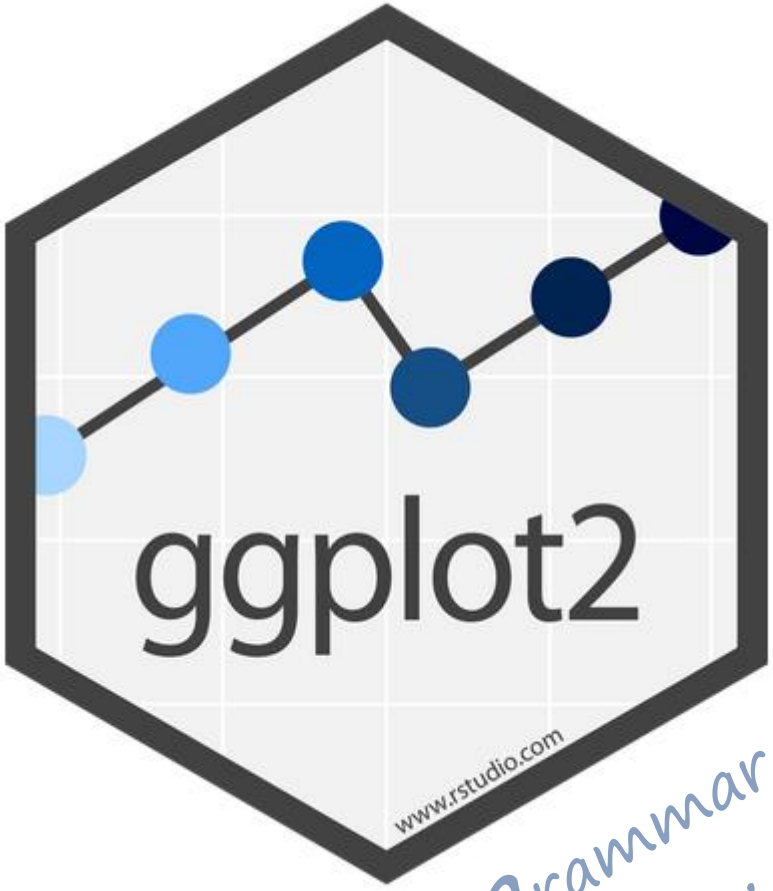
Introduction to

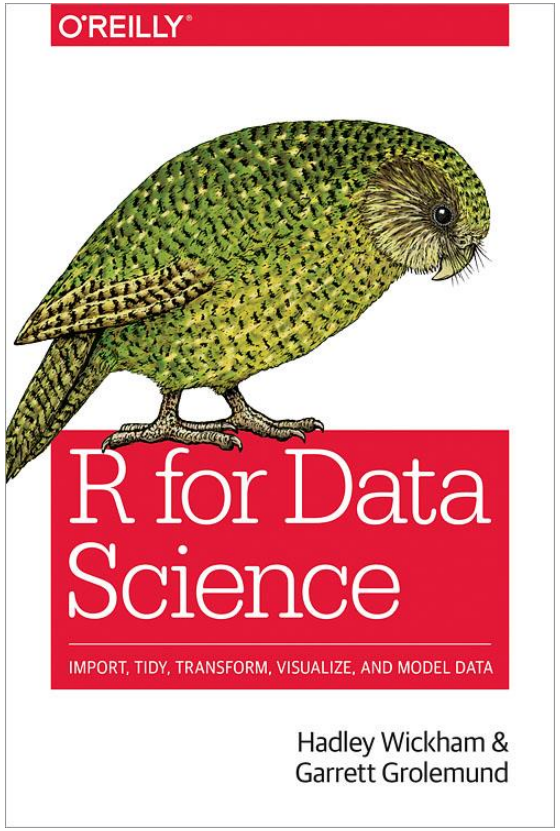


A Grammar of Graphics

Andrew Jones | Strategy Unit

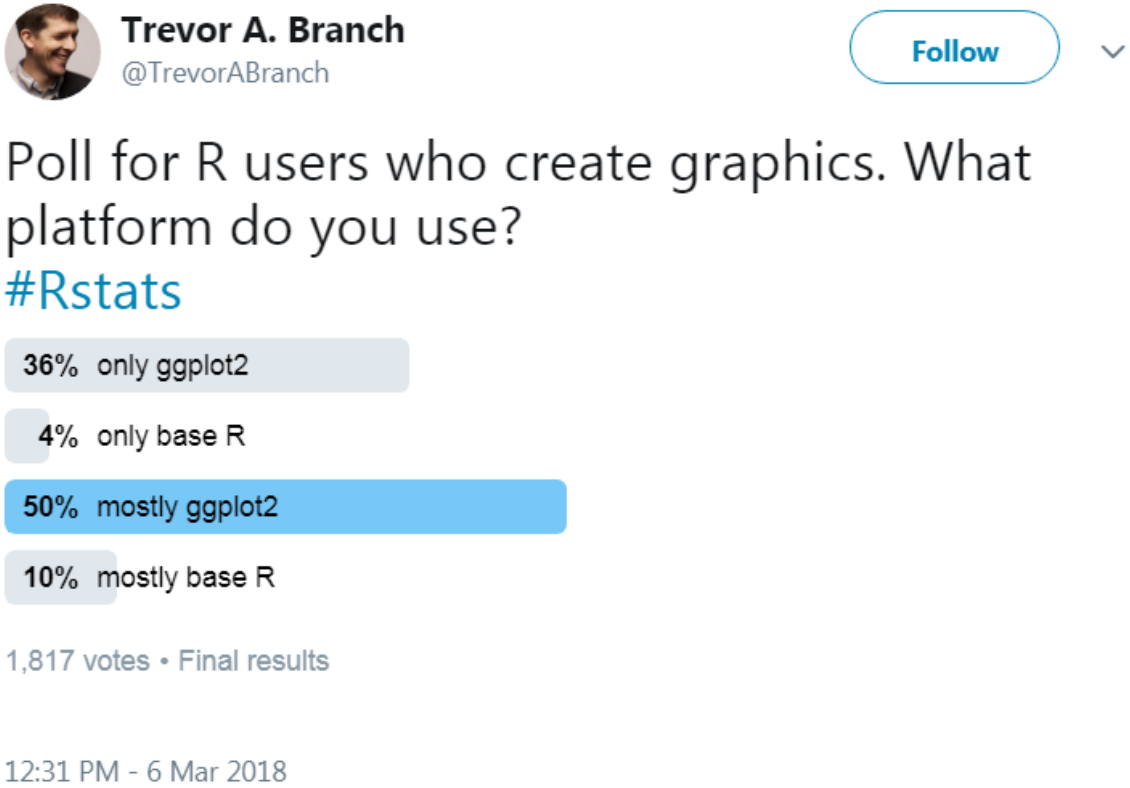
Acknowledgement

This session shadows Chapter 3 of the excellent:



ggplot2

Is one of several plotting systems in R



Why is ggplot popular?

