

# A simple guide to writing a script

Thomas Battram (he/him)

Thanks to Gwen Fernandes for assistance

[bristol.ac.uk](http://bristol.ac.uk)



# Outline

- Making a new R script
- Commenting your scripts
- What to think about before starting to script
- How the top of your script should look
- Making your code easier to read
- Saving your scripts

# Making a new R script

- The usual R Studio screen has four windows:
  - CONSOLE
  - WORKPLACE AND HISTORY
  - FILES, PLOTS, PACKAGES AND HELP
  - **R SCRIPT AND DATA VIEW** (this is where you keep a record of your work. For Stata users, this would be like your do-file, for SPSS users it is like the syntax and for SAS users, the SAS program)

RStudio

Project: (None)

Environment History Connections

Global Environment

Environment is empty

```
1 install.packages("forestplot")
2
3 install.packages("grid")
4 install.packages("magrittr")
5 install.packages("checkmate")
6
7 TAC_CMC2 <-
8   structure(list(
9     mean = c(NA, NA, 3.36, 0.66, 3.94, 6.45, 7.96, NA, 3.72),
10    lower = c(NA, NA, 0.19, 0.11, 0.17, 2.10, 1.32, NA, 0.96),
11    upper = c(NA, NA, 58.19, 3.93, 89.91, 19.79, 48.09, NA, 14.37)),
12    Names = c("mean", "lower", "upper"))
13
```

53:15 (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.

