More ggplot2: Facets, Geoms, Stats, Positioning and Coords

Group 6

April 10, 2019

```
library(tidyverse)
library(gridExtra)
library(maps)
library(mapproj)
rm(list = ls())
```

Group tasks

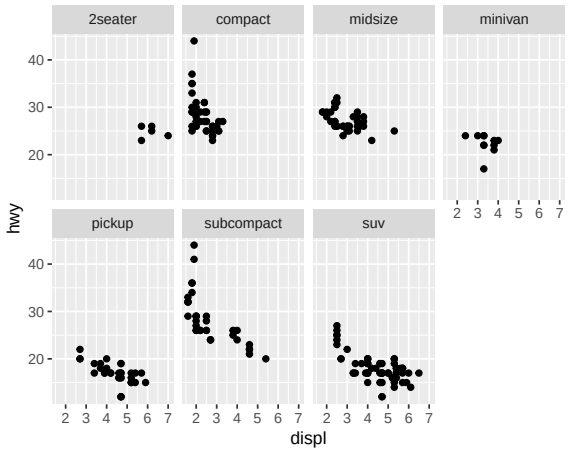
Section	Group member
3.5	Martin
3.6	Lorne
3.7	Chenyang
3.8	Finn
3.9	Sam
3.10	Emerson

R4DS 3.5 Facets (Martin)

3.5 Summary

A way to add additional variables to a plot is by splitting it into facets. Facets are panels that display one subset of the data. Itis a useful tool for categorical variables. There are two alternatives: facet a plot by a single variable using facet_wrap() function and in the first argument include ~ followed by the variable name (should be discrete). The second case arise if we want to facet the plot on a combination of 2 discrete variables, we will use facet_grid() function and inside the 2 variables names separated by ~. For example:

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

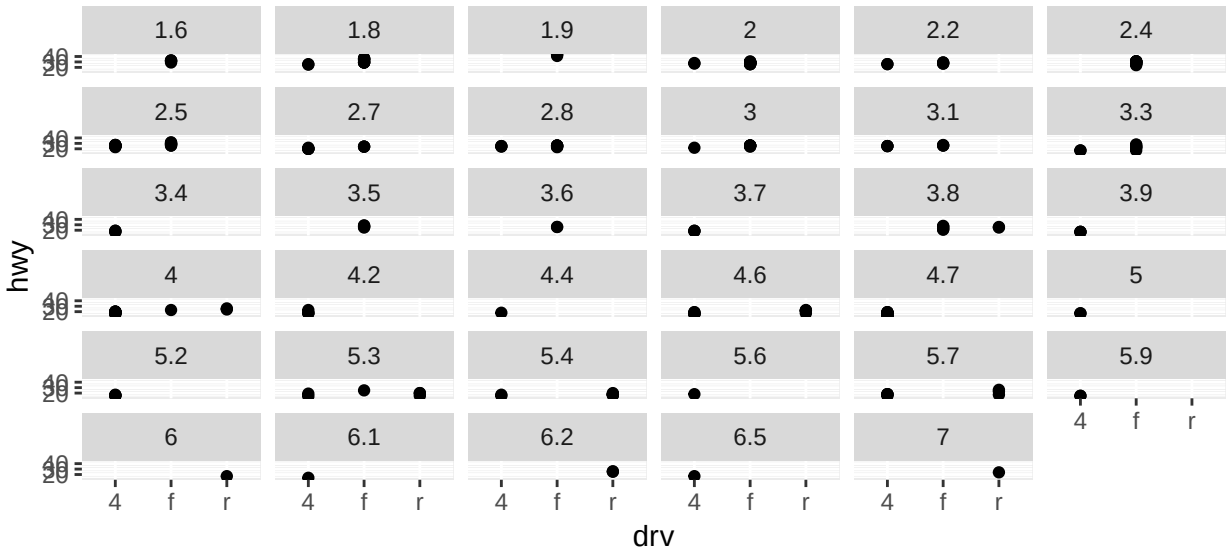


3.5.1 Exercises

a

It would create a panel per value of the continuous variable, discretizing the variable. Ex:

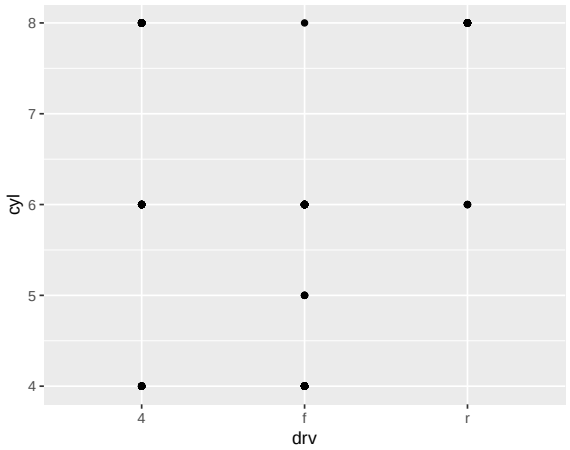
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = drv, y = hwy)) +
  facet_wrap(~displ)
```



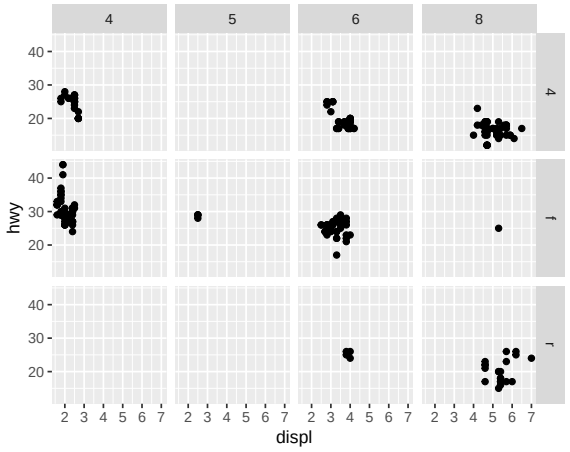
b

Means that these combinations of the levels between these two factors are not presented in the data. The plot of the question shows exactly which combinations of the levels of these two variables are presented in the dataframe and which ones not.

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



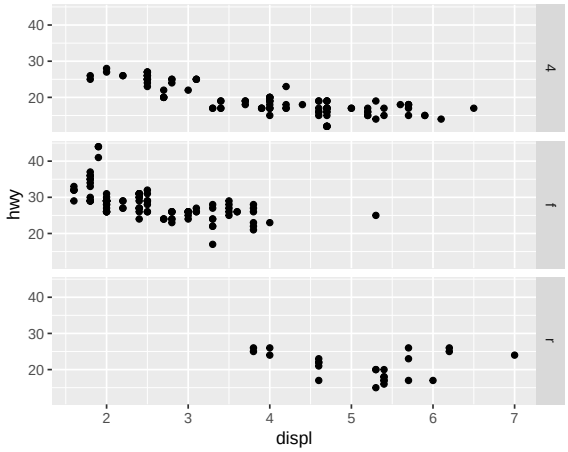

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  facet_grid(drv ~ cyl)
```



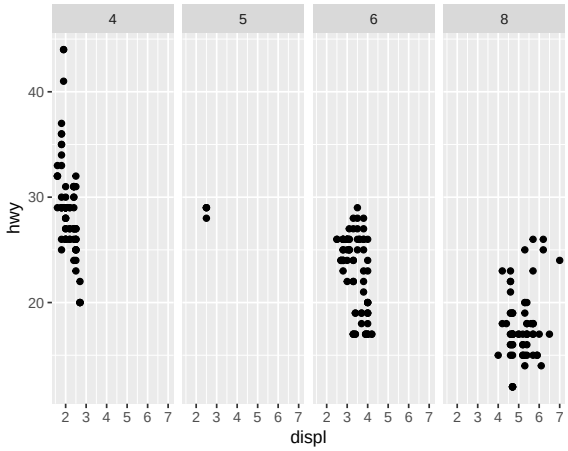
c

It plots a graph for each of the levels of the drv and for the cyl. The only difference is that the legend indicating the level of the variable, in the first case is on the left side and in the later case on the top. Finally, the use of facet_grid(. ~ cyl) is the same as the facet_wrap(~cyl), just change how arrange the panels.

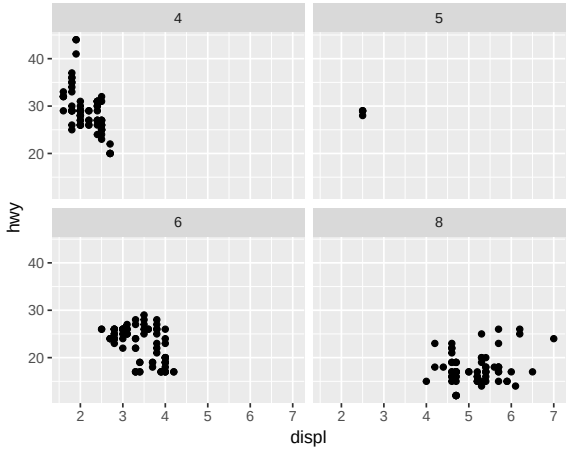
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  facet_grid(drv ~ .)
```




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

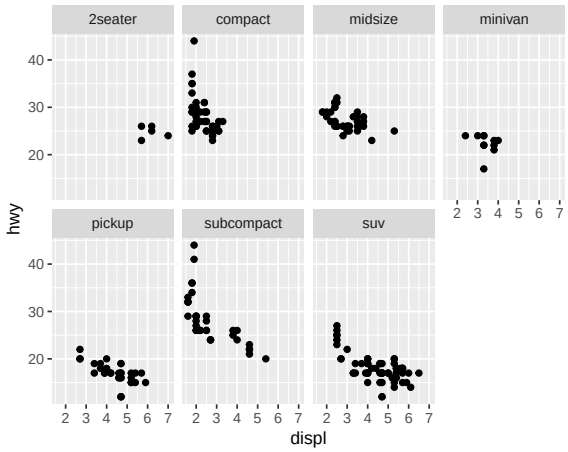



