STAT167 Lab #3 - Spring 2025

Ethan Choi

2025/4/18

Contents

Discussion/Lab #3 instructions

This week, we will review some `ggplot2` examples.

- First, download the `rmd` file from Canvas.
- Open this `rmd` file in RStudio and click `Knit -> Knit to PDF` to render it to PDF format. You need to have `LaTex` installed on the computer to render it to PDF format. If not, you can also render it to HTML format.

- Read this `rmd` file and the rendered `pdf`/`html` file side-by-side, to see how this document was generated!
- Be sure to play with this document! Change it. Break it. Fix it. The best way to learn R Markdown (or really almost anything) is to try, fail, then find out what you did wrong.
- Read over the `ggplot2` example code and check the output. If you have any questions about certain functions or parameters, it is the time to ask!
- There are some exercises through out this document. Replace **INSERT_YOUR_ANSWER** with your own answers. Knit the file, and check your results.

Please comment your R code thoroughly, and follow the R coding style guideline (https://google.github.io/styleguide/Rguide.xml). Partial credit will be deducted for insufficient commenting or poor coding styles.

Lab submission guideline

- After you completed all exercises, save your file to `FirstnameLastname-SID-lab3.rmd` and save the rendered pdf file to `FirstnameLastname-SID-lab3.pdf`. If you can not knit it to pdf, knit it to html first and then print/save it to pdf format.
- Submit **BOTH your source `rmd` file and the knitted `pdf` file** to **GradeScope**. Do NOT create a zip file.
- You can submit multiple times, you last submission will be graded.

Lecture Review - `ggplot2`

Load the `tidyverse` package

```
# install the packages first if you have not done it yet.
#install.packages("tidyverse")
#install.packages("gridExtra")

library(tidyverse) # for ggplot2
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.4
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error

library(gridExtra) # for grid.arrange()
##
## Attaching package: 'gridExtra'
##
## The following object is masked from 'package:dplyr':
##
##     combine
```

The `mpg` data set

This data set contains fuel economy data 1999 - 2008 for 38 popular car models.

```
?mpg
## starting httpd help server ... done
glimpse(mpg) # get a glimpse of the mpg data
## Rows: 234
## Columns: 11
## $ manufacturer <chr> "audi", "audi", "audi", "audi", "audi", "audi", "audi", "~
## $ model        <chr> "a4", "a4", "a4", "a4", "a4", "a4", "a4", "a4 quattro", "~
## $ displ        <dbl> 1.8, 1.8, 2.0, 2.0, 2.8, 2.8, 3.1, 1.8, 1.8, 2.0, 2.0, 2.~
## $ year         <int> 1999, 1999, 2008, 2008, 1999, 1999, 2008, 1999, 1999, 200~
## $ cyl          <int> 4, 4, 4, 4, 6, 6, 6, 4, 4, 4, 4, 6, 6, 6, 6, 6, 6, 8, 8, ~
## $ trans        <chr> "auto(l5)", "manual(m5)", "manual(m6)", "auto(av)", "auto~
## $ drv          <chr> "f", "f", "f", "f", "f", "f", "f", "4", "4", "4", "4", "4~
## $ cty          <int> 18, 21, 20, 21, 16, 18, 18, 18, 16, 20, 19, 15, 17, 17, 1~
## $ hwy          <int> 29, 29, 31, 30, 26, 26, 27, 26, 25, 28, 27, 25, 25, 25, 2~
## $ fl           <chr> "p", "p", "p", "p", "p", "p", "p", "p", "p", "p", "p", "p~
## $ class        <chr> "compact", "compact", "compact", "compact", "compact", "c~
```

