# ggplot2 : Introductory course

Aurélien Béliard & Gabriel Pires

11/12/2020

## Loading packages

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.4     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0
```

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

```
library(ggplot2)
library(plotly)
```

```
##
## Attaching package: ’plotly’
```

```
## The following object is masked from ’package:ggplot2’:
##
##     last_plot
```

```
## The following object is masked from ’package:stats’:
##
##     filter
```

```
## The following object is masked from ’package:graphics’:
##
##     layout
```

```
library(ggpubr)
library(extrafont)
```

```
## Registering fonts with R
```

# Visualizing dataset

```
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(l~ f        18    29 p     comp~
##  2 audi         a4         1.8  1999     4 manual~ f        21    29 p     comp~
##  3 audi         a4         2    2008     4 manual~ f        20    31 p     comp~
##  4 audi         a4         2    2008     4 auto(a~ f        21    30 p     comp~
##  5 audi         a4         2.8  1999     6 auto(l~ f        16    26 p     comp~
##  6 audi         a4         2.8  1999     6 manual~ f        18    26 p     comp~
##  7 audi         a4         3.1  2008     6 auto(a~ f        18    27 p     comp~
##  8 audi         a4 quat~   1.8  1999     4 manual~ 4        18    26 p     comp~
##  9 audi         a4 quat~   1.8  1999     4 auto(l~ 4        16    25 p     comp~
## 10 audi         a4 quat~   2    2008     4 manual~ 4        20    28 p     comp~
## # ... with 224 more rows
```

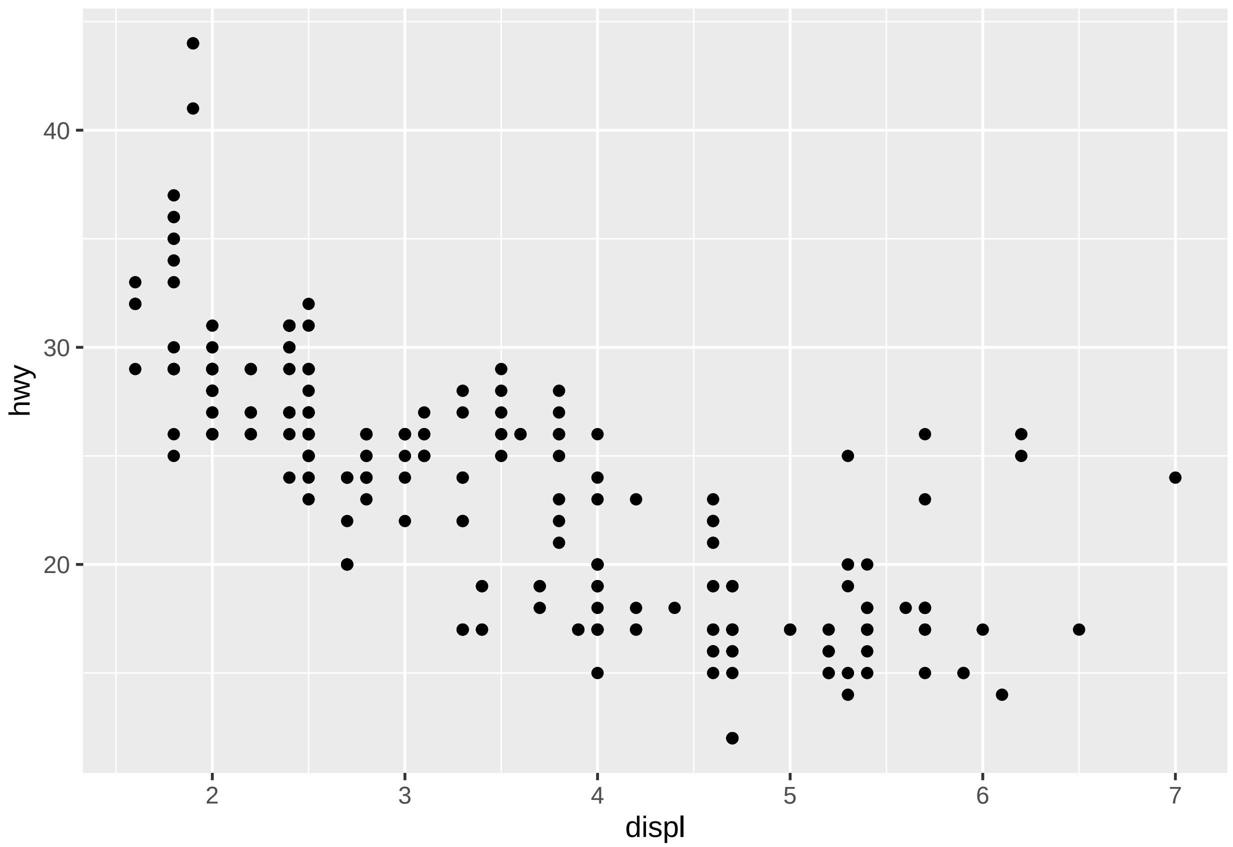
```
summary(mpg)
```

