MA615 Assignment 2

Megha Pandit

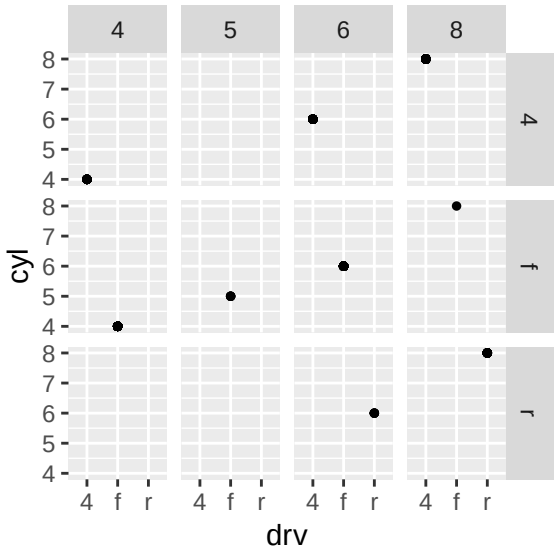
September 22, 2018

R for Data Science

3.5.1 Exercises

2. What do the empty cells in plot with facet_grid(drv ~ cyl) mean? How do they relate to this plot?

```
library(ggplot2)
data(mpg)
ggplot(data = mpg) +
  geom_point(mapping = aes(x = drv, y = cyl), size = 1)+
  facet_grid(drv ~ cyl)
```

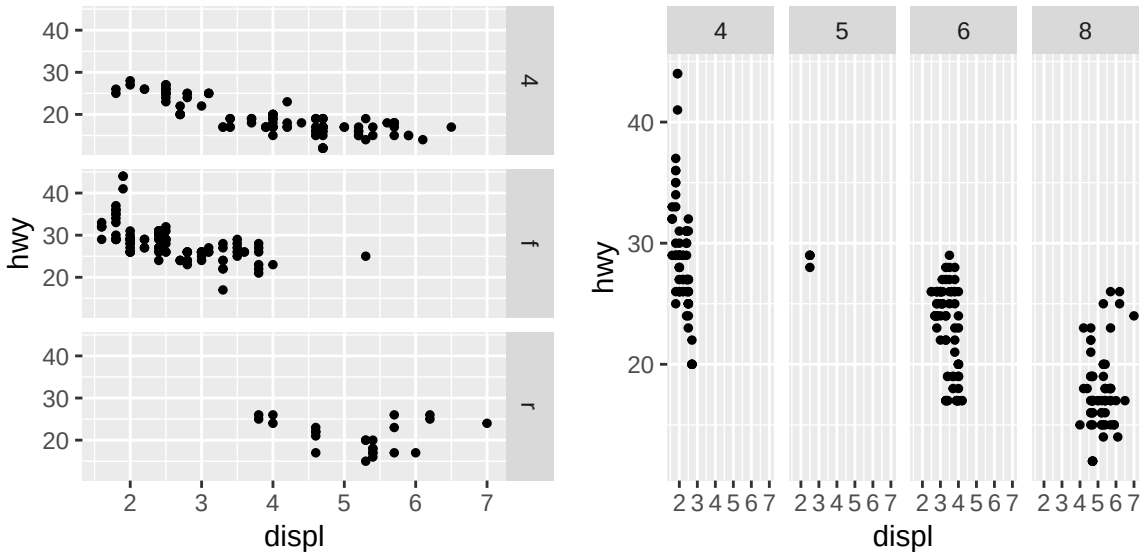


The empty cells in the plot mean that there were no observations for those particular combinations of values, or, there are no rows in the original dataset that correspond to those particular combinations of variable values.

In this plot, the empty cells mean that there are no rear wheel drive cars which have 4 cylinders or 5 cylinders and there are no 4 wheel drive cars which have 5 cylinders.