> |

Files Plots Packages Help Viewer

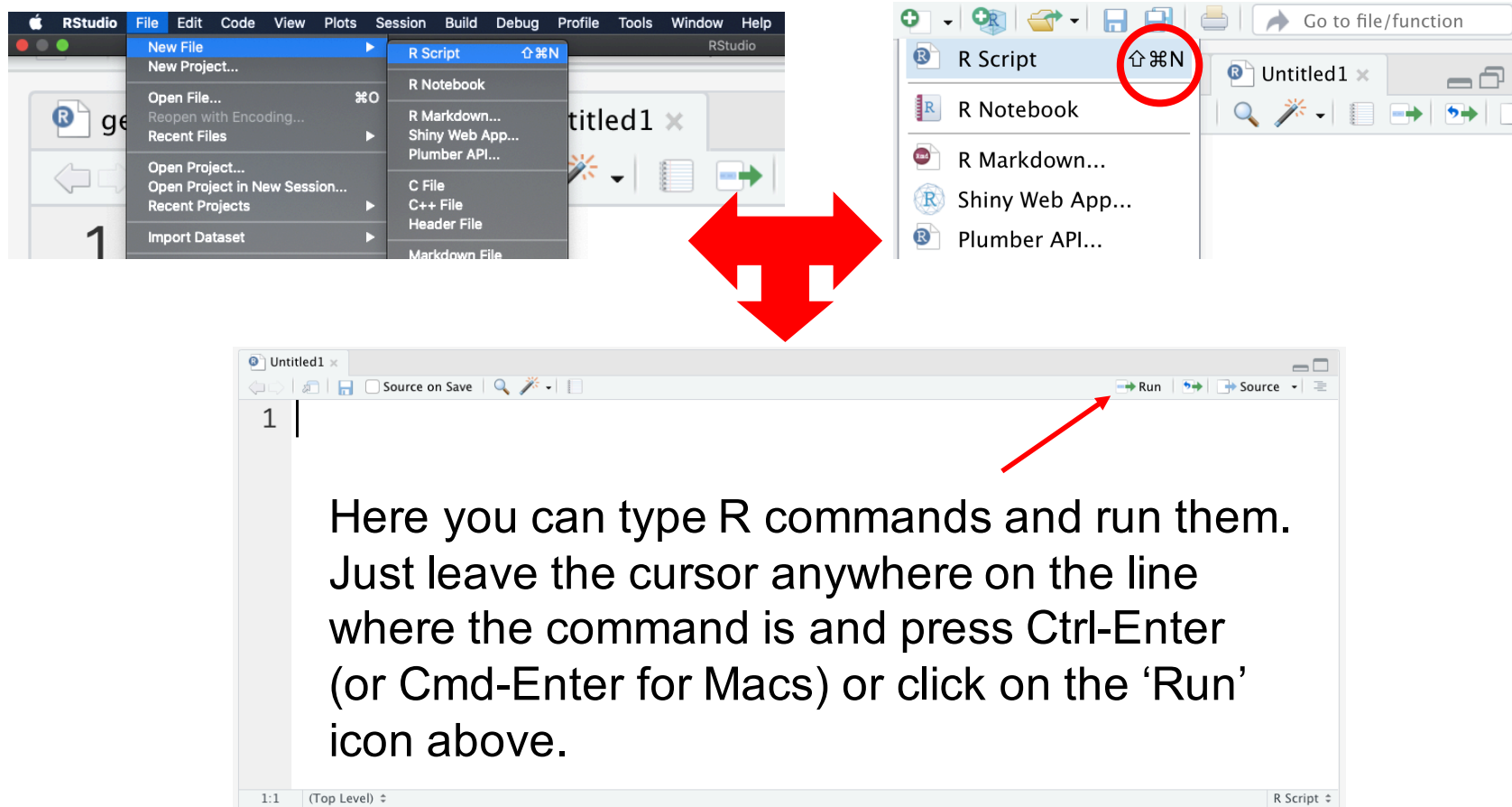
New Folder Delete Rename More

Home

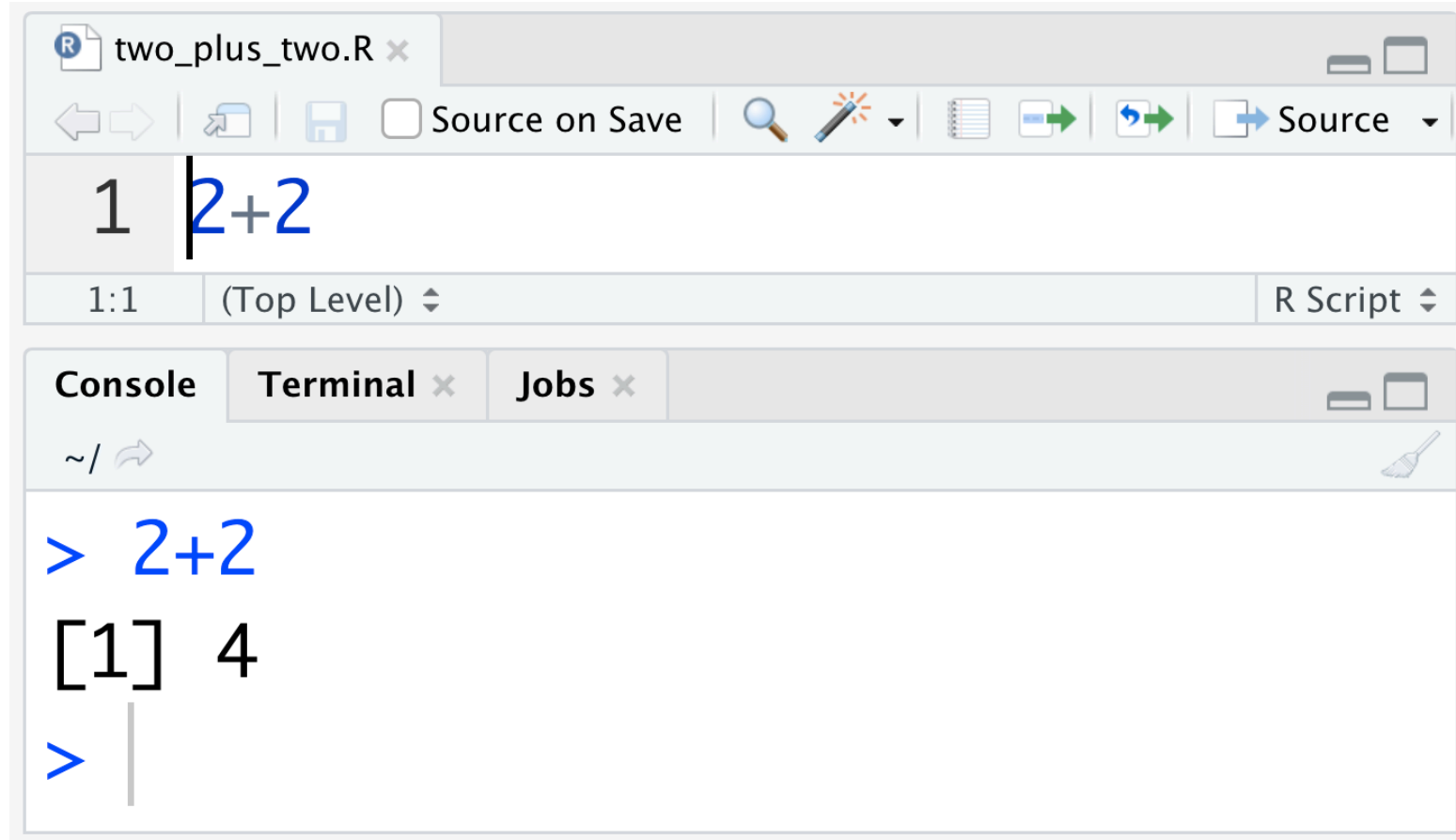
	Name	Size	Modified
<input type="checkbox"/>	.Rhistory	8.8 KB	Jul 8, 2020, 4:57 PM
<input type="checkbox"/>	Applications		
<input type="checkbox"/>	cveda		
<input type="checkbox"/>	Desktop		
<input type="checkbox"/>	Documents		
<input type="checkbox"/>	Downloads		
<input type="checkbox"/>	Library		
<input type="checkbox"/>	Logs		
<input type="checkbox"/>	Movies		
<input type="checkbox"/>	Music		
<input type="checkbox"/>	Pictures		
<input type="checkbox"/>	Public		
<input type="checkbox"/>	R script example.R	2.2 KB	Oct 2, 2020, 10:46 AM
<input type="checkbox"/>	X-RAYstataoutput.docx	491.8 KB	May 4, 2016, 6:56 AM
<input type="checkbox"/>	Zotero		

