ST720 Data Science

Data Vaisualization with ggplot2

Seung Jun Shin (sjshin@krea.ac.kr)

Department of Statistics, Korea University

Introduction

- `ggplot2` is visualization package in R
- R has several systems for making graphs, but ggplot2 is one of the most elegant and most versatile.
- ggplot2 implements the grammar of graphics.

package : tidyverse

- `tidyverse` is a collection of packages for data science: vidualization, transformation, and modelling

```
install.packages("tidyverse")
library(tidyverse)
```

- includes `ggplot2`

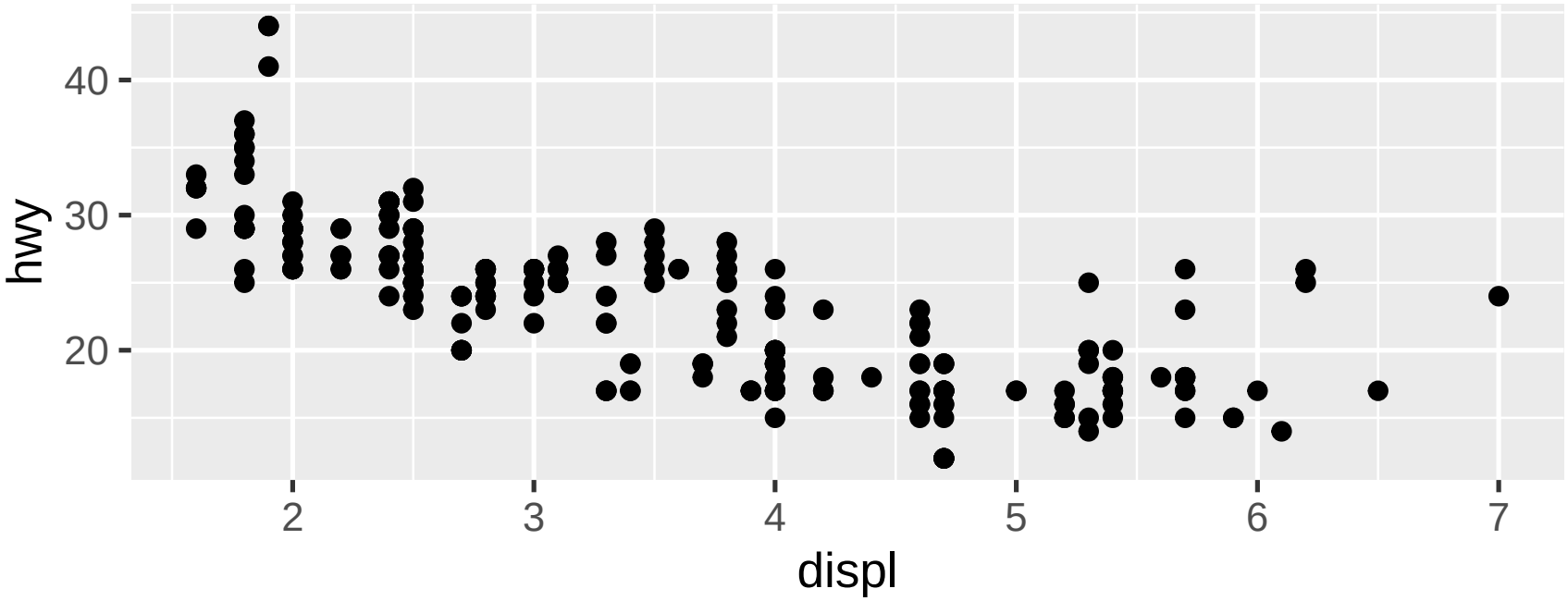
mpg dataset

- collected by EPA on 38 models of cars
- displ : a car's engine size, in liters
- hwy : a car's fuel efficiency on the highway,in miles per gallon(mpg)

```
## # A tibble: 234 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         a4      1.8  1999     4 auto~ f        18    29 p     comp~
##  2 audi         a4      1.8  1999     4 manu~ f        21    29 p     comp~
##  3 audi         a4      2    2008     4 manu~ f        20    31 p     comp~
##  4 audi         a4      2    2008     4 auto~ f        21    30 p     comp~
##  5 audi         a4      2.8  1999     6 auto~ f        16    26 p     comp~
##  6 audi         a4      2.8  1999     6 manu~ f        18    26 p     comp~
##  7 audi         a4      3.1  2008     6 auto~ f        18    27 p     comp~
##  8 audi         a4 q~   1.8  1999     4 manu~ 4        18    26 p     comp~
##  9 audi         a4 q~   1.8  1999     4 auto~ 4        16    25 p     comp~
## 10 audi         a4 q~   2    2008     4 manu~ 4        20    28 p     comp~
## # ... with 224 more rows
```

Creating a ggplot

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



- negative relationship between engine size(dipl) and fuel efficiency(hwy)

Aesthetic Mapping

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

- **ggplot(data = mpg)** : creates an empty graph
- **geom_point()** : adds a layer of points to your plot

Aesthetic Mapping

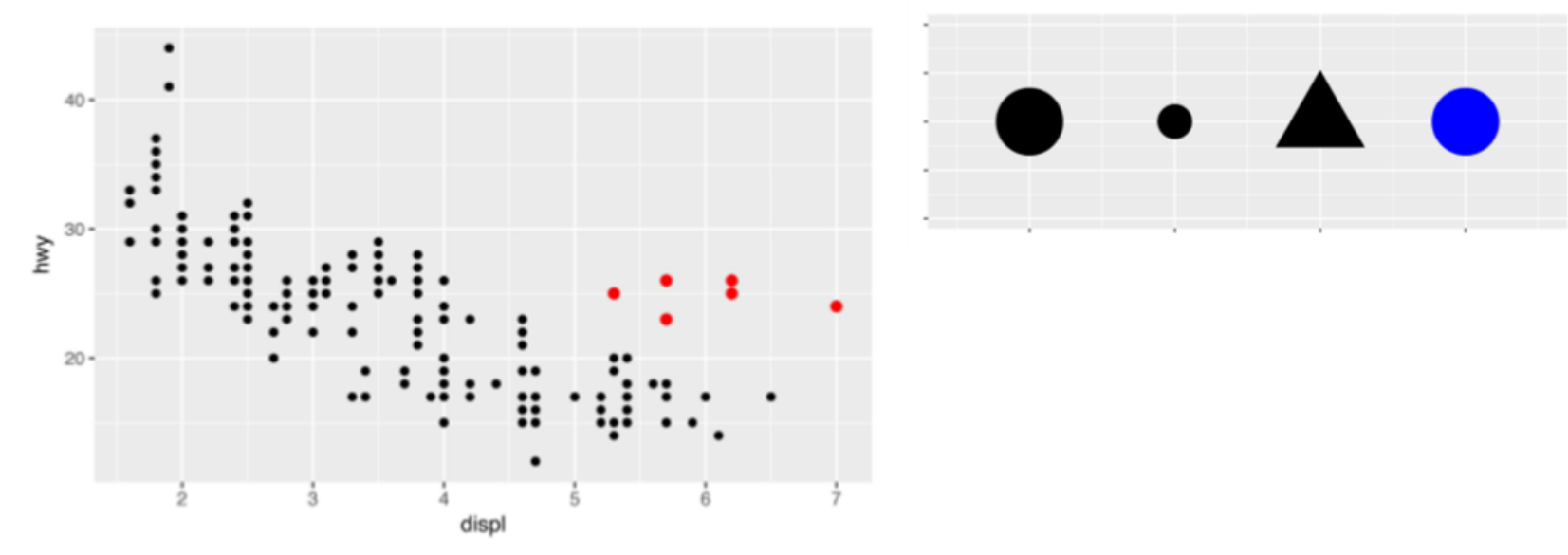
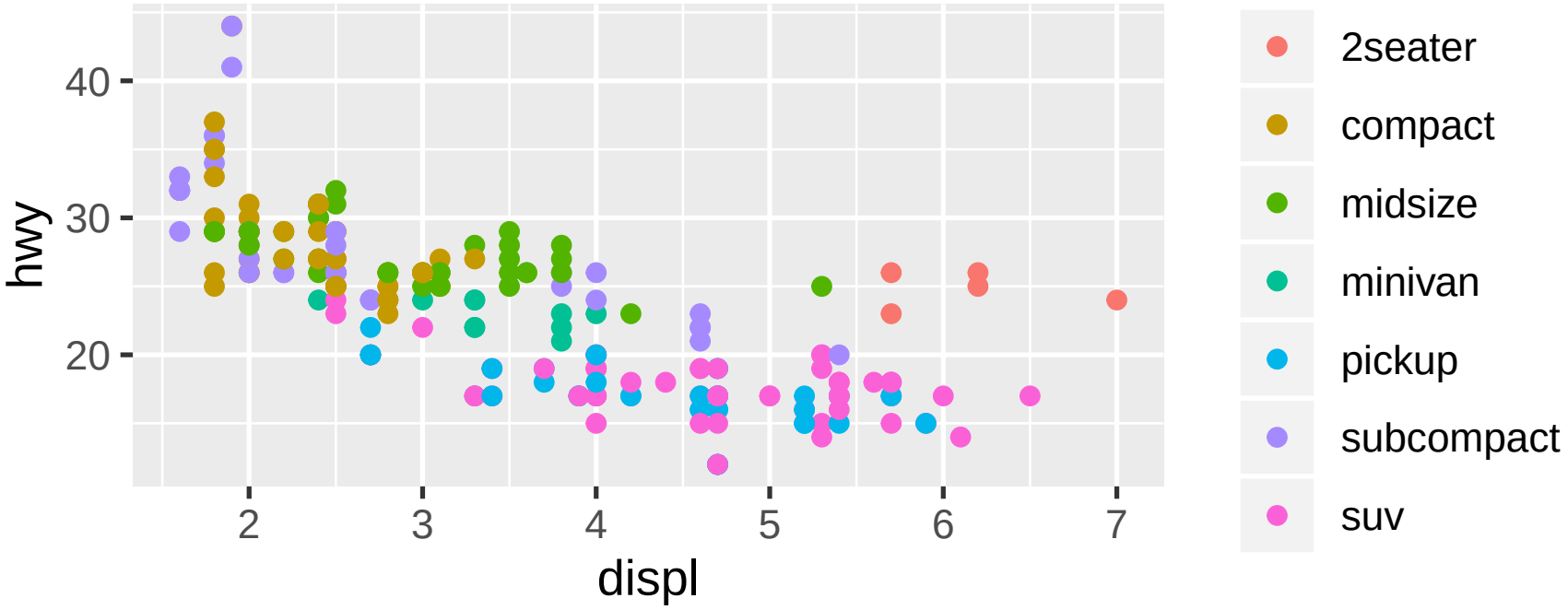


Figure 1: aesthetic

- Does the red dots are SUV ?
- mapping it to an aesthetic in two-dimensional scatterplot
- **level** : aesthetic properties (point's size, shape, and color)

Aesthetic Mapping (color/colour)

