problemset1a.R

anirudh

2023-09-19

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.3     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
# First steps - R
# Introduction to R & Data Visualization

# Part 1: R Questions
# Question 1: First Steps with R
# Part 4: Generating sequences of numbers and logical vectors
# 1.
seq(1:15)
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
```

```
# 2.
2*1:15
```

```
##  [1]  2  4  6  8 10 12 14 16 18 20 22 24 26 28 30
```

```
(2*1):15
```

```
##  [1]  2  3  4  5  6  7  8  9 10 11 12 13 14 15
```

```
# There is a difference here. In the first one, it's skipping the value by 2 numbers.
# In the second one, it's starting from the number 2 instead of 1.
# 3.
# A logical vector only consists of values 0 and 1 or True or False
vec <- runif(10, min = 0, max = 3)
vec_more_1.5 <- vec[vec > 1.5]
vec_more_1.5
```

```
## [1] 2.403222 2.209297 2.622650 2.898517
```

```
# 4.
vec_equal_1.2 <- vec[vec == 1.2]
vec_equal_1.2
```

```
## numeric(0)
```

```
# I was expecting a 0 because 1.2 is a very specific number to expect in a randomly generated vector
# 5.
?seq
seq(from = 15, to = 1)
```

```
##  [1] 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1
```

```
# Part 5: Arrays & Matrices
# 1.
x <- runif(100)
x
```

```
##   [1] 0.659478828 0.380226362 0.800692990 0.785761222 0.600894168 0.904707217
##   [7] 0.284712873 0.725831782 0.318796805 0.851004535 0.807857111 0.814198268
##  [13] 0.734832927 0.283199710 0.924963735 0.162544179 0.342285622 0.990165519
##  [19] 0.702170232 0.630128546 0.253773024 0.308152263 0.043414640 0.227923824
##  [25] 0.973692215 0.019666349 0.007388633 0.210812387 0.019205903 0.583211335
##  [31] 0.913057485 0.174814483 0.052148457 0.924308672 0.938043522 0.833015616
##  [37] 0.830134898 0.326950809 0.260545353 0.070827676 0.190997397 0.342037965
##  [43] 0.879228737 0.624236160 0.316455274 0.210685254 0.164217372 0.462215300
##  [49] 0.310923022 0.576316157 0.238342189 0.848432293 0.329860216 0.690994091
##  [55] 0.058853804 0.047236713 0.083546347 0.115713447 0.327155857 0.024377362
##  [61] 0.975626413 0.518508662 0.050742500 0.623748417 0.016563896 0.893278242
##  [67] 0.228000259 0.265731229 0.578588570 0.842822355 0.104046014 0.406164668
##  [73] 0.594460348 0.254221747 0.207385767 0.289859073 0.501668638 0.943765767
##  [79] 0.296401632 0.697809929 0.635466391 0.880022825 0.326436355 0.328959376
##  [85] 0.823319244 0.373875335 0.579558837 0.194886493 0.800521270 0.343988652
##  [91] 0.576767597 0.314426817 0.864148528 0.322703713 0.658320538 0.663131212
##  [97] 0.735854509 0.203508660 0.970714187 0.823556699
```

```
# 2.
matrix_x <- matrix(x, nrow = 10)
matrix_x
```

```
##            [,1]      [,2]        [,3]       [,4]      [,5]       [,6]      [,7]
##  [1,] 0.6594788 0.8078571 0.253773024 0.91305749 0.1909974 0.23834219 0.9756264
##  [2,] 0.3802264 0.8141983 0.308152263 0.17481448 0.3420380 0.84843229 0.5185087
##  [3,] 0.8006930 0.7348329 0.043414640 0.05214846 0.8792287 0.32986022 0.0507425
##  [4,] 0.7857612 0.2831997 0.227923824 0.92430867 0.6242362 0.69099409 0.6237484
##  [5,] 0.6008942 0.9249637 0.973692215 0.93804352 0.3164553 0.05885380 0.0165639
##  [6,] 0.9047072 0.1625442 0.019666349 0.83301562 0.2106853 0.04723671 0.8932782
##  [7,] 0.2847129 0.3422856 0.007388633 0.83013490 0.1642174 0.08354635 0.2280003
##  [8,] 0.7258318 0.9901655 0.210812387 0.32695081 0.4622153 0.11571345 0.2657312
##  [9,] 0.3187968 0.7021702 0.019205903 0.26054535 0.3109230 0.32715586 0.5785886
## [10,] 0.8510045 0.6301285 0.583211335 0.07082768 0.5763162 0.02437736 0.8428224
##            [,8]      [,9]     [,10]
##  [1,] 0.1040460 0.6354664 0.5767676
##  [2,] 0.4061647 0.8800228 0.3144268
##  [3,] 0.5944603 0.3264364 0.8641485
##  [4,] 0.2542217 0.3289594 0.3227037
##  [5,] 0.2073858 0.8233192 0.6583205
##  [6,] 0.2898591 0.3738753 0.6631312
##  [7,] 0.5016686 0.5795588 0.7358545
##  [8,] 0.9437658 0.1948865 0.2035087
##  [9,] 0.2964016 0.8005213 0.9707142
## [10,] 0.6978099 0.3439887 0.8235567
```

```
dim(matrix_x)
```

```
## [1] 10 10
```

```
# 3.
y <- matrix(runif(100), 10, 10)
# 4.
# entire column
y[0,]
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
```

```
# entire row
y[,0]
```

```
##
##  [1,]
##  [2,]
##  [3,]
##  [4,]
##  [5,]
##  [6,]
##  [7,]
##  [8,]
##  [9,]
## [10,]
```

```
# single element
y[0,1]
```

```
## numeric(0)
```

```
# range of rows or columns
y[0:2, 0:4]
```

```
##           [,1]      [,2]      [,3]      [,4]
## [1,] 0.7968570 0.9212767 0.6823411 0.4418082
## [2,] 0.1728649 0.2200920 0.2215528 0.6061150
```

```
# 5.
t(y)
```

```
##             [,1]      [,2]       [,3]      [,4]      [,5]       [,6]       [,7]
##  [1,] 0.79685701 0.1728649 0.87578904 0.6959279 0.7600220 0.23532950 0.77857844
##  [2,] 0.92127670 0.2200920 0.91410048 0.2217077 0.5799692 0.89248025 0.70558761
##  [3,] 0.68234114 0.2215528 0.87926976 0.4350800 0.8786261 0.74051907 0.76129400
##  [4,] 0.44180819 0.6061150 0.92364682 0.9502615 0.5233822 0.29902525 0.50676110
##  [5,] 0.09774047 0.9485971 0.32964212 0.4493628 0.1469271 0.68934373 0.61363313
##  [6,] 0.57022465 0.5548237 0.98581614 0.3519196 0.9349678 0.06831423 0.09488395
##  [7,] 0.79057356 0.2556447 0.09482889 0.9727837 0.9473751 0.19442846 0.22069528
##  [8,] 0.78001966 0.3250503 0.50257397 0.9632758 0.4546930 0.26004971 0.16289764
##  [9,] 0.17029558 0.4724135 0.31265122 0.8593613 0.5559640 0.23818317 0.72675383
## [10,] 0.01520497 0.1229970 0.85465740 0.7034005 0.6937164 0.63232937 0.50927153
##             [,8]      [,9]      [,10]
##  [1,] 0.81813397 0.4586118 0.82407474
##  [2,] 0.86122413 0.8053837 0.33005530
##  [3,] 0.50251328 0.8392831 0.04591512
##  [4,] 0.56156307 0.2405766 0.06305951
##  [5,] 0.11714718 0.8932037 0.76804530
##  [6,] 0.34385035 0.9930892 0.39536212
##  [7,] 0.40769378 0.3726629 0.38666612
##  [8,] 0.06807029 0.3033542 0.68106720
##  [9,] 0.23121343 0.2542687 0.39967714
## [10,] 0.34928806 0.9481925 0.95433873
```

```
# 6.
a <- matrix(runif(4), 2, 2)
b <- matrix(runif(4), 2, 2)
a*b # element wise
```

```
##           [,1]      [,2]
## [1,] 0.1719298 0.2987201
## [2,] 0.1151530 0.6396910
```

```
a %*% b # mathematical matrix multiplication
```

```
##           [,1]      [,2]
## [1,] 0.5444728 0.4937583
## [2,] 0.8017800 0.7320253
```

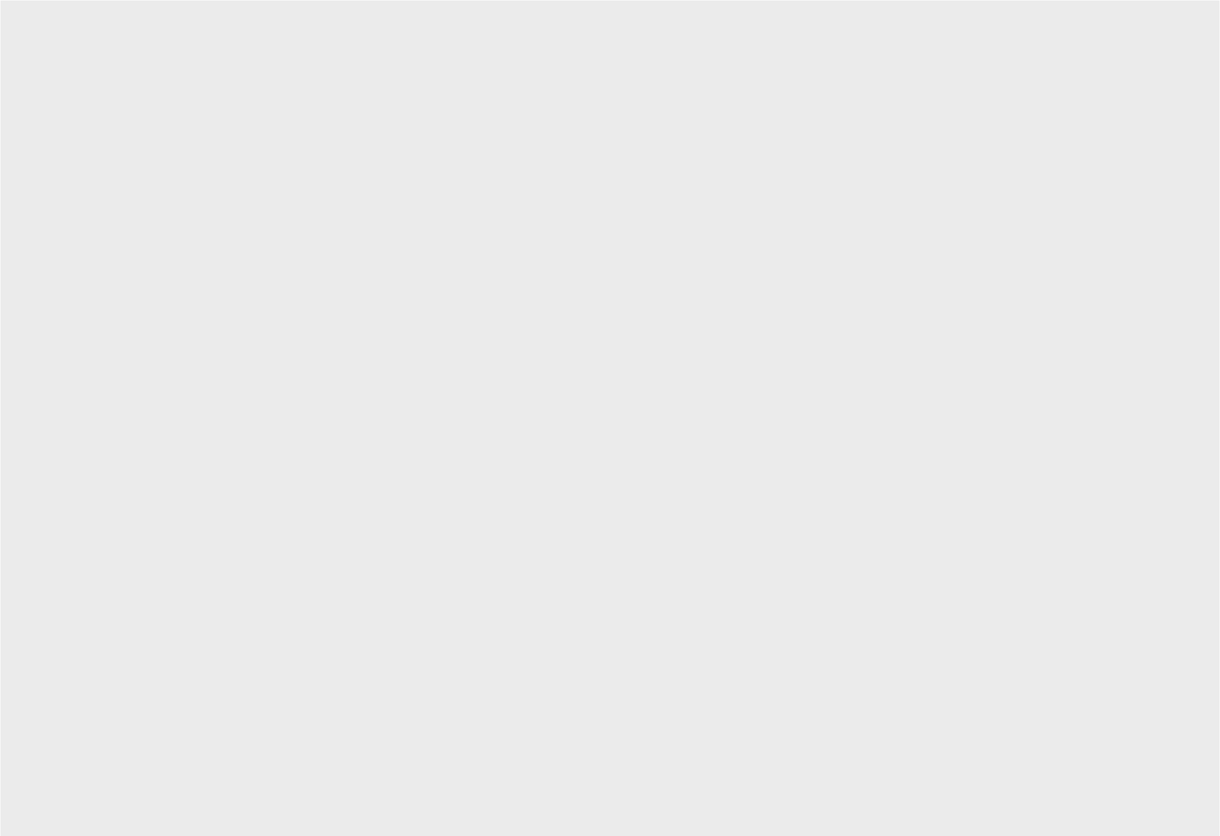
```
# 7.
rm(list=ls())

