COMP824 2023 Week 4 Data Visualisation

Dr Sarah Marshall
sarah.marshall@aut.ac.nz WZ Level 9

Department of Mathematical Sciences
Auckland University of Technology

Overview

The Process of Analytics

The Process of Analytics - with annotations

Tidyverse

Visualise

Reading

Chapter 2 and 3, Wickham and Grolemund (2020), R for Data Science https://r4ds.had.co.nz/

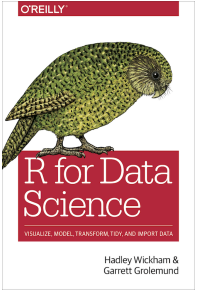
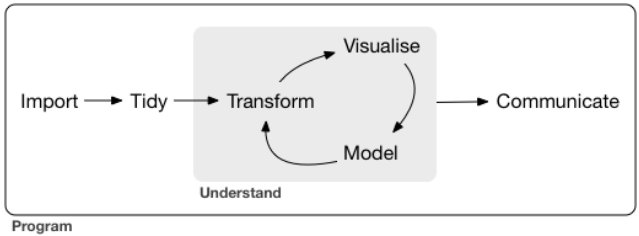


Figure 1: http://r4ds.had.co.nz/

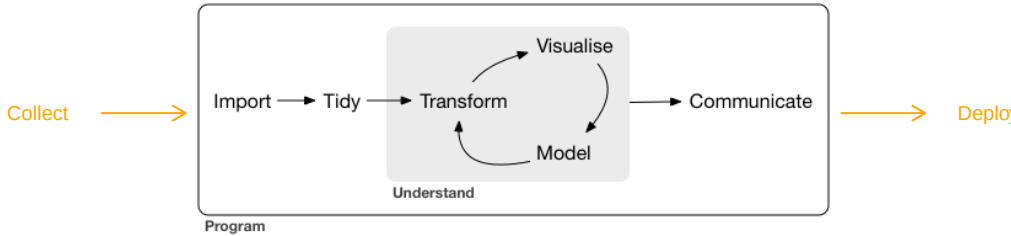
Learning objectives

- Know what Tidyverse is and how to install it within RStudio
- Know how to install and load packages in R
- Create plots using `ggplot2`
- Customise plots using `ggplot2`
- Appreciate and apply principles of data visualisation (independent learning - video)

The Process of Analytics



The Process of Analytics - with annotations



Tidyverse

Definition: “The tidyverse is an opinionated collection of R packages designed for data science. All packages share an underlying design philosophy, grammar, and data structures.” https://www.tidyverse.org/

Figure 4: https://www.tidyverse.org/

Install Tidyverse

```
install.packages("tidyverse") # do this once per computer
library(tidyverse)  # do this each time you start RStudio
```

Visualise

The first step of any data analysis is to **explore** the data. Visualisation is a good way to do this.

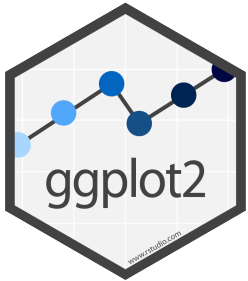


Figure 5: https://www.tidyverse.org/

ggplot2 implements the “Grammar of graphics”

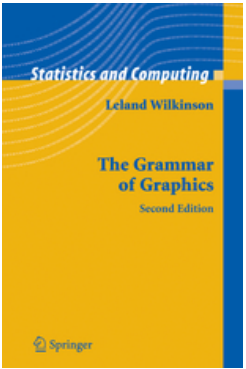


Figure 6: Grammar of Graphics Book

This article gives a nice overview of the philosophy behind `ggplot2`.
https://www.r-bloggers.com/a-simple-introduction-to-the-graphing-philosophy-of-ggplot2/

Install ggplot2

```
install.packages("ggplot2") # do this once per computer
library(ggplot2)  # do this each time you start RStudio
```

Note: if you have installed and load `tidyverse` then you don't need to install and load `ggplot2` as it is part of `tidyverse`.

Example: Cars

Discussion: Do cars with big engines use more fuel than cars with small engines?

Source: https://www.flickr.com/photos/92622665```N08/16013517632

Example: Cars - In-built dataset

Dataset: `mpg`

tibbles = special dataframes

Discussion: What do you think `int`,`dbl`, and `chr` represent on the following slide?

Example: Inspecting the mpg data

```
# A tibble: 234 x 11
  manufacturer model displ  year   cyl trans drv
  <chr>        <chr> <dbl> <int> <int> <chr> <chr>
1 audi         a4      1.8  1999     4 auto~ f
2 audi         a4      1.8  1999     4 manu~ f
3 audi         a4      2    2008     4 manu~ f
4 audi         a4      2    2008     4 auto~ f
# ... with 230 more rows, and 4 more variables:
#   cty <int>, hwy <int>, fl <chr>, class <chr>
```

- `displ` engine size in litres
- `hwy` efficiency on a highway in miles per gallon (mpg)

Example: Cars - variables

```
?mpg
```

Fuel economy data from 1999 to 2008 for 38 popular models of cars

Description

This dataset contains a subset of the fuel economy data that the EPA makes available on https://fueleconomy.gov/. It contains only models which had a new release every year between 1999 and 2008 - this was used as a proxy for the popularity of the car.

