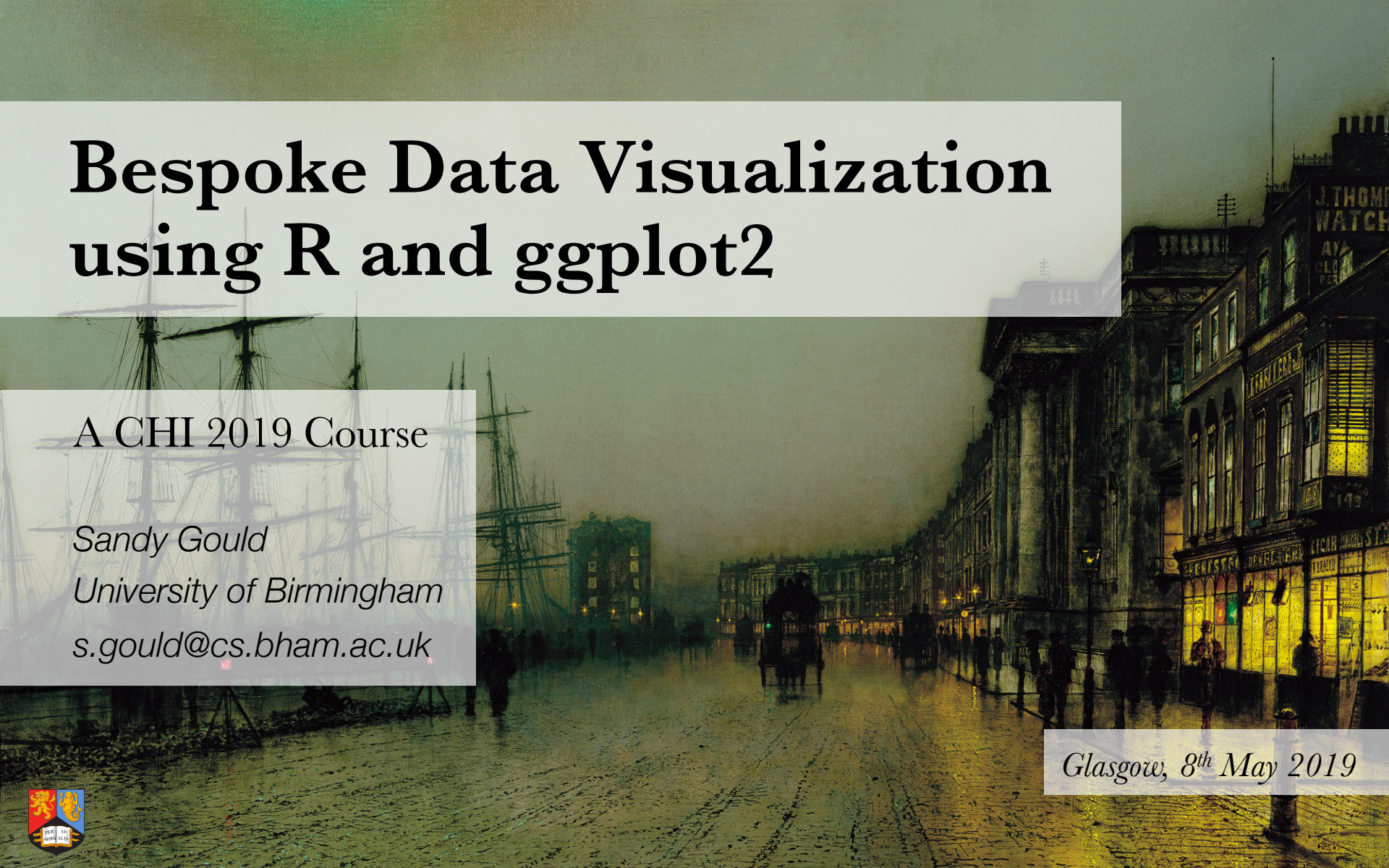


Bespoke Data Visualization using R and ggplot2

A CHI 2019 Course

Sandy Gould
University of Birmingham
s.gould@cs.bham.ac.uk

Glasgow, 8th May 2019

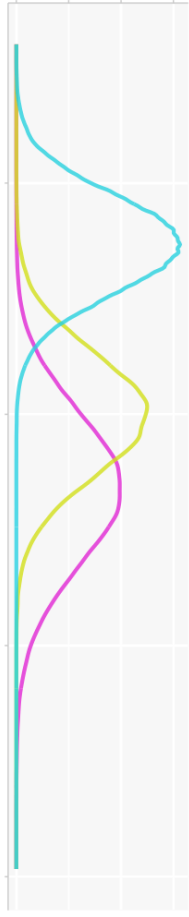
This course

Welcome!

Thank you for joining this course!

There are two parts to it:

- I want to talk to you about our approach to visualization in this course.
- Then I want us to work through a set of practical tasks. This will be the vast majority of our time.



Glossary

Term	Meaning
R	A statistical programming language
RStudio	An environment for writing R programs
ggplot2	A library for creating visualizations using R
Declarative visualization	A way of specifying visualizations

Materials

http://sjjg.uk/chi19-course

Click the link for materials. You'll need credentials.

