Assignment 2-Exercise

Jinfei Xue

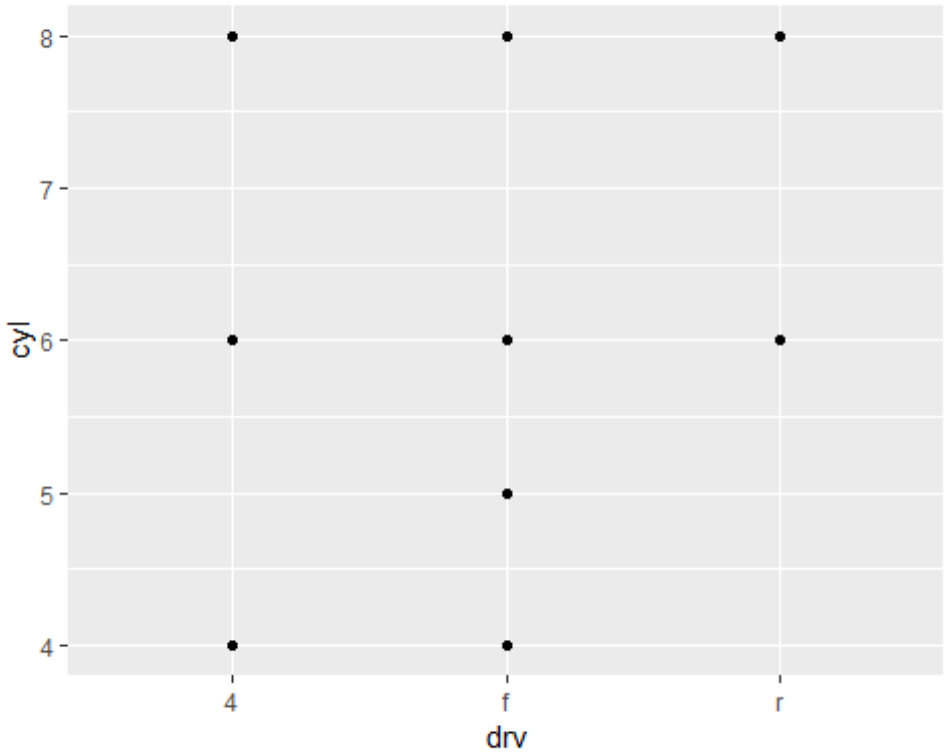
September 24, 2018

3.5.1

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

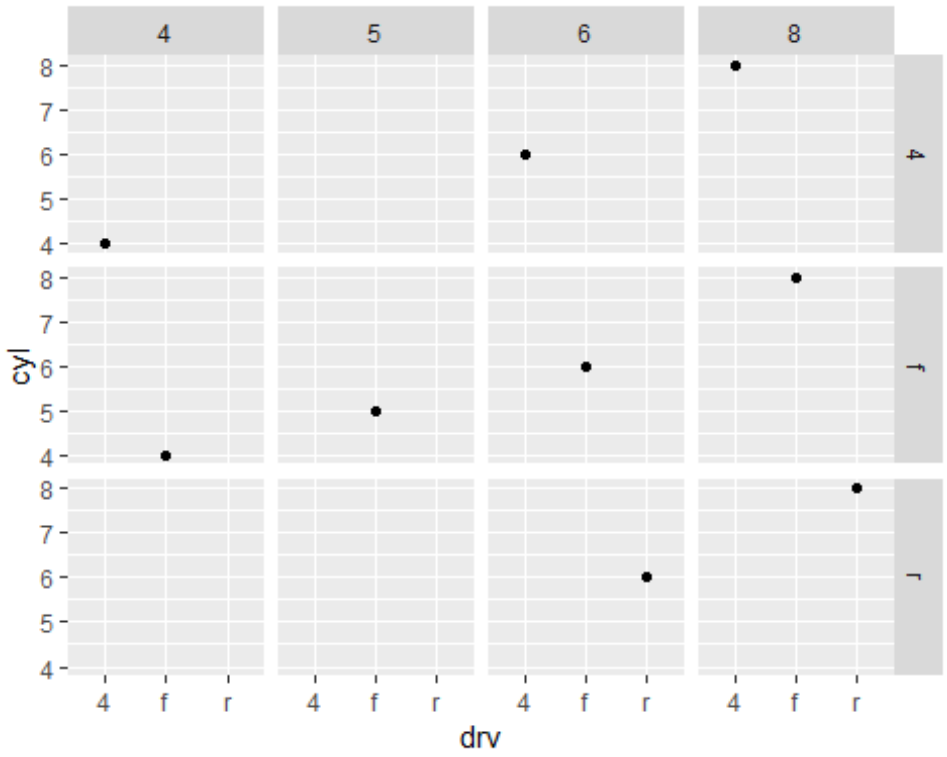
the original plot

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



the plot with facet_grid(drv ~ cyl)

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