1. Well designed and supported
2. Highly versatile
3. Attractive graphics (with a little work)

1. Why start with ggplot: http://varianceexplained.org/r/teach_ggplot2_to_beginners/
2. Argument against: https://simplystatistics.org/2016/02/11/why-i-dont-use-ggplot2/

ggplot2

ggplot2 is part of the tidyverse.

So, at the top of your script type:

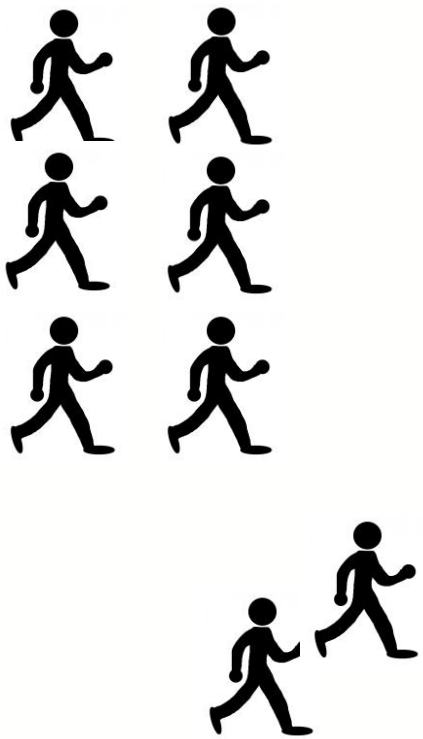
```
library(tidyverse)
```

Project 1:

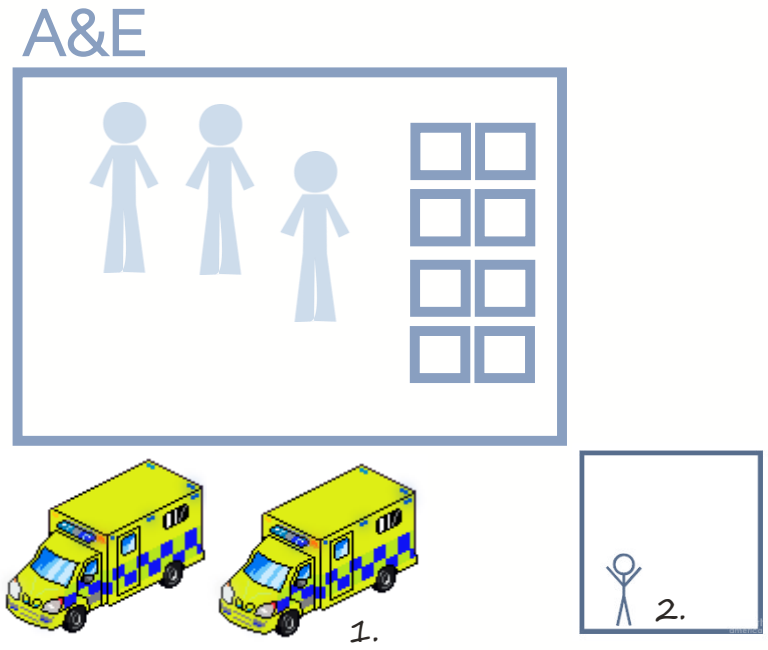
Let’s explore a perennial challenge for the NHS:

Pressures in A&E

Demand



Capacity



Data: Capacity in A&E

The dataset we loaded earlier, capacity_ae[1], shows changes in the capacity of A&E departments from 2017 to 2018.

1. Closely based on datasets collected by the NHS Benchmarking Network

Data: Capacity in A&E

The object named capacity_ae is a data frame.

What is a data frame?

A data frame stores tabular data:

Variables

Observations

id	sex	score
1	F	10.24
2	F	5.98
3	M	7.62

tibble = data frame

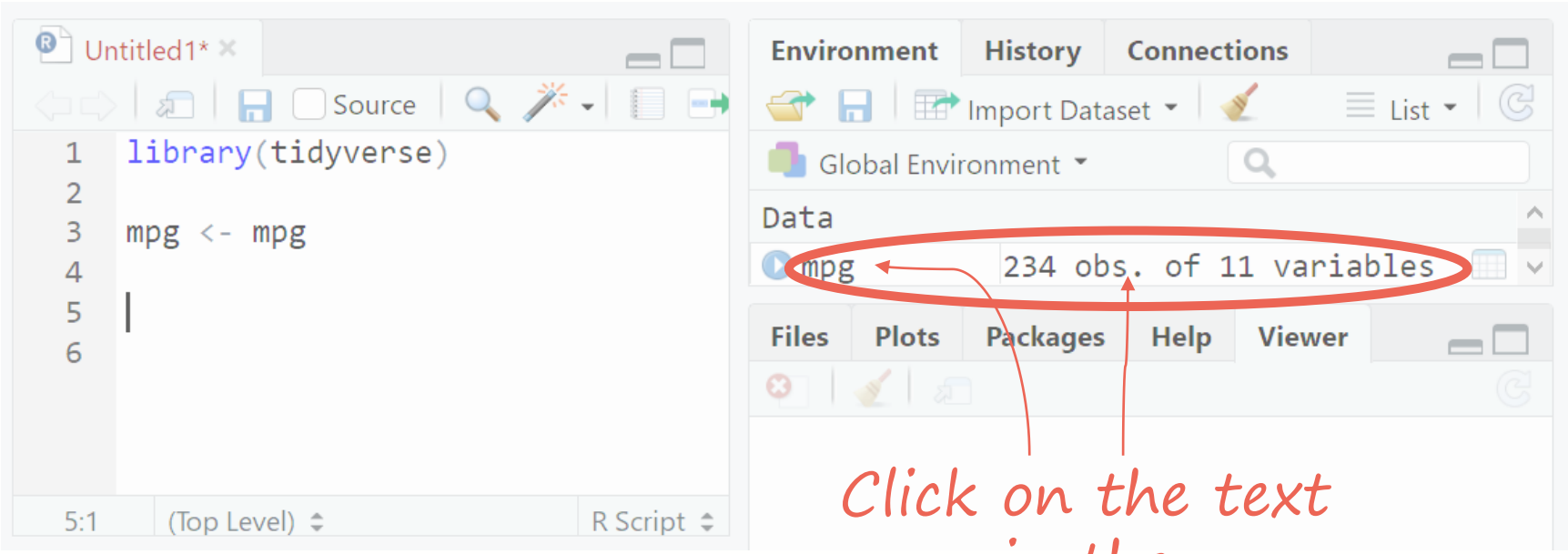
In the tidyverse you may see the term “tibble”.

