INST0065
Data Visualization and GIS

Week 5: Bar charts and more

Dr. Oliver Duke-Williams

o.duke-williams@ucl.ac.uk
(Please use Moodle forums for messages about this module)

Twitter: @oliver_dw

INFORMATION STUDIES

Celebrating 100 years

- So far we have predominantly looked at scatterplots
- **This week** we will use another scatterplot to look at two things
  - Merging data
  - Adjusting colours
- We will then look more properly at bar charts

Last week

- Last week we also introduced RMarkdown
- The examples this week will be developed as RMarkdown files

RMarkdown files

- You will be provided (via Moodle) with RMarkdown files and associated data files
  - You should use these files to follow the examples
  - Sometimes you will be asked to add code to the markdown file

Assessment

- Choose (individually) a data visualization
- Show that visualization and explain
  - What it shows
  - How it shows it, using the terms outlined by Few (2004)
    - Visual attributes
    - 'Quantitative messages'
  - Strengths or weaknesses of the visualization
    - Ease of understanding
    - Assumptions that might be embedded in the data

Assessment

- Word count: maximum 1000 words
  - Do not include captions, footnotes, bibliography
- Advice
  - Choose a visualization of data that will be straightforward to describe in terms of Few(2004)

Merging data

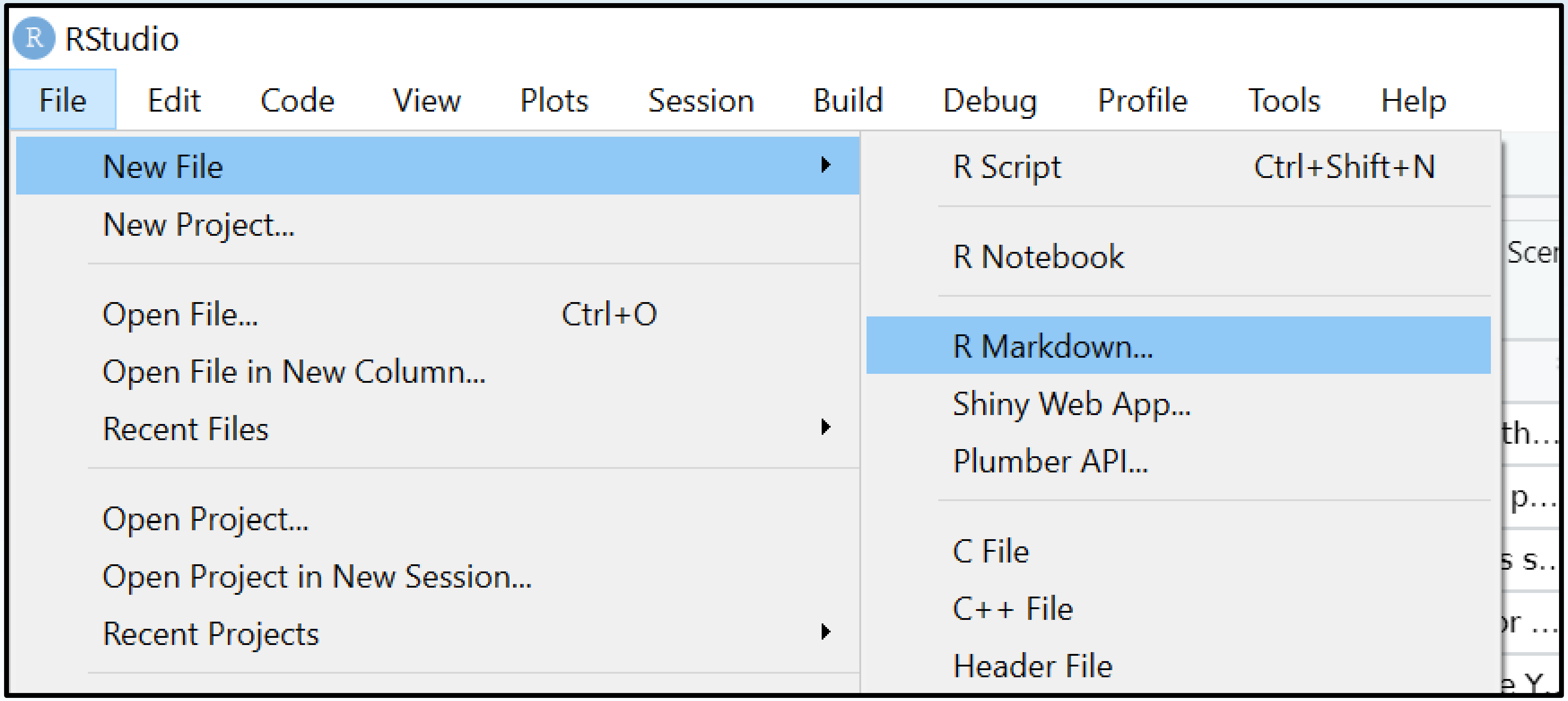
- A common scenario in analysis is that we will have data from multiple sources that we want to join together
- We will look at an example of this, and how we can use merge() to join data
- We will take a file of constituency level data relating to the 2016 EU Referendum, and join it to a file of data containing 2015 General Election results

