Homework-2: *Your Name Here*

All of these homework problems are from the *R for Data Science* book. The section numbers (e.g., “3.2.4 Exercises”) refer to sections in this book. Although the questions are based on those in the book, some questions ask for additional details or analysis.

When solving these problems, you are allowed to use any method from the book or class, even if that method wasn’t yet covered when the exercise was presented in the book.

Write answers that are as complete as possible. If a graph is helpful for formalizing the solution, provide the graph. If a table is helpful, provide a table. In the text part of the answer, outline the progression in your thinking as you perform the analysis.

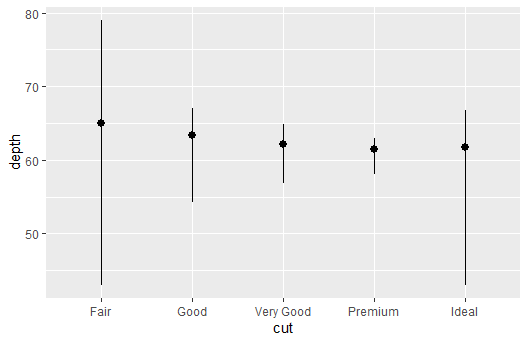
Note that you should type your answers in RStudiop, by typing into the file **Homework-2.sa.Rmd**.

## 3.7.1 Exercises

### (1) 3.7.1 Exercise 1, (10 pts)

Consider the following plot:

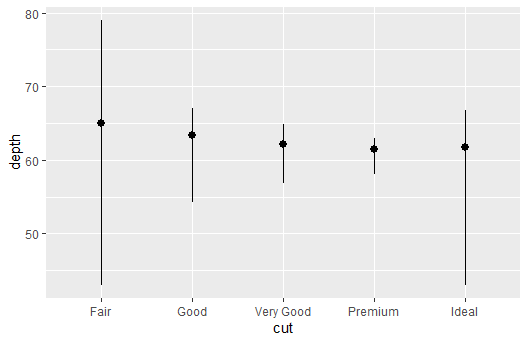
ggplot(data = diamonds) +  
 stat\_summary(  
 mapping = aes(x = cut, y = depth),  
 fun.min = min,  
 fun.max = max,  
 fun = median  
 )



What is the default geom associated with stat\_summary()? # The geometry associated would be a line. More specifically a hybrid of a frequency and bar plot.

How could you rewrite the plot code so that it drew the same graph, but used the default geom instead of stat\_summary()? # geom\_pointrange() = stat\_summary().

ggplot(data = diamonds) +  
 geom\_pointrange(  
 mapping = aes(x = cut, y = depth),  
 stat = "summary",  
 fun.min = min,  
 fun.max = max,  
 fun = median)

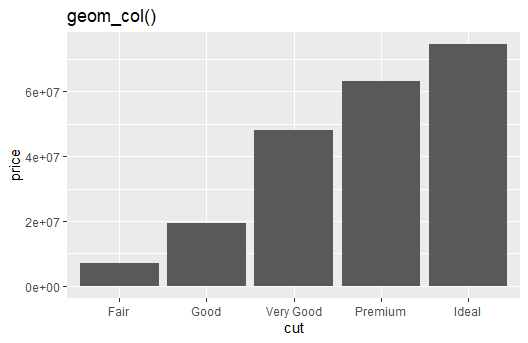


(Note: Ed made use of summarize(), but other solutions may also exist.)

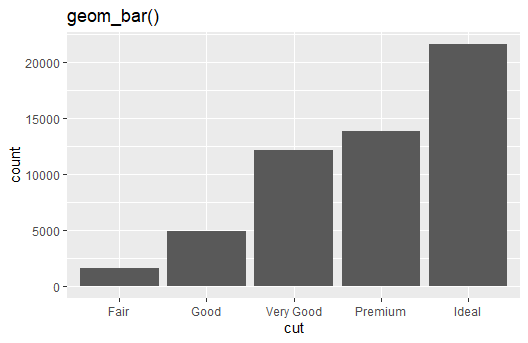
### (2) 3.7.1 Exercise 2, (5 pts)

What does geom\_col() do, and how is it different from geom\_bar()? # The main difference between the functions is that geom\_col() maps a categorical variable to a continuous one while geom\_bar() just counts the number of a specific variable.

