



How to use R Statistics

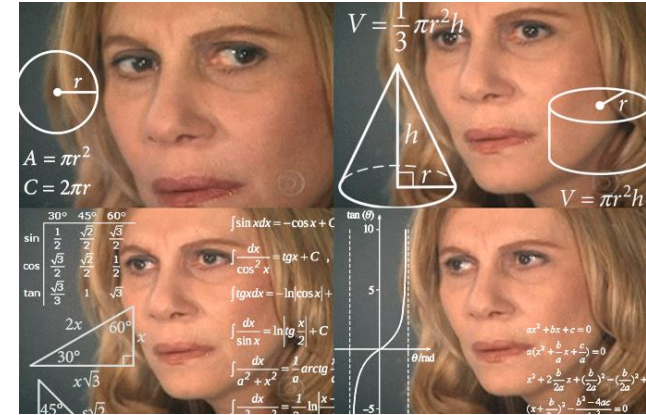
Cognition and Emotion Group

School of Psychology

Curtin University

Why we here?

- I want to learn R...
 - but I don't know where to start
 - but I got stuck and gave up
 - but it takes too much time to learn
 - but I can't write scripts and coding scares me



Reasons to learn

- It can do all the stats, for free!
- And it's not limited to stats.
 - Pre-process and organise raw data (less copy and pasting)
 - Make nice plots
- Growing popularity
 - Feel included
 - Promotes data and analysis sharing
- You also learn programming
 - Transferable skill – dealing with variables, using functions, for loops.

Purpose of the Series

- Make R accessible.
- Make it practical.
- Keep content simple and use relevant examples.
- Create opportunity to ask questions and work through problems.

Disclaimer: It's my first time. Nnot a programming, statistics or teaching expert.

Session 1: Objectives

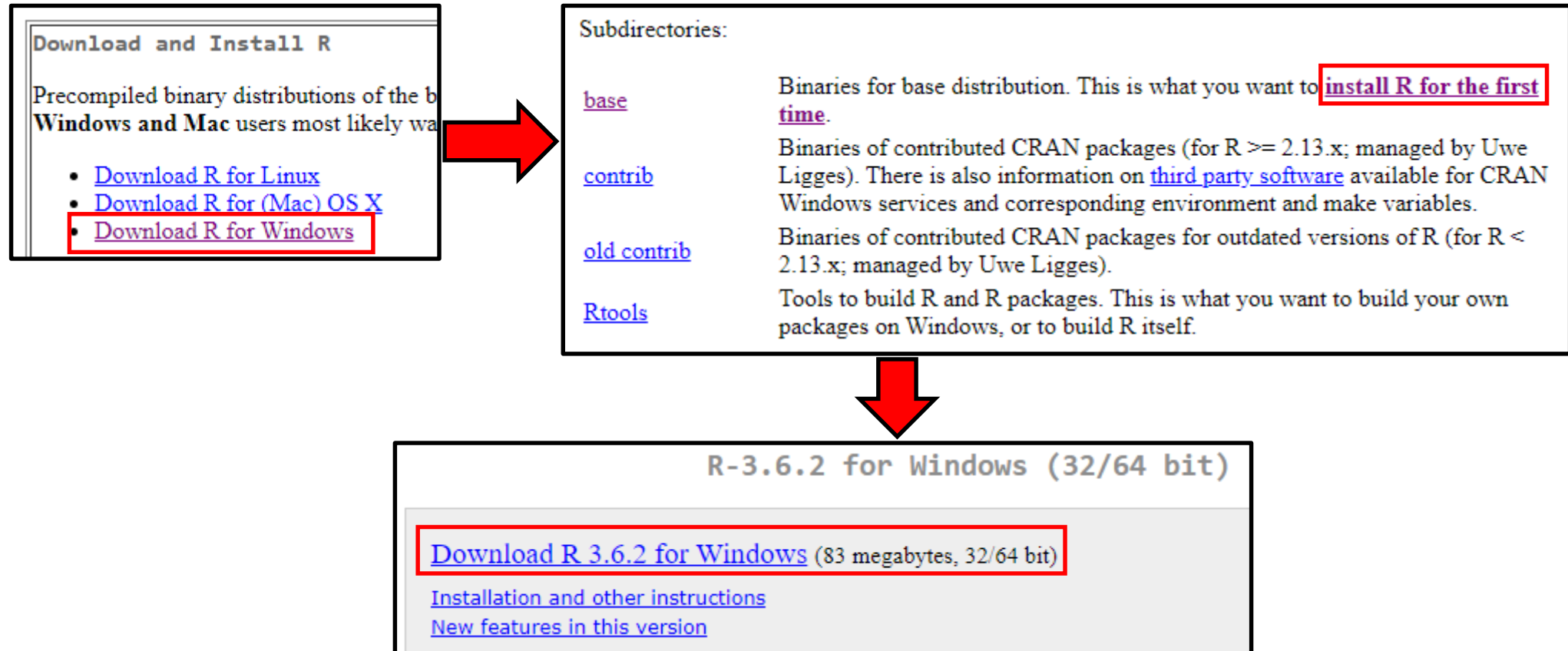
- Install R and R Studio
- Learn how to navigate the program
- Open an Excel data file
- Learn how to transform data between wide- to long-format
- Save data to Excel file

