Note with R4DS

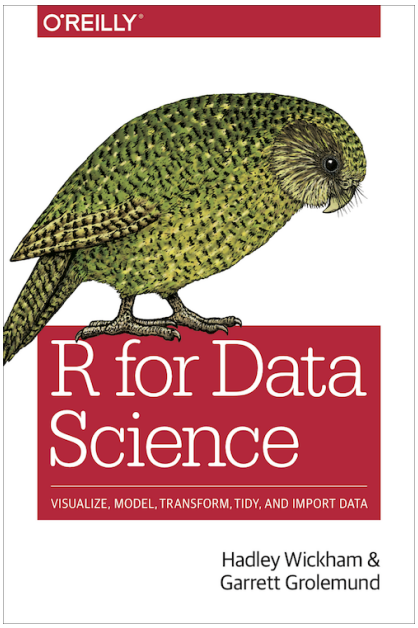
2019-05-22

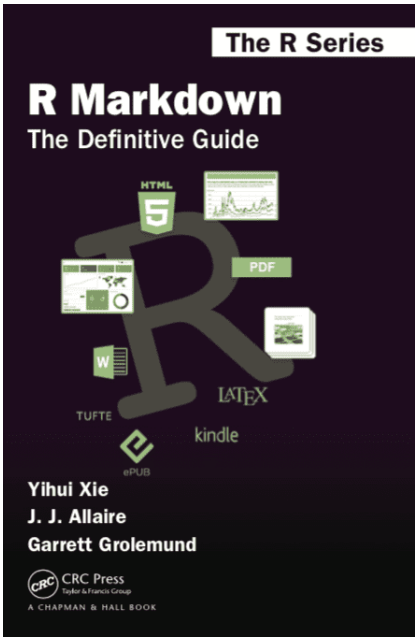
Contents

About this notebook

This notebook is my practice after reading those books:

- **R for Data Science**
- **R Markdown: The Definitive Guide**
- **bookdown: Authoring Books and Technical Documents with R Markdown**





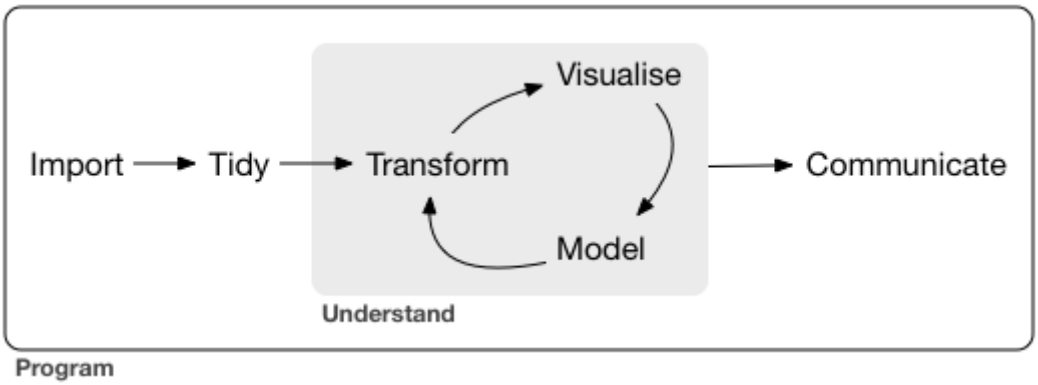
Chapter 1

Introduction

Data science is an exciting discipline that allows you to turn raw data into understanding, insight, and knowledge. The goal of "R for Data Science" is to help you learn the most important tools in R that will allow you to do data science. After reading this book, you'll have the tools to tackle a wide variety of data science challenges, using the best parts of R.

Data Science Project process:

Data science is a huge field, and there's no way you can master it by reading a single book. The goal of this book is to give you a solid foundation in the most important tools. Our model of the tools needed in a typical data science project looks something like this:



What you won't learn

- Big Data
- Python, Julia, and friends
- Non-rectangular data
- Hypothesis confirmation

Prerequisites

R

- Download R from CRAN: https://cran.r-project.org
- Cloud mirror: https://cloud.r-project.org (which automatically figures it out for you.)

RStudio

- Download and install it from http://www.rstudio.com/download
- RStudio IDE Cheat Sheet: https://www.rstudio.com/resources/cheatsheets/#ide

The tidyverse packages

Install the tidyverse packages:

```
if (!require("tidyverse")) install.packages("tidyverse")
```

Load it with the library() function:

```
library(tidyverse)
## Registered S3 methods overwritten by 'ggplot2':
##   method         from
##   [.quosures     rlang
##   c.quosures     rlang
##   print.quosures rlang
## -- Attaching packages --------------------------------- tidyverse 1.2.1 --
## v ggplot2 3.1.1     v purrr   0.3.2
## v tibble  2.1.1     v dplyr   0.8.1
## v tidyr   0.8.3     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
## -- Conflicts ------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

Update the packages:

```
tidyverse_update()
```

Other packages

In this book we'll use three data packages from outside the tidyverse:

```
install.packages(c("nycflights13", "gapminder", "Lahman"))
```

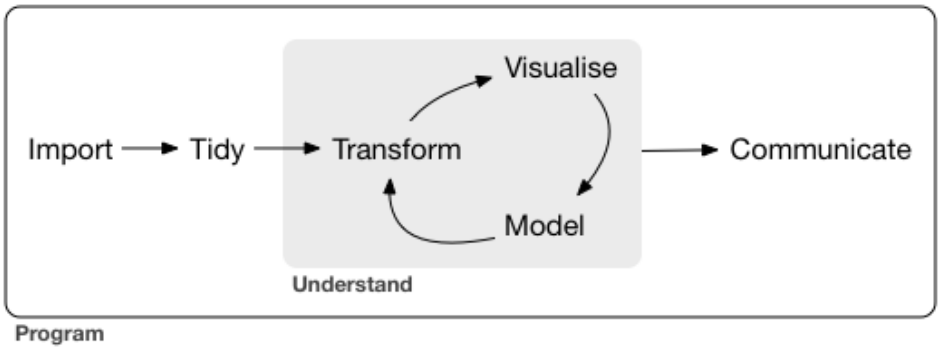
Part I

Explore

Chapter 2

Introduction

The goal of the first part of this book is to get you up to speed with the basic tools of **data exploration** as quickly as possible. Data exploration is the art of looking at your data, rapidly generating hypotheses, quickly testing them, then repeating again and again and again. The goal of data exploration is to generate many promising leads that you can later explore in more depth.



In this part of the book you will learn some useful tools that have an immediate payoff:

- Visualisation is a great place to start with R programming, because the payoff is so clear: you get to make elegant and informative plots that help you understand data. In data visualisation you'll dive into visualisation, learning the basic structure of a ggplot2 plot, and powerful techniques for turning data into plots.
- Visualisation alone is typically not enough, so in data transformation you'll learn the key verbs that allow you to select important variables, filter out key observations, create new variables, and compute summaries.
- Finally, in [exploratory data analysis], you'll combine visualisation and transformation with your curiosity and scepticism to ask and answer interesting questions about data.

Modelling is an important part of the exploratory process, but you don't have the skills to effectively learn or apply it yet. We'll come back to it in modelling, once you're better equipped with more data wrangling and programming tools.

Nestled among these three chapters that teach you the tools of exploration are three chapters that focus on your R workflow. In workflow: basics, [workflow: scripts], and [workflow: projects] you'll learn good practices for writing and organising your R code. These will set you up for success in the long run, as they'll give you the tools to stay organised when you tackle real projects.

Chapter 3

Data visualisation

3.2.4 Exercises

1. Run `ggplot(data = mpg)`. What do you see?

```
   ggplot(data = mpg)
```


   empty graph, because we don't set the aesthetic mapping for plot.

2. How many rows are in `mpg`? How many columns?

```
   dim(mpg)
## [1] 234  11
```

rows: 234, columns: 11

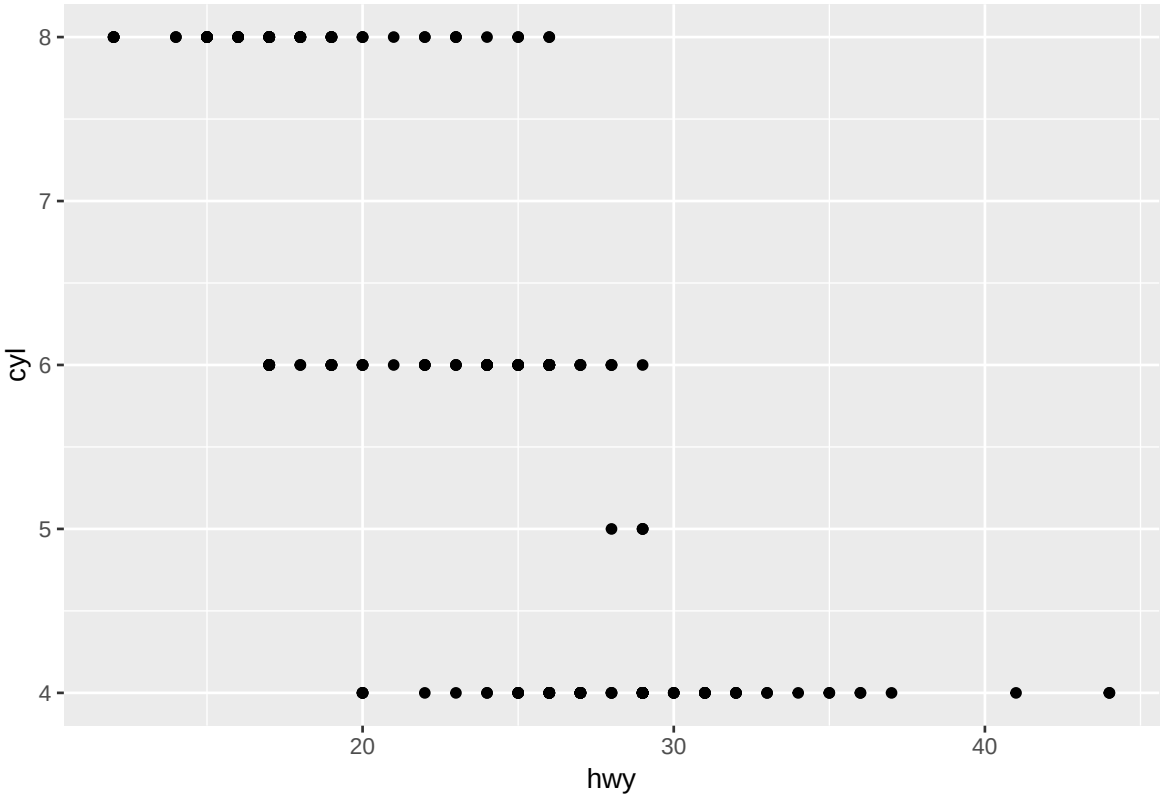
3. What does the `drv` variable describe? Read the help for `?mpg` to find out.