Username: chi2019

Password: ggplot

Declarative visualizations

Specifying visualizations declaratively means that:

- Output is **predictable**
  - *e.g., if we want to redraw*
- It's easy to completely change the visualization without having to mess around with the underlying data.
- We can create more complex visualizations with a higher degree of **reliability**
- Text-based declarations will always be **readable**

The ggplot2 approach

ggplot2 takes a declarative approach to the creation of visualizations.

The emphasis is on:

- A logical structure to the creation of visualizations
- Structuring of the data that is being used
- High quality defaults

‘The Grammar of Graphics’

Hadley Wickham, the developer of of ggplot2 has published on the philosophy of ggplot2 in his book ‘The Grammar of Graphics’.

Essentially, everything is split into:

- Aesthetics *(what is going to be shown and how it will be shown)*
- Geometries *(the form of plots that will represent the data)*

I find the labels for these confusing at times!

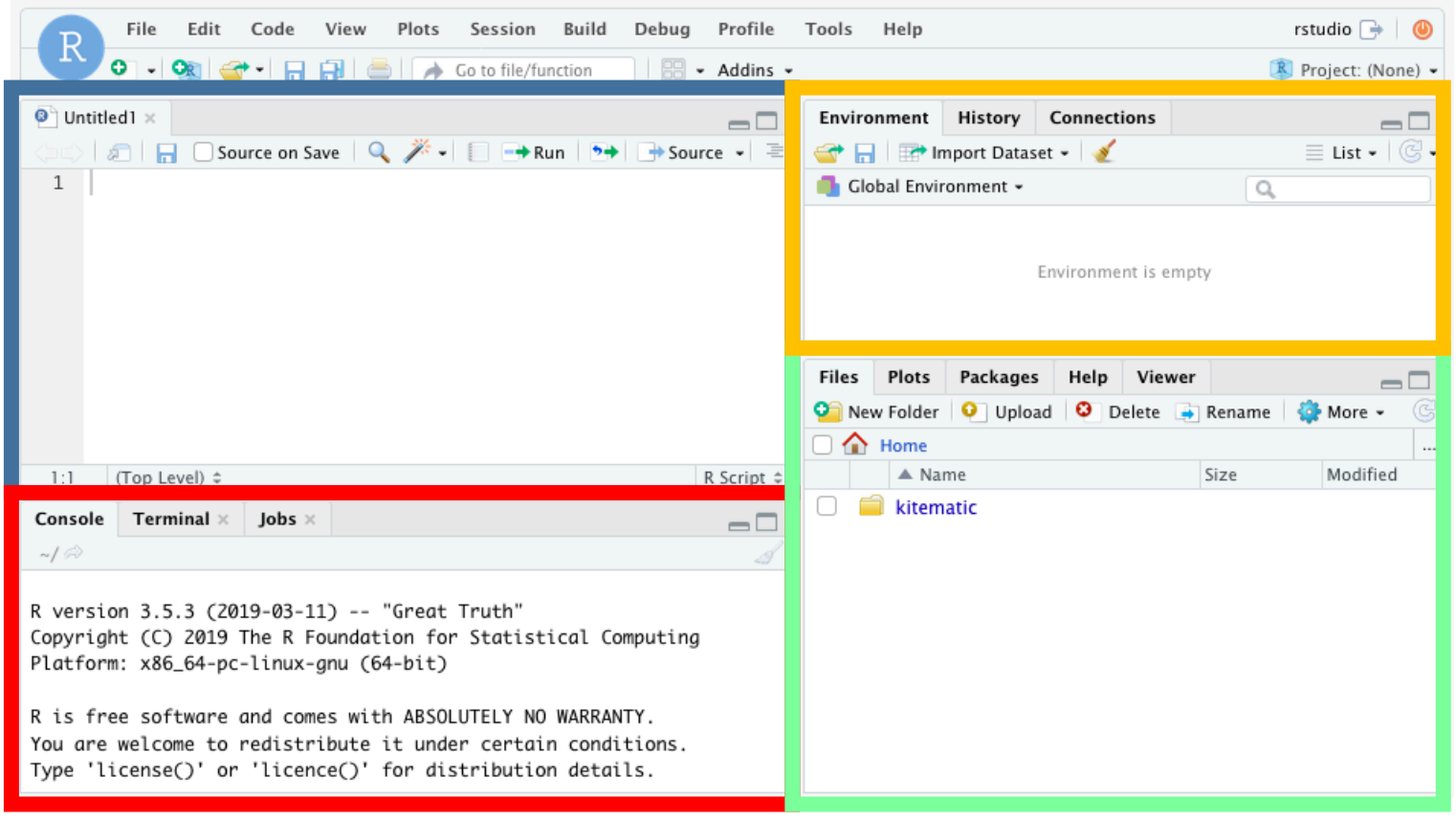
Writing code for R

- R is written using plaintext files.
- They can be written using a text editor and run from the command line.
- They can be written and run from R's default editing environment.
- They can be written and run from RStudio, a complete development environment for R.

Writing code for R

- The goal of today is not to teach you programming.
- It is to give you an understanding of how plots can be pieced together using a variety of ggplot2 commands.
- You will leave with an understanding of the logic of ggplot2 as well as plenty of template code to get you started.
- The website will stay up for you to refer to in future.