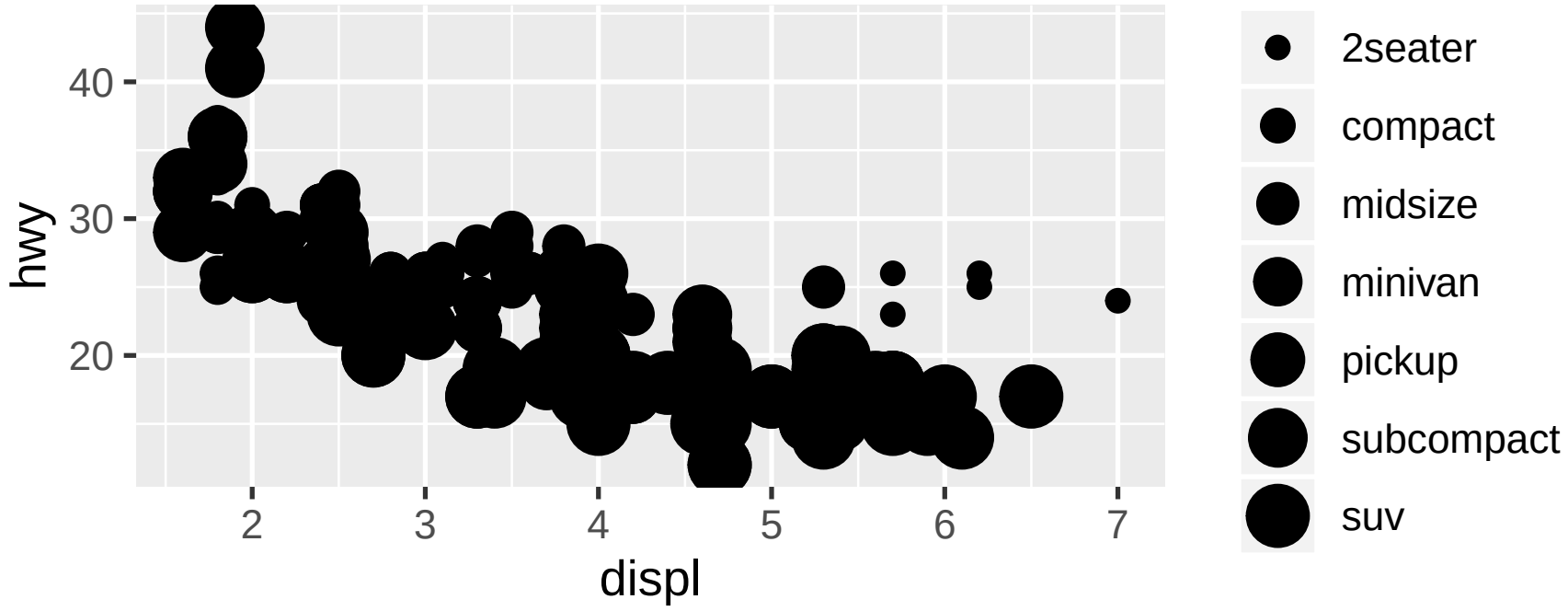
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, color = class))
```



- **aes()** : map an aesthetic to a variable
- mapping class by the color

Aesthetic Mapping (size)

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

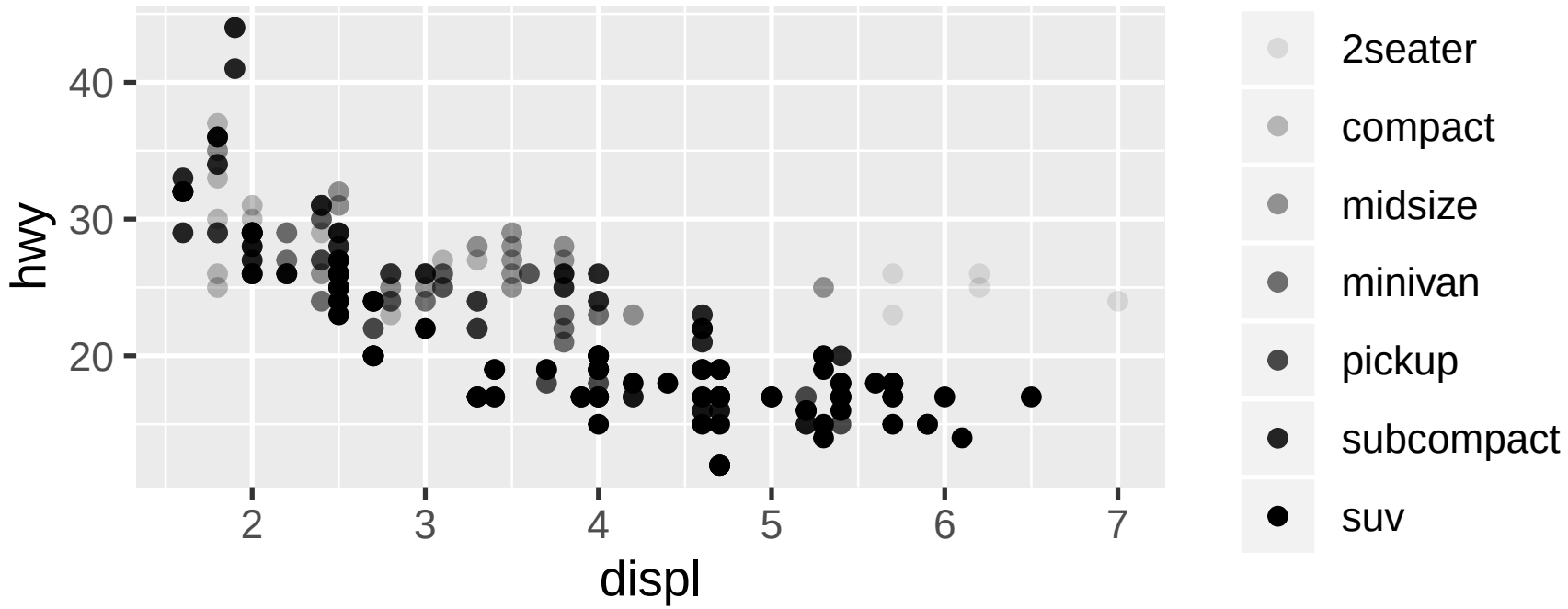


- mapping class by the size
- warning : mapping an unordered cariable to an ordered aesthetic (size) is not a good idea

Aesthetic Mapping (alpha)

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

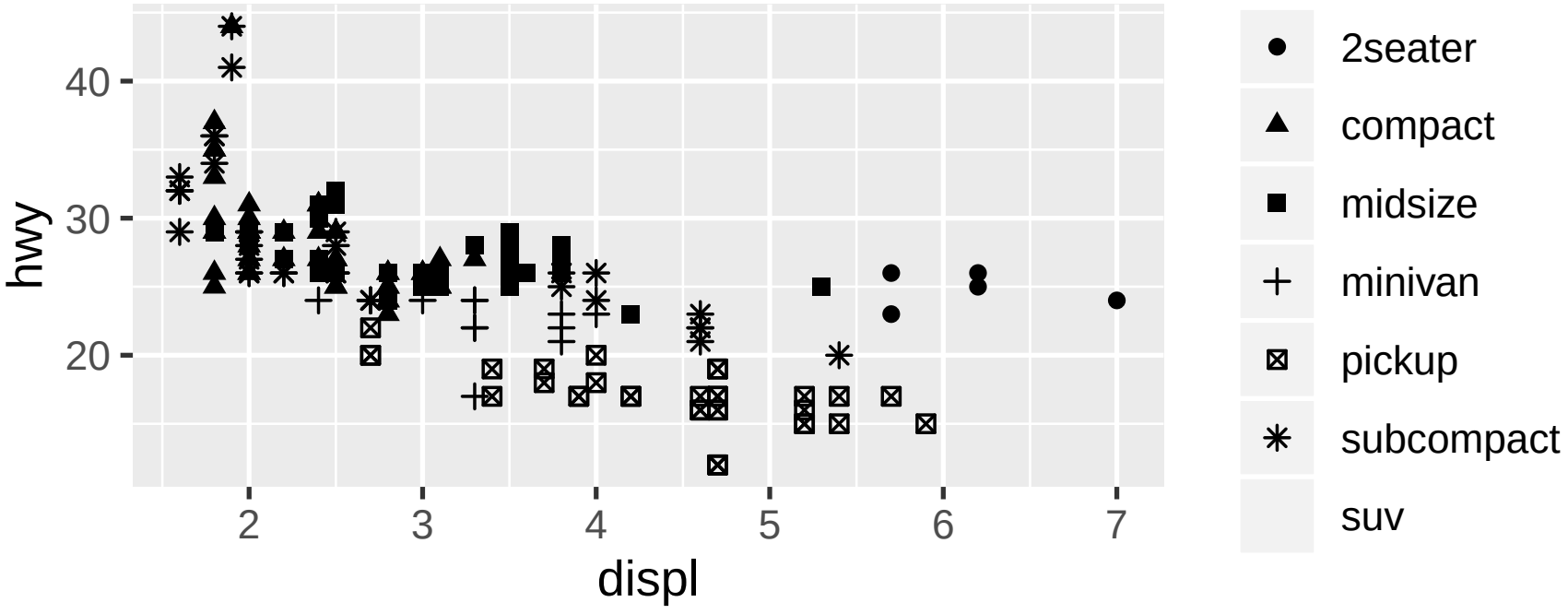
```
## Warning: Using alpha for a discrete variable is not advised.
```



- mapping class by the transparency of the points

Aesthetic Mapping (shape)

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



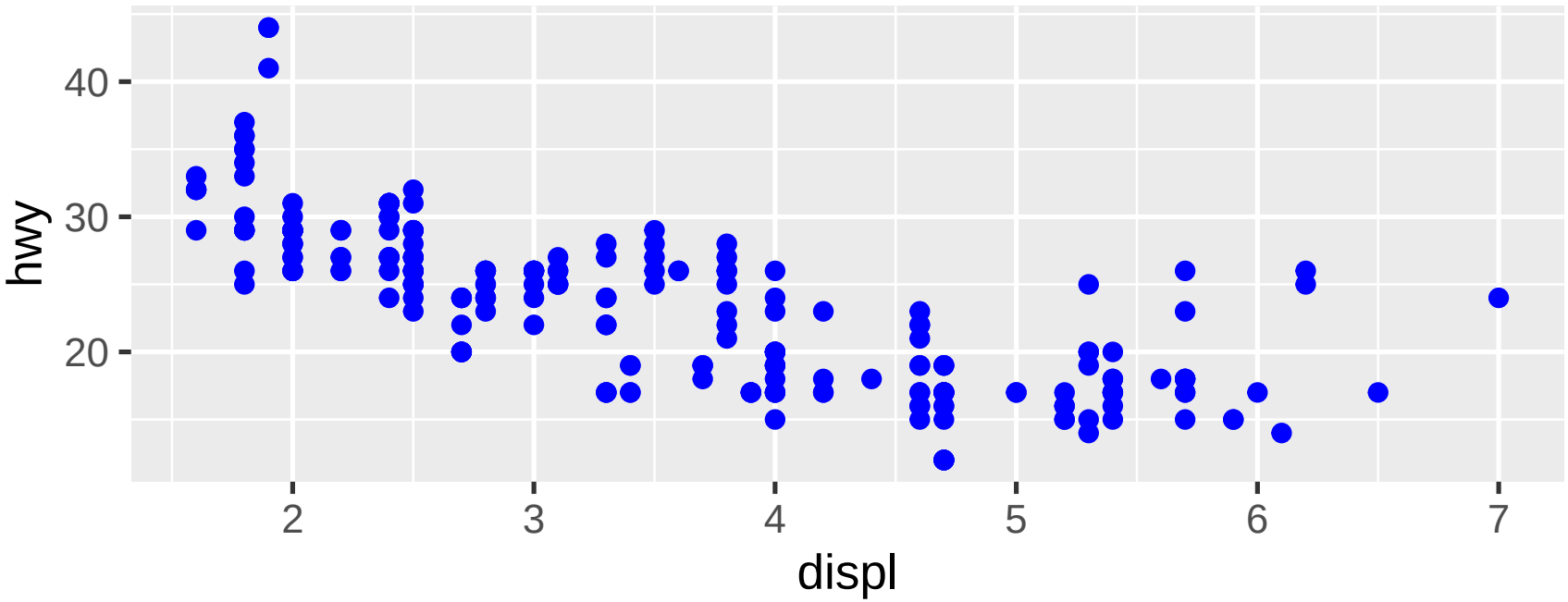
- mapping class by the shape
- only use six shapes by default(cf:`scale_shape_manual`)
- additional groups will go unploted

Aesthetic Mapping

- **aes()** : associate the name of the aesthetic with a variable to display.
- selects a reasonable scale to use with the aesthetic.
- it construcnts a legend that explain the mapping b/w levels and values.
- For x and y, it creates an axis line as a legend

Aesthetic Mapping

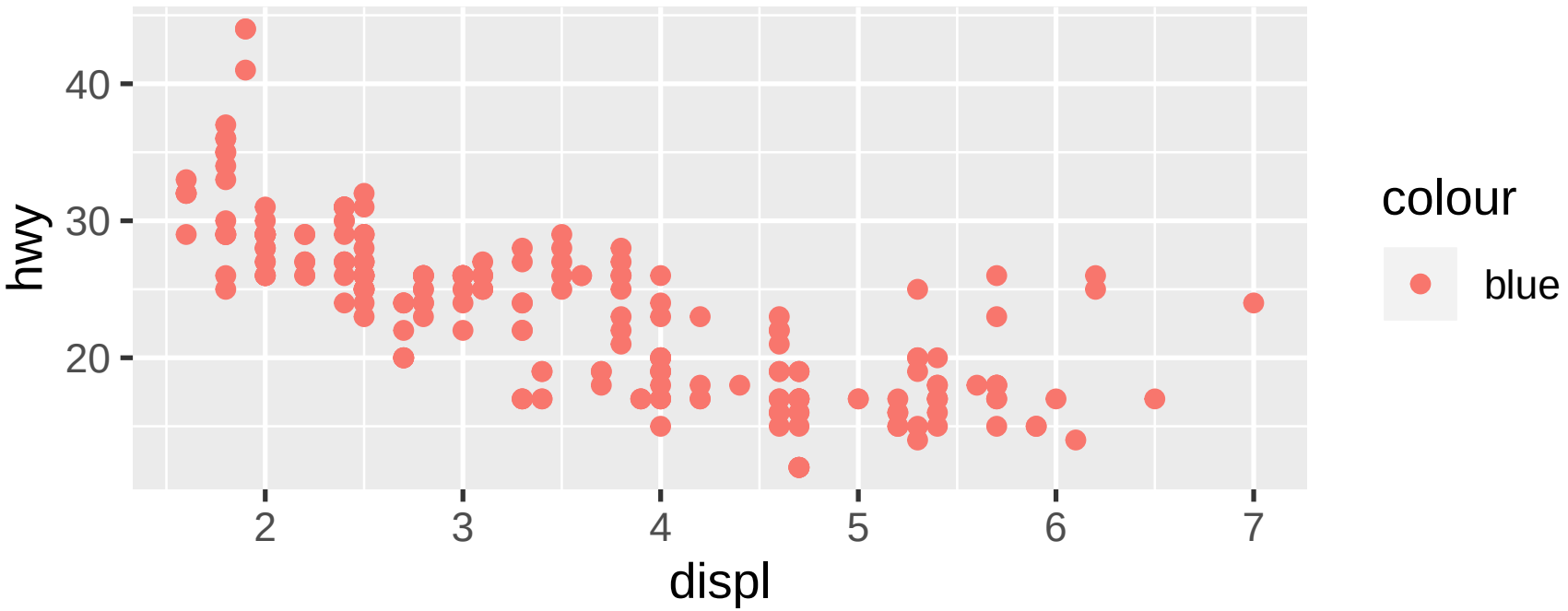
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy), color = "blue")
```



- set the aesthetic properties of geom manually
- but in here, the color doesn't have information about a variable.

Aesthetic Mapping

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



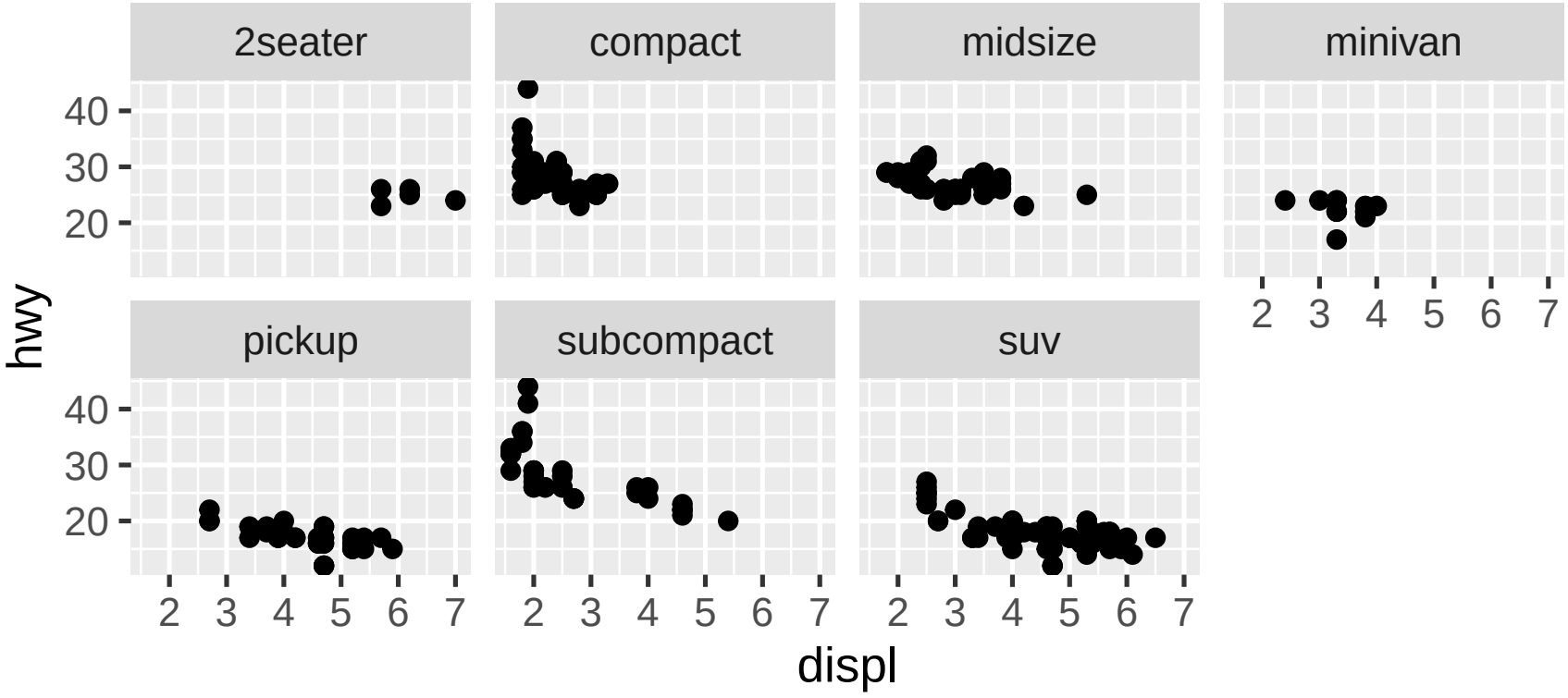
- to set an aesthetic manually, it goes outside of aes()

Facets

- One way to add additional variables is with aesthetics
- Another way, particularly useful for categorical variables, is to split your plot into facets, subplots that each display one subset of the data.

