Note with R4DS

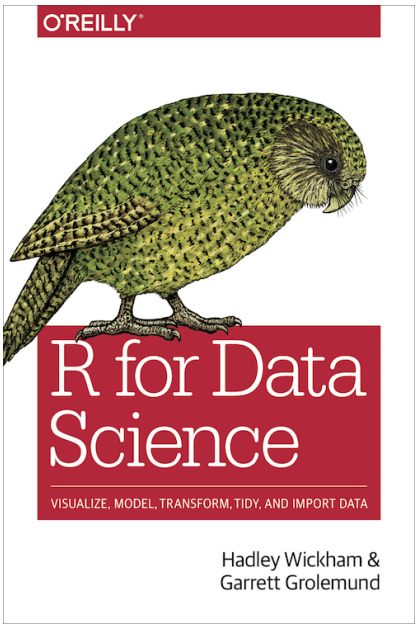
2019-05-18

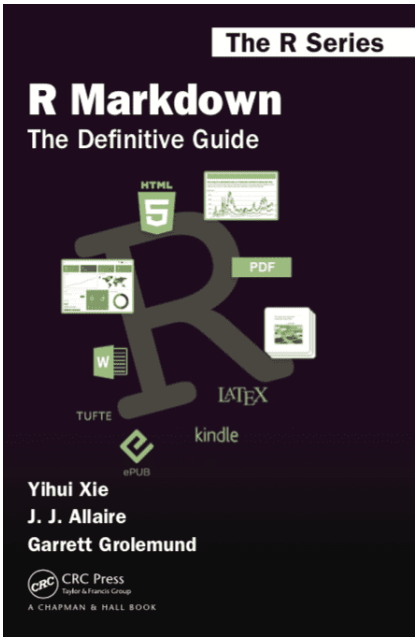
Contents

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**





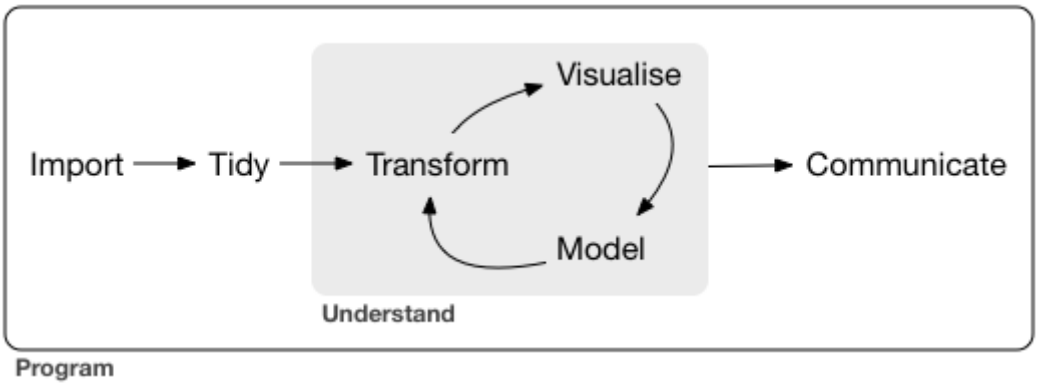
Chapter 1

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"))
```

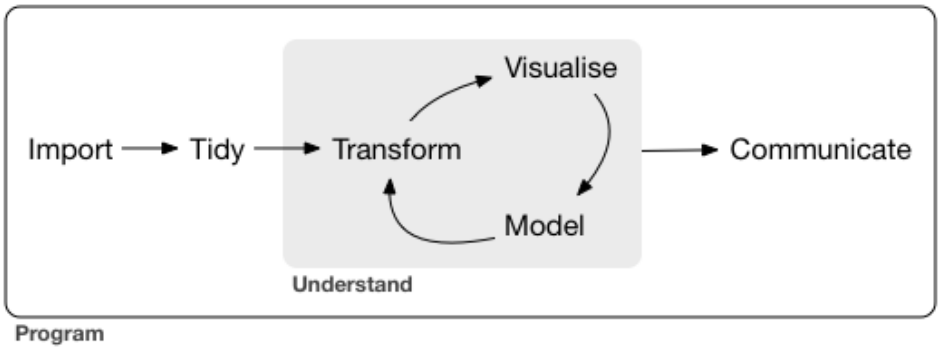
Part I

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

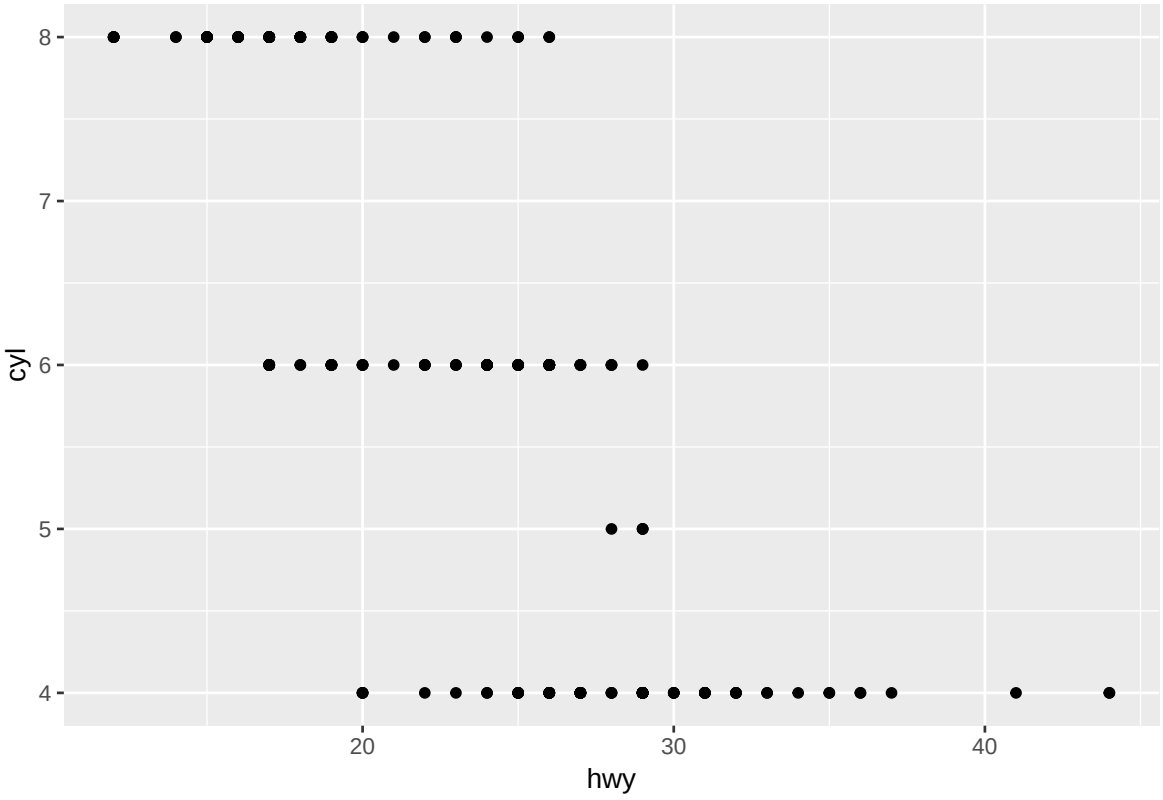
3. What does the `drv` variable describe? Read the help for `?mpg` to find out.

