COMPSCIX 415.2 Homework 3

Sanatan Das

February 17, 2018

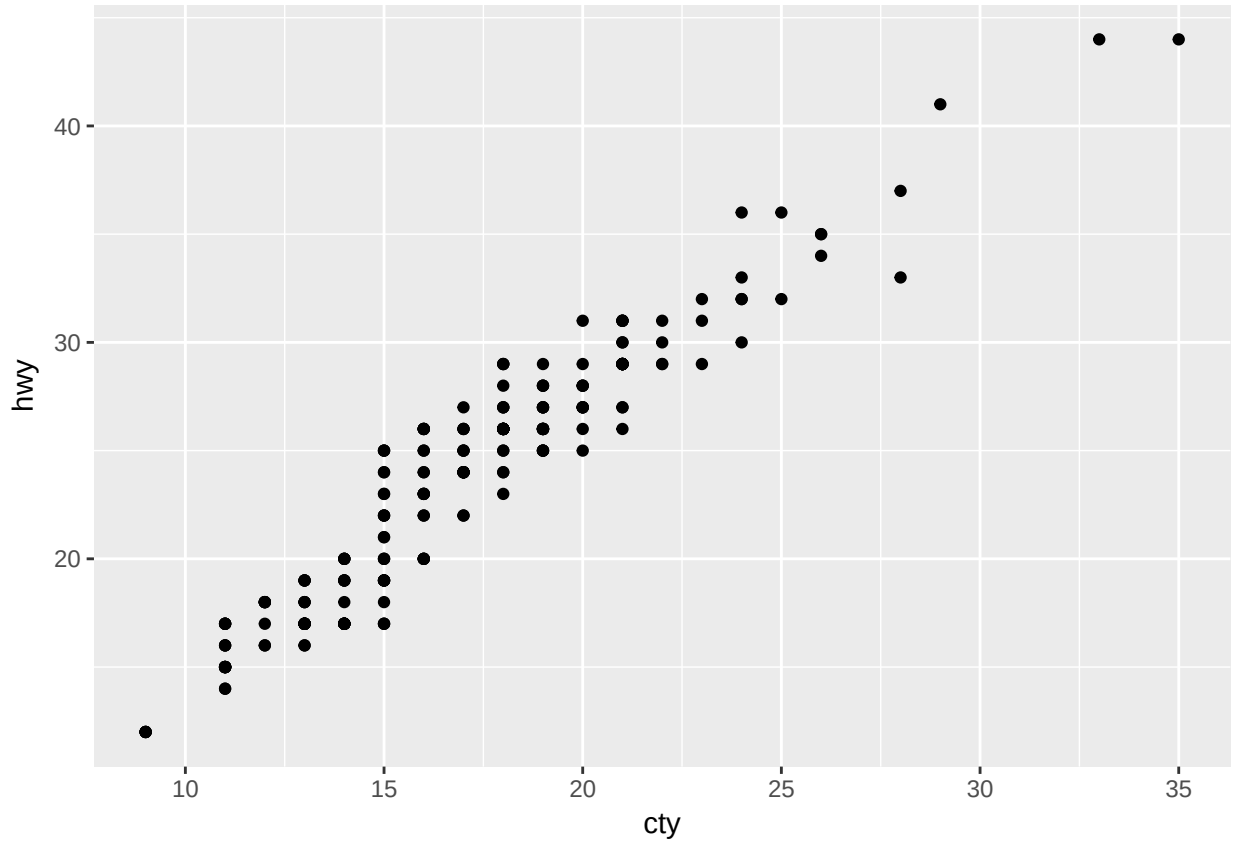
Load packages

```
library(tidyverse)
library(nycflights13)
```

3.8.1 Exercises

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

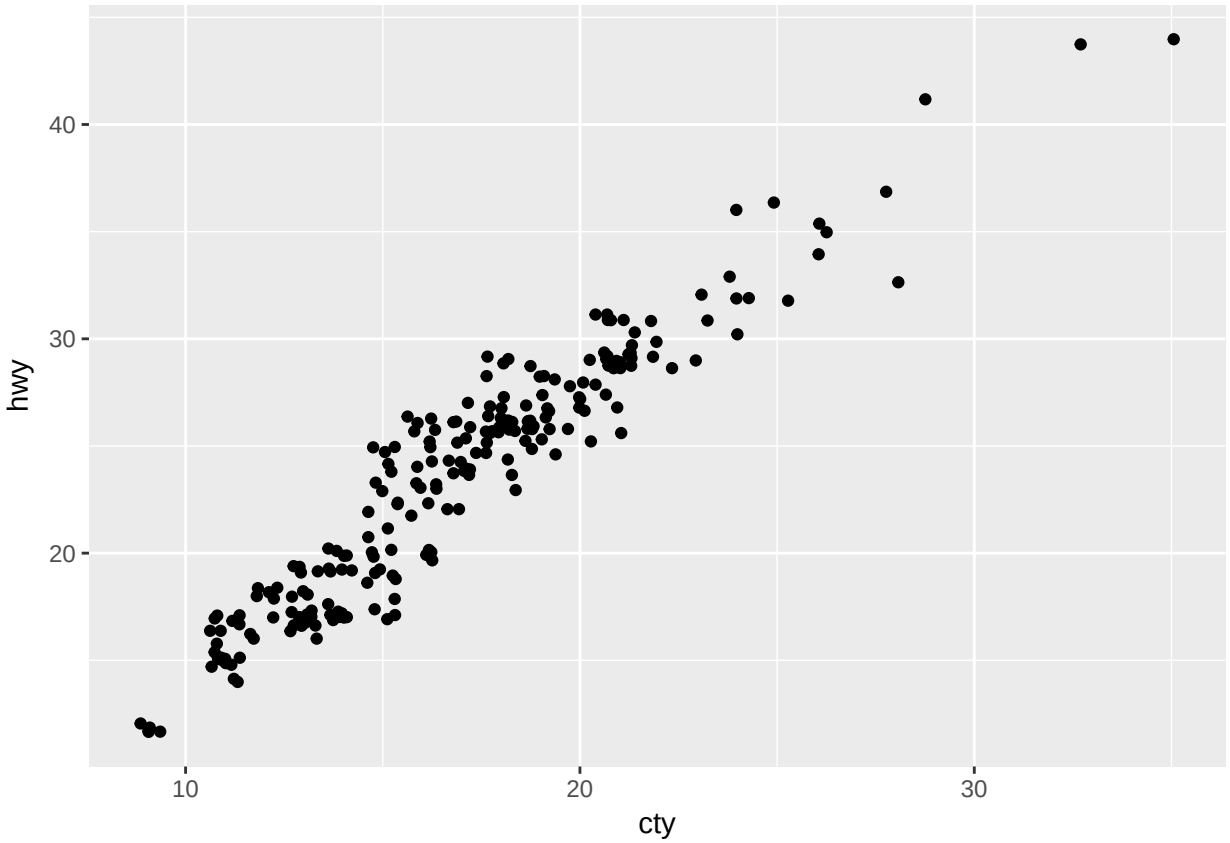
```
ggplot(data = mpg, mapping = aes(x = cty, y = hwy)) +
  geom_point()
```



