Group 1 Project 2 - Data Carpentry & Visualization Basics

Janna Jerngian, Ryo Iwata, Sequioa Smith

Group 1 members:

Janna Jernigan, Ryo Iwata, Sequioa Smith Note who is assigned leader Janna Jernigan

Group Assignment 2

Instructions - Everyone should work on their own computer on a separate copy of this, then work together to create a single very well polished version to submit for the assignment. The assigned group leader should submit the assignment for the group, but every individual should thoroughly understand every answer. In most cases, everyone should roughly be on the same question as the same time when working together simultaneously (e.g., in class or during group meetings). When asked, you (all) need to be ready to explain the thought process behind the plan, approach, revisions, and final code that was written. Now is the time to build your understanding of the language and approach to these data carpentry and visualization methods. My expectation is that every submitted assignment will be nearly perfect given how we will be approaching this series of problems both in class and out.

I have copied together all the questions in one place so everyone doesn't have to waste their time doing so, but these are directly from the reading (R4DS 2E). I am leaving the formatting plain, so refer to the book itself for the example plots and pretty formatting.

Please make it easy to grade this! Your text answers should be in the main document (i.e., not in a code chunk), and **please make them bold so they stand out**. Write code, including using comments, as if you are writing code that will be published. Build good habits now!

Important note: These chapters cover important advice but contain few exercises. As with all these chapters, please read them carefully as the information contained will be extremely helpful moving forward! You'll hopefully notice we've already discussed much of this once, hopefully the repetition helps reinforce the importance.

4.6

1

Restyle the following pipelines following the guidelines above.

```
# Old code below
flights|>filter(dest=="IAH")|>group_by(year,month,day)|>summarize(n=n(),
delay=mean(arr_delay,na.rm=TRUE))|>filter(n>10)
```

```
## `summarise()` has grouped output by 'year', 'month'. You can override using the
## `.groups` argument.
```

```
## # A tibble: 365 x 5
## # Groups:   year, month [12]
##     year month   day     n delay
##    <int> <int> <int> <int> <dbl>
```

```
##  1  2013     1     1    20 17.8
##  2  2013     1     2    20  7
##  3  2013     1     3    19 18.3
##  4  2013     1     4    20 -3.2
##  5  2013     1     5    13 20.2
##  6  2013     1     6    18  9.28
##  7  2013     1     7    19 -7.74
##  8  2013     1     8    19  7.79
##  9  2013     1     9    19 18.1
## 10  2013     1    10    19  6.68
## # i 355 more rows
```

```
flights|>filter(carrier=="UA",dest%in%c("IAH","HOU"),sched_dep_time>
0900,sched_arr_time<2000)|>group_by(flight)|>summarize(delay=mean(
arr_delay,na.rm=TRUE),cancelled=sum(is.na(arr_delay)),n=n())|>filter(n>10)
```

```
## # A tibble: 74 x 4
##    flight delay cancelled     n
##     <int> <dbl>     <int> <int>
##  1     53 12.5          2    18
##  2    112 14.1          0    14
##  3    205 -1.71         0    14
##  4    235 -5.36         0    14
##  5    255 -9.47         0    15
##  6    268 38.6          1    15
##  7    292  6.57         0    21
##  8    318 10.7          1    20
##  9    337 20.1          2    21
## 10    370 17.5          0    11
## # i 64 more rows
```

```
# Restyled code below
flights |>
  filter(dest == "IAH") |>
  group_by(
    year,
    month,
    day) |>
  summarize(
    n = n(),
    delay = mean(arr_delay, na.rm = TRUE)) |>
  filter(n > 10)
```

```
## `summarise()` has grouped output by 'year', 'month'. You can override using the
## `.groups` argument.
```

```
## # A tibble: 365 x 5
## # Groups:   year, month [12]
##     year month   day     n delay
##    <int> <int> <int> <int> <dbl>
##  1  2013     1     1    20 17.8
##  2  2013     1     2    20  7
##  3  2013     1     3    19 18.3
##  4  2013     1     4    20 -3.2
##  5  2013     1     5    13 20.2
##  6  2013     1     6    18  9.28
```

```
##  7  2013     1     7    19 -7.74
##  8  2013     1     8    19  7.79
##  9  2013     1     9    19 18.1
## 10  2013     1    10    19  6.68
## # i 355 more rows
```

```
flights |>
  filter(
    carrier == "UA",
    dest%in%c("IAH", "HOU"),
    sched_dep_time > 0900,
    sched_arr_time < 2000) |>
  group_by(flight) |>
  summarize(
    delay = mean(arr_delay, na.rm = TRUE),
    cancelled = sum(is.na(arr_delay)),
    n = n()) |>
  filter(n > 10)
```

```
## # A tibble: 74 x 4
##    flight delay cancelled     n
##     <int> <dbl>     <int> <int>
##  1     53 12.5          2    18
##  2    112 14.1          0    14
##  3    205 -1.71         0    14
##  4    235 -5.36         0    14
##  5    255 -9.47         0    15
##  6    268 38.6          1    15
##  7    292  6.57         0    21
##  8    318 10.7          1    20
##  9    337 20.1          2    21
## 10    370 17.5          0    11
## # i 64 more rows
```

5.2.1

1

For each of the sample tables, describe what each observation and each column represents.

Answer: Table 1: Each observation is the number of cases and the population for a given country and a given year. `country`- Name of country of data point. `year`- Calendar year of data point. cases- Number of TB cases in the associated country and year. `population`- Number of people in the associated country

Table 2: Each observation is the number of cases or the population for a given country and a given year. `country`- Name of country of data point. `year`- Calendar year of data point. `type`- The variable name of `count` which can be `case` or `population` for the current observation. `count`- the value of the variable specified by `type` for the current observation.

Table 3: Each observation is the rate of TB cases for the population for a given country and a given year. `country`- Name of country of data point. `year`- Calendar year of data point. cases- Number of TB cases in the associated country and year. `rate`- the number of TB cases that a given country and year has divided by the population, or the rate of TB cases.

2

Sketch out the process you'd use to calculate the rate for table2 and table3. You will need to perform four operations: **You haven't yet learned all the functions you'd need to actually perform these operations, but you should still be able to think through the transformations you'd need.**

a Extract the number of TB cases per country per year. #### b Extract the matching population per country per year. #### c Divide cases by population, and multiply by 10000. #### d Store back in the appropriate place.

Answer: Table 2: Identify rows in table2 where the type is "cases." Extract the corresponding count values for cases into a new tibble. Identify rows in table2 where the type is "population." Extract the corresponding count values for population into a new tibble. Create a new tibble from the two new tibbles by "joining" the rows that have the same country and year. Perform the calculation and save it to a new column: rate = cases/population × 10000 for each corresponding country and year. Create a new column in table2 named rate. Store the calculated rates in the rate column for the respective country and year.

Answer: Table 3: Extract the numerator (cases) from the rate column in table3 by splitting the text by / Extract the denominator (population) from the rate column in table3 by splitting the text by / Perform the calculation and save it to a new column: rate = cases/population × 10000 for each corresponding country and year. Overwrite the existing rate column in table3 with the newly calculated rates.

You can skip chapter 7 (though it can be useful, we will not rely on it within the constraints of this course). Chapter 8 contains some advice I disagree with that I've seen severely hinder students in the past.

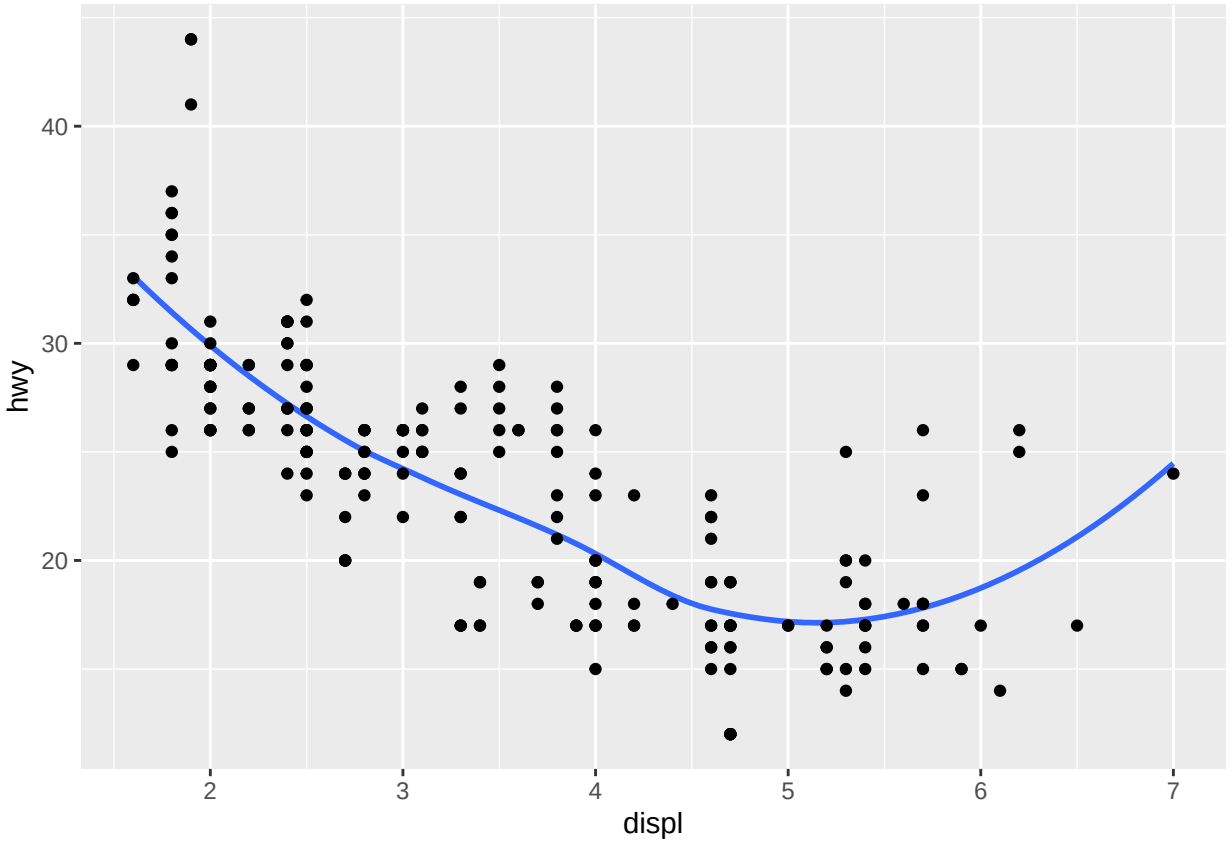
9.3.1

4

Recreate the R code necessary to generate the following graphs. Note that wherever a categorical variable is used in the plot, it's drv. (see book for images. Note this is 6 separate plots)

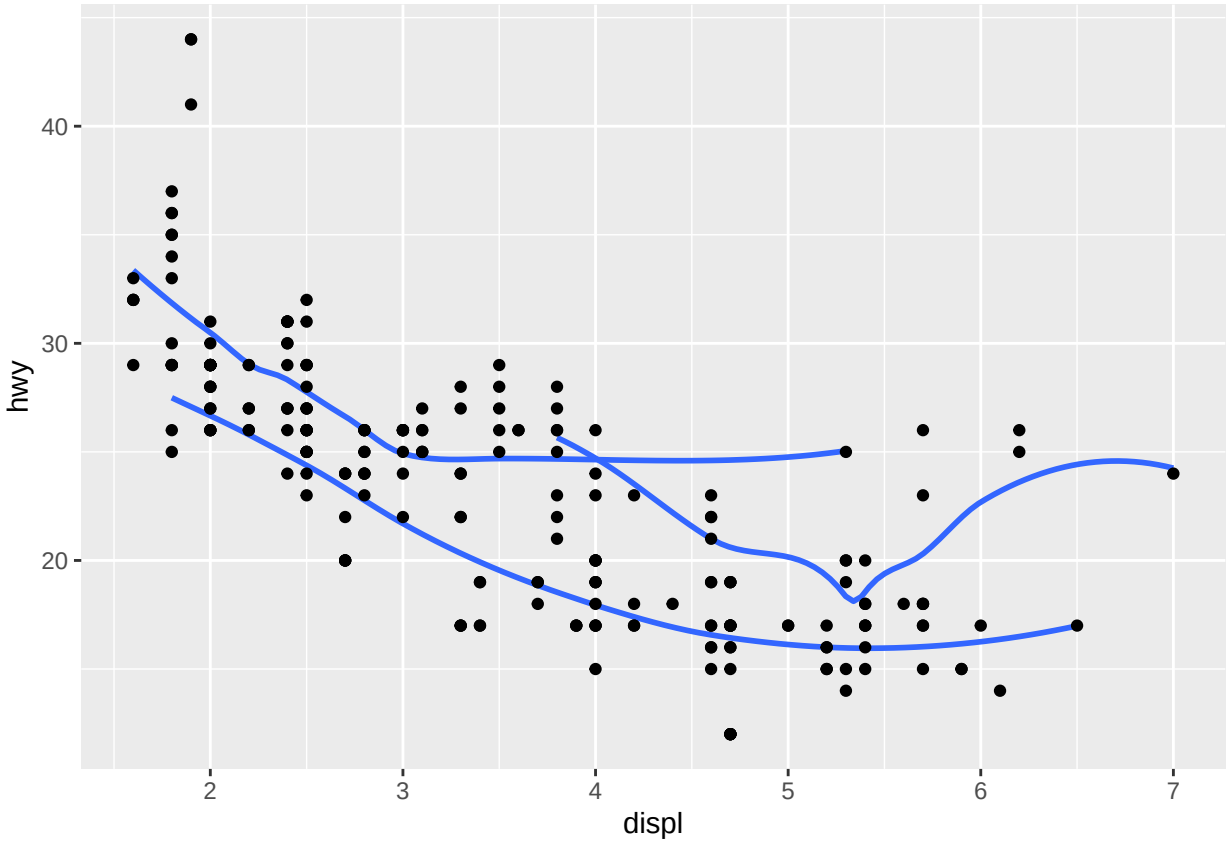
```
ggplot(mpg, aes(x = displ, y = hwy)) +
  geom_smooth(se = FALSE ) +
  geom_point()
```

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



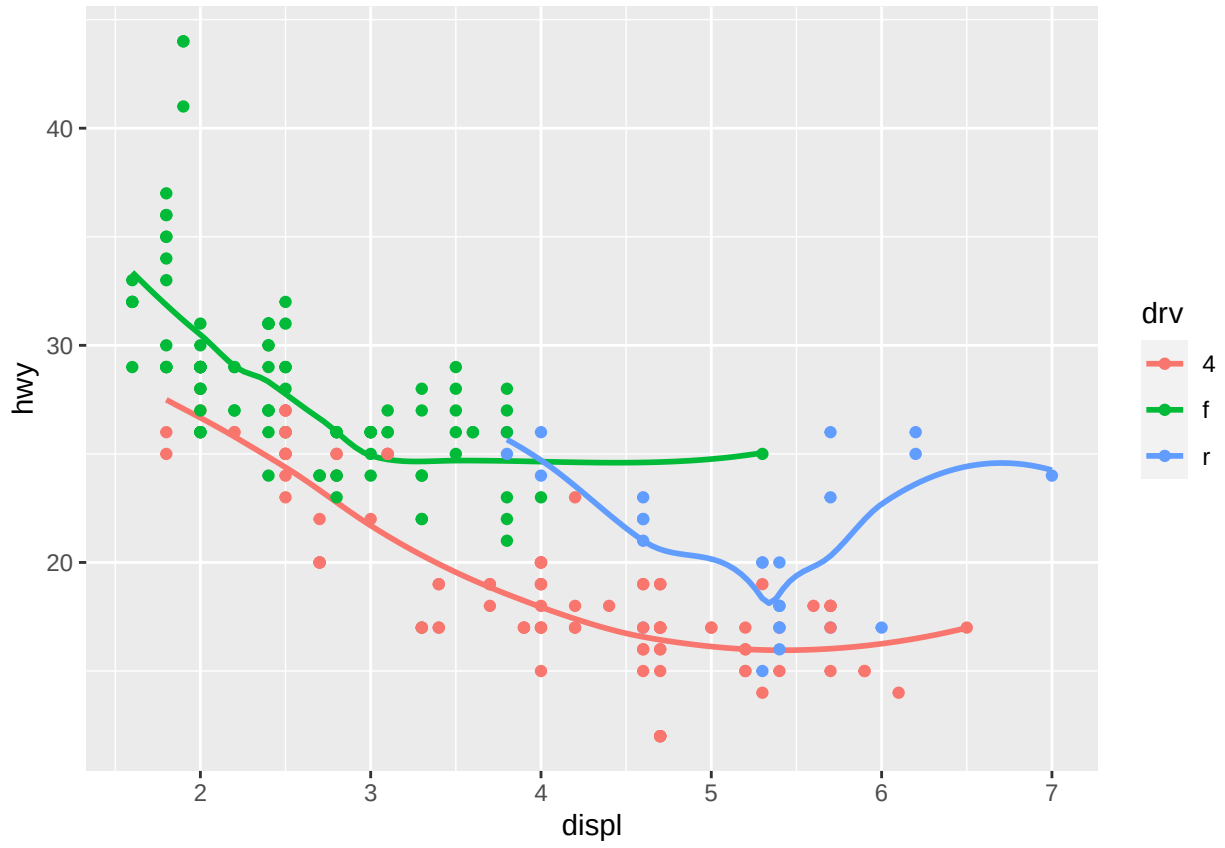

```
ggplot(mpg, aes(x = displ, y = hwy)) +
  geom_smooth(aes(group = drv), se=FALSE) +
  geom_point()
```

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



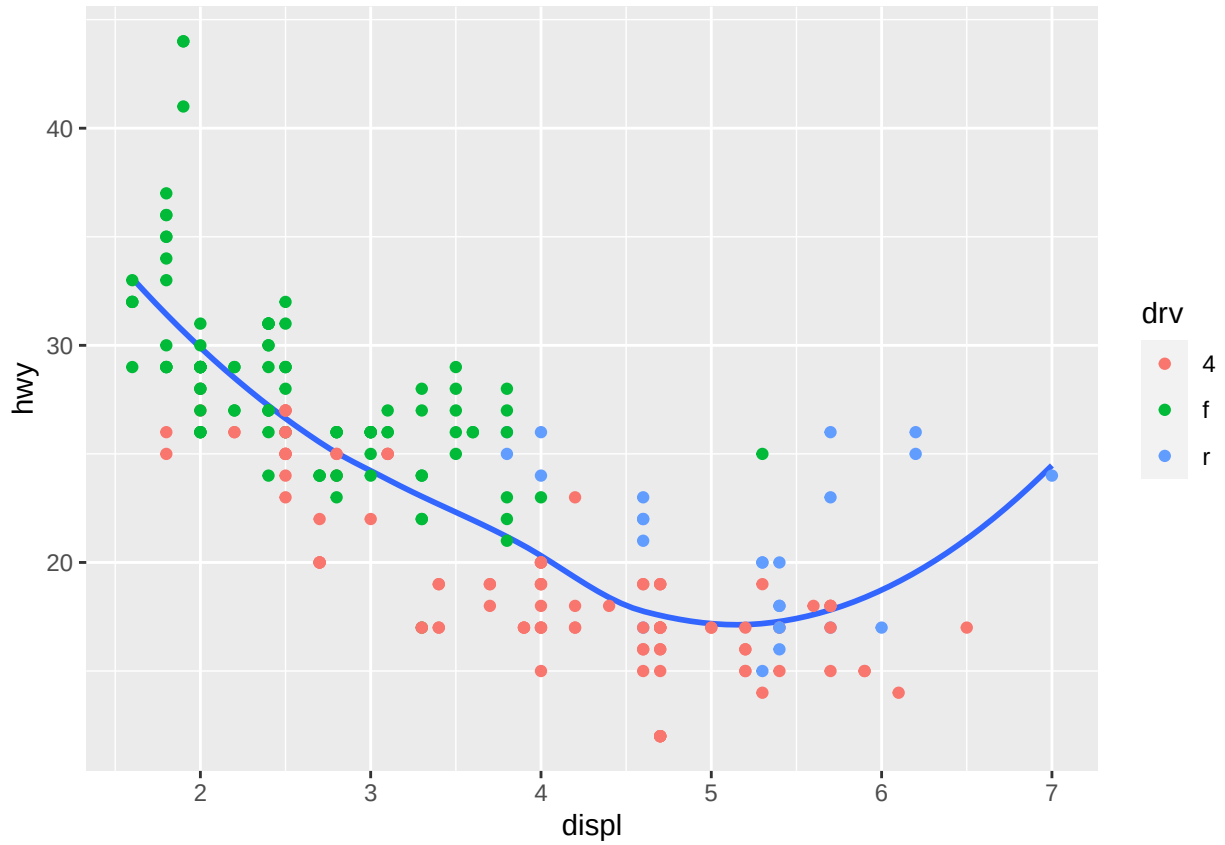

```
ggplot(mpg, aes(x = displ, y = hwy)) +
  geom_smooth(aes(color = drv), se=FALSE) +
  geom_point(aes(color = drv))
```