Notes

# Making a new R script



# Making a new R script

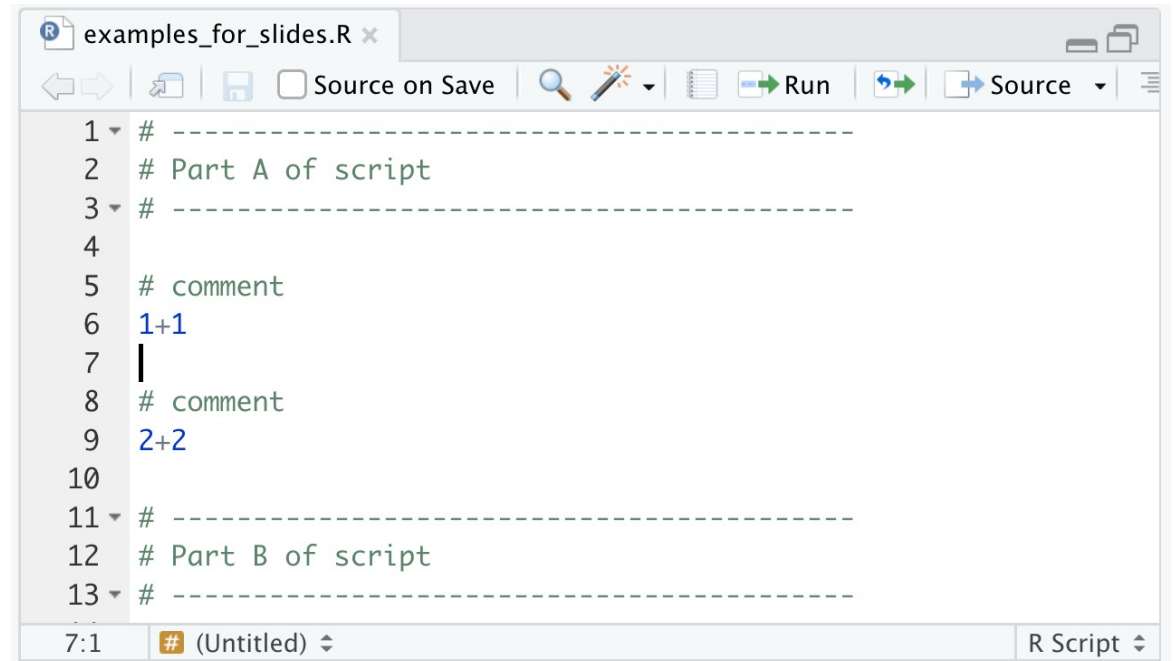


The screenshot displays the RStudio environment. At the top, a tab for 'two\_plus\_two.R' is active. The editor window shows a single line of code: `1 2+2`. Below the editor, the 'Console' pane is visible, showing the execution of the code: `> 2+2` followed by the output `[1] 4`. The 'Terminal' and 'Jobs' panes are also present but empty.

```
two_plus_two.R x
Source on Save
1 2+2
1:1 (Top Level) R Script
Console Terminal x Jobs x
~/
> 2+2
[1] 4
>
```

