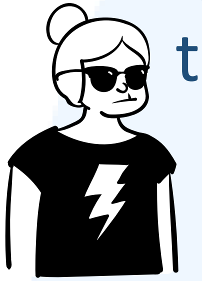
Welcome to LSM3257 !

Many people think that the sign over our module "gate" reads…

Dante (1320). *"Divine Comedy".*

But there is hope!

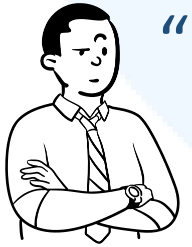


Student A

“After this module, I **can analyse data** through R, it is **very useful**”

“What has been taught will be **very useful and applicable** in other modules/ future projects – good skill to acquire.”

Student C



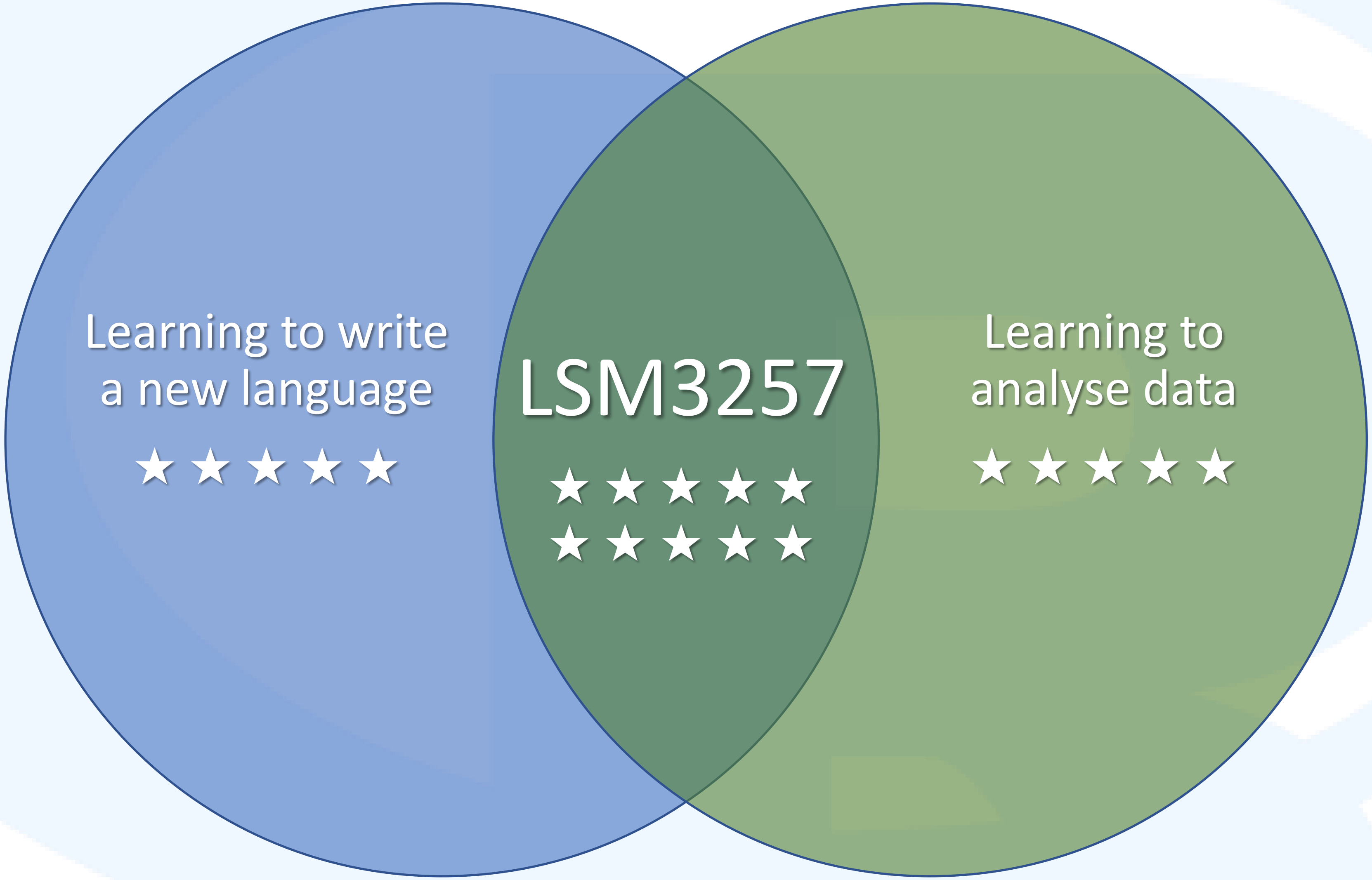
Student B

“**Challenging** but **very applicable and useful**”

“**Challenging** but interesting.”

Student D

Why is LSM3257 challenging?



Learning to write a new language

★ ★ ★ ★ ★

LSM3257

★ ★ ★ ★ ★
★ ★ ★ ★ ★

Learning to analyse data

★ ★ ★ ★ ★

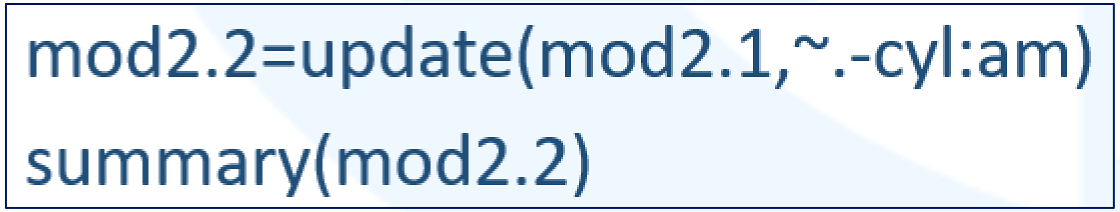
Take home message: be prepared to work hard … but it will be rewarding!

Module Overview

Designed to teach practical skills: so you'll **learn by trying and doing**!

At the end of this module, you will be able to:

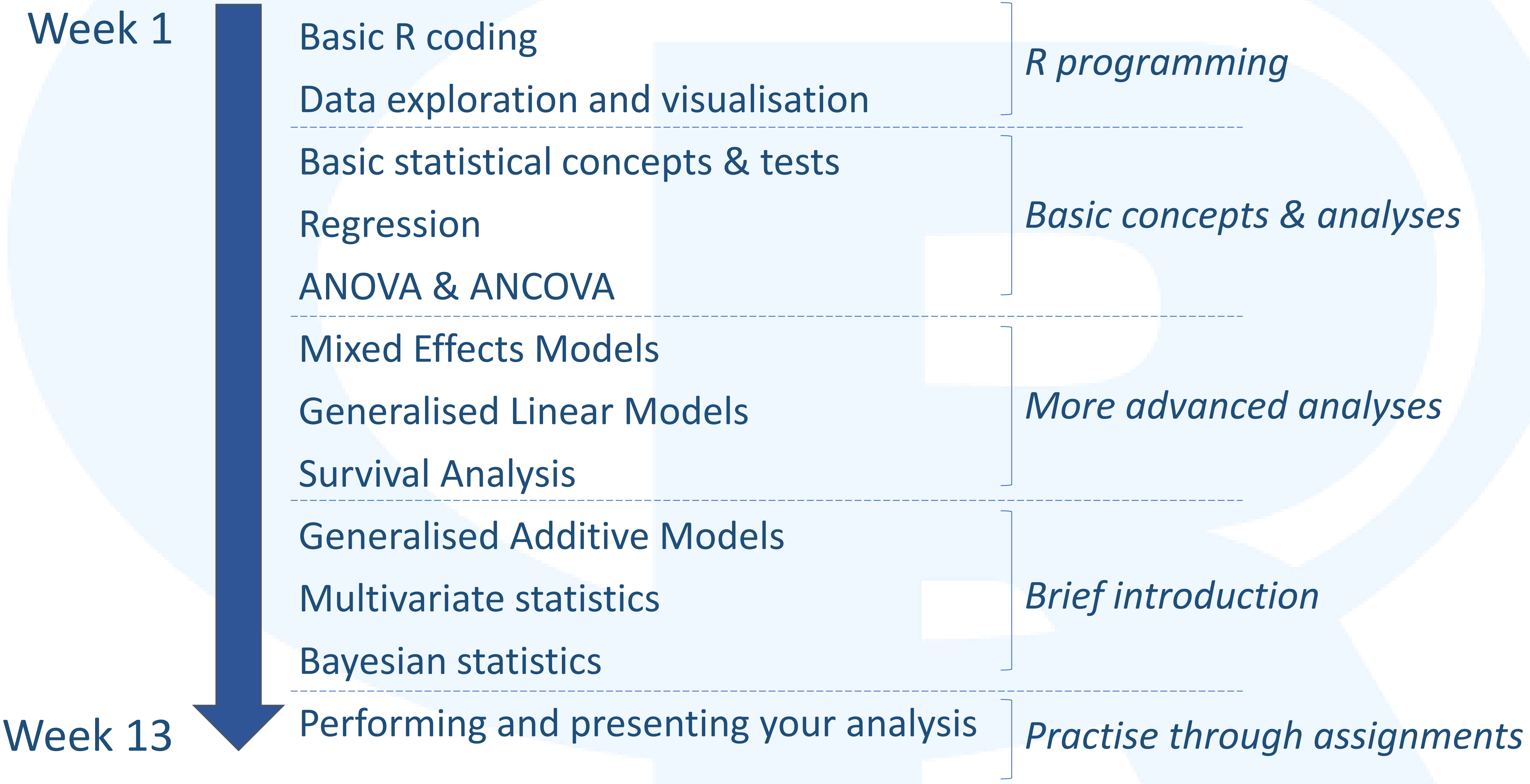
- Understand R code



```
mod2.2=update(mod2.1,~.-cyl:am)
summary(mod2.2)
```

- Write R code to perform and troubleshoot a data analysis
- Identify and install new R packages that may be relevant to your study
- Critically analyse the quality of data and data analyses
- Bring a study through: from research question → identifying relevant data → data analysis & interpretation → conclusions

Module Overview



Week 1

Basic R coding

Data exploration and visualisation

R programming

Basic statistical concepts & tests

Regression

ANOVA & ANCOVA

Basic concepts & analyses

Mixed Effects Models

Generalised Linear Models

Survival Analysis

More advanced analyses

Generalised Additive Models

Multivariate statistics

Bayesian statistics

Brief introduction

Performing and presenting your analysis

Practise through assignments

Week 13

Module Overview

Practical, Practical, Practical!

The Teaching Team

Dr Ian Chan

Lecturer

ianchan@nus.edu.sg

Ma Jinqi

Teaching Assistant

jinqima@nus.edu.sg

Su Tingting

Teaching Assistant

e0983569@u.nus.edu

Housekeeping

Class issues

Layout not ideal: I will Zoom every class (and put the recordings up) so you can get a closer look on your own screen

Starved for power outlets: power extension bars for everyone!

8am timing: sorry, I'd change it too if I had the choice…

Fonts used

I will send slides in pptx format. If words in your slides look out of alignment, make sure you have the following fonts installed:

- Agency FB
- Calibri (Body)

-Courier New

Basic Data Handling in R

Lecture 1

LSM3257

AY22/23; Sem 2 | Ian Z.W. Chan

Summary (Learning Objectives)

What is R?

- How to analyse data in R

Installing R

Starting an R Project

- Working folder
- R scripts and syntax
- Installing packages

Importing datasets in R: read_xl(), read.table(), read.csv()

Preparing your data

- Calculator
- Functions
- Objects: Vectors, Matrices, Dataframes
- Conditionals and loops

What is R?

What is R

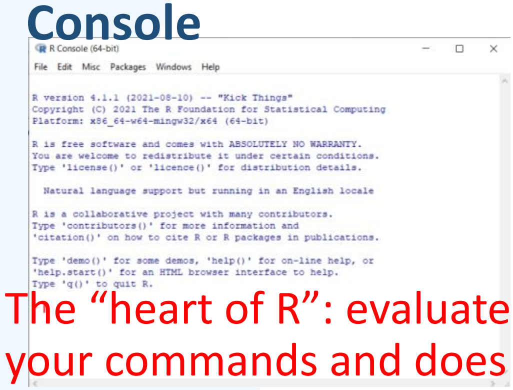
Note: “aka” = “also known as”

A program (aka “environment”) and programming language mainly used for data analysis/modelling