ANSWER 1:

In the above plot many of the data points overlap. We can jitter the points by adding some slight random noise, which will improve the overall visualization as below.

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



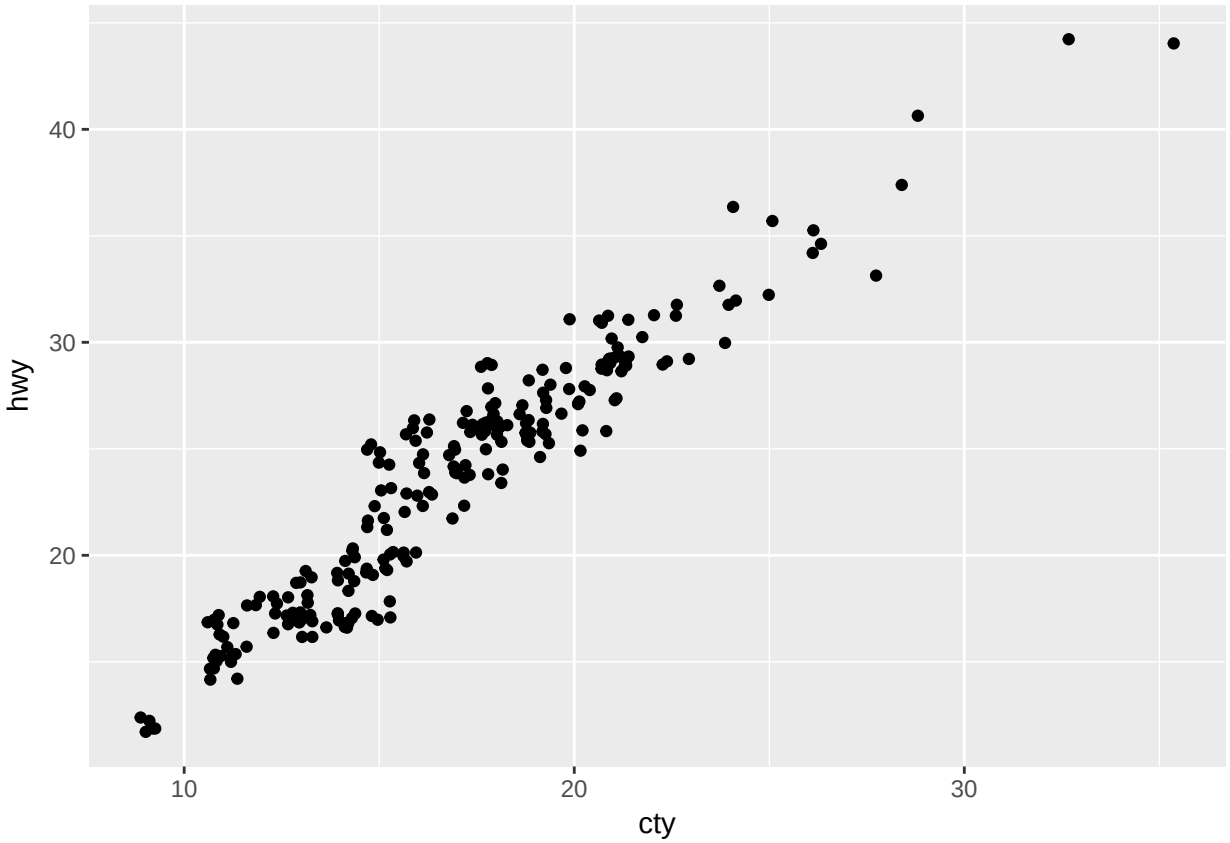
QUESTION 2: What parameters to geom_jitter() control the amount of jittering?