```
?mpg
```

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

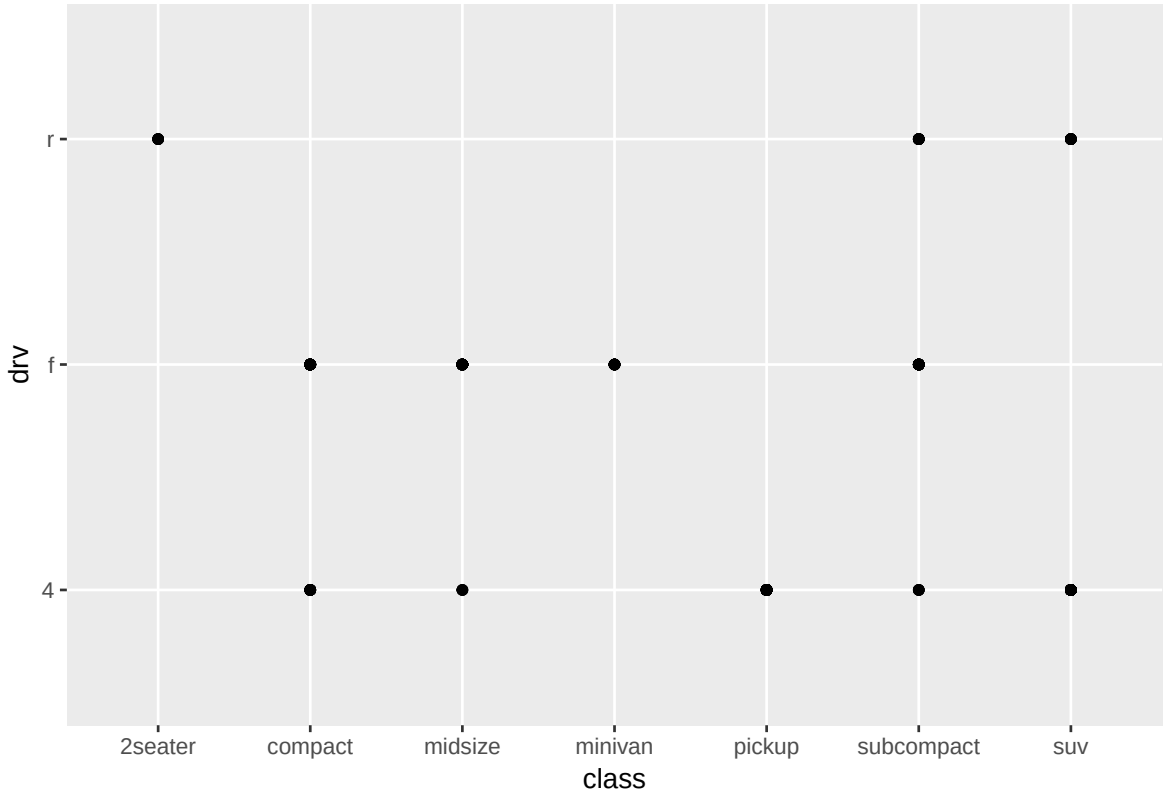
4. Make a scatterplot of `hwy` vs `cyl`.

```
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?