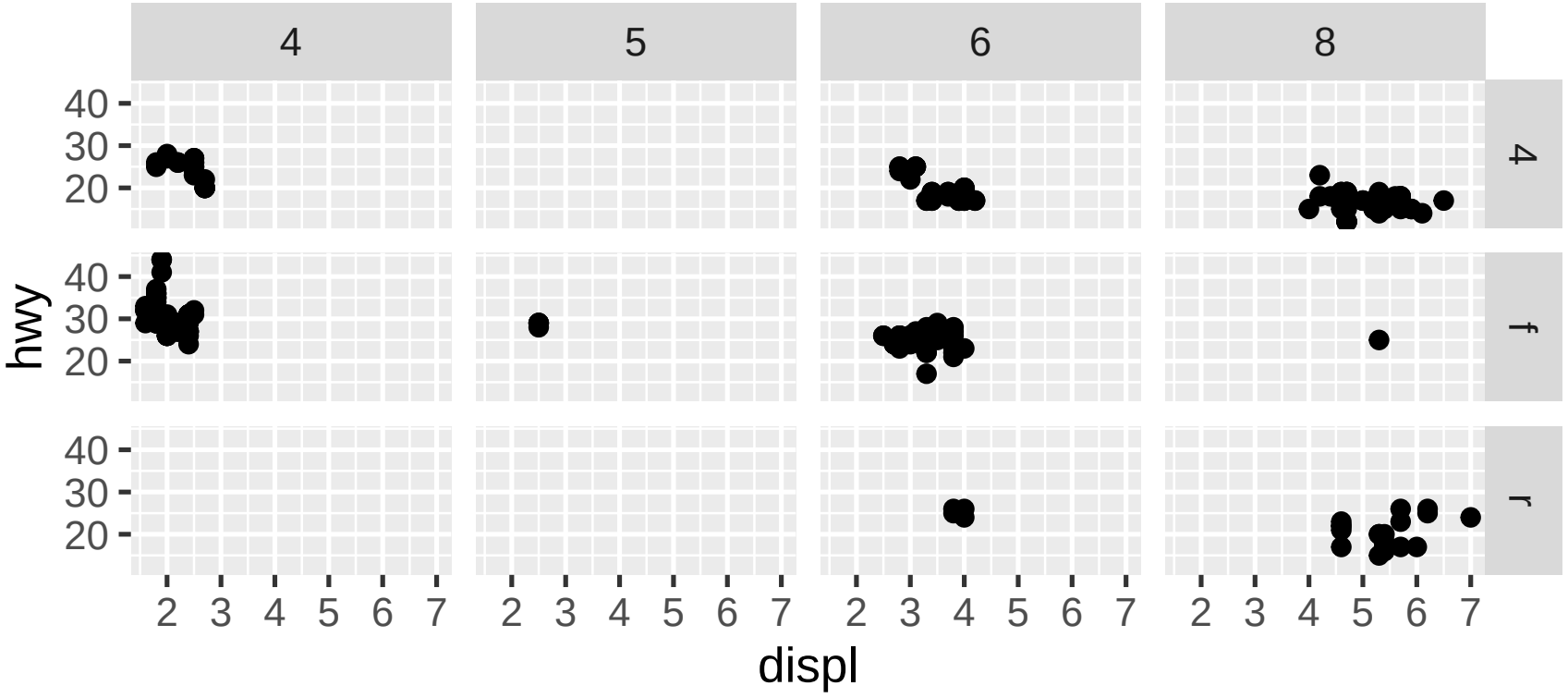
facet_wrap()

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



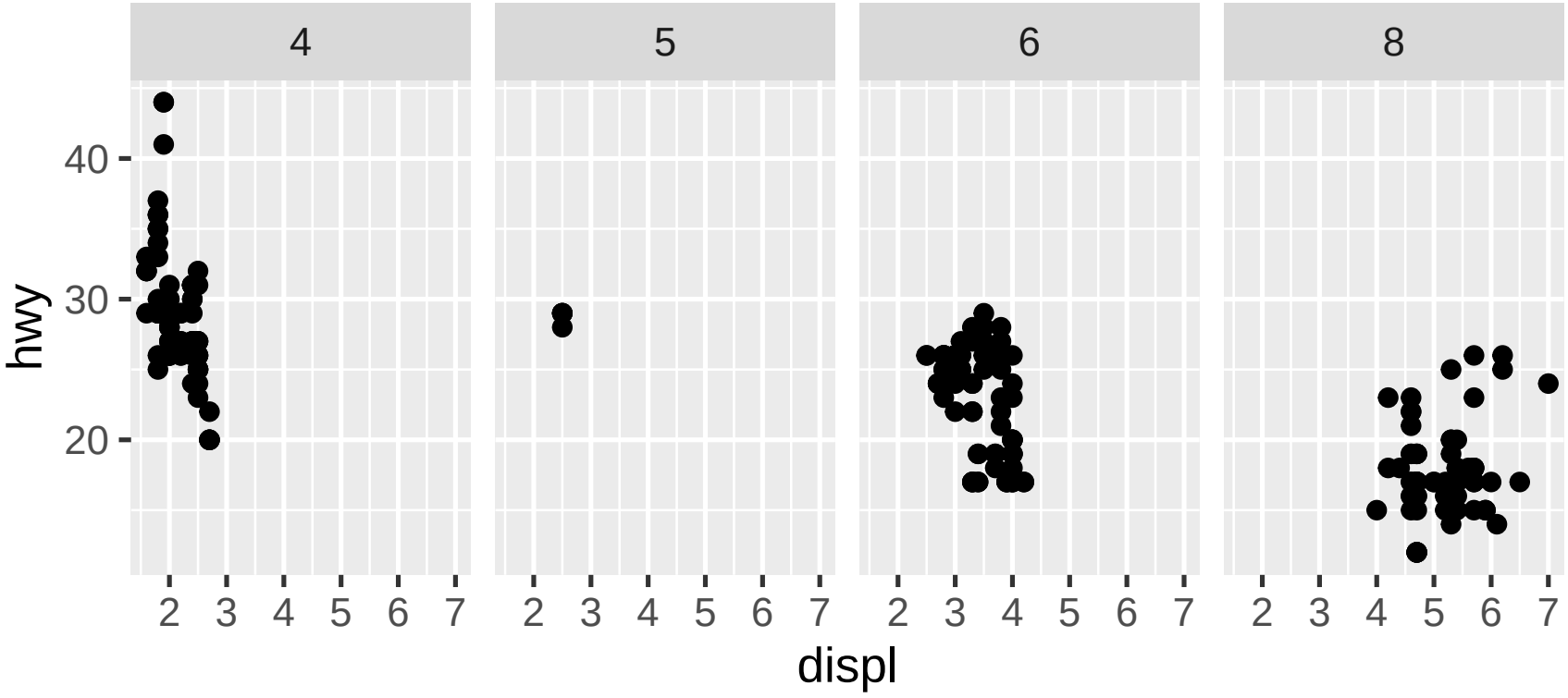
facet_grid()

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



facet_grid()

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

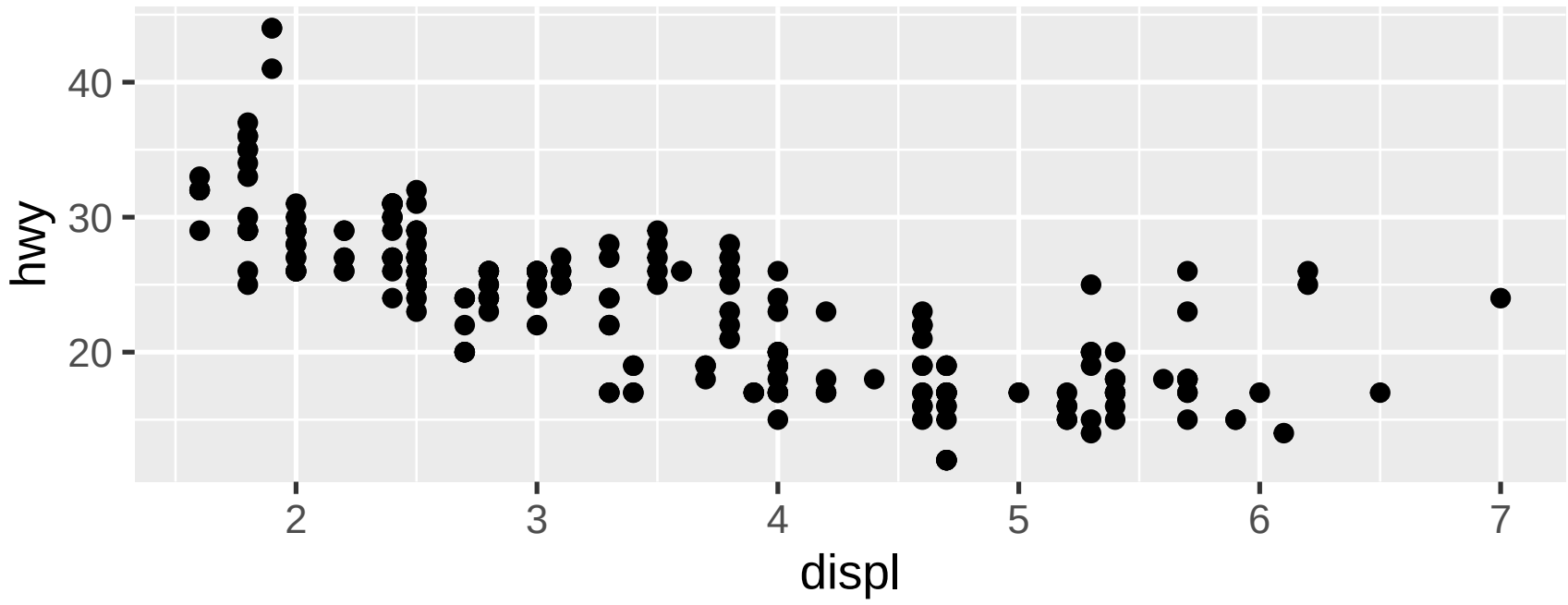


Geometric Objects

- `geom` : geometrical object
  - `bar_geom()` : bar charts
  - `line_geom()` : line charts
  - `boxplot_geom()` : boxplots
  - `point_geom()` : scattetplots
  - `smoothe_geom()` : smoothe fitted line
  - ggplot2 provieds over 30 geoms < https://www.ggplot2-exts.org >

point_geom()