Usage

```
mpg
```

Format

A data frame with 234 rows and 11 variables:

manufacturer
: manufacturer name

model
: model name

displ
: engine displacement, in litres

year
: year of manufacture

cyl
: number of cylinders

trans
: type of transmission

drv
: the type of drive train, where f = front-wheel drive, r = rear whee

cty
: city miles per gallon

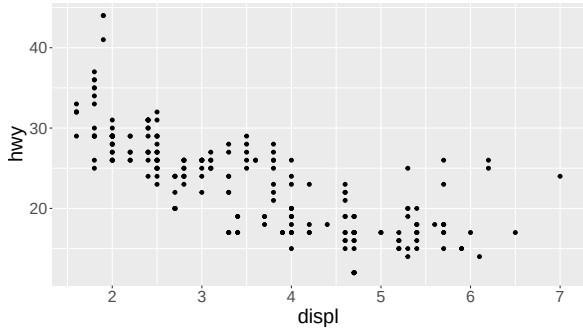
hwy
: highway miles per gallon

fl
: fuel type

class
: "type" of car

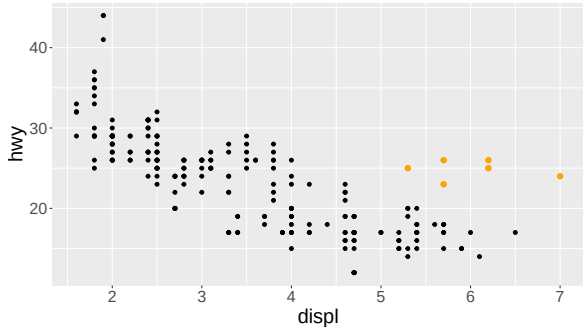
Create a scatterplot using `ggplot2`

```
# create empty plot and specify dataset
ggplot(data = mpg) +
# add layer to the plot
   geom_point(mapping = aes(x = displ, y = hwy))
```



Describe the relationship in this graph.

Examine a scatterplot using `ggplot2`



Discussion: Notice the group of points which have a slightly higher than expected miles per gallon than expected. Discuss some reasons why this might be the case.

Aesthetic Mappings

`mapping = aes(x = ..., y = ...)}` specifies how the data is mapped to the x and y axes.

Basic structure of ggplot

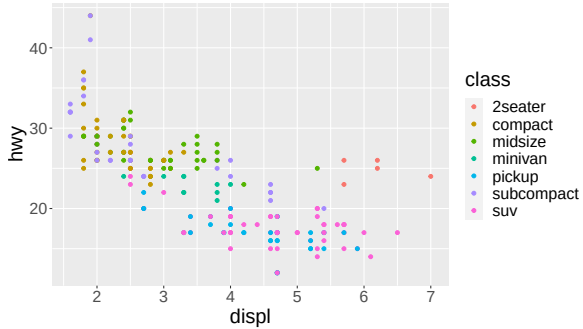
```
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

e.g.

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy))
```

Adding colour

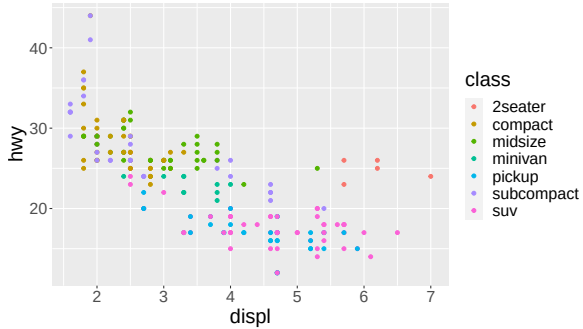
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, color = class))
```



- What does this graph tell you about the points that we identified?

Adding colour

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, color = class))
```



- What does this graph tell you about the points that we identified?
- They are 2 seaters, i.e. sports cars - large engines but small bodies, so have a good mpg.

Transparency and size

```
# Left
(a <- ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, alpha = class)))
```

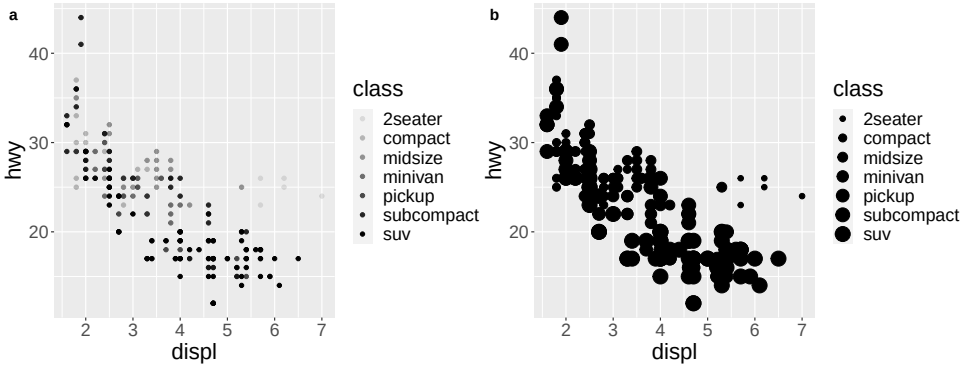
```
Warning: Using alpha for a discrete variable is
not advised.
```

```
# Right
(b <- ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, size = class)))
```

```
Warning: Using size for a discrete variable is not
advised.
```