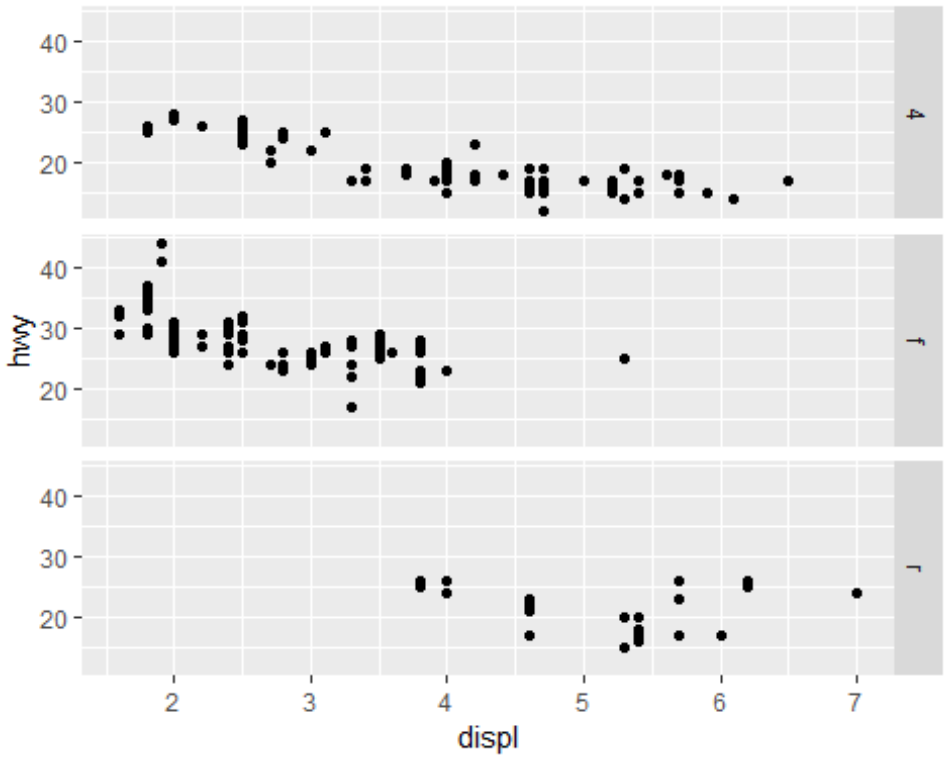
There exist several empty cells in plots shown above. They mean there are no observations in the data that have that specific combination of values.

For instance, in these plots we can say that there are no vehicles with 4 or 5 cylinders that are also rear wheel drive (r).