```
?mpg
```

drv: f = front-wheel drive, r = rear wheel drive, 4 = 4wd

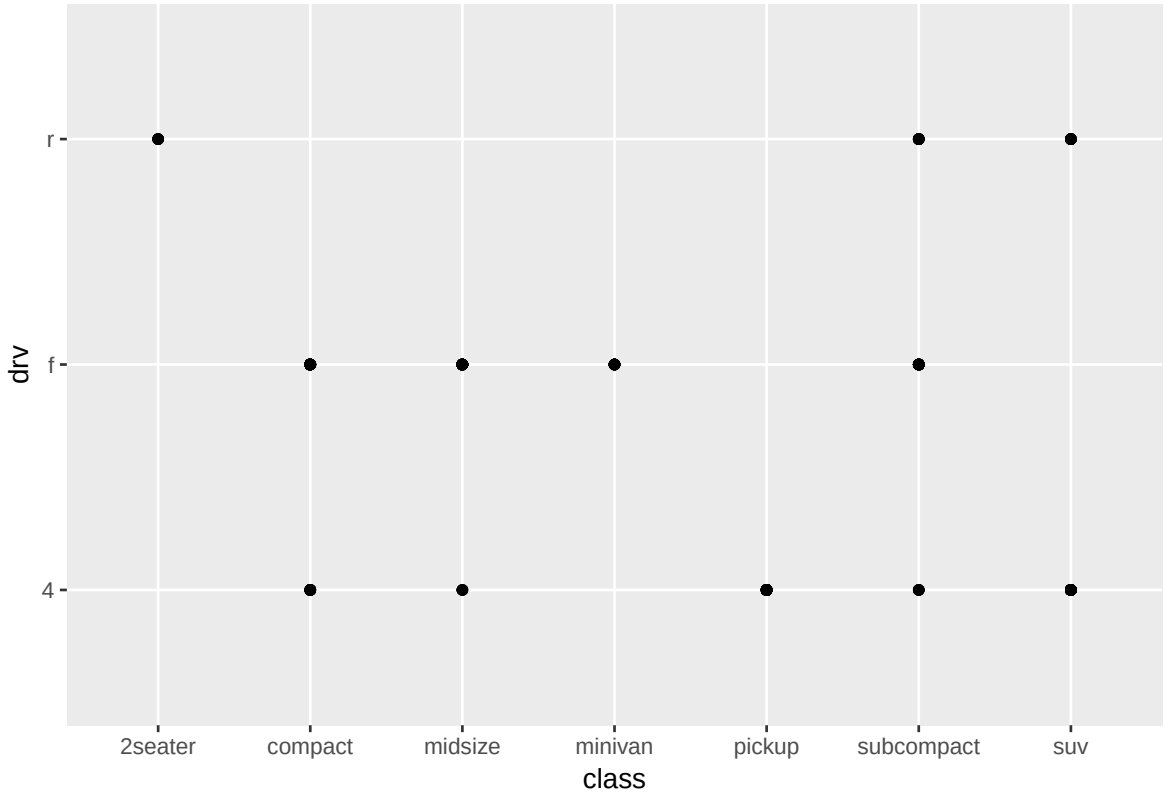
4. Make a scatterplot of `hwy` vs `cyl`.

```
  ggplot(mpg) +
geom_point(aes(x = hwy, y = cyl))
```



5. What happens if you make a scatterplot of `class` vs `drv`? Why is the plot not useful?

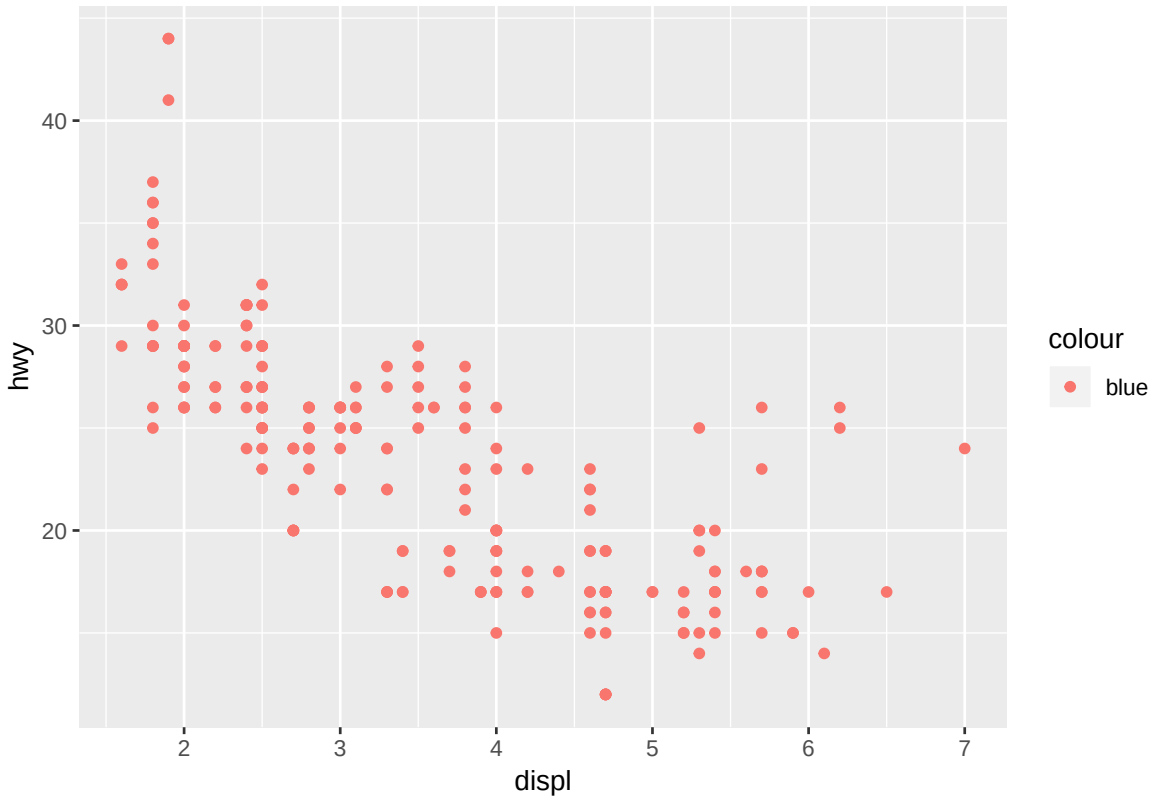
```
  ggplot(mpg) +
geom_point(aes(class, drv))
```



3.3.1 Exercises

1. What's gone wrong with this code? Why are the points not blue?

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



Need to put color attribute outside the aes()
Because the color argument was set within aes(), not geom_point()

2. Which variables in `mpg` are categorical? Which variables are continuous? (Hint: type `?mpg` to read the documentation for the dataset). How can you see this information when you run `mpg`?

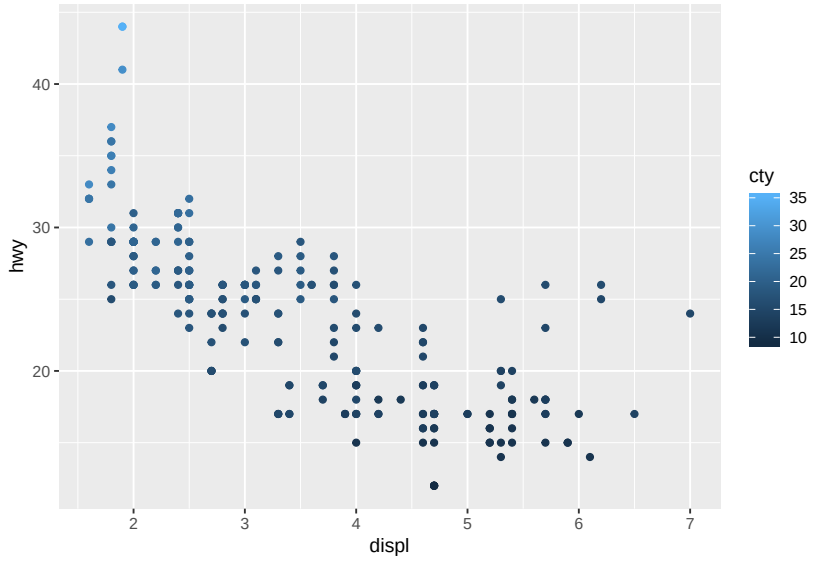
```
   str(mpg)
## Classes 'tbl_df', 'tbl' and 'data.frame':    234 obs. of  11 variables:
##  $ manufacturer: chr  "audi" "audi" "audi" "audi" ...
##  $ model       : chr  "a4" "a4" "a4" "a4" ...
##  $ displ       : num  1.8 1.8 2 2 2.8 2.8 3.1 1.8 1.8 2 ...
##  $ year        : int  1999 1999 2008 2008 1999 1999 2008 1999 1999 2008 ...
##  $ cyl         : int  4 4 4 4 6 6 6 4 4 4 ...
##  $ trans       : chr  "auto(l5)" "manual(m5)" "manual(m6)" "auto(av)" ...
##  $ drv         : chr  "f" "f" "f" "f" ...
##  $ cty         : int  18 21 20 21 16 18 18 18 16 20 ...
##  $ hwy         : int  29 29 31 30 26 26 27 26 25 28 ...
##  $ fl          : chr  "p" "p" "p" "p" ...
##  $ class       : chr  "compact" "compact" "compact" "compact" ...
```

Categorical: manufacturer, model, trans, drv, fl, class
Continuous: displ, cyl, cty, hwy

3. Map a continuous variable to `color`, `size`, and `shape`. How do these aesthetics behave differently for categorical vs. continuous variables?

color:

