COMPSCIX 415.2 Homework 3

Rajat Jain

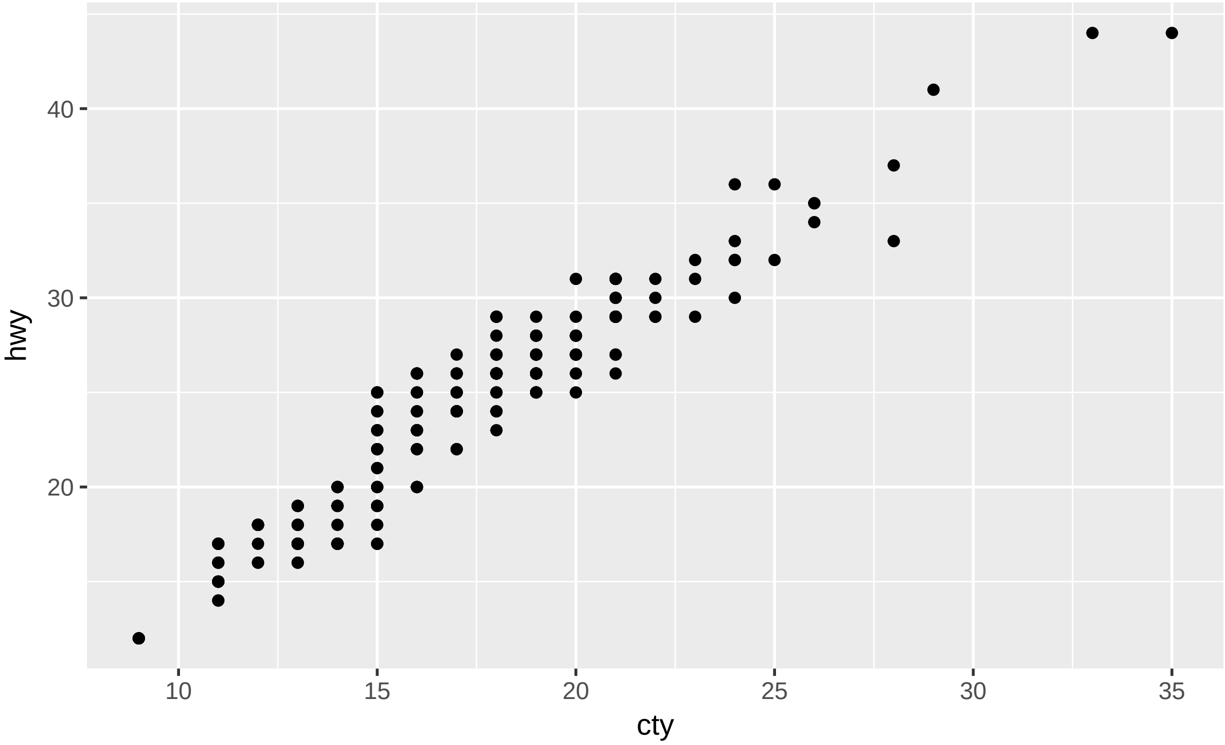
June 25th, 2018

Contents

Section 3.8.1 Exercises

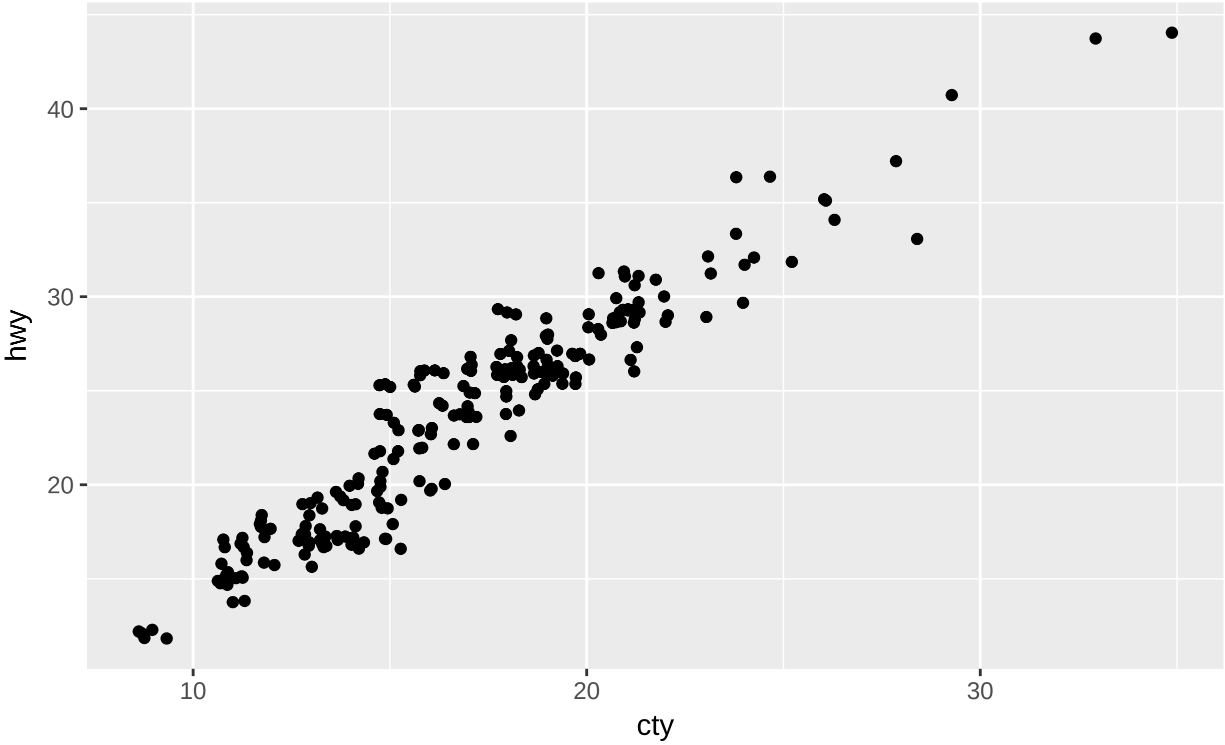
1. **What is the problem with this plot? How could you improve it?**

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



This plot has many overlapping point which do not show the areas of concentration properly. It can be inproved by adding jitter to the position adjustment.

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



2. **What parameters to `geom_jitter()` control the amount of jittering?**

The parameters to `geom_jitter()` which control the amount of jittering are `width` and `height`.