RStudio


File Edit Code View Plots Session Build Debug Profile Tools Help
rstudio
Go to file/function
Addins
Project: (None)
Untitled1
Source on Save
Run
Source
1
1:1
(Top Level)
R Script
Environment History Connections
Import Dataset
List
Global Environment
Environment is empty
Files Plots Packages Help Viewer
New Folder
Upload
Delete
Rename
More
Home
Name
Size
Modified
kitematic
Console Terminal Jobs
~/
R version 3.5.3 (2019-03-11) -- "Great Truth"
Copyright (C) 2019 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (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.

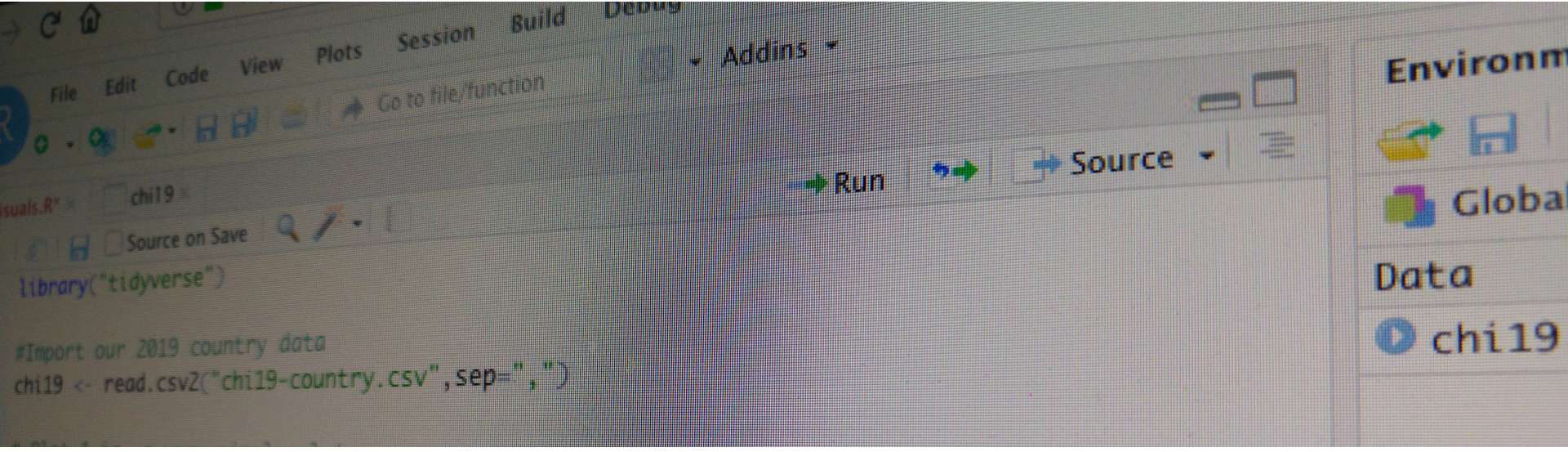
Getting RStudio up and running

We're going to be using RStudio in the browser.

https://rstudio.sjjg.uk

Log in with the details that I have given you.

Let's have a look at an example together



Our dataset

We're going to be visualizing some CHI data.

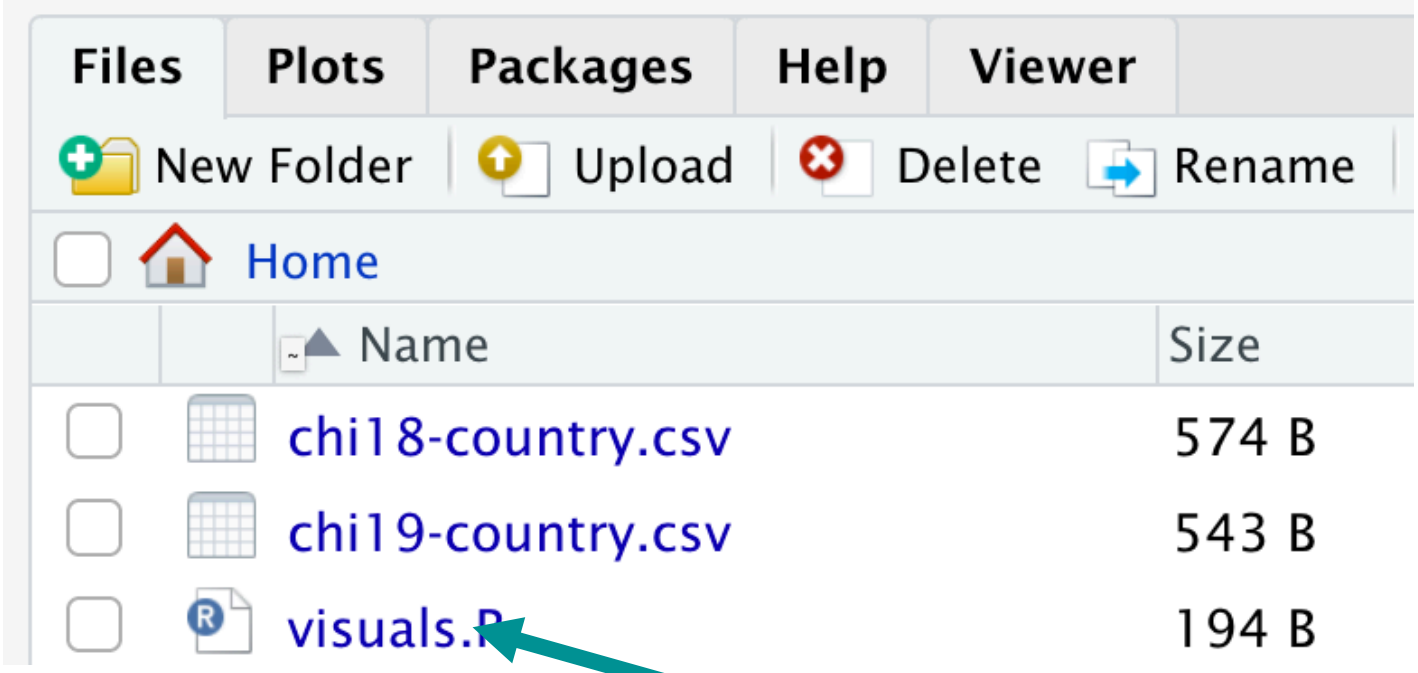
They're data about how publications by country in 2018 and 2019.