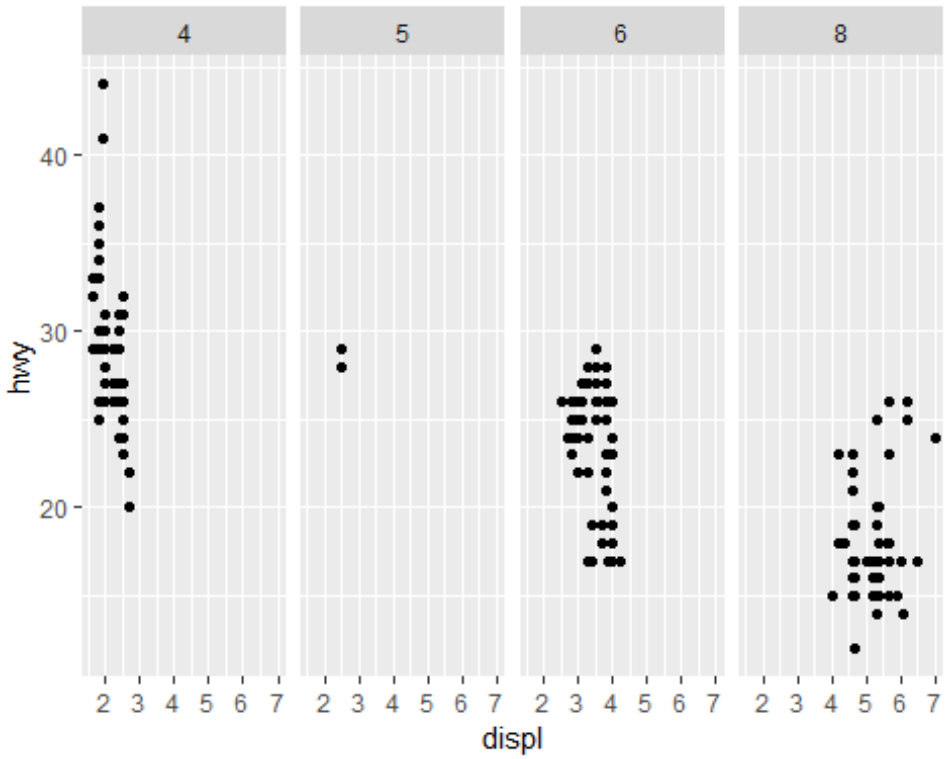
The plot is similar to the original one because each facet only appears to have a single data point.

