# COMPSCIX 415.2 Homework 3

*Vishnu Vardhan*

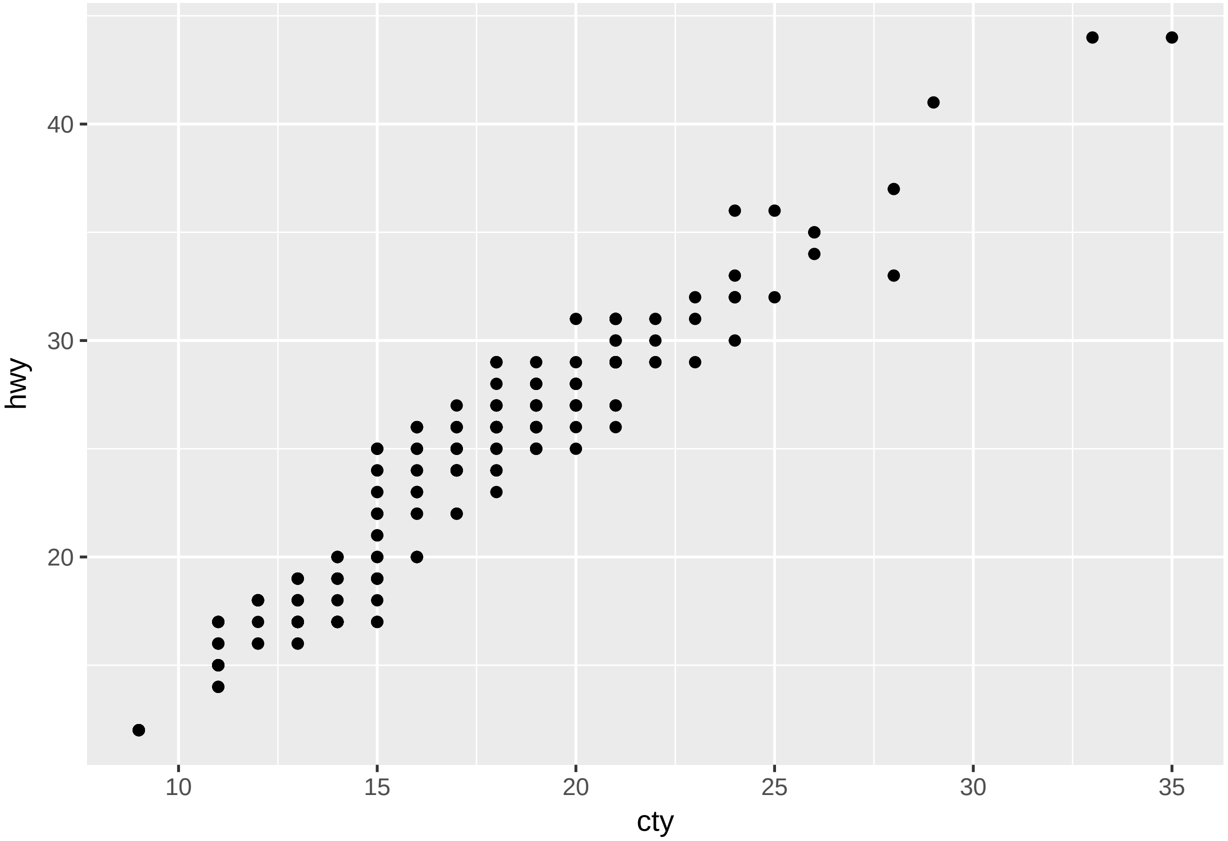
*2/18/2018*

## Section 3.8.1

### Q 3.8.1.1

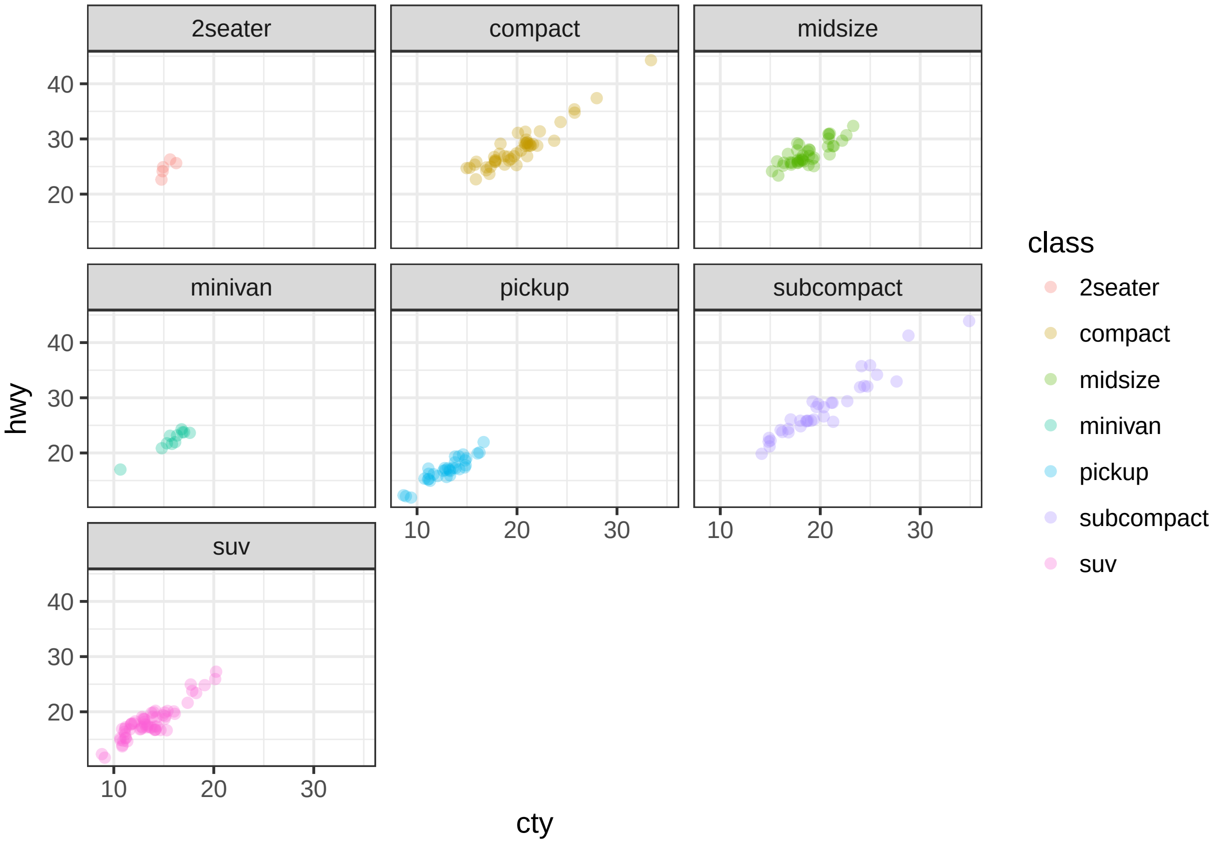
What is the problem with this plot? How could you improve it?