```
##  manufacturer          model               displ            year     
##  Length:234         Length:234         Min.   :1.600   Min.   :1999  
##  Class :character   Class :character   1st Qu.:2.400   1st Qu.:1999  
##  Mode  :character   Mode  :character   Median :3.300   Median :2004  
##                                        Mean   :3.472   Mean   :2004  
##                                        3rd Qu.:4.600   3rd Qu.:2008  
##                                        Max.   :7.000   Max.   :2008  
##       cyl           trans               drv                 cty       
##  Min.   :4.000   Length:234         Length:234         Min.   : 9.00  
##  1st Qu.:4.000   Class :character   Class :character   1st Qu.:14.00  
##  Median :6.000   Mode  :character   Mode  :character   Median :17.00  
##  Mean   :5.889                                         Mean   :16.86  
##  3rd Qu.:8.000                                         3rd Qu.:19.00  
##  Max.   :8.000                                         Max.   :35.00  
##       hwy             fl               class          
##  Min.   :12.00   Length:234         Length:234        
##  1st Qu.:18.00   Class :character   Class :character  
##  Median :24.00   Mode  :character   Mode  :character  
##  Mean   :23.44                                        
##  3rd Qu.:27.00                                        
##  Max.   :44.00                                        
```

# Dotplots

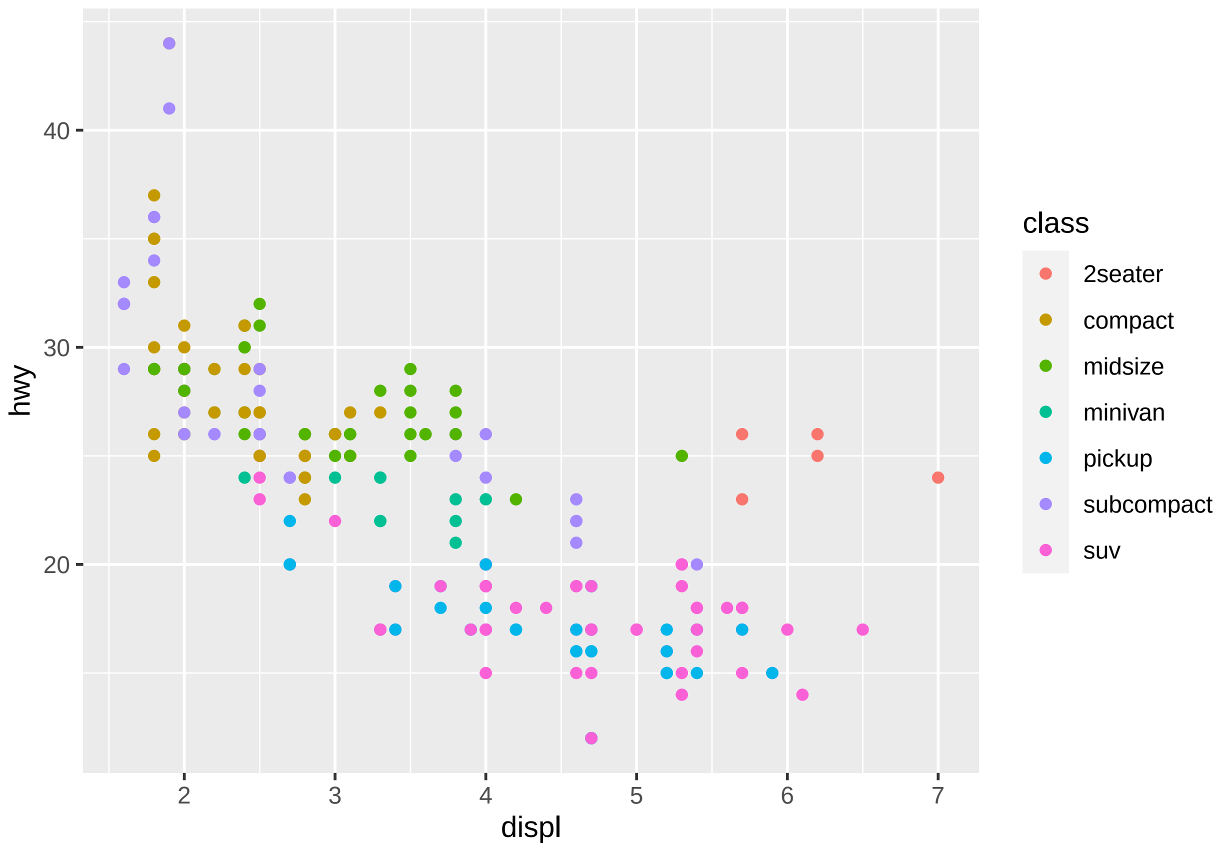
You can create a dot plot using the following function :

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



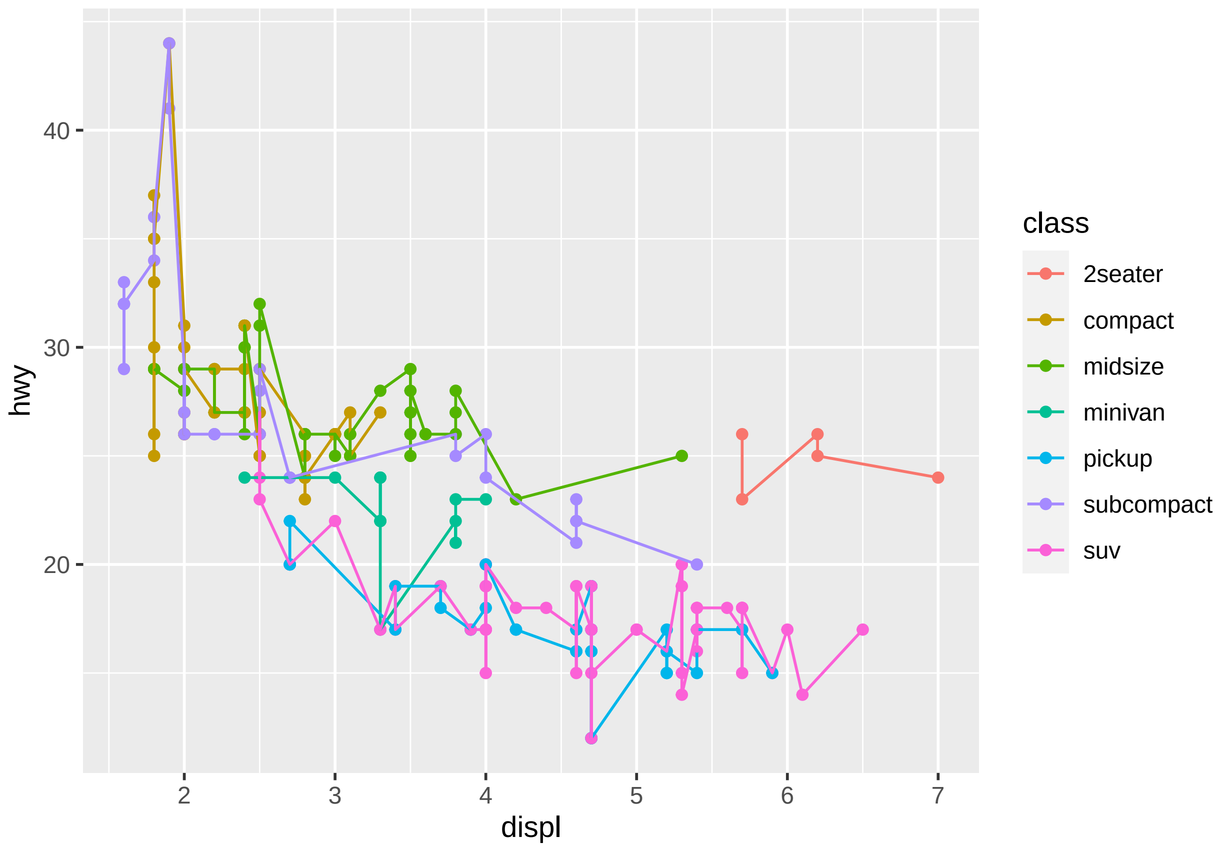
Pretty basic, now let's add some colors :

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



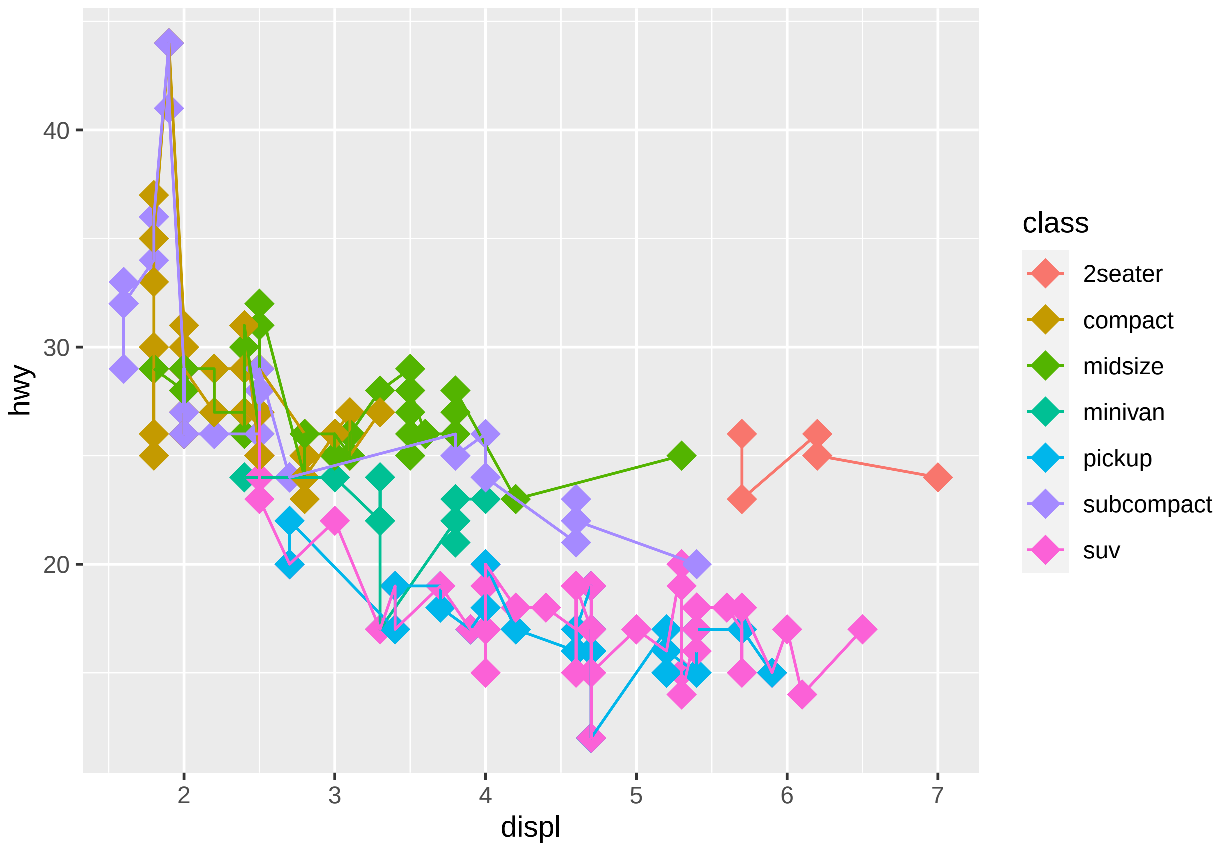
We can also add another layer as :

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



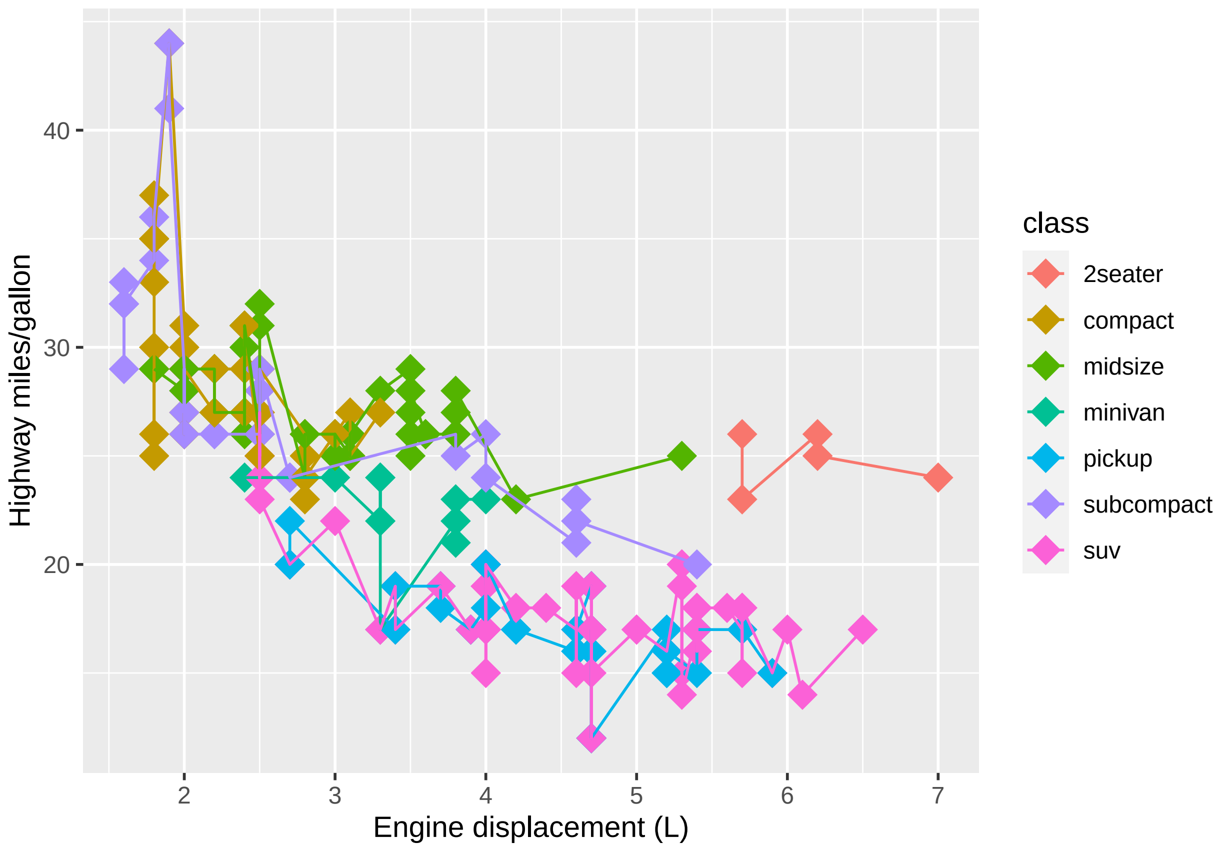
Let’s now play with the shape and size !