The `diamonds` data set

This data set contains the prices and other attributes of almost 54,000 diamonds.

```
?diamonds
glimpse(diamonds) # get a glimpse of the data
## Rows: 53,940
## Columns: 10
## $ carat   <dbl> 0.23, 0.21, 0.23, 0.29, 0.31, 0.24, 0.24, 0.26, 0.22, 0.23, 0.~
## $ cut     <ord> Ideal, Premium, Good, Premium, Good, Very Good, Very Good, Ver~
## $ color   <ord> E, E, E, I, J, J, I, H, E, H, J, J, F, J, E, E, I, J, J, J, I,~
## $ clarity <ord> SI2, SI1, VS1, VS2, SI2, VVS2, VVS1, SI1, VS2, VS1, SI1, VS1, ~
## $ depth   <dbl> 61.5, 59.8, 56.9, 62.4, 63.3, 62.8, 62.3, 61.9, 65.1, 59.4, 64~
## $ table   <dbl> 55, 61, 65, 58, 58, 57, 57, 55, 61, 61, 55, 56, 61, 54, 62, 58~
## $ price   <int> 326, 326, 327, 334, 335, 336, 336, 337, 337, 338, 339, 340, 34~
## $ x       <dbl> 3.95, 3.89, 4.05, 4.20, 4.34, 3.94, 3.95, 4.07, 3.87, 4.00, 4.~
## $ y       <dbl> 3.98, 3.84, 4.07, 4.23, 4.35, 3.96, 3.98, 4.11, 3.78, 4.05, 4.~
## $ z       <dbl> 2.43, 2.31, 2.31, 2.63, 2.75, 2.48, 2.47, 2.53, 2.49, 2.39, 2.~
```

Exercise #1

Which variables in `diamonds` are categorical? Which variables are numerical?

INSERT_YOUR_ANSWER Cut, Color, and Clarity are categorical. Carat, Depth, Table, Price, x, y, and z are numerical

The complete graphing template in `ggplot2`

```
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(
     mapping = aes(<MAPPINGS>),
     stat = <STAT>,         # optional
     position = <POSITION> # optional
  ) +
  <COORDINATE_FUNCTION> +  # optional
  <FACET_FUNCTION> +       # optional
  <SCALE_FUNCTION> +       # optional
  <THEME_FUNCTION>         # optional
```

geom vs stat functions

Statistical transformation is a alternative way to build a layer.

Most geoms and stats come in pairs that are almost always used in concert.

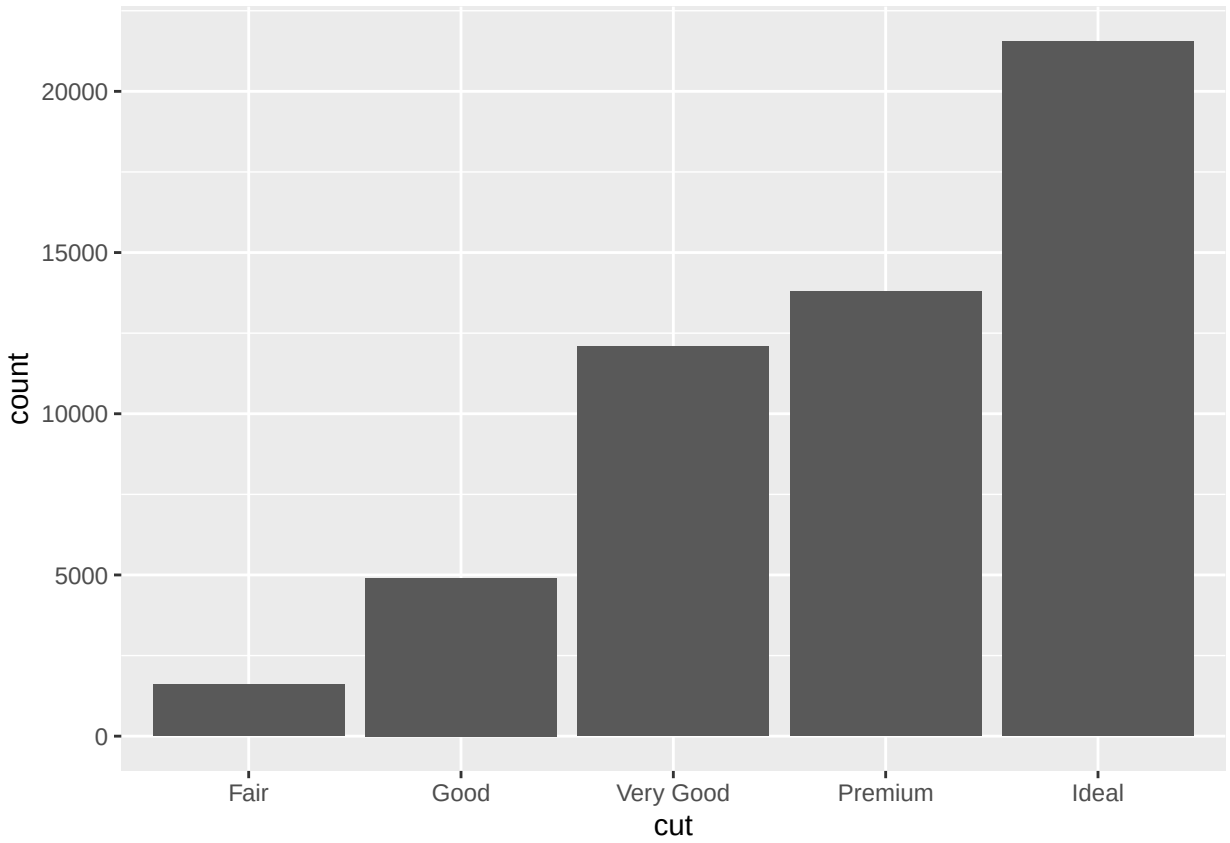
- every geom has a default stat
- every stat has a default geom.

