R4DS works

Abu Hanifa

2023-07-28

Chapter - 1

Example Codes

Loading library

```
library(tidyverse)
```

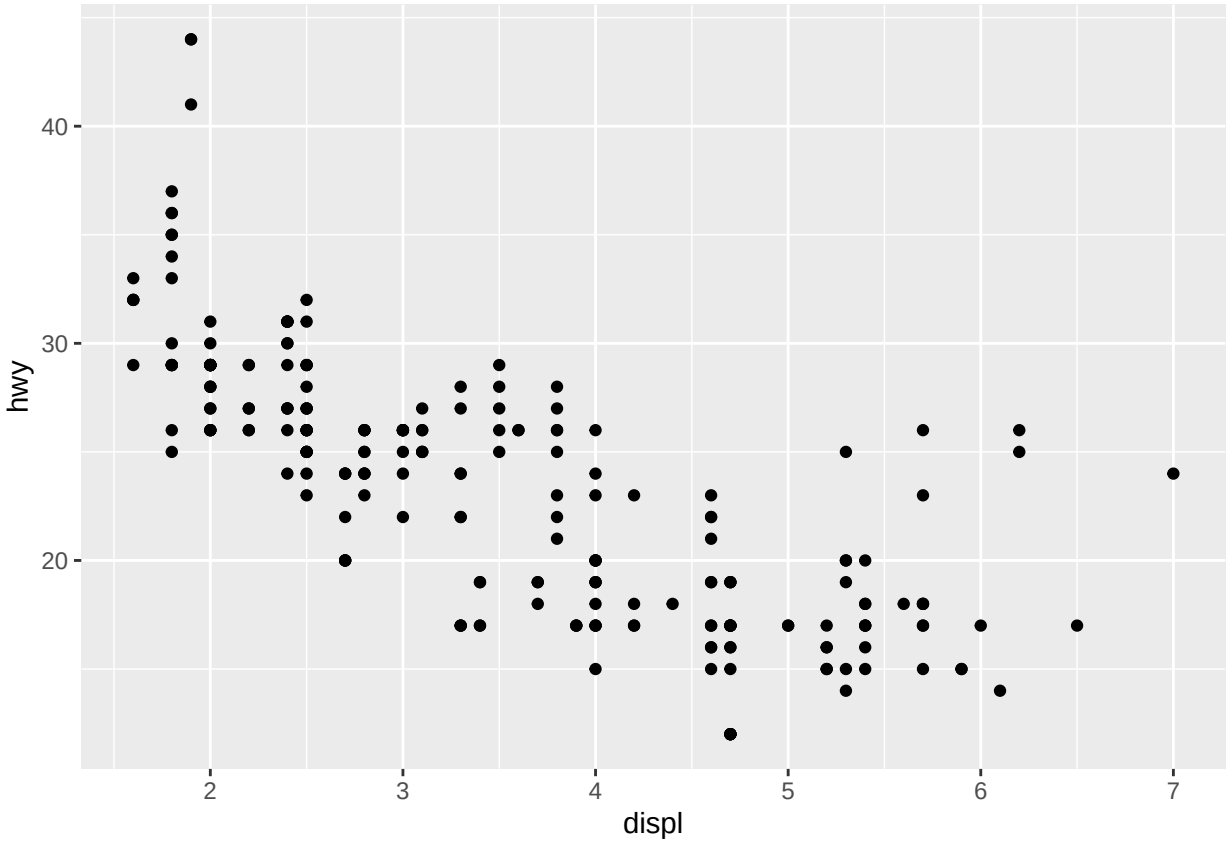
Using Mpg data set

```
mpg
```

```
## # A tibble: 234 x 11
##    manufacturer model      displ  year   cyl trans drv     cty   hwy fl    class
##    <chr>        <chr>      <dbl> <int> <int> <chr> <chr> <int> <int> <chr> <chr>
##  1 audi         a4           1.8  1999     4 auto~ f        18    29 p     comp~
##  2 audi         a4           1.8  1999     4 manu~ f        21    29 p     comp~
##  3 audi         a4           2    2008     4 manu~ f        20    31 p     comp~
##  4 audi         a4           2    2008     4 auto~ f        21    30 p     comp~
##  5 audi         a4           2.8  1999     6 auto~ f        16    26 p     comp~
##  6 audi         a4           2.8  1999     6 manu~ f        18    26 p     comp~
##  7 audi         a4           3.1  2008     6 auto~ f        18    27 p     comp~
##  8 audi         a4 quattro   1.8  1999     4 manu~ 4        18    26 p     comp~
##  9 audi         a4 quattro   1.8  1999     4 auto~ 4        16    25 p     comp~
## 10 audi         a4 quattro   2    2008     4 manu~ 4        20    28 p     comp~
## # ... with 224 more rows
```

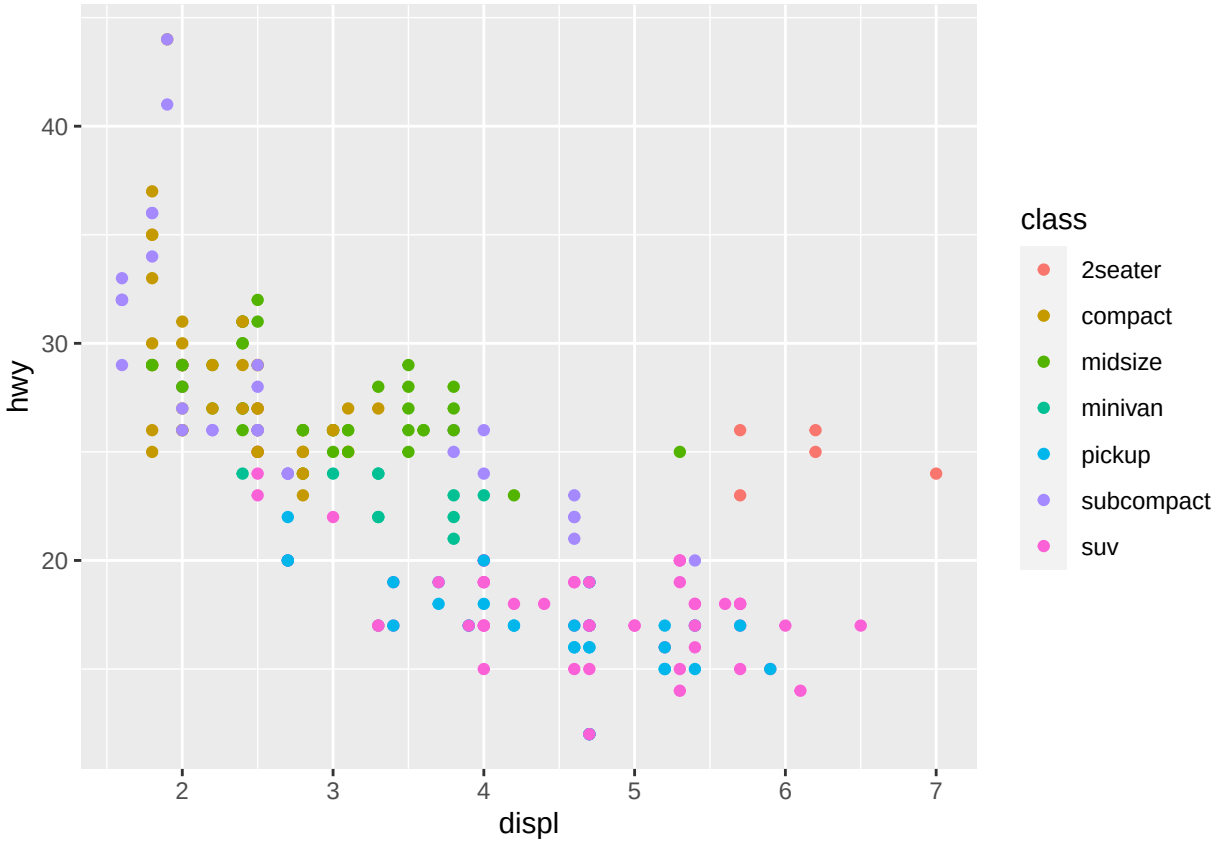
Simple scatterplot

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



Scatterplot with respect to class presenting in different color

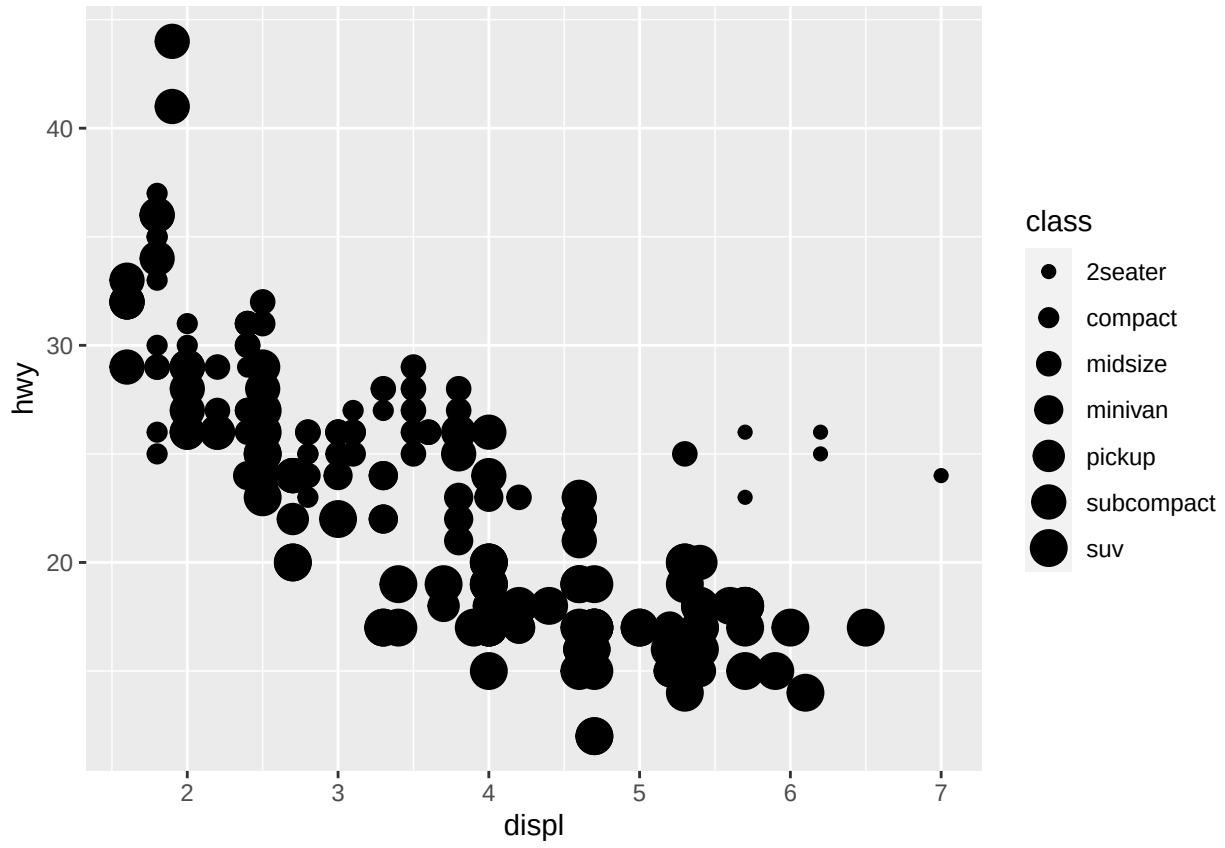
```
ggplot(data = mpg)+
  geom_point(mapping = aes(x = displ, y = hwy, color = class))
```



Scatterplot with respecct to class for different size

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

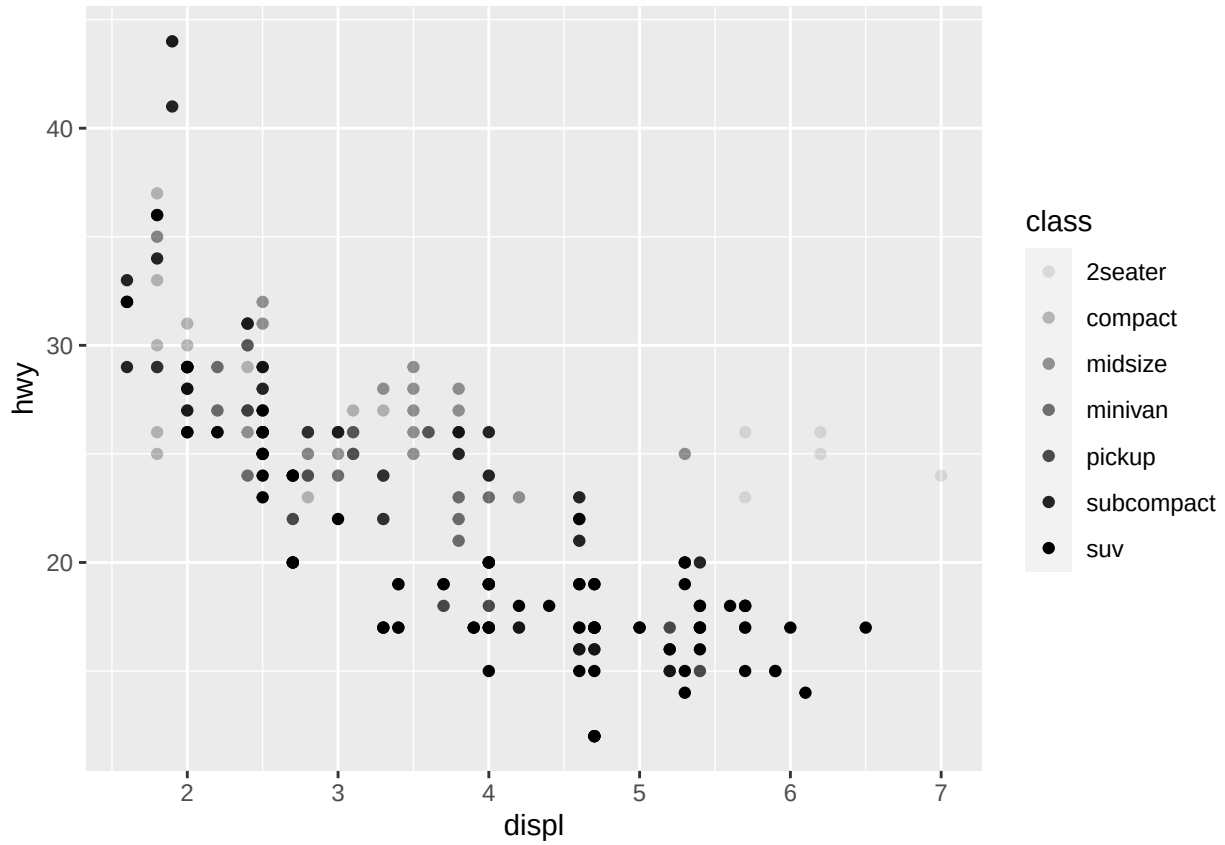
```
## Warning: Using size for a discrete variable is not advised.
```



With respect to different alpha

```
# Top
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, alpha = class))
```

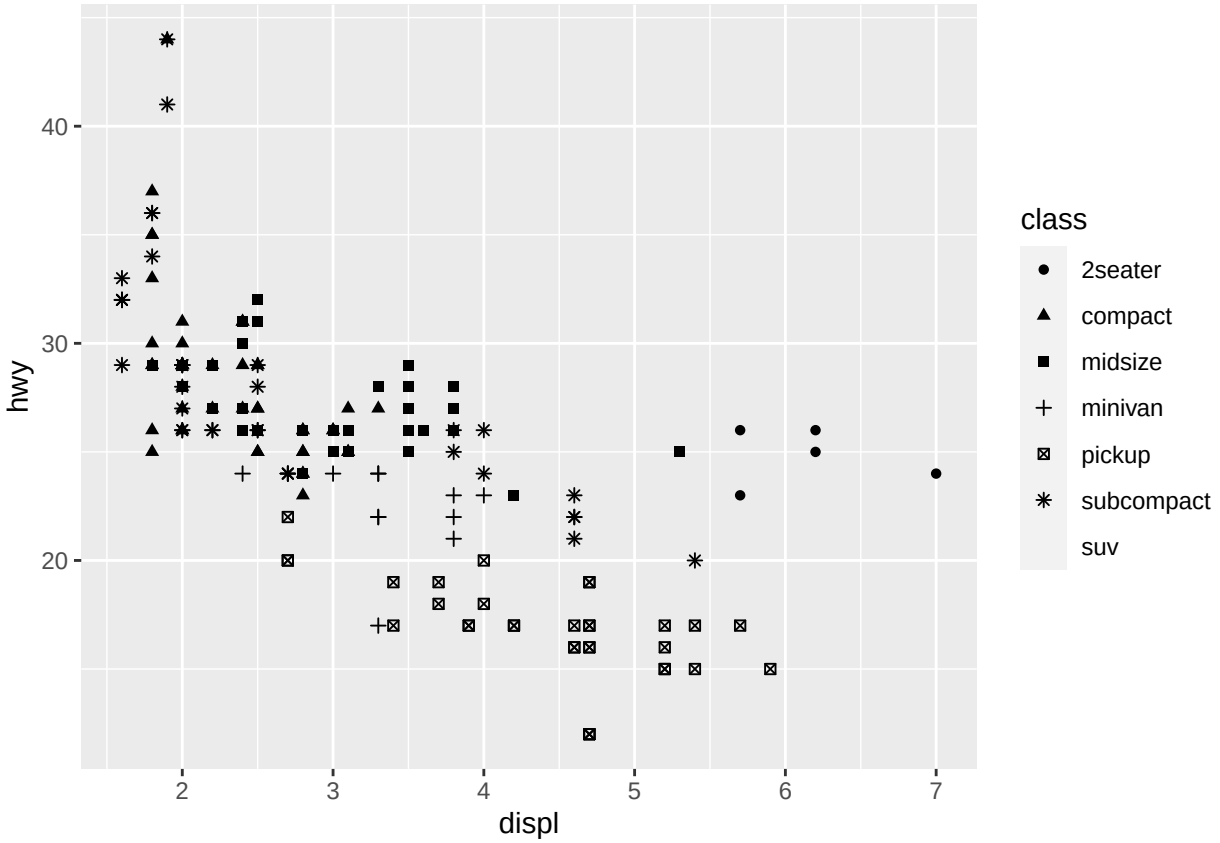
```
## Warning: Using alpha for a discrete variable is not advised.
```




```
# Bottom
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, shape = class))
```

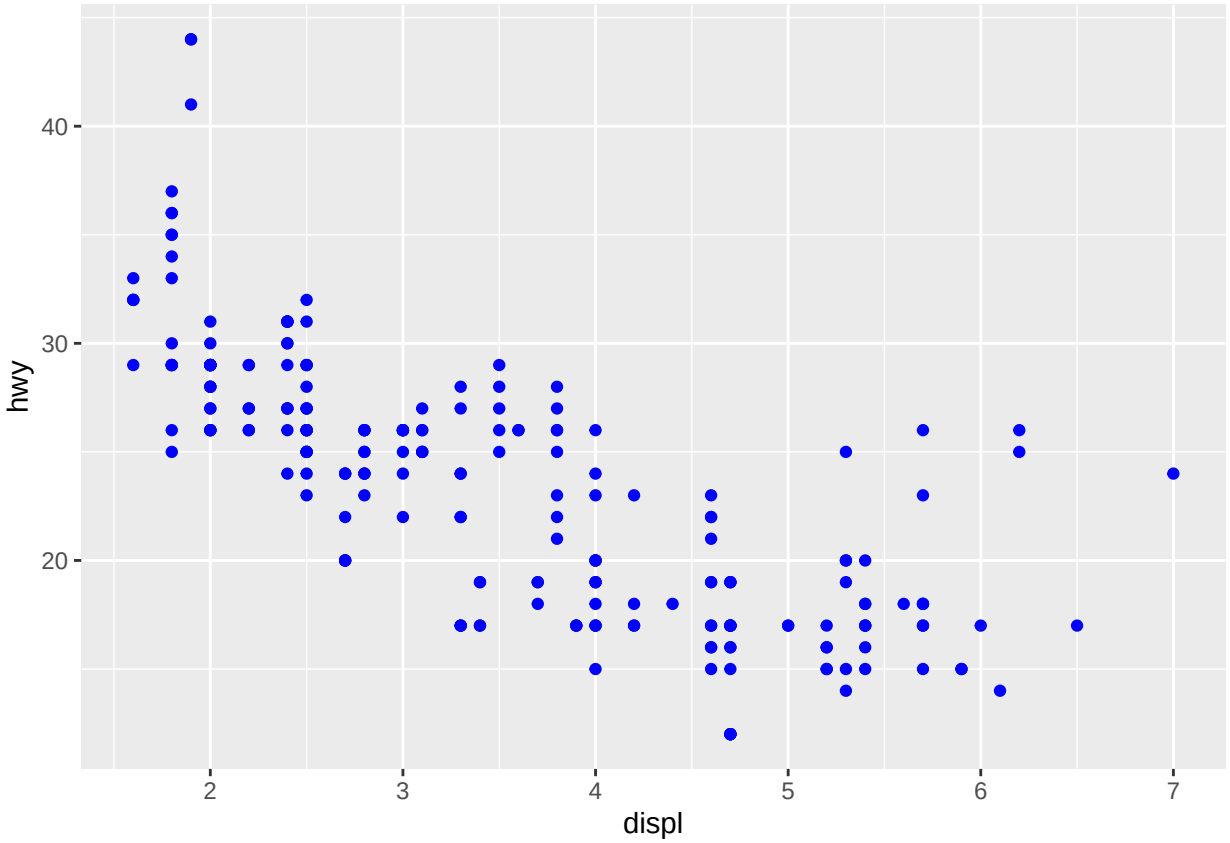
```
## Warning: The shape palette can deal with a maximum of 6 discrete values because
## more than 6 becomes difficult to discriminate; you have 7. Consider
## specifying shapes manually if you must have them.
```

```
## Warning: Removed 62 rows containing missing values (‘geom_point()‘).
```



Coloring all the point

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



This is not same as before. It colored all the points but the previous one just color with respect to the class

Excercises

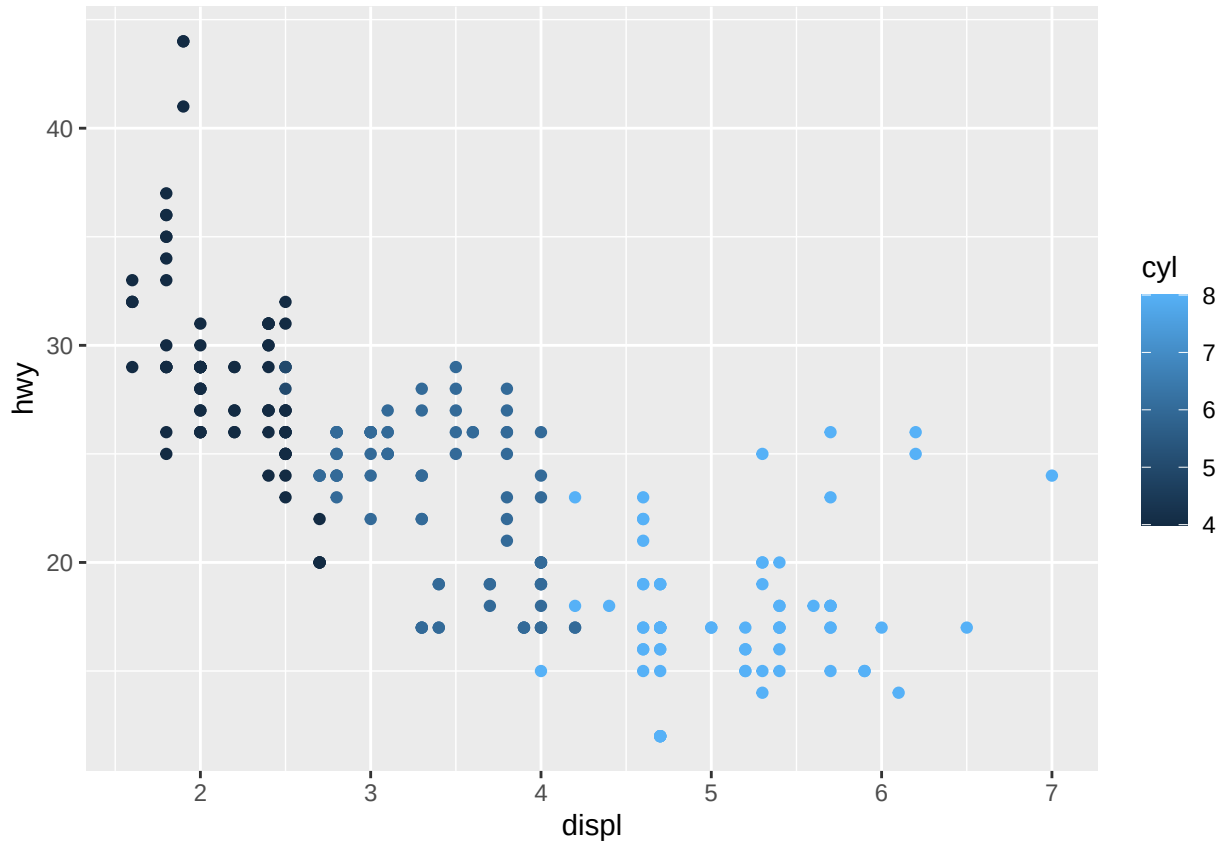
1. Colored is defined under the aesthetic function.
2.

```
str(mpg)
```