# Part 6: The Working Directory and R Projects
getwd()
```

```
## [1] "/home/anirudh/University/E401 Machine Learning For Economic Data/Week 3/problemset1a"
```

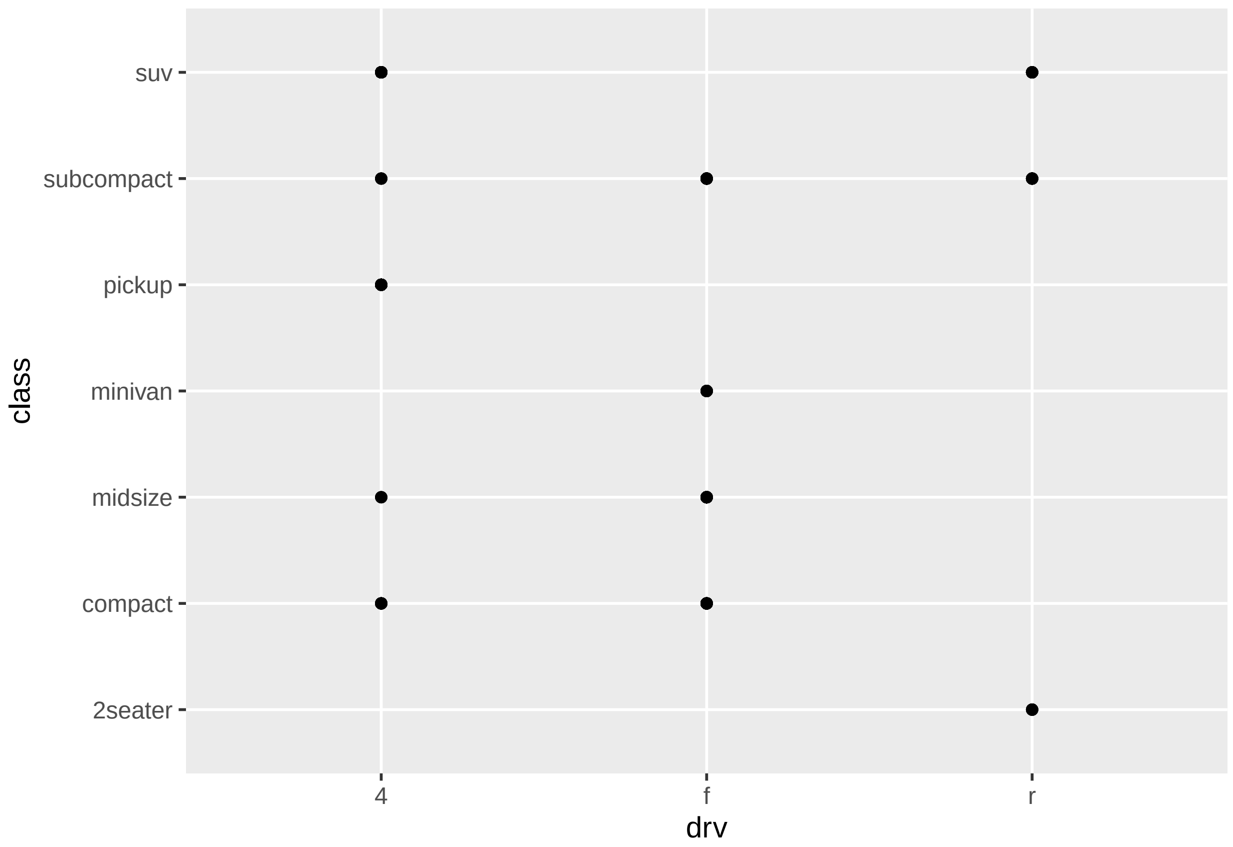
```
setwd("/home/anirudh/University/E401 Machine Learning For Economic Data/Week 3/problemset1a/r_firststeps
vgr = read_csv("vgr.csv")
```

```
## Rows: 140 Columns: 4
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (4): con, p, yv, quarter
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

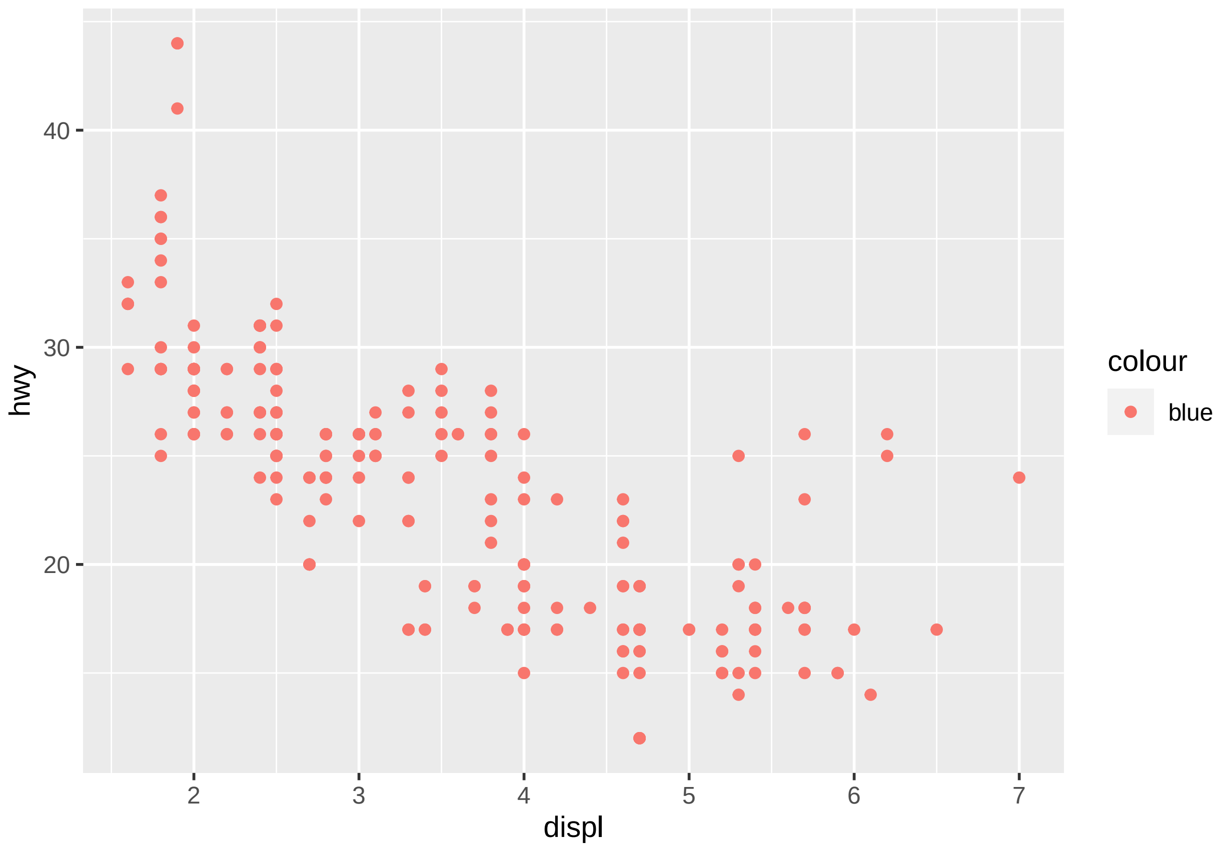
```
# Problem Set - 1a

# Question 2: Data Visualization
mpg <- mpg
diamonds <- diamonds

# Question 2.1: Visualization Basics
ggplot(data = mpg)
```



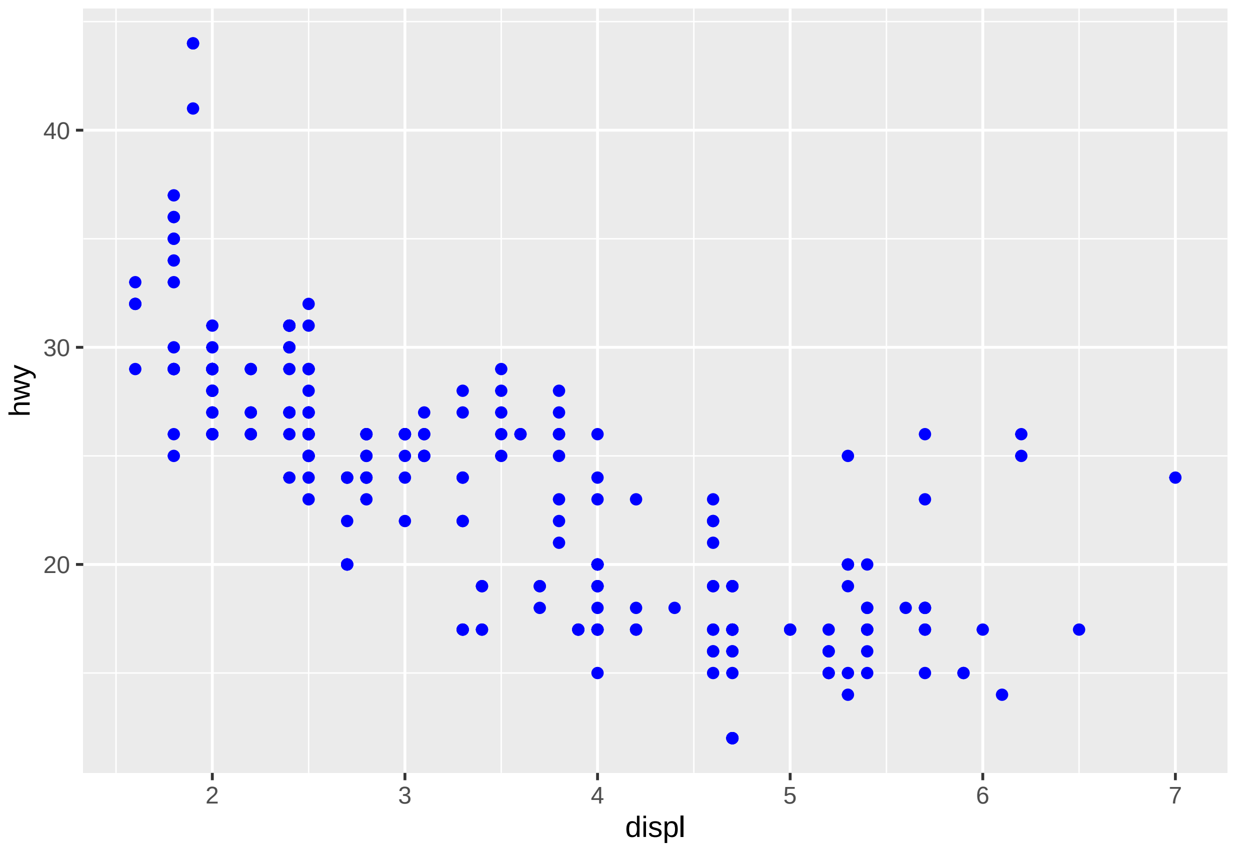
```r
# I see a gray box here with no plot being displayed because we aren't providing the structure
# information of the plot to the ggplot command
?mpg
# The drv variable is short for drive which describes the type of drive mode the car uses. For example,
# front-wheel, drive, rear wheel drive, or 4 = 4wd
ggplot(data = mpg) + geom_point(mapping = aes(x = drv, y = class))
```