```
   ggplot(mpg) +
 geom_point(aes(displ, hwy, color = cty))
```

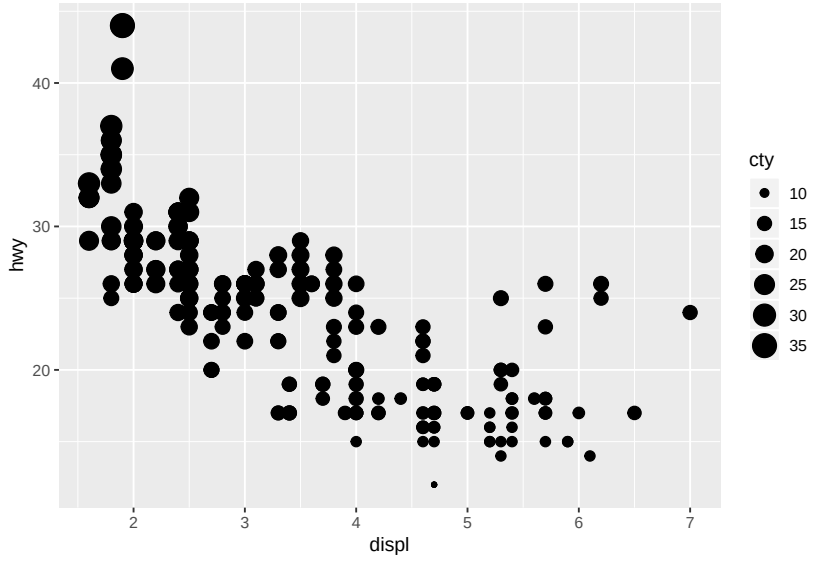


shape:

```
   ggplot(mpg) +
 geom_point(aes(displ, hwy, shape = cty))
## Error: A continuous variable can not be mapped to shape
```

size:

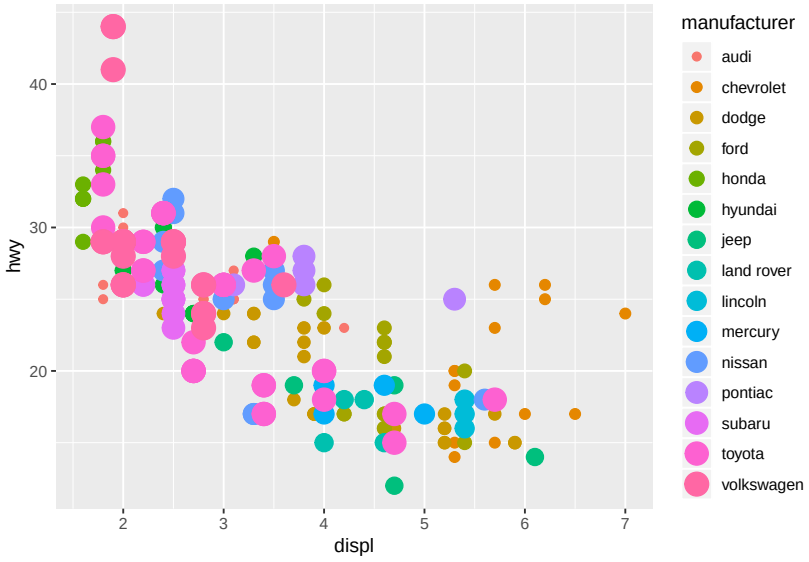
```
   ggplot(mpg) +
geom_point(aes(displ, hwy, size = cty))
```



4. What happens if you map the same variable to multiple aesthetics?

```
   ggplot(mpg) +
geom_point(aes(displ, hwy, color = manufacturer, size = manufacturer))
```

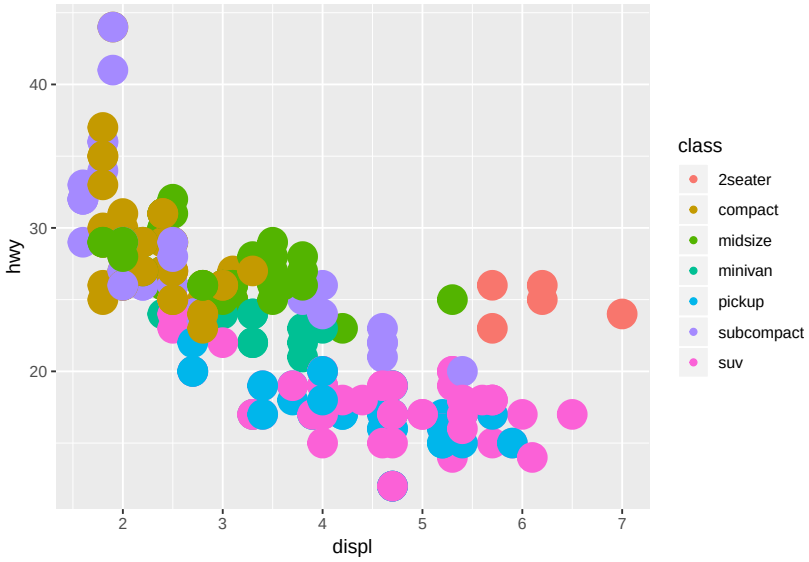
```
## Warning: Using size for a discrete variable is not advised.
```



5. What does the `stroke` aesthetic do? What shapes does it work with? (Hint: use `?geom_point`)

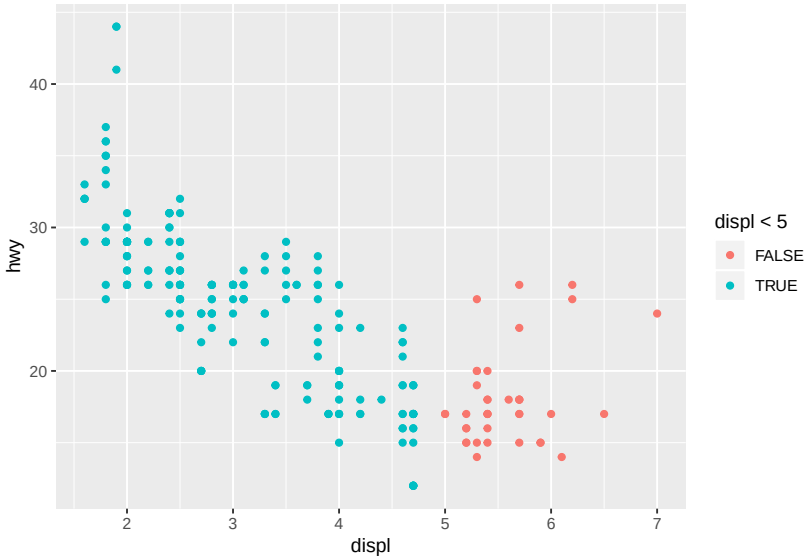
   To modify the width of the border

```
  ggplot(mpg) +
geom_point(aes(displ, hwy, color = class, stroke = 5))
```



6. What happens if you map an aesthetic to something other than a variable name, like `aes(colour = displ < 5)`? Note, you'll also need to specify x and y.

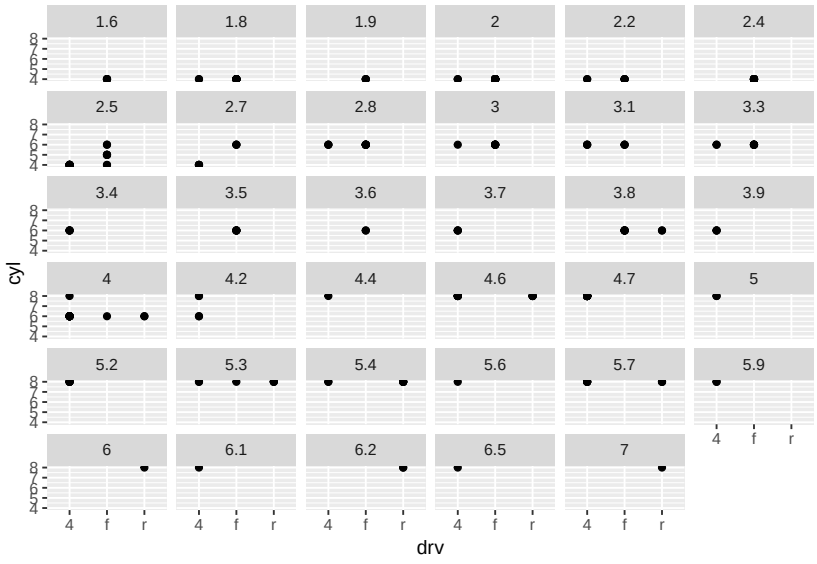
```
  ggplot(mpg) +
geom_point(aes(displ, hwy, color = displ < 5))
```



3.5.1 Exercises

1. What happens if you facet on a continuous variable?

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

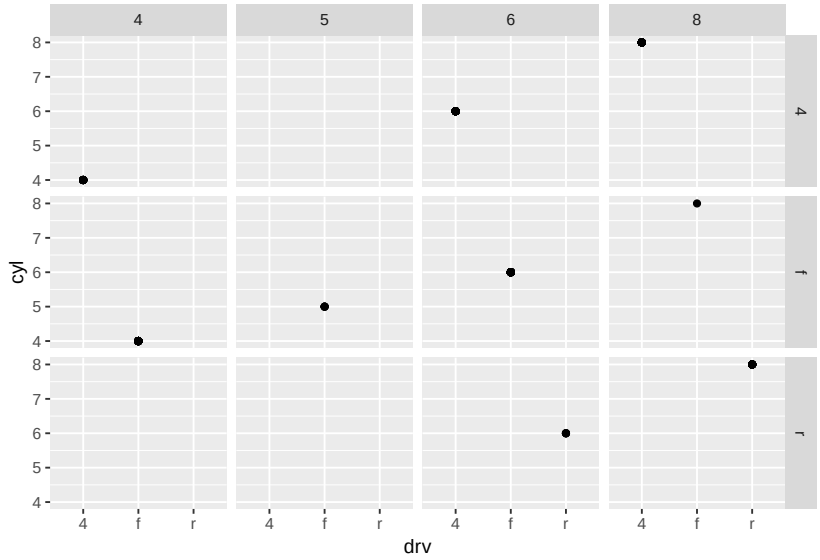


   Your graph will not make much sense. R will try to draw a separate facet for each unique value of the continuous variable. If you have too many unique values, you may crash R.

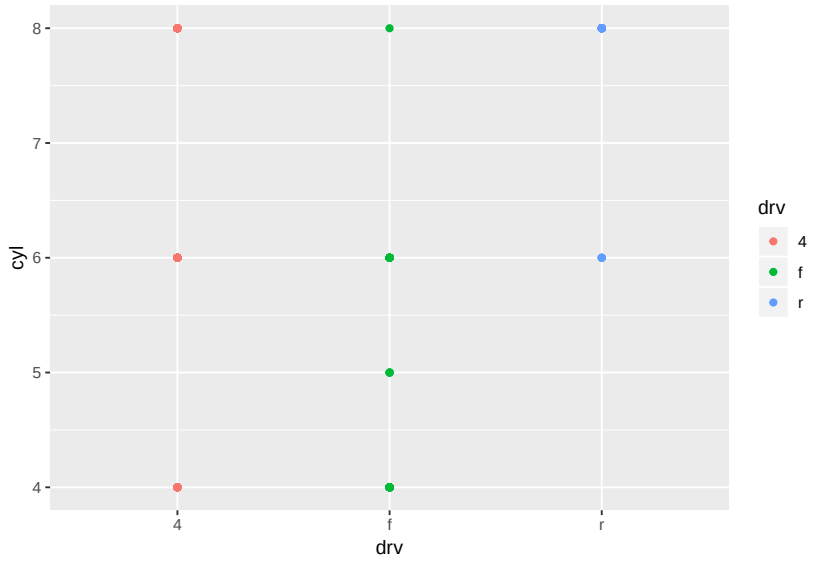
2. What do the empty cells in plot with `facet_grid(drv ~ cyl)` mean? How do they relate to this plot?

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

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