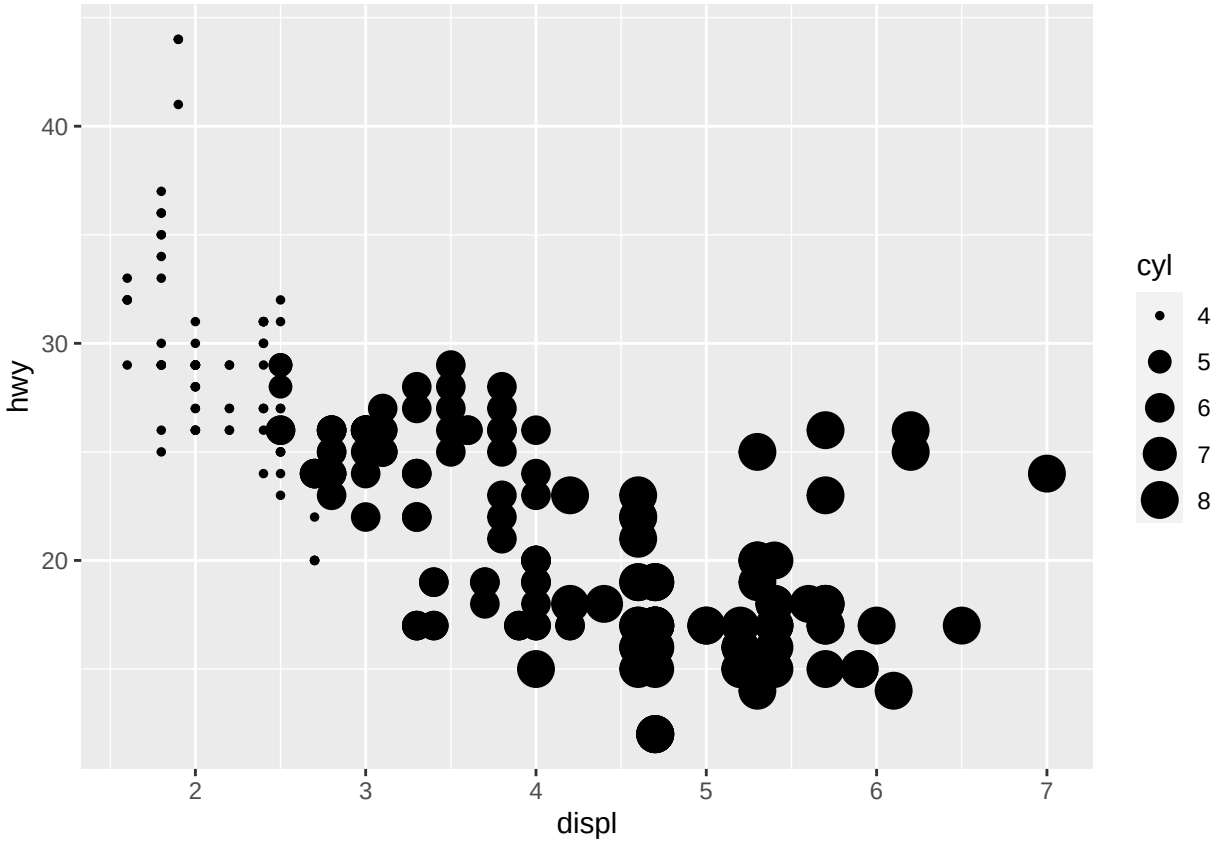
```
## tibble [234 x 11] (S3: tbl_df/tbl/data.frame)
##  $ manufacturer: chr [1:234] "audi" "audi" "audi" "audi" ...
##  $ model       : chr [1:234] "a4" "a4" "a4" "a4" ...
##  $ displ       : num [1:234] 1.8 1.8 2 2 2.8 2.8 3.1 1.8 1.8 2 ...
##  $ year        : int [1:234] 1999 1999 2008 2008 1999 1999 2008 1999 1999 2008 ...
##  $ cyl         : int [1:234] 4 4 4 4 6 6 6 4 4 4 ...
##  $ trans       : chr [1:234] "auto(l5)" "manual(m5)" "manual(m6)" "auto(av)" ...
##  $ drv         : chr [1:234] "f" "f" "f" "f" ...
##  $ cty         : int [1:234] 18 21 20 21 16 18 18 18 16 20 ...
##  $ hwy         : int [1:234] 29 29 31 30 26 26 27 26 25 28 ...
##  $ fl          : chr [1:234] "p" "p" "p" "p" ...
##  $ class       : chr [1:234] "compact" "compact" "compact" "compact" ...
```

3.

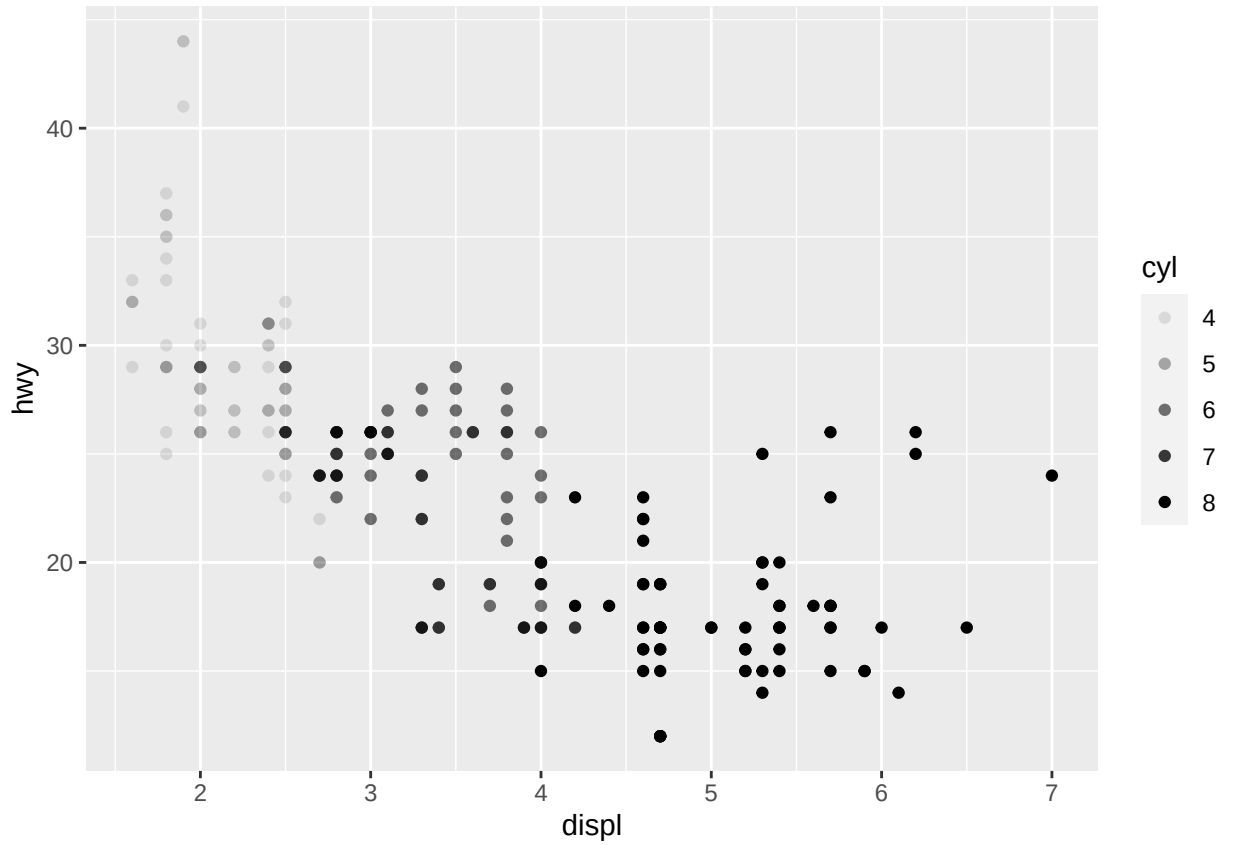
```
ggplot(data = mpg)+
  geom_point(mapping = aes(x = displ, y = hwy, color = cyl))
```




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



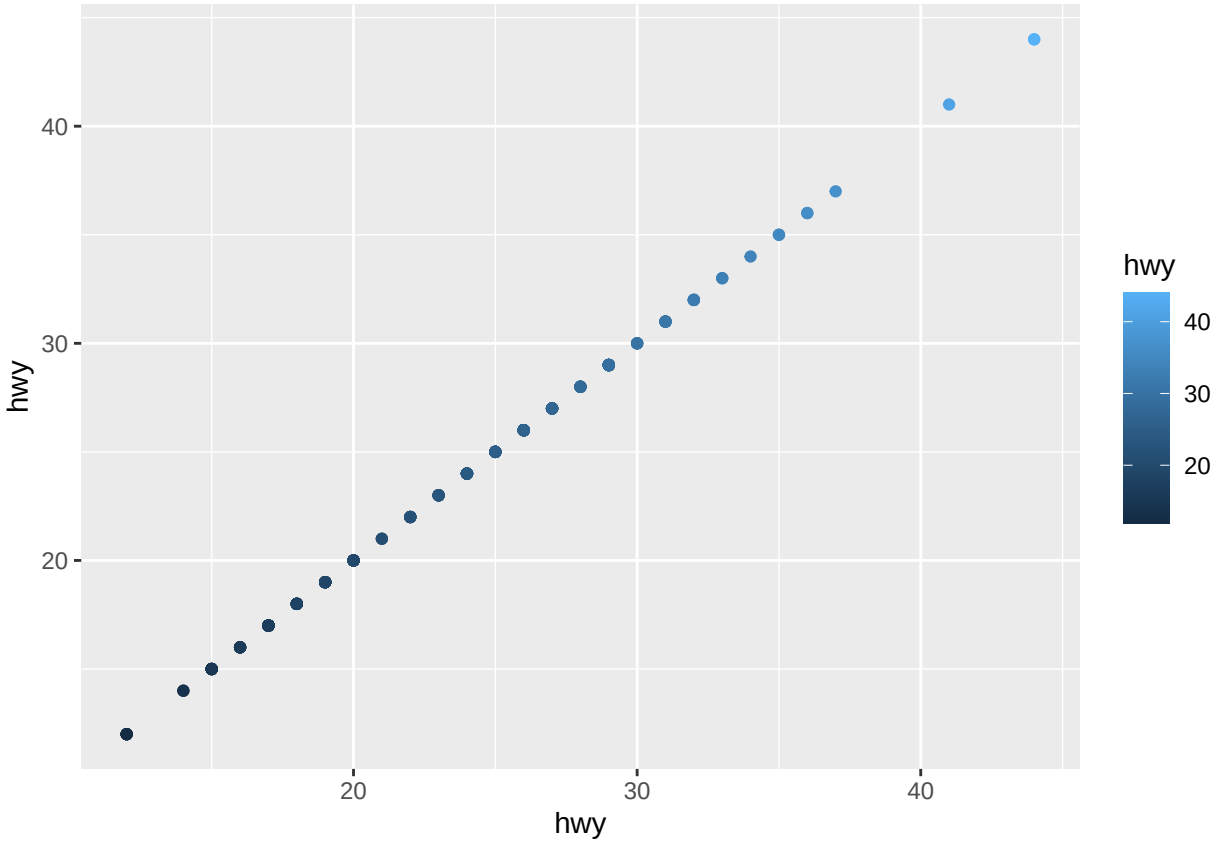

```
ggplot(data = mpg)+
  geom_point(mapping = aes(x = displ, y = hwy, alpha = cyl))
```



This changes with respect to the magnitude of the cyl.

4.

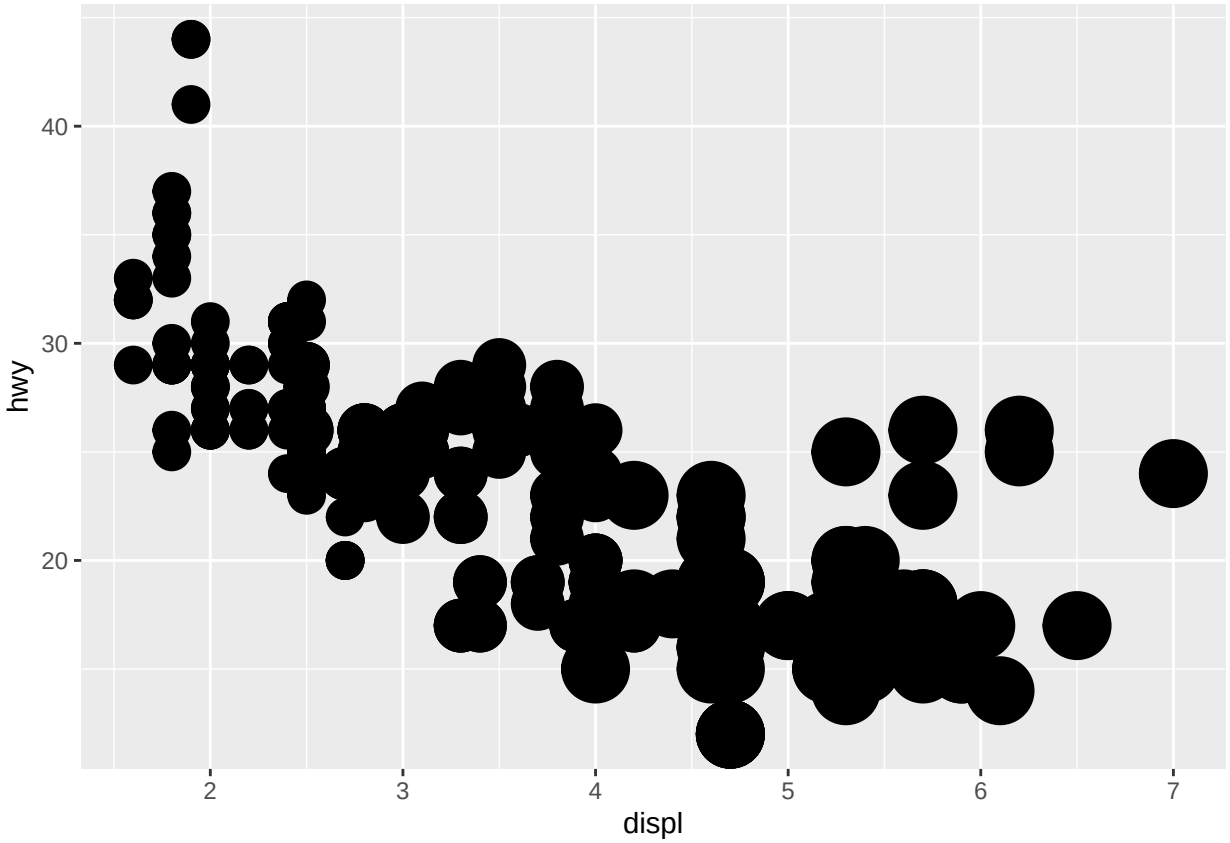
```
ggplot(data = mpg)+
  geom_point(mapping = aes(x = hwy, y = hwy, color = hwy))
```



We get a straight line with respect to different color for different magnitude

5.

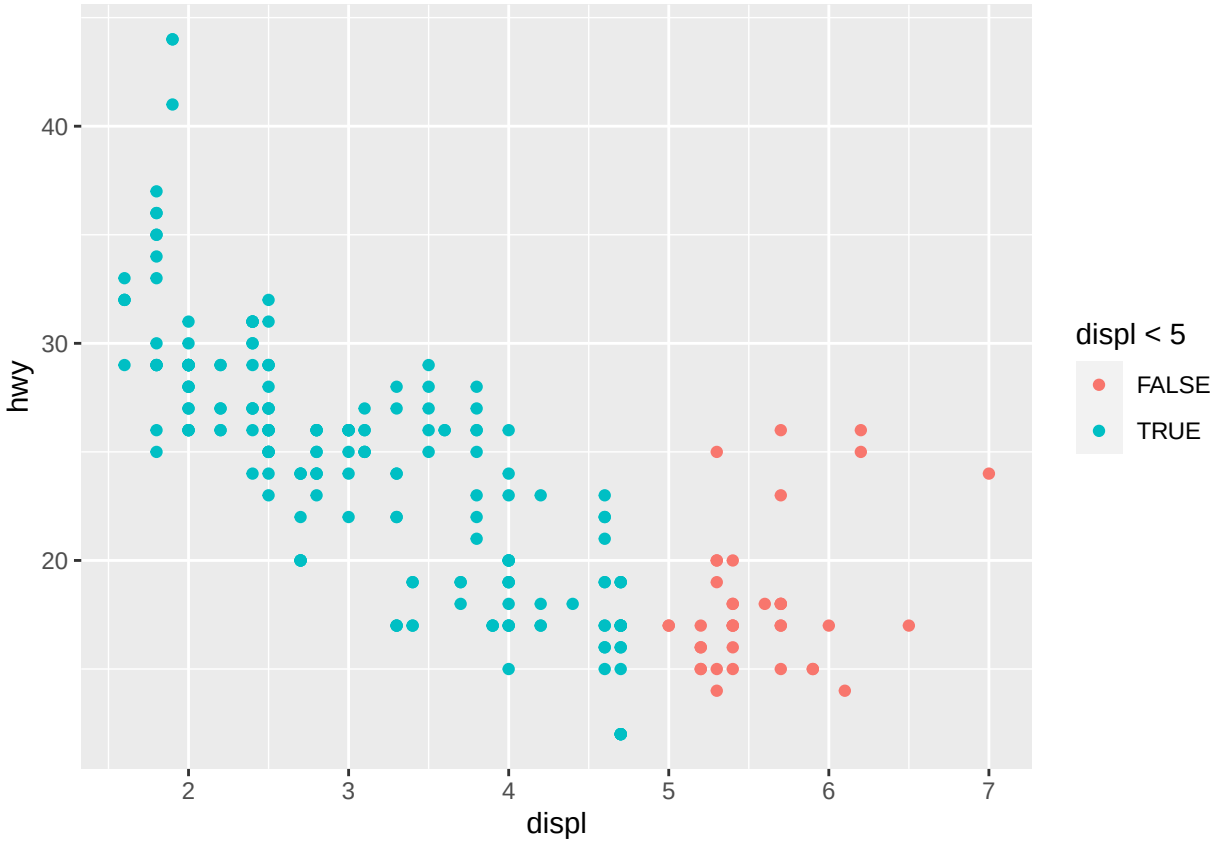
```
ggplot(data = mpg)+
  geom_point(mapping = aes(x = displ, y = hwy, stroke = cyl))
```



It plots everything with respect to the size of the data point.

6.

```
ggplot(data = mpg)+
  geom_point(mapping = aes(x = displ, y = hwy, color = displ < 5))
```

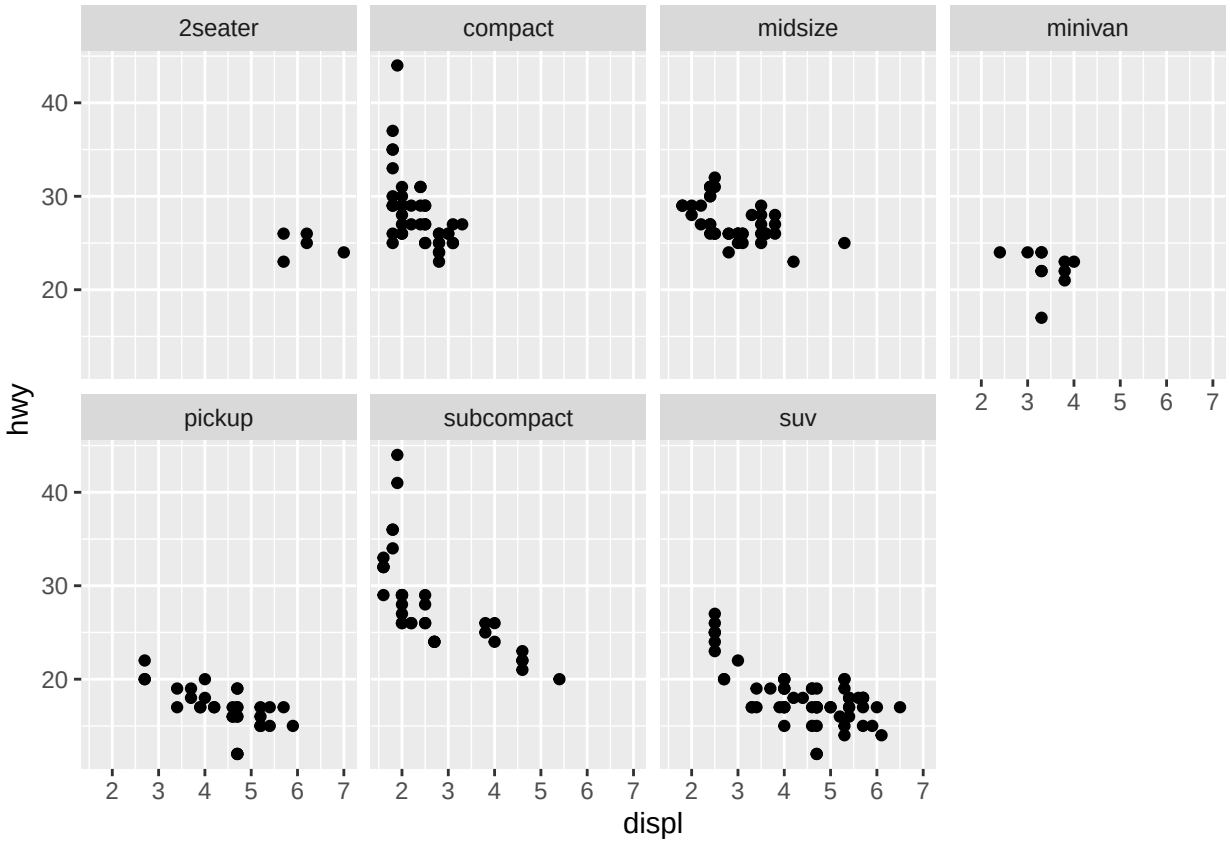


It plots everything but colored the values for which the argument is false.

