Data Visualisation using R

Lecture-2

Suman Rakshit

Scool of EECMS, Curtin University

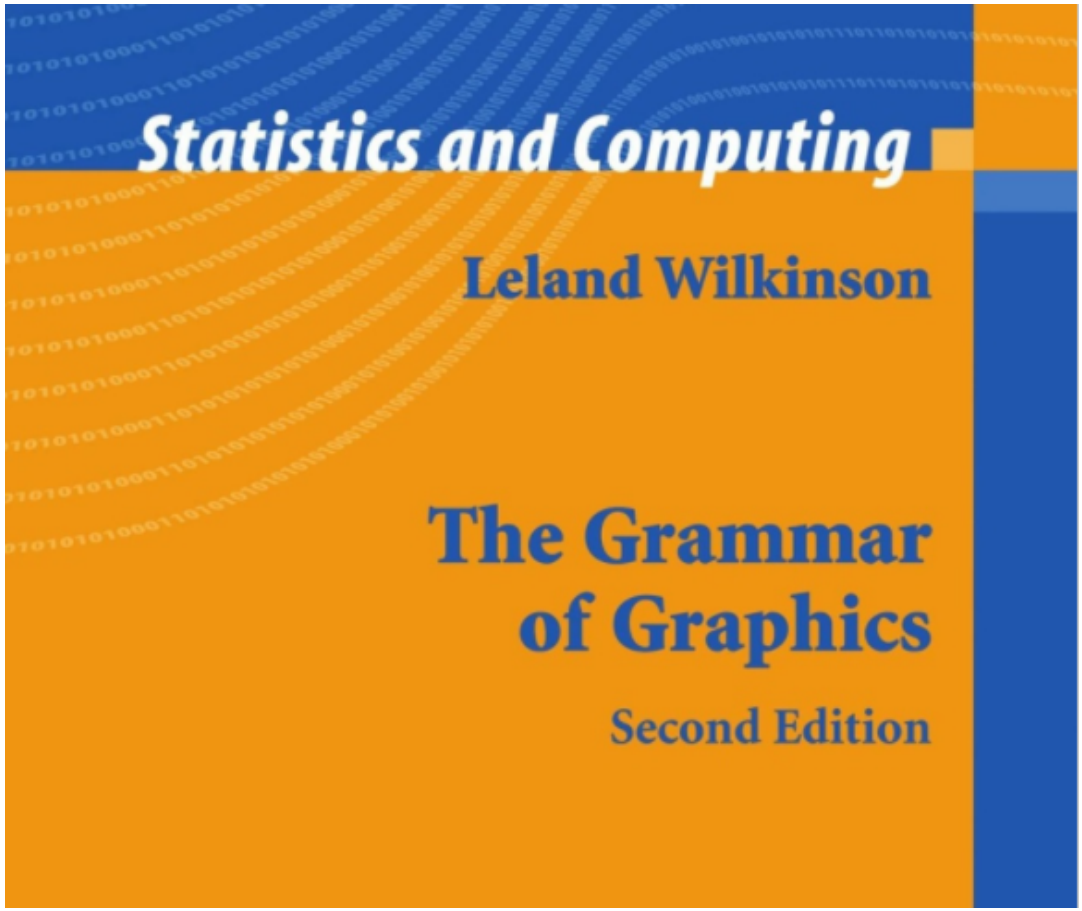
February 8, 2024

Outline

1. Introduction to Ggplot2
2. Key three elements of any Ggplot2 figure
3. Key element-1: Data
4. Key element-2: Aesthetics
5. Key element-3: Geometry
6. Summary

Why Ggplot2?

The first and the most important reason is that the R-package **Ggplot2** follows the *Grammar of Graphics*.



Grammar of English Language

- Grammar provides the key components of a language, using which you can write new sentences, essays, or best selling novels.
- Take the example of the famous sentence (pangram):

The quick brown fox jumps over the lazy dog.

- Every word has a clear grammatical definition:

The	quick brown	fox	jumps	over	the	lazy	dog
ARTICLE	ADJECTIVE	NOUN	VERB	PREPOSITION	ARTICLE	ADJECTIVE	NOUN

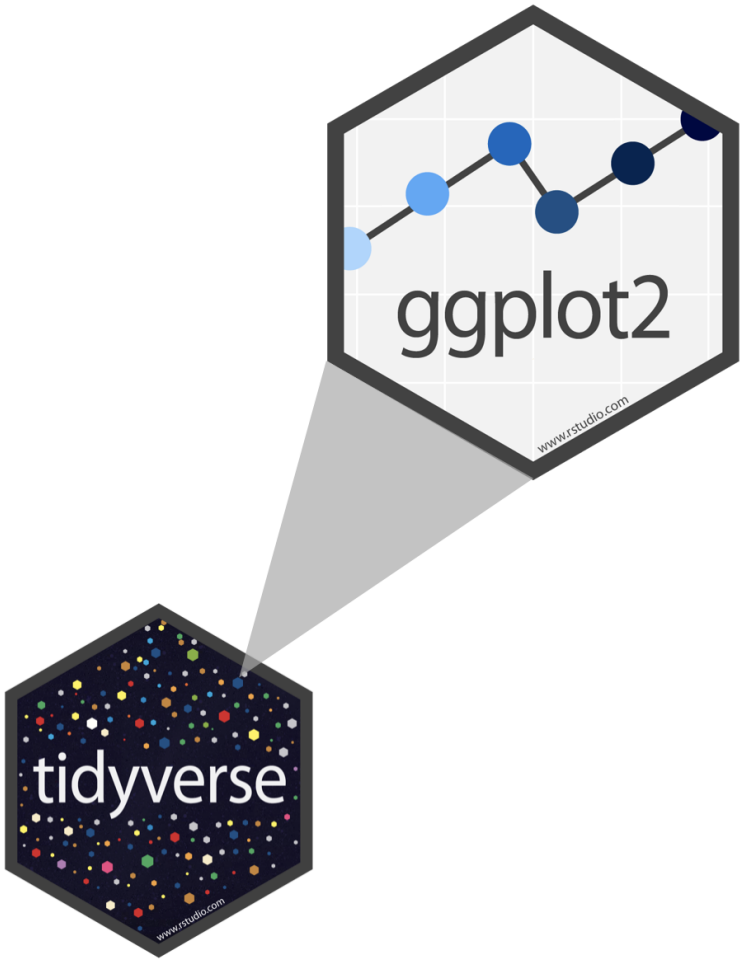
- The sentence conveys a specific message, and to change the message, we need to change its grammatical components.

Grammar of Graphics

The same framework holds true for creating visualisations in ggplot2 — the grammer of graphics provides all the individual components (like noun, verb, and adjective) and the rules to assemble them for producing meaningful visualisations.

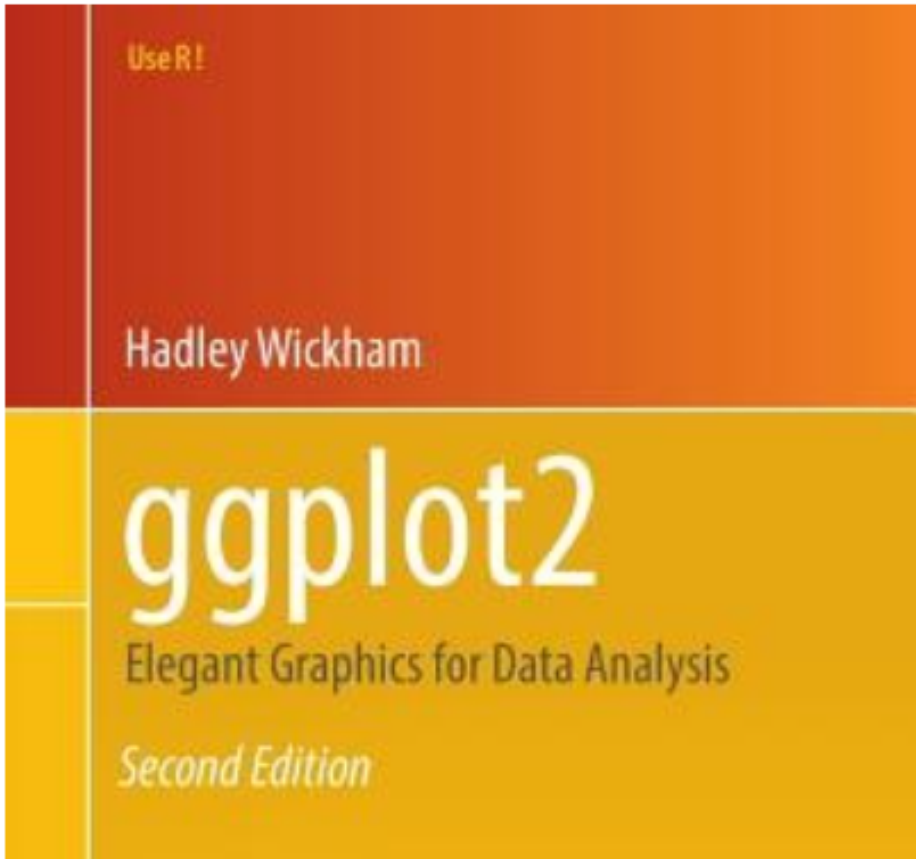
Ggplot2 is part of Tidyverse

The second reason is that Ggplot2 works well with other R-packages of Tidyverse — this saves time in writing R-code for visualisation.



Several great resources available

The third and last reason is that, great resources are available for ggplot2 — ask questions on stackoverflow: https://stackoverflow.com/questions/tagged/ggplot2



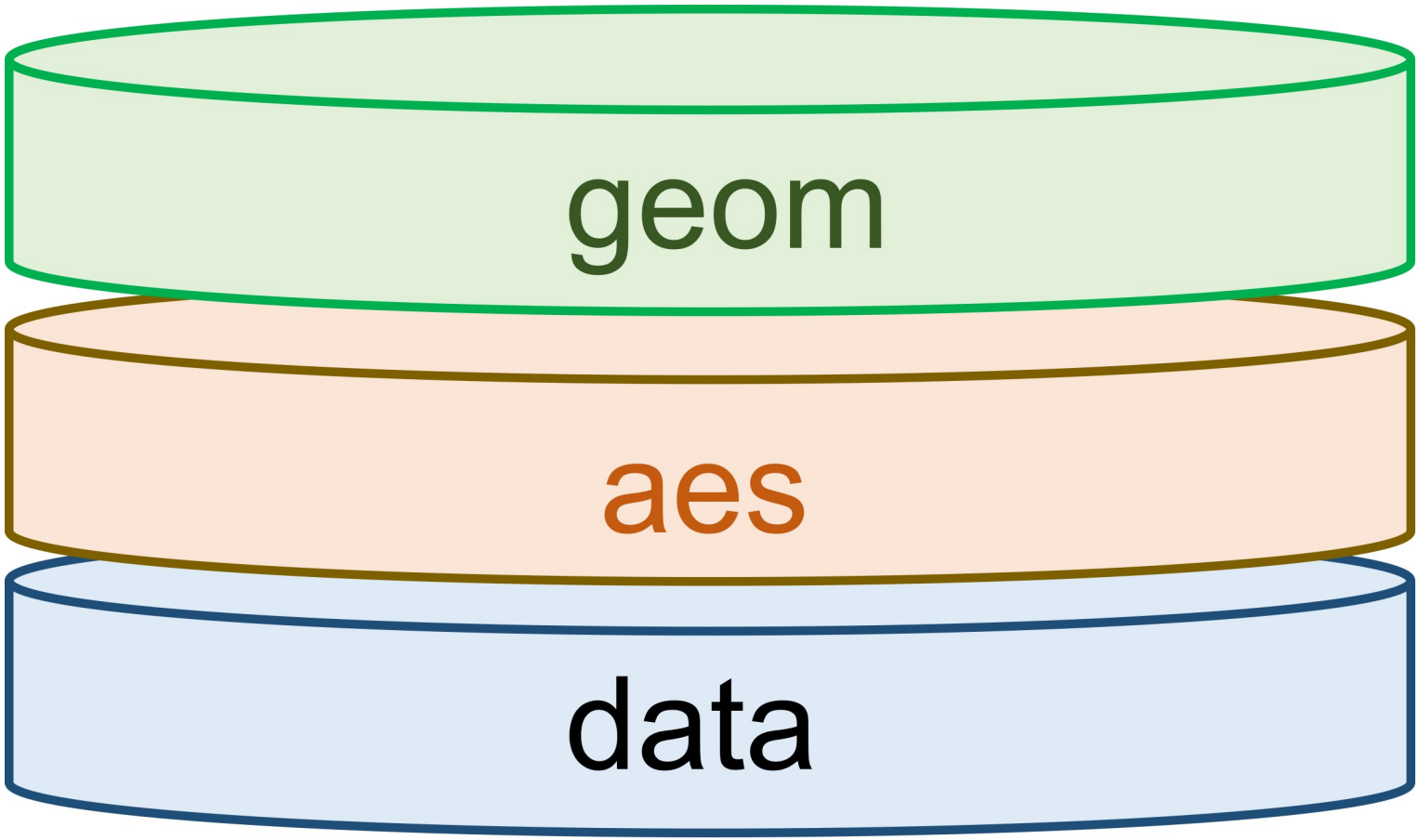
Three essential components/layers

The three essential components of any ggplot2 visualisation are: (i) `data`, (ii) `aes` (aesthetic mappings), and (iii) `geom_*` (geometry).