```
  ggplot(data = mpg) +
geom_point(mapping = aes(x = drv, y = cyl, color = drv))
```

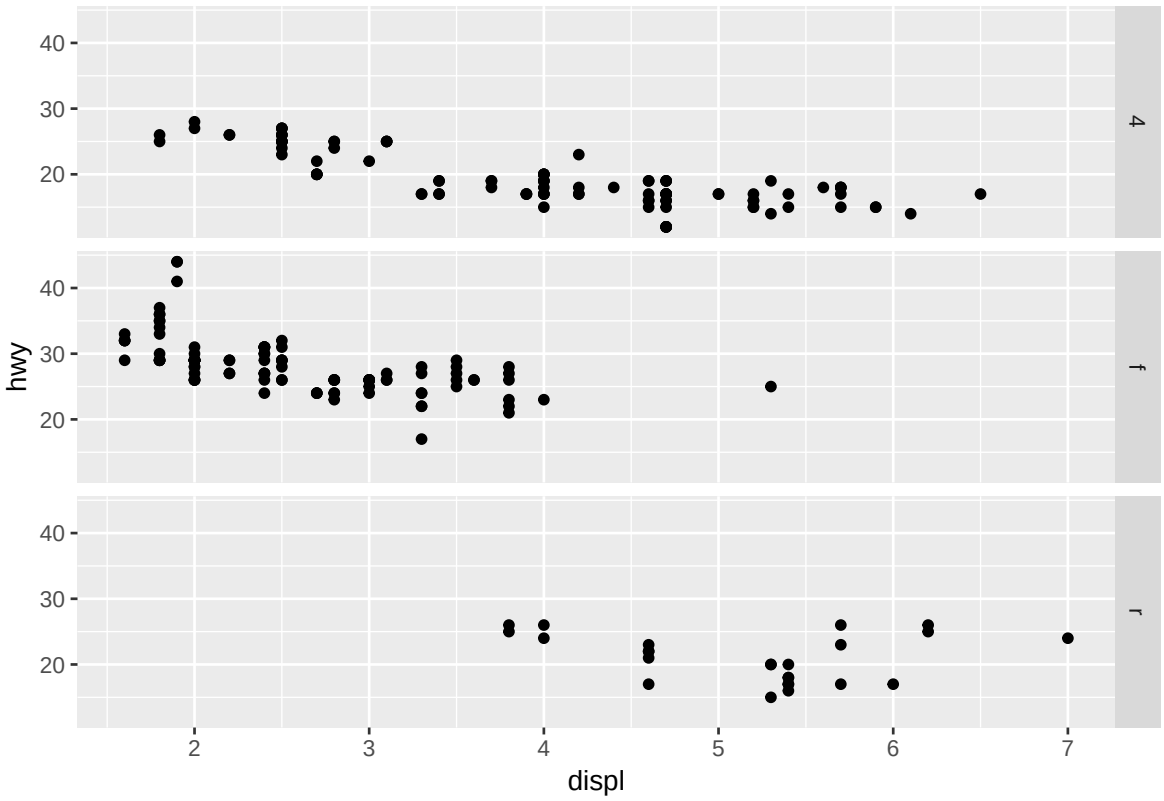


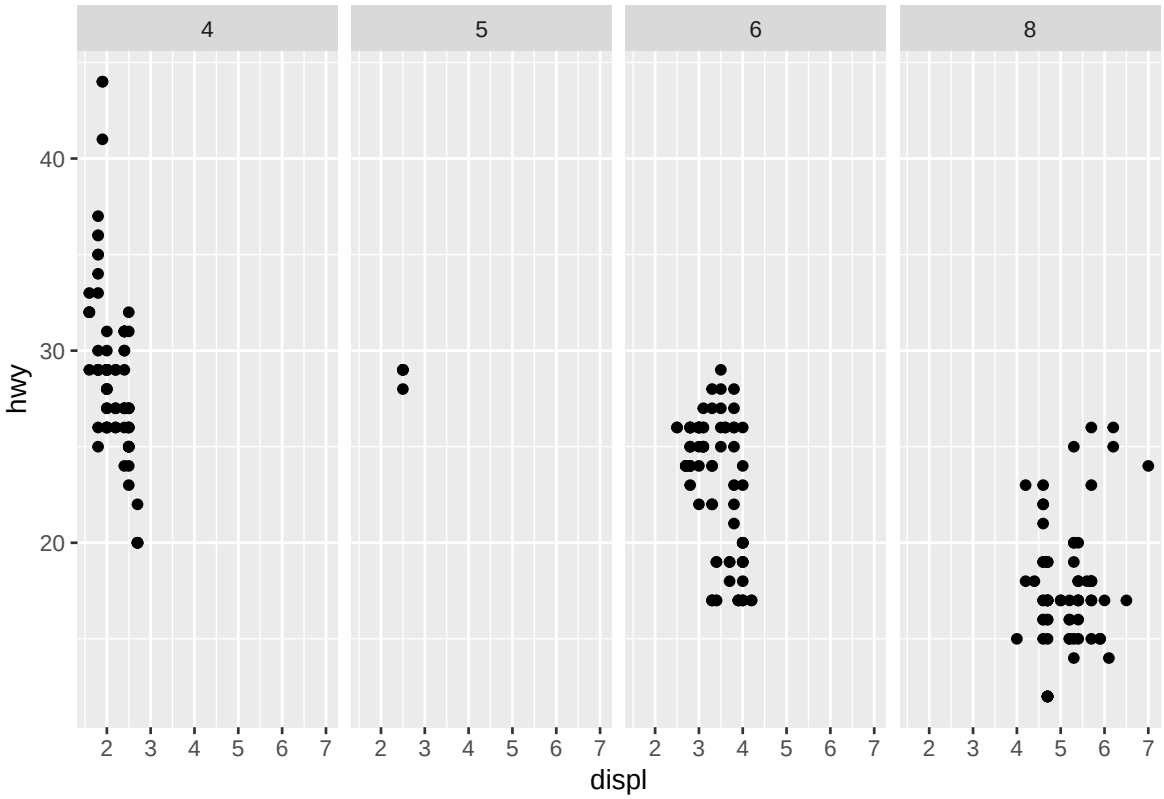
empty cells mean that there are no relation between drv and cyl. no 4 cylinders with rear wheel drive

3. What plots does the following code make? What does . do?

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

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


hwy
40
30
20
4
f
r
2
3
4
5
6
7
displ


4
5
6
8
hwy
40
30
20
2
3
4
5
6
7
displ

Display the plot on the horizontal and/or vertical direction . acts a placeholder for no variable

4. Take the first faceted plot in this section:

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

What are the advantages to using faceting instead of the colour aesthetic?

Faceting splits the data into separate grids and better visualizes trends within each individual facet.

What are the disadvantages?

disadvantage is that by doing so, it is harder to visualize the overall relationship across facets.

How might the balance change if you had a larger dataset?

The color aesthetic is fine when your dataset is small, but with larger datasets points may begin to overlap with one another. In this situation with a colored plot, jittering may not be sufficient because of the additional color aesthetic.

5. Read `?facet_wrap`. What does `nrow` do? What does `ncol` do?

nrow and ncol will show the row numbers and column numbers in the split plot

What other options control the layout of the individual panels?

as.table determines the starting facet to begin filling the plot, and dir determines the starting direction for filling in the plot (horizontal or vertical).

Why doesn't `facet_grid()` have `nrow` and `ncol` arguments?

6. When using `facet_grid()` you should usually put the variable with more unique levels in the columns. Why?

This will extend the plot vertically, where you typically have more viewing space. If you extend it horizontally, the plot will be compressed and harder to view.

3.6.1 Exercises

1. What geom would you use to draw a line chart? A boxplot? A histogram? An area chart?

line chart -> geom_line() boxplot -> geom_boxplot() histrogram -> geom_histogram() area chart -> geom_area()

2. Run this code in your head and predict what the output will look like. Then, run the code in R and check your predictions.

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

3. What does show.legend = FALSE do? What happens if you remove it? Why do you think I used it earlier in the chapter?

show.legend = FALSE, it will set the legend graph unable to see. If remove it, then the plot will show the legend

4. What does the se argument to geom_smooth() do?

Display confidence interval around smooth.

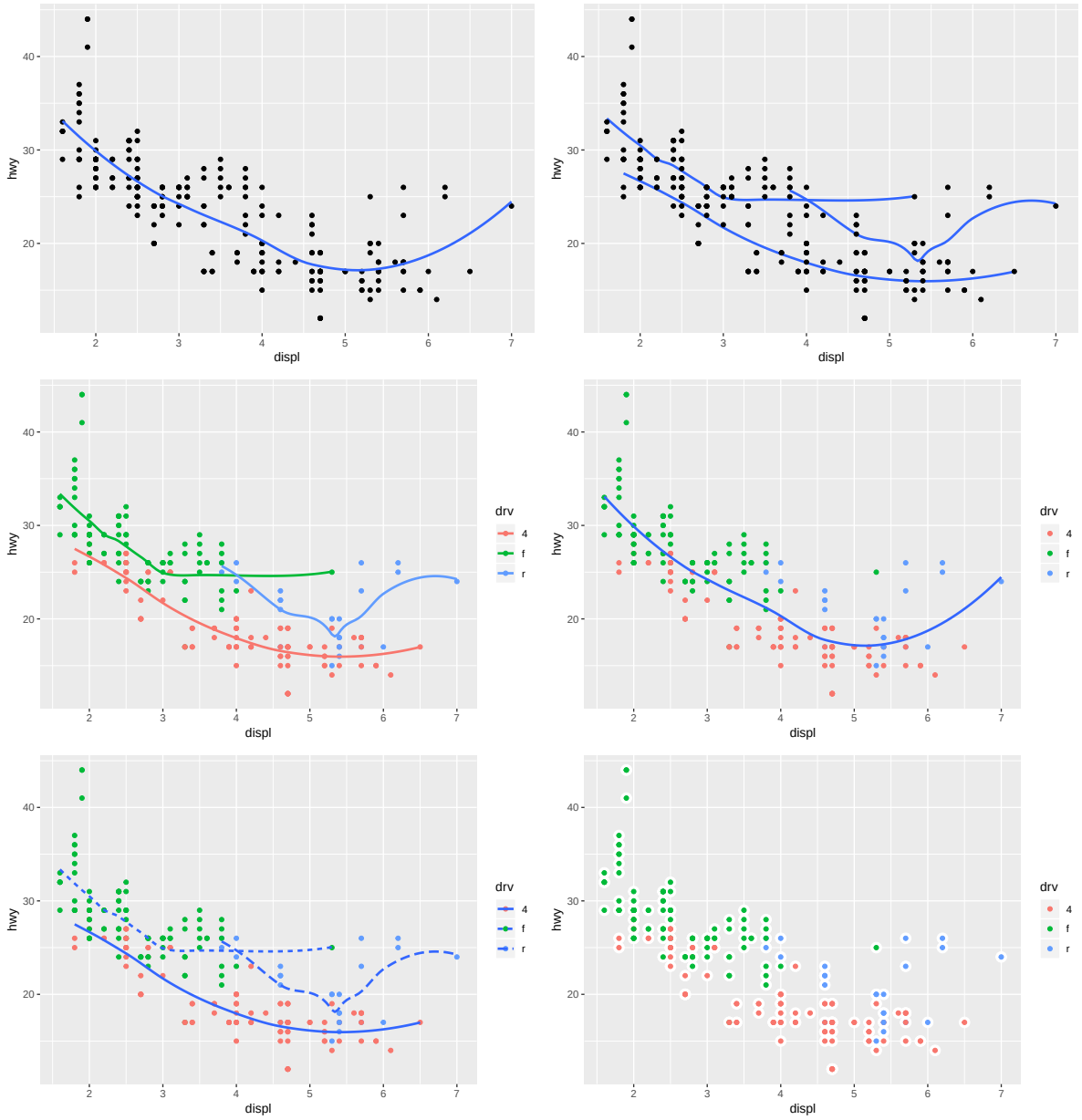
5. Will these two graphs look different? Why/why not?

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

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

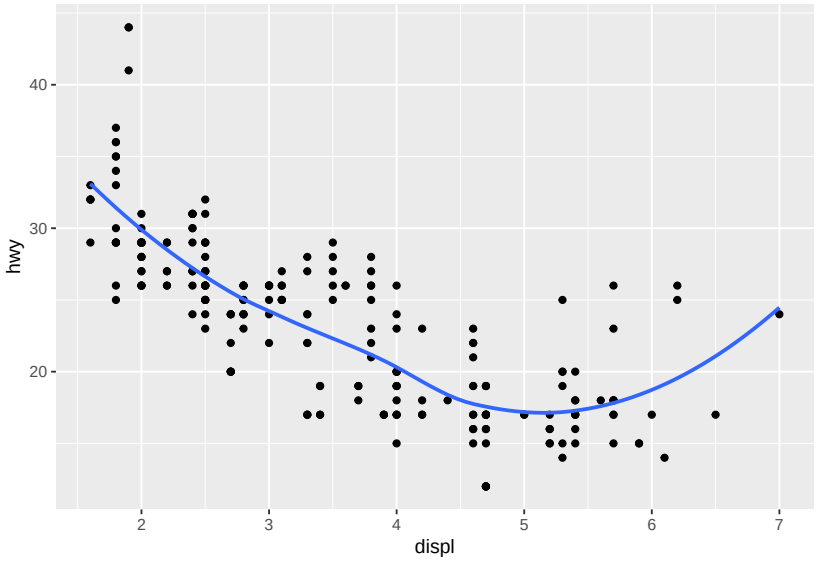
No, because they use the same data and mapping settings. The only difference is that by storing it in the ggplot() function, it is automatically reused for each layer.

6. Recreate the R code necessary to generate the following graphs.



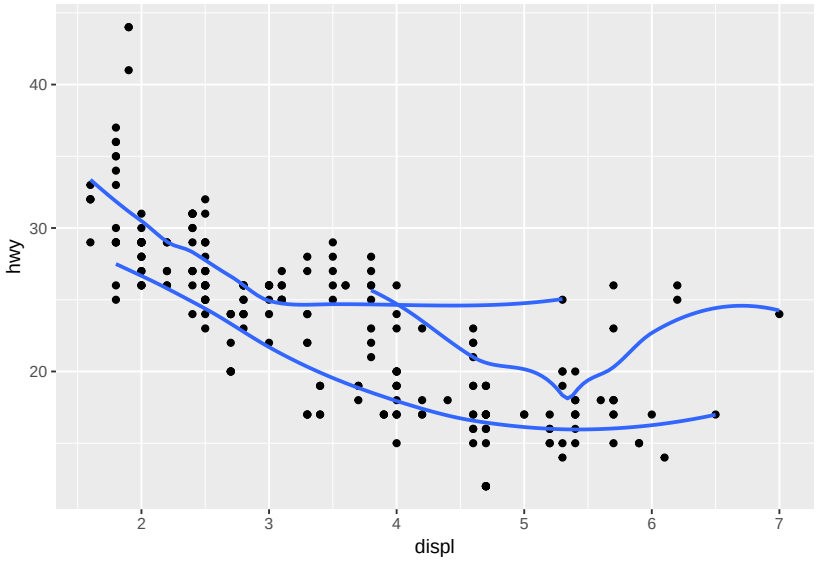
```
### (1)
ggplot(data = mpg, mapping = aes(x = displ,  y = hwy)) +
  geom_point() +
  geom_smooth(se = FALSE)
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```




```
### (2)
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_point() +
  geom_smooth(mapping = aes(group = drv), se = FALSE)
```