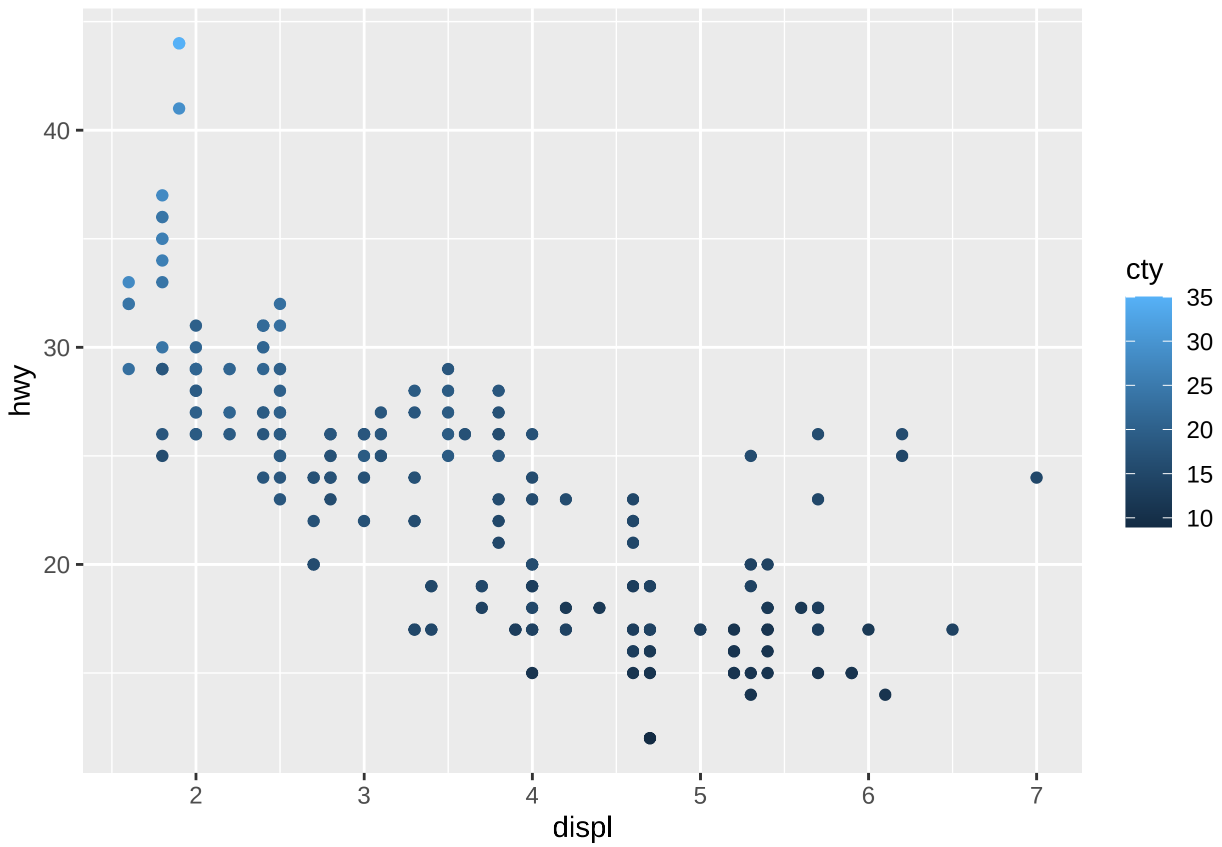
```
# We see a plot categorizing vehicles into specific categories like SUV, pickup truck based on their dr
# type. However, due to overlapping points, we don't see all 234 vehicle entries, making it challenging
# clear understanding of how the actual number of vehicles are categorized.

# Question 2.2: Aesthetic Mappings
# 1.
ggplot(data = mpg) + geom_point(mapping = aes(x = displ, y = hwy, color = "blue"))
```



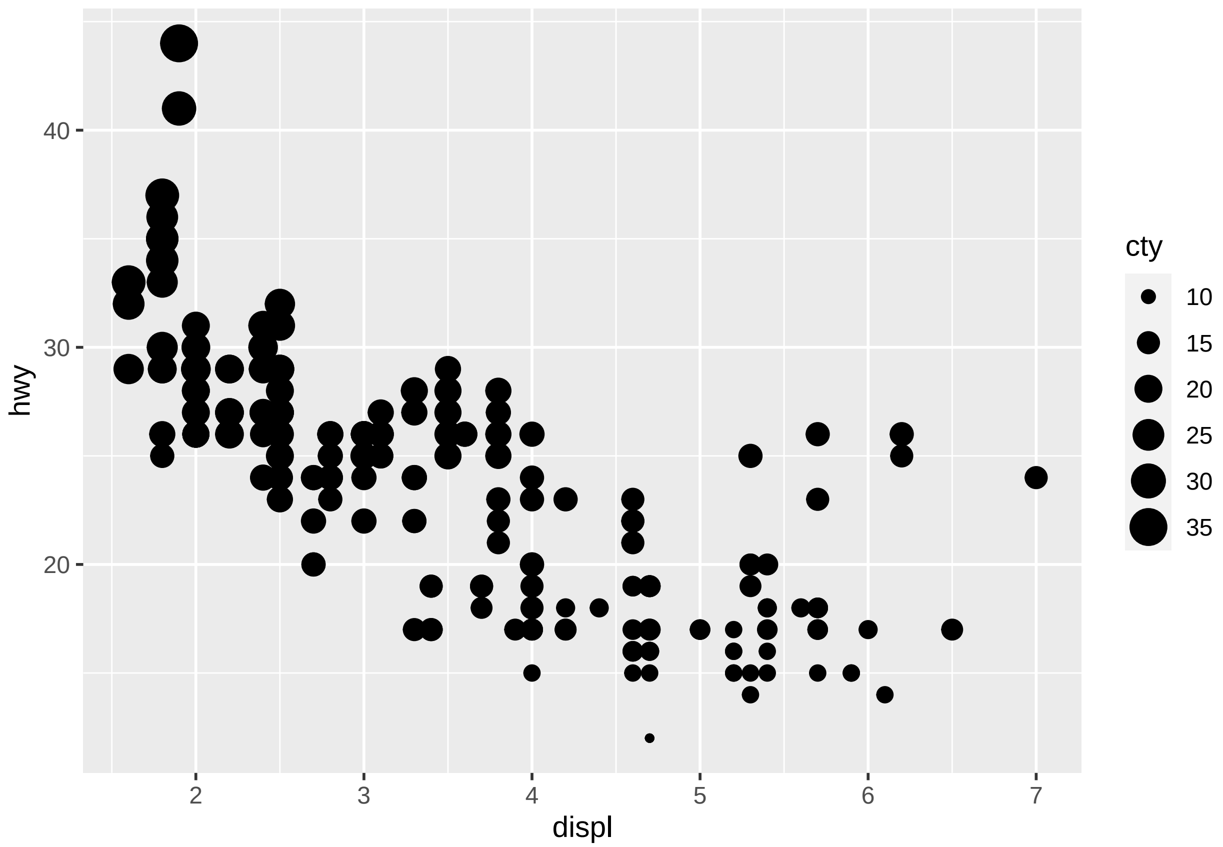
```
# The reason it doesn't show blue colour is because the "color" argument is not part of the command aes
# The below line of code shows how it should actually be typed for the colour blue to be used.
ggplot(data = mpg) + geom_point(mapping = aes(x = displ, y = hwy), color = "blue")
```



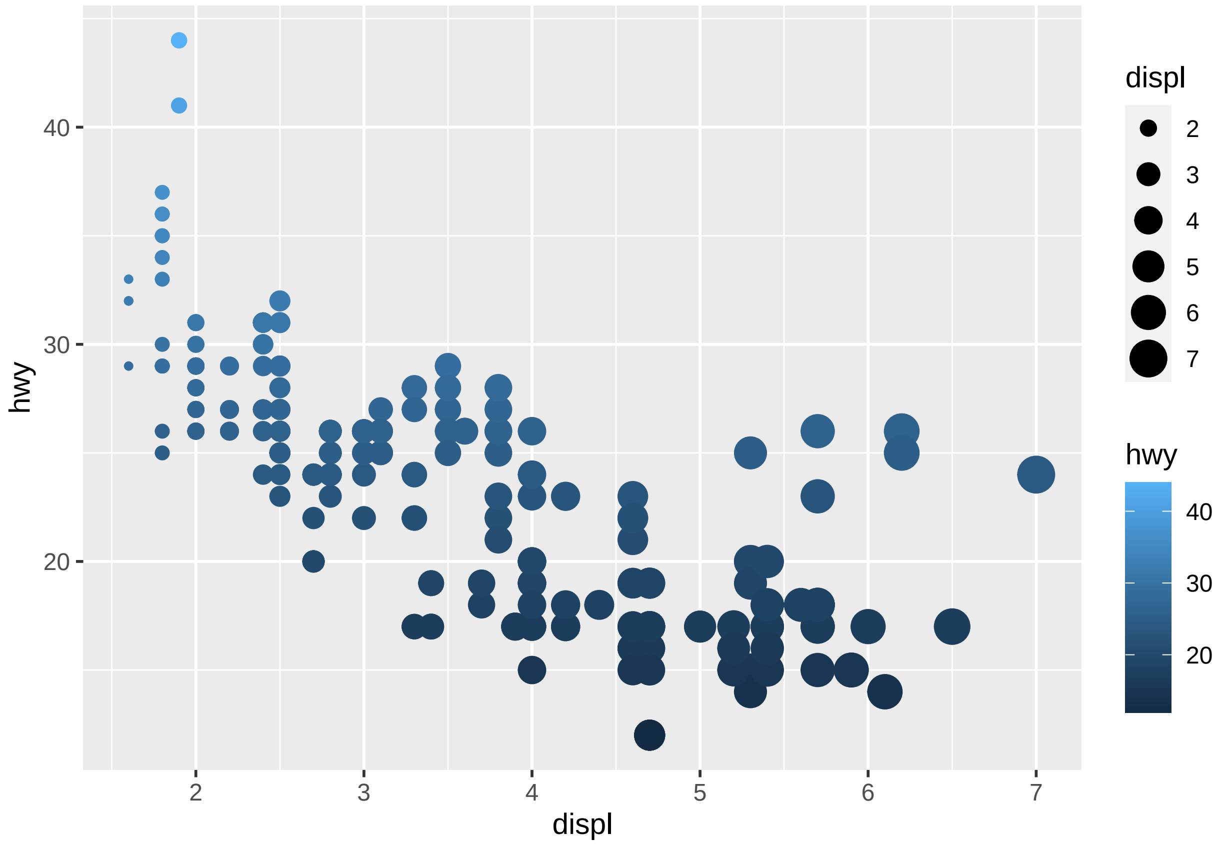
```
# 2.
?mpg
# categorical variables: manufacturer, model, drv, class, trans, fl
# continuous variables: displ, year, cyl, cty, hwy

# 3.
ggplot(mpg, aes(x = displ, y = hwy, colour = cty)) + geom_point()
```