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)
```



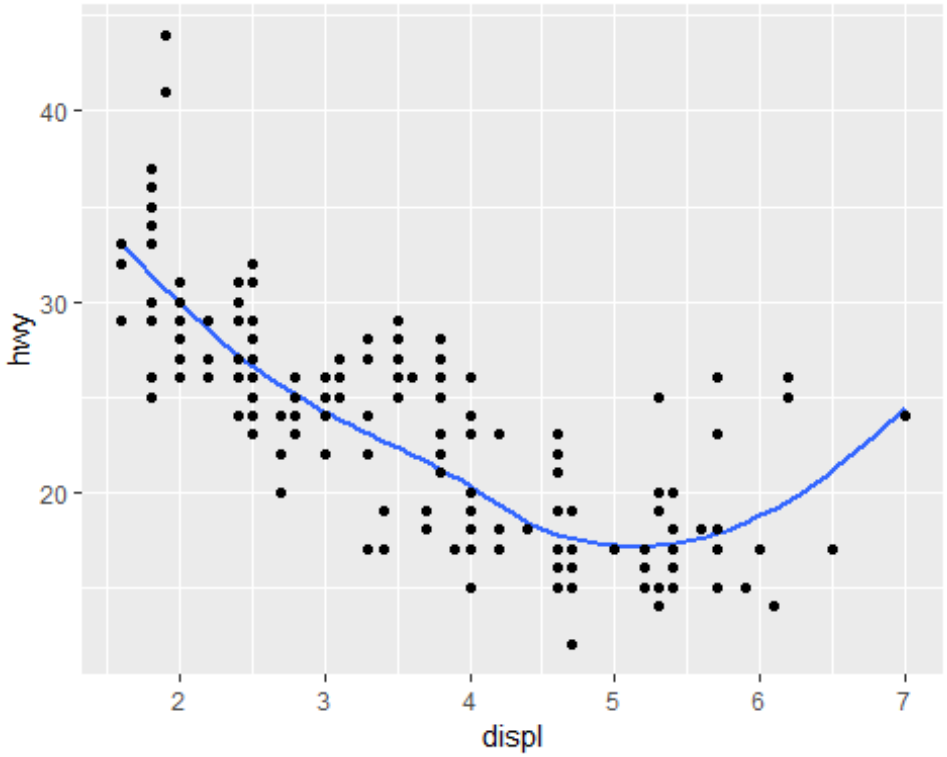
. represents no variable. That is to say, this results in a plot faceted on a single dimension rather than an N by N grid in facet_grid() function.

3.6.1

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

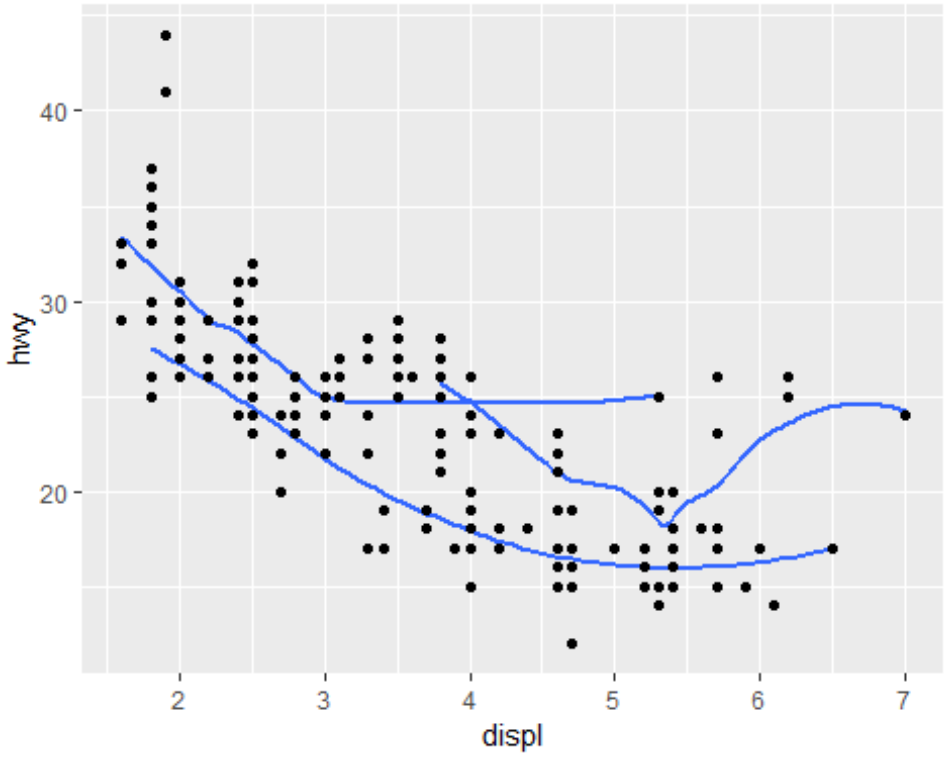
```
#plot 1
ggplot(data = mpg, mapping = aes(x=displ,y=hwy)) +
  geom_smooth(se = FALSE)+
  geom_point()