ggplot(data = diamonds,mapping = aes(x = cut, y = price)) +  
 geom\_col() + ggtitle("geom\_col()")



ggplot(data = diamonds,mapping = aes(x = cut)) +  
 geom\_bar() + ggtitle("geom\_bar()")



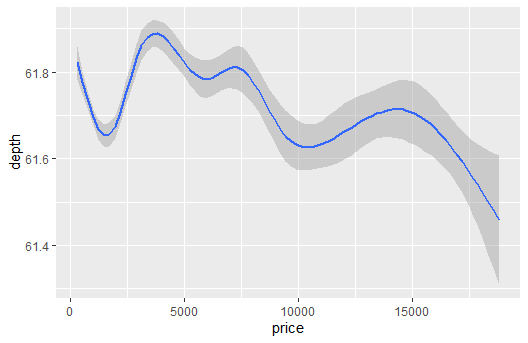
# The main difference between the functions is that geom\_col() maps a categorical variable to a continuous one while geom\_bar() just counts the number of a specific variable.

### (3) 3.7.1 Exercise 4, (5 pts)

What variables does stat\_smooth() compute? What parameters control its behavior? # stat\_smooth(), or geom\_smooth() takes the plot from two continous variables and attempts to map the pattern with a line rather than a scatterplot.

ggplot(data = diamonds) +  
 geom\_smooth(mapping = aes(x = price, y = depth))

## `geom\_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'



### (4) 3.7.1 Exercise 5, (10 pts)

In our proportion bar chart, we must set group = 1. Why? In other words, what is wrong with these two plots? # Group represents the percentage of data, group = 1 allows for all the data to appear meaning that the below plots could be missing important information.

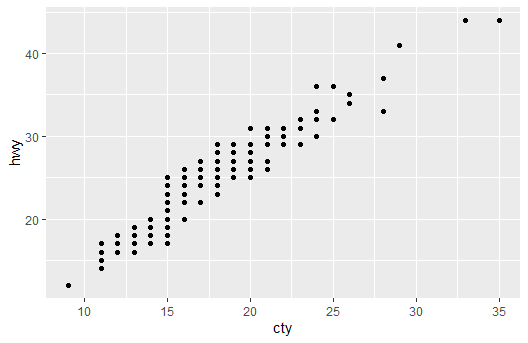
ggplot(data = diamonds) +   
 geom\_bar(mapping = aes(x = cut, y = after\_stat(prop)))  
ggplot(data = diamonds) +   
 geom\_bar(mapping = aes(x = cut, fill = color, y = after\_stat(prop)))

## 3.8.1 Exercises

### (5) 3.8.1 Exercise 1, (10 pts)

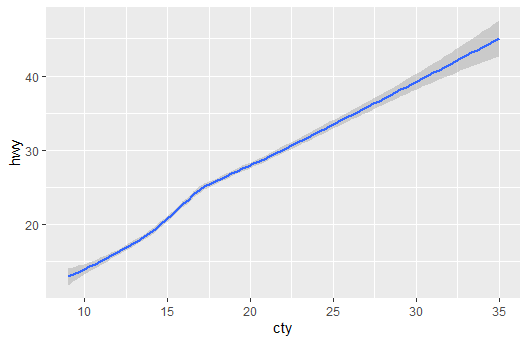
Given the context of this chapter, what is the problem with this plot? How could you improve it? # In my opinion, the scatterplot does not capture the linear element of these components as well as geom\_smooth() would capture it

ggplot(data = mpg, mapping = aes(x = cty, y = hwy)) +   
 geom\_point()



ggplot(data = mpg, mapping = aes(x = cty, y = hwy)) +   
 geom\_smooth()

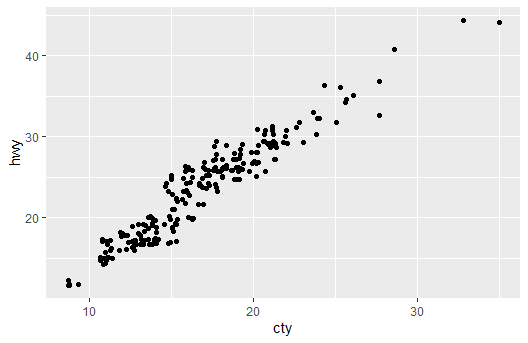
## `geom\_smooth()` using method = 'loess' and formula 'y ~ x'