Ggplot2 components	Colloquial usage	R-Object	Example
data	Dataset containing all variable information	data.frame or tibble	ggplot(data = ...)
aes	Variables used for plotting and colouring the plot elements	R-function	ggplot(aes(x, y, colour))
geom_*	Type of plot or chart we like to construct	R-function	geom_point, geom_line, geom_bar, geom_boxplot

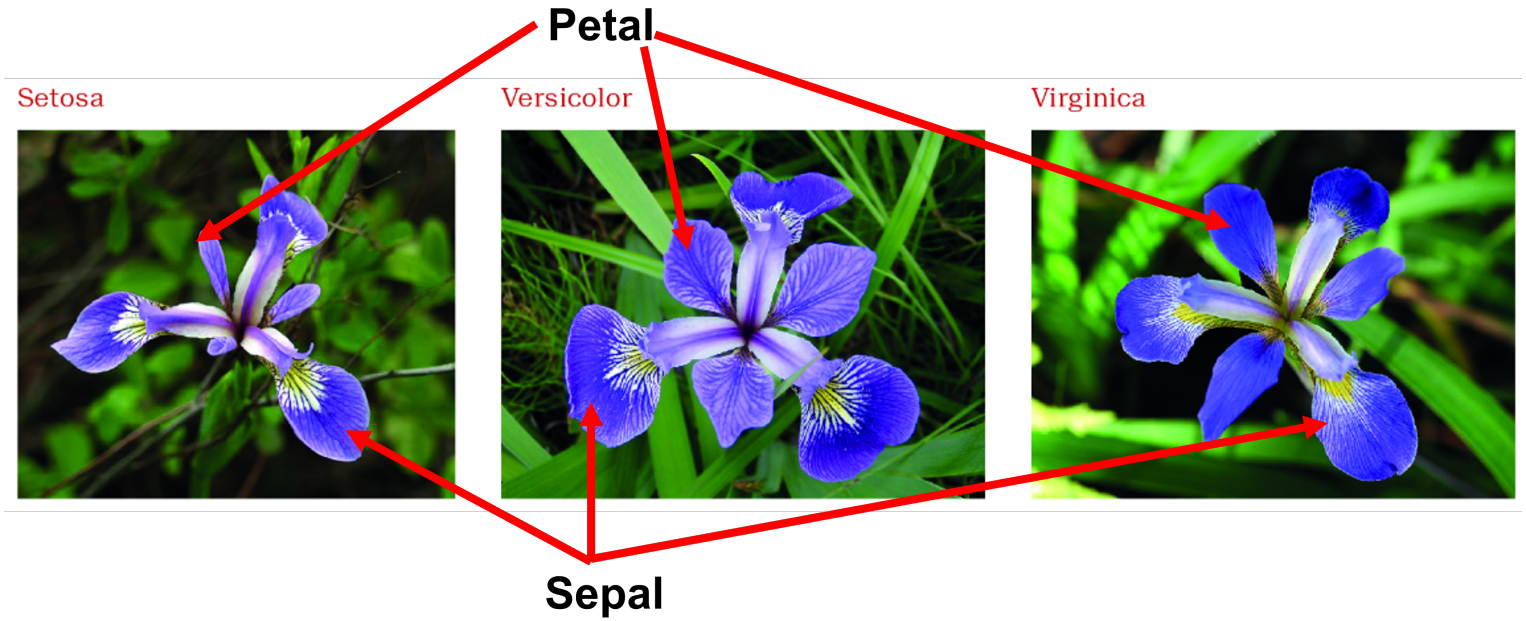
Combining three essential layers

We start with `data` layer, and add the `aes` layer on top of that, and finally, we add the `geom` layer on top.



Example: Iris dataset

We can use the three main components of ggplot2 to create a very quick exploratory plot using the iris dataset.



We like to investigate the relationship between Sepal.Length and Sepal.Width.

Iris data: Sepal.Length versus Sepal.Width

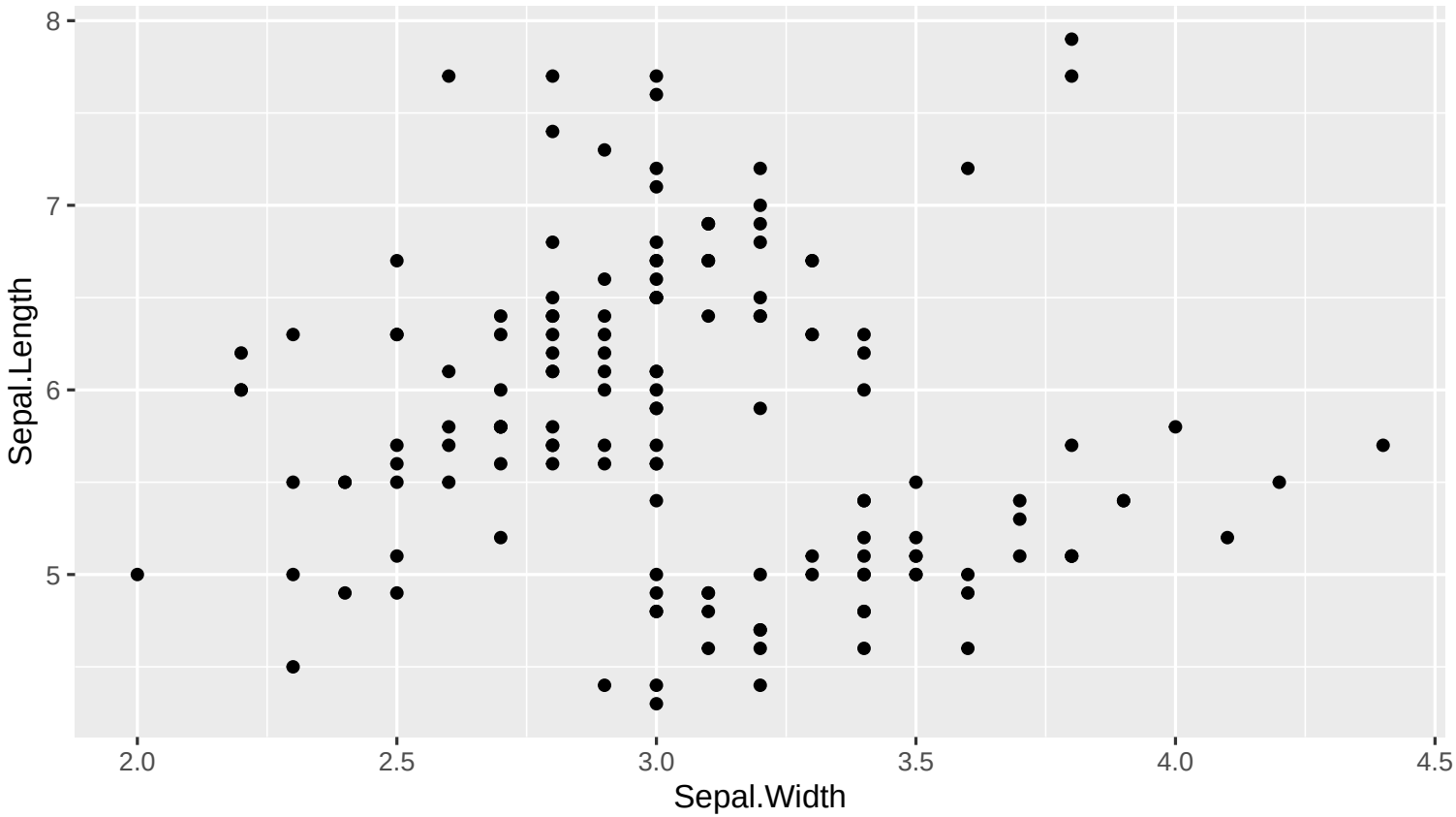


Figure 1: Barebone scatter plot of Sepal.Length versus Sepal.Width using three main components of ggplot2.

Can you guess the code?

- What is the data component?
- What are the aesthetic mappings?
- What is the geometry component?

Ggplot2 code of scatter plot

```
# Sepal.Length versus Sepal.Width
ggplot(data = iris,
       mapping=aes(x = Sepal.Width,
                   y = Sepal.Length)) +
       geom_point()
```

```
# Shorter version of the same code
iris %>%
ggplot(aes(Sepal.Width, Sepal.Length)) +
geom_point()
```

Let's see another example: mtcars

```
> str(mtcars)
'data.frame':   32 obs. of  11 variables:
 $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3
 $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
 $ disp: num  160 160 108 258 360 ...
 $ hp  : num  110 110 93 110 175 105 245 62
 $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.
 $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
 $ qsec: num  16.5 17 18.6 19.4 17 ...
 $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
 $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
 $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
 $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
```

Figure 2: Aim is to investigate the relationship between mpg (miles per gallon) and cyl (number of cylinders).

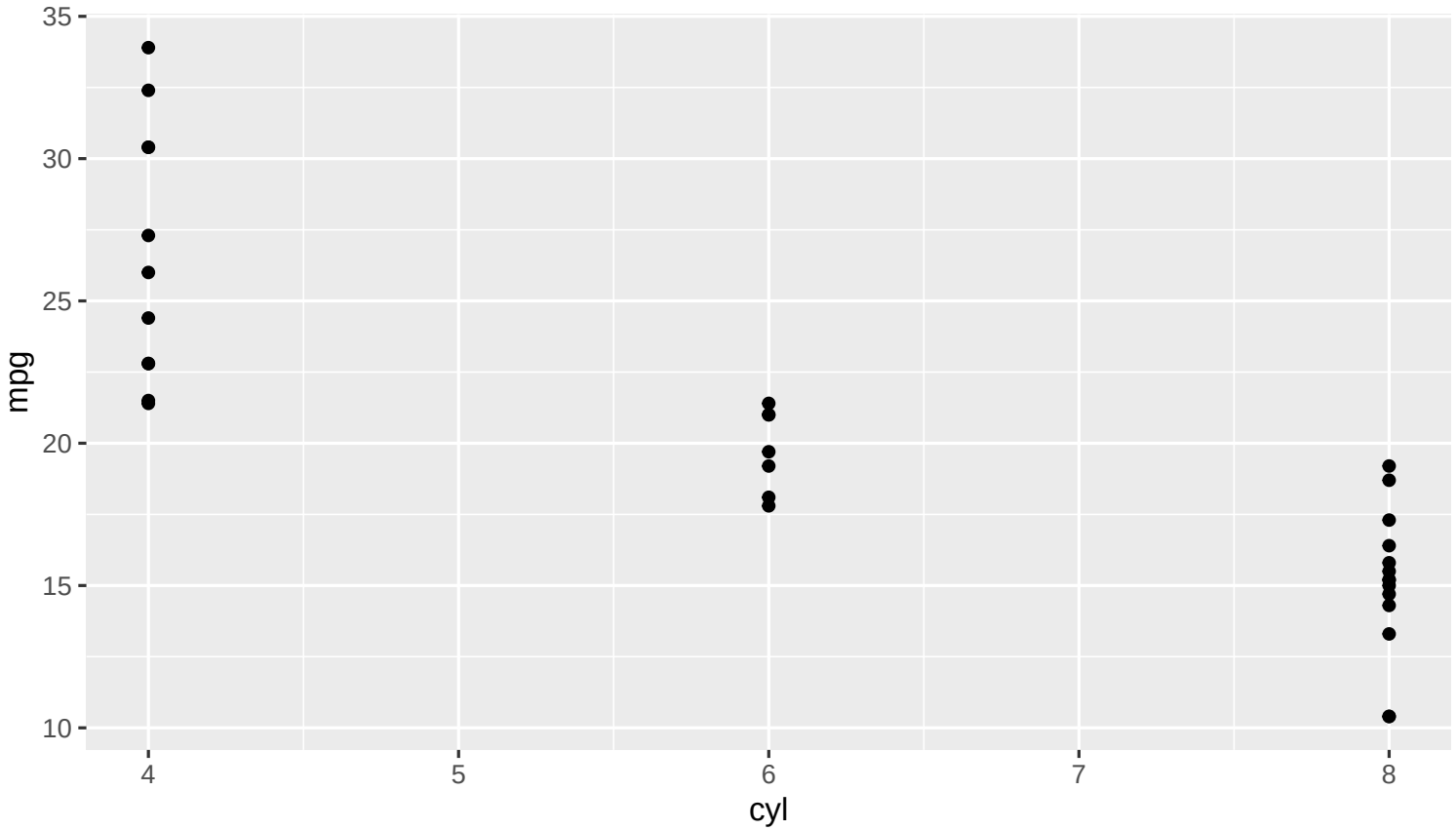
RCode — scatter plot of mpg against cyl

we start with the `mtcars` data layer. We map cyl to x-axis aesthetics and mpg to y-axis aesthetics. For obtaining a scatter plot, we add the `geom_point()` layer on top.

```
# Scatter plot: mpg versus cyl
mtcars %>%
ggplot(aes(x= cyl, y = mpg)) +
geom_point()
```

Scatter plot: mpg versus cyl

Can you spot any problem with this plot?



cyl should be a categorical variable

```
mtcars %>%
ggplot(aes(as.character(cyl), mpg)) +
geom_point()
```