Faceting

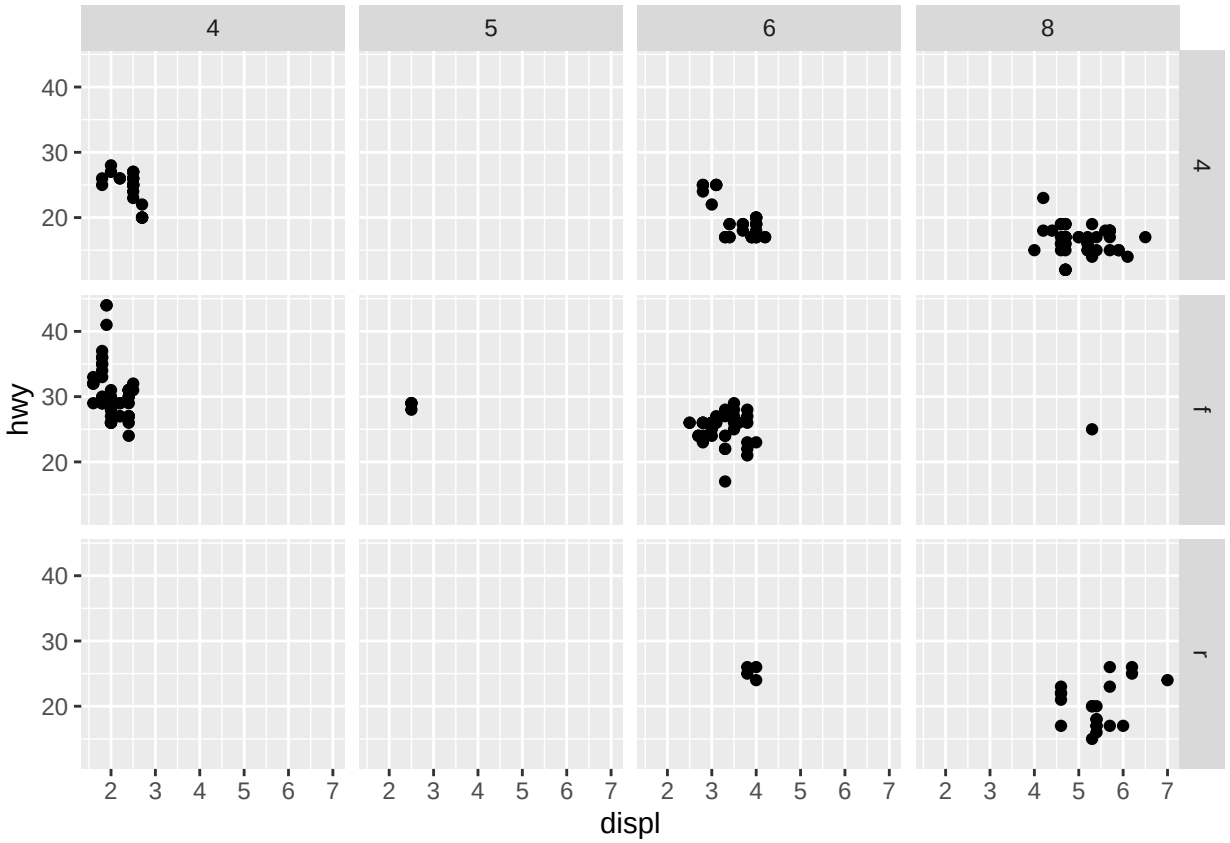
Facet wrapping

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  facet_wrap(~ class, nrow = 2)
```



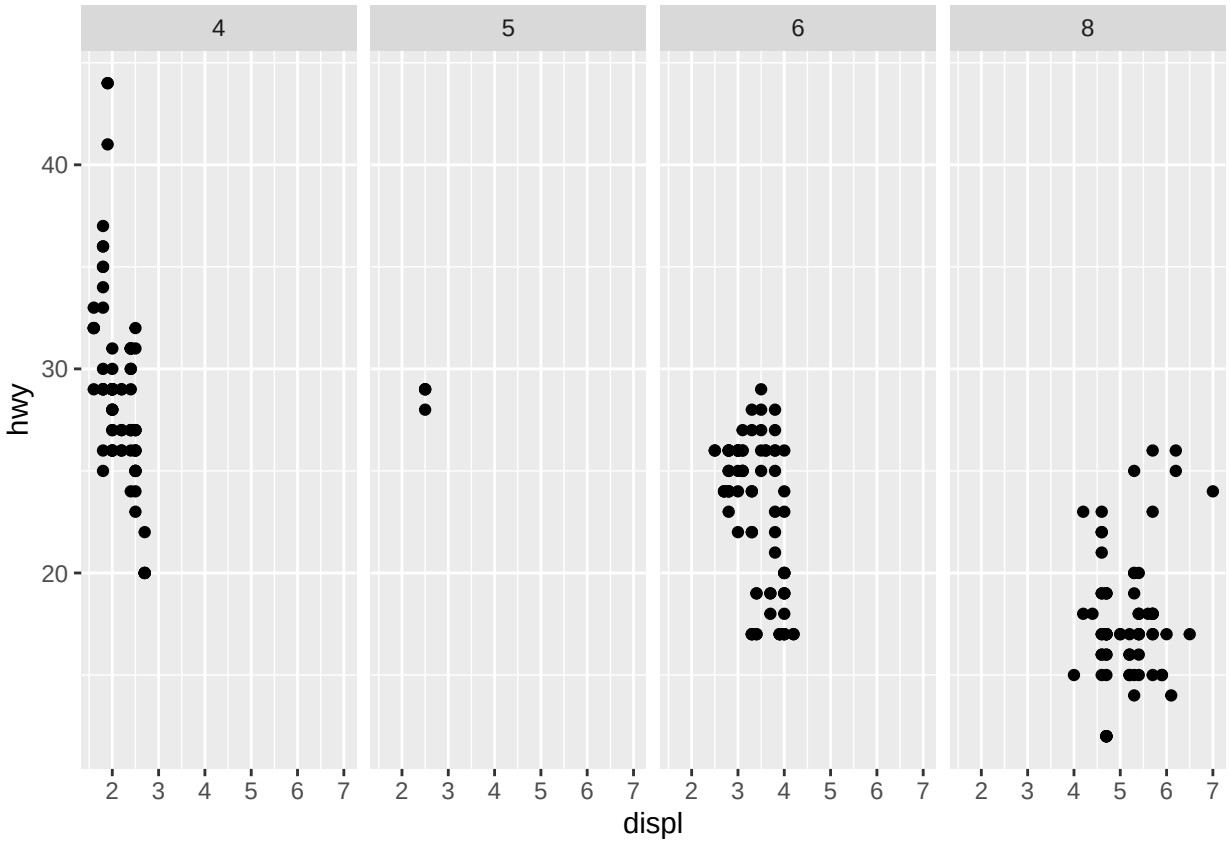
Facet griding for two variable

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



Facet gridding without specifing any column

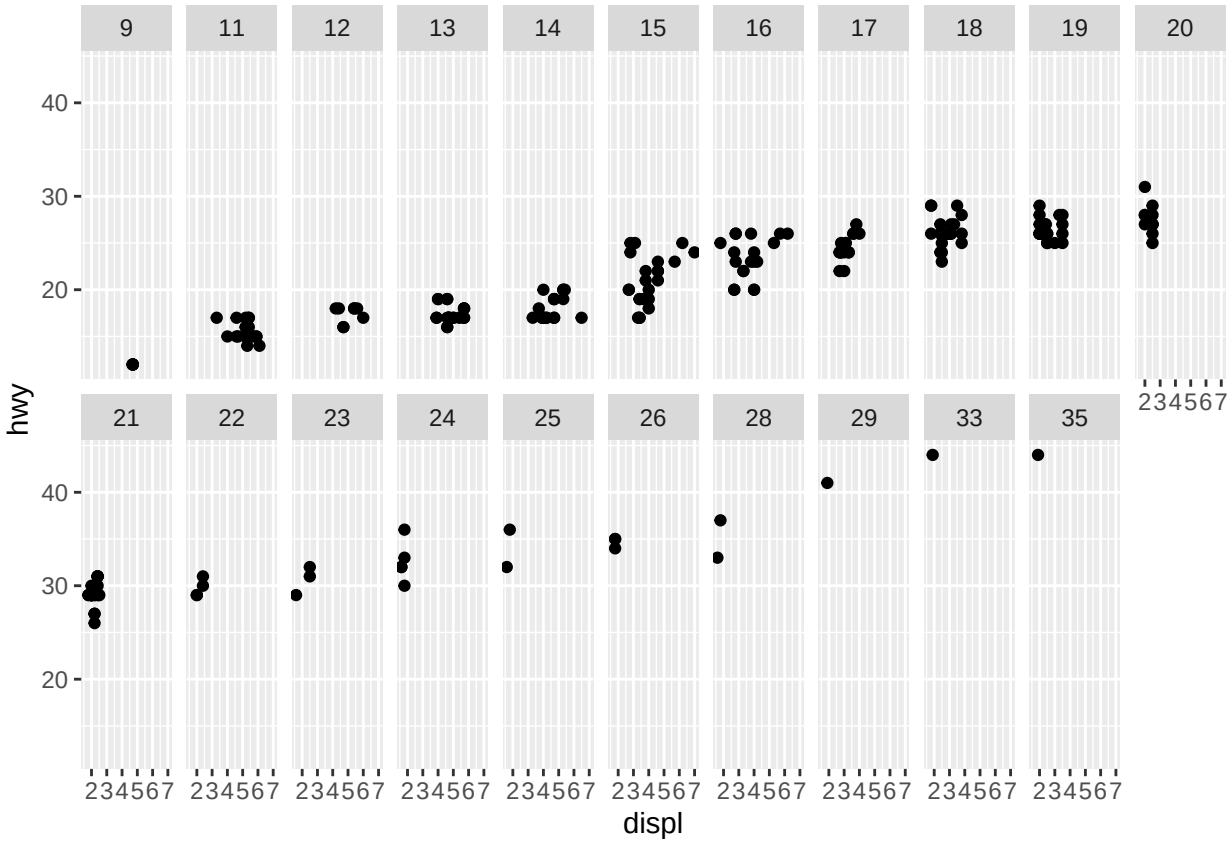
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  facet_grid(. ~ cyl)
```



Exercise

1.

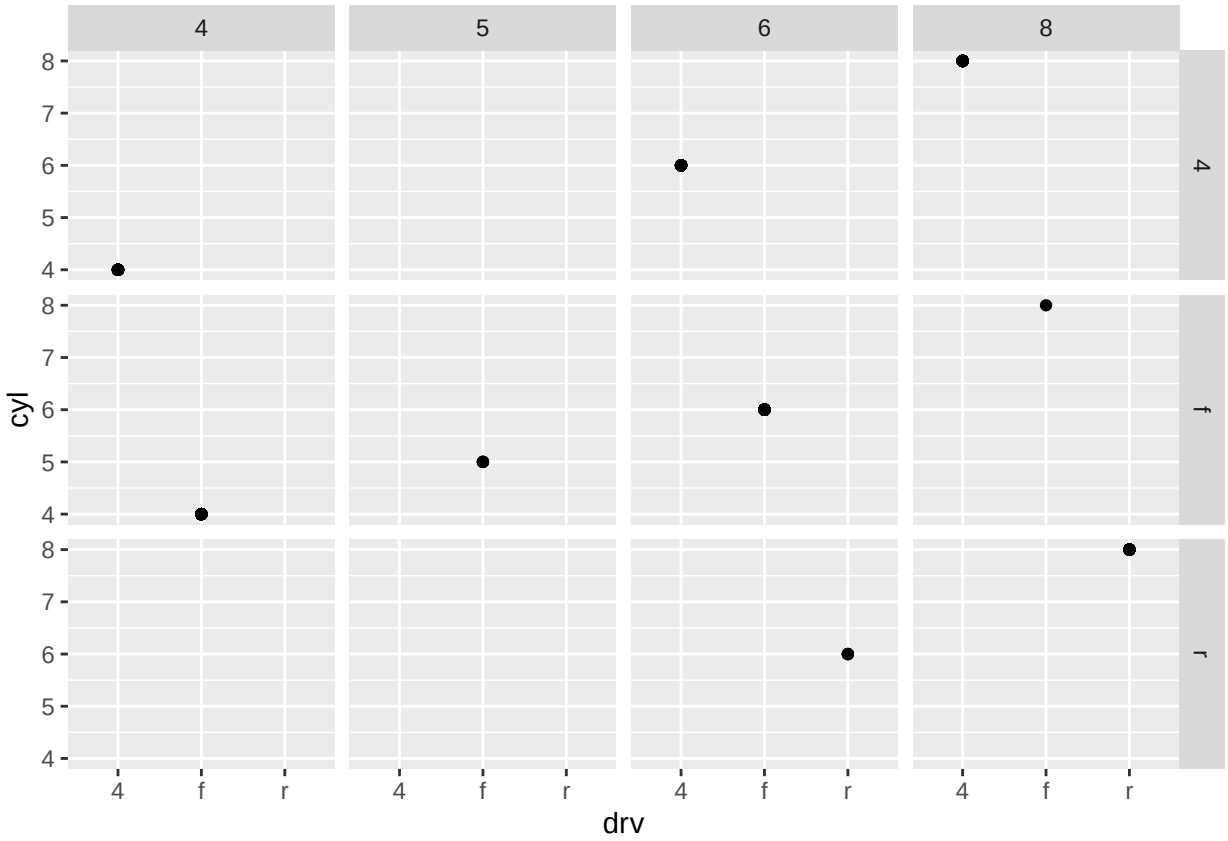
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  facet_wrap(~ cty, nrow = 2)
```



It creates the plots for every values in the variable.

2.

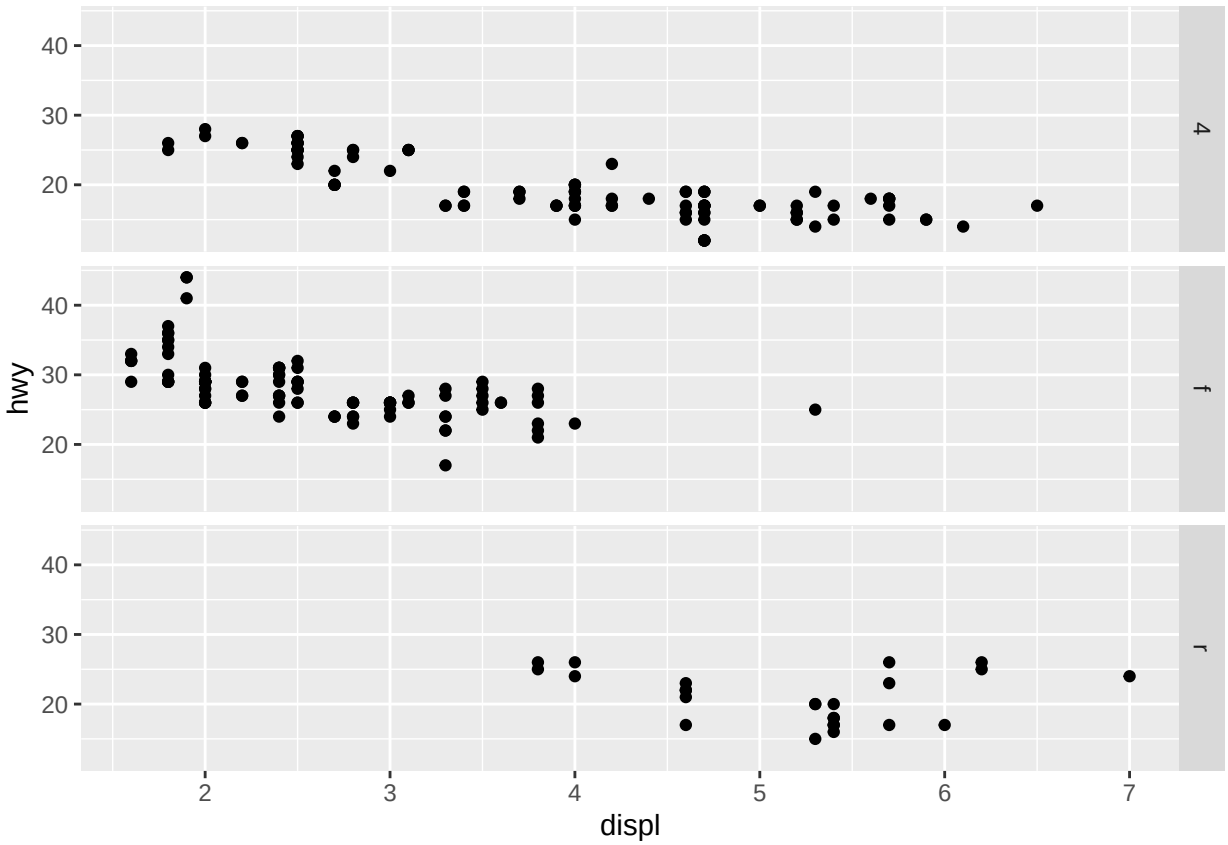
```
ggplot(data = mpg)+
  geom_point(mapping = aes(x = drv, y = cyl))+
  facet_grid(drv ~ cyl)
```



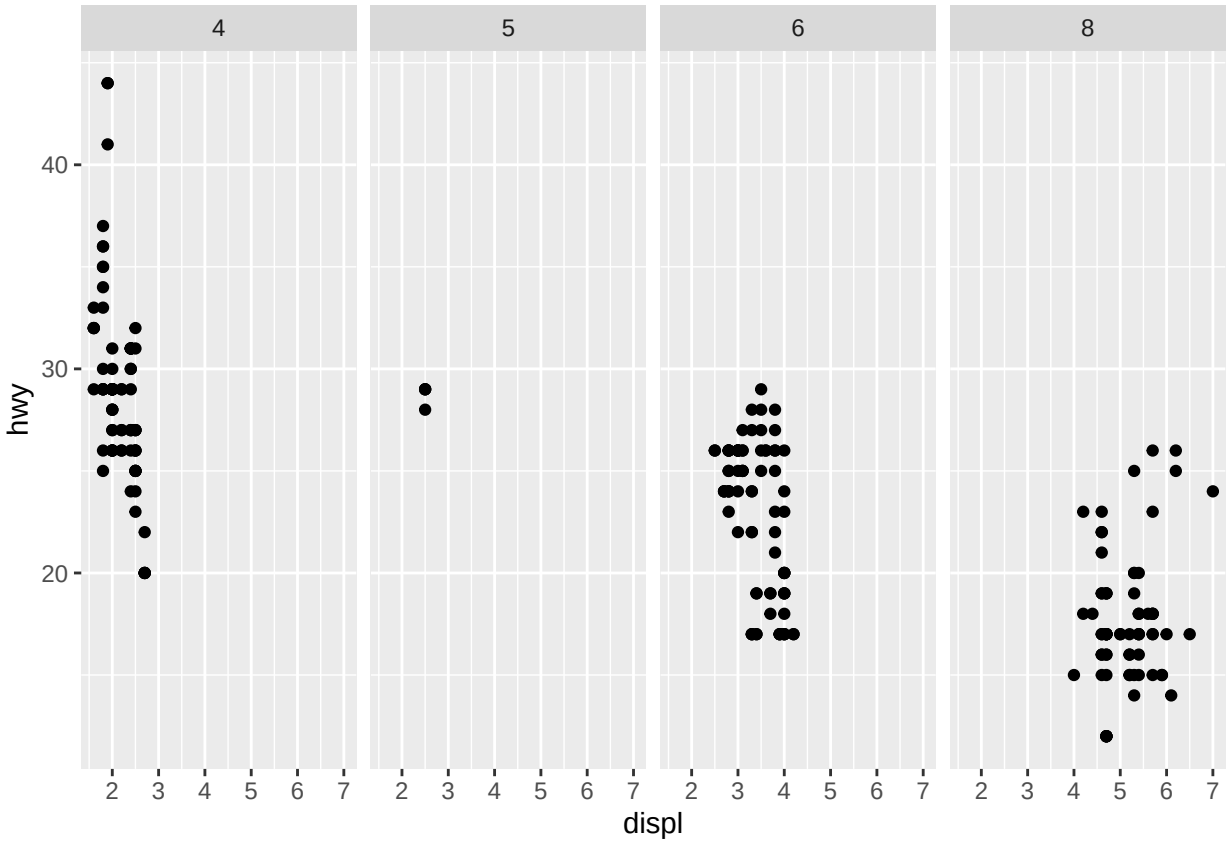
The empty set is basically the cells contains the data points

3.

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



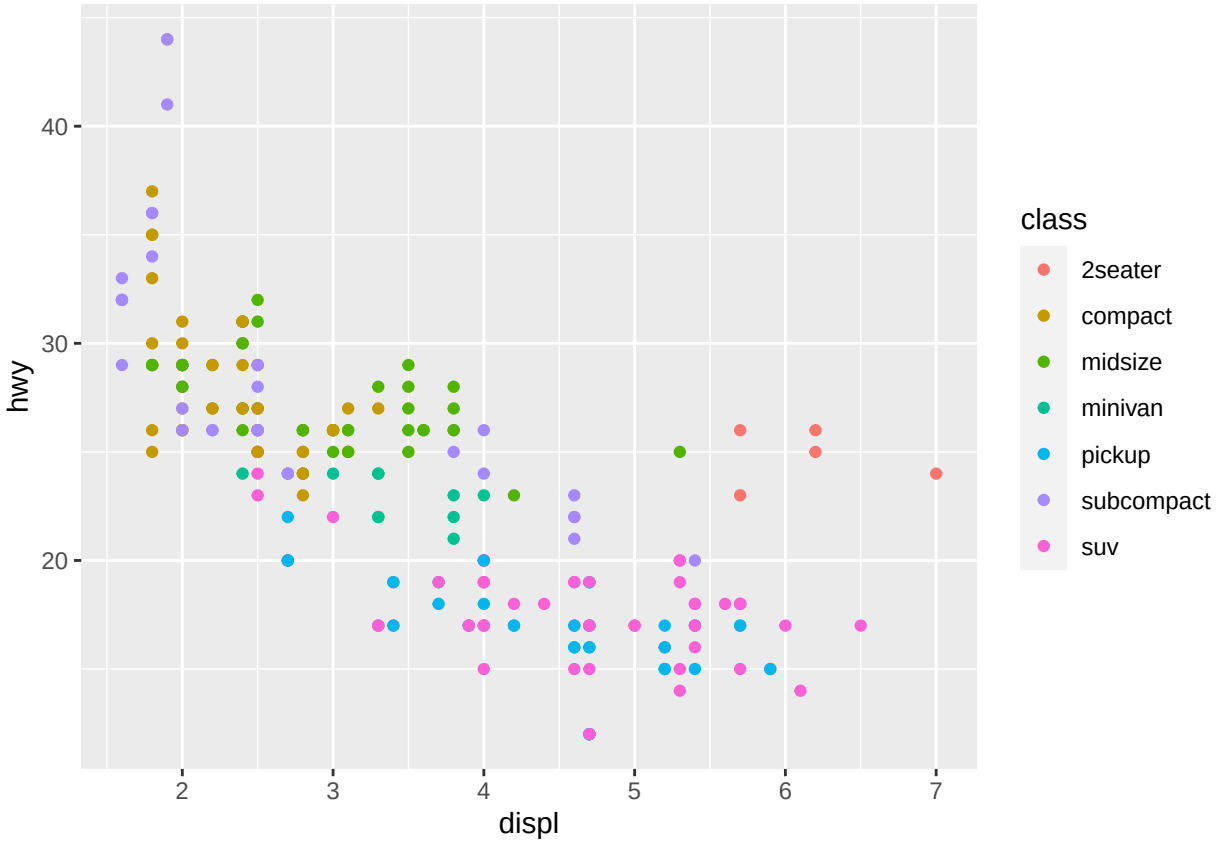

```
ggplot(data = mpg) +
geom_point(mapping = aes(x = displ, y = hwy)) +
facet_grid(. ~ cyl)
```



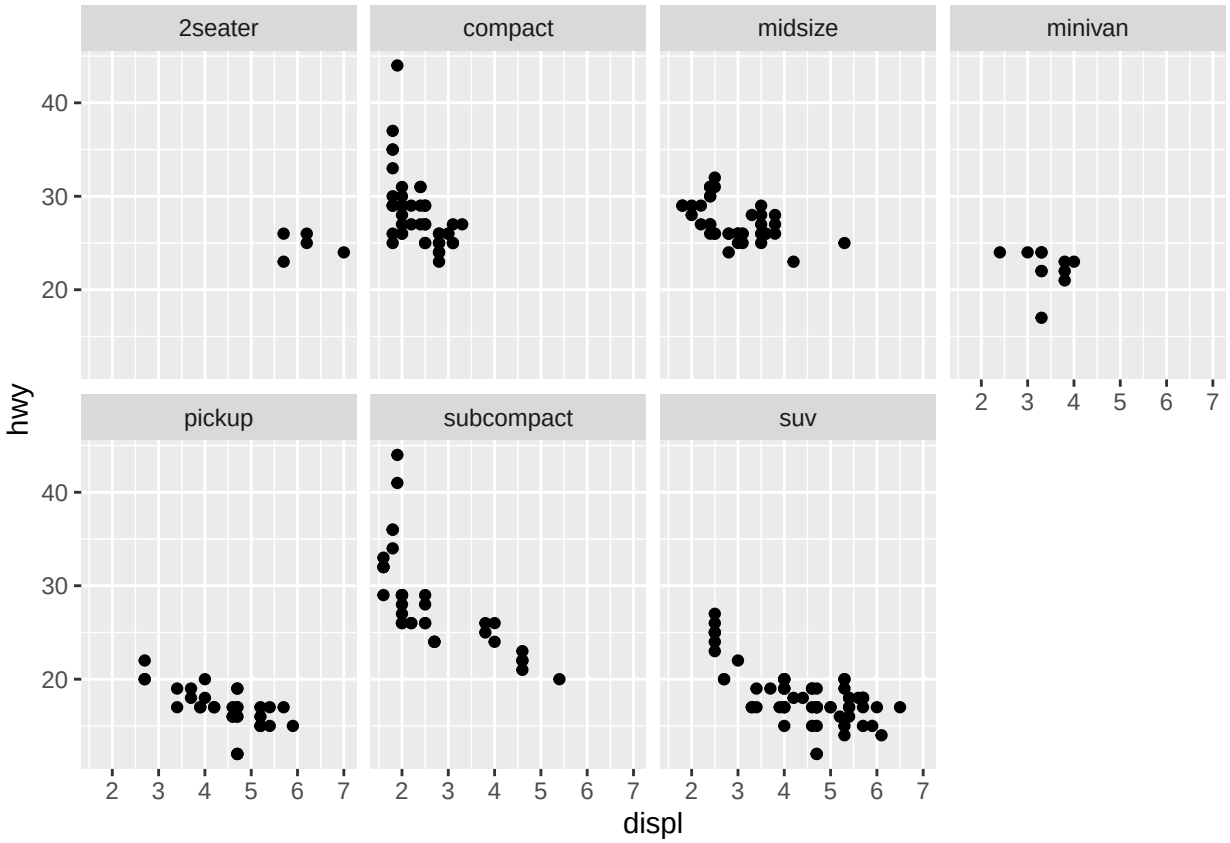
First one facets the graph with respect to `drv` in a horizontal way. Others is the opposite and it facets the graph in the vertical way with corresponding to the values.

4.

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



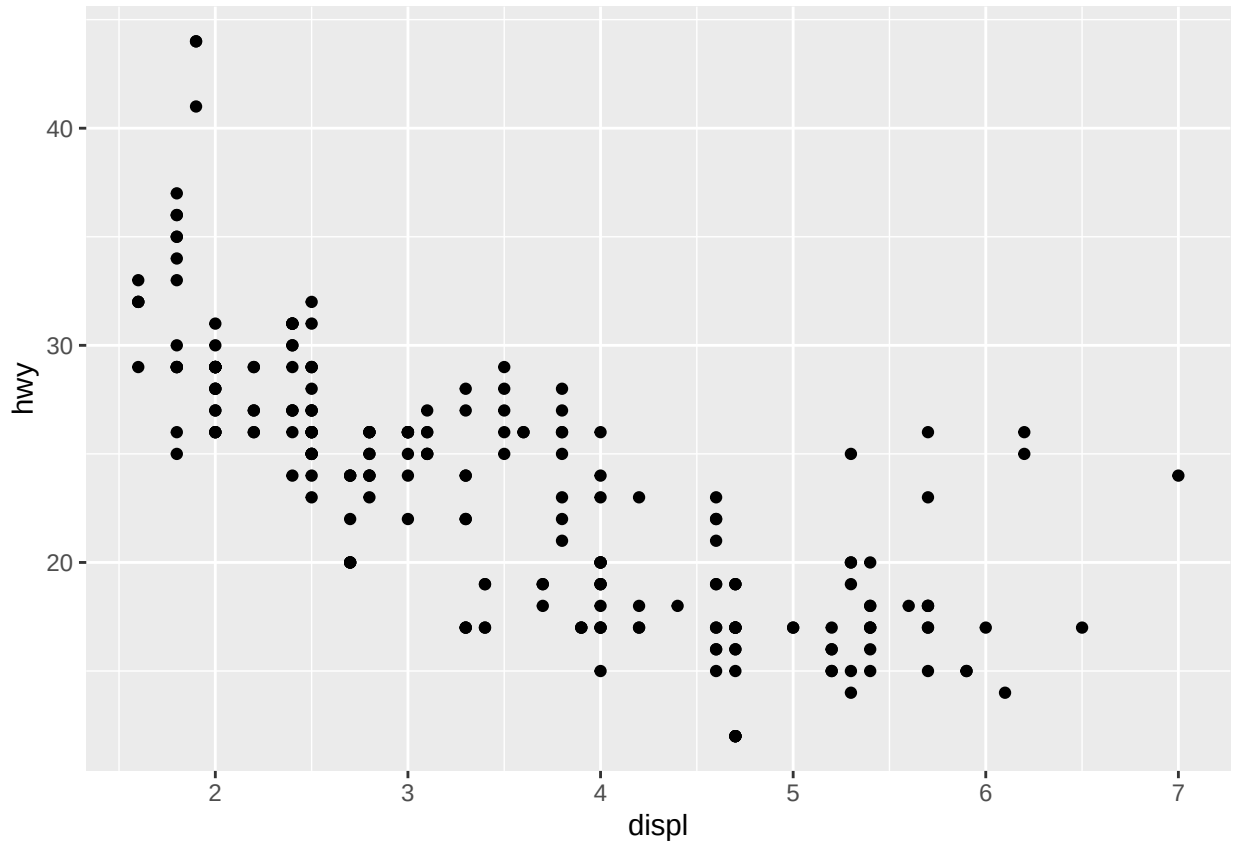