ANSWER 2: *width* and *height* are the geom_jitter() parameters to control the amount of jittering (horizontal and vertical)

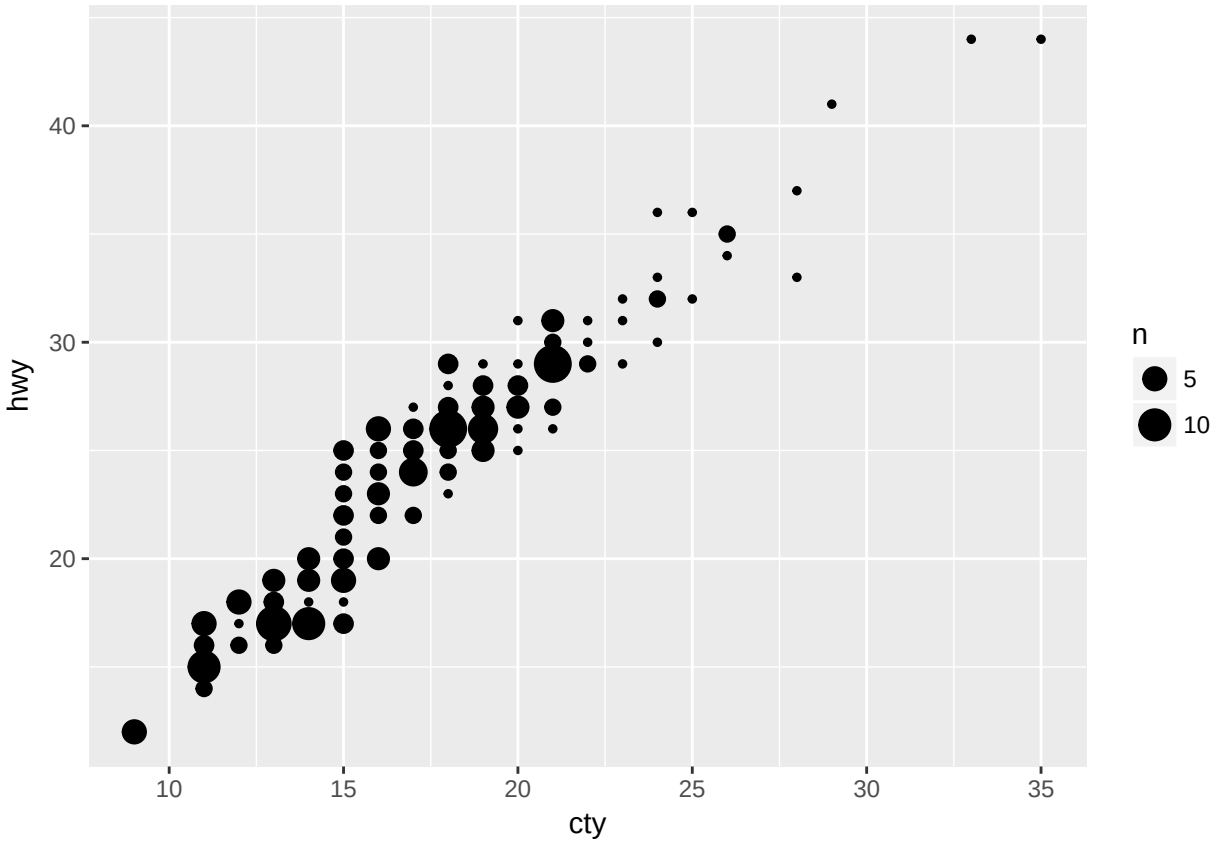
QUESTION 3: Compare and contrast geom_jitter() with geom_count().

ANSWER 3: geom_count() counts the number of observations at each location rather than adding random noise, then maps the count to point area. It makes larger points the more observations are located at that area, so the number of visible points is equal to geom_point(). Please see the below graphs.