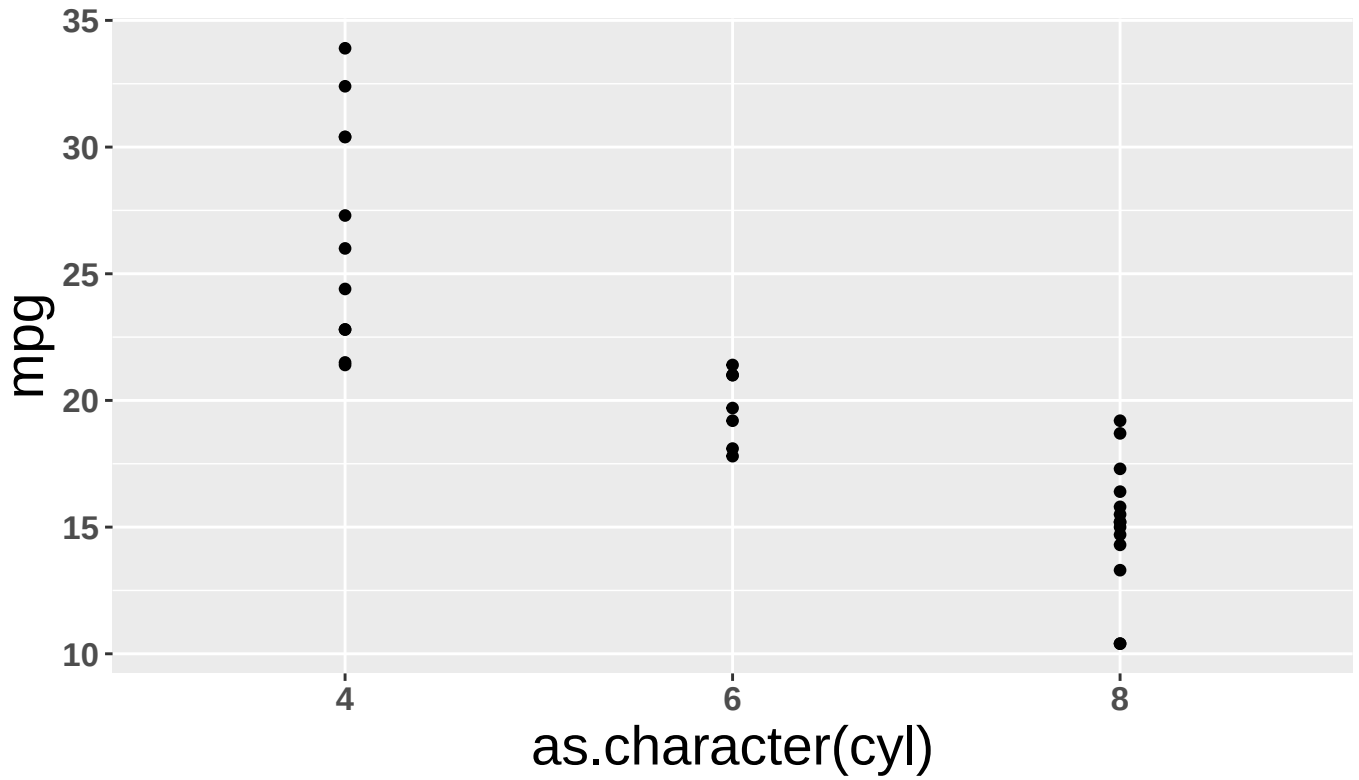
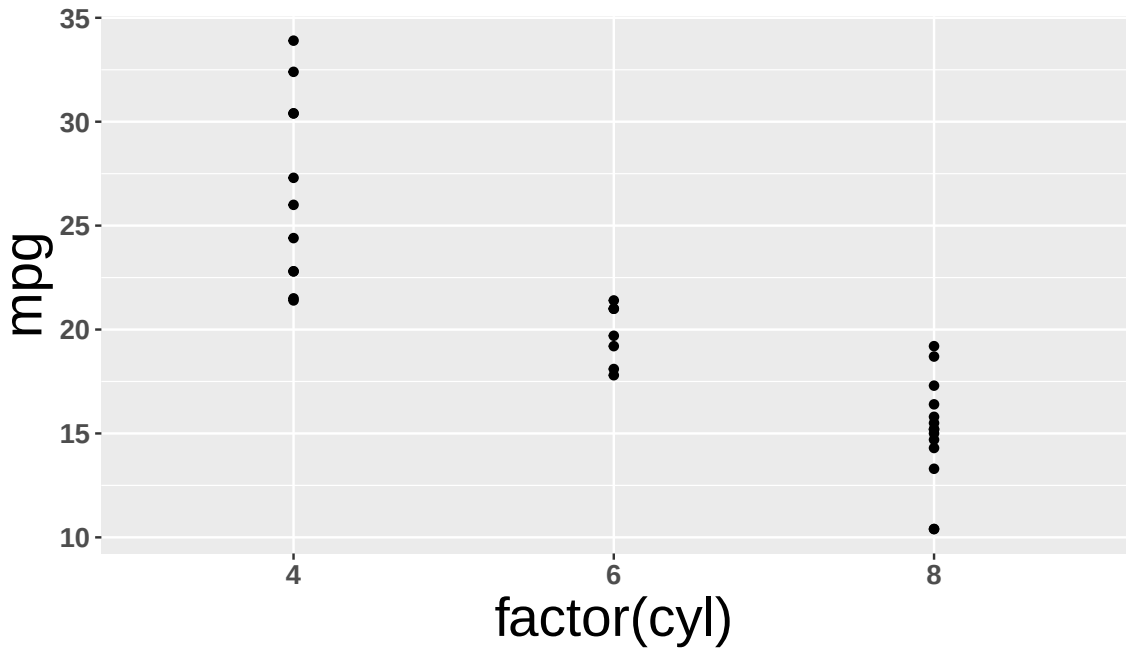


Figure 3: cyl levels are ordered alphabetically.

Use `factor()` to convert `cyl`

Use `factor()` rather than `as.character()` to turn `cyl` into a categorical variable

```
mtcars %>%
ggplot(aes(factor(cyl), mpg)) +
geom_point()
```

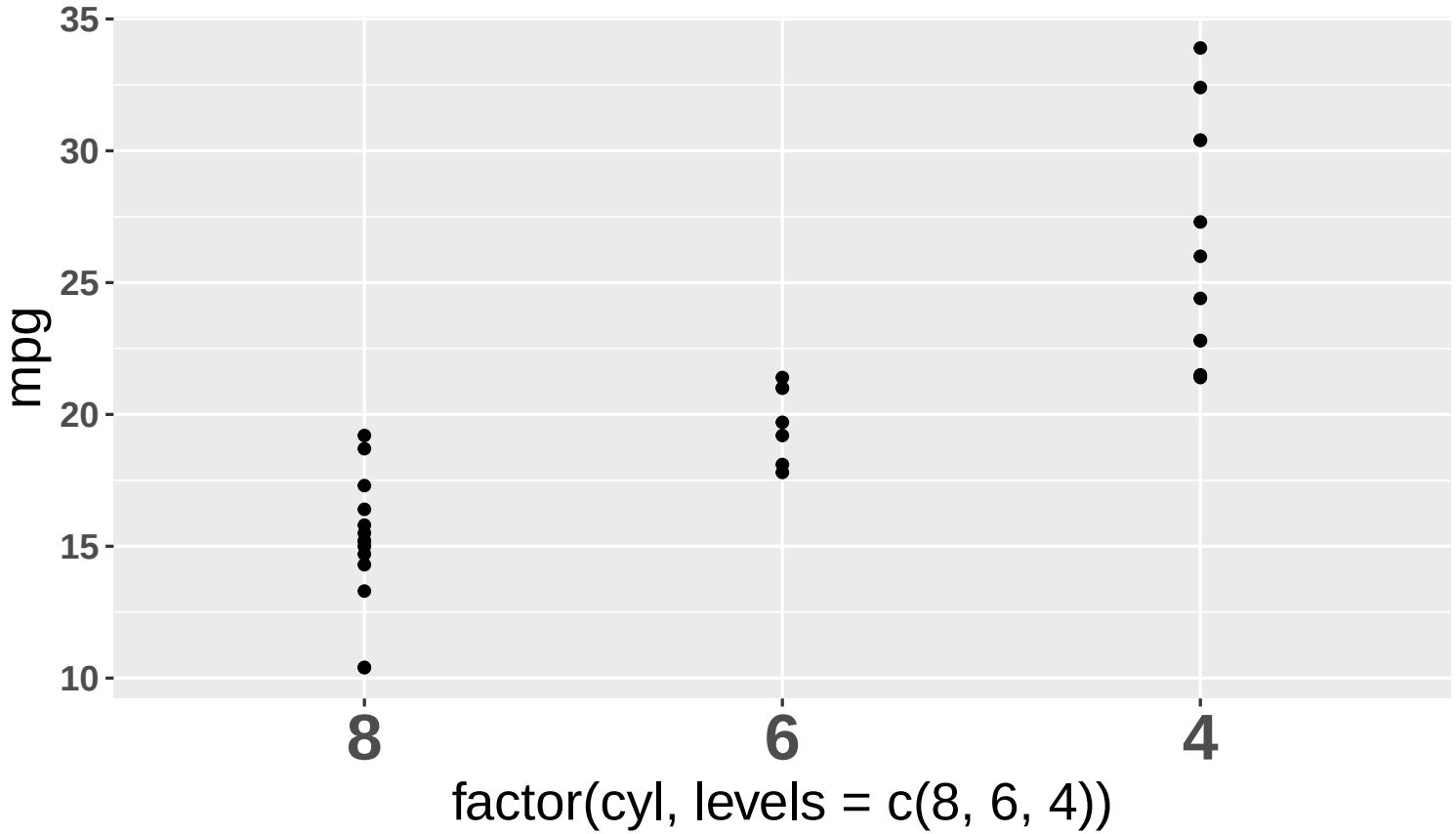


Advantage of using `factor`?

What is the advantage of using `factor()` instead of `as.character()`?

Ordering levels of categorical variables

You can change the default ordering of the levels of a categorical variable using `factor()`.



More aes mapping: color, size, and fill

- We learnt about the aesthetic mappings onto the x and y axes.
- The next three most useful aesthetic mappings are color, size, and fill.

aes	Continuous variable	Categorical variable
color	✓	✓
size	✓	✗
shape	✗	✓

Map categorical `cyl` onto `color` aesthetic

Aim: Use `mtcars data layer` to investigate how `disp` (engine displacement measured in cubic inches) influences `mpg` (miles per gallon) for different `cyl` (number of cylinders) values.

RCode: map categorical cyl onto color

Note the color aesthetic inside the aes function.

```
# Scatter plot: mpg versus disp
mtcars %>%
ggplot(aes(x= disp,
           y = mpg,
       color = factor(cyl))) +
geom_point()
```

Mapping categorical `cyl` onto `color`

Legend title reflects the mapping of `cyl` onto `color`.

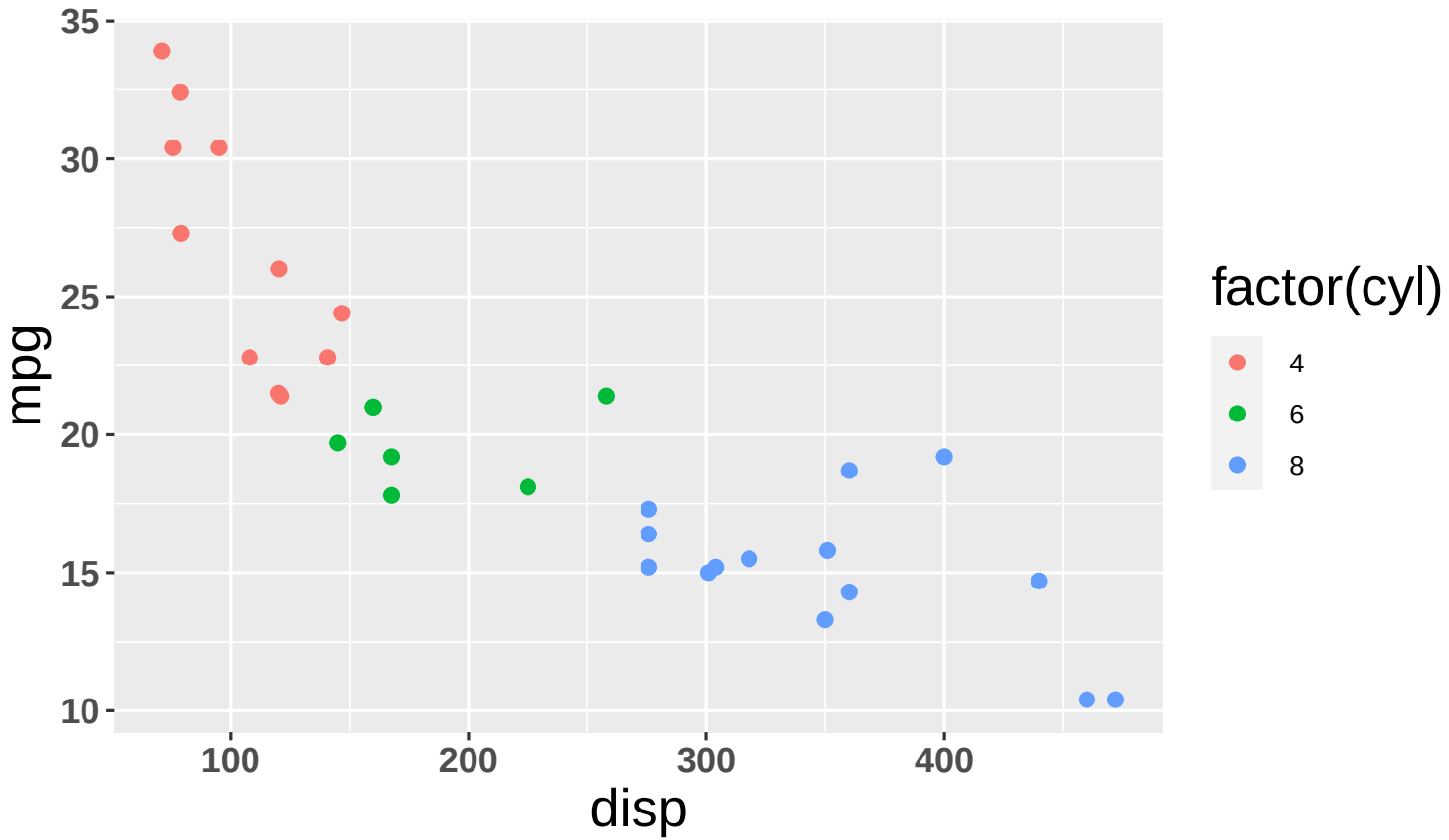


Figure 4: Cars with larger `cyl` values are less fuel-efficient (low `mpg`) but exhibit higher engine displacement (`disp`).

Mapping continuous `disp` onto `color`

Gradient color scheme reflects the mapping of the continuous `disp` onto `color`.