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



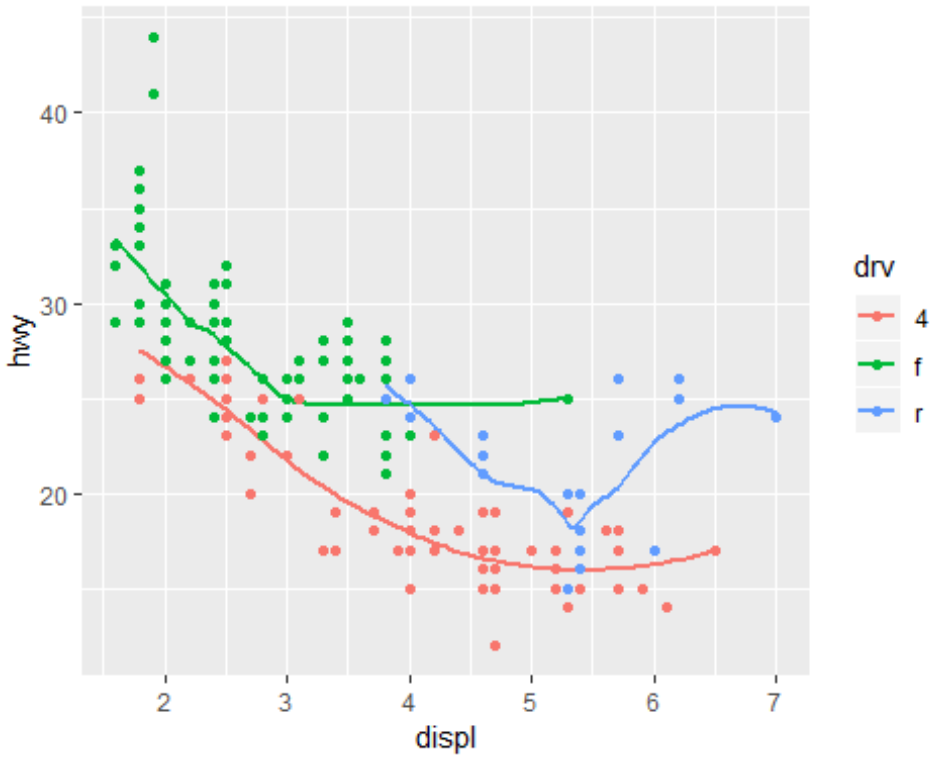

```
#plot 2
ggplot(data = mpg, mapping = aes(x=displ,y=hwy,group=drv)) +
  geom_smooth(se = FALSE)+
  geom_point()

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



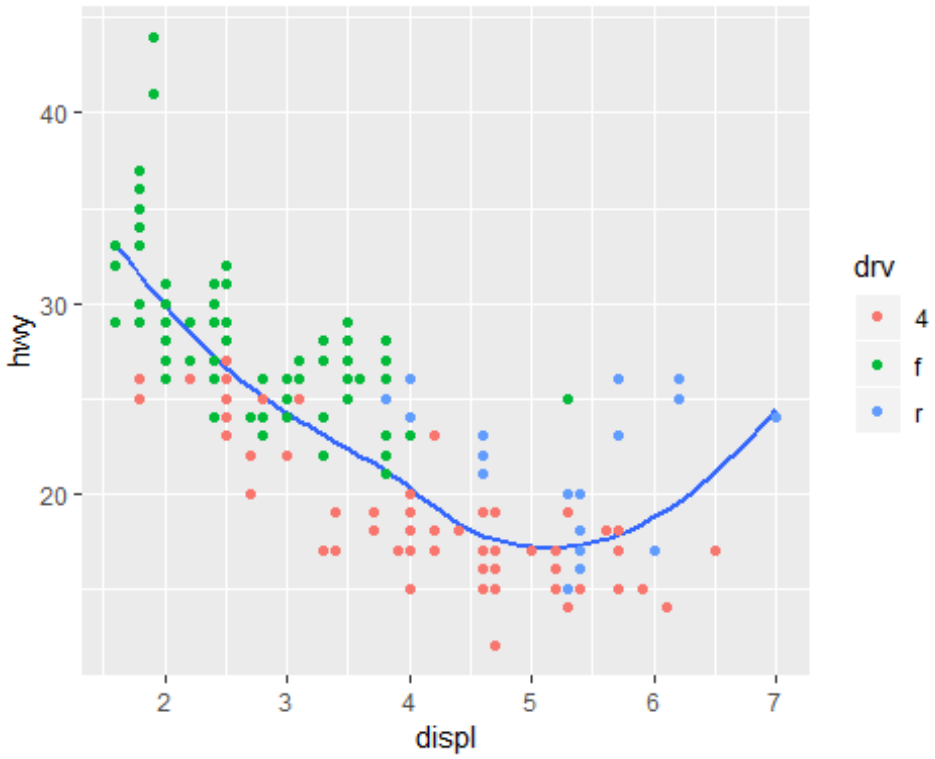

```
#plot 3
ggplot(data = mpg, mapping = aes(x=displ,y=hwy,group=drv)) +
  geom_smooth(se = FALSE,aes(colour=drv))+
  geom_point(aes(colour=drv))