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




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

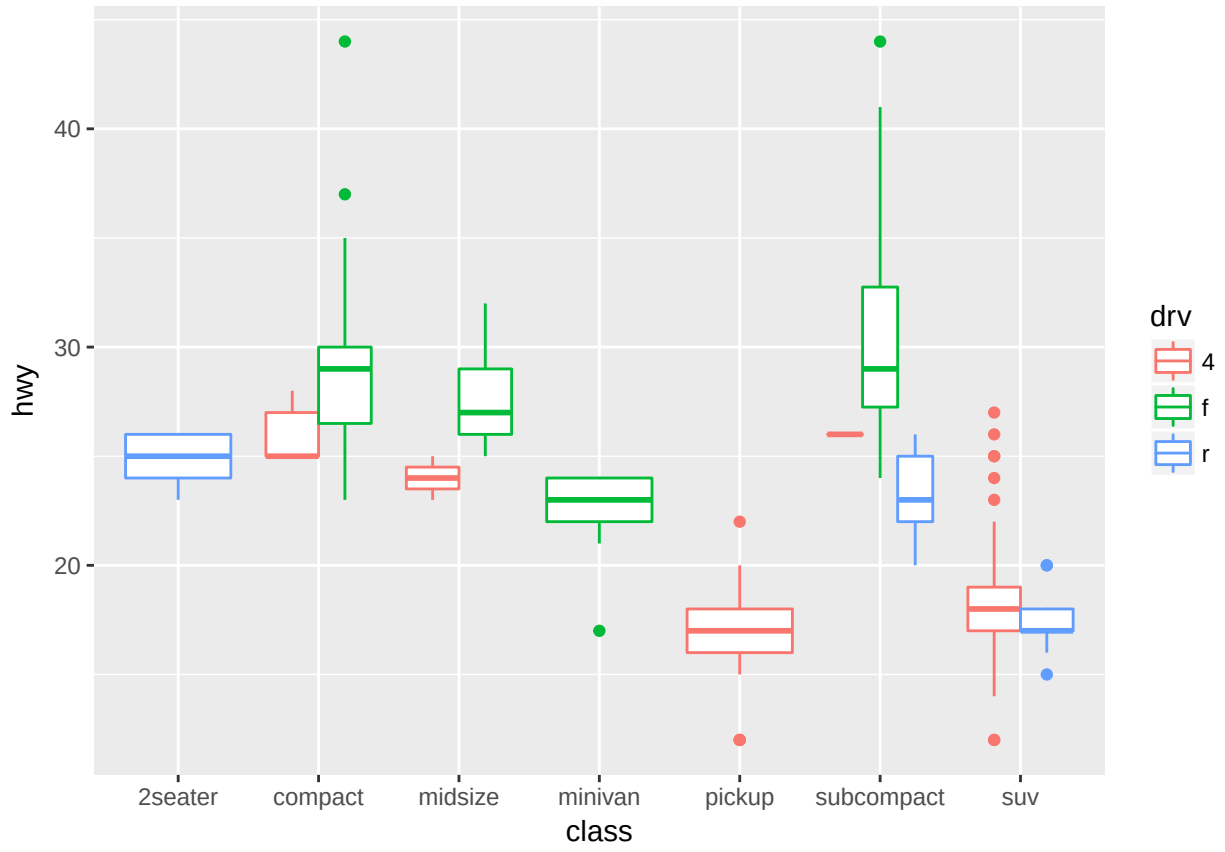


QUESTION 4: What's the default position adjustment for geom_boxplot()? Create a visualisation of the mpg dataset that demonstrates it.

ANSWER 4: The default position adjustment for geom_boxplot() is position_dodge(). Please see below.