You can typically use geoms without worrying about the underlying statistical transformation.

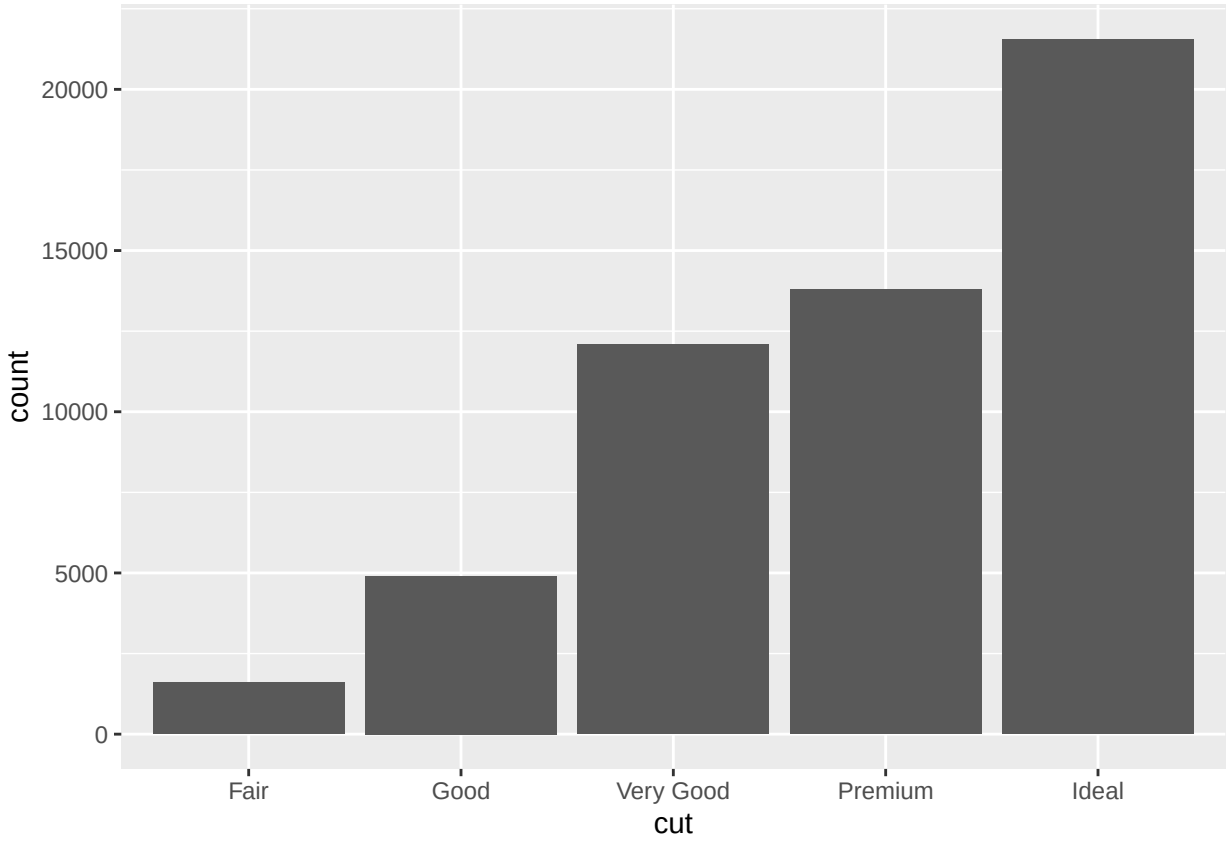
You can also use geoms and stats interchangeably.