The 2019 data is based on Kashyap Todi's processing of data provided by the CHI Data Chair, Frederik Brudy.

The 2018 data are based on Kashyap Todi's data.

Thanks to both!

Loading CHI 2019 data

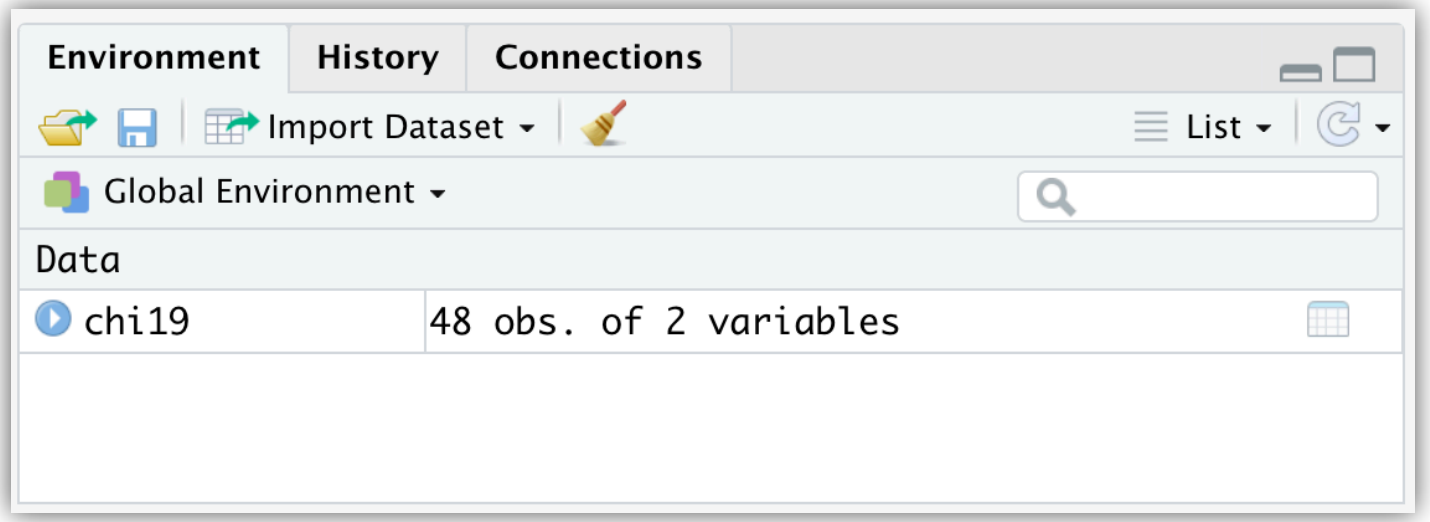


Click this!

CSV Import code

```
library("tidyverse")

#Import our 2019 country data
chi19 <- read.csv2("chi19-country.csv",sep=",")
```



CHI 2019 Country Data

Index

Country Variable

Pubs Variable

n	Country	Pubs
1	United States	359
2	United Kingdom	125
3	Canada	74
4	Germany	68
5	Australia	41
6	Japan	32
7	China	25
8	Republic of Korea	23
9	Denmark	22
10	Netherlands	21

