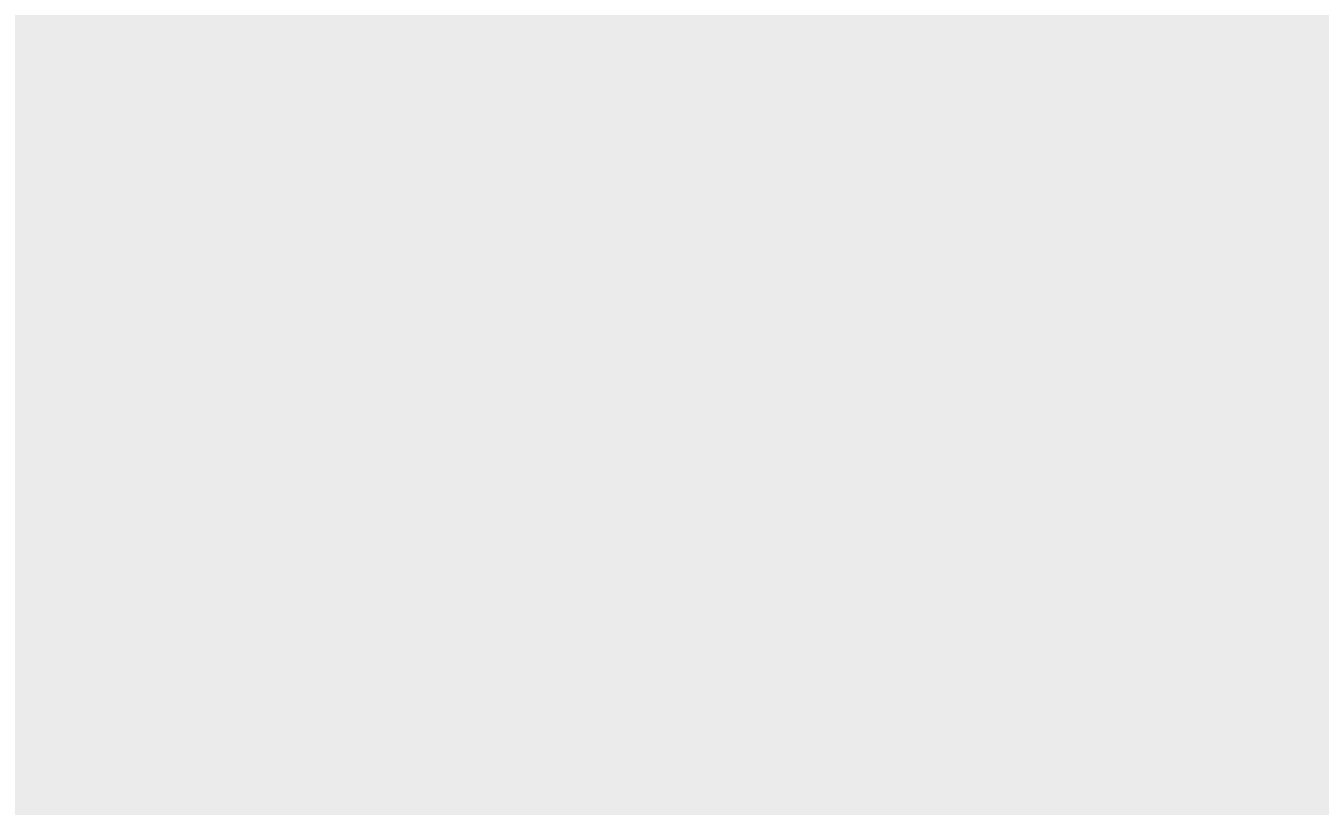
**שאלה 1**

Run ggplot(data = mpg) what do you see?

**ggplot**(data = mpg)



This code creates an empty plot. The ggplot() function creates the background of the plot, but since no layers were specified with geom function, nothing is drawn.

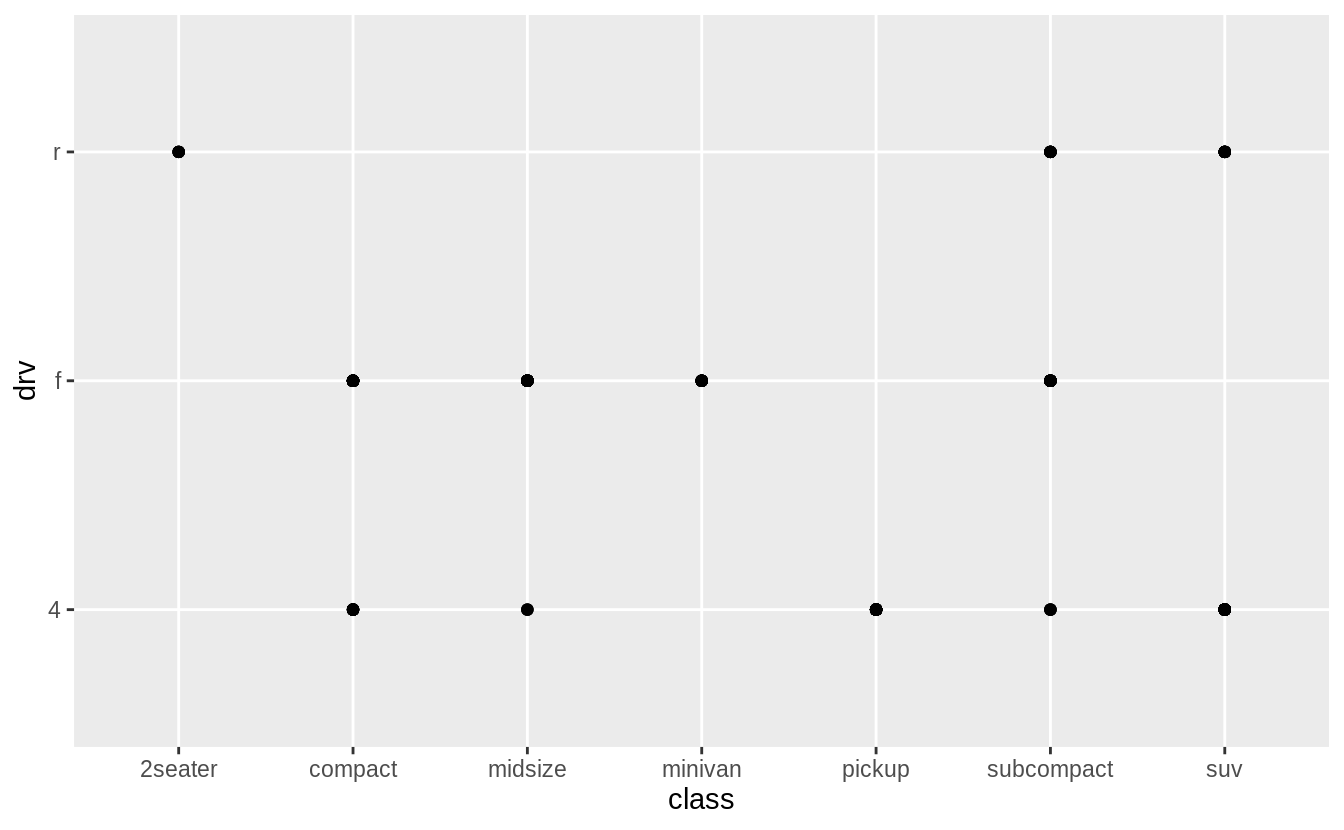
**שאלה 2**

What happens if you make a scatter plot of class vs drv? Why is the plot not useful?

The resulting scatterplot has only a few points.

**ggplot**(mpg, **aes**(x = class, y = drv)) +

**geom\_point**()



A scatter plot is not a useful display of these variables since both drv and class are categorical variables. Since categorical variables typically take a small number of values, there are a limited number of unique combinations of (x, y) values that can be displayed. In this data, drv takes 3 values and class takes 7 values, meaning that there are only 21 values that could be plotted on a scatterplot of drv vs. class. In this data, there 12 values of (drv, class) are observed.

**count**(mpg, drv, class)

*#> # A tibble: 12 x 3*

*#> drv class n*

*#> <chr> <chr> <int>*

*#> 1 4 compact 12*

*#> 2 4 midsize 3*

*#> 3 4 pickup 33*

*#> 4 4 subcompact 4*

*#> 5 4 suv 51*

*#> 6 f compact 35*

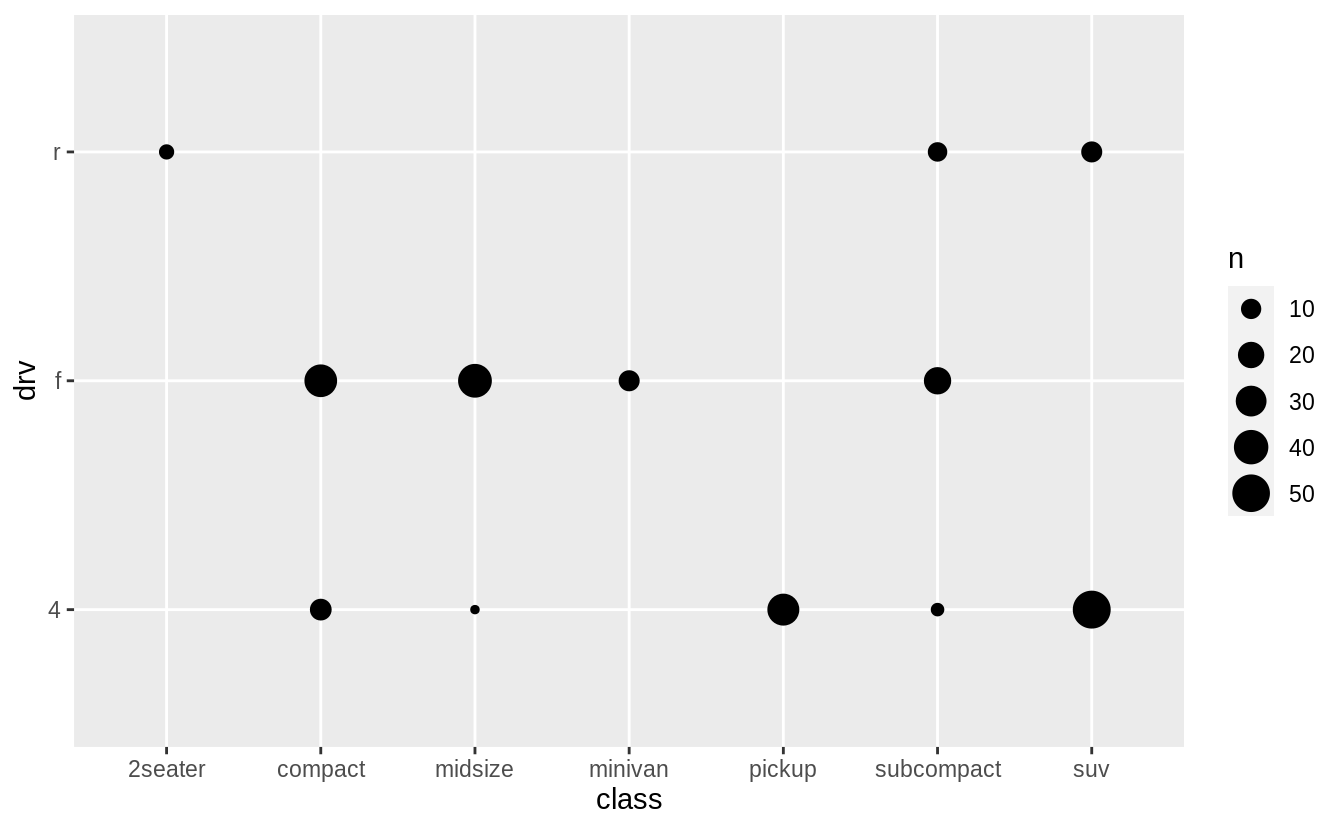
*#> # … with 6 more rows*

A simple scatter plot does not show how many observations there are for each (x, y) value. As such, scatterplots work best for plotting a continuous x and a continuous y variable, and when all (x, y) values are unique.

**Warning:** The following code uses functions introduced in a later section. Come back to this after reading section [7.5.2](https://r4ds.had.co.nz/exploratory-data-analysis.html#two-categorical-variables), which introduces methods for plotting two categorical variables. The first is geom\_count() which is similar to a scatterplot but uses the size of the points to show the number of observations at an (x, y) point.

