/R.framework/Versions/3.5/Resources/library'. The 'Install dependencies' checkbox is checked. At the bottom of the dialog are 'Install' and 'Cancel' buttons. Below the dialog, a list of packages is visible, including 'ada', 'additivityTests', 'The R Package Ada for Stochastic Boosting', and 'Additivity Tests in the Two Way Anova with...'. The right side of the screen shows the RStudio environment, history, and connections tabs.

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



The screenshot shows the RStudio environment with the 'BasicR-Tamara.R*' script open in the top-left pane, containing code related to package loading and usage. The bottom-left pane shows the 'Console' output, which includes messages about attaching the 'SparseM' package and loading required packages like MASS and locfit. The right side of the interface features the 'Packages' tab, which lists the 'System Library' with various R packages. One package, 'abc', is highlighted with a red arrow pointing to its entry in the list.

Name	Description	Version
abc	Tools for Approximate Bayesian Computation (ABC)	2.1
abc.data	Data Only: Tools for Approximate Bayesian Computation (ABC)	1.0
abind	Combine Multidimensional Arrays	1.4-5
acepack	ACE and AVAS for Selecting Multiple Regression Transformations	1.4.1
acss.data	Data Only: Algorithmic Complexity of Short Strings (Computed via Coding Theorem Method)	1.0
actuar	Actuarial Functions and Heavy Tailed Distributions	2.3-1
ada	The R Package Ada for Stochastic Boosting	2.0-5
additivityTests	Additivity Tests in the Two Way Anova with	1.1-4

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

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)`

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

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

Exercise 3. Read csv file

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

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

this command opens a csv file

this is the name of the file we want to open

```
head(got)
```

Check the first 6 lines of the object *got*

```
str(got)
```

Check the structure of the object *got*

4 Columns

```
Console Terminal x
~/Google Drive/Data_analysis_in_R_for_Marine_Science/Day1_ IntroR_RStudio_Rrmd/BasicRCourse/ ↗
> head(got)
      Name           Role Dead_S1 Age_actor
1 Emilia_Clarke Daenerys_Tragaryen FALSE    32
2 Kit_Harington        Jon_Snow FALSE    32
3 Maisie_Williams       Arya_Stark FALSE    21
4 Peter_Dinklage     Tyrion_Lannister FALSE    49
5 Jason_Momoa          Khal_Drogo TRUE     39
6 Sean_Bean            Ned_Stark  TRUE     59
> str(got)
'data.frame': 6 obs. of 4 variables:
 $ Name    : Factor w/ 6 levels "Emilia_Clarke",...: 1 3 4 5 2 6
 $ Role    : Factor w/ 6 levels "Arya_Stark","Daenerys_Tragaryen",...: 2 3 1 6 4 5
 $ Dead_S1 : logi  FALSE FALSE FALSE FALSE TRUE TRUE
 $ Age_actor: int  32 32 21 49 39 59
>
```

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)

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

- **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)
```

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

- **character** (anything with letters)

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

Character/name vectors always go with ""

- **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)

```
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)
```

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

- **character** (anything with letters)

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

Character/name vectors always go with ""

- **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
```

6. Data classes

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

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

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

- **character** (anything with letters)

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

Character/name vectors always go with ""

- **logical** (TRUE, FALSE)

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

class() will tell you the class of your data

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

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

```
levels(e)
```

```
> fall summer winter
```

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_Targaryen	FALSE	32
2	Kit_Harington	Jon_Snow	FALSE	32
3	Maisie_Williams	Arya_Stark	FALSE	21
4	Peter_Dinklage	Tyrion_Lannister	FALSE	49
5	Jason_Momoa	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_Targaryen	FALSE	32
2	Kit	Character	Character	Numeric
3	Maisie_Williams	Arya_Stark	TRUE	11
4	Peter_Dinklage	Tyrion_Lannister	FALSE	49
5	Jason_Momoa	Khal_Drogo	TRUE	39
6	Sean_Bean	Ned_Stark	TRUE	59

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	
Array	2 or more dimensions	Data types: numeric, character logic, factor
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
Matrix	2 dimensions	Data types: numeric, character logic, factor
Array	2 or more dimensions	numeric, character logic, factor
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_Targaryen	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_Targaryen	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 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 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
```

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

3. Create a matrix

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)}
```

name of function

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)}
```

name of function

Note the use of {}

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

argument: the temperature in Fahrenheit

Create your own functions

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

name of function

```
fahrenheit_to_celsius <- function(temp_F) {  
  temp_C <- ((temp_F - 32) * (5 / 9)) ← calculate the temperature in Celsius  
  return(temp_C)  
}
```

Note the use of {}

argument: the temperature in Fahrenheit

Create your own functions

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

Note the use of {}

```
fahrenheit_to_celsius <- function(temp_F) {  
  temp_C <- ((temp_F - 32) * (5 / 9)) ← calculate the temperature in Celsius  
  return(temp_C) ← return the temperature in Celsius  
}  
  
argument: the temperature in Fahrenheit
```

Create your own functions

```
function(arg1, arg2) {  
  do something with arg1 and arg2  
  return(result)}  
  
name of function  
  
fahrenheit_to_celsius <- function(temp_F) {  
  temp_C <- ((temp_F - 32) * (5 / 9)) ← calculate the temperature in Celsius  
  return(temp_C) ← return the temperature in Celsius  
}  
  
fahrenheit_to_celsius(300)  
[1] 148.8889
```

Note the use of {}

argument: the temperature in Fahrenheit

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

```
fahrenheit_to_celsius <- function(temp_F) {  
  temp_C <- ((temp_F - 32) * (5 / 9)) ← calculate the temperature in Celsius  
  return(temp_C) ← return the temperature in Celsius  
}
```

argument: the temperature in Fahrenheit