```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  facet_wrap(~cyl)
```



d

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



Faceting allows to see more clearly the relations between 2 variables and the discrete variable, in particular, when the discrete variable has too many levels. In this case, using colors can be have many overlapping and makes difficult to see the relationship between the three variables. On the other hand, if you have few categories and observations on the discrete data we are faceting, color might be easier to read, especially if the labels of the facet are not clear and easier to interpret.Finally, in a larger data set, it might depends on how many categories of the discrete variable we have and how many observations in each category: if we have many observations in each category (does not matter if we have few or several categories), facets may be clearer as we will avoid overlap. But if we have many categories but few observations in each one, facets might be not that necessary and colors may be better.

e

nrow and ncol control the numbers of rows and columns in facet_wrap (for facet_grid nrow and ncol are controlled by the levels of each variable). Another layout option for facet_wrap is dir, which controls the placement of the individual panels.

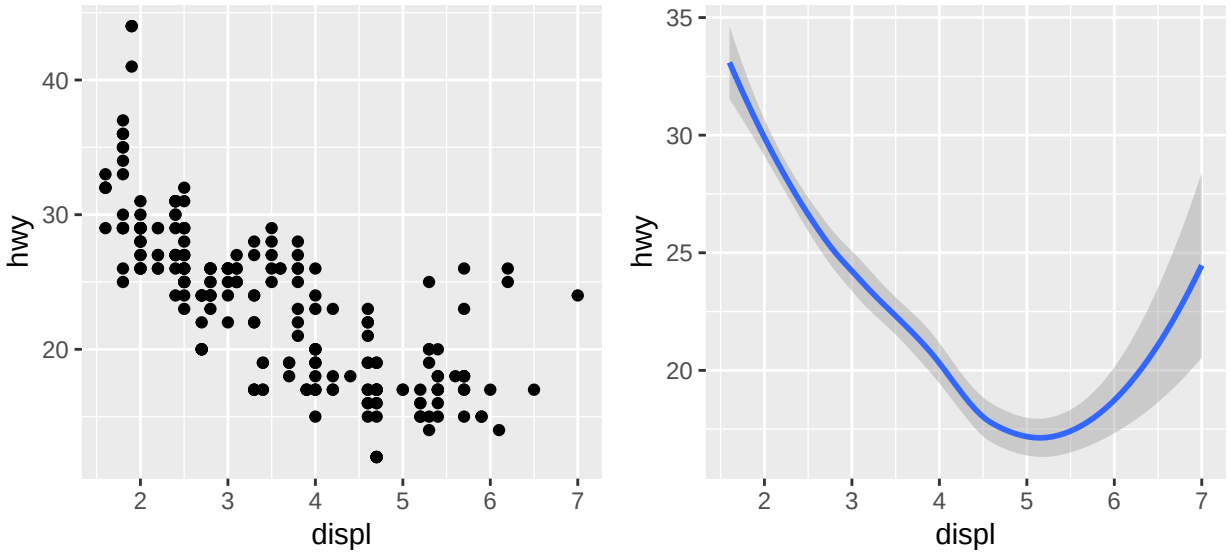
f

Screens generally are larger than taller.

section 3.6 Geometric objects (Lorne)

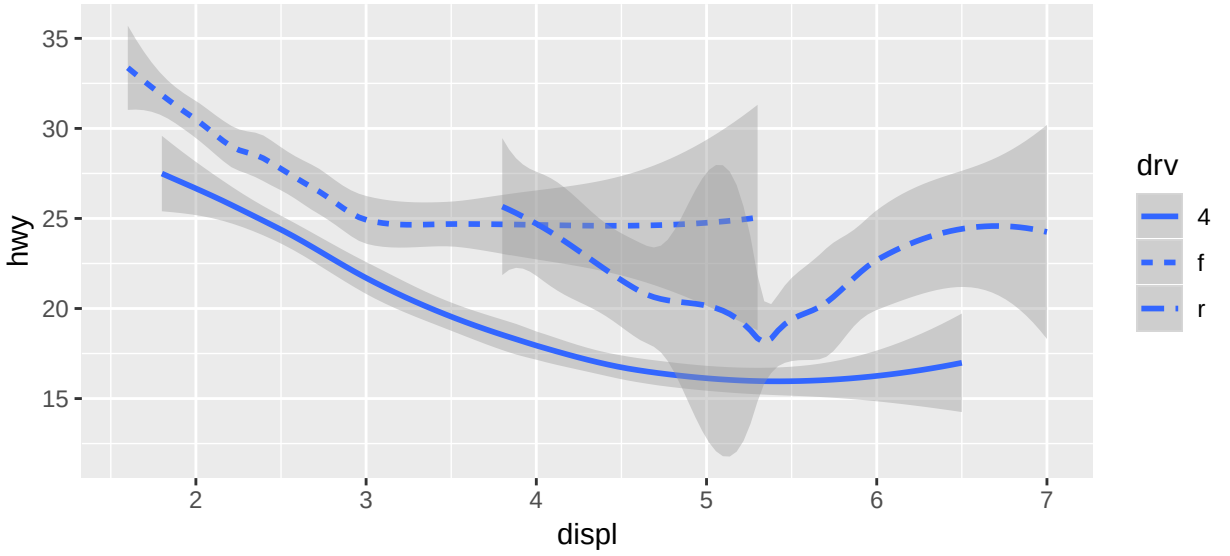
Geoms characterize plots by the way they represent data, as in boxplots use boxplot geoms. Change the geom in the plot by changing the geom:

```
a <- ggplot(mpg) +
   geom_point(aes(displ, hwy))
b <- ggplot(mpg) +
   geom_smooth(aes(displ, hwy))
grid.arrange(a,b, nrow = 1)
```



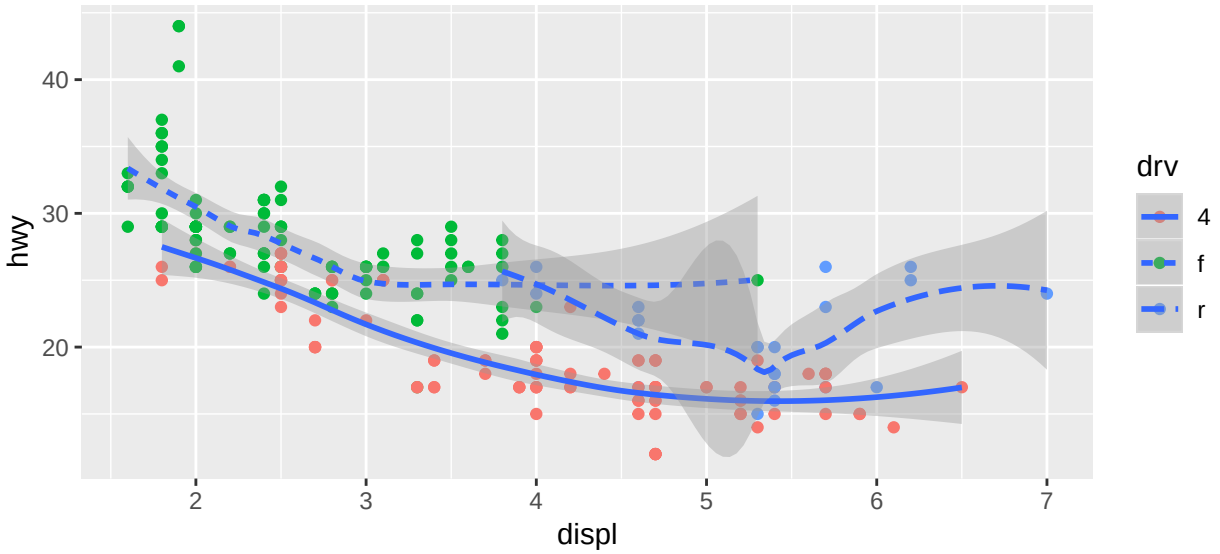
different aesthetics for different geoms

```
ggplot(mpg) +
  geom_smooth(aes(displ, hwy, linetype = drv))
```



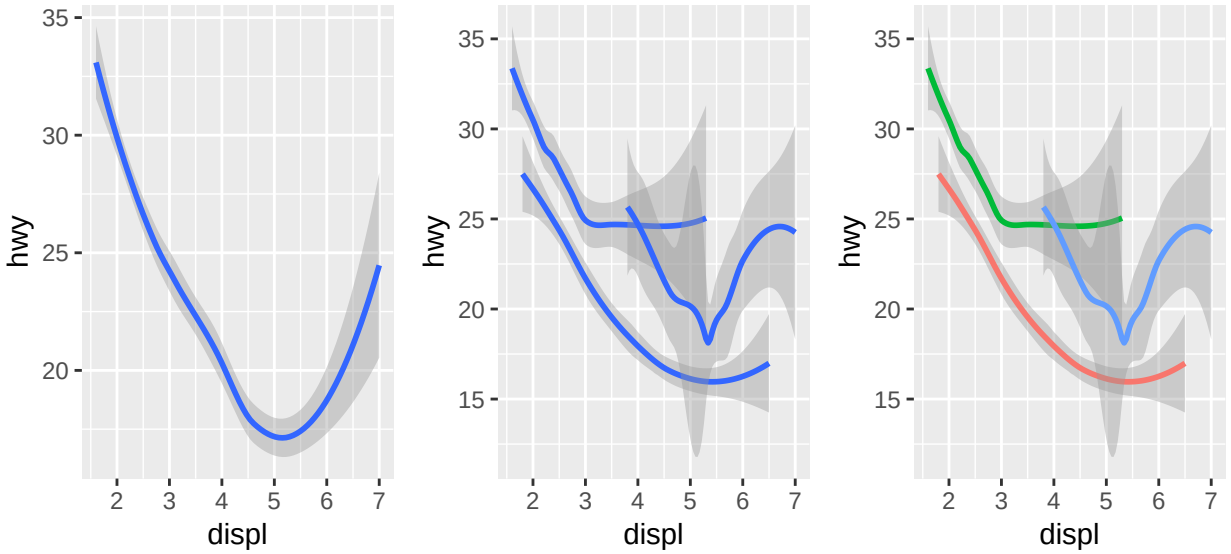
combining geoms to show both raw and smoothed data

```
ggplot(mpg, aes(displ, hwy)) +
  geom_point(aes(color = drv)) +
  geom_smooth(aes(linetype = drv))
```



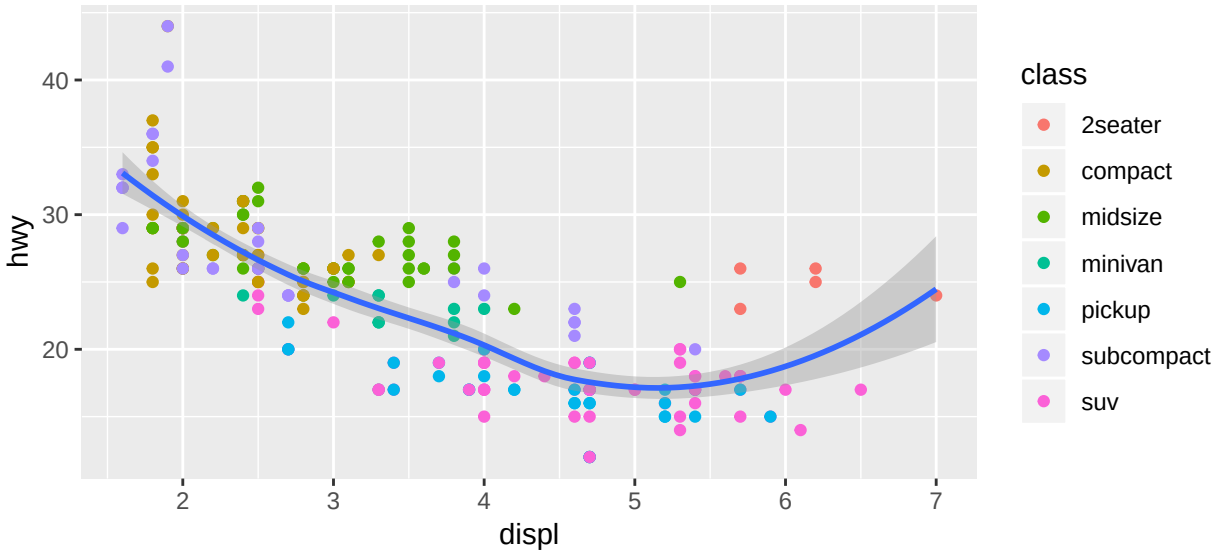
You can use the `group` aesthetic, but other aesthetics will do this and assign levels.