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



**Answer:**

```
ggplot(data = mpg, mapping = aes(x = cty, y = hwy)) +
  geom_jitter(mapping = aes(color = class ), alpha = 0.3) +
  facet_wrap(~ class) +
  theme_bw()
```



**Q 3.8.1.2**

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

**Answer:**

Width & height

As outlined in the help:

Amount of vertical and horizontal jitter. The jitter is added in both positive and negative directions, so the total spread is twice the value specified here. If omitted, defaults to 40% of the resolution of the data: this means the jitter values will occupy 80% of the implied bins. Categorical data is aligned on the integers, so a width or height of 0.5 will spread the data so it's not possible to see the distinction between the categories.

**Q 3.8.1.3**

Compare and contrast geom_jitter() with geom_count().

**Answer:**

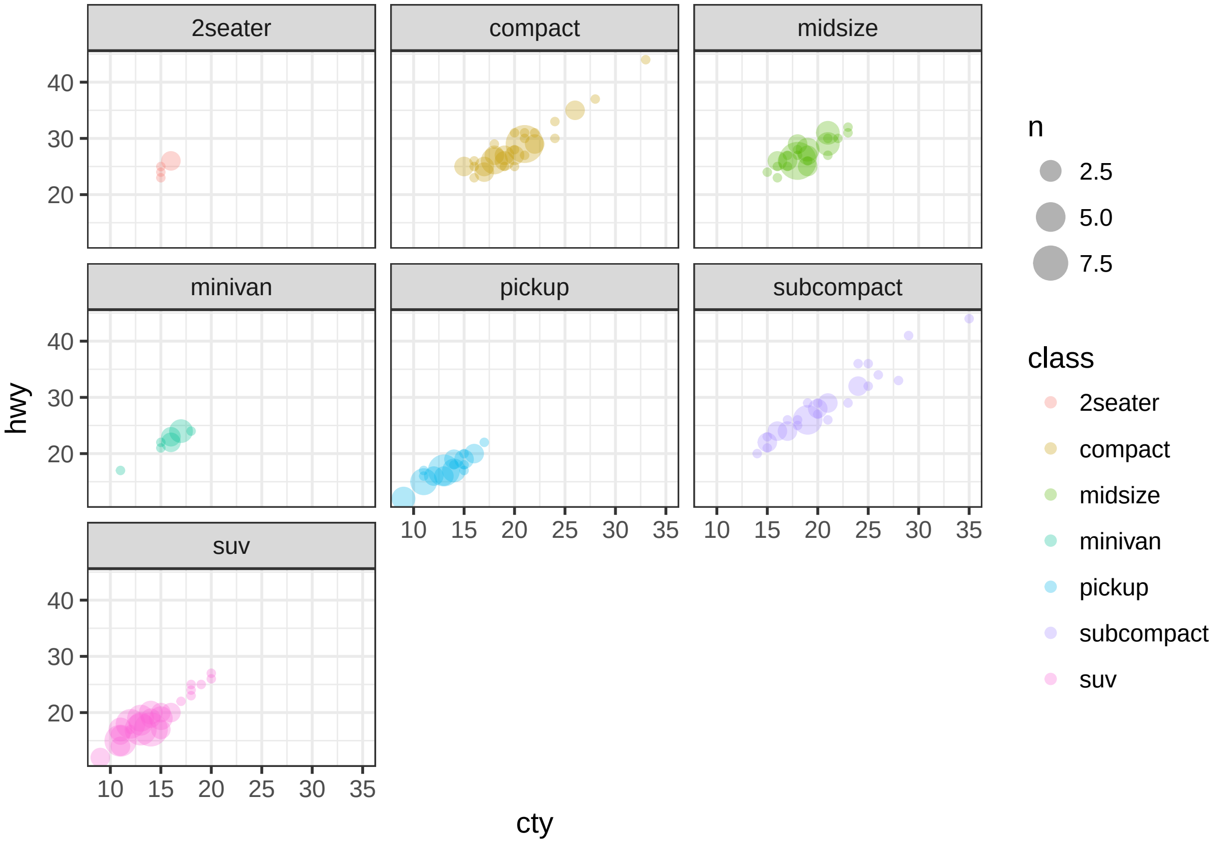
Jitter and count are two ways in which we can try to capture overlapping variables. Jitter is good when many datapoints are close to each other, and a small displacement allows a good visual conclusion to be reached.

geom_count on the other hand is more precise, but is harder to interpret if many of the points are too close together. geom_count tries to be more accurate, but communicates less when points are close to each other.