We’ll take “tibble” to be synonymous with “data frame”

id	sex	score
1	F	10.24
2	F	5.98
3	M	7.62

Viewing the data frame

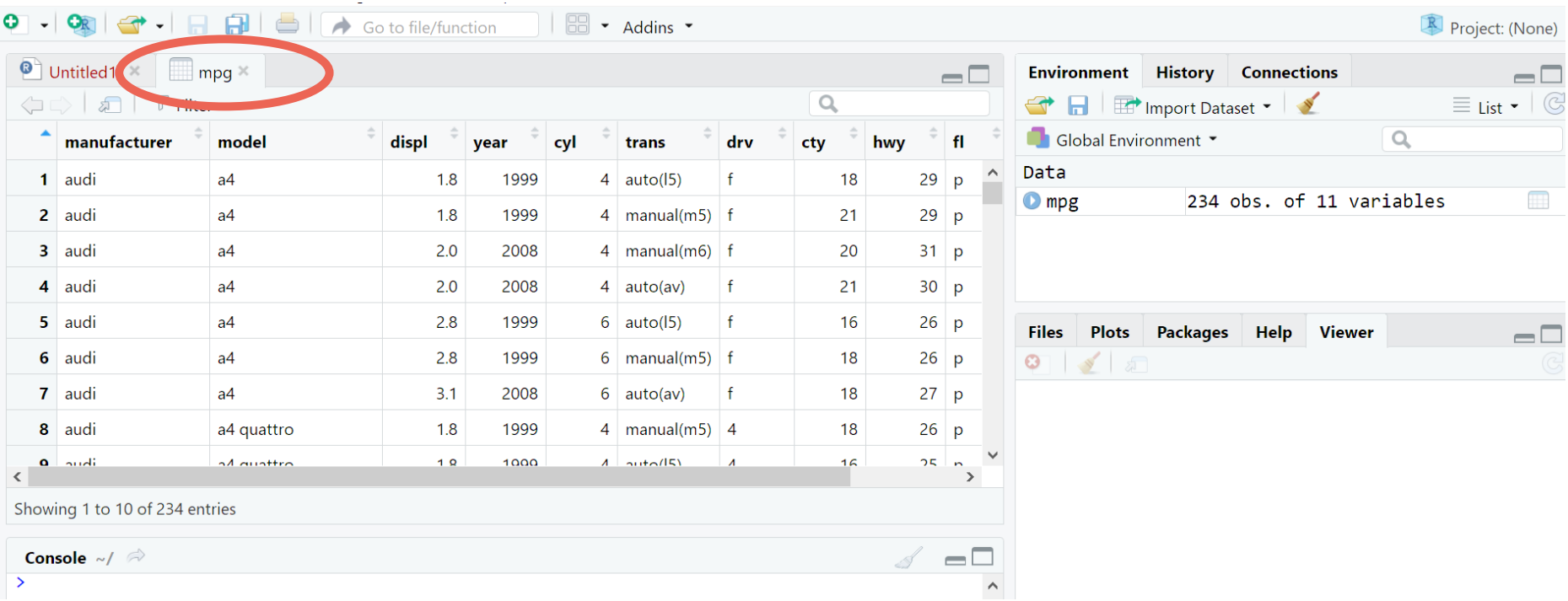
Option 1:



Click on the text in the "Environment" pane

Viewing the data frame

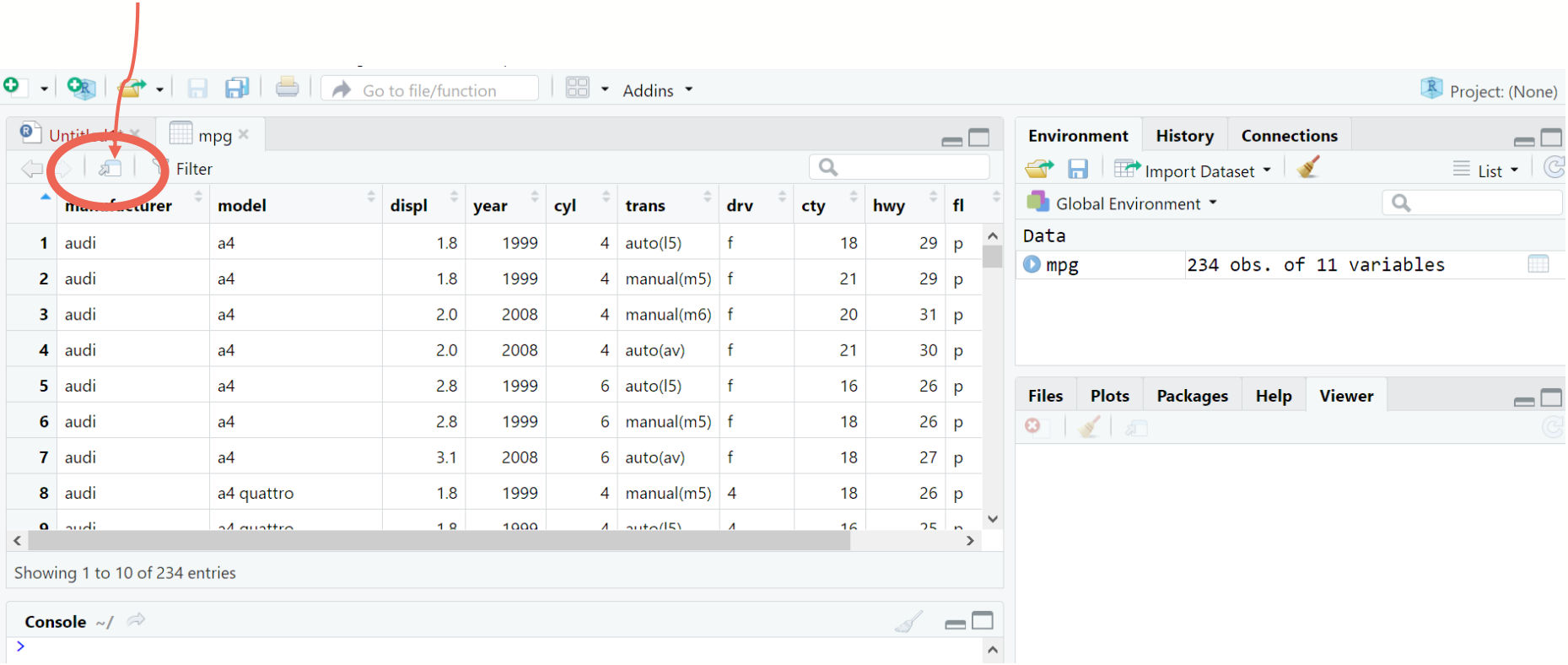
This brings up a view of the data in a new tab:



	manufacturer	model	displ	year	cyl	trans	drv	cty	hwy	fl
1	audi	a4	1.8	1999	4	auto(l5)	f	18	29	p
2	audi	a4	1.8	1999	4	manual(m5)	f	21	29	p
3	audi	a4	2.0	2008	4	manual(m6)	f	20	31	p
4	audi	a4	2.0	2008	4	auto(av)	f	21	30	p
5	audi	a4	2.8	1999	6	auto(l5)	f	16	26	p
6	audi	a4	2.8	1999	6	manual(m5)	f	18	26	p
7	audi	a4	3.1	2008	6	auto(av)	f	18	27	p
8	audi	a4 quattro	1.8	1999	4	manual(m5)	4	18	26	p

Viewing the data frame

Click here to show the data frame in a new window*



	manufacturer	model	displ	year	cyl	trans	drv	cty	hwy	fl
1	audi	a4	1.8	1999	4	auto(l5)	f	18	29	p
2	audi	a4	1.8	1999	4	manual(m5)	f	21	29	p
3	audi	a4	2.0	2008	4	manual(m6)	f	20	31	p
4	audi	a4	2.0	2008	4	auto(av)	f	21	30	p
5	audi	a4	2.8	1999	6	auto(l5)	f	16	26	p
6	audi	a4	2.8	1999	6	manual(m5)	f	18	26	p
7	audi	a4	3.1	2008	6	auto(av)	f	18	27	p
8	audi	a4 quattro	1.8	1999	4	manual(m5)	4	18	26	p

*Useful when using with multiple monitors

Viewing data: Option 2

Type the name of the dataset in editor/console, and run the line (Ctrl + Enter).