```
fahrenheit_to_celsius(300)
```

```
[1] 148.8889
```

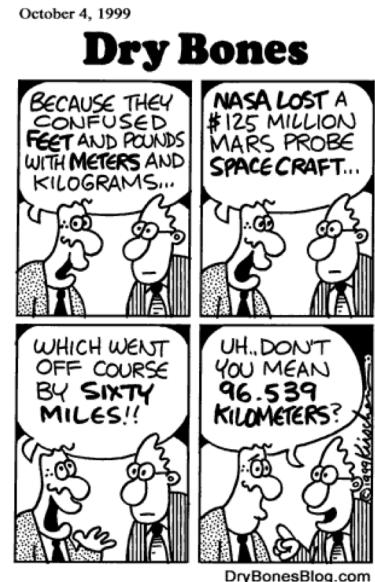
```
fahrenheit_to_celsius(c(300,20,100,800))
```

```
[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
($1 \text{ Lb f/s} = 1.36 \text{ N m/s}$)

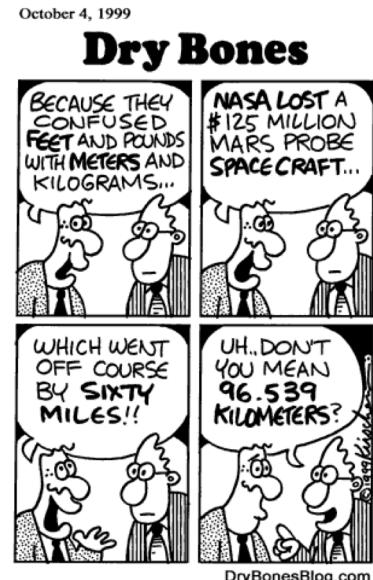


Exercise 9: make a function

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)  
}
```



9. Add columns to a dataframe

	ColA	ColB	ColC	ColD
R1				
R2				
R3				
R4				
R5				

mydata

+

	ColE
R1	
R2	
R3	
R4	
R5	

newcolumn

=

	ColA	ColB	ColC	ColD	ColE
R1					
R2					
R3					
R4					
R5					

newdata

9. Add columns to a dataframe

	ColA	ColB	ColC	ColD
R1				
R2				
R3				
R4				
R5				

mydata

	ColE
R1	
R2	
R3	
R4	
R5	

newcolumn

	ColA	ColB	ColC	ColD	ColE
R1					
R2					
R3					
R4					
R5					

newdata

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

name we give our new dataframe object
cbind = column bind

new column we want to add
dataframe to which we want to add the column

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

9. Add columns to a dataframe

The diagram illustrates the process of adding a new column to an existing dataframe. On the left, a 5x4 grid labeled "mydata" contains four columns (ColA, ColB, ColC, ColD) and five rows (R1 to R5), all colored green. In the center, a vertical stack of five orange boxes labeled "newcolumn" contains one column (ColE) and five rows (R1 to R5). To the right, an equals sign separates the inputs from the result. The result, labeled "newdata", is a 5x5 grid containing the original four columns of "mydata" followed by the new column "ColE", which is colored orange. The rows are labeled R1 through R5.

	ColA	ColB	ColC	ColD
R1				
R2				
R3				
R4				
R5				

+

	ColE
R1	
R2	
R3	
R4	
R5	

=

	ColA	ColB	ColC	ColD	ColE
R1					
R2					
R3					
R4					
R5					

newdata

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

name we give our new dataframe object
cbind = column bind

new column we want to add
dataframe to which we want to add the column

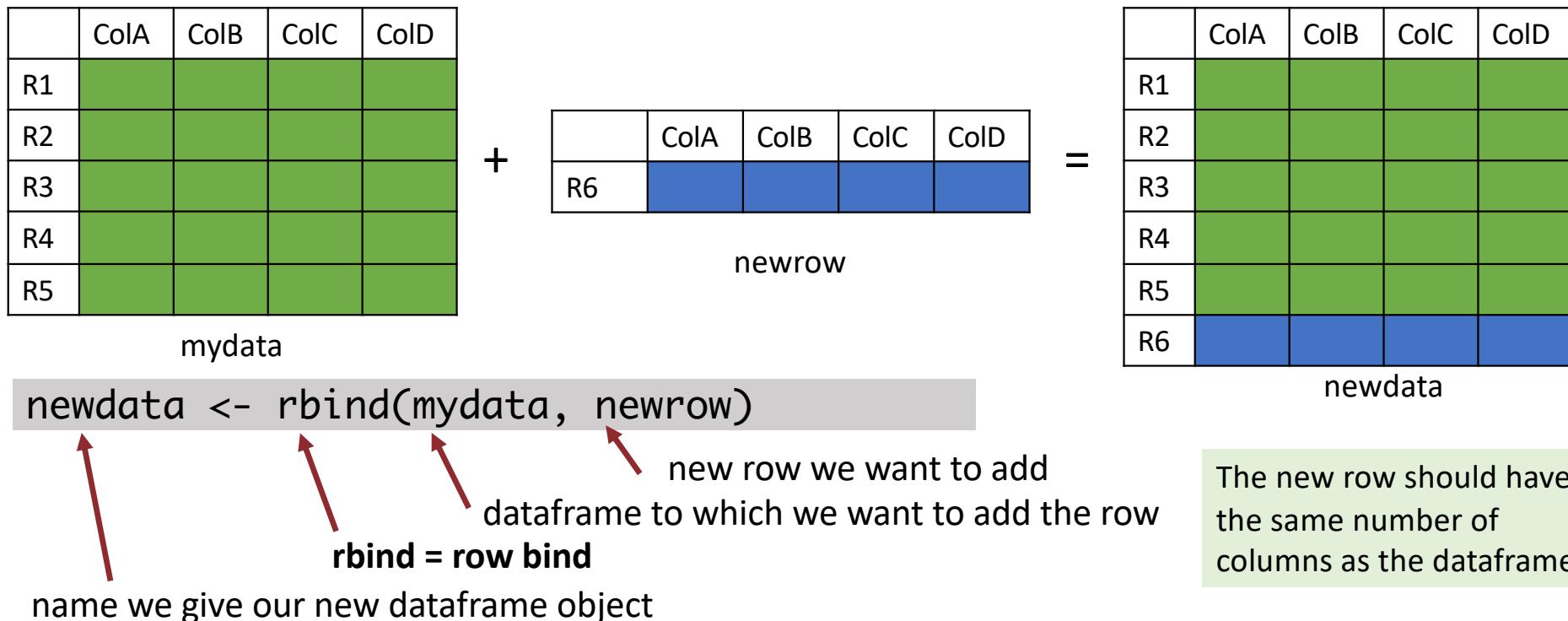
The new column should have the same
number of rows as the dataframe

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

	C1	C2	C3	C4	
	Name	Role	Dead_S1	Age_actor	
got =	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

	C1	C2	C3	C4	C5
	Name	Role	Dead_S1	Age_actor	Gender
gotGender =	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	Male

10. Add rows to a dataframe



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

	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

	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!!!!



KeepCalmAndPosters.com

125

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

Access data in a dataframe:

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

11. Subset a dataframe

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

by index = `got[,2]`

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

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

All the rows
in column 2

11. Subset a dataframe

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

by index = `got[,2]`

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

by value in a column = `subset(got, Gender == "Female")`

`got[got$Gender == "Female",]`

`filter(got, Gender == "Female")`

* tidyverse

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

All the rows
in column 2

All rows with
the females

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. Typically **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()
```