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



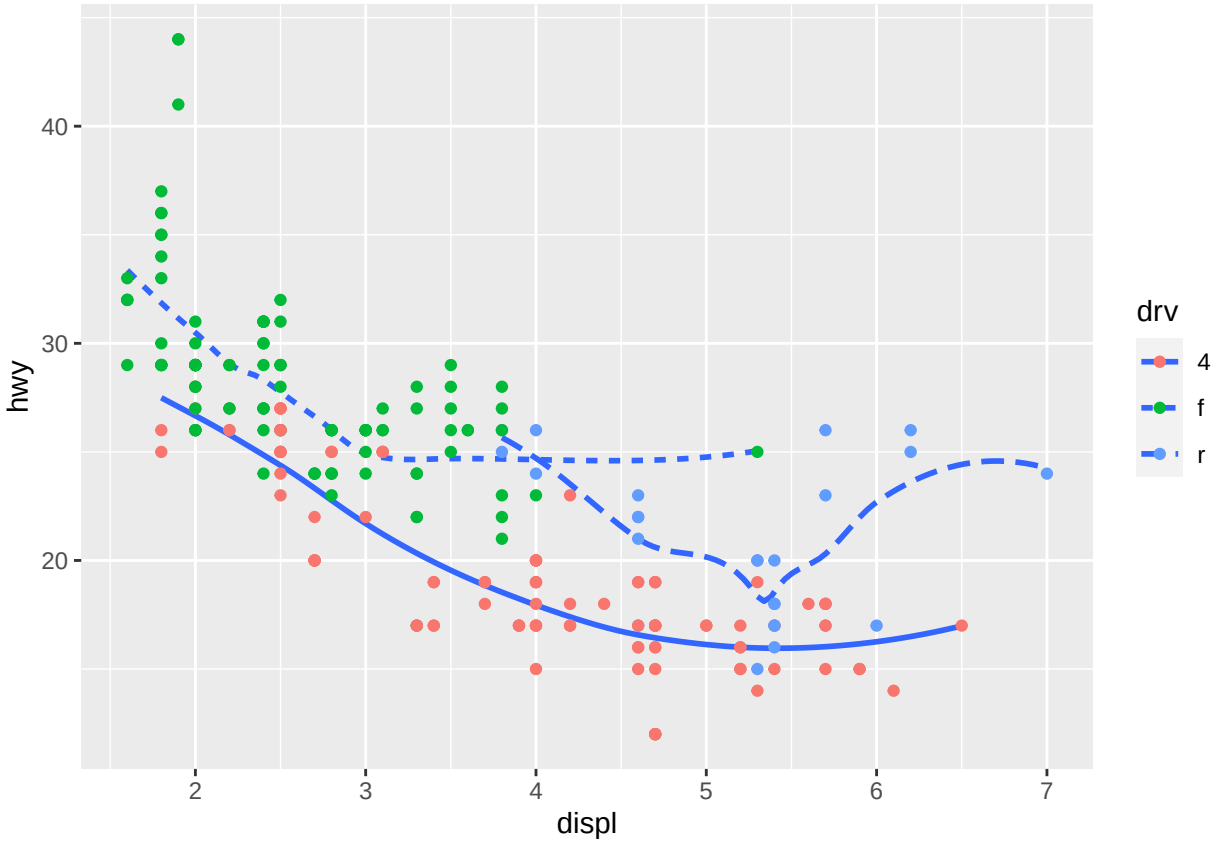

```
ggplot(mpg, aes(x = displ, y = hwy)) +
  geom_smooth(se = FALSE ) +
  geom_point(aes(color = drv))
```

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

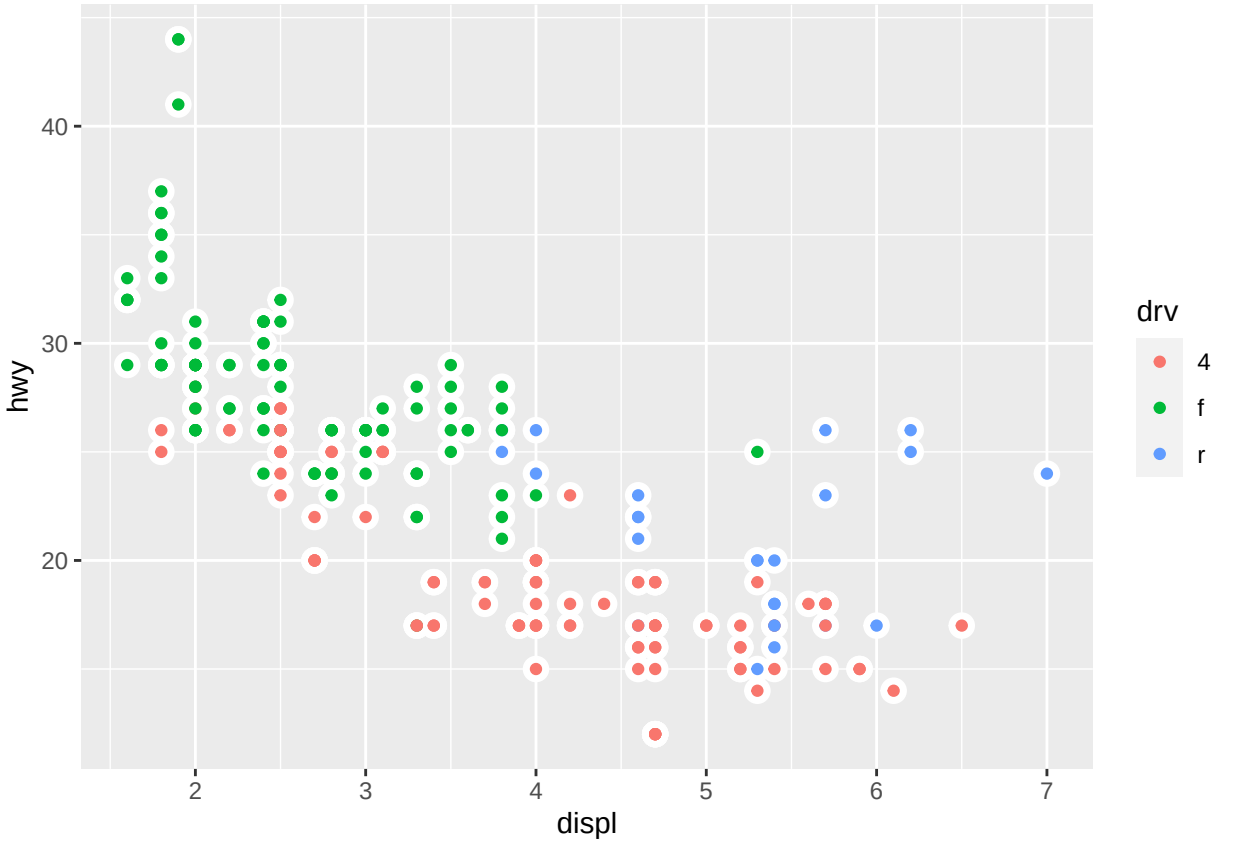



```
ggplot(mpg, aes(x = displ, y = hwy)) +
  geom_smooth(aes(linetype = drv), se=FALSE) +
  geom_point(aes(color = drv))
```

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




```
ggplot(mpg, aes(x = displ, y = hwy)) +
  geom_point(colour = "white", size = 4) +
  geom_point(aes(color = drv))
```



9.4.1

1

What happens if you facet on a continuous variable?

Answer: Faceting by a continuous variable results in one facet per each unique value of the continuous variable.

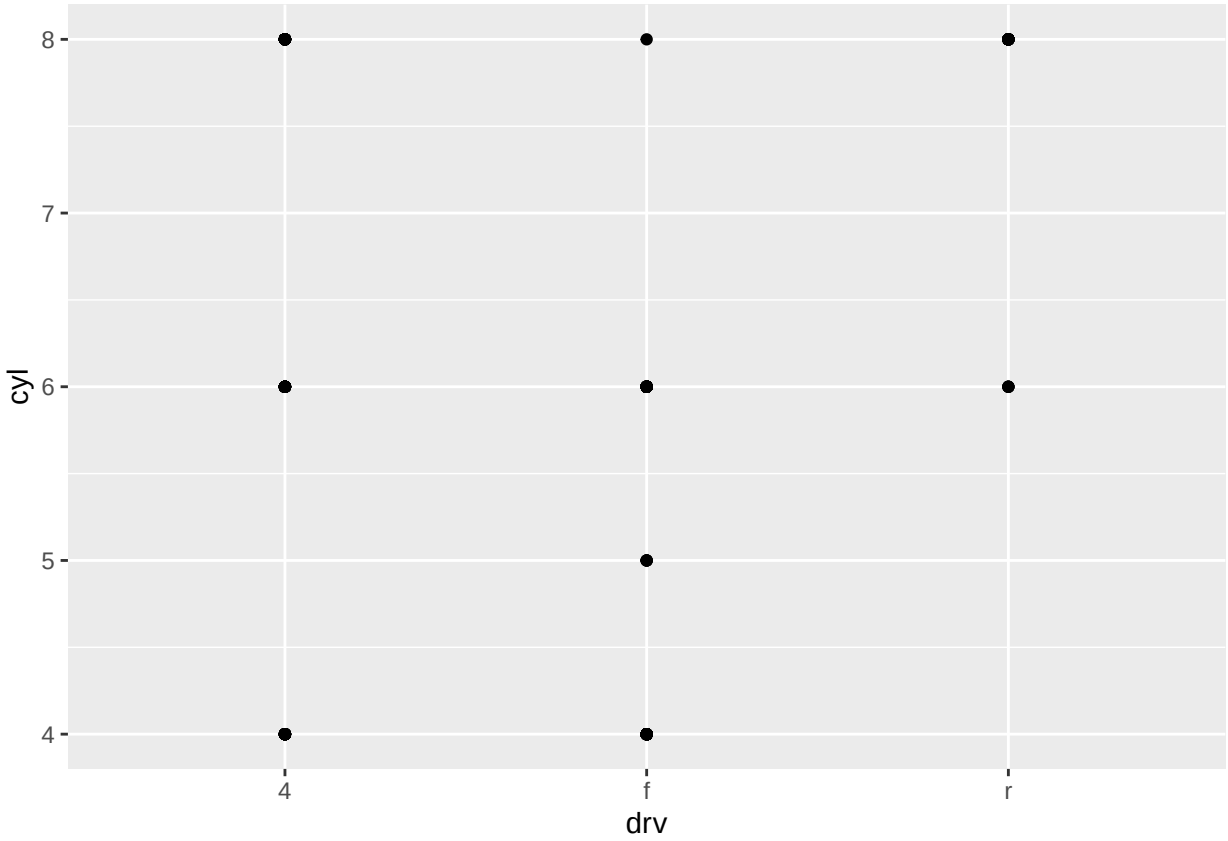
```
#?Facet
```

2

What do the empty cells in the plot above with facet_grid(drv ~ cyl) mean? Run the following code. How do they relate to the resulting plot?

Answer: The empty cells indicate that there are no data points with that combination of `cyl` and `drv`. The following code is related because there are no dots at the intersection of the `cyl` and `drv` pair that had an empty cell in the plot above.

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

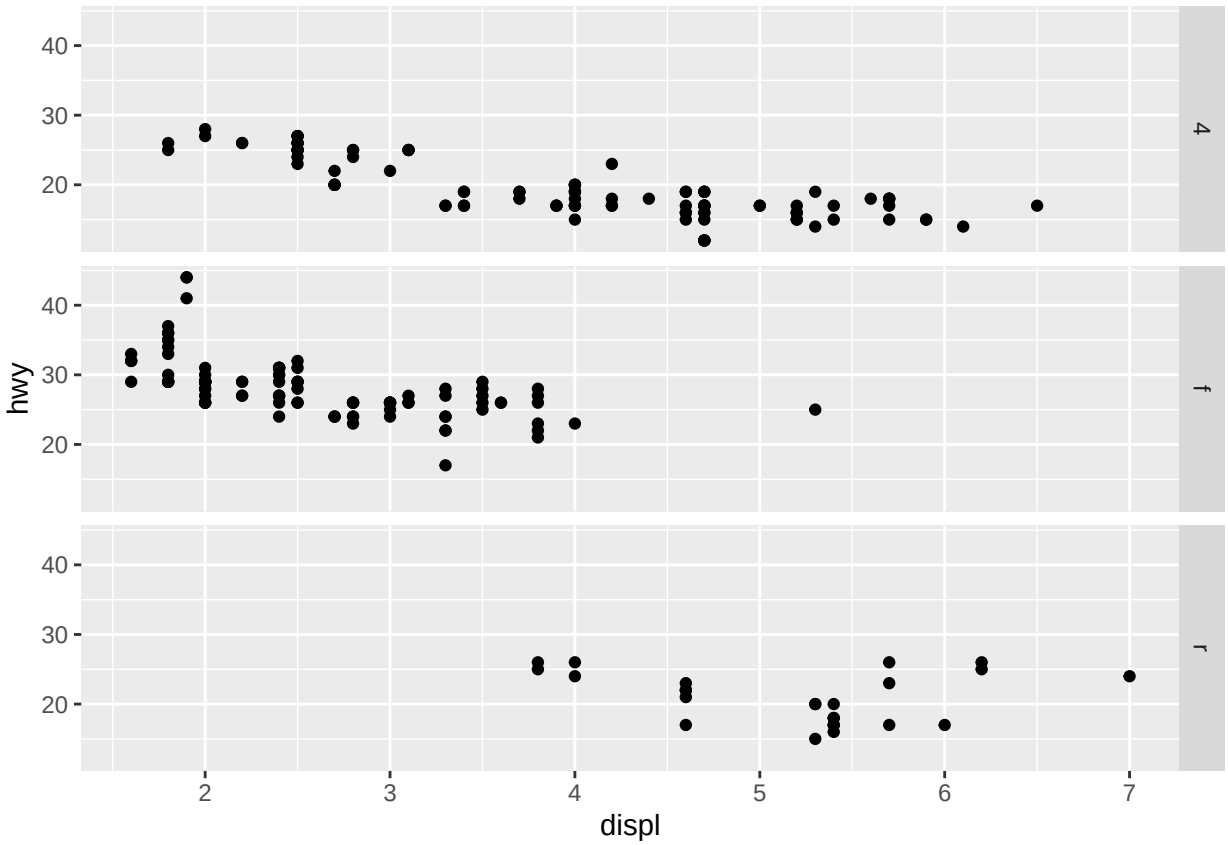


3

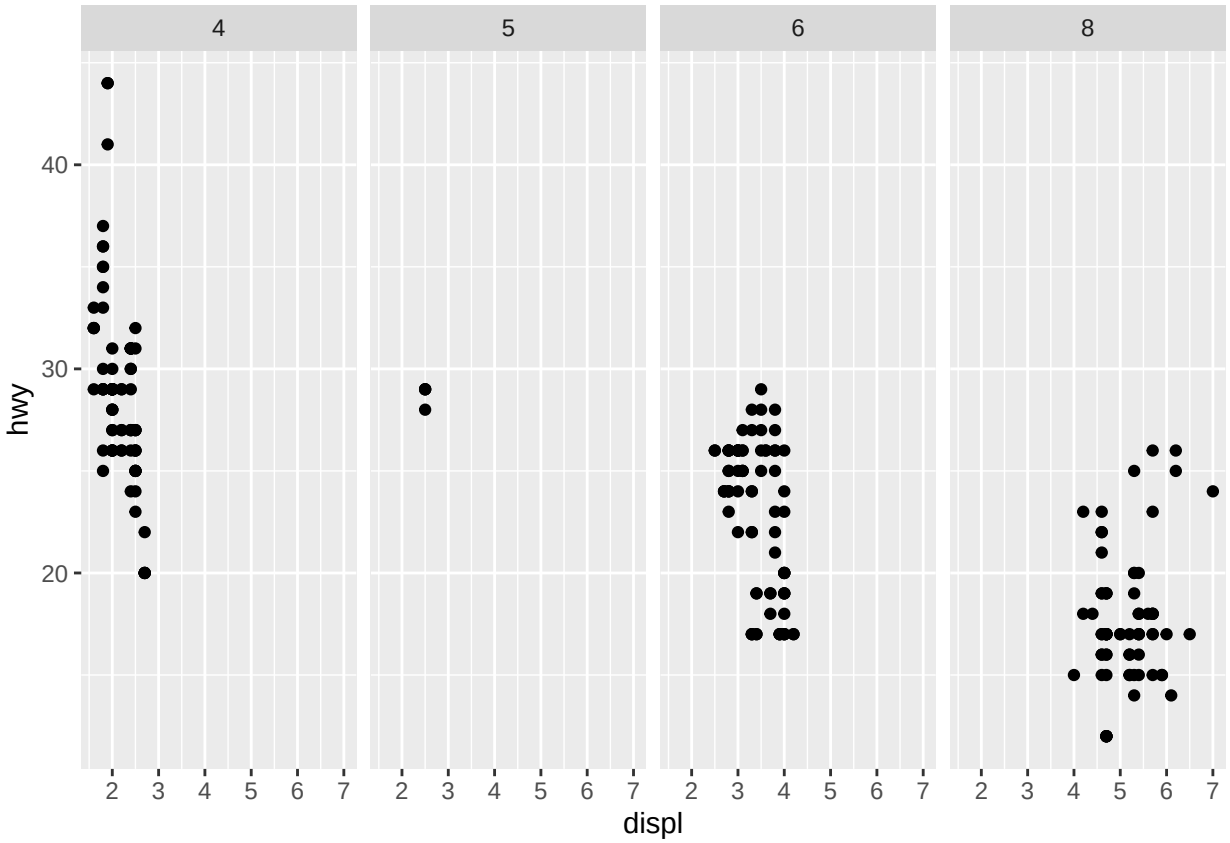
What plots does the following code make? What does . do?

Answer: Plot 1 makes a facet grid that splits horizontally as rows the scatter plot of `displ` as x and `hwy` as y by the different `drv`. Plot 2 makes a facet grid that splits vertically as columns the scatter plot of `displ` as x and `hwy` as y by the different `cyl`. The . makes it so that the `facet_grid` does not split in the dimension that . is in. If the . is before the ~ then it will only facet as columns and rows if after the ~.

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




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



4

Take the first faceted plot in this section:

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

What are the advantages to using faceting instead of the color aesthetic? What are the disadvantages? How might the balance change if you had a larger dataset?

Answer: The advantage of using faceting instead of coloring the different labels is that if there's a lot of overlap between the different groups for the variables being plotted, then some groups might hide other groups depending on the order plotted. The disadvantage is more difficulty comparing between the groups because they are on separate plots.

5

Read ?facet_wrap. What does nrow do? What does ncol do? What other options control the layout of the individual panels? Why doesn't facet_grid() have nrow and ncol arguments?

```
#?facet_wrap
```

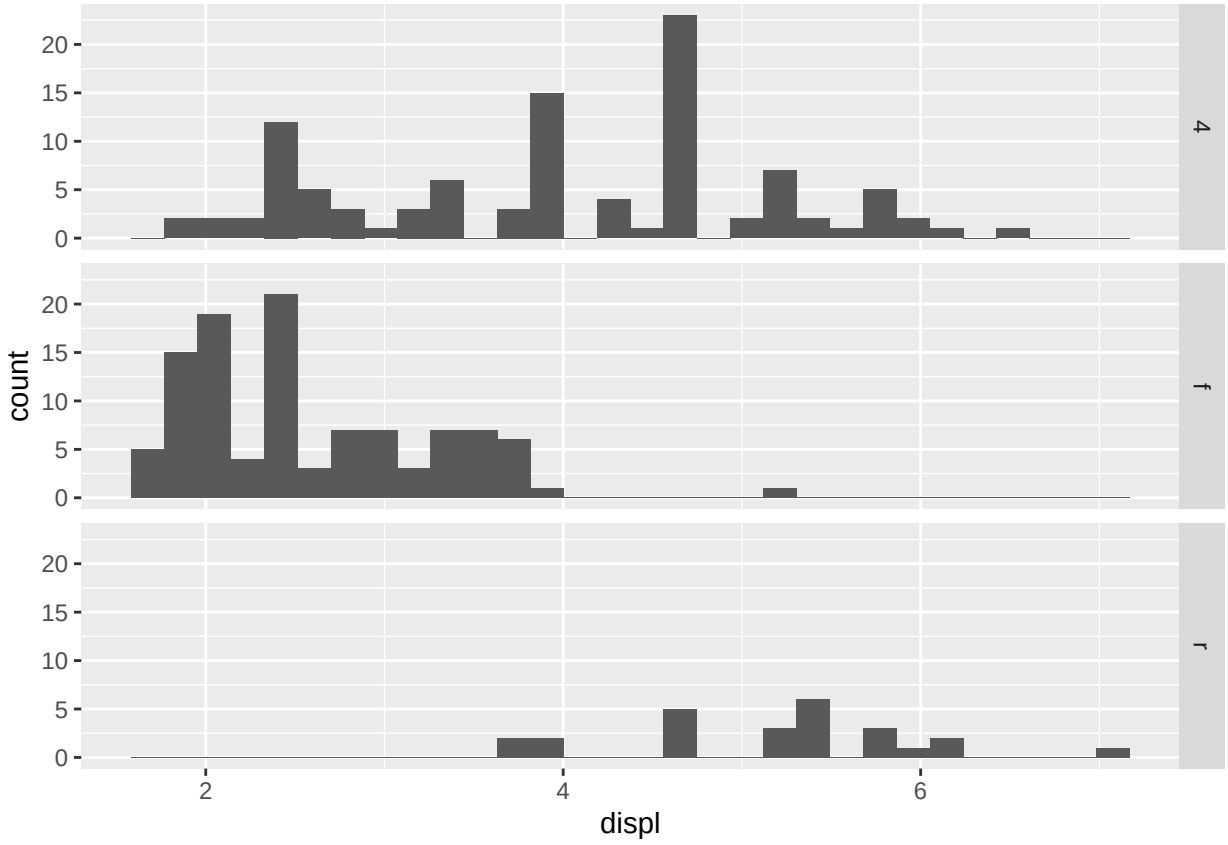
Answer: `nrow` sets the number of rows in the facet wrap plot and `ncol` sets the number of columns in the facet grid. You can also change the layout by putting different variables in different orders as the facet. `facet_grid` does not have `nrow` and `ncol` because the number of rows and columns are determined by the number of groups in the variables that are being grouped by.

6

Which of the following plots makes it easier to compare engine size (displ) across cars with different drive trains? What does this say about when to place a faceting variable across rows or columns?

