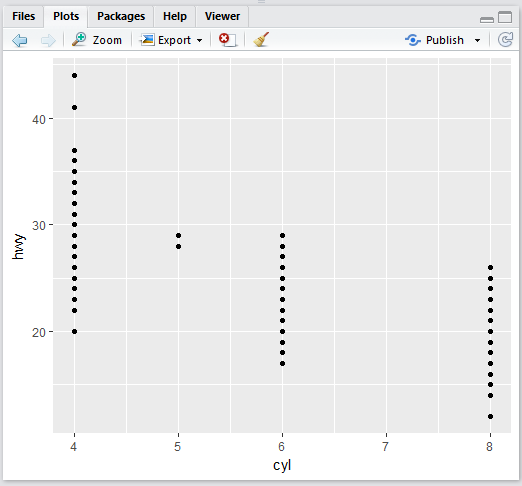
3.2.4

1. Nothing

2. nrow(mpg) = 234. Ncol(mpg) = 11

3. f = front-wheel drive, r = rear wheel drive, 4 = 4wd

4.



5. This plot isn’t useful because class and drv are categorical variables.

3.3.1

1. `color` is inside of aes(), it needs to be outside.

2. Categorical variables: manufacturer, model, trans, dvr, fl, and class. Continuous variables: displ, year, cyl, cty, and hwy. You can see this when running `mpg` by the variable types within <>.

3. Continuous variables will have a color scale and categorical variables will have unique colors for each level.

4. Each level will have a unique color and shape, further helping differentiate them on the plot.

5. Stroke controls the boarder width. It’s additive with the size aesthetic.

6. Creates two levels for color coding, TRUE or FALSE.

3.5.1

1. You get a grid for each value that exists in your data set for the continuous variable.

2. There are no data points within that combination of the two variables that you are faceting by. Both plots described will explain which combination of those two variables don’t have any data points. The second plot is a much simpler way to get to that answer.

3. The `.` serves as a placeholder and allows you to determine if you want your plots to be lined up vertically or horizontally.

4. Advantages to using faceting allows you to view the data points of each facet separately – easier to identify trends. Disadvantage is you now need many plots to see the same data. As the data set becomes larger, you will have more facets, making it much more unwieldy to use faceting.

5. `nrow` and `ncol` determine how many rows and colums you want your facets to be displayed in. Other options that control the layout of individual panels include scales and shrink. `facet\_grid()` doesn’t have `nrow` and `ncol` because each facet is a combination of two different variables – that will predetermine the number of rows and columns you will have.

6. Most displays are wider than they are tall, so it fits cleaner to have more columns than rows.

3.6.1

1. Draw a line chart with geom\_line or \_smooth. Boxplot with \_boxplot. Historgram with \_histogram. Area chart with \_area.

2. Expect it to layer a line graph over a scatter plot.

3. show.legend = FALSE will prevent the legend from being shown.

4. se determines whether the confidence interval is shown around the line.

5. They display the same thing because you’re just moving where you call what dataset you’re working with.

6.

a. ggplot(data=mpg) + geom\_point(mapping = aes(x=displ, y=hwy)) + geom\_smooth(mapping=aes(x=displ,y=hwy),se=FALSE)

b. ggplot(data=mpg) + geom\_point(mapping = aes(x=displ, y=hwy)) + geom\_smooth(mapping=aes(x=displ,y=hwy,group=drv),se=FALSE)

c. ggplot(data=mpg) + geom\_point(mapping = aes(x=displ, y=hwy, color=drv)) + geom\_smooth(mapping=aes(x=displ,y=hwy,color=drv),se=FALSE)

d. ggplot(data=mpg) + geom\_point(mapping = aes(x=displ, y=hwy, color=drv)) + geom\_smooth(mapping=aes(x=displ,y=hwy),se=FALSE)

e. ggplot(data=mpg) + geom\_point(mapping = aes(x=displ, y=hwy, color=drv)) + geom\_smooth(mapping=aes(x=displ,y=hwy,linetype=drv),se=FALSE)

f. ggplot(data=mpg) + geom\_point(mapping = aes(x=displ, y=hwy, color=drv))

3.7.1

1. Default geom for stat\_summary() is point range. ggplot(data=diamonds) + geom\_pointrange(mapping = aes(x=cut,y=depth), fun.ymin = min, fun.ymax=max, fun.y=median,stat="summary")

2. geom\_col represents values, where geom\_bar represents counts

3. geom\_bar and geom\_col = stat\_count. Geom\_bin2d = stat\_bin\_2d. geom\_boxplot = stat\_boxplot. Geom\_contour = stat\_contour. Geom\_count = stat\_sum. Geom\_density\_2d = stat\_density\_2d. Geom\_density = stat\_density. Geom\_hex = stat\_bin\_hex. Geom\_freqpoly and geom\_histogram = stat\_bin. Geom\_qq = stat\_qq. Geom\_quantile = stat\_quantile. Geom\_smooth = stat\_smooth. Geom\_violin = stat\_ydensity. Geom\_sf = stat\_sf

4. Geom\_smooth calculates predicted values, confidence interval, and standard error.

5. The proportions are created off their individual groups instead of in reference to the entire set of data. Everything will have a proportion of 100% without it.