# Commenting

- Code with a # before it does not get run
- This is useful for making your scripts much easier to read!
- Comment on WHY and WHAT (to start with)
  - Start with many comments! Also, use comments to split up the script to make it clearer



The screenshot shows an R script editor window titled 'examples\_for\_slides.R'. The script contains the following lines:

```
1 # -----  
2 # Part A of script  
3 # -----  
4  
5 # comment  
6 1+1  
7 |  
8 # comment  
9 2+2  
10  
11 # -----  
12 # Part B of script  
13 # -----
```

The editor interface includes a toolbar with icons for navigation, saving, and running code. The status bar at the bottom indicates the current line is 7:1 and the file is an 'R Script'.

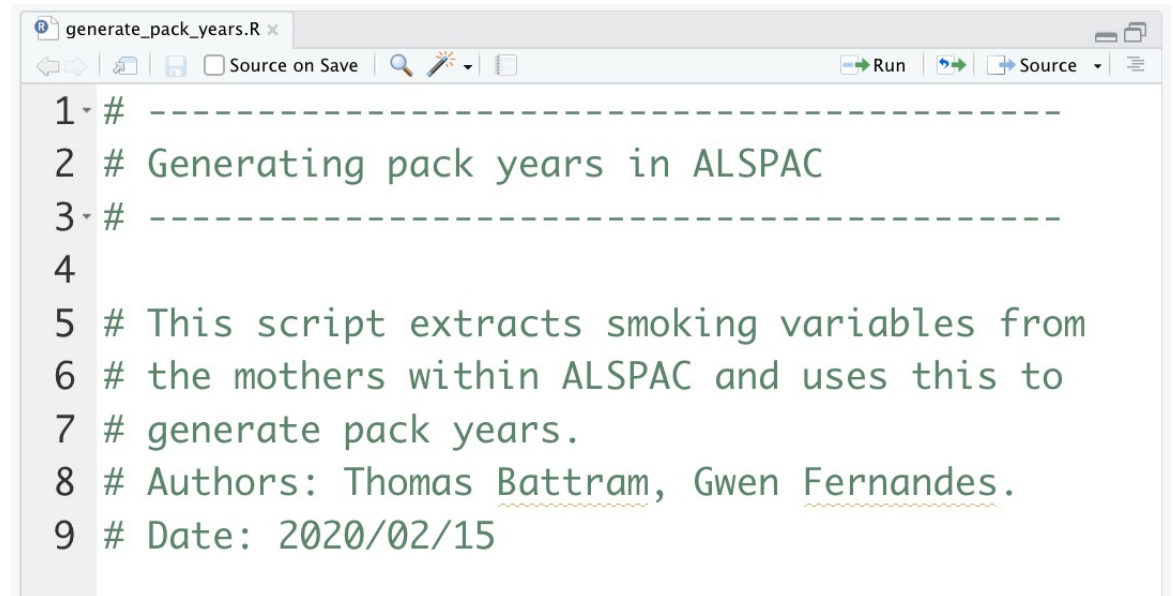
# 3 steps before starting

1. Why am I writing this script?
2. What do I need to write this script?
3. How am I going to write this script?



# Step 1: Why am I writing this script?

- Think about it and write it down!
  - e.g. clean a dataset OR assess association between x and y
- Give the script a good name, write a descriptive title + add a couple of lines that describe the purpose of the script



```
1 # -----  
2 # Generating pack years in ALSPAC  
3 # -----  
4  
5 # This script extracts smoking variables from  
6 # the mothers within ALSPAC and uses this to  
7 # generate pack years.  
8 # Authors: Thomas Battram, Gwen Fernandes.  
9 # Date: 2020/02/15
```

## Step 2: What do I need to write this script?

- Datasets!
- Using only base R can make things difficult... Packages!
- Packages are made by others and are there to make your life easier