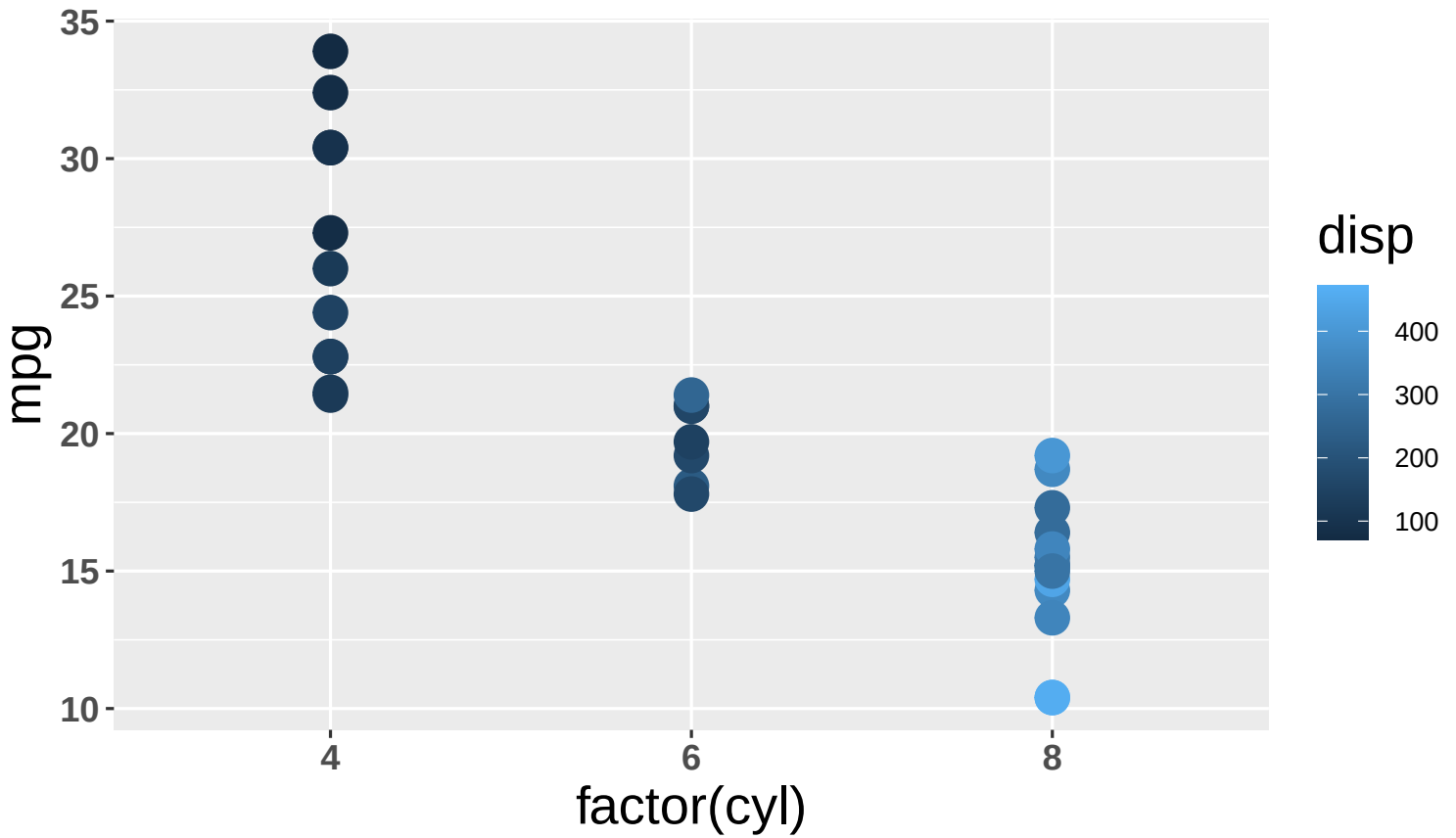


Figure 5: The value of `disp` gradually increases as the shade of blue gets from darker to lighter.

Curtin University

Example data: mpg from ggplot2

```
> str(mpg)
tibble[,11] [234 x 11] (S3: tbl_df/tbl/data.frame)
 $ manufacturer: chr [1:234] "audi" "audi" "audi" "audi" ..
 $ model       : chr [1:234] "a4" "a4" "a4" "a4" ...
 $ displ       : num [1:234] 1.8 1.8 2 2 2.8 2.8 3.1 1.8 1.
 $ year        : int [1:234] 1999 1999 2008 2008 1999 1999
 $ cyl         : int [1:234] 4 4 4 4 6 6 6 4 4 4 ...
 $ trans       : chr [1:234] "auto(l5)" "manual(m5)" "manua
 $ drv         : chr [1:234] "f" "f" "f" "f" ...
 $ cty         : int [1:234] 18 21 20 21 16 18 18 18 16 20
 $ hwy         : int [1:234] 29 29 31 30 26 26 27 26 25 28
 $ fl          : chr [1:234] "p" "p" "p" "p" ...
 $ class       : chr [1:234] "compact" "compact" "compact"
```

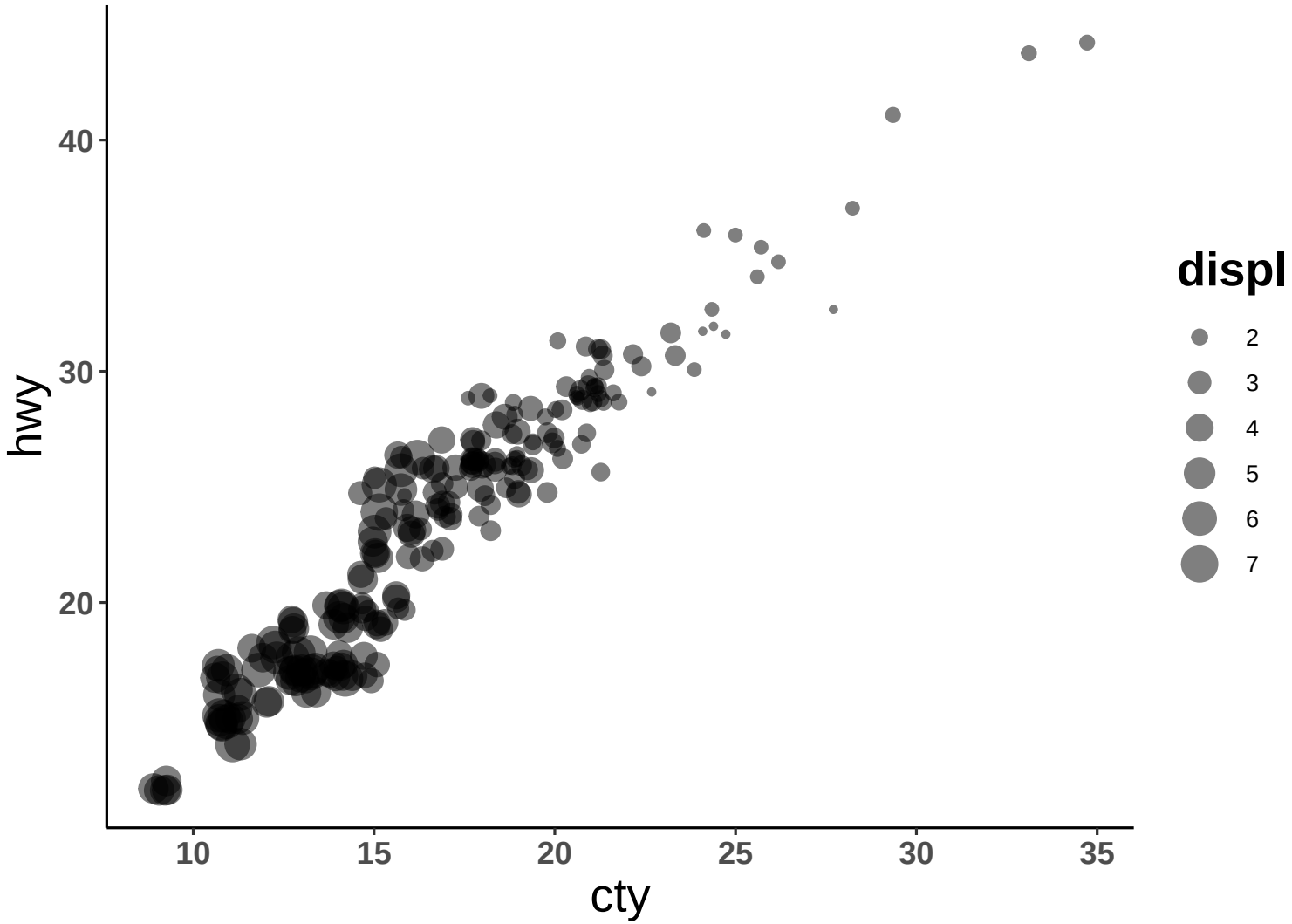
Figure 6: Aim is to investigate *highway miles per gallon* (hwy) as a function of *city miles per gallon* (cty).

RCode: mapping continuous displ to size

Note the size aesthetic inside the aes() function, and also the new geom layer.

```
# Scatter plot: hwy versus cty
mpg %>%
ggplot(aes(x = cty,
           y = hwy,
        size = displ)) +
geom_jitter()
```

Scatter plot: displ mapped to size



Avoid mapping factors onto size

```
>    mpg %>%
+    ggplot(aes(x = cty, y=hwy, size = factor(drv))) +
+    geom_jitter()
Warning message:
Using size for a discrete variable is not advised.
```

Example: map factors onto shape

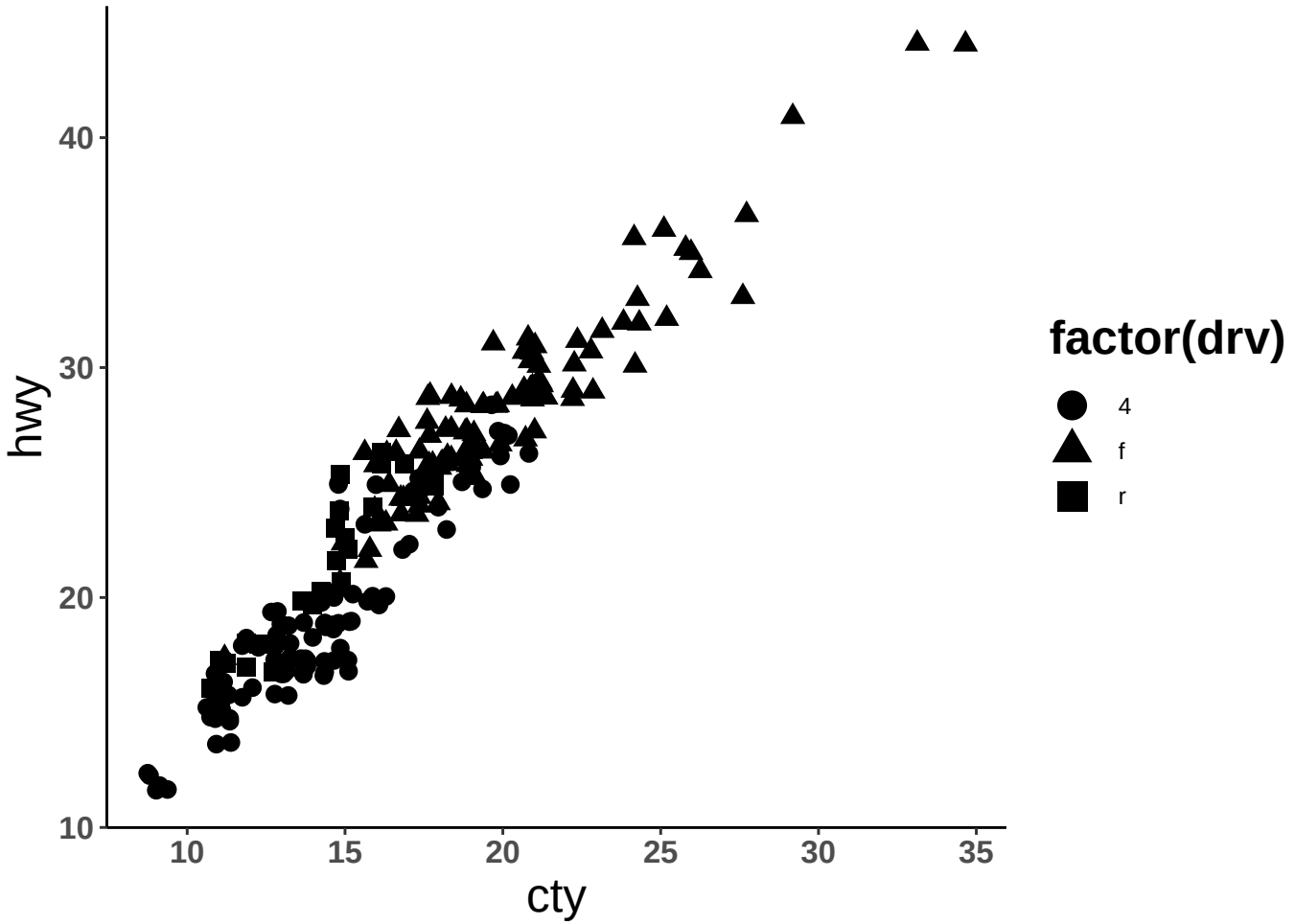
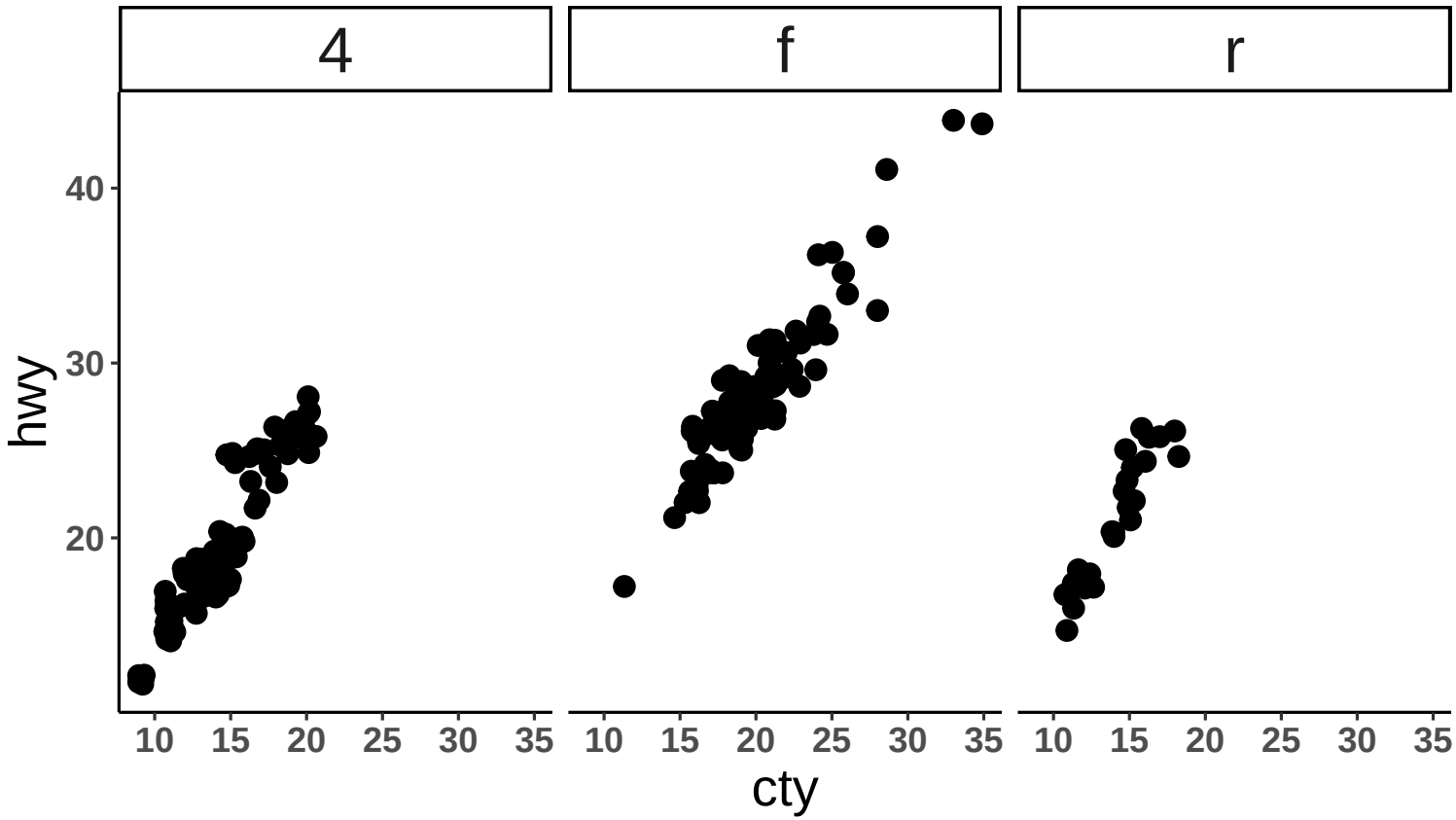