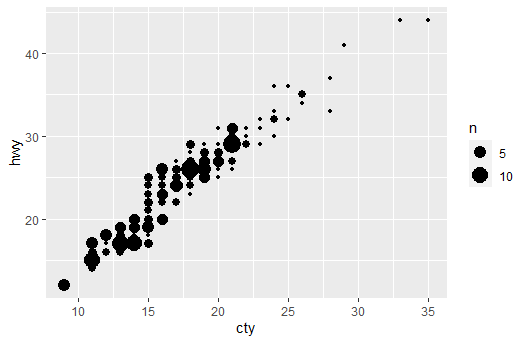
### (6) 3.8.1 Exercise 3, (10 pts)

Compare and contrast geom\_jitter() with geom\_count(). Demonstrate the kind of graph that geom\_count() creates. # geom\_jitter() and geom\_count() are both variations of geom\_point(). The geom\_jitter() command adds variation to the data points so that it is easier for the intrepreter of the graph to see correlation in small datasets. The command geom\_count() is used describe density of the data by increasing the size of points. Personally, I see the use in geom\_jitter() but I would much prefer to use geom\_count() to cover up missing points in the graph.

ggplot(data = mpg, mapping = aes(x = cty, y = hwy)) +   
 geom\_jitter()



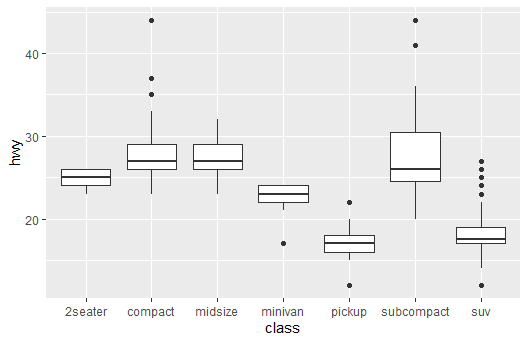
ggplot(data = mpg, mapping = aes(x = cty, y = hwy)) +   
 geom\_count()



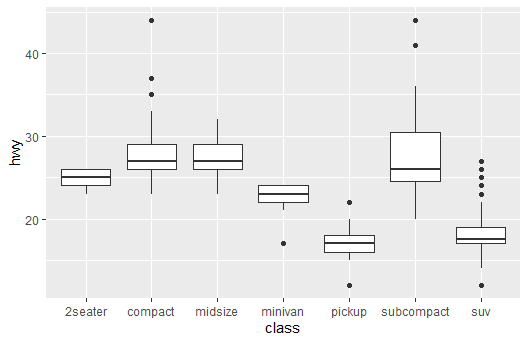
### (7) 3.8.1 Exercise 4, (10 pts)

What is the default position adjustment for geom\_boxplot()? Create a visualization of the mpg data set that demonstrates it. # The default position adjustment is “dodge2” this is used to make sure each boxplot will have space between each other. If the classes could interconnect, then changing from “dodge2” to “identity” will allow the boxplots to stack.

ggplot(data = mpg, mapping = aes(x = class, y = hwy)) +   
 geom\_boxplot()



ggplot(data = mpg, mapping = aes(x = class, y = hwy)) +   
 geom\_boxplot(position = "identity")

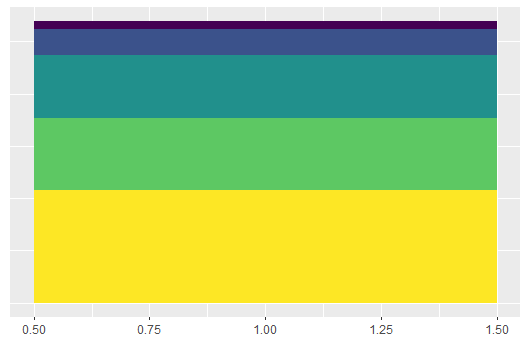


## 3.9.1 Exercises

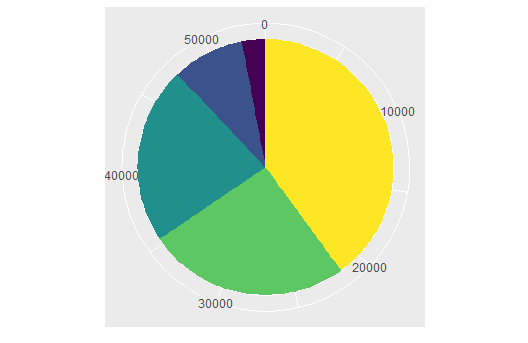
### (8) 3.9.1 Exercise 1, (10 pts)