**ggplot**(mpg, **aes**(x = class, y = drv)) +

**geom\_count**()



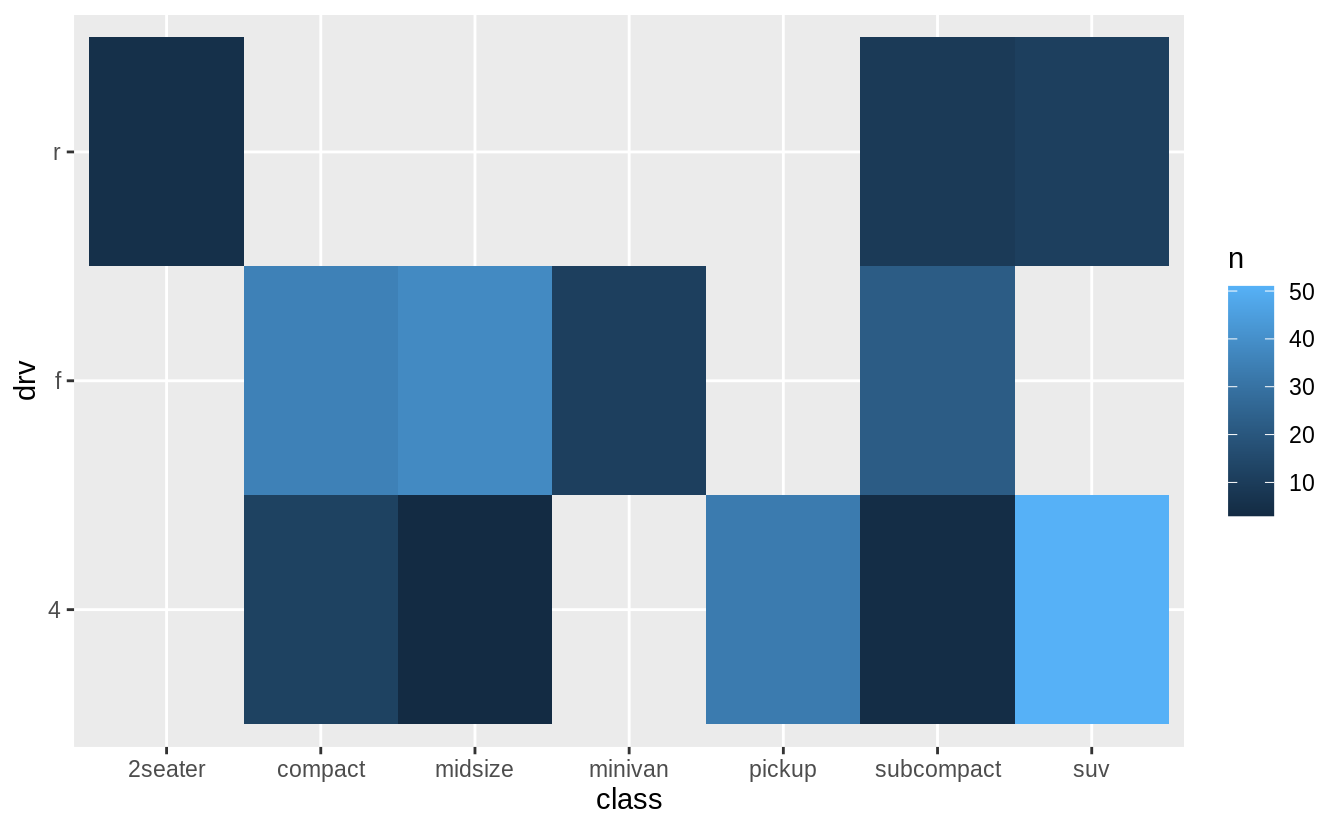
The second is geom\_tile() which uses a color scale to show the number of observations with each (x, y) value.

mpg %>%

**count**(class, drv) %>%

**ggplot**(**aes**(x = class, y = drv)) +

**geom\_tile**(mapping = **aes**(fill = n))



In the previous plot, there are many missing tiles. These missing tiles represent unobserved combinations of class and drv values. These missing values are not unknown, but represent values of (class, drv) where n = 0. The complete() function in the tidyr package adds new rows to a data frame for missing combinations of columns. The following code adds rows for missing combinations of class and drv and uses the fill argument to set n = 0 for those new rows.

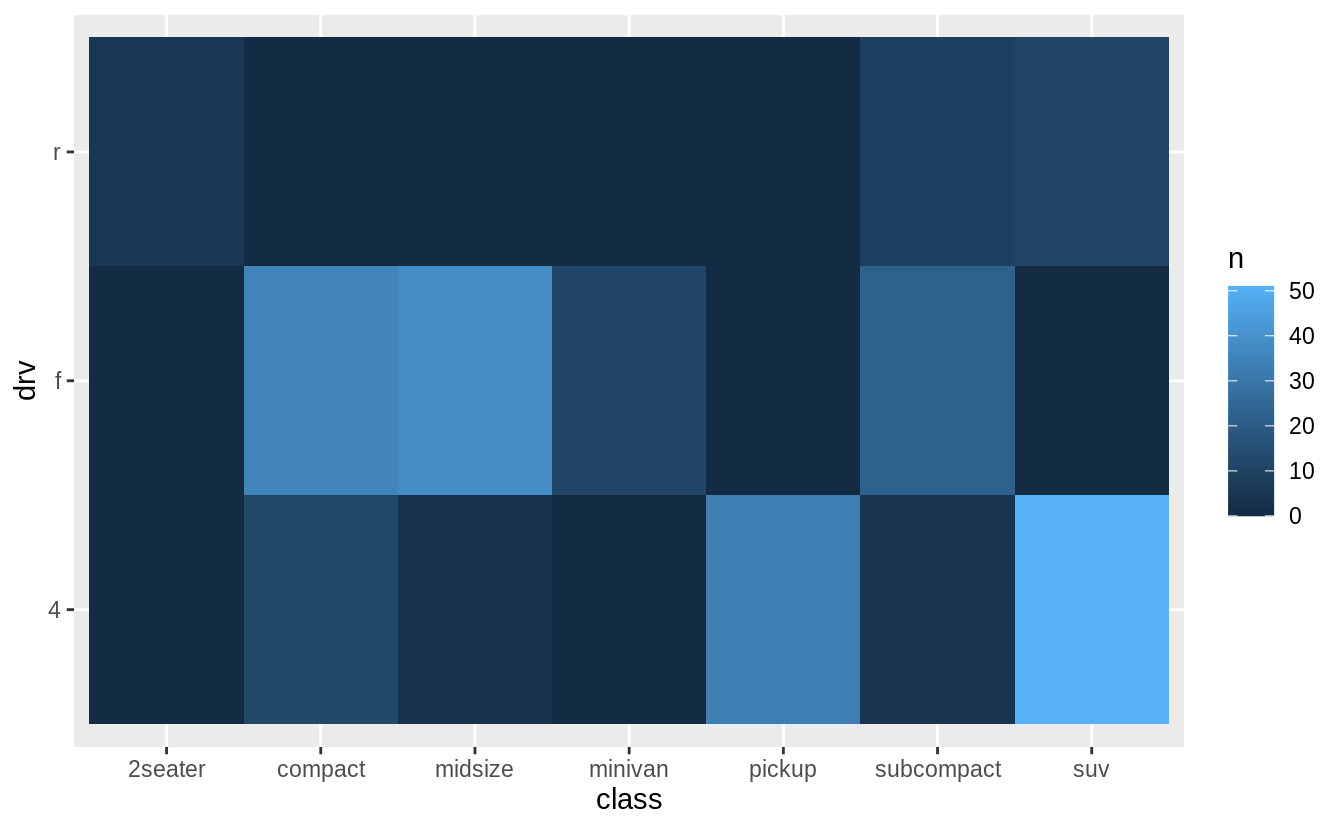
mpg %>%

**count**(class, drv) %>%

**complete**(class, drv, fill = **list**(n = 0)) %>%

**ggplot**(**aes**(x = class, y = drv)) +

**geom\_tile**(mapping = **aes**(fill = n))

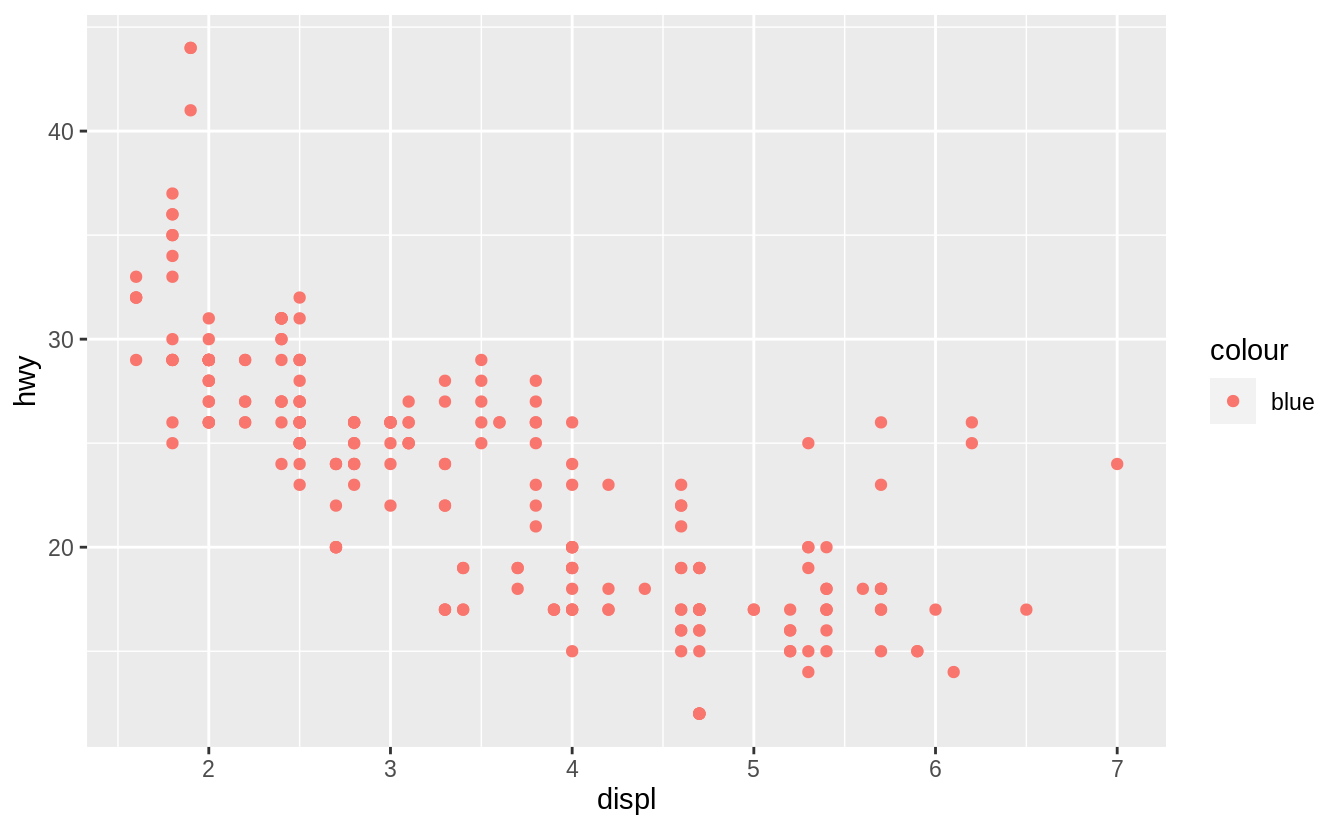


**שאלה 3**

What’s gone wrong with this code? Why are the points not blue?

**ggplot**(data = mpg) +

**geom\_point**(mapping = **aes**(x = displ, y = hwy, colour = "blue"))

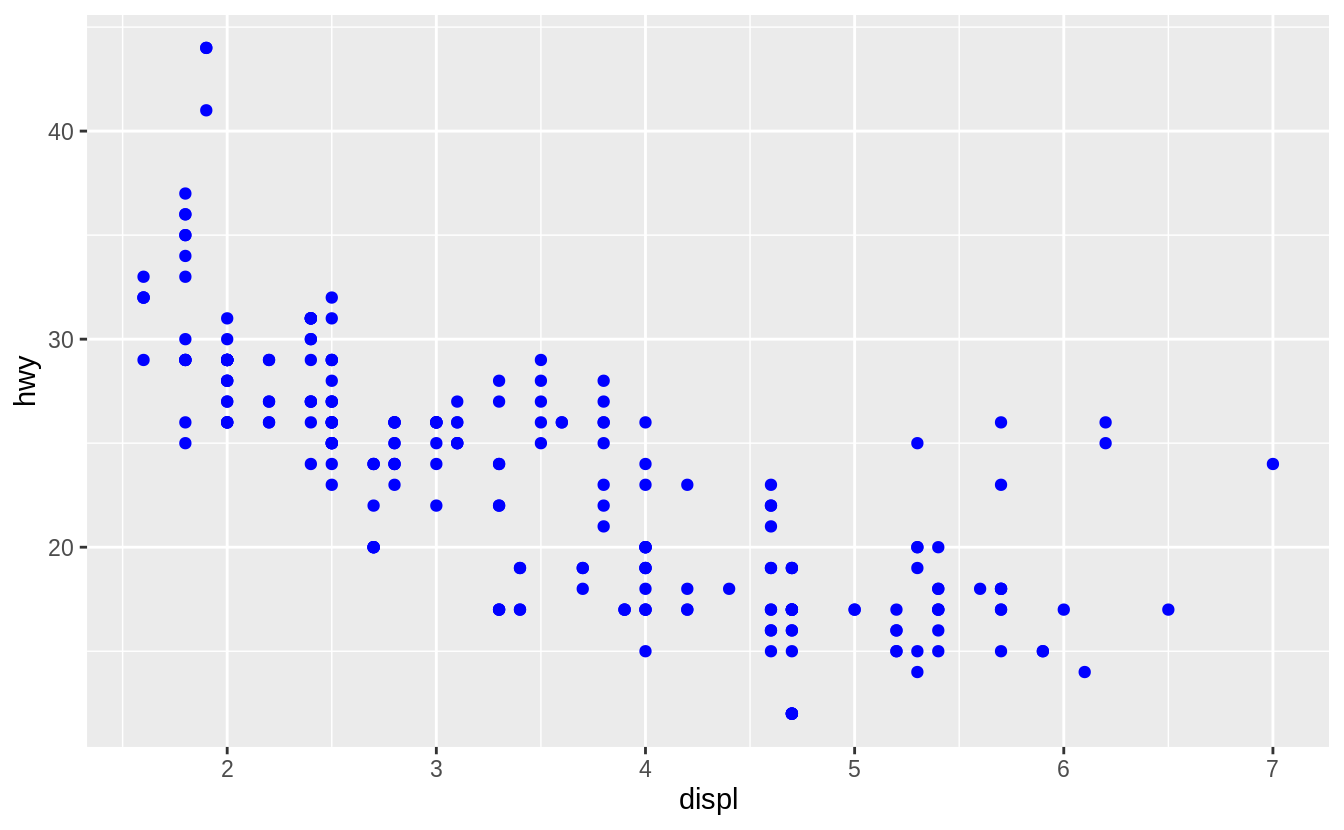


The argumentcolour = "blue" is included within the mapping argument, and as such, it is treated as an aesthetic, which is a mapping between a variable and a value. In the expression, colour = "blue", "blue" is interpreted as a categorical variable which only takes a single value "blue". If this is confusing, consider how colour = 1:234 and colour = 1 are interpreted by aes().