Figure 7: Hard to distinguish between four wheel (4) and rear wheel (r) drives.

Faceting — conditional plots

The best way to investigate the relationship between `hwy` and `cty` for the three driving categories (`drv`) is by using the `facet_wrap()` function.



Curtin University

RCode: facet_wrap() example

```
# Scatter plot: hwy versus cty
mpg %>%
ggplot(aes(x = cty, y = hwy)) +
geom_jitter() +
facet_wrap(~ factor(drv))
```

facet_wrap() versus facet_grid()

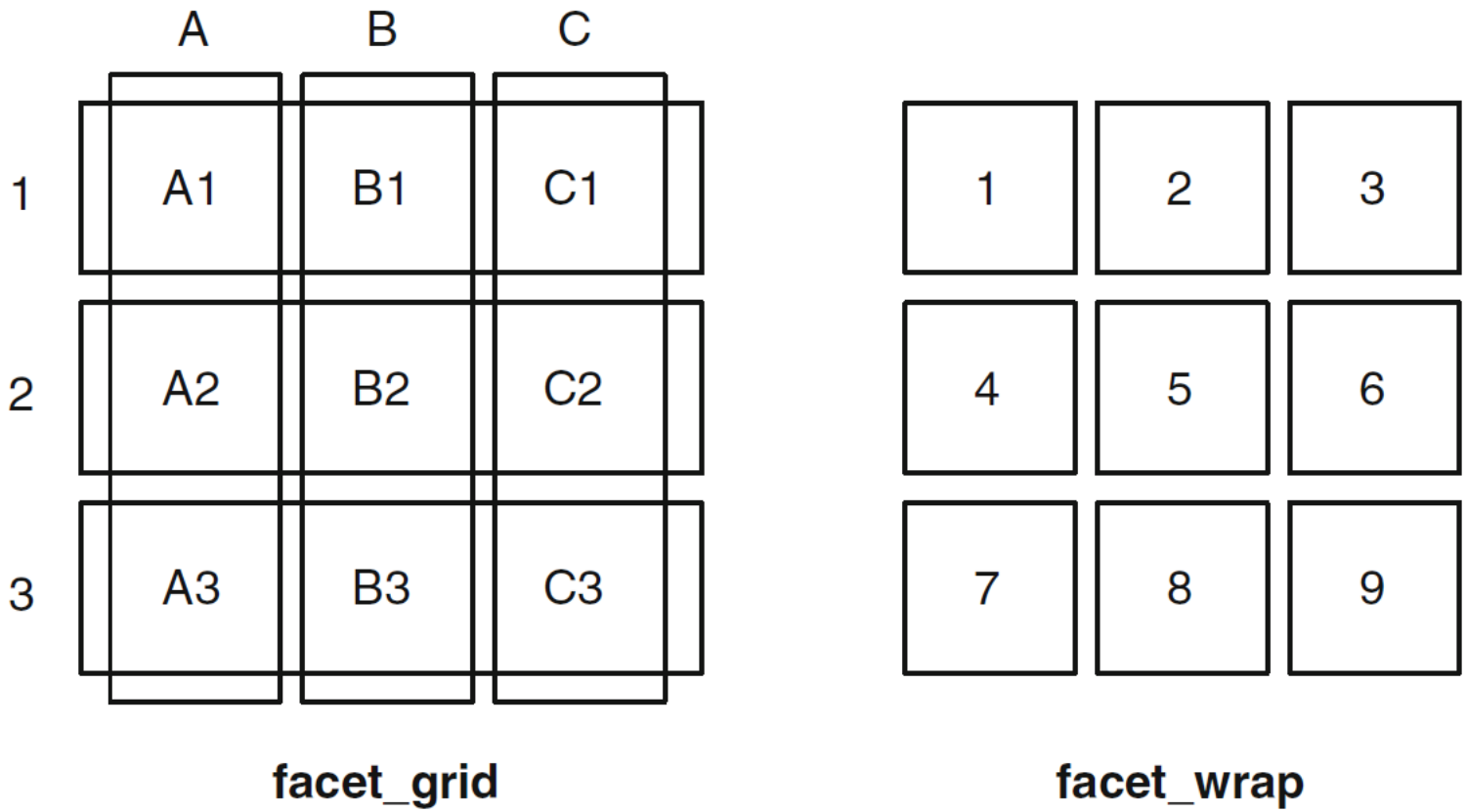


Figure 8: `facet_grid()` is appropriate for including two factors into faceting — from both row and column directions.

Faceting using `facet_grid()`

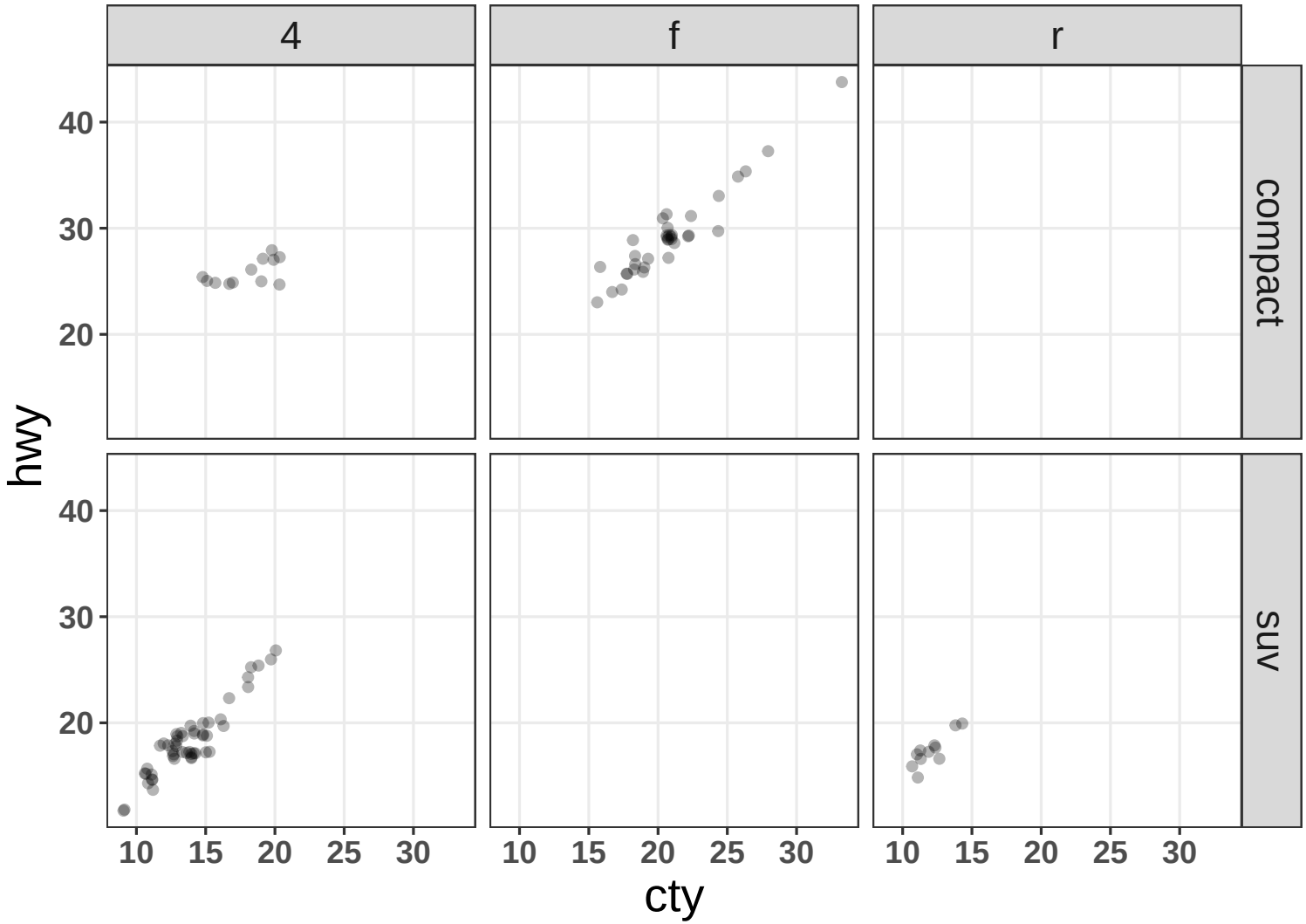


Figure 9: `hwy` vs `cty` vs `drv` vs `class`.

RCode: facet_grid() example

```
# Scatter plot: hwy versus cty
mpg %>%
ggplot(aes(x = cty, y = hwy)) +
geom_jitter() +
facet_grid(class ~ drv)
```

`color` versus `fill` aesthetics

- Often there is a confusion between the `color` aesthetic and the `fill` aesthetic.
- `color` aesthetic typically changes the color of the outline of a plotting object.
- `fill` aesthetic changes the color of the body of a plotting object.

color versus fill aesthetics

- Often there is a confusion between the color aesthetic and the fill aesthetic.
- color aesthetic typically changes the color of the outline of a plotting object.
- fill aesthetic changes the color of the body of a plotting object.

`color` versus `fill` aesthetics

- Often there is a confusion between the `color` aesthetic and the `fill` aesthetic.
- `color` aesthetic typically changes the color of the outline of a plotting object.
- `fill` aesthetic changes the color of the body of a plotting object.

Example: `color` versus `fill` aesthetics

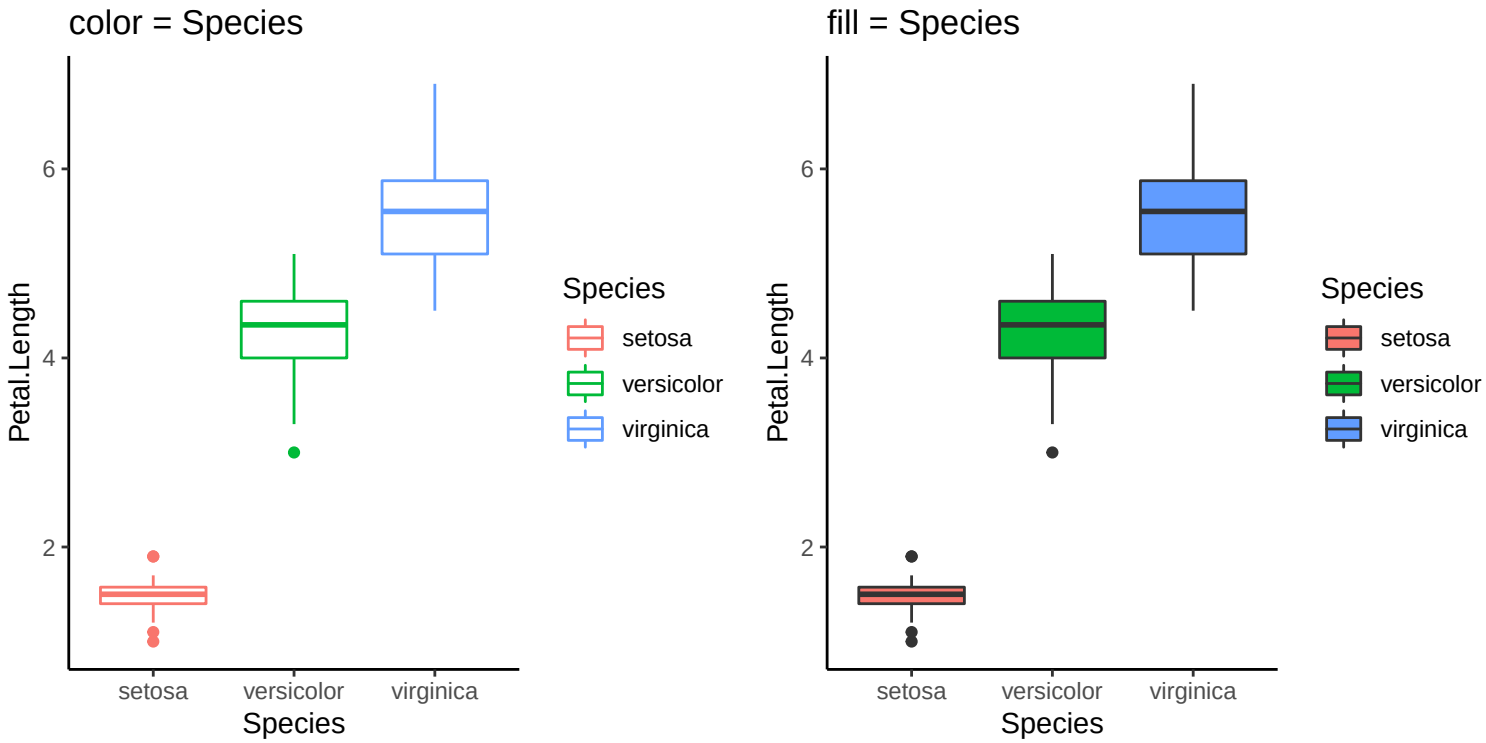


Figure 10: Boxplots of Petal.Length for three iris species using `color = Species` (*left*) and `fill = Species` (*right*).