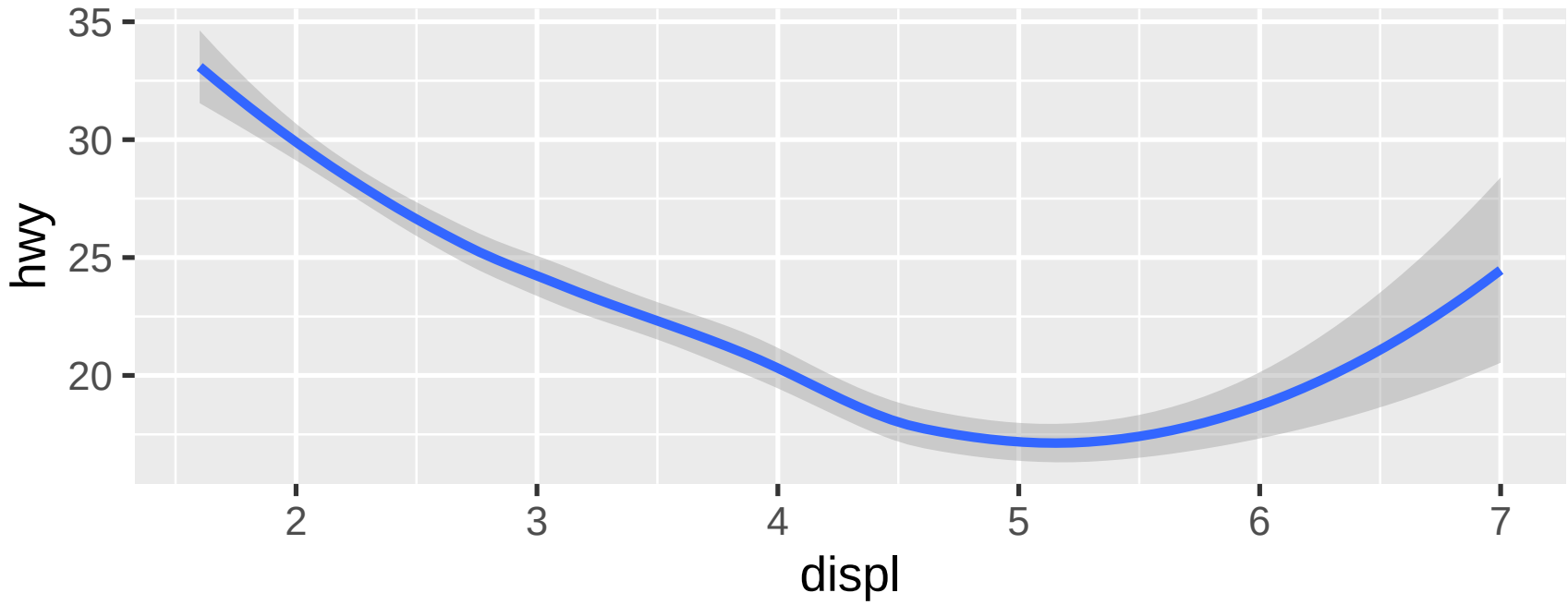
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy))
```



smooth_geom()

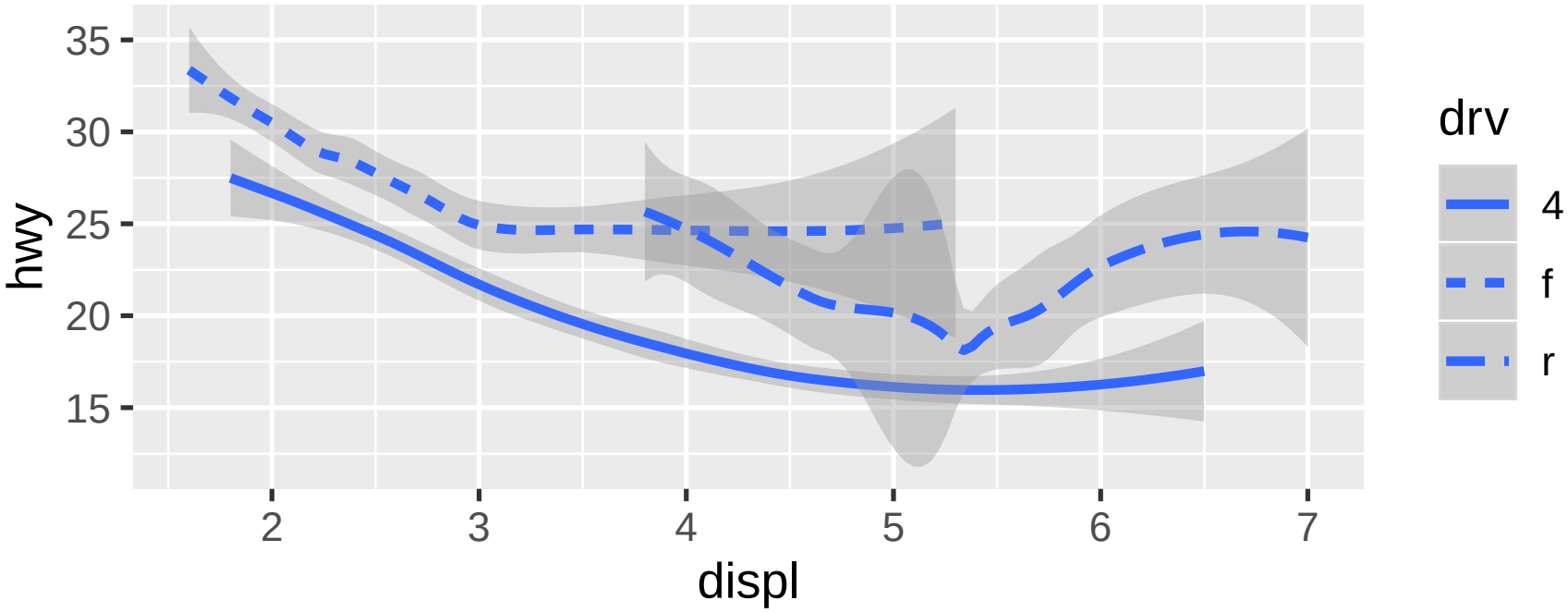
```
ggplot(data = mpg) +
  geom_smooth(mapping = aes(x = displ, y = hwy))
```

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



smooth_geom()

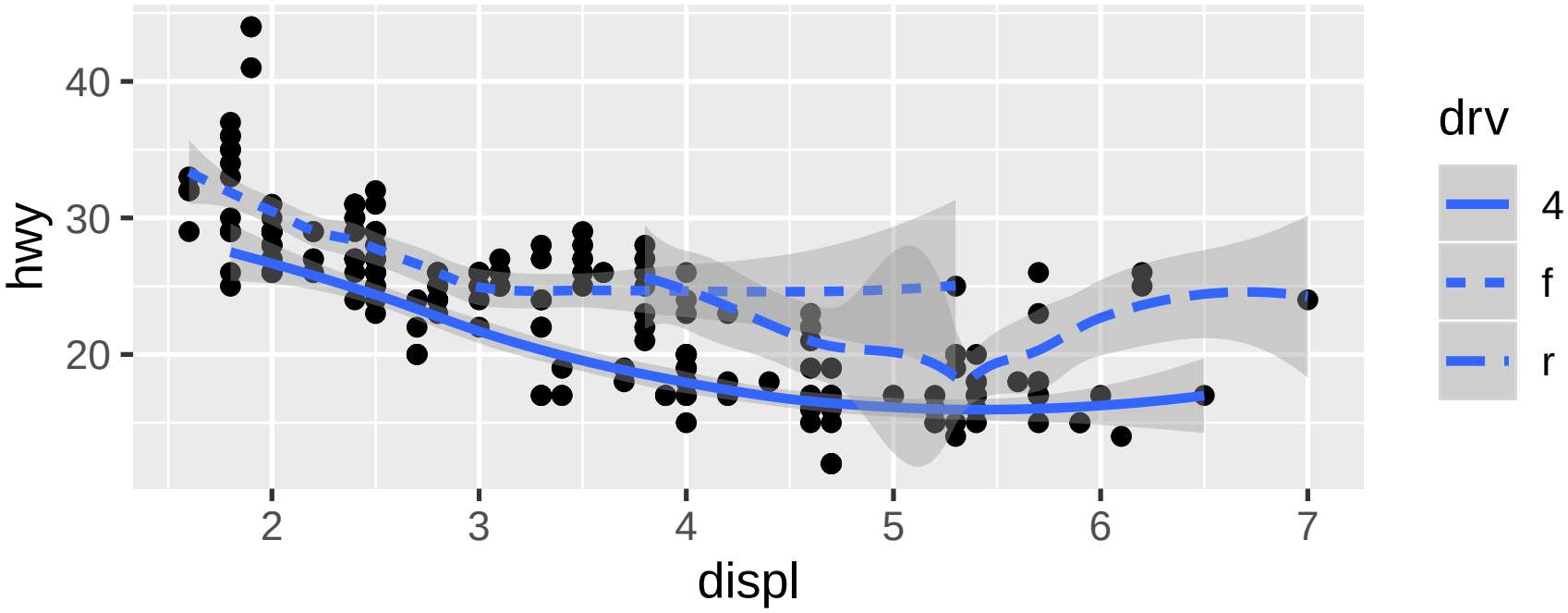
```
ggplot(data = mpg) +
  geom_smooth(mapping = aes(x = displ, y = hwy, linetype = drv))
```



- separates the cars into three lines based on **drv** values.
- 4 : stands for four-wheel drive, f : front-wheel drive, r : rear-wheel drive

smooth_geom()

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



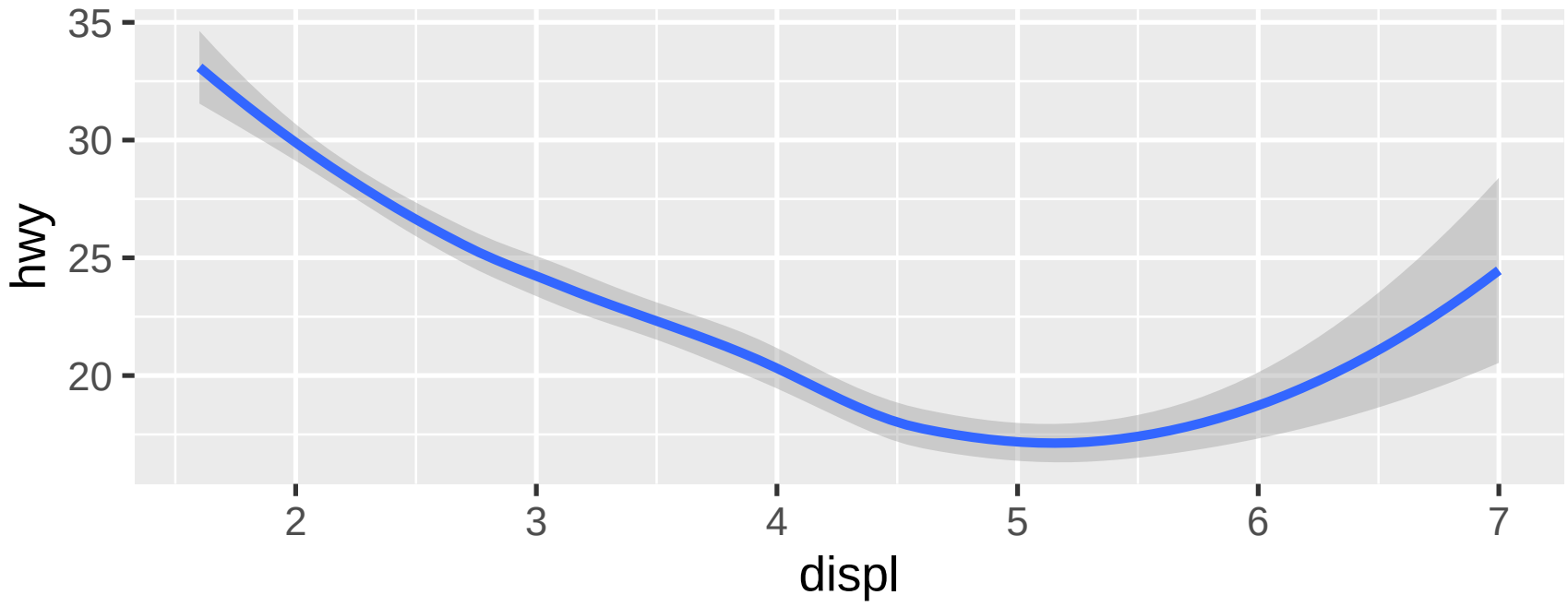
- plot contains two geons in the same graph (next section)

`smooth_geom()` with `group`

- **group** aesthetic to a categorical variable to draw multiple objects
- draw a separate object for each unique value of the grouping variable
- automatically group the data (ex) linetype, color

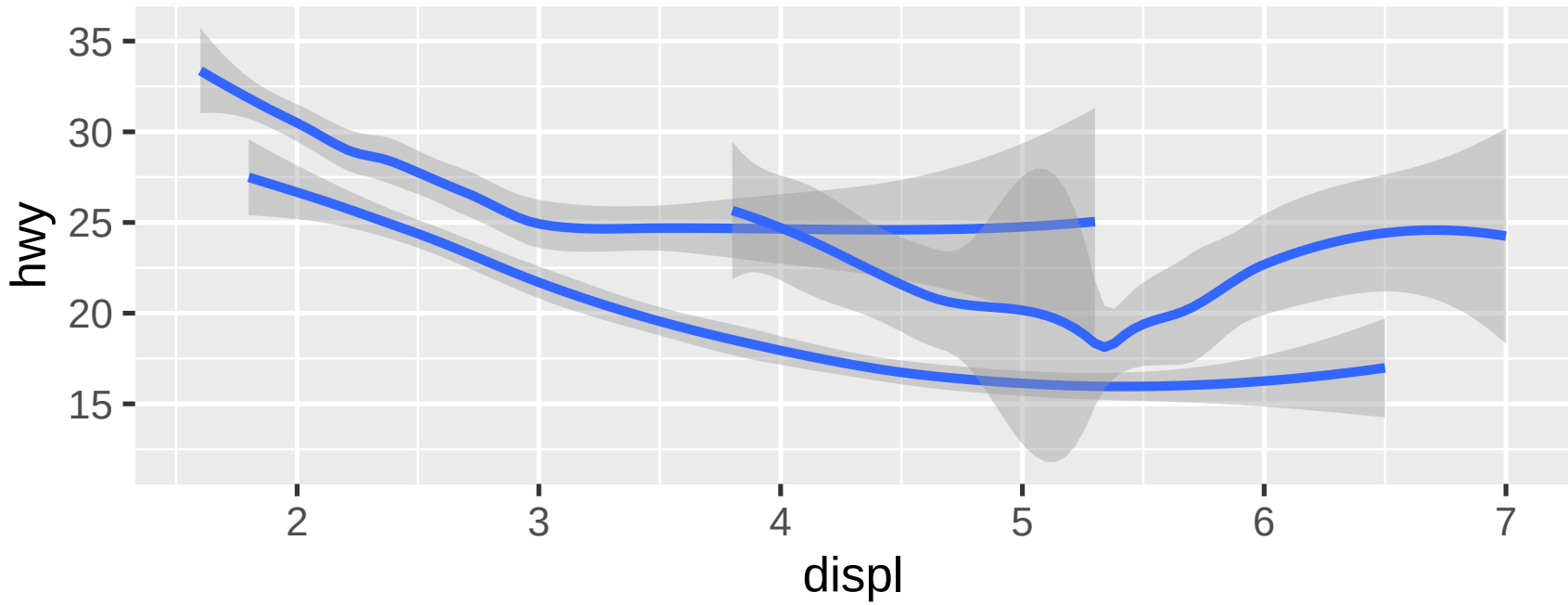
smooth_geom() with group

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



smooth_geom() with group

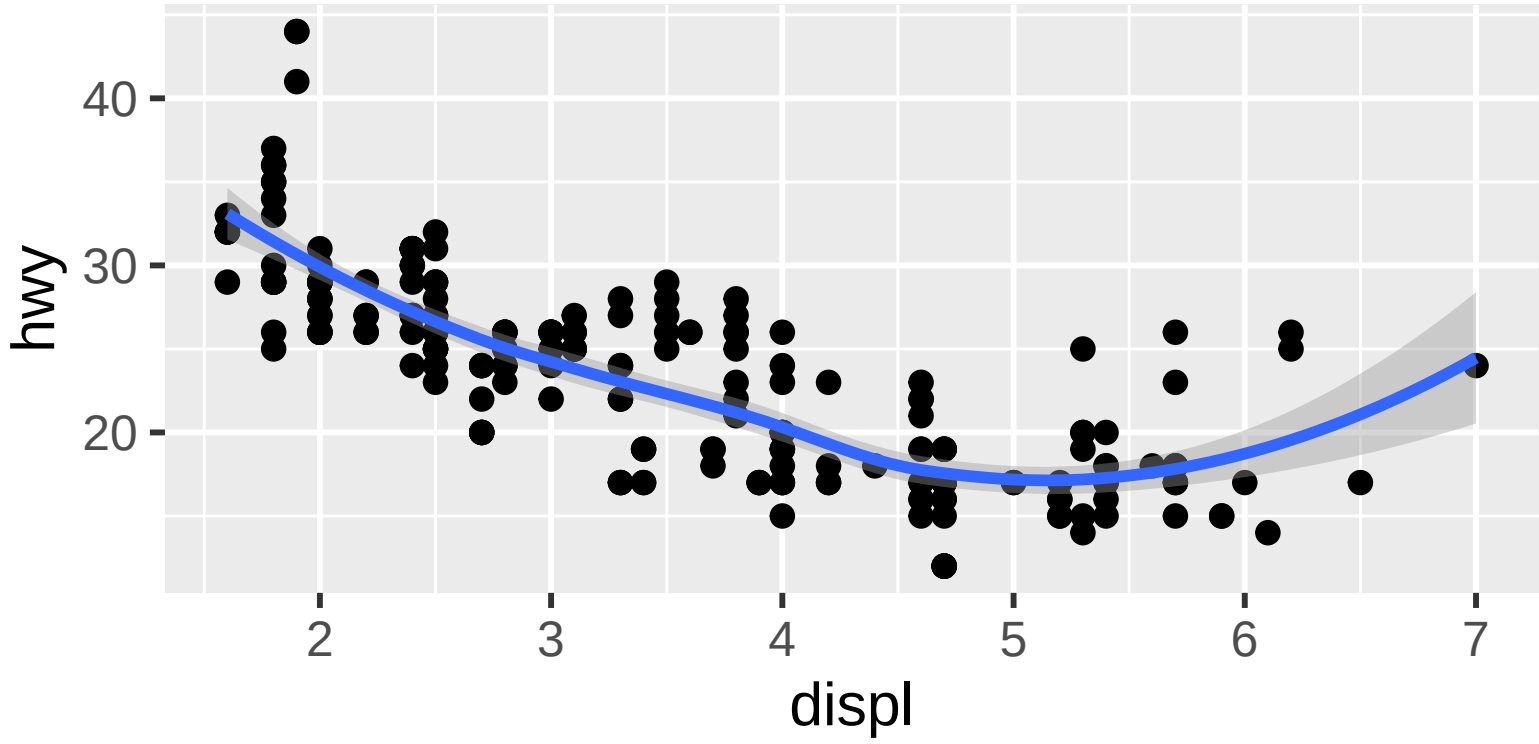
```
ggplot(data = mpg) +
  geom_smooth(mapping = aes(x = displ, y = hwy, group = drv))
```



Multiple geom

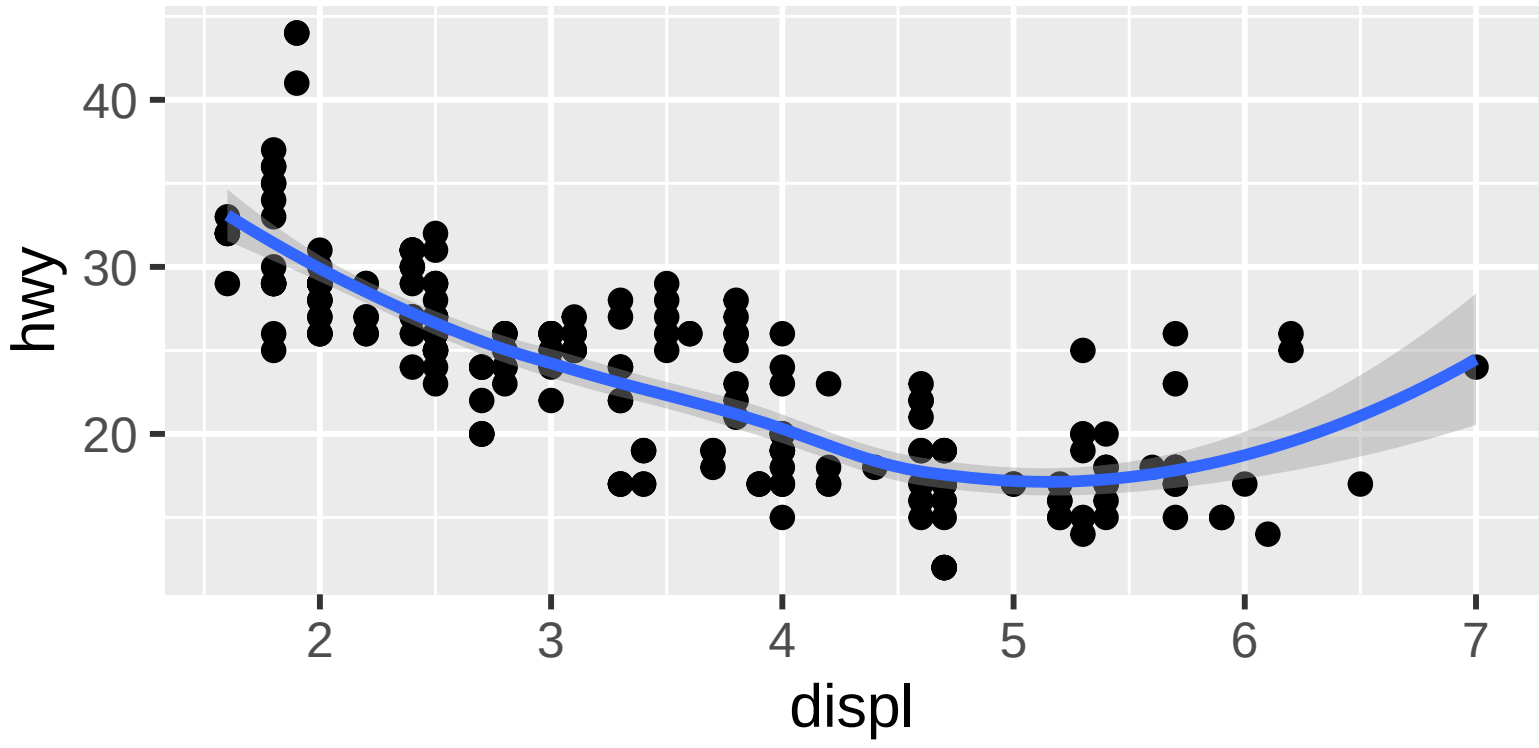
- Multiple geoms in the same plot

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



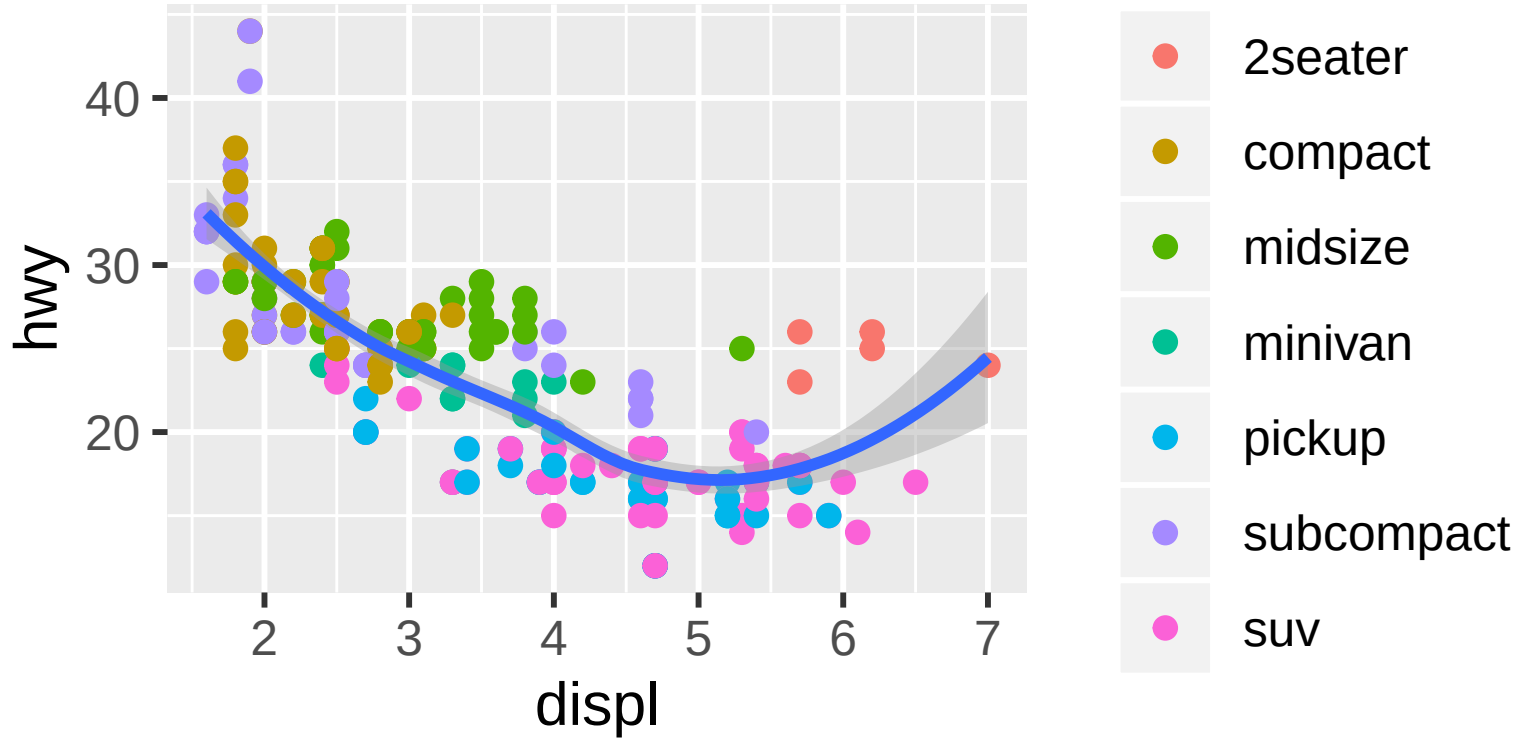
Multiple geom

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



Multiple geom

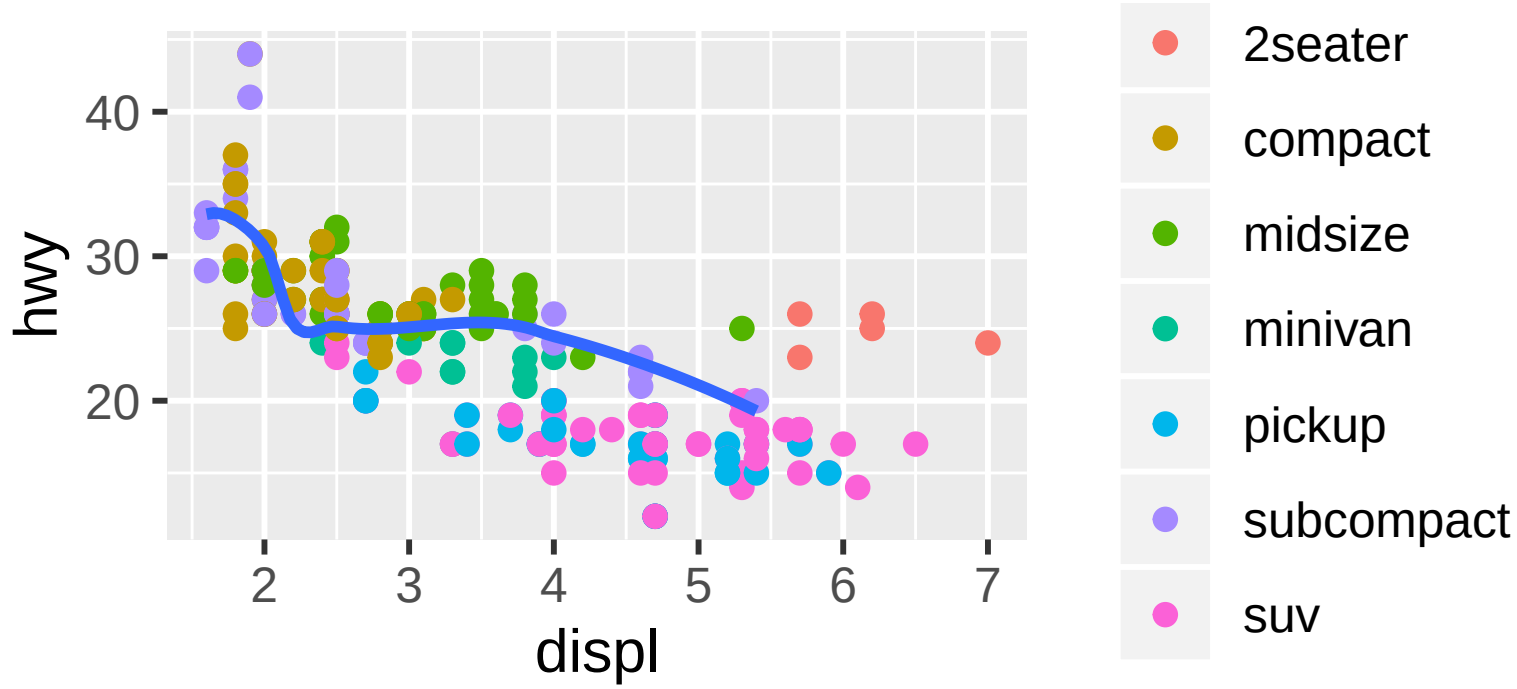
```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_point(mapping = aes(color = class)) +
  geom_smooth()
```



- Mappings in a geom function as local mappings for the layer.

Multiple geom with different data

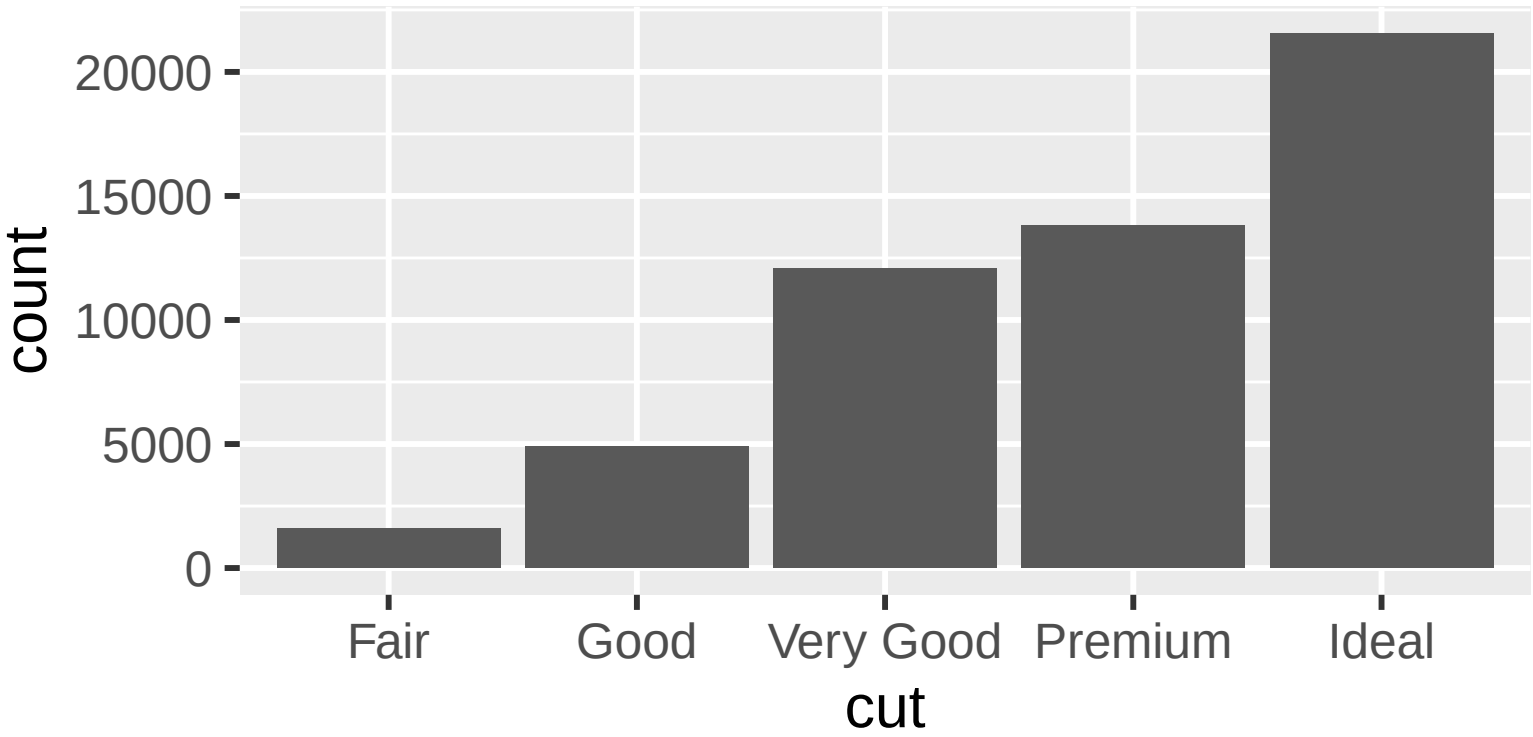
```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_point(mapping = aes(color = class)) +
  geom_smooth(
    data = filter(mpg, class == "subcompact"), se = FALSE  )
```



- Same idea to different data for each layer

Statistical Transformation

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



Statistical Transformation

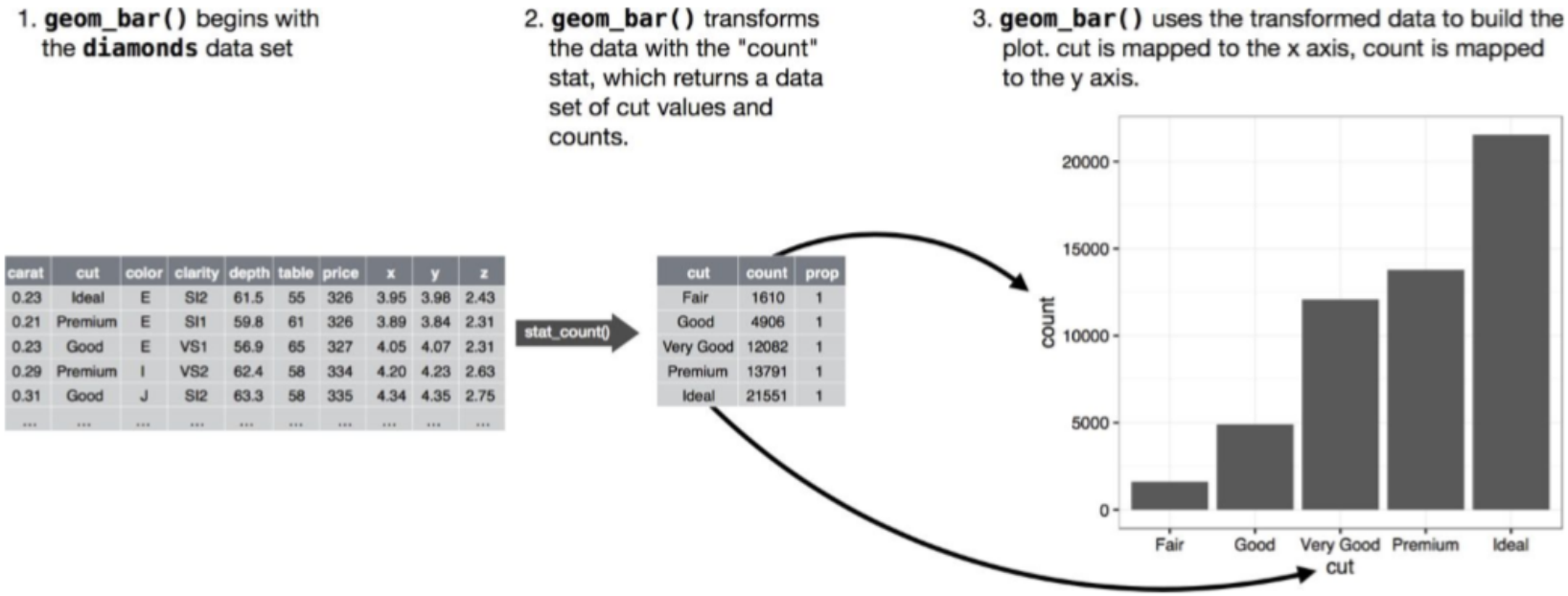
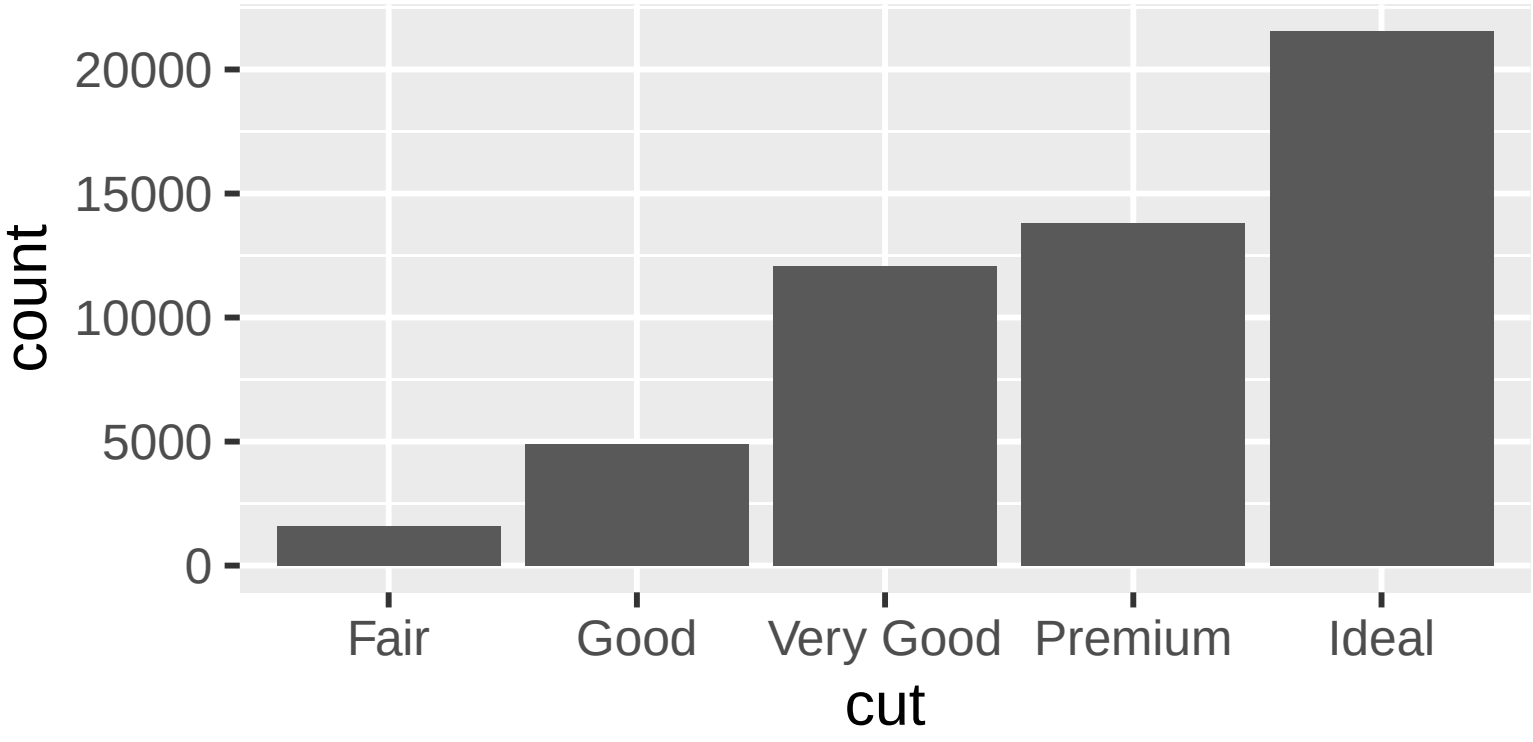


Figure 2: geom_bar()

Statistical Transformation (stat_count)

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



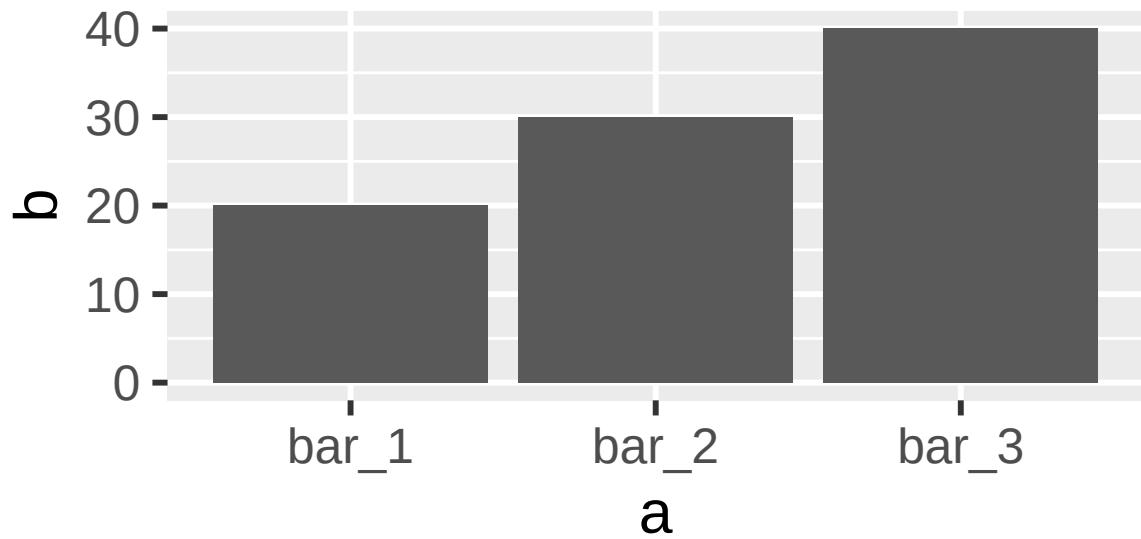
- Same as `geom_bar()`

Statistical Transformation