Run

Prints data frame to console

Untitled1* | 0823_ggplot_cubicles.R

Source on Save | Run | Source

```
1  library(tidyverse)
2
3  capacity_ae <- read_csv("capacity_ae.csv")
4
5  capacity_ae
6
7
8
9
10
11
12
```

20:1 (Top Level) R Script

Console | Terminal | Jobs

C:/Users/andrew.jones/OneDrive - Midlands and Lancashire CSU/2019_projects/0815_r_workshop2.0/

```
> capacity_ae
# A tibble: 68 x 5
    site attendance2018 staff_increase dcubicles   dwait
   <dbl>          <dbl> <lgl>              <dbl>   <dbl>
 1     1          81600 TRUE                  50 -59.9
 2     2         114600 FALSE                 49 -32.8
 3     3          83400 FALSE                 47   3.39
```

Q1. How many sites?

Q2. Do we understand the variable names?

(and what they mean)

Q1. How many sites?

Q2. Do we understand the variable names?

difference in average ___
from 2017 to 2018

id

site	attendance2018	staff_increase	dcubicles	dwait

Did # staff increase
between 2017 and 2018

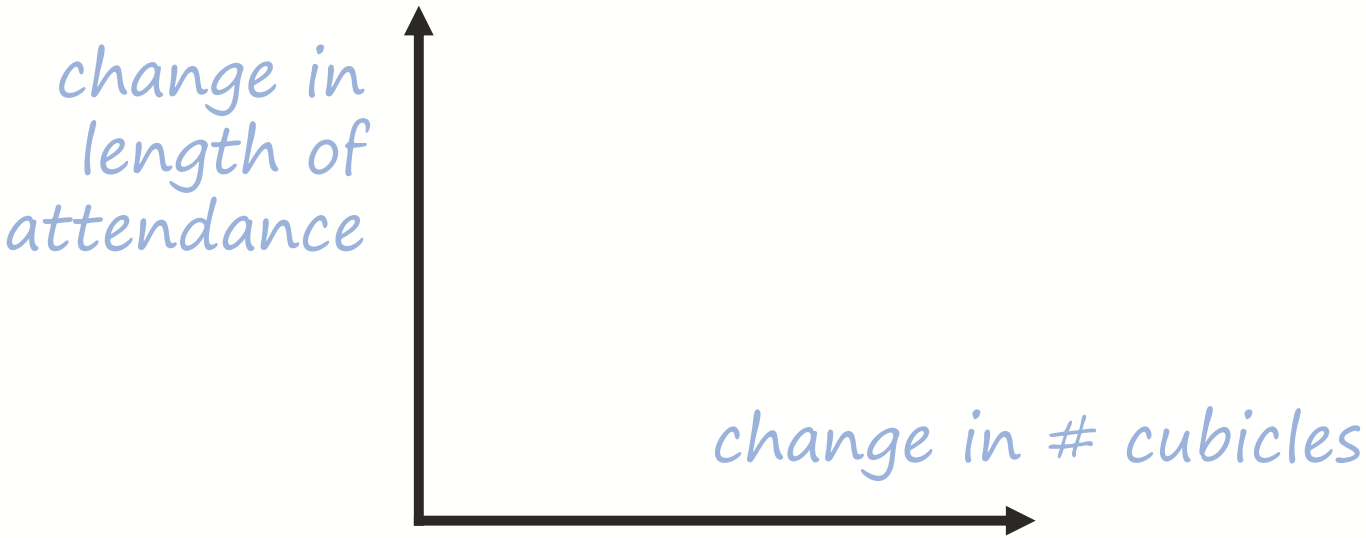
“The simple graph has bought more information to the data analyst’s mind than any other device”

– John Tukey

Graphics with ggplot2

Q. Is a change in the number of cubicles available in A&E associated with a change in length of attendance ?

Q. Is a change in the number of cubicles available in A&E associated with a change in length of attendance ?



Q. Is a change in the number of cubicles available in A&E associated with a change in length of attendance ?

Note that R is case sensitive

```
ggplot(data = capacity_ae) +
  geom_point(aes(x = dcubicles, y = dwait))
```

Breakdown

1. We begin our plot with `ggplot()`

2. Inside `ggplot()` we can specify our dataset

3. Next, we add layer(s) with +

```
ggplot(data = capacity_ae) +
geom_point(aes(x = dcubicles, y = dwait))
```

How do we move from data to graphic?

(Pen and paper exercise)

Create a graphic from the data below:

(Pen and paper exercise)

year	time (sec)
1930	12.0
1960	11.3
1990	10.5

Create a graphic from the data below.

Now, try to note down all the subtle (unconscious?) choices you made when creating your graphic.

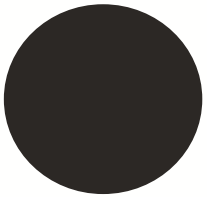
year	time (sec)
1930	12.0
1960	11.3
1990	10.5

Choices

1. What shape will represent the data?

Choices

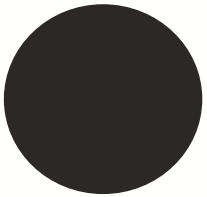
1. What shape will represent the data?