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



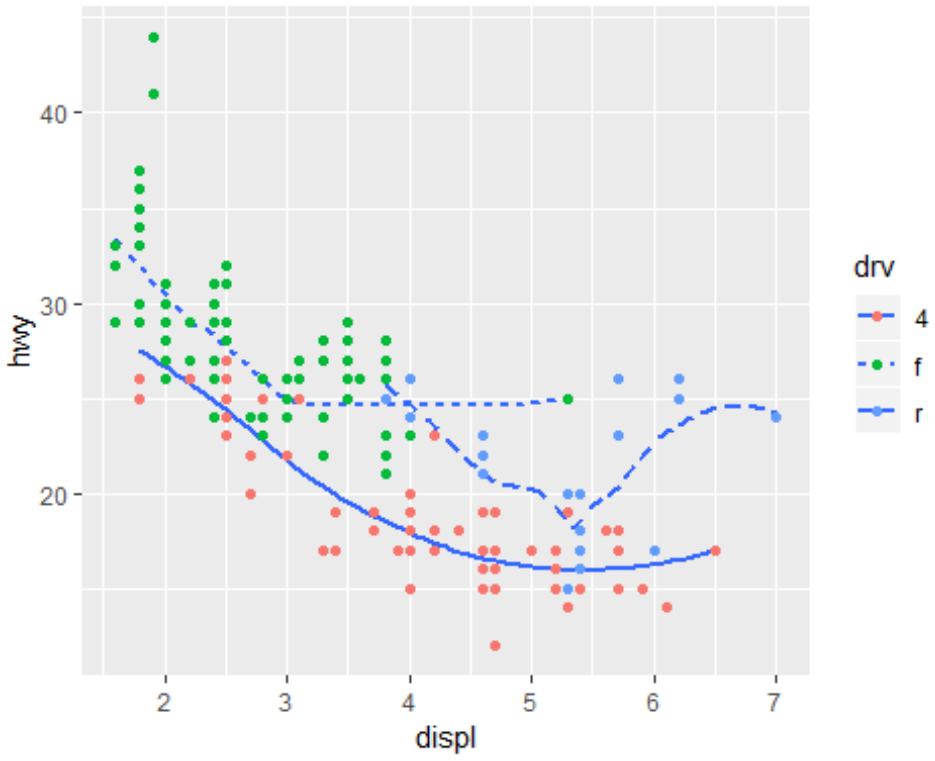

```
#plot 4
ggplot(data = mpg, mapping = aes(x=displ,y=hwy)) +
  geom_smooth(se = FALSE)+
  geom_point(aes(colour=drv))

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

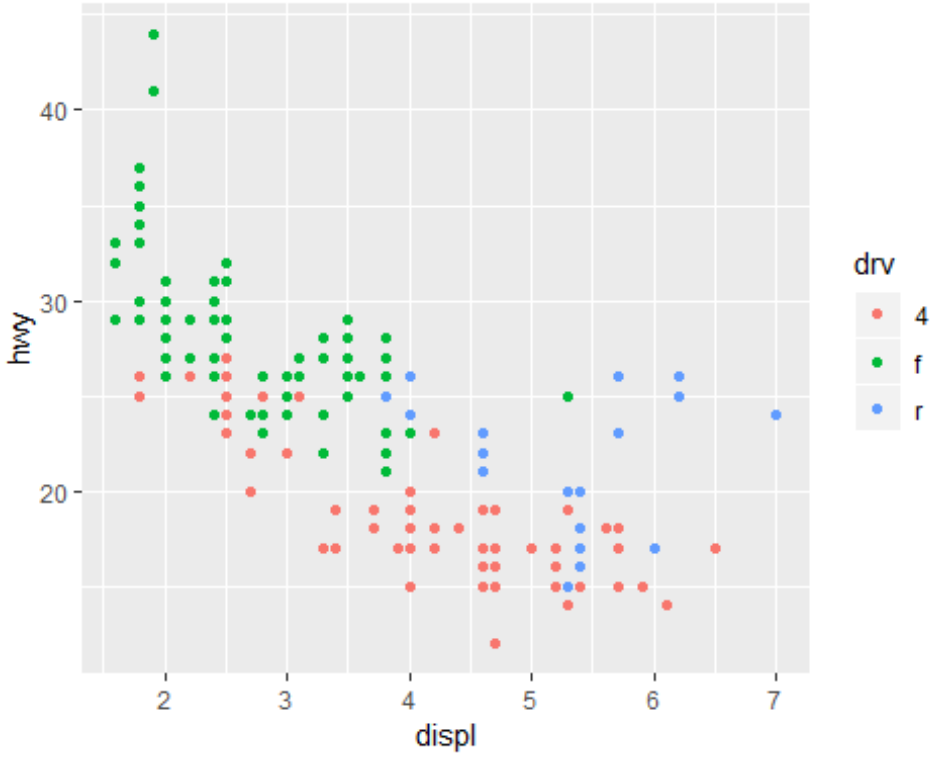