Example: `geom_bar()` calls `stat_count()` make bar plot for a categorical variable

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



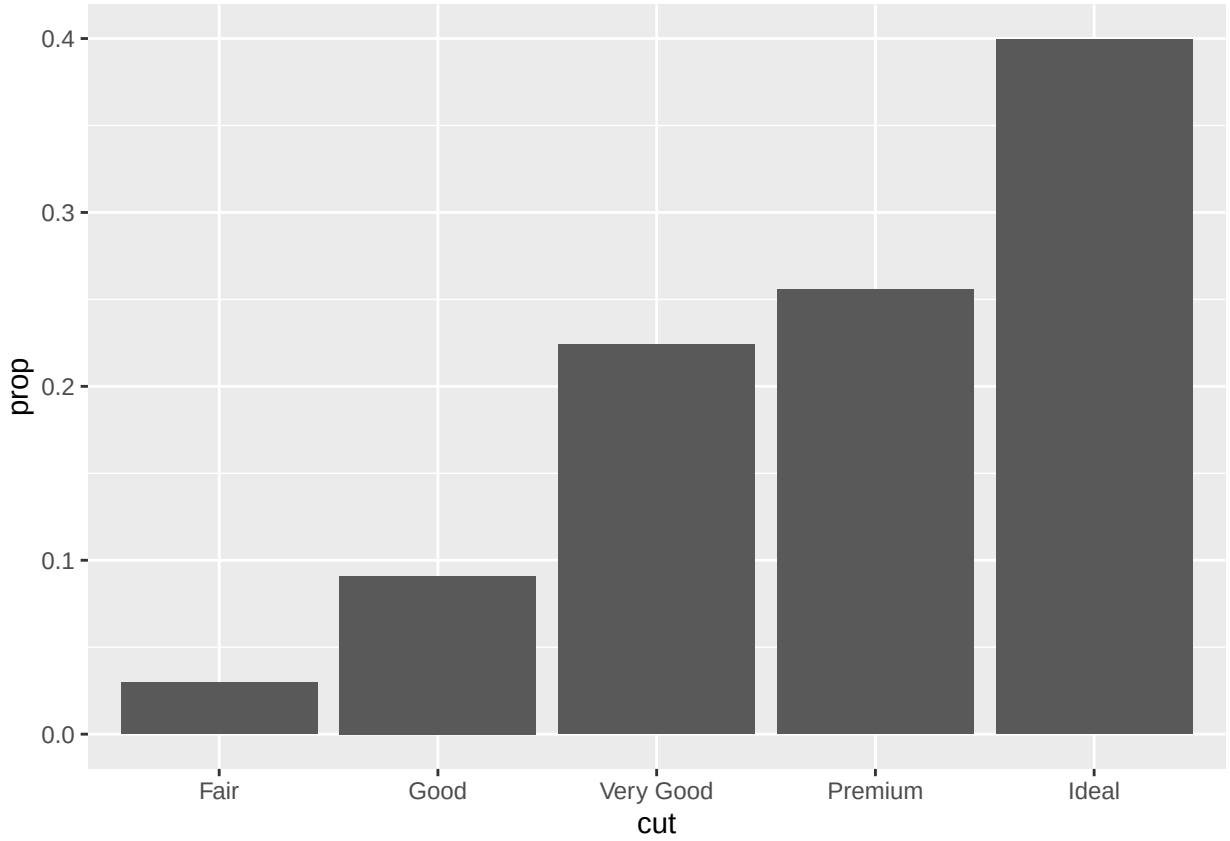

```
ggplot(data = diamonds) +
  stat_count(mapping = aes(x = cut))
```



Override default stat function

To make a proportion bar chart (relative frequency histogram), we need to override the default `group` argument and map the new variable `prop` (computed by `stat_count()`) to the `y` aesthetic.