geometric object

Choices

1. What shape will represent the data? (geom)
2. What visual (aesthetic) attributes do we give to the geom?



Choices

1. What shape will represent the data? (geom)
2. What visual (aesthetic) attributes do we give to the geom?

Choices

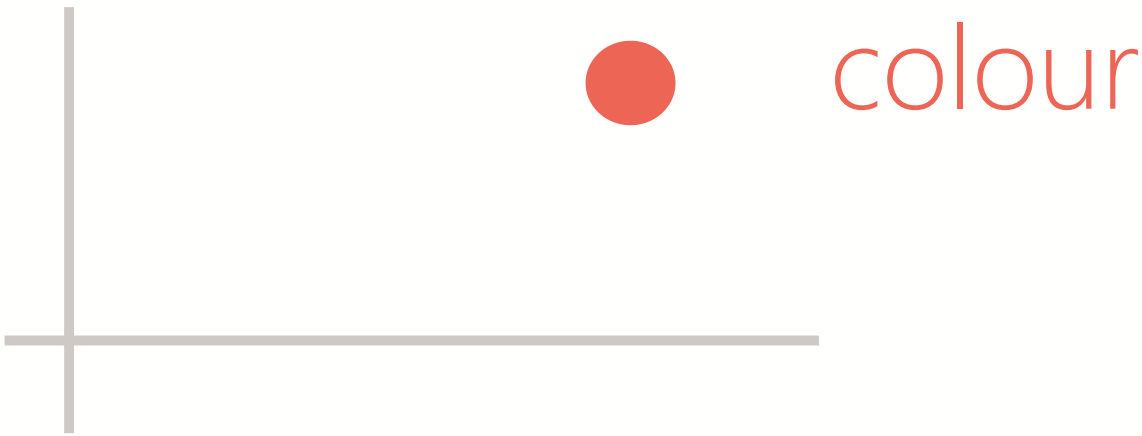
1. What shape will represent the data? (geom)
2. What visual (aesthetic) attributes do we give to the geom?

Choices

1. What shape will represent the data? (geom)
2. What visual (aesthetic) attributes do we give to the geom?

Choices

1. What shape will represent the data? (geom)
2. What visual (aesthetic) attributes do we give to the geom?



A statistical graphic

(aesthetic attributes of)

Data variables → geometric objects.

A statistical graphic

(aesthetic
attributes of)

Data variables → geometric objects.

```
ggplot(data = capacity_ae) +
geom_point(aes(x = dcubicles, y = dwait))
```

Here, other aesthetic
properties (size, colour, etc.)
are set by default

Functions ( )

`ggplot()`, **`geom_point()`**, and **`aes()`** are functions.

Running a function does something

Functions are given zero or more inputs (arguments)

Arguments of a function are separated by commas

Functions ( )

You can explicitly name arguments;

```
ggplot(data = capacity_ae) + ...
```

Or not:

```
ggplot(capacity_ae) + ...
```

(provided you have the arguments in the correct order!)

Functions ( )

Here, we have provided **ggplot()** *with one named argument*

```
ggplot(data = capacity_ae) +
  geom_point(aes(x = dcubicles, y = dwait))
```

And given **aes()** *two named arguments*

Unspecified (yet required) arguments will often revert to default values

Shorthand

Since ggplot2 knows the order of essential arguments, I will tend not to name arguments from now on. So:

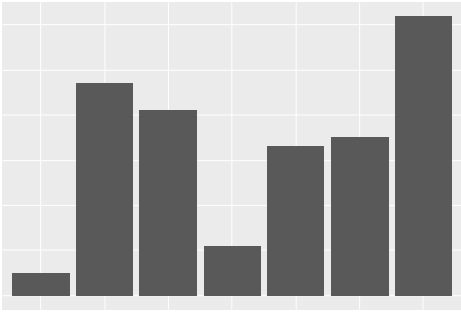
no need for "data ="

```
ggplot(capacity_ae) +
  geom_point(aes(dcubicles, dwait))
```

x goes first y goes second

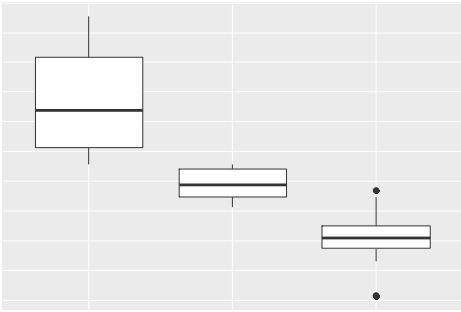
geoms

We tend to describe plots in terms of the *geom* used:

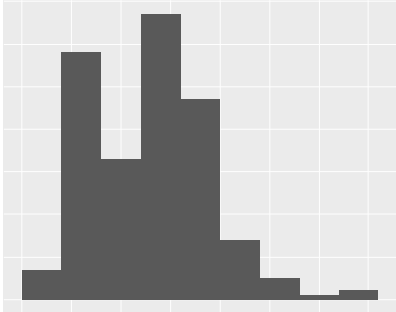


`geom_bar()`

`geom_line()`



`geom_boxplot()`



`geom_histogram()`

Layering geoms

We can display more than one geom in a plot:

to add a layer

```
ggplot(capacity_ae) +
  geom_point(aes(dcubicles, dwait)) +
  geom_line(aes(dcubicles, dwait))
```

then specify another geom

Your turn

This is our current plot:

```
ggplot(data = capacity_ae) +
 geom_point(aes(x = dcubicles, y = dwait))+
 ... ... ...(...(... ...  ))
```

Add a **`geom_smooth`** layer (to help identify patterns)

Hint: Don't forget the aes() values in the new layer

Extension: If you prefer, re-write in shorthand

Your turn

A geom_smooth layer can help us identify patterns. Add **`geom_smooth()`** to our current plot:

```
ggplot(data = capacity_ae) +
geom_point(aes(x = dcubicles, y = dwait)) +
geom_smooth(aes(x = dcubicles, y = dwait))
```