For example, reading data into R can be tricky depending on how the data is stored, but packages can make this easier!

```
3 install.packages("packagename")|
```

# Reading data into R

Depending on what form the data is in, you have to use different functions to read in the data. Data you get may be in:

- excel spreadsheets
- comma seperated value (csv) files
- text separated value (tsv) files
- spss files
- stata files
- images (e.g. .png files)

```
11 # load packages
12 library(haven)
13
14 # read in data
15 df <- read_dta("my_data.dta")
```

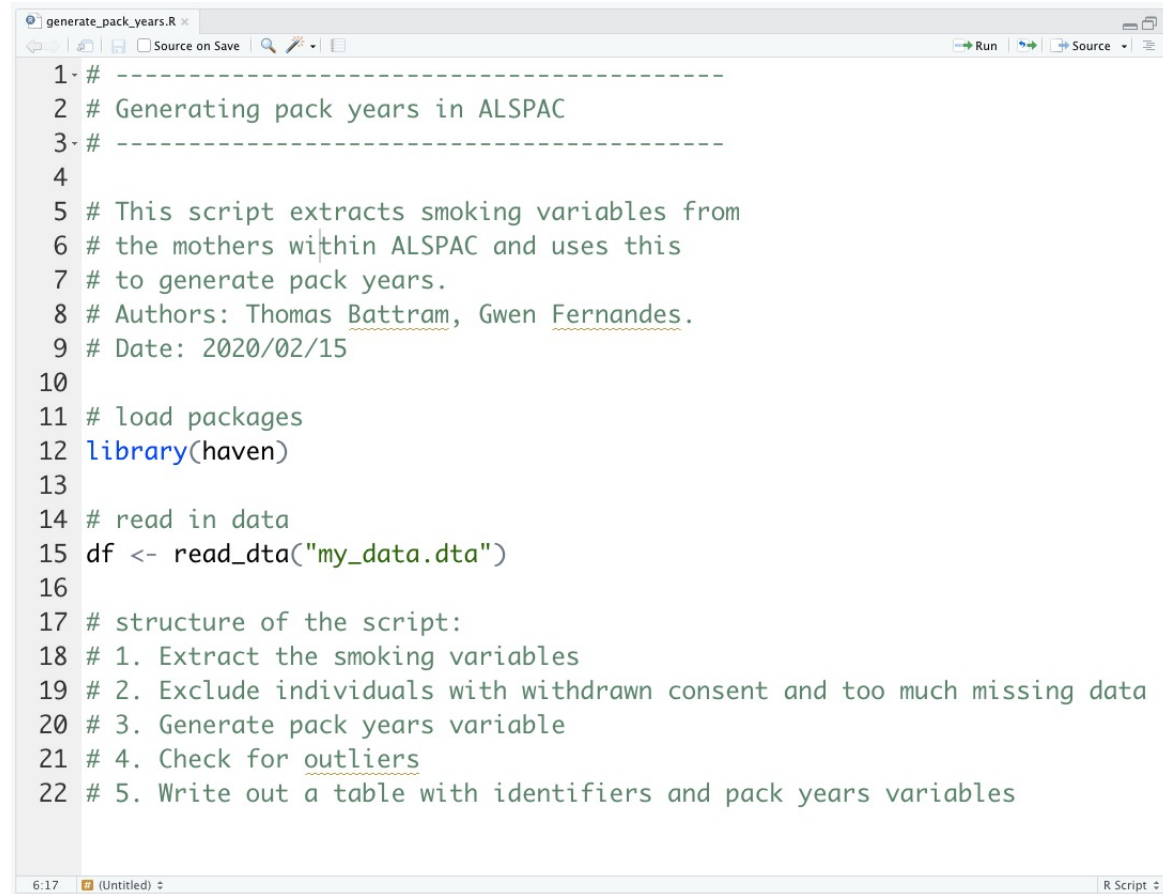
These need different functions to read them in, some of which are only available with certain packages.