3. What plots does the following code make? What does . do?

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy), size = 1) +
  facet_grid(drv ~ .)
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy), size = 1) +
  facet_grid(. ~ cyl)
```



The first code plots engine displacement vs highway miles per gallon for 4 wheel, front wheel and rear wheel drive cars. The second code plots the engine displacement vs highway miles per gallon for cars with 4,5,6,and 8 cylinders.

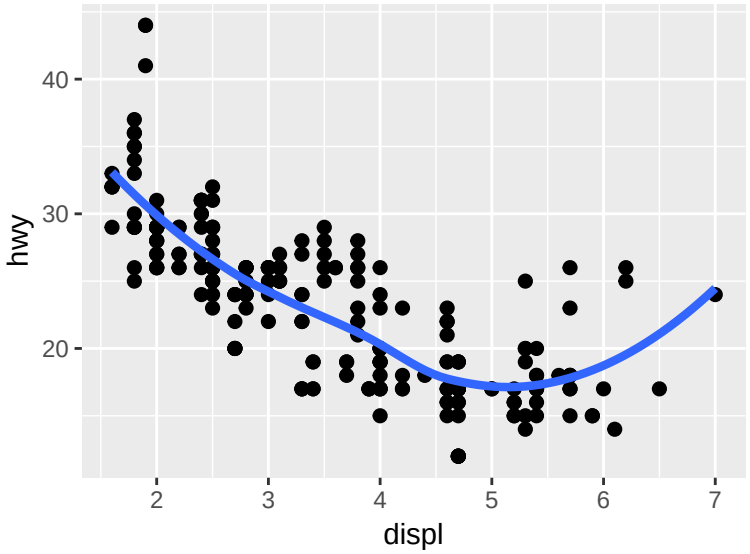
The . is like a placeholder that controls whether the faceting happens row wise or column wise. In the plot on the left side, (drv .) plots the drv as rows, whereas in the plot on the right side, (. cyl) plots the cyl as columns

3.6.1 Exercises

6. Recreate the R code necessary to generate the following graphs.

```
ggplot(mpg)+
  geom_point(aes(x = displ, y = hwy), size = 2)+
  geom_smooth(aes(x= displ, y = hwy), se = FALSE, lwd = 1.5)
```

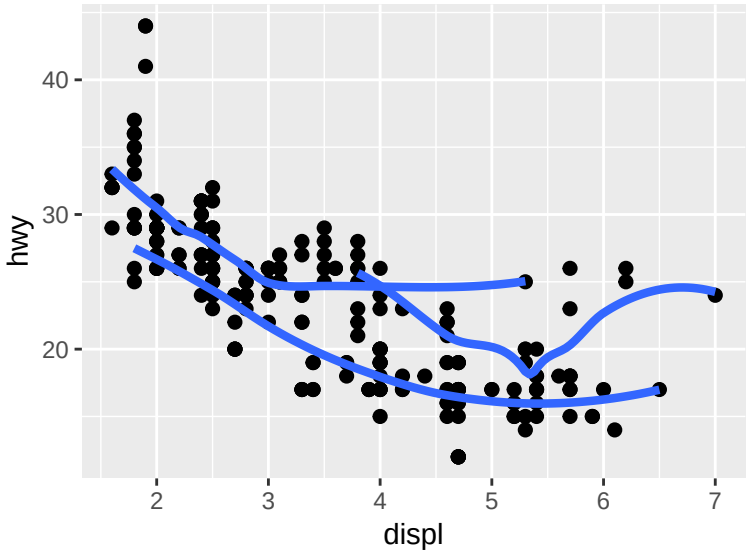
```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



```
ggplot(mpg)+
  geom_point(aes(x = displ, y = hwy), size = 2)+
  geom_smooth(aes(x= displ, y = hwy, class = drv), se = FALSE, lwd = 1.5)
```

```
## Warning: Ignoring unknown aesthetics: class
```