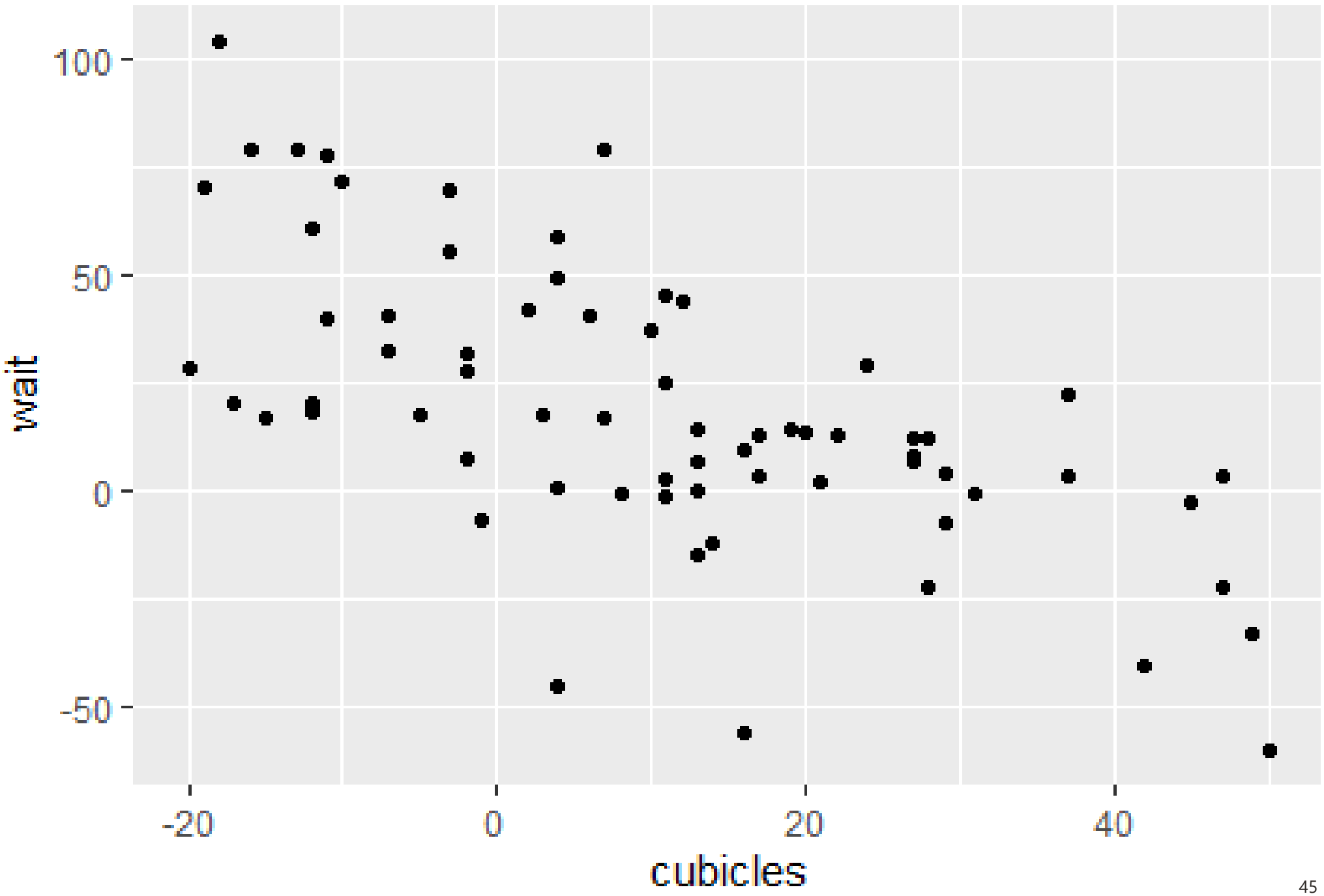
One more thing

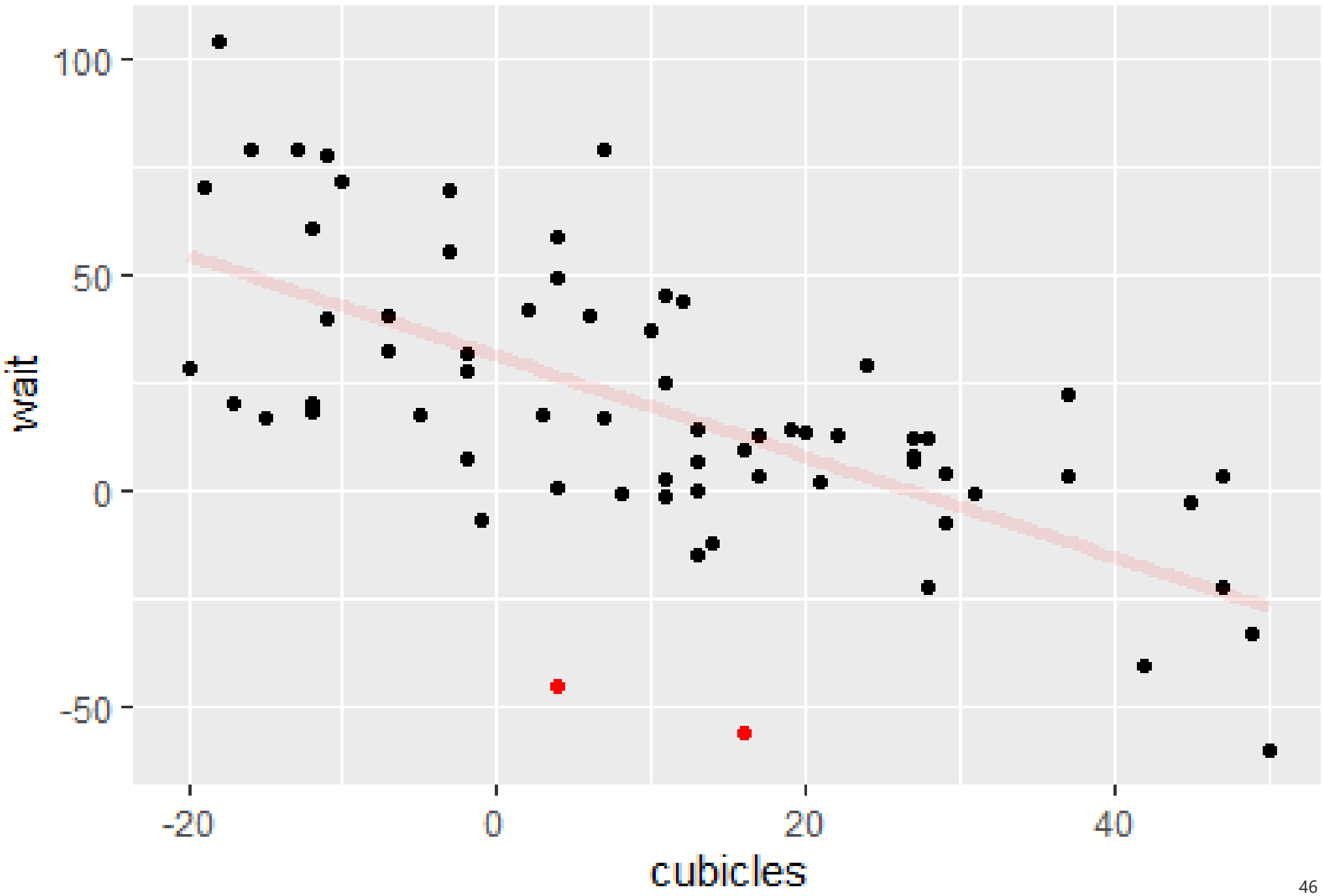
We'd probably prefer a linear fit

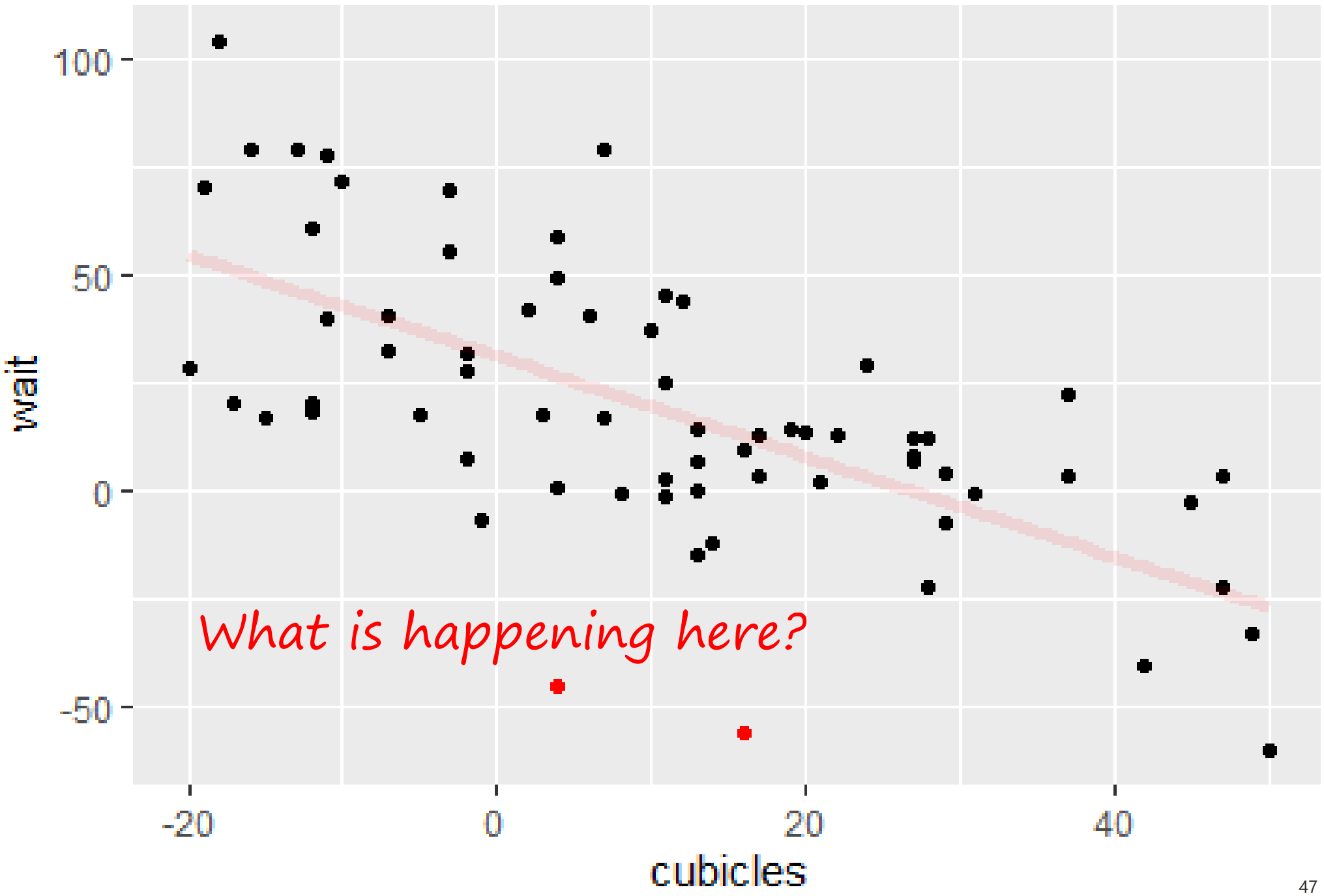
rather than a non-linear fit:

```
ggplot(capacity_ae)+
  geom_point(aes(dcubicles, dwait)) +
  geom_smooth(aes(dcubicles, dwait),
                method = "lm")
```

Fit a linear model


100
50
0
-50
wait
-20
0
20
40
cubicles


100
50
0
-50
wait
-20
0
20
40
cubicles


100
50
0
-50
wait
-20
0
20
40
cubicles
What is happening here?

Ideas?

Hypothesis

The two sites have seen staffing increases

aesthetic attribute

We can map point colour to the **staff_increase** variable to find out.

So each point is given a colour which will depend on the value of **staff_increase**.

Adding another dimension

Put an argument <u>inside</u> **`aes()`** if you want a visual attribute to change with different values of a variable.

```
ggplot(capacity_ae) +
 geom_point(
 aes(dcubicles, dwait, colour = staff_increase))
```

We could have equally have chosen `size` *or* `shape` *– but these make graphic less clear*

Outcome

The two sites in question have indeed seen an increase in staff levels.

Important distinction

If you want a visual attribute to be applied across the whole plot, the argument goes <u>outside</u> **`aes()`**

```
ggplot(capacity_ae) +
    geom_point(
                aes(dcubicles, dwait),
                colour = “red”
                )
```

Important distinction

If you want a visual attribute to be applied across the whole plot, the argument goes <u>outside</u> **`aes()`**

```
ggplot(capacity_ae) +
    geom_point(
                aes(dcubicles, dwait),
                size = 4
                )
```

Small multiples

Another way to visualise the relationship between multiple variables is with a **`facet_wrap()`** layer:

```
ggplot(capacity_ae) +
  geom_point(aes(dcubicles, dwait)) +
  facet_wrap(vars(staff_increase))
```

1. Set up our basic 2-D graphic

2. Then, tell **facet_wrap** you want a panel for each category of "staff increase"

Small multiples

Another way to visualise multiple variables simultaneously is with a `facet_wrap()` layer:

```
ggplot(capacity_ae) +
  geom_point(aes(dcubicles, dwait)) +
  facet_wrap(vars(staff_increase),