```
ggplot(data = mpg) +
geom_point(mapping = aes(x = displ, y = hwy)) +
facet_wrap(~ class, nrow = 2)
```



First graph colors the point with respect to the class and the second one divide the graphs with respect to the class. For big data I would recomend using faceting.

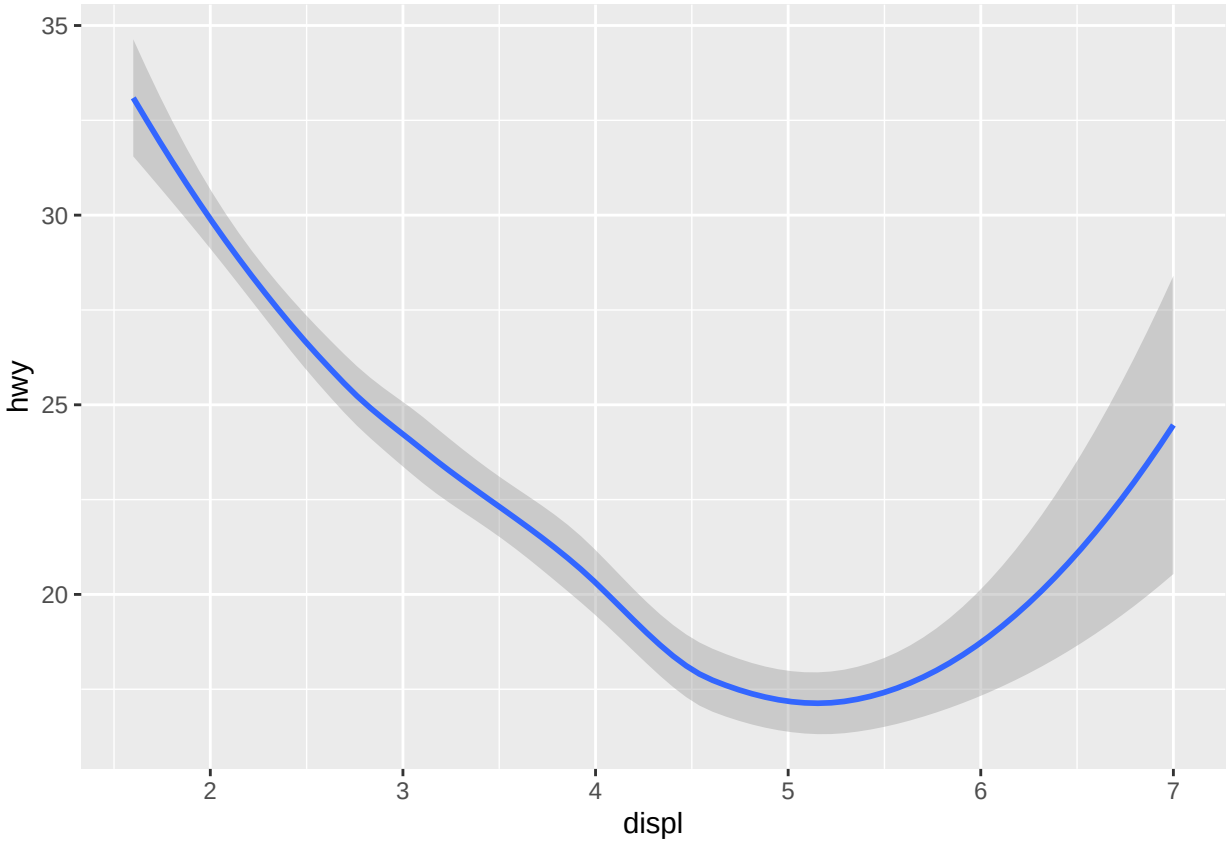
Geometric Objects

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



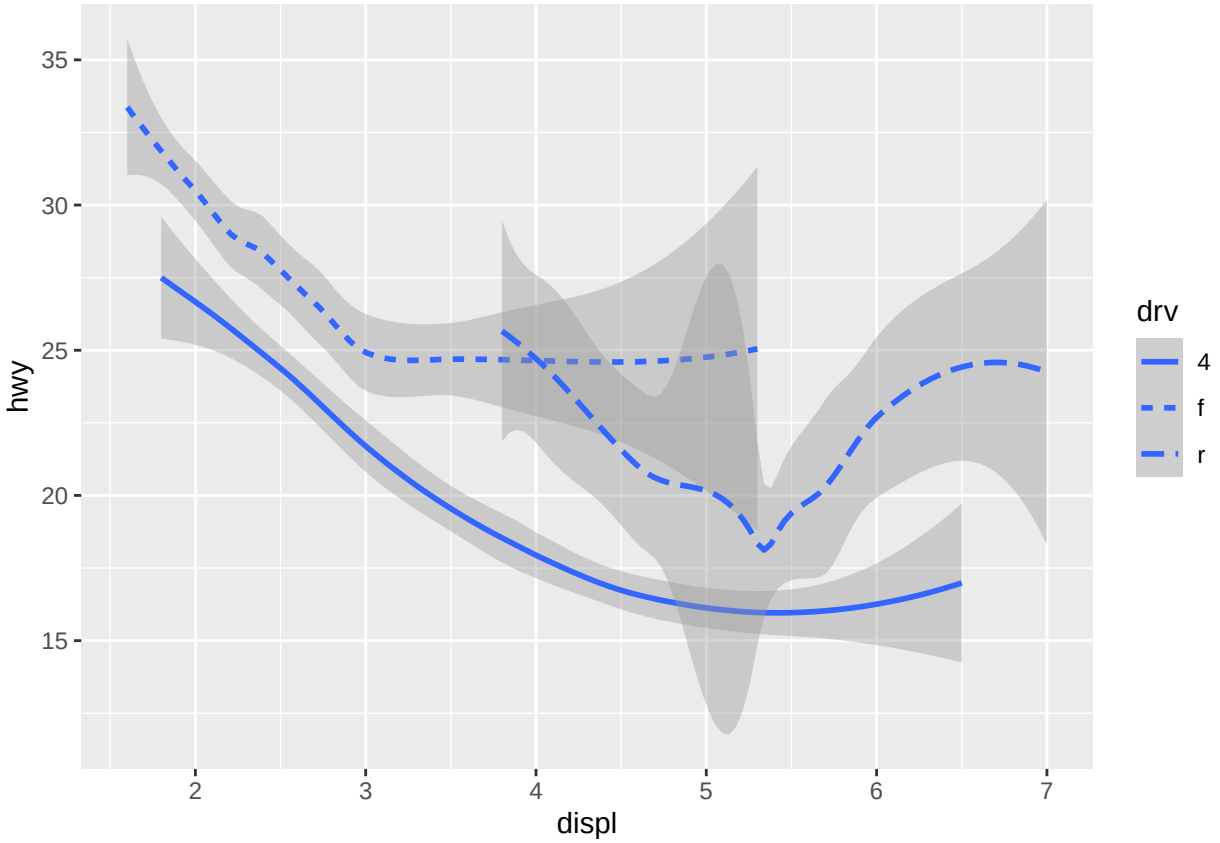

```
# right
ggplot(data = mpg) +
geom_smooth(mapping = aes(x = displ, y = hwy))
```

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



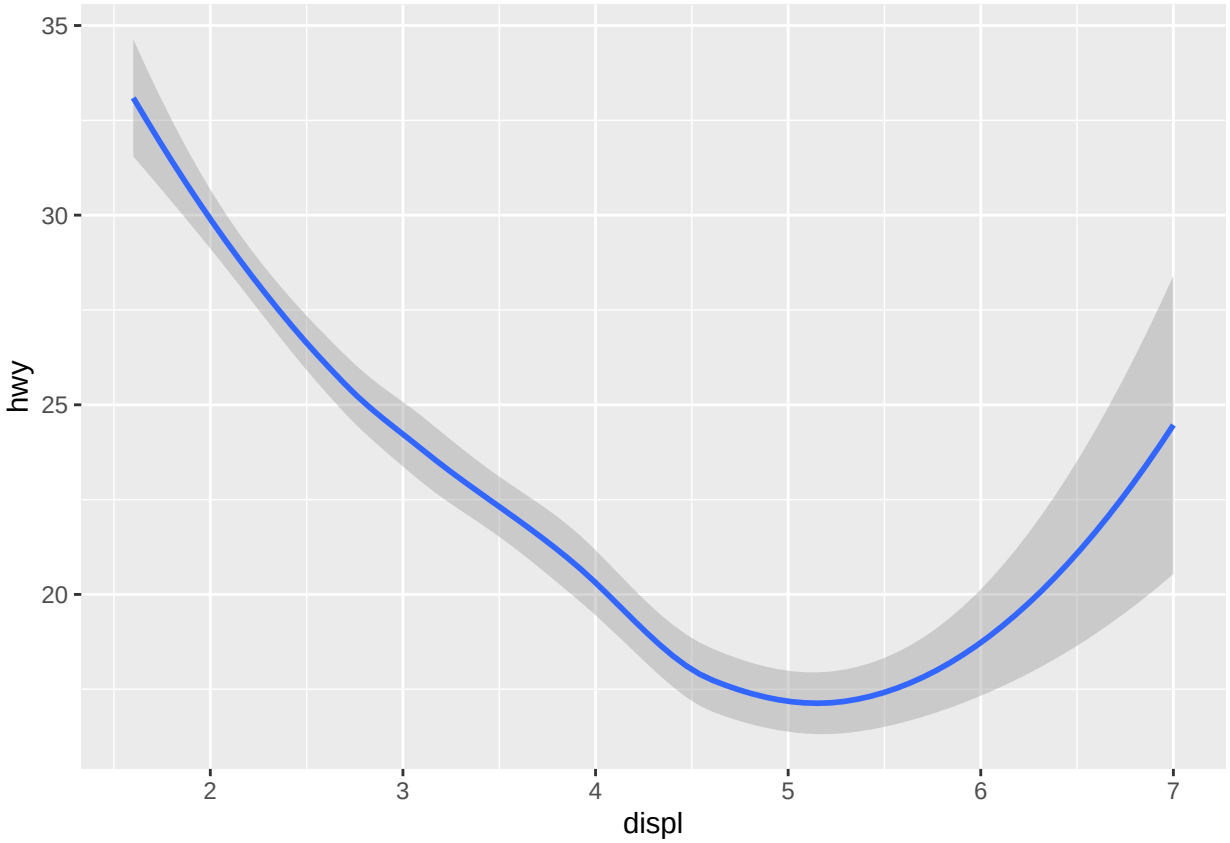

```
# right
ggplot(data = mpg) +
geom_smooth(mapping = aes(x = displ, y = hwy, linetype = drv))
```

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



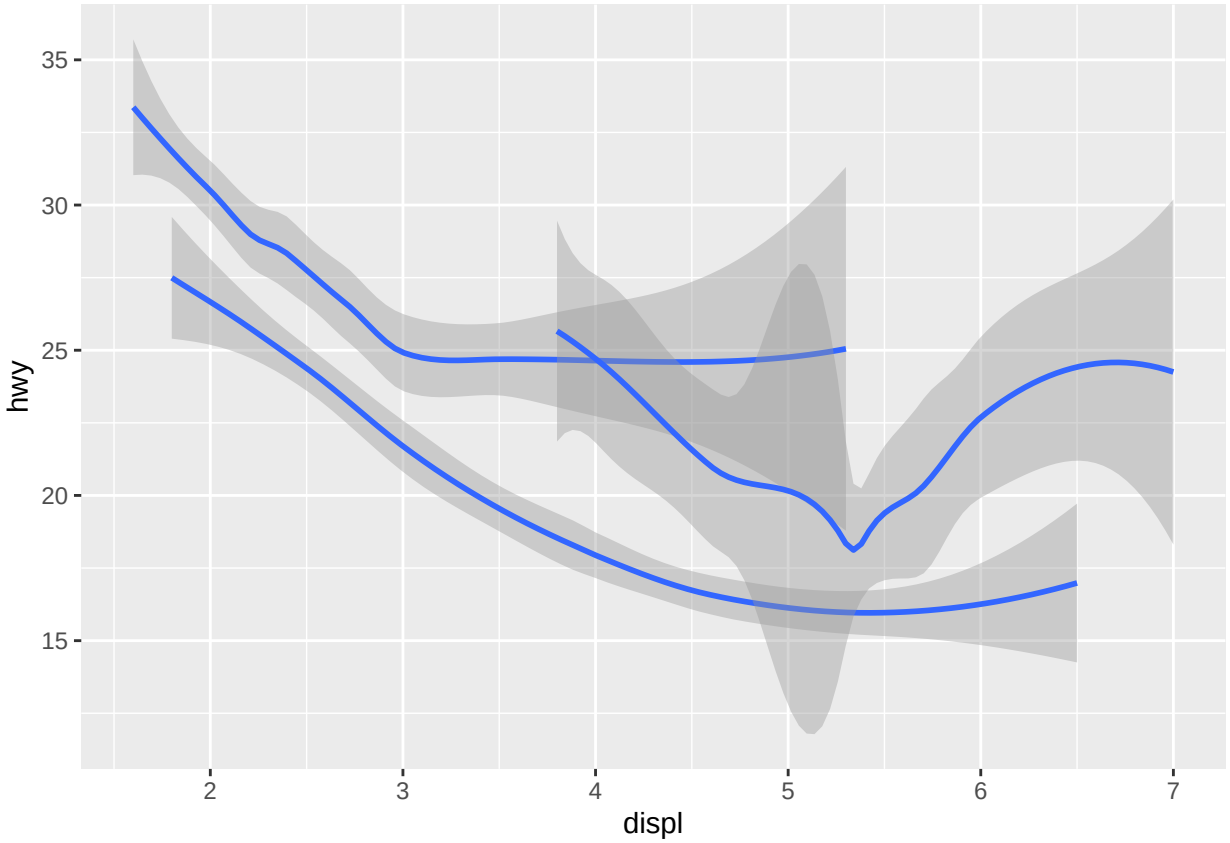

```
ggplot(data = mpg) +
geom_smooth(mapping = aes(x = displ, y = hwy))
```

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



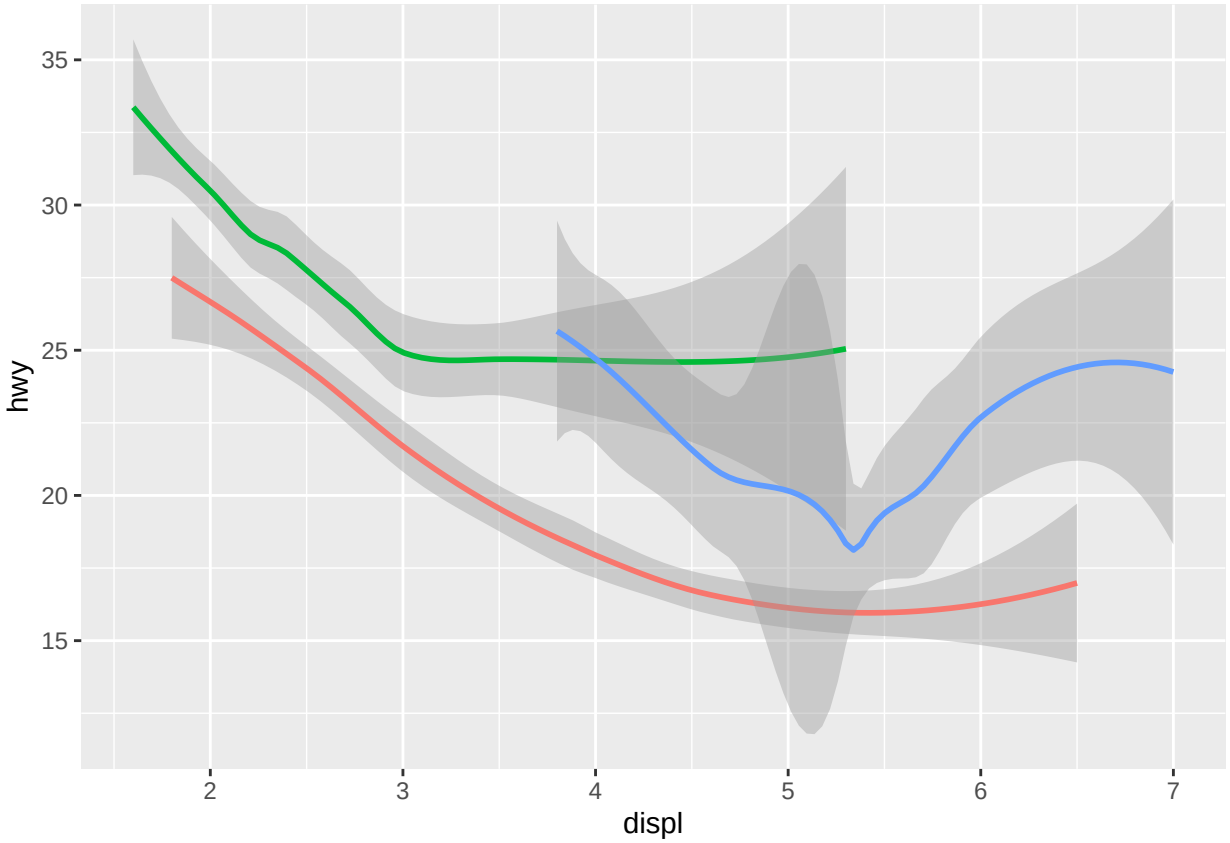

```
ggplot(data = mpg) +
geom_smooth(mapping = aes(x = displ, y = hwy, group = drv))
```

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



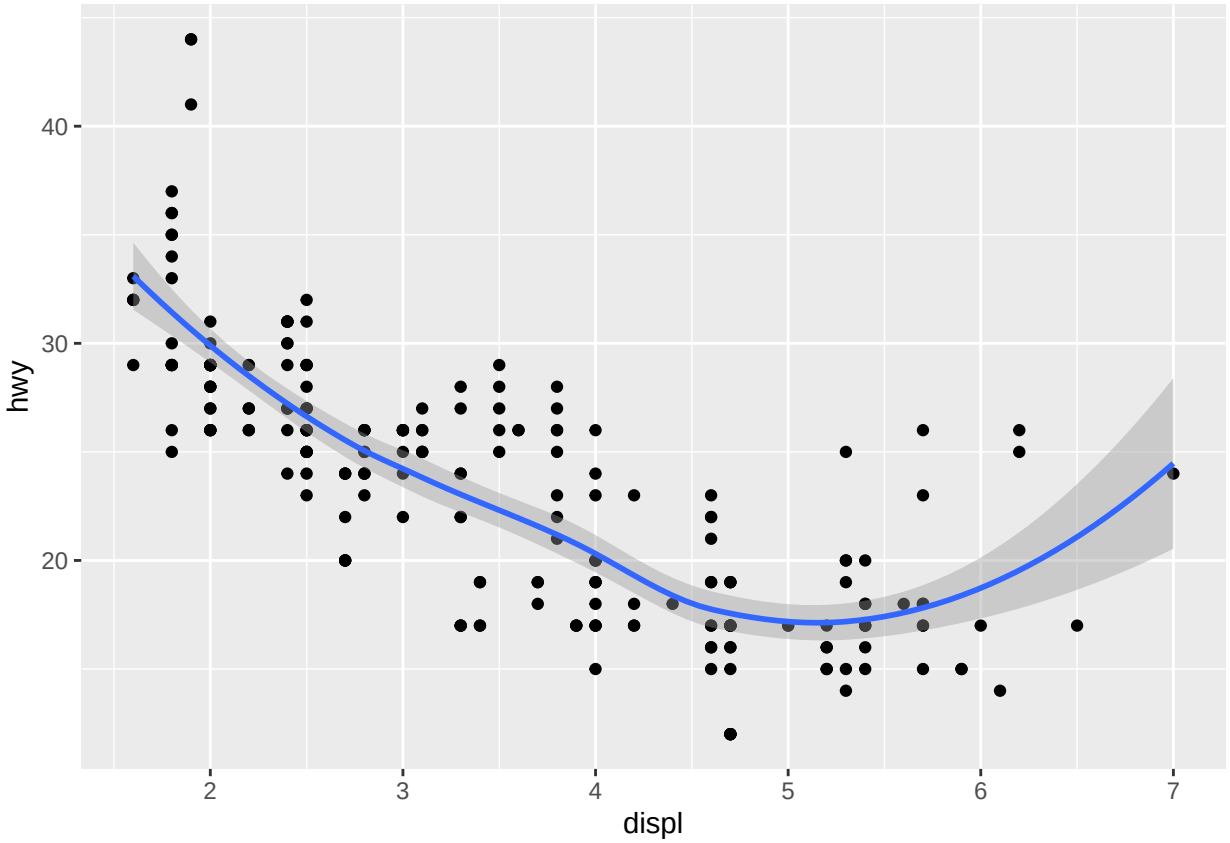

```
ggplot(data = mpg) +
geom_smooth(
mapping = aes(x = displ, y = hwy, color = drv),
show.legend = FALSE
)
```

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



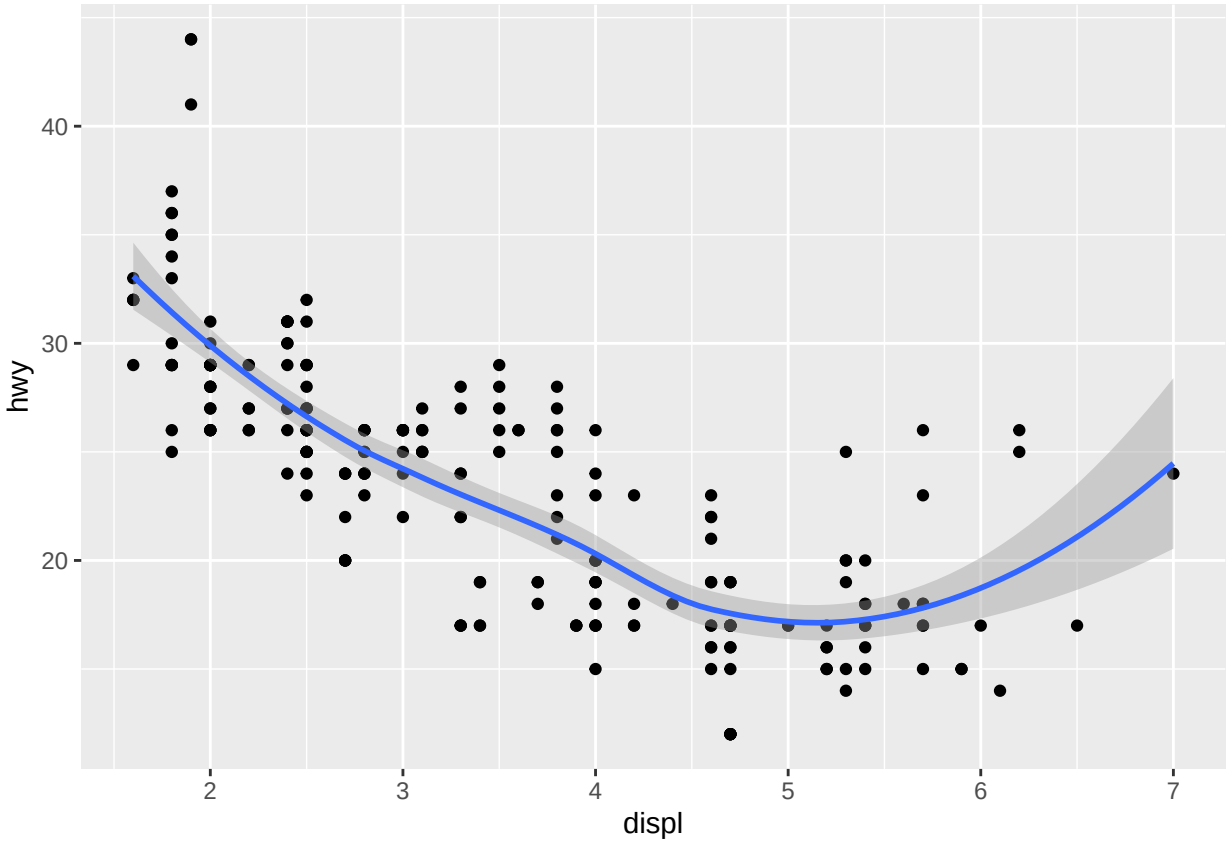