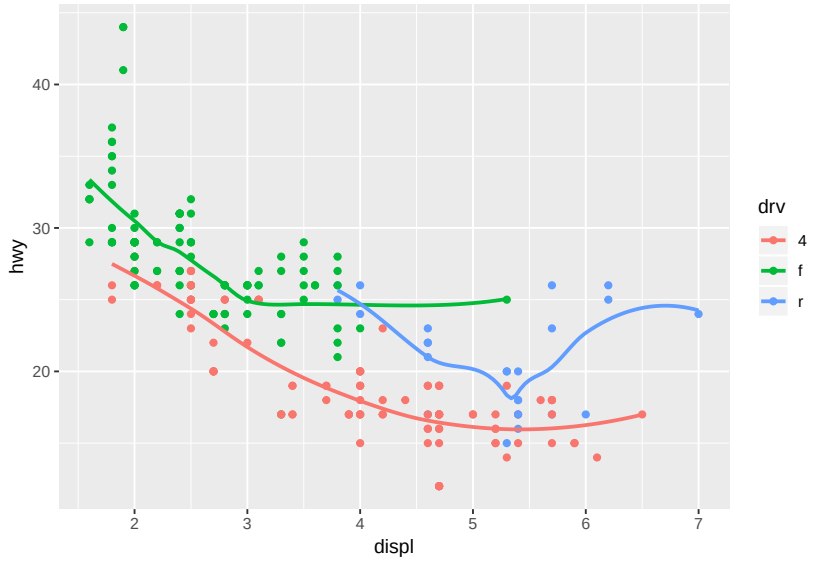
```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```




```
### (3)
ggplot(data = mpg, mapping = aes(x = displ, y = hwy, color = drv)) +
  geom_point() +
```

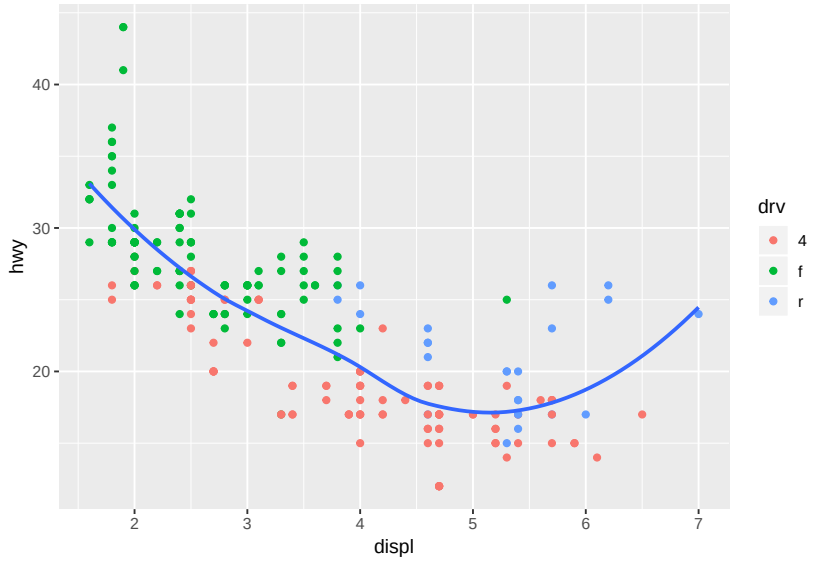
```
  geom_smooth(se = FALSE)
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



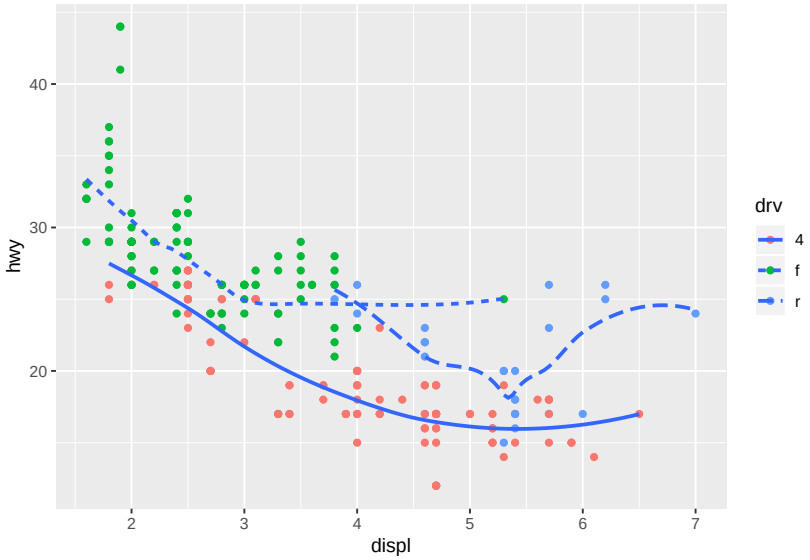

```
### (4)
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_point(mapping = aes(color = drv)) +
  geom_smooth(se = FALSE)
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

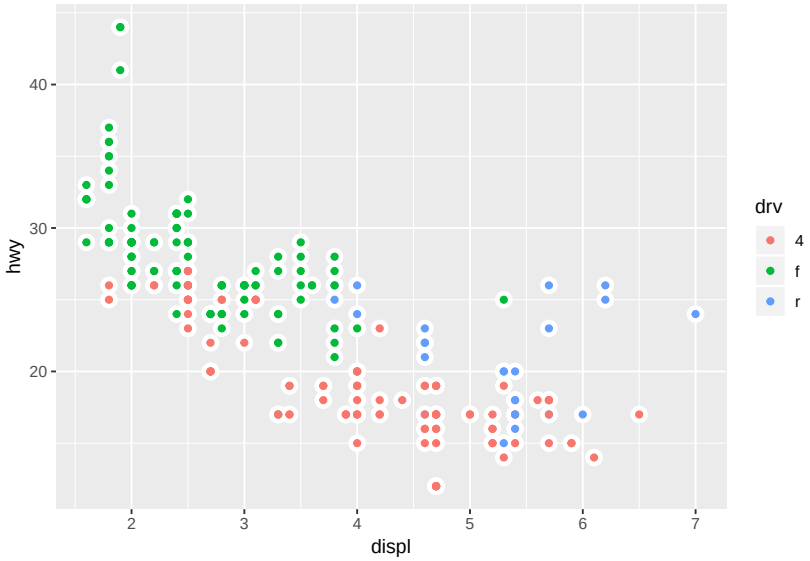