The following code does produces the expected result.

**ggplot**(data = mpg) +

**geom\_point**(mapping = **aes**(x = displ, y = hwy), colour = "blue")



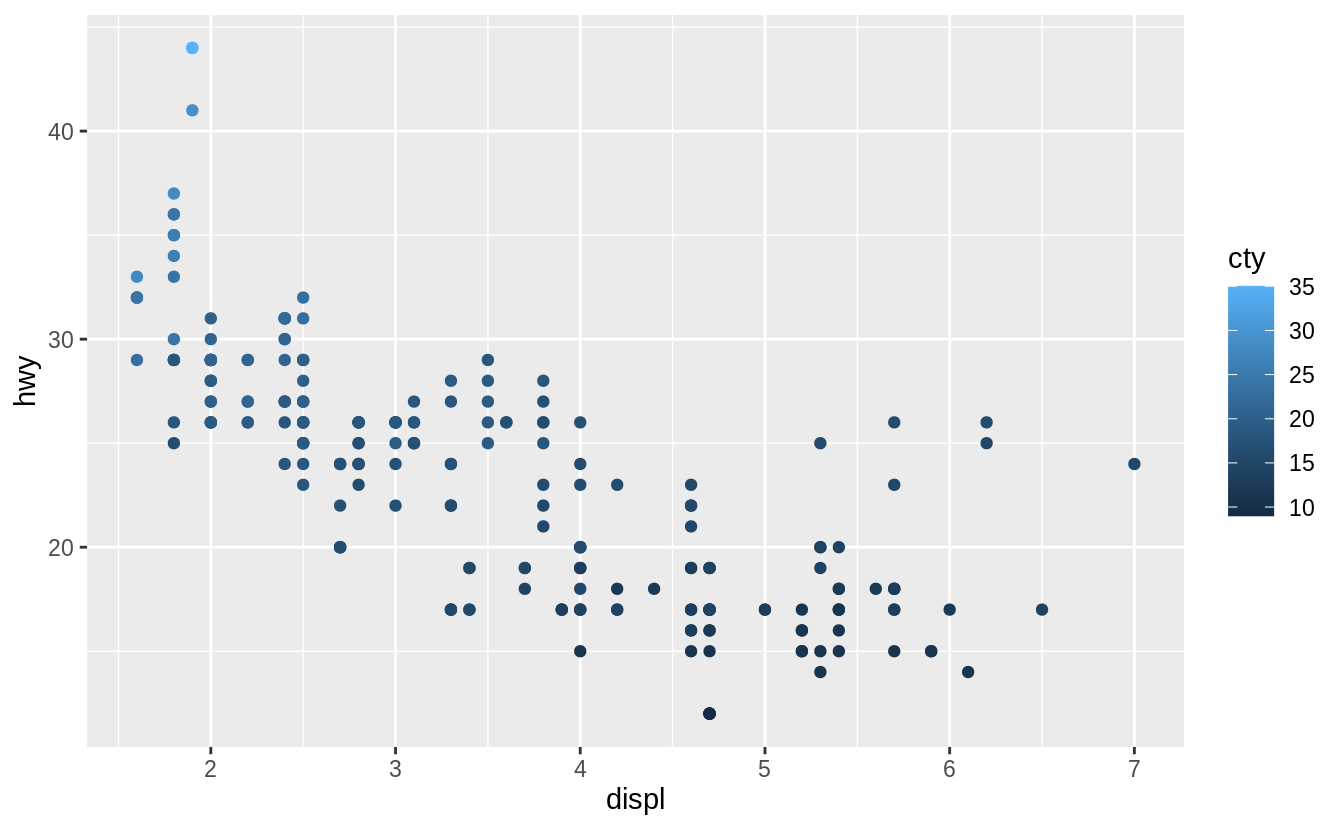
**שאלה 4**

Map a continuous variable to color, size, and shape. How do these aesthetics behave differently for categorical vs. continuous variables?

The variable cty, city highway miles per gallon, is a continuous variable.

**ggplot**(mpg, **aes**(x = displ, y = hwy, colour = cty)) +

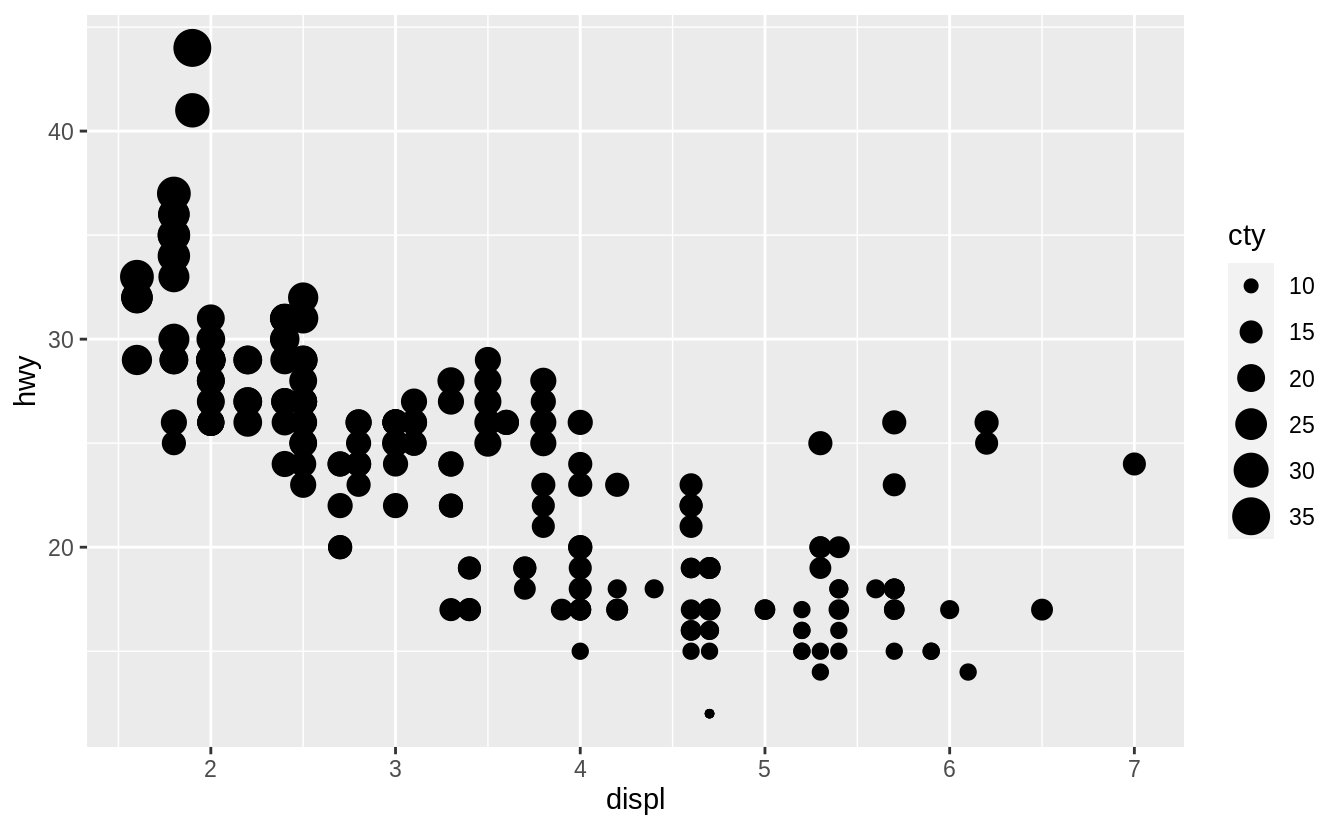
**geom\_point**()



Instead of using discrete colors, the continuous variable uses a scale that varies from a light to dark blue color.

**ggplot**(mpg, **aes**(x = displ, y = hwy, size = cty)) +

**geom\_point**()



When mapped to size, the sizes of the points vary continuously as a function of their size.

**ggplot**(mpg, **aes**(x = displ, y = hwy, shape = cty)) +

**geom\_point**()

*#> Error: A continuous variable can not be mapped to shape*

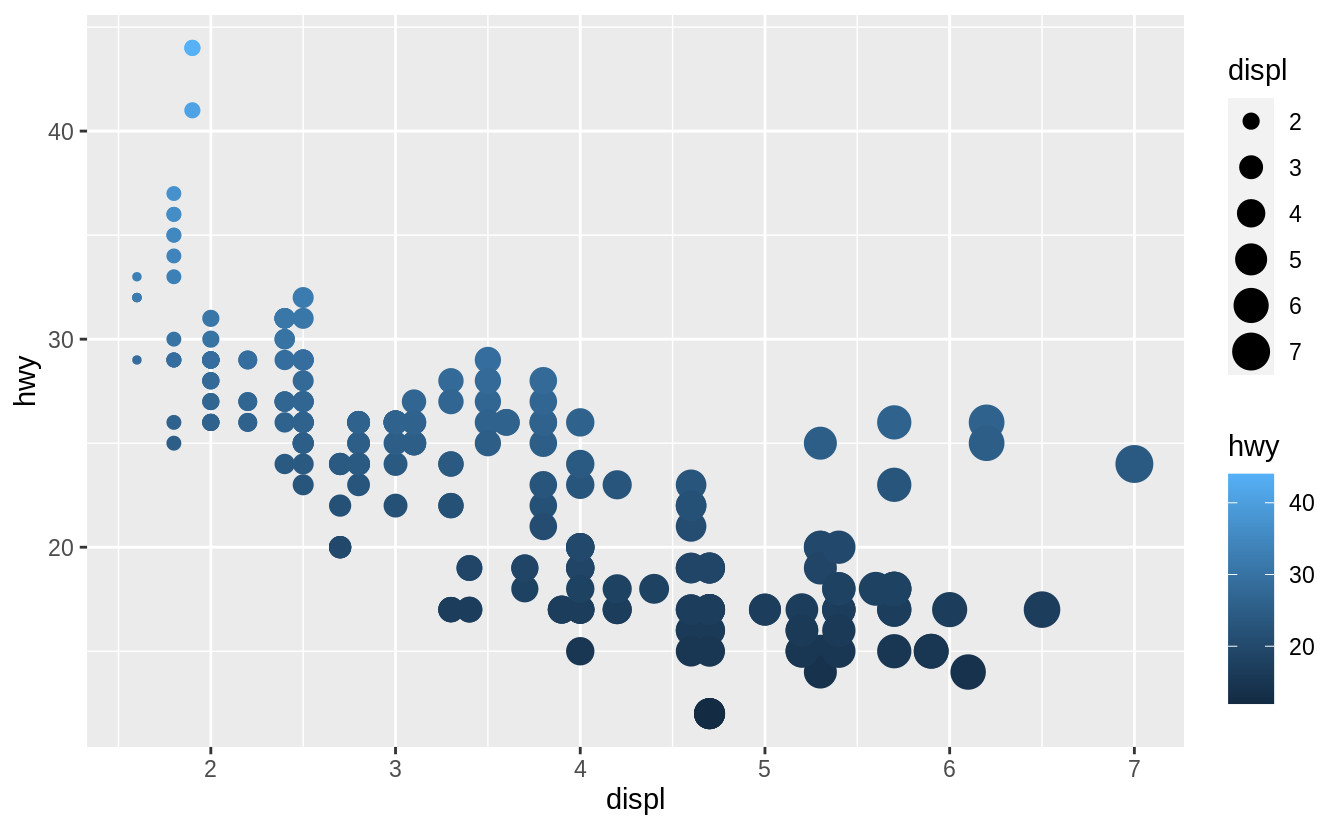
When a continuous value is mapped to shape, it gives an error. Though we could split a continuous variable into discrete categories and use a shape aesthetic, this would conceptually not make sense. A numeric variable has an order, but shapes do not. It is clear that smaller points correspond to smaller values, or once the color scale is given, which colors correspond to larger or smaller values. But it is not clear whether a square is greater or less than a circle.

**שאלה 5**

What happens if you map the same variable to multiple aesthetics?