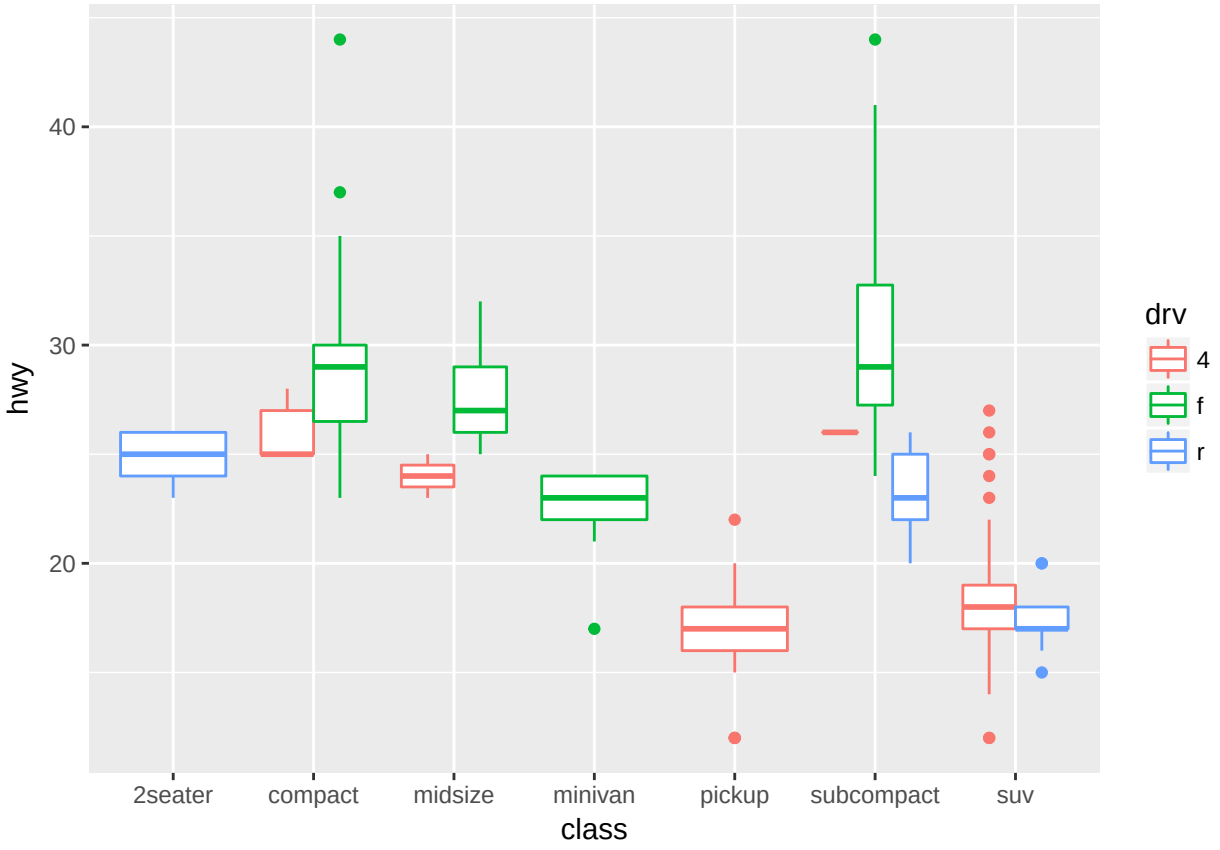
without position adjustment

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



with position adjustment

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



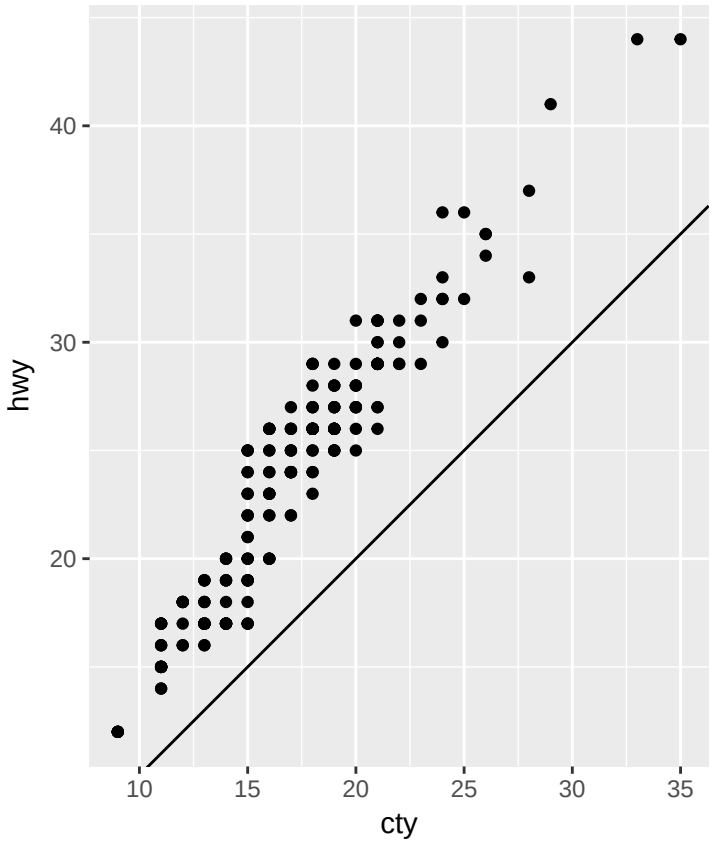
3.9.1 Exercises

QUESTION 2: What does labs() do?

ANSWER 2: *labs()* adds labels to the graph. A title, subtitle, a label for the x-axis and y-axis, as well as a caption can be added to a plot using labs().

QUESTION 4: 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?

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



ANSWER 4: In the above plot, the relationship is approximately linear, though overall cars have slightly better highway mileage than city mileage.

coord_fixed() draws equal intervals on the x-axis and y-axis so they are directly comparable.

geom_abline() draws a line that has an intercept of 0 and slope of 1. In the above plot, this tells us that automobile gas efficiency is on average slightly higher for highways than city driving.

4.4 Exercises

QUESTION 1: Why does this code not work?

my_variable <- 10 my_varıable #> Error in eval(expr, envir, enclos): object 'my_varıable' not found

ANSWER 2:The second line has a typo. It should be my_variable, not my_varıable.

QUESTION 2: Tweak each of the following R commands so that they run correctly:

library(tidyverse)

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

fliter(mpg, cyl = 8)

filter(diamond, carat > 3)

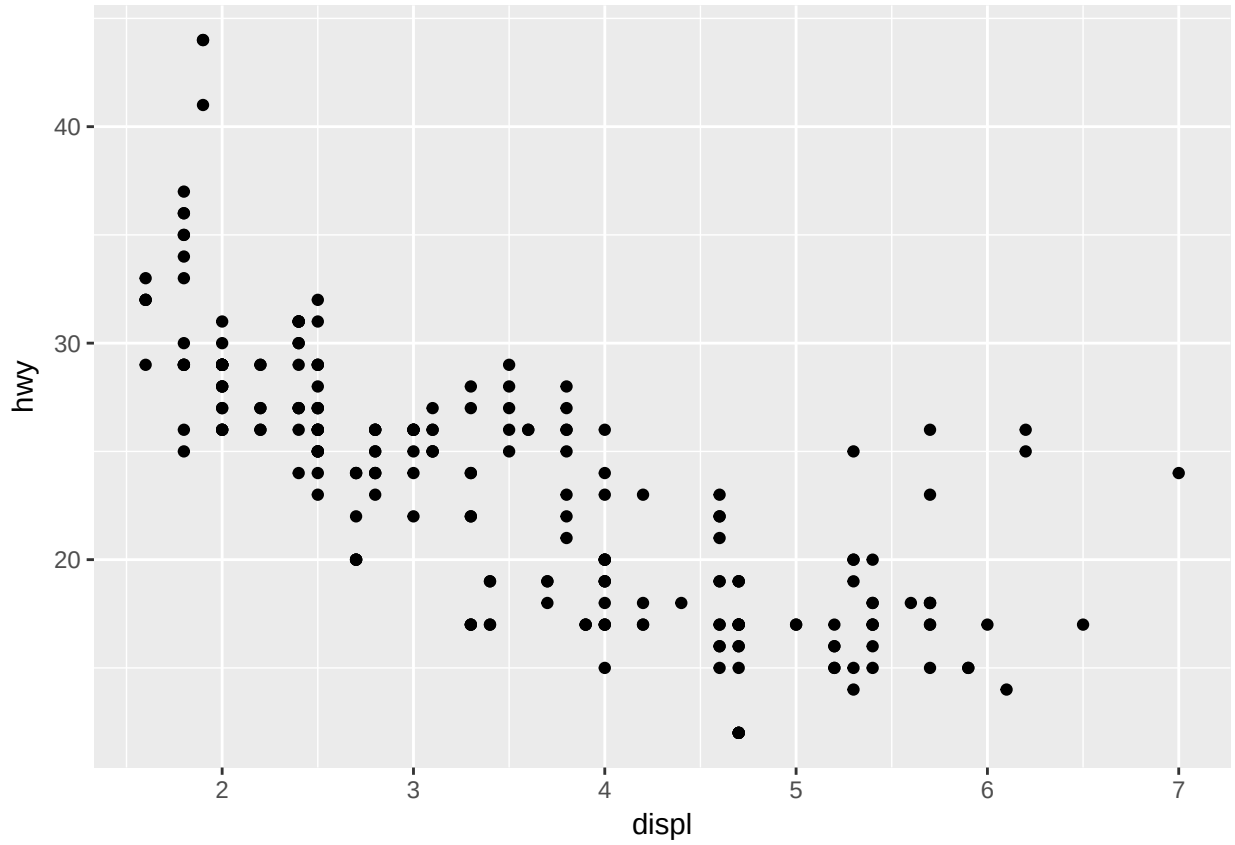
ANSWER 2:

```
library(tidyverse)

# incorrect
```

```
# ggplot(dota = mpg) +
  # geom_point(mapping = aes(x = displ, y = hwy))
# Error in structure(list(data = data, layers = list(), scales = scales_list(), : argument "data" is mi

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