3.8.1

1. Points overlap. Jitter them.

2. Width and height control the amount of jittering.

3. Geom\_count counts the number of obstervations and makes those points larger. Jitter will help you see more points on the screen. Count will make the overlapped points stand out.

4. Default position adjustment for geom\_boxplot is dodge. ggplot(data=mpg) + geom\_boxplot(mapping = aes(x=class, y=hwy, color=drv))

3.9.1

1. ggplot(data = diamonds) + geom\_bar(mapping = aes(x = cut, fill = clarity), position = "fill") + coord\_polar()

2. Labs adds labels and a caption to your graph.

3. Quickmap preserves straight lines and therefore is faster to compute, but less accurate.

4. Plot tells you there is a positive, linear correlation between hwy and city mpg. Coord\_fixed() keeps the scale proportional. Geom\_abline draws a line that has an intercept of 0 and a slope of 1 to use as a reference point.

4.4

1. my\_variable was spelled wrong

2. data is spelled dOta. Filter is spelled fLilter.

3. It pulls up a list of shortcuts. Also can go to help -> keyboard shortcuts help

5.2.4

* 1. Filter(flights, arr\_delay>=120)
  2. Filter(flights, dest ==”IAH” | dest=”HOU”)
  3. Filter(flights, carrier==”UA” | carrier==”AA” | carrier==”DL”)
  4. Filter(flights, month==7 | month==8 | month==9)
  5. Filter(flights, arr\_delay>=120 & dep\_delay == 0)
  6. Filter(flights, dep\_time>=0 & dep\_time<= 6\*60)

1. It’s a shortcut for an inclusive filtering on one variable. It could simplify the last item from the previous question.
2. 8255 rows with an NA dep\_time. They are also missing delays and arrival times. They could represent cancelled flights.
3. Anything raised to the 0th power is always 1. Since NA can be any value, will you always get the same result regardless of what value NA is? If yes, the response is not missing. If no, the value is missing.

5.3.1

1. arrange(df,desc(is.na(variable)))

2. arrange(flights,desc(dep\_delay)). Arrange(flights,dep\_time)

3. arrange(flights,air\_time)

4. arrange(flights, desc(distance)). Arrange(flights, distance)

5.4.1

1. Write out each column individually. Starts with dep and starts with arr.

2. Duplicate columns are automatically removed from select()

3. One\_of allows you to use a vector with the column names created outside of the select statement.

4. Doesn’t surprise me. Case doesn’t matter.

5.5.2

1. transmute(flights, sched\_dep\_time = (sched\_dep\_time %/% 100) \* 60 + sched\_dep\_time %% 100, dep\_time = (dep\_time %/% 100) \* 60 + dep\_time %% 100)

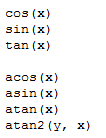
2. They shouldn’t match up because arr\_time and dep\_time aren’t continuous. Fix it by creating a new column for them that are continuous (like in the question above), then subtract them.

3. Difference between dep\_time and sched\_dep\_time should equal dep\_delay after converting the first two into continuous variables.

4. arrange(mutate(flights,rank = min\_rank(dep\_delay)),desc(rank)). I’d rather have the earliest entry win so there aren’t any duplicate ranks. So I can use arrange(mutate(flights,rank = rank(dep\_delay,ties.method="first")),desc(rank))

5. The vectors aren’t the same length, so R will recycle the first vector until it is the same length. Then math is done.

6. These



5.6.7

1.

a. group\_by(flights,flight) %>% summarize(early\_15 = sum(arr\_delay <= - 15, na.rm = TRUE) / n(),late\_15 = sum(arr\_delay >= 15,na.rm = TRUE) / n()) %>% filter(early\_15 == .5, late\_15 == .5)

b. group\_by(flights,flight) %>% summarize(late\_10 = sum(arr\_delay == 10, na.rm = TRUE) / n()) %>% filter(late\_10 == 1)

c. group\_by(flights,flight) %>% summarize(early\_30 = sum(arr\_delay <= - 30, na.rm = TRUE) / n(),late\_30 = sum(arr\_delay >=30,na.rm = TRUE) / n()) %>% filter(early\_30 == .5, late\_30 == .5)

d. group\_by(flights,flight) %>% summarize(on\_time = sum(arr\_delay == 0, na.rm = TRUE) / n(),late\_2hour = sum(arr\_delay >= 120, na.rm = TRUE) / n()) %>% filter(on\_time == 99, late\_2hour == .01)

e. You fly with the intent to reach a destination, not to leave your current location. Therefore, arrival delay is more important.

2.

a. group\_by(not\_cancelled, dest) %>% summarize(the\_count = n()) %>% select(dest,the\_count)

b. group\_by(not\_cancelled, tailnum) %>% summarize(n=sum(distance, na.rm = TRUE)) %>% select(tailnum,n)

3. If dep\_delay is NA and arr\_delay has a value, it means a flight arrived. It’s possible instead of a 0, we had an NA for dep\_delay. Arr\_delay is more important.

4. group\_by(flights,day) %>% summarize(prop\_flights = sum(is.na(dep\_delay) | is.na(arr\_delay), na.rm = TRUE)/n(),avg\_delay = mean(dep\_delay, na.rm = TRUE)) %>% ggplot() + geom\_point(mapping=aes(x=avg\_delay,y=prop\_flights))

5.