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

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

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()
```

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

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

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
<code>read.csv()</code>	<code>write.csv()</code>	csv file
<code>read.table()</code>	<code>write.table()</code>	txt file
<code>load()</code>	<code>save()</code>	1 or several objects as Rdata
<code>load()</code>	<code>save.image()</code>	Workspace as Rdata
<code>readRDS()</code>	<code>saveRDS</code>	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 assignment table (or function annotation for metagenomes/transcriptomes). The first column will also be the OTU name (this has to match the OTU table) and the columns will be the taxonomic assignments
- 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.

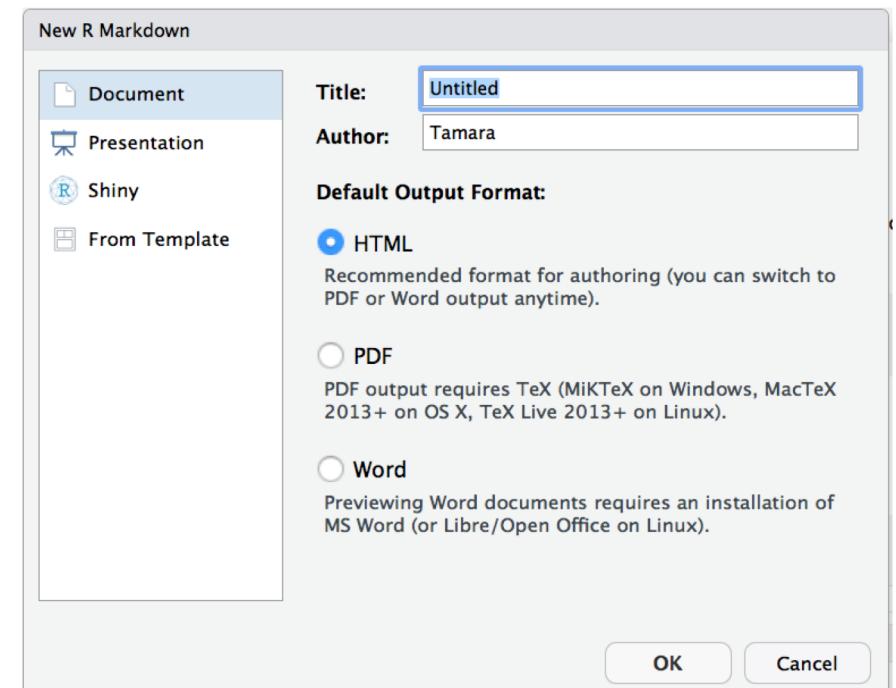
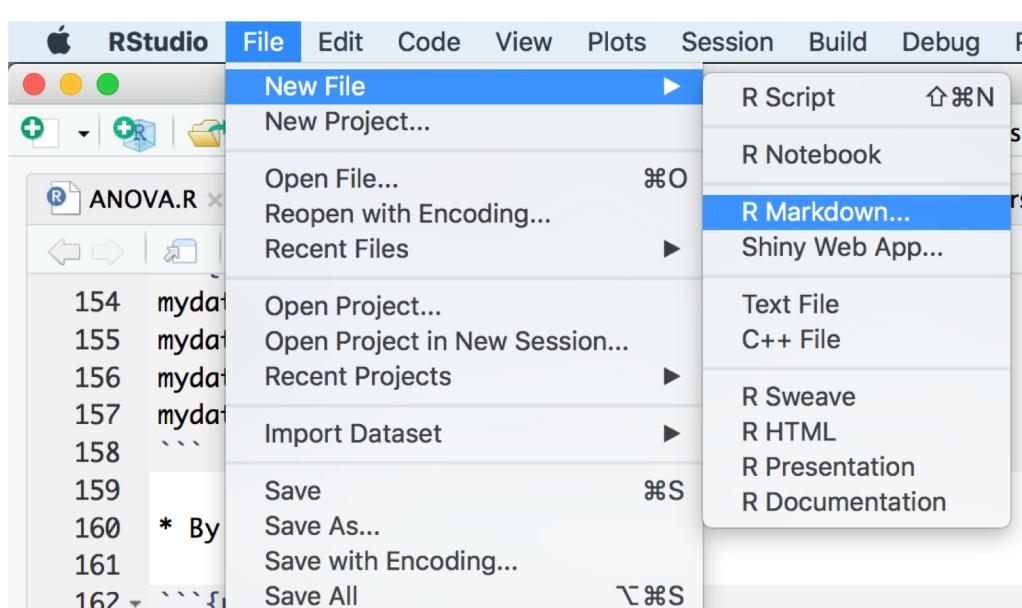
1. LOADING AND PREPARING DATA

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/>

The screenshot shows the RStudio interface with the following components:

- Top Bar:** RStudio, File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Window, Help.
- Toolbar:** Addins, Go to file/function, Insert, Run, Knit, Environment, History, Connections, Import Dataset, List, Global Environment, Environment is empty, Files, Plots, Packages, Help, Viewer.
- Code Editor:** Untitled2.Rmd, displaying R Markdown code. The code includes a YAML header, setup code for knitr, and two code chunks. The second code chunk uses the `cars` dataset.
- Console:** Shows the R startup message and a prompt (>).

The screenshot shows the RStudio interface with an R Markdown document open in the left pane. A red callout box highlights the text "Space in white will appear as text".