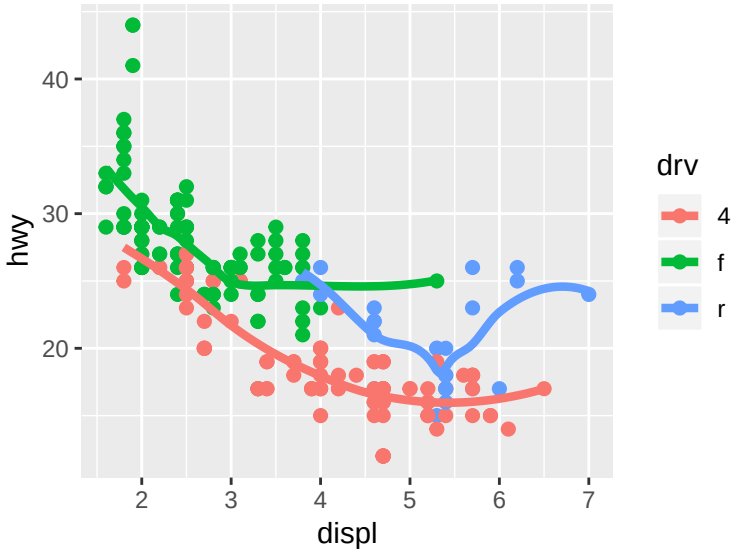
```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```




```
ggplot(mpg)+
  geom_point(aes(x = displ, y = hwy, color = drv), size = 2)+
  geom_smooth(aes(x= displ, y = hwy, class = drv, color = drv), se = FALSE, lwd = 1.5)
```

```
## Warning: Ignoring unknown aesthetics: class
```

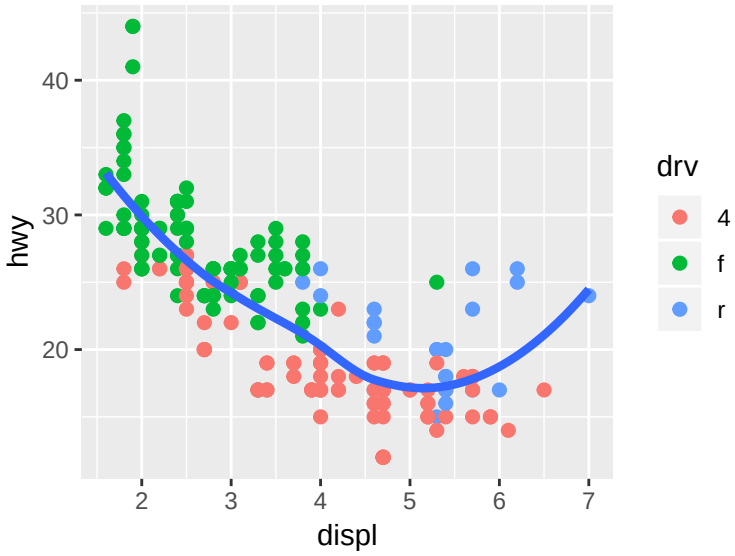
```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```




```
ggplot(mpg)+
  geom_point(aes(x = displ, y = hwy, color = drv), size = 2)+
```