```
gg <- ggplot(mpg, aes(displ, hwy))
c <- gg + geom_smooth()
d <- gg + geom_smooth(aes(group = drv))
e <- gg + geom_smooth(aes(color = drv), show.legend = FALSE)
grid.arrange(c, d, e, nrow = 1)
```



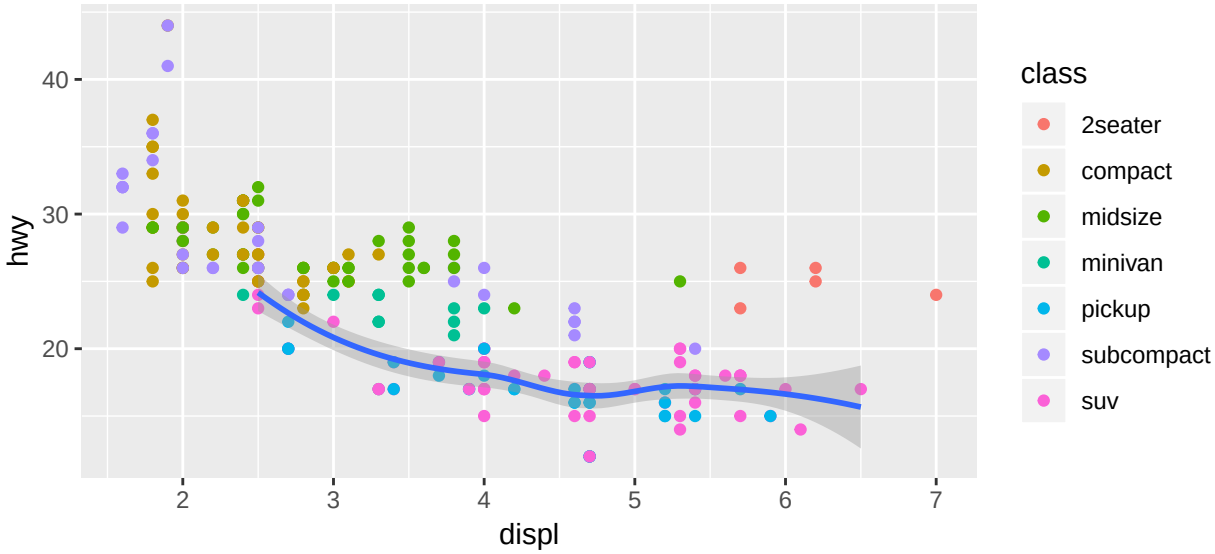
Save mappings duplicated across geoms to ggplot, then add geom-specific aesthetics to the geom.

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



Keeping going with this, we can display different subsets of the data in different geoms.

```
ggplot(mpg, aes(displ, hwy)) +
  geom_point(aes(color = class)) +
  geom_smooth(data = filter(mpg, class == "suv"))
```



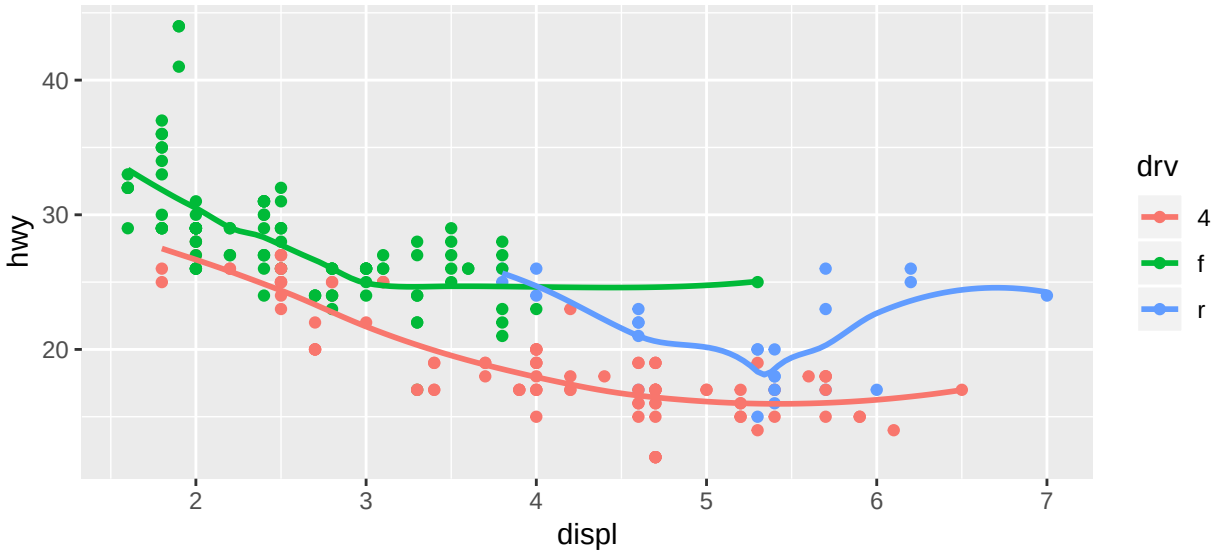
3.6.1 exercises

1. what geom would you use to draw a line chart? Answer: `geom_line()`.

A boxplot? Answer: `geom_boxplot()`.
A histogram? Answer: `geom_hist()`.
An area chart? Answer: `geom_area()`

2. Predict, then run the code below. Answer: we'll get color levels assigned to each of the 3 unique values in `drv` both in points and in separate smoothed lines.

```
ggplot(mpg, aes(displ, 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?

Answers: It removes the legend that would otherwise accompany the mapping of variables to aesthetics other than `x` and `y`. Removing it gives the graph more room in the allotted space, as in the triplet comparing `group` and other aesthetics.

4. What does the se argument do? Answer: it determines whether standard error bands are shown.

5. Will these 2 graphs look different?

Answer: Nope, same data, same aesthetics.

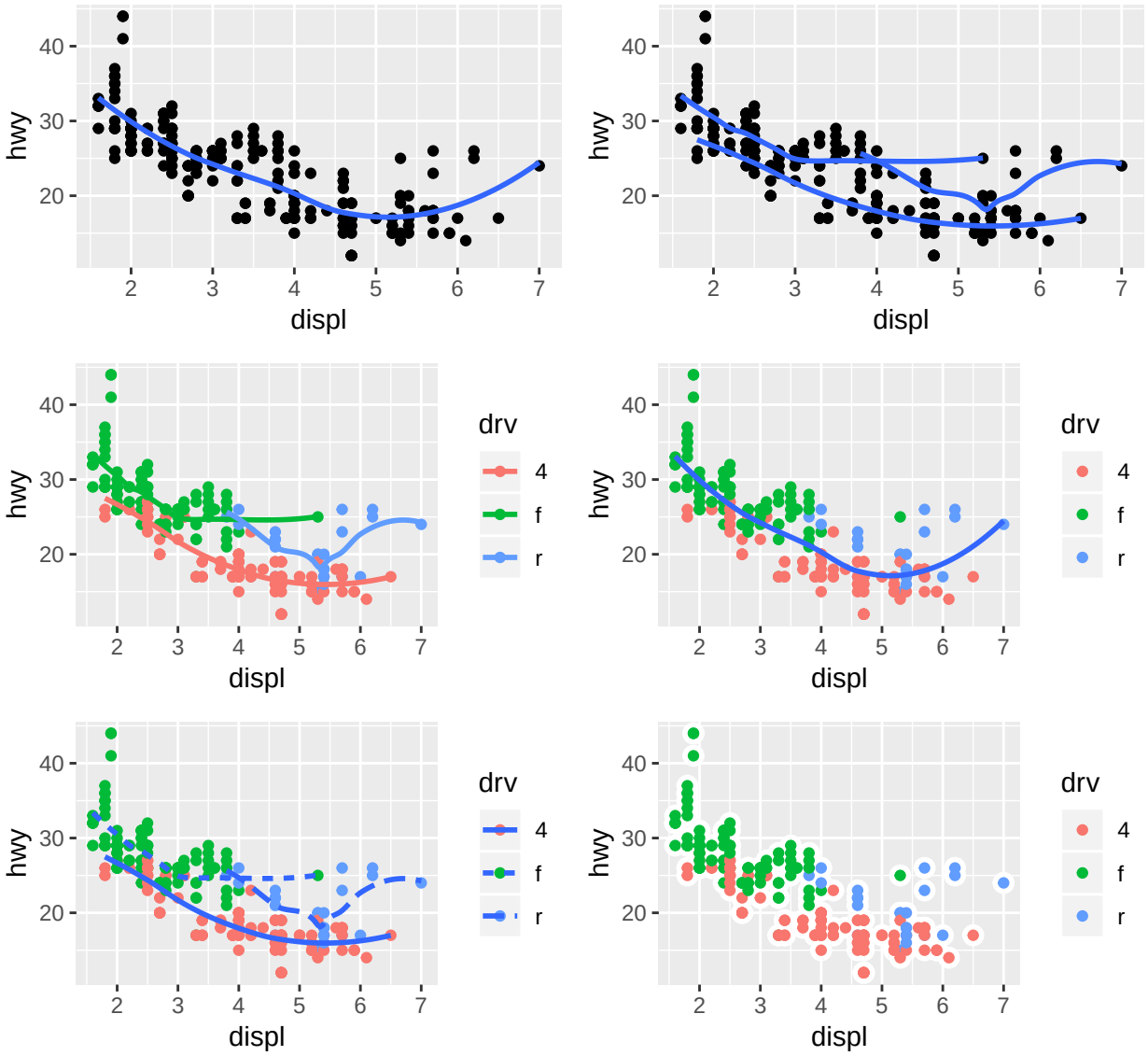
```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_point() +
  geom_smooth()

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

6. recreate the R code necessary for the displaye plots.

Answer:

```
gg <- ggplot(mpg, aes(displ, hwy))
e <- gg +
  geom_point() +
  geom_smooth(se = FALSE)
```

```
f <- gg +
  geom_point() +
  geom_smooth(aes(group = drv), se = FALSE)
g <- gg +
  geom_point(aes(color = drv)) +
  geom_smooth(aes(color = drv), se = FALSE)
h <- gg +
  geom_point(aes(color = drv)) +
  geom_smooth(se = FALSE)
i <- gg +
  geom_point(aes(color = drv)) +
  geom_smooth(aes(linetype = drv), se = FALSE)
j <- gg +
  geom_point(color = "white", stroke = 2) +
  geom_point(aes(color = drv))