```
#plot 5
ggplot(data = mpg, mapping = aes(x=displ,y=hwy,group=drv)) +
  geom_smooth(se = FALSE,aes(linetype=drv))+
  geom_point(aes(colour=drv))

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




```r
#plot 6
ggplot(data = mpg, mapping = aes(x=displ,y=hwy)) +
  geom_point(aes(colour=drv))
```



5.2.4

1. Find all flights that:

1> Had an arrival delay of two or more hours

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

```
## -- Attaching packages ---------------------------------------------
-------- tidyverse 1.2.1 --

## √ tibble  1.4.2     √ purrr   0.2.5
## √ tidyr   0.8.1     √ dplyr   0.7.6
## √ readr   1.1.1     √ stringr 1.3.1
## √ tibble  1.4.2     √ forcats 0.3.0

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

```
flights_1<-filter(flights, arr_delay>=120)
flights_1
```

```
## # 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_2<-filter(flights, dest %in% c('IAH', 'HOU'))
flights_2
```

```
## # 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 <i
nt>,
## #   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

```
flights_3<-filter(flights, carrier %in% c('UA', 'AA', 'DL'))
flights_3
```

```
## # 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_4<-filter(flights, month %in% c(7, 8, 9))
flights_4
```

```
## # 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_5<-filter(flights, arr_delay > 120, dep_delay <= 0)
flights_5
```

```
## # 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

```
flights_6<-filter(flights, dep_delay >= 60, dep_delay-arr_delay > 30)
flights_6
```

```
## # 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 <i
nt>,
## #   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_7<-filter(flights, dep_time >= 2400 | dep_time <= 600)
flights_7
```

```
## # 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 <i
nt>,
## #   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 shorter, faster way of testing two inequalities at once: it tests if its first argument is greater than or equal to its second position, and less than or equal to its third position.

```
flights_7<-filter(flights, !between(dep_time, 601, 2359))
flights_7
```

```
## # 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 <i
nt>,
## #   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?

```
sum(is.na(flights$dep_time))
```

```
## [1] 8255
```

```
map_dbl(flights, ~ sum(is.na(.x)))
```

```
##           year          month            day       dep_time sched_de
p_time
##              0              0              0           8255
     0
##      dep_delay       arr_time sched_arr_time      arr_delay        c
arrier
##           8255           8713              0           9430
     0
##         flight        tailnum         origin           dest       ai
r_time
##              0           2512              0              0
  9430
##       distance           hour         minute      time_hour
##              0              0              0              0
```

There are 8255 flights which have a missing dep_time.

Also, the table shows 8255 flights have a missing dep_delay, 8713 flights have a missing arr_time, 9430 flights have a missing arr_delay, and 9430 flights have a missing air_time.

Those rows represent that some flights are failed to depart or arrive.

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 is not missing because anything to the power of 0 is 1.

NA | TRUE is not missing because the whole expression means NA or TRUE, it will return TRUE.(as long as one of the terms is true, the expression evaluates to TRUE)

FALSE & NA is not missing because the whole expression means FALSE and NA, it will return FLASE.(as long as one of the terms is false, the expression evaluates to FALSE)

The general rule is that whenever there is a logical expressions, if one can be tested, then the result shouldn't be NA. And any operation that the results is determined, regardless of the number, the inputting NA does not affect the result.

$NA \cdot 0$ *could be argued to be because the NA could represent Inf, and* $Inf \cdot 0$ *is NaN (Not a Number), rather than NA.*