```
# incorrect
# fliter(mpg, cyl = 8)

#Error in fliter(mpg, cyl = 8) : could not find function "fliter" Calls: <Anonymous> ... handle -> with

# incorrect
# filter(diamond, carat > 3)

# Error in filter(diamond, carat > 3) : object 'diamond' not found Calls: <Anonymous> ... withCallingHa

# correct
filter(mpg, cyl == 8)
```

```
## # A tibble: 70 x 11
##    manufacturer model     displ  year   cyl trans  drv     cty   hwy fl
##    <chr>        <chr>     <dbl> <int> <int> <chr>  <chr> <int> <int> <chr>
##  1 audi         a6 quatt~  4.20  2008     8 auto(~ 4        16    23 p
##  2 chevrolet    c1500 su~  5.30  2008     8 auto(~ r        14    20 r
```

```
##  3 chevrolet    c1500 su~  5.30  2008     8 auto(~ r        11    15 e
##  4 chevrolet    c1500 su~  5.30  2008     8 auto(~ r        14    20 r
##  5 chevrolet    c1500 su~  5.70  1999     8 auto(~ r        13    17 r
##  6 chevrolet    c1500 su~  6.00  2008     8 auto(~ r        12    17 r
##  7 chevrolet    corvette   5.70  1999     8 manua~ r        16    26 p
##  8 chevrolet    corvette   5.70  1999     8 auto(~ r        15    23 p
##  9 chevrolet    corvette   6.20  2008     8 manua~ r        16    26 p
## 10 chevrolet    corvette   6.20  2008     8 auto(~ r        15    25 p
## # ... with 60 more rows, and 1 more variable: class <chr>
```

```
filter(diamonds, carat > 3)
```

```
## # A tibble: 32 x 10
##    carat cut     color clarity depth table price     x     y     z
##    <dbl> <ord>   <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
##  1  3.01 Premium I     I1       62.7  58.0  8040  9.10  8.97  5.67
##  2  3.11 Fair    J     I1       65.9  57.0  9823  9.15  9.02  5.98
##  3  3.01 Premium F     I1       62.2  56.0  9925  9.24  9.13  5.73
##  4  3.05 Premium E     I1       60.9  58.0 10453  9.26  9.25  5.66
##  5  3.02 Fair    I     I1       65.2  56.0 10577  9.11  9.02  5.91
##  6  3.01 Fair    H     I1       56.1  62.0 10761  9.54  9.38  5.31
##  7  3.65 Fair    H     I1       67.1  53.0 11668  9.53  9.48  6.38
##  8  3.24 Premium H     I1       62.1  58.0 12300  9.44  9.40  5.85
##  9  3.22 Ideal   I     I1       62.6  55.0 12545  9.49  9.42  5.92
## 10  3.50 Ideal   H     I1       62.8  57.0 12587  9.65  9.59  6.03
## # ... with 22 more rows
```

5.2.4 Exercises

QUESTION 1: Find all flights that

1. Had an arrival delay of two or more hours
2. Flew to Houston (IAH or HOU)
3. Were operated by United, American, or Delta
4. Departed in summer (July, August, and September)
5. Arrived more than two hours late, but didn't leave late
6. Were delayed by at least an hour, but made up over 30 minutes in flight
7. Departed between midnight and 6am (inclusive)

1->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
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013     1     1      811            630     101       1047
##  2  2013     1     1      848           1835     853       1001
##  3  2013     1     1      957            733     144       1056
##  4  2013     1     1     1114            900     134       1447
##  5  2013     1     1     1505           1310     115       1638
##  6  2013     1     1     1525           1340     105       1831
##  7  2013     1     1     1549           1445      64.0     1912
##  8  2013     1     1     1558           1359     119       1718
##  9  2013     1     1     1732           1630      62.0     2028
## 10  2013     1     1     1803           1620     103       2008
```

```
## # ... with 10,190 more rows, and 12 more variables: sched_arr_time <int>,
## #   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>
```

2->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
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013     1     1      517            515      2.00      830
##  2  2013     1     1      533            529      4.00      850
##  3  2013     1     1      623            627   -  4.00      933
##  4  2013     1     1      728            732   -  4.00     1041
##  5  2013     1     1      739            739      0        1104
##  6  2013     1     1      908            908      0        1228
##  7  2013     1     1     1028           1026      2.00     1350
##  8  2013     1     1     1044           1045   -  1.00     1352
##  9  2013     1     1     1114            900    134        1447
## 10  2013     1     1     1205           1200      5.00     1503
## # ... with 9,303 more rows, and 12 more variables: sched_arr_time <int>,
## #   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>
```

3->operated by United, American, or Delta

```
filter(flights, carrier == "UA" |
         carrier == "AA" |
         carrier == "DL")
```