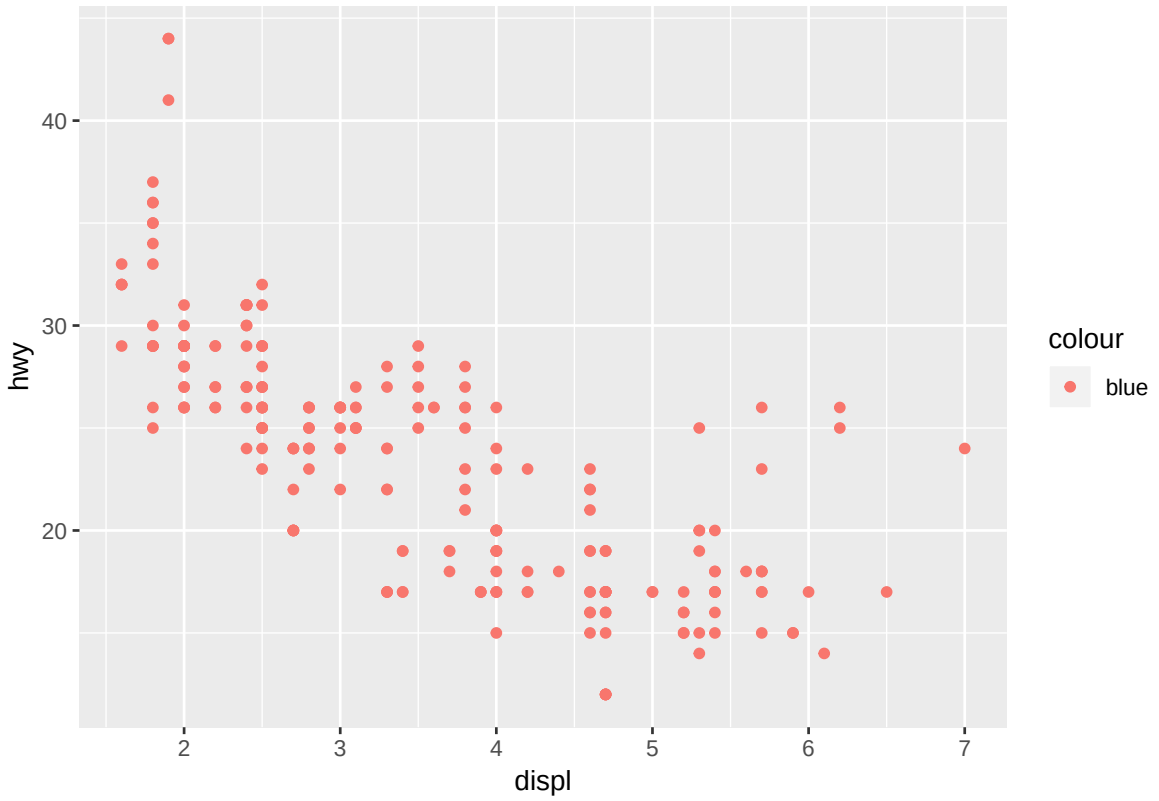
```
ggplot(mpg) +
  geom_point(aes(class, drv))
```



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`?

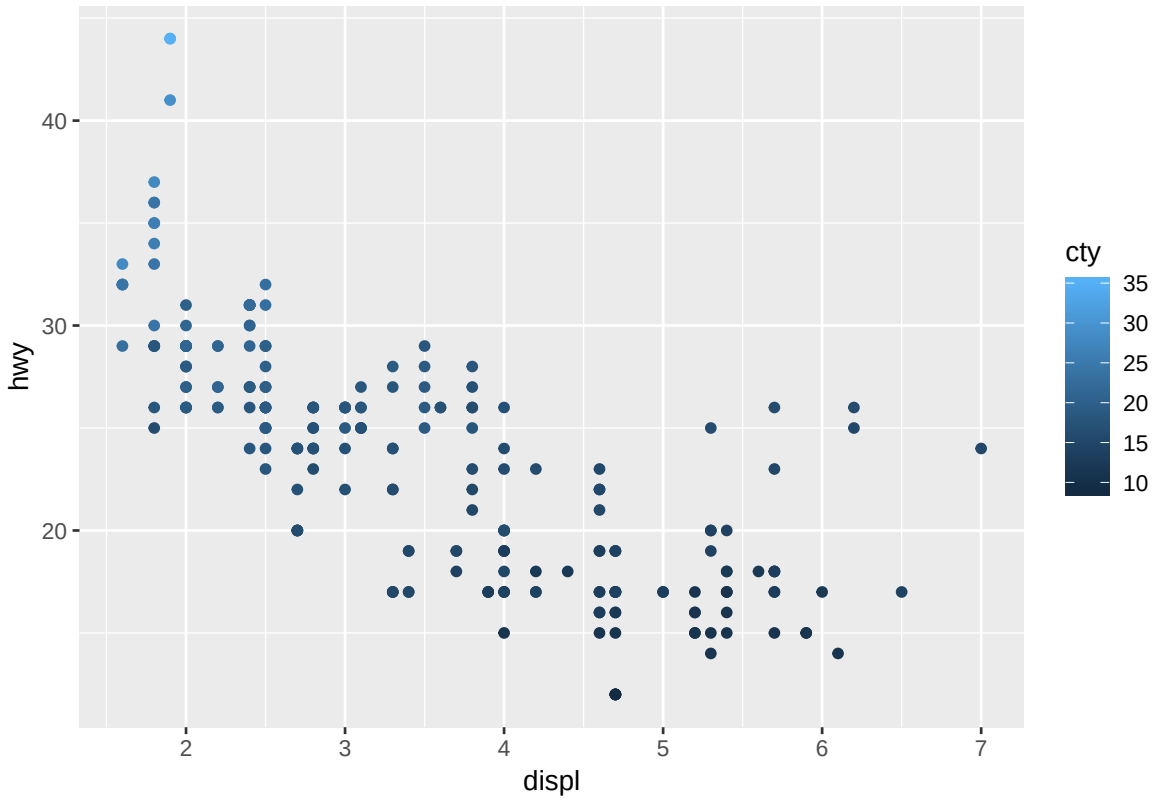
```
str(mpg)
## Classes 'tbl_df', 'tbl' and 'data.frame': 234 obs. of  11 variables:
##  $ manufacturer: chr  "audi" "audi" "audi" "audi" ...
##  $ model       : chr  "a4" "a4" "a4" "a4" ...
##  $ displ       : num  1.8 1.8 2 2 2.8 2.8 3.1 1.8 1.8 2 ...
##  $ year        : int  1999 1999 2008 2008 1999 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))
```