```
1 ---  
2 title: "Untitled"  
3 author: "Tamara"  
4 date: "1/22/2019"  
5 output: html_document  
6 ---  
7  
8 ```{r setup, include=FALSE}  
9 knitr::opts_chunk$set(echo = TRUE)  
10 ```  
11  
12 ## R Markdown  
13  
14 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>.  
15  
16 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:  
17  
18 ```{r cars}  
19 summary(cars)  
2:1 # Untitled
```

Environment is empty

Files Plots Packages Help Viewer

152

The screenshot shows the RStudio interface with an R Markdown document open. The code editor displays the following R Markdown code:

```
1 ---  
2 title: "Untitled"  
3 author: "Tamara"  
4 date: "1/22/2019"  
5 output: html_document  
6 ---  
7  
8 ```{r setup, include=FALSE}  
9 knitr::opts_chunk$set(echo = TRUE)  
10 ...  
11  
12 ## R Markdown  
13  
14 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>.  
15  
16 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:  
17  
18 ```{r cars}  
19 summary(cars)  
2:1 # Untitled
```

The rendered output in the preview pane shows two pink boxes with text: "Space in white will appear as text" and "Space in gray is code". The RStudio interface also includes the Environment, History, and Connections panes, and a Console/Terminal pane at the bottom.

A screenshot of the RStudio interface. The main window shows an R Markdown document titled "Untitled2". The code chunk at the top is:

```
1 ---  
2 title: "Untitled"  
3 author: "Tamara"  
4 date: "1/22/2019"  
5 output: html_document  
6 ---  
7  
8 ```{r setup, include=FALSE}  
9 knitr::opts_chunk$set(echo = TRUE)  
10 ```  
11  
12 ## R Markdown  
13  
14 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>.  
15  
16 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:  
17  
18 ```{r cars}  
19 summary(cars)  
2:1 # Untitled
```

The "Insert" button in the toolbar is circled in red, and a pink callout bubble points to it with the text "Insert pieces of R code".

The RStudio interface includes the following panels:

- Top Bar:** RStudio, File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Window, Help.
- Toolbar:** Addins, Go to file/function, Knit, Insert (circled), Run, Environment, History, Connections.
- Left Panel:** Global Environment (empty).
- Right Panel:** Files, Plots, Packages, Help, Viewer.
- Bottom Panels:** Console (showing R help text) and Terminal.

A screenshot of the RStudio interface. The main window shows an R Markdown file named "Untitled2". The code includes a YAML header and several R code chunks. A pink callout box with the text "Press Knit to see how the file looks" points to the "Knit" button in the toolbar above the editor. The RStudio environment is visible on the right, showing the Global Environment and other panels.

Press Knit to see how the file looks