Turn a stacked bar chart into a pie chart using coord\_polar().

bar2 <- ggplot(data=diamonds) +  
 geom\_bar(mapping = aes(x=1,fill = cut), show.legend = FALSE, width = 1) + # Changing cut to x=1  
 theme(axis.ticks.y = element\_blank(), axis.text.y=element\_blank()) + # Changing Theme  
 labs(x=NULL, y=NULL)  
  
bar2



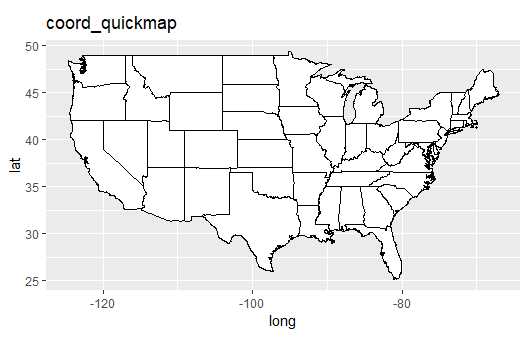
bar2 + coord\_polar(theta = "y")



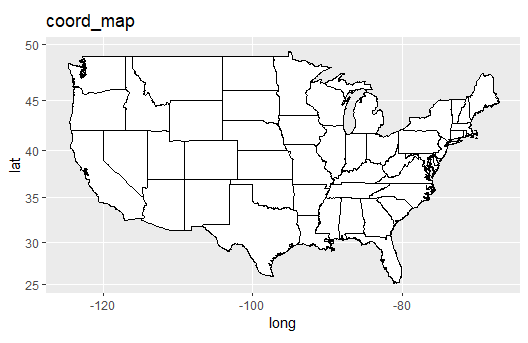
### (9) 3.9.1 Exercise 3, (5 pts)

What is the difference between coord\_quickmap() and coord\_map()? # They are the same function but coord\_quickmap() is predetermined algorithm to prevent distortion of the straightlines from the curvature of the earth.

state <- as\_tibble(map\_data("state"))  
  
ggplot(state, aes(x = long, lat, group=group)) +   
 geom\_polygon(fill="white", colour = 'black') +  
 coord\_quickmap() +  
 ggtitle("coord\_quickmap")



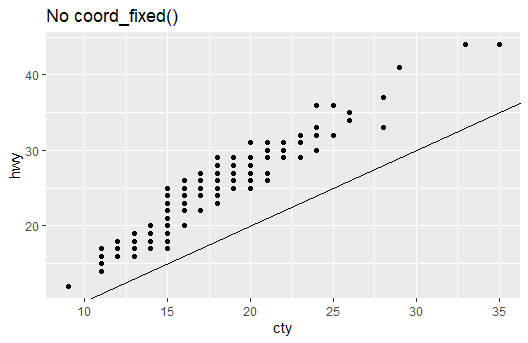
ggplot(state, aes(x = long, lat, group=group)) +   
 geom\_polygon(fill="white", colour = 'black') +  
 coord\_map() +  
 ggtitle("coord\_map")



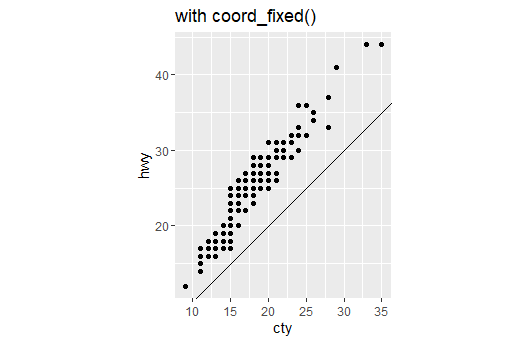
### (10) 3.9.1 Exercise 4, (10 pts)

What does the plot below tell you about the relationship between city and highway mpg? Why is coord\_fixed() important? What does geom\_abline() do? # The graph tells me the data has a linear relationship. It is important to have coord\_fixed() because it allows us directly compare x and y (since they are the same metric) without any distortion. The geom\_abline() command adds the linear relationship between the data, if the coord\_fixed() function is used the slope can be gathered from the abline allowing for more information to be gathered.