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



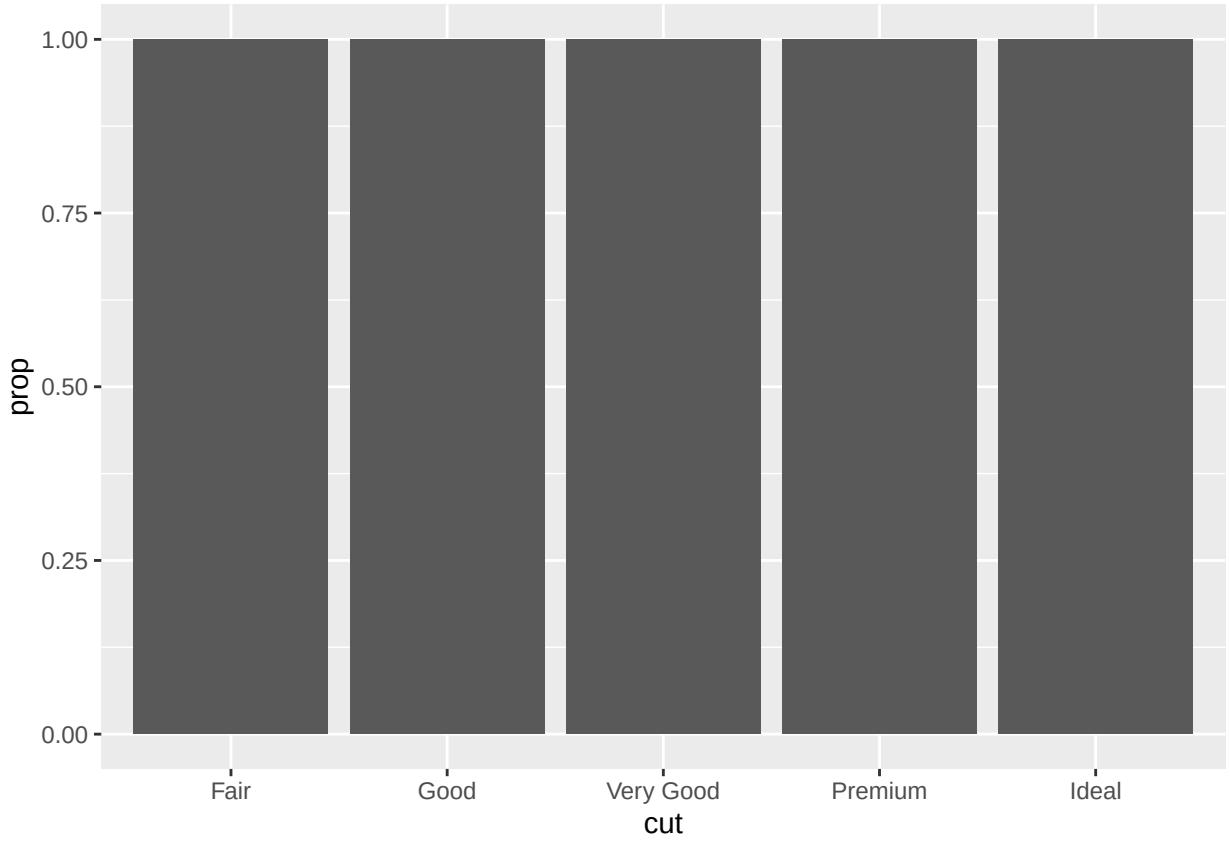
Exercise #2

(a) Why do we need to set `group = 1`?

INSERT_YOUR_ANSWER So we can process the dataset as a whole, rather than splitting it into different groups. If I were to change it, the proportions would be off.

(b) If we remove `group = 1`, what is the problem with the output graph?

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



INSERT_YOUR_ANSWER Our bar heights all become the same, and again, the proportions are incorrectly calculated