```
ggplot(mpg)+
  geom_point(aes(x=displ, y=hwy, color=class), shape="diamond", size=5)+
  geom_line(aes(x=displ, y=hwy, color=class))
```



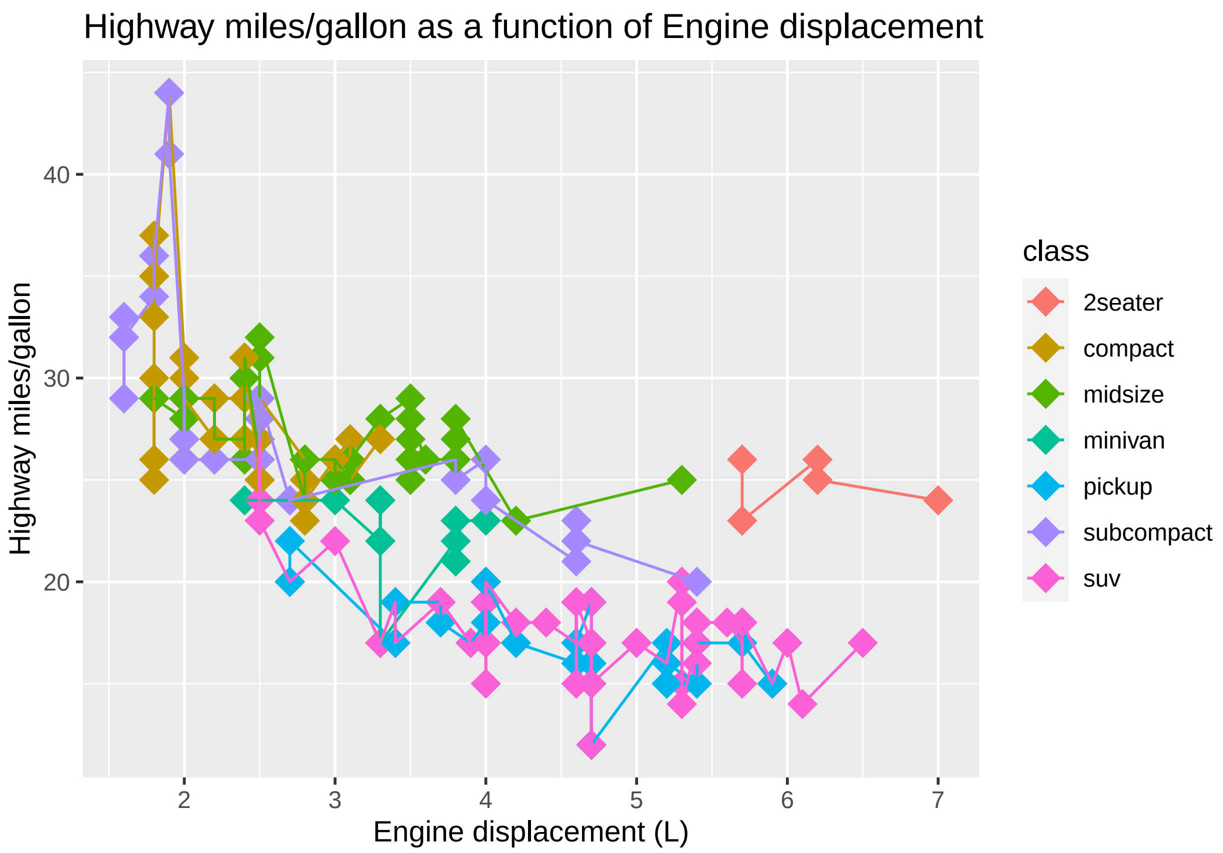
Now that we have the plot we wanted, let's arrange it to make it prettier and easier to read. First, let's change the names of the axes to make it clearier.