ggplot(data = mpg, mapping = aes(x = cty, y = hwy)) +  
 geom\_point() +   
 geom\_abline() +  
 ggtitle("No coord\_fixed()")



ggplot(data = mpg, mapping = aes(x = cty, y = hwy)) +  
 geom\_point() +   
 geom\_abline() +  
 coord\_fixed() +  
 ggtitle("with coord\_fixed()")



## 5.2.4 Exercises

### (11) 5.2.4 Exercise 1, 70 pts (10 each))

For the nycflights13::flights data set, find all flights that:

flights

## # A tibble: 336,776 x 19  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time sched\_arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int> <int>  
## 1 2013 1 1 517 515 2 830 819  
## 2 2013 1 1 533 529 4 850 830  
## 3 2013 1 1 542 540 2 923 850  
## 4 2013 1 1 544 545 -1 1004 1022  
## 5 2013 1 1 554 600 -6 812 837  
## 6 2013 1 1 554 558 -4 740 728  
## 7 2013 1 1 555 600 -5 913 854  
## 8 2013 1 1 557 600 -3 709 723  
## 9 2013 1 1 557 600 -3 838 846  
## 10 2013 1 1 558 600 -2 753 745  
## # ... with 336,766 more rows, and 11 more variables: arr\_delay <dbl>,  
## # carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,  
## # air\_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time\_hour <dttm>

1. Had an arrival delay of two or more hours

filter(flights, arr\_delay >= 120)

## # A tibble: 10,200 x 19  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time sched\_arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int> <int>  
## 1 2013 1 1 811 630 101 1047 830  
## 2 2013 1 1 848 1835 853 1001 1950  
## 3 2013 1 1 957 733 144 1056 853  
## 4 2013 1 1 1114 900 134 1447 1222  
## 5 2013 1 1 1505 1310 115 1638 1431  
## 6 2013 1 1 1525 1340 105 1831 1626  
## 7 2013 1 1 1549 1445 64 1912 1656  
## 8 2013 1 1 1558 1359 119 1718 1515  
## 9 2013 1 1 1732 1630 62 2028 1825  
## 10 2013 1 1 1803 1620 103 2008 1750  
## # ... with 10,190 more rows, and 11 more variables: arr\_delay <dbl>,  
## # carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,  
## # air\_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time\_hour <dttm>

1. Flew to Houston (IAH or HOU)

filter(flights, dest == 'IAH' | dest == 'HOU')

## # A tibble: 9,313 x 19  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time sched\_arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int> <int>  
## 1 2013 1 1 517 515 2 830 819  
## 2 2013 1 1 533 529 4 850 830  
## 3 2013 1 1 623 627 -4 933 932  
## 4 2013 1 1 728 732 -4 1041 1038  
## 5 2013 1 1 739 739 0 1104 1038  
## 6 2013 1 1 908 908 0 1228 1219  
## 7 2013 1 1 1028 1026 2 1350 1339  
## 8 2013 1 1 1044 1045 -1 1352 1351  
## 9 2013 1 1 1114 900 134 1447 1222  
## 10 2013 1 1 1205 1200 5 1503 1505  
## # ... with 9,303 more rows, and 11 more variables: arr\_delay <dbl>,  
## # carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,  
## # air\_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time\_hour <dttm>

1. Were operated by United, American, or Delta

filter(flights, carrier == 'DL' | carrier == 'AA' | carrier == 'US')

## # A tibble: 101,375 x 19  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time sched\_arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int> <int>  
## 1 2013 1 1 542 540 2 923 850  
## 2 2013 1 1 554 600 -6 812 837  
## 3 2013 1 1 558 600 -2 753 745  
## 4 2013 1 1 559 600 -1 941 910  
## 5 2013 1 1 602 610 -8 812 820  
## 6 2013 1 1 606 610 -4 858 910  
## 7 2013 1 1 606 610 -4 837 845  
## 8 2013 1 1 615 615 0 833 842  
## 9 2013 1 1 622 630 -8 1017 1014  
## 10 2013 1 1 623 610 13 920 915  
## # ... with 101,365 more rows, and 11 more variables: arr\_delay <dbl>,  
## # carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,  
## # air\_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time\_hour <dttm>

1. Departed in summer (July, August, and September)

filter(flights, month <= 9, month >= 7)