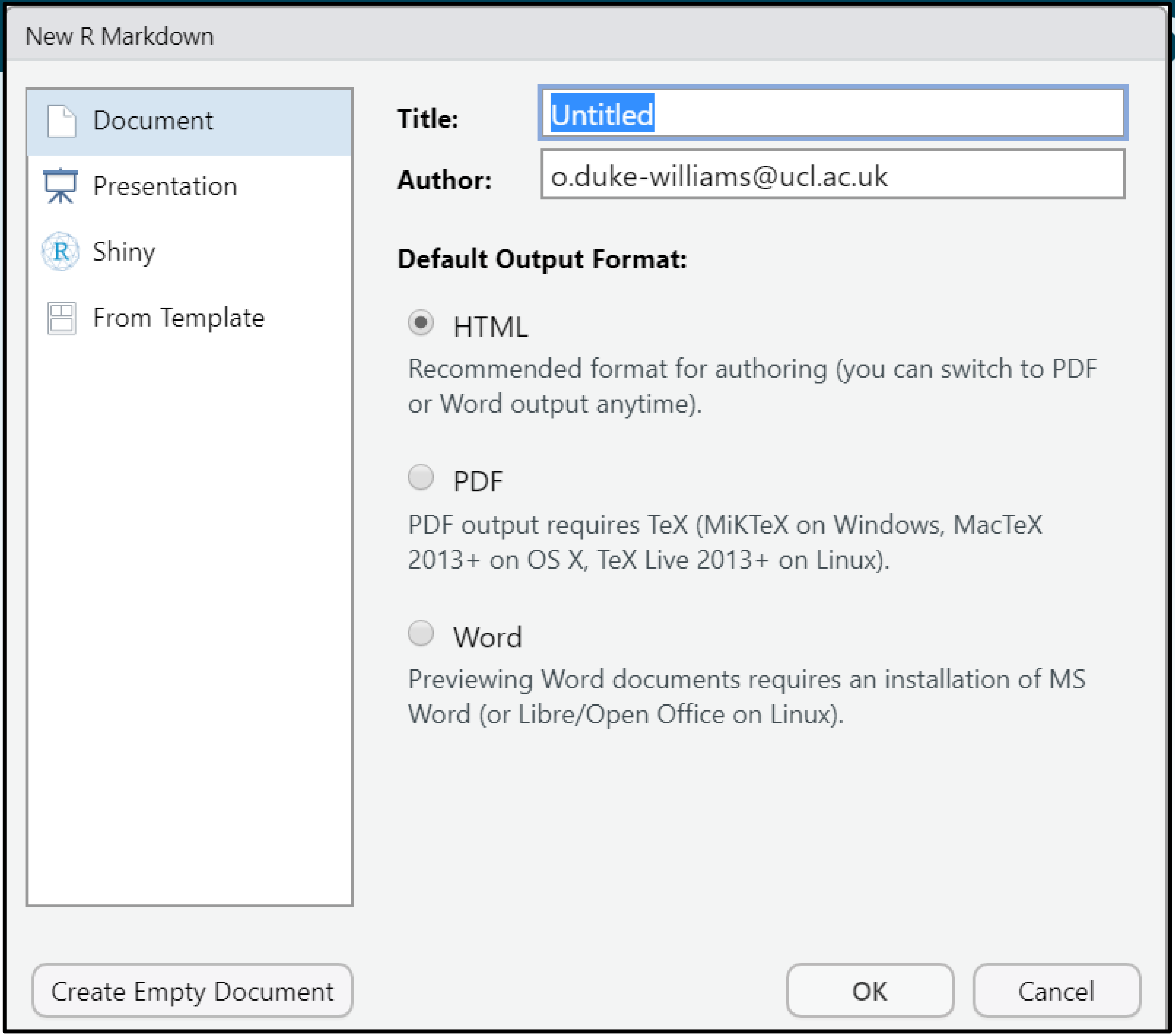
Getting started with RMarkdown

- As a reminder, we can either use an existing template, simply start writing a file from scratch, or use the example 'new file' template in RStudio
- We'll do the latter
  - In RStudio
    - File -> New file -> RMarkdown
    - Set title and check author details
  - In RMarkdown file
    - Delete everything after initial options

Getting started with RMarkdown

- This example will show how we use RMarkdown and how we use R to produce the outputs we are looking for
  - Some screenshots will be used to illustrate the RMarkdown process
  - Mostly we will concentrate on the actual code
  - Screenshots show how the file was created; the finished file is on Moodle


RStudio
File
Edit
Code
View
Plots
Session
Build
Debug
Profile
Tools
Help
New File
New Project...
Open File... Ctrl+O
Open File in New Column...
Recent Files
Open Project...
Open Project in New Session...
Recent Projects
R Script Ctrl+Shift+N
R Notebook
R Markdown...
Shiny Web App...
Plumber API...
C File
C++ File
Header File



New R Markdown

Document
Presentation
Shiny
From Template

Title: Untitled

Author: o.duke-williams@ucl.ac.uk

Default Output Format:

(●) HTML

Recommended format for authoring (you can switch to PDF or Word output anytime).

( ) PDF

PDF output requires TeX (MiKTeX on Windows, MacTeX 2013+ on OS X, TeX Live 2013+ on Linux).

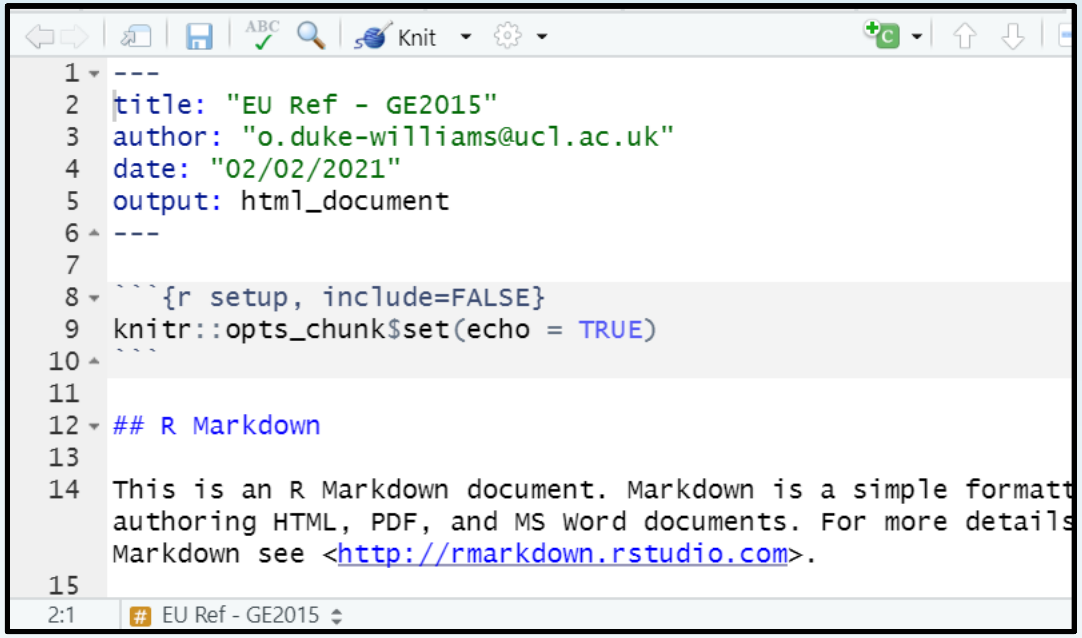
( ) Word

Previewing Word documents requires an installation of MS Word (or Libre/Open Office on Linux).

Create Empty Document

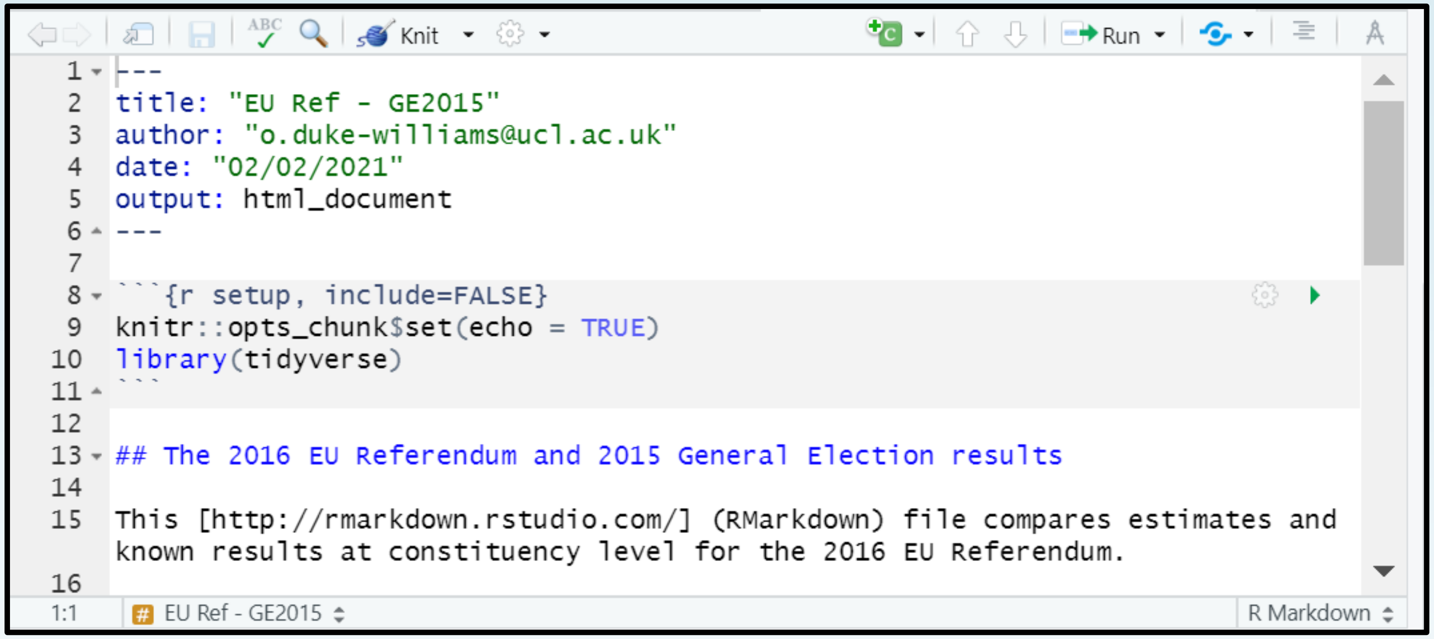
OK

Cancel



~~~
---
title: "EU Ref - GE2015"
author: "o.duke-williams@ucl.ac.uk"
date: "02/02/2021"
output: html_document
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```

## R Markdown

This is an R Markdown document. Markdown is a simple formatt
authoring HTML, PDF, and MS Word documents. For more details
Markdown see <http://rmarkdown.rstudio.com>.