```
ggplot(mpg)+
  geom_point(aes(x=displ, y=hwy, color=class), shape="diamond", size=5)+
  geom_line(aes(x=displ, y=hwy, color=class))+
  xlab("Engine displacement (L)")+
  ylab("Highway miles/gallon")
```



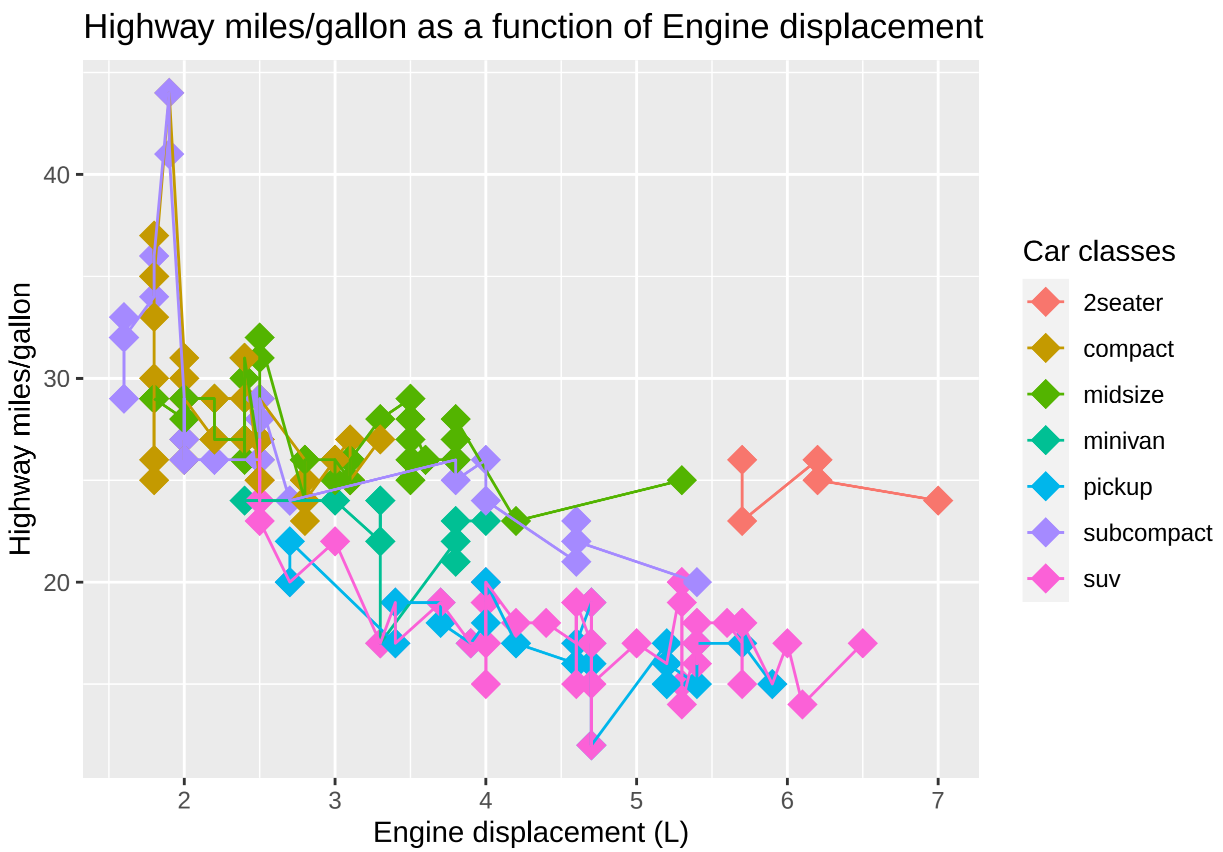
We could also add a title.