```
ggplot(data = mpg) +
geom_point(mapping = aes(x = displ, y = hwy)) +
geom_smooth(mapping = aes(x = displ, y = hwy))
```

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



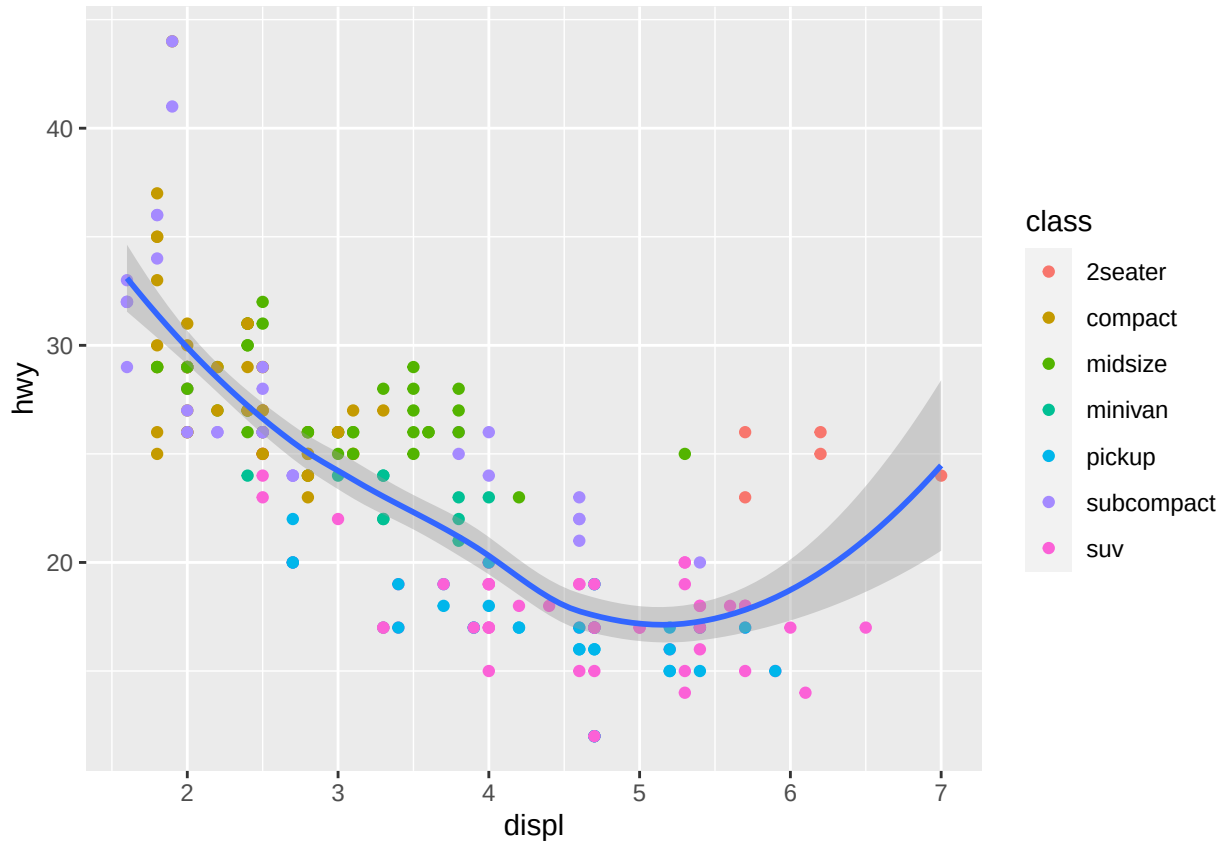

```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
geom_point() +
geom_smooth()
```

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



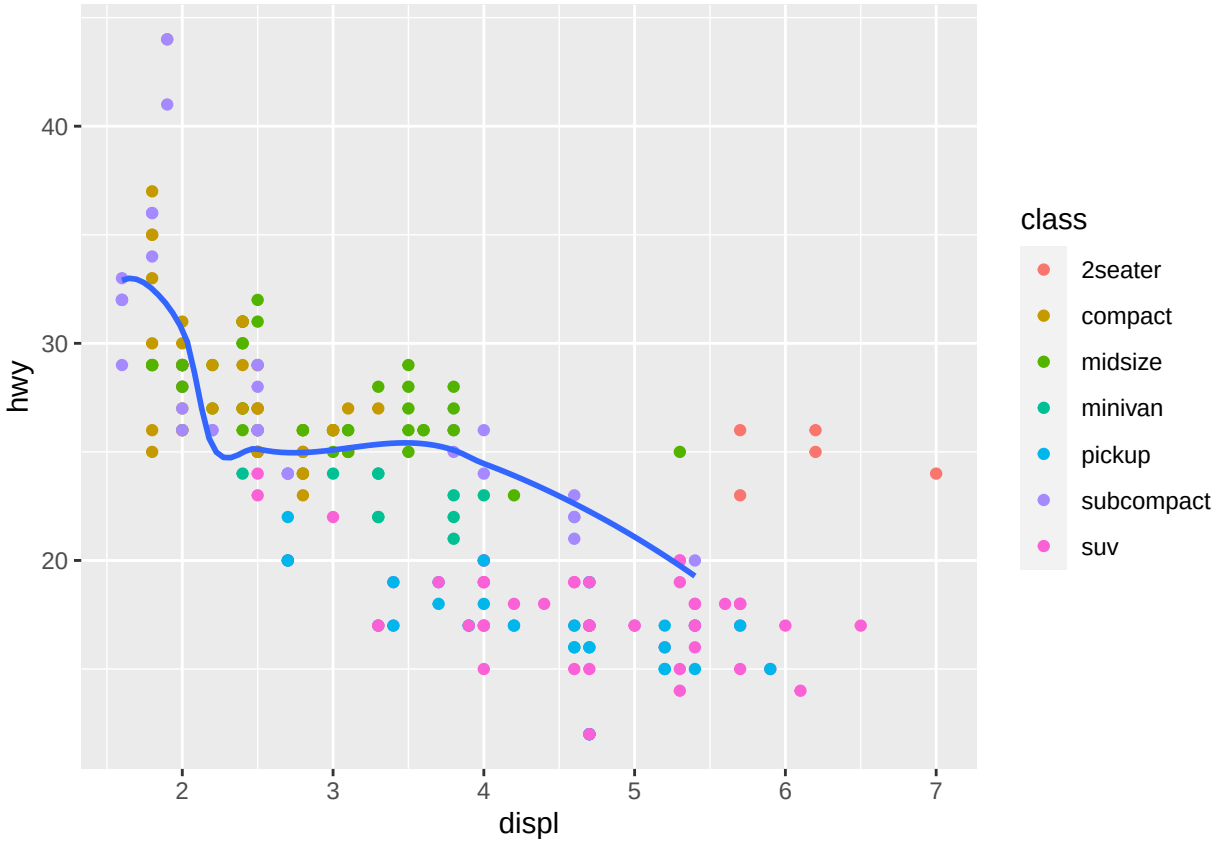

```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
geom_point(mapping = aes(color = class)) +
geom_smooth()
```

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




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

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

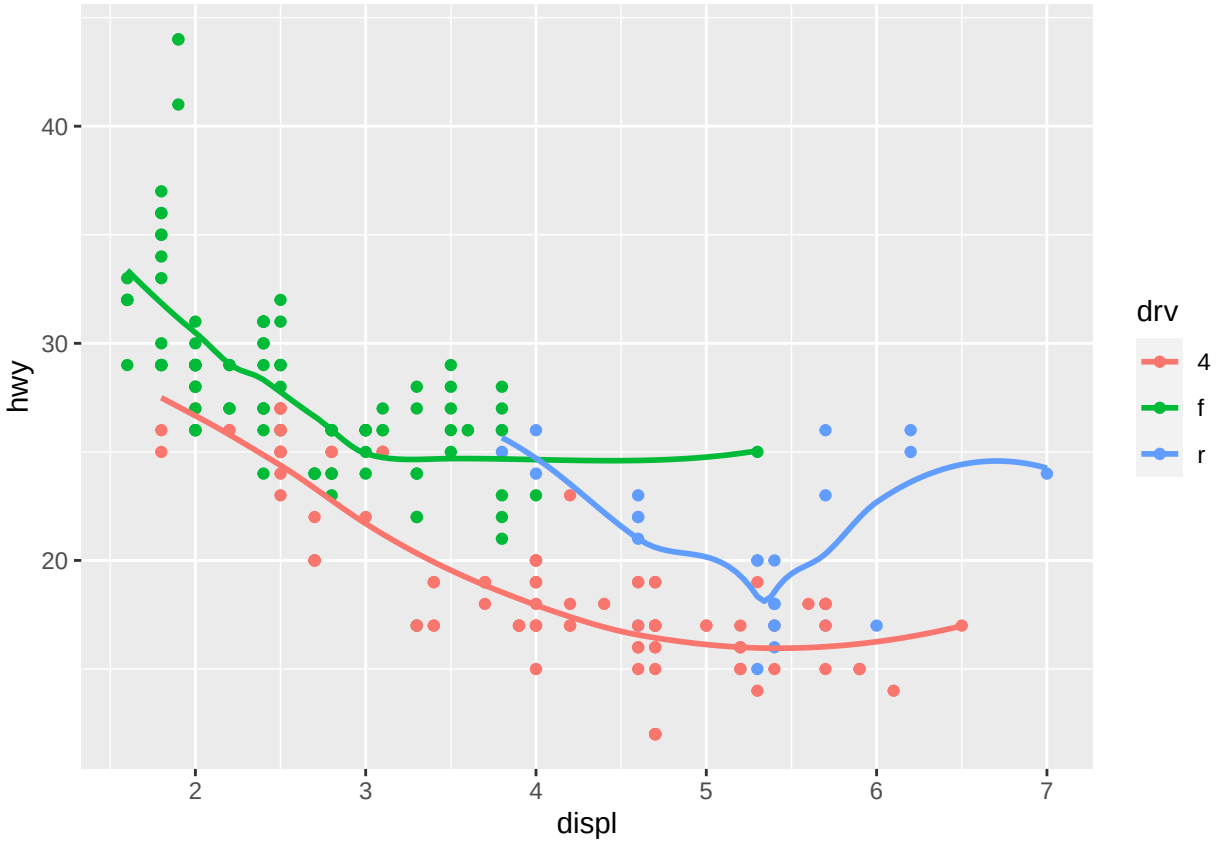


Exercises

This will create scattarplot and line with different class as a legend.

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

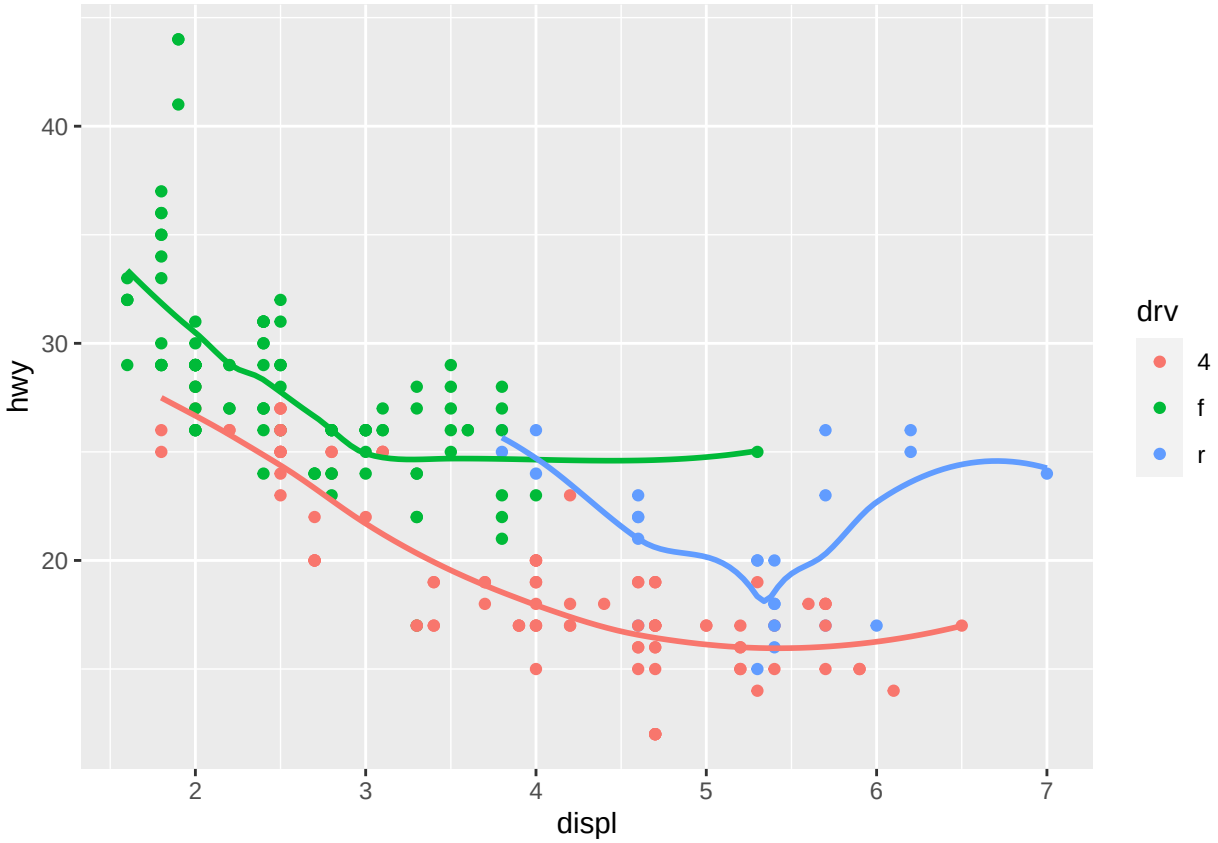
```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



3. Does not show smooth lines legend as we said it to be false.

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

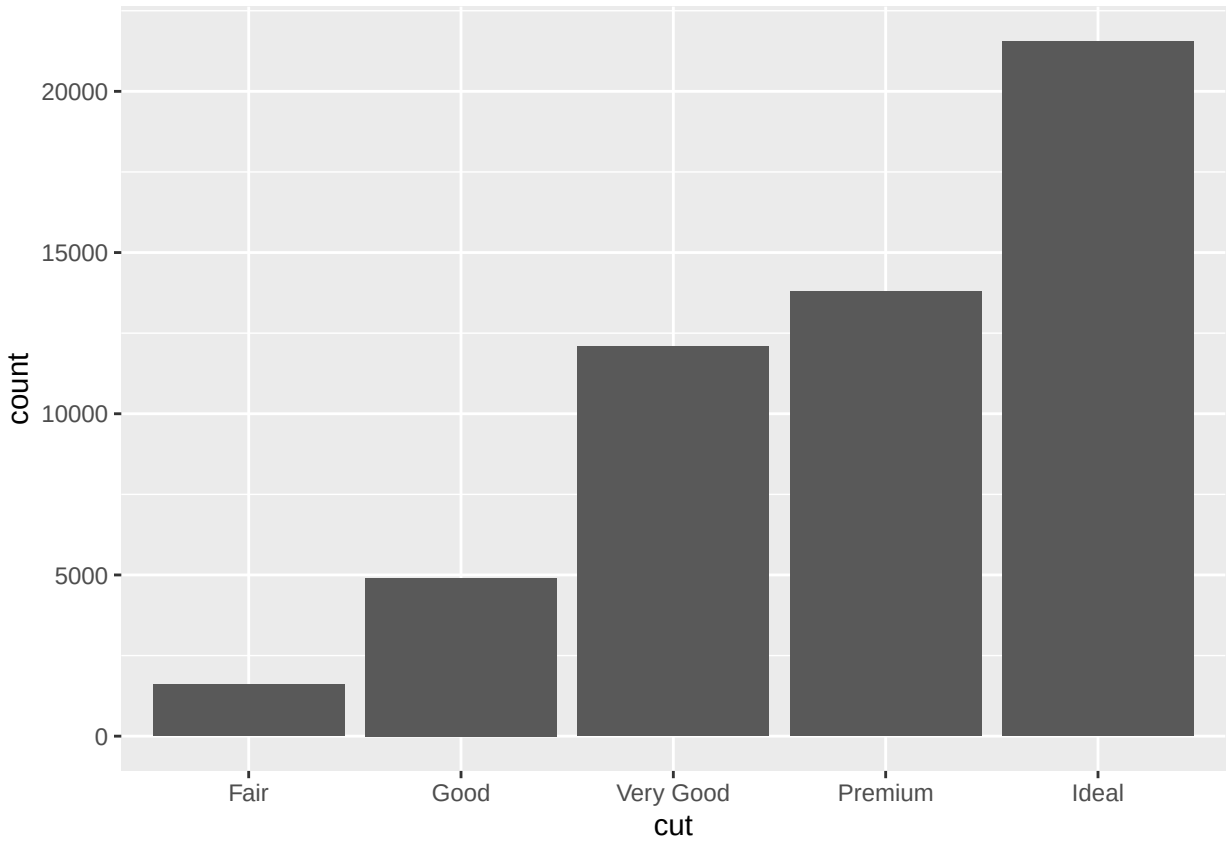
```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



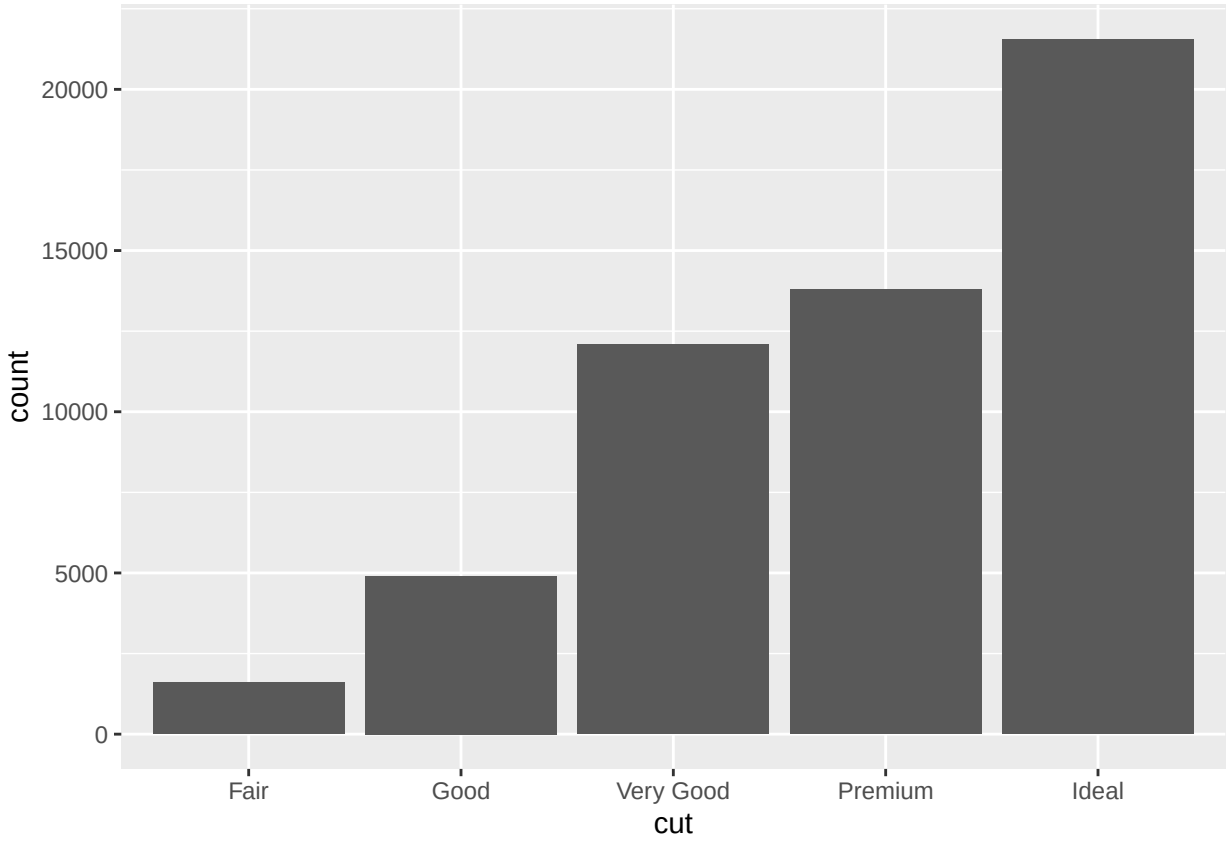
4. se argument creates confidence interval along the lines.
5. They are the same two plots. First ones aesthetics are globally specified and the second is specified in locally.
6. Skipped as I know how to do it.

Statistical Transformation

```
ggplot(data = diamonds) +
geom_bar(mapping = aes(x = cut))
```



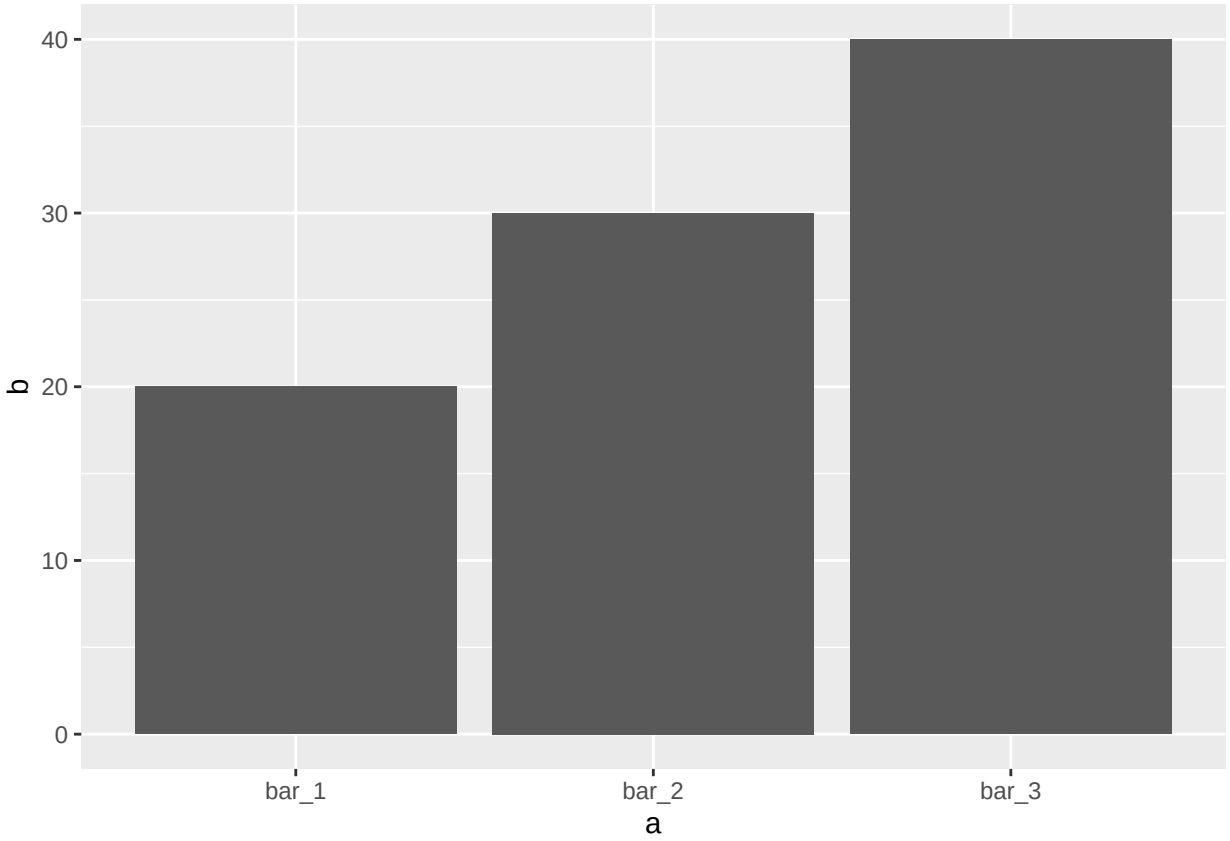