grid.arrange(e, f, g, h, i, j, nrow = 3)
```

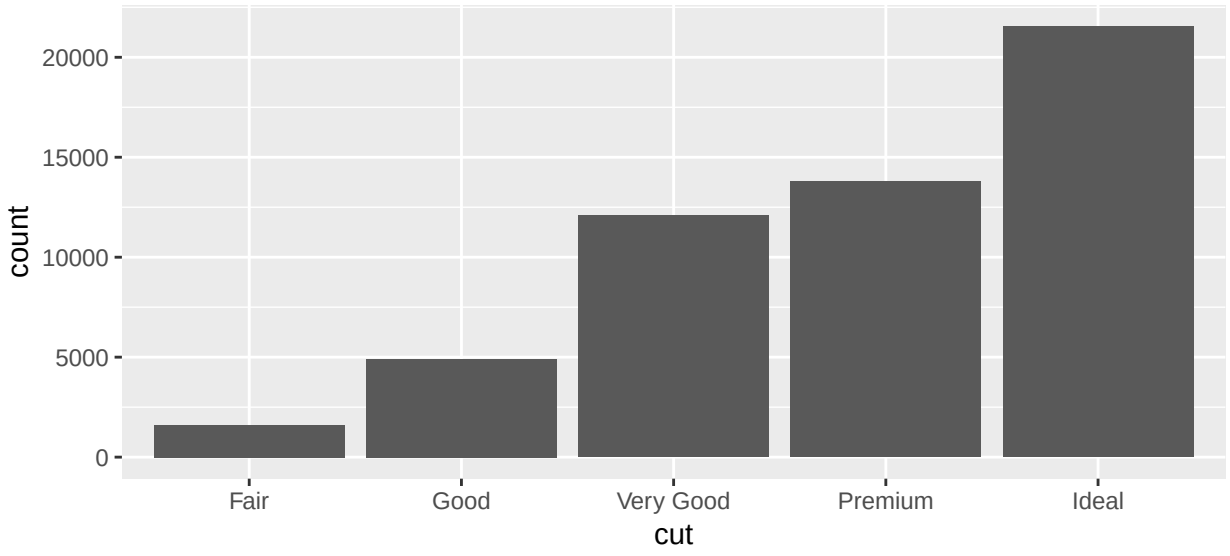

hwy
displ
drv
4
f
r

section 3.7 Statistical transformations (Chenyang)

###Summary: Many graphs, like scatterplots, plot the raw values of your dataset. Other graphs, like bar charts, calculate new values to plot: (1) bar charts, histograms, and frequency polygons bin your data and then plot bin counts, the number of points that fall in each bin. (2) smoothers fit a model to your data and then plot predictions from the model. (3) boxplots compute a robust summary of the distribution and then display a specially formatted box.

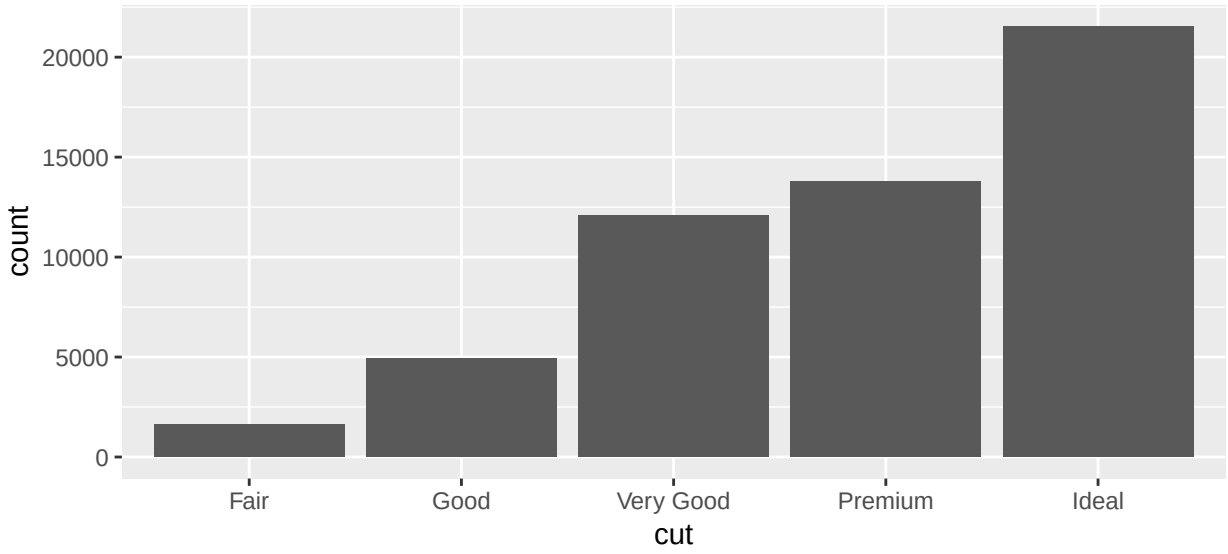
The following chart displays the total number of diamonds in the `diamonds` dataset, grouped by `cut`.On the x-axis, the chart displays `cut`, a variable from `diamonds`. On the y-axis, it displays count, but count is not a variable in diamonds

```
library(ggplot2)
library(tidyverse)
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut))
```



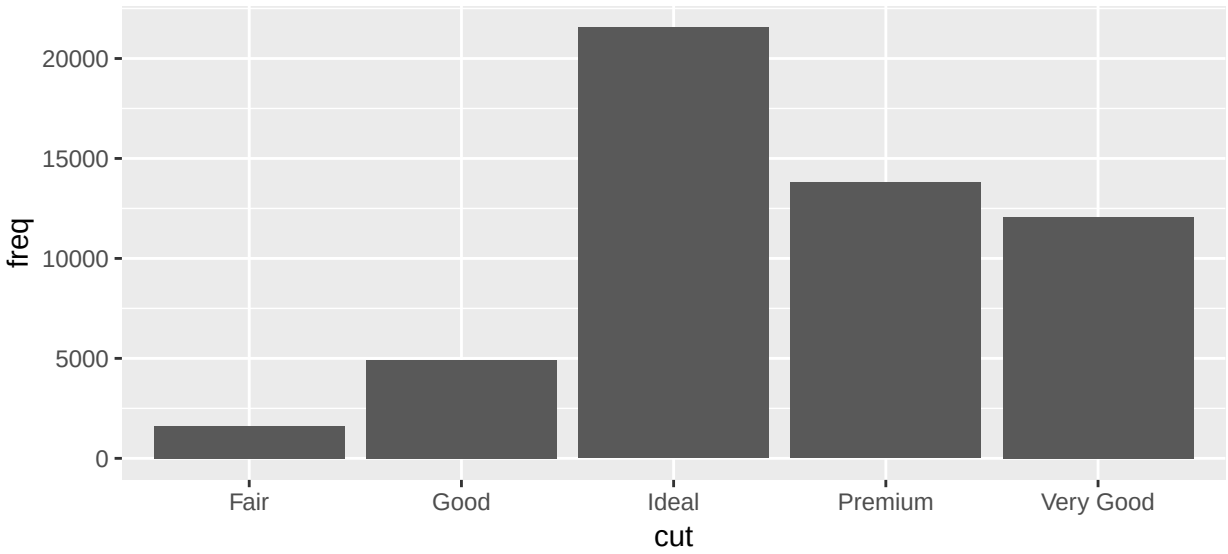
You can generally use geoms and stats interchangeably. For example, you can recreate the previous plot using `stat_count()` instead of `geom_bar()`

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



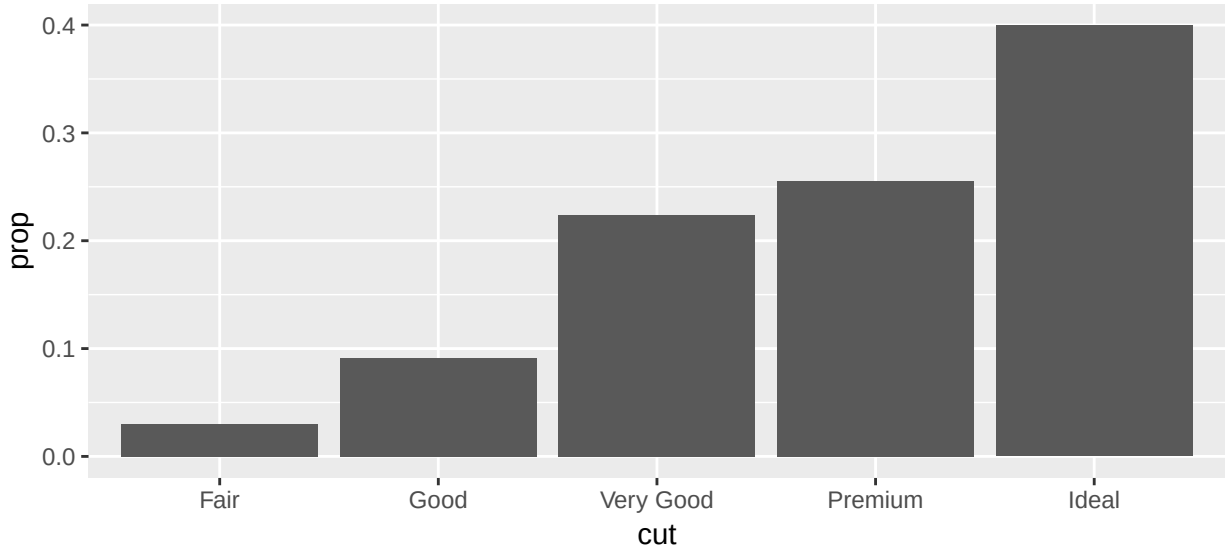
Change the stat of `geom_bar()` from count (the default) to identity. This lets me map the height of the bars to the raw values of a y variable.

```
demo <- tribble(
  ~cut,          ~freq,
  "Fair",        1610,
  "Good",        4906,
  "Very Good",   12082,
  "Premium",     13791,
  "Ideal",       21551
)

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



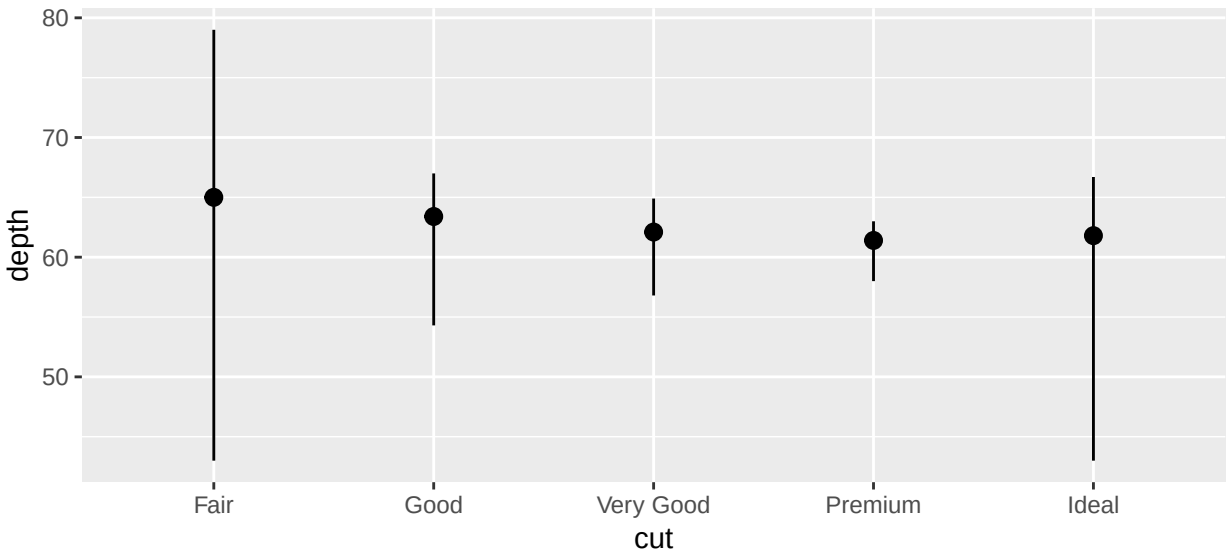
Display a bar chart of proportion, rather than count

