Exercise in R for Data Science

Yifu Dong

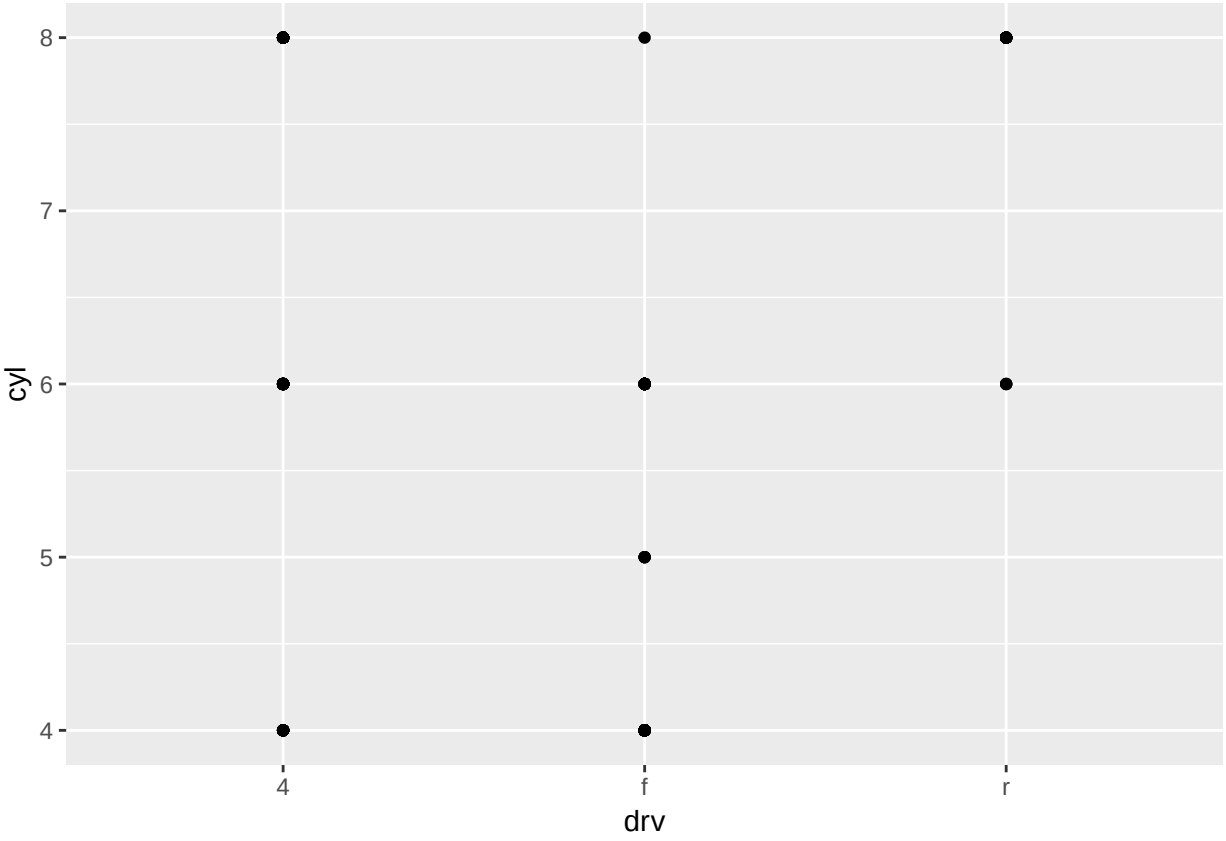
September 23, 2018

3.5.1 Exercises

2

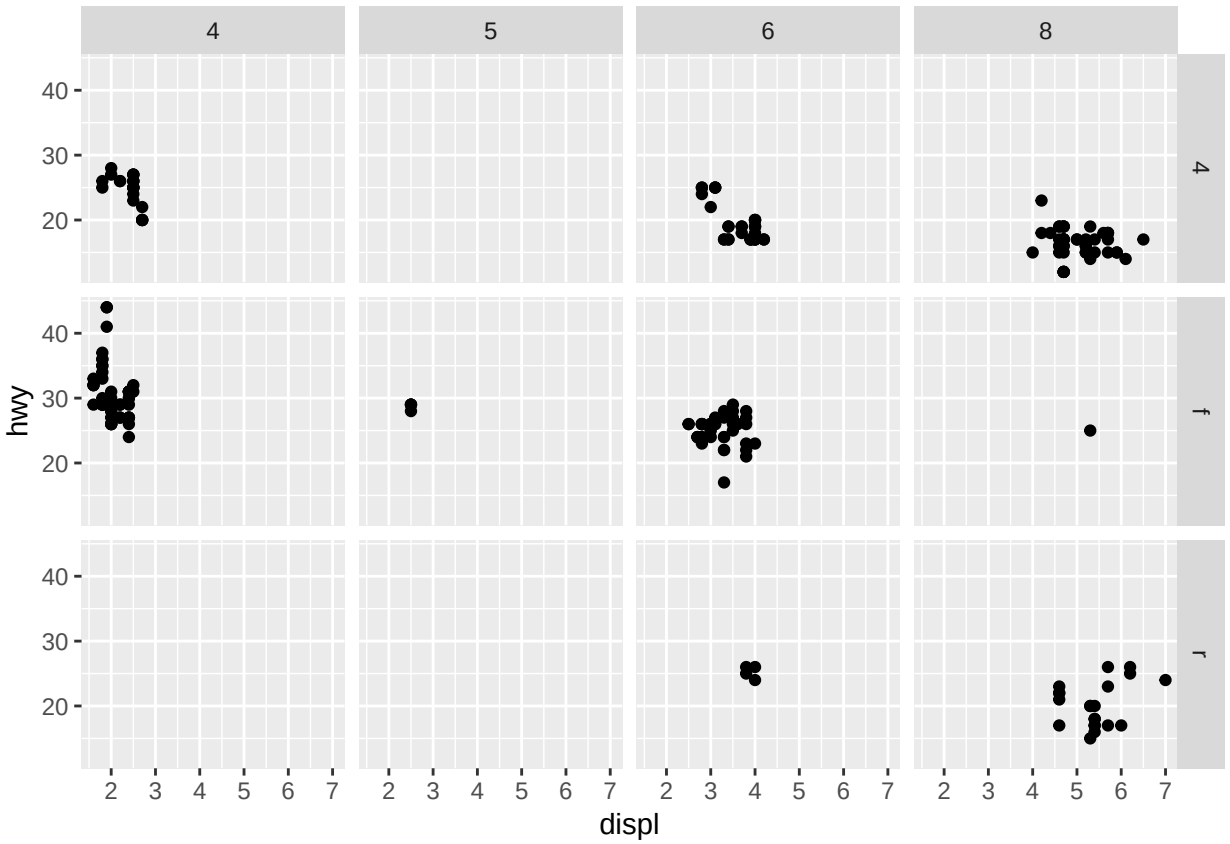
What do the empty cells in plot with facet_grid(drv ~ cyl) mean? How do they relate to this plot?

```
ggplot(data=mpg)+
  geom_point(mapping=aes(x=drv,y=cyl))
```



Solution:

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

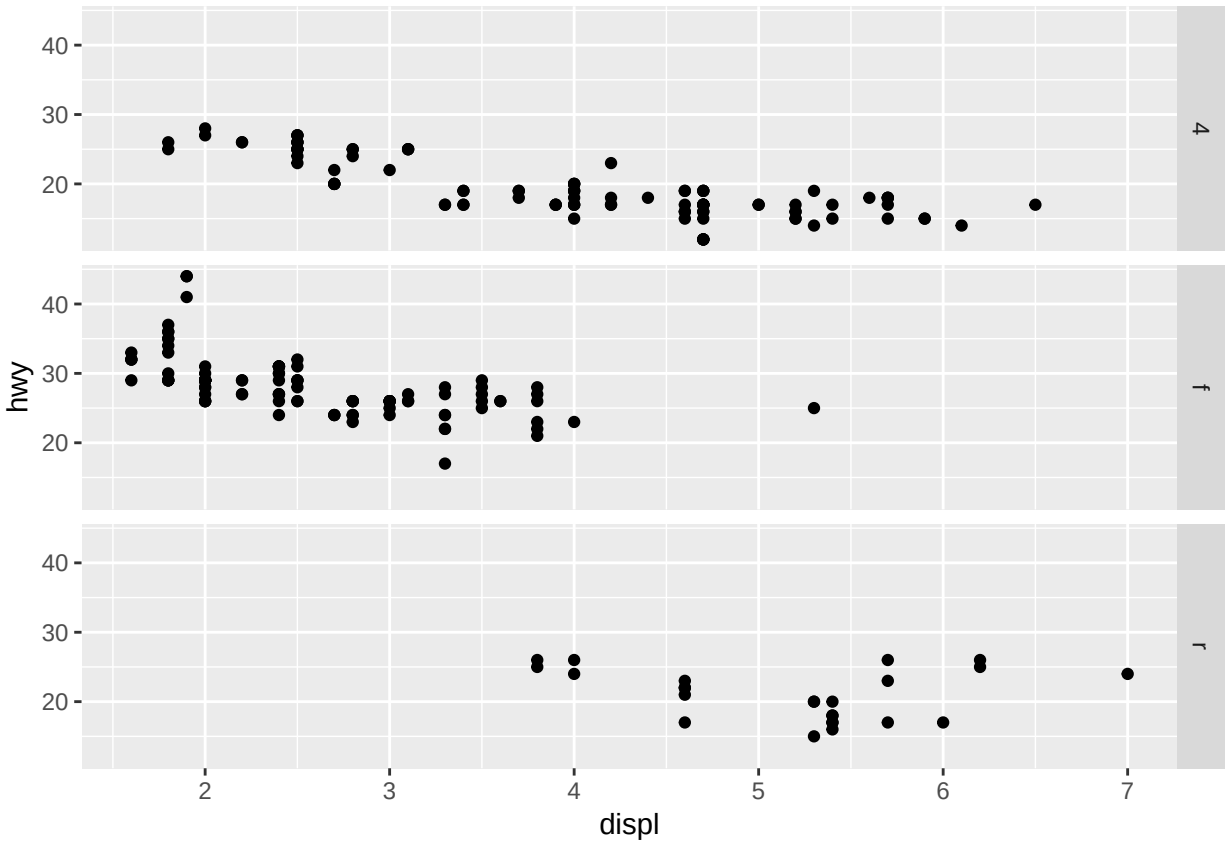


There are some empty cells in this plot. Empty cells imply that there were no rows with that combination of values in the original dataset. For example, the plot with drv=4 and cyl=r is empty, thus it means that in the original dataset we don't have such kind of data whose drv=4 while cyl=r. And it is related to the plot given above since the plot above provides the coordinate