```
ggplot(data = diamonds) +
stat_count(mapping = aes(x = cut))
```



They worked as same.

For overriding the default count we can use stat identity to plot the y values the raw values. Example of this is given below.

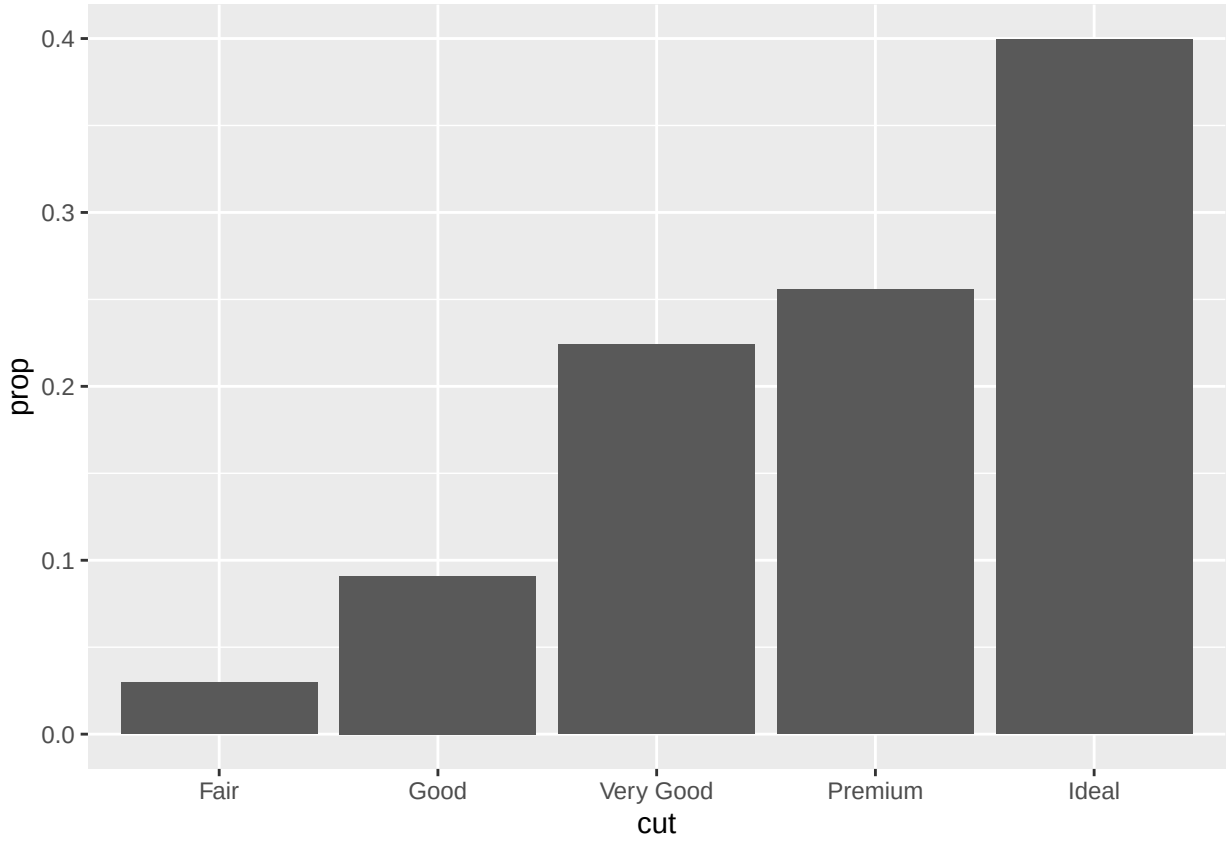
```
demo <- tribble(
~a, ~b,
"bar_1", 20,
"bar_2", 30,
"bar_3", 40
)

ggplot(data = demo) +
geom_bar(
mapping = aes(x = a, y = b), stat = "identity"
)
```



We can plot proportion instead of bar count or stat.

Like this,

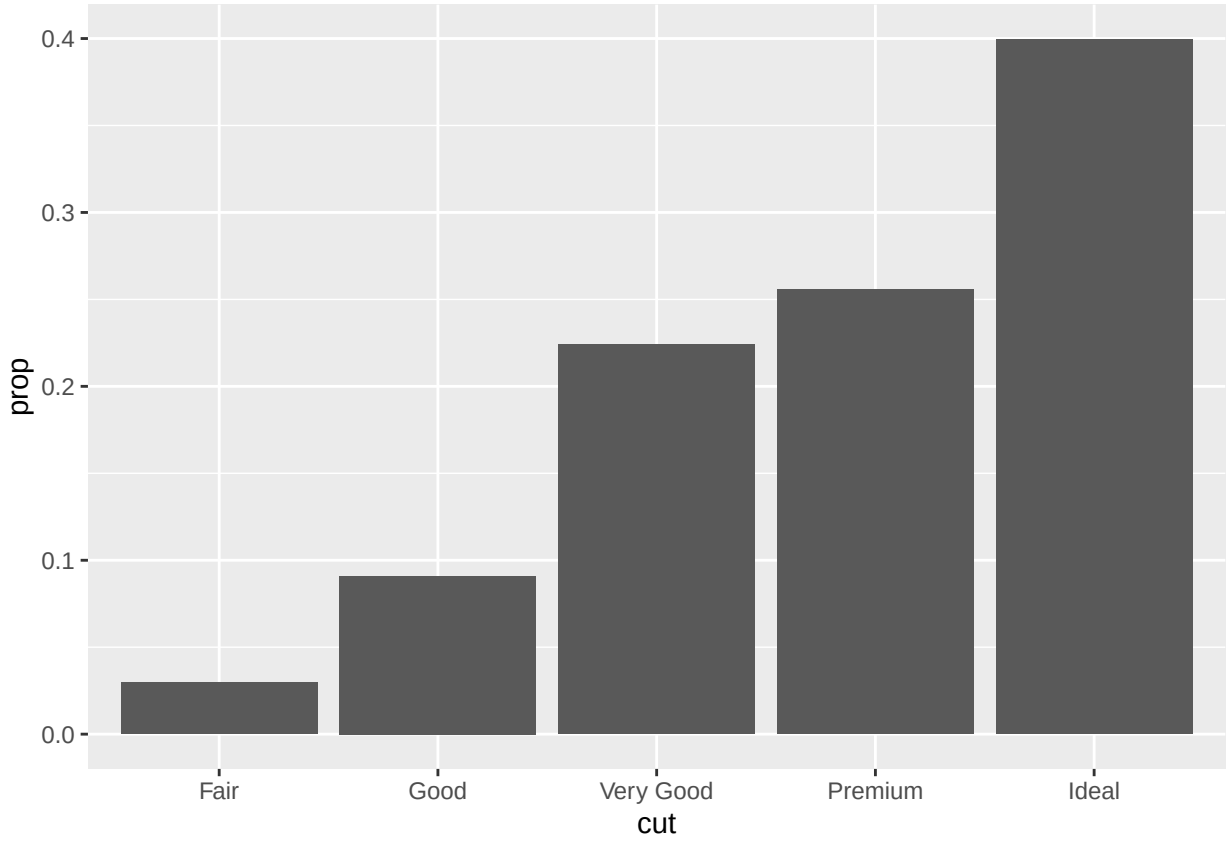
```
ggplot(data = diamonds) +
geom_bar(
mapping = aes(x = cut, y = after_stat(prop), group = 1)
)
```



or,

```
ggplot(data = diamonds) +
geom_bar(
mapping = aes(x = cut, y = ..prop.., group = 1)
)
```

```
## Warning: The dot-dot notation (‘..prop..‘) was deprecated in ggplot2 3.4.0.
## i Please use ‘after_stat(prop)‘ instead.
```



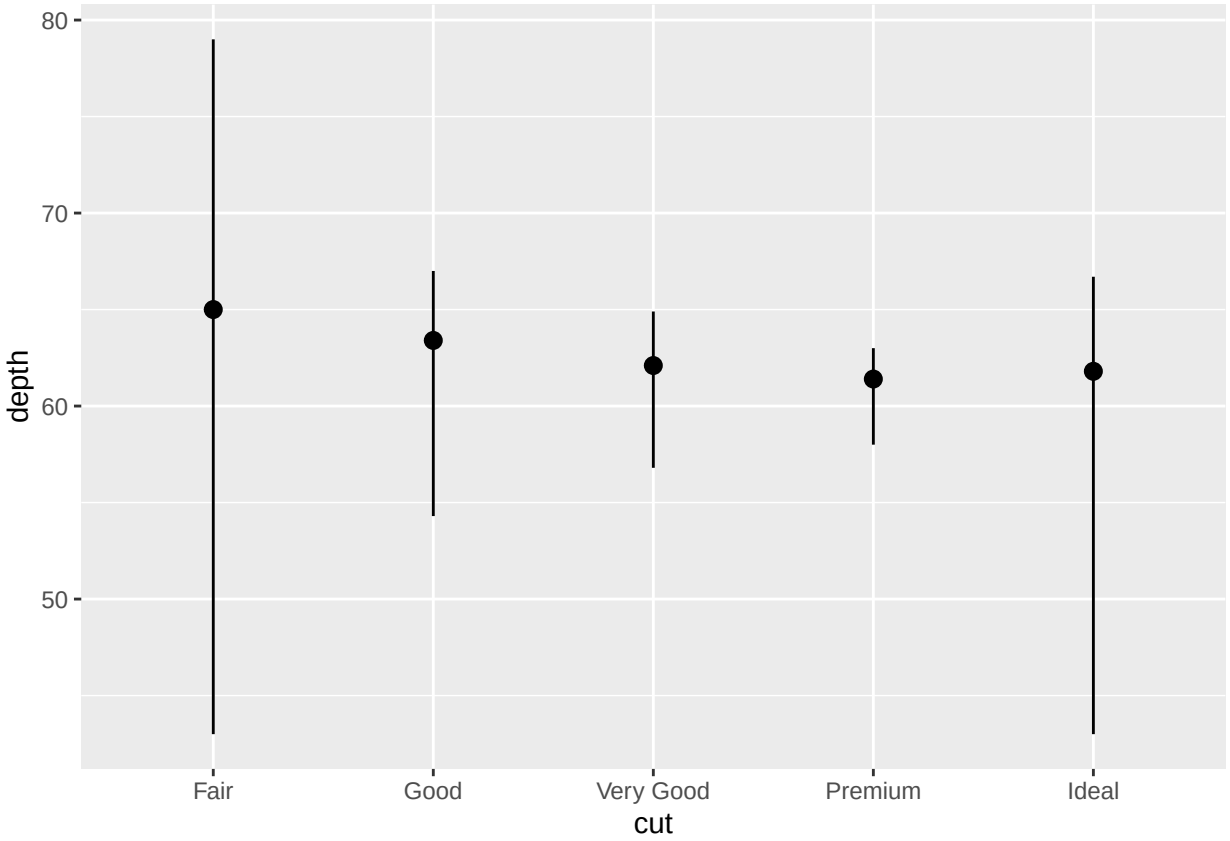
To plot the summaries of the y values for unique x values we can use stat_summary,

```
ggplot(data = diamonds) +
stat_summary(
mapping = aes(x = cut, y = depth),
fun.ymin = min,
fun.ymax = max,
fun.y = median
)
```

```
## Warning: The ‘fun.y‘ argument of ‘stat_summary()‘ is deprecated as of ggplot2 3.3.0.
## i Please use the ‘fun‘ argument instead.
```

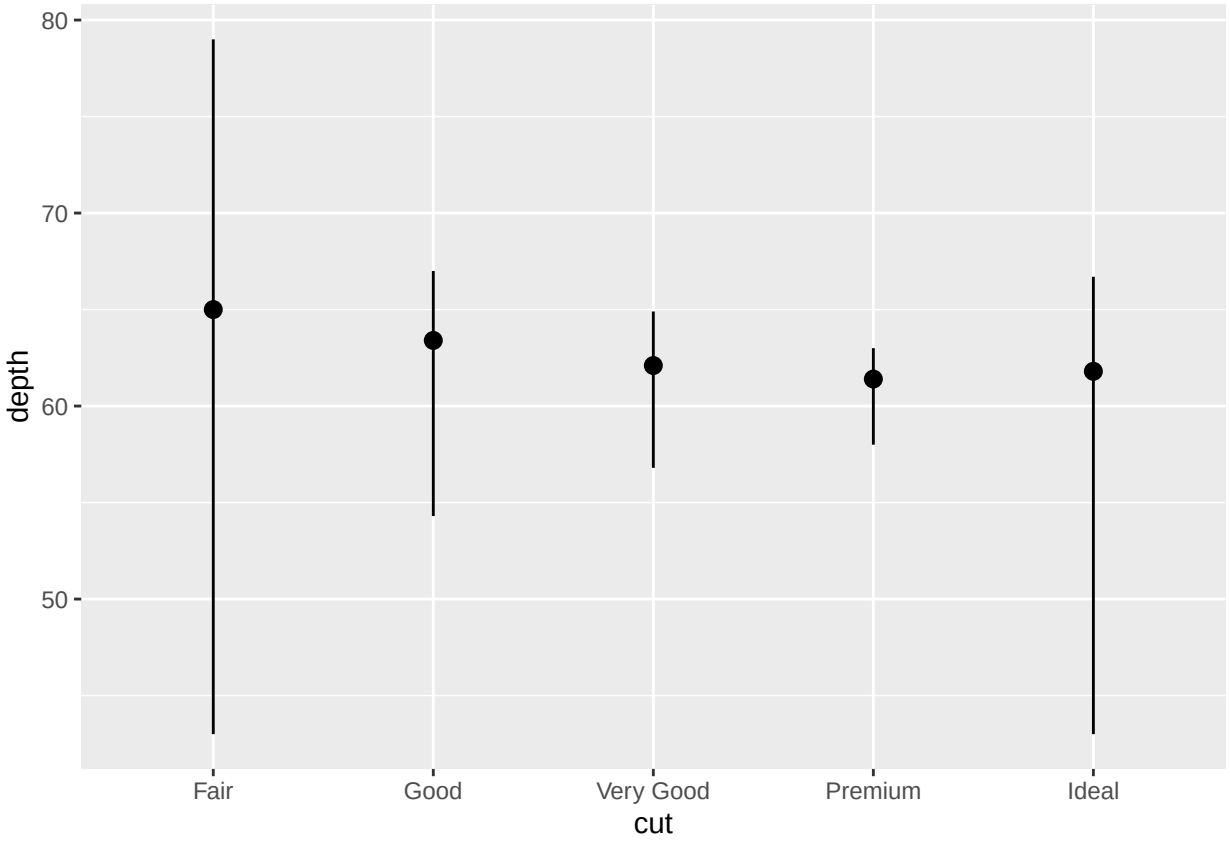
```
## Warning: The ‘fun.ymin‘ argument of ‘stat_summary()‘ is deprecated as of ggplot2 3.3.0.
## i Please use the ‘fun.min‘ argument instead.
```

```
## Warning: The ‘fun.ymax‘ argument of ‘stat_summary()‘ is deprecated as of ggplot2 3.3.0.
## i Please use the ‘fun.max‘ argument instead.
```



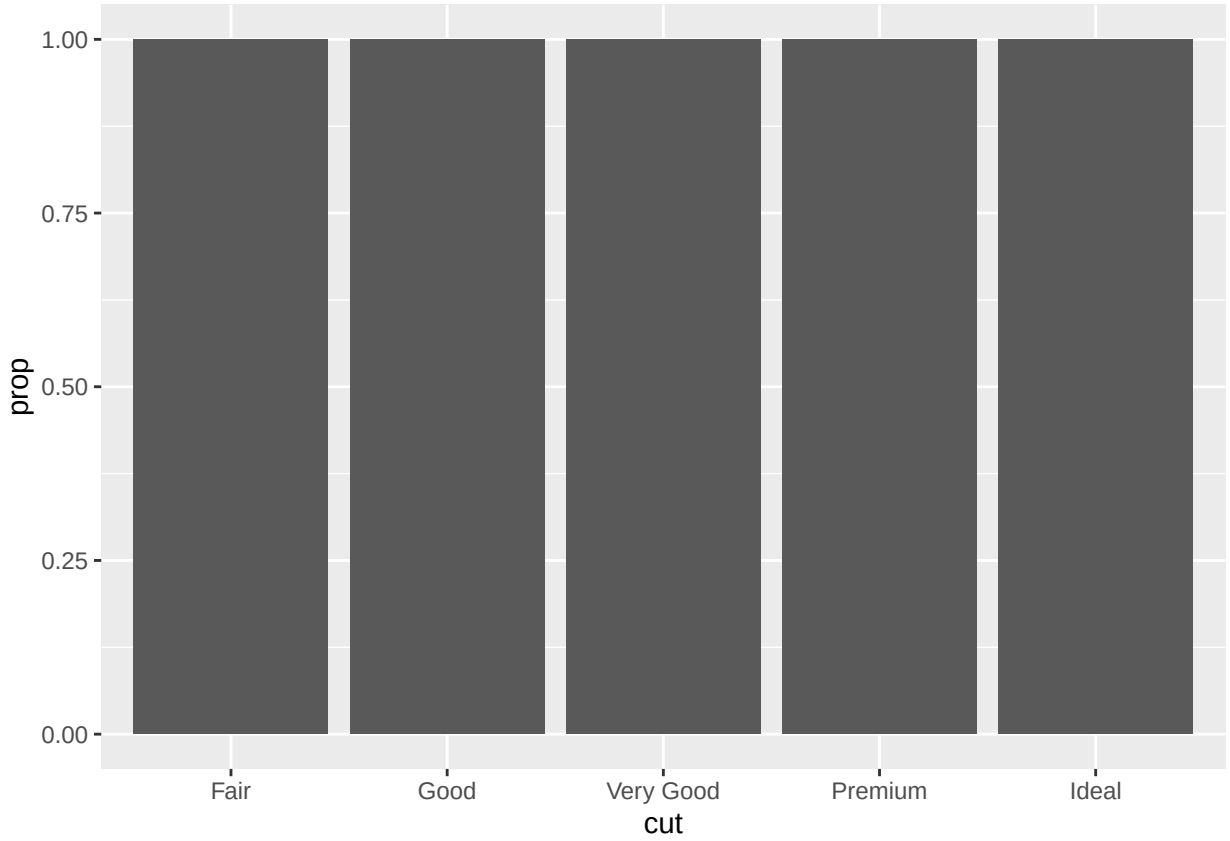
For avoid warning we can use

```
ggplot(data = diamonds) +
stat_summary(
mapping = aes(x = cut, y = depth),
fun.min = min,
fun.max = max,
fun = median
)
```

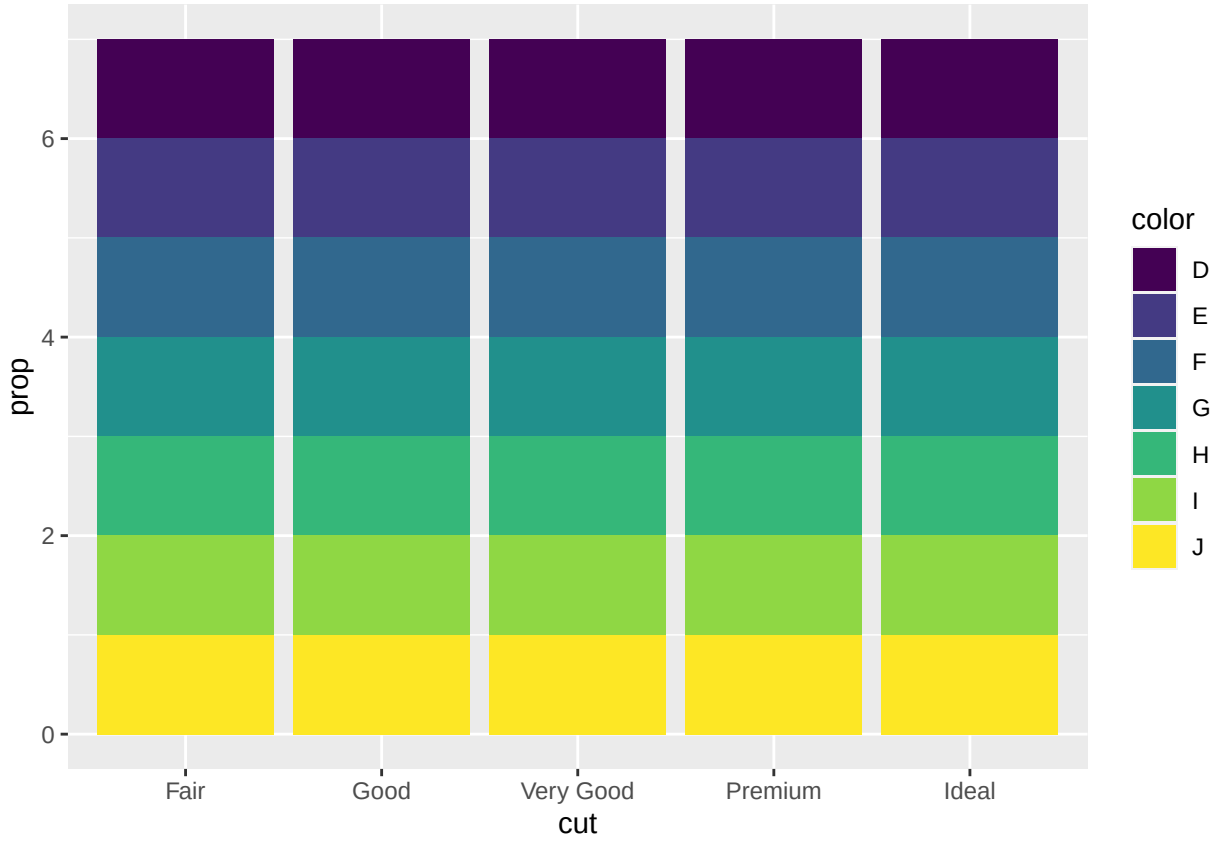


?stat_bin gives the description of the statistical work option.