~~~

- Make sure that the tidyverse library is loaded
- Delete content
- Start to replace text; use markdown formatting
  - See https://rstudio.com/wp-content/uploads/2015/02/rmarkdown-cheatsheet.pdf



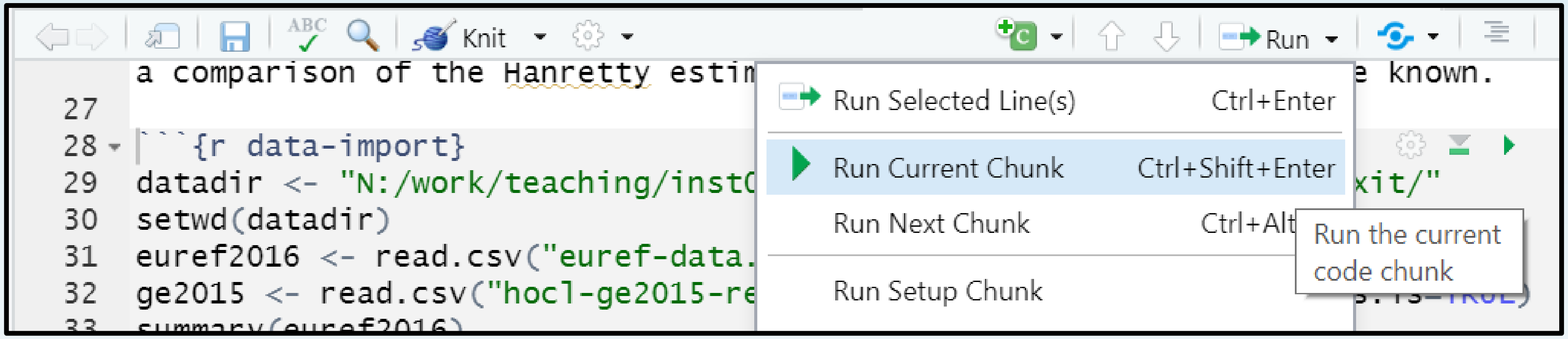
```
 1  ---
 2  title: "EU Ref - GE2015"
 3  author: "o.duke-williams@ucl.ac.uk"
 4  date: "02/02/2021"
 5  output: html_document
 6  ---
 7
 8  ```{r setup, include=FALSE}
 9  knitr::opts_chunk$set(echo = TRUE)
10  library(tidyverse)
11  ```
12
13  ## The 2016 EU Referendum and 2015 General Election results
14
15  This [http://rmarkdown.rstudio.com/] (RMarkdown) file compares estimates and
    known results at constituency level for the 2016 EU Referendum.
16
```

- Here we have
  - Loaded the tidyverse library (line 10)
  - Started to replace some of the text (line 13 onwards)
- Assume that you are going to give the results to someone else
  - Make sure that you explain what you are doing, and the data

~~~
a comparison of the Hanretty estimates and the actual results where
known.

```{r data-import}
datadir <- "N:/work/teaching/inst0065-dataviz-gis/week5/ge2015-
brexit/" # datadir will need to be changed for each person
setwd(datadir)
euref2016 <- read.csv("euref-data.csv",header=TRUE,as.is=TRUE)
ge2015 <- read.csv("hocl-ge2015-results-
summary.csv",header=TRUE,as.is=TRUE)
summary(euref2016)
summary(ge2015)
```
## Comparing the estimates and observed results

How good were Chris Hanretty's initial estimates when compared to
later observed actual constituency level results? We will use a
~~~

- Note how we intersperse commentary and R code

Working with RMarkdown


Knit
Run
Run Selected Line(s) Ctrl+Enter
Run Current Chunk Ctrl+Shift+Enter
Run Next Chunk Ctrl+Alt
Run Setup Chunk
Run the current code chunk

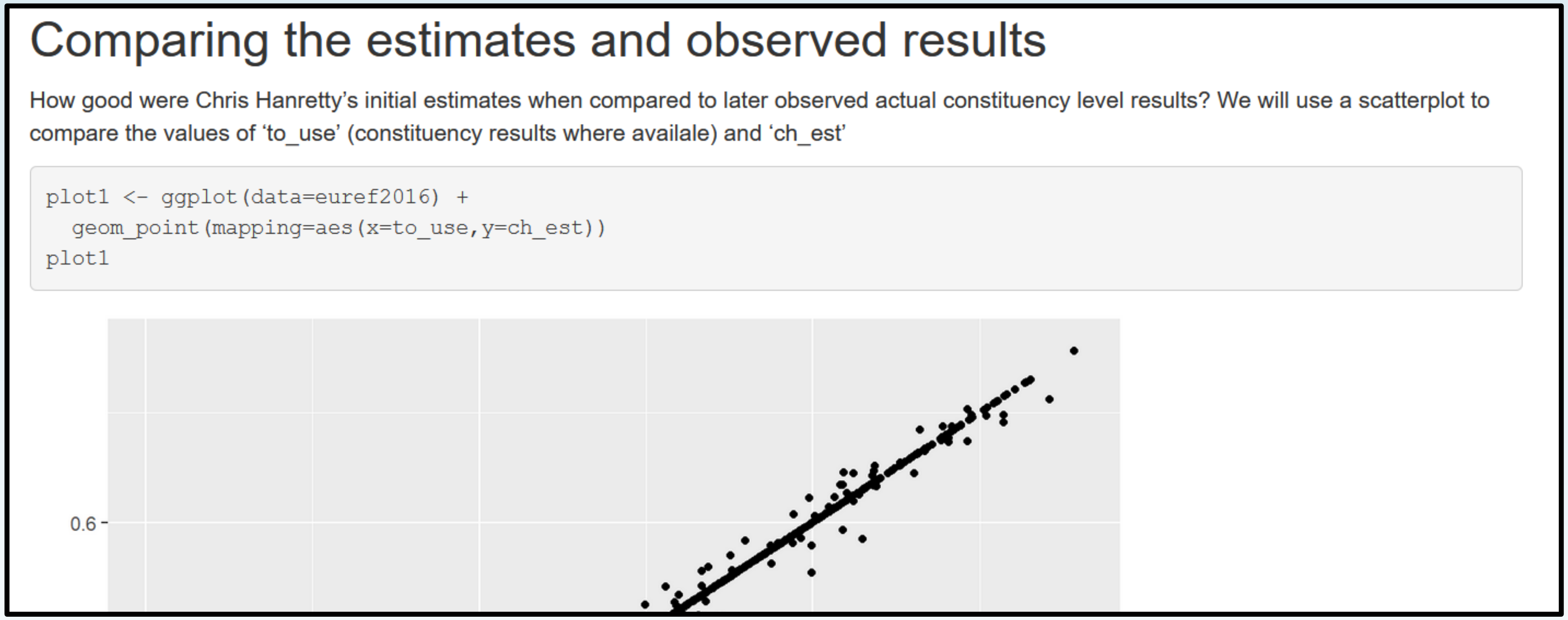
~~~
a comparison of the Hanretty estim                          e known.