```
  ### (5)
  ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
    geom_point(mapping = aes(color = drv)) +
    geom_smooth(mapping = aes(linetype = drv), se = FALSE)
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```




```
### (6)
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_point(color = 'white', size =4) +
  geom_point(mapping = aes(color = drv))
```



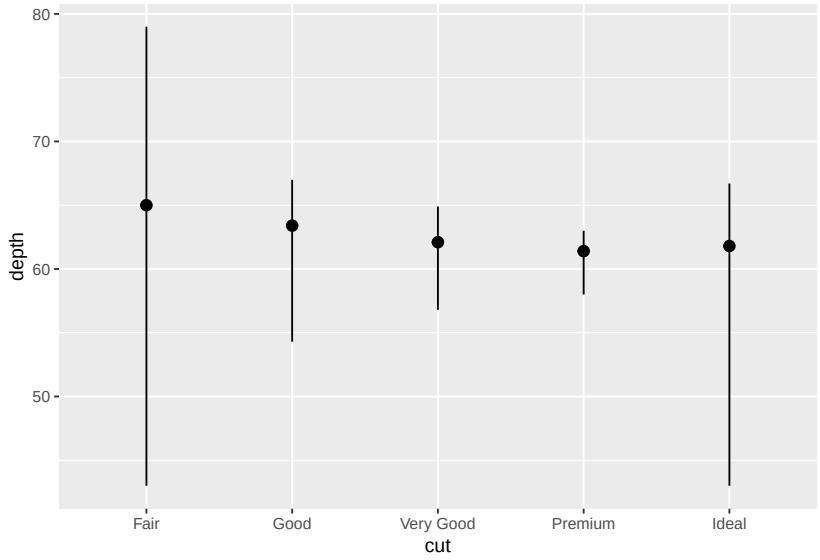
3.7.1 Exercises

1. What is the default geom associated with `stat_summary()`? How could you rewrite the previous plot to use that geom function instead of the stat function?

   Use "?stat_summary()", you'll find the poperty of default geom is geom_pointrange()

```
ggplot(data = diamonds) +
    geom_pointrange (
    mapping = aes(x = cut, y = depth),
    stat = 'summary',
```

```
    fun.ymin = min,
    fun.ymax = max,
    fun.y = median
    )
```



2. What does `geom_col()` do? How is it different to `geom_bar()`?

   There are two types of bar charts: geom_bar() and geom_col(). geom_bar() makes the height of the bar proportional to the number of cases in each group (or if the weight aesthetic is supplied, the sum of the weights). If you want the heights of the bars to represent values in the data, use geom_col() instead. geom_bar() uses stat_count() by default: it counts the number of cases at each x position.

3. Most geoms and stats come in pairs that are almost always used in concert. Read through the documentation and make a list of all the pairs. What do they have in common?

4. What variables does `stat_smooth()` compute? What parameters control its behaviour?

   by '?stat_smooth' -> find computed variables

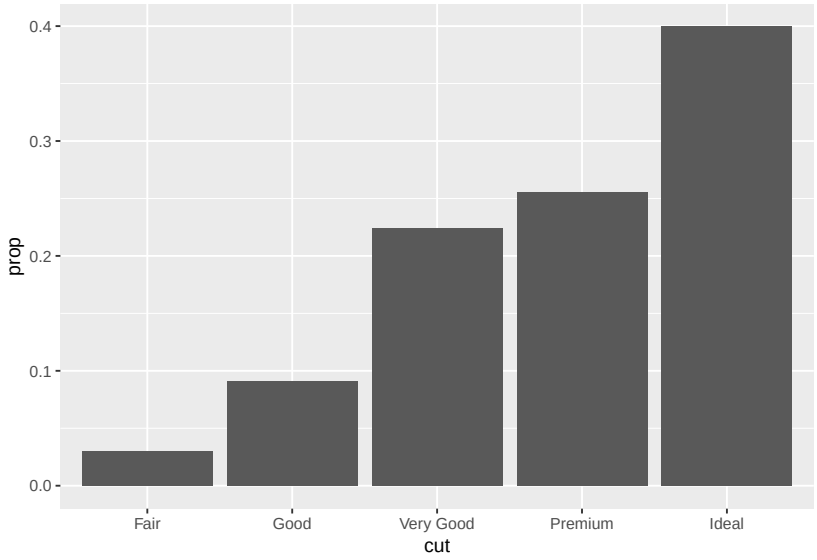
   (1) y: predicted value
   (2) ymin: lower pointwise confidence interval around the mean
   (3) ymax: upper pointwise confidence interval around the mean
   (4) se: standard errorS

5. In our proportion bar chart, we need to set `group = 1`. Why? In other words what is the problem with these two graphs?

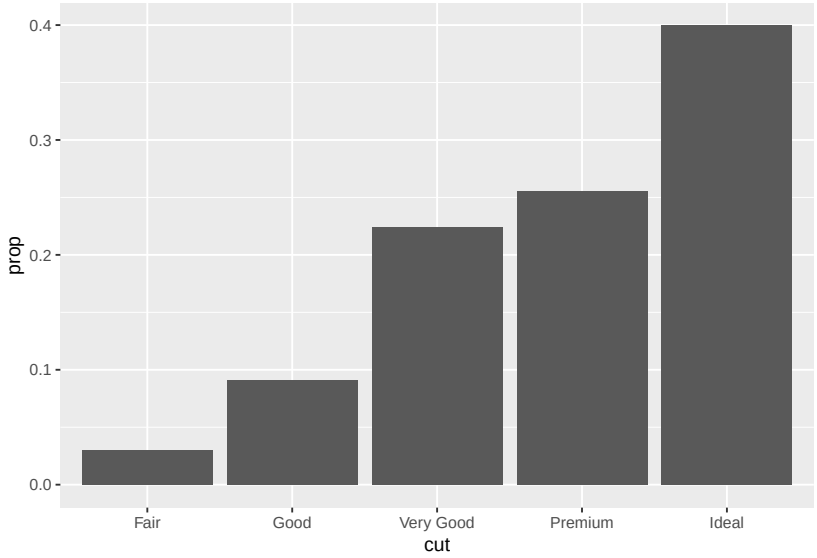
```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, y = ..prop..))
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, fill = color, y = ..prop..))
```

   If we fail to set group = 1, the proportions for each cut are calculated using the complete dataset, rather than each subset of cut.

```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, y = ..prop.., group = 1))
```



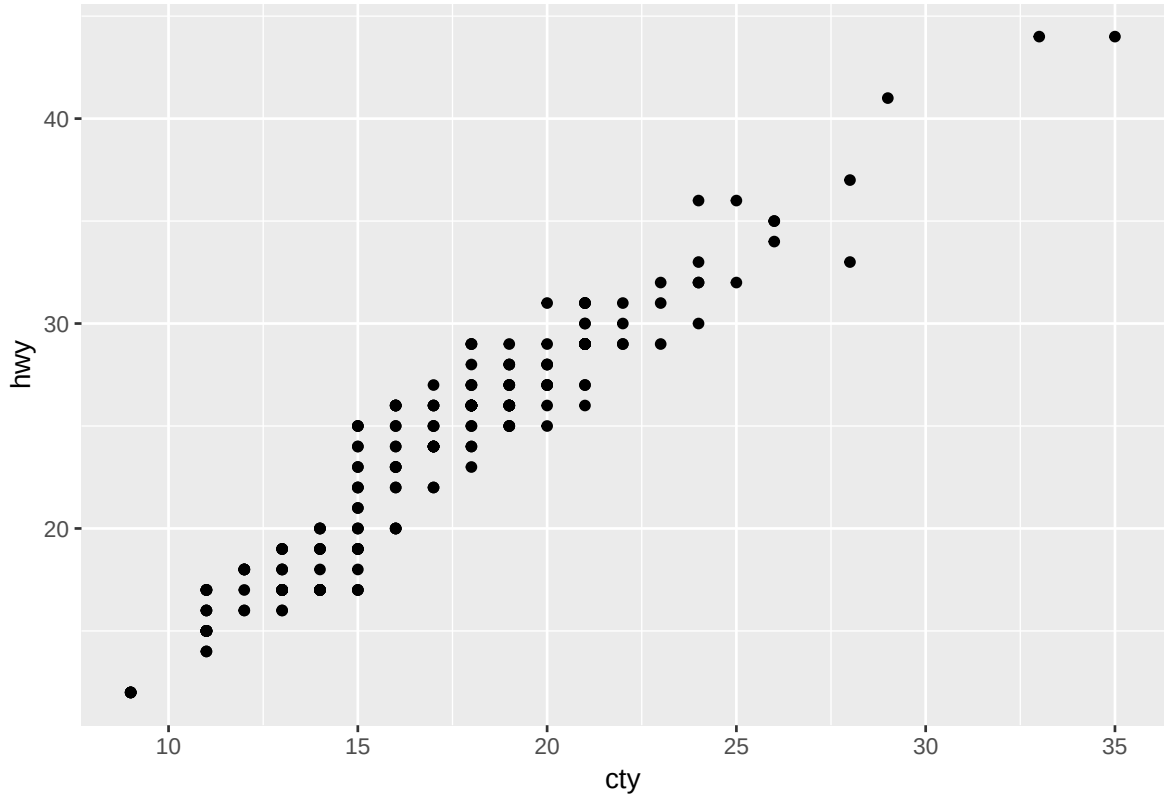

```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, fill = color, y = ..prop.., group = 1))
```