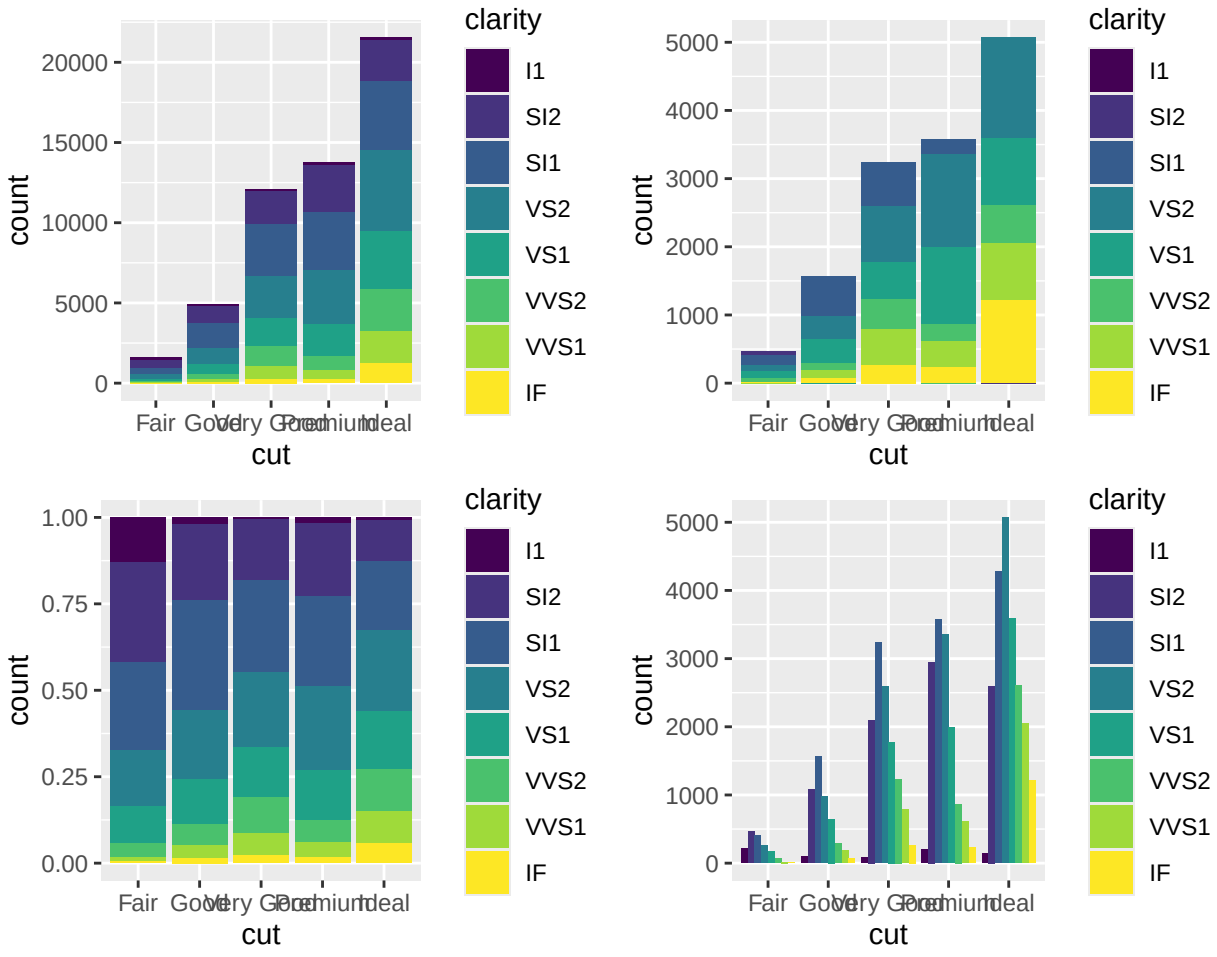
Example: position adjustment options for `geom_bar()`

```
?geom_bar
```

The position argument specifies the position adjustment of bars, rectangles.

- **default**: position = "stack"
- position = "identity" will place each object exactly where it falls in the context of the graph.
- position = "fill" works like stacking, but makes each set of stacked bars the same height.
- position = "dodge" places overlapping objects directly beside one another. the bars are automatically stacked. Each colored rectangle represents a combination of cut and clarity.

```
gg <- ggplot(data = diamonds,
             mapping = aes(x = cut, fill = clarity))
plot1 <- gg + geom_bar(position="stack")
plot2 <- gg + geom_bar(position="identity")
plot3 <- gg + geom_bar(position="fill")
plot4 <- gg + geom_bar(position="dodge")
grid.arrange(plot1, plot2, plot3, plot4, ncol=2)
```