R Studio

Base R

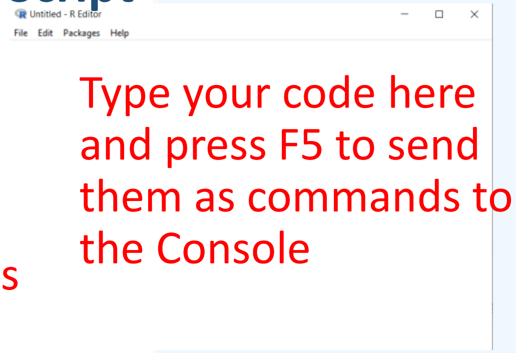
Console



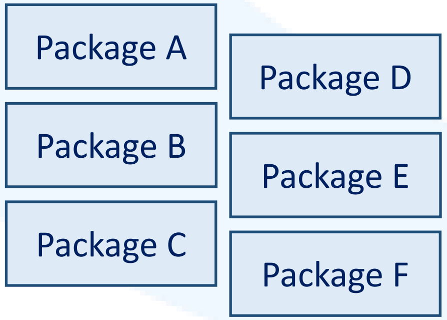
```
R version 4.1.1 (2021-08-10) -- "Kick Things"
Copyright (C) 2021 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

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 “heart of R”: evaluates your commands and does the analysis

Script

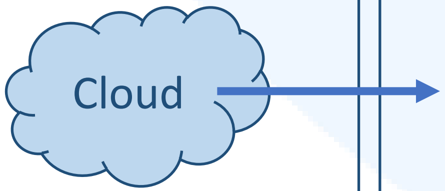


Type your code here and press F5 to send them as commands to the Console



Packages run your commands and do analysis

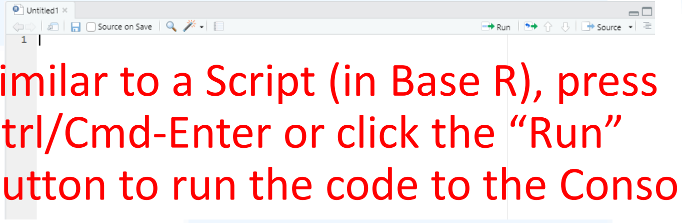
Some packages are preinstalled in Base R



Install more packages to run more sophisticated analyses

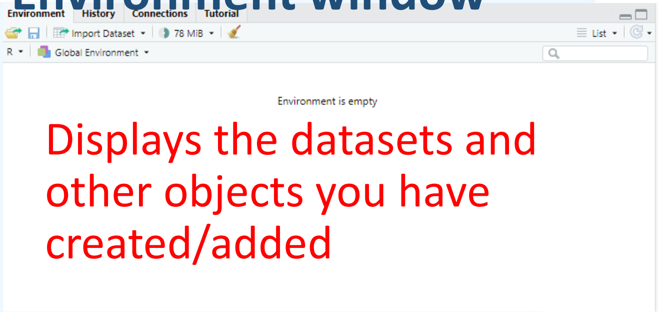
Graphic User Interface

Source window



Similar to a Script (in Base R), press Ctrl/Cmd-Enter or click the “Run” button to run the code to the Console

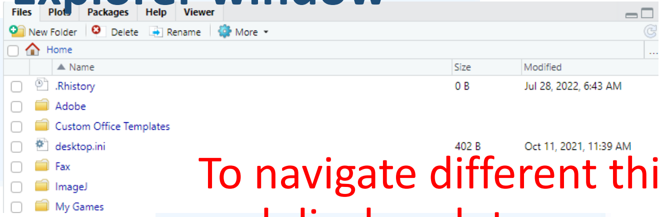
Environment window



Environment is empty

Displays the datasets and other objects you have created/added

Explorer window



Name	Size	Modified
.Rhistory	0 B	Jul 28, 2022, 6:43 AM
Adobe		
Custom Office Templates		
desktop.ini	402 B	Oct 11, 2021, 11:39 AM
Fax		
ImageJ		
My Games		

To navigate different things and display plots

What is R

Base R vs. R Studio

- Base R is just text; R Studio adds colours, autocomplete, error detection, etc.

- R Studio gives you more help so I recommend you start with that (I personally use Base R but I'm just a dinosaur)

- R Studio is a separate layer on top so you have to **install Base R first then install R Studio**

R in Windows vs. R in Mac

- Runs in both but with some minor differences (different installation file, Ctrl-Enter vs. Cmd-Enter, directories are specified differently, etc.)

- In both cases, you need to know how to navigate the folders and files in your computer!

What is R

R vs. Python

R

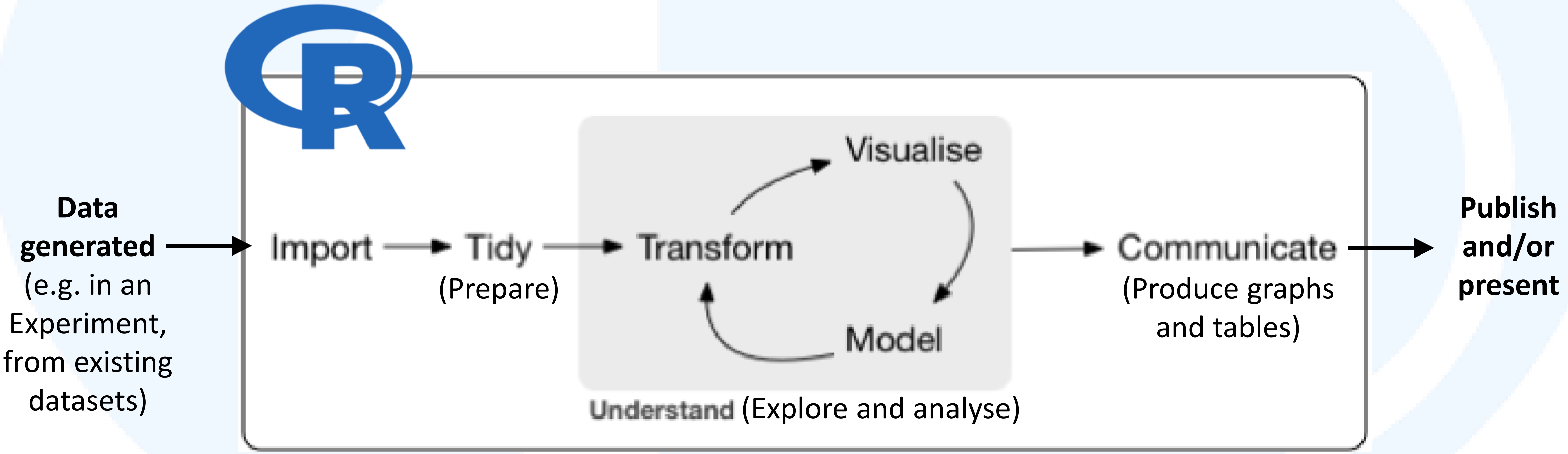
- Programming for data analysis
- More, and more powerful, statistical packages
- More complex language
- Cunning, intelligent, detective work

Python

- General purpose programming language
- Better for machine learning, web apps, integration with outside apps
- Slightly simpler syntax
- Strong, elegant, wide range of abilities

But they both have one thing in common…

How R fits into research and data science…



Data generated (e.g. in an Experiment, from existing datasets)

Import

Tidy (Prepare)

Transform

Visualise

Model

Understand (Explore and analyse)

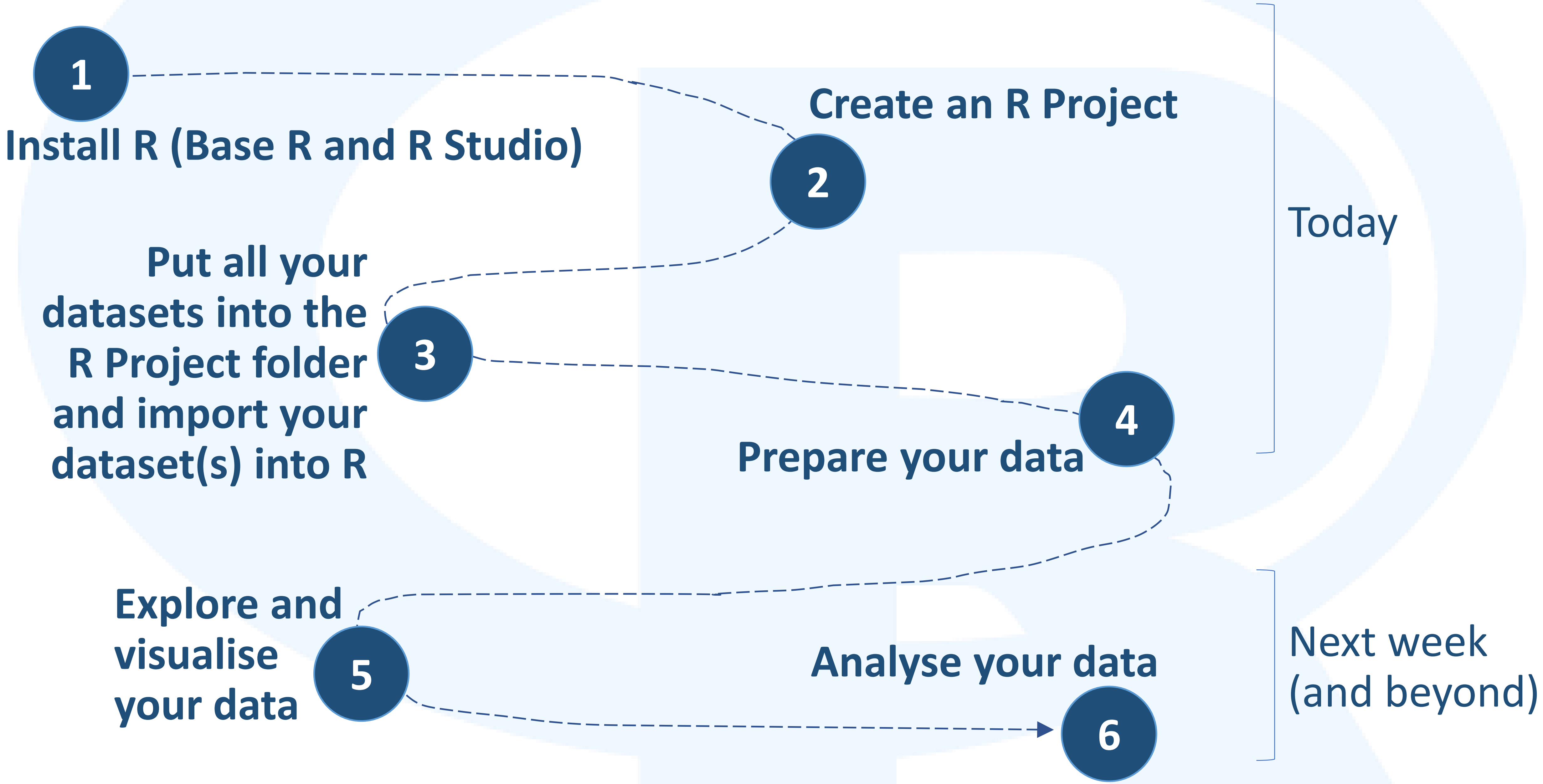
Communicate (Produce graphs and tables)

Publish and/or present

Grolemund (2017)

This module introduces you to this whole process

How to analyse data in R



1

Install R (Base R and R Studio)

2

Create an R Project

3

Put all your datasets into the R Project folder and import your dataset(s) into R

4

Prepare your data

5

Explore and visualise your data

6

Analyse your data

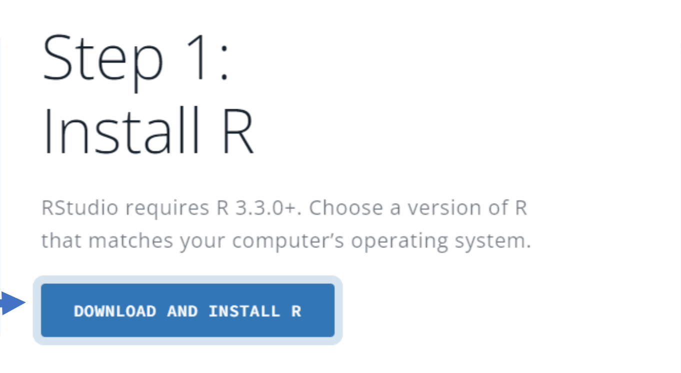
Today

Next week (and beyond)

Installing R

Download and installation

1) Go to: https://posit.co/download/rstudio-desktop/



Step 1:
Install R

RStudio requires R 3.3.0+. Choose a version of R that matches your computer's operating system.

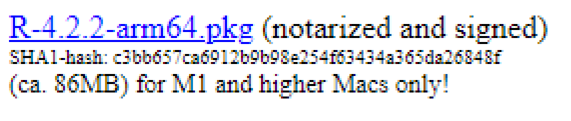
DOWNLOAD AND INSTALL R

2) Install Base R first:

- Click the “DOWNLOAD AND INSTALL R” button
- Download the software

For Macs

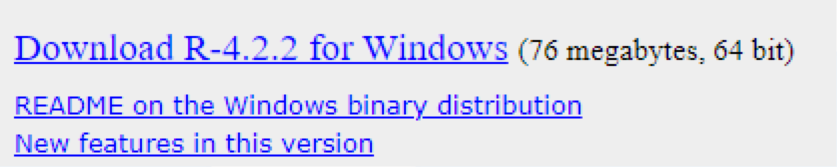
- Click “Download R for macOS”
- Click the latest package (e.g. “R-4.2.2.pkg”)



R-4.2.2-arm64.pkg (notarized and signed)
SHA1-hash: c3bb657ca6912b9b98e254f63434a365da26848f
(ca. 86MB) for M1 and higher Macs only!

For Windows

- Click “Download R for Windows”
- Click “base”
- Click the “Download R-X.X.X for Windows”



Download R-4.2.2 for Windows (76 megabytes, 64 bit)

README on the Windows binary distribution
New features in this version

- Complete the installation on your computer

Download and installation

3) Install Rstudio:

- Download the installer

For Windows

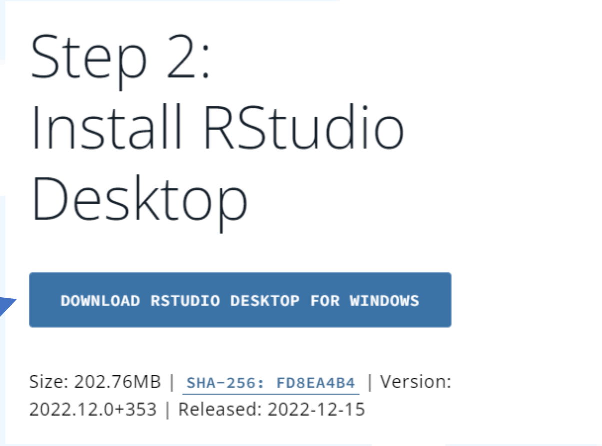
- Click the “DOWNLOAD RSTUDIO DESKTOP FOR WINDOWS” button

(or you can scroll down and click this download link manually)

For Macs

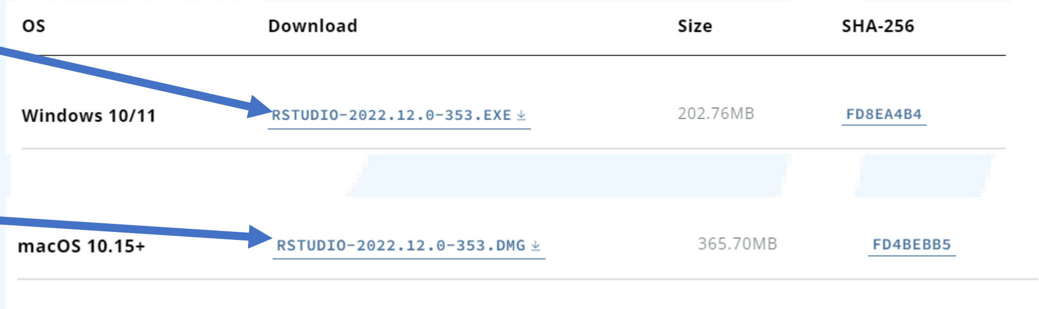
- Scroll down and click the macOS 10.15+ download link

- Complete the Installation on your computer



Step 2:
Install RStudio Desktop

Size: 202.76MB | SHA-256: FD8EA4B4 | Version: 2022.12.0+353 | Released: 2022-12-15



OS	Download	Size	SHA-256
Windows 10/11	RSTUDIO-2022.12.0-353.EXE ⤓	202.76MB	FD8EA4B4
macOS 10.15+	RSTUDIO-2022.12.0-353.DMG ⤓	365.70MB	FD4BEBB5

4) Start up R Studio

- R studio will automatically start up Base R inside R Studio

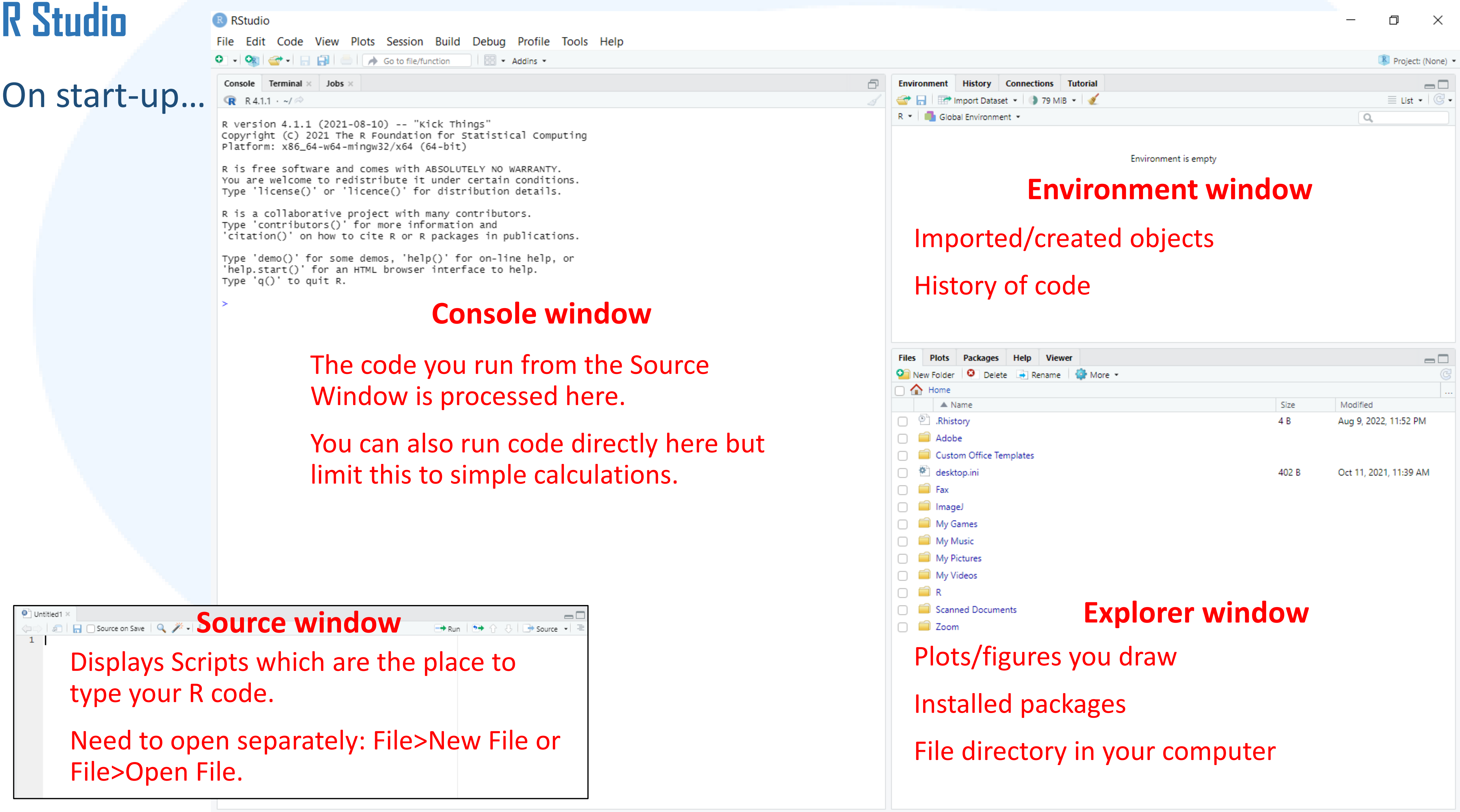
- If you keep using R Studio, you will probably never have to start up Base R manually ever

Base R icon

R Studio icon

R Studio

On start-up…



```
R version 4.1.1 (2021-08-10) -- "Kick Things"
Copyright (C) 2021 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (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.