To summarize - geom_jitter: Good when many points overlap, but the distannce between these overlap points is also less. Is less accurate, but better at trends

geom_count: good when many points overlap, but also the distance between the overlap points is more. Is more accurate

```
ggplot(data = mpg, mapping = aes(x = cty, y = hwy)) +
  geom_count(mapping = aes(color = class ), alpha = 0.3) +
  facet_wrap(~ class) +
  theme_bw()
```



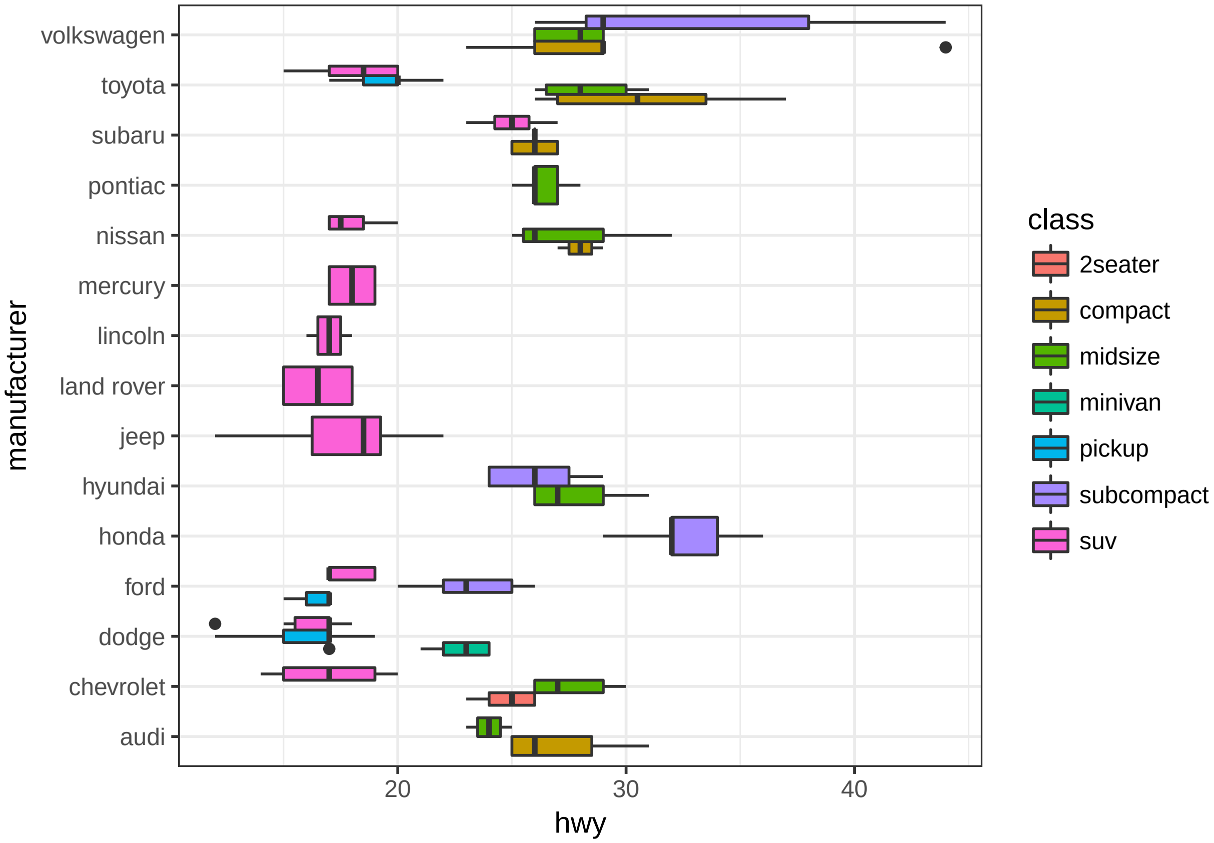
**Q 3.8.1.4**

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