RCode: color versus fill aesthetics

```
# Basic Boxplot: Petal.Length
   vs Species
base_plt <- ggplot(iris, aes(x
   = Species, y = Petal.Length))

# color = Species
base_plt + geom_boxplot(aes(
   color = Species))

# fill = Species
base_plt + geom_boxplot(aes(
   fill = Species))
```

Three key points from the last slide

- Any ggplot can be saved as an R object/variable.
- ggplot objects can be extended using the + operator.
- We can call `aes()` inside the geom layer as well.

Example: `color` versus `fill` aesthetics

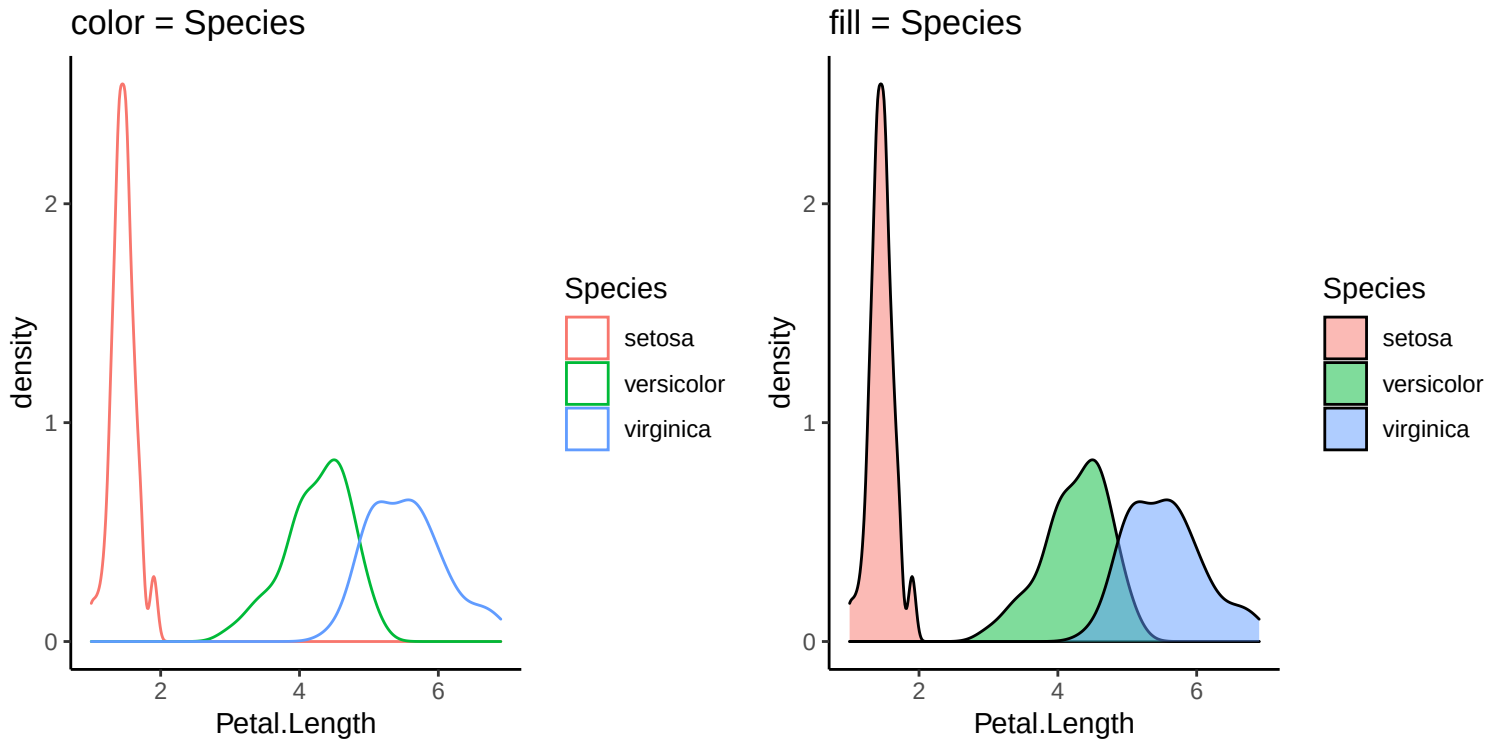


Figure 11: Density plots of Petal.Length for three iris species using `color = Species` (*left*) and `fill = Species` (*right*).

RCode: color vs fill in geom_density

```
# Petal.Length densities
base_plt <- ggplot(iris, aes(x
   = Petal.Length))

# color = Species
base_plt + geom_density(aes(
   color = Species))

# fill = Species
base_plt + geom_density(aes(
   fill = Species))
```

color vs fill in scatter plot

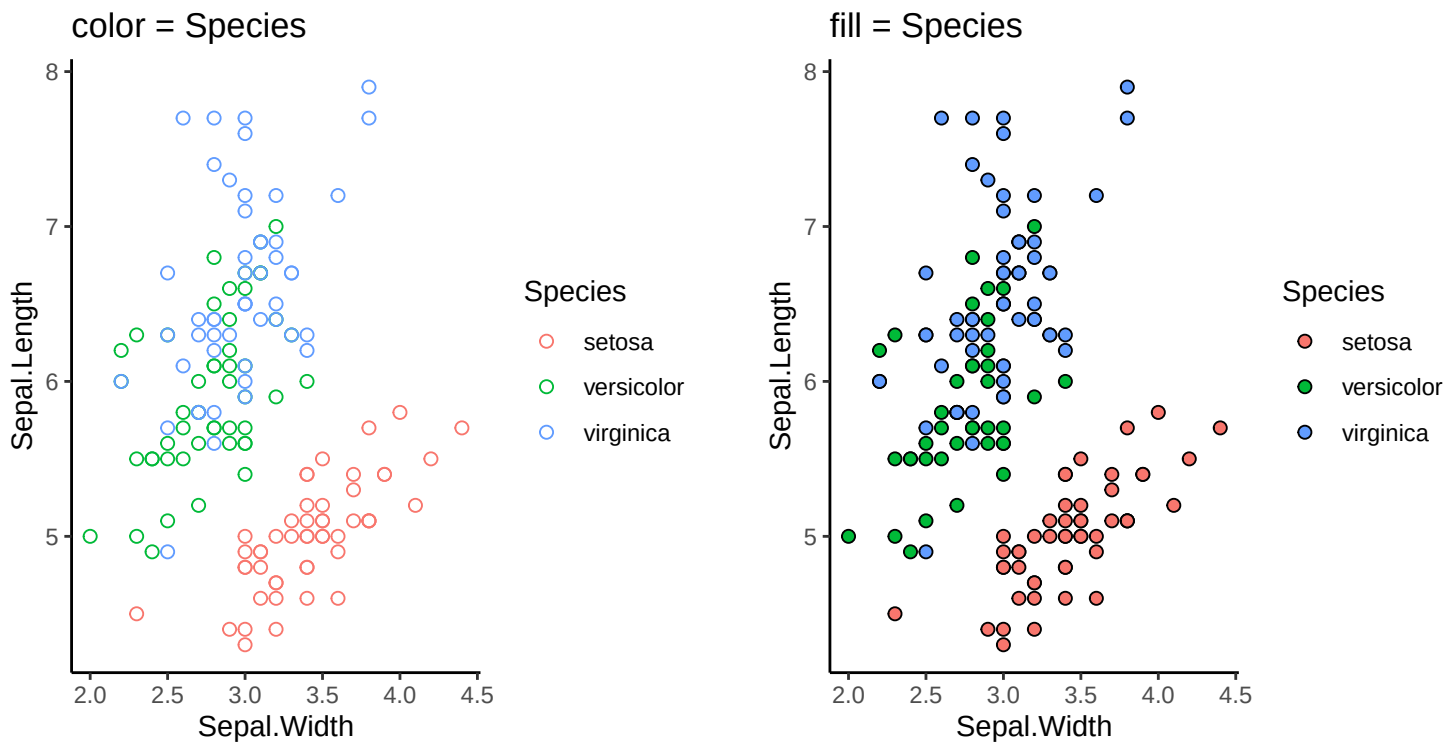


Figure 12: Scatter plots of Sepal.Length vs Sepal.Width using `color = Species` (*left*) and `fill = Species` (*right*); it is important to use a point `shape` that has both `color` and `fill` attributes.

RCode: color vs fill in geom_point

```
# Sepal.Length vs Petal.Length
base_plt <- ggplot(iris, aes(x = Petal.
    Length, y = Sepal.Length))

# color = Species
base_plt + geom_point(aes(color =
    Species), shape = 21)

# fill = Species
base_plt + geom_point(aes(fill = Species
    ), shape = 21)
```

Note `shape = 21` outside the `aes()` function.

Aesthetics vs Attributes

- It is common to confuse between aesthetics mapping and visible attributes.
- The confusion is often due to the fact that all visual aesthetics can be used as fixed visible attributes.
- Fixed attributes specify how all plotting components should look, e.g., colors of all points, size of all points, or transparency of all points. This is different than aesthetic mapping.
- To specify a fixed attribute, call it outside the `aes()` function.

Aesthetics vs Attributes

- It is common to confuse between aesthetics mapping and visible attributes.
- The confusion is often due to the fact that all visual aesthetics can be used as fixed visible attributes.
- Fixed attributes specify how all plotting components should look, e.g., colors of all points, size of all points, or transparency of all points. This is different than aesthetic mapping.
- To specify a fixed attribute, call it outside the `aes()` function.

Aesthetics vs Attributes

- It is common to confuse between aesthetics mapping and visible attributes.
- The confusion is often due to the fact that all visual aesthetics can be used as fixed visible attributes.
- Fixed attributes specify how all plotting components should look, e.g., colors of all points, size of all points, or transparency of all points. This is different than aesthetic mapping.
- To specify a fixed attribute, call it outside the `aes()` function.

Aesthetics vs Attributes

- It is common to confuse between aesthetics mapping and visible attributes.
- The confusion is often due to the fact that all visual aesthetics can be used as fixed visible attributes.
- Fixed attributes specify how all plotting components should look, e.g., colors of all points, size of all points, or transparency of all points. This is different than aesthetic mapping.
- To specify a fixed attribute, call it outside the `aes()` function.

Common attributes: `alpha`, `shape` and `size`

- In a scatter plot, the attributes `alpha`, `shape` and `size` can be controlled to produce better visualisation when there is overplotting (i.e., points lie on top of each other).
- `alpha` controls the transparency — `alpha = 0` means completely transparent, while `alpha = 1` means completely opaque. A suitable value between 0 and 1 should be chosen.
- `shape` controls the point shape, e.g., `shape = 1` plots open circle, while `shape = 21` plots closed circle.
- `size` controls the size of geom objects.

Example: `alpha` controls transparency

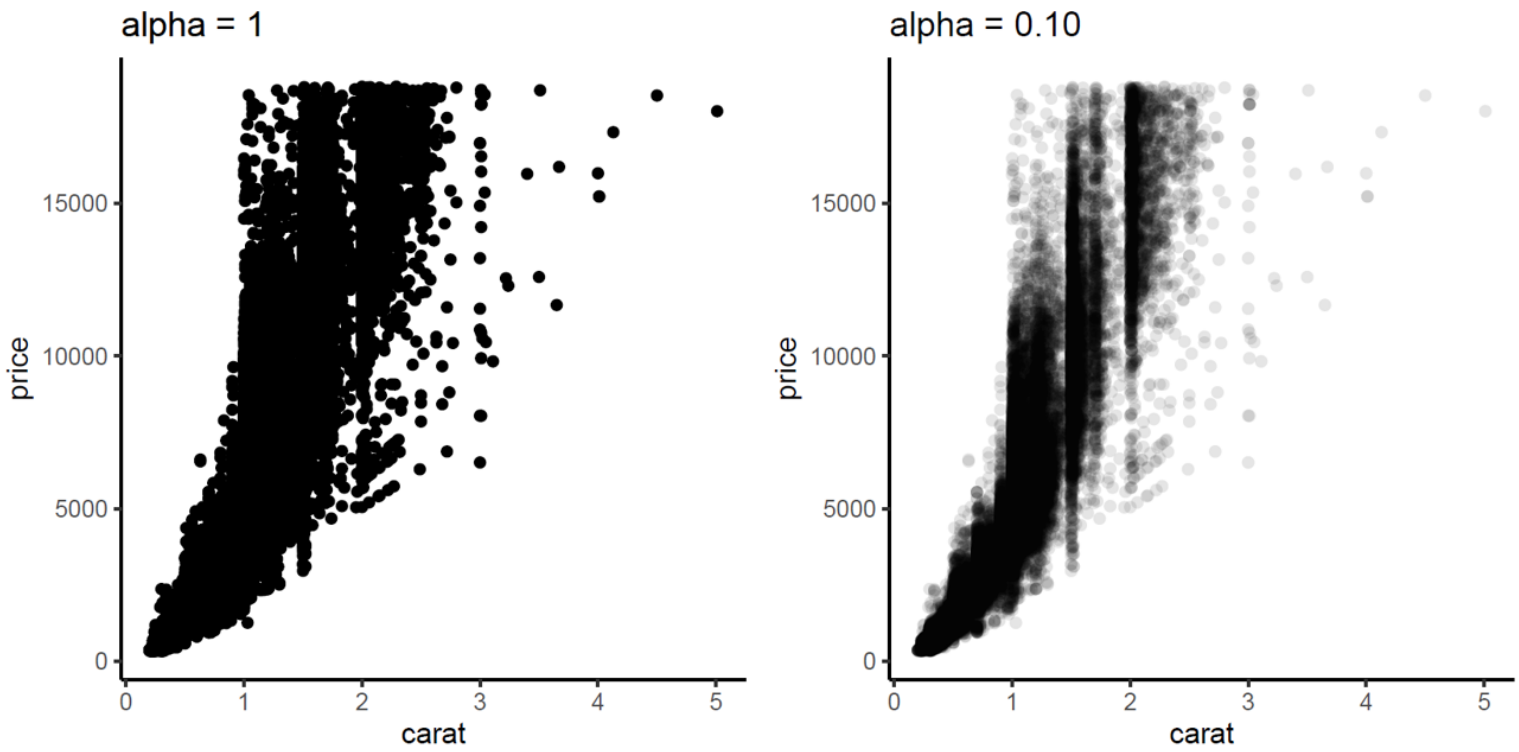


Figure 13: Diamonds price plotted against carat (weight of the diamond between 0.2 and 5.01) using the `diamonds` dataset from the `ggplot2` package.