```
ggplot(mpg)+
  geom_point(aes(x=displ, y=hwy, color=class), shape="diamond", size=5)+
  geom_line(aes(x=displ, y=hwy, color=class))+
  xlab("Engine displacement (L)")+
  ylab("Highway miles/gallon")+
  ggtitle("Highway miles/gallon as a function of Engine displacement")
```



But that's a lot of functions to add to our ggplot object. We could also use the function *labs()*.

```
ggplot(mpg)+
  geom_point(aes(x=displ, y=hwy, color=class), shape="diamond", size=5)+
  geom_line(aes(x=displ, y=hwy, color=class))+
  labs(x="Engine displacement (L)",
  y="Highway miles/gallon",
  title="Highway miles/gallon as a function of Engine displacement",
  color="Car classes")
```

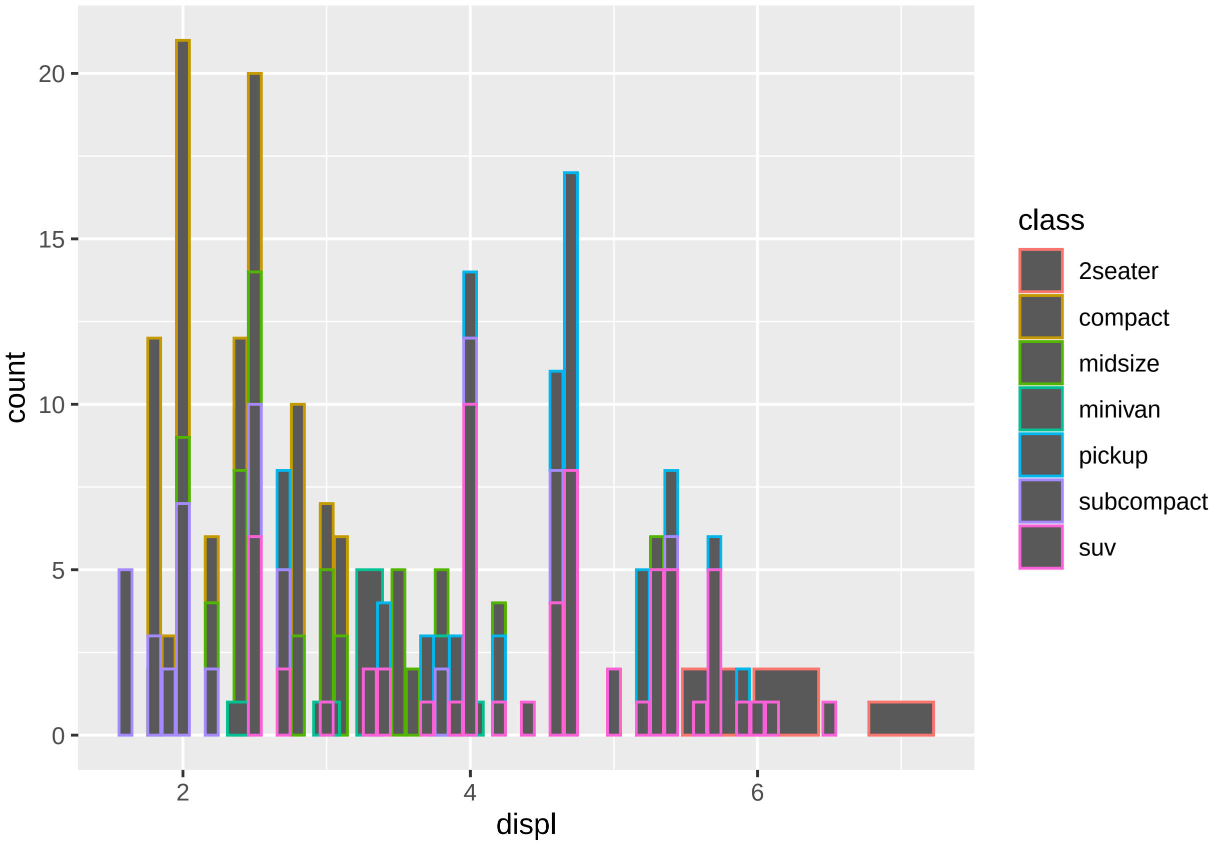


With this function we can also replace the name of the aesthetics, color, shape, size, fill, group, etc.

## Barplots

```
ggplot(mpg)+
  geom_bar(aes(x=displ, color=class))
```

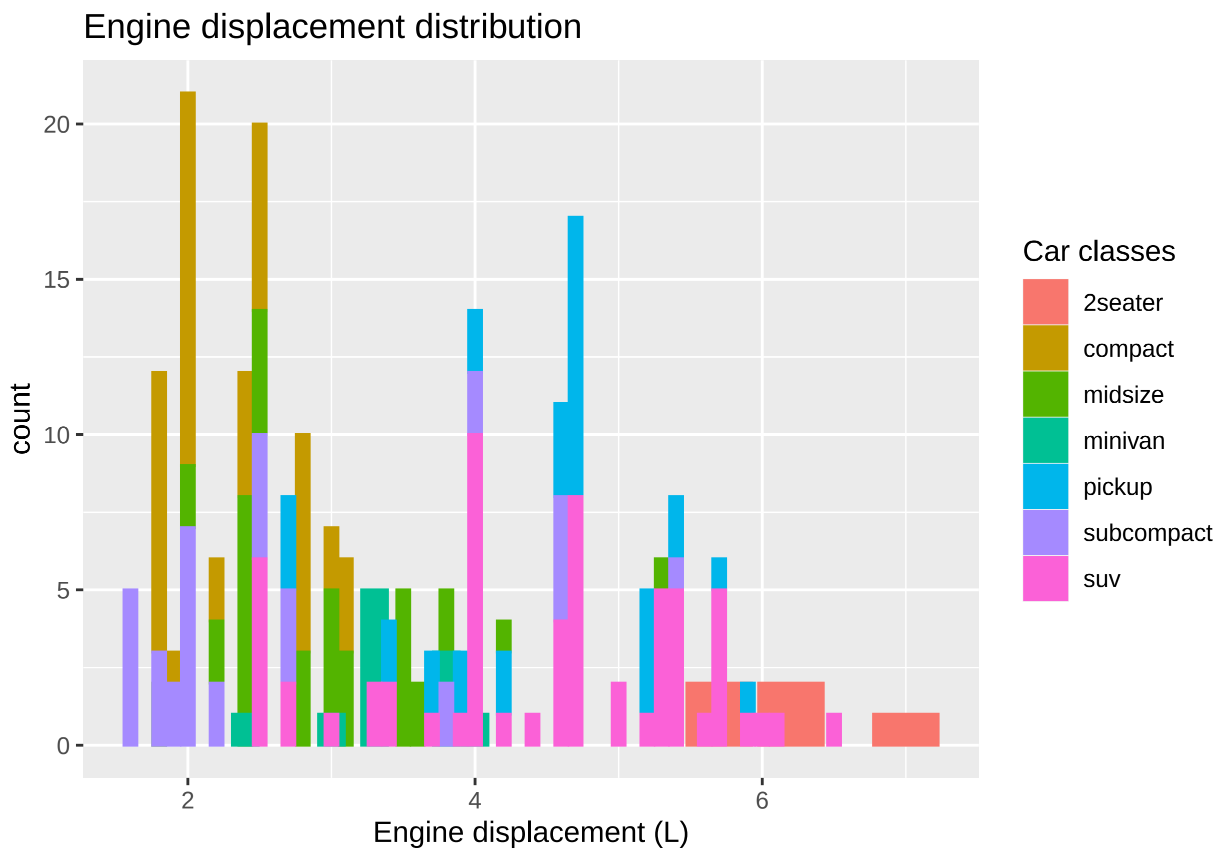
```
## Warning: position_stack requires non-overlapping x intervals
```



As you can see, by filling only the color aesthetics, we only color the contour of the bars. To fill the bars with a color, we can use the fill aesthetic as follow :