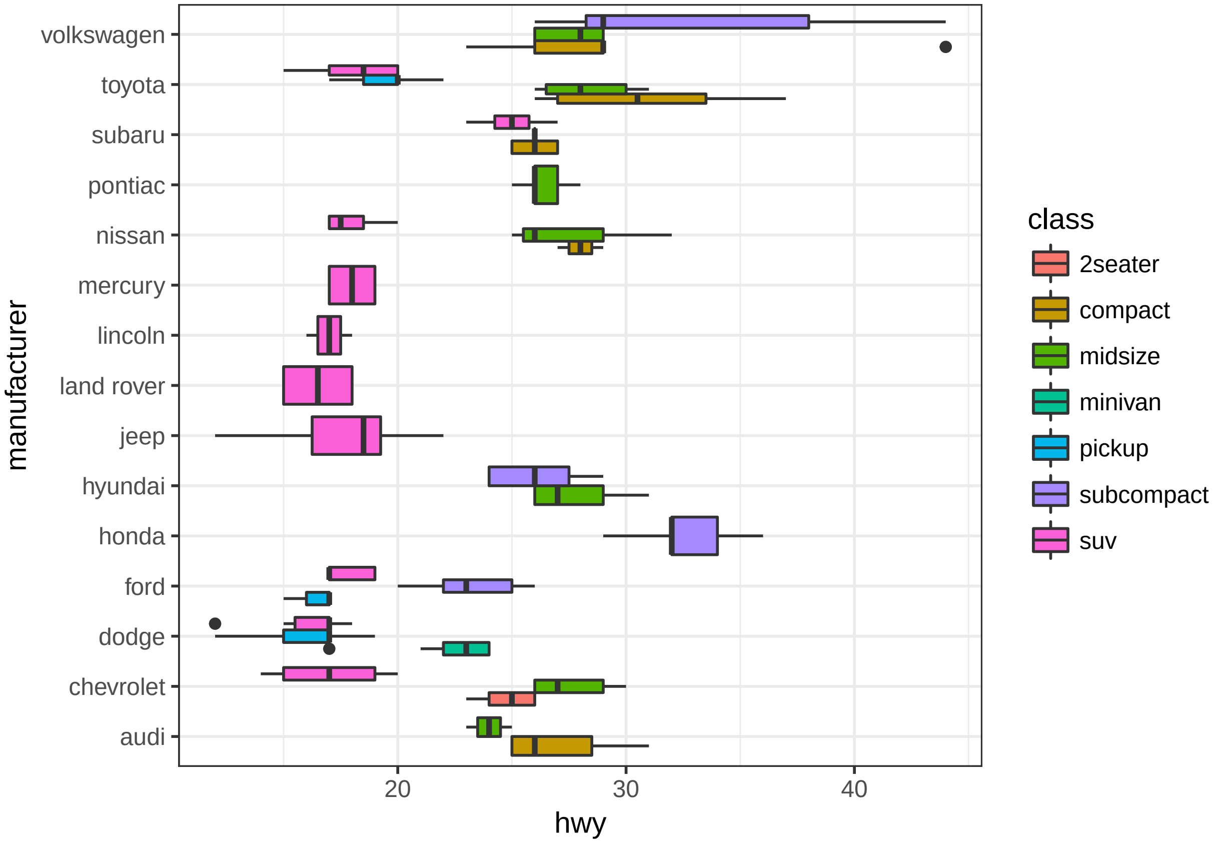
**Answer:**

The default position for a boxplot is "dodge". As you can see, the below two graphs are identical.

```
ggplot(data = mpg, mapping = aes( x = manufacturer,y = hwy))  +
  geom_boxplot(mapping = aes(fill = class)) +
  coord_flip() +
  theme_bw()
```

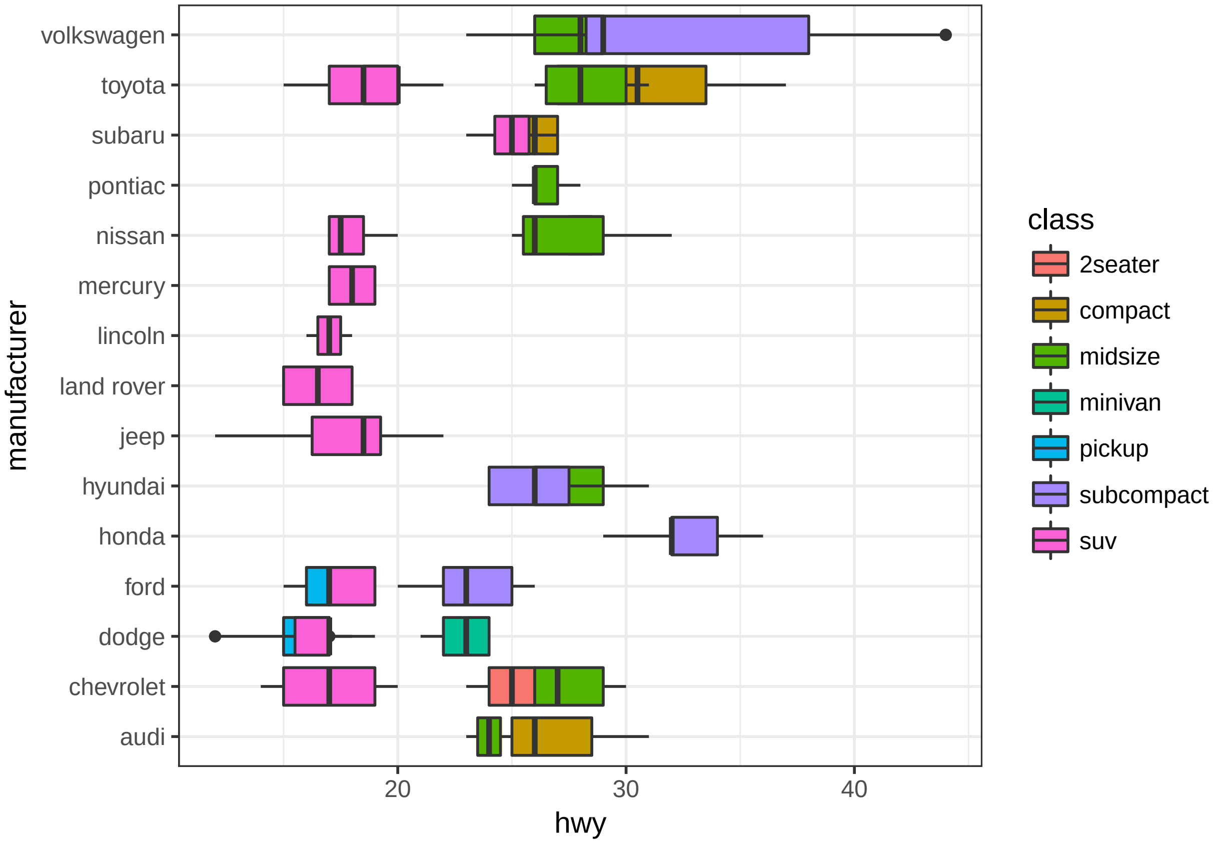


```
ggplot(data = mpg, mapping = aes( x = manufacturer,y = hwy))  +
  geom_boxplot(mapping = aes(fill = class), position="dodge") +
  coord_flip() +
  theme_bw()
```