             ncol = 1)
```

facet_wrap is used with categorical variables

Demonstrating geoms: (note: these are simple, unpolished graphics)

Q. How are “wait” values distributed? Histogram

```
ggplot(capacity_ae) +
  geom_histogram(aes(dwait))
```

Q. How are “wait” values distributed? Histogram

```
ggplot(capacity_ae) +
  geom_histogram(aes(dwait),
                 binwidth = 10)
```

Q. Number of attendances by site? Bar plot

```
ggplot(capacity_ae) +
  geom_col(aes(x = site,
               y = attendance2018))
```

Q. Number of attendances by site? Bar plot

```
ggplot(capacity_ae)+
  geom_col(
    aes(x = reorder(site, attendance2018),
        y = attendance2018))
```

Reorder **site** *by* **attendances**

Q. Distribution of “wait” for each value of staff level? Box plot

```
ggplot(capacity_ae) +
  geom_boxplot(
      aes(staff_increase, dwait))
```

Plot labels

```
ggplot(capacity_ae) +
  geom_boxplot(aes(staff_increase, dwait))+
  labs(
      title = “Do changes in staffing …”,
      y = “Waiting time”
       )
```

Add layer to your ggplot

or: pdf, jpg as you wish

```
... +
ggsave("plot_name.png")
```

By default: saves a plot in the same dimensions as plot window.

In future, you'll wish to add height, width and "units" arguments to specify plot dimensions:

```
ggsave(plot_name.png, units = "cm",
height = 10, width = 8)
```

What does this function do?

C:/Users/andrew.jones/OneDrive - Midlands and Lancashire CSU/2019_projects/0815_r_workshop2.0 - RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function Addins

Untitled1* title

Source on Save Run Source

```
60
61
62 ggplot(capacity_ae, aes(staff_increase, dwait)) +
63   geom_boxplot()+
64   labs(
65     title = "Waiting times by staff level",
66     y = "Waiting time"
67     )
68
69
70 ggsav
71 ◇ ggsave {ggplot2}
72
73
74
75
76
77
78
```

```
ggsave(filename, plot = last_plot(), device = NULL, path =
    NULL, scale = 1, width = NA, height = NA, units =
    c("in", "cm", "mm"), dpi = 300, limitsize = TRUE, ...)
```

`ggsave()` is a convenient function for saving a plot. It defaults to saving the last plot that you displayed, using the size of the current graphics device. It also guesses the type of graphics device from the extension.