**ggplot**(mpg, **aes**(x = displ, y = hwy, colour = hwy, size = displ)) +

**geom\_point**()



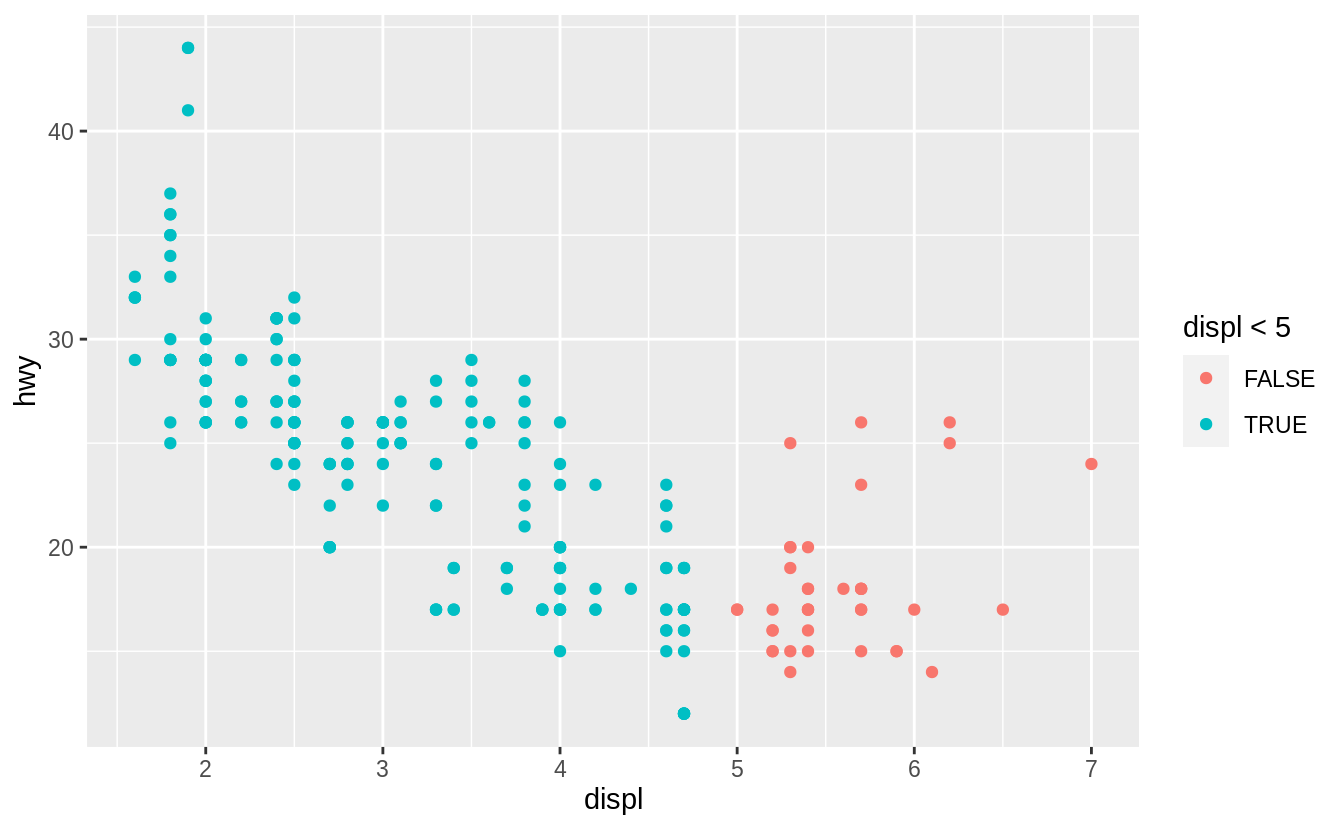
In the above plot, hwy is mapped to both location on the y-axis and color, and displ is mapped to both location on the x-axis and size. The code works and produces a plot, even if it is a bad one. Mapping a single variable to multiple aesthetics is redundant. Because it is redundant information, in most cases avoid mapping a single variable to multiple aesthetics.

**שאלה 6**

What happens if you map an aesthetic to something other than a variable name, like aes(colour = displ < 5)?

**ggplot**(mpg, **aes**(x = displ, y = hwy, colour = displ < 5)) +

**geom\_point**()



Aesthetics can also be mapped to expressions like displ < 5. The ggplot() function behaves as if a temporary variable was added to the data with values equal to the result of the expression. In this case, the result of displ < 5 is a logical variable which takes values of TRUE or FALSE.

This also explains why, in Exercise 3 the expression colour = "blue" created a categorical variable with only one category: “blue”.

**שאלה 7**

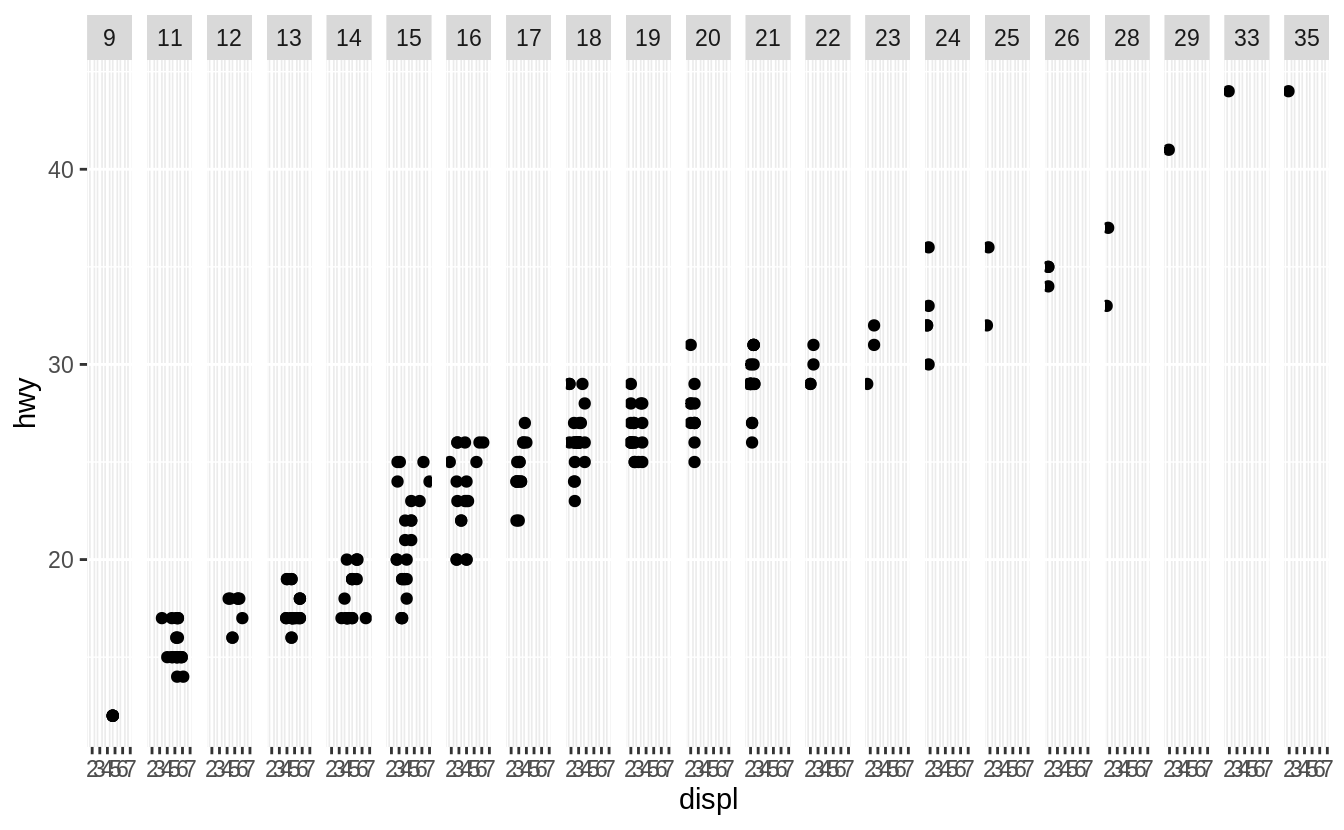
What happens if you facet on a continuous variable?

Let’s see.

**ggplot**(mpg, **aes**(x = displ, y = hwy)) +

**geom\_point**() +

**facet\_grid**(. ~ cty)



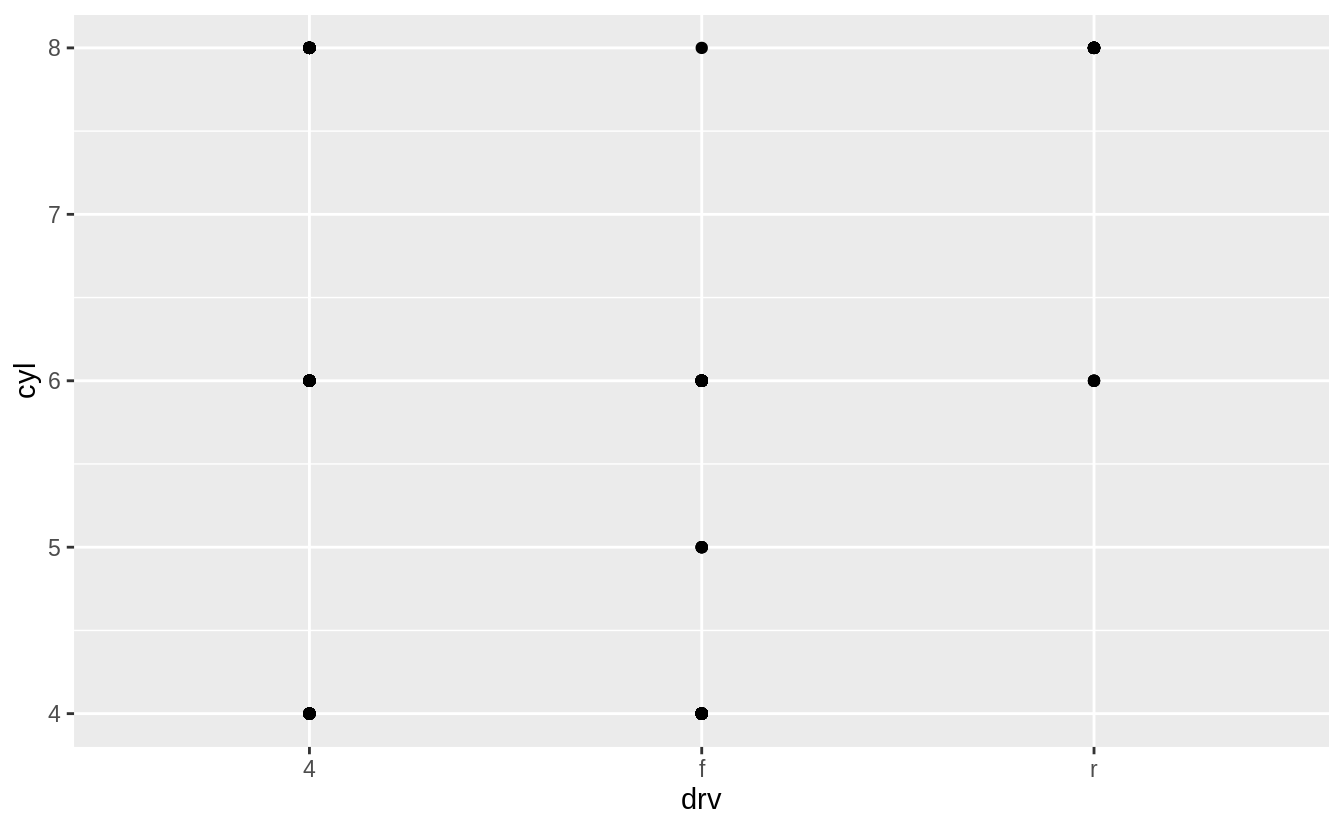
The continuous variable is converted to a categorical variable, and the plot contains a facet for each distinct value.

**שאלה 8**

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

**ggplot**(data = mpg) +

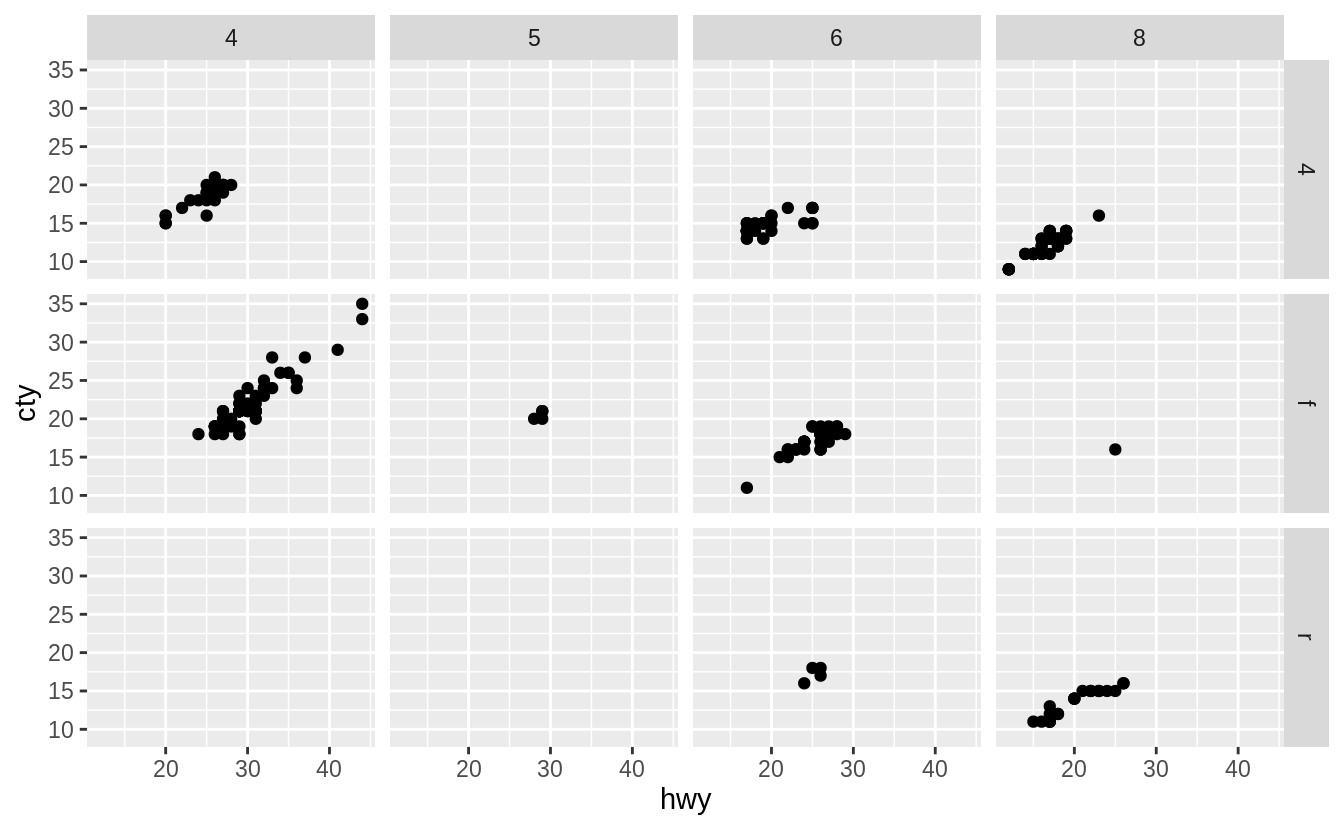
**geom\_point**(mapping = **aes**(x = drv, y = cyl))