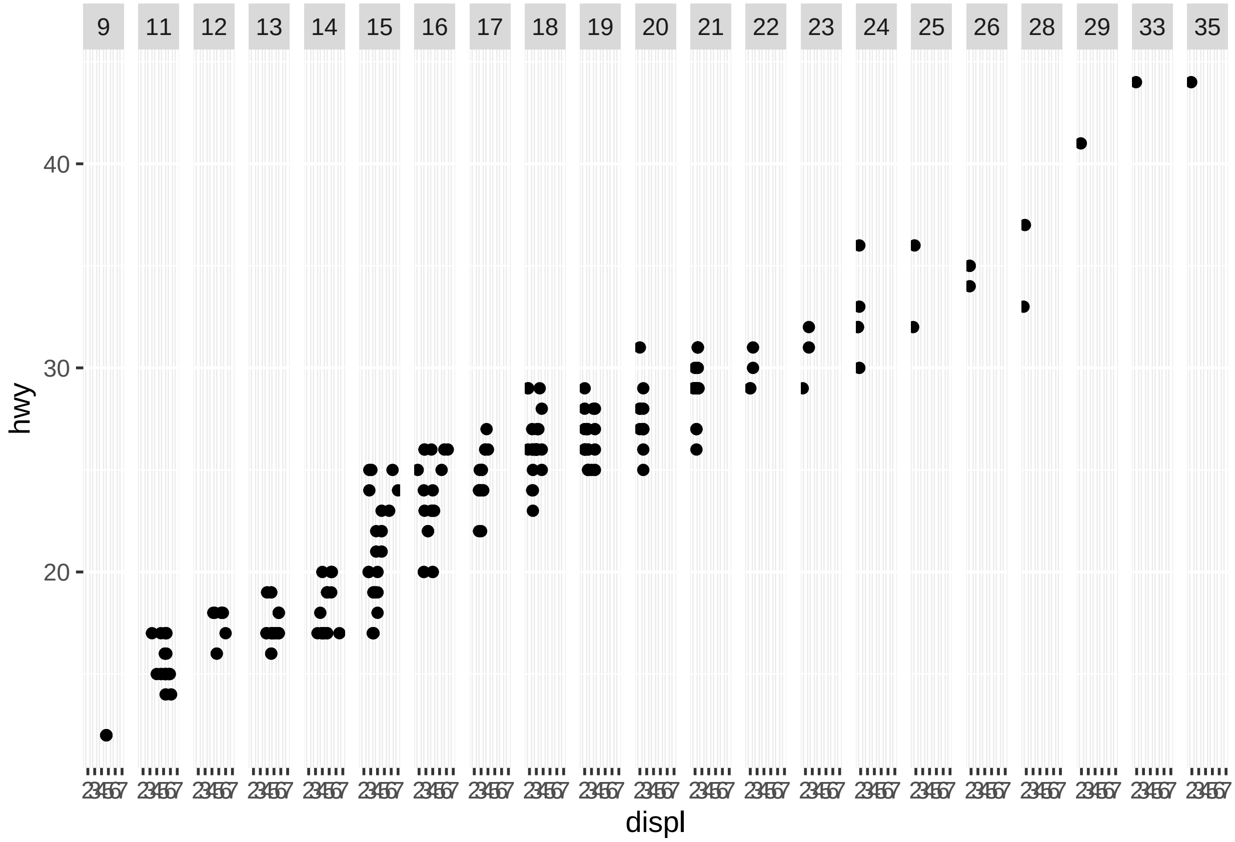
```
# cty here is a continuous variable that colours the points from light to dark. Light for points high a
# Dark for points low.
ggplot(mpg, aes(x = displ, y = hwy, size = cty)) + geom_point()
```



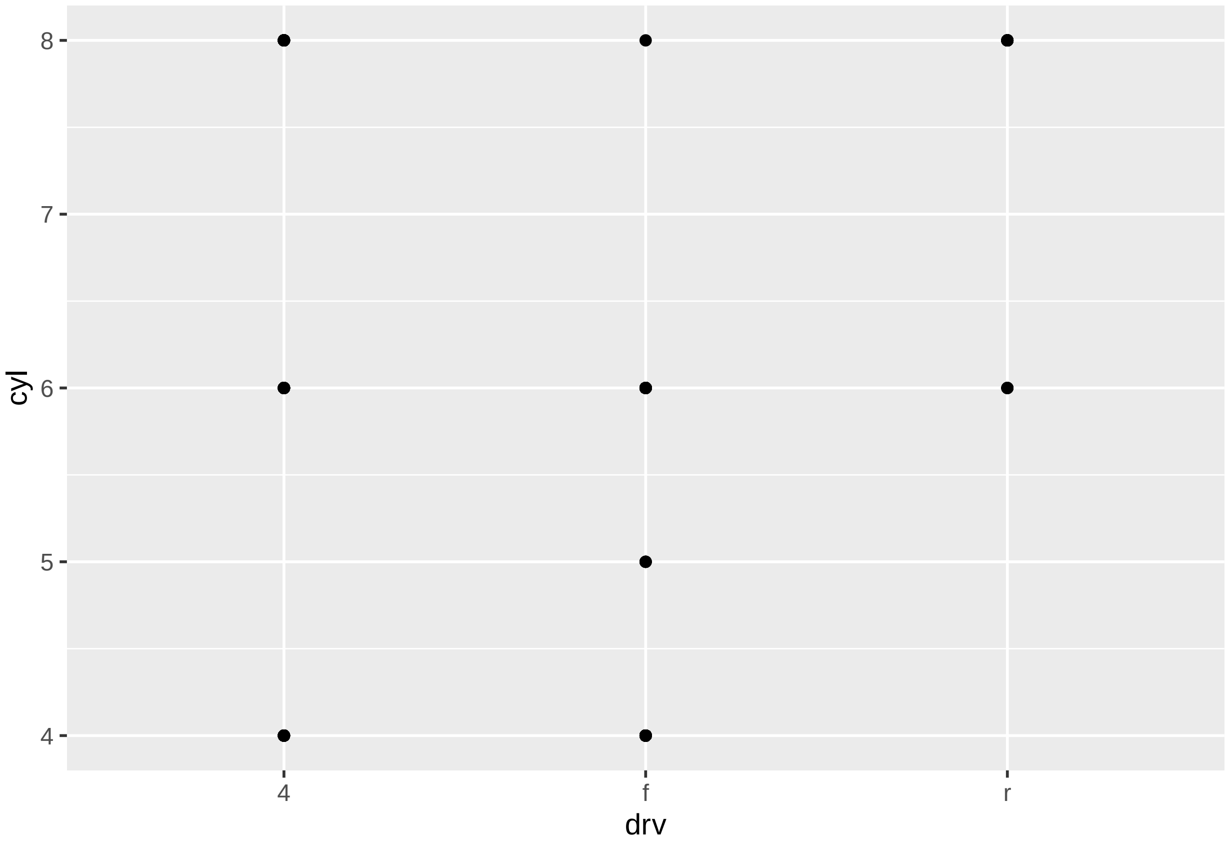
```
# when cty is used for size, the size values become continuous according to the corresponding y value's

# 4.
ggplot(mpg, aes(x = displ, y = hwy, colour = hwy, size = displ)) + geom_point()
```



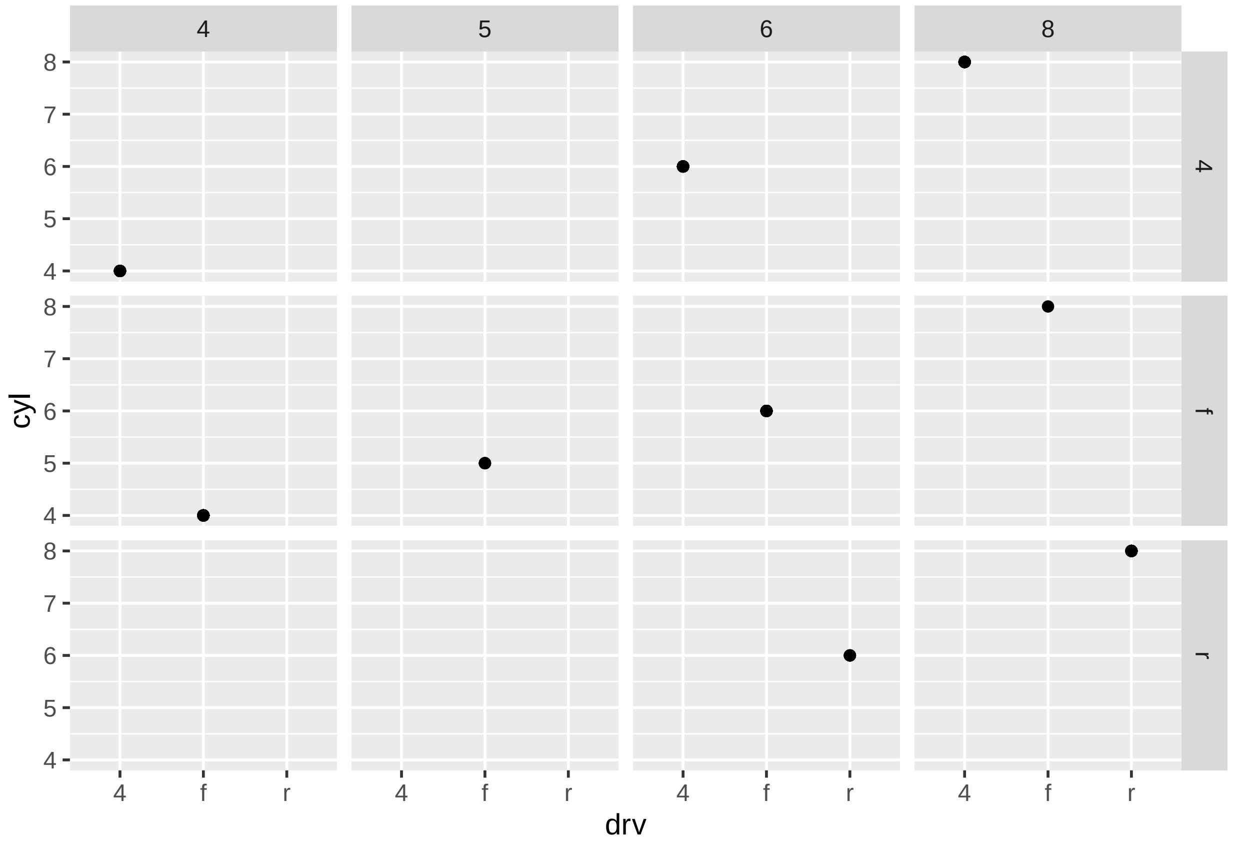
```r
# The issue with mapping a single varialbe to multiple aesthetics is that there is redundancy. So it's
# avoid this approach.

# Question 2.3: Facets

#1.
ggplot(mpg, aes(x = displ, y = hwy)) + geom_point() + facet_grid(. ~ cty)
```