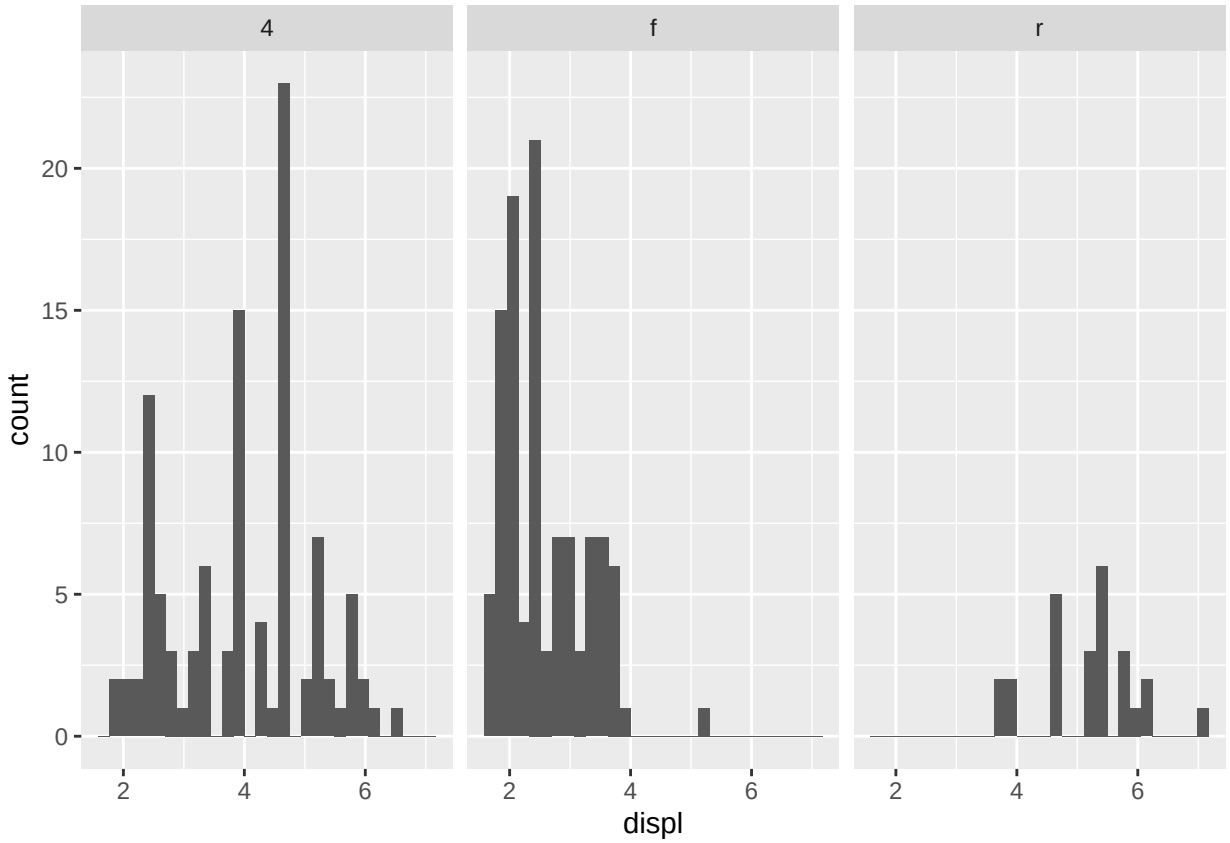
```
ggplot(mpg, aes(x = displ)) +
  geom_histogram() +
  facet_grid(drv ~ .)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
ggplot(mpg, aes(x = displ)) +
  geom_histogram() +
  facet_grid(. ~ drv)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



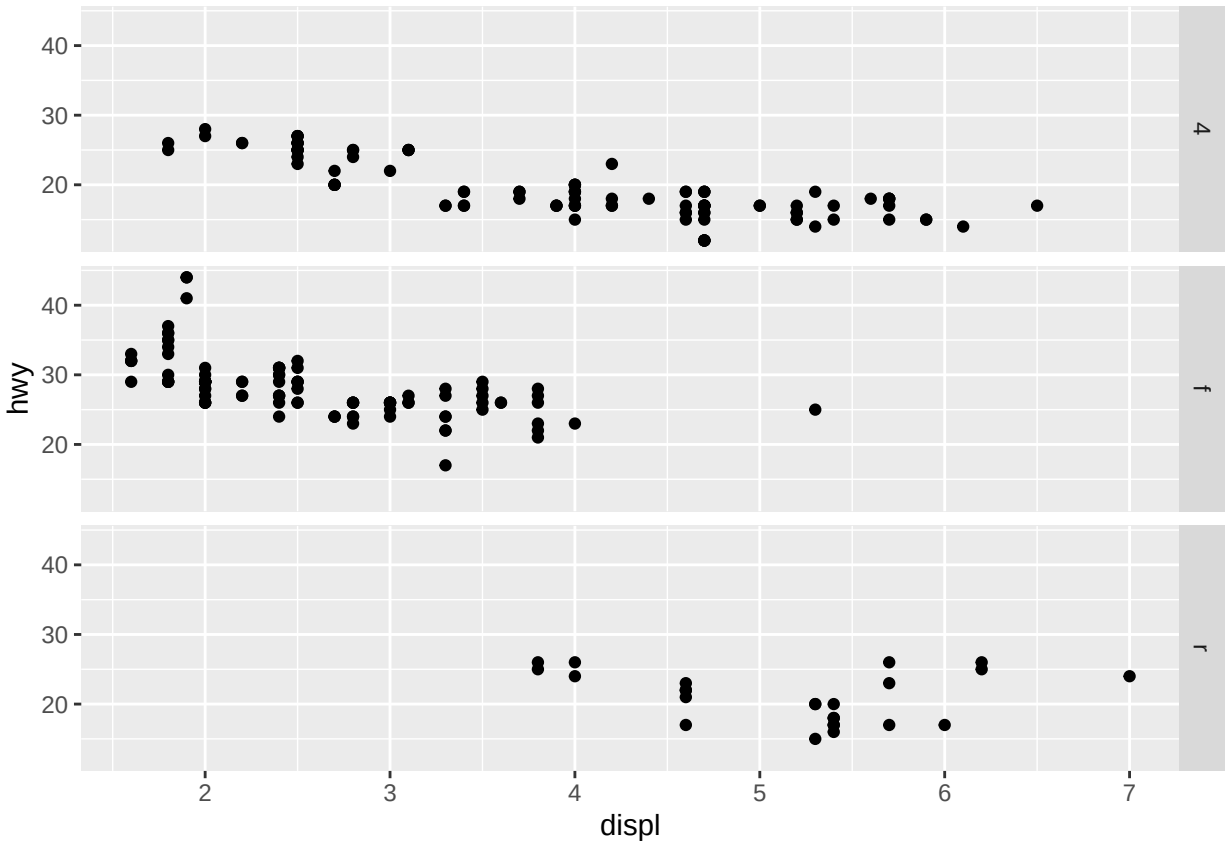
Answer: The first plot is easier to compare `displ` because the variable being compared is on the same axis. Comparison would be a matter of looking vertically at each `displ` bin to see how it varies across drive trains. This suggests that faceting should happen with columns if you're comparing across rows for a given value or with rows if you're comparing across columns for a given value.

7

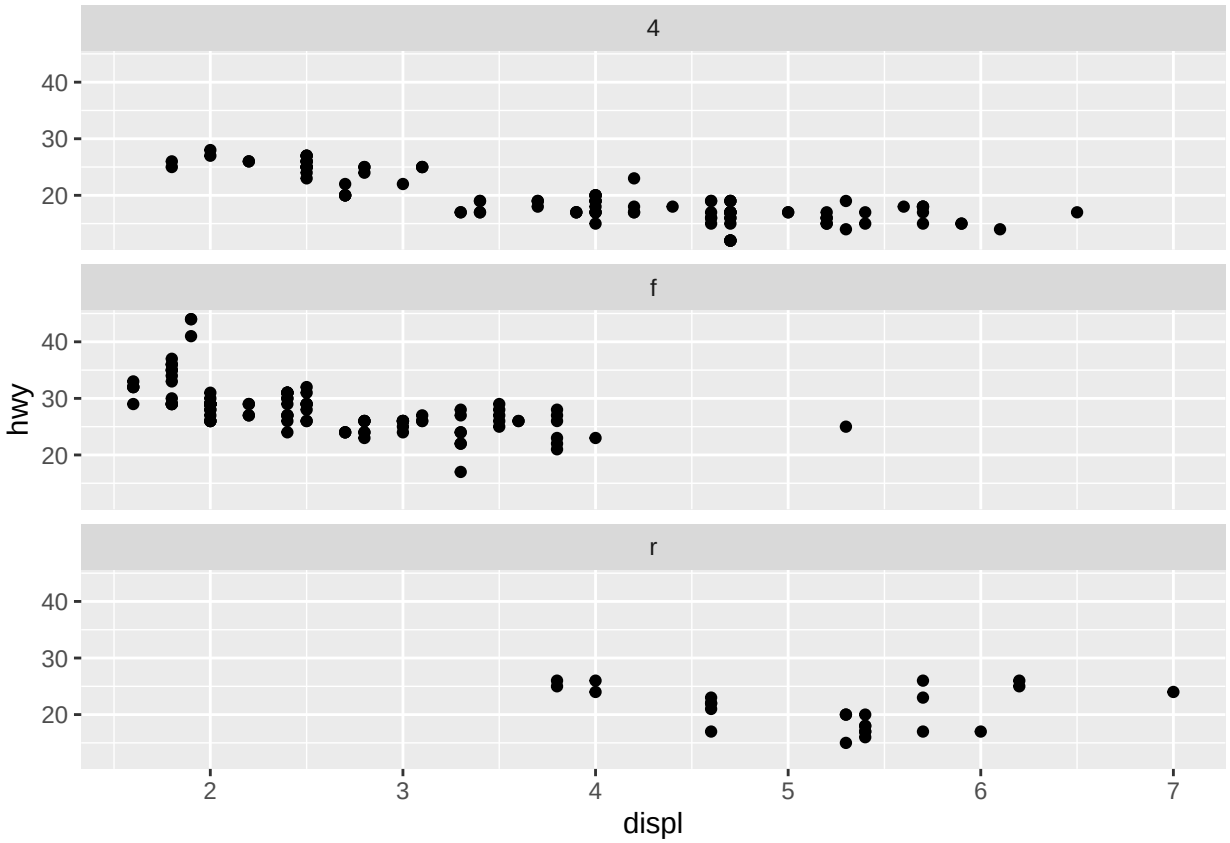
Recreate the following plot using facet_wrap() instead of facet_grid(). How do the positions of the facet labels change?

Answer: The facet labels move from being to the right of each group's plot to the top of the plots.

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




```
ggplot(mpg) +
  geom_point(aes(x = displ, y = hwy)) +
  facet_wrap(~ drv, nrow=3)
```



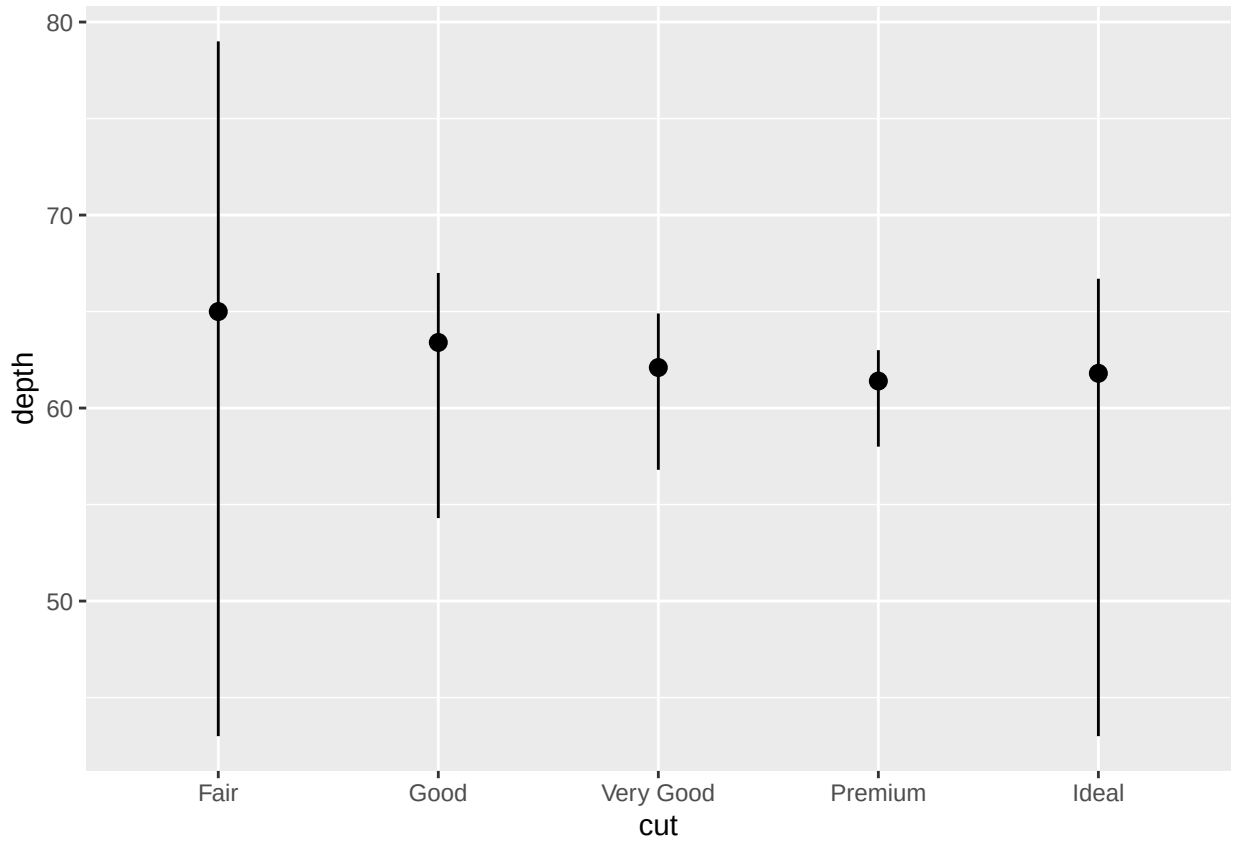
1

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?

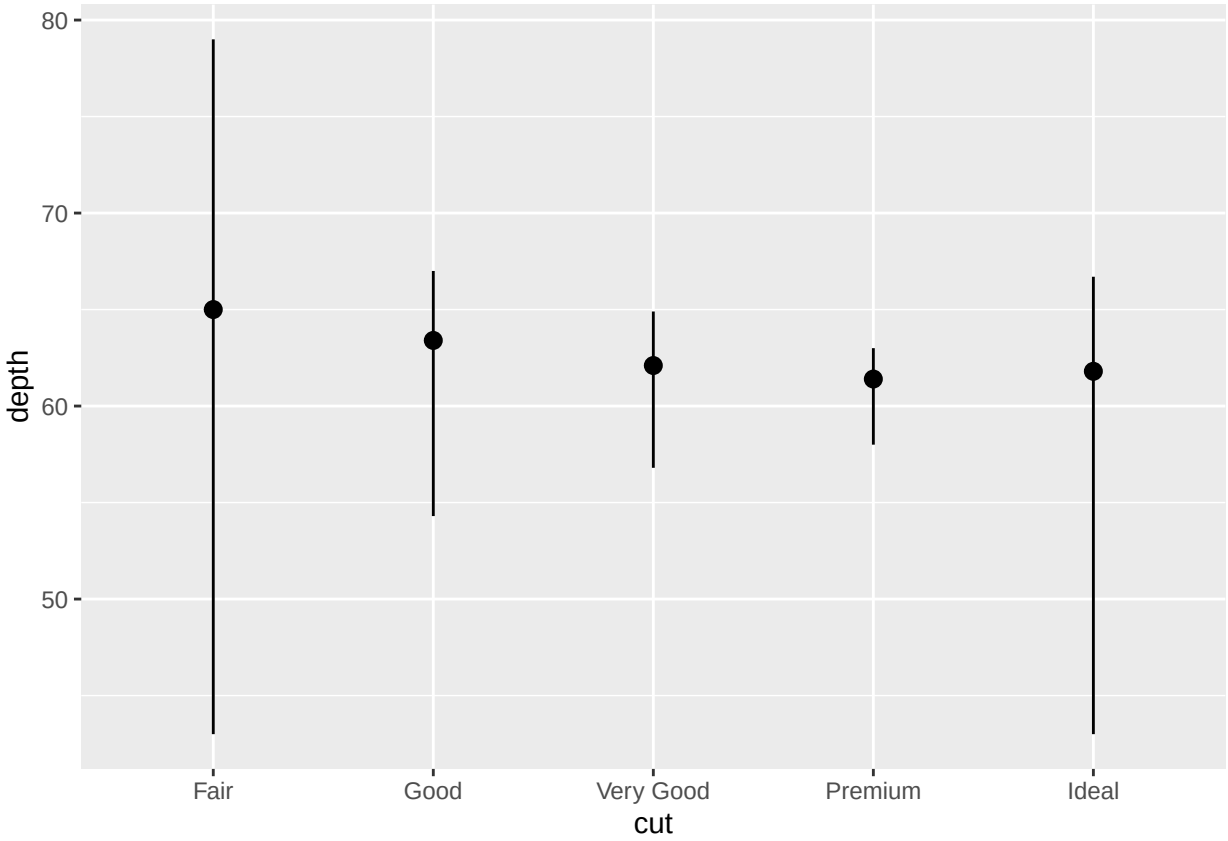
geom_pointrange() is the default geom associated with `stat_summary()`. You could rewrite the function with `geom_pointrange` by calculating the min, max, and median of depths for each cut. Then you would plot using these values with the aesthetic mapping of `aes(ymin = lower, ymax = upper)`

```
#?stat_summary

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




```
ggplot(diamonds) +
  geom_pointrange(
    aes(x = cut, y = depth),
    stat = "summary",
    fun.min = min,
    fun.max = max,
    fun = median
  )
```



2

What does geom_col() do? How is it different from geom_bar()?

Answer: `geom_col` uses the value in the data directly as the height of bars. `geom_bar` instead uses the count of each group as the height for the bar of the group.

```
#?geom_col
#?geom_bar
```

3

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

```
#?geom_pointrange
#?geom_col
#?geom_histogram
#?geom_point
#?geom_smooth
#?geom_density
#?geom_boxplot
```

Answer: geom_bar() and stat_count(). geom_histogram() and stat_bin(). geom_point() and stat_identity(). geom_smooth() and stat_smooth(). geom_density() and stat_density(). geom_boxplot() and stat_boxplot().

What they have in common is that the geom visualizes the outcome of the statistical transfor-

mation performed by the stat, often with the stat being the default for that geom, allowing for direct data visualization with minimal specification.

4

What variables does stat_smooth() compute? What arguments control its behavior?

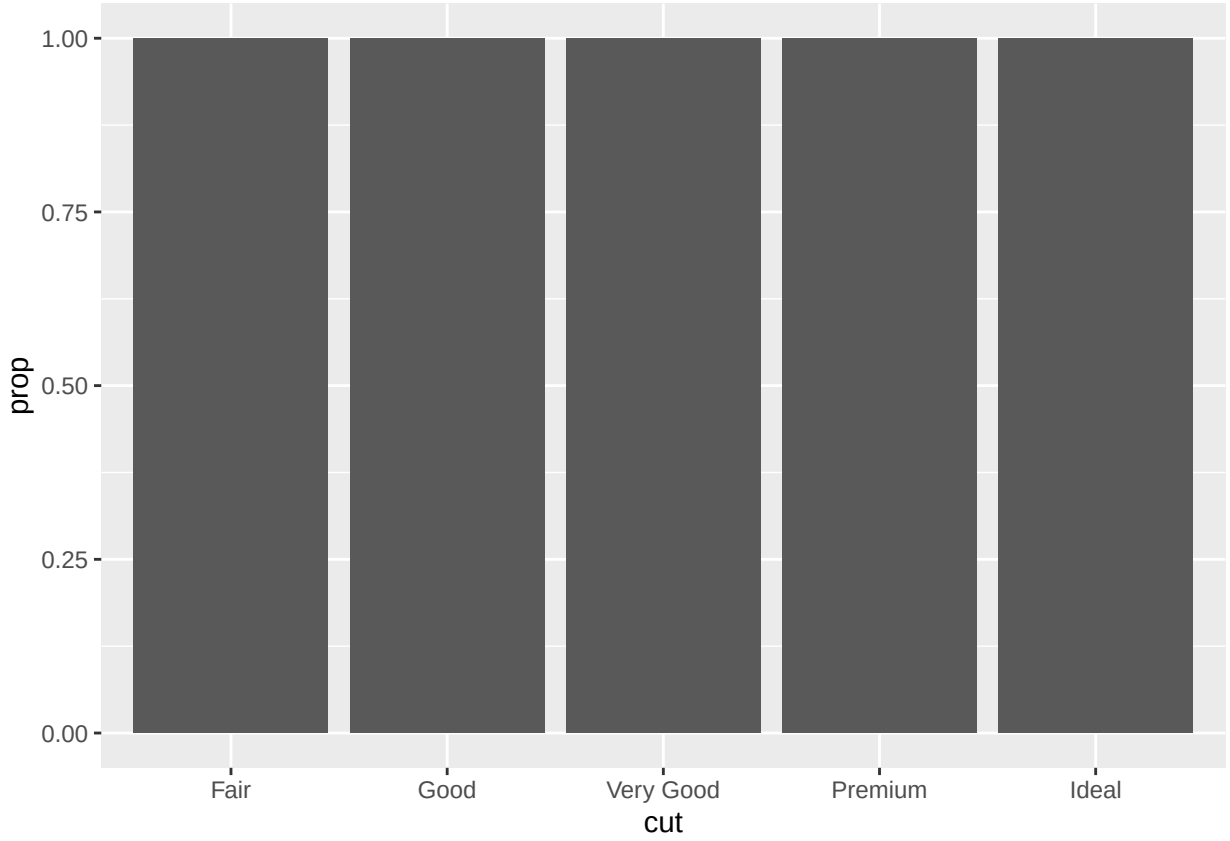
Answer: `stat_smooth` computes a smoothed line that computes the y based off the x variable. There are multiple ways of smoothing based on the specified `method`. There methods include: lm() for linear smooths, glm() for generalised linear smooths, and loess() for local smooths. The `method`: Specifies the smoothing method. `formula`: Defines the formula to use in the smoothing function. `se`: Determines whether to display the standard error around the smooth.

```
#?stat_smooth
```