**ggplot**(data = mpg) +

**geom\_point**(mapping = **aes**(x = hwy, y = cty)) +

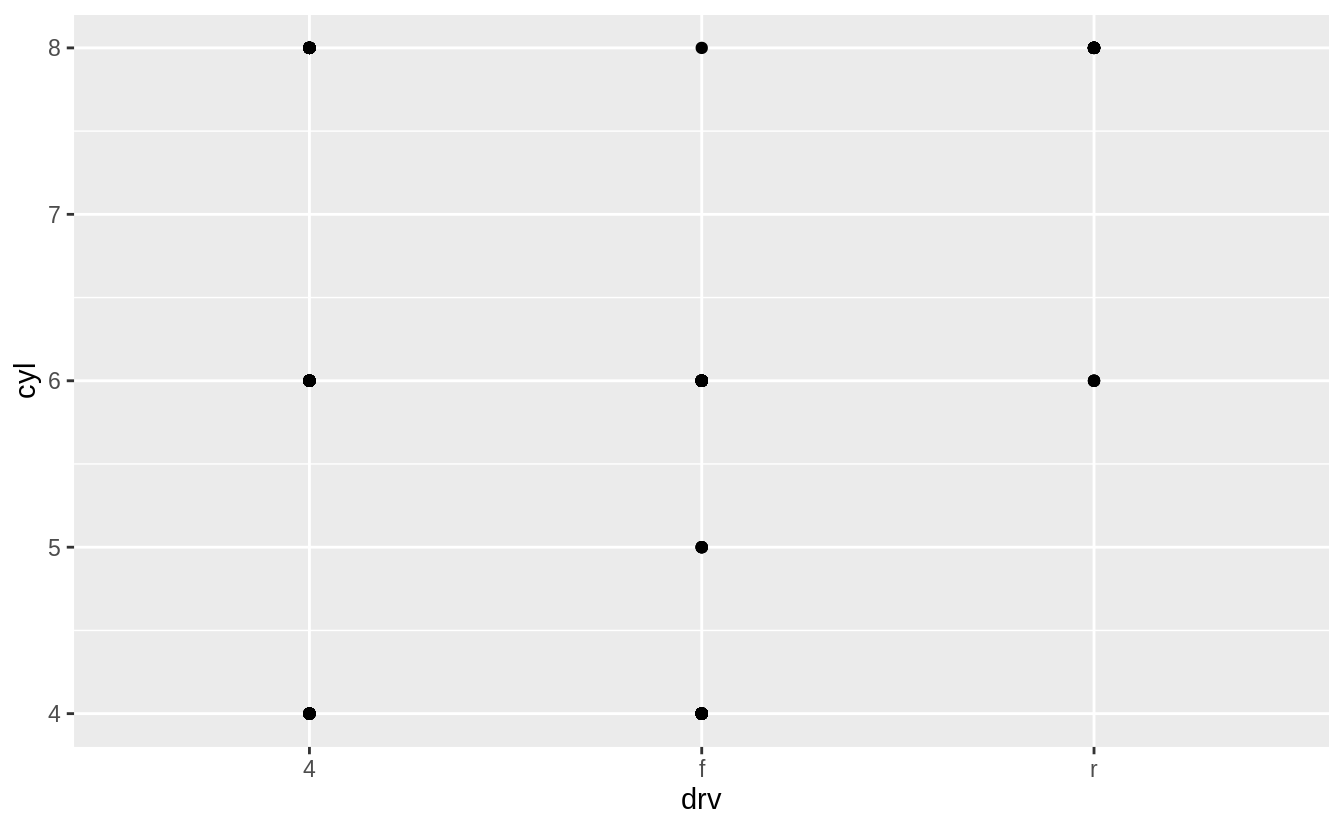
**facet\_grid**(drv ~ cyl)



The empty cells (facets) in this plot are combinations of drv and cyl that have no observations. These are the same locations in the scatter plot of drv and cyl that have no points.

**ggplot**(data = mpg) +

**geom\_point**(mapping = **aes**(x = drv, y = cyl))



**שאלה 9**

Read ?facet\_wrap. What does nrow do? What does ncol do? What other options control the layout of the individual panels? Why doesn’t facet\_grid() have nrow and ncol variables?

The arguments nrow (ncol) determines the number of rows (columns) to use when laying out the facets. It is necessary since facet\_wrap() only facets on one variable.

The nrow and ncol arguments are unnecessary for facet\_grid() since the number of unique values of the variables specified in the function determines the number of rows and columns.

**שאלה 10**

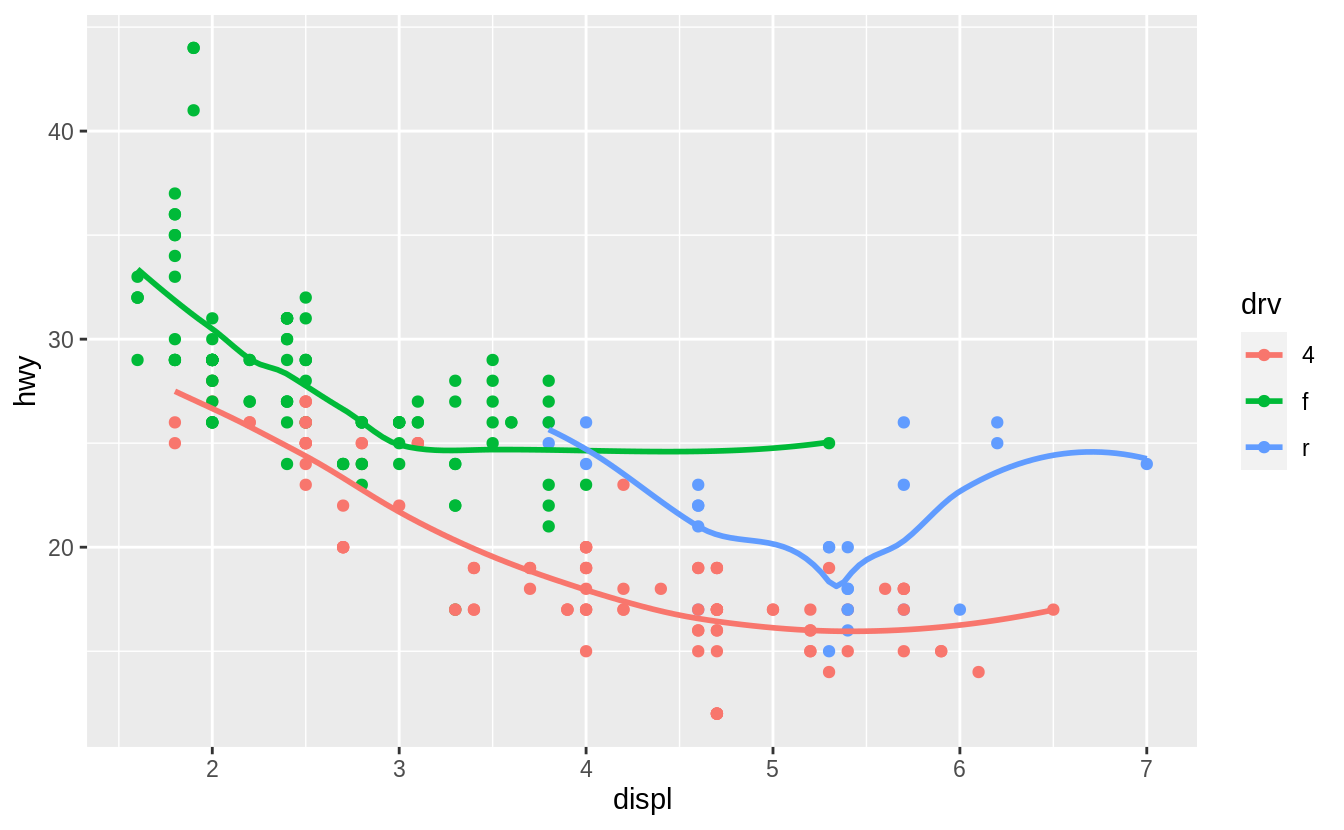
Run this code in your head and predict what the output will look like. Then, run the code in R and check your predictions.

**ggplot**(data = mpg, mapping = **aes**(x = displ, y = hwy, colour = drv)) +

**geom\_point**() +

**geom\_smooth**(se = FALSE)

*#> `geom\_smooth()` using method = 'loess' and formula 'y ~ x'*



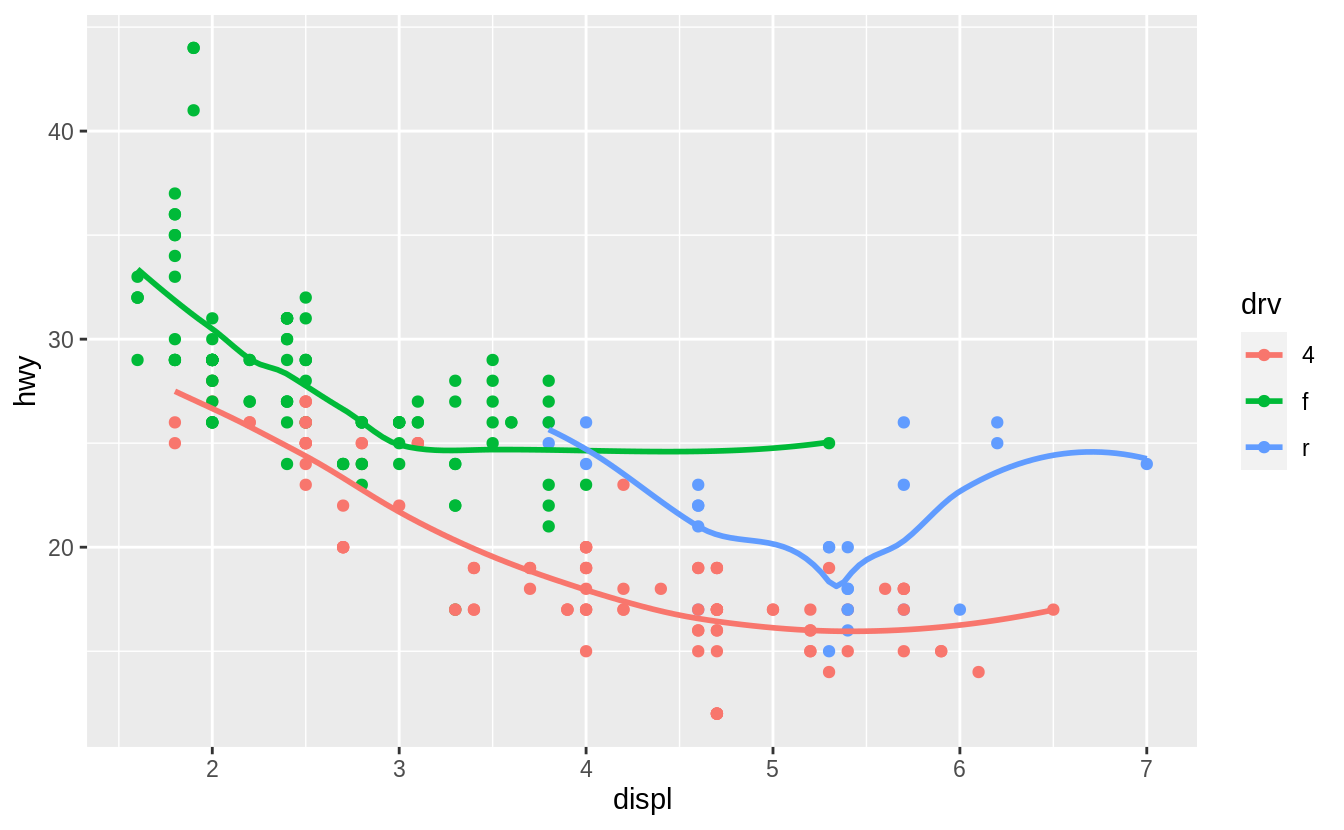
This code produces a scatter plot with displ on the x-axis, hwy on the y-axis, and the points colored by drv. There will be a smooth line, without standard errors, fit through each drv group.

**ggplot**(data = mpg, mapping = **aes**(x = displ, y = hwy, colour = drv)) +

**geom\_point**() +

**geom\_smooth**(se = FALSE)

*#> `geom\_smooth()` using method = 'loess' and formula 'y ~ x'*



**שאלה 11**

Will these two graphs look different? Why/why not?