```
1 ---  
2 title: "Untitled"  
3 author: "Tamara"  
4 date: "1/22/2019"  
5 output: html_document  
6 ---  
7  
8 ```{r setup, include=FALSE}  
9 knitr::opts_chunk$set(echo = TRUE)  
10 ```  
11  
12 ## R Markdown  
13  
14 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>.  
15  
16 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:  
17  
18 ```{r cars}  
19 summary(cars)  
2:1 # Untitled
```

Press the green arrow to run the selected chunk of code

A screenshot of the RStudio interface. On the left, the code editor shows an R Markdown file named 'Untitled2'. A red arrow points from a pink callout box containing the text 'Press the green arrow to run the selected chunk of code' to the green 'Run' button in the toolbar above the code editor. The 'Run' button is highlighted with a red circle. The code editor contains several chunks of R code and text. The right side of the interface includes the Environment, History, and Connections panes, and the Global Environment pane which states 'Environment is empty'. At the bottom, the Console and Terminal panes are visible.

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 customizable 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**

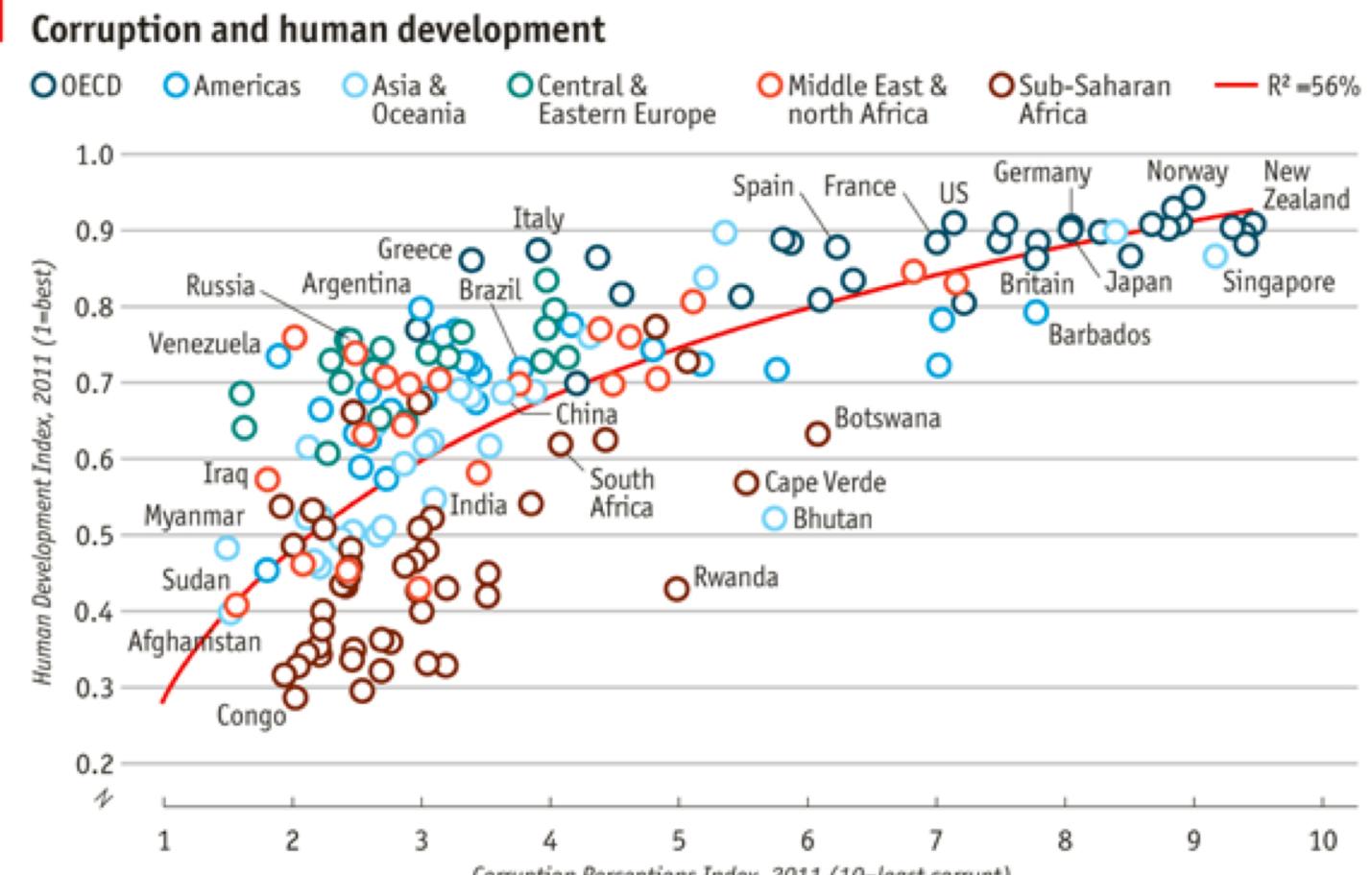
Main options

- 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

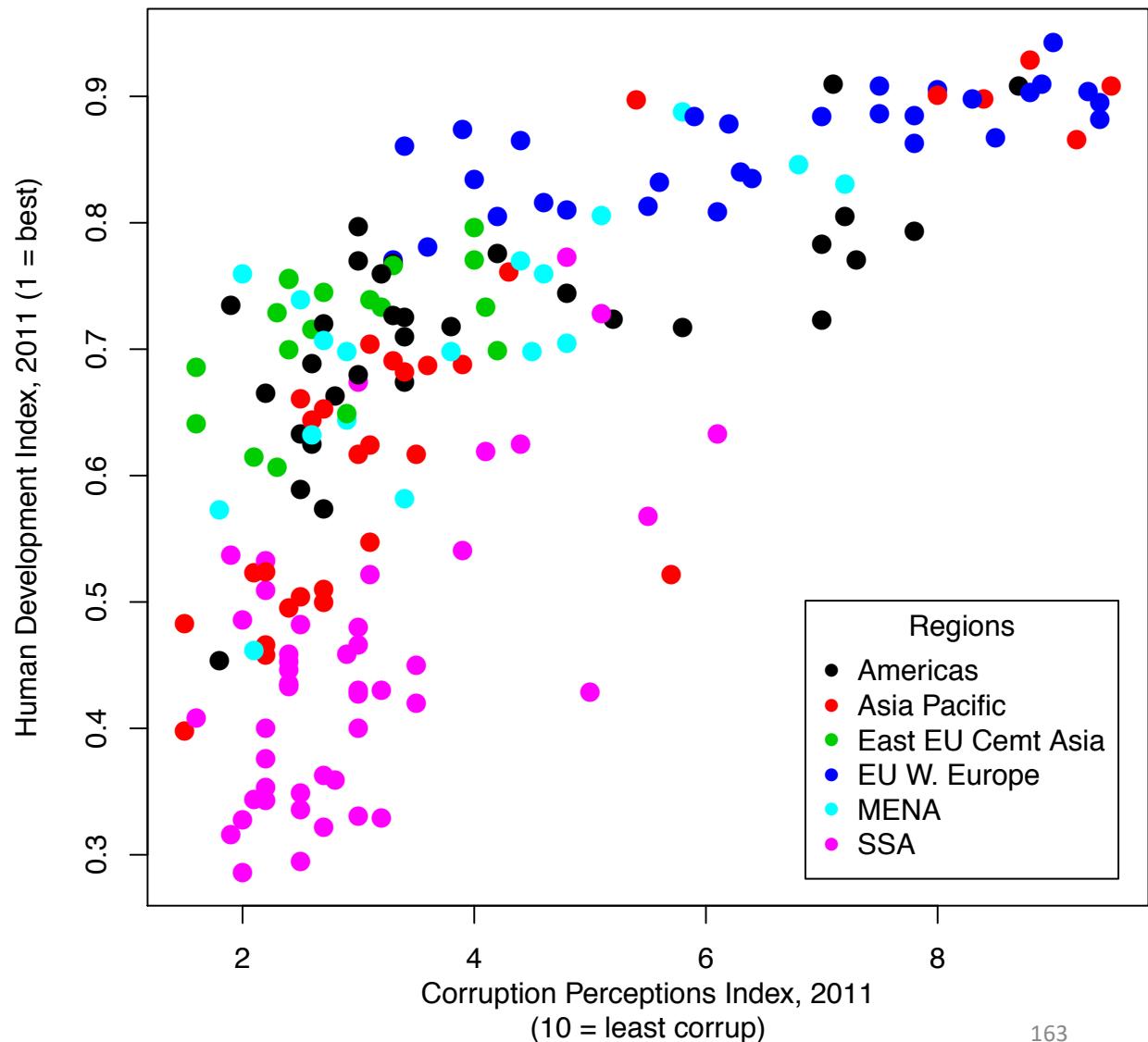


Sources: Transparency International; UN Human Development Report

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 corrupt)",
      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

R colors
<http://research.stowers-institute.org/efg/R/Color/Chart>

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75
76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125
126	127	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150
151	152	153	154	155	156	157	158	159	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	192	193	194	195	196	197	198	199	200
201	202	203	204	205	206	207	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	224	225
226	227	228	229	230	231	232	233	234	235	236	237	238	239	240	241	242	243	244	245	246	247	248	249	250
251	252	253	254	255	256	257	258	259	260	261	262	263	264	265	266	267	268	269	270	271	272	273	274	275
276	277	278	279	280	281	282	283	284	285	286	287	288	289	290	291	292	293	294	295	296	297	298	299	300
301	302	303	304	305	306	307	308	309	310	311	312	313	314	315	316	317	318	319	320	321	322	323	324	325
326	327	328	329	330	331	332	333	334	335	336	337	338	339	340	341	342	343	344	345	346	347	348	349	350
351	352	353	354	355	356	357	358	359	360	361	362	363	364	365	366	367	368	369	370	371	372	373	374	375
376	377	378	379	380	381	382	383	384	385	386	387	388	389	390	391	392	393	394	395	396	397	398	399	400
401	402	403	404	405	406	407	408	409	410	411	412	413	414	415	416	417	418	419	420	421	422	423	424	425
426	427	428	429	430	431	432	433	434	435	436	437	438	439	440	441	442	443	444	445	446	447	448	449	450
451	452	453	454	455	456	457	458	459	460	461	462	463	464	465	466	467	468	469	470	471	472	473	474	475
476	477	478	479	480	481	482	483	484	485	486	487	488	489	490	491	492	493	494	495	496	497	498	499	500
501	502	503	504	505	506	507	508	509	510	511	512	513	514	515	516	517	518	519	520	521	522	523	524	525
526	527	528	529	530	531	532	533	534	535	536	537	538	539	540	541	542	543	544	545	546	547	548	549	550
551	552	553	554	555	556	557	558	559	560	561	562	563	564	565	566	567	568	569	570	571	572	573	574	575
576	577	578	579	580	581	582	583	584	585	586	587	588	589	590	591	592	593	594	595	596	597	598	599	600
601	602	603	604	605	606	607	608	609	610	611	612	613	614	615	616	617	618	619	620	621	622	623	624	625
626	627	628	629	630	631	632	633	634	635	636	637	638	639	640	641	642	643	644	645	646	647	648	649	650
651	652	653	654	655	656	657																		

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	#FFEFDDB	255	239	219
5	antiquewhite2	#EEDFCC	238	223	204
6	antiquewhite3	#CDC0B0	205	192	176
7	antiquewhite4	#8B8378	139	131	120
8	aquamarine	#7FFFDD	127	255	212
9	aquamarine1	#7FFFDD	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

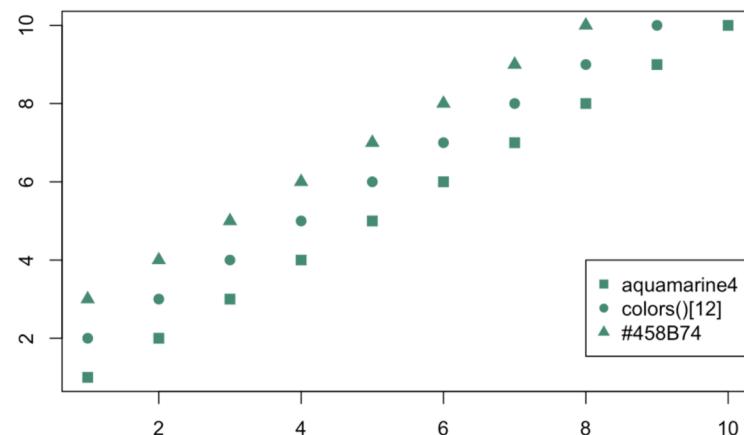
Exercise. Access individual colors

- 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)
```