Changing position to identity results in the below.

```r
ggplot(data = mpg, mapping = aes( x = manufacturer,y = hwy)) +
  geom_boxplot(mapping = aes(fill = class), position="identity") +
  coord_flip() +
  theme_bw()
```



## Section 3.9.1

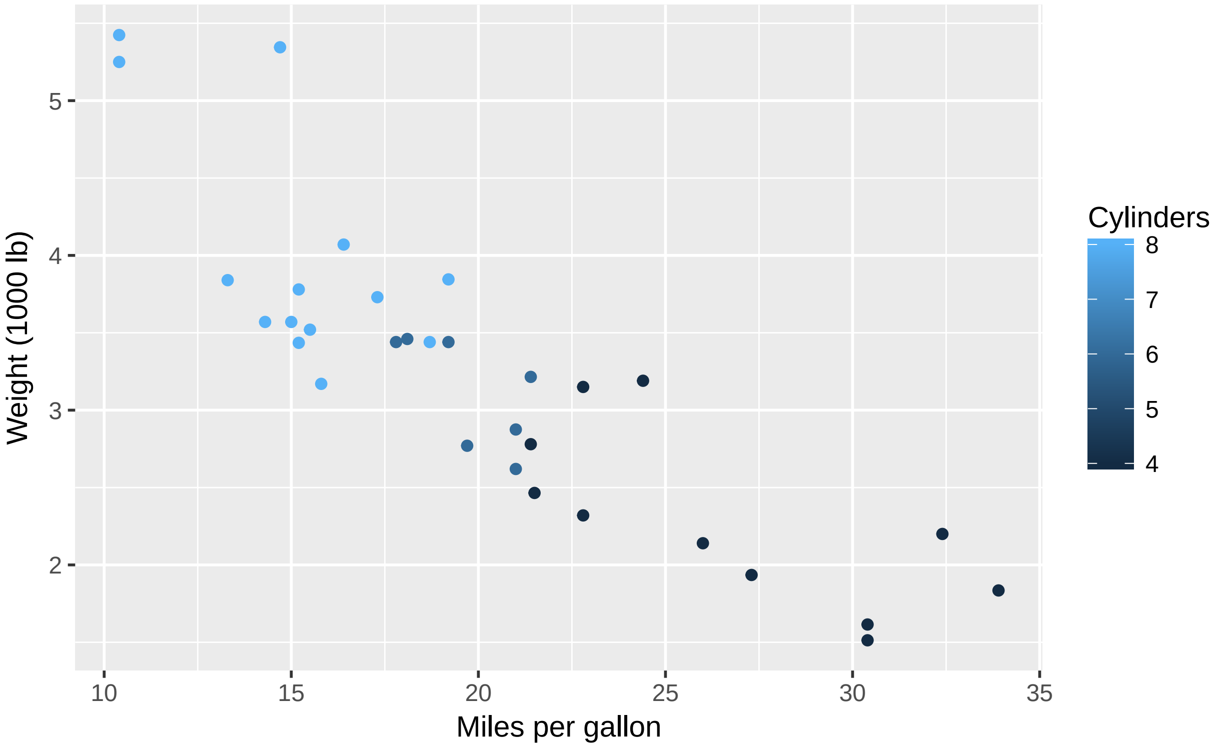
### Q 3.9.1.2

What does labs() do? Read the documentation.

**Answer:**

labs() allows us to control the title, sub-title, labels for the x & y axis, footnotes via caption, and the label for the legend. Here are is a modified form of the examples from the documentation.

```
ggplot(mtcars, aes(x=mpg,y=wt, color = cyl)) +
  geom_point() +
  labs(x = "Miles per gallon", y = "Weight (1000 lb)", color = "Cylinders") +
  labs(caption = "\n From the 1974 Motor Trend US magazine for 32 automobiles \n (1973-1974 Models)")
```

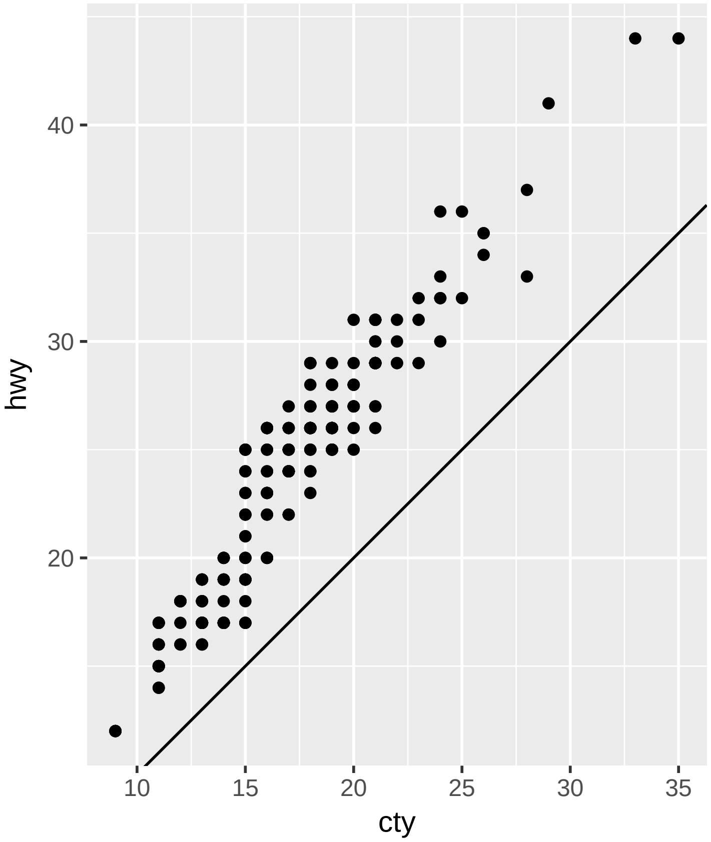


From the 1974 Motor Trend US magazine for 32 automobiles
(1973−1974 Models)

### Q 3.9.1.4

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?