3. **Compare and contrast `geom_jitter()` with `geom_count()`.**

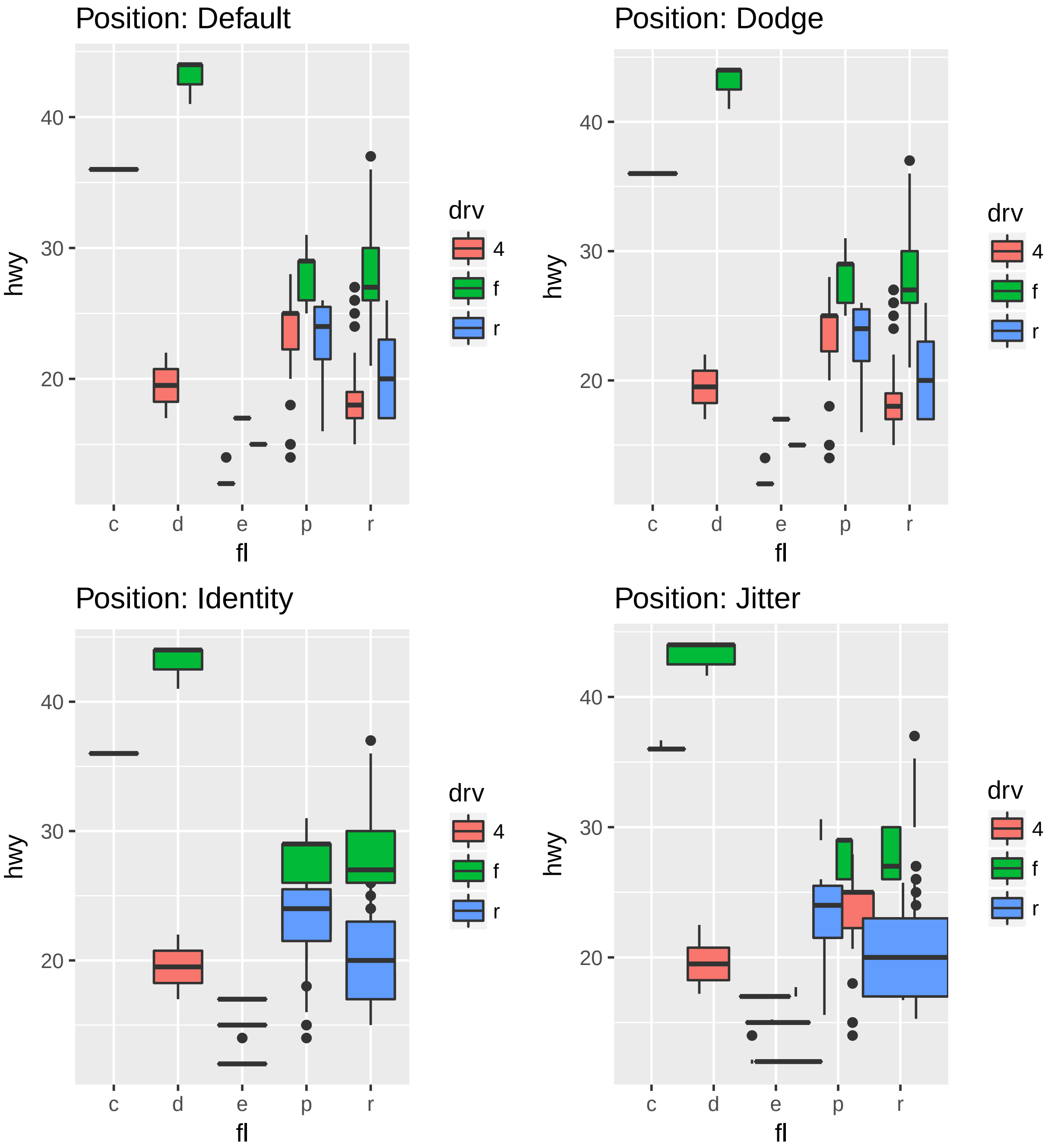
`geom_jitter` randomly moves the overlapping points slightly to avoid overlapping whereas, `geom_count` counts the overlapping points at a given point and maps them to the size of a single point. The determinism of `geom_count` makes it useful in discrete situations, but it does not work when the points are not exactly overlapping but are very close.

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

The default position adjustment for `geom_boxplot()` is `position_dodge`. Here is a demonstration:

```
my_plot <- ggplot(data = mpg, mapping = aes(x = fl, y = hwy, fill = drv))

# Plot with default position adjustment.
dflt <- my_plot + ggtitle("Position: Default") +
  geom_boxplot()

# Plot with dodge position adjustment.
ddge <- my_plot + ggtitle("Position: Dodge") +
  geom_boxplot(position = "dodge")
```

```
# Plot with identity position adjustment.
idnt <- my_plot + ggtitle("Position: Identity") +
  geom_boxplot(position = "identity")

# Plot with jitter position adjustment.
jter <- my_plot + ggtitle("Position: Jitter") +
  geom_boxplot(position = "jitter")

grid.arrange(dflt, ddge, idnt, jter, nrow = 2, ncol = 2)
```



Position: Default

Position: Dodge

Position: Identity

Position: Jitter

In the above plots position adjustment `Dodge` generated the plot looking exactly as with default position adjustment.