```{r data-import}
datadir <- "N:/work/teaching/instC                          xit/"
setwd(datadir)
euref2016 <- read.csv("euref-data.
ge2015 <- read.csv("hocl-ge2015-re
summary(euref2016)
~~~

- Each chunk should do a distinct task, rather than have a monolithic amount of code
- Each chunk should have a **unique chunk name**
- Each chunk can be tested as you write the code
- You should check that the whole file works as expected before passing it on

From RMarkdown file to knitted output…

~~~
```{r basic-plot}
plot1 <- ggplot(data=euref2016) +
  geom_point(mapping=aes(x=to_use,y=ch_est))
plot1
```


There's a heavy line along the diagonal where we have a lot
of results. Lets use alpha to try to make it easier to read.
~~~

- This is an excerpt from our RMarkdown file
- There is some R code in a chunk, indicated by backticks: ```
- This is creating a plot and assigning it to an object
- When we use that object name, the plot will be displayed



Comparing the estimates and observed results

How good were Chris Hanretty's initial estimates when compared to later observed actual constituency level results? We will use a scatterplot to compare the values of 'to_use' (constituency results where availale) and 'ch_est'

```
plot1 <- ggplot(data=euref2016) +
  geom_point(mapping=aes(x=to_use,y=ch_est))
plot1
```

- This is part of the output after we 'knit' the RMarkdown file
- The R code from the file is repeated
- The output is merged in

Filtering data

~~~
```{r filter-data}
euref_filt <- euref2016 %>% filter(result == "Yes")
```


We'll now replot the data.

```{r filtered-plot}
plot3 <- ggplot(data=euref_filt) +
  geom_point(mapping=aes(x=to_use,y=ch_est))
plot3
```
~~~

- Here, we have filtered our euref2016 dataset
- We only retain lines for which 'result' is 'Yes'
- We then plot that as before

RMarkdown files as a research record

- We can use RMarkdown files as a notebook to document our work
- In this example, we have produced a plot and concluded that it is too busy
- We then filter the data
- Writing the RMarkdown file is not a 'one-off' process, they can be ongoing and reflective

Back to our example…

- We have produced a plot of estimated vs observed data
- In this case, we might wonder whether voting behaviour has patterns related to the referendum vote
- We could colour each dot according to the party that won that seat
- But
  - Party information isn't in the referendum data
  - We have that information in the GE2015 results data that we also loaded (because we knew we'd get to this point..)
  - We need to merge the two together

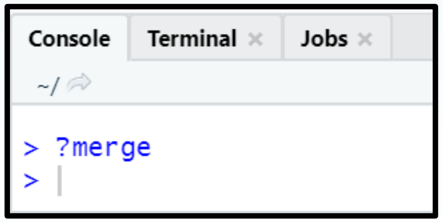
Merging data

~~~
###Merging data

We join the data based on the constituency, using the common
field 'ons_id'

```{r merge-data}
refge <- merge(euref2016,ge2015,by="ons_id")
```
~~~

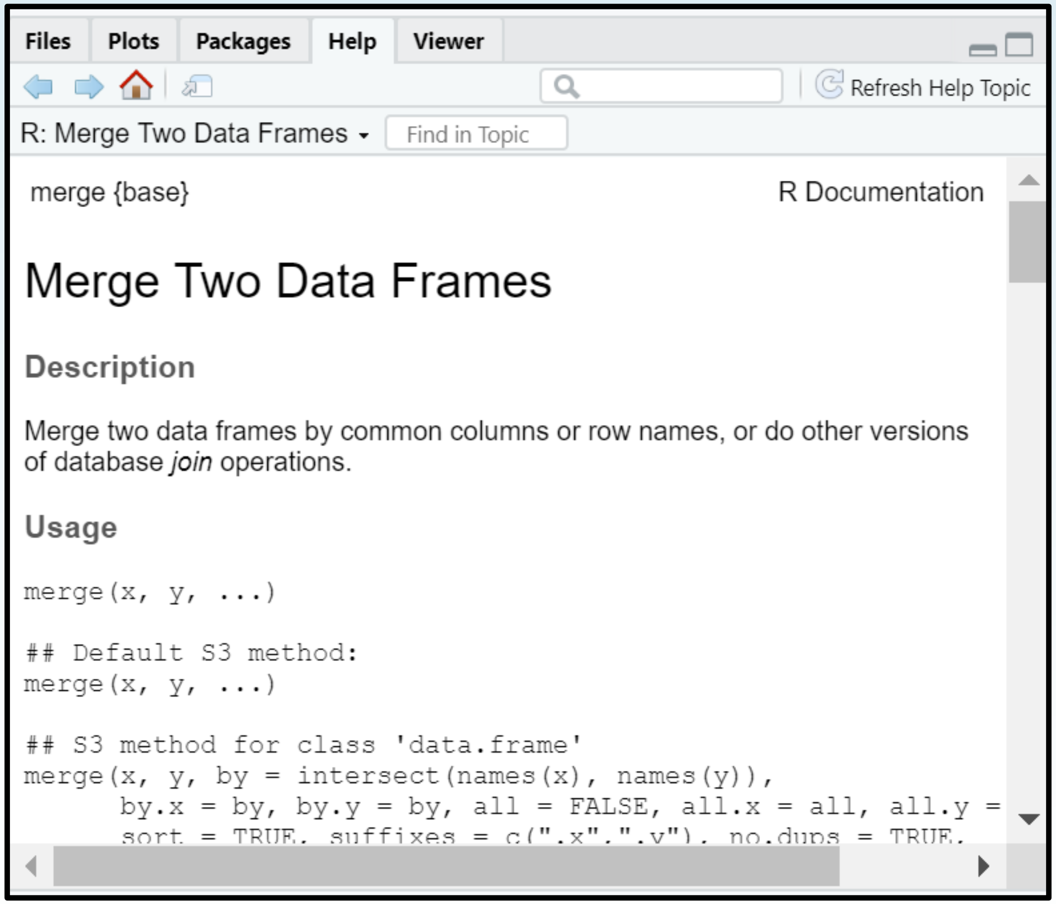
- See R documentation for merge()
  - https://www.rdocumentation.org/packages/data.table/versions/1.13.6/topics/merge

Documentation

- As well as online sources, in R we can get documentation on any command using '?command'



```
> ?merge
>
```



merge {base} R Documentation

Merge Two Data Frames

Description

Merge two data frames by common columns or row names, or do other versions of database *join* operations.

Usage

```
merge(x, y, ...)