between drv and cyl. Hence, the empty coordinate is correspondingly the empty cells in a plot with facet_grid(drv~cyl)

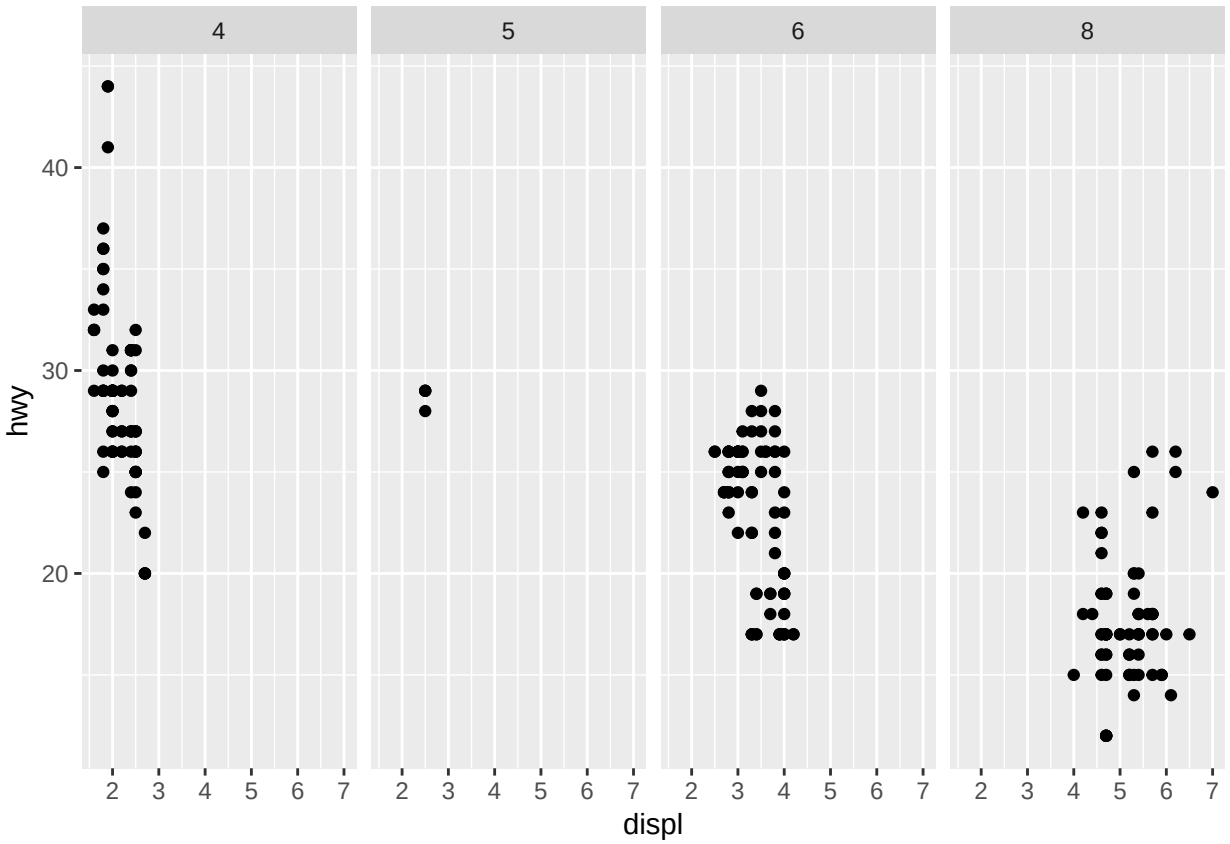
3

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

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




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



Solution: We can know from the plots above that . is a placeholder so that we can have a facet in only one dimension which is given by drv or cyl.