```
(demo <- tribble(~a, ~b, "bar_1", 20, "bar_2", 30, "bar_3", 40 ))
```

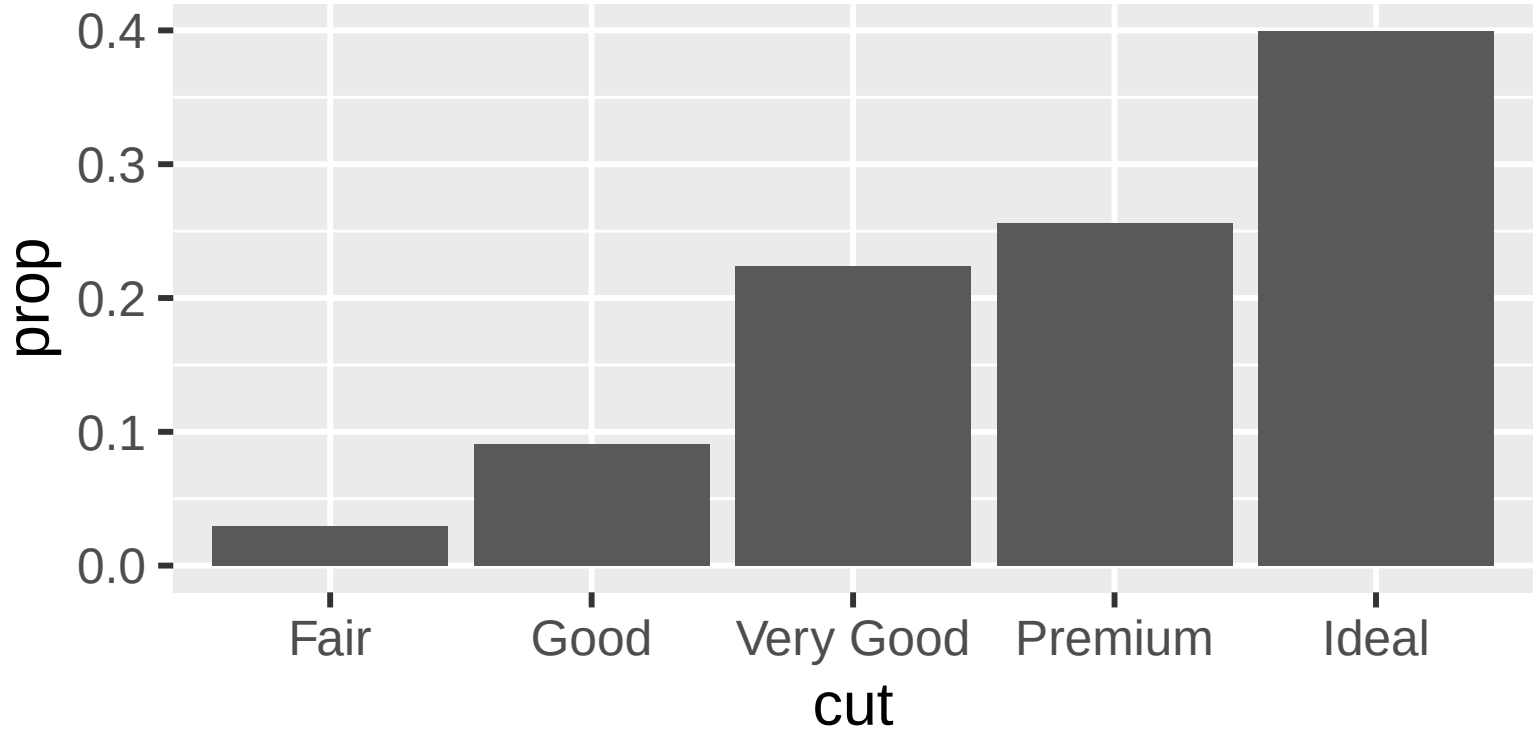
```
## # A tibble: 3 x 2
##   a           b
##   <chr> <dbl>
## 1 bar_1    20
## 2 bar_2    30
## 3 bar_3    40
```