## Default S3 method:
merge(x, y, ...)

## S3 method for class 'data.frame'
merge(x, y, by = intersect(names(x), names(y)),
      by.x = by, by.y = by, all = FALSE, all.x = all, all.y =
      sort = TRUE, suffixes = c(".x",".y"), no.dups = TRUE,
```

Merging data

~~~
```{r merge-data}
refge <- merge(euref2016,ge2015,by="ons_id")
```
~~~

- To merge, we first identify the two tables to merge
- These are identified internally as 'x' and 'y'
- We can specify these explicitly if we want to

```
merge(x="euref2016",y="ge2015"…)
```

Merging data

~~~
```{r merge-data}
refge <- merge(euref2016,ge2015,by="ons_id")
```
~~~

- Next we need to identify a field with common values on which to merge, using "by="
- In this case we have a field with the same name in both input tables
- If the relevant fields have different names, we can use by.x= and by.y=

```
merge(x="tabA",y="tabB",by.x="some_id",by.y="id"…)
```

Merging data

- Where the key values (i.e. the common field(s)) in both tables are fully consistent, the merge should make sense
- What happens if one of the tables does not have a value for each id in the other table?
  - In the exercises, we have a separate RMarkdown file, which creates two sample data frames and merges them together
  - You are asked to try different options for merge()

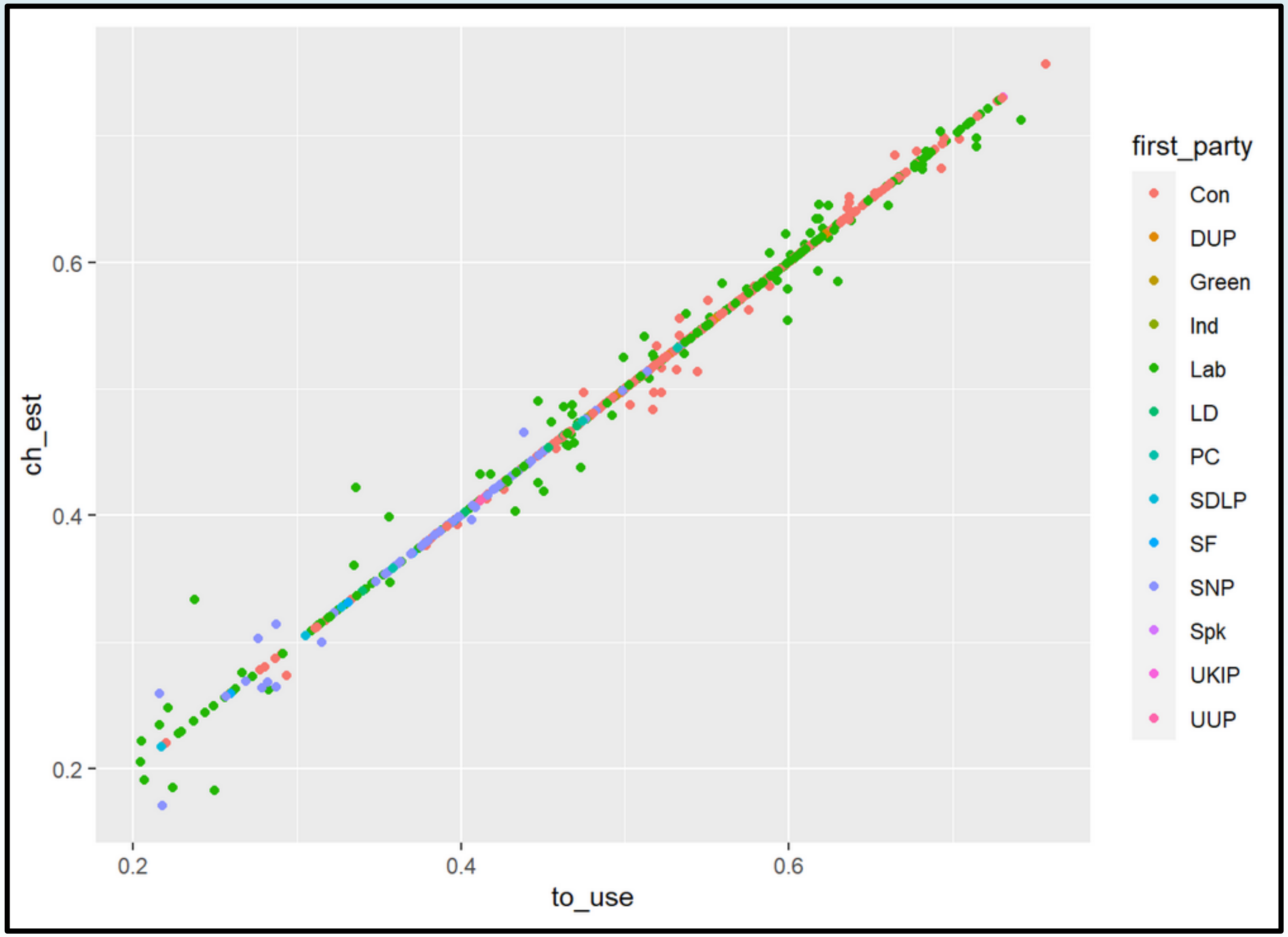
Plotting merged data

- Our merged data file contains values from both input datasets
  - If there are fields in the two tables with the same column name (they may or may not be related) R will include both and add .x and .y suffixes
- We can now plot that merged data set

Colour by category

~~~
```{r plot-parties-all}
plot4 <- ggplot(data=refge) +

geom_point(mapping=aes(x=to_use,y=ch_est,colour=first_party)
)
plot4
```
~~~

- We have added 'colour=first_party' to our plot command
- This tells R two things
  - We want plotted dots to be coloured
  - That is should use 'first_[arty' to do this. Every record that has the same value in this field will be plotted the same colour
  - 'first_party' is a field in the results data that tells us which party won that seat



- The dots are coloured, but not the colours we might want!

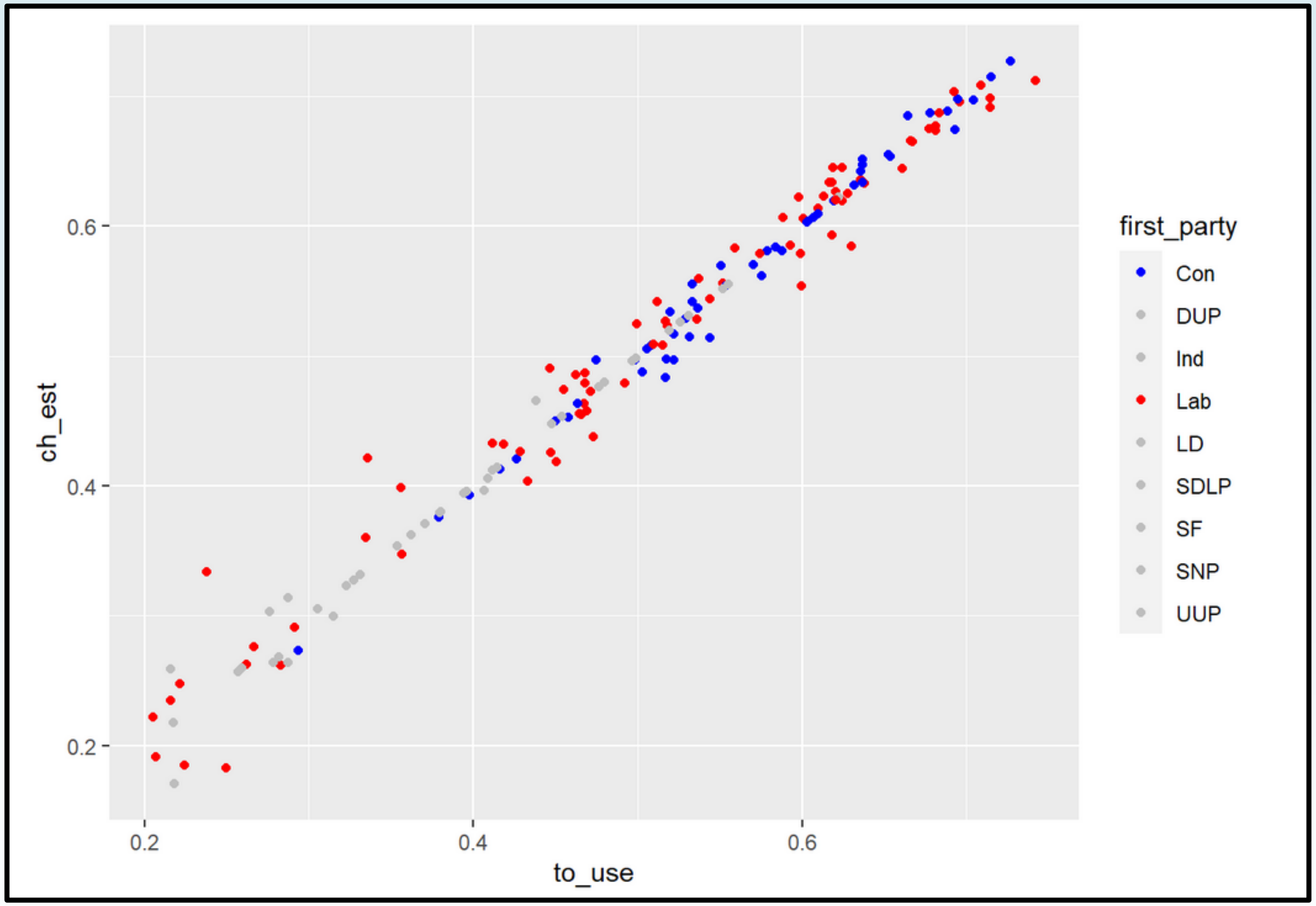
Adjusting colours

- There are various ways of adjusting colour
- In the RMarkdown file, it is explained that we have used **scale_colour_manual()** to override the default colours
- This requires a vector of colour definitions; the length of that vector needs to be the same as the number of categories we have
- We need to look at our results in order to build it; again, this is the RMarkdown file as a reflective document

Using scale_colour_manual()