# Step 3: How am I going to write this script?

- Linked to why you are writing it and what you need to write it!
- Write out each step

```
17 # structure of the script:
18 # 1. Extract the smoking variables
19 # 2. Exclude individuals with withdrawn
20 #    consent and too much missing data
21 # 3. Generate pack years variable
22 # 4. Check for outliers
23 # 5. Write out a table with identifiers
24 #    and pack years variables
```

# Top of the script



```
1 # -----  
2 # Generating pack years in ALSPAC  
3 # -----  
4  
5 # This script extracts smoking variables from  
6 # the mothers within ALSPAC and uses this  
7 # to generate pack years.  
8 # Authors: Thomas Battram, Gwen Fernandes.  
9 # Date: 2020/02/15  
10  
11 # load packages  
12 library(haven)  
13  
14 # read in data  
15 df <- read_dta("my_data.dta")  
16  
17 # structure of the script:  
18 # 1. Extract the smoking variables  
19 # 2. Exclude individuals with withdrawn consent and too much missing data  
20 # 3. Generate pack years variable  
21 # 4. Check for outliers  
22 # 5. Write out a table with identifiers and pack years variables
```

# Make your code easy to read

1. Use a consistent style when writing code

```
a_var <- c(1, 2, 3)
a.var <- c(1, 2, 3)
aVar <- c(1, 2, 3)
```

2. Use spaces appropriately

```
x <-c(1,2,51,124,4124)
x <- c(1, 2, 51, 124, 4124)
x <-      c(1      ,2      ,      51,124      , 4124)
```

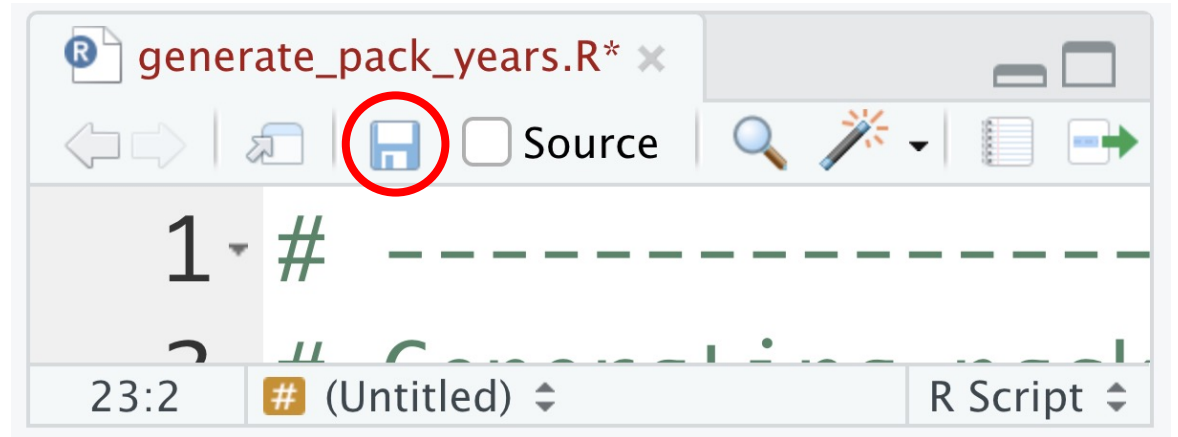
3. Use indents appropriately

```
my_plot <- plot(x = my_data$x, y = mydata$y, xlab = "I

my_plot <- plot(x = my_data$x,
                y = my_data$y,
                xlab = "My X-axis label",
                ylab = "My Y-axis label",
                main = "My plot title")
```

# Saving scripts

- It's important to save scripts as you go and you can always come back to them and send them to other people.
- Give the script a good name and save it regularly!



CTRL (or CMD) + S

# Summary

3 questions to think about before writing a script:

1. Why am I writing this script?
2. What do I need to write this script?
3. How am I going to write this script?

- Comment your code a lot (using #)
- Make your code easy to read
- Save your scripts regularly



# Git

- One way to share scripts with the world (or just your collaborators) is through [GitHub](#).
- Slides are available here:  
[https://github.com/thomasbattram/how to write a script](https://github.com/thomasbattram/how_to_write_a_script)
- For the curious:
  - Rationale for github: <https://guides.github.com/introduction/git-handbook/>
  - Quickstart guide: <https://guides.github.com/activities/hello-world/>

# Any questions?



# On to the practical!!

- [Bad script](#)
- [Good script](#)

Things to consider for the bad script:

- a. What is the point of this script?
- b. What is it about? Can you tell by the commands? Could you infer, if you had to?
- c. How could you make this script better?
- d. Is this something you could share with a new user/researcher/analyst?