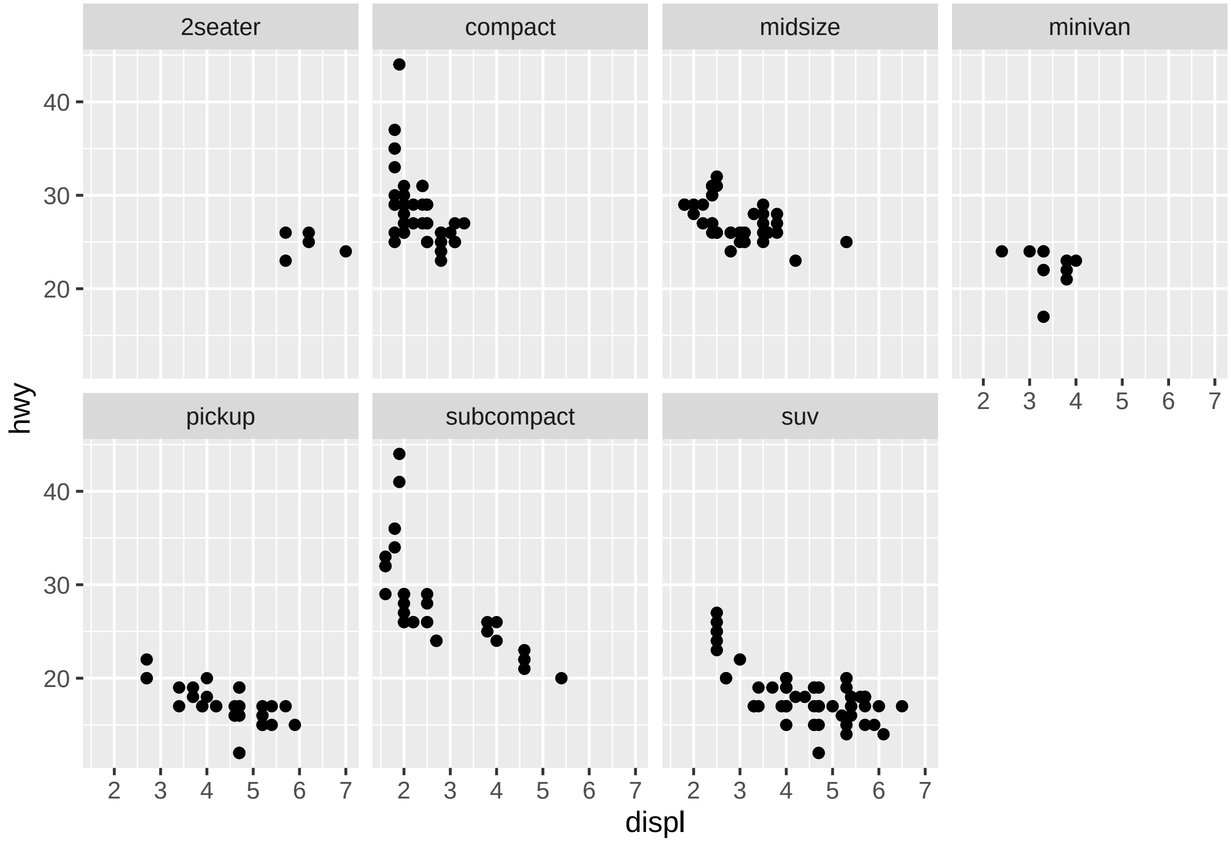
```
# If we facet a continuous variable, we can see that it gets converted into a categorical variable.

#2.
ggplot(data = mpg) + geom_point(mapping = aes(x = drv, y = cyl))
```



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

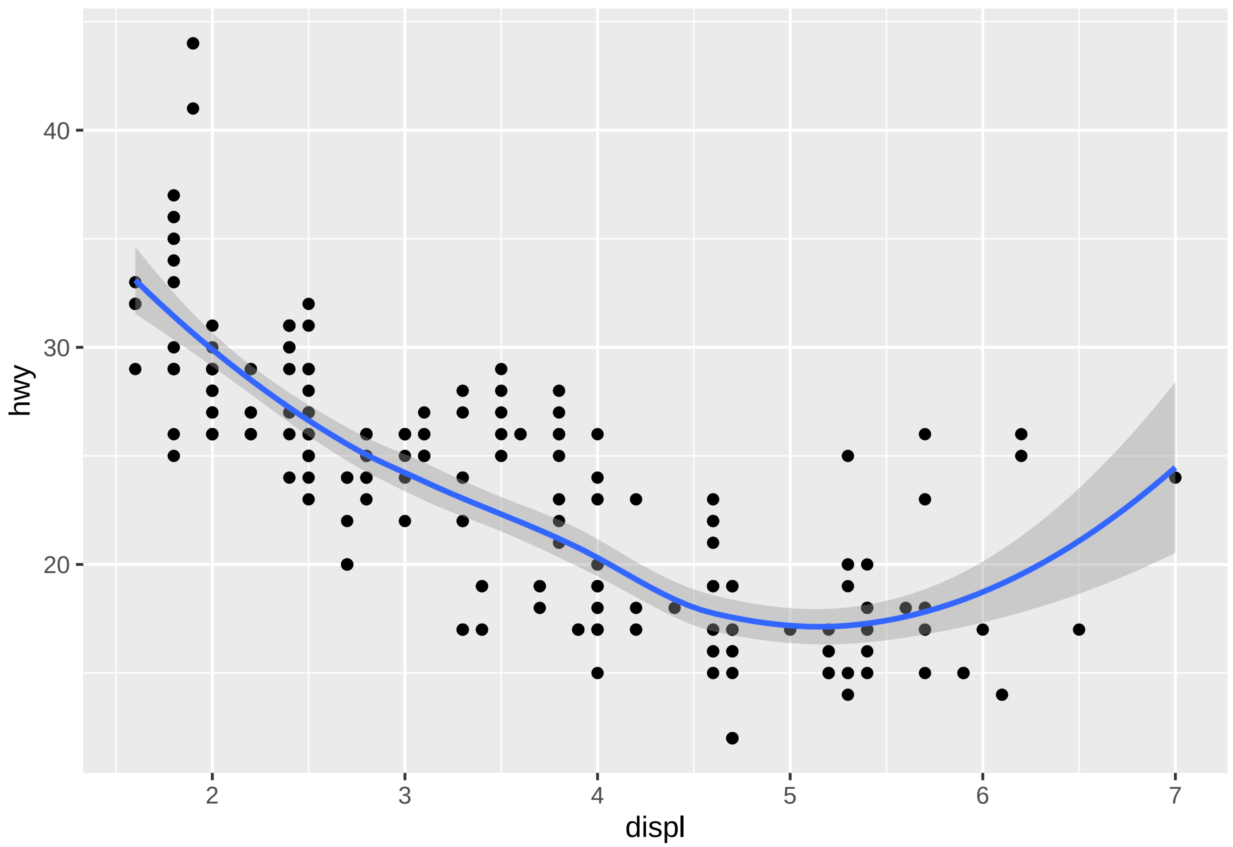


```
# The empty cells in this plot are data points in drv vs cyl that have no observations.
# These are the same locations in the scatter plot of drv and cyl that have no points.

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



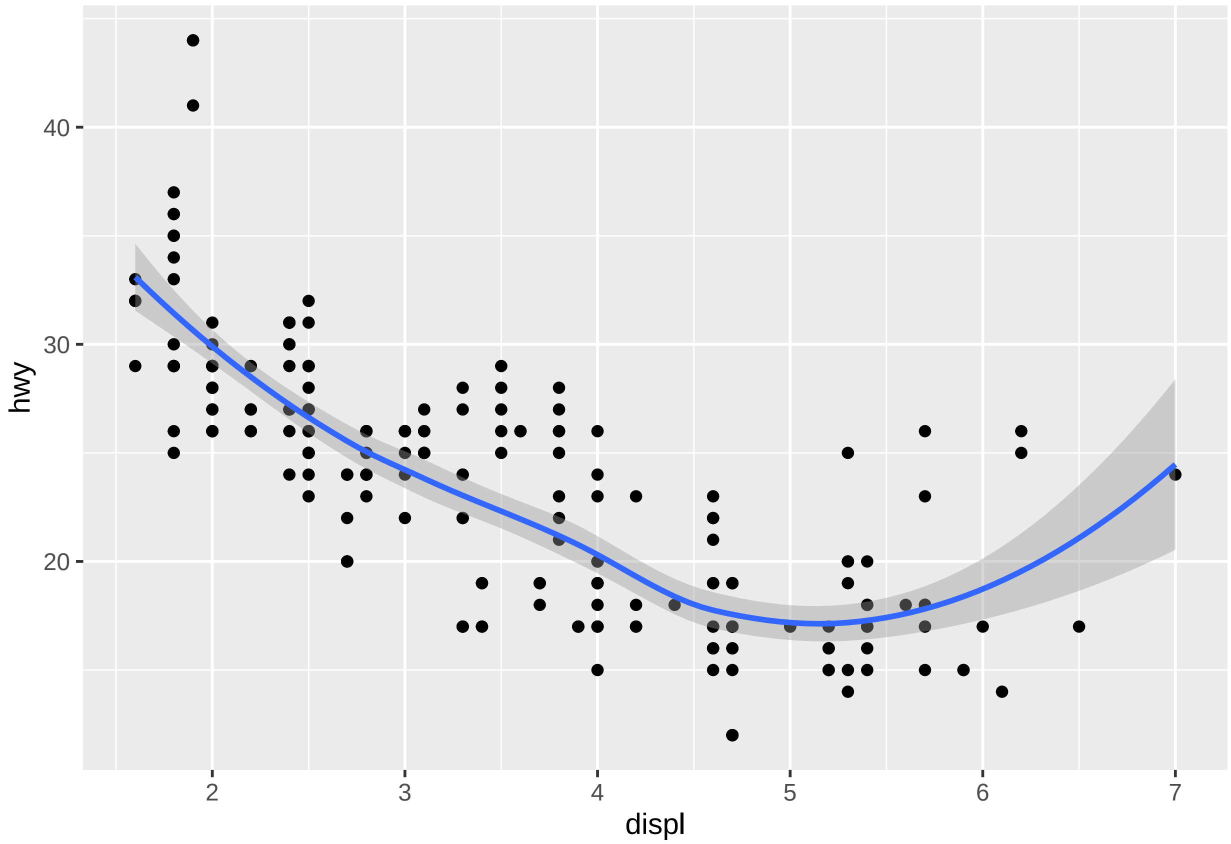
```
# The advantage would be we can encode them into distinct categorical variables
# The disadvantage would be that it makes it difficult to compare the datapoints between categories
# But as the no. of categories increase, differences in colours decrease making it difficult to tell th
# datapoints apart