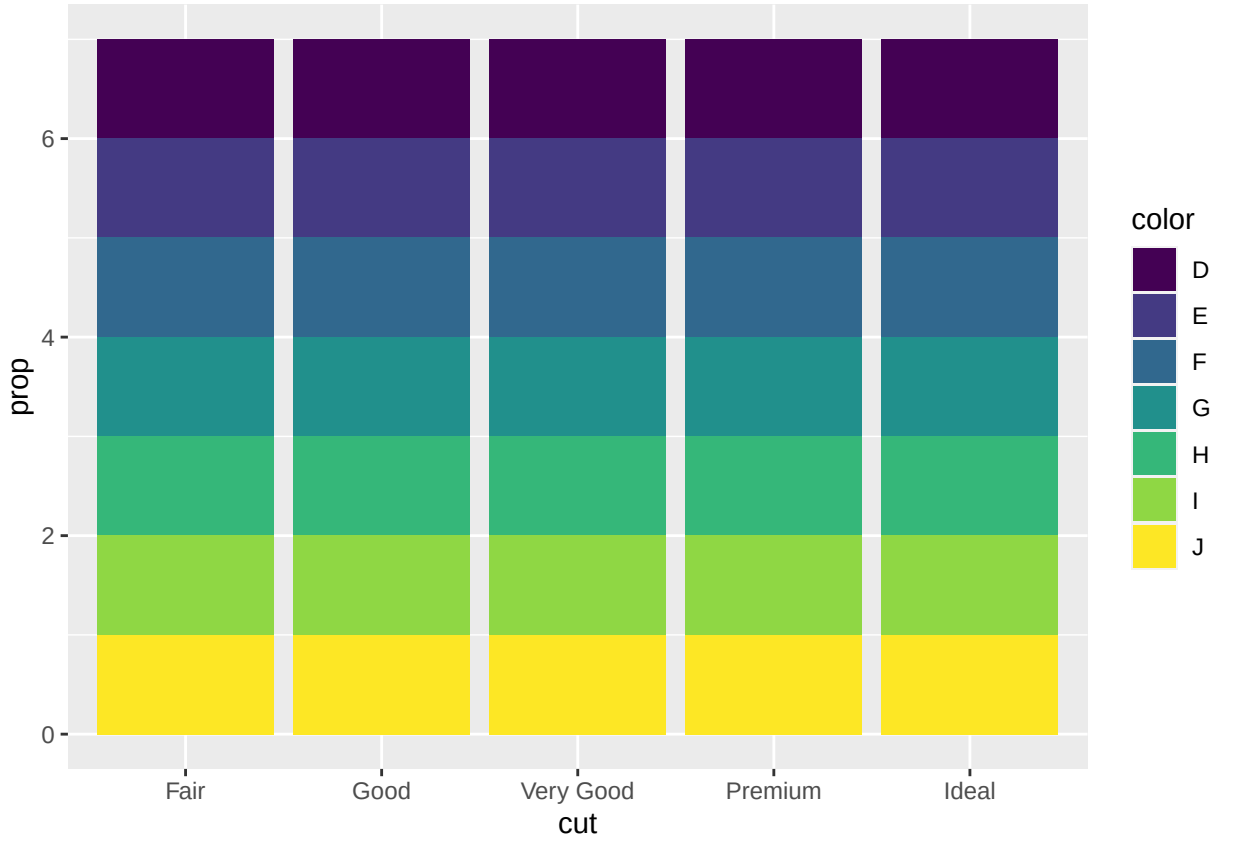
4

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

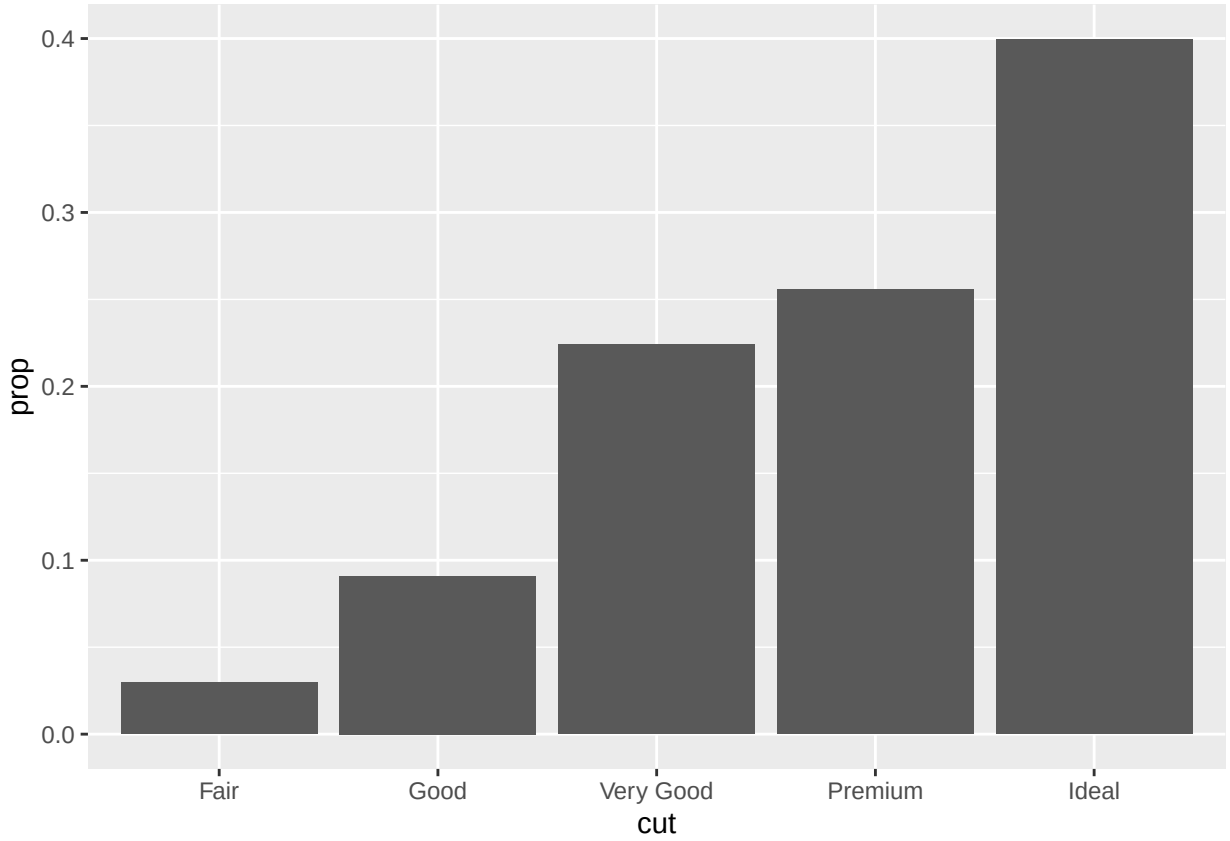
```
#?ggplot
#?after_stat
ggplot(diamonds, aes(x = cut, y = after_stat(prop))) +
      geom_bar()
```




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




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



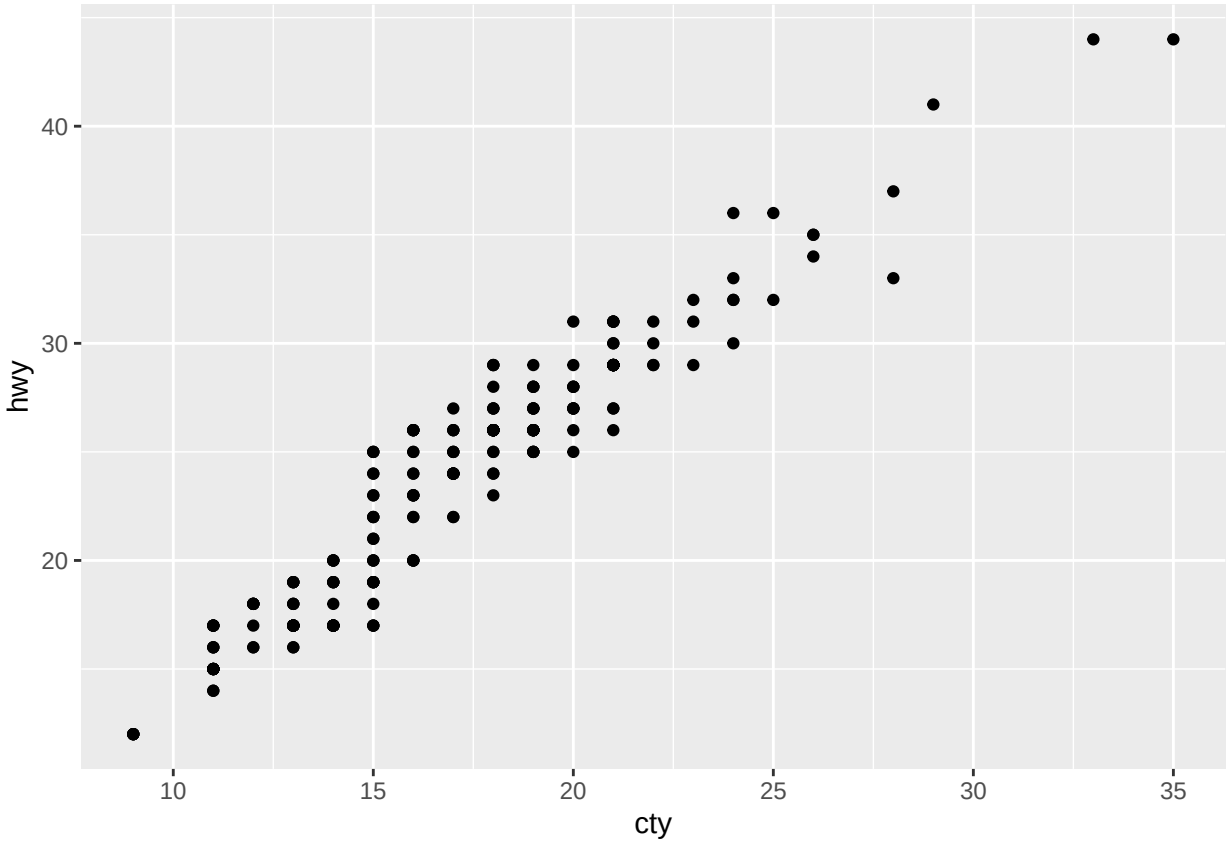
Answer: The issue with plot 1. Setting group = 1 in the proportion bar chart ensures that the proportion calculations are done over the entire dataset rather than separately within each group defined by another aesthetic. Without group = 1, the first graph would treat each cut as a separate group for proportion calculation. The issue with plot 2 is, each combination of cut and color would be treated as a separate group, leading to incorrect proportion calculations where proportions are calculated within each cut and color combination, not globally across all cut values.

9.6.1 Exercises

1

What is the problem with the following plot? How could you improve it?

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

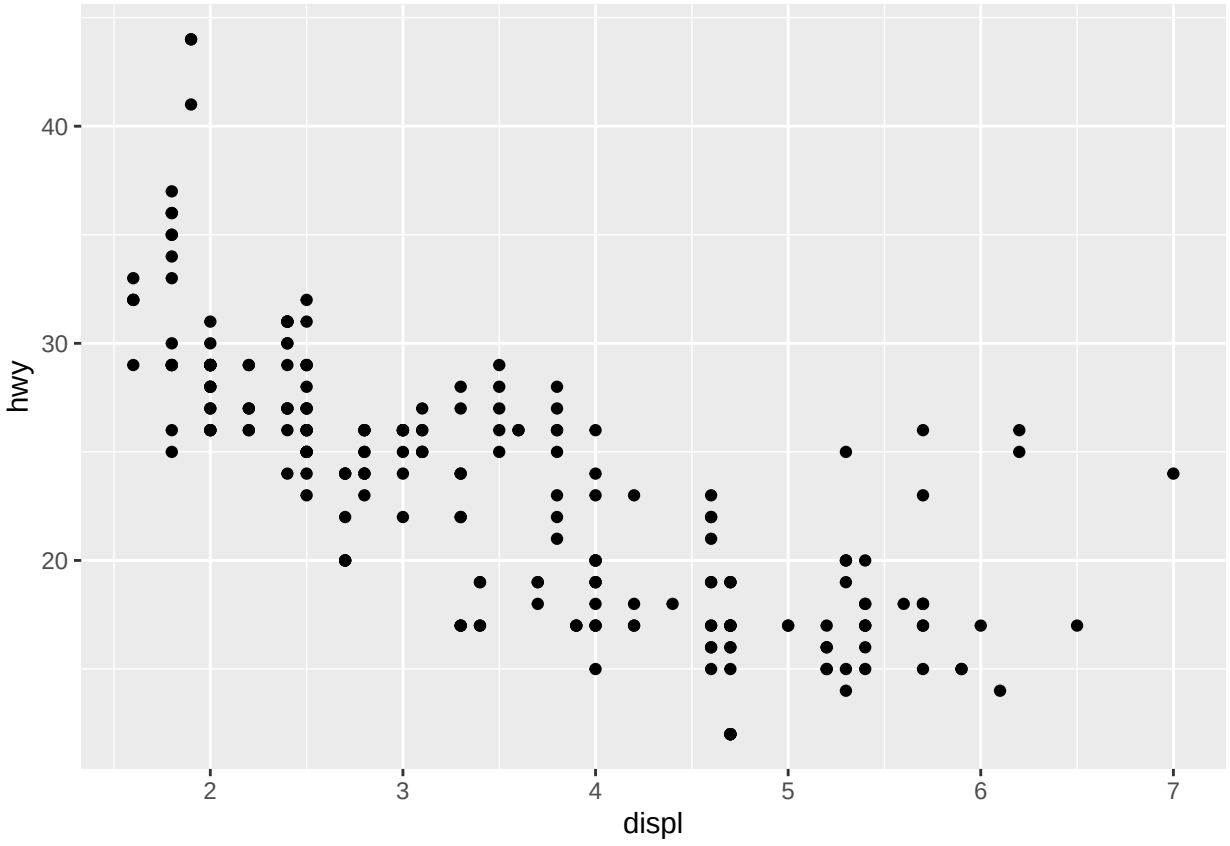


Answer: If you're quickly looking at the relationship of all cars' `cty` and `hwy`, this may be adequate. But improvements could include utilizing color or shape to distinguish between different groups, or incorporating a trend line to highlight any relationship between `cty` and `hwy`. We could also facet by car groupings.

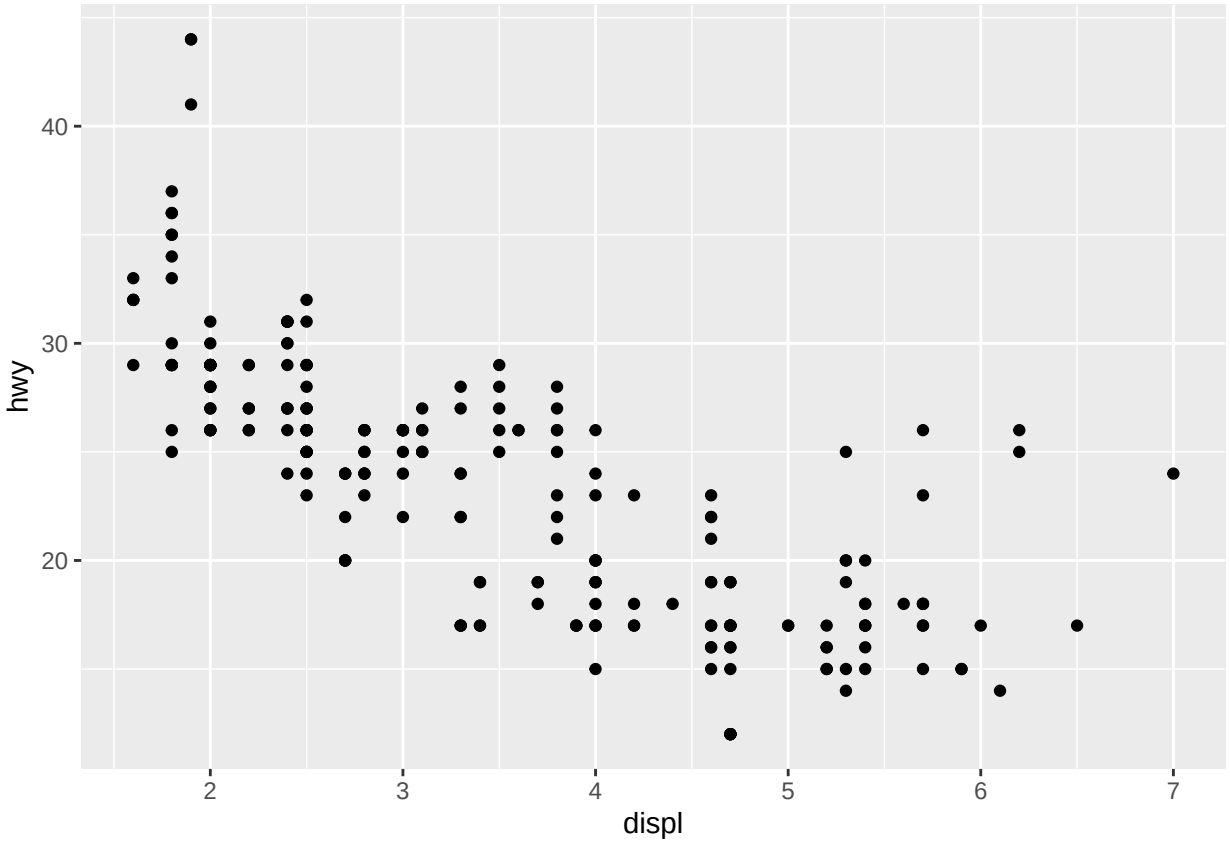
2

What, if anything, is the difference between the two plots? Why?

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




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



Answer: There seems to be no difference between the plots. The `position = "identity"` argument in the `geom_point()` function is the default behavior,

3

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

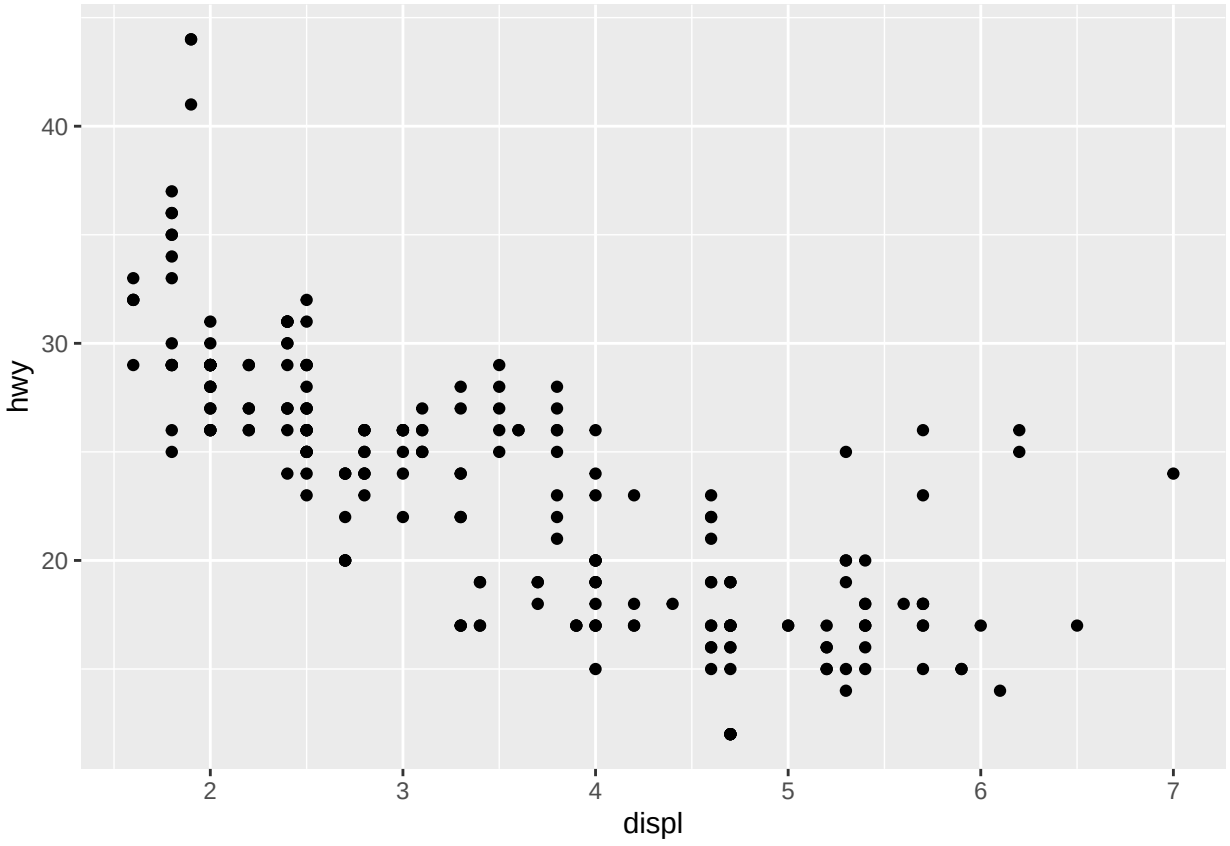
```
#?geom_jitter
```

Answer: `width` and `height` controls the amount of jittering.

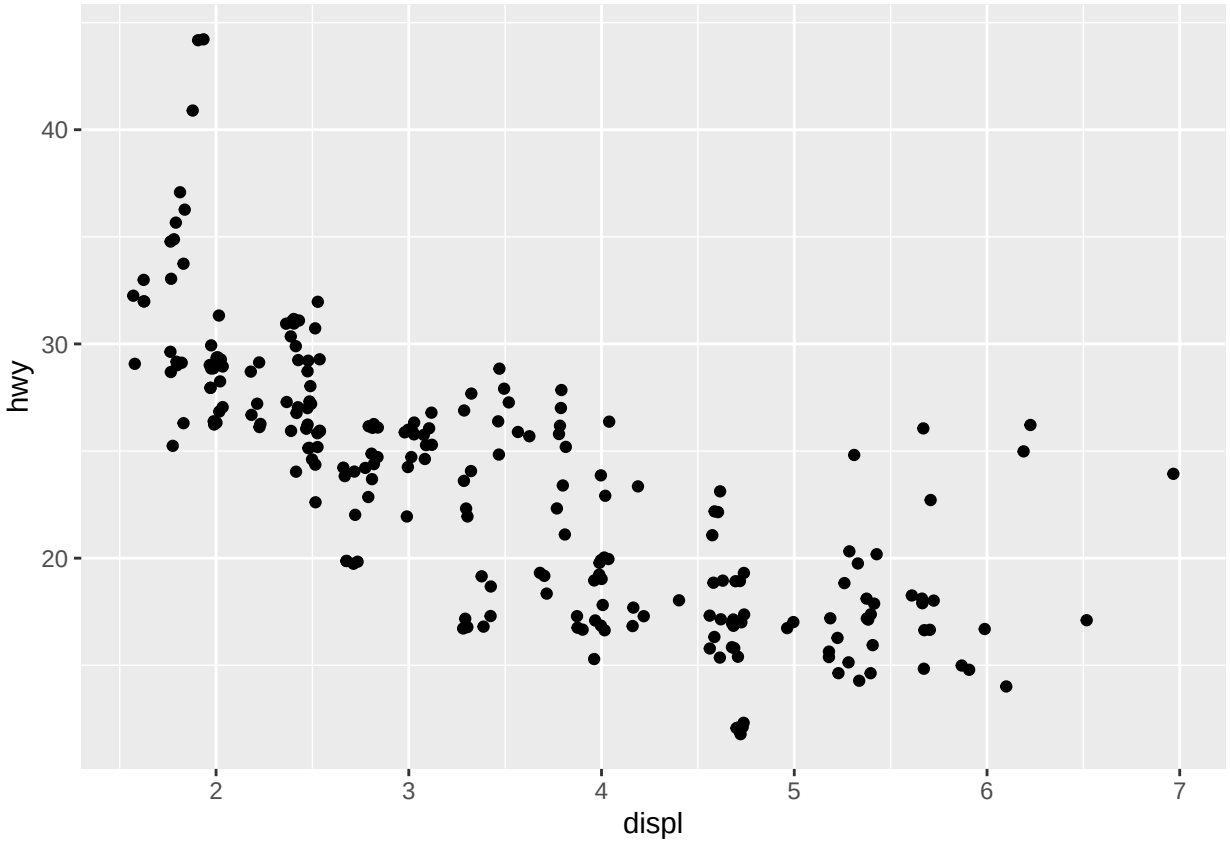
4

Compare and contrast geom_jitter() with geom_count().