~~~
```{r plot-parties-filt-colour}
cols <-
c("blue","grey","grey","red","grey","grey","grey","grey","grey")
plot6 <- ggplot(data=refge_filt) +
  geom_point(mapping=aes(x=to_use,y=ch_est,colour=first_party)) +
  scale_colour_manual(values=cols)
plot6
```
~~~

- We build a vector 'cols'
- We have to ensure it is the right length, and that we follow the same order
- Actually using that vector is then fairly easy



- Two categories have had colours applied

Bar charts

- Scatterplots require a series of x.y observations, and generally plot each one
- Bar charts present an aggregate of date
  - R, by, default, assumes data is a series of individual records, and counts how many fall into each group
  - We can also have data that contain a pre-calculated total

Data sources

- We will mostly use the 'diamonds' data set
  - This is a sample data set included in the tidyverse group of libraries
- It contains a set of individual observations about diamonds
- This is also used as an example in *R for Data Science*

Exploring bar charts

- As with the EU Referendum example, we will use an RMarkdown file to store R commands and outputs
- This should be consulted in conjunction with these slides/videos
- During the practical period, you should work through these RMarkdown files making sure you understand them

A bar chart

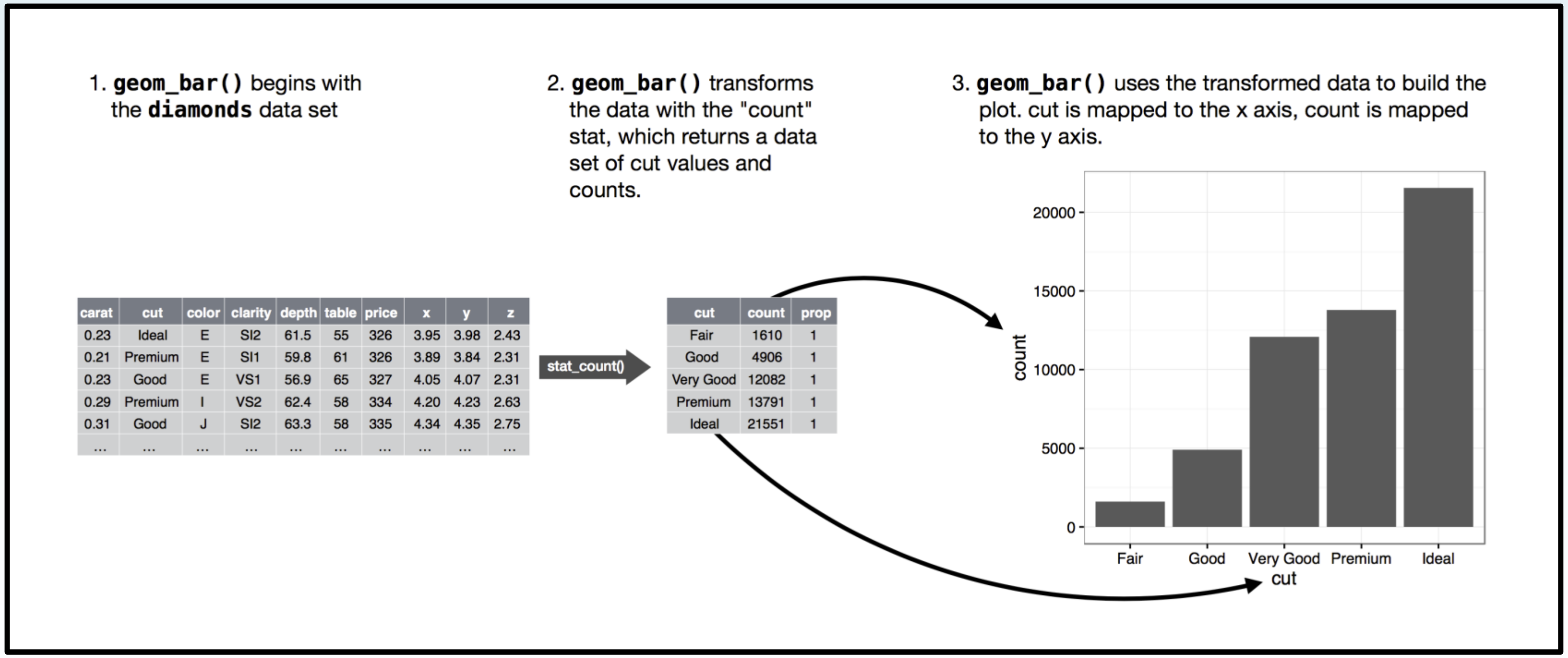
- We can create bar charts using geom_bar()
- We use this in a similar way to geom_point()