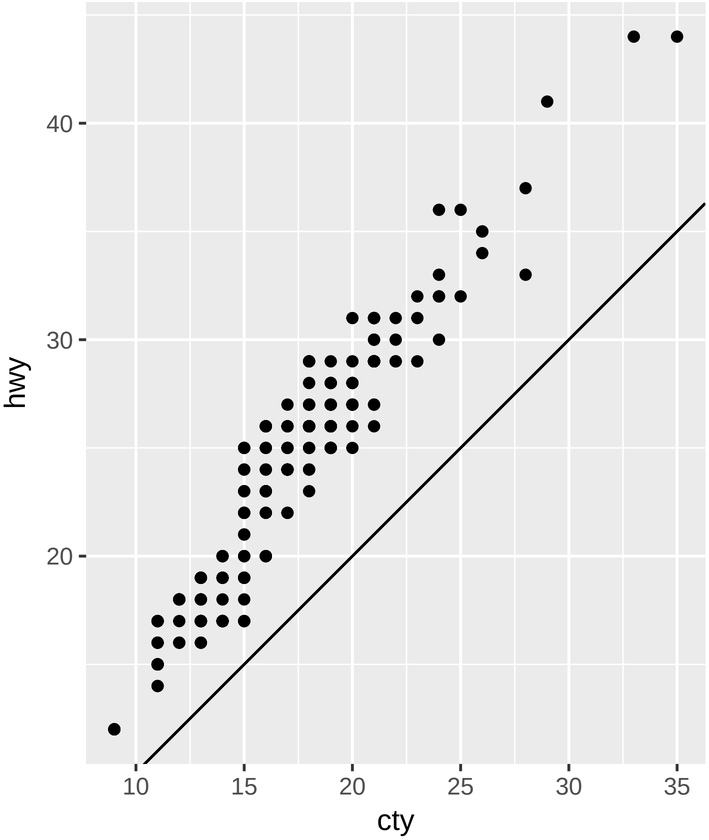
Section 3.9.1 Exercises

2. **What does `labs()` do? Read the documentation.**

`labs()` function from `ggplot2` package is used to modify axis, legend, and plot labels.

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()
```



This plot tells us that the city and highway mpg are positively correlated, meaning if a vehicle has a higher city mpg, it is also expected to have a higher highway mpg.

`coord_fixed()` is important because both city and highway mpg have the same unit and hence one unit of each should be represented by the same length on the axes for the best visualization. `coord_fixed()` does exactly that for us.

`geom_abline()` creates a reference line shown on the plot. Since it is called without `intercept` and `slope` arguments, it generates a default reference line passing through the origin and with a slope of 1 (45°). So this line represents all the point on the plot where city mpg would be equal to highway mpg. From the plot we can also observe that all the plotted points are above this line, which means that the highway mpg is always higer than the city mpg. If we observe closely, we can also see that the scatter plot is more or less parallel to the reference line. This tells us that the highway mpg is higher than the city mpg for a constant additive offset.

Section 4.4 Exercises

1. **Why does this code not work?**

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

Look carefully! (This may seem like an exercise in pointlessness, but training your brain to notice even the tiniest difference will pay off when programming.)

This code does not work because the name of the variable while printing it is mis-spelled.

```
my_variable <- 10
my_variable
```

```
## [1] 10
```

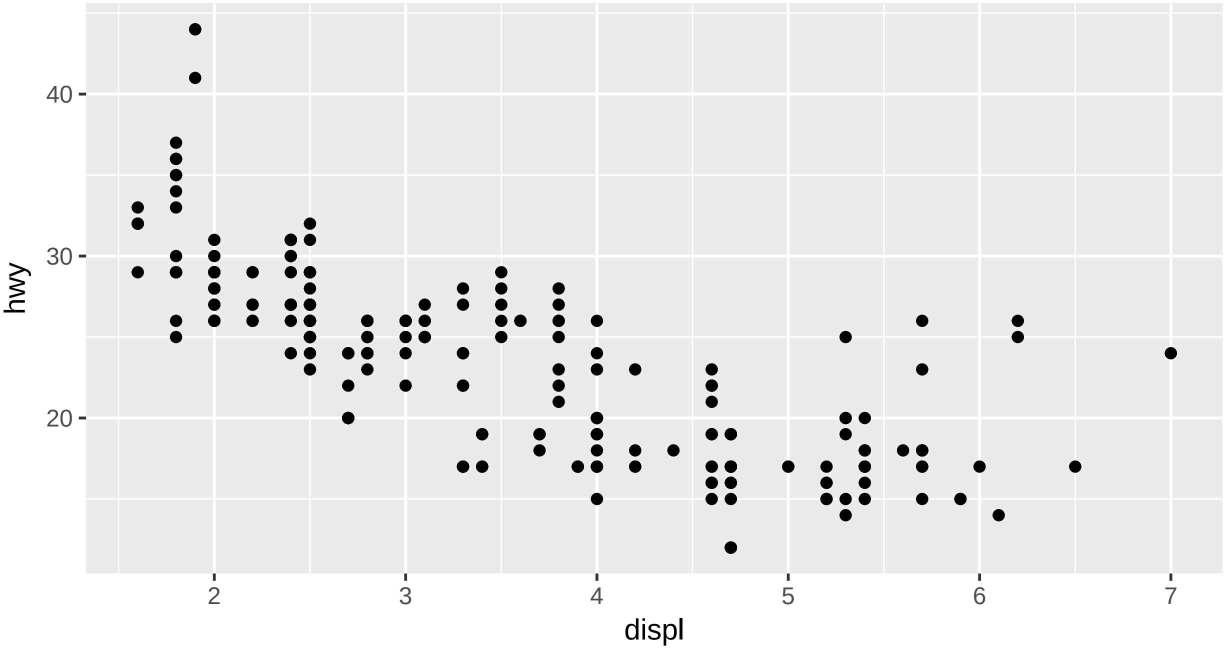
The above code is its fixed version which works.

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)
```