```
ggplot(data = demo) +  geom_bar(
  mapping = aes(x = a, y = b), stat = "identity"  )
```



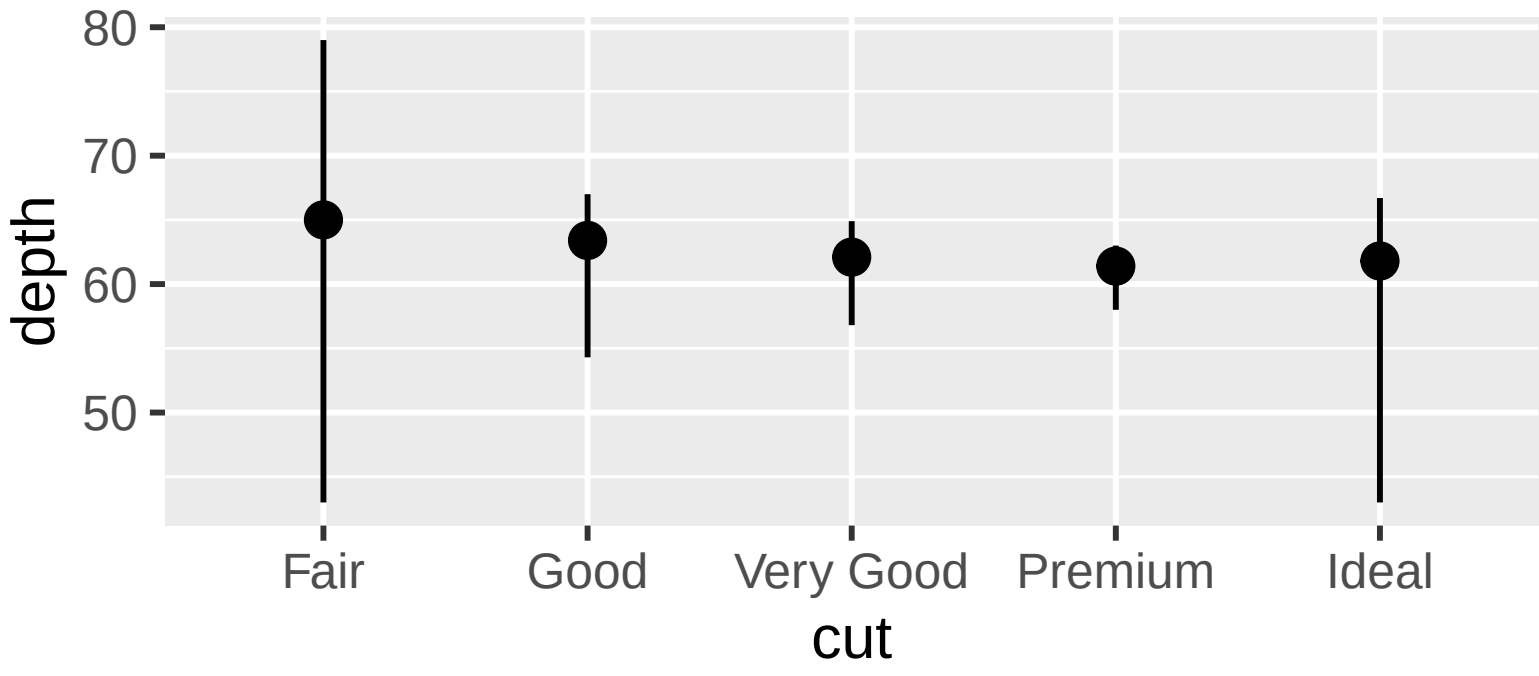
Statistical Transformation (proportion)

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



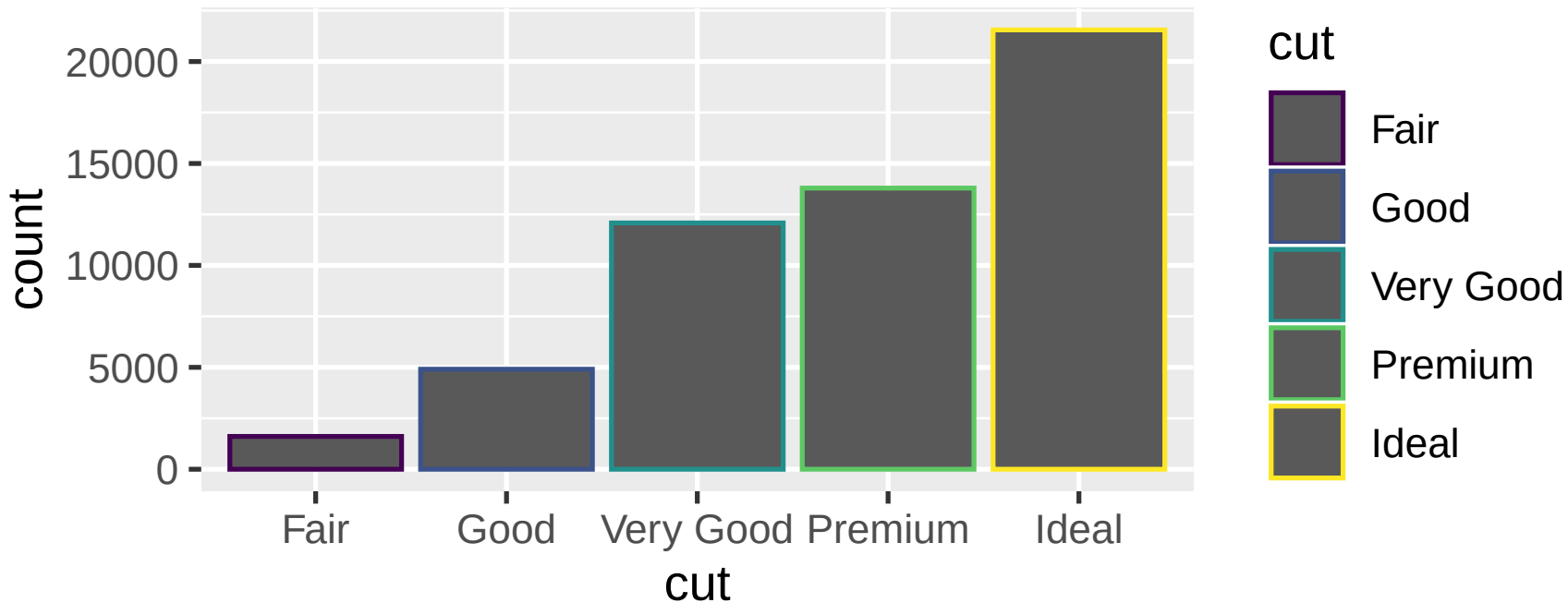
Statistical Transformation (`stat_summary()`)