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 is empty

Name	Size	Modified
.Rhistory	4 B	Aug 9, 2022, 11:52 PM
Adobe		
Custom Office Templates		
desktop.ini	402 B	Oct 11, 2021, 11:39 AM
Fax		
ImageJ		
My Games		
My Music		
My Pictures		
My Videos		
R		
Scanned Documents		
Zoom		

Console window

The code you run from the Source Window is processed here.

You can also run code directly here but limit this to simple calculations.

Environment window

Imported/created objects

History of code

Source window

Displays Scripts which are the place to type your R code.

Need to open separately: File>New File or File>Open File.

Explorer window

Plots/figures you draw

Installed packages

File directory in your computer

First-time startup settings

Turn on all diagnostics (we need all the help we can get).

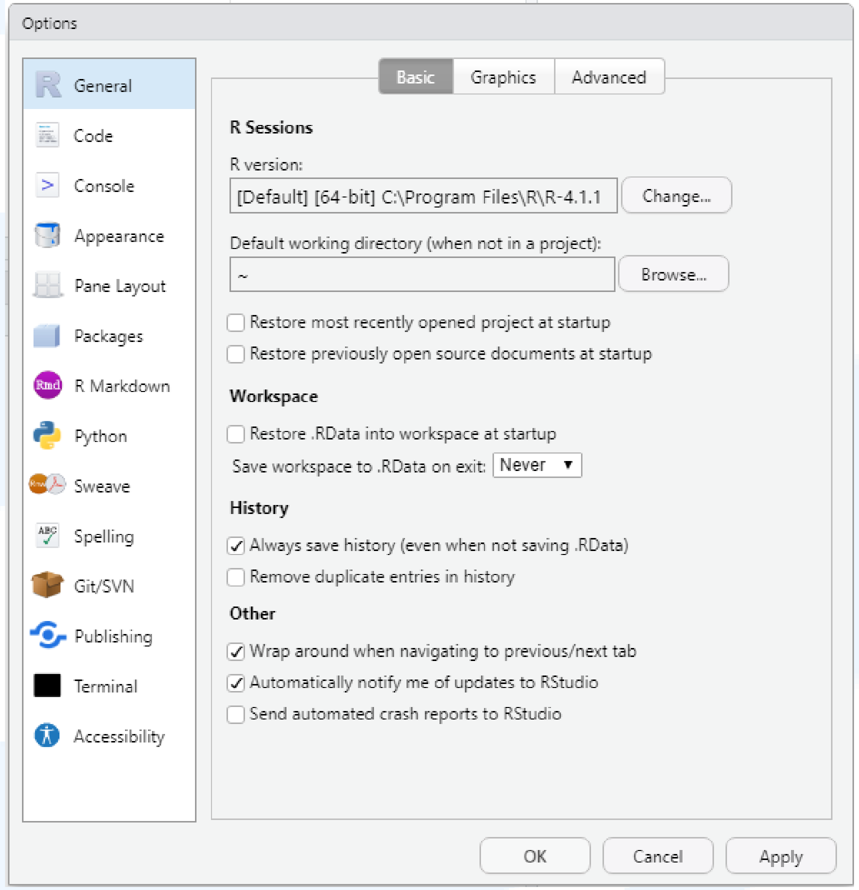
- Click Tools>Global Options>Code>Diagnostics>Turn on everything

Clean up our workflow.

- Click Tools>Global Options>General>Basic
- Uncheck “Restore most recently opened project at startup”
- Uncheck “Restore previously open source documents at startup”
- Uncheck “Restore .RData into workspace at startup”
- Set “Save workspace to .RData on exit:” to “Never”

Cheatsheets (pdf downloads) are available in R Studio.

- Click Help>Cheatsheets



Options

General | Basic

R Sessions

R version: [Default] [64-bit] C:\Program Files\R\R-4.1.1

Default working directory (when not in a project): ~

□ Restore most recently opened project at startup

□ Restore previously open source documents at startup

Workspace

□ Restore .RData into workspace at startup

Save workspace to .RData on exit: Never

History

☑ Always save history (even when not saving .RData)

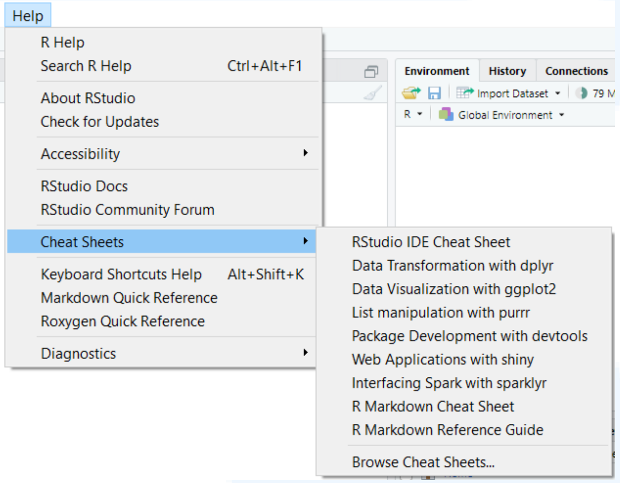
□ Remove duplicate entries in history

Other

☑ Wrap around when navigating to previous/next tab

☑ Automatically notify me of updates to RStudio

□ Send automated crash reports to RStudio



Starting an R Project

Create a new project

When starting a new analysis, first create a new R Project (one experiment → one R Project).

- Create a folder where you want to store all your data for the study
- Click File>New Project>Existing Directory
- Click Browse>Select the folder you created>Click Open
- Click Create Project (you should see a .Rproj file appear in that folder)

Pro tip: you may want to select the option to view file extensions to help you identify things more easily

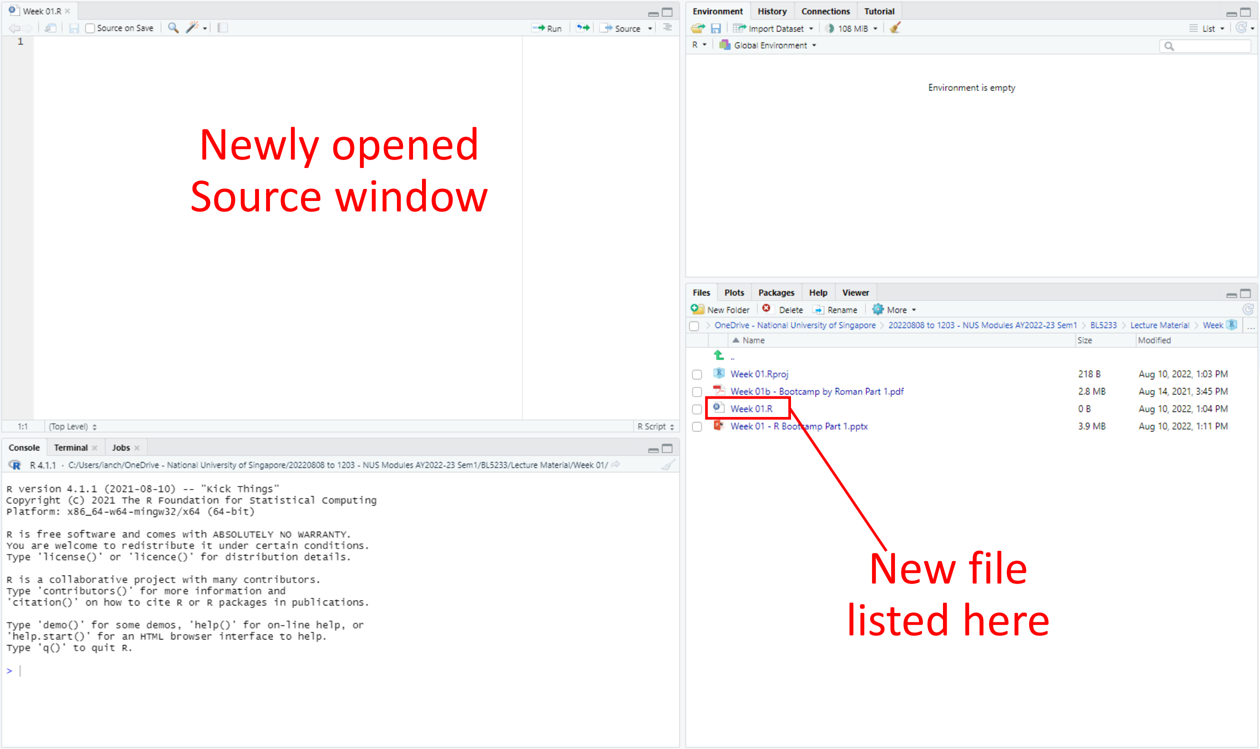
This is your Working Folder for this R Project

- Put all your datasets for your analysis in here
- Remember where to find it!

Create a new script

The first thing to do in a new project is to create a new script

- File>New File>R Script
- A new Source window appears in R Studio
- Click the (very small) Save icon>Name your file whatever you want>Check that the script appears under Files in the Explorer window (you should also see it in your folder)



Newly opened Source window

New file listed here

Week 01.R
Source on Save
Run
Source
1
1:1
(Top Level)
R Script
Environment
History
Connections
Tutorial
Import Dataset
108 MiB
List
Global Environment
Environment is empty
Files
Plots
Packages
Help
Viewer
New Folder
Delete
Rename
More
OneDrive - National University of Singapore > 20220808 to 1203 - NUS Modules AY2022-23 Sem1 > BL5233 > Lecture Material > Week
Console
Terminal
Jobs
R 4.1.1 · C:/Users/ianch/OneDrive - National University of Singapore/20220808 to 1203 - NUS Modules AY2022-23 Sem1/BL5233/Lecture Material/Week 01/

Name	Size	Modified
..		
Week 01.Rproj	218 B	Aug 10, 2022, 1:03 PM
Week 01b - Bootcamp by Roman Part 1.pdf	2.8 MB	Aug 14, 2021, 3:45 PM
Week 01.R	0 B	Aug 10, 2022, 1:04 PM
Week 01 - R Bootcamp Part 1.pptx	3.9 MB	Aug 10, 2022, 1:11 PM

```
R version 4.1.1 (2021-08-10) -- "Kick Things"
Copyright (C) 2021 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (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.

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.

> 
```

Pro tip: Continually use Ctrl-S / Cmd-S to save your script while you work!

You type commands into the Script first and then run them to the Command Line

- It is not good practice to type code directly into the Console (except for simple calculations you don't want to save)

R syntax: what to type in the Source window?

Commands to tell R to do things: e.g.

```
dataset = read.table(“Data.txt”,header=T)
View(dataset) #This allows me to see my data
```

- Case sensitive: <dataset> ≠ <Dataset>.

- Commenting (using #) is useful!

- Commands are separated by a line break: one command, one line (you can get around this by using “+”, but more on this next time).

- If a command is not complete and you run it (e.g. you forgot to add a final close bracket), a “+” appears on the right of the line in the Console. If this happens, you get stuck. Press Esc and make the corrections.

- Both “<-” and “=” mean equals. “<-” is more explicit (you are assigning the item on the right to the item on the left) but I use “=”.

  - Shortcut for “<-” in R Studio: Alt- – (hold down Alt and press minus)

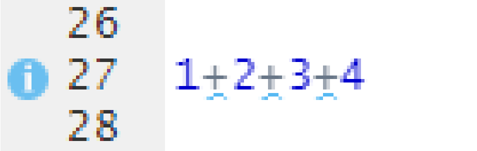
Commenting is writing text that is NOT run by R. The text is just for users to read.

We use it: (1) to remember what we are doing, and (2) to tell people reading our code what we are doing. Very important and good practice to use comments!

To tell R that some text is a comment, add a hex (#) at the beginning of the line or from the point where the comment starts.

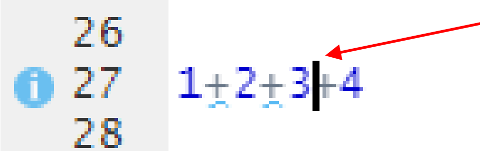
Running code from the Source window to the Console

Let’s say you want to calculate the value of “1+2+3+4”, so you type it into your Source window.



```

1+2+3+4

```

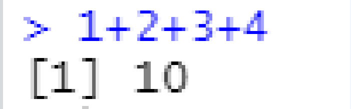
To run the whole line, place your cursor anywhere on the line and then press Ctrl-Enter or Cmd-Enter.



```

1+2+3+4

```

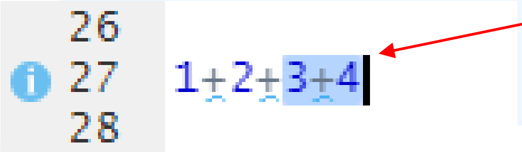
Cursor here

Output in Console:



```
> 1+2+3+4
[1] 10
```

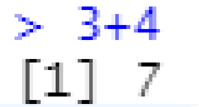
If you want to run only part of the line (e.g. only “3+4”), highlight that part and then press Ctrl-Enter or Cmd-Enter.



```

1+2+3+4

```

Highlighted here

Output in Console:



```
> 3+4
[1] 7
```

If you want to run multiple lines, highlight them (you can press Ctrl-A or Cmd-A to highlight all lines), then press Ctrl-Enter or Cmd-Enter.