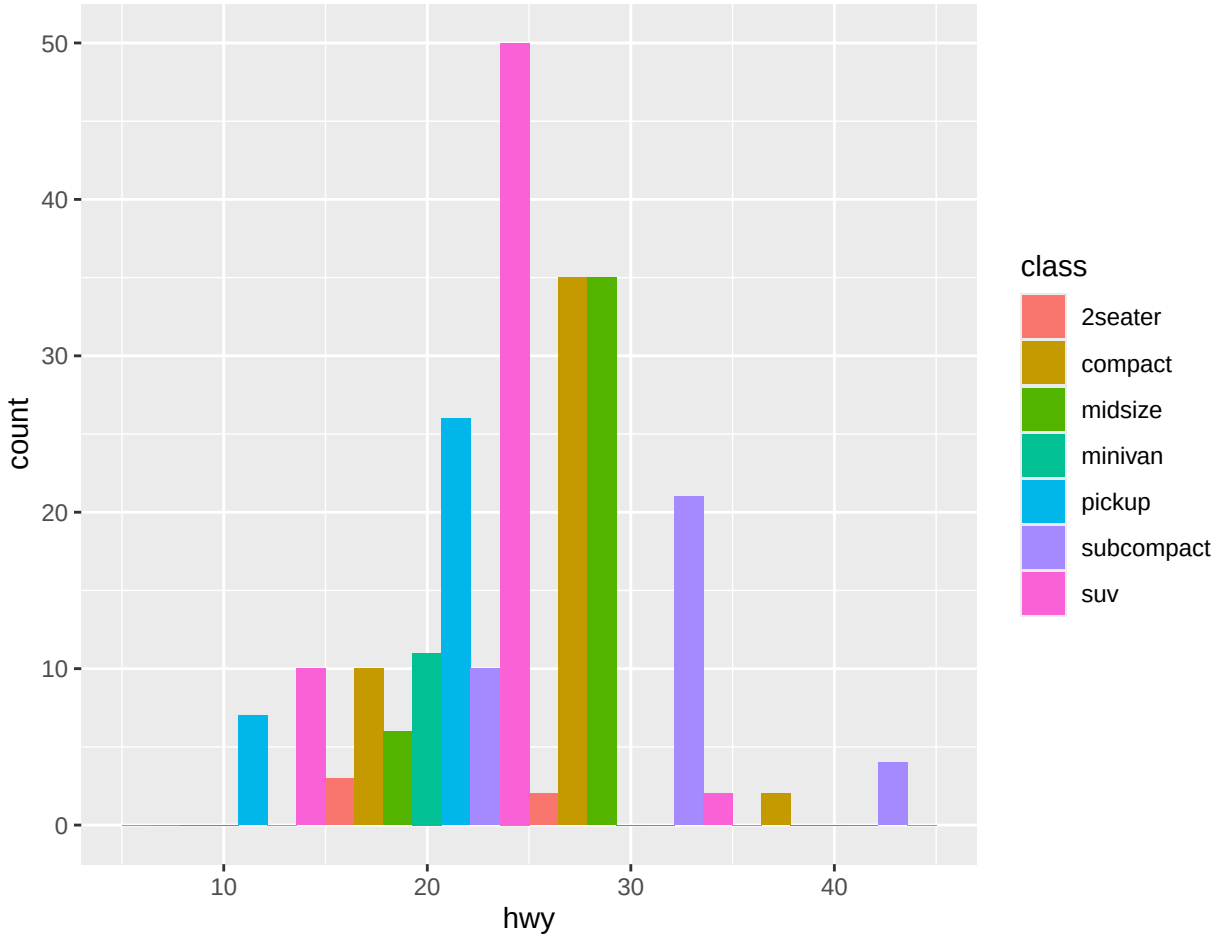
Exercise #3

Compare the "stack" position adjustment and the "identical" position adjustment. What is the difference? Why the y-axis scales are different?

INSERT_YOUR_ANSWER Stack position adjustment displays the graph so that the bars of clarity level are, well, stacked on top of one another. The identical position adjustment shows each cut and clarity combo individually rather than stacked. The y axis scales are different because stack gathers the values in a clumped fashion which results in extended totals, whereas identical doesn't and has lesser totals.