# Question 2.4:
# 1.
# For line chart, geom_line()
# For boxplot, geom_boxplot()
# For histogram, geom_histogram()
# For area chart, geom_area()

#2.
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_point() +
  geom_smooth()
```

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



```r
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'
```

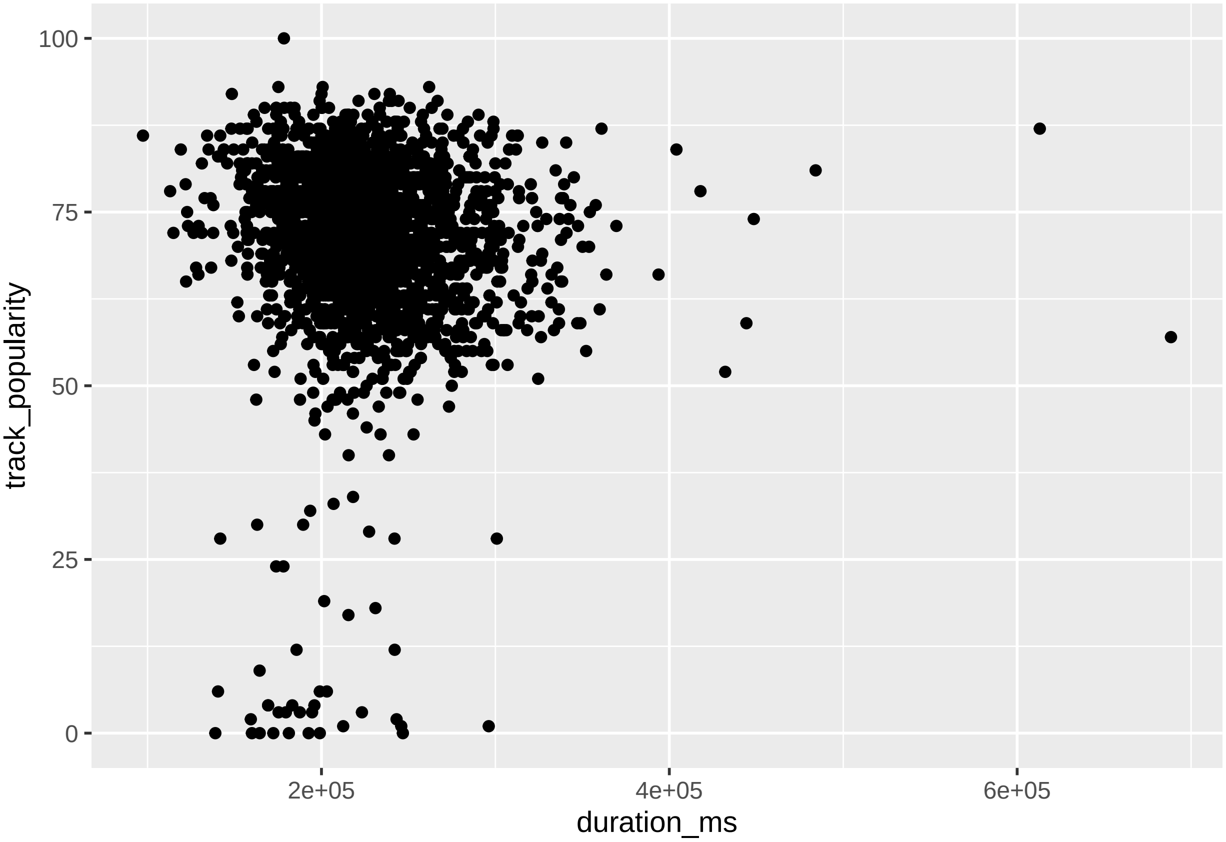


```
# They don't look different because geom point and smooth because the dataset is still the same

# Part 2: Your Project
# Question 3:
getwd()
```

```
## [1] "/home/anirudh/University/E401 Machine Learning For Economic Data/Week 3/problemset1a/r_firststep
```

```
setwd("/home/anirudh/University/E401 Machine Learning For Economic Data/Week 3/problemset1a/part_2_datas
spotify <- read.csv("playlist_2010to2022.csv")
ggplot(data = spotify) + geom_point(mapping = aes(x = duration_ms, y = track_popularity))
```

```
## Warning: Removed 1 rows containing missing values (`geom_point()`).
```

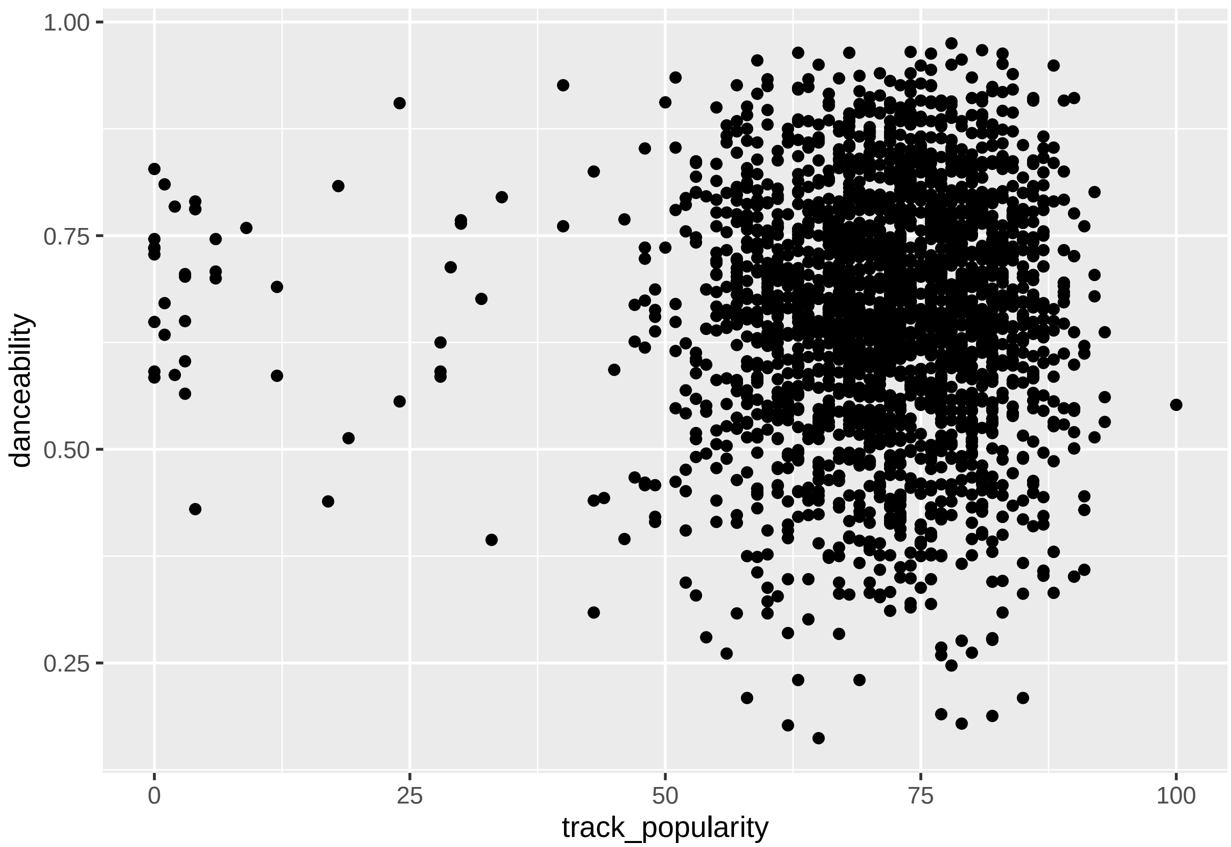


```
# The no.of milliseconds seems to play a role in the songs popularity. It looks like songs which have a
# around 200000ms/200s/3.33 minutes are popular.

musical_attributes <- c("danceability", "energy", "key", "loudness", "mode", "speechiness", "acousticne
musical_attributes
```

```
##  [1] "danceability"     "energy"           "key"              "loudness"
##  [5] "mode"             "speechiness"      "acousticness"     "instrumentalness"
##  [9] "liveness"         "valence"          "tempo"
```

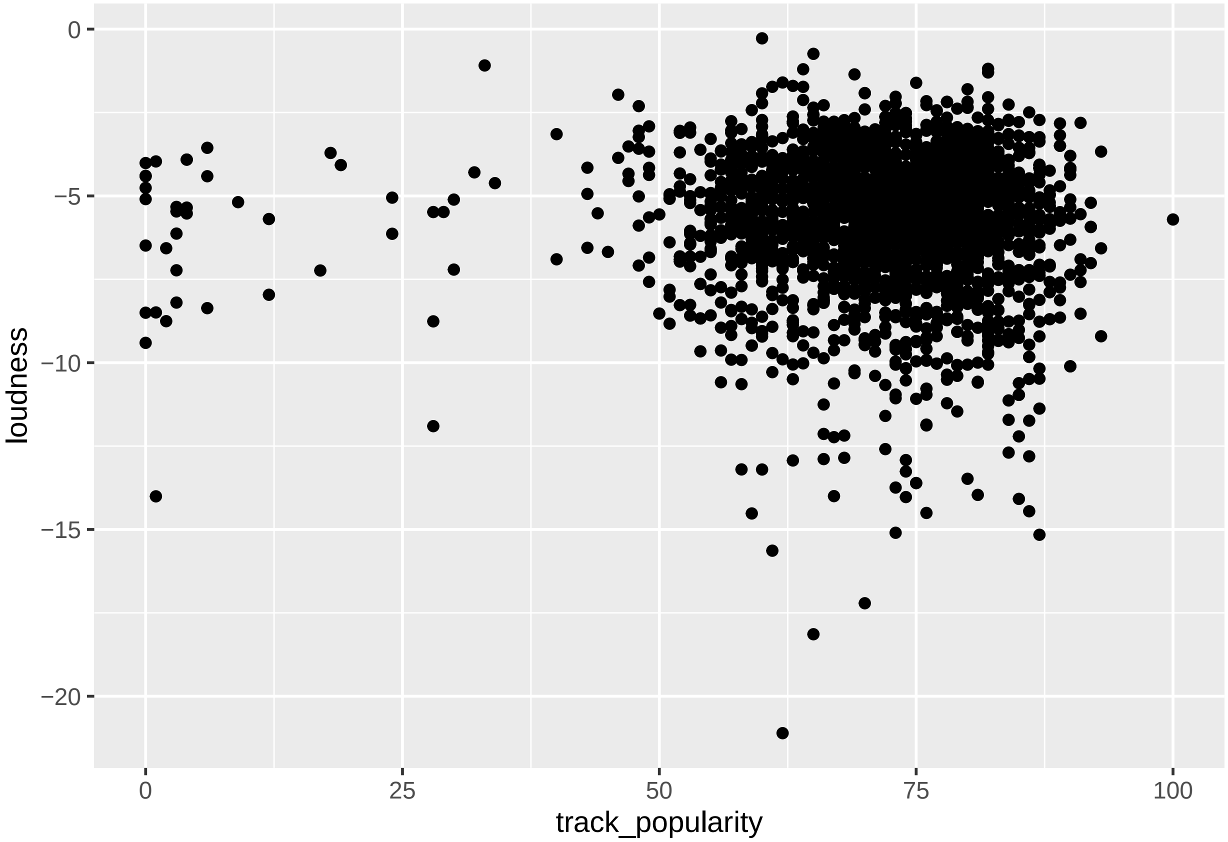
```
ggplot(data = spotify) + geom_point(mapping = aes(x = track_popularity, y = danceability))
```