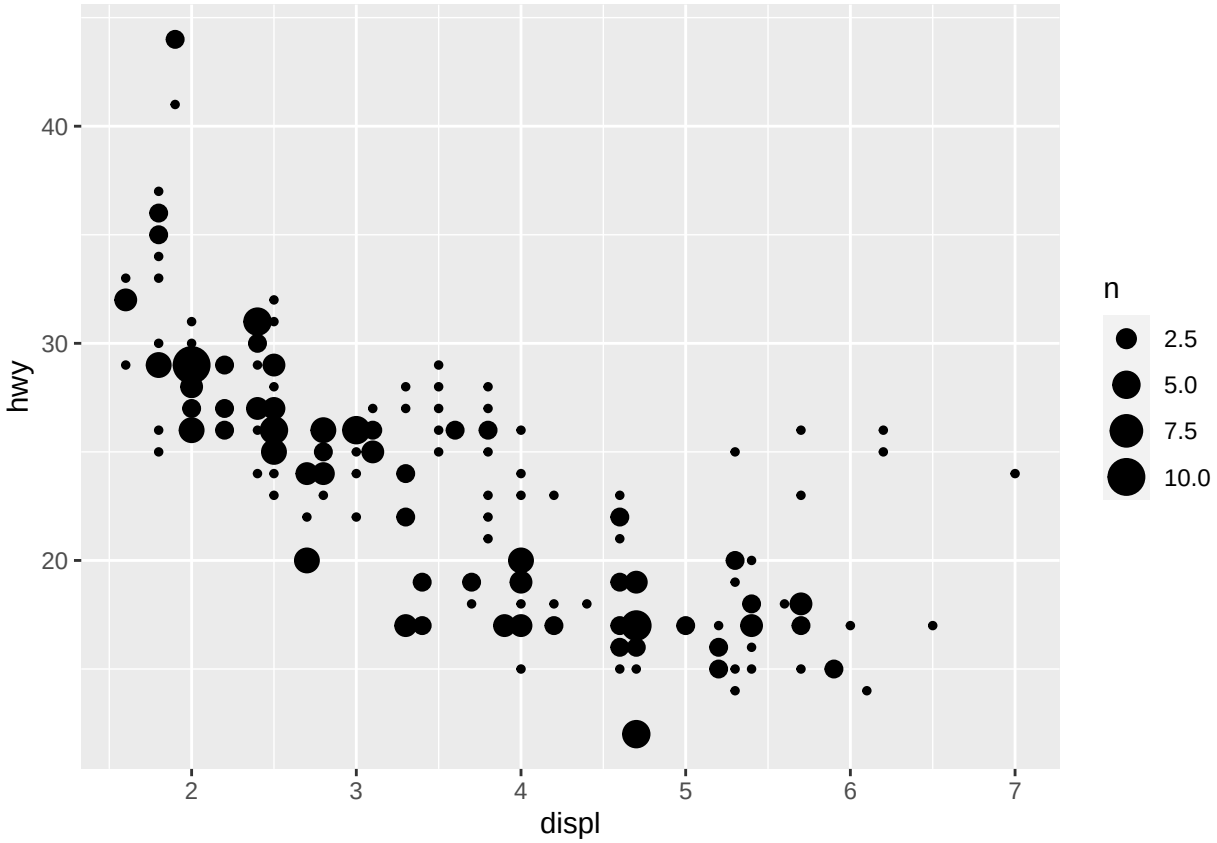
```
ggplot(mpg, aes(x = displ, y = hwy)) +
  geom_point()
```




```
ggplot(mpg, aes(x = displ, y = hwy)) +
  geom_jitter()
```




```
ggplot(mpg, aes(x = displ, y = hwy)) +
  geom_count()
```

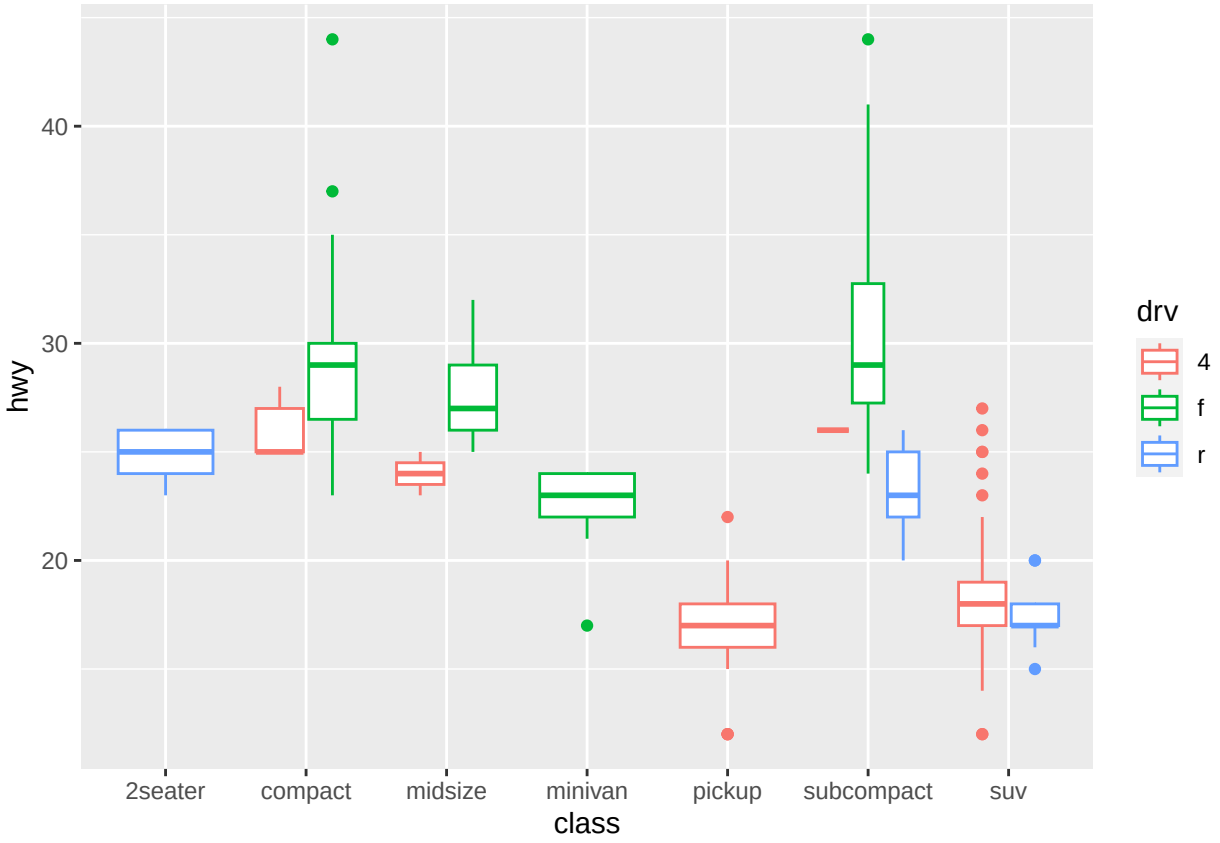


Answer: They are both variations of `geom_point`. `geom_jitter` creates a scatter plot using `geom_point` and adds a small random variation which helps with distinguishing between multiple points that are on the same location in a plot. `geom_count` creates a scatter plot using `geom_point` that creates dots that are different sizes based on the number of data points that occupy a given area.

5

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

```
#?geom_boxplot
ggplot(mpg, aes(class, hwy)) +
  geom_boxplot(aes(colour = drv))
```



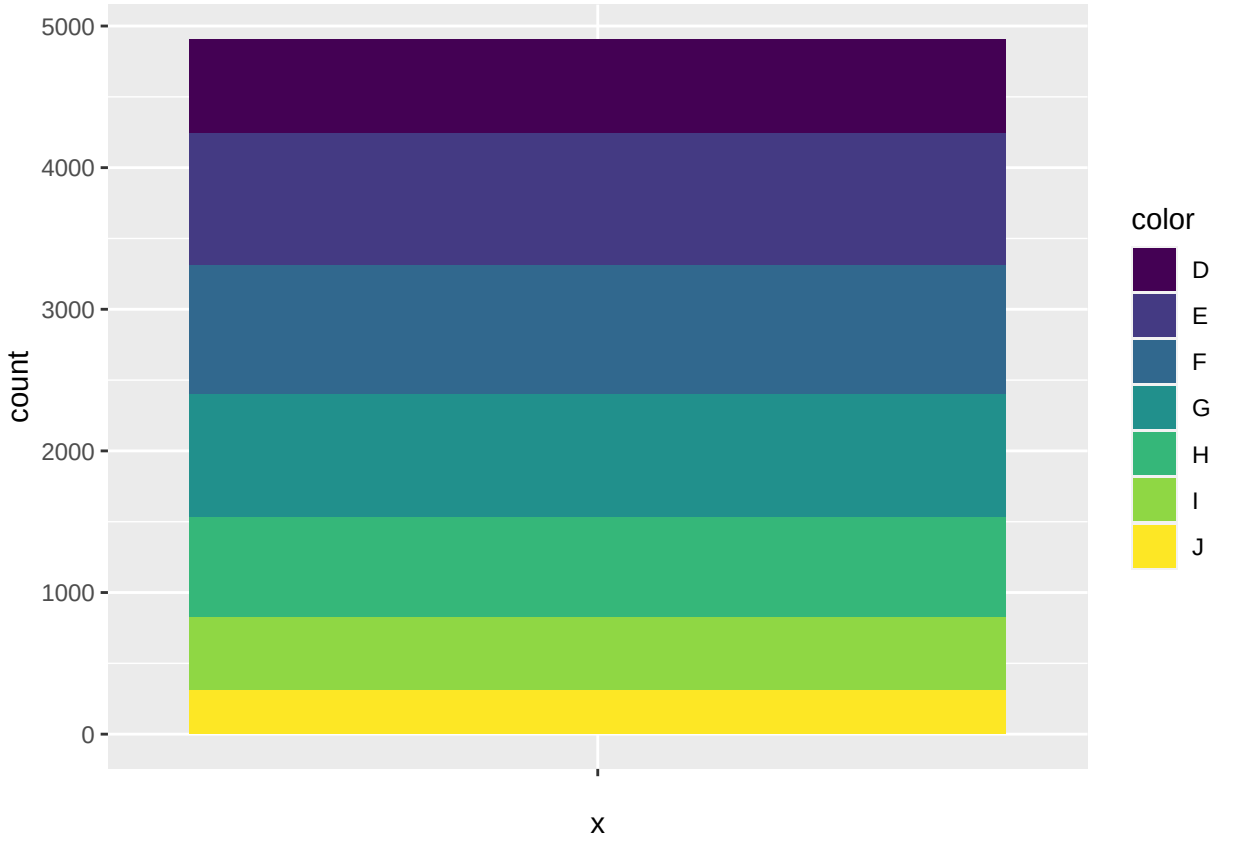
Answer: The default position adjustment for `geom_boxplot` is `dodge2` which automatically configures the boxplots so that they don't overlap and puts them side by side.

9.7.1 Exercises

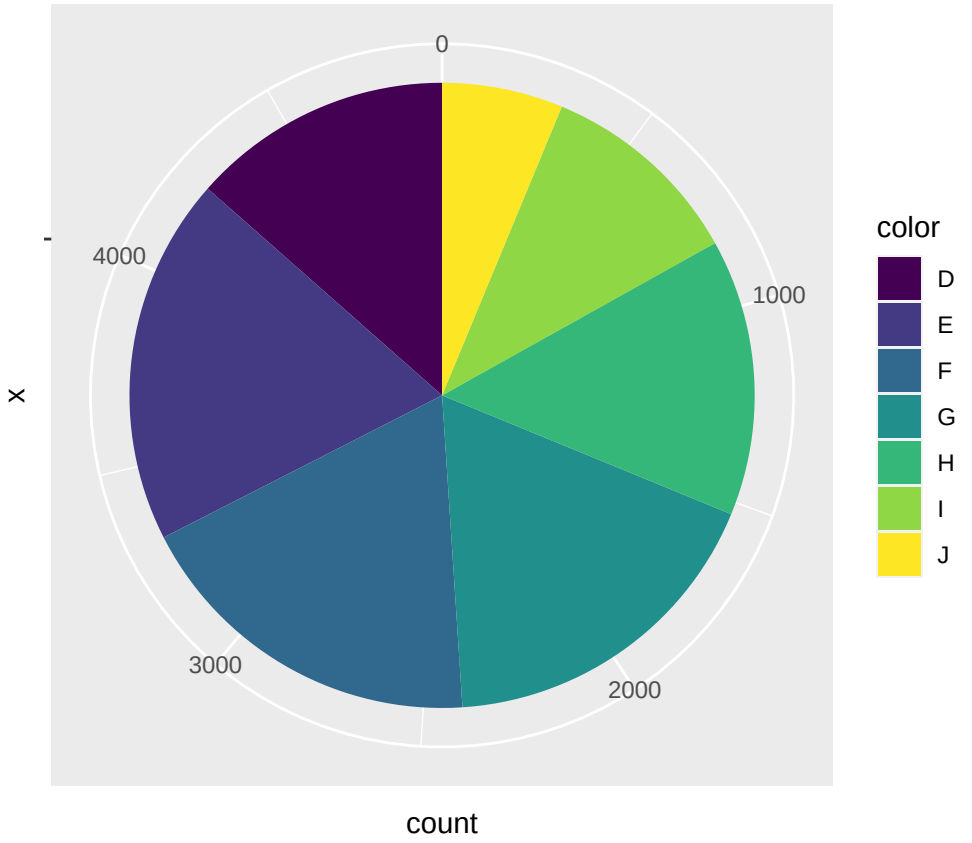
1

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

```
ggplot(filter(diamonds, cut == "Good"), aes(x = "", fill = color)) +
  geom_bar(width = 1)
```




```
ggplot(filter(diamonds, cut == "Good"), aes(x = "", fill = color)) +
  geom_bar(width = 1) +
  coord_polar(theta = "y")
```



2

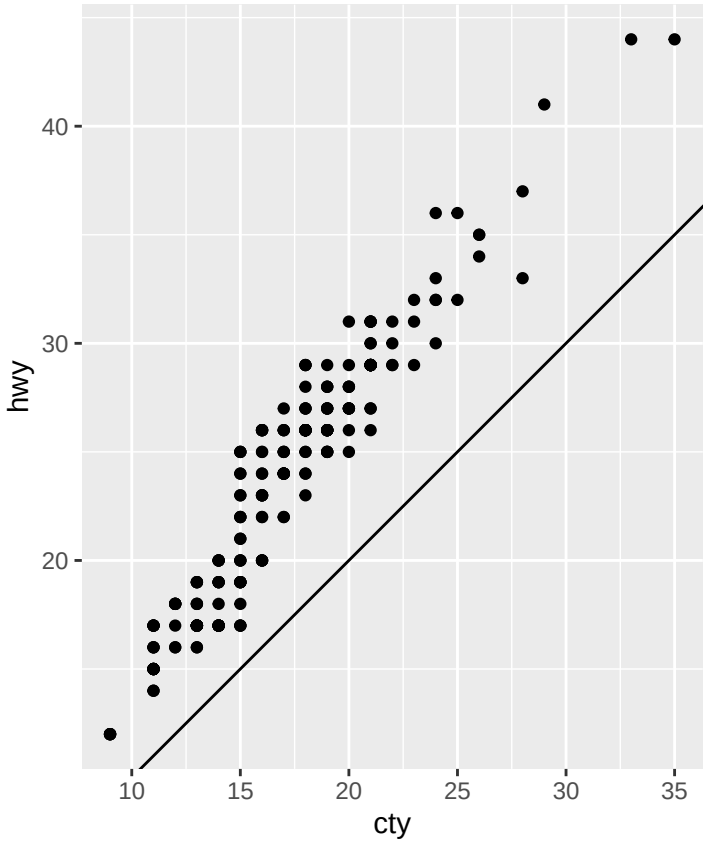
What's the difference between coord_quickmap() and coord_map()?

Answer: `coord_map()` projects spherical areas onto a 2D map without preserving the straight lines. It can handle larger geographic areas with more precision than `coord_quickmap()`. `coord_quickmap()` is used for more approximate plots that preserve straight lines from maps which requires less computation at the cost of accuracy. It is most suitable for mapping smaller areas that are closer to the equator.

3

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

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




```
#?coord_fixed
#?geom_abline
```

Answer: coord_fixed() uses a fixed scale coordinate system where the number of units on the x and y-axes are equivalent. Since all the points are above the line, the highway mileage is always greater than city mileage for these cars.geom_abline() adds reference lines (sometimes called rules) to a plot, either horizontal, vertical, or diagonal (specified by slope and intercept).

10.3.3 Exercises

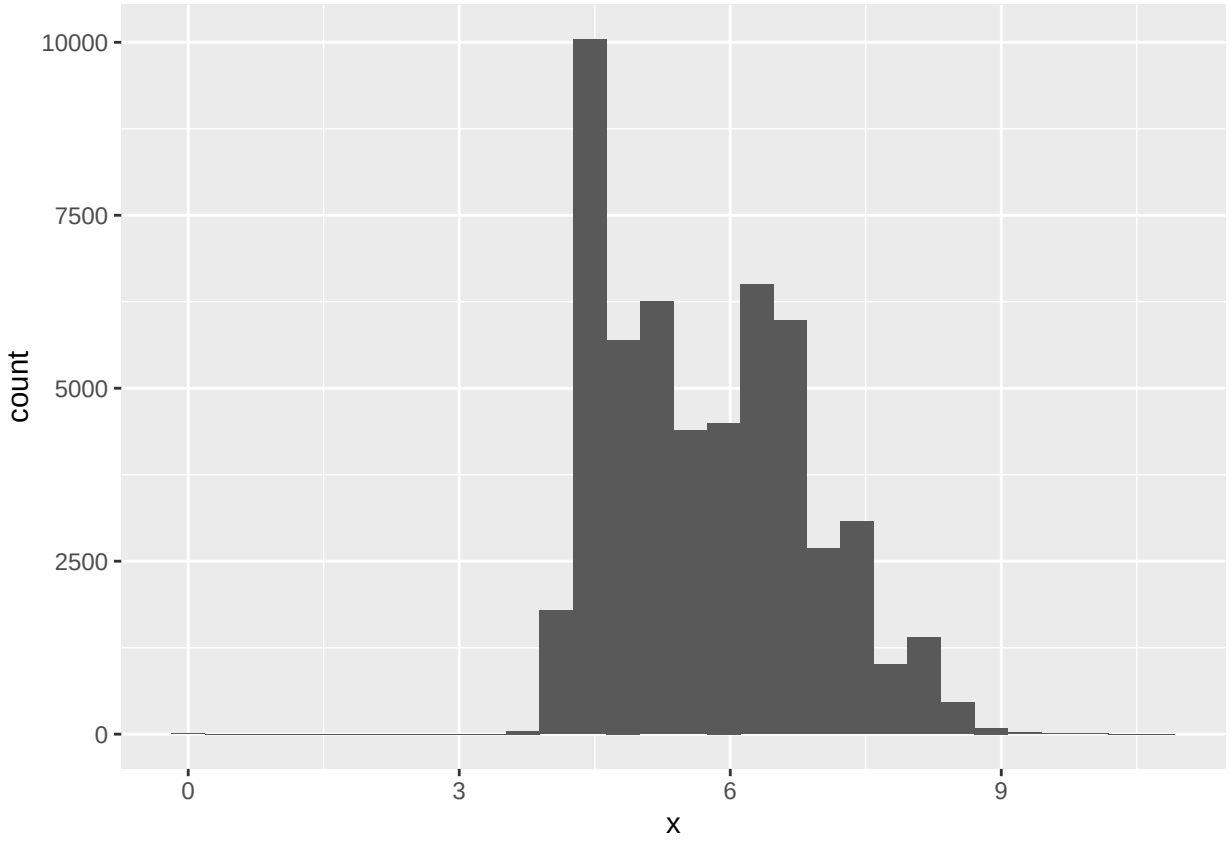
1

Explore the distribution of each of the x, y, and z variables in diamonds. What do you learn? Think about a diamond and how you might decide which dimension is the length, width, and depth.

Answer: To explore the distribution of the x, y, and z variables in the diamonds dataset, you can create histograms for each variable. Additionally, you can use summary statistics to gain insights into the distribution. By examining the histograms and summary statistics, you can learn about the central tendency, spread, and shape of the distributions for the x, y, and z variables. This information is valuable for understanding the variation in the dimensions of the diamonds in the dataset.x: Represents the length. y: Represents the width.z: Represents the depth.I used the documentation to determine these.