```
## # A tibble: 139,504 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013     1     1      517            515      2.00      830
##  2  2013     1     1      533            529      4.00      850
##  3  2013     1     1      542            540      2.00      923
##  4  2013     1     1      554            600     -6.00      812
##  5  2013     1     1      554            558     -4.00      740
##  6  2013     1     1      558            600     -2.00      753
##  7  2013     1     1      558            600     -2.00      924
##  8  2013     1     1      558            600     -2.00      923
##  9  2013     1     1      559            600     -1.00      941
## 10  2013     1     1      559            600     -1.00      854
## # ... with 139,494 more rows, and 12 more variables: sched_arr_time <int>,
## #   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>
```

4->Departed in summer (July, August, and September)

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

```
## # A tibble: 86,326 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
```

```
##  1  2013     7     1        1           2029    212         236
##  2  2013     7     1        2           2359      3.00      344
##  3  2013     7     1       29           2245    104         151
##  4  2013     7     1       43           2130    193         322
##  5  2013     7     1       44           2150    174         300
##  6  2013     7     1       46           2051    235         304
##  7  2013     7     1       48           2001    287         308
##  8  2013     7     1       58           2155    183         335
##  9  2013     7     1      100           2146    194         327
## 10  2013     7     1      100           2245    135         337
## # ... with 86,316 more rows, and 12 more variables: sched_arr_time <int>,
## #   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->Arrived more than two hours late, but didn't leave late

```
filter(flights, arr_delay >= 120, dep_delay <= 0)
```

```
## # A tibble: 29 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013     1    27     1419           1420     -1.00     1754
##  2  2013    10     7     1350           1350      0        1736
##  3  2013    10     7     1357           1359     -2.00     1858
##  4  2013    10    16      657            700     -3.00     1258
##  5  2013    11     1      658            700     -2.00     1329
##  6  2013     3    18     1844           1847     -3.00       39
##  7  2013     4    17     1635           1640     -5.00     2049
##  8  2013     4    18      558            600     -2.00     1149
##  9  2013     4    18      655            700     -5.00     1213
## 10  2013     5    22     1827           1830     -3.00     2217
## # ... with 19 more rows, and 12 more variables: sched_arr_time <int>,
## #   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>
```

6->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
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013     1     1     1716           1545      91.0     2140
##  2  2013     1     1     2205           1720     285         46
##  3  2013     1     1     2326           2130     116        131
##  4  2013     1     3     1503           1221     162       1803
##  5  2013     1     3     1821           1530     171       2131
##  6  2013     1     3     1839           1700      99.0     2056
##  7  2013     1     3     1850           1745      65.0     2148
##  8  2013     1     3     1923           1815      68.0     2036
##  9  2013     1     3     1941           1759     102       2246
## 10  2013     1     3     1950           1845      65.0     2228
## # ... with 2,064 more rows, and 12 more variables: sched_arr_time <int>,
## #   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>
```

7->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
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013     1     1      517            515      2.00      830
##  2  2013     1     1      533            529      4.00      850
##  3  2013     1     1      542            540      2.00      923
##  4  2013     1     1      544            545     -1.00     1004
##  5  2013     1     1      554            600     -6.00      812
##  6  2013     1     1      554            558     -4.00      740
##  7  2013     1     1      555            600     -5.00      913
##  8  2013     1     1      557            600     -3.00      709
##  9  2013     1     1      557            600     -3.00      838
## 10  2013     1     1      558            600     -2.00      753
## # ... with 9,334 more rows, and 12 more variables: sched_arr_time <int>,
## #   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>
```

QUESTION 2: Another useful dplyr filtering helper is between(). What does it do? Can you use it to simplify the code needed to answer the previous challenges?

ANSWER 2: It is an easy way to find observations between two values. We can use it as below.

Departed between midnight and 6am (inclusive)

```
# without between
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
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013     1     1      517            515      2.00      830
##  2  2013     1     1      533            529      4.00      850
##  3  2013     1     1      542            540      2.00      923
##  4  2013     1     1      544            545     -1.00     1004
##  5  2013     1     1      554            600     -6.00      812
##  6  2013     1     1      554            558     -4.00      740
##  7  2013     1     1      555            600     -5.00      913
##  8  2013     1     1      557            600     -3.00      709
##  9  2013     1     1      557            600     -3.00      838
## 10  2013     1     1      558            600     -2.00      753
## # ... with 9,334 more rows, and 12 more variables: sched_arr_time <int>,
## #   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>
```

```
# with between
filter(flights, between(dep_time, 0, 600))
```

```
## # A tibble: 9,344 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
```

```
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013     1     1      517            515      2.00      830
##  2  2013     1     1      533            529      4.00      850
##  3  2013     1     1      542            540      2.00      923
##  4  2013     1     1      544            545     -1.00     1004
##  5  2013     1     1      554            600     -6.00      812
##  6  2013     1     1      554            558     -4.00      740
##  7  2013     1     1      555            600     -5.00      913
##  8  2013     1     1      557            600     -3.00      709
##  9  2013     1     1      557            600     -3.00      838
## 10  2013     1     1      558            600     -2.00      753
## # ... with 9,334 more rows, and 12 more variables: sched_arr_time <int>,
## #   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>
```

QUESTION 3: How many flights have a missing dep_time? What other variables are missing? What might these rows represent?

ANSWER 3:

flights have a missing dep_time

```
filter(flights, is.na(dep_time))
```

```
## # A tibble: 8,255 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013     1     1       NA           1630        NA       NA
##  2  2013     1     1       NA           1935        NA       NA
##  3  2013     1     1       NA           1500        NA       NA
##  4  2013     1     1       NA            600        NA       NA
##  5  2013     1     2       NA           1540        NA       NA
##  6  2013     1     2       NA           1620        NA       NA
##  7  2013     1     2       NA           1355        NA       NA
##  8  2013     1     2       NA           1420        NA       NA
##  9  2013     1     2       NA           1321        NA       NA
## 10  2013     1     2       NA           1545        NA       NA
## # ... with 8,245 more rows, and 12 more variables: sched_arr_time <int>,
## #   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>
```

The other variables with missing values are arrival time and departure/arrival delay.