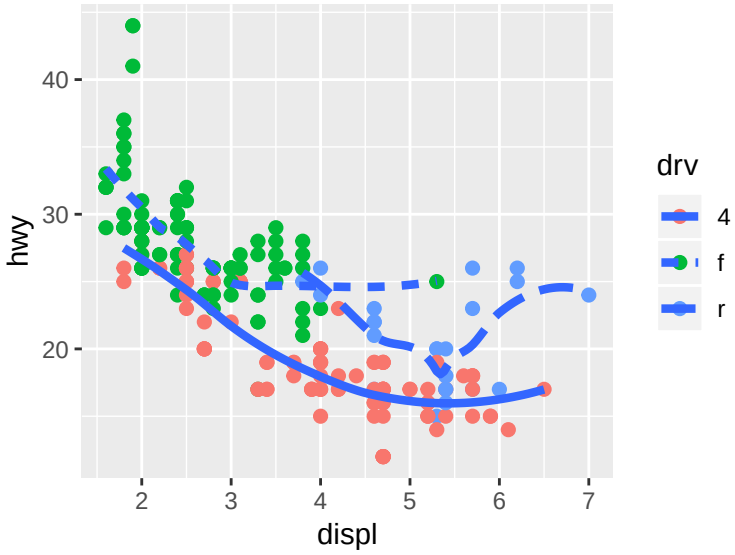
```
  geom_smooth(aes(x= displ, y = hwy), se = FALSE, lwd = 1.5)
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

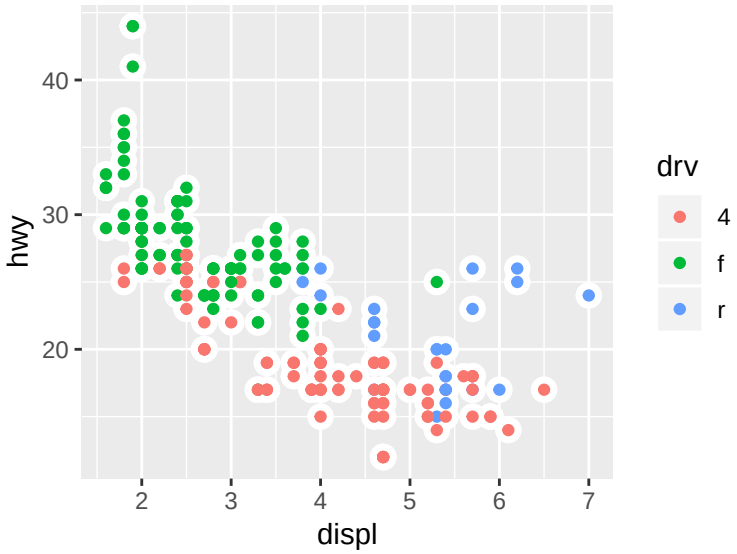



```
ggplot(mpg)+
  geom_point(aes(x = displ, y = hwy, color = drv), size = 2)+
  geom_smooth(aes(x= displ, y = hwy, linetype = drv), se = FALSE, lwd = 1.5)
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```




```
ggplot(mpg, aes(x = displ, y = hwy))+
  geom_point(size = 4, color = "white")+
  geom_point(aes(color = drv))
```



5.2.4 Exercises

Find all flights that:

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidyverse)
```

```
## -- Attaching packages ------------------------------------------------------------------------------ ti
```

```
## v tibble  1.4.2     v purrr   0.2.5
## v tidyr   0.8.1     v stringr 1.3.1
## v readr   1.1.1     v forcats 0.3.0
```

```
## -- Conflicts --------------------------------------------------------------------------------- tidyverse
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(nycflights13)
data(flights)
```

1. Had an arrival delay of two or more hours

```
#Flights that 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)

```
#Flights that flew to Houston
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 Delta8

```
#Flights that 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)

```
#Flights that 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      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

```
#Flights that arrived more than 2 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

If a flight is delayed by an hour, then it should arrive an hour late if it did not make up any time in the air. Therefore, if the flight made up over 30 mins in air, then the difference between the delay in departure and delay in arrival should be more than 30 mins.

```
#Flights that were delayed atleast by an hour, but made up over 30 minutes in flight
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)

```
#Flights that departed between midnight and 6 am
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>
```

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?

The between filtering helper is an equivalent to less than and greater than. For example, it could have been used in the part 4 of the previous question for flights that departed in summer. between could have been used as follows:

```
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>
```

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

```
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>
```

Along with dep time, dep delay, arr time, arr delay and air time variabels are missing. The rows with missing values correspond to cancelled flights.

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 to the power 0 is 1 because anything to the power 0 is 1. NA or TRUE evaluates to TRUE and hence is not missing. FALSE and NA evaluates to FALSE and hence is not missing.