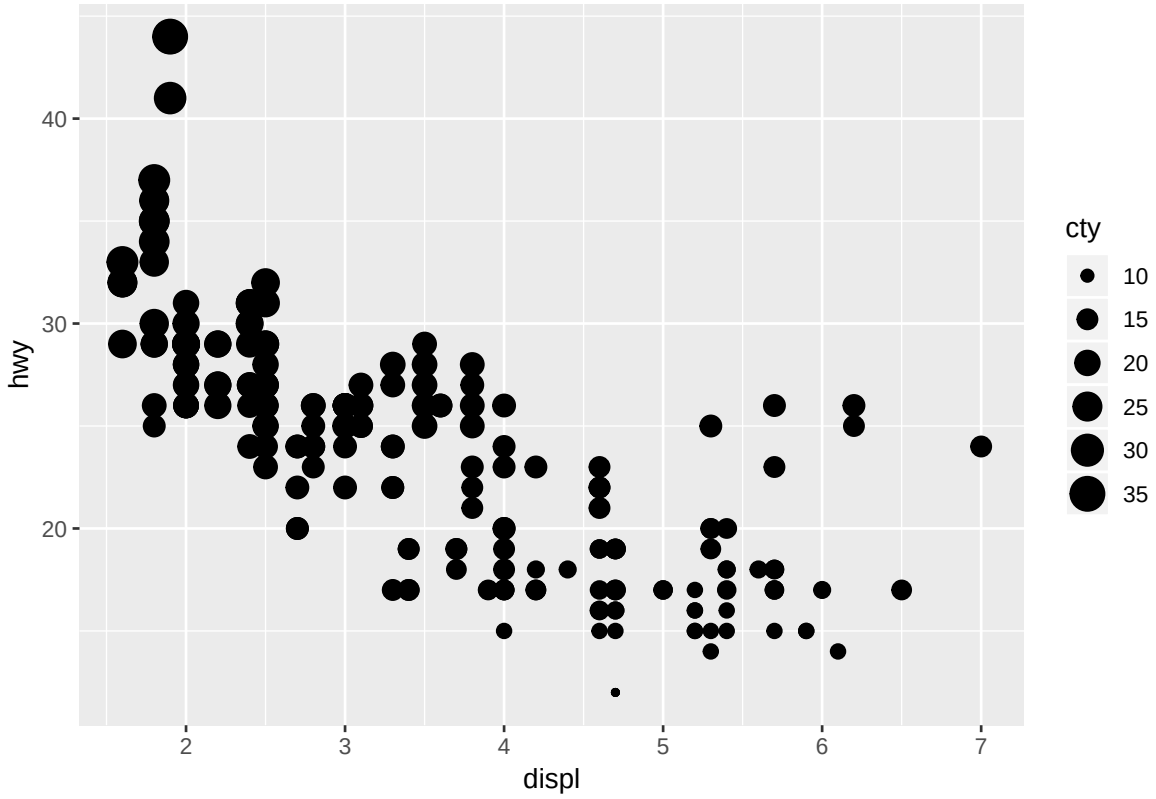


shape:

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

size:

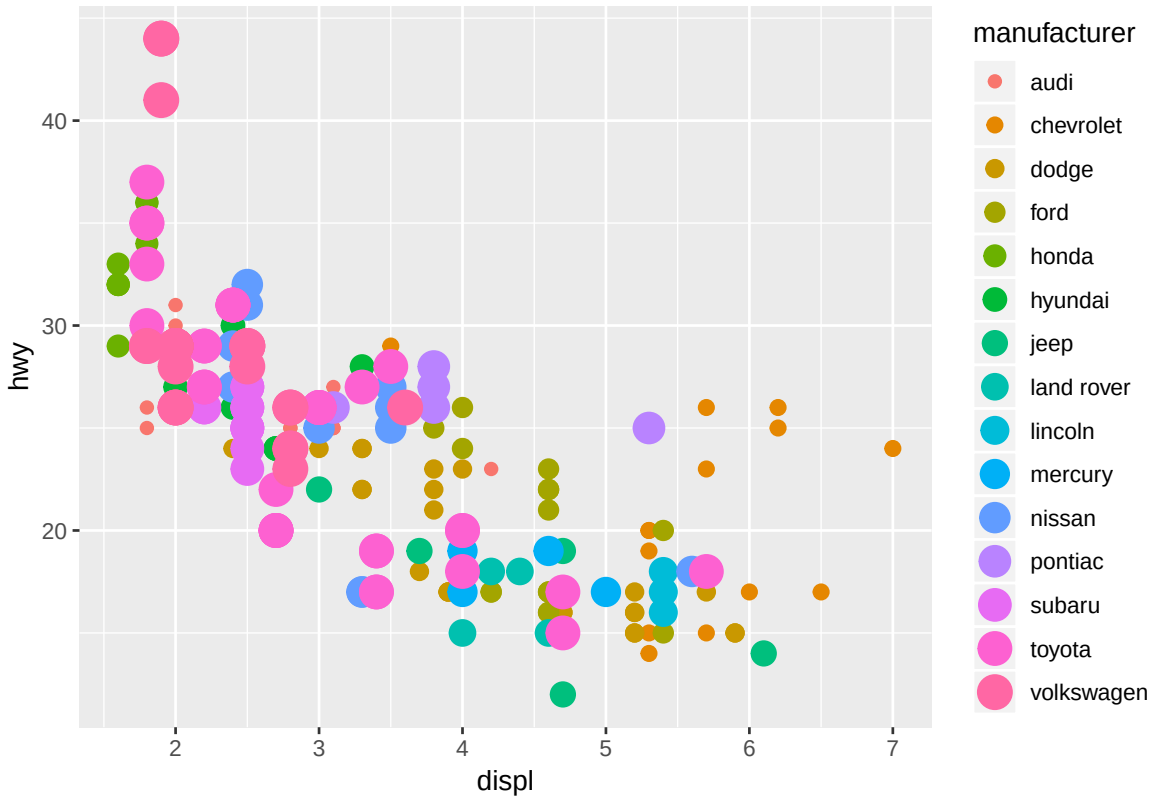
```
ggplot(mpg) +
  geom_point(aes(displ, hwy, size = cty))
```



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

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

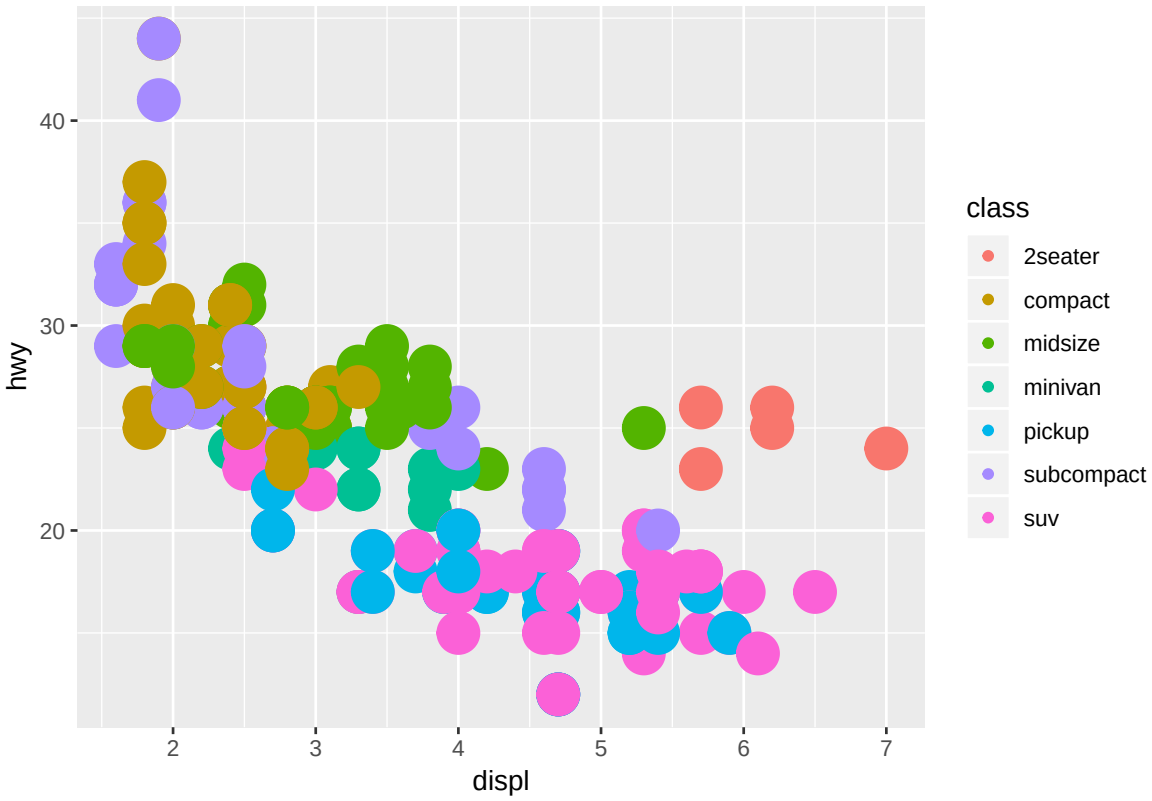
```
## Warning: Using size for a discrete variable is not advised.
```