Installing packages in R

Packages allow you to do things,
 e.g. manipulate your data, do fancy plots, run advanced analyses

Some packages are automatically installed when you first install Base R, other packages you have to install yourself

Installation is a 2-step process:

With open and close inverted commas

# Step 1: Install the package

```
install.packages("readxl") #This package contains the read_excel() command
```

#Choose “Cloud” and click Enter (if asked)

#Step 2: Load the package (if you don’t do this, you cannot use it)

```
library(readxl)
```

Without inverted commas

To resume working on your project

1) Open the R Project file

Option 1: Open R Studio>File>Open Project

Option 2: Open the .Rproj file in the folder directly (easier)

2) In the Explorer window (within R Studio), open the R Script (.R file) in the R project folder

- All your previously saved code will appear in your Source window

3) Re-run whatever code is needed

- Re-load packages

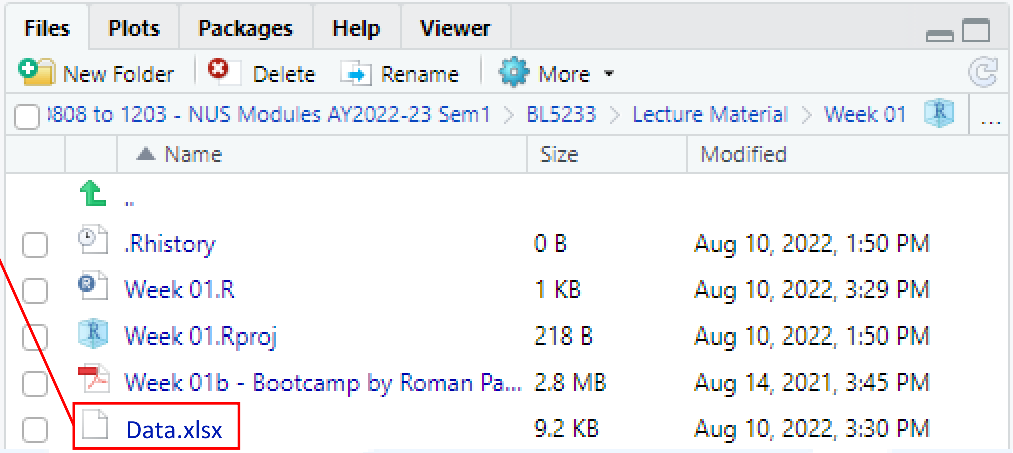
- Re-import datasets, re-create objects

Importing datasets into R

Put datasets into the Working Folder

Copy and paste the dataset (e.g. “Data.xlsx”) into your R Project folder, it should appear under Files in the Explorer window.

Dataset will appear here



808 to 1203 - NUS Modules AY2022-23 Sem1 > BL5233 > Lecture Material > Week 01

Name	Size	Modified
..		
.Rhistory	0 B	Aug 10, 2022, 1:50 PM
Week 01.R	1 KB	Aug 10, 2022, 3:29 PM
Week 01.Rproj	218 B	Aug 10, 2022, 1:50 PM
Week 01b - Bootcamp by Roman Pa...	2.8 MB	Aug 14, 2021, 3:45 PM
Data.xlsx	9.2 KB	Aug 10, 2022, 3:30 PM

Now that your dataset is in your R Project Folder (aka your Working Directory), you can import the data into R using commands in R.

If you’re using Base R, you will need to set your Working Directory to where your excel file is located using the **setwd()** command. For example, if I have stored my file “Data.xlsx” on my desktop, I will need to run:

In Windows

```
setwd(“C:\\Users\\Ian\\Desktop”)
```

#You have to use either two back slashes (\\) or one forward slash (/) to indicate folder levels

In Mac

```
setwd(“~/Desktop”)
```

#You use one forward slash (/) to indicate folder levels

Import the dataset into R

To read my Excel dataset (“Data.xlsx”) from the Working Folder into R:

Step 1: Install the “readxl” package.

```
install.packages("readxl") #This package contains the read_excel() command
library(readxl)
```

Step 2: Use the read_excel() command to import the dataset into R and save it to an object named <data1>.

```
data1 = read_excel(“Data.xlsx”) #The dataset is saved in an Object <data1>
data1 #Running this line of code will allow me to view the dataset in <data1>
```

Remember to save your data to an object, otherwise you won’t be able to use it later. You can choose to have spaces around the “=“ or not. R understands both.

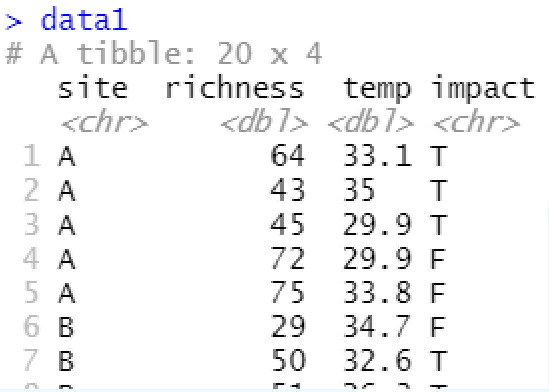
Two other (more difficult, but more traditional) ways to get data from Excel into R:

1) Save your file as a tab-delimited .txt file in Excel (File>Save as) → Use read.table()
   Must specify that your variables have a header row, e.g. read.table("Data.txt",header=T).

2) Save your file as a CSV .csv file in Excel (File>Save as) → Use read.csv()
   No need to specify, e.g. read.csv("Data.csv")

View the imported dataset

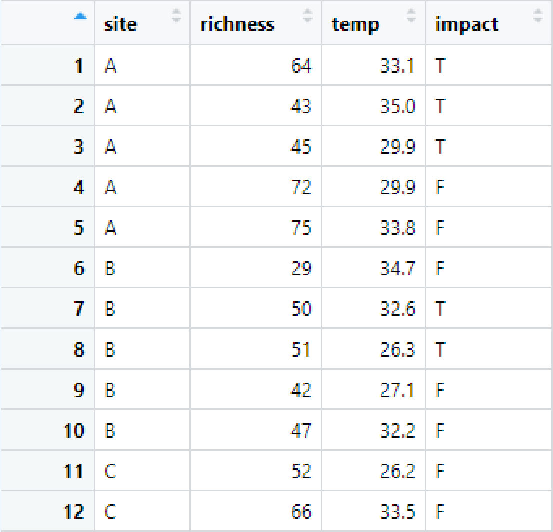
```
data1 #Shows either the first 10 rows or the whole dataset
head(data1) #Shows the first 6 rows of the dataset
tail(data1) #Shows the last 6 rows of the dataset
```



```
> data1
# A tibble: 20 x 4
  site  richness  temp impact
  <chr>    <dbl> <dbl> <chr>
1 A           64  33.1 T
2 A           43  35   T
3 A           45  29.9 T
4 A           72  29.9 F
5 A           75  33.8 F
6 B           29  34.7 F
7 B           50  32.6 T
```

```
View(data1) #Shows the dataset in a nice table in the Source window
```

- Can also click on the dataset in the Environment window



	site	richness	temp	impact
1	A	64	33.1	T
2	A	43	35.0	T
3	A	45	29.9	T
4	A	72	29.9	F
5	A	75	33.8	F
6	B	29	34.7	F
7	B	50	32.6	T
8	B	51	26.3	T
9	B	42	27.1	F
10	B	47	32.2	F
11	C	52	26.2	F
12	C	66	33.5	F

```
str(data1)
```

- Gives you important information about the dataset and variables (we’ll learn more about this later)

Preparing the data

This is like learning a language, you’re not going to remember all the words immediately—don’t be discouraged!

Just remember it can be done and refer back here to see the exact code required.

R is a calculator

```
5+7 #This code outputs the answer in the Console
9-1
3*4
7/2
2^3
(5+7)/4^2
4%%3 #Modulo, i.e. finds the remainder
```

Assigning values to objects

```
e1=5+7 #This code assigns the answer to the object <e1>
e2=9-1
e3=3*4
e4=7/2
e5=2^3
e6=4%%3
```

Object naming rules

You can name your objects anything you want, except you…

- CANNOT begin with a number or symbol: 1data or $data

- SHOULD NOT contain a symbol (&, $, <, etc.)

- CANNOT have spaces in the name

Tips

Use names that make sense: height, age

Use a naming convention: this_is_snake_case, thisIsCamelCase, this.uses.periods

Functions do things to objects

```
log(e1)
exp(e2)
sqrt(4)
log(e3,10) #Log base 10 of e3
factorial(5)
floor(5.1) #Rounds down
ceiling(5.1) #Rounds up
abs(e3)
cos(e4)
sin(e1)
tan(e2)
acos(0)
round(x,digits=0)
```

This “log()” is a function. <e1> is an object that the function is performed on.

“digits” is an argument that must be provided to the round() function. Different functions need different arguments, and you just have to slowly learn them (like learning grammar in a language).

Objects

The **nouns in R** (i.e. the things that store information).

The information can be one number, many numbers arranged in a straight line, many numbers arranged in a grid, etc.

Functions

The **verbs in R** (i.e. the actions done to the Objects). Different functions are designed to act on different types of Objects.

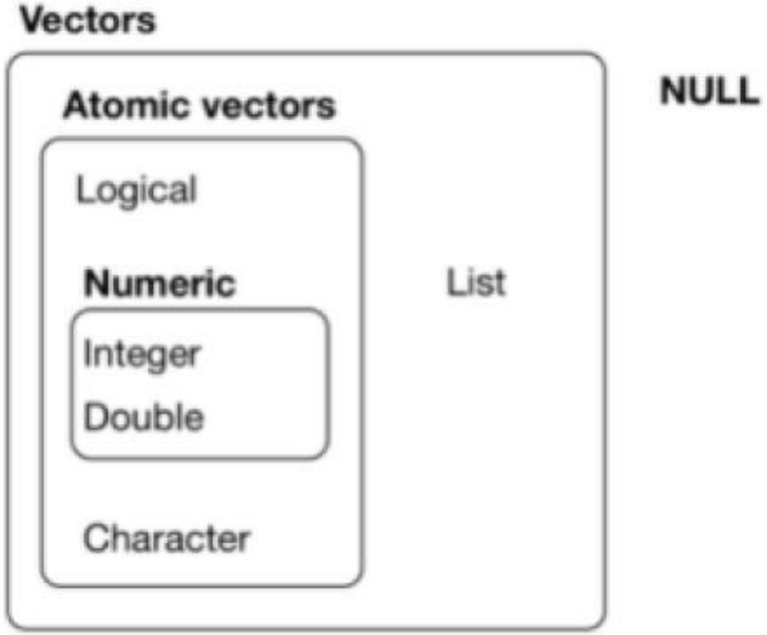
Different types of objects

Vector: many numbers (or other info) arranged in a one-dimensional row

```
v1=c(2,5,8,3,7,4,9,4) #A vector of with 8 elements, i.e. length=8
length(v1) #This would return 8
```

- A single value is a vector of length=1

- Vectors can store different types of information: numbers without decimal places (Integer), numbers with decimal places (Double), TRUE/FALSE (Logical), words/words with numbers (Character)



Matrix: multiple numbers (or other info) arranged in a two-dimensional rectangle

- Can only contain one type of information, e.g.:

Vector 1:	2	5	8	3	7	4	9	4
Vector 2:	1	7	4	4	9	6	3	7

Different types of objects

Dataframe: multiple numbers (and/or other info) arranged in a two-dimensional rectangle

- Different vectors can contain different types of information

Vector 1:	2	5	8	3	7	4	9	4
Vector 2:	A	B	A	A	B	A	B	B
Vector 3:	True	False	False	False	True	False	True	False

- This is the type of object that is created when you read datasets in using read.table() and read.csv()

Tibble: a Dataframe with a few slight differences (used in Tidyverse)

- This is the type of object we will work with most of the time to store datasets because read_excel() creates this type of object

A word on Tidyverse

Tidyverse is a collection of R packages written by a lot of different researchers to try to make coding in R more friendly.

Some researchers feel that Tidyverse is the best thing in the world since sliced bread. I disagree: I think some aspects are great, some not so. So I'll be introducing to you the great ones (e.g. ggplot2) and leaving out the others.

After the course, when you are more familiar with R, you can look up Tidyverse for yourself (https://www.tidyverse.org/) and decide whether you want to use all its functions.

Some common functions for vectors

```
length() #Check the number of elements in your vector
mean()
median()
max()
min()
range() #Returns the smallest and largest values
sd() #Returns the standard deviation of the vector
var() #Returns the variance of the vector
sum()
prod() #Returns the product of all the elements
sort() #Returns a vector with the elements sorted from smallest to largest
rank(v1) #Returns the rank of each element from smallest (1) to largest (8,
         #because there are 8 elements in <v1>)
quantile()
sample(x,replace=T) #Samples values from the vector and replaces them
match(v1,v2) #Checks whether the elements in 2 vectors are identical
```

If I write something like that (with two brackets after), you know I’m referring to a function

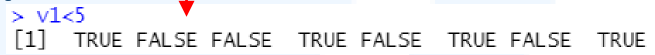
Extract values from vectors (subsetting)

To extract one or more element(s), we can use square brackets []:

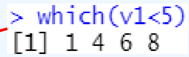
```
v1=c(2,5,8,3,7,4,9,4)
v1[1] #Returns the first element: 2
v1[4] #Returns the fourth element: 3
v1[c(1,4)] #Returns a vector containing the first and fourth element: c(2,3)
```

To extract elements based on their value, we can use Logicals:

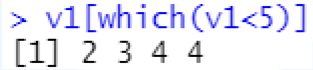
Values 1, 4, 6 and 8 are smaller than 5, therefore TRUE



```
> v1<5
[1]  TRUE FALSE FALSE  TRUE FALSE  TRUE FALSE  TRUE
```

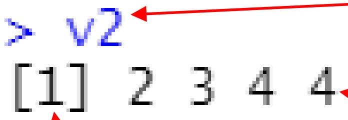


```
> which(v1<5)
[1] 1 4 6 8
```



```
> v1[which(v1<5)]
[1] 2 3 4 4
```

```
v1[v1<5] #Extracts the elements which are smaller than 5
v1[which(v1<5)] #This is equivalent to "v1[c(1,4,6,8)]"
v2=v1[v1<5] #To save all the values smaller than 5 in <v1> to <v2>
v2
```



```
> v2
[1] 2 3 4 4
```

The command that was run: this command tells R to display whatever is in the vector <v2>

The values of the four elements in the vector

The numbering of the first element in this row (useful with long vectors that need more than one row to be displayed)

Editing values of elements

To change one value in the vector, we can use the square brackets again:

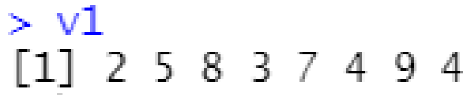
```
v1[1]=3 #This changes the value of the first element from 2 to 3
```

To change all values in the vector:

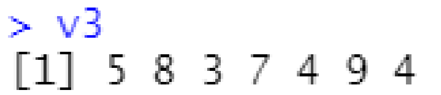
```
v1=v1+2 #Adds 2 to all the elements in <v1>
```

Removing an element

```
v3=v1[-1] #Removes the 1st element of <v1> and saves everything else in <v3>
```



```
> v1
[1] 2 5 8 3 7 4 9 4
```



```
> v3
[1] 5 8 3 7 4 9 4
```

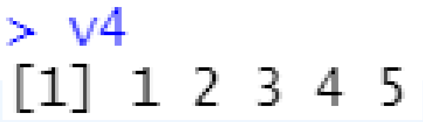
But these kind of manipulations in R are a little inconvenient. My suggestion is to just do it in Excel and reimport the entire dataset.