```
ggplot(data = diamonds) +
  stat_summary(
    mapping = aes(x = cut, y = depth),
    fun.ymin = min,
    fun.ymax = max,
    fun.y = median
    )
```



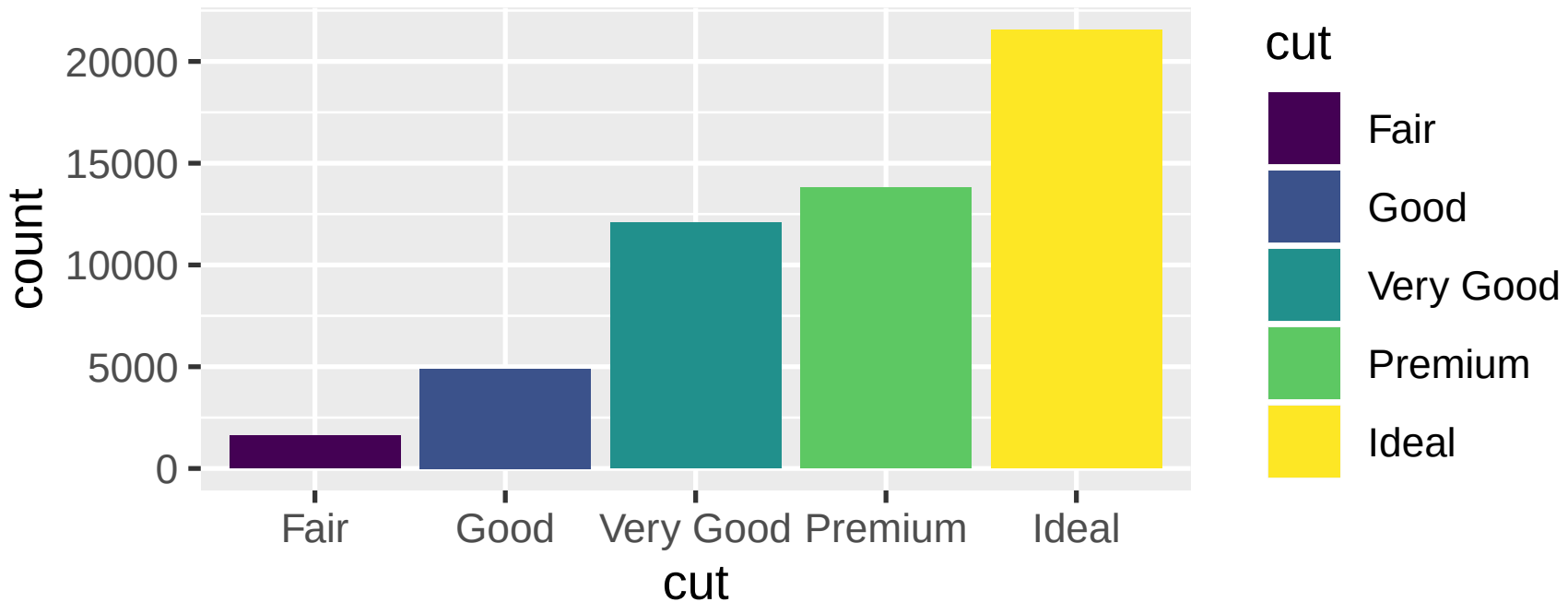
Position Adjustmnets (geom_bar())

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



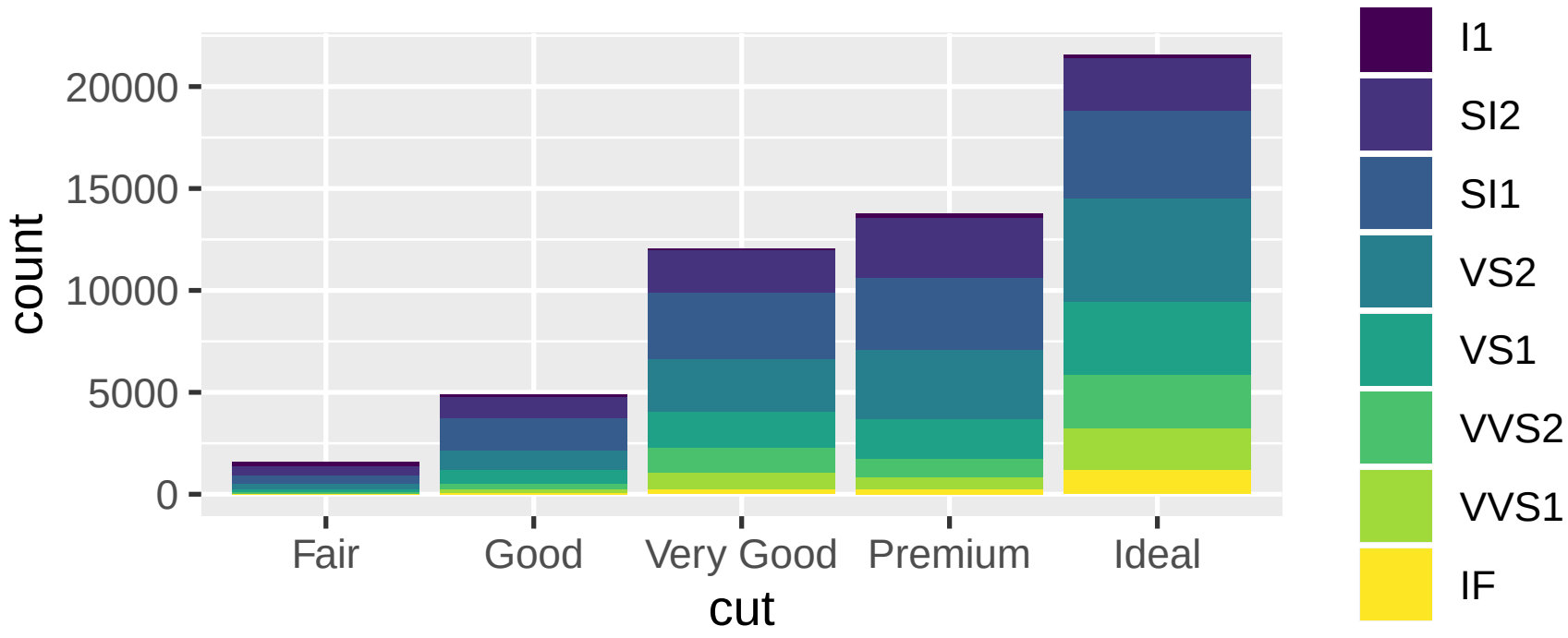
Position Adjustmnets (geom_bar())

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



Position Adjustmnets (geom_bar())

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

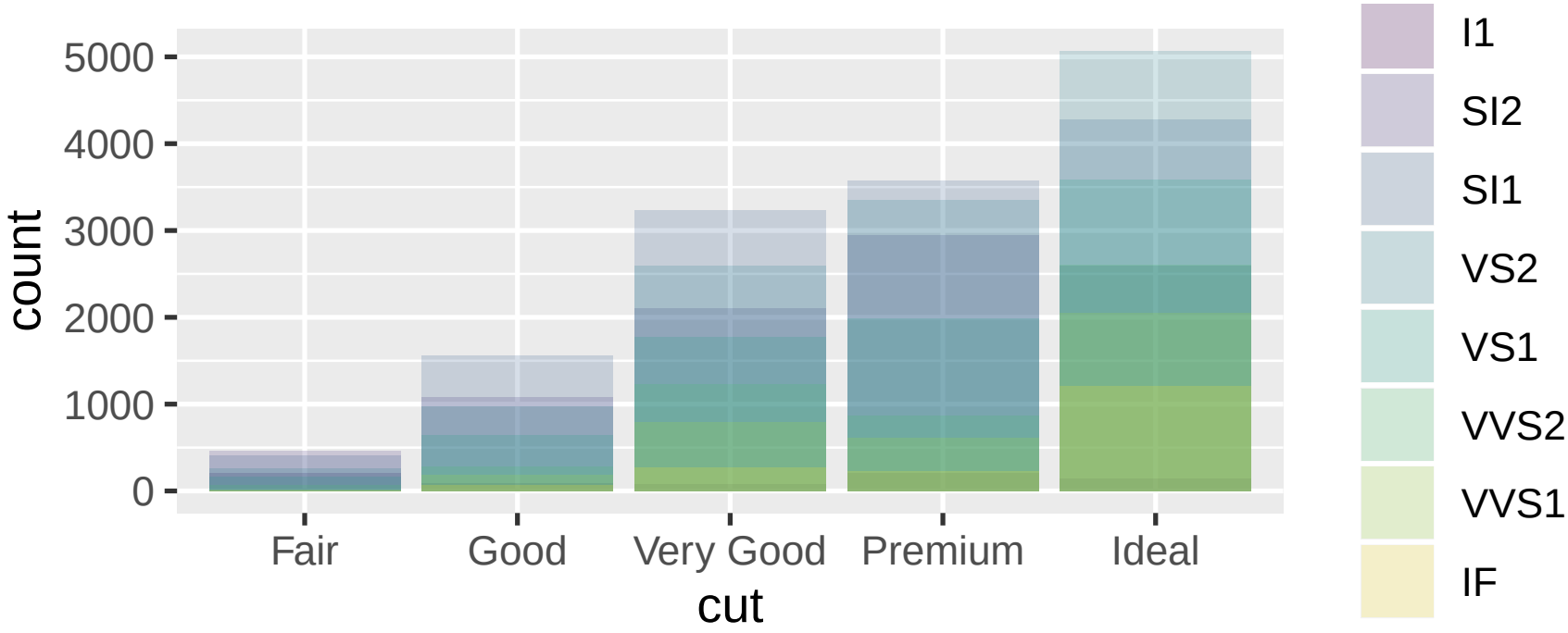


- each colored rectangle represents a combination of **cut** and **clarity**
- stacking is performed atuomatically by **position** argument.
- “identity”, “dodge”, “fill”

geom_bar() with position = "identity"

- place each object exactly where it falls in the context of the graph
- not useful in bars but uesful for 2D geoms like points

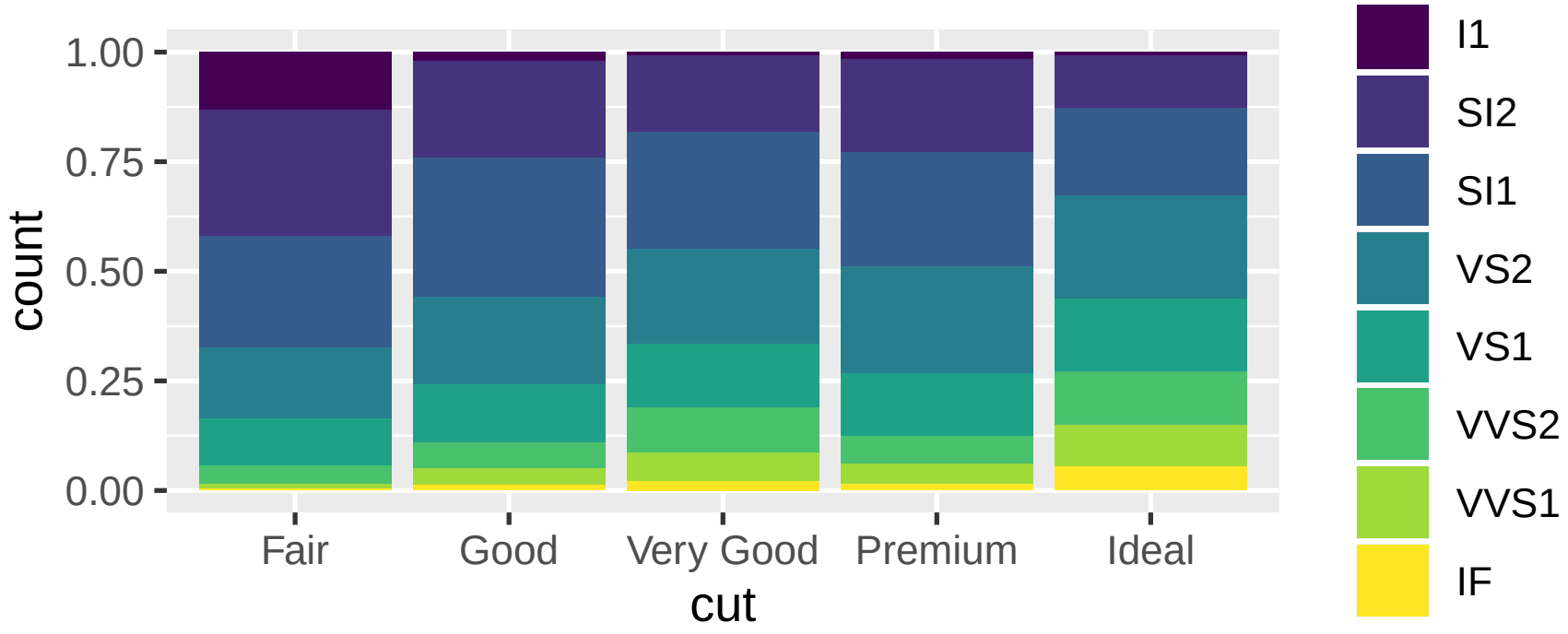
```
ggplot(  data = diamonds,
         mapping = aes(x = cut, fill = clarity) ) +
  geom_bar(alpha = 1/5, position = "identity")
```



- *alpha* : slightly transparent

geom_bar() with position = ""

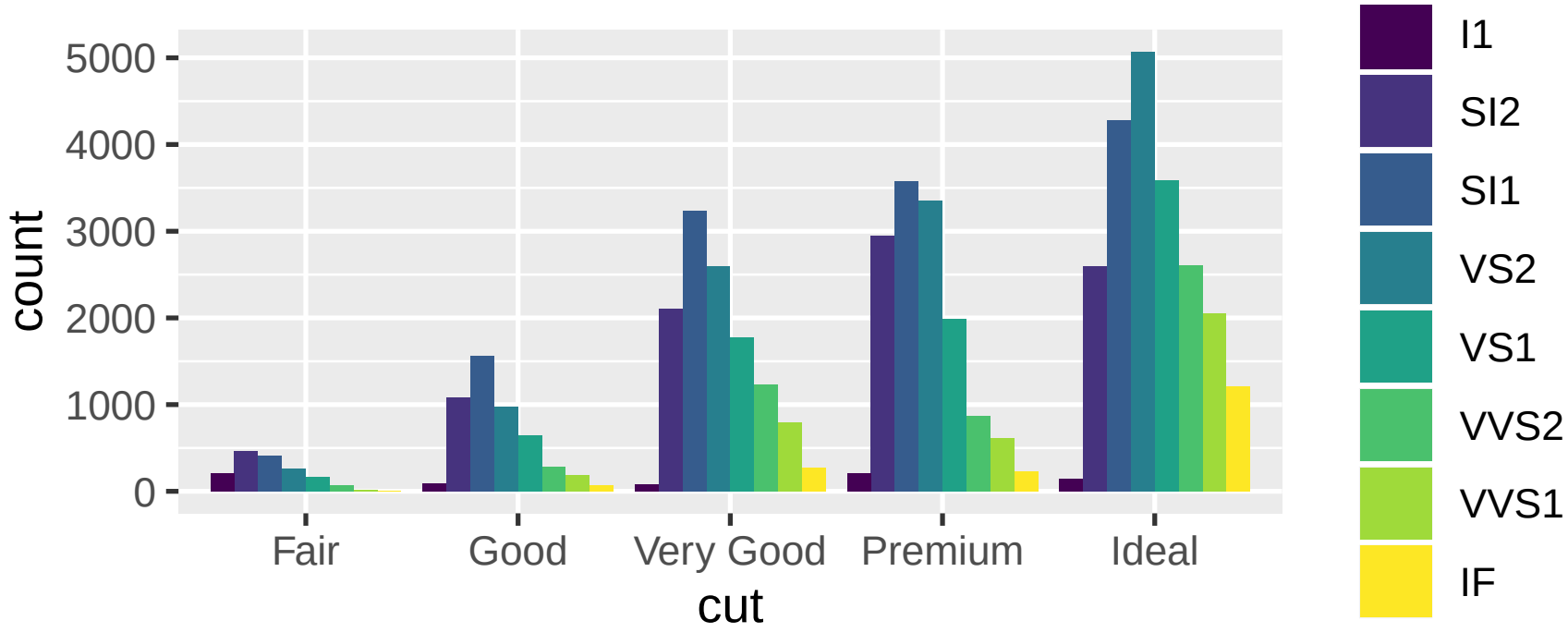
```
ggplot(data = diamonds) +  geom_bar(
  mapping = aes(x = cut, fill = clarity), position = "fill"  )
```



- works like stacking but makes each set of stackted bars the same height.
- easier to compare proportions across groups

geom_bar() with position="dodge"

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

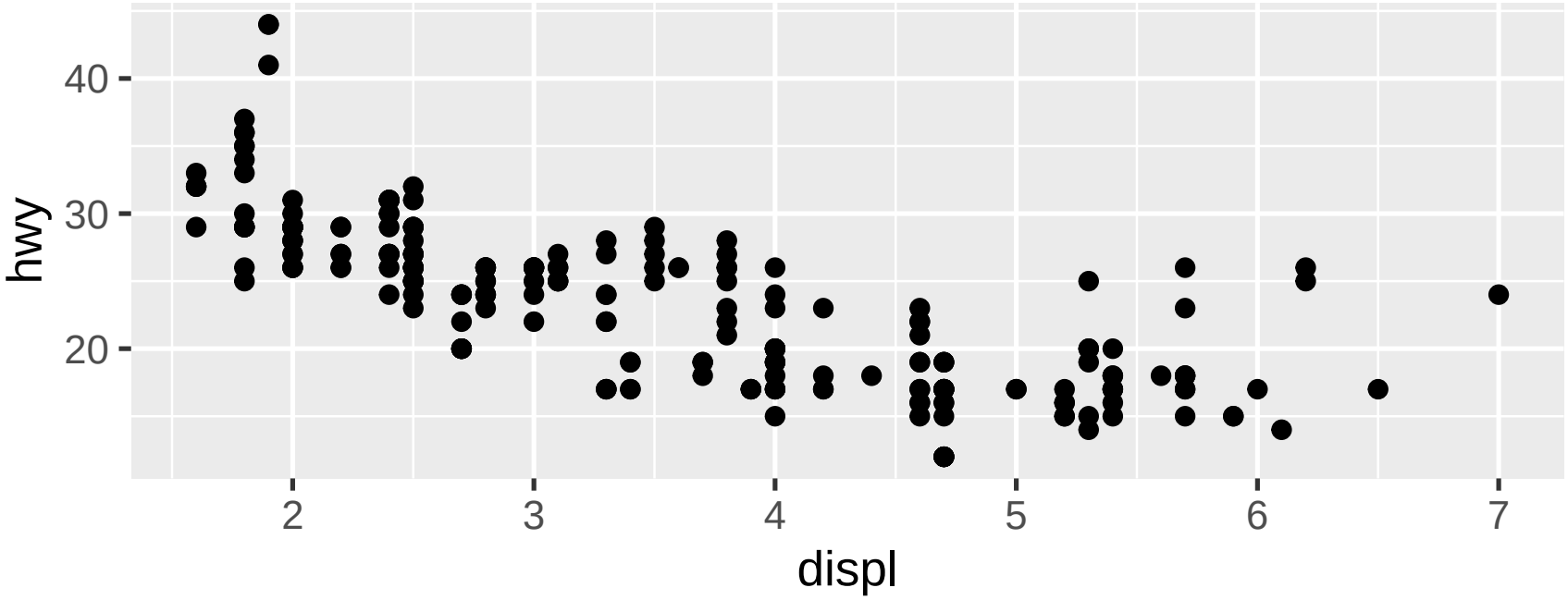


- places overlapping objects directly beside on another
- easier to compare individual values

Position Adjustmnets (geo_point)

- scatterplot

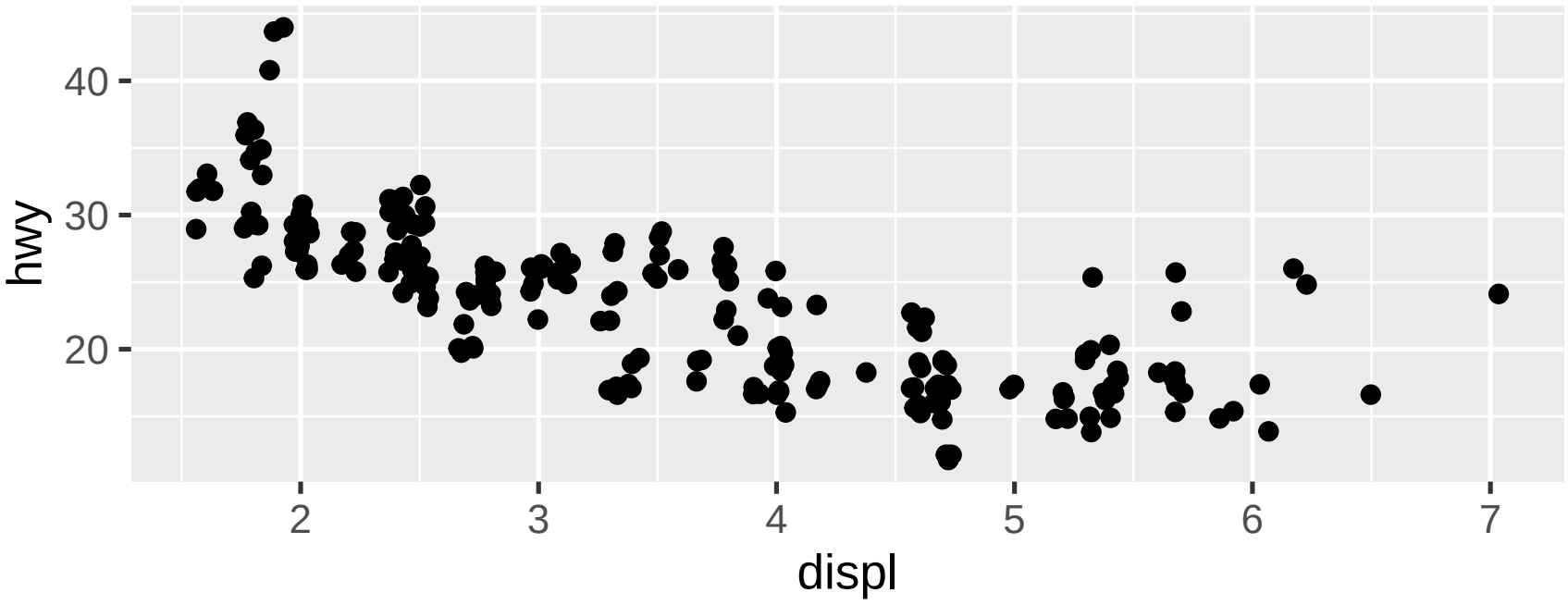
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy))
```