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



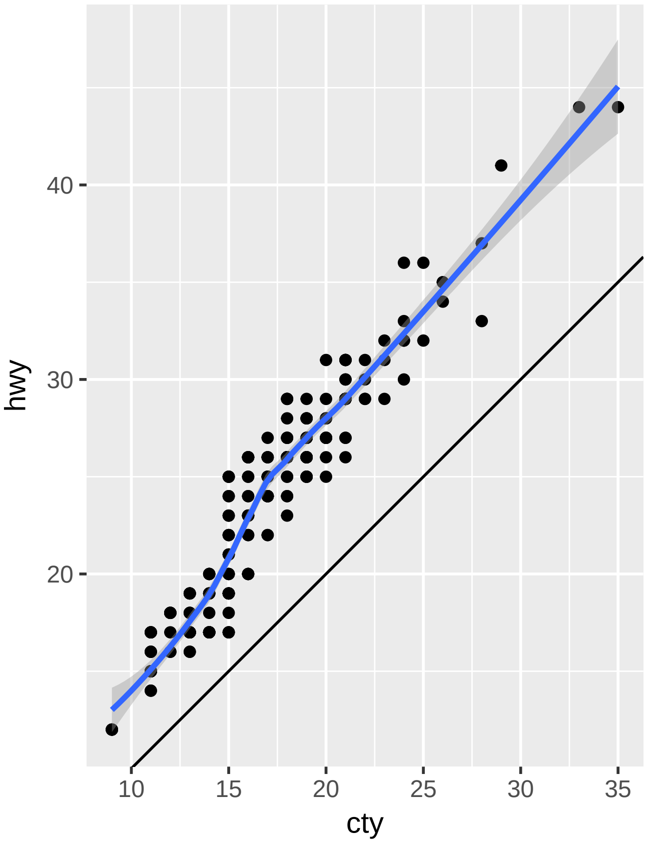
**Answer:**

As illustrated in the graph below, as city miles increase, higy miles increase propotionally, though highway miles increase faster between 15 & 20 city miles.

coord_fixed is important so that we are comparing similar things, and the slope shows the real world slope. geom_abline, plots a reference line that makes it easier to compare the data with a referece.

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

```
## `geom_smooth()` using method = 'loess'
```



## Section 4.4

### Q 4.4.1

Why does this code not work?

my_variable <- 10 my_varıable

**Answer:**

The variable is defined with ‘i’, but in the usage, the ‘i’ is replaced with the number ‘l’ (or something similar).

### Q 4.4.2

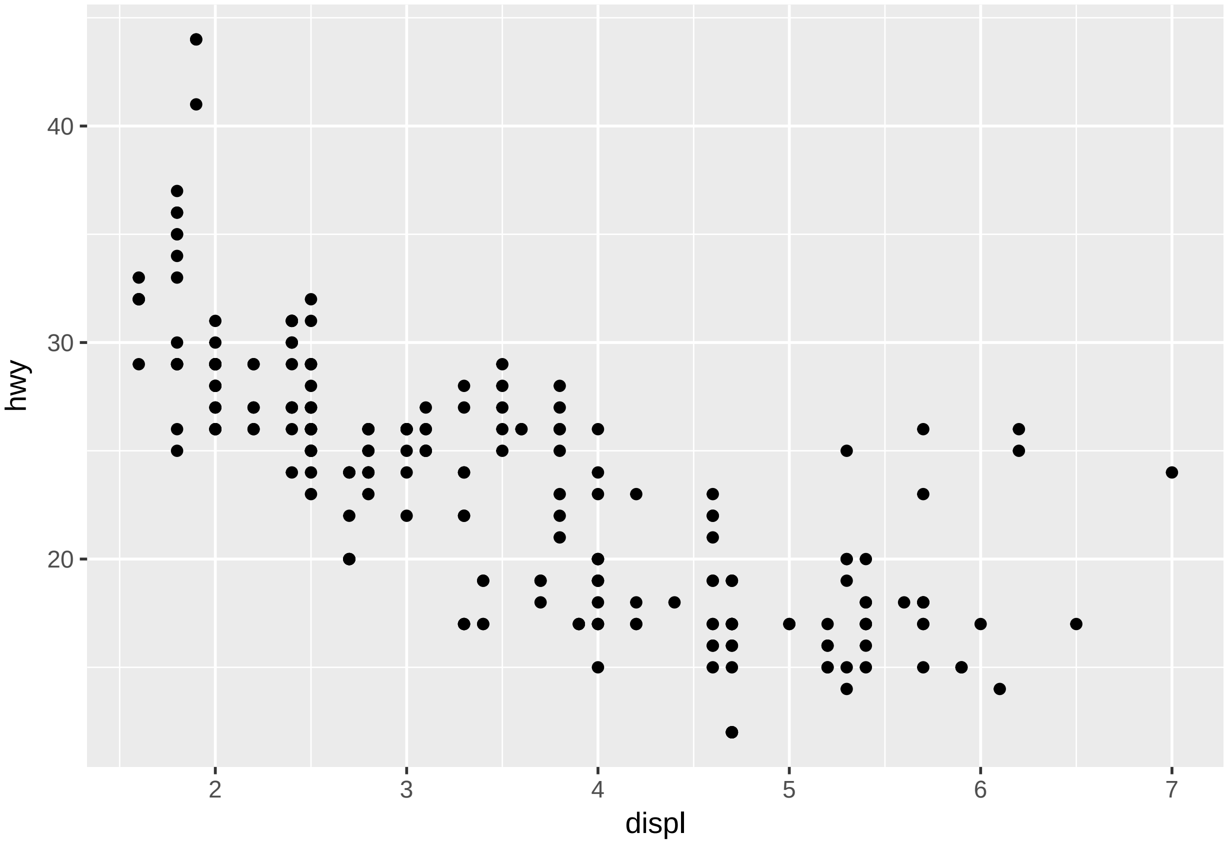
Tweak each of the following R commands so that they run correctly:

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

fliter(mpg, cyl = 8) filter(diamond, carat > 3)

**Answer:**

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



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

```
## # A tibble: 70 x 11
##    manufacturer              model displ  year   cyl      trans  drv
##           <chr>              <chr> <dbl> <int> <int>      <chr> <chr>
##  1         audi         a6 quattro   4.2  2008     8   auto(s6)     4
##  2    chevrolet c1500 suburban 2wd   5.3  2008     8   auto(l4)     r
##  3    chevrolet c1500 suburban 2wd   5.3  2008     8   auto(l4)     r
##  4    chevrolet c1500 suburban 2wd   5.3  2008     8   auto(l4)     r
##  5    chevrolet c1500 suburban 2wd   5.7  1999     8   auto(l4)     r
##  6    chevrolet c1500 suburban 2wd   6.0  2008     8   auto(l4)     r
##  7    chevrolet           corvette   5.7  1999     8 manual(m6)     r
##  8    chevrolet           corvette   5.7  1999     8   auto(l4)     r
##  9    chevrolet           corvette   6.2  2008     8 manual(m6)     r
## 10    chevrolet           corvette   6.2  2008     8   auto(s6)     r
## # ... with 60 more rows, and 4 more variables: cty <int>, hwy <int>,
## #   fl <chr>, 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  8040  9.10  8.97  5.67
##  2  3.11    Fair     J      I1  65.9    57  9823  9.15  9.02  5.98
##  3  3.01 Premium     F      I1  62.2    56  9925  9.24  9.13  5.73
##  4  3.05 Premium     E      I1  60.9    58 10453  9.26  9.25  5.66
##  5  3.02    Fair     I      I1  65.2    56 10577  9.11  9.02  5.91
##  6  3.01     Fair     H      I1  56.1     62 10761  9.54  9.38  5.31
##  7  3.65     Fair     H      I1  67.1     53 11668  9.53  9.48  6.38
##  8  3.24 Premium     H      I1  62.1     58 12300  9.44  9.40  5.85
##  9  3.22    Ideal     I      I1  62.6     55 12545  9.49  9.42  5.92
## 10  3.50    Ideal     H      I1  62.8     57 12587  9.65  9.59  6.03
## # ... with 22 more rows
```

## Section 5.2.4

### Q 5.2.4.1

Find all flights that

Had an arrival delay of two or more hours

Flew to Houston (IAH or HOU)

Were operated by United, American, or Delta

Departed in summer (July, August, and September)

Arrived more than two hours late, but didn't leave late

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

Departed between midnight and 6am (inclusive)

**Answer:**

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
##    <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     1912
##  8  2013     1     1     1558           1359       119     1718
##  9  2013     1     1     1732           1630        62     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>
```

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      830
## 2  2013     1     1      533            529         4      850
## 3  2013     1     1      623            627        -4      933
## 4  2013     1     1      728            732        -4     1041
## 5  2013     1     1      739            739         0     1104
## 6  2013     1     1      908            908         0     1228
## 7  2013     1     1     1028           1026         2     1350
## 8  2013     1     1     1044           1045        -1     1352
## 9  2013     1     1     1114            900       134     1447
## 10  2013     1     1     1205           1200         5     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>
```

Were 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      830
## 2  2013     1     1      533            529         4      850
## 3  2013     1     1      542            540         2      923
## 4  2013     1     1      554            600        -6      812
## 5  2013     1     1      554            558        -4      740
## 6  2013     1     1      558            600        -2      753
## 7  2013     1     1      558            600        -2      924
## 8  2013     1     1      558            600        -2      923
## 9  2013     1     1      559            600        -1      941
## 10  2013     1     1      559            600        -1      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>
```

Departed in summer (July, August, and September)

```
filter(flights,  (month == 7 | month == 8 | 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      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>
```

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     1754
##  2  2013    10     7     1350           1350         0     1736
##  3  2013    10     7     1357           1359        -2     1858
##  4  2013    10    16      657            700        -3     1258
##  5  2013    11     1      658            700        -2     1329
##  6  2013     3    18     1844           1847        -3       39
##  7  2013     4    17     1635           1640        -5     2049
##  8  2013     4    18      558            600        -2     1149
##  9  2013     4    18      655            700        -5     1213
## 10  2013     5    22     1827           1830        -3     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>
```