Press F1 for additional help

70:6 (Top Level) R Script

What does this function do?

C:/Users/andrew.jones/OneDrive - Midlands and Lancashire CSU/2019_projects/0815_r_workshop2.0 - RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function | Addins | 0815_r_workshop2.0

Untitled1* | title

Source on Save | Run | Source

```
56
57
58
59
60
61
62 ggplot(capacity_ae, aes(staff_increase, dwait)) +
63   geom_boxplot()+
64   labs(
65     title = "Waiting times by staff level",
66     y = "Waiting time"
67     )
68
69
70 ggsave()
71
72
73
74
75
76
77
78
```

70:5 (Top Level) R Script

Have your cursor in the function name and press F1

Environment | History | Connections

Files | Plots | Packages | Help | Viewer

R: Save a ggplot (or other grid object) with sensible defaults | Find in Topic

`...` Other arguments passed on to the graphics device function, as specified by `device`.

Details

Note: Filenames with page numbers can be generated by including a C integer format expression, such as `%03d` (as in the default file name for most R graphics devices, see e.g. `png()`). Thus, `filename = "figure%03d.png"` will produce successive filenames `figure001.png`, `figure002.png`, `figure003.png`, etc. To write a filename containing the `%` sign, use `%%`. For example, `filename = "figure-100%%.png"` will produce the filename `figure-100%.png`.

Examples

```
## Not run:
ggplot(mtcars, aes(mpg, wt)) + geom_point()

ggsave("mtcars.pdf")
ggsave("mtcars.png")
```

What does this function do?

C:/Users/andrew.jones/OneDrive - Midlands and Lancashire CSU/2019_projects/0815_r_workshop2.0 - RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function | Addins | 0815_r_workshop2.0

Untitled1* | title

Source on Save | Run | Source

```
ggplot(capacity_ae, aes(staff_increase, dwait)) +
  geom_boxplot()+
  labs(
    title = "Waiting times by staff level",
    y = "Waiting time"
    )


ggsave()
```

70:5 (Top Level) R Script

Have your cursor in the function name and press F1

Environment | History | Connections

Files | Plots | Packages | Help | Viewer

R: Save a ggplot (or other grid object) with sensible defaults | Find in Topic

`...` Other arguments passed on to the graphics device function, as specified by `device`.

Details

Note: Filenames with page numbers can be generated by including a C integer format expression, such as `%03d` (as in the default file name for most R graphics devices, see e.g. `png()`). Thus, `filename = "figure%03d.png"` will produce successive filenames `figure001.png`, `figure002.png`, `figure003.png`, etc. To write a filename containing the `%` sign, use `%%`. For example, `filename = "figure-100%%.png"` will produce the filename `figure-100%.png`.

Examples

```
## Not run:
ggplot(mtcars, aes(mpg, wt)) + geom_point()

ggsave("mtcars.pdf")
ggsave("mtcars.png")
```

Note: typing ? before the function and executing will do the same

Save your script!

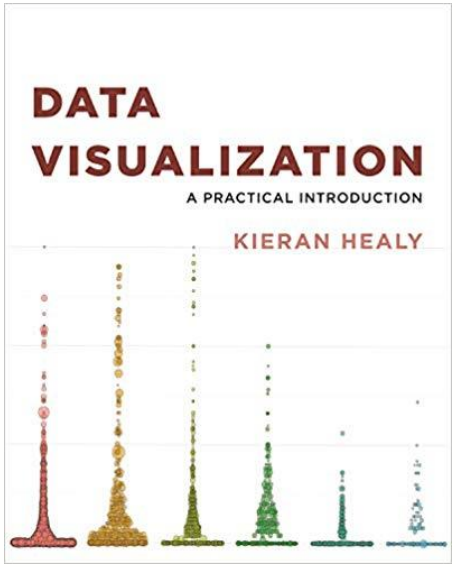
Think of your script[1] as the "real" part of your analysis.

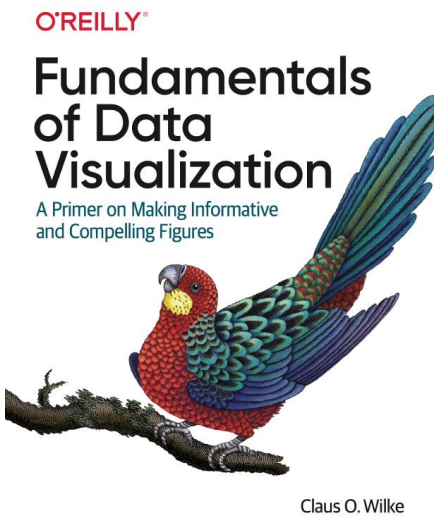
File → Save As... → ggplot_intro.R

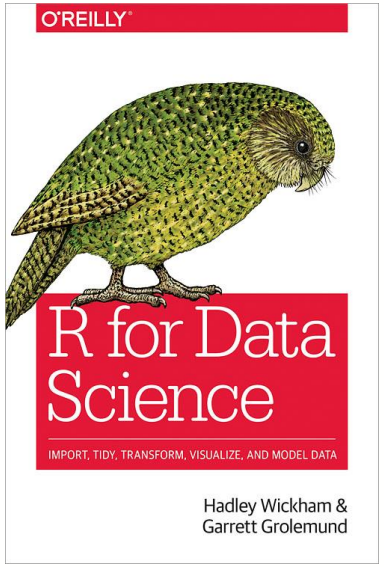
Or, shortcut:
Ctrl + s

1. Rather than the objects you create

Recommended Reading







http://socviz.co/

https://r4ds.had.co.nz/

https://serialmentor.com/dataviz/

All freely available online (hardcopies also recommended)

accidental aRt

https://twitter.com/accidental_aRt

This work is licensed as

End

Your turn

A geom_smooth layer can help us identify patterns.

Add **`geom_smooth()`** to our current plot:

```
ggplot(capacity_ae) +
 geom_point(aes(dcubicles, dwait)) +
 geom_smooth(aes(dcubicles, dwait))
```

More succinct but...

Layering geoms

A geom_smooth layer can help us identify patterns.

Add **`geom_smooth()`** to our current plot:

duplication!

```
ggplot(capacity_ae) +
  geom_point(aes(dcubicles, dwait)) +
  geom_smooth(aes(dcubicles, dwait))
```

Layering geoms

To avoid duplication, we can pass the common local **`aes()`** arguments to **`ggplot()`** to make them global

```
ggplot(capacity_ae, aes(dcubicles, dwait))+
  geom_point() +
  geom_smooth()
```

In the **`ggplot()`** *function,* **`aes()`** *is the standard 2nd argument*