Creating vectors from scratch

Debugging Time!

Can you spot the error in this slide?

These all create the same vector:



```
> v4
[1] 1 2 3 4 5
```

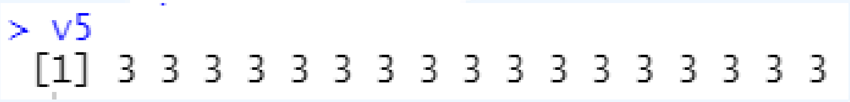
```
v4=1:5
v4=c(1,2,3,4,5)
v4=seq(1,2,1) #seq(x,y,z): go from x to y, increasing by z each time
```

You just have to remember that this is how you string a few different values together in R

The seq() function requires you to provide 3 values for these 3 arguments.

Or you can create a vector filled with only one value:

```
v5=rep(3,17) #rep(x,y): repeat x y-times
```

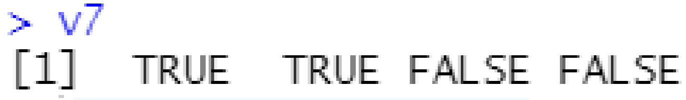


```
> v5
 [1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
```

- We sometimes use this to create vectors of 0s before we assign values to each element.

To create vectors of words (characters) or True/False data:

```
v6=c("Red","Blue","Green","Yellow") #Must use inverted commas around each
v7=c(T,T,F,F) #True/False type data do not require inverted commas
#Can also be entered as TRUE and FALSE
```

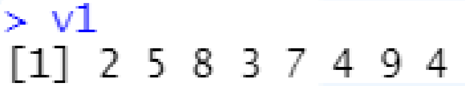


```
> v7
[1]  TRUE  TRUE FALSE FALSE
```

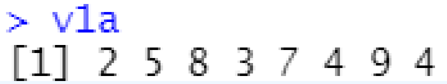
Combining vectors to make matrices

If you start with different vectors:

```
v1a=v1 #Duplicate <v1>
```

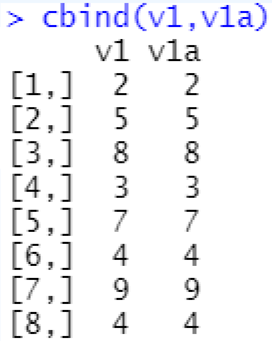


```
> v1
[1] 2 5 8 3 7 4 9 4
```



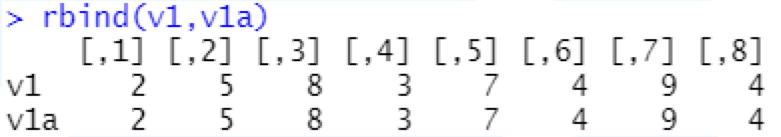
```
> v1a
[1] 2 5 8 3 7 4 9 4
```

```
m1=cbind(v1,v1a) #Put the vectors together as two columns
```



```
> cbind(v1,v1a)
     v1 v1a
[1,]  2   2
[2,]  5   5
[3,]  8   8
[4,]  3   3
[5,]  7   7
[6,]  4   4
[7,]  9   9
[8,]  4   4
```

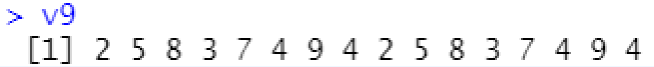
```
m2=rbind(v1,v1a) #Put the vectors together as two rows
```



```
> rbind(v1,v1a)
    [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
v1     2    5    8    3    7    4    9    4
v1a    2    5    8    3    7    4    9    4
```

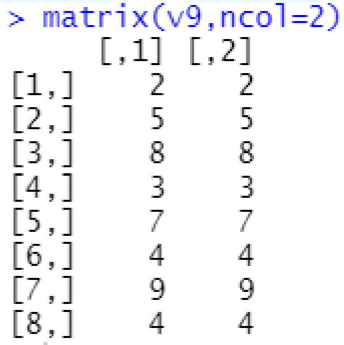
If you start from one long vector of all the values:

```
v9=c(v1,v1) #<v9> is <v1> repeated twice
```



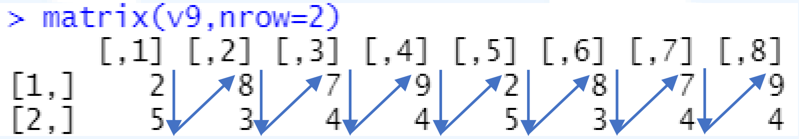
```
> v9
 [1] 2 5 8 3 7 4 9 4 2 5 8 3 7 4 9 4
```

```
m3=matrix(v9,ncol=2)
#uses <v9> to create a matrix with the specified number of columns
```



```
> matrix(v9,ncol=2)
     [,1] [,2]
[1,]    2    2
[2,]    5    5
[3,]    8    8
[4,]    3    3
[5,]    7    7
[6,]    4    4
[7,]    9    9
[8,]    4    4
```

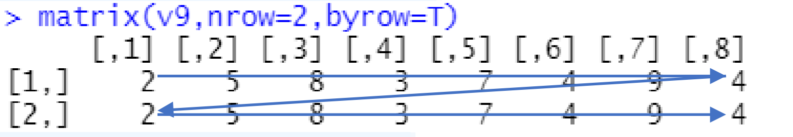
```
m4=matrix(v9,nrow=2) #To create a matrix of 2 rows instead
#not quite right because R fills up the elements top down first
```



```
> matrix(v9,nrow=2)
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
[1,]    2    8    7    9    2    8    7    9
[2,]    5    3    4    4    5    3    4    4
```

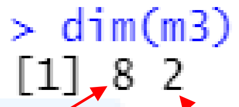
```
m5=matrix(v9,nrow=2,byrow=T)
#this is right!
```

Setting this argument to True tells R to fill the matrix by rows



```
> matrix(v9,nrow=2,byrow=T)
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
[1,]    2    5    8    3    7    4    9    4
[2,]    2    5    8    3    7    4    9    4
```

```
dim(m3) #tells you the number of rows and columns in the matrix
```



```
> dim(m3)
[1] 8 2
```

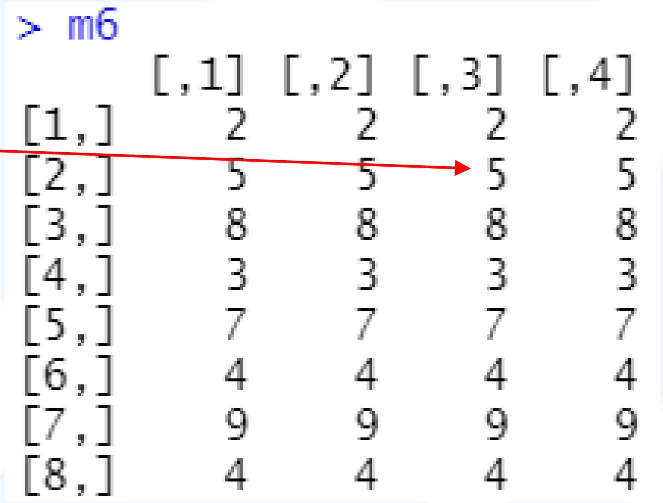
Rows

Columns

Extracting elements and vectors from a matrix

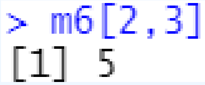
Referring to elements using the notation: [row,column]

```
m6=cbind(m3,m3)
```



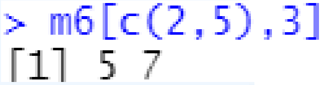
```
> m6
     [,1] [,2] [,3] [,4]
[1,]    2    2    2    2
[2,]    5    5    5    5
[3,]    8    8    8    8
[4,]    3    3    3    3
[5,]    7    7    7    7
[6,]    4    4    4    4
[7,]    9    9    9    9
[8,]    4    4    4    4
```

```
m6[2,3] #Value in Row 2 of Column 3
```



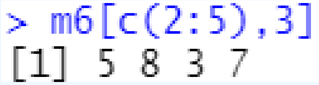
```
> m6[2,3]
[1] 5
```

```
m6[c(2,5),3] #Values in Rows 2 and 5 of Column 3
```



```
> m6[c(2,5),3]
[1] 5 7
```

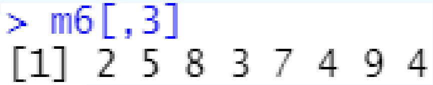
```
m6[c(2:5),3] #Values in Rows 2 to 5 of Column 3
```



```
> m6[c(2:5),3]
[1] 5 8 3 7
```

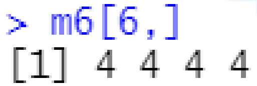
```
#Leaving an argument out tells R that you want ALL rows/columns
```

```
m6[,3] #Values in all rows of Column 3
```



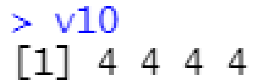
```
> m6[,3]
[1] 2 5 8 3 7 4 9 4
```

```
m6[6,] #Values in all columns of Row 6
```



```
> m6[6,]
[1] 4 4 4 4
```

```
v10=m6[6,] #You can save the rows/columns as new vectors
```

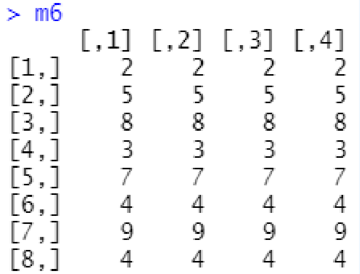


```
> v10
[1] 4 4 4 4
```

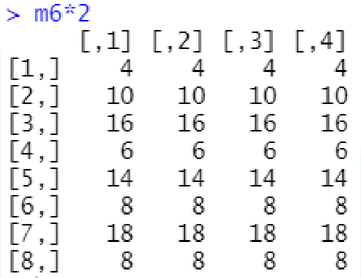
Working with whole matrices

Similar to vectors, you can add to, subtract from, multiply, etc. all elements in a vector at the same time:

```
m7=m6*2
#This multiplies <m6> by 2 and stores it in <m7>
```



```
> m6
     [,1] [,2] [,3] [,4]
[1,]    2    2    2    2
[2,]    5    5    5    5
[3,]    8    8    8    8
[4,]    3    3    3    3
[5,]    7    7    7    7
[6,]    4    4    4    4
[7,]    9    9    9    9
[8,]    4    4    4    4
```



```
> m6*2
     [,1] [,2] [,3] [,4]
[1,]    4    4    4    4
[2,]   10   10   10   10
[3,]   16   16   16   16
[4,]    6    6    6    6
[5,]   14   14   14   14
[6,]    8    8    8    8
[7,]   18   18   18   18
[8,]    8    8    8    8
```

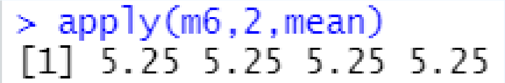
If you want to apply a function by row/column, e.g. calculate the mean of all the columns in <m6>, you could do this manually for one column at a time (tedious), or you could use apply():

```
apply(m6,2,mean)
```

In this matrix

By columns (for rows, use “1” instead)

Calculate the mean. This could instead be another function that works on vectors: e.g. sd, var, median

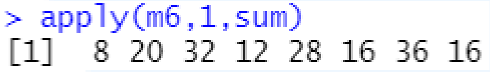


```
> apply(m6,2,mean)
[1] 5.25 5.25 5.25 5.25
```

Mean of column 1

Mean of column 2

```
apply(m6,1,sum) #This calculates the sum of each row
```



```
> apply(m6,1,sum)
[1]  8 20 32 12 28 16 36 16
```

Extracting elements and vectors from a Dataframe

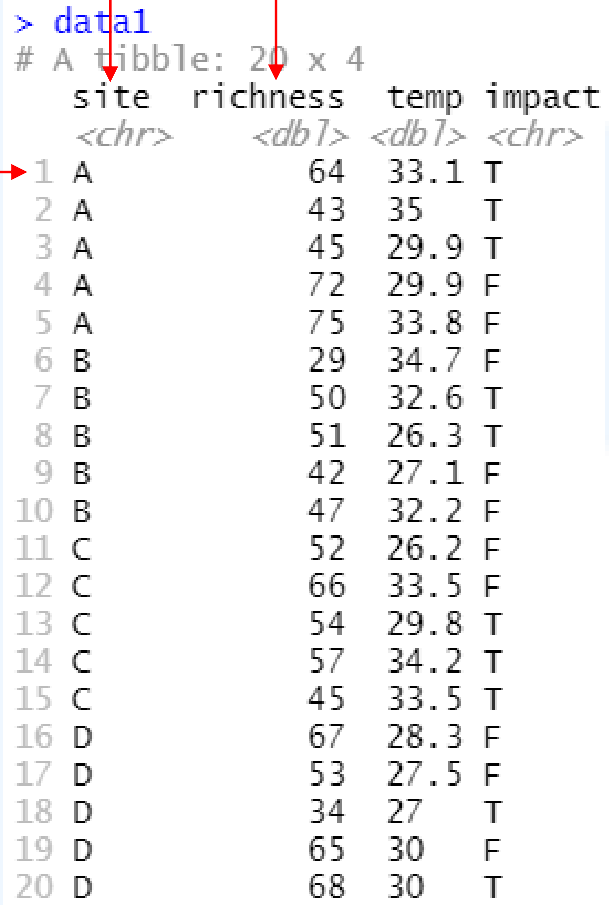
Identical to what you can do with a matrix

```
data1[11,1] #returns C
data1[6,3] #returns 34.7
```

Column 1

Column 2

Row 1

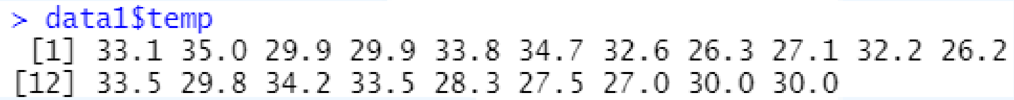


```
> data1
# A tibble: 20 x 4
   site  richness  temp impact
   <chr>    <dbl> <dbl> <chr> 
 1 A           64  33.1 T     
 2 A           43  35   T     
 3 A           45  29.9 T     
 4 A           72  29.9 F     
 5 A           75  33.8 F     
 6 B           29  34.7 F     
 7 B           50  32.6 T     
 8 B           51  26.3 T     
 9 B           42  27.1 F     
10 B           47  32.2 F     
11 C           52  26.2 F     
12 C           66  33.5 F     
13 C           54  29.8 T     
14 C           57  34.2 T     
15 C           45  33.5 T     
16 D           67  28.3 F     
17 D           53  27.5 F     
18 D           34  27   T     
19 D           65  30   F     
20 D           68  30   T     
```

Difference: the columns (variables) now have names

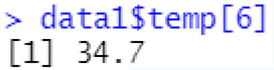
- You can use variable names to refer to columns: “$variable”

```
data1$temp #returns the whole of the <temp> column
```



```
> data1$temp
 [1] 33.1 35.0 29.9 29.9 33.8 34.7 32.6 26.3 27.1 32.2 26.2
[12] 33.5 29.8 34.2 33.5 28.3 27.5 27.0 30.0 30.0
```

```
data1$temp[6] #returns the 6th value in <temp>
```



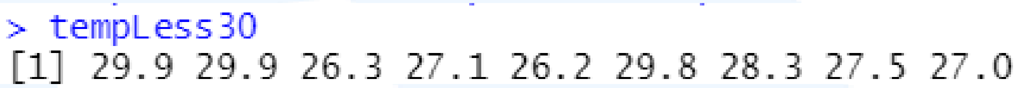
```
> data1$temp[6]
[1] 34.7
```

```
#Can combine this with Logicals to get all values <30

tempLess30=data1$temp[data1$temp<30]
```

1) Give me the position of the elements in <data1$temp> that are <30

2) Give me the values in those elements

3) Store the values in this new object



```
> tempLess30
[1] 29.9 29.9 26.3 27.1 26.2 29.8 28.3 27.5 27.0
```

Create a new variable (column) in a Dataframe

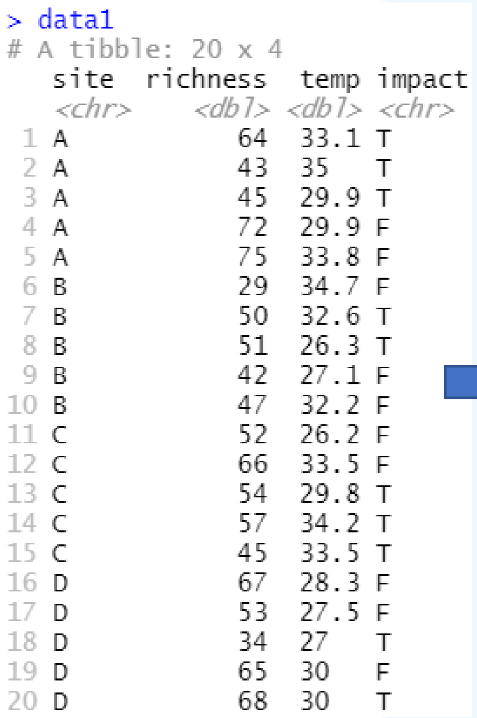
To create a new column, all you have to do is specify a new variable.