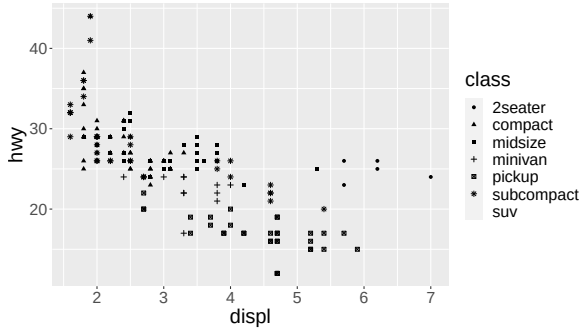
Notice the warning message associated with these options.

Transparency and size



Shape

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, shape = class))
```

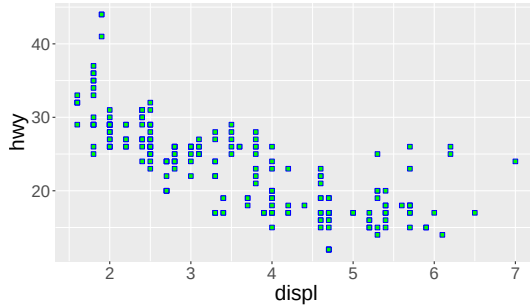


Notice the warning message associated with this option.

Manually modify aesthetics

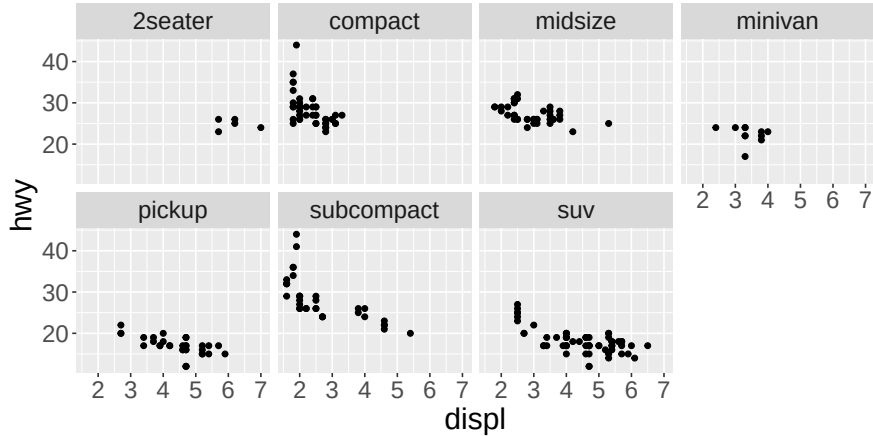
- Add aesthetic options (color, shape, size etc) *outside* `aes()`.
- For aesthetics not related to values in dataset
- See `?geom_point` for more details.

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy),
             color = "blue", shape=22, size=2, fill="green")
```



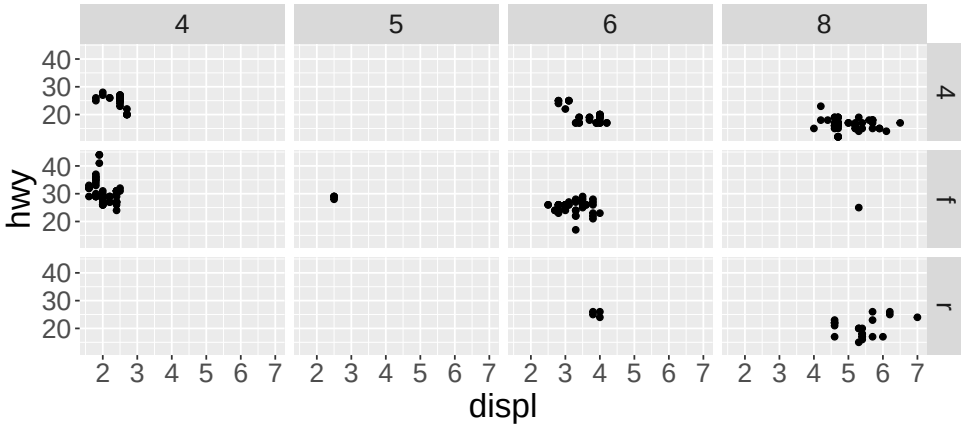
Facets: `facet_wrap()` - subplots by single discrete variable

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  facet_wrap(~ class, nrow = 2)
```



Facets: `facet_grid()` - subplots by two discrete variables

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  facet_grid(drv ~ cyl)
```