3.8.1 Exercises

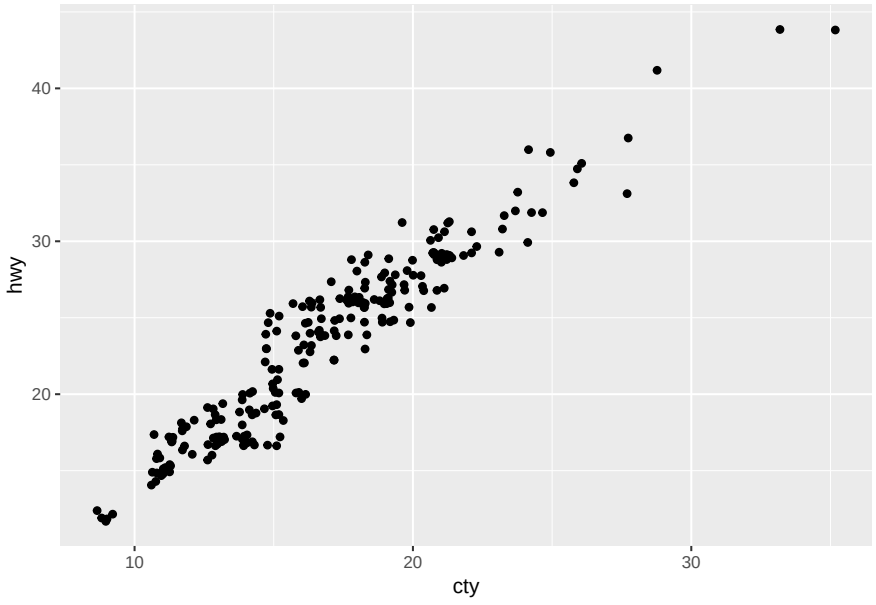
1. What is the problem with this plot? How could you improve it?

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

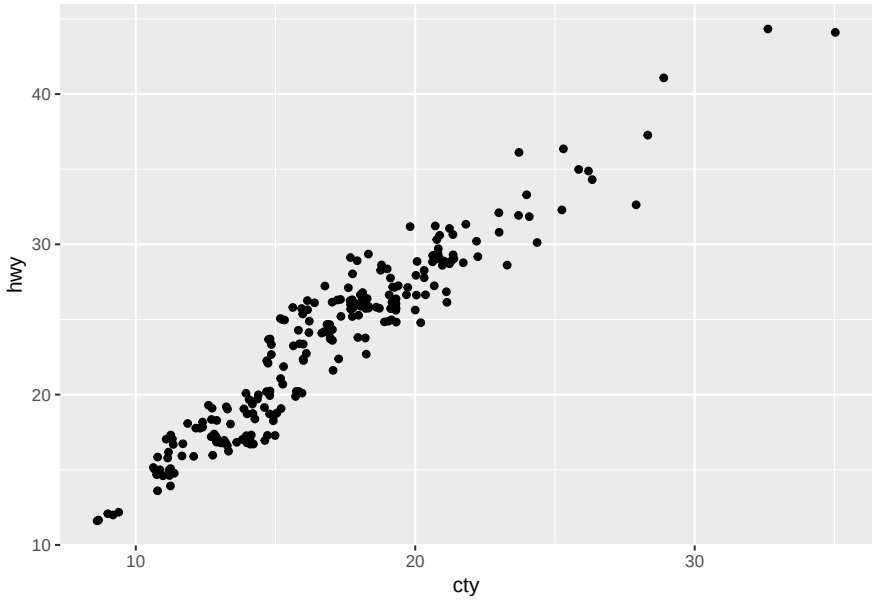


Many of the data points overlap

```
ggplot(data = mpg, mapping = aes(x = cty, y = hwy)) +
  geom_point(position = 'jitter')
```




```
ggplot(data = mpg, mapping = aes(x = cty, y = hwy)) +
  geom_jitter()
```

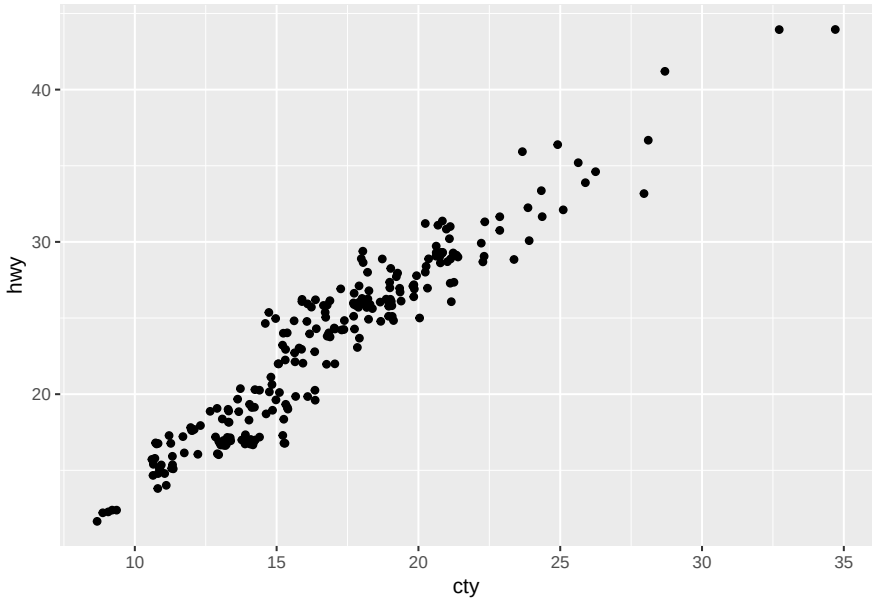


2. What parameters to `geom_jitter()` control the amount of jittering?

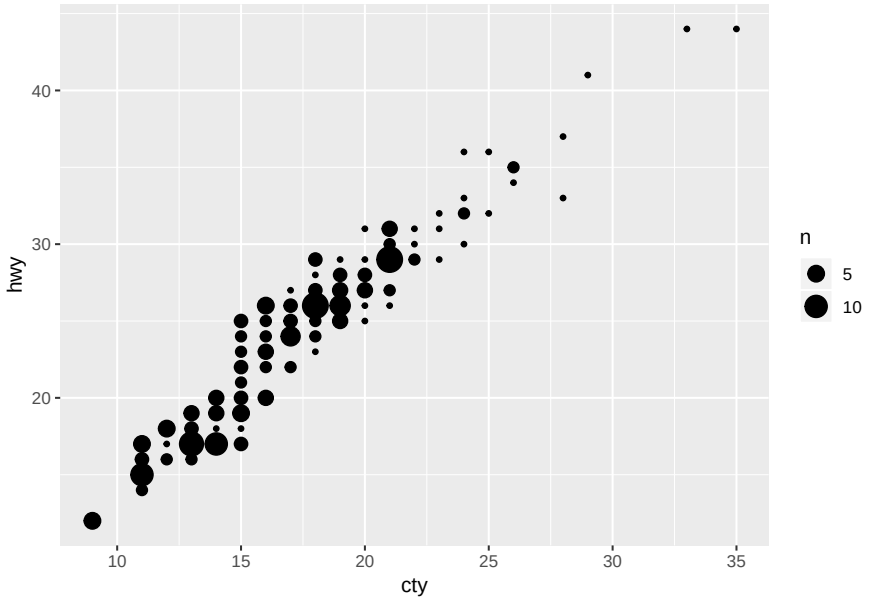
   width and height

3. Compare and contrast `geom_jitter()` with `geom_count()`.

```
ggplot(data = mpg, mapping = aes(x = cty, y = hwy)) +
  geom_jitter()
```




```
ggplot(data = mpg, mapping = aes(x = cty, y = hwy)) +
  geom_count()
```

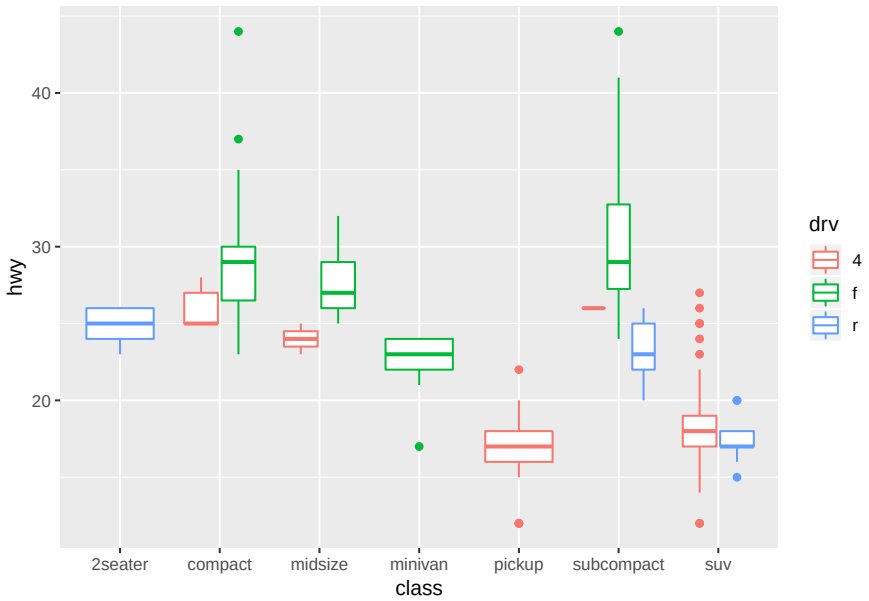


> This is a variant geom_point() that counts the number of observations at each location, then maps the count to point area. It useful when you have discrete data and overplotting.

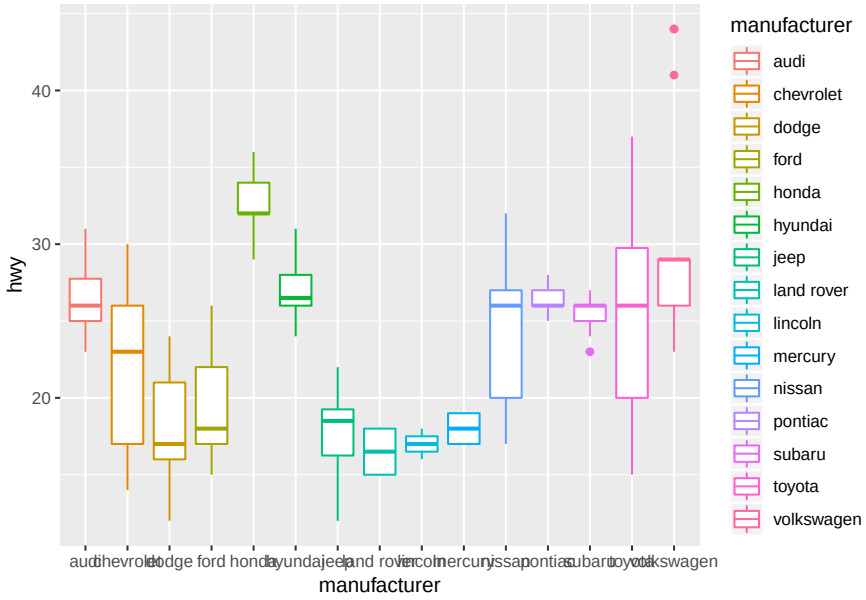
4. What's the default position adjustment for `geom_boxplot()`? Create a visualisation of the `mpg` dataset that demonstrates it.