Here is the fixed and working code:

```
# Package tidyverse is pre-loaded in this document.
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy))
```



```r
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>
```

```r
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 Exercises

1. **Find all flights that**

    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
##    <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>
```

2. Flew to Houston (IAH or HOU)

```
filter(flights, dest %in% c("IAH", "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>
```

3. Were operated by United, American, or Delta

```
filter(flights, carrier %in% c("UA", "AA", "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>
```

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

```
filter(flights, month %in% 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>
```

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.     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>
```

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

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

```
## # A tibble: 1,844 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     2015           1915       60.     2135
##  9  2013     1     3     2257           2000      177.       45
## 10  2013     1     4     1917           1700      137.     2135
## # ... with 1,834 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)

```r
filter(flights, dep_time <= 600 | dep_time == 2400)
```

```
## # A tibble: 9,373 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 9,363 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. **How many flights have a missing `dep_time`? What other variables are missing? What might these rows represent?**

```r
summary(flights)
```

```
##       year          month             day           dep_time
##  Min.   :2013   Min.   : 1.000   Min.   : 1.00   Min.   :   1
##  1st Qu.:2013   1st Qu.: 4.000   1st Qu.: 8.00   1st Qu.: 907
##  Median :2013   Median : 7.000   Median :16.00   Median :1401
##  Mean   :2013   Mean   : 6.549   Mean   :15.71   Mean   :1349
##  3rd Qu.:2013   3rd Qu.:10.000   3rd Qu.:23.00   3rd Qu.:1744
##  Max.   :2013   Max.   :12.000   Max.   :31.00   Max.   :2400
##                                                  NA's   :8255
##  sched_dep_time   dep_delay          arr_time    sched_arr_time
##  Min.   : 106   Min.   : -43.00   Min.   :   1   Min.   :   1
##  1st Qu.: 906   1st Qu.:  -5.00   1st Qu.:1104   1st Qu.:1124
##  Median :1359   Median :  -2.00   Median :1535   Median :1556
##  Mean   :1344   Mean   :  12.64   Mean   :1502   Mean   :1536
##  3rd Qu.:1729   3rd Qu.:  11.00   3rd Qu.:1940   3rd Qu.:1945
##  Max.   :2359   Max.   :1301.00   Max.   :2400   Max.   :2359
##                 NA's   :8255      NA's   :8713
##    arr_delay          carrier              flight       tailnum
##  Min.   : -86.000   Length:336776      Min.   :   1   Length:336776
##  1st Qu.: -17.000   Class :character   1st Qu.: 553   Class :character
##  Median :  -5.000   Mode  :character   Median :1496   Mode  :character
##  Mean   :   6.895                      Mean   :1972
##  3rd Qu.:  14.000                      3rd Qu.:3465
##  Max.   :1272.000                      Max.   :8500
##  NA's   :9430
##     origin              dest              air_time        distance
##  Length:336776      Length:336776      Min.   : 20.0   Min.   :  17
##  Class :character   Class :character   1st Qu.: 82.0   1st Qu.: 502
##  Mode  :character   Mode  :character   Median :129.0   Median : 872
##                                        Mean   :150.7   Mean   :1040
##                                        3rd Qu.:192.0   3rd Qu.:1389
##                                        Max.   :695.0   Max.   :4983
##                                        NA's   :9430
##       hour           minute        time_hour
##  Min.   : 1.00   Min.   : 0.00   Min.   :2013-01-01 05:00:00
##  1st Qu.: 9.00   1st Qu.: 8.00   1st Qu.:2013-04-04 13:00:00
##  Median :13.00   Median :29.00   Median :2013-07-03 10:00:00
##  Mean   :13.18   Mean   :26.23   Mean   :2013-07-03 05:02:36
##  3rd Qu.:17.00   3rd Qu.:44.00   3rd Qu.:2013-10-01 07:00:00
##  Max.   :23.00   Max.   :59.00   Max.   :2013-12-31 23:00:00
##
```

8255 flights have missing `dep_time`. Other variables with missing values are: `dep_delay`, `arr_time`, `arr_delay` & `air_time`.

Since only those variables seem to have missing values which are related to actual flight instances and not just the scheduled details, they may represent the cancelled flights. Alternatively, they could just be errors in data entry while recording those values at flight departure and/or arrival. Or a mix of both.

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!)**