```
## Warning: Removed 1 rows containing missing values (`geom_point()`).
```



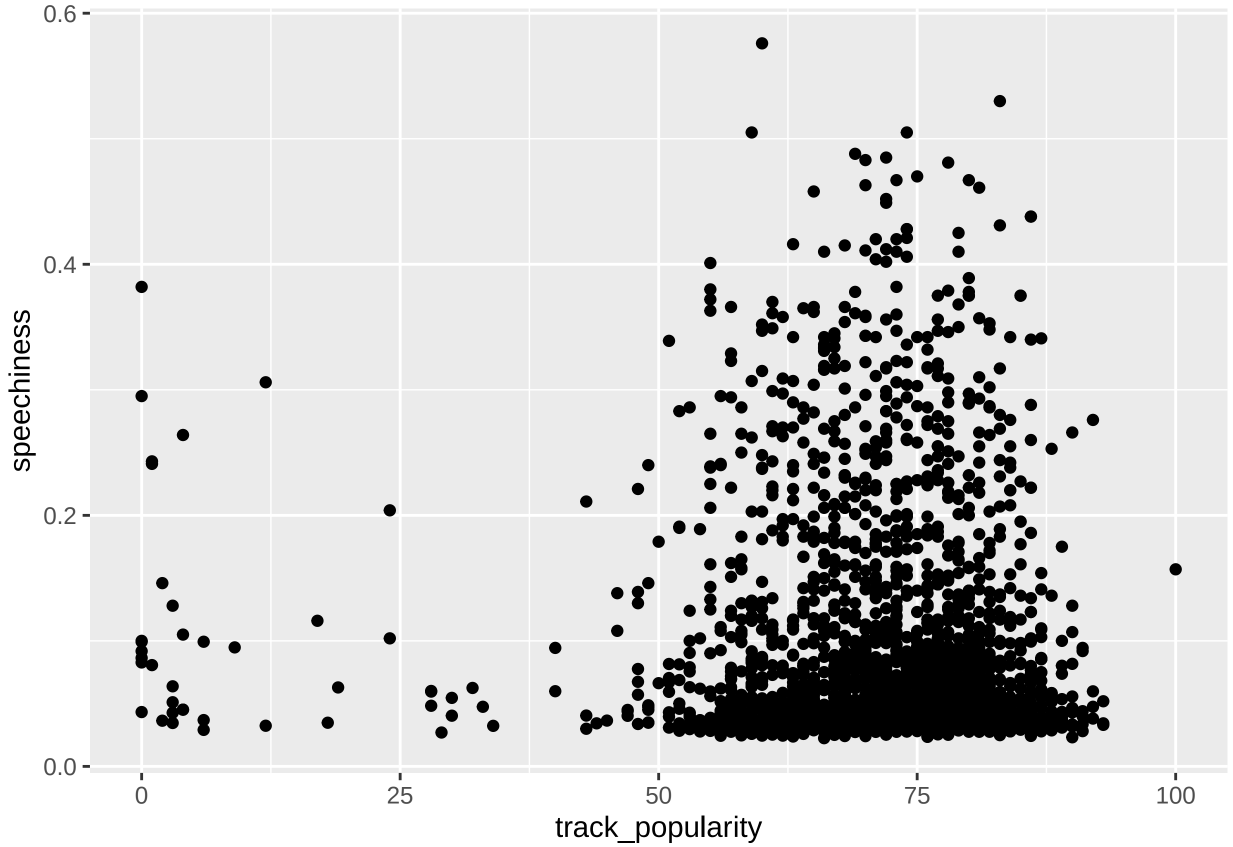
```
ggplot(data = spotify) + geom_point(mapping = aes(x = track_popularity, y = loudness))
```

```
## Warning: Removed 1 rows containing missing values (`geom_point()`).
```



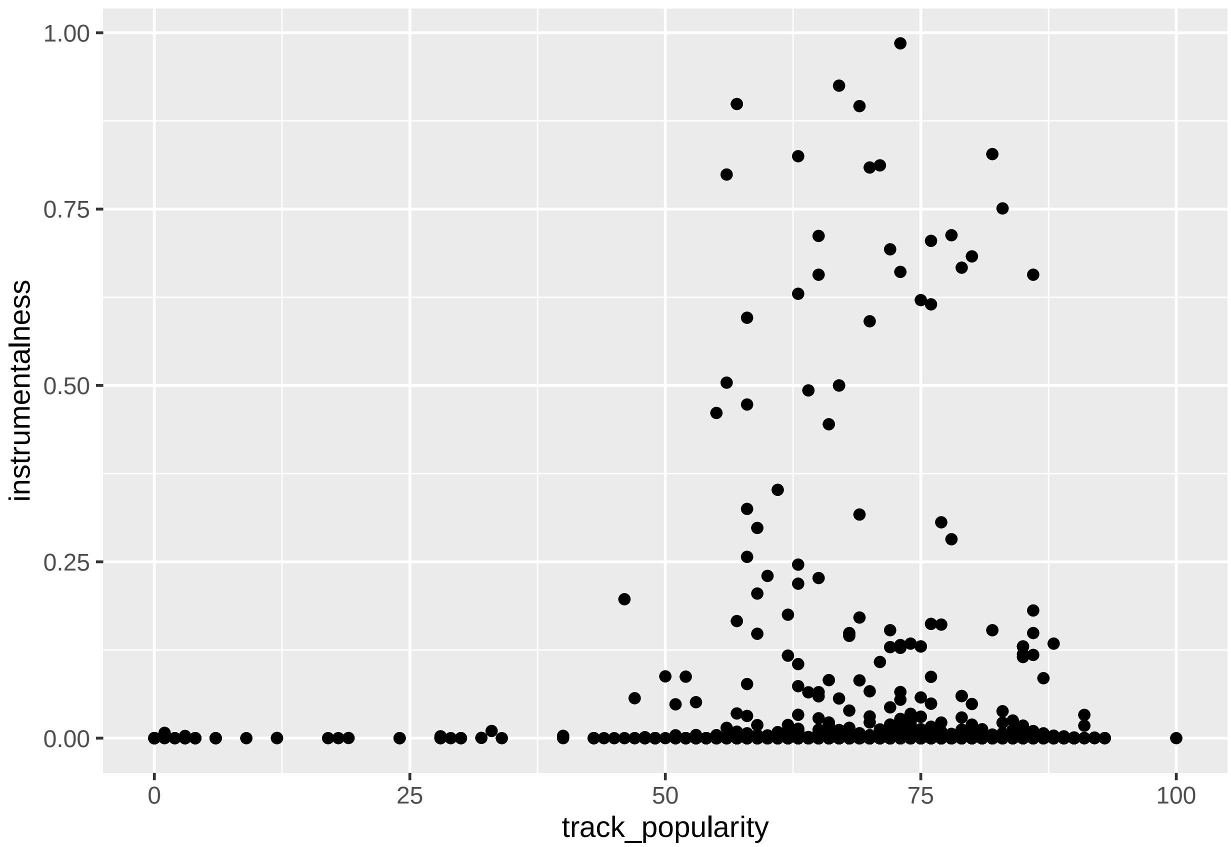
```
ggplot(data = spotify) + geom_point(mapping = aes(x = track_popularity, y = speechiness))
```