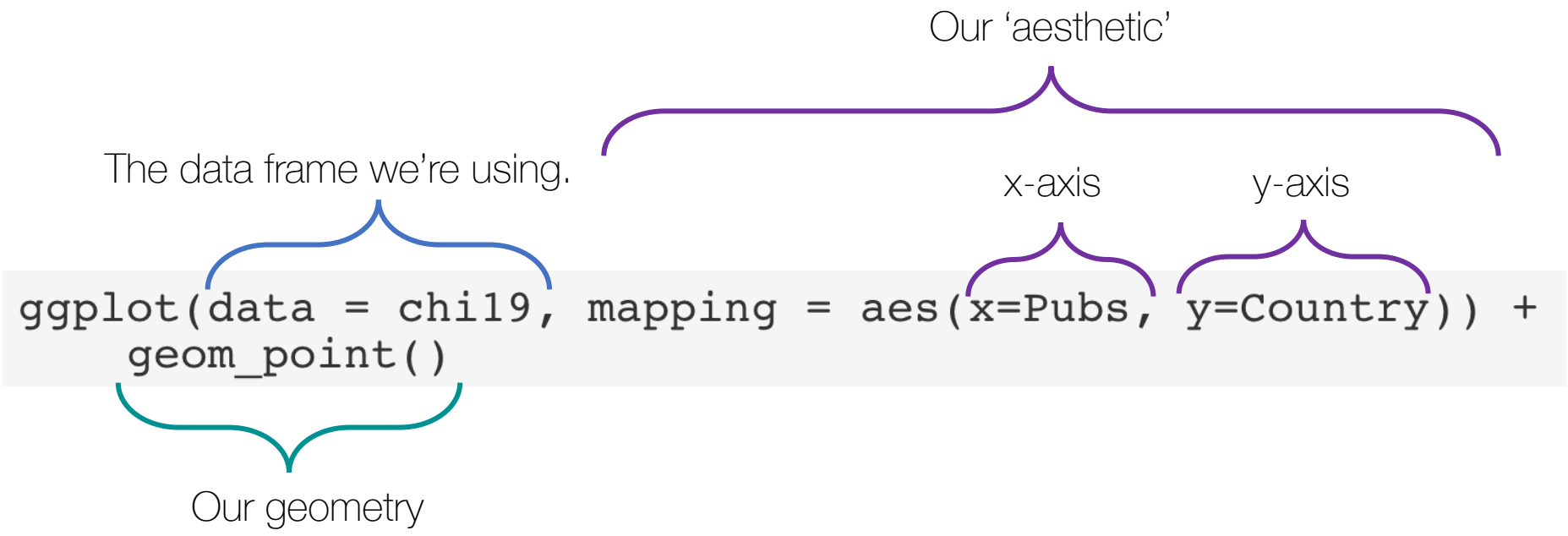
ggplot2

ggplot2 visualizations are created using the `ggplot()` function.

This function defines:

- The data we are using
- The aesthetics of the visualization we will create

The ggplot2 function



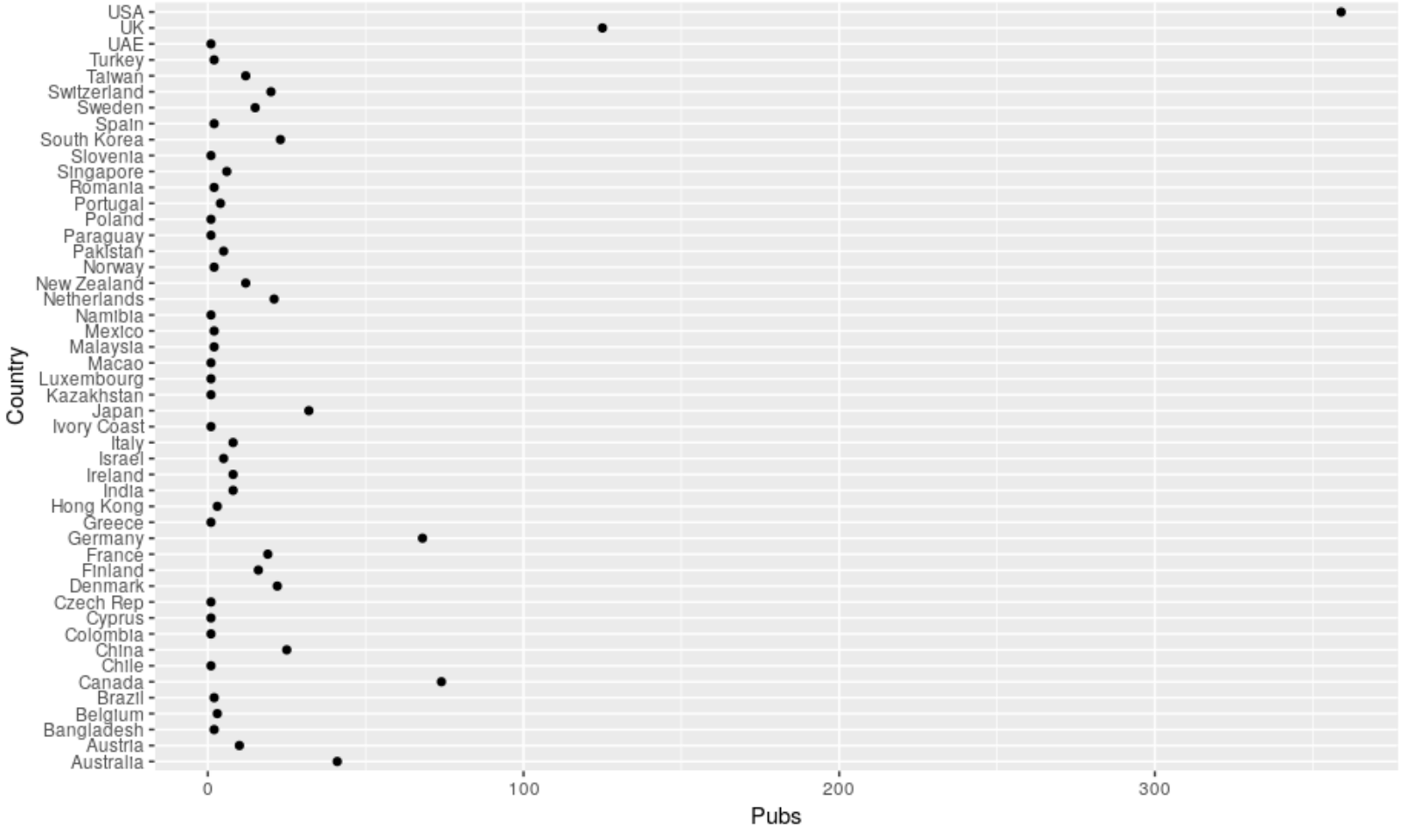
ggplot2 geometries

```
geom_abline()
geom_hline()
geom_vline()
geom_bar()
geom_col()
stat_count()
geom_bin2d()
stat_bin_2d()
geom_blank()
geom_boxplot()
stat_boxplot()
geom_contour()
stat_contour()
geom_count()
geom_ribbon()
stat_sum()
geom_density()
stat_density()
geom_density_2d()
stat_density_2d()
geom_dotplot()
geom_errorbarh()
geom_hex()
stat_bin_hex()
geom_freqpoly()
geom_histogram()
stat_bin()
geom_jitter()
geom_crossbar()
geom_area()
geom_errorbar()
geom_linerange()
geom_pointrange()
geom_map()
geom_path()
geom_line()
geom_step()
geom_point()
geom_polygon()
geom_qq_line()
stat_qq_line()
geom_qq() stat_qq()
geom_quantile()
stat_quantile()
geom_rug()
geom_segment()
geom_curve()
geom_smooth()
stat_smooth()
geom_spoke()
geom_label()
geom_text()
geom_raster()
geom_rect() geom_tile()
geom_violin()
stat_ydensity()
stat_sf() geom_sf()
geom_sf_label()
geom_sf_text()
coord_sf()
```