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

```
filter(flights,  (arr_delay < (dep_delay-30)) & (dep_delay > 60))
```

```
## # A tibble: 1,819 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     2205           1720       285       46
##  2  2013     1     1     2326           2130       116      131
##  3  2013     1     3     1503           1221       162     1803
##  4  2013     1     3     1839           1700        99     2056
##  5  2013     1     3     1850           1745        65     2148
##  6  2013     1     3     1941           1759       102     2246
##  7  2013     1     3     1950           1845        65     2228
##  8  2013     1     3     2257           2000       177       45
##  9  2013     1     4     1917           1700       137     2135
## 10  2013     1     4     2010           1745       145     2257
## # ... with 1,809 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>
```

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      830
##  2  2013     1     1      533            529         4      850
##  3  2013     1     1      542            540         2      923
##  4  2013     1     1      544            545        -1     1004
##  5  2013     1     1      554            600        -6      812
##  6  2013     1     1      554            558        -4      740
##  7  2013     1     1      555            600        -5      913
##  8  2013     1     1      557            600        -3      709
##  9  2013     1     1      557            600        -3      838
## 10  2013     1     1      558            600        -2      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>
```

### Q 5.2.4.2

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:**

between() is a shortcut for x >= left & x <= right.

```
filter(flights,  between(month,7,9))
```

```
## # 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      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>
```

```
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      830
##  2  2013     1     1      533            529         4      850
##  3  2013     1     1      542            540         2      923
##  4  2013     1     1      544            545        -1     1004
##  5  2013     1     1      554            600        -6      812
##  6  2013     1     1      554            558        -4      740
##  7  2013     1     1      555            600        -5      913
##  8  2013     1     1      557            600        -3      709
##  9  2013     1     1      557            600        -3      838
## 10  2013     1     1      558            600        -2      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>
```

### Q 5.2.4.3

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

**Answer:**

8,255 flights have dep_time missing. Other variables that are mising are: arr_time, dep_delay, arr_delay, tailnum, air_time.

tailnum - likely represents private flights, that don't have a tail num. air_time would be missing, if either dep_time or arr_time is missing, since it represents the amount of time in the air.

```
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>
```

```
colSums(is.na(flights))
```

```
##           year          month            day       dep_time sched_dep_time
##              0              0              0           8255              0
##      dep_delay       arr_time sched_arr_time      arr_delay        carrier
##           8255           8713              0           9430              0
##         flight        tailnum         origin           dest       air_time
##              0           2512              0              0           9430
##       distance           hour         minute      time_hour
##              0              0              0              0
```

### Q 5.2.4.4

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:**

NA ˆ 0 is alwyas 1. So it makes sense to be valid. NA | TRUE will always evaluate to true, irrespective of the NA value. So again makes sense FALSE & NA will always evaluate to false, irrespective of the NA value. Makes sense to not be missing. In general, if the answer is predictable - any evaluation with NA is not missing.

## Section 5.4.1

### Q 5.4.1.1

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

**Answer:**

```
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
```

```
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      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
```

### Q 5.4.1.2

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

**Answer:**

select will only consider the first occurrence, and ignore other occurences. This is how the everything() call can be used to reorder the columns.

```
select(flights, tailnum, tailnum)
```

```
## # A tibble: 336,776 x 1
##    tailnum
##      <chr>
##  1  N14228
##  2  N24211
##  3  N619AA
##  4  N804JB
##  5  N668DN
##  6  N39463
##  7  N516JB
##  8  N829AS
##  9  N593JB
## 10  N3ALAA
## # ... with 336,766 more rows
```

```
select(flights, tailnum, everything())
```

```
## # A tibble: 336,776 x 19
##    tailnum  year month   day dep_time sched_dep_time dep_delay arr_time
##      <chr> <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  N14228  2013     1     1      517            515         2      830
##  2  N24211  2013     1     1      533            529         4      850
##  3  N619AA  2013     1     1      542            540         2      923
##  4  N804JB  2013     1     1      544            545        -1     1004
##  5  N668DN  2013     1     1      554            600        -6      812
##  6  N39463  2013     1     1      554            558        -4      740
##  7  N516JB  2013     1     1      555            600        -5      913
##  8  N829AS  2013     1     1      557            600        -3      709
##  9  N593JB  2013     1     1      557            600        -3      838
## 10  N3ALAA  2013     1     1      558            600        -2      753
## # ... with 336,766 more rows, and 11 more variables: sched_arr_time <int>,
## #   arr_delay <dbl>, carrier <chr>, flight <int>, origin <chr>,
## #   dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>,
## #   time_hour <dttm>
```

### Q 5.4.1.3

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:**

one_of has the signature one_of(. . . , vars=current_vars()). The first parameter (. . . ) signifies one or more character arrays. current_vars(), unless specified, evaluates to all the columns within the current select call.

one_of looks at the character arrays, combines all the character arrays, and removes duplicates, and displays columns that are present in the 'vars' column list (by defaul all the columns). Errors (i.e garbage columns) are printed out as warnings.

The key use seems to be described by this stackoverflow thread: https://stackoverflow.com/questions/45865892/why-is-one-of-called-that

select(flights,garbage) will throw an error select(flights, one_of(c("garbage"))) will only warn.

This allows for UI driven development, without the developer having to check for the existence of the column, before executing a select.

### Q 5.4.1.4

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"))

**Answer:**

Yes, it is quite surprising that 'select' is not case sensitive, though the help is clear that case is an input parameter, and the default is to ignore case. Executing this with 'ignore.case = FALSE', results in the expected output.

```
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
##  9      557            600      838            846      140
## 10      558            600      753            745      138
## # ... with 336,766 more rows, and 1 more variables: time_hour <dttm>
```

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

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

**Assignment complete**