It is most likely these are scheduled flights that never flew.

QUESTION 4: 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!)

ANSWER 4:

- *NA ^ 0* - by definition anything to the 0th power is 1.
- *NA | TRUE* - as long as one condition is TRUE, the result is TRUE. By definition, TRUE is TRUE.
- *FALSE & NA* - NA indicates the absence of a value, so the conditional expression ignores it.
- In general, any operation on a missing value becomes a missing value. Hence. NA * 0 is NA. In conditional expressions, missing values are simply ignored.

5.4.1 Exercises

QUESTION 1: Brainstorm as many ways as possible to select dep_time, dep_delay, arr_time, and arr_delay from flights.

ANSWER 1:

```
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      2.00      830     11.0
##  2      533      4.00      850     20.0
##  3      542      2.00      923     33.0
##  4      544     -1.00     1004    -18.0
##  5      554     -6.00      812    -25.0
##  6      554     -4.00      740     12.0
##  7      555     -5.00      913     19.0
##  8      557     -3.00      709    -14.0
##  9      557     -3.00      838    - 8.00
## 10      558     -2.00      753      8.00
## # ... with 336,766 more rows
```

```
select(flights, starts_with("dep"), starts_with("arr"))
```

```
## # A tibble: 336,776 x 4
##    dep_time dep_delay arr_time arr_delay
##       <int>     <dbl>    <int>     <dbl>
##  1      517      2.00      830     11.0
##  2      533      4.00      850     20.0
##  3      542      2.00      923     33.0
##  4      544     -1.00     1004    -18.0
##  5      554     -6.00      812    -25.0
##  6      554     -4.00      740     12.0
##  7      555     -5.00      913     19.0
##  8      557     -3.00      709    -14.0
##  9      557     -3.00      838    - 8.00
## 10      558     -2.00      753      8.00
## # ... with 336,766 more rows
```

```
select(flights, ends_with("delay"))
```

```
## # A tibble: 336,776 x 2
##    dep_delay arr_delay
##        <dbl>     <dbl>
##  1      2.00     11.0
##  2      4.00     20.0
##  3      2.00     33.0
##  4     -1.00    -18.0
##  5     -6.00    -25.0
##  6     -4.00     12.0
##  7     -5.00     19.0
##  8     -3.00    -14.0
##  9     -3.00    - 8.00
## 10     -2.00      8.00
## # ... with 336,766 more rows
```

```
select(flights, contains("delay"))
```

```
## # A tibble: 336,776 x 2
##    dep_delay arr_delay
##        <dbl>     <dbl>
##  1      2.00     11.0
##  2      4.00     20.0
##  3      2.00     33.0
##  4     -1.00    -18.0
##  5     -6.00    -25.0
##  6     -4.00     12.0
##  7     -5.00     19.0
##  8     -3.00    -14.0
##  9     -3.00    - 8.00
## 10     -2.00      8.00
## # ... with 336,766 more rows
```

QUESTION 2: What happens if you include the name of a variable multiple times in a select() call?

ANSWER 2: It is included only a single time in the result data frame even if a variable is included multiple times in a select() call.

QUESTION 3: What does the one_of() function do? Why might it be helpful in conjunction with this vector?

vars <- c("year", "month", "day", "dep_delay", "arr_delay")

ANSWER 3: It selects any variable which matches one of the strings in the vector. Please see the example below.

```
vars <- c("year", "month", "day", "dep_delay", "arr_delay")
select(flights, one_of(vars))
```

```
## # A tibble: 336,776 x 5
##     year month   day dep_delay arr_delay
##    <int> <int> <int>     <dbl>     <dbl>
##  1  2013     1     1      2.00     11.0
##  2  2013     1     1      4.00     20.0
##  3  2013     1     1      2.00     33.0
##  4  2013     1     1     -1.00    -18.0
##  5  2013     1     1     -6.00    -25.0
##  6  2013     1     1     -4.00     12.0
##  7  2013     1     1     -5.00     19.0
##  8  2013     1     1     -3.00    -14.0
##  9  2013     1     1     -3.00    - 8.00
## 10  2013     1     1     -2.00      8.00
## # ... with 336,766 more rows
```

QUESTION 4: Does the result of running the following code surprise you? How do the select helpers deal with case by default? How can you change that default?

```
select(flights, contains("TIME"))
```

```
## # A tibble: 336,776 x 6
##    dep_time sched_dep_time arr_time sched_arr_time air_time
##       <int>          <int>    <int>          <int>    <dbl>
##  1      517            515      830            819    227
##  2      533            529      850            830    227
```

```
##  3      542            540      923            850    160
##  4      544            545     1004           1022    183
##  5      554            600      812            837    116
##  6      554            558      740            728    150
##  7      555            600      913            854    158
##  8      557            600      709            723     53.0
##  9      557            600      838            846    140
## 10      558            600      753            745    138
## # ... with 336,766 more rows, and 1 more variable: time_hour <dttm>
```

ANSWER 4: By default the select helpers ignore case as in above select() call. To adhere to case, we have to use *ignore.case = FALSE* in the helper function. For example:

```
select(flights, contains("TIME", ignore.case = FALSE))
```

```
## # A tibble: 336,776 x 0
```

END OF HW3 ASSIGNMENT