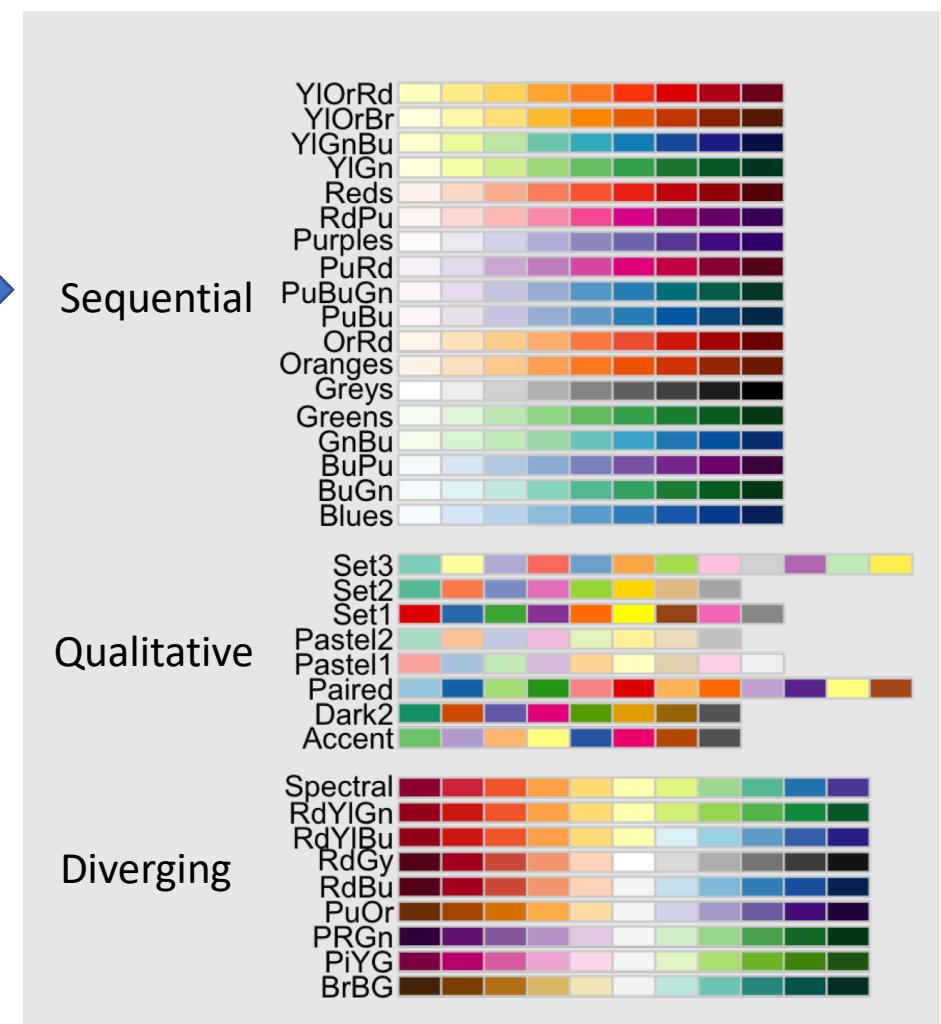
```
plot(x=1:3, y=1:3, col=colors()[c(91,12,641)], pch=15,cex=1.2)
```