3.6.1 Exercise

6

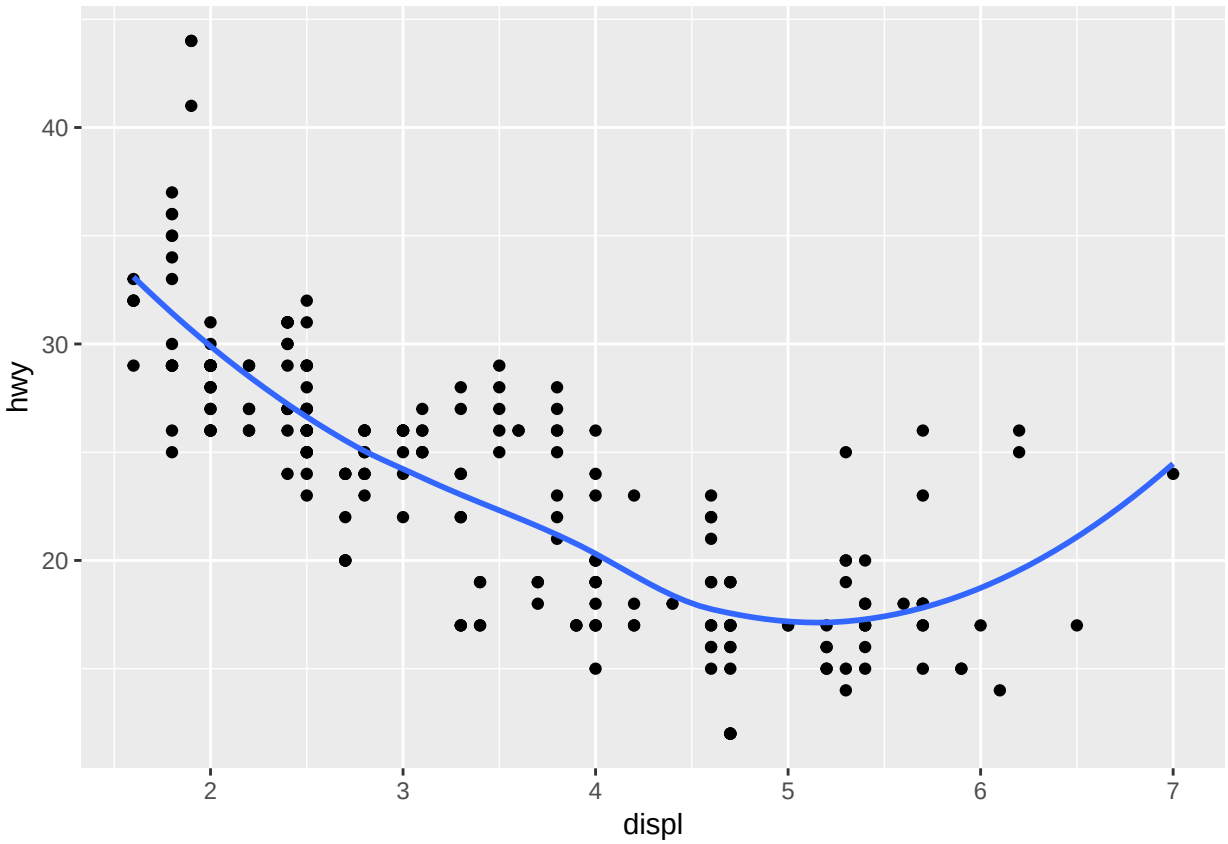
Recreate the R code necessary to generate the following graphs

Solution:

plot1:

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

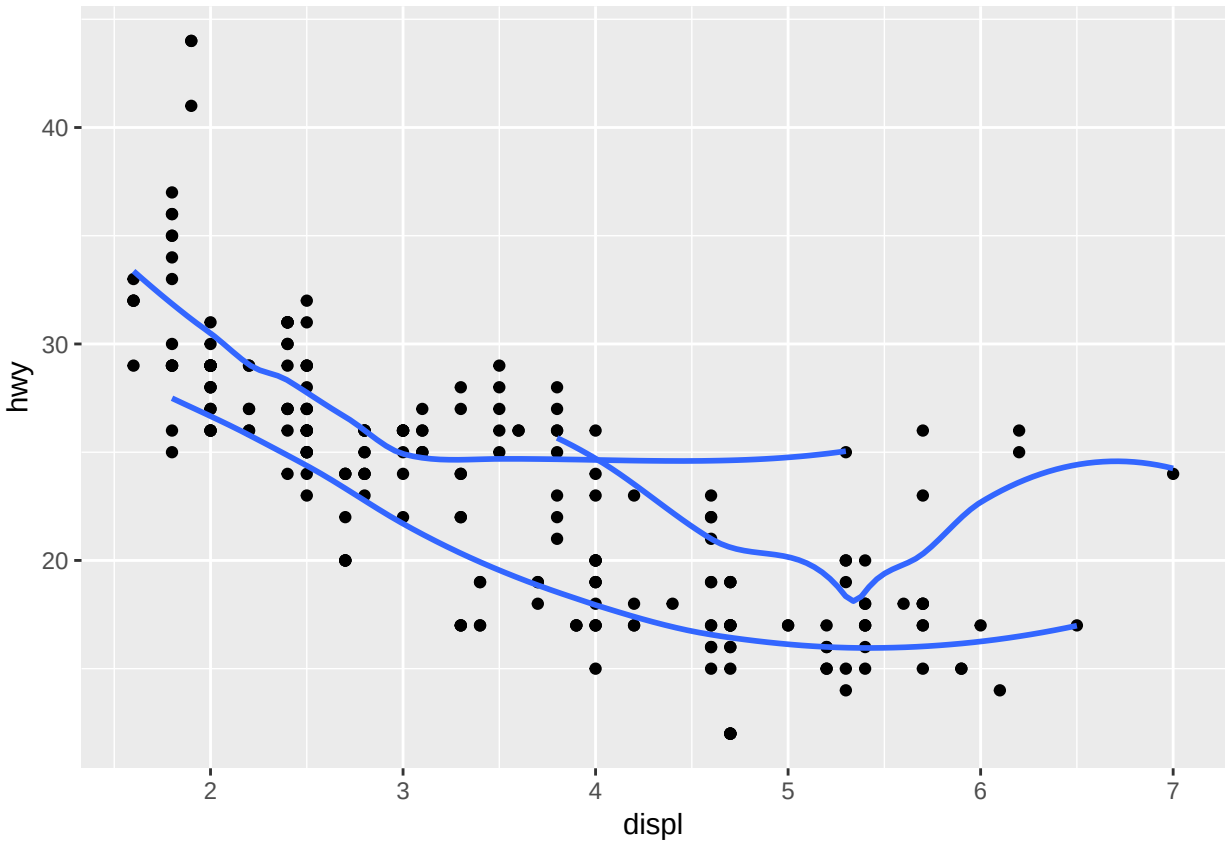
```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



plot2:

```
ggplot(data=mpg)+
  geom_point(mapping=aes(x=displ,y=hwy))+
  geom_smooth(mapping=aes(x=displ,y=hwy,group=drv),se=FALSE)
```

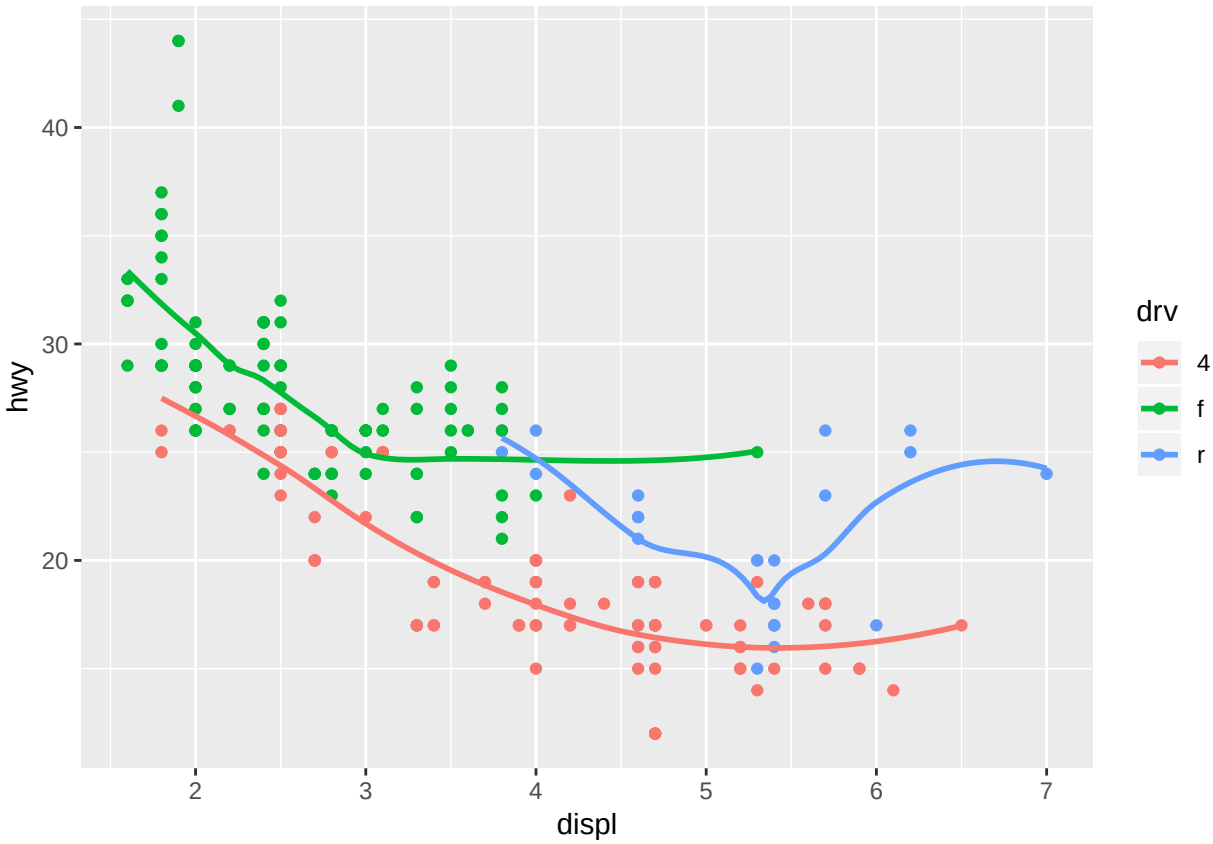
```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