```
NA ^ 0
```

```
## [1] 1
```

```
NA | TRUE
```

```
## [1] TRUE
```

```
FALSE & NA
```

```
## [1] FALSE
```

```
NA * 0
```

```
## [1] NA
```

`NA ^ 0` is not missing because anything to the power of 0 is 1 irrespective of what the value is. However, this is not true for `Inf ^ 0`. It is still indeterminate form. So this seems to be an incorrect evaluation by R.

`NA | TRUE` is not missing because the `|` expression evaluates to `TRUE` if either parts of the expression evaluate to `TRUE` and since `TRUE` is always `TRUE` it doesn't matter what the other part is.

Similarly, `FALSE & NA` is not missing because `&` expression evaluates to `FALSE` if either parts of the expression evaluates to `FALSE` irrespective of the value of the other part.

`NA * 0` evaluates to `NA` because NA could take any value including `Inf` and `Inf * 0` is indeterminate form.

Section 5.4.1 Exercises

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

```
# Define column names vector.
col_names <- c("dep_time",  "dep_delay", "arr_time", "arr_delay")


#1. Using [] with column indexes.
flights[, c(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
```

```
#2. Using [] with column names vector.
flights[, col_names]
```

```
## # 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
```

```
#3. Using select with column indexes.
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
```

```
#4. Using select with column names vector.
select(flights, col_names)
```

```
## # 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
```

```
#5. Using select with column names mentioned as ... arguments.
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
```

```
#6. Using select with pipe operator and column indexes.
flights %>% select(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
```

```
#7. Using select with pipe operator and column names vector.
flights %>% select(col_names)
```

```
## # 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
```

```
#8. Using select with pipe operator and column names mentioned as ... arguments
flights %>% select(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
```

```
#9. Using starts_with() select helper.
flights %>% select(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
```

```
#10. Using one_of() select helper.
flights %>% select(one_of(col_names))
```

```
## # 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
```

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")
```

`one_of()` is a Select Helper function from `dplyr` package. It allows for guessing or subset-matching. (Courtesy: StackOverflow). It returns all the columns with names which match the vector provided to it. For example, in conjunction with the vector above, it could be used with `select()` as below:

```
flights %>% select(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.       11.
##  2  2013     1     1        4.       20.
##  3  2013     1     1        2.       33.
##  4  2013     1     1       -1.      -18.
##  5  2013     1     1       -6.      -25.
##  6  2013     1     1       -4.       12.
##  7  2013     1     1       -5.       19.
##  8  2013     1     1       -3.      -14.
##  9  2013     1     1       -3.       -8.
## 10  2013     1     1       -2.        8.
## # ... with 336,766 more rows
```