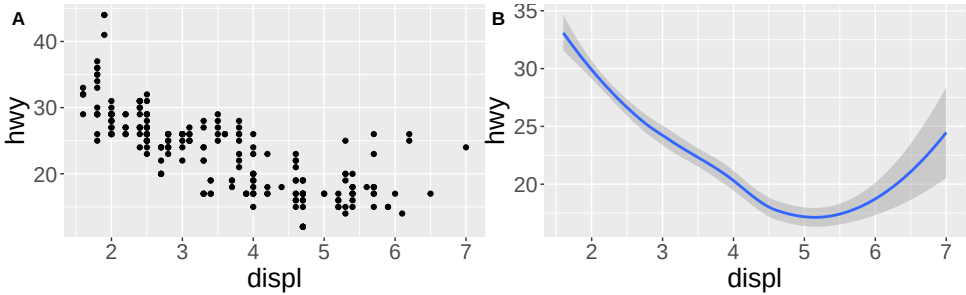
Geometric Objects: geoms

`geoms` specify the type of plot (e.g. scatterplot, boxplot, histogram etc).

- `geom_point`
- `geom_smooth`
- `geom_boxplot`
- `geom_histogram`
- `geom_line`
- and many more!!

Geometries - example `geom_point` and `geom_smooth`

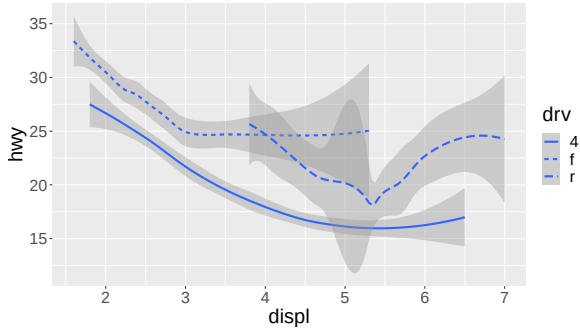
```
A <- ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy))
B <- ggplot(data = mpg) +
  geom_smooth(mapping = aes(x = displ, y = hwy))
cowplot::plot_grid(A, B, labels = "AUTO")
```



The `cowplot` package is useful for position multiple plots in documents.

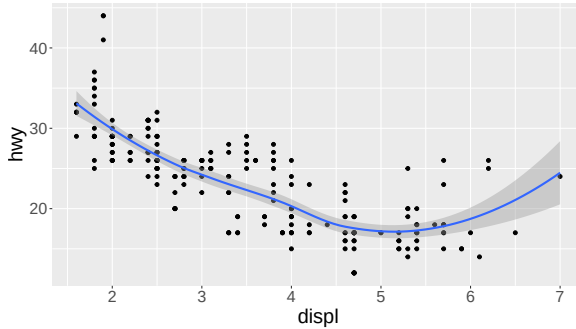
Geometries – geom_smooth with a 3rd variable

```
ggplot(data = mpg) +
  geom_smooth(mapping = aes(x = displ, y = hwy, linetype = drv))
```



Two geometries on the same plot

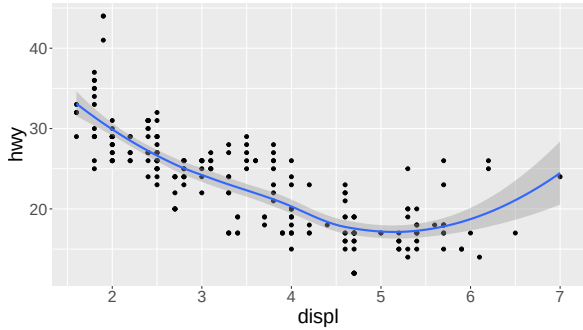
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  geom_smooth(mapping = aes(x = displ, y = hwy))
```



Two geometries on the same plot - better code

Add “global” mapping to ggplot() to avoid duplication.

```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_point() +
  geom_smooth()
```



Statistical Transformations and Bar Charts

Some functions will perform statistical transformations in order to create a plot.

Example: `geom_bar` *counts* the number in each category.

Example: Diamonds (dataset within `ggplot2`)

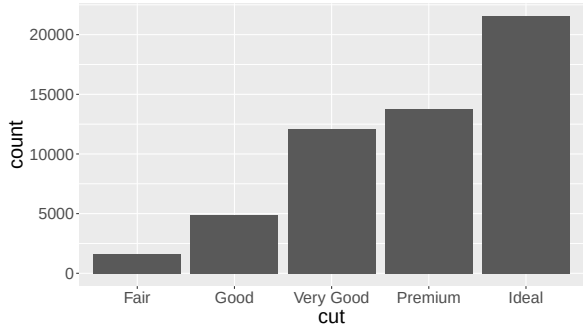
```
diamonds
```