The ggplot2 function

Now select the code and run it:

```
ggplot(data = chi19, mapping = aes(x=Pubs, y=Country)) +
    geom_point()
```


USA
UK
UAE
Turkey
Taiwan
Switzerland
Sweden
Spain
South Korea
Slovenia
Singapore
Romania
Portugal
Poland
Paraguay
Pakistan
Norway
New Zealand
Netherlands
Namibia
Mexico
Malaysia
Macao
Luxembourg
Kazakhstan
Japan
Ivory Coast
Italy
Israel
Ireland
India
Hong Kong
Greece
Germany
France
Finland
Denmark
Czech Rep
Cyprus
Colombia
China
Chile
Canada
Brazil
Belgium
Bangladesh
Austria
Australia
Country
0
100
200
300
Pubs

Let's try and build this out a bit

You should now have a basic plot. It's far from perfect though

Let's pick up the webpage at **5.3 Getting some order to proceedings**

You're going to:

- Get the data into a sensible order
- Have a play around with different geometries

Stop when you get to **6. Controlling the appearance of plots**

Controlling the look of plots

There are three ways of controlling the look of plots:

- Through aesthetics
- Through geometries
- Through themes

} We'll look at these first

Appearance in geometries

```
ggplot(data = chi19, mapping = aes(x=reorder(Countr
  geom_point(colour="Orange", size=3) +
  coord_flip()
```

Color of points

Size of points

Appearance in geometries

On the website, work from

6.1 Appearance as part of geometries

Until you reach the end of:

6.2 Controlling appearance with themes

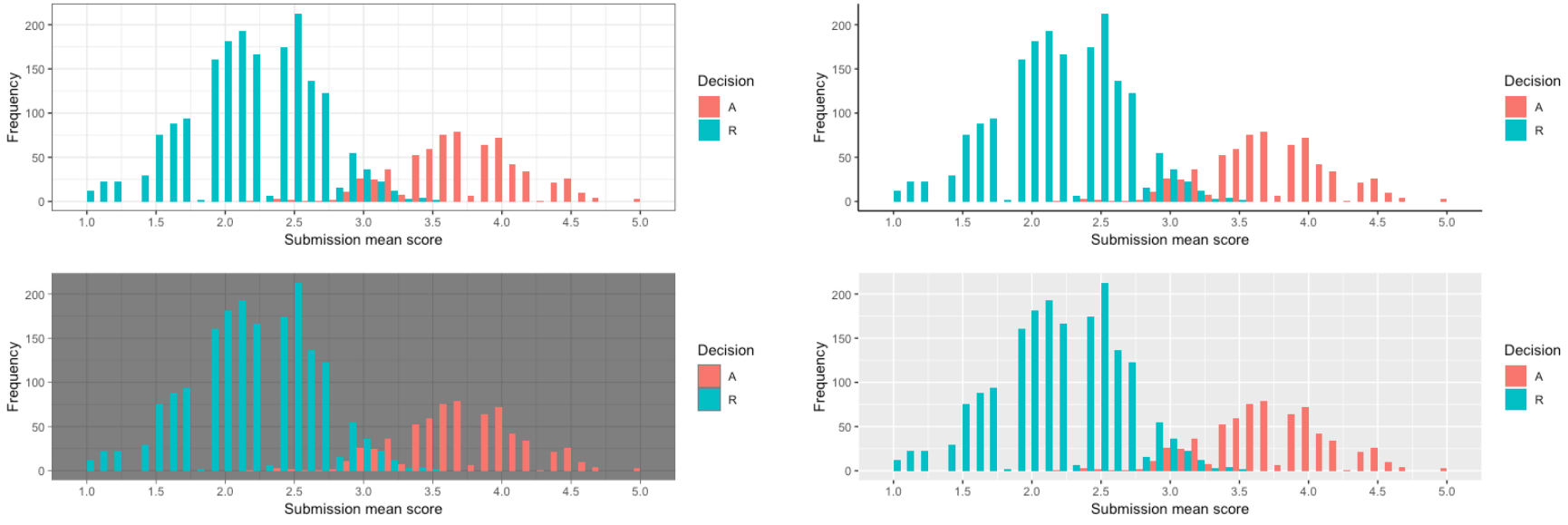
Controlling appearance with themes

Themes control the **overall** appearance of your plot.

- Gridlines
- Background
- Axes
- Ticks
- Labels

Controlling appearance with themes

There are built-in themes, like `theme_classic()`, `theme_dark()`, `theme_bw()` and `theme_gray ()`.



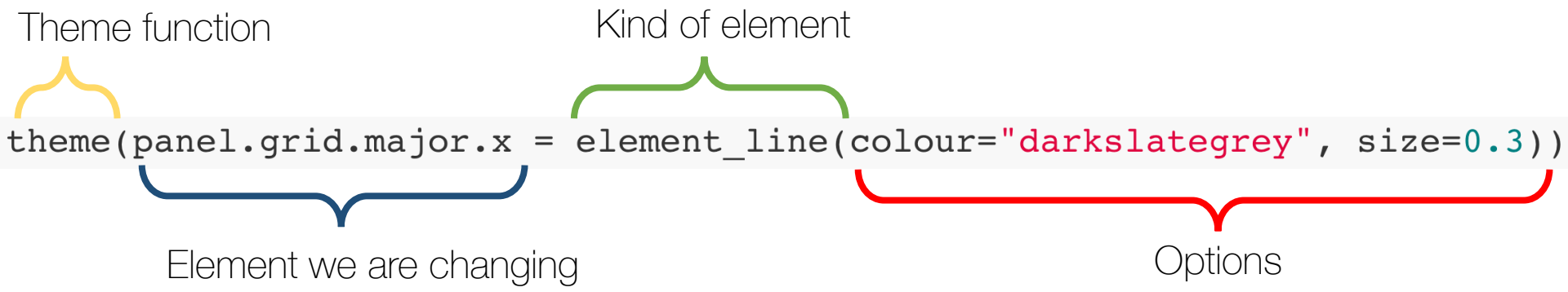
Controlling appearance with themes

Themes are made up of a number of components in a hierarchical form. (e.g., `legend.box.background` or `plot.caption`).

They are drawn using elements:

- Lines are `element_line()`
- Rectangles are `element_rect()`
- etc

Controlling appearance with themes



Controlling appearance with themes

On the website, work until you reach:

7. A plot drawing on two datasets

Plotting multiple variables

So far we have looked at a very simple plot.

- It has a categorical variable.
- And a continuous variable.

What if we want to look at many year's worth of data?

Enter the **CHImbined** dataset with 2019 and 2018 data!

CHI 2019 Country Data

Index

Country Variable

Year Variable

Pubs Variable

n	Country	Year	Pubs
1	Australia	2018	23
2	Australia	2019	41
3	Austria	2018	8
4	Austria	2019	10
5	Belgium	2018	3
6	Belgium	2019	3
7	Brazil	2018	1
8	Brazil	2019	2
9	Canada	2018	78
10	Canada	2019	74

Plotting multiple variables

There are three ways of controlling the look of plots:

- Through aesthetics 
- Through geometries
- Through themes

Plotting multiple variables

aes function

New ‘colour’ aes addition

```
aes(x=reorder(Country, Pubs), y=Pubs, colour=factor(Year)))
```