```
ggplot(data = diamonds) +
geom_bar(mapping = aes(x = cut, y = ..prop..))
```



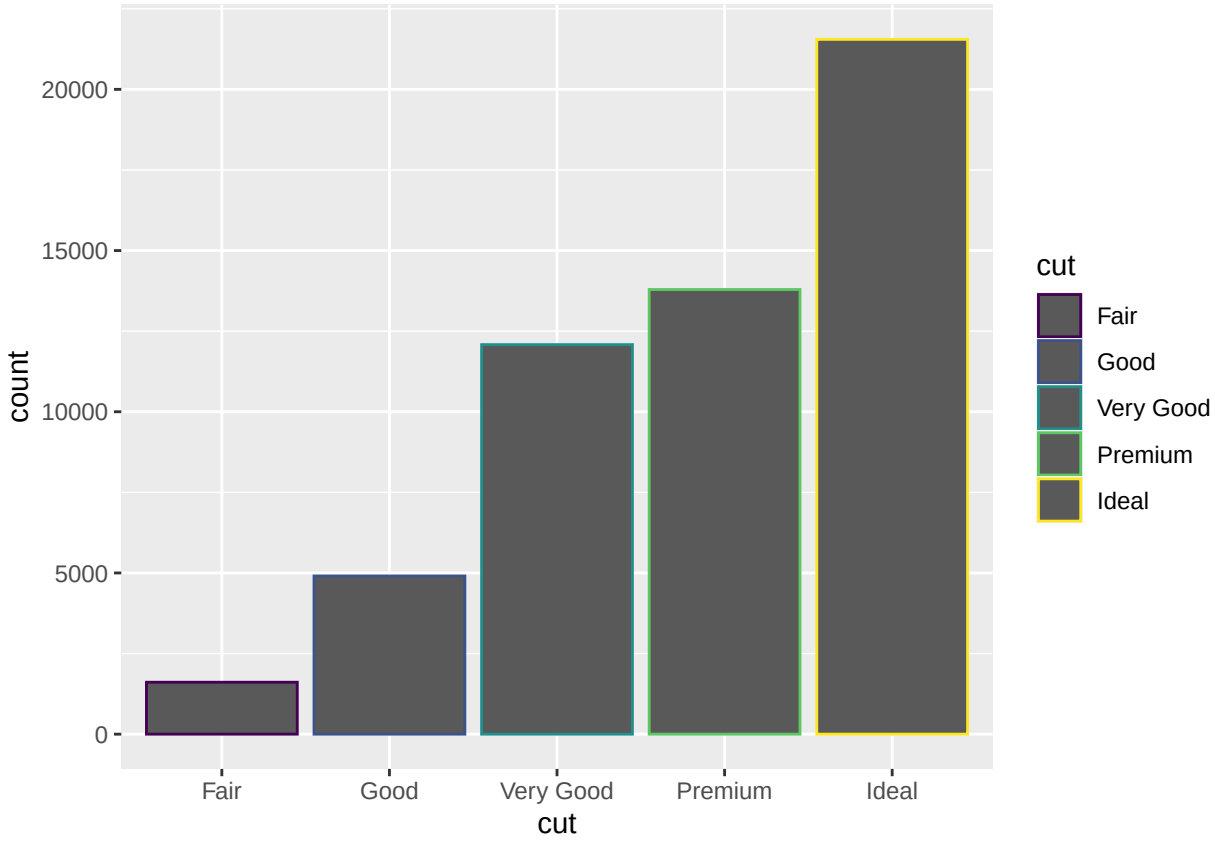

```
#
ggplot(data = diamonds) +
geom_bar(
mapping = aes(x = cut, fill = color, y = ..prop..)
)
```



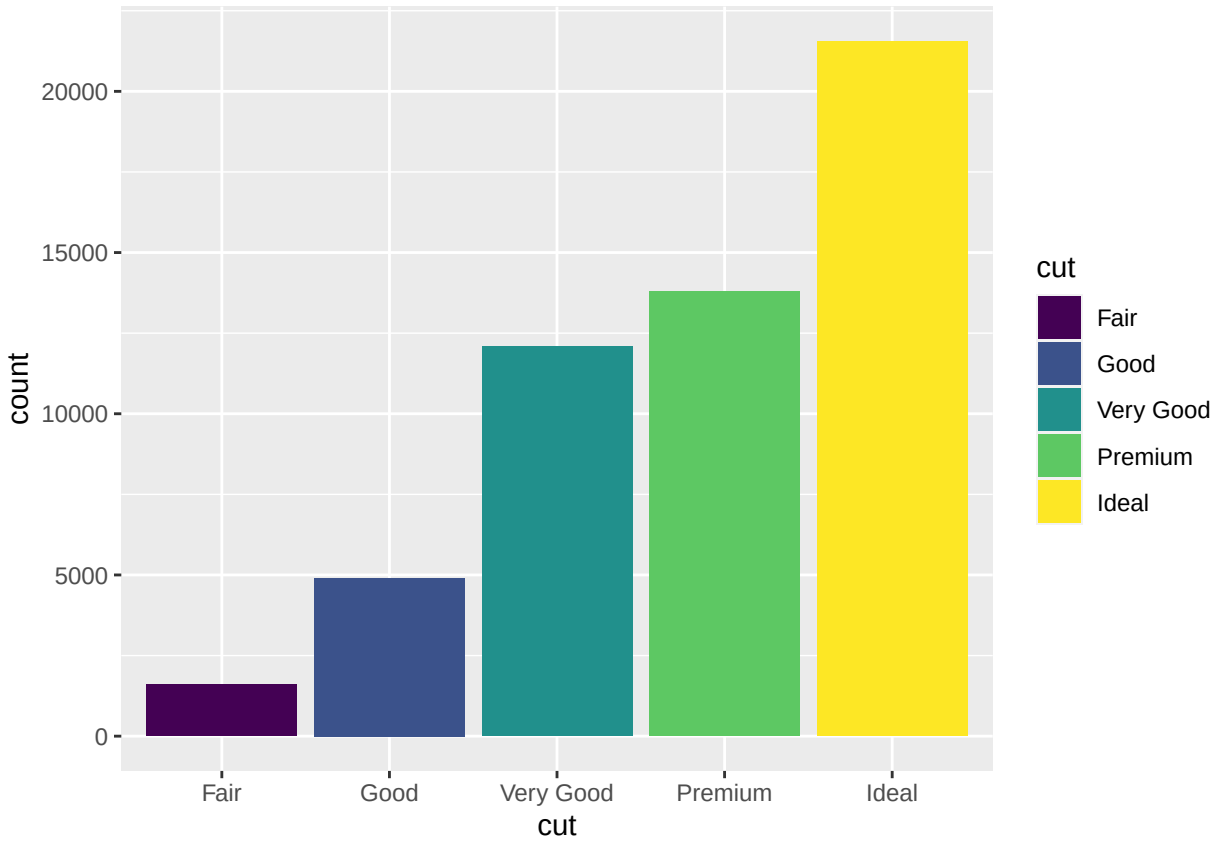
The difference is the later graph fills with color for different value in variable color in diamonds data set.

Position adjustment

```
ggplot(data = diamonds) +
geom_bar(mapping = aes(x = cut, color = cut))
```

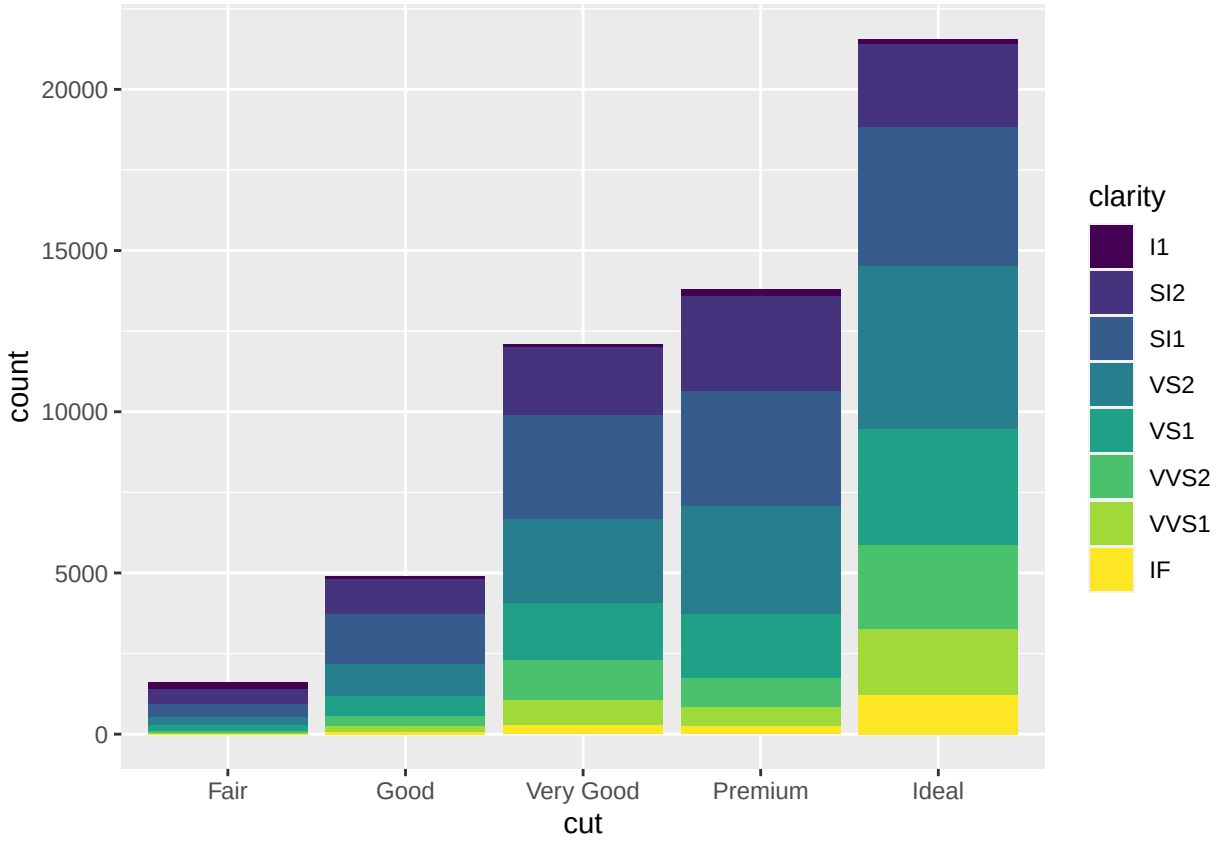



```
#

ggplot(data = diamonds) +
geom_bar(mapping = aes(x = cut, fill = cut))
```



mapping the fill with respect to the another variable

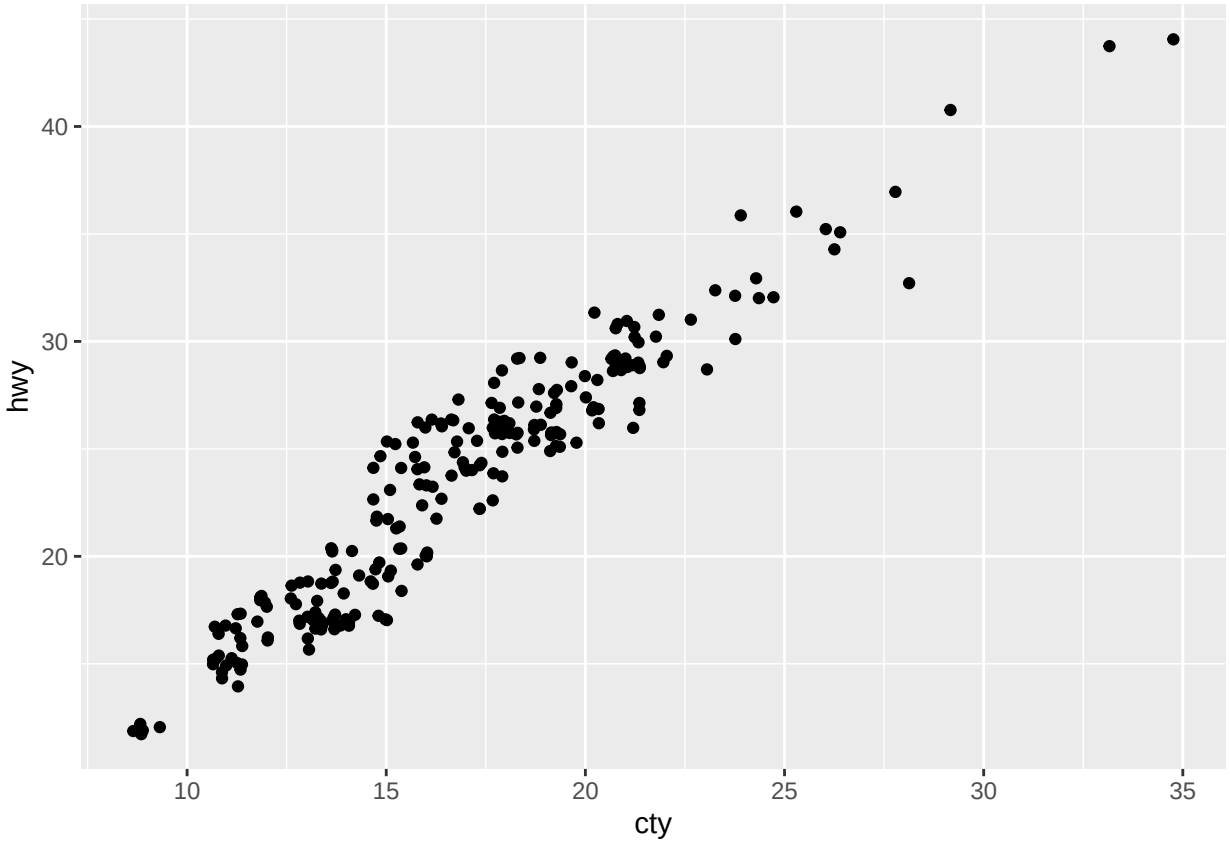
```
ggplot(data = diamonds) +
geom_bar(mapping = aes(x = cut, fill = clarity))
```



Exercise

1.

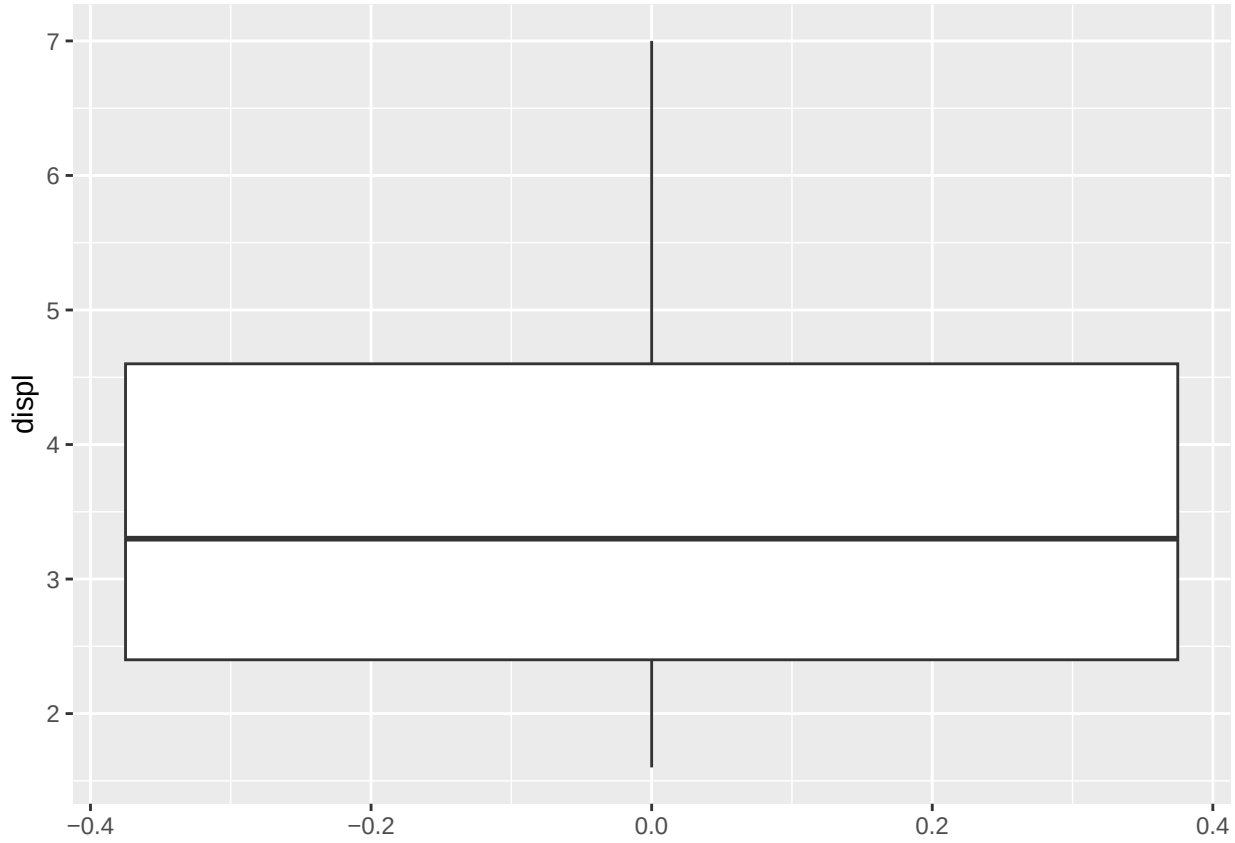
```
ggplot(data = mpg, mapping = aes(x = cty, y = hwy)) +
geom_point(position = "jitter")
```



Should add jitter.

4.

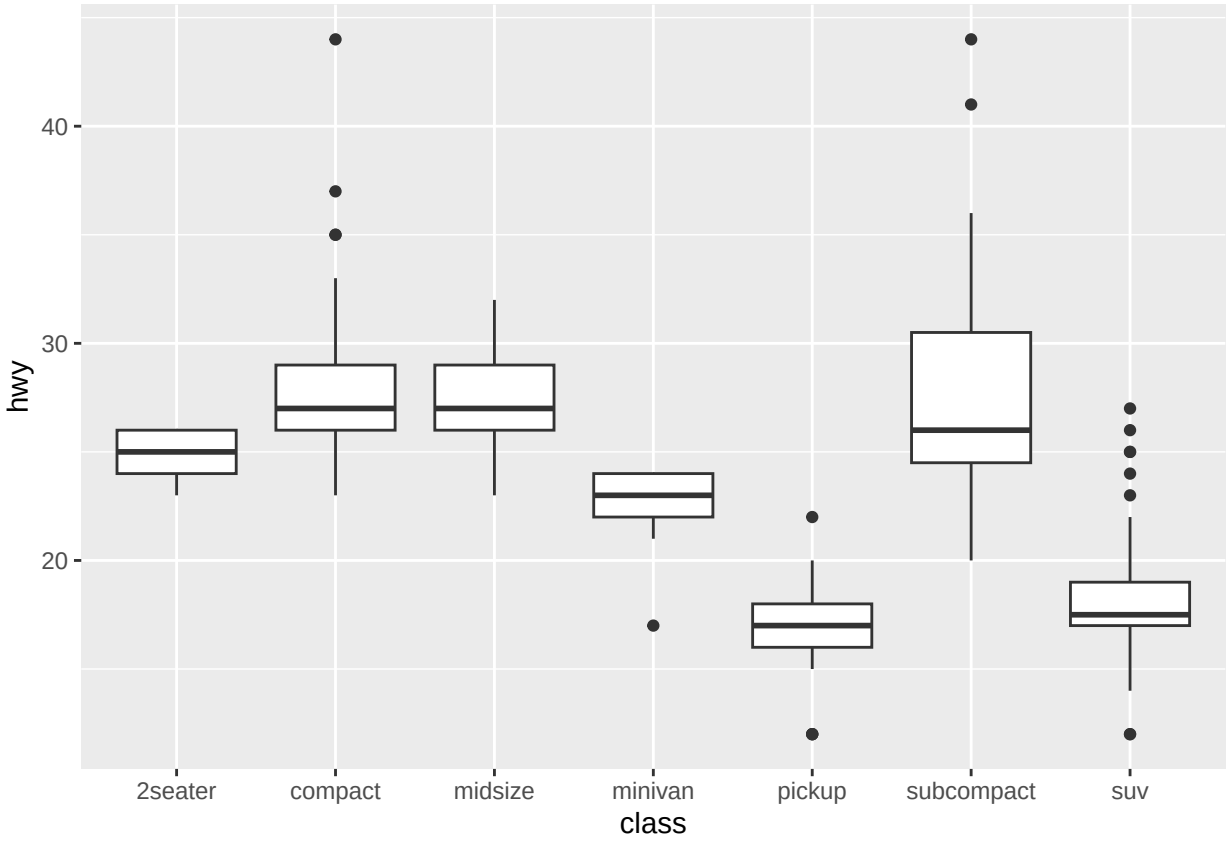
```
ggplot(data = mpg, mapping = aes(y = displ))+
  geom_boxplot()
```



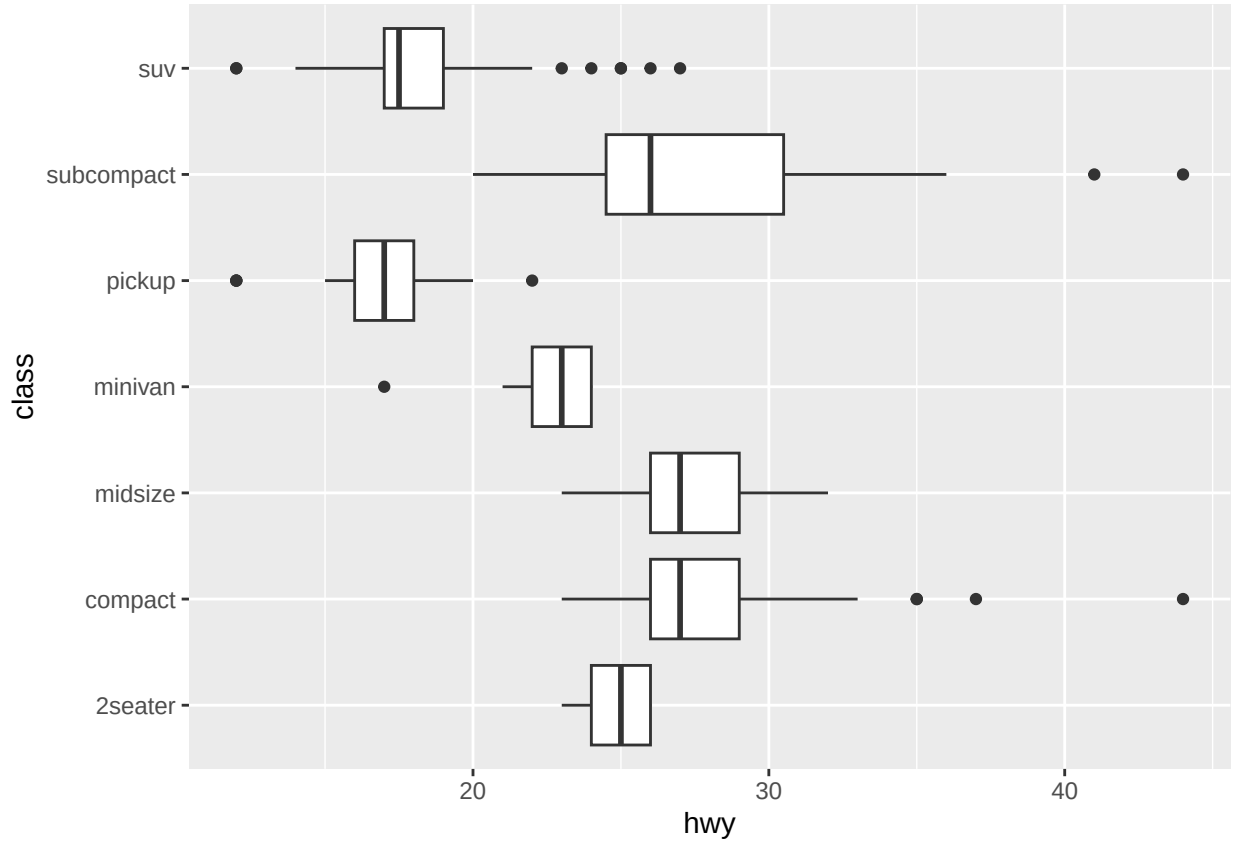
`coord_flip` switch the x and y axes.

Flipping the axes for the bloxplot.

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

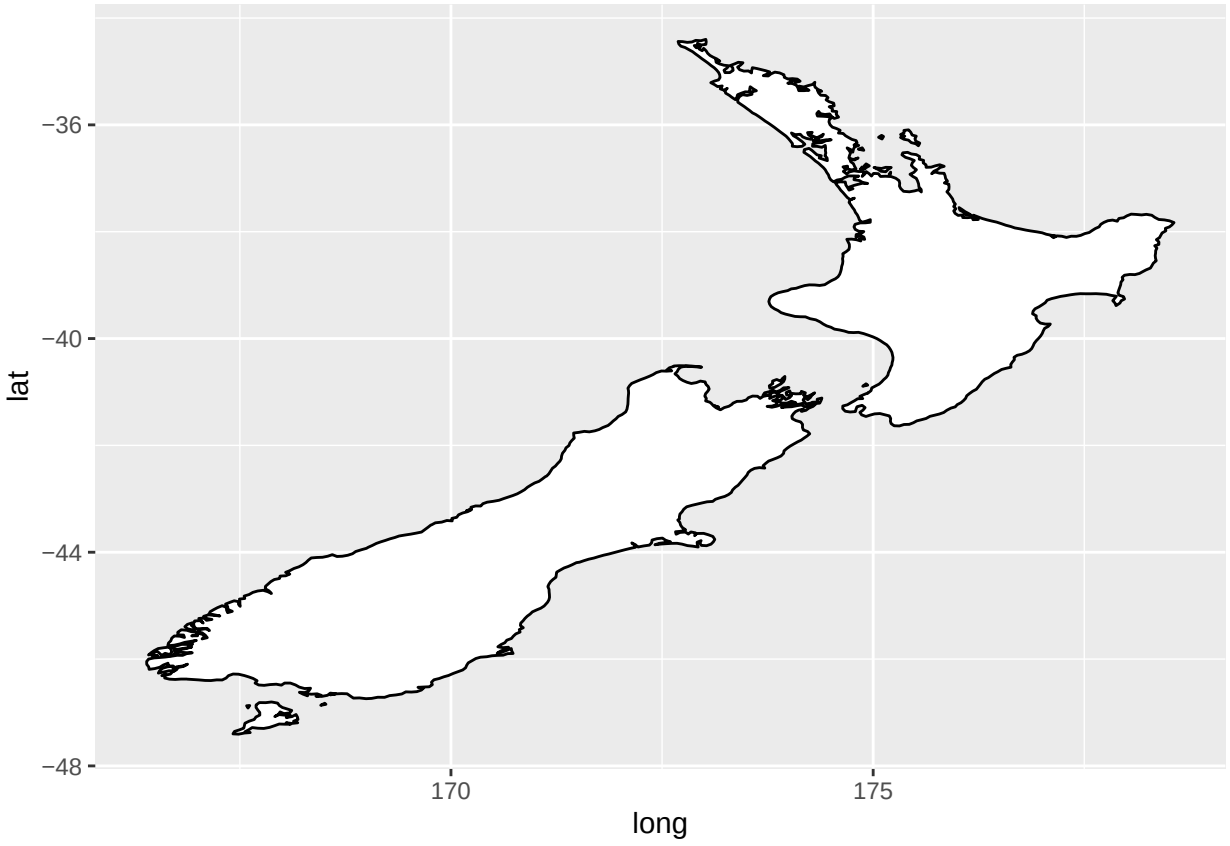