Download Tutorial Materials from GitHub

- https://github.com/angnguyen/ce_group
- Clone or download > download .zip
- Contains sample data, sample code, power-point slides.
 - Open slides to go at your own pace.
 - Use for future reference

Install R (Windows)

Go here: <https://cran.curtin.edu.au/>



Install R (Mac OS)

- Download and run **R-3.6.2.pkg** (or most recent version)
- Install with default settings

Install R Studio (Windows)

Go here: <https://rstudio.com/products/rstudio/download/#download>

RStudio Desktop 1.2.5033 - [Release Notes](#)

1. Install R. RStudio requires R 3.0.1+.
2. Download RStudio Desktop. Recommended for your system:



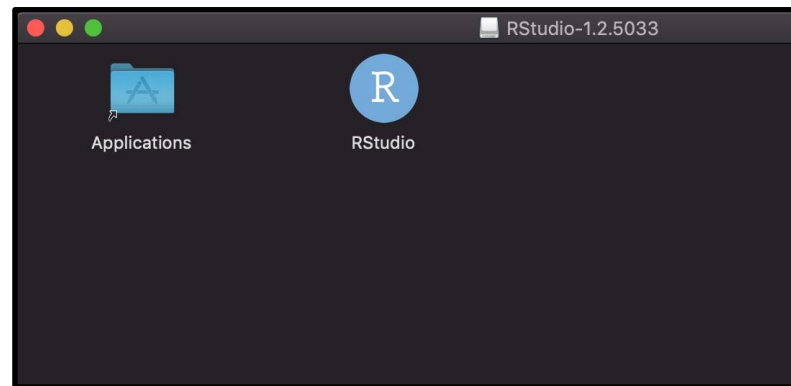
DOWNLOAD RSTUDIO FOR WINDOWS

1.2.5033 | 149.83MB

Requires Windows 10/8/7 (64-bit)

Install R Studio (Mac OS)

- Run the **.dmg** file
- Drag the RStudio icon to your “Applications” folder



Open R Studio

The screenshot displays the RStudio application window. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. Below the menu is a toolbar with icons for file operations and running code. The main editor pane shows an R script with the following content:

```
1 #####
2 ### R Tutorial - 1 ###
3 #####
4
5 # Code for R Tutorial 1 by An Nguyen and Aaron McInnes for Cognition and Emotion Group
6 # Disclaimer: We are not teaching or coding experts
7
8 rm(list = ls()) # clears environment
9 cat("\014") # clears console
10
11
12 1+1
13
14 hello <- 50
15 there <- 20
16 hello + there
17
18 var1 <- "hello"
19 var1
20
21 #####
22 ### R Tutorial - How to load an excel file ###
23 #####
24
25 rm(list = ls()) # clears environment
26 cat("\014") # clears console
27
28
29 library(openxlsx) # loads package with functions to open excel files - DOCUMENTATION: https://cran.r-project.org/web/packages/openxlsx/op
30
31
32 data_folder <- "E:/Cognition_emotion_group_2020/R_tutorial_1/data_is_here/" # define folder where the data file lives - NOTE: use forward
33 data_file <- "example_data_single_sheet.xlsx" # define the file name
34 full_file_path <- paste0(data_folder,data_file) # combine folder path and file name to get the full file path
35
```