```
ggplot(mpg)+
  geom_bar(aes(x=displ, color=class, fill=class))+
  labs(x="Engine displacement (L)",
       color="Car classes",
       fill="Car classes",
       title="Engine displacement distribution")
```

```
## Warning: position_stack requires non-overlapping x intervals
```

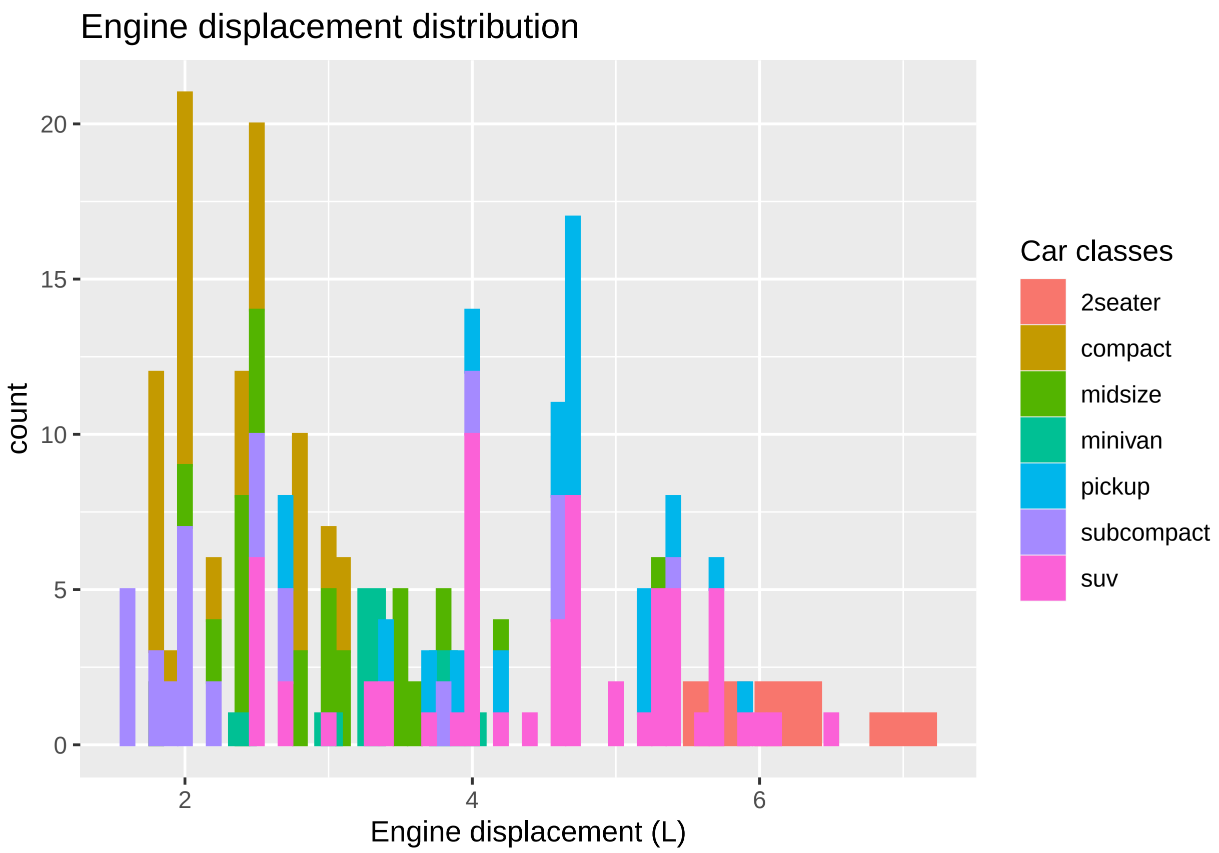


NB : This occurs because displ is numeric, so the ggplot object assumes a continuous x-axis, and aesthetic parameter group is based on the only known discrete variable (fill = class).

The $x$ axis can be treated either as a continuous variable or a discrete variable. To treat the $x$ axis as a continuous variable, use the *as.numeric()* function as follow :

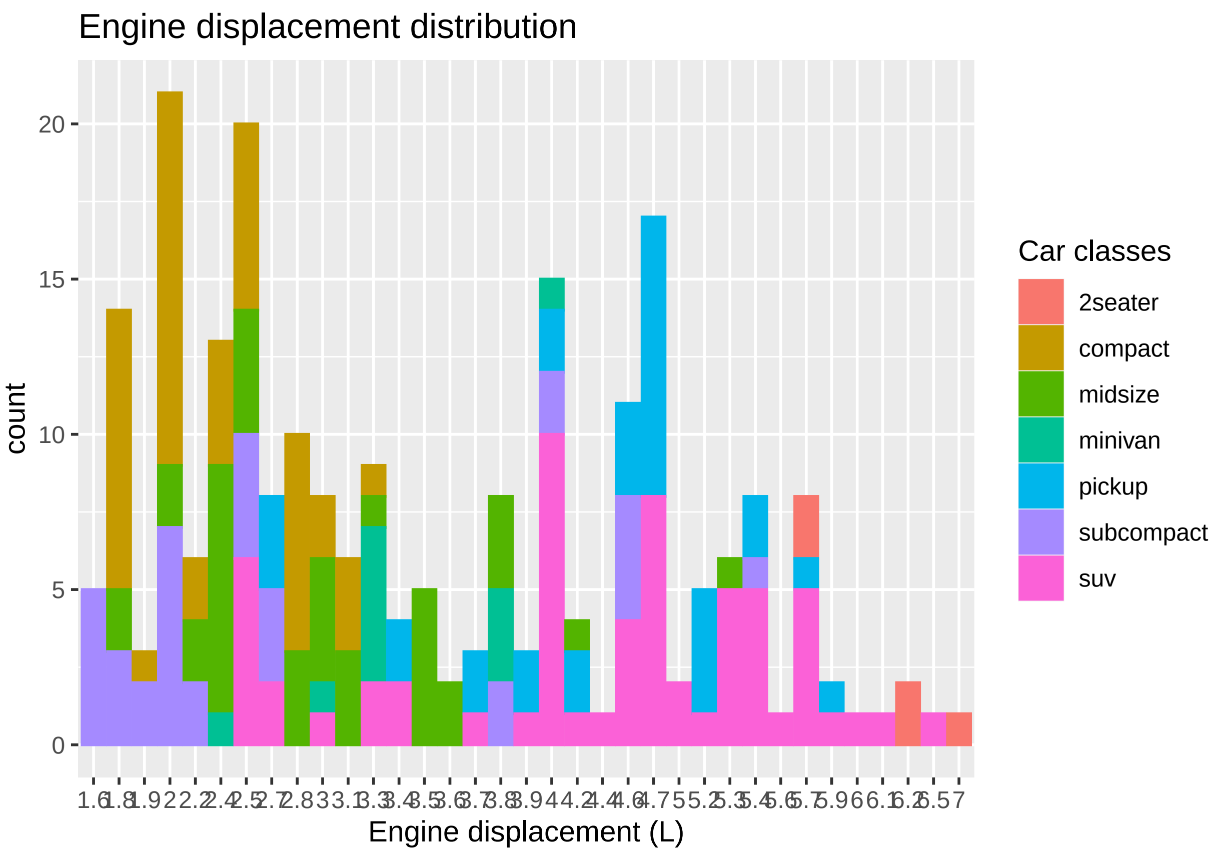
```
mpg$displ<-as.numeric(mpg$displ)