## # A tibble: 86,326 x 19  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time sched\_arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int> <int>  
## 1 2013 7 1 1 2029 212 236 2359  
## 2 2013 7 1 2 2359 3 344 344  
## 3 2013 7 1 29 2245 104 151 1  
## 4 2013 7 1 43 2130 193 322 14  
## 5 2013 7 1 44 2150 174 300 100  
## 6 2013 7 1 46 2051 235 304 2358  
## 7 2013 7 1 48 2001 287 308 2305  
## 8 2013 7 1 58 2155 183 335 43  
## 9 2013 7 1 100 2146 194 327 30  
## 10 2013 7 1 100 2245 135 337 135  
## # ... with 86,316 more rows, and 11 more variables: arr\_delay <dbl>,  
## # carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,  
## # air\_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time\_hour <dttm>

1. Arrived more than two hours late, but didn’t leave late

filter(flights, dep\_delay == 0, arr\_delay >= 120)

## # A tibble: 3 x 19  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time sched\_arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int> <int>  
## 1 2013 10 7 1350 1350 0 1736 1526  
## 2 2013 5 23 1810 1810 0 2208 2000  
## 3 2013 7 1 905 905 0 1443 1223  
## # ... with 11 more variables: arr\_delay <dbl>, carrier <chr>, flight <int>,  
## # tailnum <chr>, origin <chr>, dest <chr>, air\_time <dbl>, distance <dbl>,  
## # hour <dbl>, minute <dbl>, time\_hour <dttm>

1. Were delayed by at least an hour, but made up over 30 minutes in flight

filter(flights, dep\_delay >= 60, dep\_delay - arr\_delay >= 30)

## # A tibble: 2,074 x 19  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time sched\_arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int> <int>  
## 1 2013 1 1 1716 1545 91 2140 2039  
## 2 2013 1 1 2205 1720 285 46 2040  
## 3 2013 1 1 2326 2130 116 131 18  
## 4 2013 1 3 1503 1221 162 1803 1555  
## 5 2013 1 3 1821 1530 171 2131 1910  
## 6 2013 1 3 1839 1700 99 2056 1950  
## 7 2013 1 3 1850 1745 65 2148 2120  
## 8 2013 1 3 1923 1815 68 2036 1958  
## 9 2013 1 3 1941 1759 102 2246 2139  
## 10 2013 1 3 1950 1845 65 2228 2227  
## # ... with 2,064 more rows, and 11 more variables: arr\_delay <dbl>,  
## # carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,  
## # air\_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time\_hour <dttm>

1. Departed between midnight and 6am (inclusive)

filter(flights, dep\_time >= 0, dep\_time <= 600)

## # A tibble: 9,344 x 19  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time sched\_arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int> <int>  
## 1 2013 1 1 517 515 2 830 819  
## 2 2013 1 1 533 529 4 850 830  
## 3 2013 1 1 542 540 2 923 850  
## 4 2013 1 1 544 545 -1 1004 1022  
## 5 2013 1 1 554 600 -6 812 837  
## 6 2013 1 1 554 558 -4 740 728  
## 7 2013 1 1 555 600 -5 913 854  
## 8 2013 1 1 557 600 -3 709 723  
## 9 2013 1 1 557 600 -3 838 846  
## 10 2013 1 1 558 600 -2 753 745  
## # ... with 9,334 more rows, and 11 more variables: arr\_delay <dbl>,  
## # carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,  
## # air\_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time\_hour <dttm>

### (12) 5.2.4 Exercise 4, (5 pts)

Why is NA ^ 0 not missing? Why is NA | TRUE not missing? Why is FALSE & NA not missing? Can you figure out the general rule? (NA \* 0 is a tricky counterexample!)

#NA ^ 0 = 1 # because all values raised to zero are one  
#NA | TRUE = TRUE # because "|" is an or statement and "or TRUE" is always true.  
#NA & FALSE = FALSE # because "," is an and function meaning "and FALSE" will always be false.  
  