Let’s say we want add a new variable called <country> to <data1> with the first 10 values being “X” and the next 10 values being “Y”:

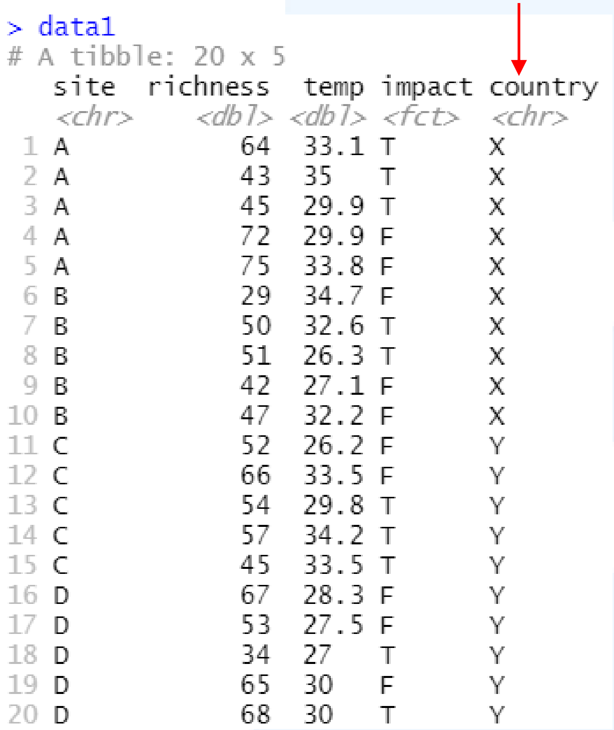
```
cy=c(rep("X",10),rep("Y",10)) #creates a vector <cy> with the specified data
data1$country=cy #This command creates the new variable
```

The dataframe previously did not have a variable called <country>

I’m assigning the values in <cy> to this new variable



```
> data1
# A tibble: 20 x 4
   site  richness  temp impact
   <chr>    <dbl> <dbl> <chr> 
 1 A           64  33.1 T     
 2 A           43  35   T     
 3 A           45  29.9 T     
 4 A           72  29.9 F     
 5 A           75  33.8 F     
 6 B           29  34.7 F     
 7 B           50  32.6 T     
 8 B           51  26.3 T     
 9 B           42  27.1 F     
10 B           47  32.2 F     
11 C           52  26.2 F     
12 C           66  33.5 F     
13 C           54  29.8 T     
14 C           57  34.2 T     
15 C           45  33.5 T     
16 D           67  28.3 F     
17 D           53  27.5 F     
18 D           34  27   T     
19 D           65  30   F     
20 D           68  30   T     
```



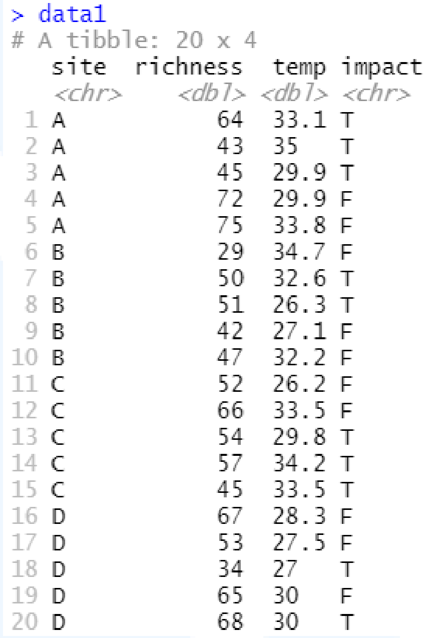
```
> data1
# A tibble: 20 x 5
   site  richness  temp impact country
   <chr>    <dbl> <dbl> <fct>  <chr>  
 1 A           64  33.1 T      X      
 2 A           43  35   T      X      
 3 A           45  29.9 T      X      
 4 A           72  29.9 F      X      
 5 A           75  33.8 F      X      
 6 B           29  34.7 F      X      
 7 B           50  32.6 T      X      
 8 B           51  26.3 T      X      
 9 B           42  27.1 F      X      
10 B           47  32.2 F      X      
11 C           52  26.2 F      Y      
12 C           66  33.5 F      Y      
13 C           54  29.8 T      Y      
14 C           57  34.2 T      Y      
15 C           45  33.5 T      Y      
16 D           67  28.3 F      Y      
17 D           53  27.5 F      Y      
18 D           34  27   T      Y      
19 D           65  30   F      Y      
20 D           68  30   T      Y      
```

<country> added here

Working with Dataframes

With dataframes, we seldom want to calculate the mean for ALL columns because they have different types of variables. Usually we want to calculate the mean (or sd, sum, prod, etc.) of one specific column:

```
mean(data1$richness) #to calculate the mean of richness
```



```
> data1
# A tibble: 20 x 4
   site  richness  temp impact
   <chr>    <dbl> <dbl> <chr> 
 1 A           64  33.1 T     
 2 A           43  35   T     
 3 A           45  29.9 T     
 4 A           72  29.9 F     
 5 A           75  33.8 F     
 6 B           29  34.7 F     
 7 B           50  32.6 T     
 8 B           51  26.3 T     
 9 B           42  27.1 F     
10 B           47  32.2 F     
11 C           52  26.2 F     
12 C           66  33.5 F     
13 C           54  29.8 T     
14 C           57  34.2 T     
15 C           45  33.5 T     
16 D           67  28.3 F     
17 D           53  27.5 F     
18 D           34  27   T     
19 D           65  30   F     
20 D           68  30   T     
```

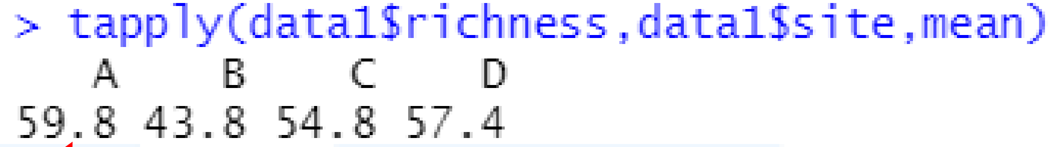
Another thing we often want to do is apply the calculation based on groupings defined by another variable. For example, you want to find the mean <richness> of <site> A, B, C and D separately. You can use tapply():

```
tapply(data1$richness,data1$site,mean)
```

For this variable

Grouped according to this variable

Calculate this



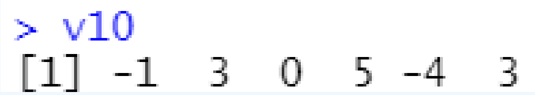
```
> tapply(data1$richness,data1$site,mean)
   A    B    C    D 
59.8 43.8 54.8 57.4 
```

Mean richness for <site> A

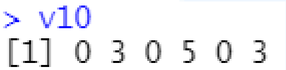
Conditionals

Often, you may want to apply a transformation to certain elements in a vector.

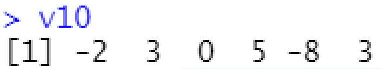
```
v10=c(-1,3,0,5,-4,3)
#I want to change all elements <0 to 0
v10[v10<0]=0
#I want to multiply all elements <0 by 2
v10[v10<0]=v10[v10<0]*2
```



```
> v10
[1] -1  3  0  5 -4  3
```



```
> v10
[1] 0 3 0 5 0 3
```



```
> v10
[1] -2  3  0  5 -8  3
```

Operators in R

For a list of operators (equals, not equal, greater than, less than or equals, etc.): Google, or see HERE.

You may also want to perform a certain command only if a condition is met.

```
if(mean(v10)>5){
  v10=v10*2
}else if(mean(v10)>=0){
  v10=v10*1.5
}else{
  v10=v10+1
}
```

If this condition is met

Do this command

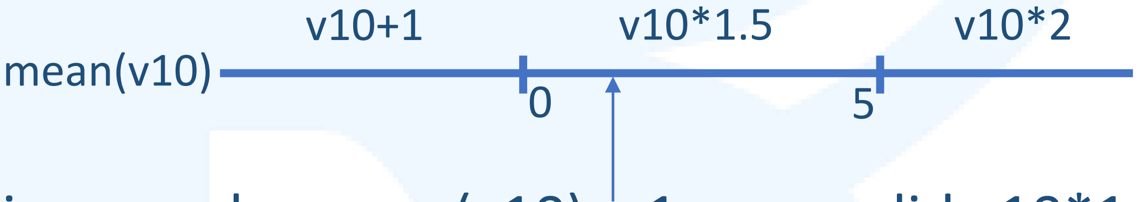
Otherwise, if this condition is met

Do this command

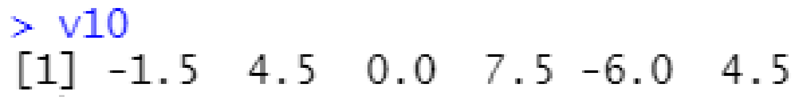
Otherwise

Do this command

What I want to do:



In this example, mean(v10) = 1, so we did v10*1.5:



```
> v10
[1] -1.5  4.5  0.0  7.5 -6.0  4.5
```

Loops

To do things repeatedly (iteration) in R.

- Loops are great but they tend to slow down your code so avoid them if you can (e.g. use apply() or perform transformations on whole vectors/matrices like we learnt previously).

- We will use very little loops in the semester.

2 main types:

1) "While" loops (when you do NOT know how many times you need to do something)

```
While(this condition is true){
      Do these instructions
}
```

This curly bracket denotes the start of the commands to be repeated

This curly bracket denotes the end of the commands to be repeated

2) "For" loops (when you know how many times you need something repeated)

```
For(as long as this condition is true){
      Do these instructions
}
```

“While” loop

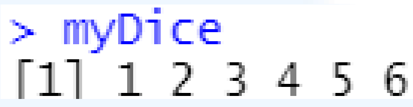
Imagine we want to roll a dice until we get a 6 and see how many times we rolled:

```
myDice=1:6 #First we create the dice
sample(myDice,1,replace=T) #This ‘rolls’ the dice
```

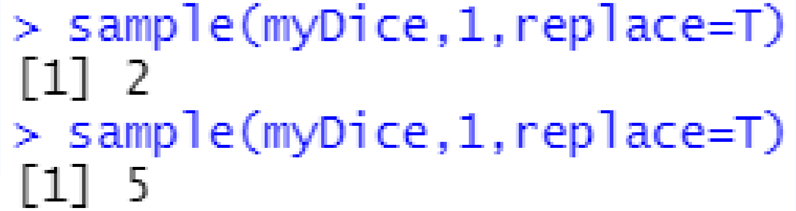
From the values in <myDice>

Choose 1 value

And make the chosen value available again (not really needed for this example)



```
> myDice
[1] 1 2 3 4 5 6
```



```
> sample(myDice,1,replace=T)
[1] 2
> sample(myDice,1,replace=T)
[1] 5
```

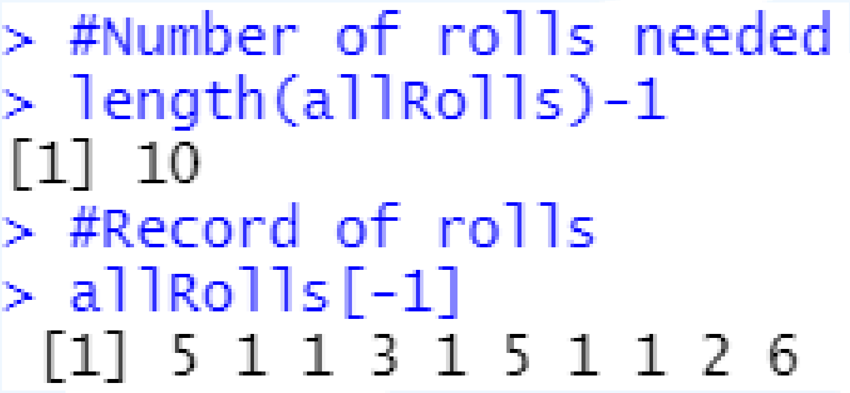
```
#We don’t know how many times to roll, so we use a While loop
allRolls=0
lastRoll=0
while(lastRoll!=6){
  lastRoll=sample(myDice,1)
  allRolls=c(allRolls,lastRoll)
}
#Number of rolls needed
length(allRolls[-1])
#Record of rolls
allRolls[-1]
```

This means not equals

This is the loop: note how the curly brackets are indented. This is to help us see clearer.

Outputs the number of rolls needed to get a 6

Outputs the history of all the rolls



```
> #Number of rolls needed
> length(allRolls)-1
[1] 10
> #Record of rolls
> allRolls[-1]
 [1] 5 1 1 3 1 5 1 1 2 6
```

"While" loop: a closer look…

This vector is used to store the history of all our rolls

As long as <lastRoll> is not 6, do all the commands in the curly brackets

This is used to store the most recent roll

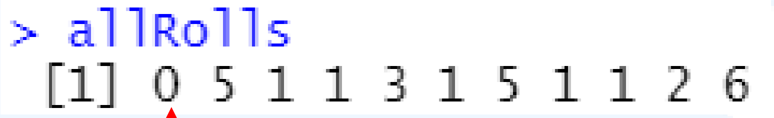
Roll the dice and store the result in <lastRoll>

```
allRolls=0
lastRoll=0
while(lastRoll!=6){
  lastRoll=sample(myDice,1)
  allRolls=c(allRolls,lastRoll)
}
#Number of rolls needed
length(allRolls[-1])
#Record of rolls
allRolls[-1]
```

Goes back to check whether <lastRoll> = 6.
- If not, it carries out the commands again.
- If it is, it exits the loop and does the next instruction after the loop.