```
# A tibble: 53,940 x 10
  carat cut       color clarity depth table price
  <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int>
1  0.23 Ideal     E     SI2      61.5    55   326
2  0.21 Premium   E     SI1      59.8    61   326
3  0.23 Good      E     VS1      56.9    65   327
4  0.29 Premium   I     VS2      62.4    58   334
# ... with 53,936 more rows, and 3 more
#   variables: x <dbl>, y <dbl>, z <dbl>
```

Bar Chart

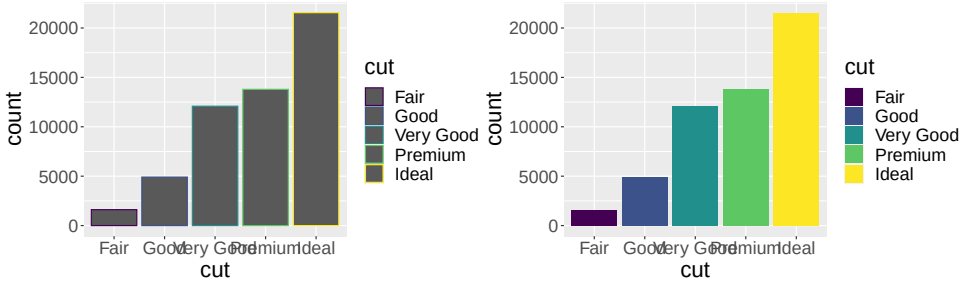
- By default, `geom_bar` shows the *count*. See `?geom_bar` for other options.

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



Adding colour to a bar chart: fill vs colour

```
a <- ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, colour=cut))
b <- ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, fill=cut))
cowplot::plot_grid(a, b)
```



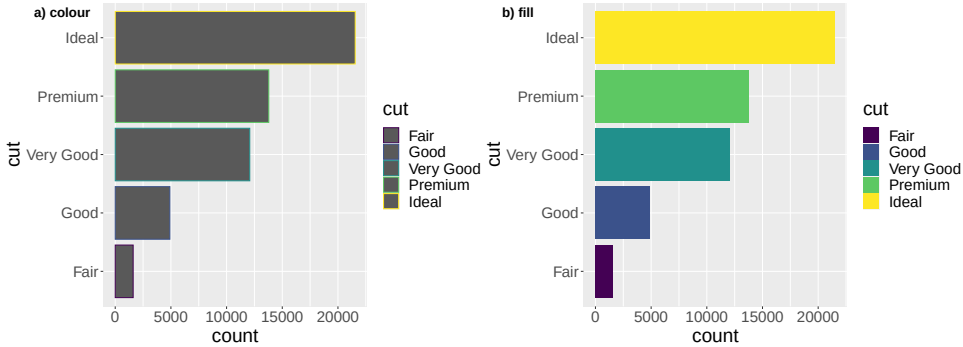
Flipping a bar chart

```
a <- ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, colour=cut)) +
  coord_flip()

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

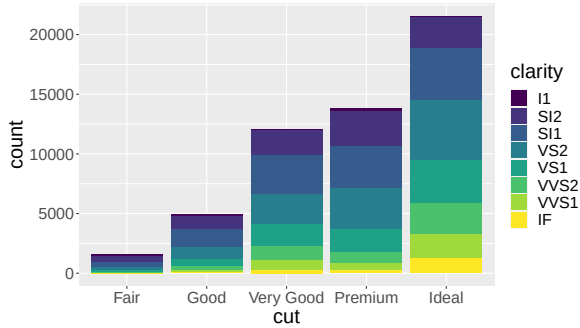
Flipping a bar chart

```
cowplot::plot_grid(a, b, labels = c("a) colour", "b) fill"))
```