The console pane at the bottom shows the output of the script execution:

```
~/RStudio
~/RStudio
~/RStudio
[workspace loaded from ~/.RData]
Loading required package: lmerTest
Loading required package: lme4
Loading required package: Matrix
Attaching package: 'lmerTest'
The following object is masked from 'package:lme4':
  lmer
The following object is masked from 'package:stats':
  step
> |
```

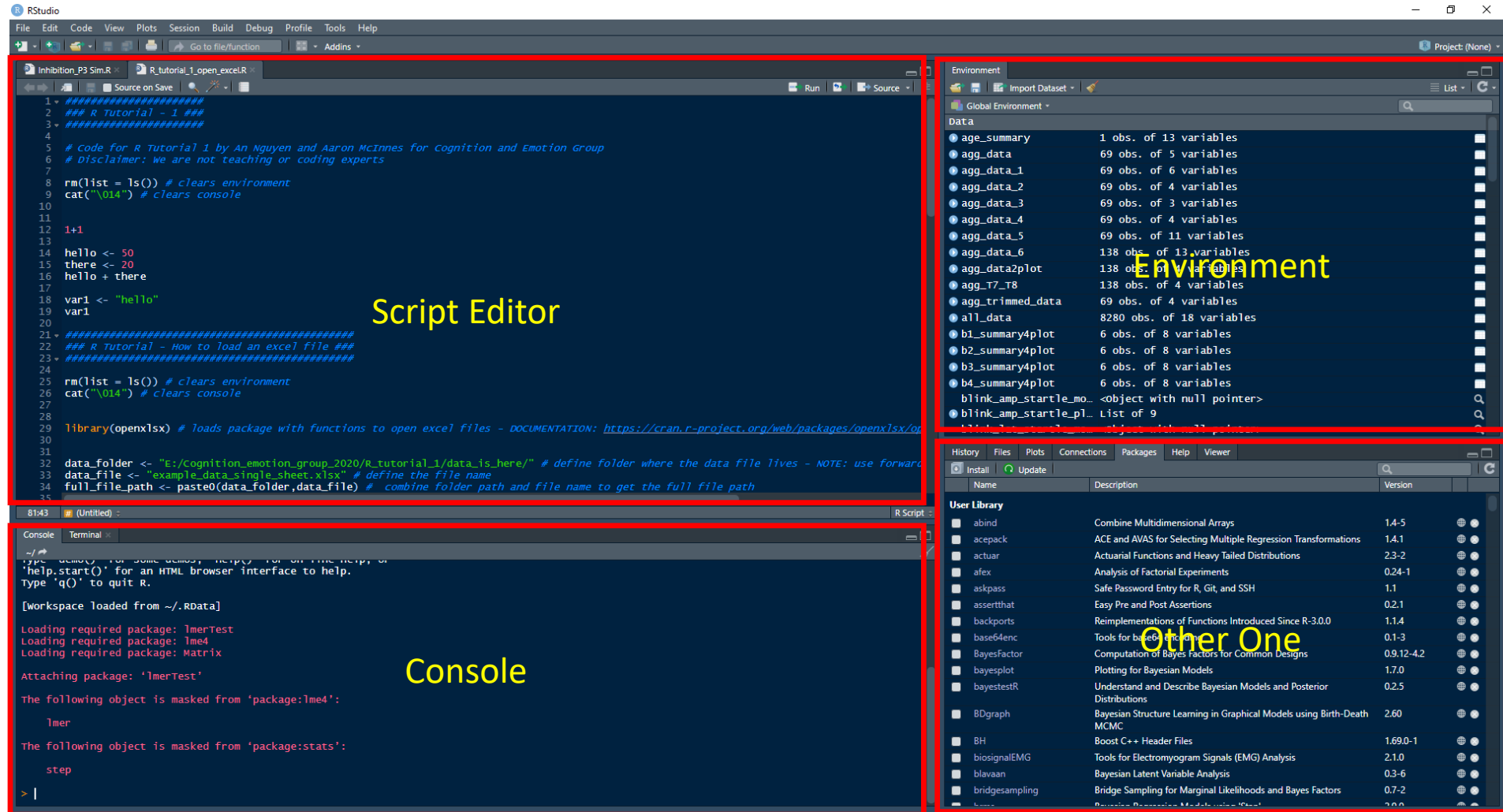
The Environment pane on the right shows the Global Environment with the following objects:

Object	Description
age_summary	1 obs. of 13 variables
agg_data	69 obs. of 5 variables
agg_data_1	69 obs. of 6 variables
agg_data_2	69 obs. of 4 variables
agg_data_3	69 obs. of 3 variables
agg_data_4	69 obs. of 4 variables
agg_data_5	69 obs. of 11 variables
agg_data_6	138 obs. of 13 variables
agg_data2plot	138 obs. of 4 variables
agg_T7_T8	138 obs. of 4 variables
agg_trimmed_data	69 obs. of 4 variables
all_data	8280 obs. of 18 variables
b1_summary4plot	6 obs. of 8 variables
b2_summary4plot	6 obs. of 8 variables
b3_summary4plot	6 obs. of 8 variables
b4_summary4plot	6 obs. of 8 variables
blink_amp_startle_mo...	<object with null pointer>
blink_amp_startle_pl...	List of 9
blink_lat_startle_mo...	<object with null pointer>

The Packages pane at the bottom right shows the User Library with the following packages:

Name	Description	Version
abind	Combine Multidimensional Arrays	1.4-5
acepack	ACE and AVAS for Selecting Multiple Regression Transformations	1.4.1
actuar	Actuarial Functions and Heavy Tailed Distributions	2.3-2
afex	Analysis of Factorial Experiments	0.24-1
askpass	Safe Password Entry for R, Git, and SSH	1.1
assertthat	Easy Pre and Post Assertions	0.2.1
backports	Reimplementations of Functions Introduced Since R-3.0.0	1.1.4
base64enc	Tools for base64 encoding	0.1-3
BayesFactor	Computation of Bayes Factors for Common Designs	0.9.12-4.2
bayesplot	Plotting for Bayesian Models	1.7.0
bayestestR	Understand and Describe Bayesian Models and Posterior Distributions	0.2.5
BDgraph	Bayesian Structure Learning in Graphical Models using Birth-Death MCMC	2.60
BH	Boost C++ Header Files	1.69.0-1
biosignalEMG	Tools for Electromyogram Signals (EMG) Analysis	2.1.0
blavaan	Bayesian Latent Variable Analysis	0.3-6
bridgesampling	Bridge Sampling for Marginal Likelihoods and Bayes Factors	0.7-2

Navigating the Interface



Navigating the Interface

- **Script Editor** (*only appears if you have a script open*)
 - Write and save R scripts
- **Console**
 - Type in and run commands
 - Output shows up here
- **Environment**
 - Existing variables and data show up here
- **Other One**
 - Plots show up here and Install packages

Customise the look of R Studio

- Tools > Global Options > Appearance
- Go nuts – I like Cobalt

Using the Console

- Use it like a calculator – output is *not* saved

```
> 1+1      > (50+100+60+200)*4      > mean( c(2,2,5,5,7,8,9) )  
[1] 2      [1] 1640                      [1] 5.428571
```