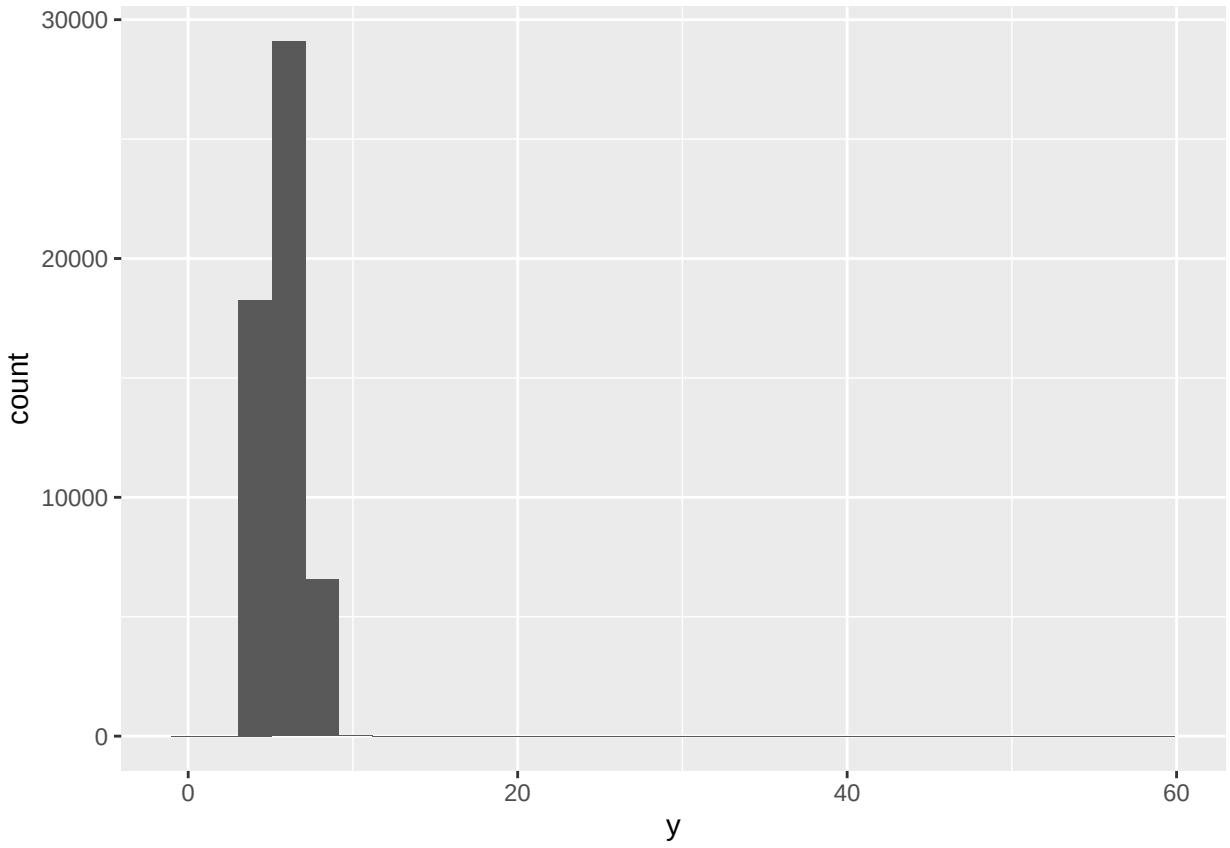
```
# Create histograms for x, y, and z
ggplot(diamonds, aes(x = x)) +
  geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



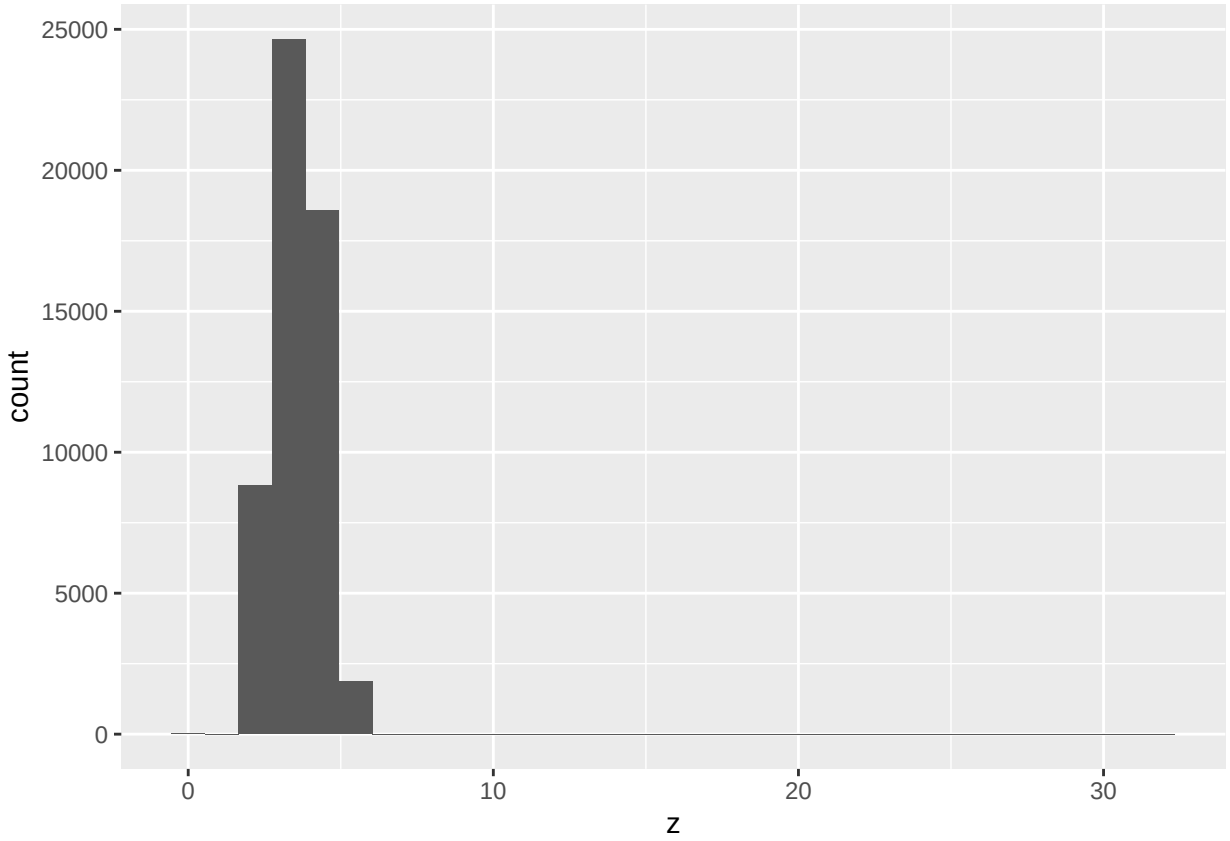

```
ggplot(diamonds, aes(x = y)) +
  geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
ggplot(diamonds, aes(x = z)) +
  geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
# Summary statistics for x, y, and z
summary(diamonds$x)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   4.710   5.700   5.731   6.540  10.740
```

```
summary(diamonds$y)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   4.720   5.710   5.735   6.540  58.900
```

```
summary(diamonds$z)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   2.910   3.530   3.539   4.040  31.800
```

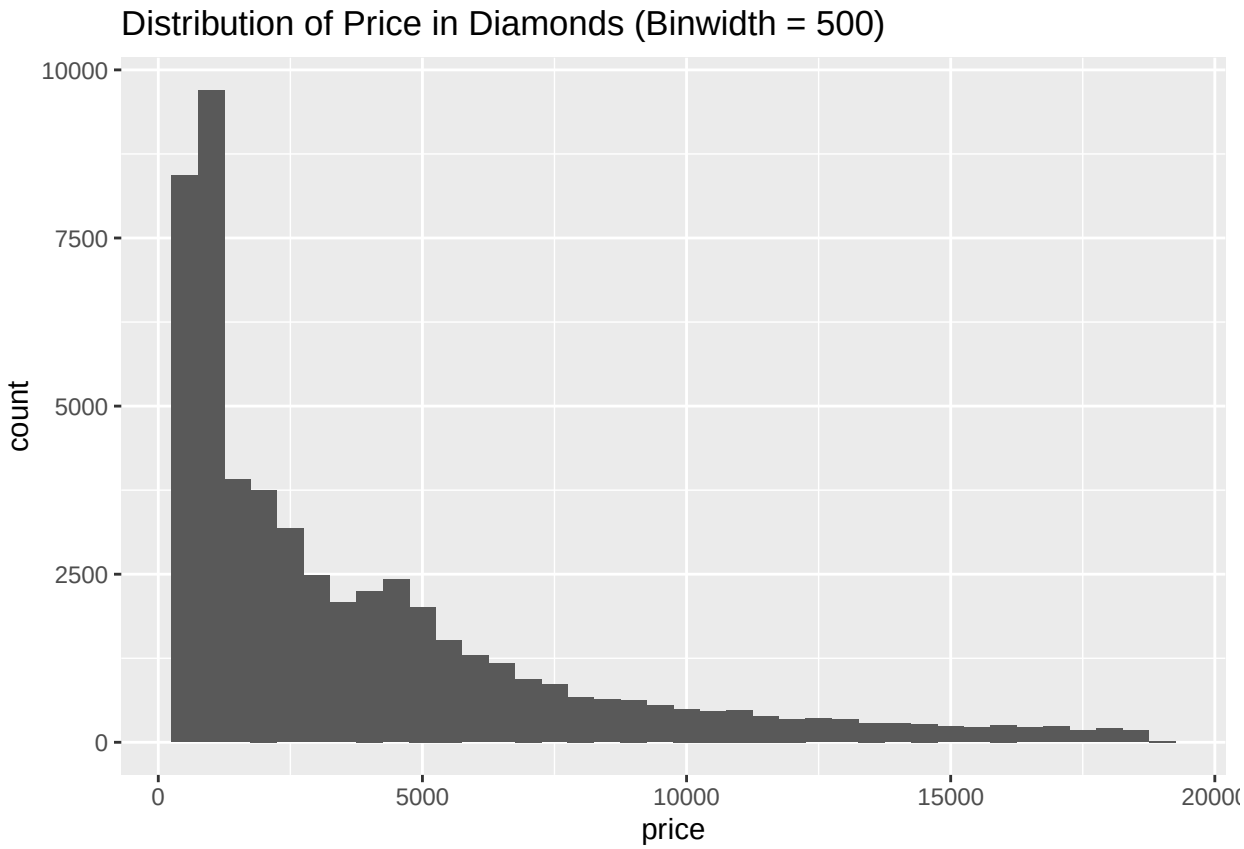
```
#?diamonds
```

2

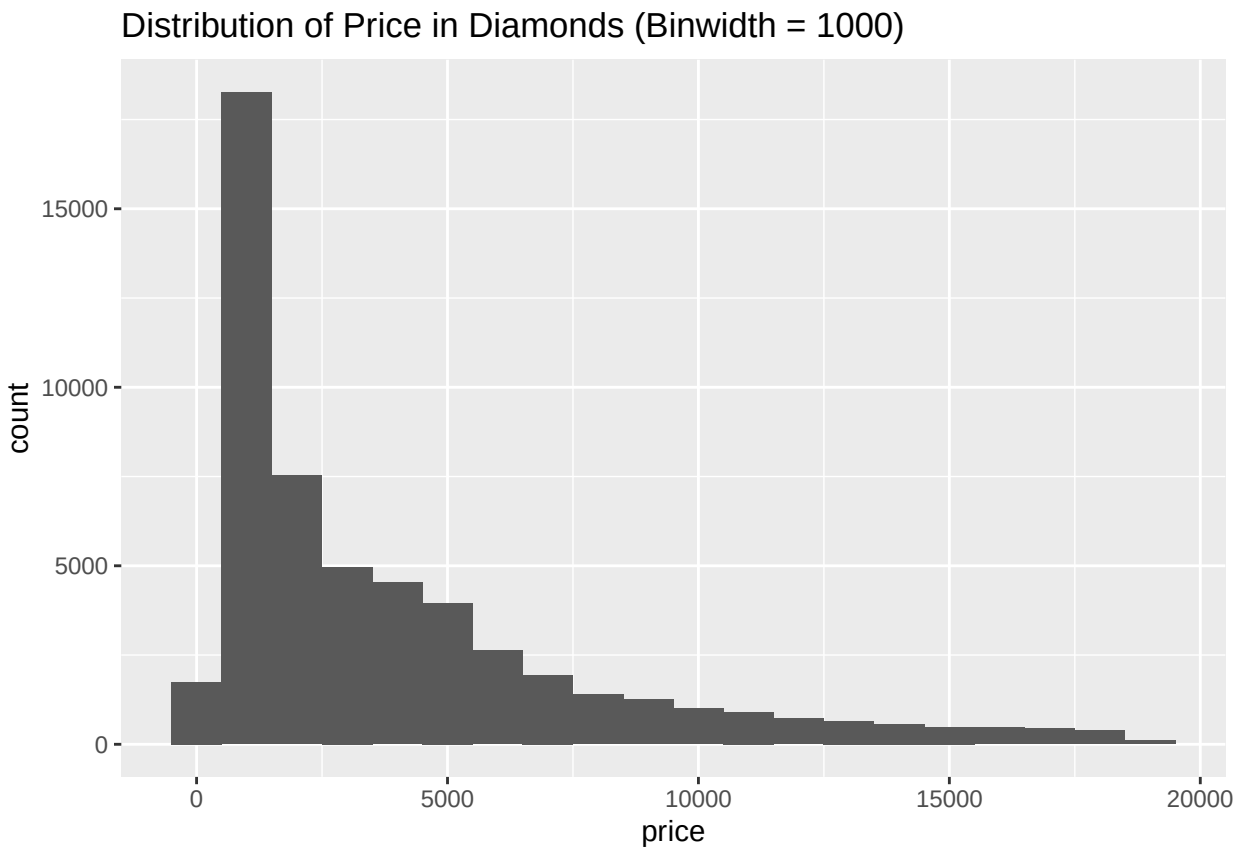
Explore the distribution of price. Do you discover anything unusual or surprising? (Hint: Carefully think about the binwidth and make sure you try a wide range of values.)

Answer: Smaller binwidths provide more detail, while larger binwidths may smooth out the distribution. It it surprising that there is a steep dip in number of diamonds after $1000

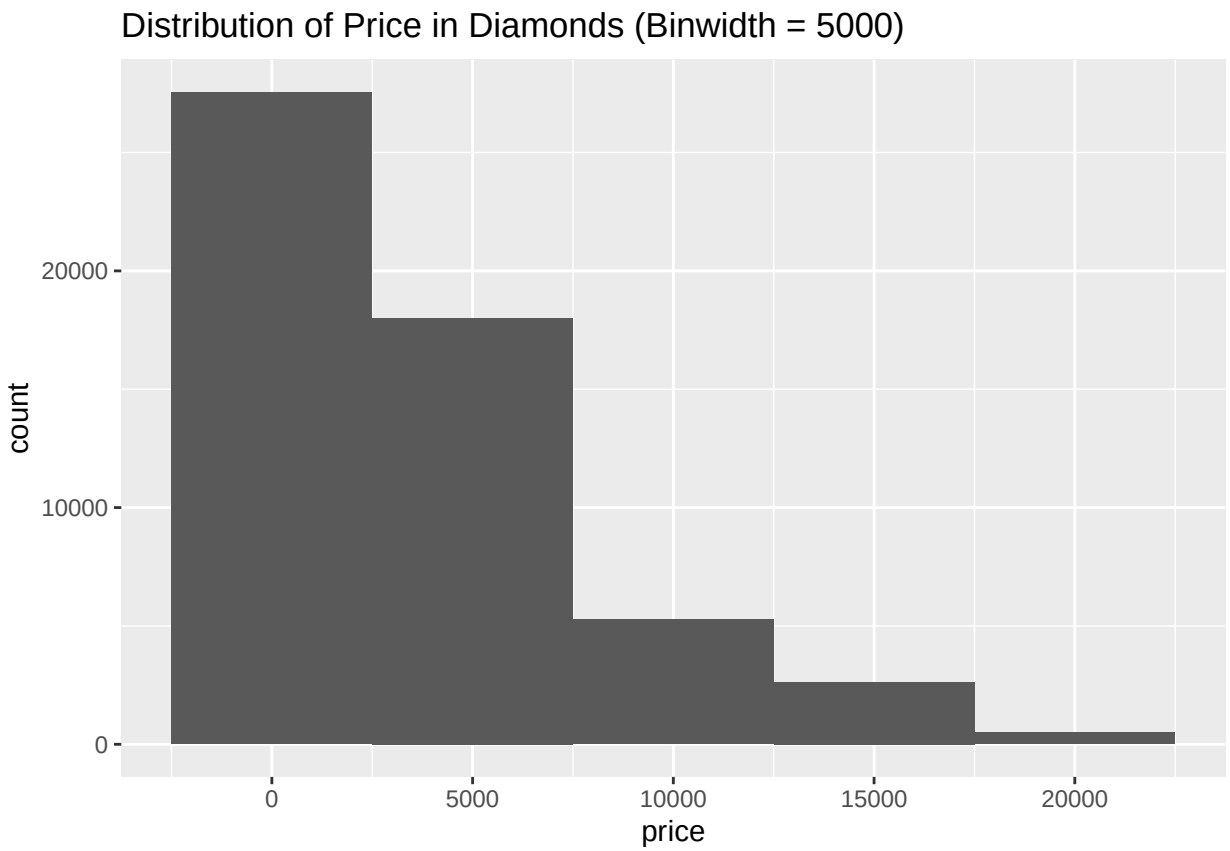
```
ggplot(diamonds, aes(x = price)) +
  geom_histogram(binwidth = 500) +
  labs(title = "Distribution of Price in Diamonds (Binwidth = 500)")
```




```
ggplot(diamonds, aes(x = price)) +
  geom_histogram(binwidth = 1000) +
  labs(title = "Distribution of Price in Diamonds (Binwidth = 1000)")
```




```
ggplot(diamonds, aes(x = price)) +
  geom_histogram(binwidth = 5000) +
  labs(title = "Distribution of Price in Diamonds (Binwidth = 5000)")
```



3

How many diamonds are 0.99 carat? How many are 1 carat? What do you think is the cause of the difference?

There are only 23 .99 carats and there are 1558 1 carats. The cause of the difference could be due to rounding errors or human error while inputting the data and or recording the data.

```
# Count of diamonds with 0.99 carats
diamonds |>
  filter(carat == 0.99) |>
  nrow()
```

```
## [1] 23
```

```
# Count of diamonds with 1 carat
diamonds |>
  filter(carat == 1) |>
  nrow()
```

```
## [1] 1558
```

4

Compare and contrast coord_cartesian() vs. xlim() or ylim() when zooming in on a histogram. What happens if you leave binwidth unset? What happens if you try and zoom so only half a bar shows?

`coord_cartesian` adjusts the visible portion of the plot without changing the underlying data. This means that all data points are still considered for calculations like statistics or binning in the case of histograms.Whereas `xlim()` or `ylim()` directly modifies the limits of the data

that will be considered for plotting. Data points outside these limits are excluded from the visualization and calculations. If you leave the binwidth unset it will not affect the binning of the data. The binwidth is determined based on the original data range. For `coord_cartesian` if you try to zoom in so that only half a bar shows, the plot will display the visible portion without modifying the underlying data. For `xlim()` or `ylim()` if you try to zoom in so that only half a bar shows, the statistics are calculated only for the visible portion.

10.4.1 Exercises

1

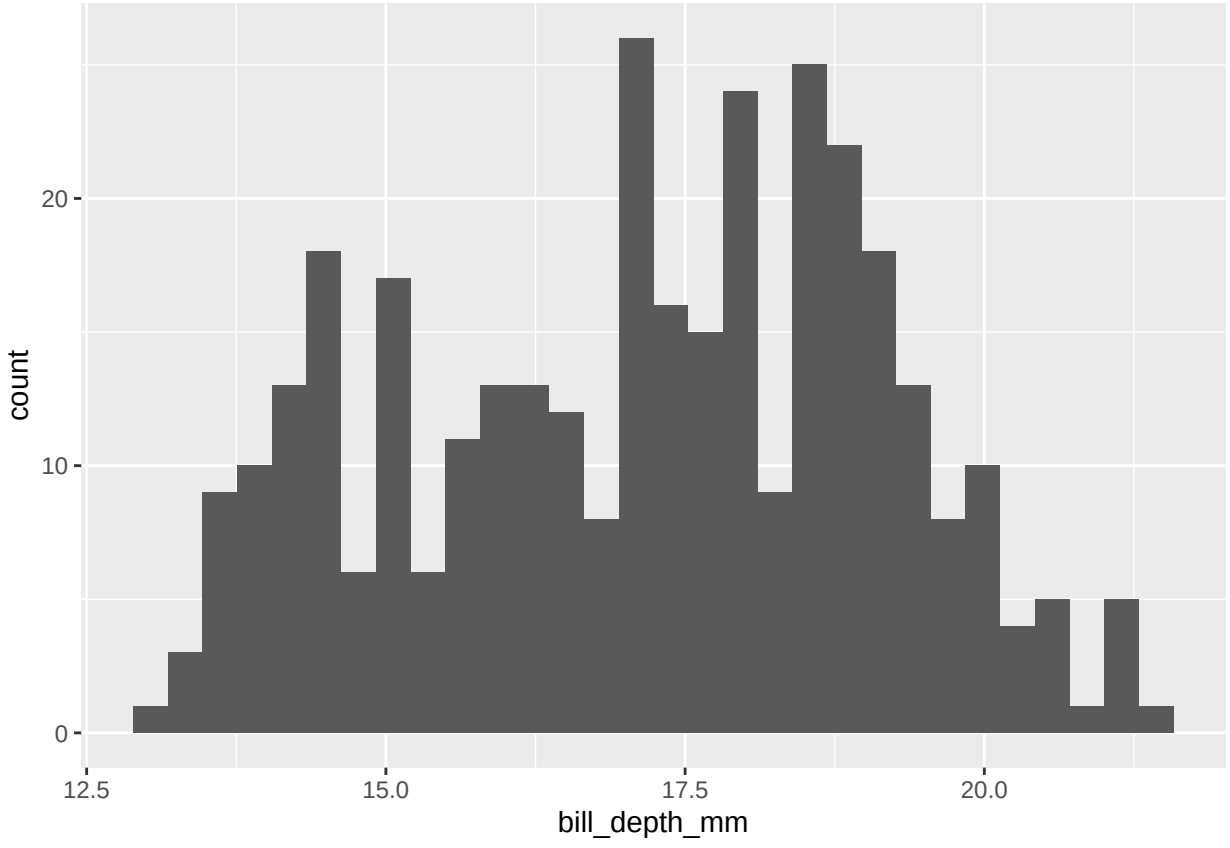
What happens to missing values in a histogram? What happens to missing values in a bar chart? Why is there a difference in how missing values are handled in histograms and bar charts?

In a histogram, missing values are typically ignored. In a bar chart, missing values may be explicitly represented as a separate category. Each bar corresponds to a category, and the handling of missing values depends on whether they are visibly included or labeled as "Missing." The difference in treatment stems from the distinct nature of the data, with histograms focusing on numerical distributions and bar charts emphasizing categorical comparisons.

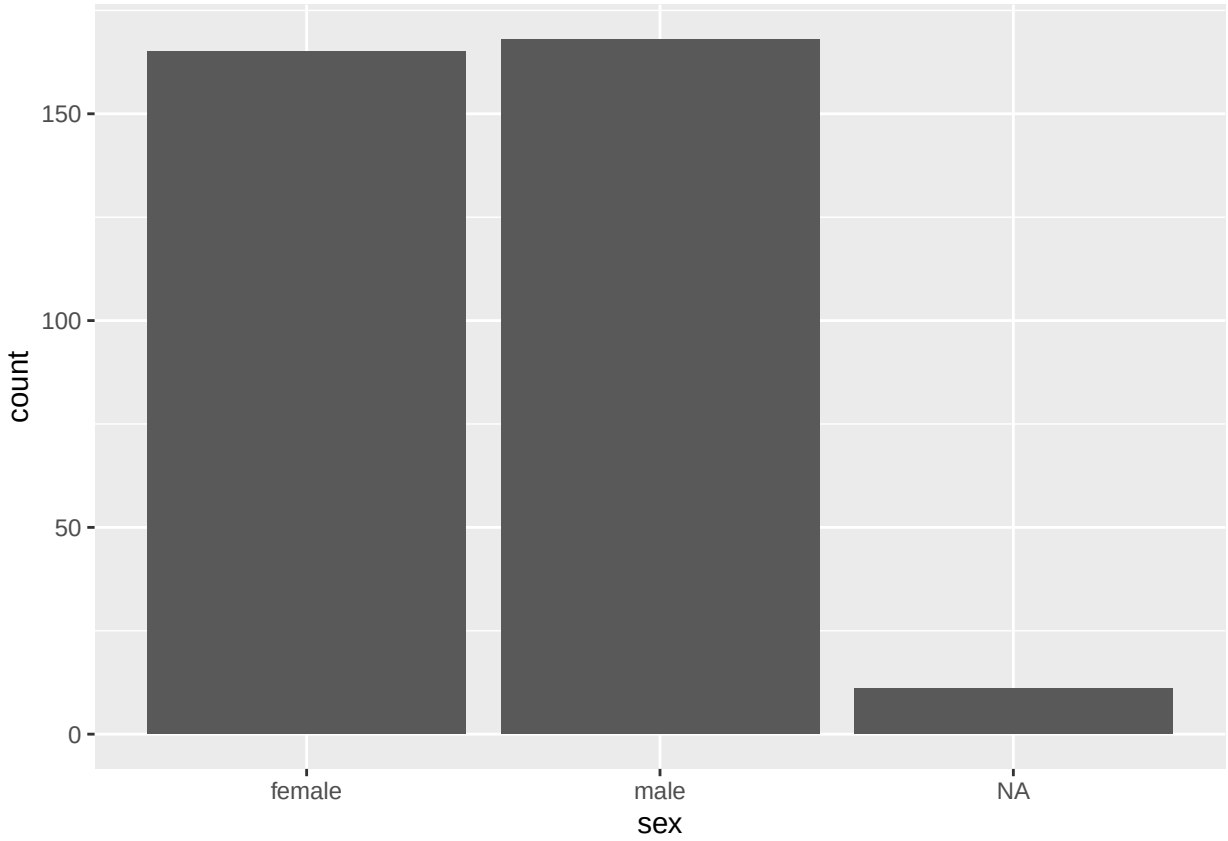
```
ggplot(penguins, mapping=aes(bill_depth_mm))+
  geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## Warning: Removed 2 rows containing non-finite values (`stat_bin()`).
```