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

**geom\_point**() +

**geom\_smooth**()

**ggplot**() +

**geom\_point**(data = mpg, mapping = **aes**(x = displ, y = hwy)) +

**geom\_smooth**(data = mpg, mapping = **aes**(x = displ, y = hwy))

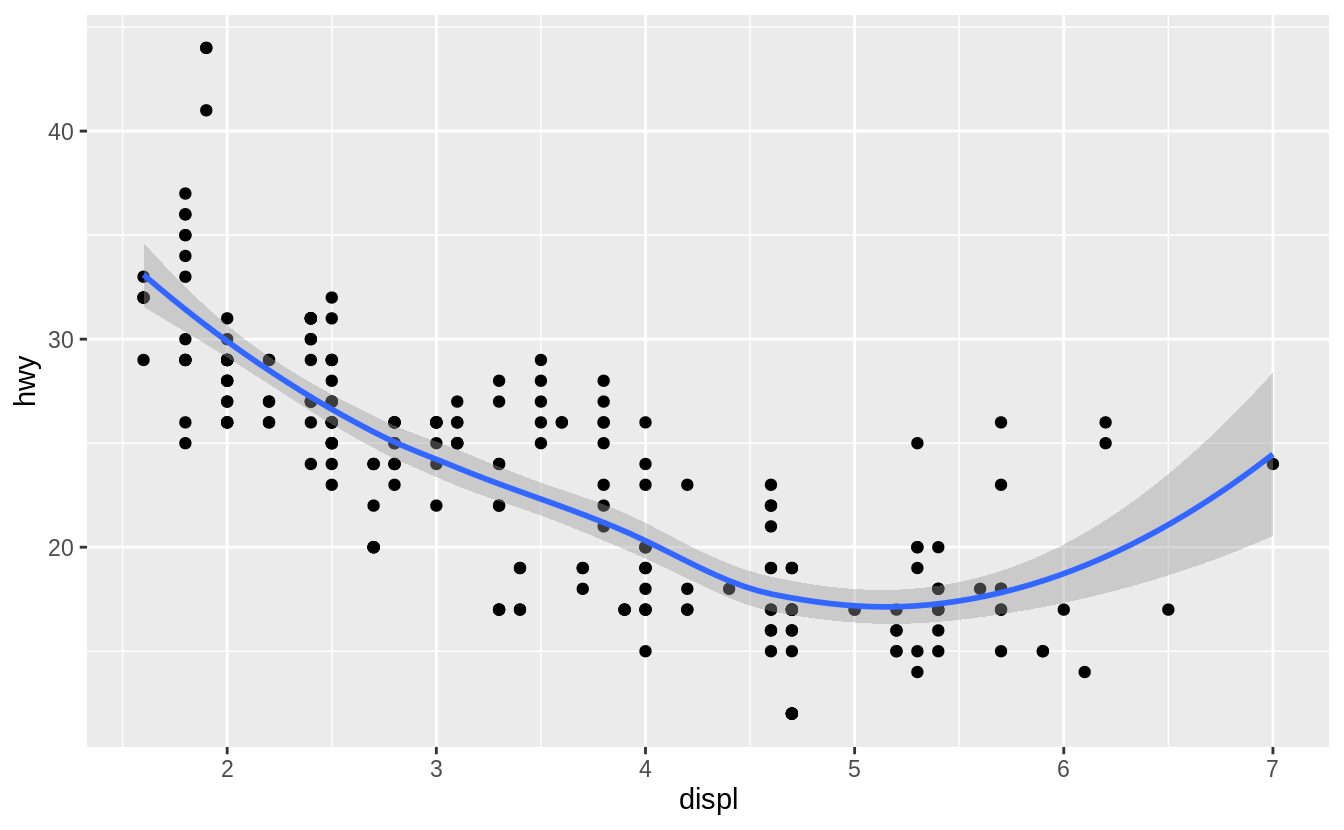
No. Because both geom\_point() and geom\_smooth() will use the same data and mappings. They will inherit those options from the ggplot() object, so the mappings don’t need to specified again.

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

**geom\_point**() +

**geom\_smooth**()

*#> `geom\_smooth()` using method = 'loess' and formula 'y ~ x'*

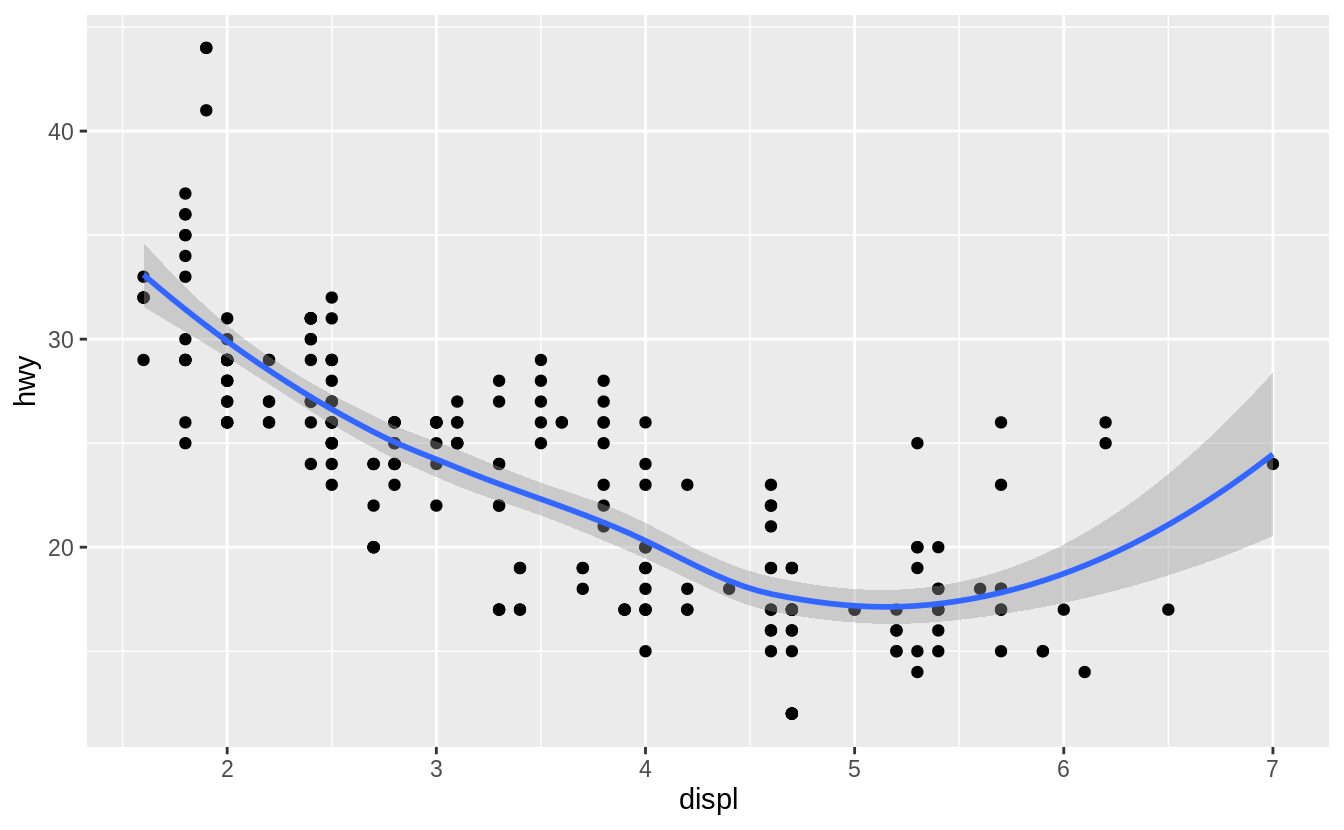


**ggplot**() +

**geom\_point**(data = mpg, mapping = **aes**(x = displ, y = hwy)) +

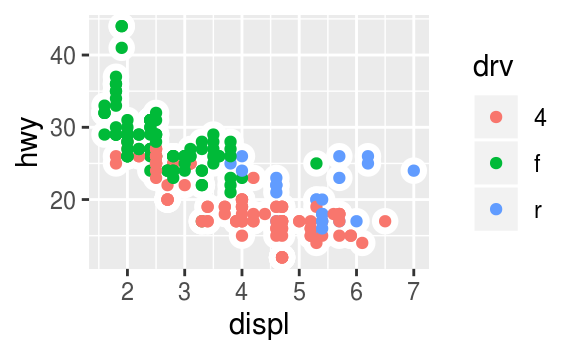
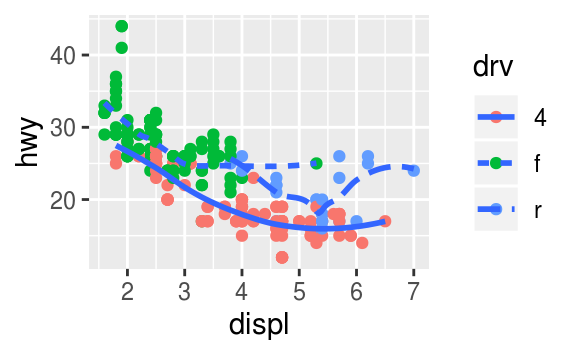
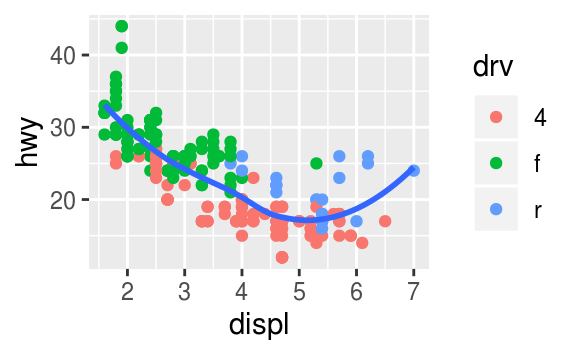
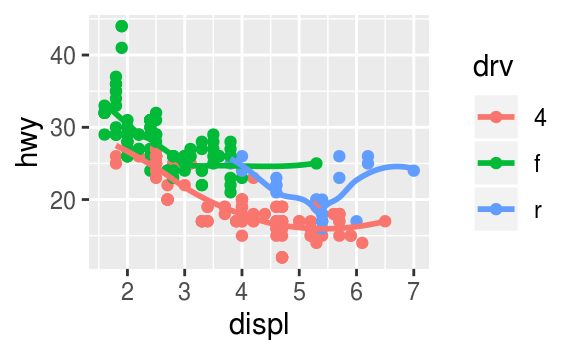
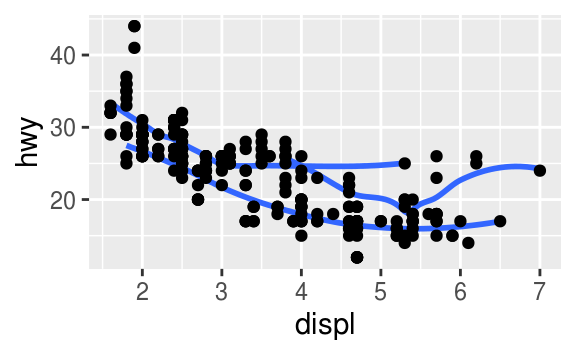
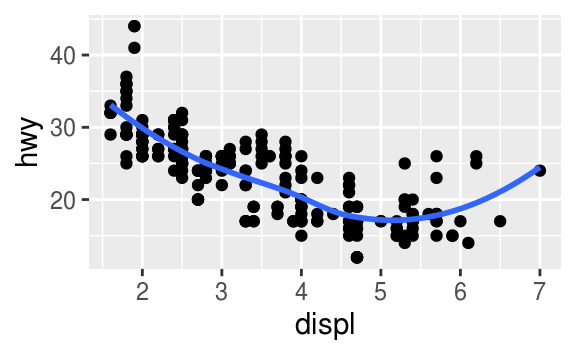
**geom\_smooth**(data = mpg, mapping = **aes**(x = displ, y = hwy))

*#> `geom\_smooth()` using method = 'loess' and formula 'y ~ x'*



**שאלה 12**

Recreate the R code necessary to generate the following graphs.