plot3:

```
ggplot(data=mpg)+
  geom_point(mapping = aes(x=displ,y=hwy,color=drv))+
  geom_smooth(mapping = aes(x=displ,y=hwy,color=drv), se=FALSE)
```

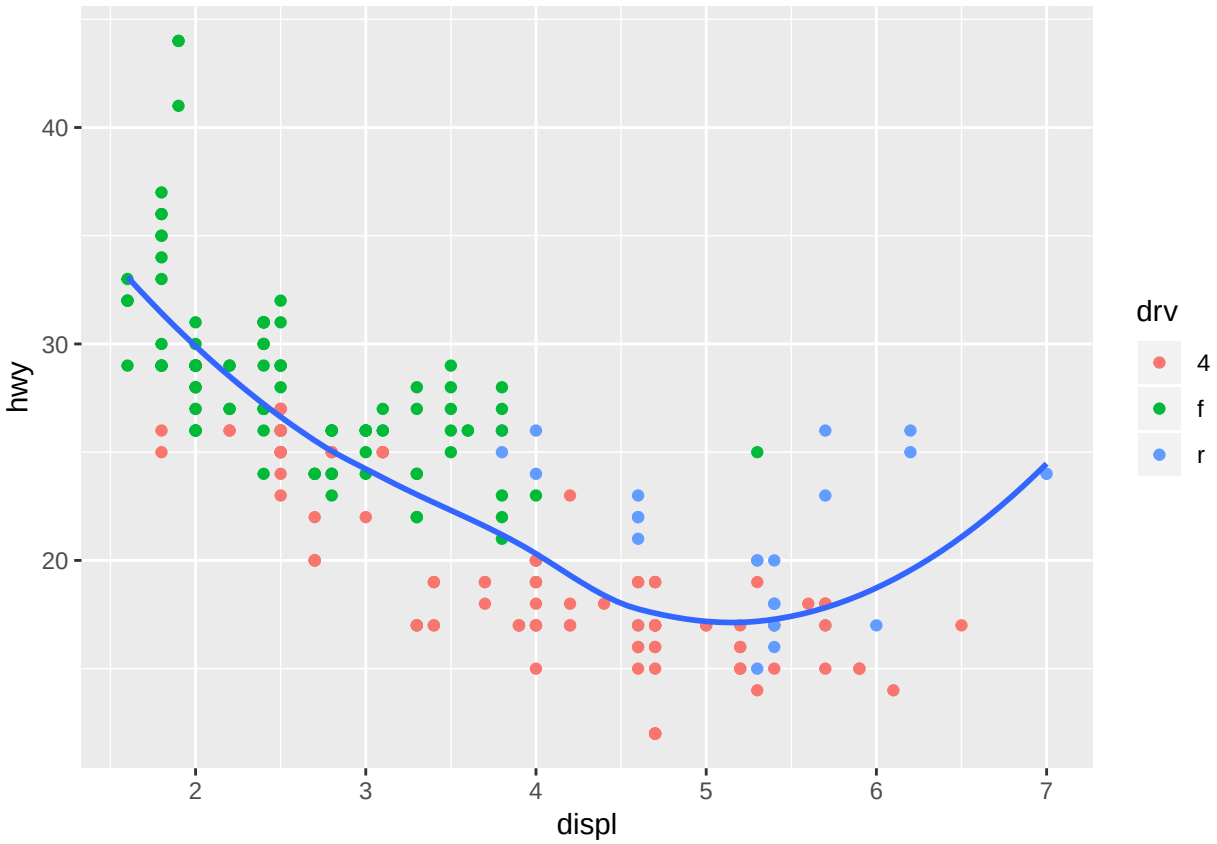
```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



plot4:

```
ggplot(data=mpg)+
  geom_point(mapping = aes(x=displ,y=hwy,color=drv))+
  geom_smooth(mapping = aes(x=displ,y=hwy), se=FALSE)
```

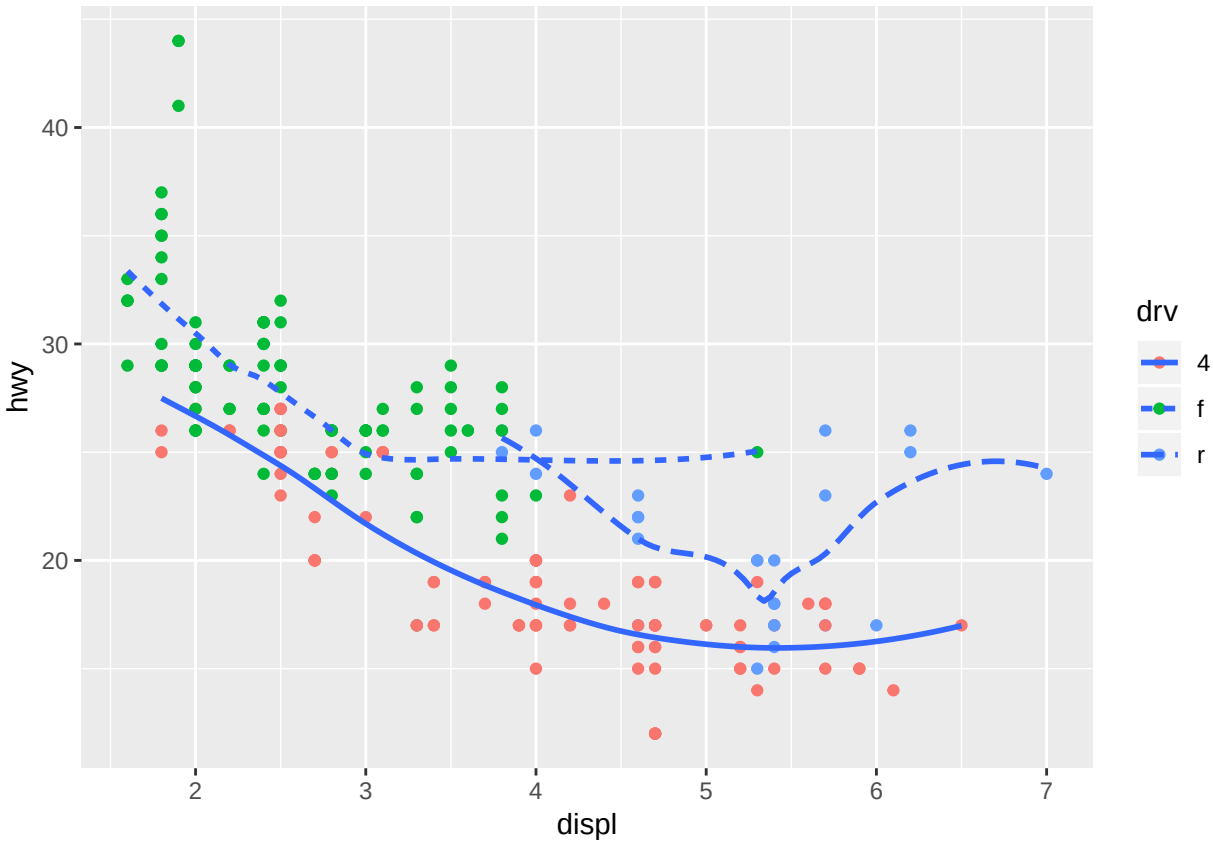
```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



plot5:

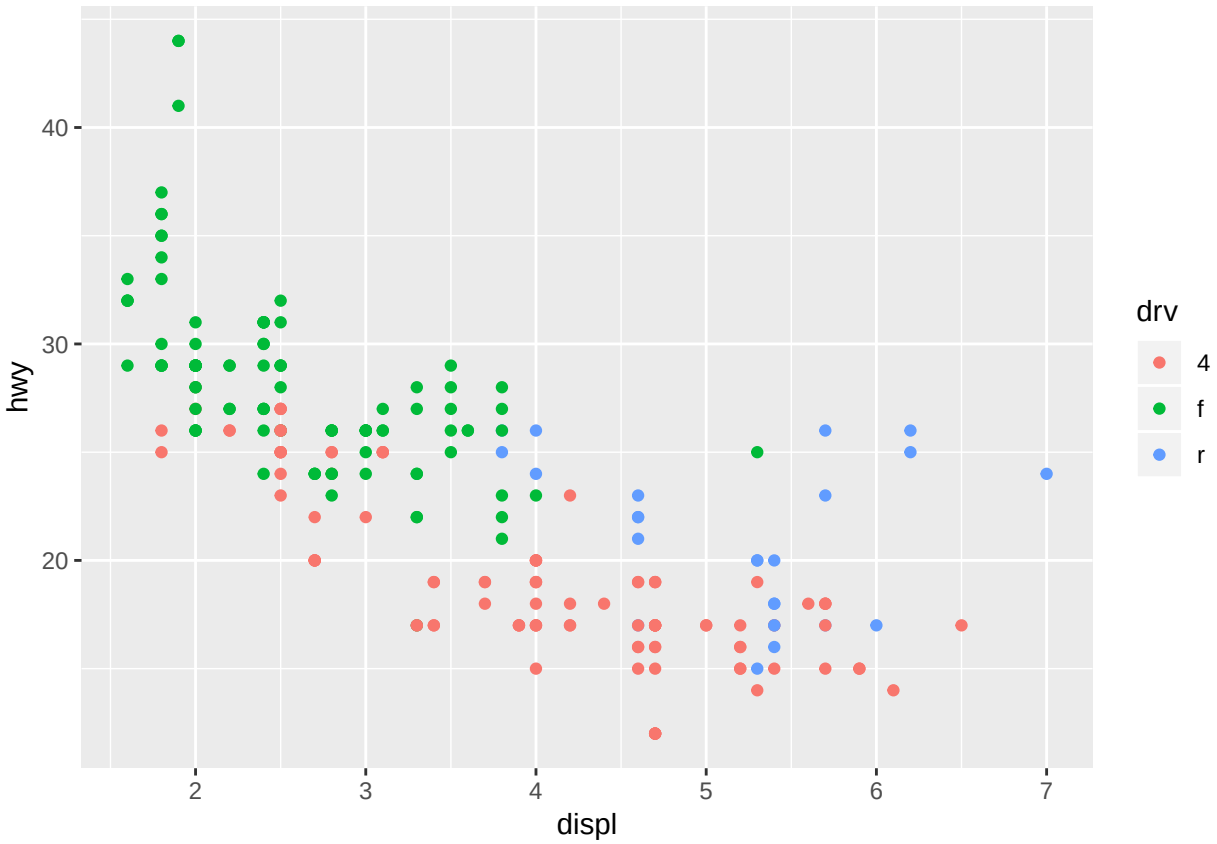
```
ggplot(data=mpg)+
  geom_point(mapping = aes(x=displ,y=hwy,color=drv))+
  geom_smooth(mapping = aes(x=displ,y=hwy,linetype=drv), se=FALSE)
```

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



plot6:

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



5.2.4 Exercise

```
library("nycflights13")
```

1

Find all flights that:

a.Had an arrival delay of two or more hours

```
filter(flights,dep_delay>=2)
```

```
## # A tibble: 120,382 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      608            600         8      807
##  5  2013     1     1      611            600        11      945
##  6  2013     1     1      613            610         3      925
##  7  2013     1     1      623            610        13      920
##  8  2013     1     1      632            608        24      740
##  9  2013     1     1      644            636         8      931
## 10  2013     1     1      702            700         2     1058
```

```
## # ... with 120,372 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>
```

b.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>
```

c.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>
```

d.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>
```

e.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>
```

f.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: 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>
```

g.Departed between midnight and 6am (inclusive)

```
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?

between() is a way of testing two inequalities at once,

```
filter(flights, between(month, 5, 10))
```

```
## # A tibble: 172,254 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013    10     1      447            500       -13      614
##  2  2013    10     1      522            517         5      735
##  3  2013    10     1      536            545        -9      809
##  4  2013    10     1      539            545        -6      801
##  5  2013    10     1      539            545        -6      917
##  6  2013    10     1      544            550        -6      912
##  7  2013    10     1      549            600       -11      653
##  8  2013    10     1      550            600       -10      648
##  9  2013    10     1      550            600       -10      649
## 10  2013    10     1      551            600        -9      727
## # ... with 172,244 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?

```
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 05:22:54
##  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 a missing dep_time. Also from the table above, we can find that 8255 have a missing dep_delay,9430 have a missing arr_delay, and 9430 have a missing air_time, 8713 have a missing arr_time. This represents that there are more than 8000 flights didn't arrive on time.

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 not missing because anything to the power of 0 is 1, so although we didn't know the original value, we know the value of NA^0

With NA | TRUE, since the | operator returns TRUE if either of the terms are true, the whole expression returns true because the right half returns true.

For the next example, since FALSE is false, and that's enough for FALSE & NA to be evaluated false.

As for NA*0:

```
NA*0
```

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

Since we don't know the true value of NA, it can be 0, or Inf, or 100. So the output is NA.