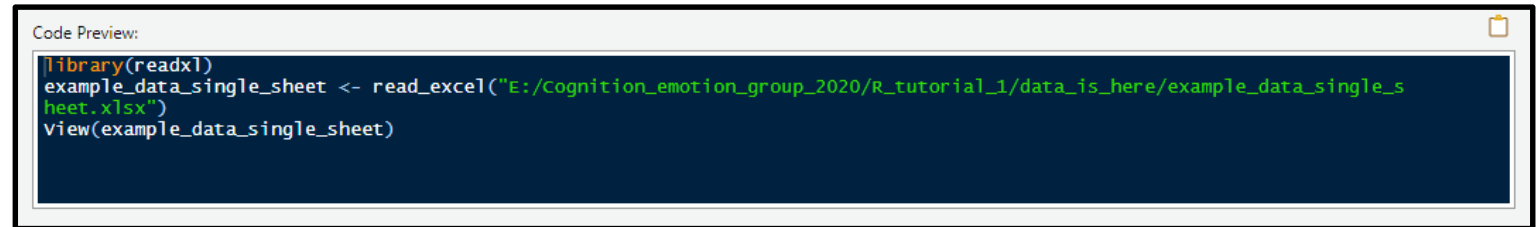
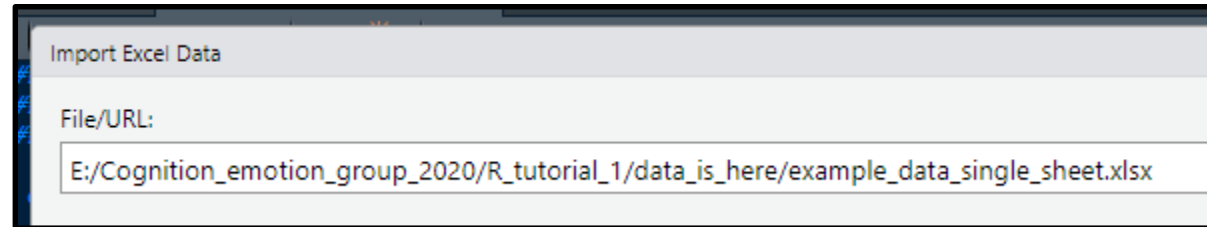
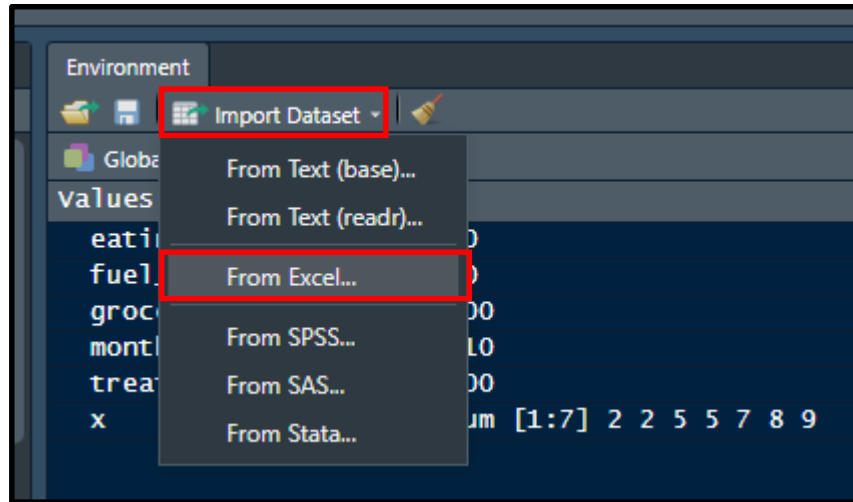
- `c()` stands for *concatenate* – use it to group things

- Output can be saved into **Variables**

```
> monthly_spend <- (50+100+60+200)*4  
>
```

- Saved in the environment (*goes away when you close R*)
- Makes code understandable
- Makes it easier to adjust values in code
 - Not just for numbers but also words
 - “**Alt + minus**” to get the arrow

Importing Excel data manually

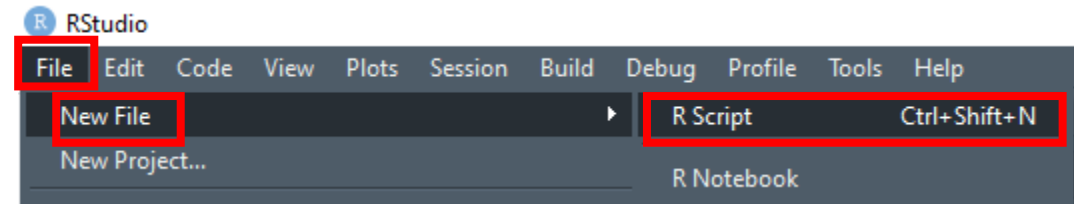


- It's loading the ***readxl*** package from library
 - What is a package? Add-on with additional functions
 - Ok and what is a function? Mini-program that does a specific task
- Using ***read_excel()*** function to open the file
- Saves data to variable called '***example_data_single_sheet***'

But we want to do that using scripts!

- First we need to create a new script.