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

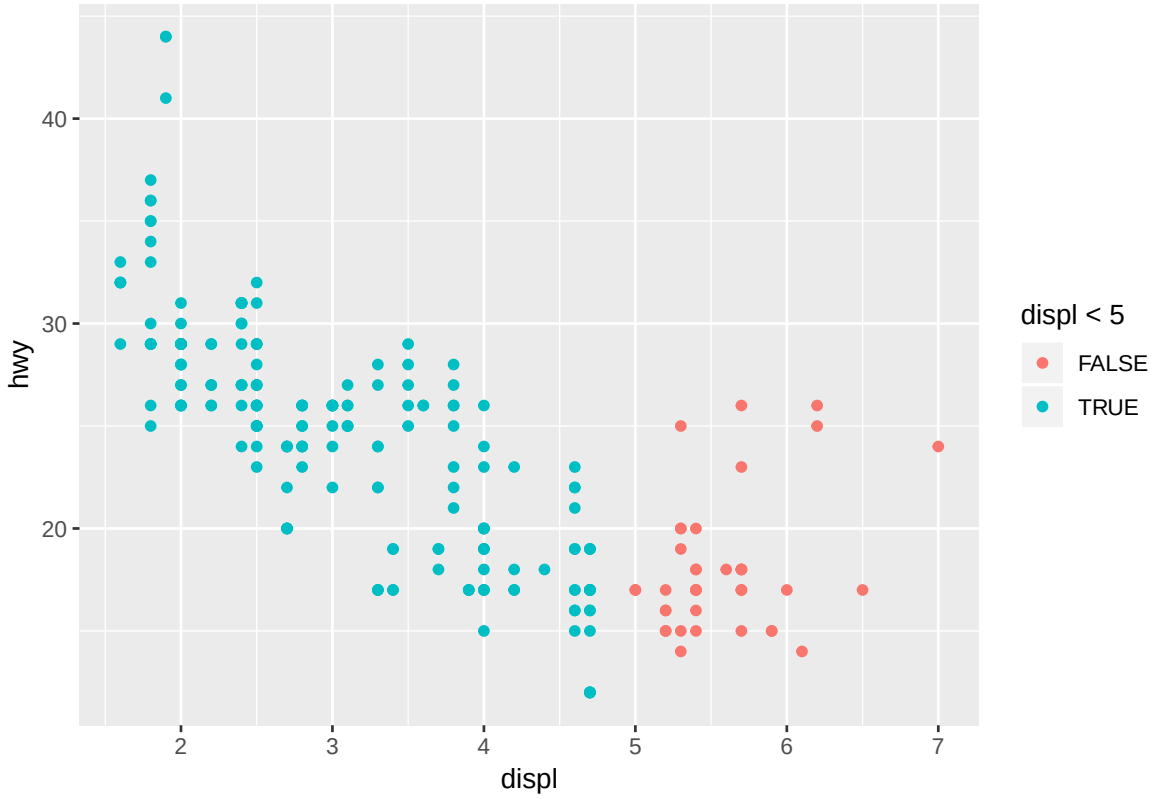
To modify the width of the border

```
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.

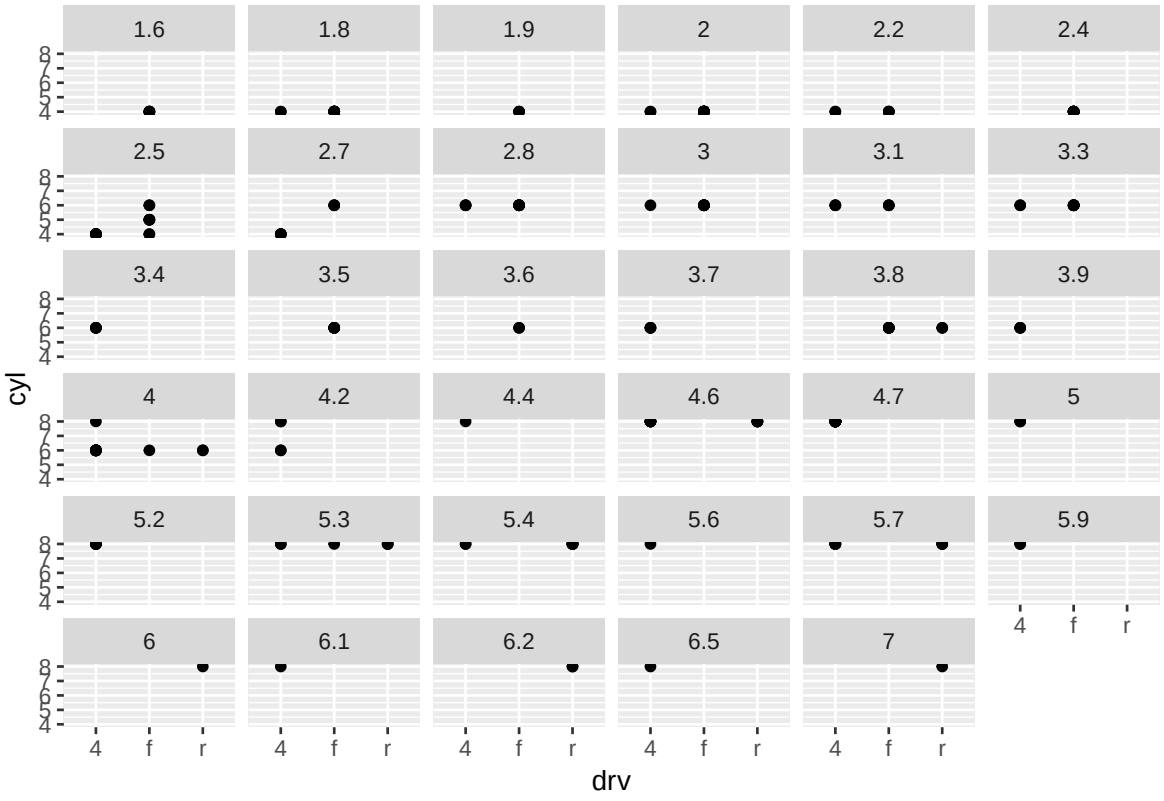
```
ggplot(mpg) +
  geom_point(aes(displ, hwy, color = displ < 5))
```



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)
```