Add the most recent roll to the history of rolls

Outputs how many elements there are in our history of rolls (minus 1)…

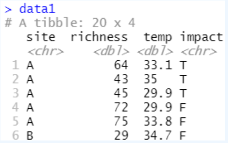


```
> allRolls
 [1] 0 5 1 1 3 1 5 1 1 2 6
```

We needed this 0 to start the vector. Now, to display the results, we remove it using [-1]. (There is another way of doing this but it involves using lists which we haven't learnt.)

Outputs the whole history without the first element

“For” loop



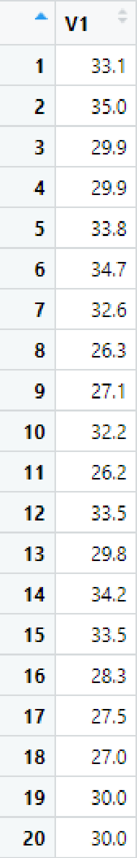
```
> data1
# A tibble: 20 x 4
  site  richness  temp impact
  <chr>    <dbl> <dbl> <chr>
1 A           64  33.1 T
2 A           43  35   T
3 A           45  29.9 T
4 A           72  29.9 F
5 A           75  33.8 F
6 B           29  34.7 F
```

From <data1>, we want to model future temperatures over the next 10 years based on current <temp>, assuming a 1% increase per year.

Creates a 20 by 10 matrix of 0s

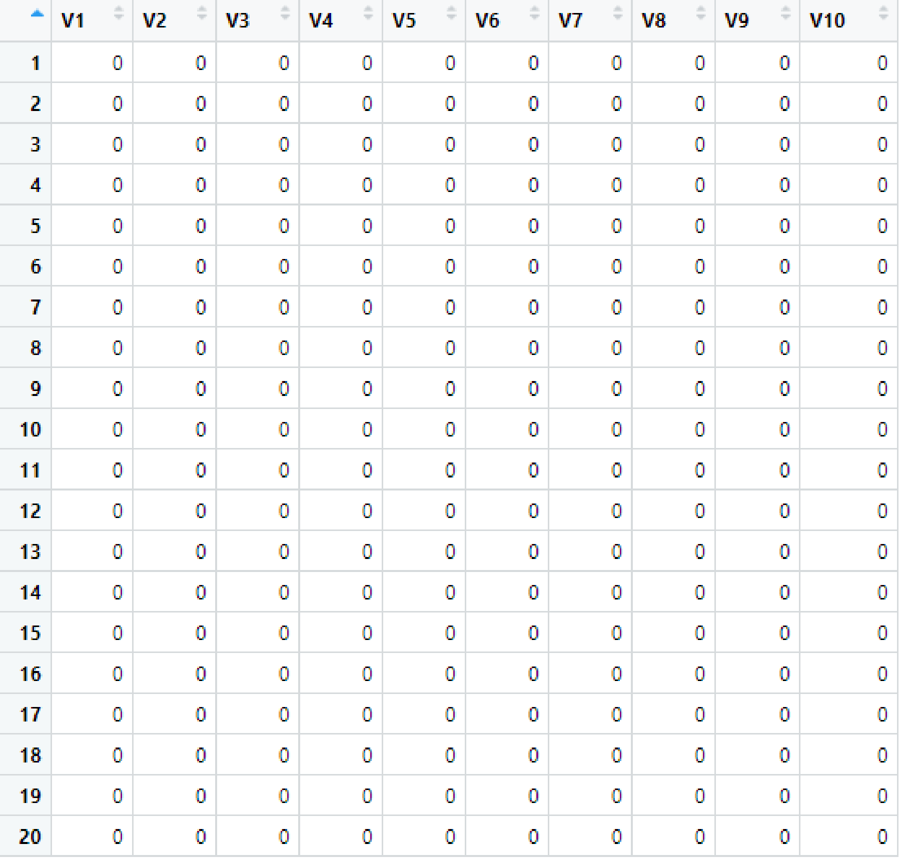
```
#Create a matrix to store the data
futureTemps=matrix(rep(0,20*10),nrow=20)
futureTemps=cbind(data1$temp,futureTemps)
```

Put another column with current temperatures in front



	V1
1	33.1
2	35.0
3	29.9
4	29.9
5	33.8
6	34.7
7	32.6
8	26.3
9	27.1
10	32.2
11	26.2
12	33.5
13	29.8
14	34.2
15	33.5
16	28.3
17	27.5
18	27.0
19	30.0
20	30.0

+



	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0

```
#Then we could do this manually
futureTemps[,2]=futureTemps[,1]*1.01
futureTemps[,3]=futureTemps[,2]*1.01
futureTemps[,4]=futureTemps[,3]*1.01
futureTemps[,5]=futureTemps[,4]*1.01
futureTemps[,6]=futureTemps[,5]*1.01
… … … …
futureTemps[,11]=futureTemps[,10]*1.01
```

```
#Or, since we know we need to do this 10
times, we could use a For loop:
for(i in 1:10){
  futureTemps[,i+1]=futureTemps[,i]*1.01
}
```

"For" loop: a closer look

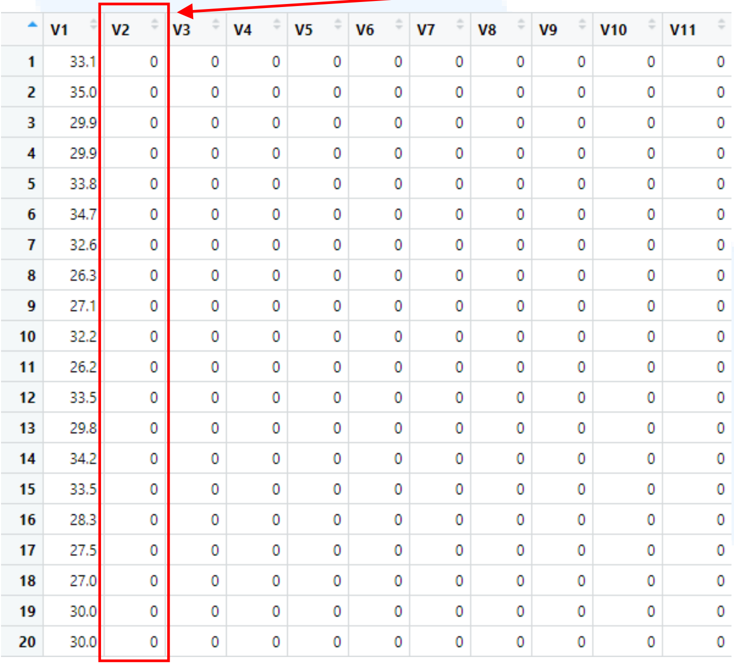
```
#The loop
for(i in 1:10){
   futureTemps[,i+1]=futureTemps[,i]*1.01
}
```

Creates a dummy variable <i> that starts at the first number, increases by 1 each time it goes through the loop, and stops when it is greater than the second number.

As long as <i> is between these two numbers (inclusive), the commands in the loop will be carried out

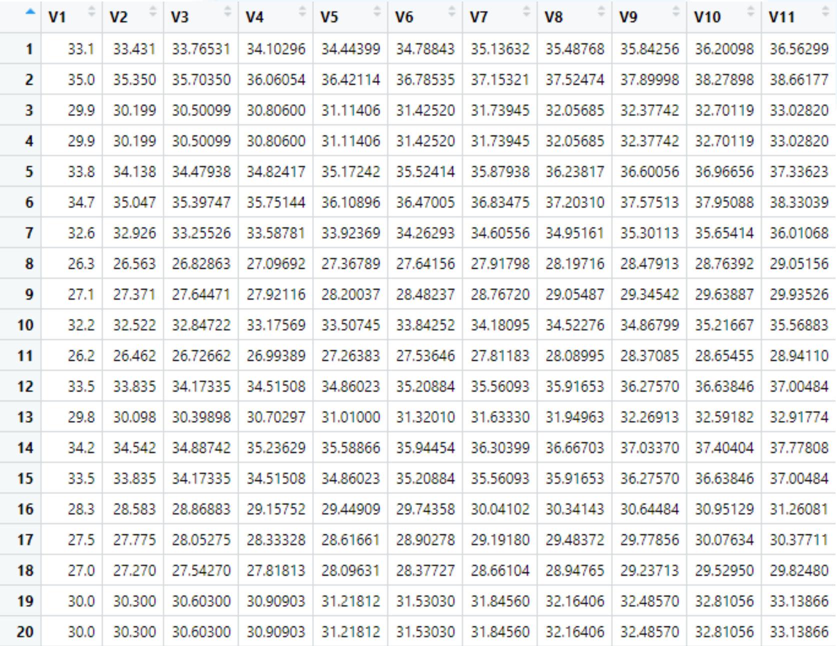
Starts at column 2. As <i> increases, it will move to column 3→4→5→etc.

Increases the values from the previous column by 1%



	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11
1	33.1	0	0	0	0	0	0	0	0	0	0
2	35.0	0	0	0	0	0	0	0	0	0	0
3	29.9	0	0	0	0	0	0	0	0	0	0
4	29.9	0	0	0	0	0	0	0	0	0	0
5	33.8	0	0	0	0	0	0	0	0	0	0
6	34.7	0	0	0	0	0	0	0	0	0	0
7	32.6	0	0	0	0	0	0	0	0	0	0
8	26.3	0	0	0	0	0	0	0	0	0	0
9	27.1	0	0	0	0	0	0	0	0	0	0
10	32.2	0	0	0	0	0	0	0	0	0	0
11	26.2	0	0	0	0	0	0	0	0	0	0
12	33.5	0	0	0	0	0	0	0	0	0	0
13	29.8	0	0	0	0	0	0	0	0	0	0
14	34.2	0	0	0	0	0	0	0	0	0	0
15	33.5	0	0	0	0	0	0	0	0	0	0
16	28.3	0	0	0	0	0	0	0	0	0	0
17	27.5	0	0	0	0	0	0	0	0	0	0
18	27.0	0	0	0	0	0	0	0	0	0	0
19	30.0	0	0	0	0	0	0	0	0	0	0
20	30.0	0	0	0	0	0	0	0	0	0	0

with one line of code



	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11
1	33.1	33.431	33.76531	34.10296	34.44399	34.78843	35.13632	35.48768	35.84256	36.20098	36.56299
2	35.0	35.350	35.70350	36.06054	36.42114	36.78535	37.15321	37.52474	37.89998	38.27898	38.66177
3	29.9	30.199	30.50099	30.80600	31.11406	31.42520	31.73945	32.05685	32.37742	32.70119	33.02820
4	29.9	30.199	30.50099	30.80600	31.11406	31.42520	31.73945	32.05685	32.37742	32.70119	33.02820
5	33.8	34.138	34.47938	34.82417	35.17242	35.52414	35.87938	36.23817	36.60056	36.96656	37.33623
6	34.7	35.047	35.39747	35.75144	36.10896	36.47005	36.83475	37.20310	37.57513	37.95088	38.33039
7	32.6	32.926	33.25526	33.58781	33.92369	34.26293	34.60556	34.95161	35.30113	35.65414	36.01068
8	26.3	26.563	26.82863	27.09692	27.36789	27.64156	27.91798	28.19716	28.47913	28.76392	29.05156
9	27.1	27.371	27.64471	27.92116	28.20037	28.48237	28.76720	29.05487	29.34542	29.63887	29.93526
10	32.2	32.522	32.84722	33.17569	33.50745	33.84252	34.18095	34.52276	34.86799	35.21667	35.56883
11	26.2	26.462	26.72662	26.99389	27.26383	27.53646	27.81183	28.08995	28.37085	28.65455	28.94110
12	33.5	33.835	34.17335	34.51508	34.86023	35.20884	35.56093	35.91653	36.27570	36.63846	37.00484
13	29.8	30.098	30.39898	30.70297	31.01000	31.32010	31.63330	31.94963	32.26913	32.59182	32.91774
14	34.2	34.542	34.88742	35.23629	35.58866	35.94454	36.30399	36.66703	37.03370	37.40404	37.77808
15	33.5	33.835	34.17335	34.51508	34.86023	35.20884	35.56093	35.91653	36.27570	36.63846	37.00484
16	28.3	28.583	28.86883	29.15752	29.44909	29.74358	30.04102	30.34143	30.64484	30.95129	31.26081
17	27.5	27.775	28.05275	28.33328	28.61661	28.90278	29.19180	29.48372	29.77856	30.07634	30.37711
18	27.0	27.270	27.54270	27.81813	28.09631	28.37727	28.66104	28.94765	29.23713	29.52950	29.82480
19	30.0	30.300	30.60300	30.90903	31.21812	31.53030	31.84560	32.16406	32.48570	32.81056	33.13866
20	30.0	30.300	30.60300	30.90903	31.21812	31.53030	31.84560	32.16406	32.48570	32.81056	33.13866

Help in R

How do I know what arguments a function needs?

- Look for built-in help in R

```
help(mean) #can be a function or a package
?mean #opens documentation in R
??mean #used if there is no documentation in R, opens from the web
```

- Vignettes (search the function name and “R vignette”)

I don’t even know what function I need!

- https://stackoverflow.com
- http://cran.r-project.org
- Google (especially error messages)

Summary (Learning Objectives)

What is R?