```
## Warning: Removed 1 rows containing missing values (`geom_point()`).
```



```
ggplot(data = spotify) + geom_point(mapping = aes(x = track_popularity, y = instrumentalness))
```

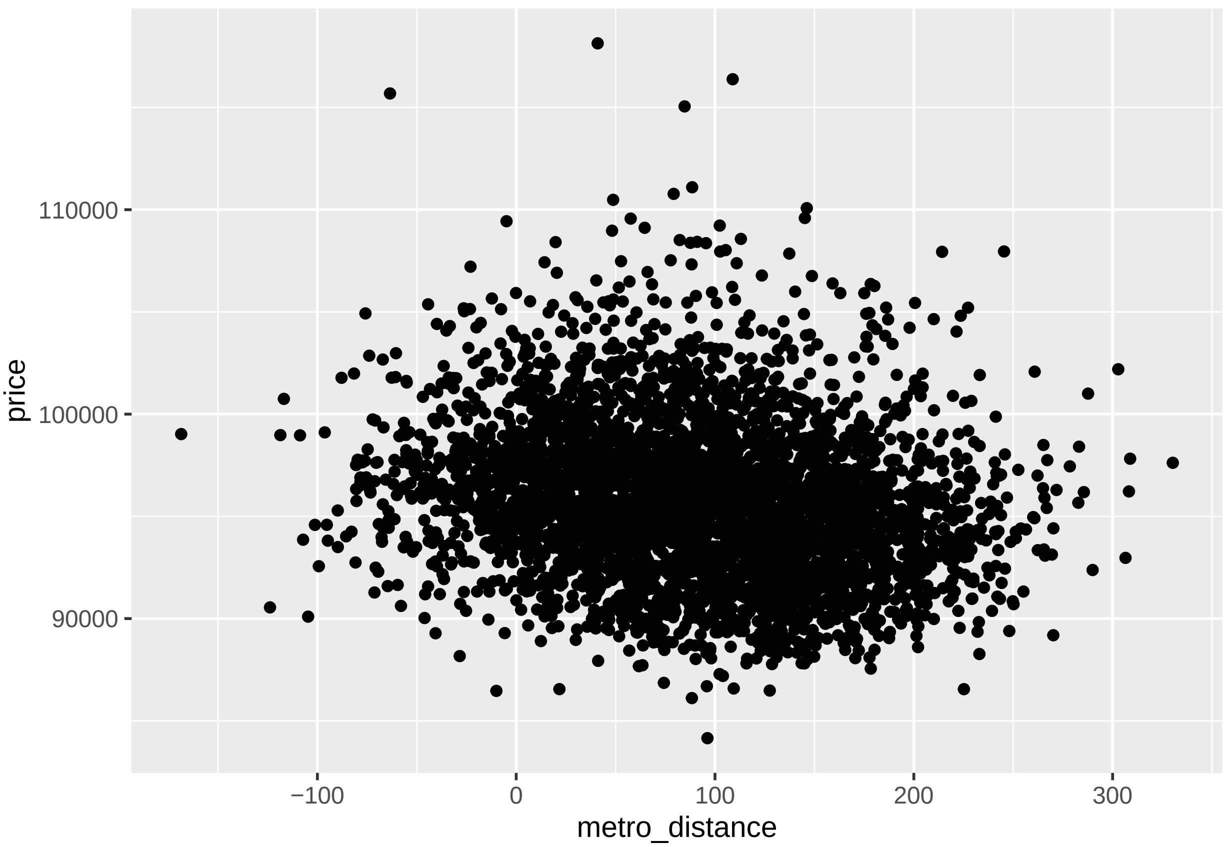
```
## Warning: Removed 1 rows containing missing values (`geom_point()`).
```



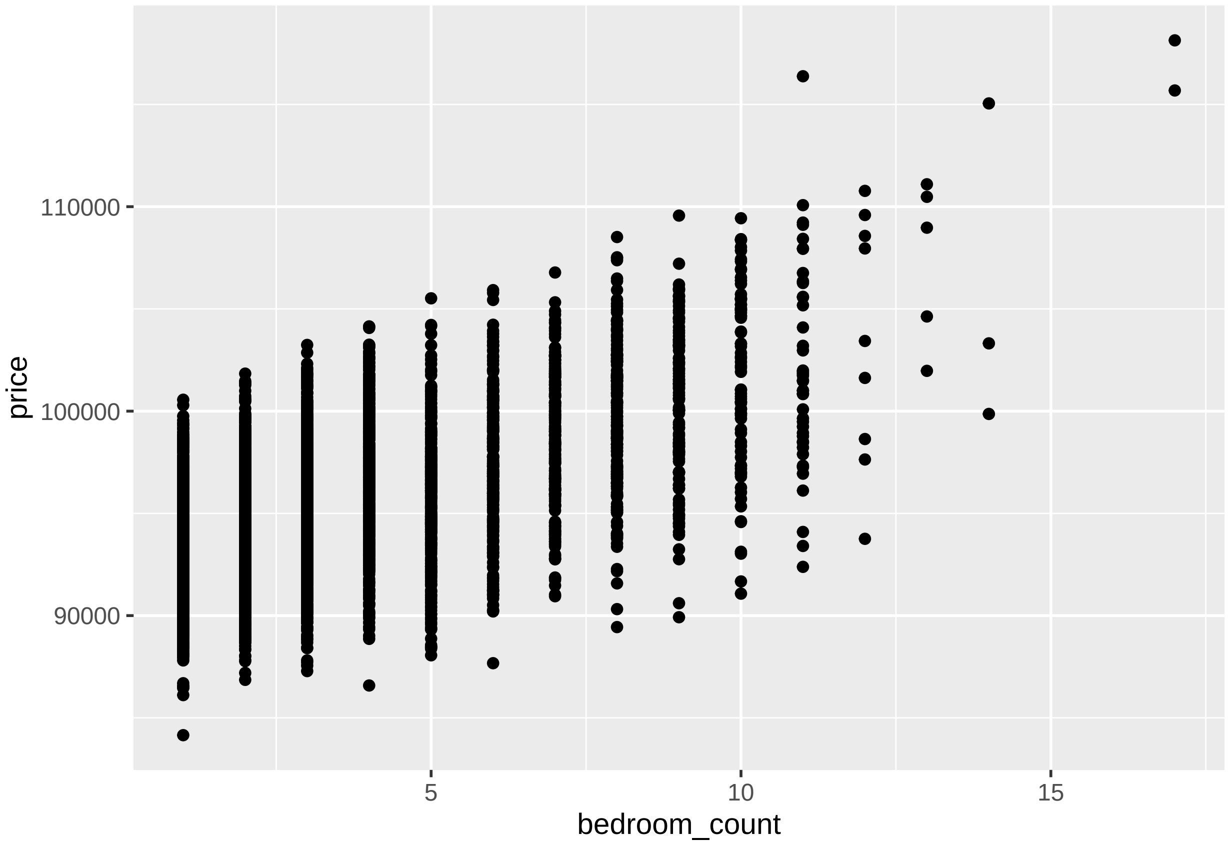
```
getwd()
```

```
## [1] "/home/anirudh/University/E401 Machine Learning For Economic Data/Week 3/problemset1a/part_2_data
```

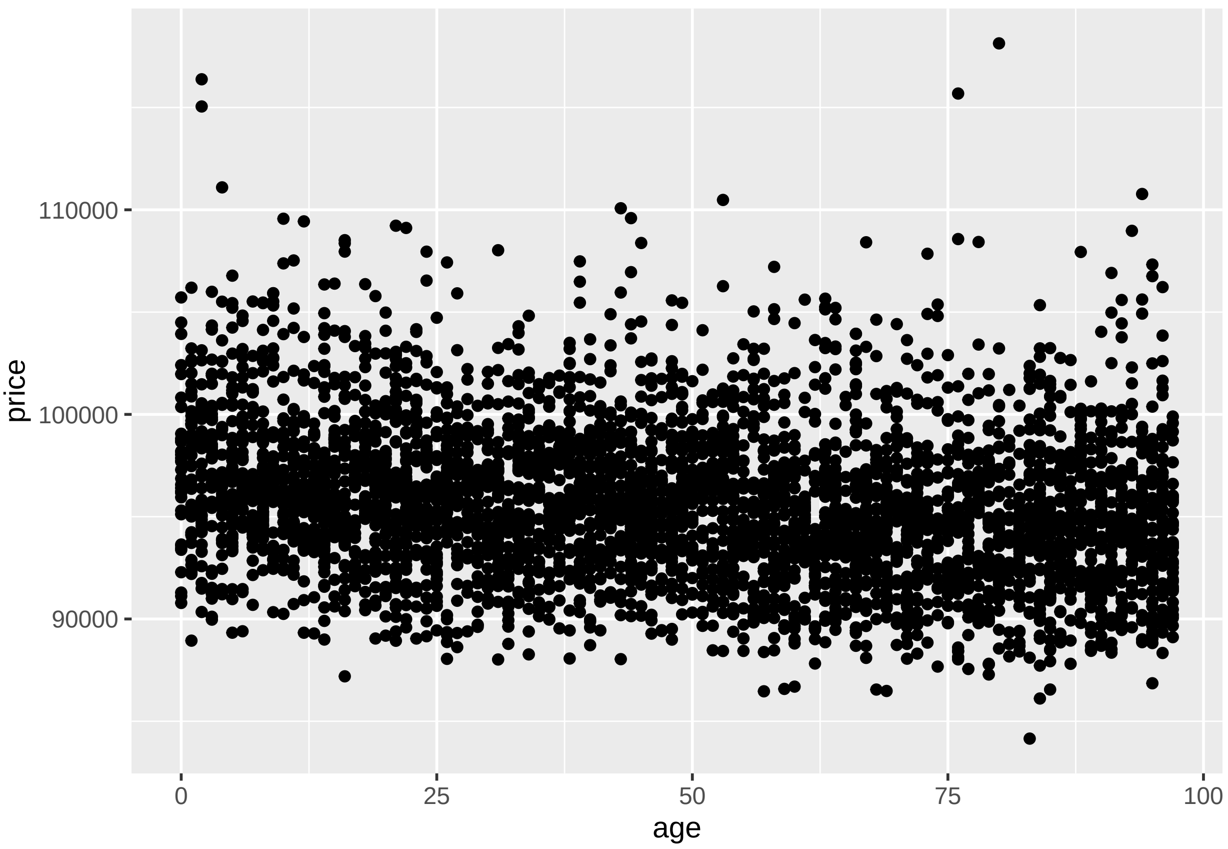
```
setwd("/home/anirudh/University/E401 Machine Learning For Economic Data/Week 3/problemset1a/part_2_data
house <- read.csv("house.csv")
ggplot(data = house) + geom_point(mapping = aes(x = metro_distance, y = price))
```



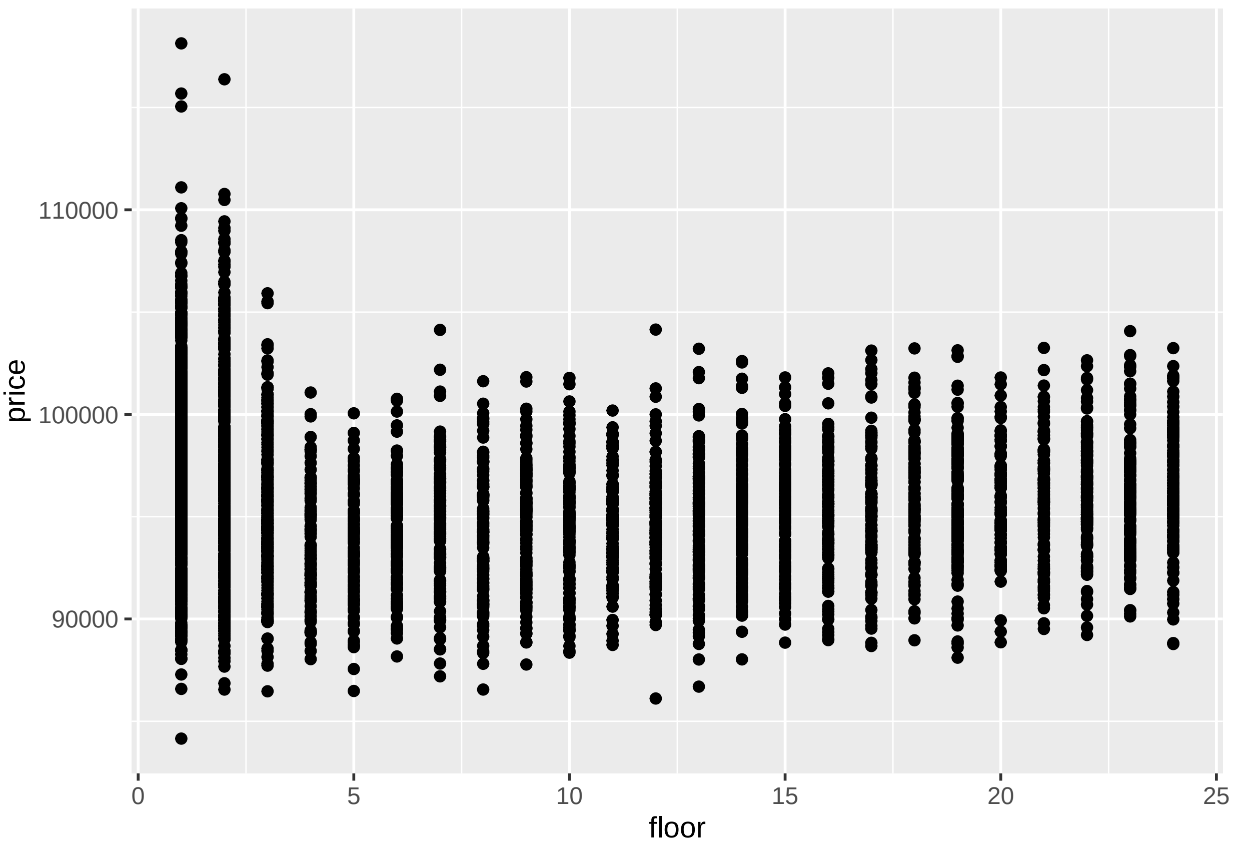
```
ggplot(data = house) + geom_point(mapping = aes(x = bedroom_count, y = price))
```



```
ggplot(data = house) + geom_point(mapping = aes(x = age, y = price))
```



```r
ggplot(data = house) + geom_point(mapping = aes(x = floor, y = price))
```


price
110000
100000
90000
0
5
10
15
20
25
floor