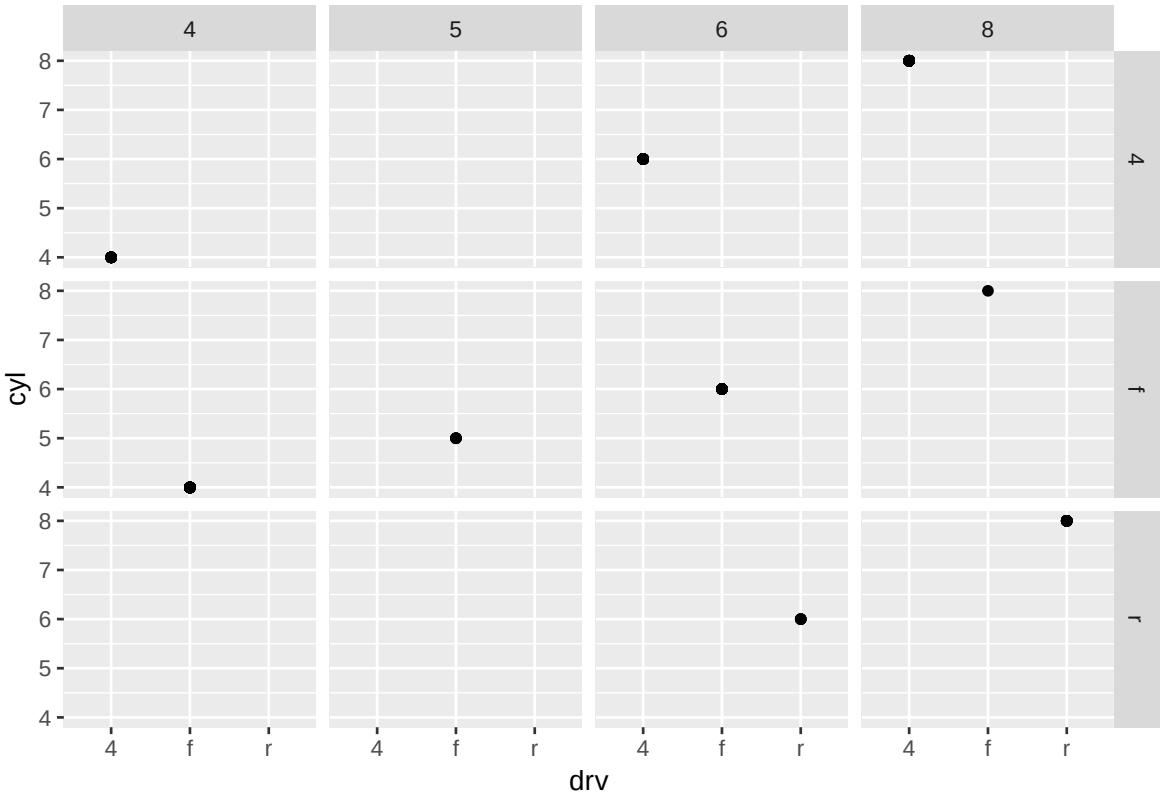


Your graph will not make much sense. R will try to draw a separate facet for each unique value of the continuous variable. If you have too many unique values, you may crash R.

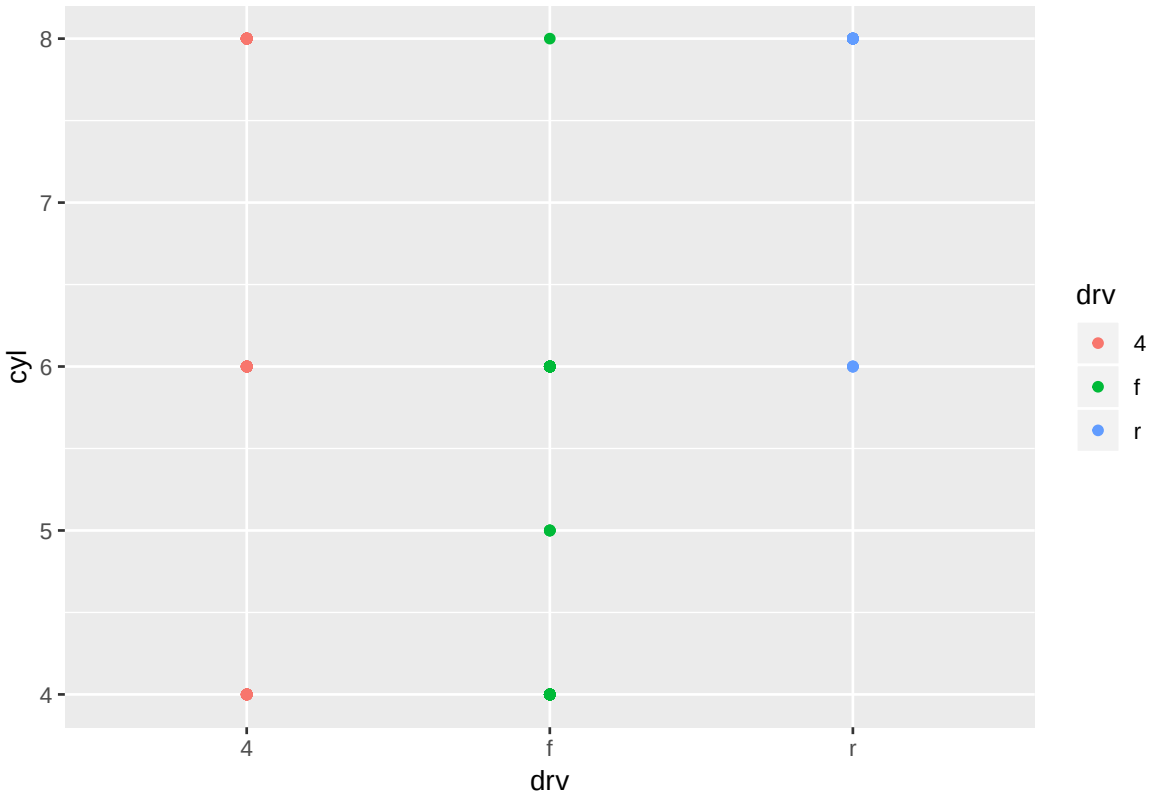
2. What do the empty cells in plot with `facet_grid(drv ~ cyl)` mean? How do they relate to this plot?

```
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))
```