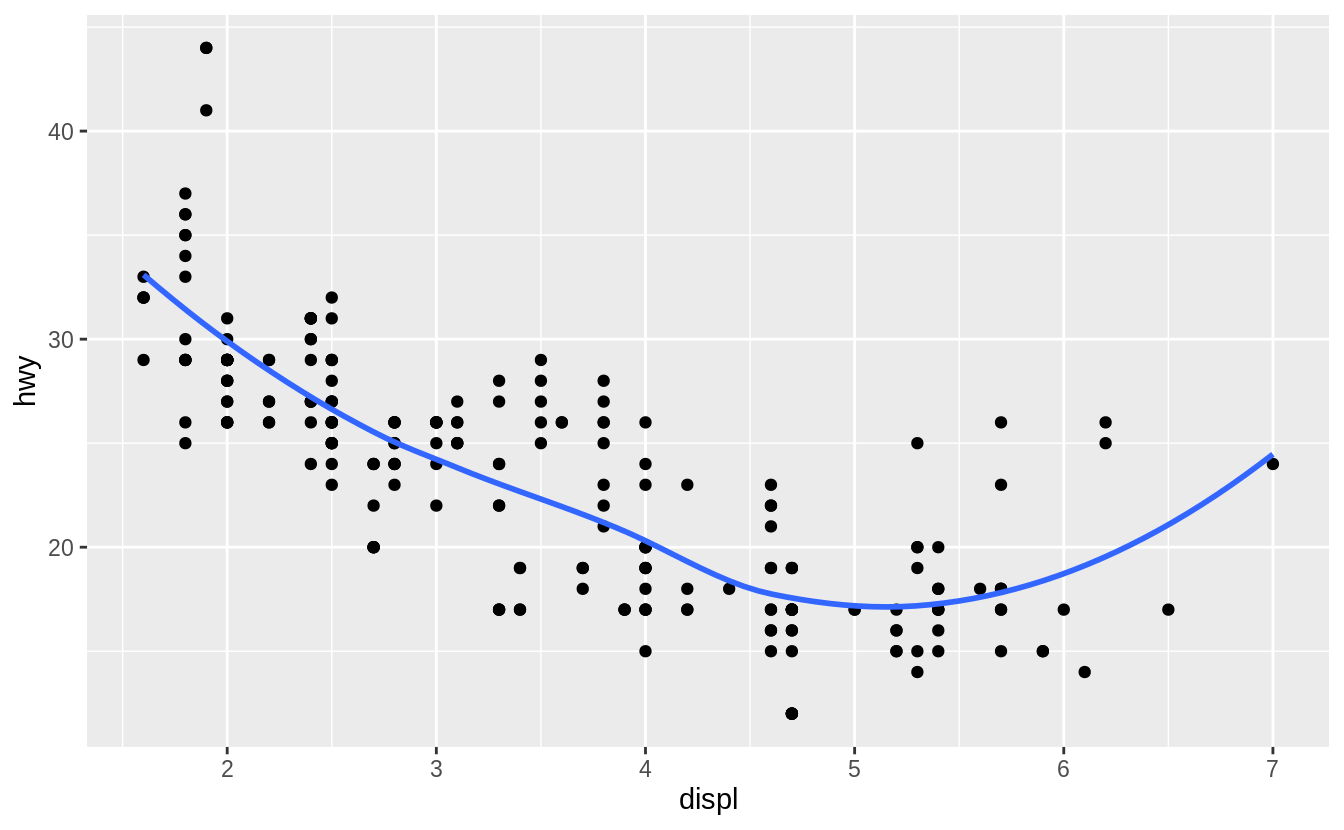


The following code will generate those plots.

**ggplot**(mpg, **aes**(x = displ, y = hwy)) +

**geom\_point**() +

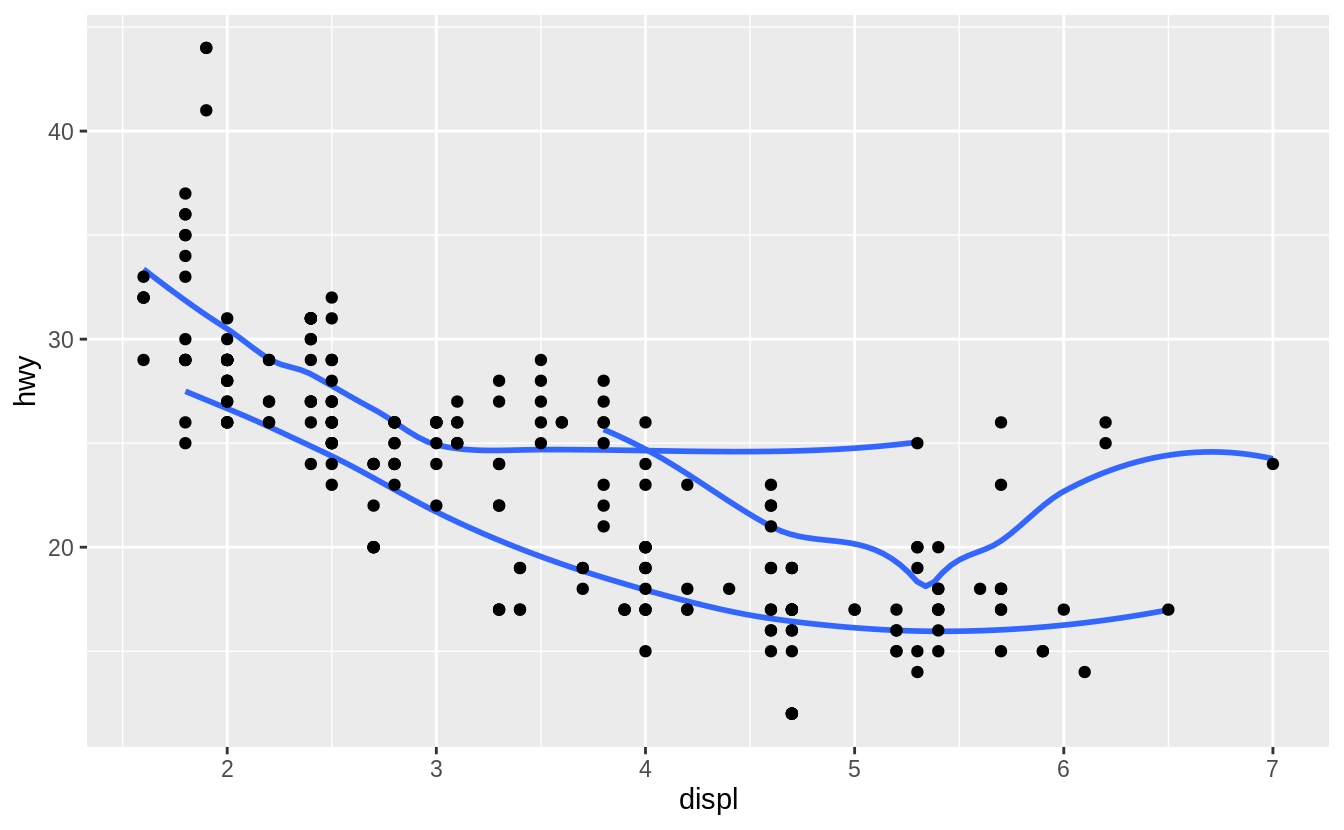
**geom\_smooth**(se = FALSE)



**ggplot**(mpg, **aes**(x = displ, y = hwy)) +

**geom\_smooth**(mapping = **aes**(group = drv), se = FALSE) +

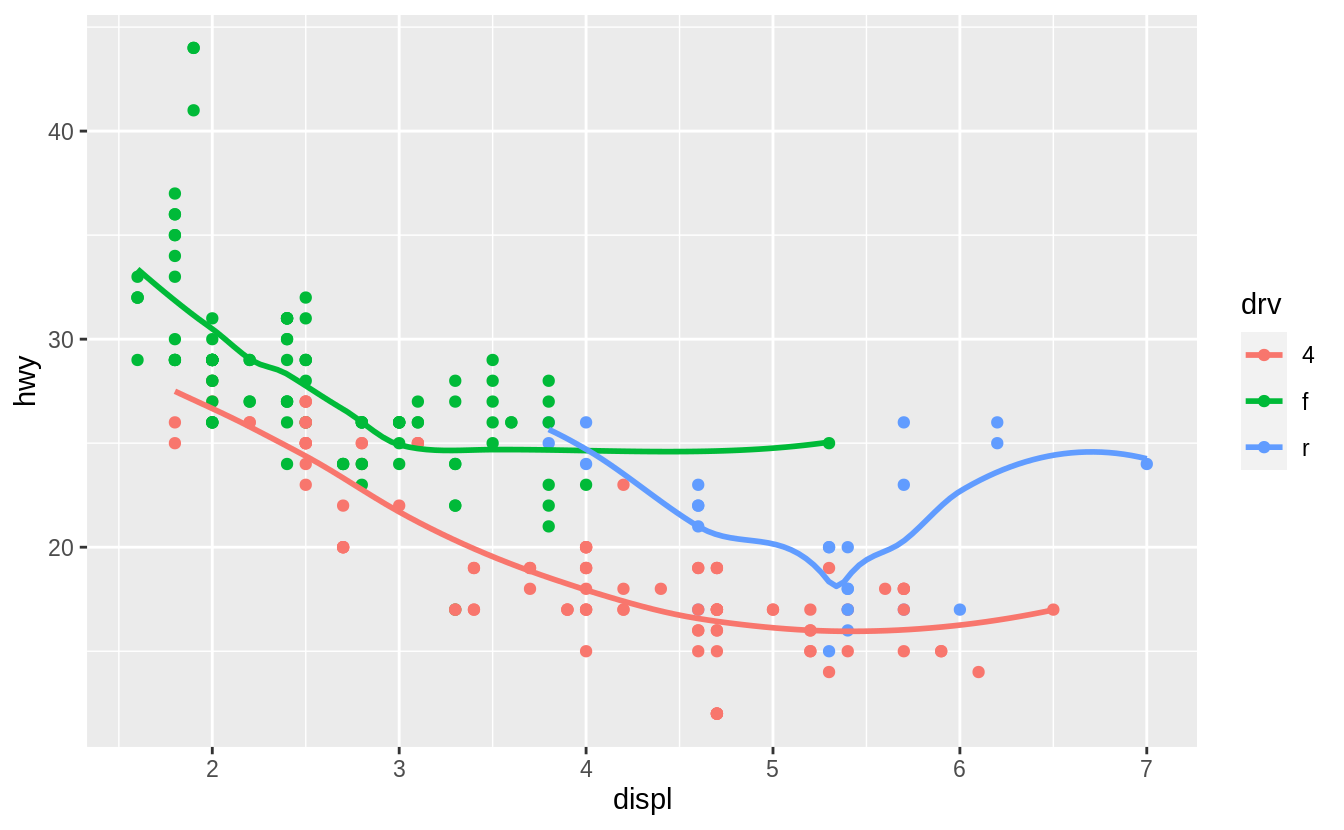
**geom\_point**()



**ggplot**(mpg, **aes**(x = displ, y = hwy, colour = drv)) +

**geom\_point**() +

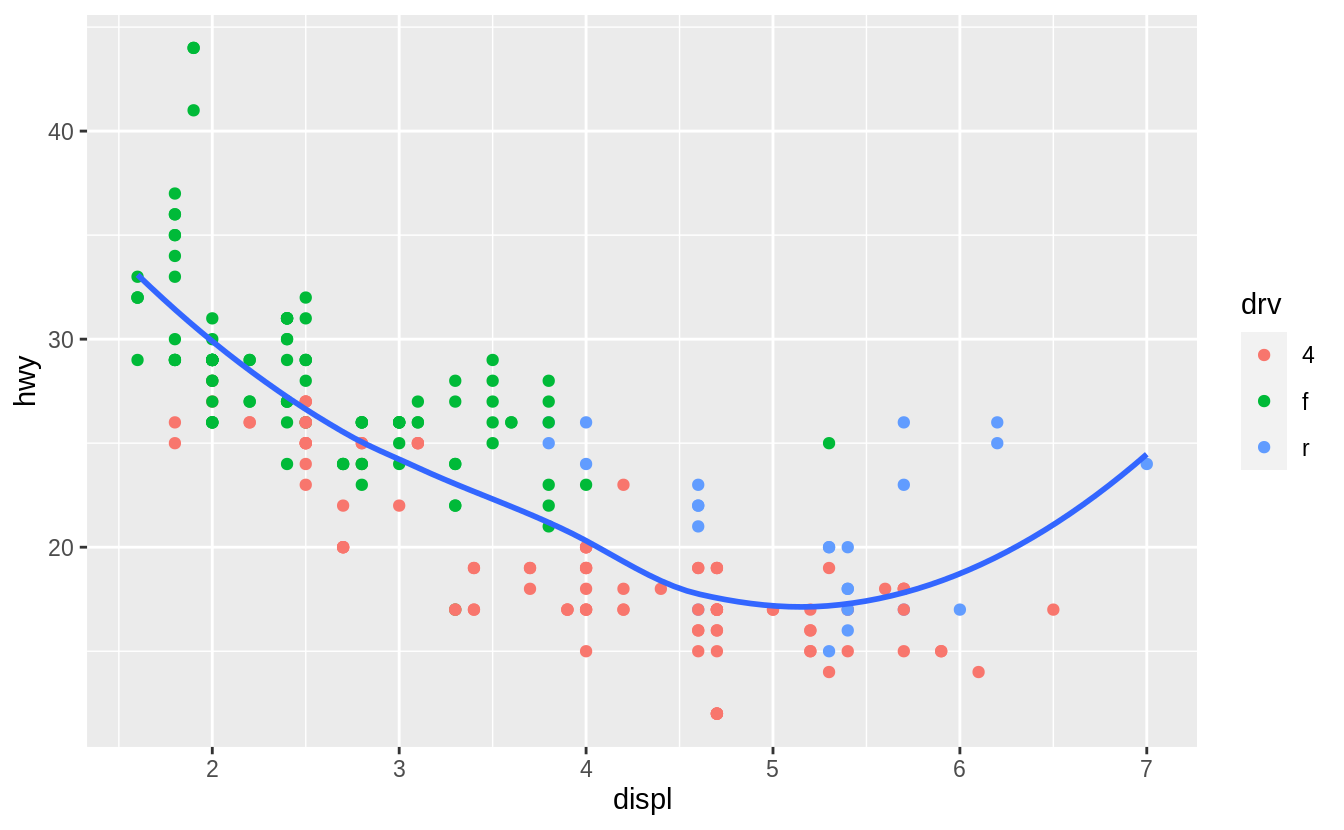
**geom\_smooth**(se = FALSE)