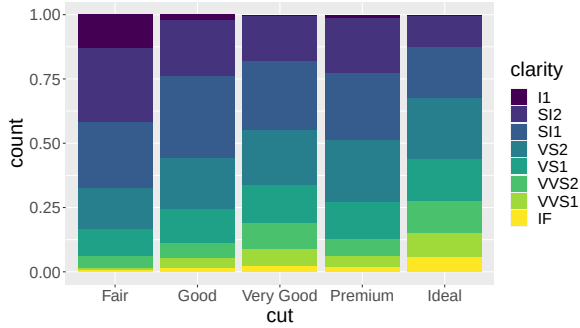
Stacked bar charts

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



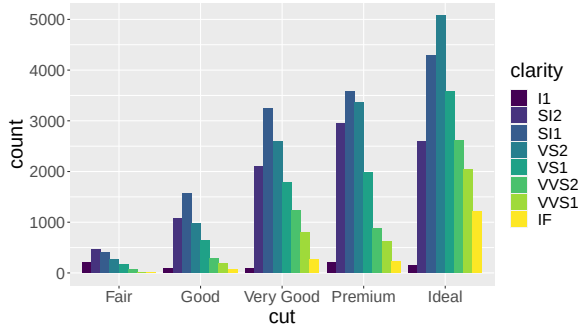
Stacked bar charts - Proportions

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



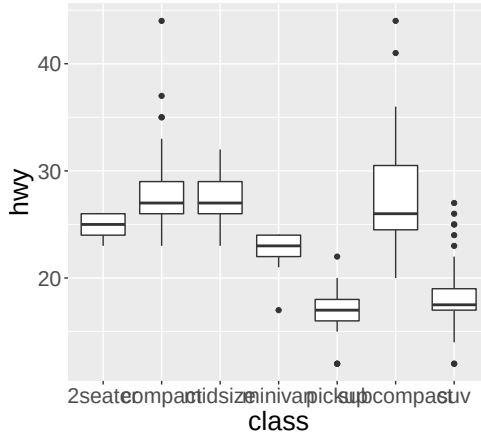
Side-by-side bar charts

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



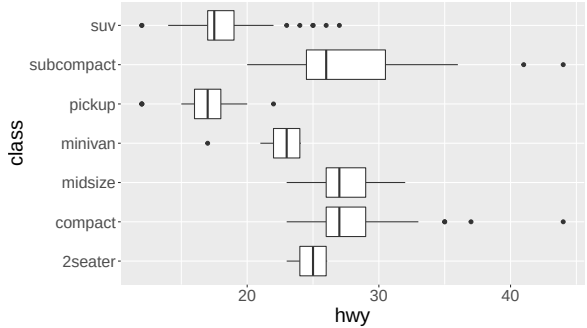
Boxplots

```
ggplot(data = mpg, mapping = aes(x = class, y = hwy)) +
  geom_boxplot()
```



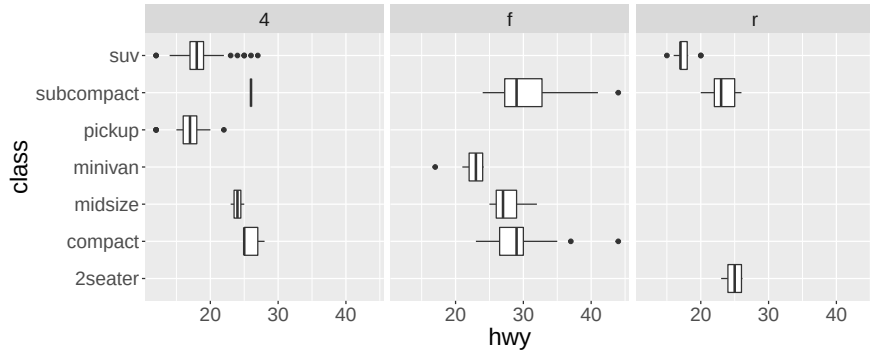
Boxplots: Flip x and y axis

```
ggplot(data = mpg, mapping = aes(x = class, y = hwy)) +
  geom_boxplot() +
  coord_flip()
```



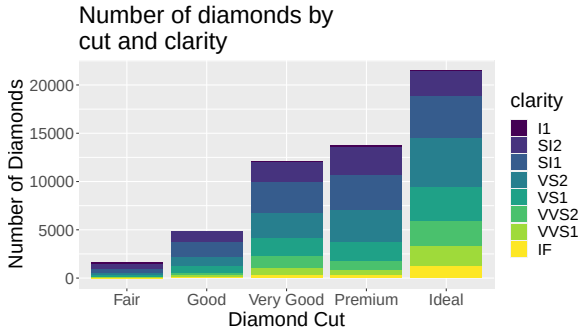
Boxplots: with facets

```
ggplot(data = mpg, mapping = aes(x = class, y = hwy)) +
  geom_boxplot() +
  coord_flip() +
  facet_wrap(~ drv)
```



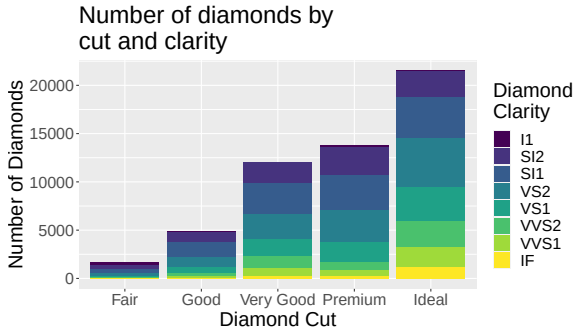
Customising Labels (method 1)

```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, fill=clarity)) +
  xlab("Diamond Cut") + ylab("Number of Diamonds") +
  ggtitle("Number of diamonds by \ncut and clarity")
```



Customising Labels (method 2)

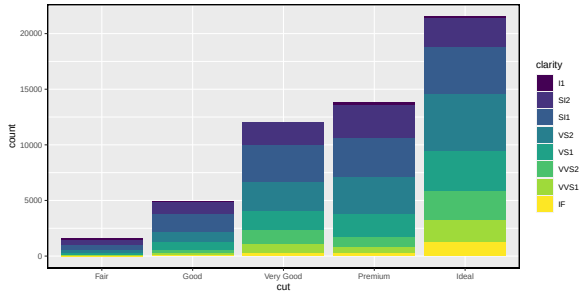
```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, fill=clarity)) +
  labs(x = "Diamond Cut",
       y = "Number of Diamonds",
       title = "Number of diamonds by \ncut and clarity",
       fill = "Diamond \nClarity")
```



Themes

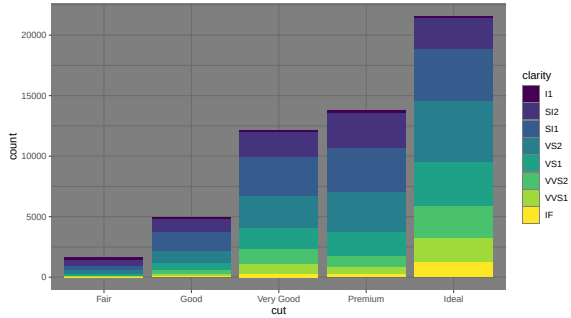
- Themes can be used to customise the look of plots, e.g. to change the size of all elements on a plot
- See `?theme` for more info

```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, fill=clarity)) +
  theme(text = element_text(size = 10),
  panel.border = element_rect(colour = "black", fill=NA, size=1))
```