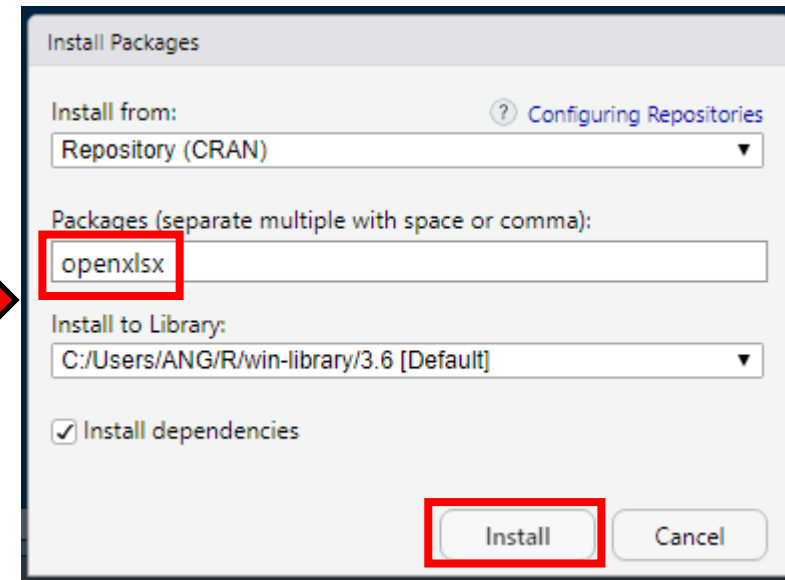
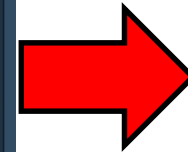
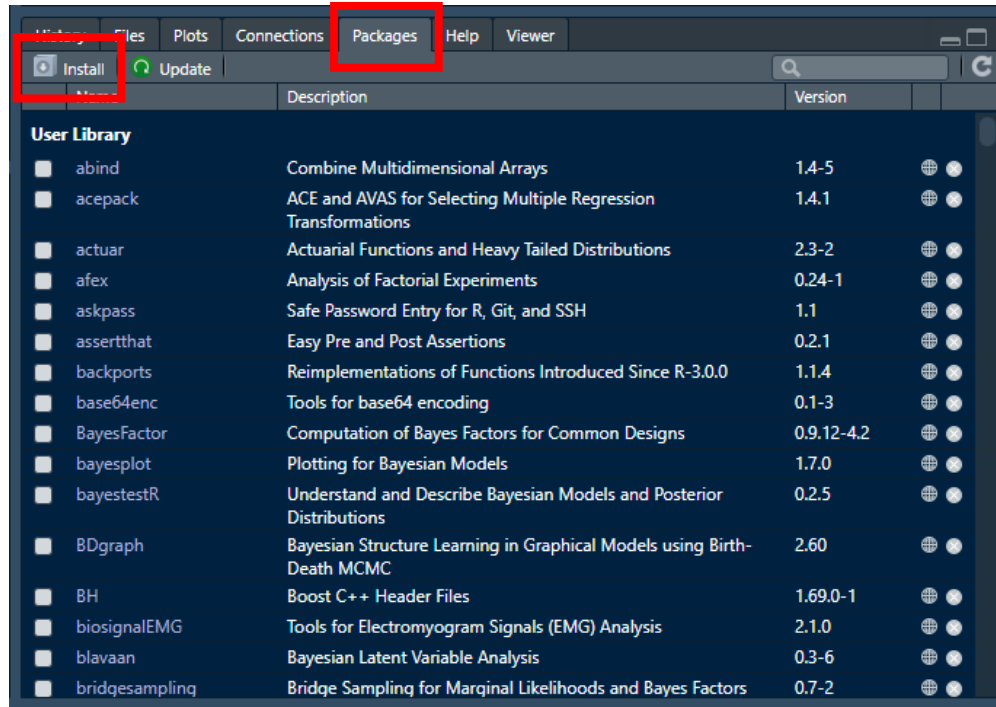
- Create New R-Script
 - File > New File > R Script



- Say what script is about
 - # this is my first script
- Save the script somewhere
 - File > Save As

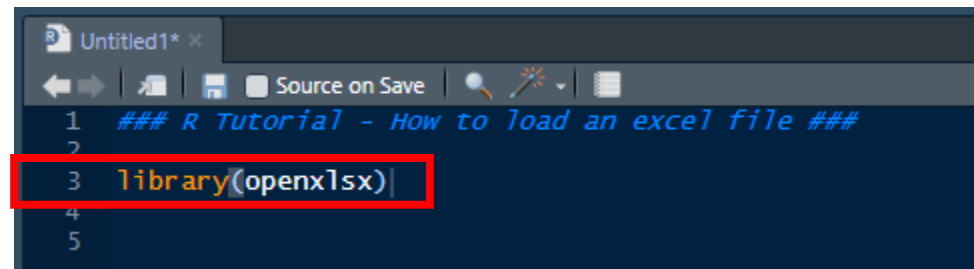
Install 'openxlsx' package

- Bottom-left > 'Packages' tab > 'Install' > type 'openxlsx' > 'Install'