Example: `geom_histogram()` to compare highway mileage across car classes

```
ggplot(data = mpg) +
  geom_histogram(mapping = aes(x = hwy, fill = class),
                 binwidth = 10,
                 position = "dodge")
```



Exercise #4

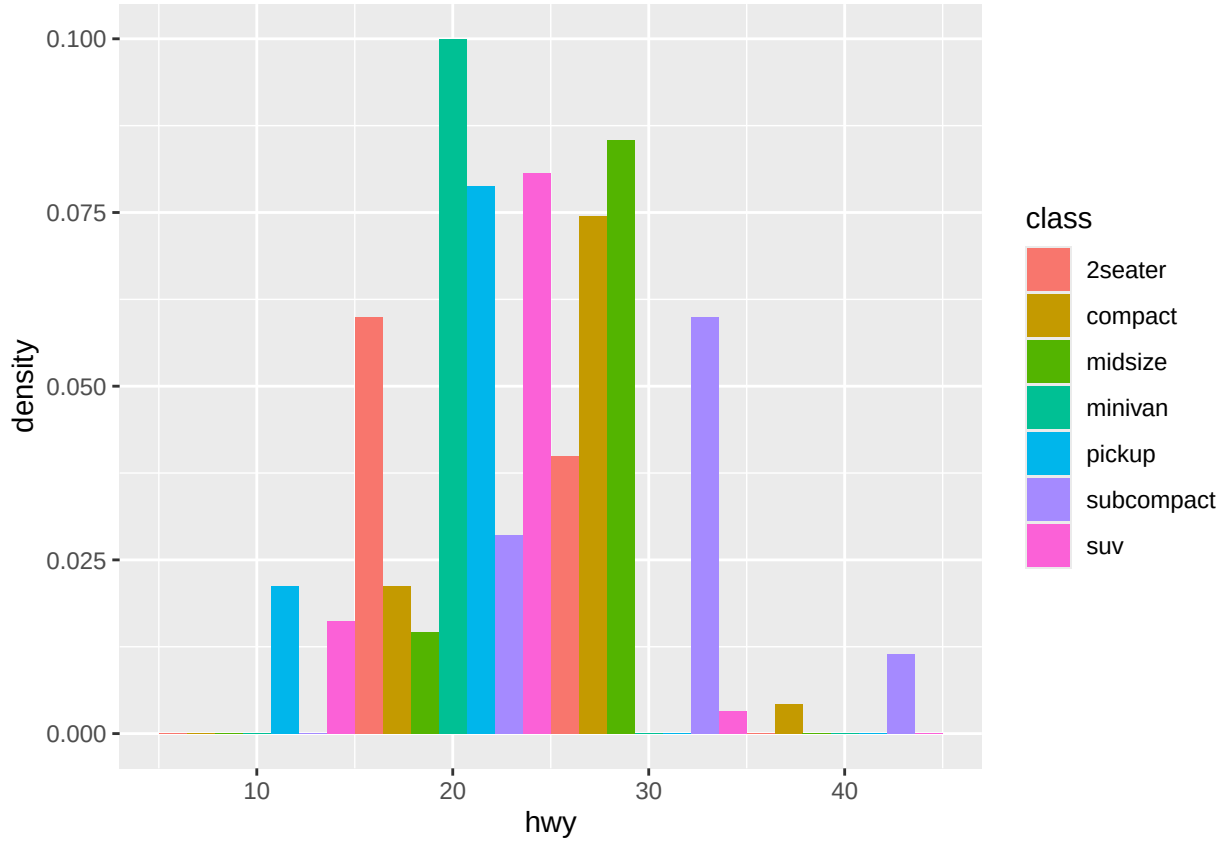
(a) Why there are gaps in the histogram?

INSERT_YOUR_ANSWER Position = dodge was used, so values, or the bars, in the histogram are next to each other intsead of on top.

(b) Can you change this frequency histogram to a density histogram?

INSERT_YOUR_ANSWER Yes? I'd just add a y =

```
ggplot(data = mpg) +
  geom_histogram(mapping = aes(x = hwy, fill = class, y = after_stat(density)), binwidth = 10, position
```



Example: `geom_freqpoly()` to compare highway mileage across car classes

```
ggplot(data = mpg) +
  geom_freqpoly(mapping = aes(x = hwy, color = class),
                binwidth = 10)
```