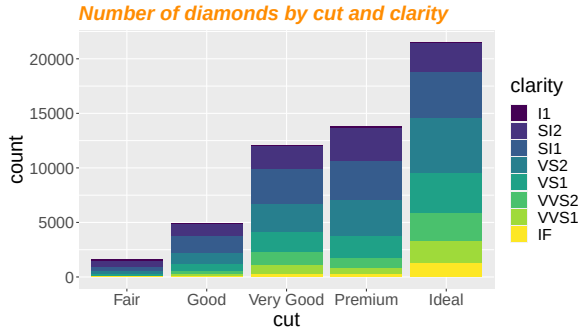
Preset Themes - e.g. Dark theme

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



Using `theme` to change customise labels

```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, fill=clarity)) +
  labs(title = "Number of diamonds by cut and clarity")+
  theme(plot.title = element_text(color="darkorange", size=20,
                                  face="bold.italic"))
```



Key elements of ggplot2

```
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(
     mapping = aes(<MAPPINGS>),
     stat = <STAT>,
     position = <POSITION>
  ) +
  <COORDINATE_FUNCTION> +
  <FACET_FUNCTION> +
  <THEME>
```

Getting help

Best way to get help: Ask your favourite search engine

e.g. Suppose you want to change the background colour of the plot

Suggested search term: *ggplot2 change background color*

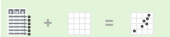
Cheatsheet: ggplot2

https://www.rstudio.com/resources/cheatsheets

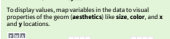
Data visualization with ggplot2 : : CHEAT SHEET

Basics

ggplot2 is based on the **grammar of graphics**, the idea that you can build every graph from the same components: a **data** set, a **coordinate system**, and **geoms**—visual marks that represent data points.



To display values, map variables in the data to visual properties of the geom (**aesthetics**) like **size**, **color**, and **x** and **y** locations.



Complete the template below to build a graph.

required

```
ggplot(data = <DATA>) +
<GEOM_FUNCTION>(mapping = aes(<MAPPINGS>),
stat = <STAT>, position = <POSITION>) +
<COORDINATE_FUNCTION> +
<FACET_FUNCTION> +
<SCALE_FUNCTION> +
<THEME_FUNCTION>
```

Not required, sensible defaults supplied

ggplot(data = mpg, **aes**(x = cty, y = hwy)) Begins a plot that you finish by adding layers to. Add one geom function per layer.

last_plot() Returns the last plot.

ggsave("plot.png", width = 5, height = 5) Saves last plot as 5' x 5' file named "plot.png" in working directory. Matches file type to file extension.

Aes Common aesthetic values.

color and **fill** - string ("red", "#RRGGBB")

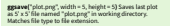
linetype - integer or string (0 = "blank", 1 = "solid", 2 = "dashed", 3 = "dotted", 4 = "dotdash", 5 = "longdash", 6 = "twodash")

linewind - string ("round", "butt", or "square")

linejoin - string ("round", "mitre", or "bevel")

size - integer (line width in mm)

shape - integer/shape name or a single character ("a")



Geoms

Use a geom function to represent data points, use the geom's aesthetic properties to represent variables. Each function returns a layer.

GRAPHICAL PRIMITIVES

```
a <- ggplot(economics, aes(date, unemploy))
b <- ggplot(seals, aes(x = long, y = lat))
```

a + geom_blank() and **a + expand_limits()** Ensure limits include values across all plots.

b + geom_curve(aes(yend = lat + 1, xend = long + 1), curvature = 1) - x, xend, y, yend, alpha, angle, color, curvature, linetype, size

a + geom_path(lineend = "butt", linejoin = "round", linemitre = 1) x, y, alpha, color, group, linetype, size

a + geom_polygon(aes(alpha = 50)) - x, y, alpha, color, fill, group, subgroup, linetype, size

b + geom_rect(aes(xmin = long, ymin = lat, xmax = long + 1, ymax = lat + 1)) - xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size

a + geom_ribbon(aes(ymin = unemploy - 900, ymax = unemploy + 900)) - x, ymax, ymin, alpha, color, fill, group, linetype, size

LINE SEGMENTS

common aesthetics: x, y, alpha, color, linetype, size

b + geom_abline(aes(intercept = 0, slope = 1)) **b + geom_hline**(aes(yintercept = lat)) **b + geom_vline**(aes(xintercept = long))

b + geom_segment(aes(yend = lat + 1, xend = long + 1)) **b + geom_spoke**(aes(angle = 1:1155, radius = 1))