- display only 126 points, even though there are 234 obs
- overplotting (hwy, displ are rounded)

geom_point with position="jitter"

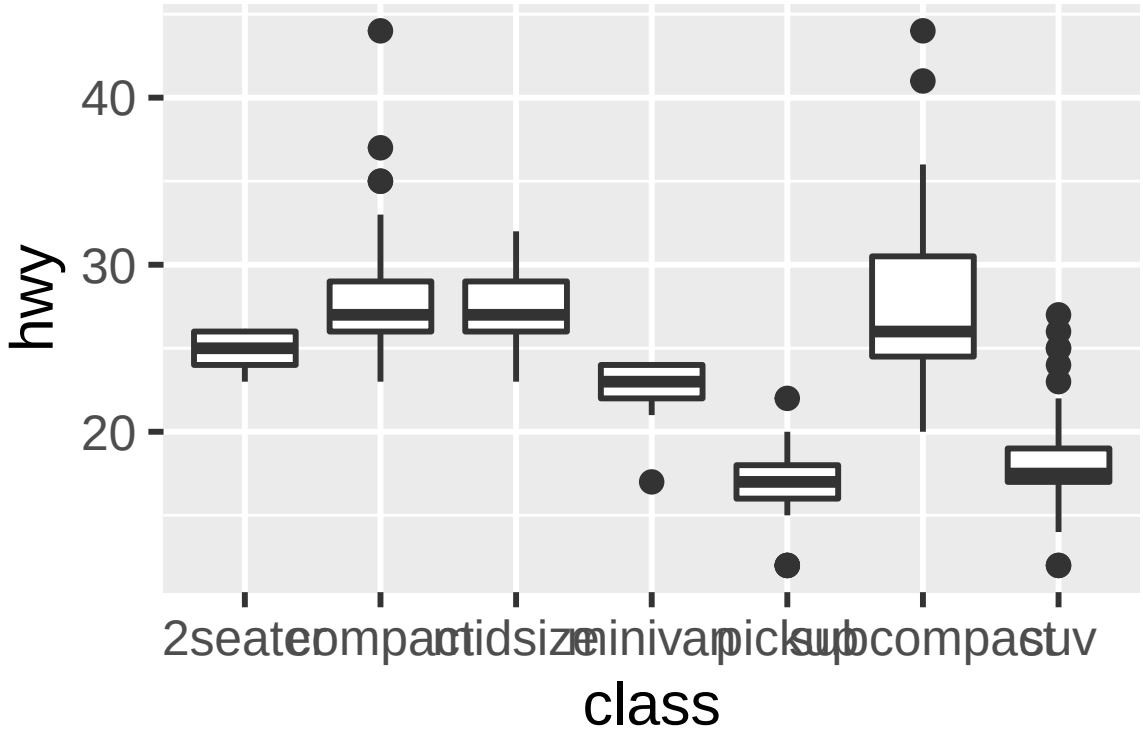
```
ggplot(data = mpg) +  geom_point(
  mapping = aes(x = displ, y = hwy),position = "jitter"  )
```



- adds a small amount of random noise to each point
- less accurate at small scales

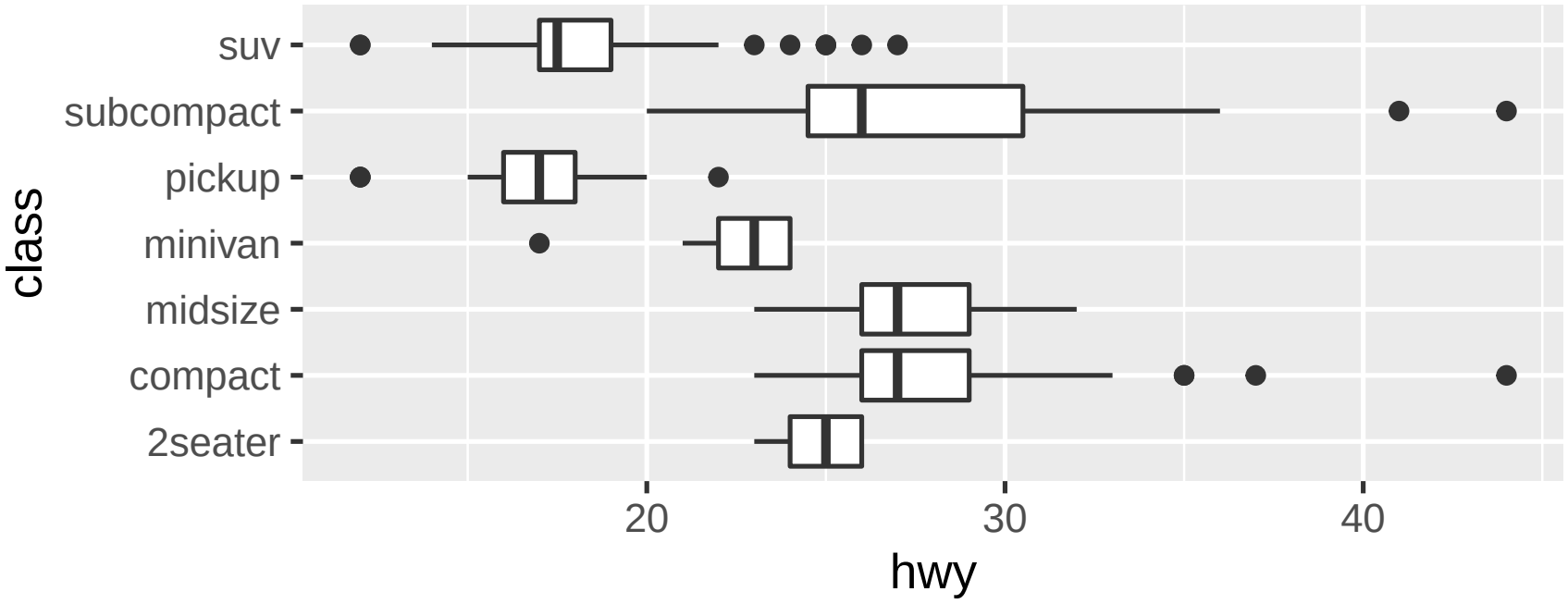
Coordinate Systems

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



coord_flip()

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



- switches the x- and y-
- useful for long label

coord_polar()

```
bar <- ggplot(data = diamonds) +
  geom_bar(
    mapping = aes(x = cut, fill = cut),
    show.legend = FALSE,
    width = 1  ) +
  theme(aspect.ratio = 1) +
  labs(x = NULL, y = NULL)
```

coord_flip() and coord_polar()

```
bar + coord_flip()
bar + coord_polar()
```

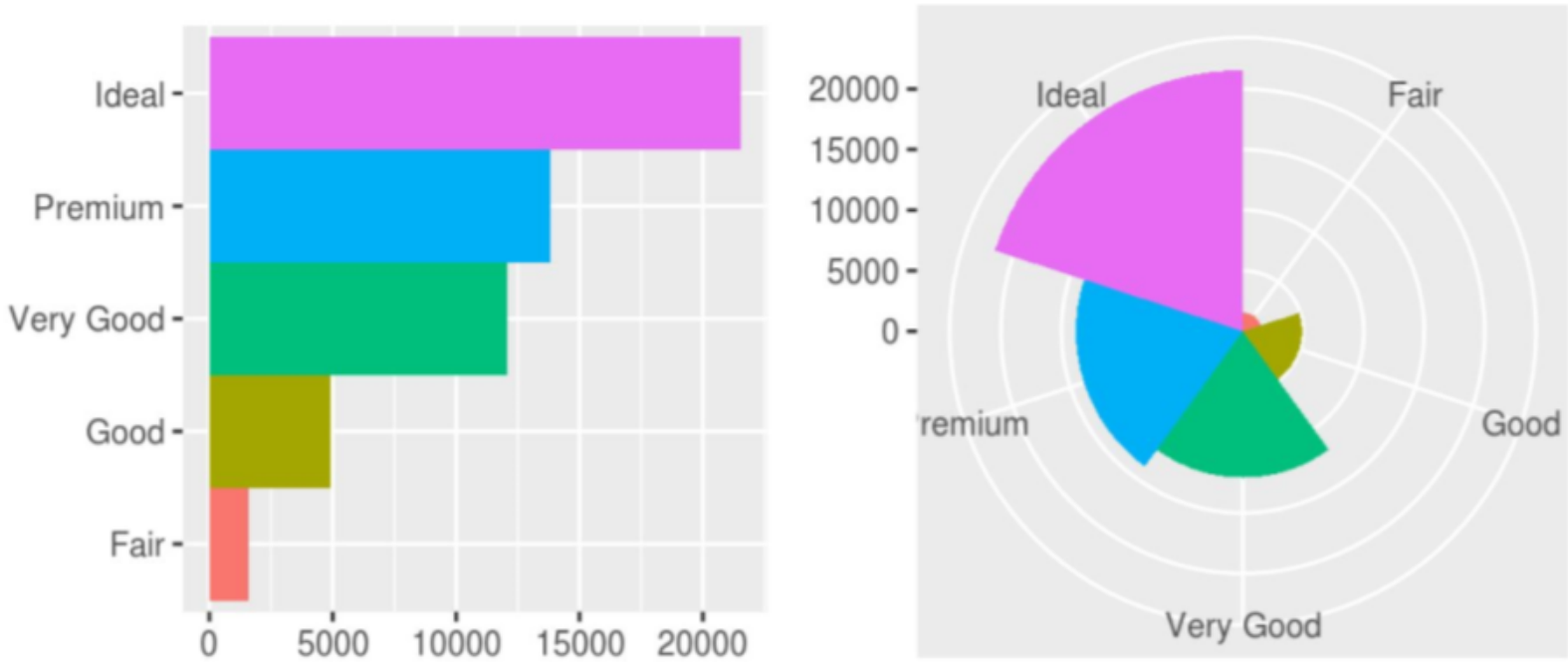


Figure 3: coord_flip(), coord_polar()

Reference

- Wickham, H. and Grolemund, G. (2017) R for Data Science, O'reilly Media Inc., Chapter 1.