```
ggplot(penguins, mapping = aes(x = sex))+
  geom_bar()
```



2

What does na.rm = TRUE do in mean() and sum()?

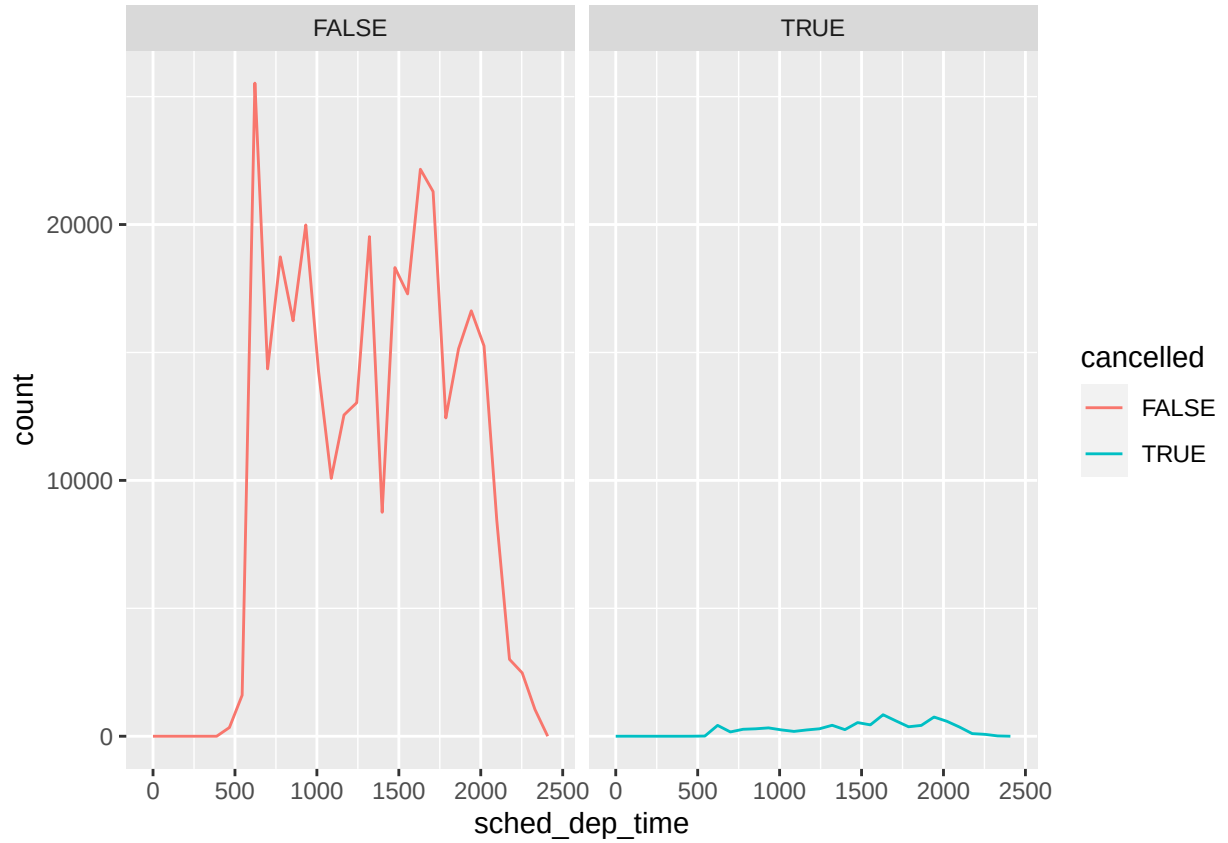
`na.rm` parameter is used in functions like `mean()` and `sum()` to specify whether missing values (`NA`) should be removed before performing the calculation. When `na.rm = TRUE`, it means that missing values are included, and the calculation is performed using all the values.

3

Recreate the frequency plot of scheduled_dep_time colored by whether the flight was cancelled or not. Also facet by the cancelled variable. Experiment with different values of the scales variable in the faceting function to mitigate the effect of more non-cancelled flights than cancelled flights.

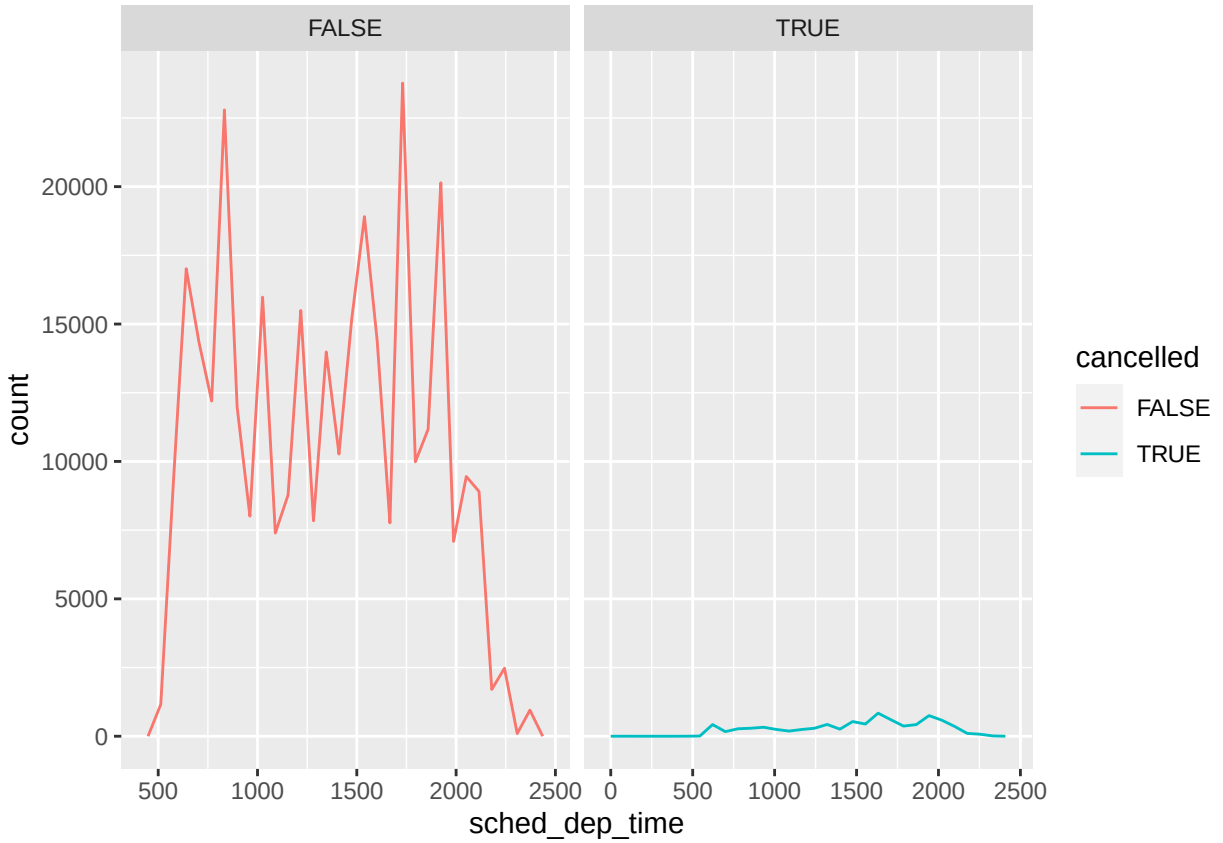
```
mutate(flights, cancelled = is.na(dep_time)) |>
  ggplot(aes(x = sched_dep_time)) +
  geom_freqpoly(aes(color = cancelled)) +
  facet_wrap(~cancelled)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
mutate(flights, cancelled = is.na(dep_time)) |>
  ggplot(aes(x = sched_dep_time)) +
  geom_freqpoly(aes(color = cancelled)) +
  facet_wrap(~cancelled, scales = "free_x")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
mutate(flights, cancelled = is.na(dep_time)) |>
  ggplot(aes(x = sched_dep_time)) +
  geom_freqpoly(aes(color = cancelled)) +
  facet_wrap(~cancelled, scales = "free_y")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```


```
mutate(flights, cancelled = is.na(dep_time)) |>
  ggplot(aes(x = sched_dep_time)) +
  geom_freqpoly(aes(color = cancelled)) +
  facet_wrap(~cancelled, scales = "free")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

10.5.2.1 Exercises

1

How could you rescale the count dataset above to more clearly show the distribution of cut within color, or color within cut?

```
count_diamonds<-
  diamonds |>
  count(color, cut)

within_color<-
  count_diamonds |>
  group_by(color) |>
  summarise(`Total within color`= sum(n))

left_join(count_diamonds, within_color) |>
  mutate(`Proportion` = `n`/`Total within color`)
```

```
## Joining with `by = join_by(color)`
```

```
## # A tibble: 35 x 5
##    color cut           n `Total within color` Proportion
##    <ord> <ord>     <int>                <int>      <dbl>
##  1 D     Fair        163                 6775     0.0241
##  2 D     Good        662                 6775     0.0977
##  3 D     Very Good  1513                 6775     0.223
##  4 D     Premium    1603                 6775     0.237
```

```
##  5 D     Ideal      2834                6775     0.418
##  6 E     Fair        224                9797     0.0229
##  7 E     Good        933                9797     0.0952
##  8 E     Very Good  2400                9797     0.245
##  9 E     Premium    2337                9797     0.239
## 10 E     Ideal      3903                9797     0.398
## # i 25 more rows
```

```
within_cut<-
  count_diamonds |>
  group_by(cut) |>
  summarise(`Total within cut`= sum(n))

left_join(count_diamonds, within_cut) |>
  mutate(`Proportion` = `n`/`Total within cut`)
```

```
## Joining with `by = join_by(cut)`
```

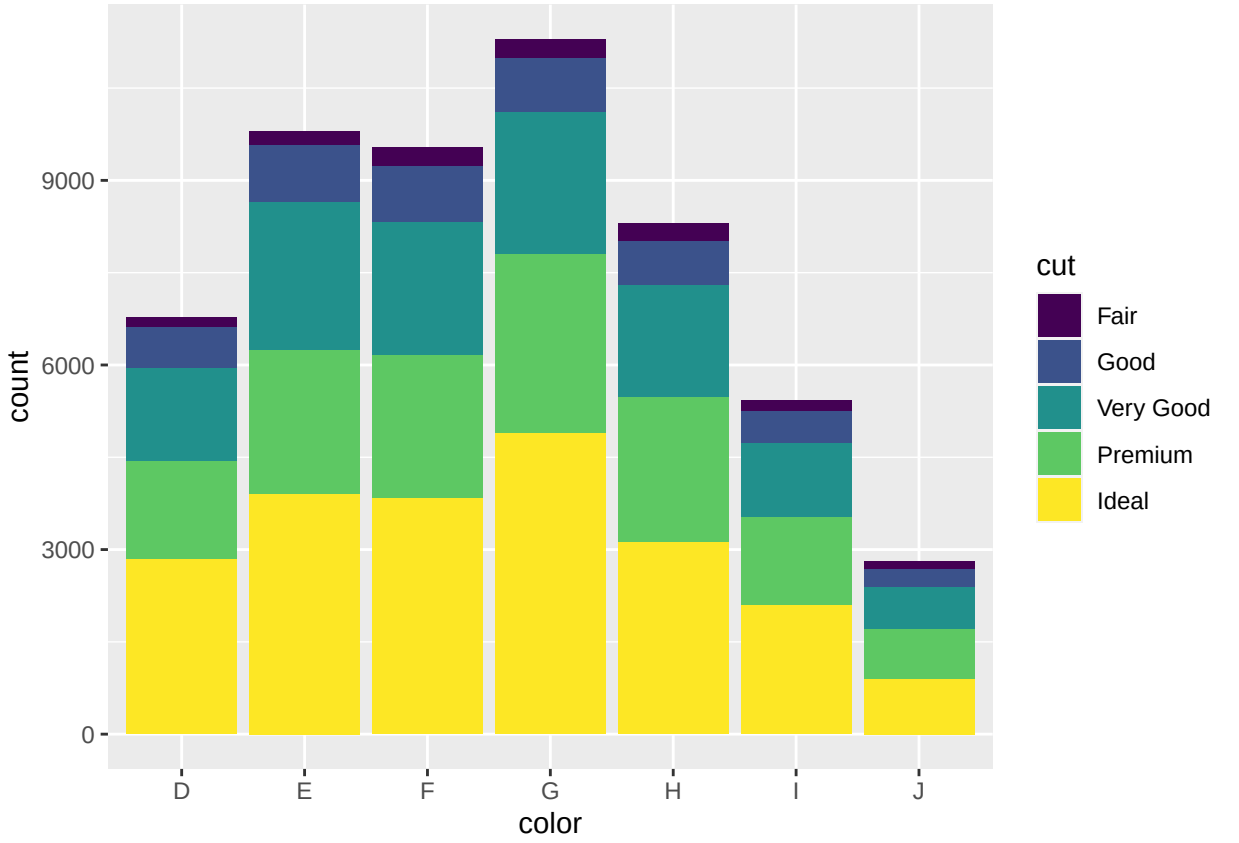
```
## # A tibble: 35 x 5
##    color cut           n `Total within cut` Proportion
##    <ord> <ord>     <int>              <int>      <dbl>
##  1 D     Fair        163               1610      0.101
##  2 D     Good        662               4906      0.135
##  3 D     Very Good  1513              12082      0.125
##  4 D     Premium    1603              13791      0.116
##  5 D     Ideal      2834              21551      0.132
##  6 E     Fair        224               1610      0.139
##  7 E     Good        933               4906      0.190
##  8 E     Very Good  2400              12082      0.199
##  9 E     Premium    2337              13791      0.169
## 10 E     Ideal      3903              21551      0.181
## # i 25 more rows
```

2

What different data insights do you get with a segmented bar chart if color is mapped to the x aesthetic and cut is mapped to the fill aesthetic? Calculate the counts that fall into each of the segments.

The insight we get from color mapped to the x aesthetic with cut being mapped to the fill aesthetic is that we can see the proportion of each cut within each color diamond. Counts for each segemnt are in the tibble below.

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




```
diamonds |>
  count(color, cut)
```

```
## # A tibble: 35 x 3
##    color cut           n
##    <ord> <ord>     <int>
##  1 D     Fair        163
##  2 D     Good        662
##  3 D     Very Good  1513
##  4 D     Premium    1603
##  5 D     Ideal      2834
##  6 E     Fair        224
##  7 E     Good        933
##  8 E     Very Good  2400
##  9 E     Premium    2337
## 10 E     Ideal      3903
## # i 25 more rows
```

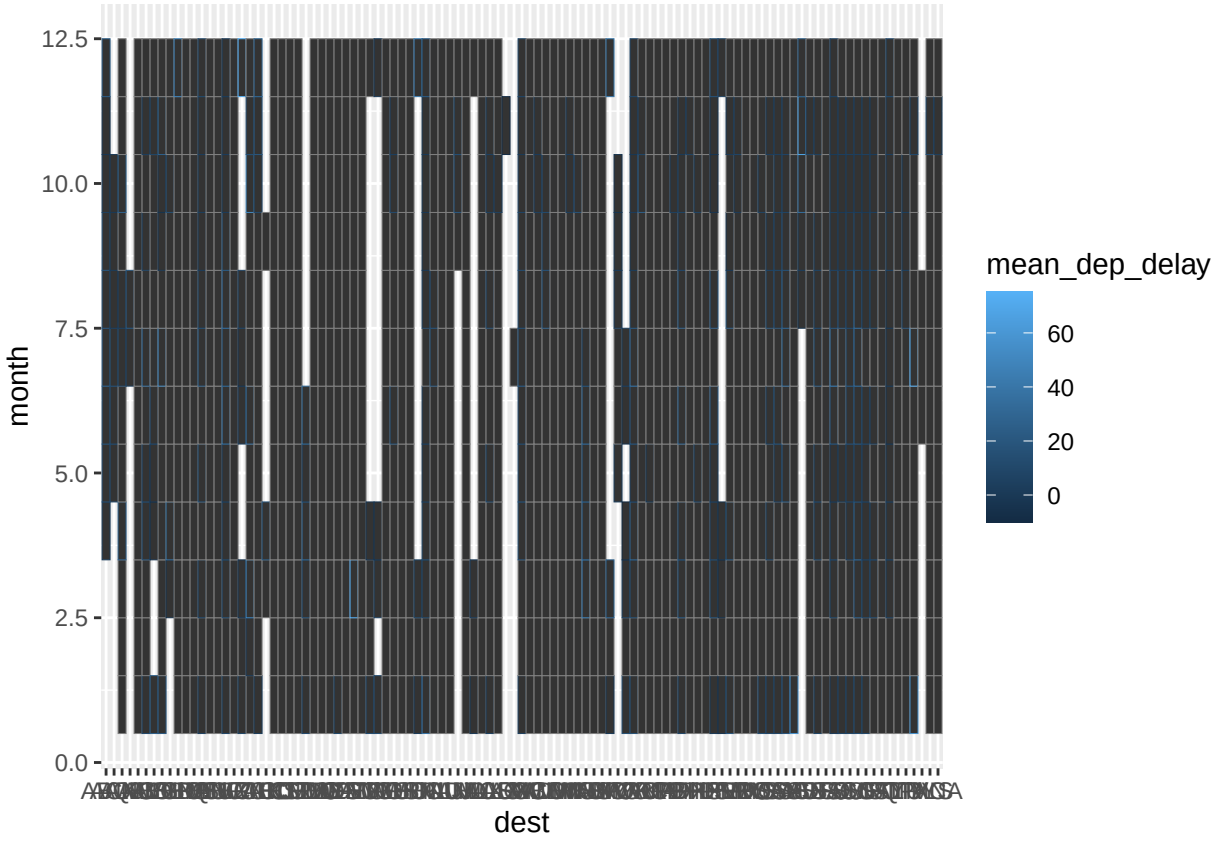
3

Use geom_tile() together with dplyr to explore how average flight departure delays vary by destination and month of year. What makes the plot difficult to read? How could you improve it?

This plot is difficult to read because there are too many destinations to read the X axis. It could be improved by creating a larger plot that has more space between each destination or multiple plots with a subsection of destinations.