# The general rule seems to be, if it applies to all numbers then it applies to NA.

## 5.3.1 Exercises

### (13) 5.3.1 Exercise 1, (10 pts)

# Checking for NA values and putting them at the top then the rest of the code

arrange(flights, desc(is.na(dep\_time)), dep\_time)

## # A tibble: 336,776 x 19  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time sched\_arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int> <int>  
## 1 2013 1 1 NA 1630 NA NA 1815  
## 2 2013 1 1 NA 1935 NA NA 2240  
## 3 2013 1 1 NA 1500 NA NA 1825  
## 4 2013 1 1 NA 600 NA NA 901  
## 5 2013 1 2 NA 1540 NA NA 1747  
## 6 2013 1 2 NA 1620 NA NA 1746  
## 7 2013 1 2 NA 1355 NA NA 1459  
## 8 2013 1 2 NA 1420 NA NA 1644  
## 9 2013 1 2 NA 1321 NA NA 1536  
## 10 2013 1 2 NA 1545 NA NA 1910  
## # ... with 336,766 more rows, and 11 more variables: arr\_delay <dbl>,  
## # carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,  
## # air\_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time\_hour <dttm>

### (14) 5.3.1 Exercise 4, (10 pts)

Which flights traveled the longest distance? Which traveled the shortest? # Top is the shortest flights since it is in ascending order, and vice-versa.

arrange(flights, distance)

## # A tibble: 336,776 x 19  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time sched\_arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int> <int>  
## 1 2013 7 27 NA 106 NA NA 245  
## 2 2013 1 3 2127 2129 -2 2222 2224  
## 3 2013 1 4 1240 1200 40 1333 1306  
## 4 2013 1 4 1829 1615 134 1937 1721  
## 5 2013 1 4 2128 2129 -1 2218 2224  
## 6 2013 1 5 1155 1200 -5 1241 1306  
## 7 2013 1 6 2125 2129 -4 2224 2224  
## 8 2013 1 7 2124 2129 -5 2212 2224  
## 9 2013 1 8 2127 2130 -3 2304 2225  
## 10 2013 1 9 2126 2129 -3 2217 2224  
## # ... with 336,766 more rows, and 11 more variables: arr\_delay <dbl>,  
## # carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,  
## # air\_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time\_hour <dttm>

arrange(flights, desc(distance))

## # A tibble: 336,776 x 19  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time sched\_arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int> <int>  
## 1 2013 1 1 857 900 -3 1516 1530  
## 2 2013 1 2 909 900 9 1525 1530  
## 3 2013 1 3 914 900 14 1504 1530  
## 4 2013 1 4 900 900 0 1516 1530  
## 5 2013 1 5 858 900 -2 1519 1530  
## 6 2013 1 6 1019 900 79 1558 1530  
## 7 2013 1 7 1042 900 102 1620 1530  
## 8 2013 1 8 901 900 1 1504 1530  
## 9 2013 1 9 641 900 1301 1242 1530  
## 10 2013 1 10 859 900 -1 1449 1530  
## # ... with 336,766 more rows, and 11 more variables: arr\_delay <dbl>,  
## # carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,  
## # air\_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time\_hour <dttm>

## 5.4.1 Exercises

### (15) 5.4.1 Exercise 1, (10 pts)

Brainstorm as many ways as possible to select dep\_time, dep\_delay, arr\_time, and arr\_delay from flights.

dplyr::select(flights, dep\_time, dep\_delay, arr\_time, arr\_delay)

## # A tibble: 336,776 x 4  
## dep\_time dep\_delay arr\_time arr\_delay  
## <int> <dbl> <int> <dbl>  
## 1 517 2 830 11  
## 2 533 4 850 20  
## 3 542 2 923 33  
## 4 544 -1 1004 -18  
## 5 554 -6 812 -25  
## 6 554 -4 740 12  
## 7 555 -5 913 19  
## 8 557 -3 709 -14  
## 9 557 -3 838 -8  
## 10 558 -2 753 8  
## # ... with 336,766 more rows

dplyr::select(flights, 4, 6, 7, 9)

## # A tibble: 336,776 x 4  
## dep\_time dep\_delay arr\_time arr\_delay  
## <int> <dbl> <int> <dbl>  
## 1 517 2 830 11  
## 2 533 4 850 20  
## 3 542 2 923 33  
## 4 544 -1 1004 -18  
## 5 554 -6 812 -25  
## 6 554 -4 740 12  
## 7 555 -5 913 19  
## 8 557 -3 709 -14  
## 9 557 -3 838 -8  
## 10 558 -2 753 8  
## # ... with 336,766 more rows

dplyr::select(flights, "dep\_time", "dep\_delay", "arr\_time", "arr\_delay")