?geom_boxplot The default position is 'dodge2'

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



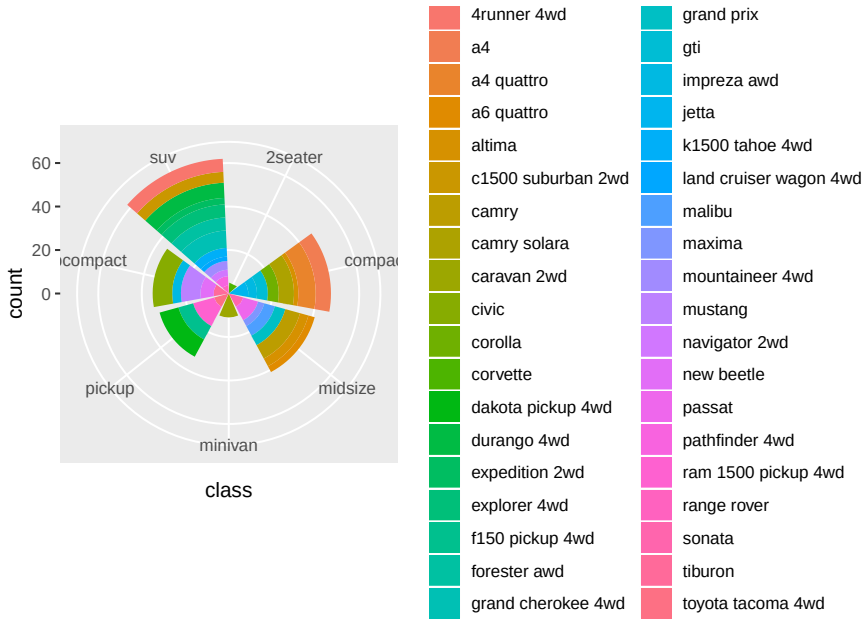

```
ggplot(data = mpg, mapping = aes(x = manufacturer, y = hwy, color = manufacturer)) +
  geom_boxplot()
```



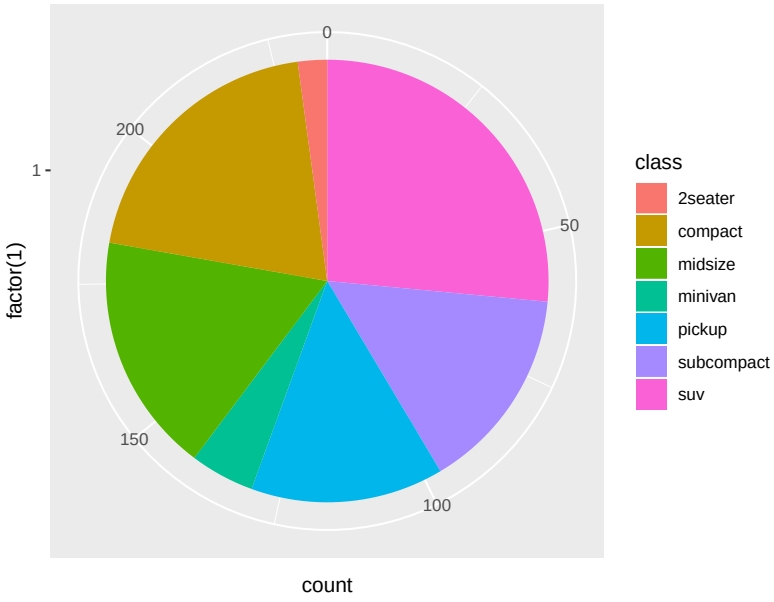
3.9.1 Exercises

1. Turn a stacked bar chart into a pie chart using `coord_polar()`.

```
ggplot(data = mpg) +
  geom_bar(mapping = aes(x = class, y = stat(count), fill = model)) +
  coord_polar()
```




```
ggplot(data = mpg, mapping = aes(x = factor(1), fill = class)) +
  geom_bar(width = 1) +
  coord_polar(theta = "y")
```



2. What does `labs()` do? Read the documentation.

```
?labs()
```

adds labels to the graph. You can add a title, subtitle, and a label for the xand y axes, as well as a caption.

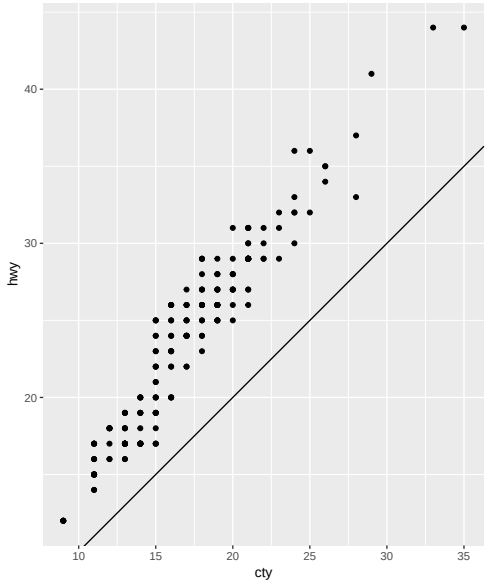
3. What's the difference between `coord_quickmap()` and `coord_map()`?

```
?coord_map
?coord_quickmap
```

coord_map projects a portion of the earth, which is approximately spherical, onto a flat 2D plane using any projection defined by the mapproj package. Map projections do not, in general, preserve straight lines, so this requires considerable computation. coord_quickmap is a quick approximation that does preserve straight lines. It works best for smaller areas closer to the equator.

4. What does the plot below tell you about the relationship between city and highway mpg? Why is `coord_fixed()` important? What does `geom_abline()` do?

```
ggplot(data = mpg, mapping = aes(x = cty, y = hwy)) +
  geom_point() +
  geom_abline() +
  coord_fixed()
```




```
?coord_fixed
?geom_abline
```

The relationships is approximately linear, though overall cars have slightly better highway mileage than city mileage. But using coord_fixed(), the plot draws equal intervals on the x and y axes so they are directly comparable. geom_abline() draws a line that, by default, has an intercept of 0 and slope of 1. This aids us in our discovery that automobile gas efficiency is on average slightly higher for highways than city driving, though the slope of the relationship

Chapter 4

Workflow: basics

Practice

1. Why does this code not work?

```
my_variable <- 10
my_varıable
```

```
## Error in eval(expr, envir, enclos): object 'my_varıable' not found
```

Look carefully! (This may seem like an exercise in pointlessness, but training your brain to notice even the tiniest difference will pay off when programming.)

not my_var ı able, instead is my_var i able.

2. Tweak each of the following R commands so that they run correctly:

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

fliter(mpg, cyl = 8)
filter(diamond, carat > 3)
```

```
filter(mpg, cyl == 8)
```

```
## # A tibble: 70 x 11
##    manufacturer model displ  year   cyl trans drv     cty   hwy fl    class
##    <chr>        <chr> <dbl> <int> <int> <chr> <chr> <int> <int> <chr> <chr>
##  1 audi         a6 q~   4.2  2008     8 auto~ 4        16    23 p     mids~
##  2 chevrolet    c150~   5.3  2008     8 auto~ r        14    20 r     suv
##  3 chevrolet    c150~   5.3  2008     8 auto~ r        11    15 e     suv
##  4 chevrolet    c150~   5.3  2008     8 auto~ r        14    20 r     suv
##  5 chevrolet    c150~   5.7  1999     8 auto~ r        13    17 r     suv
##  6 chevrolet    c150~   6    2008     8 auto~ r        12    17 r     suv
##  7 chevrolet    corv~   5.7  1999     8 manu~ r        16    26 p     2sea~
##  8 chevrolet    corv~   5.7  1999     8 auto~ r        15    23 p     2sea~
##  9 chevrolet    corv~   6.2  2008     8 manu~ r        16    26 p     2sea~
## 10 chevrolet    corv~   6.2  2008     8 auto~ r        15    25 p     2sea~
## # ... with 60 more rows
```

```
filter(diamonds, carat > 3)
```

```
## # A tibble: 32 x 10
##    carat cut     color clarity depth table price     x     y     z
##    <dbl> <ord>   <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
##  1  3.01 Premium I     I1       62.7    58  8040  9.1   8.97  5.67
##  2  3.11 Fair    J     I1       65.9    57  9823  9.15  9.02  5.98
##  3  3.01 Premium F     I1       62.2    56  9925  9.24  9.13  5.73
##  4  3.05 Premium E     I1       60.9    58 10453  9.26  9.25  5.66
##  5  3.02 Fair    I     I1       65.2    56 10577  9.11  9.02  5.91
##  6  3.01 Fair    H     I1       56.1    62 10761  9.54  9.38  5.31
##  7  3.65 Fair    H     I1       67.1    53 11668  9.53  9.48  6.38
##  8  3.24 Premium H     I1       62.1    58 12300  9.44  9.4   5.85
##  9  3.22 Ideal   I     I1       62.6    55 12545  9.49  9.42  5.92
## 10  3.5  Ideal   H     I1       62.8    57 12587  9.65  9.59  6.03
## # ... with 22 more rows
```

3. Press Alt + Shift + K. What happens? How can you get to the same place using the menus?

   In the tool bar, help > keyboard shortcuts help

Chapter 5

Data transformation

5.2.4 Exercises

1. Find all flights that
    1. Had an arrival delay of two or more hours
    2. Flew to Houston (`IAH` or `HOU`)
    3. Were operated by United, American, or Delta
    4. Departed in summer (July, August, and September)
    5. Arrived more than two hours late, but didn't leave late
    6. Were delayed by at least an hour, but made up over 30 minutes in flight
    7. Departed between midnight and 6am (inclusive)
2. Another useful dplyr filtering helper is `between()`. What does it do? Can you use it to simplify the code needed to answer the previous challenges?
3. How many flights have a missing `dep_time`? What other variables are missing? What might these rows represent?
4. Why is `NA ^ 0` not missing? Why is `NA | TRUE` not missing? Why is `FALSE & NA` not missing? Can you figure out the general rule? (`NA * 0` is a tricky counterexample!)