ONE VARIABLE continuous

```
c <- ggplot(mpg, aes(hwy)); c2 <- ggplot(mpg)
```

c + geom_area(stat = "bin") x, y, alpha, color, fill, linetype, size

c + geom_density(kernel = "gaussian") x, y, alpha, color, fill, group, linetype, size, weight

c + geom_dotplot() x, y, alpha, color, fill

c + geom_freqpoly() x, y, alpha, color, group, linetype, size

c + geom_histogram(binwidth = 5) x, y, alpha, color, fill, linetype, size, weight

c2 + geom_qq(aes(sample = hwy)) x, y, alpha, color, fill, linetype, size, weight

discrete

```
d <- ggplot(mpg, aes(fl))
```

d + geom_bar() x, alpha, color, fill, linetype, size, weight

TWO VARIABLES both continuous

```
e <- ggplot(mpg, aes(cty, hwy))
```

e + geom_label(aes(label = cty), nudge_x = 1, nudge_y = 1) - x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

e + geom_point() x, y, alpha, color, fill, shape, size, stroke

e + geom_quantile() x, y, alpha, color, group, linetype, size, weight

e + geom_rug(sides = "bl") x, y, alpha, color, linetype, size

e + geom_smooth(method = lm) x, y, alpha, color, fill, group, linetype, size, weight

e + geom_text(aes(label = cty), nudge_x = 1, nudge_y = 1) - x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

one discrete, one continuous

```
f <- ggplot(mpg, aes(class, hwy))
```

f + geom_col() x, y, alpha, color, fill, group, linetype, size

f + geom_boxplot() x, y, lower, middle, upper, ymax, ymin, alpha, color, fill, group, linetype, shape, size, weight

f + geom_dotplot(binaxis = "y", stackdir = "center") x, y, alpha, color, fill, group

f + geom_violin(scale = "area") x, y, alpha, color, fill, group, linetype, size, weight

both discrete

```
g <- ggplot(diamonds, aes(cut, color))
```

g + geom_count() x, y, alpha, color, fill, shape, size, stroke

g + geom_jitter(height = 2, width = 2) x, y, alpha, color, fill, shape, size

continuous bivariate distribution

```
h <- ggplot(diamonds, aes(carat, price))
```

h + geom_bin2d(binwidth = c(0.25, 500)) x, y, alpha, color, fill, linetype, size, weight

h + geom_density_2d() x, y, alpha, color, group, linetype, size

h + geom_hex() x, y, alpha, color, fill, size

continuous function

```
i <- ggplot(economics, aes(date, unemploy))
```

i + geom_area() x, y, alpha, color, fill, linetype, size

i + geom_line() x, y, alpha, color, group, linetype, size

i + geom_step(direction = "hv") x, y, alpha, color, group, linetype, size

visualizing error

```
df <- data.frame(grp = c("A", "B"), fit = 4:5, se = 1:2)
j <- ggplot(df, aes(grp, fit, ymin = fit - se, ymax = fit + se))
```

j + geom_crossbar(fatten = 2) - x, y, ymax, ymin, alpha, color, fill, group, linetype, size

j + geom_errorbar() - x, ymax, ymin, alpha, color, group, linetype, size, width Also **geom_errorbarh()**.

j + geom_linerange() x, ymin, ymax, alpha, color, group, linetype, size

j + geom_pointrange() - x, y, ymin, ymax, alpha, color, fill, group, linetype, shape, size

maps

```
data <- data.frame(murder = USArrests$Murder,
                   state = tolower(rownames(USArrests)))
map <- map_data("state")
k <- ggplot(data, aes(fill = murder))
```

k + geom_map(aes(map_id = state), map = map) **+ expand_limits**(x = map$long, y = map$lat) map_id, alpha, color, fill, linetype, size

THREE VARIABLES

```
seals$z <- with(seals, sqrt(delta_long^2 + delta_lat^2)); l <- ggplot(seals, aes(long, lat))
```

l + geom_contour(aes(z = z)) x, y, z, alpha, color, group, linetype, size, weight

l + geom_contour_filled(aes(fill = z)) x, y, alpha, color, fill, group, linetype, size, subgroup

l + geom_raster(aes(fill = z), hjust = 0.5, vjust = 0.5, interpolate = FALSE) x, y, alpha, fill

l + geom_tile(aes(fill = z)) x, y, alpha, color, fill, linetype, size, width

Further Reading

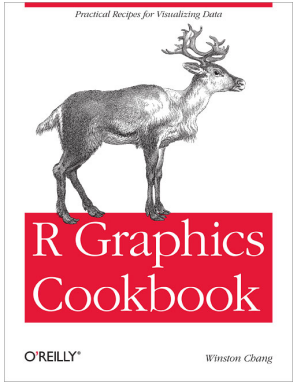


Figure 8: R Graphics Cookbook https://r-graphics.org/

Learning objectives

- Know what Tidyverse is and how to install it within RStudio
- Know how to install and load packages in R
- Create plots using `ggplot2`
- Customise plots using `ggplot2`
- Appreciate and apply principles of data visualisation (independent learning - video)

References

Wickham, Hadley, and Garrett Grolemund. 2020. *R for Data Science: Import, Tidy, Transform, Visualize, and Model Data*.