## # A tibble: 336,776 x 4  
## dep\_time dep\_delay arr\_time arr\_delay  
## <int> <dbl> <int> <dbl>  
## 1 517 2 830 11  
## 2 533 4 850 20  
## 3 542 2 923 33  
## 4 544 -1 1004 -18  
## 5 554 -6 812 -25  
## 6 554 -4 740 12  
## 7 555 -5 913 19  
## 8 557 -3 709 -14  
## 9 557 -3 838 -8  
## 10 558 -2 753 8  
## # ... with 336,766 more rows

features <- c("dep\_time", "dep\_delay", "arr\_time", "arr\_delay")  
dplyr::select(flights, all\_of(features))

## # A tibble: 336,776 x 4  
## dep\_time dep\_delay arr\_time arr\_delay  
## <int> <dbl> <int> <dbl>  
## 1 517 2 830 11  
## 2 533 4 850 20  
## 3 542 2 923 33  
## 4 544 -1 1004 -18  
## 5 554 -6 812 -25  
## 6 554 -4 740 12  
## 7 555 -5 913 19  
## 8 557 -3 709 -14  
## 9 557 -3 838 -8  
## 10 558 -2 753 8  
## # ... with 336,766 more rows

dplyr::select(flights, any\_of(features))

## # A tibble: 336,776 x 4  
## dep\_time dep\_delay arr\_time arr\_delay  
## <int> <dbl> <int> <dbl>  
## 1 517 2 830 11  
## 2 533 4 850 20  
## 3 542 2 923 33  
## 4 544 -1 1004 -18  
## 5 554 -6 812 -25  
## 6 554 -4 740 12  
## 7 555 -5 913 19  
## 8 557 -3 709 -14  
## 9 557 -3 838 -8  
## 10 558 -2 753 8  
## # ... with 336,766 more rows

### (16) 5.4.1 Exercise 3, (10 pts)

What does the any\_of() function do? Why might it be helpful in conjunction with this vector?

# This can be helpful if we need data points from specific columns, and if a column is not included then there will not be an error.

vars <- c("year", "month", "day", "dep\_delay", "arr\_delay")  
dplyr::select(flights, any\_of(vars))

## 5.5.2 Exercises

### (17) 5.5.2 Exercise 1, (10 pts)

Currently dep\_time and sched\_dep\_time are convenient to look at, but hard to compute with because they’re not really continuous numbers. Convert them to the more computationally convenient representation of number of minutes since midnight.

# First half is converting the hour side to minutes and vice versa. The data was also improplerly scaled so I converted all values minus 1440 since that was the minimum I found.

flights\_times <- mutate(flights,  
 dep\_time\_mins = (dep\_time %/% 100 \* 60 + dep\_time %% 100) %% 1440,  
 sched\_dep\_time\_mins = (sched\_dep\_time %/% 100 \* 60 + sched\_dep\_time %% 100) %% 1440)  
   
flights\_times

## # A tibble: 336,776 x 21  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time sched\_arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int> <int>  
## 1 2013 1 1 517 515 2 830 819  
## 2 2013 1 1 533 529 4 850 830  
## 3 2013 1 1 542 540 2 923 850  
## 4 2013 1 1 544 545 -1 1004 1022  
## 5 2013 1 1 554 600 -6 812 837  
## 6 2013 1 1 554 558 -4 740 728  
## 7 2013 1 1 555 600 -5 913 854  
## 8 2013 1 1 557 600 -3 709 723  
## 9 2013 1 1 557 600 -3 838 846  
## 10 2013 1 1 558 600 -2 753 745  
## # ... with 336,766 more rows, and 13 more variables: arr\_delay <dbl>,  
## # carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,  
## # air\_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time\_hour <dttm>,  
## # dep\_time\_mins <dbl>, sched\_dep\_time\_mins <dbl>

### (18) 5.5.2 Exercise 2, (10 pts)

Compare air\_time with arr\_time - dep\_time. What do you expect to see? What do you see? What do you need to do to fix it?

# I expect the values to equal one another.

# That is not the case, I suspect the reason is becasue how I changed midnight making the some one hour flights through midnight calculate for 23 hours.

# I am unsure how to fix the issue.

flights\_new <- mutate(flights\_times,  
 diff = air\_time - arr\_time + dep\_time)  
  
ggplot(data = flights\_new) +  
 geom\_point(mapping = aes(x=dep\_time, y =arr\_time, fill = diff))

## Warning: Removed 8713 rows containing missing values (geom\_point).