```
ggplot(data=diamonds) +
   geom_point(mapping=aes(x=carat,y=price))


ggplot(data=diamonds) +
   geom_bar(mapping=aes(x=cut))
```

```
ggplot(data=diamonds) +
   geom_point(mapping=aes(x=carat,y=price))


ggplot(data=diamonds) +
   geom_bar(mapping=aes(x=cut))
```

- Both start with ggplot() identitying a data set
  - Both then add a layer; geom_point() in the first case, geom_bar() in the second
- Each geom function will draw a layer, and each must have a mapping, which says how the data are mapped to visual properties
  - 'mapping' is always coupled with aes(), an *aesthetic*.
- Point layers require x and y variables
- Bar charts just require an x variable (to start with)

Transforming data



1. **`geom_bar()`** begins with the **`diamonds`** data set

2. **`geom_bar()`** transforms the data with the "count" stat, which returns a data set of cut values and counts.

3. **`geom_bar()`** uses the transformed data to build the plot. cut is mapped to the x axis, count is mapped to the y axis.

carat	cut	color	clarity	depth	table	price	x	y	z
0.23	Ideal	E	SI2	61.5	55	326	3.95	3.98	2.43
0.21	Premium	E	SI1	59.8	61	326	3.89	3.84	2.31
0.23	Good	E	VS1	56.9	65	327	4.05	4.07	2.31
0.29	Premium	I	VS2	62.4	58	334	4.20	4.23	2.63
0.31	Good	J	SI2	63.3	58	335	4.34	4.35	2.75
...	...	...	...	...	...	...	...	...	...

cut	count	prop
Fair	1610	1
Good	4906	1
Very Good	12082	1
Premium	13791	1
Ideal	21551	1

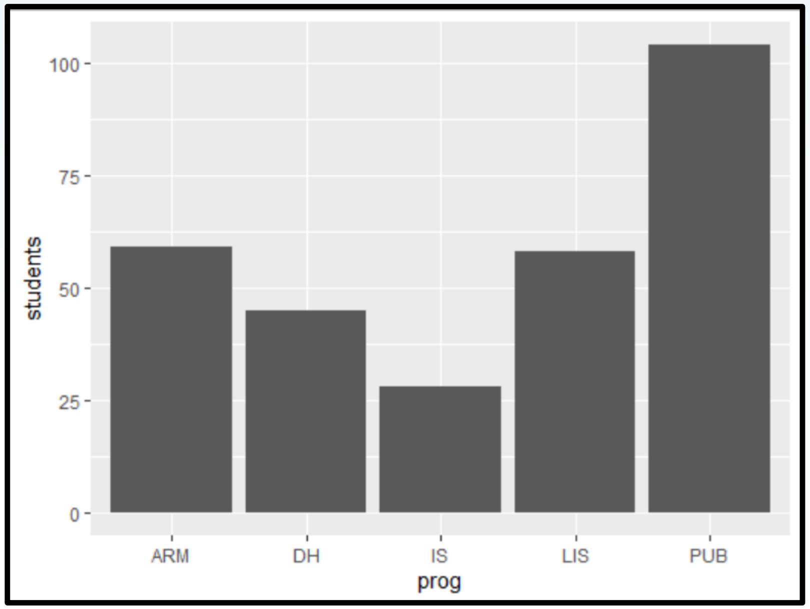
Source: R for Data Science

count vs. identity

- The default behaviour of geom_bar is to do a count of records in each category
- We can use pre-calculated values instead; we often have data like this

~~~
```{r bar-identity}
df1 <-
data.frame(prog=c("ARM","DH","IS","LIS","PUB"),students=c(59
,45,28,58,104))
plot2 <- ggplot(data = df1) +
  geom_bar(mapping=aes(x=prog,y=students),stat="identity")
```
~~~

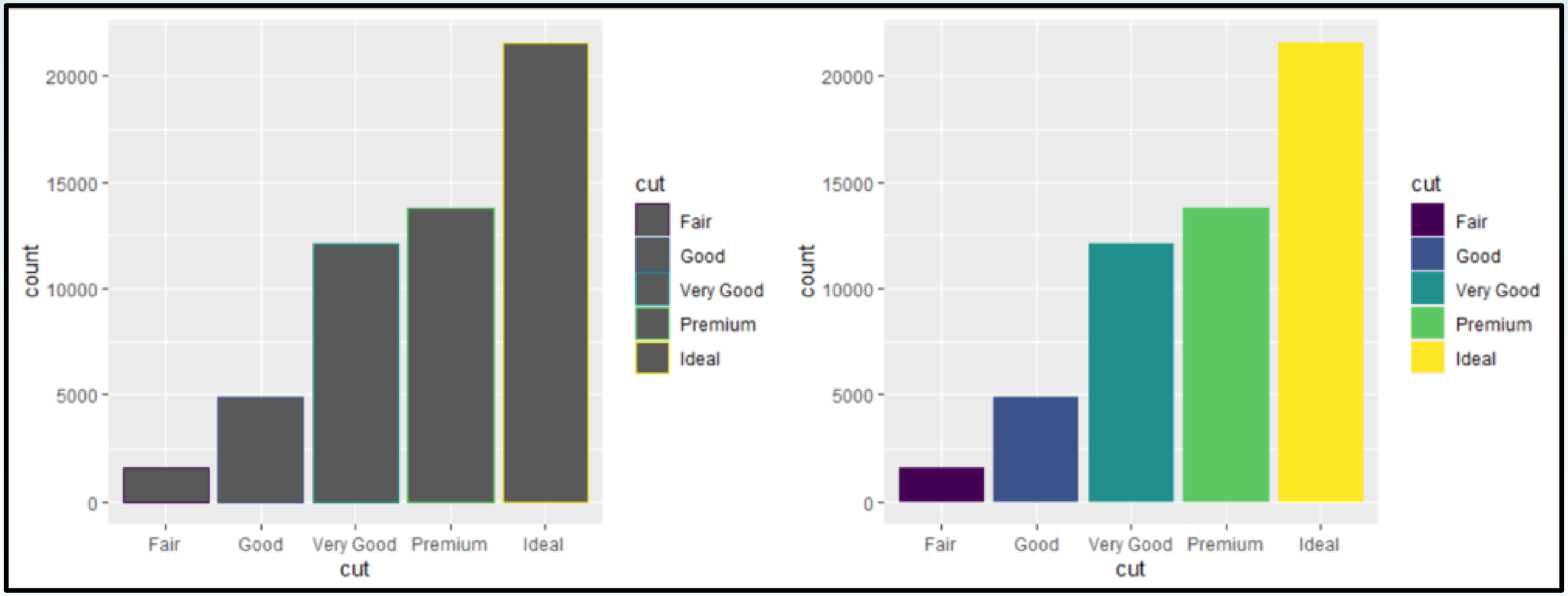
- We have some data showing the total numbers of students on different programmes
- By adding a y term, we can say which field to use to size the bar



Using colour

- We can adjust the outline and fill colours of bars related to each category by adding a colour term to the aesthetic