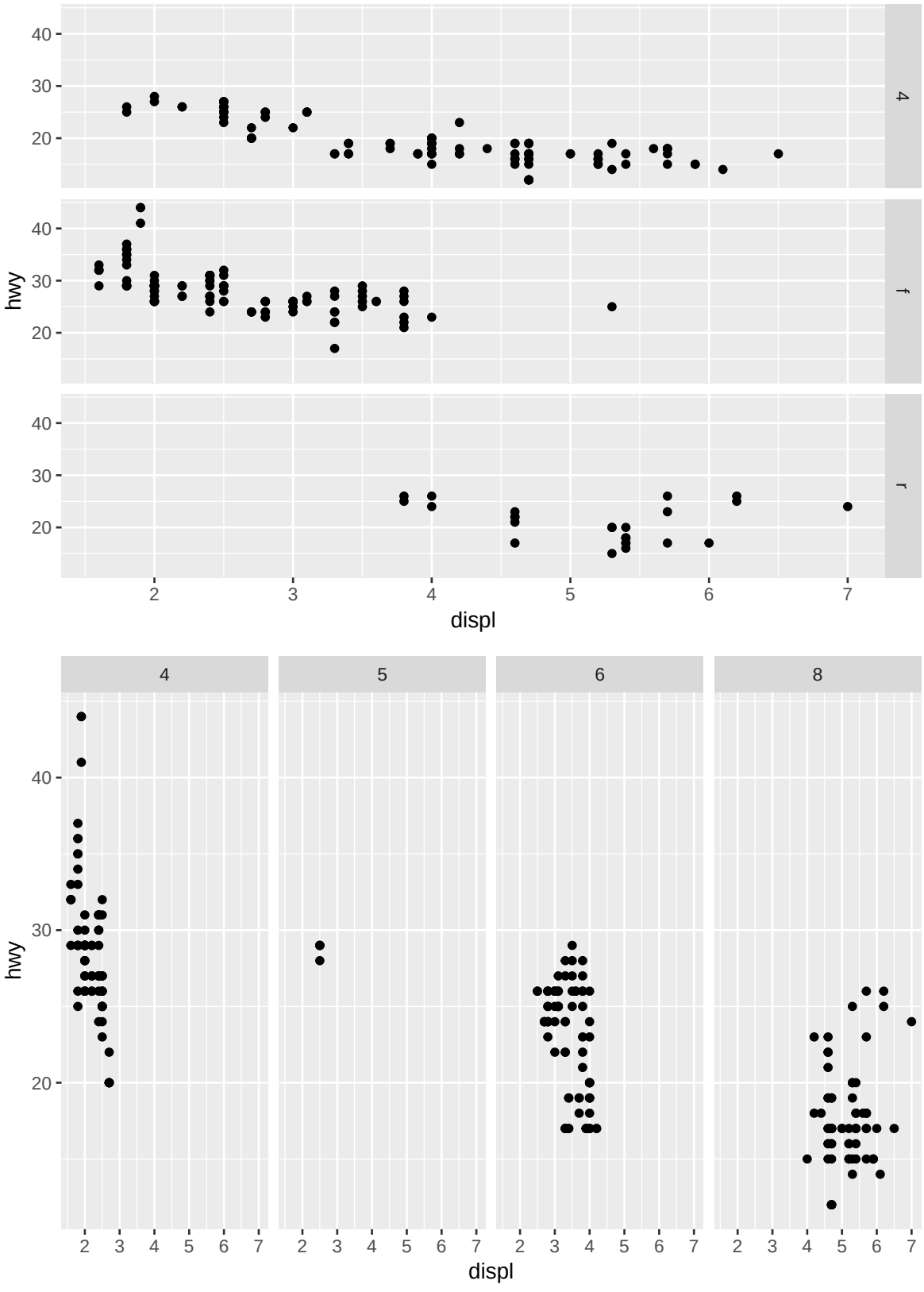


empty cells mean that there are no relation between drv and cyl. no 4 cylinders with rear wheel drive

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)
```



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.