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



Use `stat_summary()`, which summarises the y values for each unique x value

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

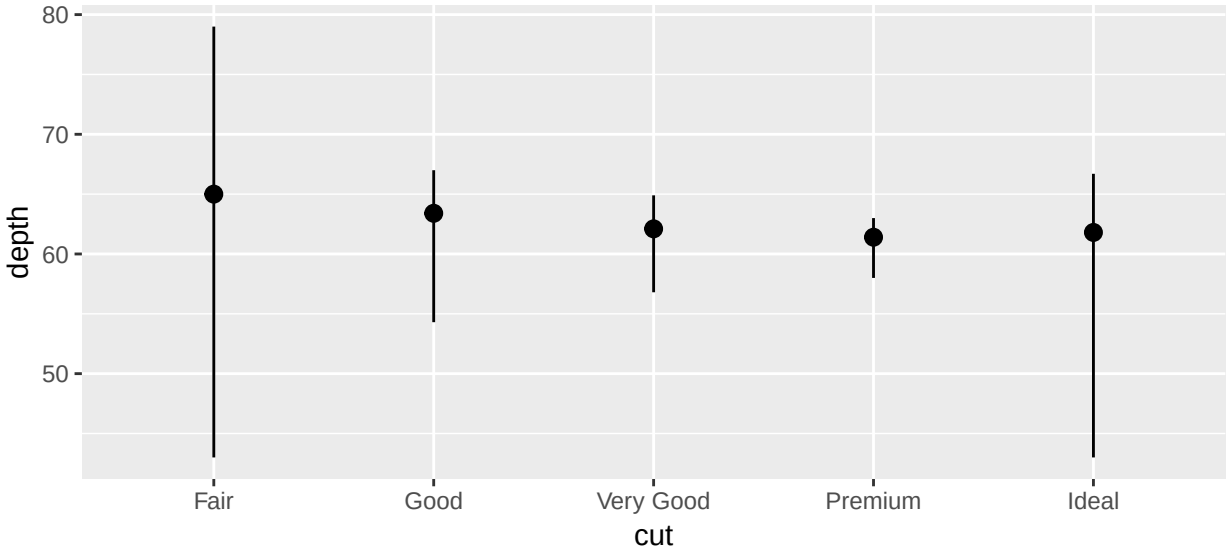


3.7.1 exercises

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

Answers: The default geom for `stat_summary()` is geom_pointrange(https://ggplot2.tidyverse.org/reference/stat_summary.html)

```
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()?

Answers: The default stat for `geom_col()` is `stat_identity()`, whereas the default stat for `geom_bar()` is `stat_bin()`. With `geom_col` you can plot the values of x variable against y variable.

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?

Answers: | geom | stat | |———————–|———————–| | `geom_bar()` | `stat_count()` | | `geom_boxplot()` | `stat_boxplot()` | | `geom_contour()` | `stat_contour()` | | `geom_count()` | `stat_sum()` | | `geom_density()` | `stat_density()` | | `geom_density_2d()` | `stat_density_2d()` | | `geom_histogram()` | `stat_bin()` | | `geom_qq_line()` | `stat_qq_line()` | | `geom_qq()` | `stat_qq()` | | `geom_quantile()` | `stat_quantile()` | | `geom_smooth()` | `stat_smooth()` | The names of geoms and stats coming in pairs are very similar.For example, `geom_boxplot()` and `stat_boxplot()`.

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

From https://www.rdocumentation.org/packages/ggplot2/versions/0.9.0/topics/stat_smooth. We know that `stat_smooth()` compute (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 error `methods` and `formula` control its behavior

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

If we do not contain `group=1` in our code, the height of different cut will be the same.

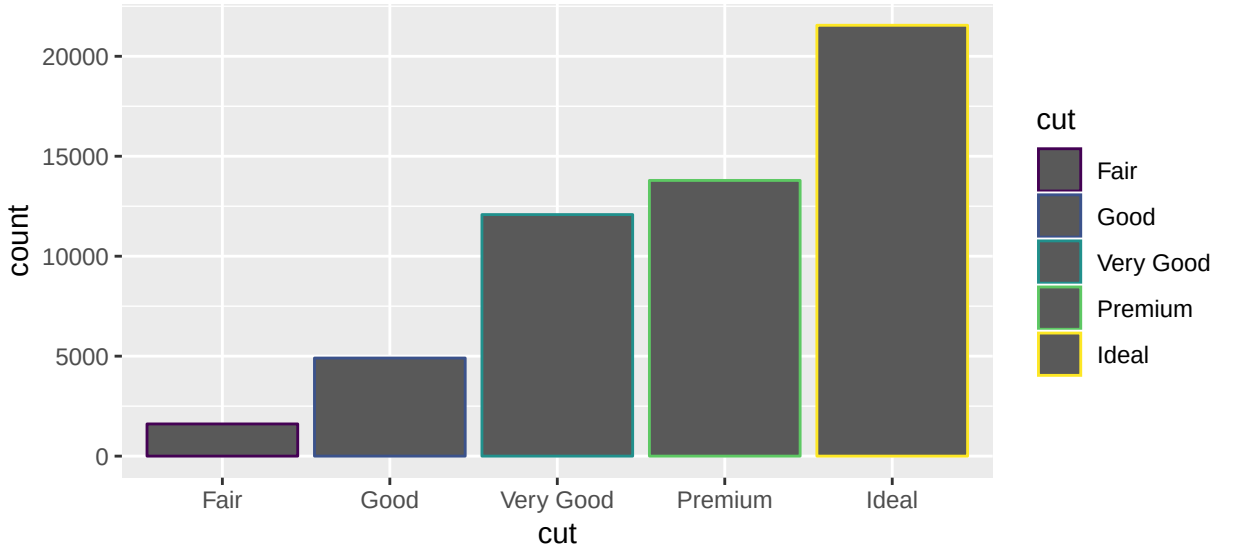
```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, y = ..prop..))

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

Section 3.8: Position adjustments

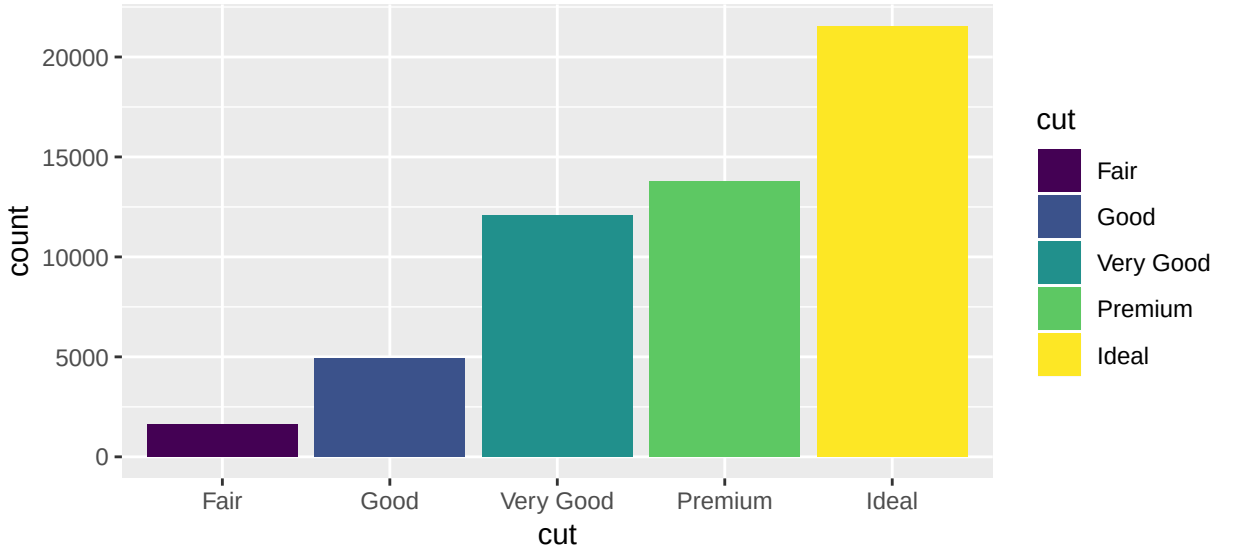
Summary

You can change the color of the bar chart by using the `colour` (which gives color to the outlines) and `fill` (which colors the whole bar) options.

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

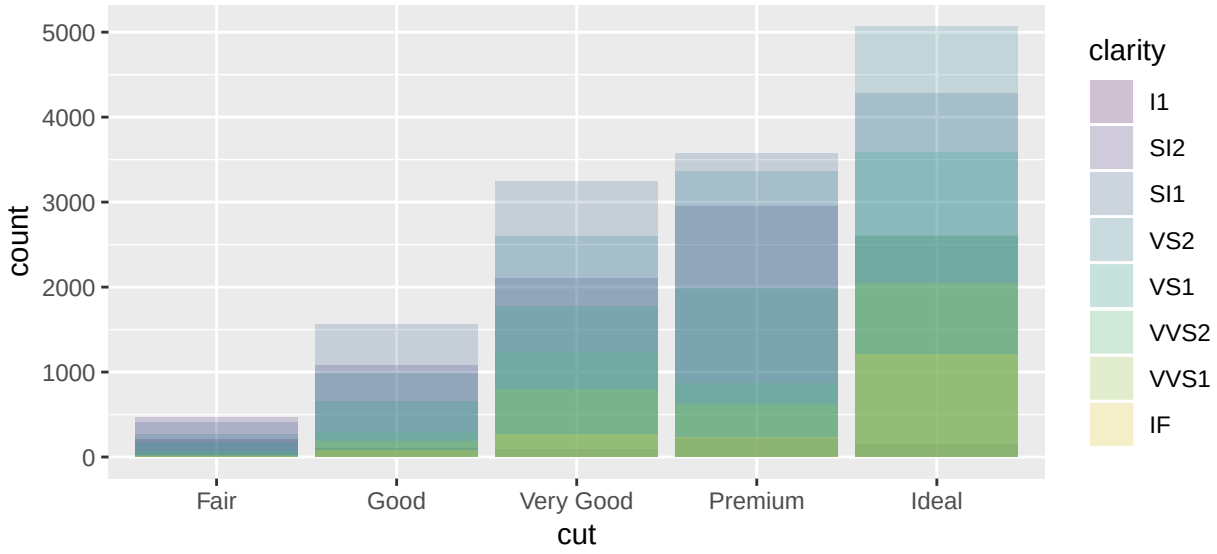



```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, fill = cut))
```

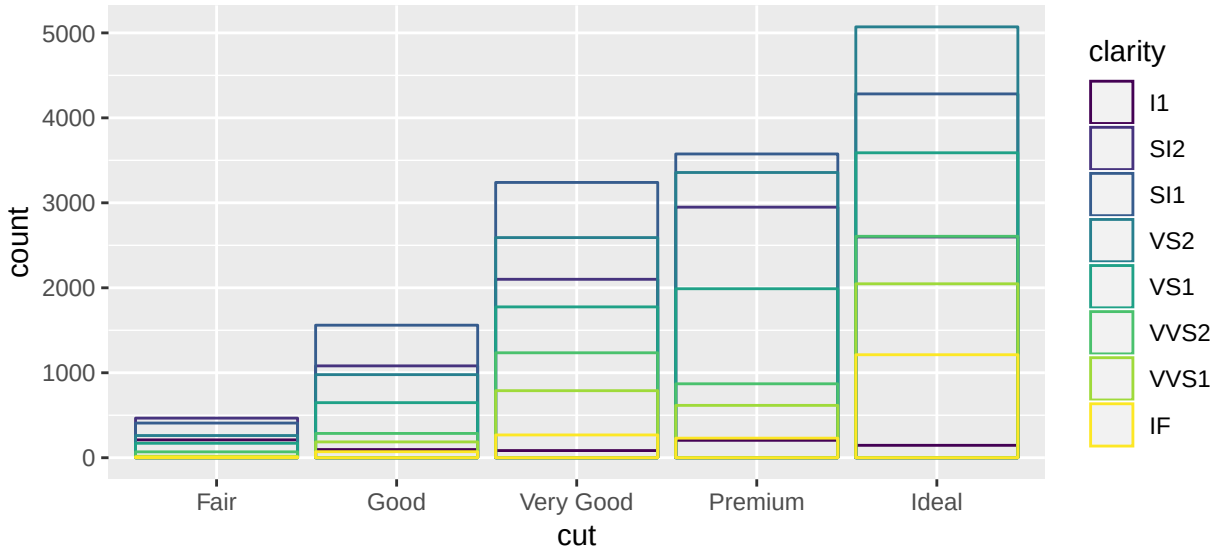


These options can also be used to create a stacked bar chart when instead of a color a varible is used as the argment for `fill` or `colour`.