- How to analyse data in R

Installing R

Starting an R Project

- Working folder
- R scripts and syntax
- Installing packages

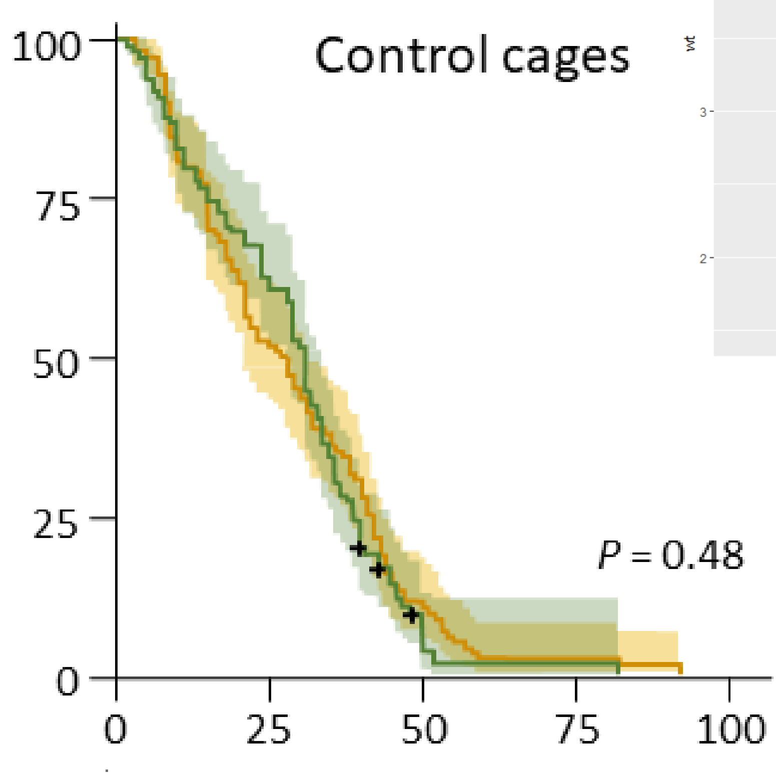
Importing datasets in R: read_xl(), read.table(), read.csv()

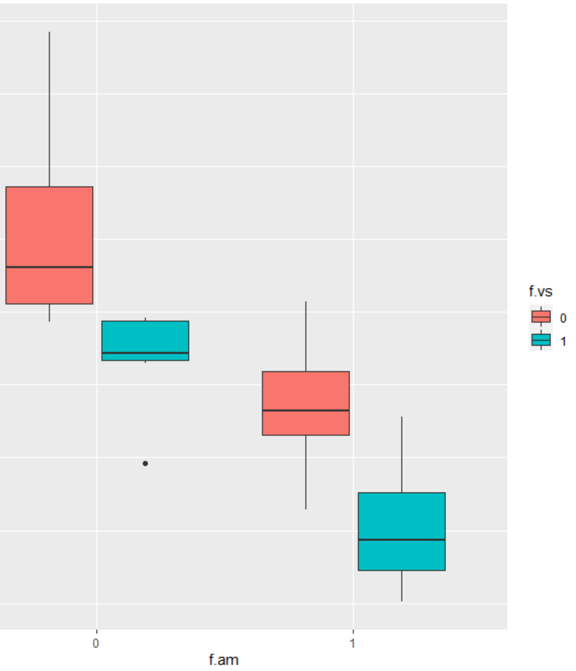
Preparing your data

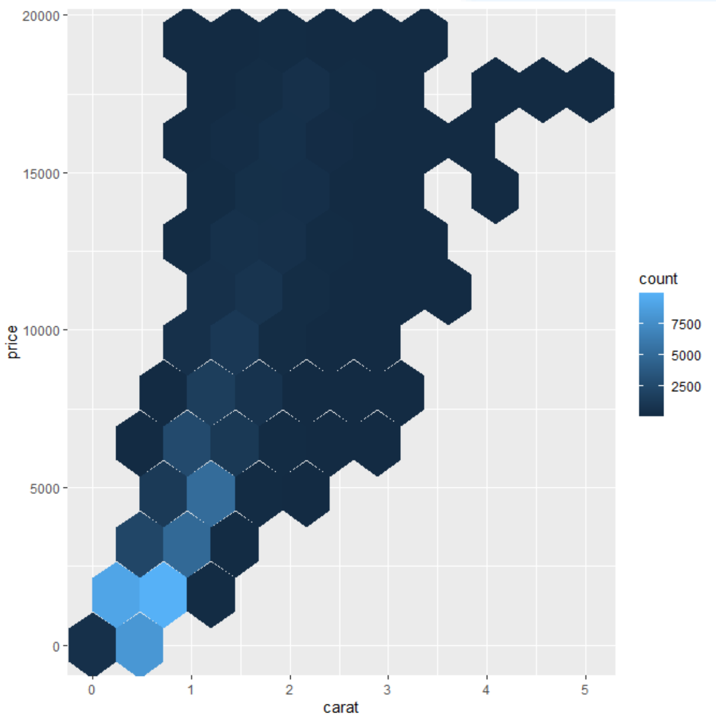
- Calculator
- Functions
- Objects: Vectors, Matrices, Dataframes
- Conditionals and loops

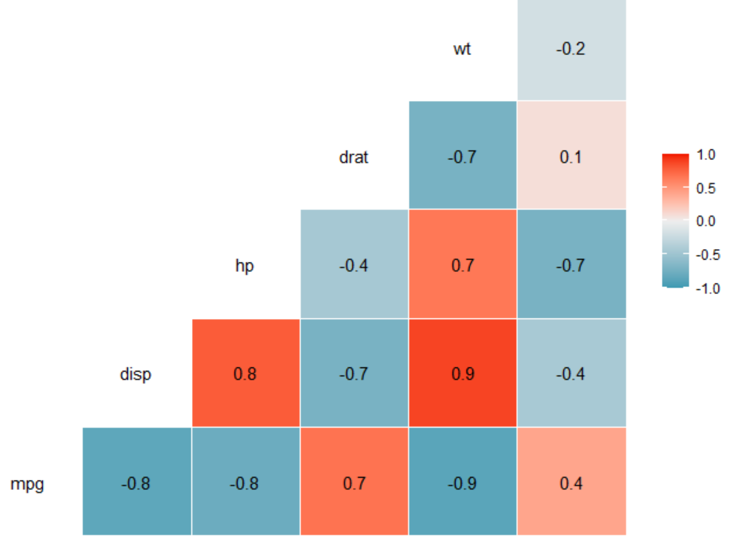
Next week…

Data exploration and visualisation!









Module Information

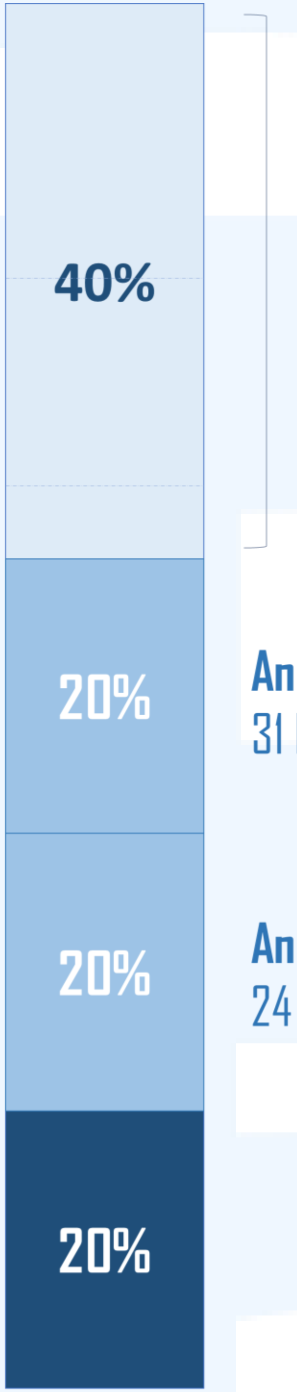
Lecture 1

LSM3257

AY22/23; Sem 2 | Ian Z.W. Chan

Assessments

100% CA



Report (Individual): 20%
28 Apr 2023, 2359h

Presentation (Group): 15%
3, 10 Apr 2023†
(Submit slides by 2359h the day before your presentation)

Proposal (Group): 5%
12 Feb 2023, 2359h

Project

Analysis Challenge 2
31 Mar 2023, 2359h

Analysis Challenge 1
24 Feb 2023, 2359h

Take home assignments:

- Similar to HEx but more challenging

- Open everything but please no discussion

Hands-On Exercises (HEx)
Throughout the course, see Schedule and Readings for specific deadlines

Assignments to practise what you learn in class

Full details in Module Guide

Hands-On Exercises (HEx) (20%)

Weeks 1, 2, 3, 6, 7 and 8 only.

Download the questions (a .R file) from Canvas before lecture. Start attempting during lecture (we will help). Continue doing at home. Submit the completed .R file on Canvas by the deadline (1 week after the class).

If pictures/plots are asked for, paste all of them onto a Word document (will be provided) and submit it too.

Marked only based on whether you’ve genuinely attempted the question, NOT on whether the answer is correct or not.

- Marks in the assignment are just indicative of how complicated the answer is

See Module Overview pdf for more details.

Analysis Challenges (20% x 2)

2 challenges (instead of 2 mid-terms from last year)

- Challenge 1: covers Weeks 1–6; released Week 6, submit by 24 Feb 2359h (2 weeks)
- Challenge 2: covers mainly Weeks 7–10; released Week 10, submit by 31 Mar 2359h (2 weeks)

Similar to the HEx but more difficult

Open universe (to make it as authentic as possible)

But please do not discuss with your friends

- This will force you to learn it for yourself so you actually benefit from the assignment

Project (35%)

In groups of 3 to 4…

Find a dataset, choose variables, formulate a research question, analyse the data then present and write a concise research article.

- Ideally, the dataset would be related to your work or research
- From supervisor/colleagues: datasets which have not yet been analysed
- Data from international agencies or published papers in open repositories (e.g. http://datadryad.org/)

1) Submit a **Proposal (5%)** as a <u>group</u>: describe background (why is your question important), the research question you came up with and the dataset to be used. Similar to an Abstract but without results and conclusions. ≤ 300 words. Due on 12 Feb 2359h.

- Refer to research papers to see how Abstracts are written
- This is mainly for me to give you feedback and make sure you're on the right track

Individual Project (35%)

2) **Presentation (15%)** as a group: describe background, research question, data used AND results and conclusions. 10 minutes followed by Q&A. Weeks 12–13 (come on the week you're presenting). Submit slides the day before your slot.

- Use nice graphics and tell an interesting story that laymen can understand

3) **Report (20%)** individually: describe background, research question, data used, and results and conclusions. ≤ ~2500 words. Due on 28 Apr 2359h.

- Refer to research papers to see how they write. If you need help with scientific writing, please email me and we will advise you.

See the Module Overview pdf for full details: refer to the rubrics often!!

Teammate grading is available as a last resort. My email/door is always open.

Key Values

Honesty

- We do not plagiarise: i.e. take existing work (even your own) and pass it off as new work for this module. When in doubt, cite your sources (even yourself!)
    - https://www.nus.edu.sg/celc/programmes/plagiarism.html
- We do not copy others’ work

Respecting others

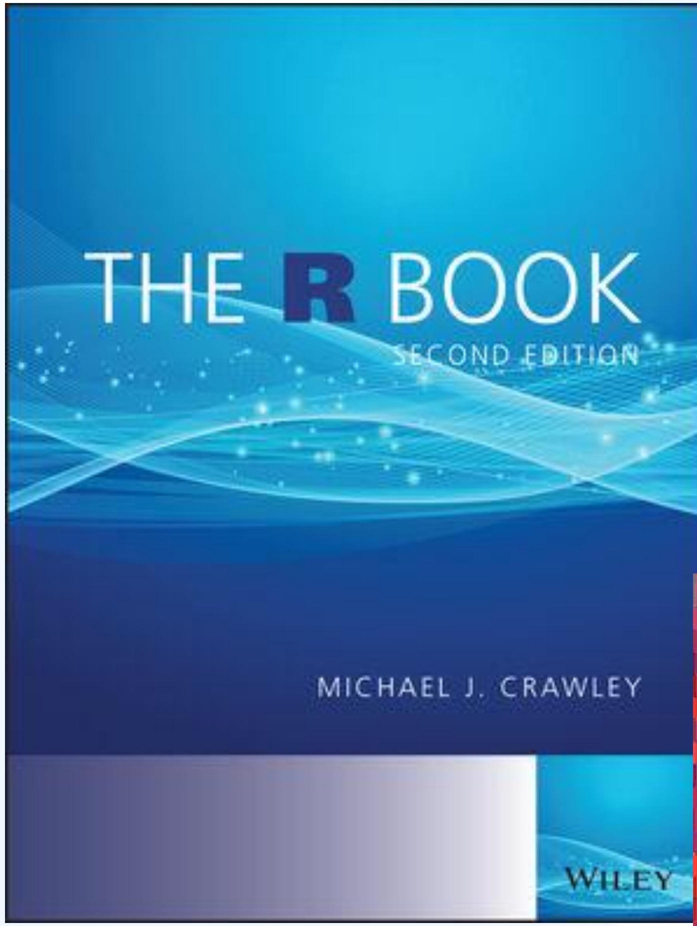
- We will foster a safe and positive learning environment: NO discrimination, NO personal attacks, critique the idea NOT the person
- We are responsible for our own learning
- We will try to help one another along the way

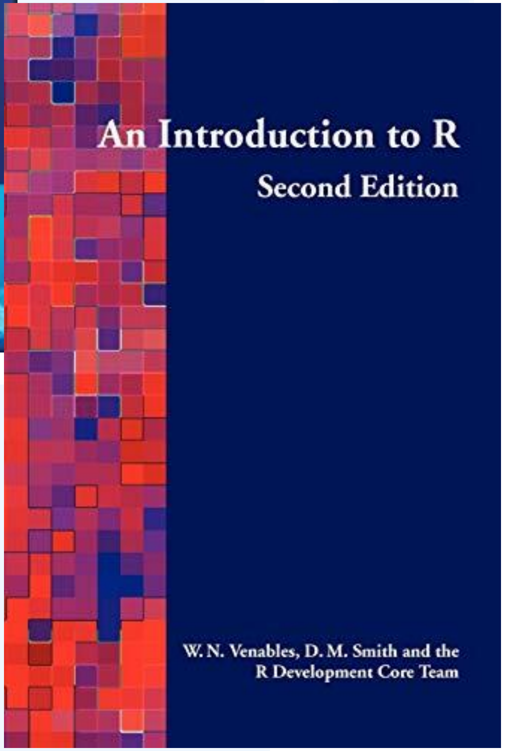
Readings

Our 2 "textbooks" are:

- The R Book (main), and
- Introduction to R (supplementary).

There are also links to videos for some weeks in the Module Overview (under Schedule and Readings).





Schedule

Wk	Date	Lectures / **Activities** / ***Assignments***	Readings
1	9 Jan 23	Lecture 1: Basic Data Handling **Project Preparation** ***HEx 1 (due 15 Jan 2359h)***	R Book: Ch 1 Intro to R: Ch 2 & 9
2	16 Jan 23	Lecture 2: Data Exploration & Visualisation ***HEx 2 (due 25 Jan 2359h)***	R Book: Ch 3.2, 4.1 to 4.4, 5.2, 5.6 & 5.8 Intro to R: Section 6, 7 & 12
3	23 Jan 23	No Class: Public Holiday	
4	30 Jan 23	Lecture 3: Basic Statistical Concepts & Tests ***HEx 3 (due 5 Feb 2359h)***	R Book: Ch 8 Watch: Probability Distributions, Parametric vs. Non-Parametric Tests, Power Analysis
5	6 Feb 23	Lecture 4: Regression ***Project Proposal (due 12 Feb 2359h)***	R Book: Ch 10 Intro to R: Section 11.1–11.5
6	13 Feb 23	Lecture 5: ANOVA & ANCOVA	R Book: Ch 11 & 12
Recess Week ***Analysis Challenge 1 (due 24 Feb 2359h)***			
7	27 Feb 23	Lecture 6: GLS & LME ***HEx 6 (due 5 Mar 2359h)***	R Book: Ch 19.1–19.9
8	6 Mar 23	Lecture 7: GLM ***HEx 7 (due 12 Mar 2359h)***	R Book: Ch 13.1 to 13.8, 14.1 & 16.1 Intro to R: Section 11.6
9	13 Mar 23	Lecture 8: GLMM & Survival Analysis ***HEx 8 (due 19 Mar 2359h)***	R Book: Ch 29 & 19.10
10	20 Mar 23	Lecture 9: Multivariate Stats, GAM & Bayesian Stats	R Book: Ch 18, 22 & 25 Watch: Multivariate Statistics, Bayesian Stats A & B
11	27 Mar 23	No Class: **Project Consultations** ***Analysis Challenge 2 (due 31 Mar 2359h)***	-
12	3 Apr 23	***Slides (due 2 Apr 2359h)*** ***Project Presentation Seminar***	-
13	10 Apr 23	***Slides (due 2 Apr 2359h)*** ***Project Presentation Seminar***	-
Reading and Exam Weeks ***Project Report (due 28 Apr 2359h)***			

Project Preparation

Group Presentations

1) Form groups of 4 (ideally) or 3 (if you really must)

2) Indicate your groupings in Canvas>BL5233>People>Groups

3) If you don’t have a group by next week, I will assign

Later on: we will assign presentation slots in Weeks 12 and 13