Stuff you recognize

Plotting multiple variables

On the website, work from

7. A plot drawing on multiple variables

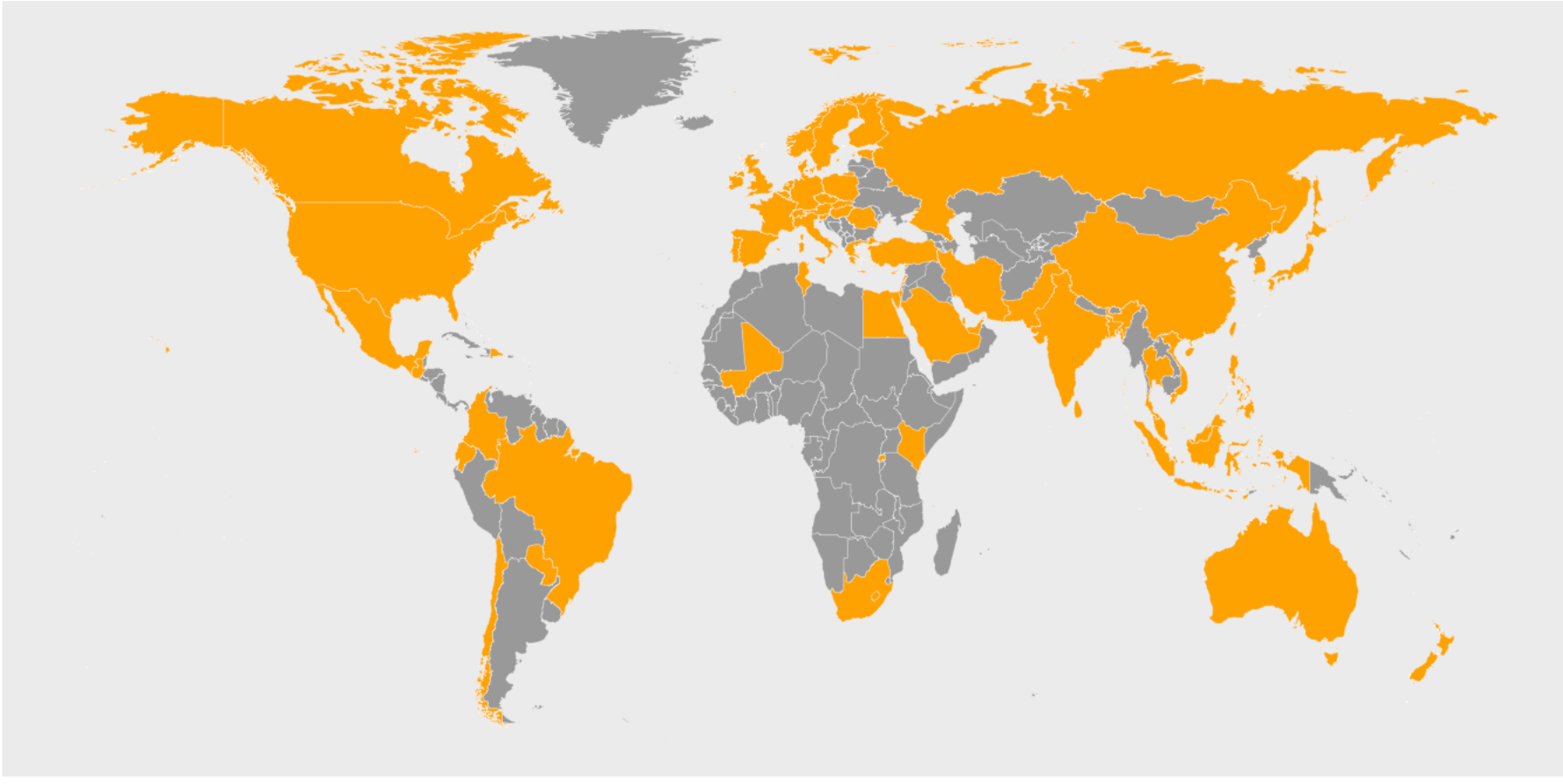
Until you reach:

8. Mapping locations of CHI authors

Mapping locations

```
#Import mapping libraries
library(maps)
```

Mapping locations



Mapping locations

```
world <- map_data("world")
world$fac <- world$region %in% chi19$Country
```

Mapping locations

We're going to use the geom_poly()* geometry for plotting our graph.

Map of the world

Longitude and latitude

```
ggplot(data = world, aes(x=long, y = lat, group = group)) +
  geom_polygon(colour="white")
```

Plot the polygons

Colour borders in white

Keep countries together

* There is a `geom_map()`, which I discuss on the website

Mapping locations

We're going to use the geom_poly()* geometry for plotting our graph.

Map of the world

Longitude and latitude

```
ggplot(data = world, aes(x=long, y = lat, group = group)) +
  geom_polygon(colour="white")
```

Plot the polygons

Colour borders in white

Keep countries together

* There is a `geom_map()`, which I discuss on the website

Plotting multiple variables

On the website, work from

8. Mapping locations of CHI authors

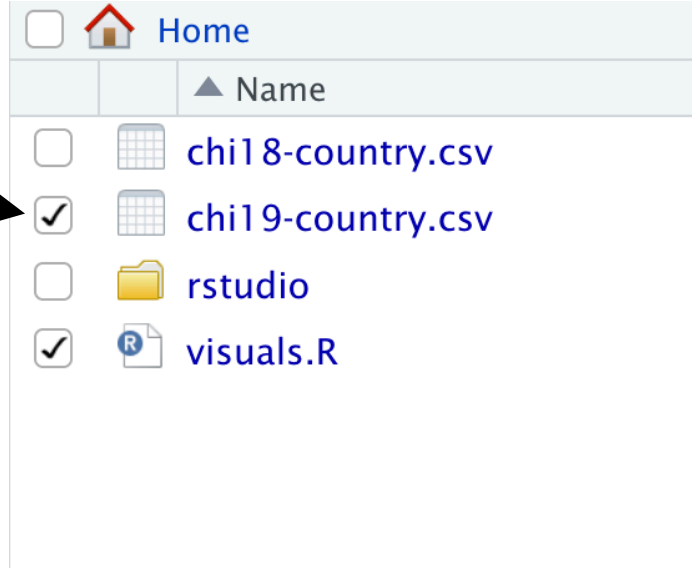
Until you reach:

8. We're done… for now!

Getting your code

Download your files before you lose them!

Select the files you want to keep



Getting your code

Download your files before you lose them!

Then click More ➔ Export…

To get a zip file.