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

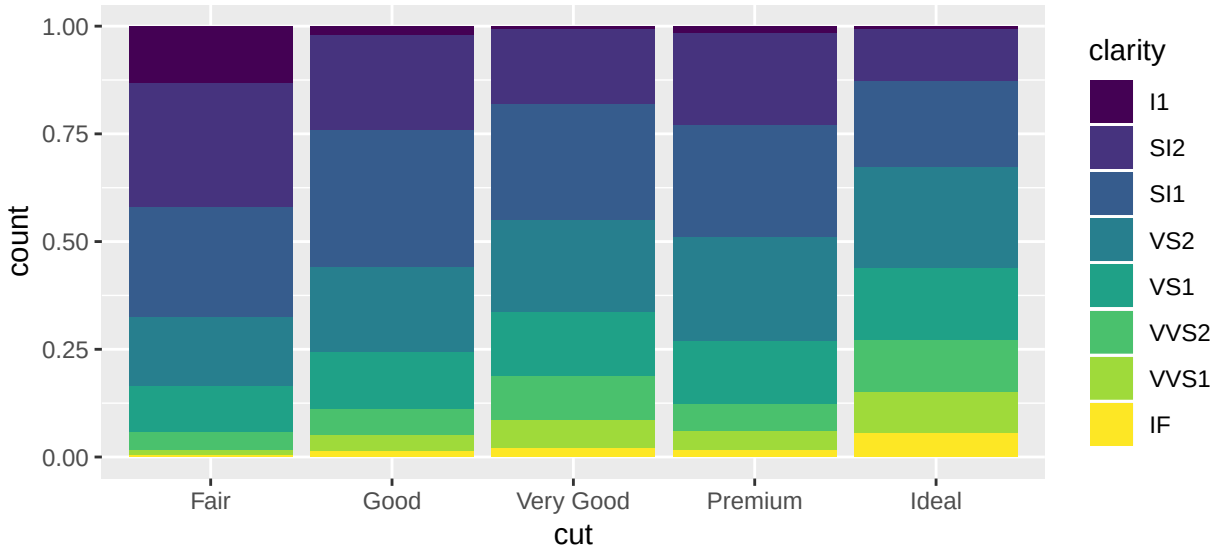



```
ggplot(data = diamonds, mapping = aes(x = cut, colour = clarity)) +
  geom_bar(fill = NA, position = "identity")
```

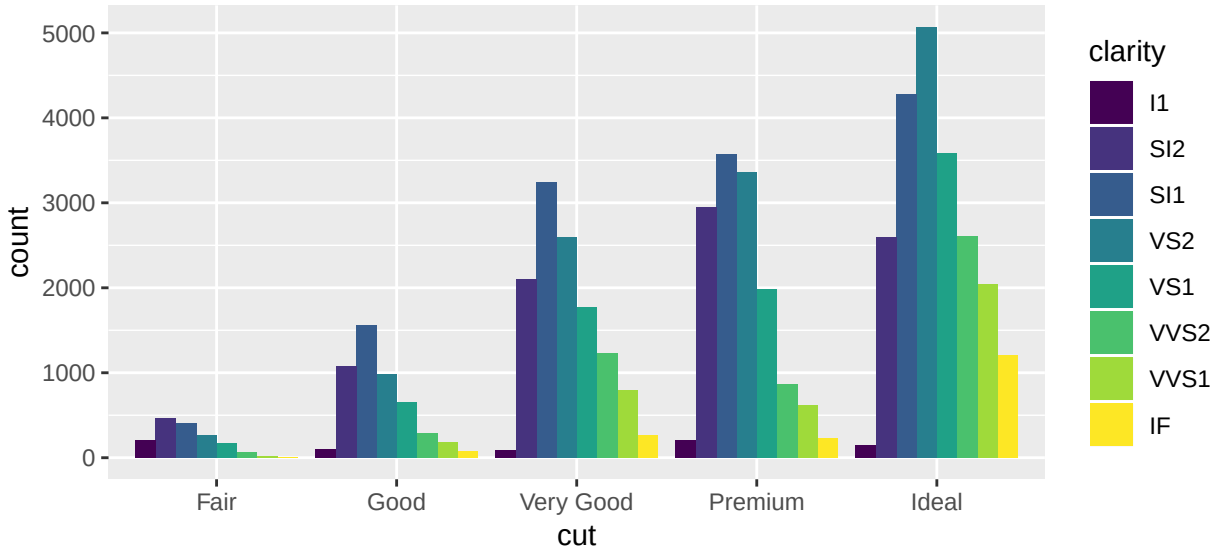


This can further be changed by using the `position` options `identity` (which creates a stcked bar chart as above), `fill` (which makes all the bars the same height to better compare the preportions of the fill variable), and `dodge` (which places overlapping objects directly beside one another).

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

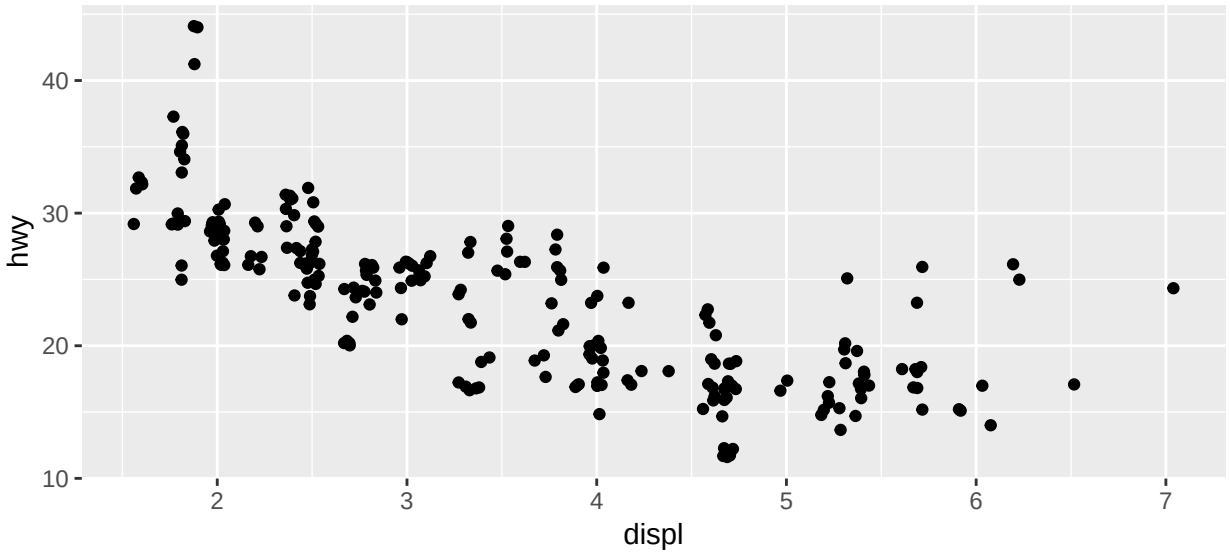



```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, fill = clarity), position = "dodge")
```



Another `position` argument that can be used for scatterplots to help with overplotting is the option `jitter` which also has a shorthand `geom_jitter()`. This adds a small amount of random noise to each point uncovering overlapped points.