Load package into R

- Type 'library(openxlsx)' into script
- Comment the code – e.g. # for opening excel
- Place cursor on that line and press **Ctrl + Enter** to run that line



```
1  ### R Tutorial - How to load an excel file ###  
2  
3  library(openxlsx)  
4  
5
```

Tell R where to look

Variable name – can give any name, but try to keep sort and informative

- `data_folder` <- “E:/Cognition_emotion_group_2020/R_tutorial_1/data_is_here/”

Folder directory – with forward slash (opposite to windows explorer)
Put your own directory – don’t copy this one.

- `data_file` <- “example_data_single_sheet.xlsx”

- `full_path` <- paste0(`data_folder` , `data_file`)

- paste0() function – literally just stitches two words together, put comma between objects

- Keep variable names short and informative
- No spaces, use underscore instead

read.xlsx() function

- `data2use <- read.xlsx(full_path)`
 - Or use your own variable name
- data2use should appear in environments section

Let's have look at the data

- Double-click variable in environment
 - Or type View(<insert variable name>)
- This is a **data-frame** – basically a table, contains many variables
- This data is based on the Go/No-go task
 - Respond quickly to Go stimulus but not No-go stimulus
 - RT recorded on each trial. Only Failed Nogo trials have RTs. Successful Nogo trials don't have RTs – hence NA.
 - Data from 5 participants, with 5 trials each

R likes data in long format

- What is long format?
 - Each variable is a separate column (participant, trial, condition, RT)
 - Each variable should only be specifying one thing.
 - Each observation is a separate row (each trial)
 - ***Long format is ideal for big datasets with many variables of different scales/type***

Participant	Trial	Condition	RT
P001	1	Congruent	230
P001	2	Neutral	235
P001	3	Incongruent	290
P002	1	Congruent	220
P002	2	Neutral	230
P002	3	Incongruent	300



Examples of not-long data

This variable is specifying 2 things – no good.

Participant	AUDIT_Q1	AUDIT_Q2	AUDIT_Q3	BISBAS_Q1	BISBAS_Q2	BISBAS_Q3
P001	4	2	2	1	1	2
P002	1	2	2	3	3	1
P003	3	4	4	2	2	2

These scores are from **separate items** and questionnaires but are on the **same row**... **NOT LONG**.

Participant	RT_Congruent	RT_Incongruent	RT_Neutral
P001	230	300	300
P002	250	290	200
P003	200	340	230

These RTs are from **separate conditions** (i.e. separate observations) but are on **same row**... **NOT LONG**.

This is what you want...

Participant	AUDIT_Q1	AUDIT_Q2	AUDIT_Q3	BISBAS_Q1	BISBAS_Q2	BISBAS_Q3
P001	4	2	2	1	1	2
P002	1	2	2	3	3	1
P003	3	4	4	2	2	2



Participant	Questionnaire	Item	Score
P001	AUDIT	1	4
P001	AUDIT	2	2
P001	AUDIT	3	2
P001	BISBAS	1	1
P001	BISBAS	2	1
P001	BISBAS	3	2

This is what you want...

Participant	RT_Congruent	RT_Neutral	RT_Incongruent
P001	230	300	300
P002	250	200	290
P003	200	230	340