ggplot(mpg)+
  geom_bar(aes(x=displ, color=class, fill=class))+
  labs(x="Engine displacement (L)",
       color="Car classes",
       fill="Car classes",
       title="Engine displacement distribution")
```

```
## Warning: position_stack requires non-overlapping x intervals
```



If you consider that your variable should be discrete, then use *as.factor()* function as follow :

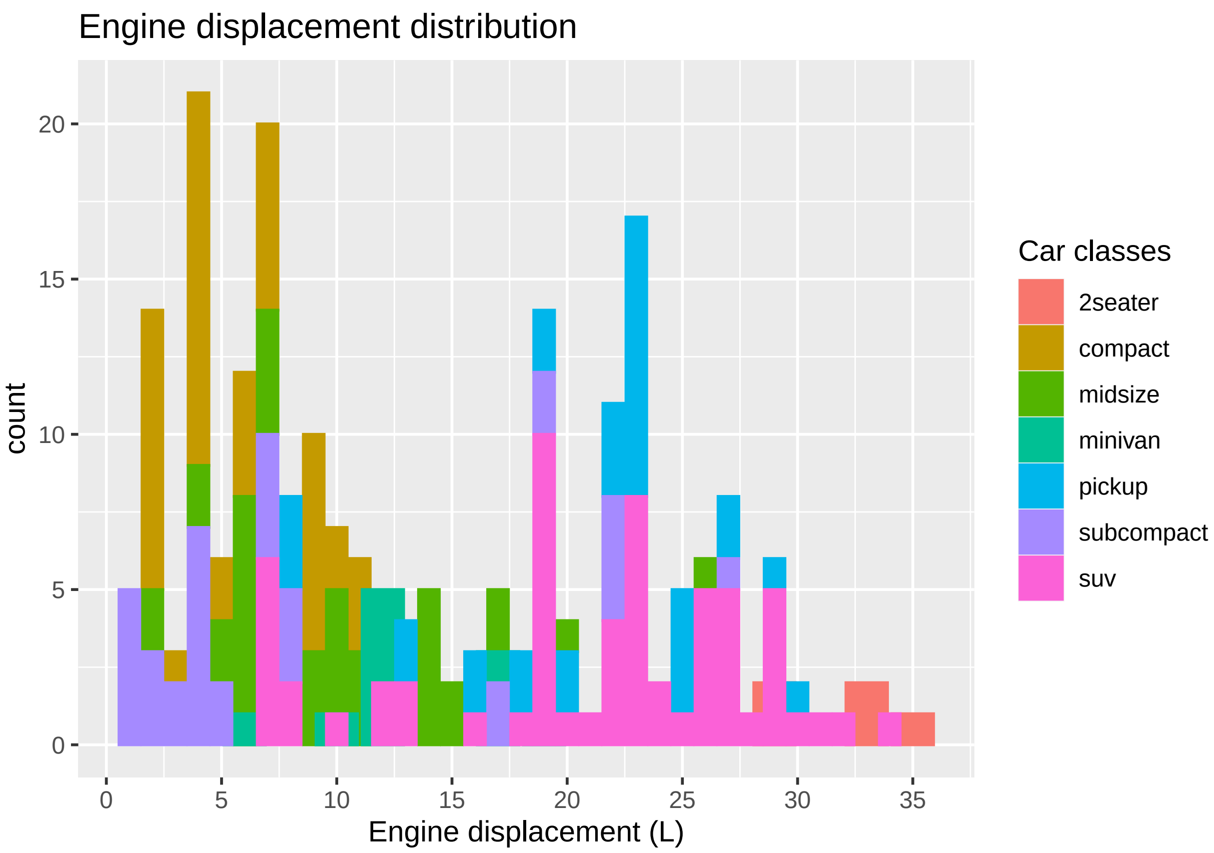
```
mpg$displ<-as.factor(mpg$displ)

ggplot(mpg)+
  geom_bar(aes(x=displ, color=class, fill=class))+
  labs(x="Engine displacement (L)",
       color="Car classes",
       fill="Car classes",
       title="Engine displacement distribution")
```



In such cases, the $x$ axis can be blurry and sometimes, the labels are not clear. To make it clearier, one could use the functions *scale_x_continuous()* or *scale_x_discrete()* as :

```
mpg$displ<-as.numeric(mpg$displ)