Package RColorBrewer

- Finding the right combination of colors can take forever

Package RColorBrewer

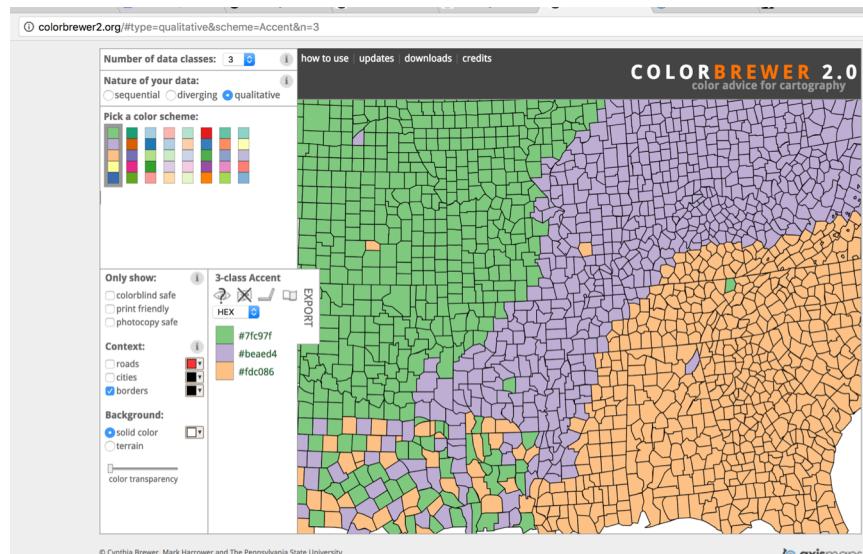
- Finding the right combination of colors can take forever
- RColorBrewer has ready made palettes →



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!)

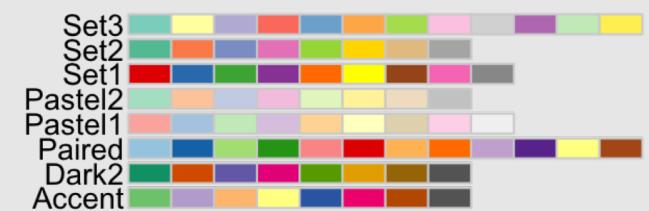
<http://colorbrewer2.org/#type=qualitative&scheme=Accent&n=3>



Sequential



Qualitative



Diverging



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

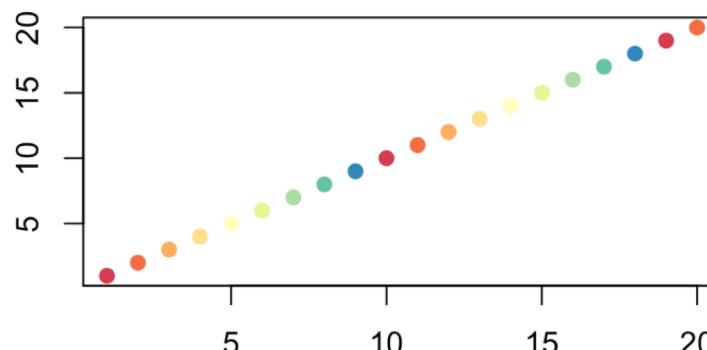
```
brewer.pal(n=9, name = "Spectral")
```

→ 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



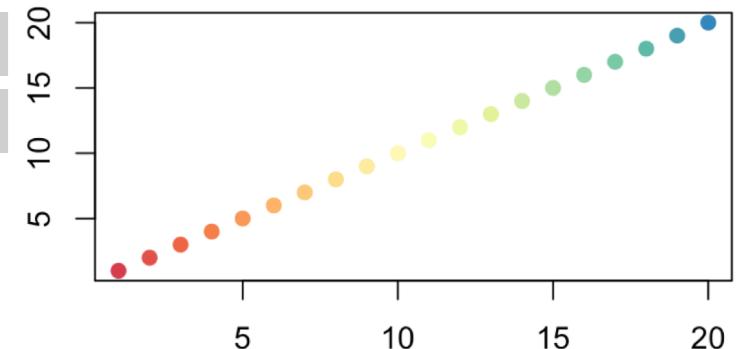
Package RColorBrewer

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:

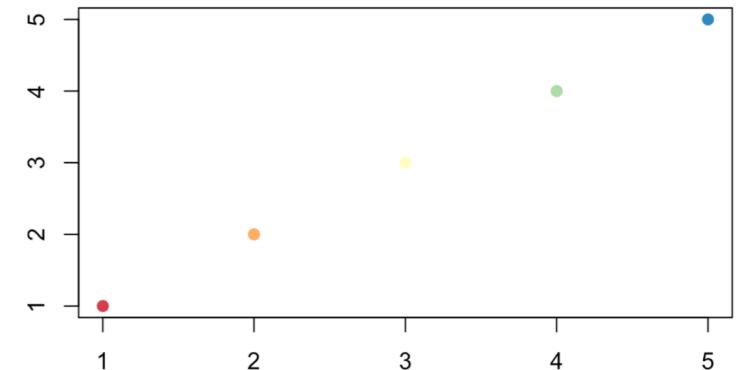
```
myPal <- colorRampPalette(brewer.pal(9, "Spectral"))
```

```
plot(1:20, col= myPal(20), pch=19)
```

```
plot(1:5, col= myPal(5), pch=19)
```

Extend

Reduce



Package RColorBrewer

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

```
myPal <- colorRampPalette(brewer.pal(9,"Spectral"))
```