RCode: alpha controls transparency

```
# price vs carat
base_plt <- ggplot(diamonds,
   aes(x = carat, y = price))

# alpha = 1
base_plt + geom_point()

# alpha = 0.10
base_plt +
geom_point(alpha = 0.10)
```

Attributes: alpha, size, shape, and color

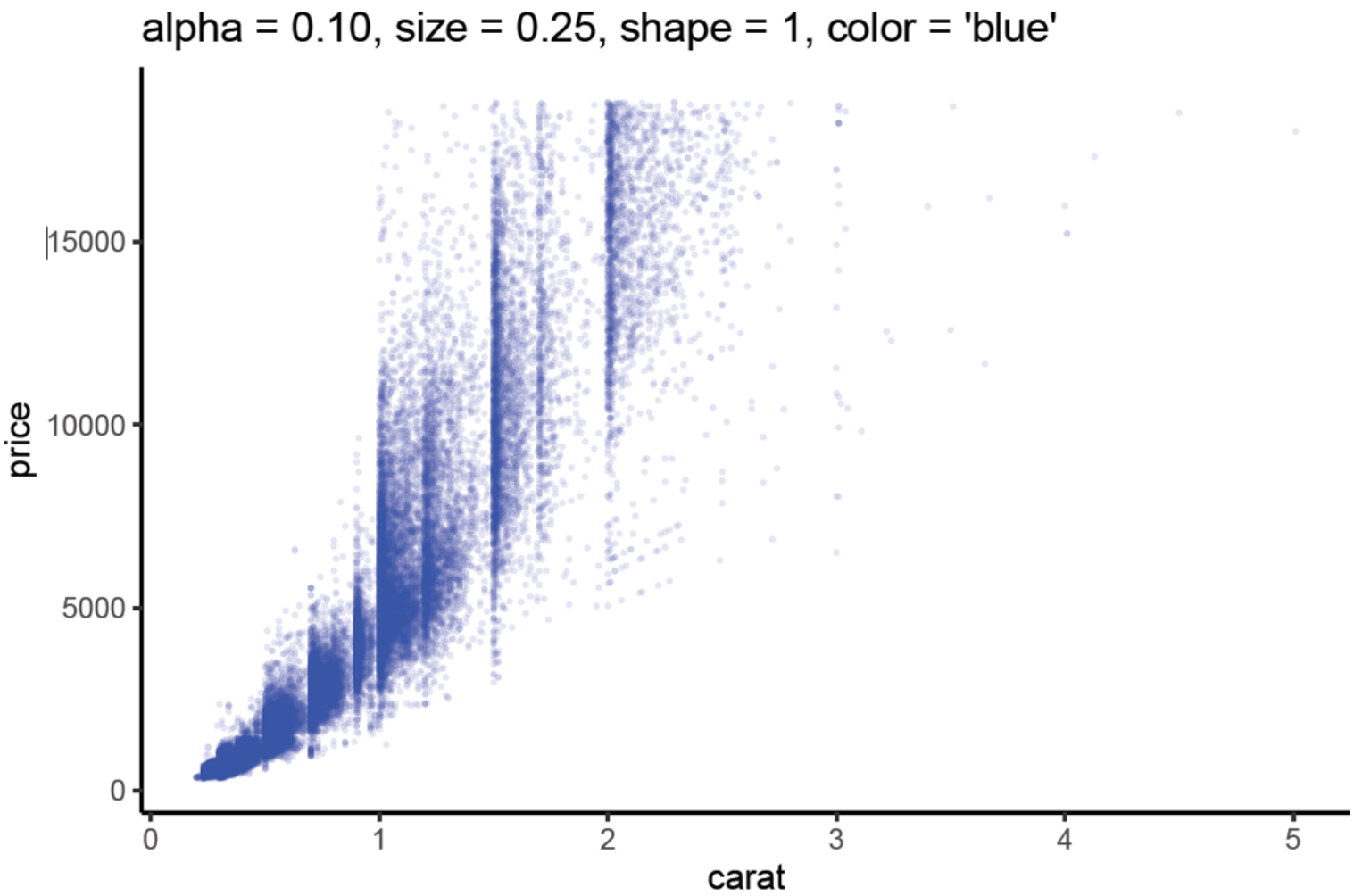


Figure 14: Diamonds price plotted against carat using the `diamonds` dataset from the `ggplot2` package.

RCode: alpha, size, shape, and color

```
# price vs carat
base_plt <- ggplot(diamonds, aes(x =
    carat, y = price))

# alpha = 0.1, size = 0.25, shape = 1,
    color = 'blue'
base_plt +
 geom_point(alpha = 0.10, size = 0.25,
            shape = 1, color = "blue")
```

Combining different geoms

We can add one geom layer on top of another geom either using the same aesthetic mappings or different aesthetics.

Combining geom_point and geom_smooth

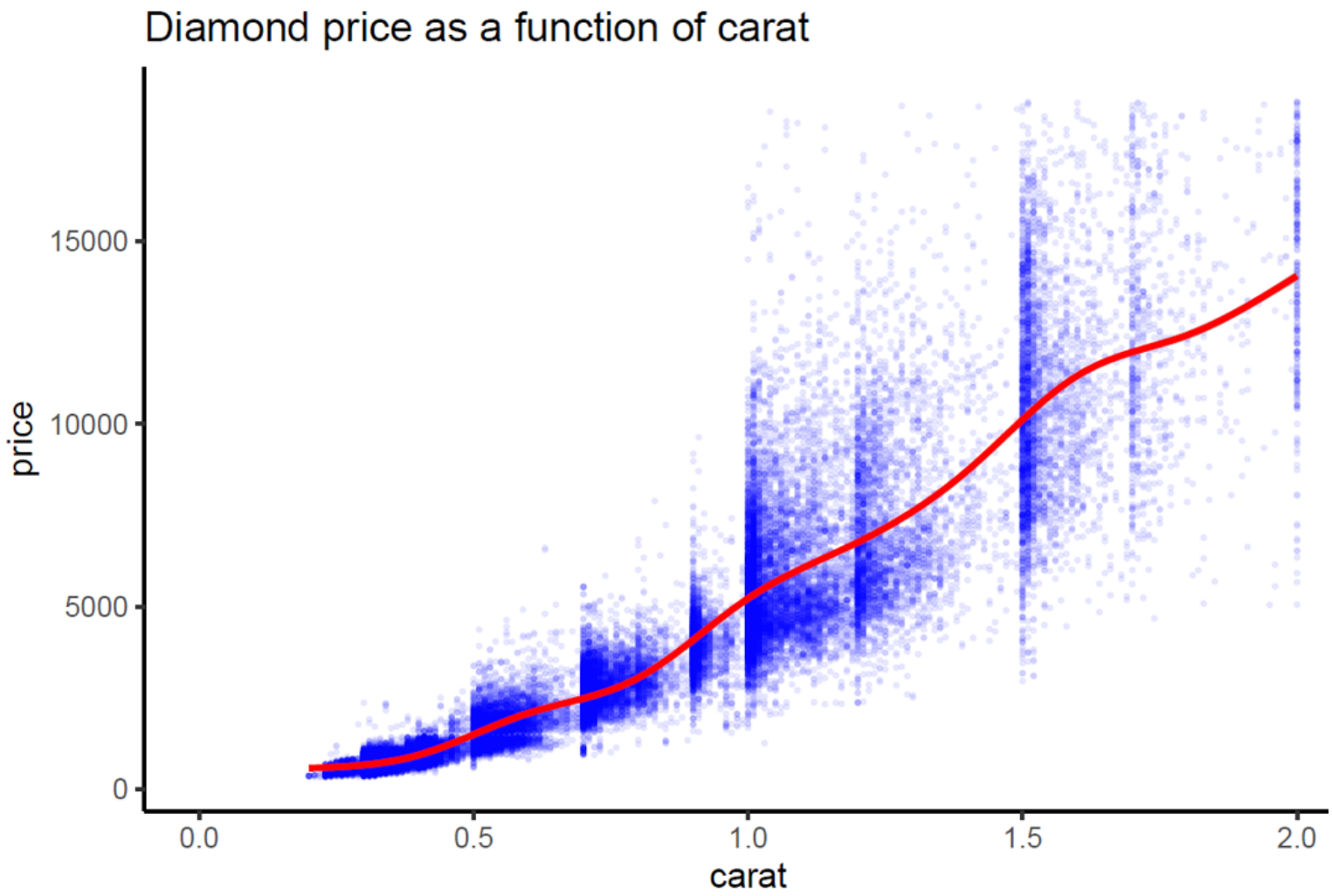


Figure 15: GAM used to predict diamond price as a function of carat.

RCode: geom_point and geom_smooth

```
# price vs carat
ggplot(diamonds, aes(x = carat, y =
    price)) +
# geom_point
geom_point(alpha = 0.10, size=0.25,
    shape=1, color = "blue") +
# geom_smooth
geom_smooth(color = "red") +
xlim(c(0, 2.0)) +
theme_classic()
```

Note: Different visible attribute specifications in geom_point() and geom_smooth().