```
flights |>
  group_by(dest, month) |>
  summarise(mean_dep_delay = mean(dep_delay)) |>
  ggplot(aes(dest, month)) +
  geom_tile(aes(color = mean_dep_delay))
```

```
## `summarise()` has grouped output by 'dest'. You can override using the
## `.groups` argument.
```



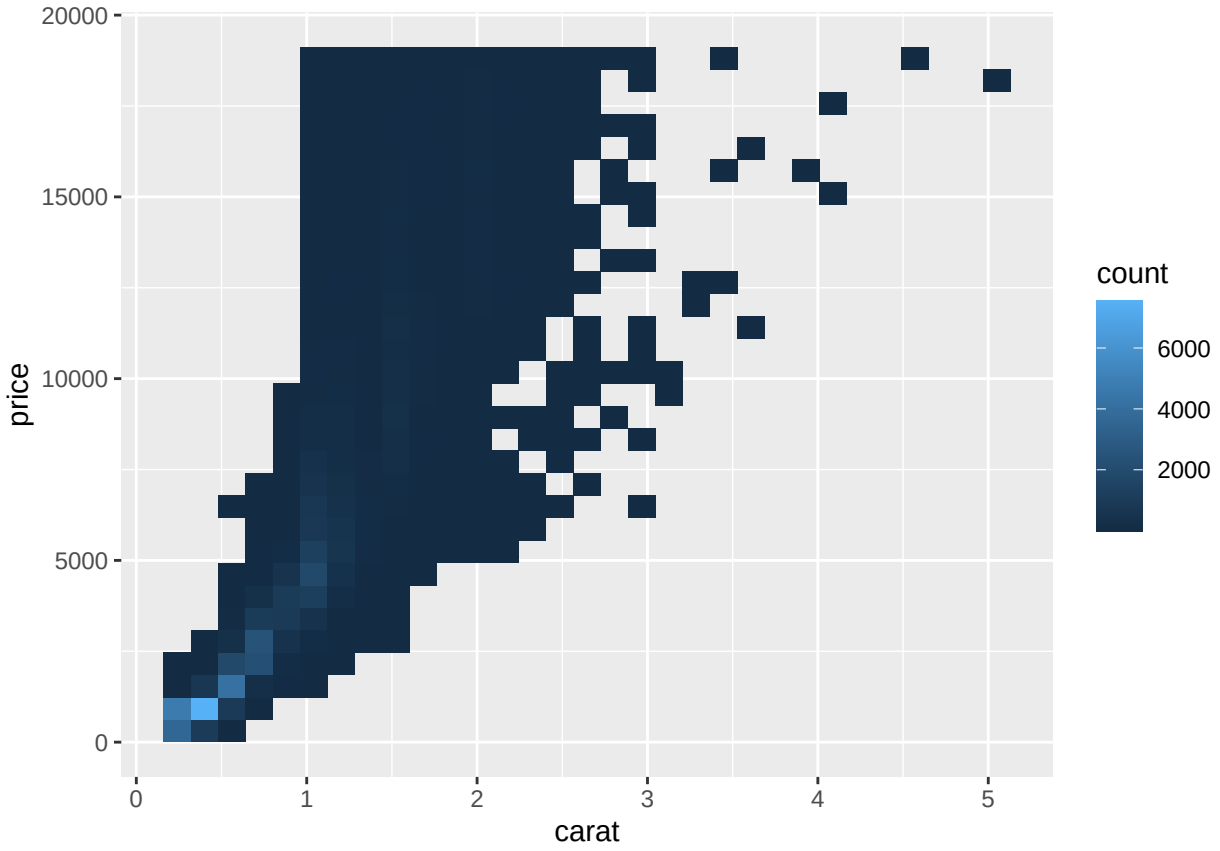
10.5.3.1 Exercises

1

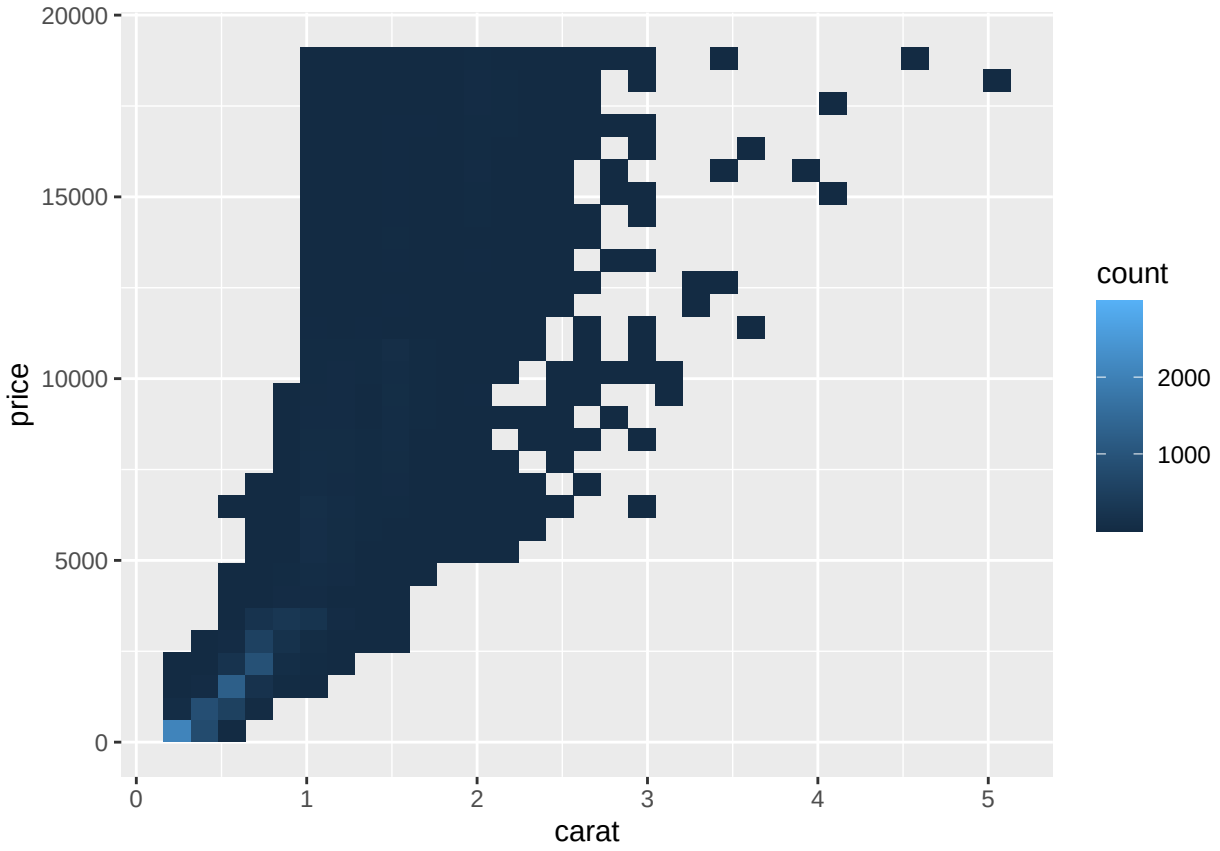
Instead of summarizing the conditional distribution with a boxplot, you could use a frequency polygon. What do you need to consider when using cut_width() vs. cut_number()? How does that impact a visualization of the 2d distribution of carat and price?

cut_width() divides a numerical variable into bins of the width you specify, while cut_number() makes n groups with (approximately) equal numbers of observations. Changing the cut_with() and/or cut_number() will change the resolution of the count gradient in a 2d distribution of variables.

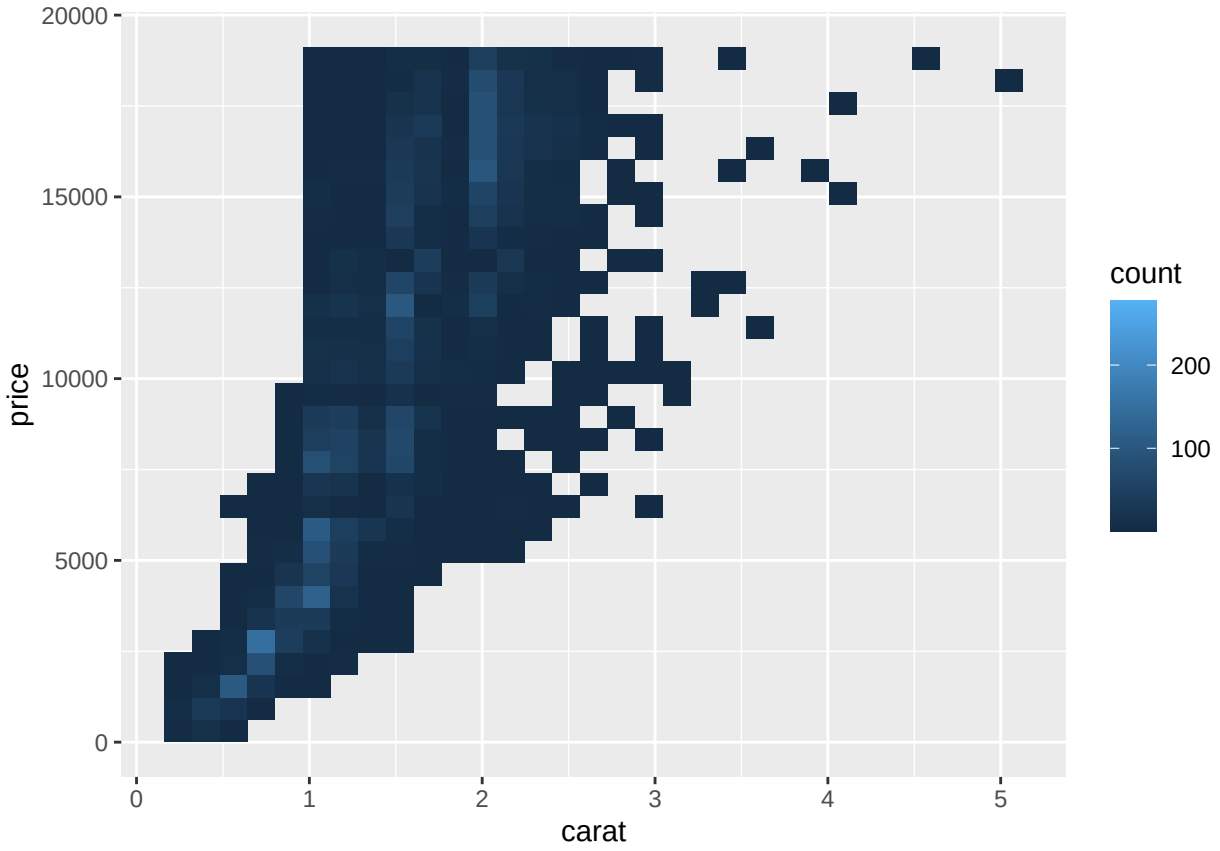
```
ggplot(diamonds, aes(carat, price))+
  geom_bin2d()
```




```
ggplot(diamonds, aes(carat, price))+
  geom_bin2d(aes(group = cut_width(`price`,200)))
```



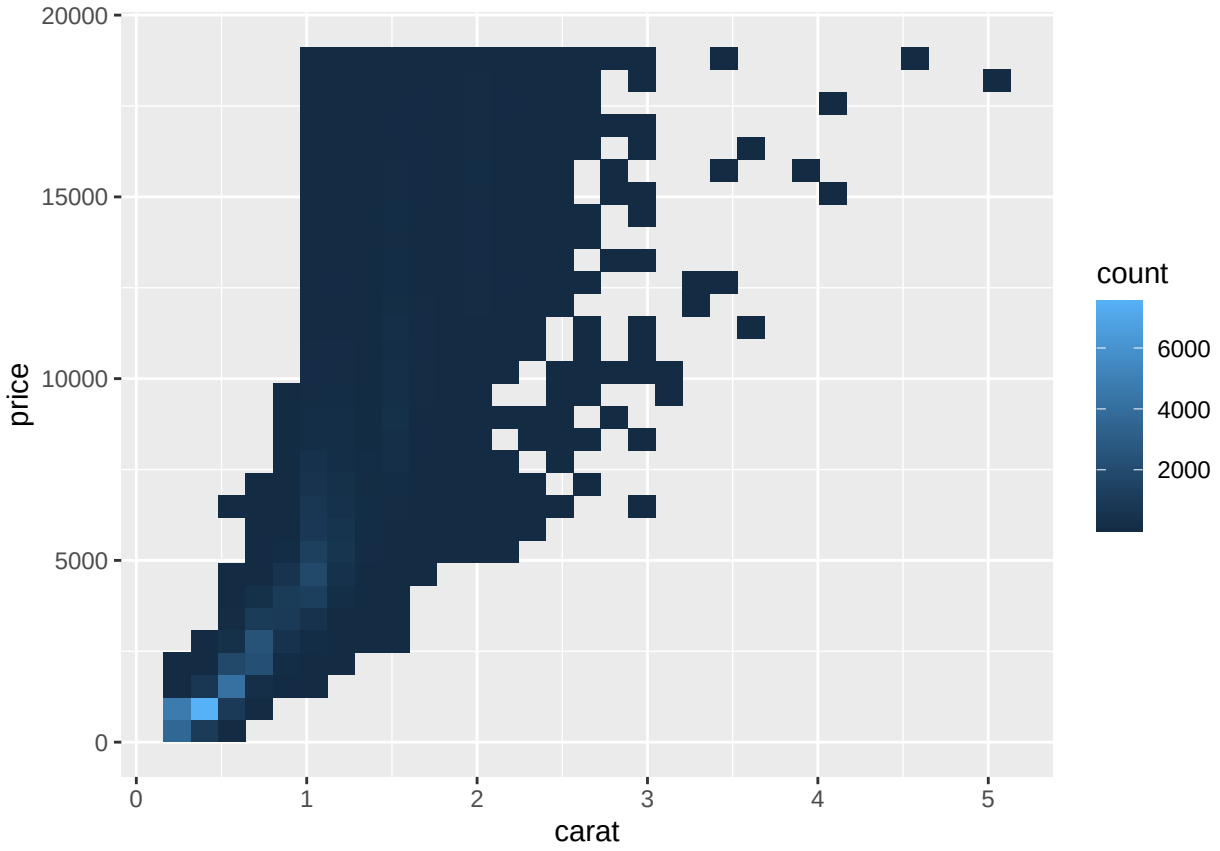

```
ggplot(diamonds, aes(carat, price))+
  geom_bin2d(aes(group = cut_number(`price`,200)))
```



2

Visualize the distribution of carat, partitioned by price.

```
ggplot(diamonds, aes(carat, price))+
  geom_bin2d()
```



3

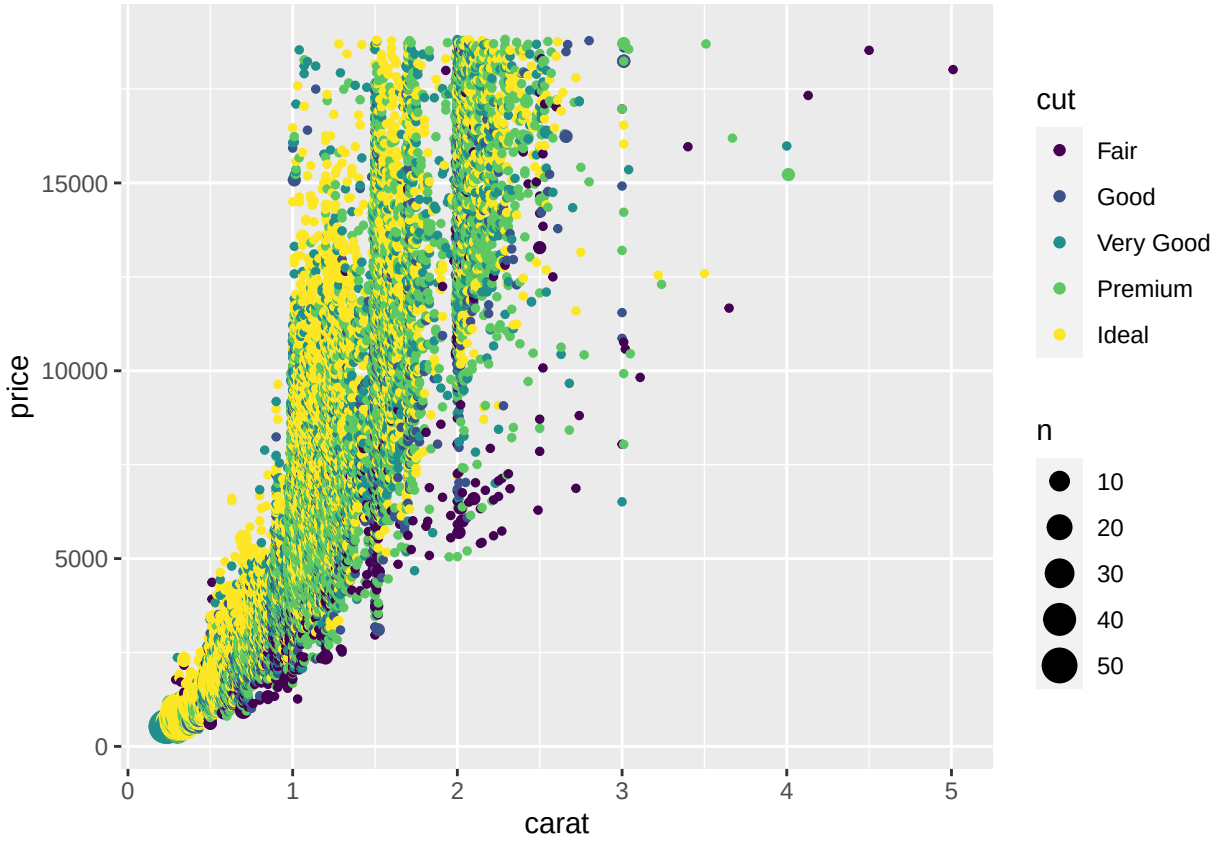
How does the price distribution of very large diamonds compare to small diamonds? Is it as you expect, or does it surprise you?

It is surprising that the range of prices for small diamonds is much larger than the range of prices for very large diamonds.

4

Combine two of the techniques you've learned to visualize the combined distribution of cut, carat, and price.

```
ggplot(diamonds, aes(carat, price, color = cut ))+
  geom_count()
```

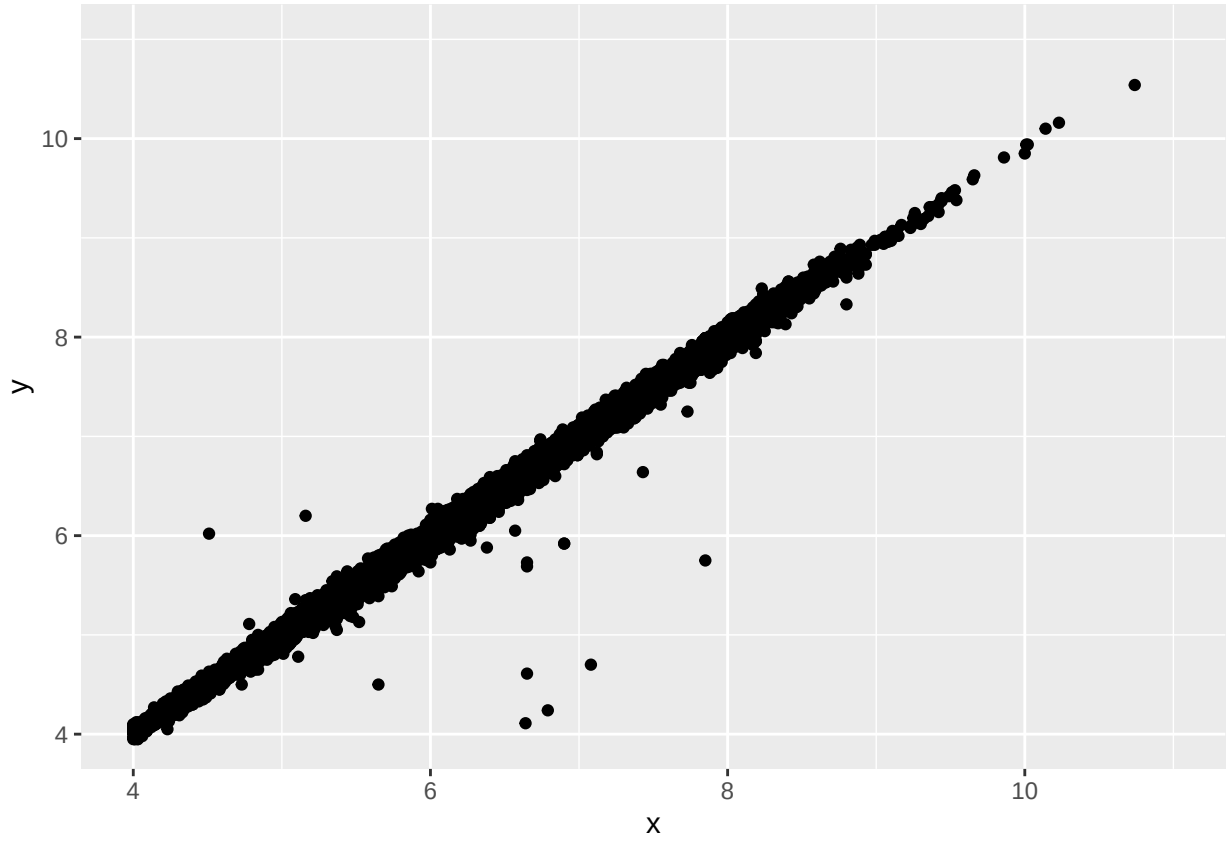


5

Two dimensional plots reveal outliers that are not visible in one dimensional plots. For example, some points in the following plot have an unusual combination of x and y values, which makes the points outliers even though their x and y values appear normal when examined separately. Why is a scatterplot a better display than a binned plot for this case?

Scatter plots allows you to see each individual observation and visually see outliers. Whereas if this was binned data those outliers would be thrown in a bin and not as easily observed.

```
diamonds |>
      filter(x >= 4) |>
      ggplot(aes(x = x, y = y)) +
      geom_point() +
      coord_cartesian(xlim = c(4, 11), ylim = c(4, 11))
```



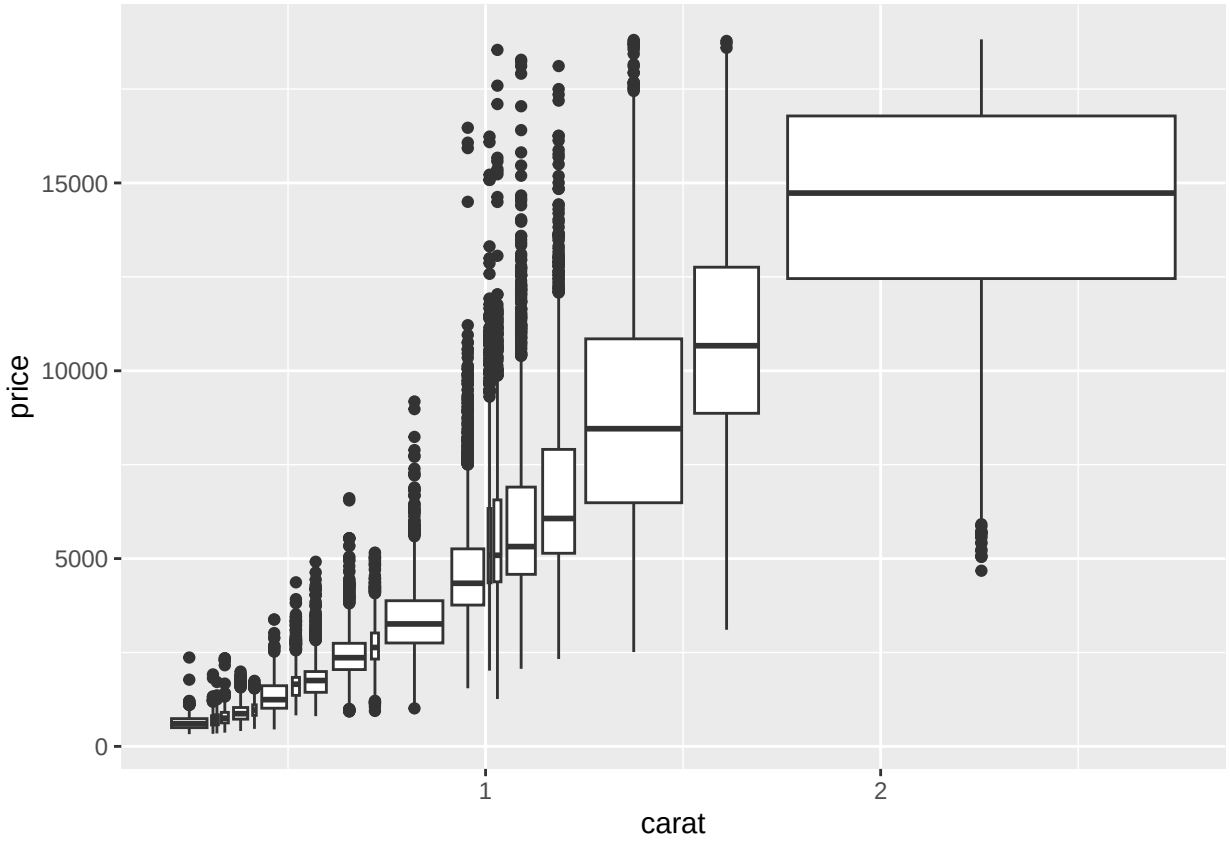
6

Instead of creating boxes of equal width with cut_width(), we could create boxes that contain roughly equal number of points with cut_number(). What are the advantages and disadvantages of this approach?

The advantage is that it is easier to see the distribution of diamonds across carats however it is difficult to see the the smaller carat boxes because there are so many of them.

```
smaller <- diamonds |>
  filter(carat < 3)

ggplot(smaller, aes(x = carat, y = price)) +
  geom_boxplot(aes(group = cut_number(carat, 20)))
```


price
15000
10000
5000
0
1
2
carat