**ggplot**(mpg, **aes**(x = displ, y = hwy)) +

**geom\_point**(**aes**(colour = drv)) +

**geom\_smooth**(se = FALSE)

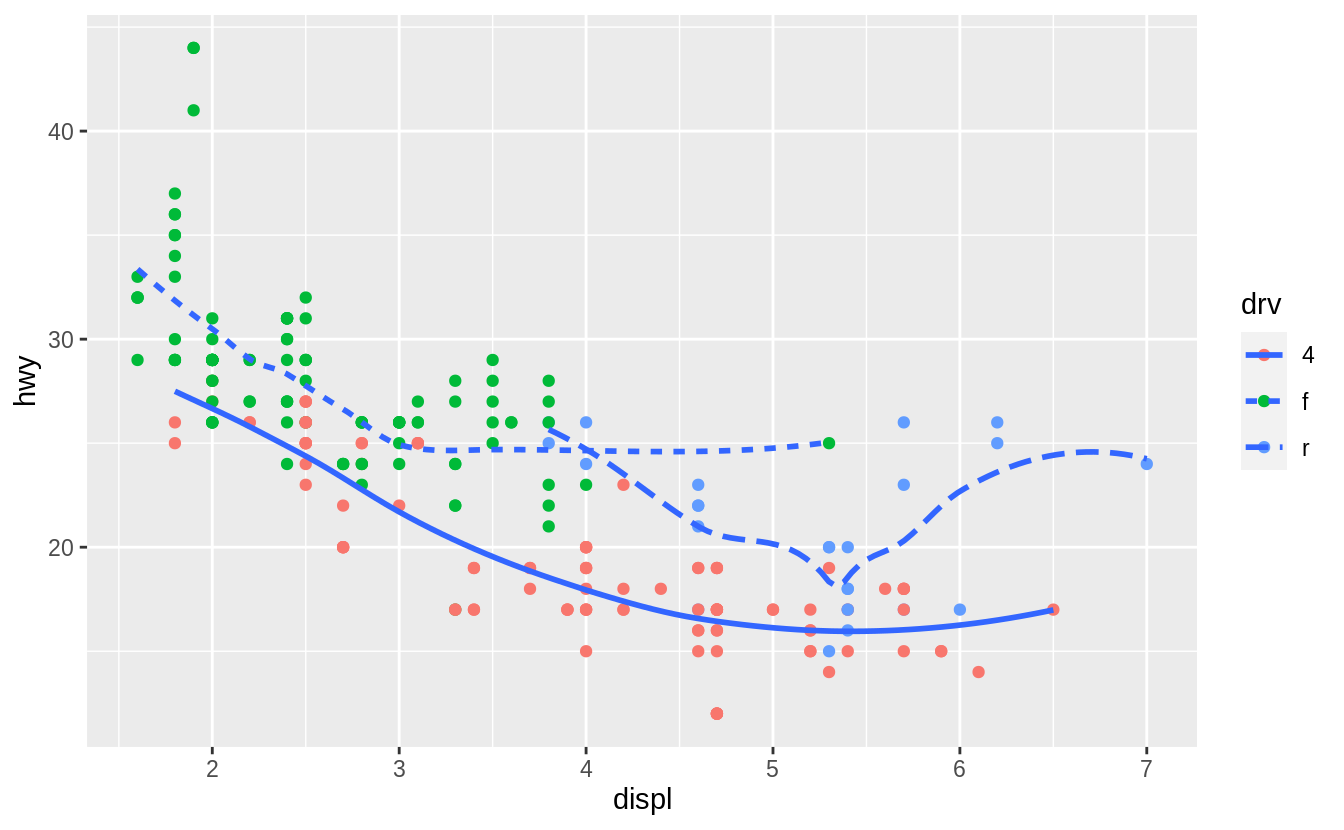


**ggplot**(mpg, **aes**(x = displ, y = hwy)) +

**geom\_point**(**aes**(colour = drv)) +

**geom\_smooth**(**aes**(linetype = drv), se = FALSE)

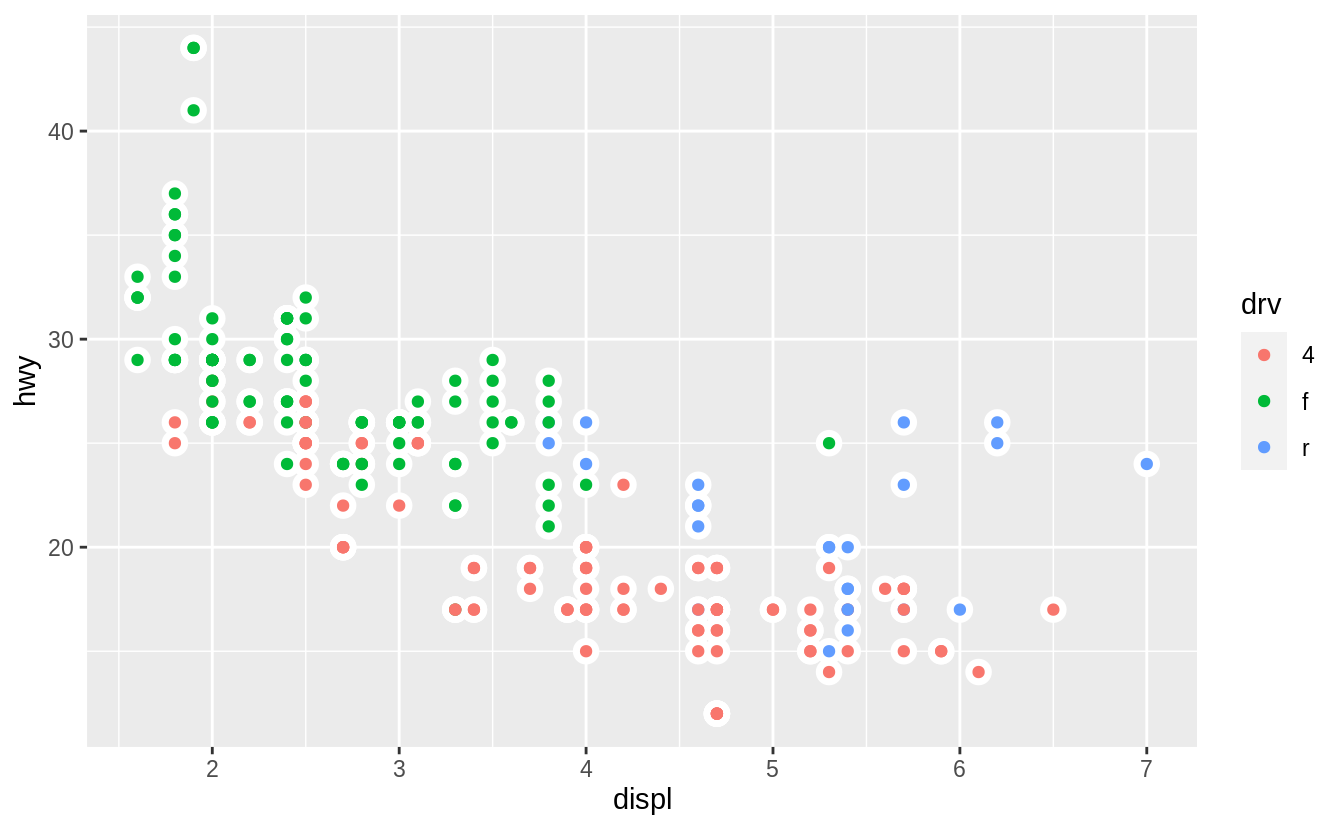
*#> `geom\_smooth()` using method = 'loess' and formula 'y ~ x'*



**ggplot**(mpg, **aes**(x = displ, y = hwy)) +

**geom\_point**(size = 4, color = "white") +

**geom\_point**(**aes**(colour = drv))



**שאלה 13**

1. ייבא את נתוני Tara אשר השתמשנו בהם בשיעור הראשון, והרץ את הקוד הבא אשר יכין את הנתונים:

tara <- readxl**::read\_xlsx**(path = "../1 introduction/1\_data/Tara.xlsx", sheet = "2010")

**names**(tara) <- **c**("date", "cat", "q\_k", "revenue", "vc", "fc")

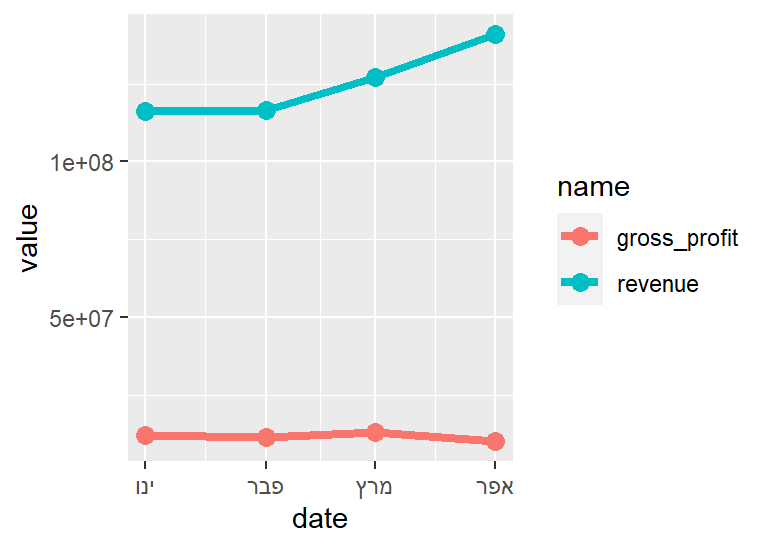
tara <- **mutate**(tara, gross\_profit = revenue **-** vc **-** fc)

tara**$**date <- **parse\_date**(tara**$**date, format = "%Y-%m-%d")

carton\_milk <- tara **%>%** **filter**(**str\_detect**(cat, pattern = "חלב קרטון")) **%>%**

**select**(date,revenue, gross\_profit)

הגרף הבא מציג את הרווח הגולמי והפדיון של חלב בקרטון. שחזר את הגרף תוך שימוש ב pivot\_longer.



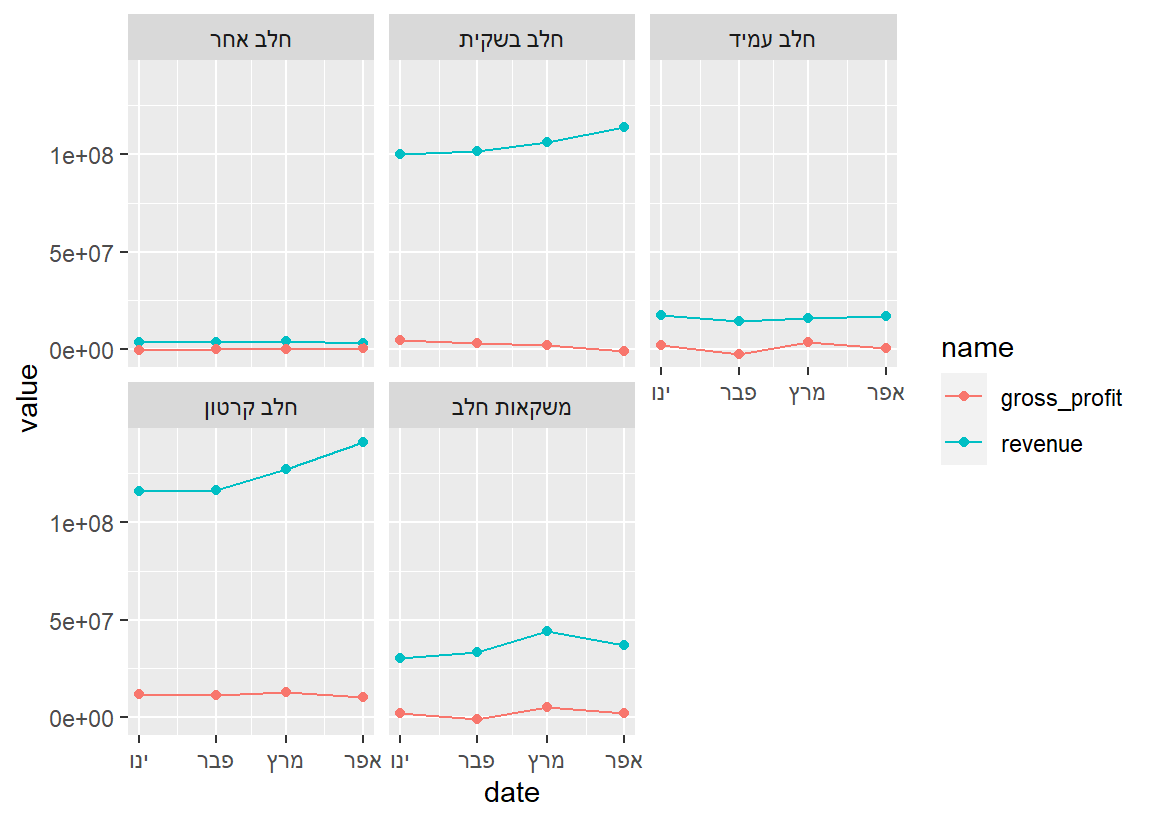
בנוסף, הקוד הבא מסנן את מעדני החלב:

milk <- tara **%>%** **filter**(**str\_detect**(cat, pattern = "חלב"))

milk <- milk **%>%** **filter**( **!str\_detect**(cat, pattern = "מעדני"))

milk <- milk **%>%** **select**(date, revenue, gross\_profit, cat)

הגרף הבא מציג את הרווח הגולמי והפדיון עבור משקאות חלב. שחזר את הגרף תוך שימוש ב- pivot\_longer ושימוש ב-facet\_wrap.



**פתרון:**

הקוד לפתרון לא הוסתר ונמצא בקובץ ה Rmd שהוצג בשיעור.

הקוד לגרף הראשון הוא:

pivot\_longer(carton\_milk,2:3) %>%

ggplot(aes(x = date, y = value, color = name)) + geom\_line(size = 1.5) + geom\_point(size = 3)

הקוד לגרף השני הוא:

milk %>% pivot\_longer(cols = c(revenue, gross\_profit)) %>%

ggplot(aes(x = date, y = value, color = name)) + geom\_point() + geom\_line() +

facet\_wrap(~ cat)