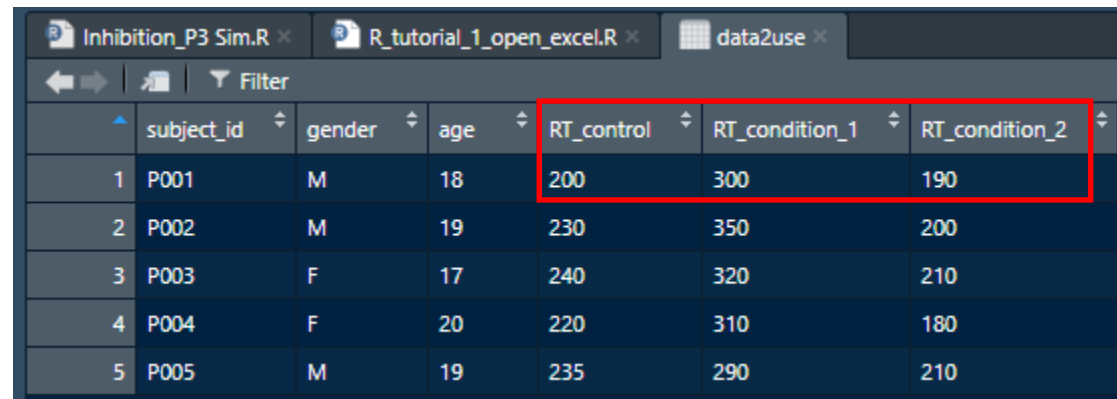
Participant	Condition	RT
P001	Congruent	230
P001	Neutral	300
P001	Incongruent	300
P002	Congruent	250
P002	Neutral	200
P002	Incongruent	290

Ok, show me how to do that.

- We need ***tidyr*** package
- We will use the ***gather()*** function – Wide to Long
- Then we will use ***spread()*** function – Long to Wide
- Links:
 - R-documentation (*explains how to use functions*):
 - <https://www.rdocumentation.org/packages/tidyr/versions/0.8.3/topics/gather>
 - Another guide to use gather() and spread() :
 - http://www.cookbook-r.com/Manipulating_data/Converting_data_between_wide_and_long_format/

Tidyr and gather()

- Install tidyr
- Load tidyr from library using library()
- Load **'gather_example_data.xlsx'** into a variable using read.xlsx()
- View data
 - not long-format, RTs for different conditions on same row
 - Want separate columns for RT and condition



The screenshot shows an RStudio window with three tabs: 'Inhibition_P3 Sim.R', 'R_tutorial_1_open_excel.R', and 'data2use'. Below the tabs is a data table with 7 columns: an index column, 'subject_id', 'gender', 'age', 'RT_control', 'RT_condition_1', and 'RT_condition_2'. The first row of data is highlighted with a red box.

	subject_id	gender	age	RT_control	RT_condition_1	RT_condition_2
1	P001	M	18	200	300	190
2	P002	M	19	230	350	200
3	P003	F	17	240	320	210
4	P004	F	20	220	310	180
5	P005	M	19	235	290	210

Tidyr and gather()

2. Creating new column for condition names

1. Variable to transform

```
• data2use_v2 <- gather(data = data2use, key = "condition",  
  value = "RT", c("RT_control", "RT_condition_1", "RT_condition_2") )
```



3. Creating new column for condition names

- Name the 'key' after the variable names
- Name 'value' after the stuff in the actual cells
- View (data2use_v2)

spread()

- Working from the last dataset
- `data2use_v3 <- spread(data = data2use_v2, key = "condition", value = "RT")`

Save variable into Excel

- `write.xlsx(data2use_v3 , full_path)`

Exercise: Lets open our own data

- Use `read.xlsx()` function to open your own file
- Use `gather()` to convert to long format (if not already long)
- If data is long, use `spread()` to convert to wide-format

- If different sheets, single file
 - There is code
- If different excel files
 - There is code
- If something else
 - ask

That's it for today

- Next Cognition and Emotion Group Meeting (6 March)
 - Discussing pre-registration
- Next R Session (13 March)
 - Plotting in R using ggplot2

Cheat sheet - Hotkeys

- **Ctrl + Enter** – Runs single line of code
- **Alt + minus** – <- - for defining variables
- **Ctrl + 3 - #** - ignores stuff after this
- **Ctrl + Shift + C** – disable/enable line of code
- **Ctrl + Shift + Enter** – Runs whole script

Cheat sheet - Functions

`paste0()` or `paste()`, `cbind()` and `rbind()`

- Lets you stick things together

`Substring()`

- Extract part of a word – good for getting subject id from filenames

`Unique()`

- Gets unique values from variable – good for checking participants or factors in large datasets

`Length()` and `dim()`

- Shows you size of variable – good for troubleshooting – check if right number of trials/participants or whatever are there

`Subset()`

- Extract a smaller part of bigger dataset – good for follow-up analyses (e.g. only want to analyse a certain condition of the data).

`Aggregate()`

- Can use to calculate mean values per participant and condition.