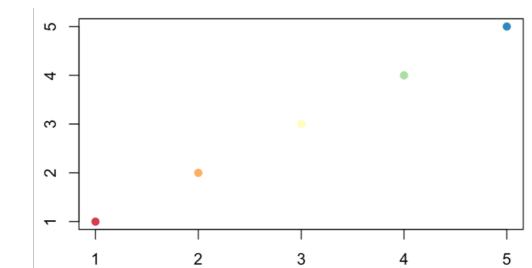
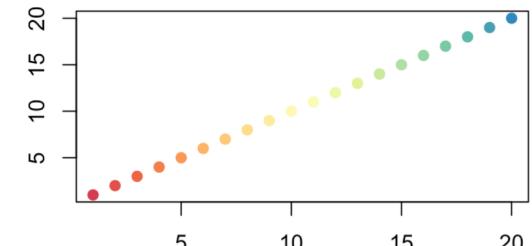
```
plot(1:20, col= myPal(20),pch=19)
```

Extend

```
plot(1:5, col= myPal(5),pch=19)
```

Reduce

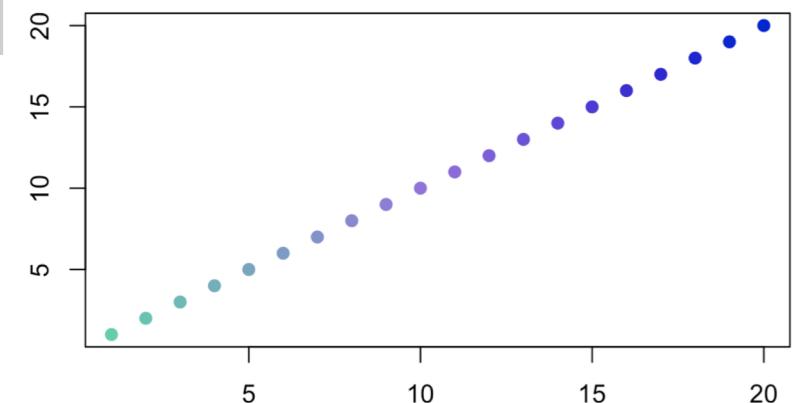
colorRampPalette()



3. Or make your own palettes!

```
myPal2 <- colorRampPalette(c("mediumaquamarine","mediumpurple","mediumblue"))
```

```
plot(1:20, col= myPal2(20),pch=19)
```



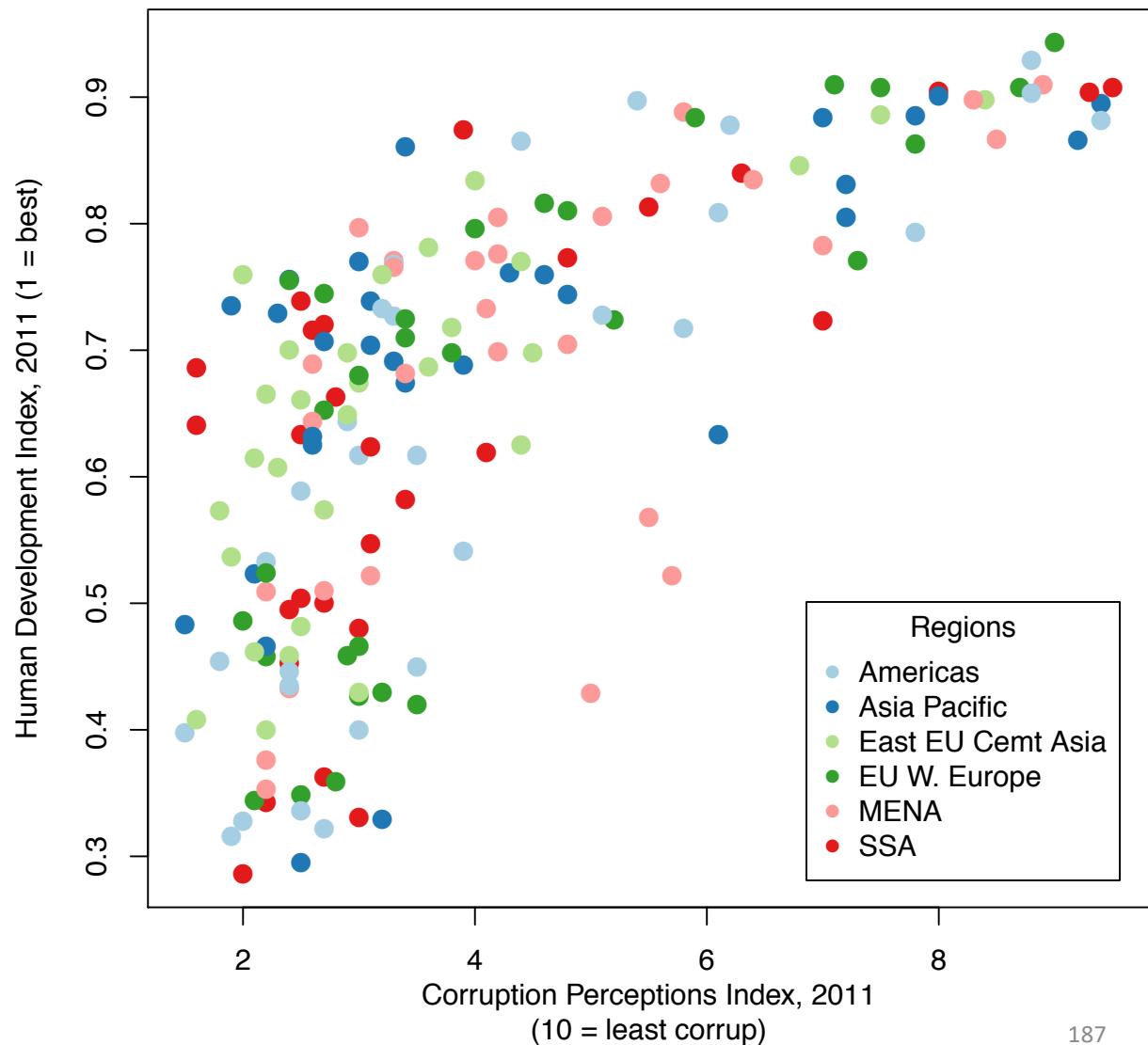
Check out all these options in the tutorial

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 corrupt)",
      ylab = "Human Development Index, 2011 (1 = best")
)
legend("bottomright", title = "Regions", legend = levels(data$Region),
col = brewer.pal(6,"Paired"), pch = 16)
```