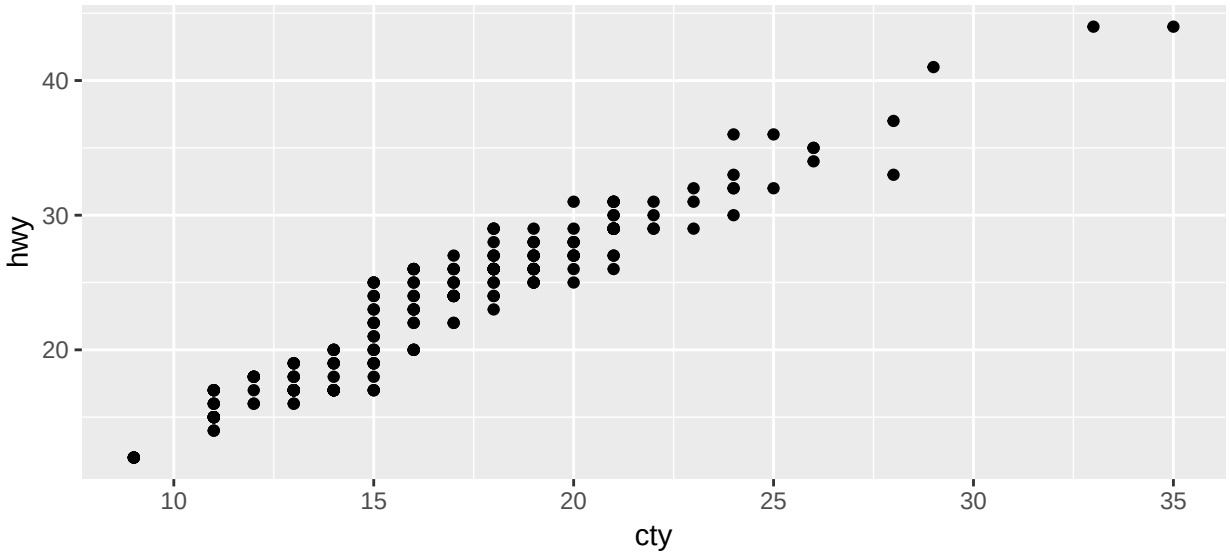
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy), position = "jitter")
```



3.8.1 Exercises

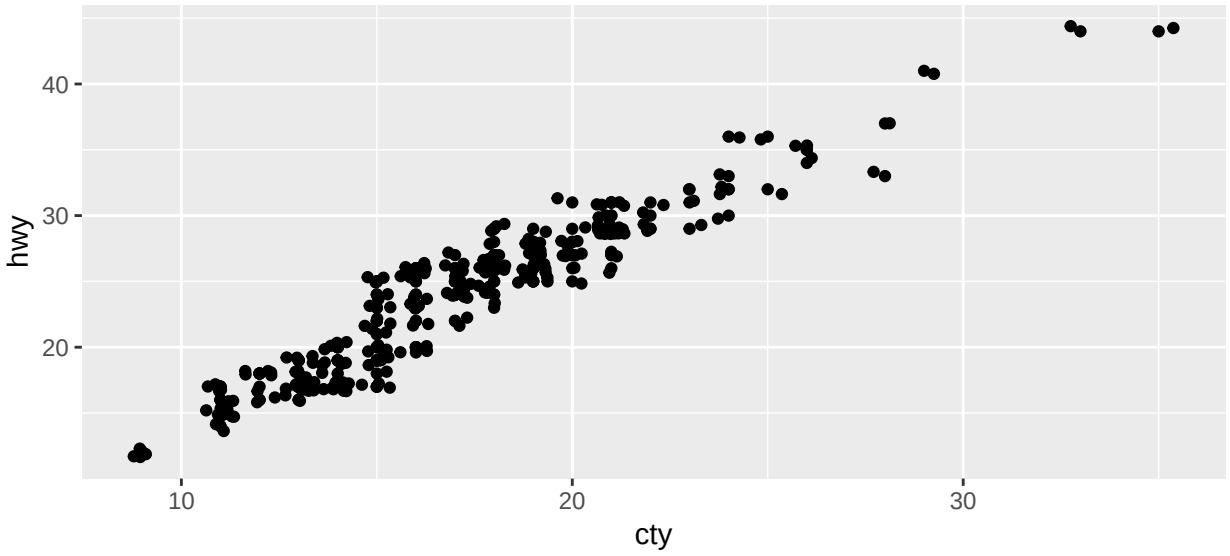
1. What is the problem with this plot? How could you improve it?

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



The problem with this plot is overplotting and it can be improved by using jitter.

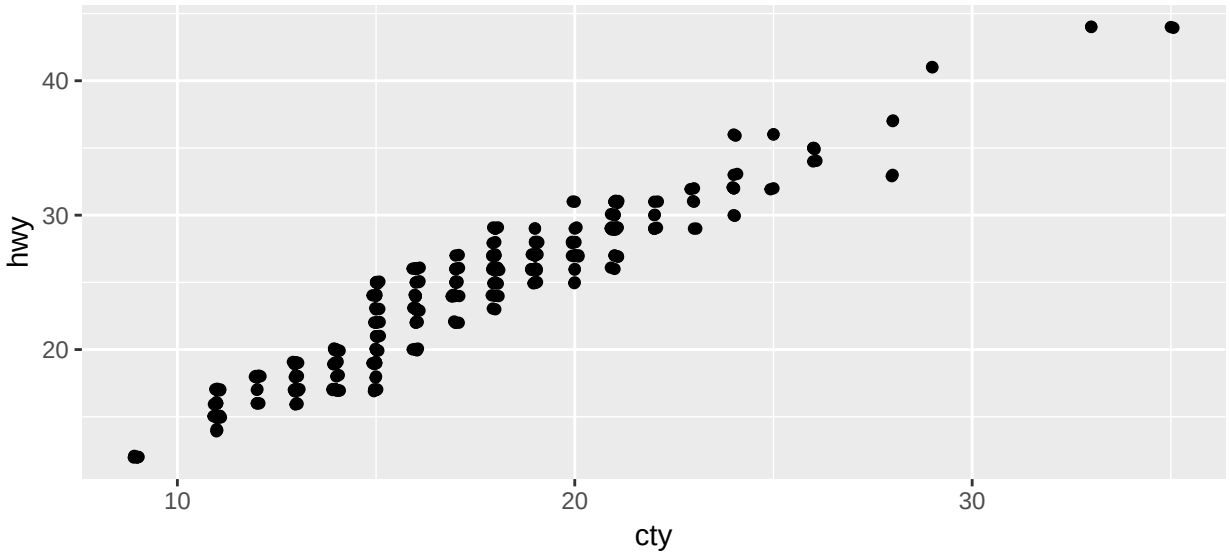
```
ggplot(data = mpg, mapping = aes(x = cty, y = hwy)) +
  geom_point() + geom_jitter()
```



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

The parameter `width` and `height` can be used to adjust the amount of jittering.

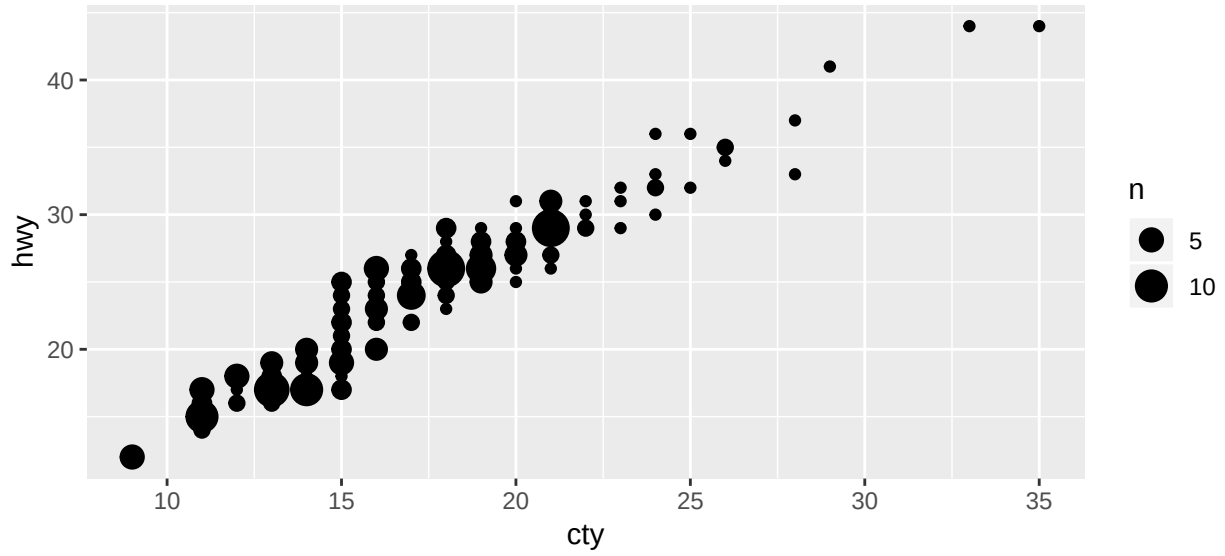
```
ggplot(data = mpg, mapping = aes(x = cty, y = hwy)) +
  geom_point() + geom_jitter(width=.1, height = .1)
```



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

While `geom_jitter()` moves each point slightly `geom_count()` replaces overlapping points with a larger point relative the the number of overlapping points like so:

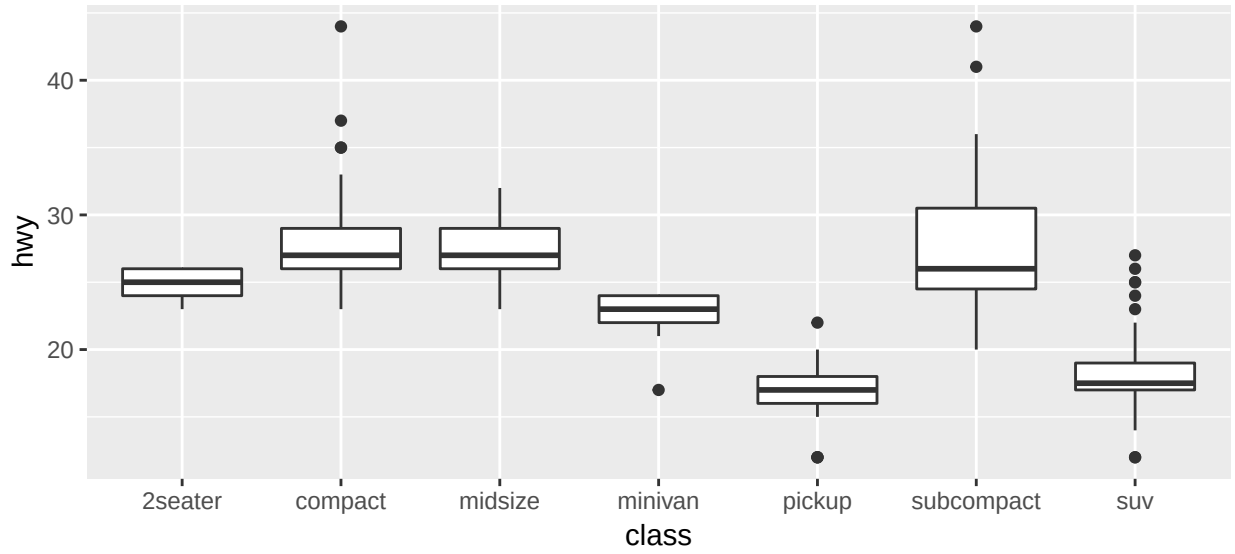
```
ggplot(data = mpg, mapping = aes(x = cty, y = hwy)) +
  geom_point() + geom_count()
```



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

The default position argument is `dodge` as demonstated by these two plots being the same:

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




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


40
30
20
hwy
2seater
compact
midsize
minivan
pickup
subcompact
suv
class

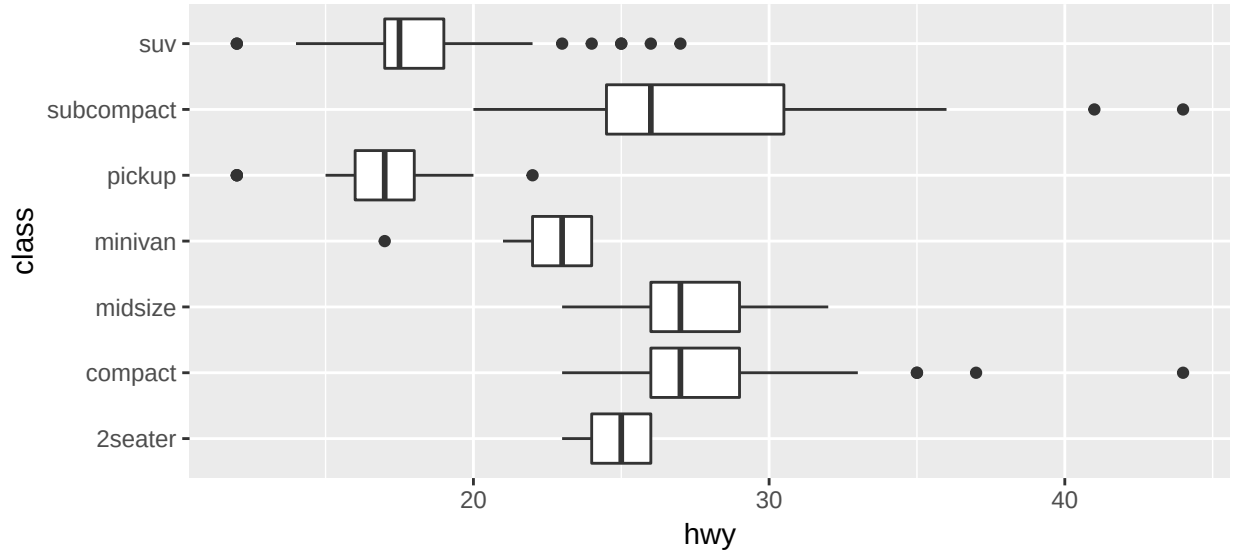
section 3.9 coordinate systems

For ggplot, coordinates can be adjusted in purose. Below are some situations where you might want to make changes.

To switches the x and y axes, use coord_flip():

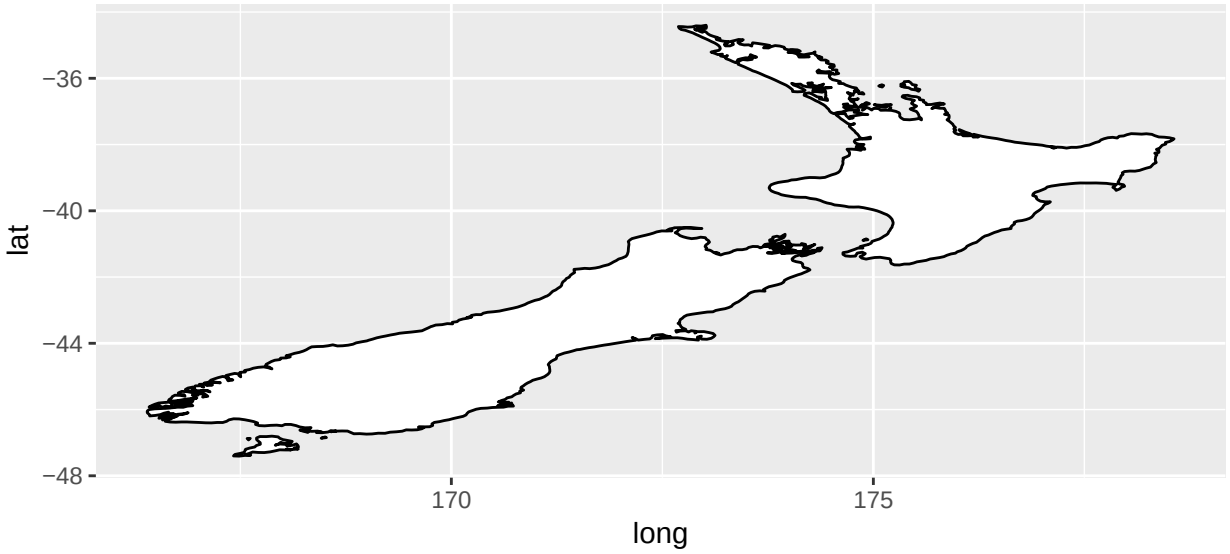
```
ggplot(data = mpg, mapping = aes(x = class, y = hwy)) +
  geom_boxplot()
```


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

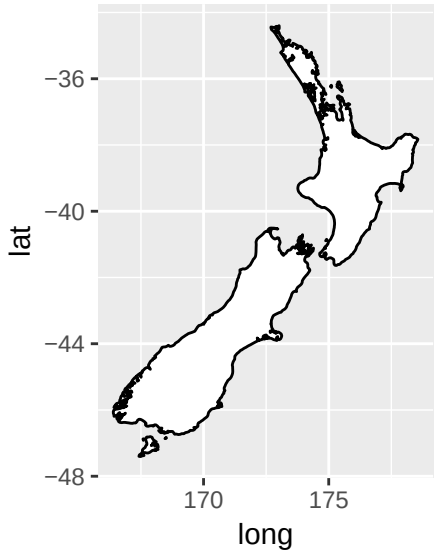


While plotting spatial data, use coord_quickmap() to sets the aspect ratio correctly for maps:

```
nz <- map_data("nz")
ggplot(nz, aes(long, lat, group = group)) +
  geom_polygon(fill = "white", colour = "black")
```



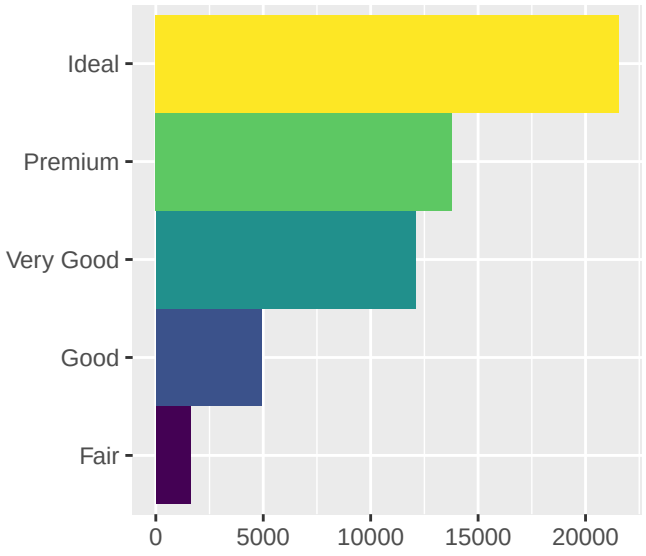

```
ggplot(nz, aes(long, lat, group = group)) +
  geom_polygon(fill = "white", colour = "black") +
  coord_quickmap()
```



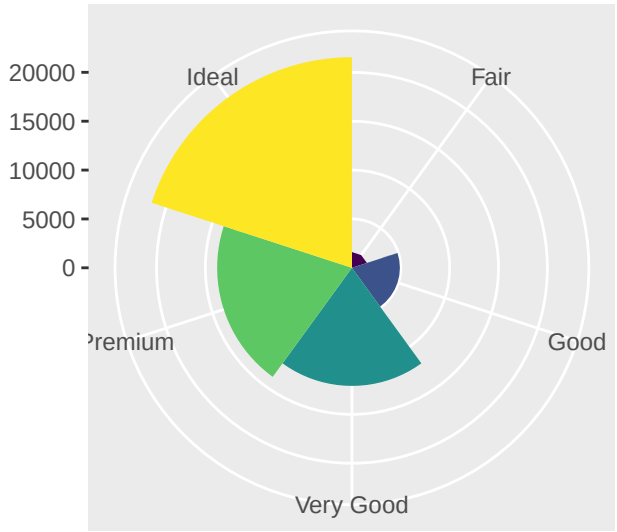
To use polar coordinates, coord_polar():

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



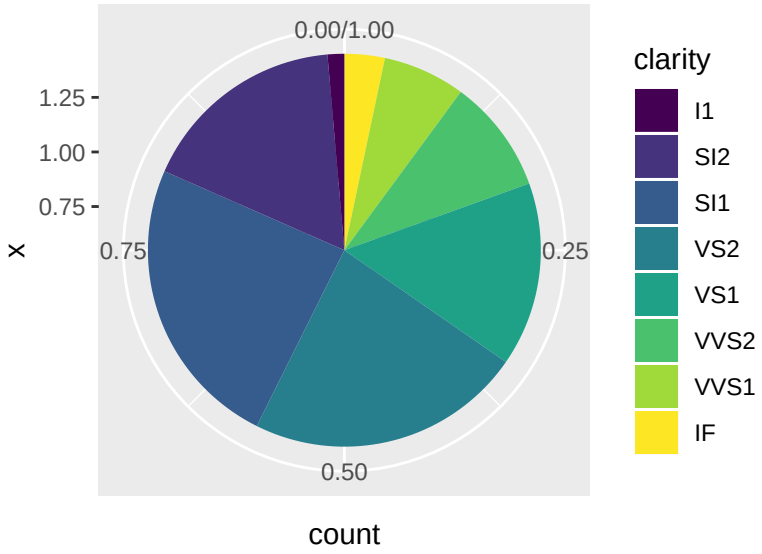

```
bar + coord_polar()
```



3.9.1 Exercise solution

1.

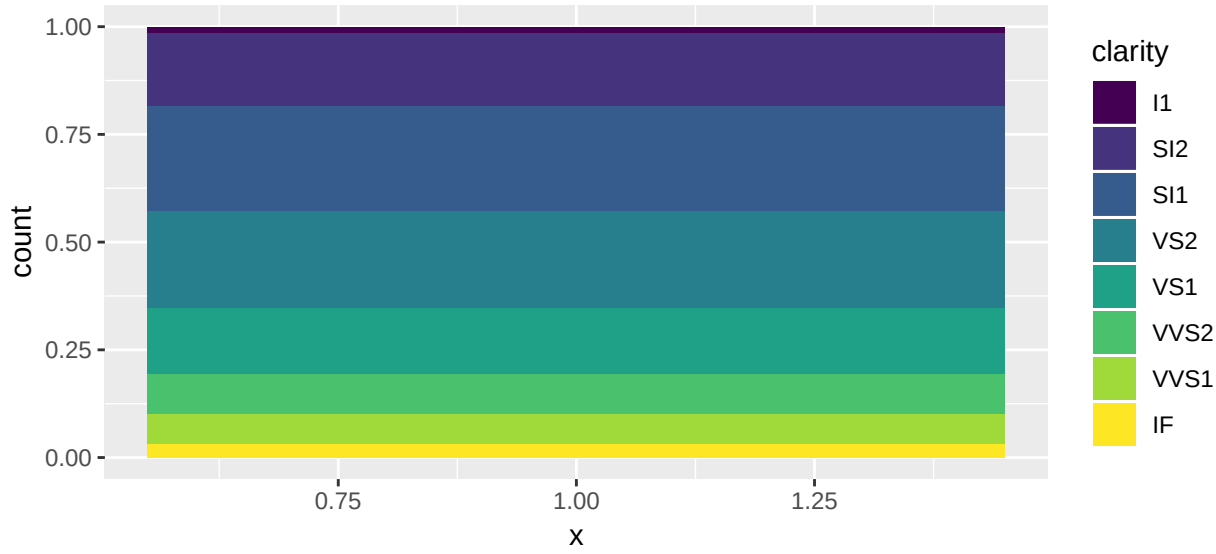
```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = 1, fill = clarity), position = "fill") +
  coord_polar(theta = "y")
```



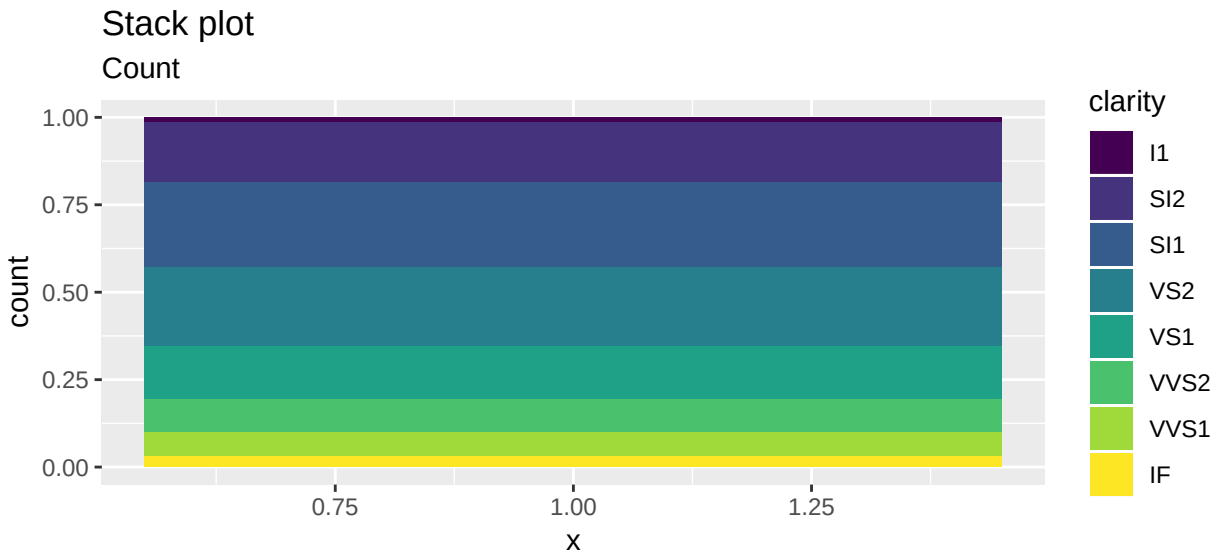
2.

Labs function can modify axis, legend, and plot labels. For example:

```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = 1, fill = clarity), position = "fill")
```




```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = 1, fill = clarity), position = "fill") +
  labs(title = "Stack plot", subtitle = "Count")
```

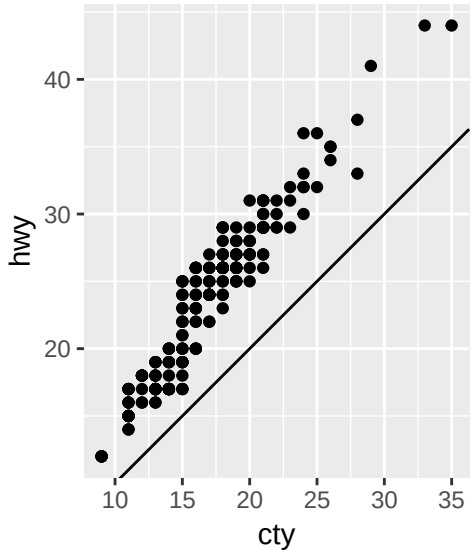


3.

They do similar jobs. Map projections in 'coord_map' require package 'mapproj' and they 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.

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



The plot shows that hwy and cty are highly correlated and the relationship is linear.
'coord_fixed' forces a specified ratio between the physical representation of data units on the axes. Here by default the ratio is 1. It ensures that one unit on the x-axis is the same length as one unit on the y-axis. In

some sense it avoided a misleading message that correlation between hwy and cty was small.
'geom_abline' added a diagonal reference line to the plot.

section 3.10

To summarize, this section goes over the layered grammar of graphics and the fact that the grammar implemented in ggplot2 allows any plot to be described as the combination of dataset, choice of geometry, mappings, transformations of data, coordinate system, and facetting scheme. The section then goes on to describe the basic process of constructing a plot using the grammar of graphics. The basic steps are:

You begin with a dataset which you might transform for easier display using some stat() function. You then decide upon an appropriate geom for the data at hand and the mappings you will use to map the variables in the data to the properties of the geom. You then select the coordinate system to place the geom in. You could also choose to facet your data, or to add more layers to your plot.