Aesthetics in geom_point and geom_smooth

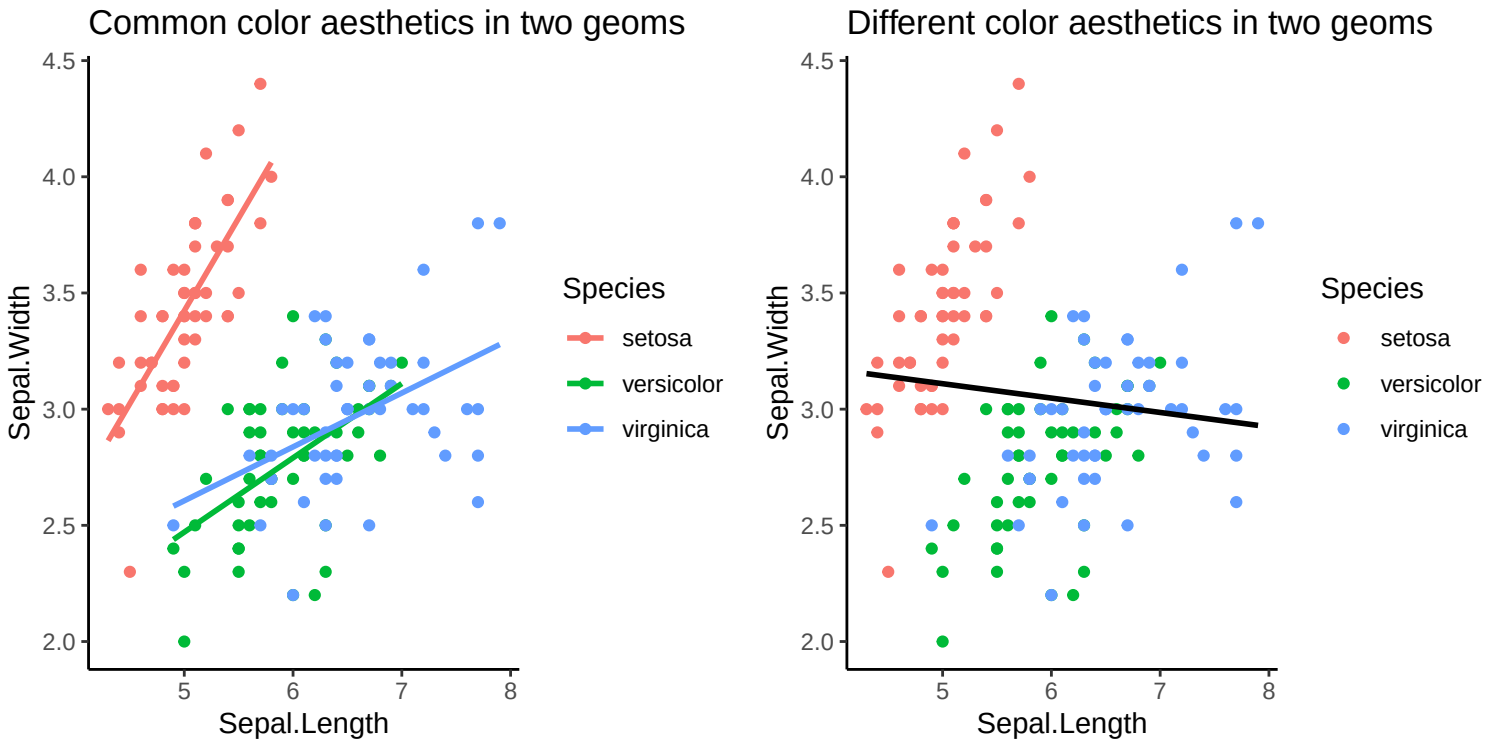


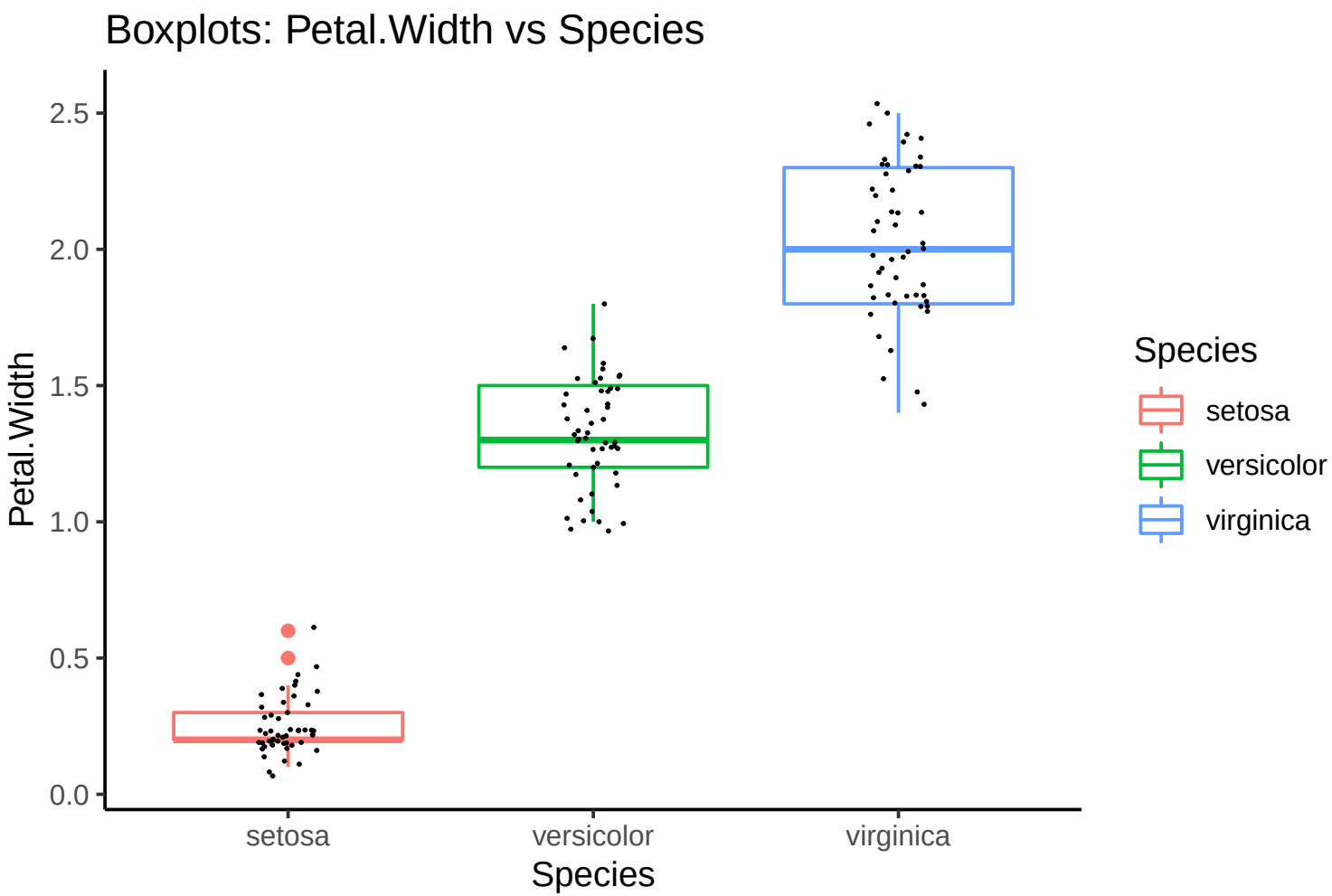
Figure 16: *Left*: Both geoms inherits same aesthetic mappings; *Right*: Only `geom_point` maps Species to color aesthetics.

Aesthetics in geom_point and geom_smooth

```
##########################################
# Same aes mapping in both geoms         #
##########################################
ggplot(iris, aes(Sepal.Length, Sepal.Width,
                 color = Species)) +
geom_point() +
geom_smooth(method = "lm", se = FALSE) +
theme_classic()
##########################################
# color=Species only in geom_point()     #
##########################################
ggplot(iris,aes(Sepal.Length,Sepal.Width)) +
geom_point(aes(color = Species)) +
geom_smooth(method = "lm", se = FALSE, color
   ="black") +
theme_classic()
```

Combining geom_boxplot and geom_jitter

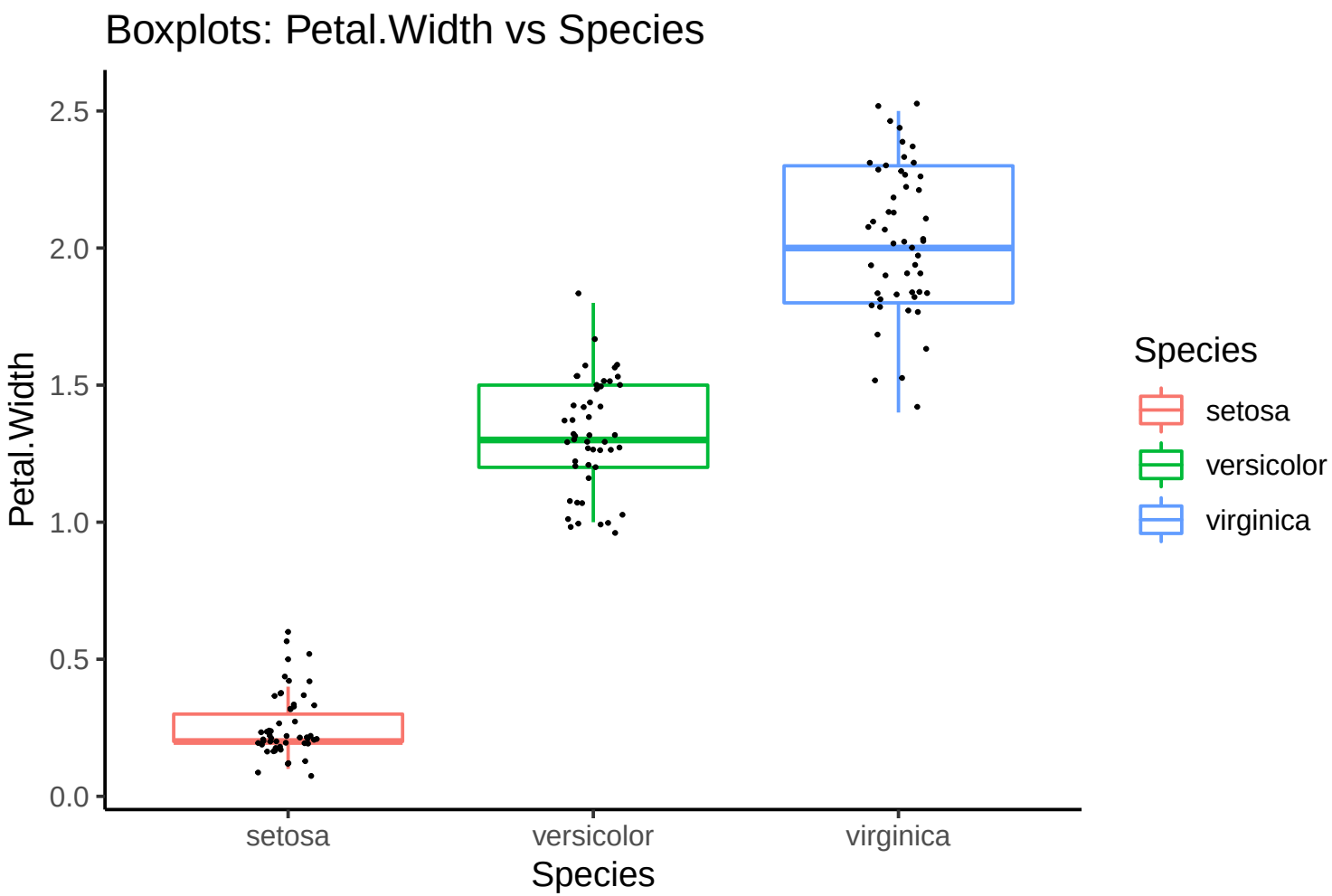
Can you guess the aesthetic mappings? Also, can you spot a small problem with this plot?



Curtin University

Resolving the outlier colour and shape issue

Use `outlier.color` and `outlier.size` in `geom_boxplot` to modify outlier color and size, respectively.



RCode: geom_boxplot and geom_jiter

```
# Petal.Width boxplots for three species
ggplot(iris,aes(Species,Petal.Width)) +
# Map Species to color in geom_boxplot
geom_boxplot(aes(color = Species),
             outlier.color = "black",
             outlier.size = 0.20) +
# Specify size and range of points
geom_jitter(size=0.20, width = 0.10) +
theme_classic()
```

A few points on jittering

- Jittered plot is an useful alternative to a scatter plot when there is overplotting.
- Because `geom_jitter()` adds random noise to points, it may not be reproducible.
- To reproduce the same pattern, we should use `set.seed()` before calling `geom_jitter()`.
- The other solution is to use `position = position_jitter(seed = 123)` inside the `geom_point()` function.

Example: combined geoms with faceting

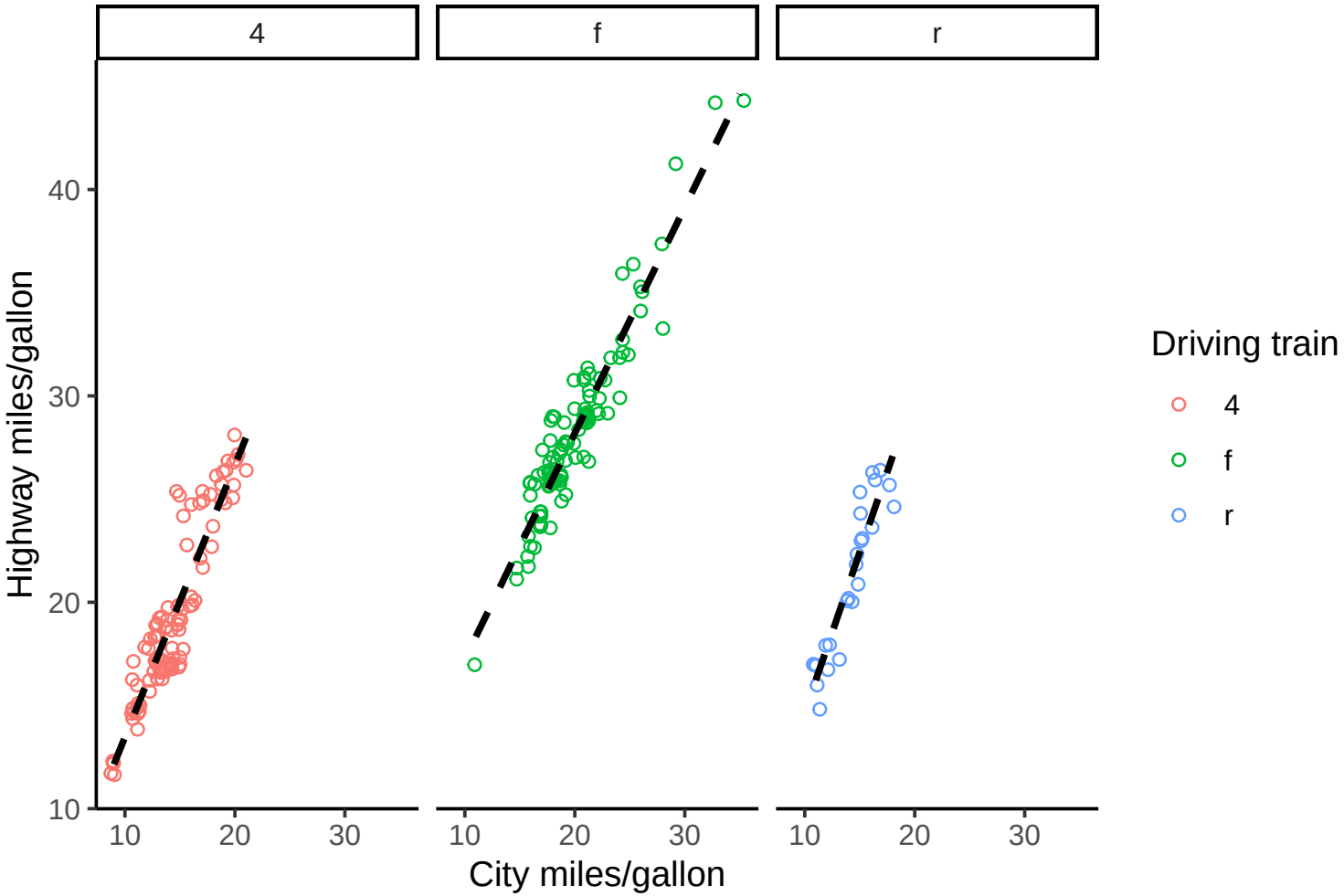


Figure 17: `hwy` vs `cty` for three `drv`s.

RCode: combined geoms with faceting

```
ggplot(mpg, aes(x = cty, y=hwy)) +
# Reproduce jittering
geom_point(aes(color=factor(drv)),
position = position_jitter(seed = 123),
shape = 1) +
# fit a linear regression
geom_smooth(method = "lm", se = FALSE,
    color="black", linetype = "dashed") +
# Faceting based on drv factor
facet_wrap(~ factor(drv)) +
theme_classic() +
labs(color = "Driving train",
      x = "City miles/gallon",
      y = "Highway miles/gallon")
```

Summary

- Ggplot2 is built using the *grammar of graphics*, and that makes it possible to build complex visualisations using the basic visualisation components.
- Ggplot figures are built by adding visualisation layer on top of each other, and we can inspect a figure at any stage of this process and make decision about the next enhancement.
- Three key components are data, aesthetic, and geometry.
- There is a subtle difference between aesthetic mapping and visible attributes.
- Many geoms can be combined to produce complex visualisations.
- Faceting is a great way to show the relationship between multiple variables.