```
#

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

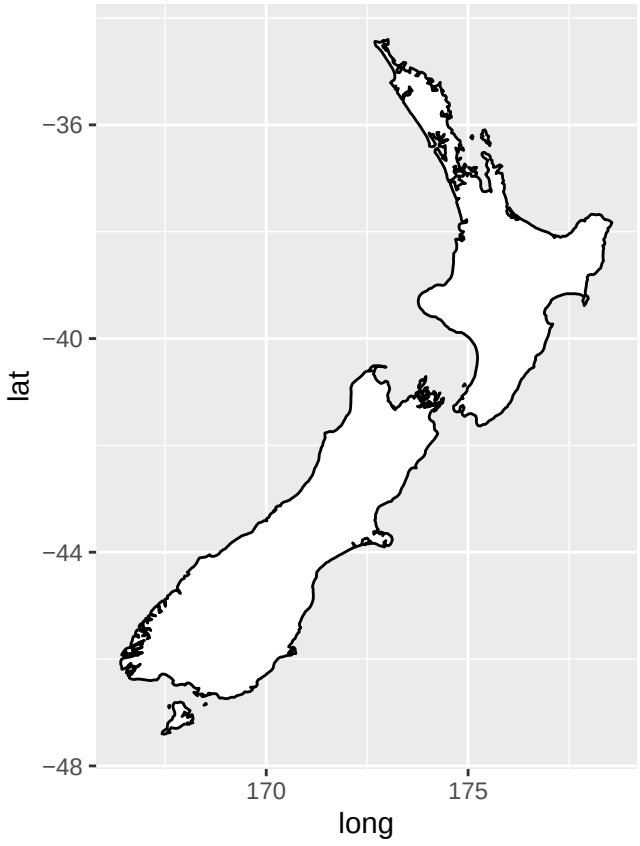


`coord_quickmap` sets the aspect ratio correctly for maps.

```
nz <- map_data("nz")

ggplot(data = nz, mapping = aes(x = long, y = lat, group = group))+
  geom_polygon(fill = "white", color = "black")
```

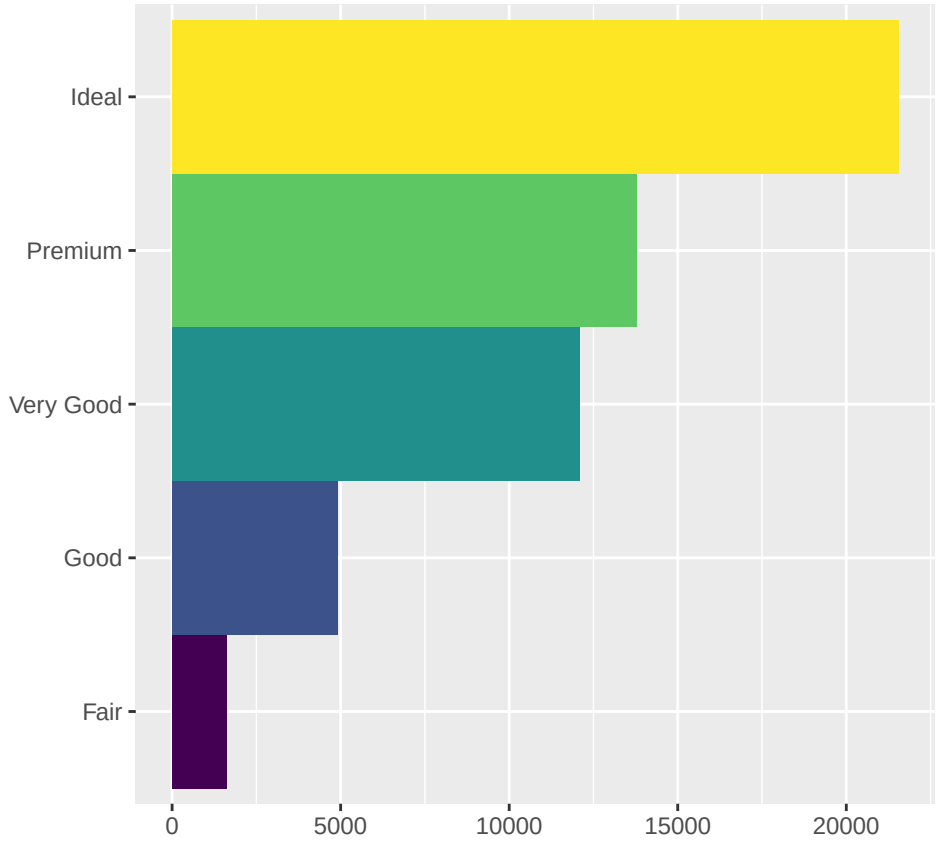



```
ggplot(nz, aes(long, lat, group = group)) +
geom_polygon(fill = "white", color = "black") +
coord_quickmap()
```

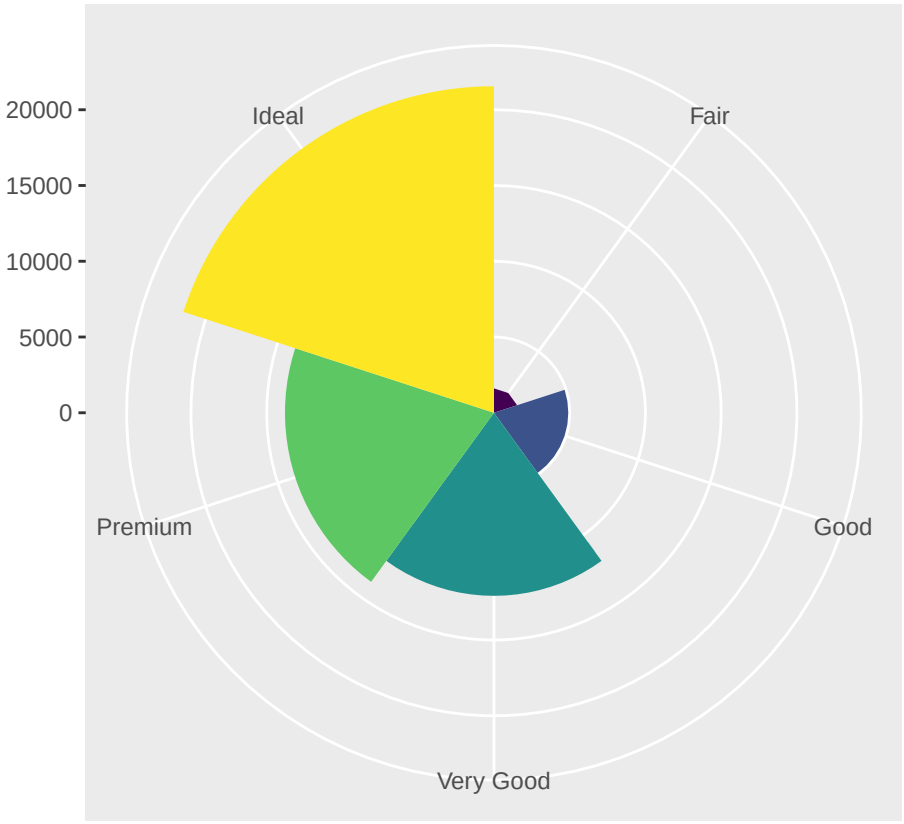


`coord_ploar()` uses polar coordinates.

```
bar <- ggplot(data = diamonds)+
  geom_bar(mapping = aes(x = cut, fill = cut),
           show.legend = F, width = 1)+
  theme(aspect.ratio = 1)+
  labs(x = NULL, y = NULL)

bar + coord_flip()
```




```
bar + coord_polar()
```



Chapter - 3

Loading the New York flight data.

In the chapter everything seems to familiar for me so I skipped it.

Exercise

1.

```
# a

flights %>%
  filter(arr_delay >= 2)
```

```
## # A tibble: 127,929 x 19
##     year month   day dep_time sched_de~1 dep_d~2 arr_t~3 sched~4 arr_d~5 carrier
##    <int> <int> <int>    <int>      <int>   <dbl>   <int>   <int>   <dbl> <chr>
##  1  2013     1     1      517        515       2     830     819      11 UA
##  2  2013     1     1      533        529       4     850     830      20 UA
##  3  2013     1     1      542        540       2     923     850      33 AA
##  4  2013     1     1      554        558      -4     740     728      12 UA
##  5  2013     1     1      555        600      -5     913     854      19 B6
##  6  2013     1     1      558        600      -2     753     745       8 AA
##  7  2013     1     1      558        600      -2     924     917       7 UA
##  8  2013     1     1      559        600      -1     941     910      31 AA
##  9  2013     1     1      600        600       0     837     825      12 MQ
```

```
## 10  2013     1     1      602        605      -3     821     805      16 MQ
## # ... with 127,919 more rows, 9 more variables: flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>, and abbreviated variable names
## #   1: sched_dep_time, 2: dep_delay, 3: arr_time, 4: sched_arr_time,
## #   5: arr_delay
```

```
# b

flights %>%
  filter(dest == c("IAH", "HOU"))
```

```
## # A tibble: 4,655 x 19
##     year month   day dep_time sched_de~1 dep_d~2 arr_t~3 sched~4 arr_d~5 carrier
##    <int> <int> <int>    <int>      <int>   <dbl>   <int>   <int>   <dbl> <chr>
##  1  2013     1     1      517        515       2     830     819      11 UA
##  2  2013     1     1      623        627      -4     933     932       1 UA
##  3  2013     1     1     1028       1026       2    1350    1339      11 UA
##  4  2013     1     1     1114        900     134    1447    1222     145 UA
##  5  2013     1     1     1208       1158      10    1540    1502      38 B6
##  6  2013     1     1     1306       1300       6    1622    1610      12 WN
##  7  2013     1     1     1527       1515      12    1854    1810      44 UA
##  8  2013     1     1     1620       1620       0    1945    1922      23 UA
##  9  2013     1     1     1725       1720       5    2045    2021      24 UA
## 10  2013     1     1     1855       1848       7    2203    2200       3 UA
## # ... with 4,645 more rows, 9 more variables: flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>, and abbreviated variable names
## #   1: sched_dep_time, 2: dep_delay, 3: arr_time, 4: sched_arr_time,
## #   5: arr_delay
```

```
# c

flights %>%
  filter(carrier == c("UA", "AA", "DL"))
```

```
## Warning: There was 1 warning in `filter()`.
## i In argument: `carrier == c("UA", "AA", "DL")`.
## Caused by warning in `carrier == c("UA", "AA", "DL")`:
## ! longer object length is not a multiple of shorter object length
```

```
## # A tibble: 46,913 x 19
##     year month   day dep_time sched_de~1 dep_d~2 arr_t~3 sched~4 arr_d~5 carrier
##    <int> <int> <int>    <int>      <int>   <dbl>   <int>   <int>   <dbl> <chr>
##  1  2013     1     1      517        515       2     830     819      11 UA
##  2  2013     1     1      558        600      -2     924     917       7 UA
##  3  2013     1     1      602        610      -8     812     820      -8 DL
##  4  2013     1     1      606        610      -4     858     910     -12 AA
##  5  2013     1     1      606        610      -4     837     845      -8 DL
##  6  2013     1     1      607        607       0     858     915     -17 UA
##  7  2013     1     1      615        615       0     833     842      -9 DL
##  8  2013     1     1      623        610      13     920     915       5 AA
##  9  2013     1     1      643        646      -3     922     940     -18 UA
## 10  2013     1     1      653        700      -7     936    1009     -33 DL
## # ... with 46,903 more rows, 9 more variables: flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
```

```
## #   minute <dbl>, time_hour <dttm>, and abbreviated variable names
## #   1: sched_dep_time, 2: dep_delay, 3: arr_time, 4: sched_arr_time,
## #   5: arr_delay
```

```
# d
```

Using the `select()` command.

```
## Selecting using `contain()` argument.
flights %>%
  filter(month %in% 7:9)
```

```
## # A tibble: 86,326 x 19
##     year month   day dep_time sched_de~1 dep_d~2 arr_t~3 sched~4 arr_d~5 carrier
##    <int> <int> <int>    <int>      <int>   <dbl>   <int>   <int>   <dbl> <chr>
##  1  2013     7     1        1       2029     212     236    2359     157 B6
##  2  2013     7     1        2       2359       3     344     344       0 B6
##  3  2013     7     1       29       2245     104     151       1     110 B6
##  4  2013     7     1       43       2130     193     322      14     188 B6
##  5  2013     7     1       44       2150     174     300     100     120 AA
##  6  2013     7     1       46       2051     235     304    2358     186 B6
##  7  2013     7     1       48       2001     287     308    2305     243 VX
##  8  2013     7     1       58       2155     183     335      43     172 B6
##  9  2013     7     1      100       2146     194     327      30     177 B6
## 10  2013     7     1      100       2245     135     337     135     122 B6
## # ... with 86,316 more rows, 9 more variables: flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>, and abbreviated variable names
## #   1: sched_dep_time, 2: dep_delay, 3: arr_time, 4: sched_arr_time,
## #   5: arr_delay
```

```
select(flights, contains("TIME"))
```

```
## # A tibble: 336,776 x 6
##    dep_time sched_dep_time arr_time sched_arr_time air_time time_hour
##       <int>          <int>    <int>          <int>    <dbl> <dttm>
##  1      517            515      830            819      227 2013-01-01 05:00:00
##  2      533            529      850            830      227 2013-01-01 05:00:00
##  3      542            540      923            850      160 2013-01-01 05:00:00
##  4      544            545     1004           1022      183 2013-01-01 05:00:00
##  5      554            600      812            837      116 2013-01-01 06:00:00
##  6      554            558      740            728      150 2013-01-01 05:00:00
##  7      555            600      913            854      158 2013-01-01 06:00:00
##  8      557            600      709            723       53 2013-01-01 06:00:00
##  9      557            600      838            846      140 2013-01-01 06:00:00
## 10      558            600      753            745      138 2013-01-01 06:00:00
## # ... with 336,766 more rows
```

```
## Selecting using `oneof()` argument
vars <- c(
"year", "month", "day", "dep_delay", "arr_delay"
)

select(flights, 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
```

We can create new variable using using `mutate()` and If we want to keep only the new variables we can use `transmutate()`

Examples are given below:

```
flights_sml <- select(flights,
year:day,
ends_with("delay"),
distance,
air_time
)

# With mutate
mutate(flights_sml,
gain = arr_delay - dep_delay,
hours = air_time / 60,
gain_per_hour = gain / hours
)
```

```
## # A tibble: 336,776 x 10
##     year month   day dep_delay arr_delay distance air_time  gain hours gain_pe~1
##    <int> <int> <int>     <dbl>     <dbl>    <dbl>    <dbl> <dbl> <dbl>     <dbl>
##  1  2013     1     1         2        11     1400      227     9 3.78       2.38
##  2  2013     1     1         4        20     1416      227    16 3.78       4.23
##  3  2013     1     1         2        33     1089      160    31 2.67      11.6
##  4  2013     1     1        -1       -18     1576      183   -17 3.05      -5.57
##  5  2013     1     1        -6       -25      762      116   -19 1.93      -9.83
##  6  2013     1     1        -4        12      719      150    16 2.5        6.4
##  7  2013     1     1        -5        19     1065      158    24 2.63       9.11
##  8  2013     1     1        -3       -14      229       53   -11 0.883    -12.5
##  9  2013     1     1        -3        -8      944      140    -5 2.33      -2.14
## 10  2013     1     1        -2         8      733      138    10 2.3        4.35
## # ... with 336,766 more rows, and abbreviated variable name 1: gain_per_hour
```

```
# With transmutate
transmute(flights,
gain = arr_delay - dep_delay,
hours = air_time / 60,
gain_per_hour = gain / hours
)
```

```
## # A tibble: 336,776 x 3
##     gain hours gain_per_hour
##    <dbl> <dbl>         <dbl>
```

```
##  1     9 3.78           2.38
##  2    16 3.78           4.23
##  3    31 2.67          11.6
##  4   -17 3.05          -5.57
##  5   -19 1.93          -9.83
##  6    16 2.5            6.4
##  7    24 2.63           9.11
##  8   -11 0.883        -12.5
##  9    -5 2.33          -2.14
## 10    10 2.3            4.35
## # ... with 336,766 more rows
```

Exercises

2.

```
comperison <- flights %>%
  mutate(fly = arr_time - dep_time,
         com = fly - air_time) %>%
  select(com)
comperison
```

```
## # A tibble: 336,776 x 1
##      com
##    <dbl>
##  1    86
##  2    90
##  3   221
##  4   277
##  5   142
##  6    36
##  7   200
##  8    99
##  9   141
## 10    57
## # ... with 336,766 more rows
```

Group Summarise with `summarize()`

```
flights %>%
  summarize(delay = mean(dep_delay, na.rm = T))
```

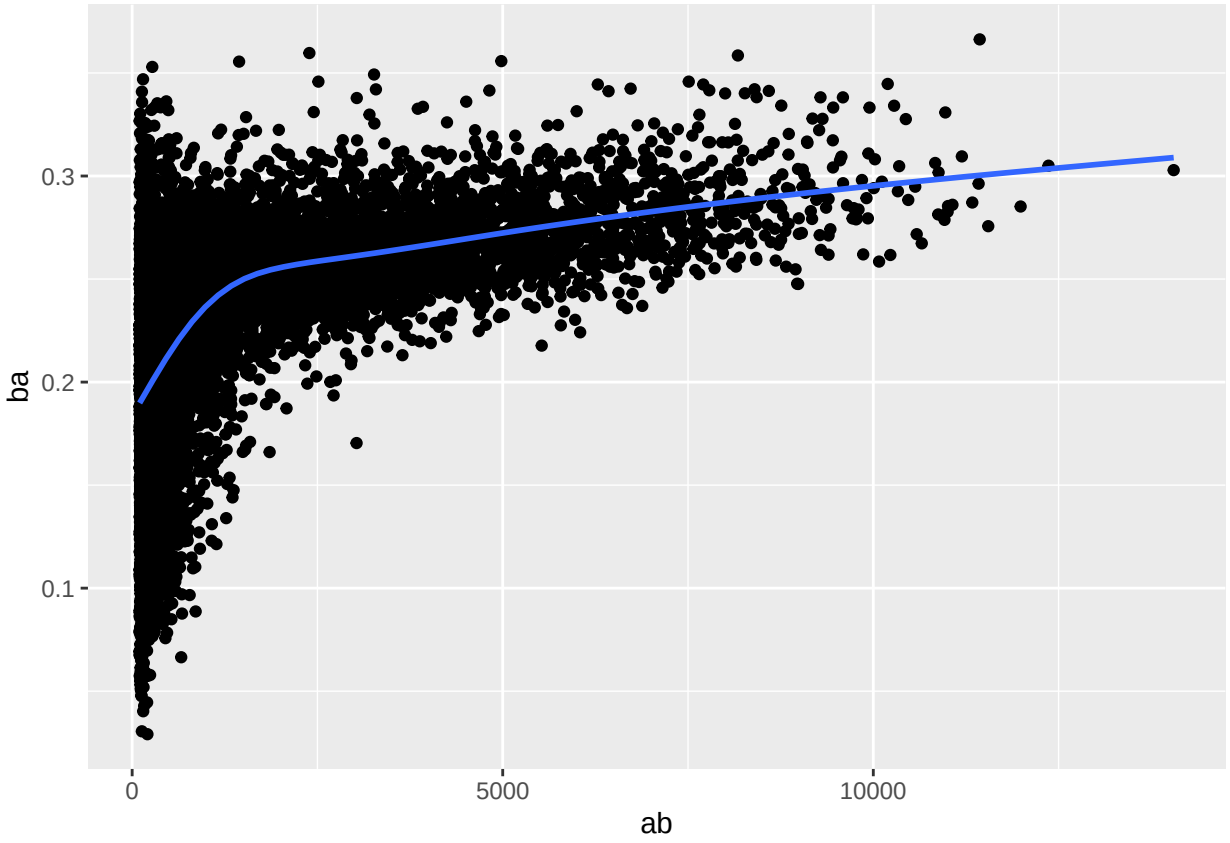
```
## # A tibble: 1 x 1
##   delay
##   <dbl>
## 1  12.6
```

Using `Leman` package for batting data summary.

```
batting <- as_tibble(Lahman::Batting)

batters <- batting %>%
              group_by(playerID) %>%
              summarise(ba = sum(H, na.rm = TRUE) /
                          sum(AB, na.rm = TRUE),
                        ab = sum(AB, na.rm = TRUE))
```

```
batters %>%
  filter(ab > 100) %>%
  ggplot(mapping = aes(x = ab, y = ba))+
  geom_point()+
  geom_smooth(se = F)
```

```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```




```
not_cancelled <- flights %>%
filter(!is.na(dep_delay), !is.na(arr_delay))

not_cancelled %>%
group_by(year, month, day) %>%
summarize(
# average delay:
avg_delay1 = mean(arr_delay),
# average positive delay:
avg_delay2 = mean(arr_delay[arr_delay > 0])
)
```

```
## `summarise()` has grouped output by 'year', 'month'. You can override using the
## `.groups` argument.
```

```
## # A tibble: 365 x 5
## # Groups:   year, month [12]
##     year month   day avg_delay1 avg_delay2
##    <int> <int> <int>      <dbl>      <dbl>
```

```
##  1  2013     1     1     12.7         32.5
##  2  2013     1     2     12.7         32.0
##  3  2013     1     3      5.73        27.7
##  4  2013     1     4     -1.93        28.3
##  5  2013     1     5     -1.53        22.6
##  6  2013     1     6      4.24        24.4
##  7  2013     1     7     -4.95        27.8
##  8  2013     1     8     -3.23        20.8
##  9  2013     1     9     -0.264       25.6
## 10  2013     1    10     -5.90        27.3
## # ... with 355 more rows
```

```
not_cancelled %>%
group_by(dest) %>%
summarize(distance_sd = sd(distance)) %>%
arrange(desc(distance_sd))
```

```
## # A tibble: 104 x 2
##    dest  distance_sd
##    <chr>       <dbl>
##  1 EGE         10.5
##  2 SAN         10.4
##  3 SFO         10.2
##  4 HNL         10.0
##  5 SEA          9.98
##  6 LAS          9.91
##  7 PDX          9.87
##  8 PHX          9.86
##  9 LAX          9.66
## 10 IND          9.46
## # ... with 94 more rows
```

```
not_cancelled %>%
group_by(year, month, day) %>%
summarize(
first = min(dep_time),
last = max(dep_time)
)
```

```
## `summarise()` has grouped output by 'year', 'month'. You can override using the
## `.groups` argument.
```

```
## # A tibble: 365 x 5
## # Groups:   year, month [12]
##     year month   day first  last
##    <int> <int> <int> <int> <int>
##  1  2013     1     1   517  2356
##  2  2013     1     2    42  2354
##  3  2013     1     3    32  2349
##  4  2013     1     4    25  2358
##  5  2013     1     5    14  2357
##  6  2013     1     6    16  2355
##  7  2013     1     7    49  2359
##  8  2013     1     8   454  2351
##  9  2013     1     9     2  2252
## 10  2013     1    10     3  2320
```

```
## # ... with 355 more rows
```

```
not_cancelled %>%
group_by(year, month, day) %>%
summarize(
first_dep = first(dep_time),
last_dep = last(dep_time)
)
```

```
## `summarise()` has grouped output by 'year', 'month'. You can override using the
## `.groups` argument.
```

```
## # A tibble: 365 x 5
## # Groups:   year, month [12]
##     year month   day first_dep last_dep
##    <int> <int> <int>     <int>    <int>
##  1  2013     1     1       517     2356
##  2  2013     1     2        42     2354
##  3  2013     1     3        32     2349
##  4  2013     1     4        25     2358
##  5  2013     1     5        14     2357
##  6  2013     1     6        16     2355
##  7  2013     1     7        49     2359
##  8  2013     1     8       454     2351
##  9  2013     1     9         2     2252
## 10  2013     1    10         3     2320
## # ... with 355 more rows
```

```
not_cancelled %>%
group_by(year, month, day) %>%
mutate(r = min_rank(desc(dep_time))) %>%
filter(r %in% range(r))
```

```
## # A tibble: 770 x 20
## # Groups:   year, month, day [365]
##     year month   day dep_time sched_de~1 dep_d~2 arr_t~3 sched~4 arr_d~5 carrier
##    <int> <int> <int>    <int>      <int>   <dbl>   <int>   <int>   <dbl> <chr>
##  1  2013     1     1      517        515       2     830     819      11 UA
##  2  2013     1     1     2356       2359      -3     425     437     -12 B6
##  3  2013     1     2       42       2359      43     518     442      36 B6
##  4  2013     1     2     2354       2359      -5     413     437     -24 B6
##  5  2013     1     3       32       2359      33     504     442      22 B6
##  6  2013     1     3     2349       2359     -10     434     445     -11 B6
##  7  2013     1     4       25       2359      26     505     442      23 B6
##  8  2013     1     4     2358       2359      -1     429     437      -8 B6
##  9  2013     1     4     2358       2359      -1     436     445      -9 B6
## 10  2013     1     5       14       2359      15     503     445      18 B6
## # ... with 760 more rows, 10 more variables: flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>, r <int>, and abbreviated variable names
## #   1: sched_dep_time, 2: dep_delay, 3: arr_time, 4: sched_arr_time,
## #   5: arr_delay
```

```
not_cancelled %>%
count(tailnum, wt = distance)
```

```
## # A tibble: 4,037 x 2
```

```
##    tailnum      n
##    <chr>    <dbl>
##  1 D942DN    3418
##  2 N0EGMQ  239143
##  3 N10156  109664
##  4 N102UW   25722
##  5 N103US   24619
##  6 N104UW   24616
##  7 N10575  139903
##  8 N105UW   23618
##  9 N107US   21677
## 10 N108UW   32070
## # ... with 4,027 more rows
```