~~~
```{r barchart-colours}
plot3 <- ggplot(data = diamonds) +
 geom_bar(mapping=aes(x=cut, colour=cut))

plot4 <- ggplot(data = diamonds) +
 geom_bar(mapping=aes(x=cut, fill=cut))
```
~~~



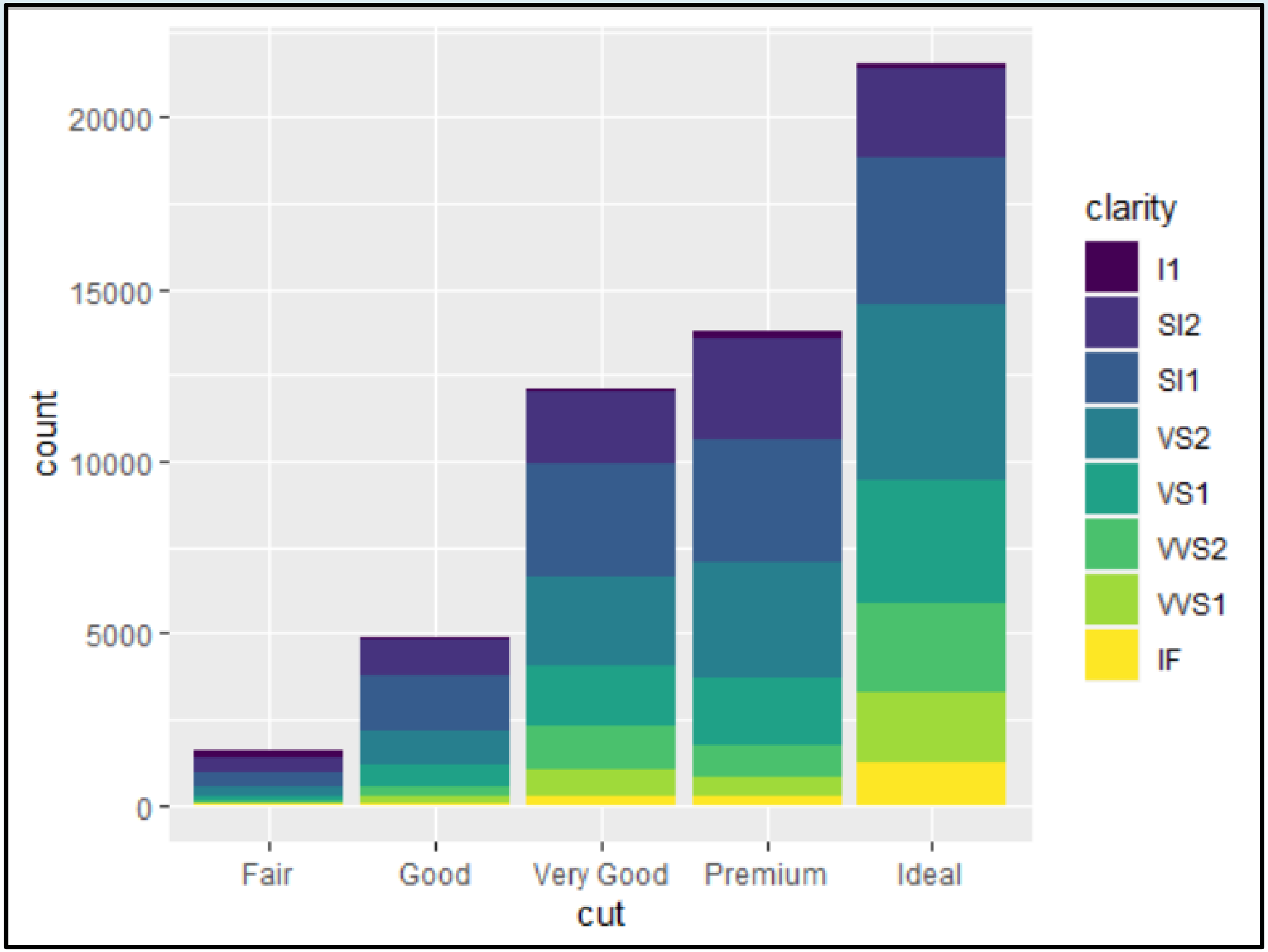
count
20000
15000
10000
5000
0
Fair
Good
Very Good
Premium
Ideal
cut
cut
Fair
Good
Very Good
Premium
Ideal
count
20000
15000
10000
5000
0
Fair
Good
Very Good
Premium
Ideal
cut
cut
Fair
Good
Very Good
Premium
Ideal

Using multiple characteristics

- In the previous graph, we set x=cut and fill=cut – both the size and the colour of the bar were being controlled by the same variable
- We don't have to do this – we can use a fill colour based on a second variable

~~~
```{r barchart-cut-clarity}
plot7 <- ggplot(data = diamonds) +
  geom_bar(mapping=aes(x=cut, fill=clarity))
plot7
```
~~~

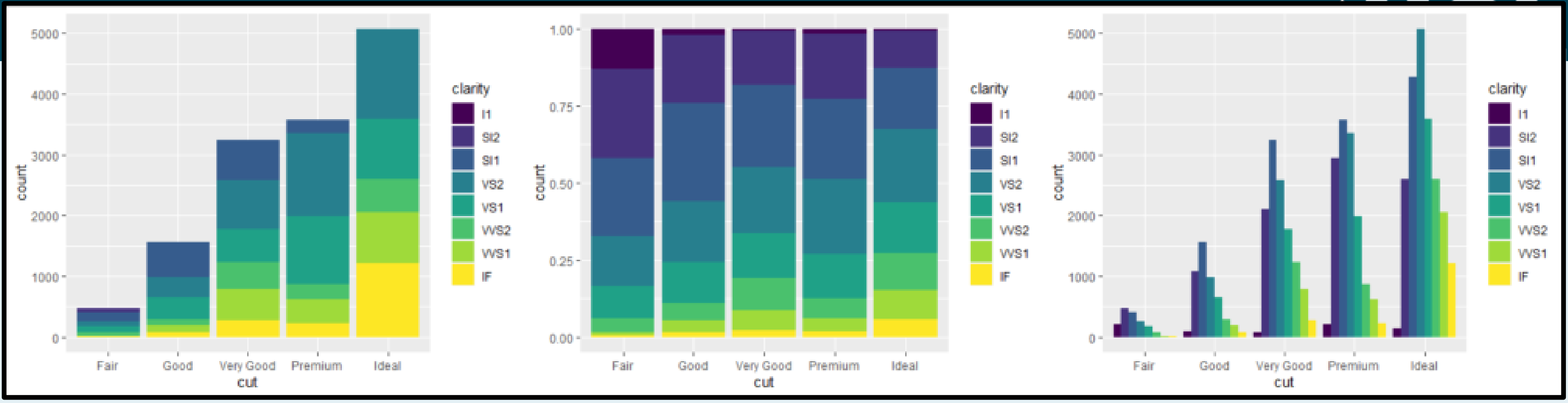
- Here we are again creating a bar chart
- We have used x=cut
  - This means we will have one bar for each value of 'cut'
  - The height of the bar will be based on the total number of records for each 'cut' type
- We have used fill=clarity
  - All records of a given 'clarity' type will be coloured the same way
  - It doesn't matter what order the original records were in, R will group them together to colour them



count
20000
15000
10000
5000
0
Fair
Good
Very Good
Premium
Ideal
cut
clarity
I1
SI2
SI1
VS2
VS1
VVS2
VVS1
IF

Alternative arrangements

- In the RMarkdown file, you will learn about other data arrangements – identity, dodge, and fill
  - These affect how the bar chart elements are positioned



```
> ggplot(data = diamonds) +
   geom_bar(mapping=aes(x=cut,fill=clarity),position="identity")
> ggplot(data = diamonds) +
   geom_bar(mapping=aes(x=cut,fill=clarity),position="fill")
> ggplot(data = diamonds) +
   geom_bar(mapping=aes(x=cut,fill=clarity),position="dodge")
```

- Note that we add these terms at the layer level, not at the aesthetic (aes) level

Options at aesthetic level and options at plot level

- We have seen that some options need to be added in the *aes* mapping, and some need to be added outside that (in the geom_bar() command)

```
ggplot(data = diamonds) +
 geom_bar(mapping=aes(x=cut, colour=cut))


ggplot(data = diamonds) +
 geom_bar(mapping=aes(x=cut,fill=clarity),position="identity")
```

Options at aesthetic level and options at plot level

- How do we know where to put commands?
- This can be confusing
- The aes() mapping is also to do with how variables are visually presented
  - Especially about how different variables within one plot will be presented differently
  - Different locations (x,y), different categories (x), different colouring based on the data

Options at aesthetic level and options at plot level

- Options that apply to the whole layer affect how everything in that layer looks
  - Colour of all bars / points etc
  - How different variables are going to be assembled together

```
ggplot(data = diamonds) +
 geom_bar(mapping=aes(x=cut,fill=clarity),position="identity")
```

Appearance based on data values

Arrangement of plot independent of data values