ggplot(mpg)+
  geom_bar(aes(x=displ, color=class, fill=class))+
  labs(x="Engine displacement (L)",
       color="Car classes",
       fill="Car classes",
       title="Engine displacement distribution")+
  scale_x_continuous(breaks=seq(0, max(mpg$displ), 5))
```

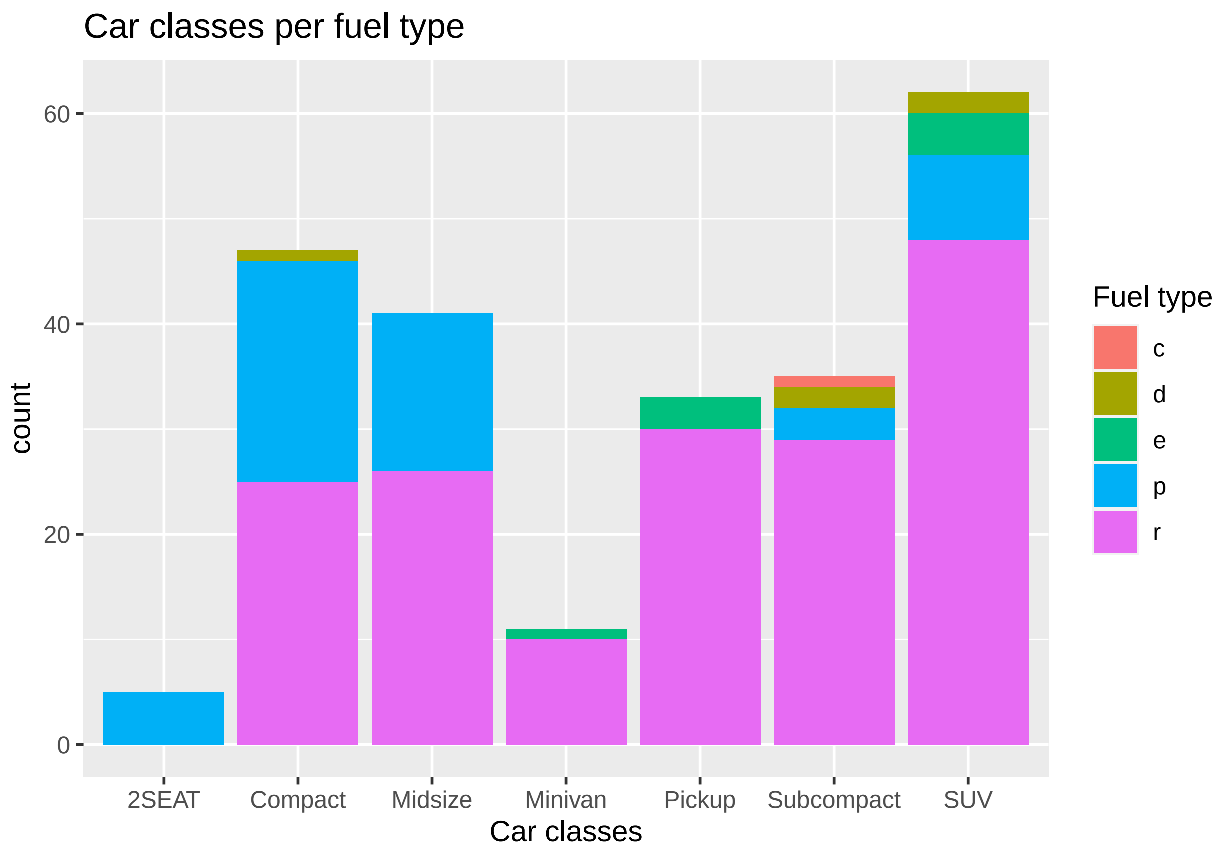
```
## Warning: position_stack requires non-overlapping x intervals
```



NB : Here, I call the max function to get the maximum value displayed on the $x$ axis. By doing so, we suppose that the maximum value can change from a dataset to another, so it's better to never use a maximum value chosen by hand.

Of course, here it doesn't make sense to convert the engine displacement values as factors. So we will try with another variable which is more suitable for this kind of work. We will use the class variable for our discrete variable. With *scale_x_discrete()* we can assign another name to each label as follow :

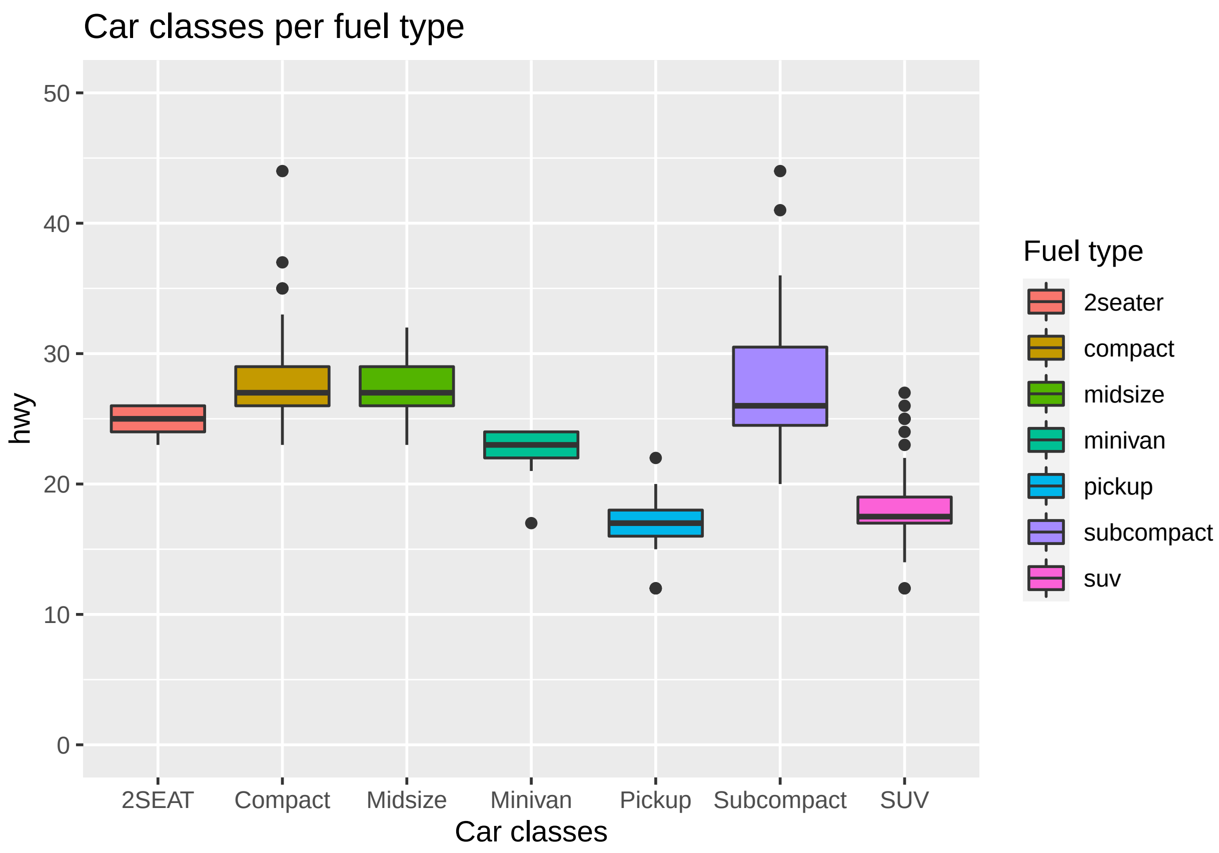
```
ggplot(mpg)+
  geom_bar(aes(x=class, fill=fl))+
  labs(x="Car classes",
       color="Fuel type",
       fill="Fuel type",
       title="Car classes per fuel type")+
  scale_x_discrete(labels=c("2seater"="2SEAT",
                            "compact"="Compact",
                            "midsize"="Midsize",
                            "minivan"="Minivan",
                            "pickup"="Pickup",
                            "subcompact"="Subcompact",
                            "suv"="SUV"))
```



## Boxplot

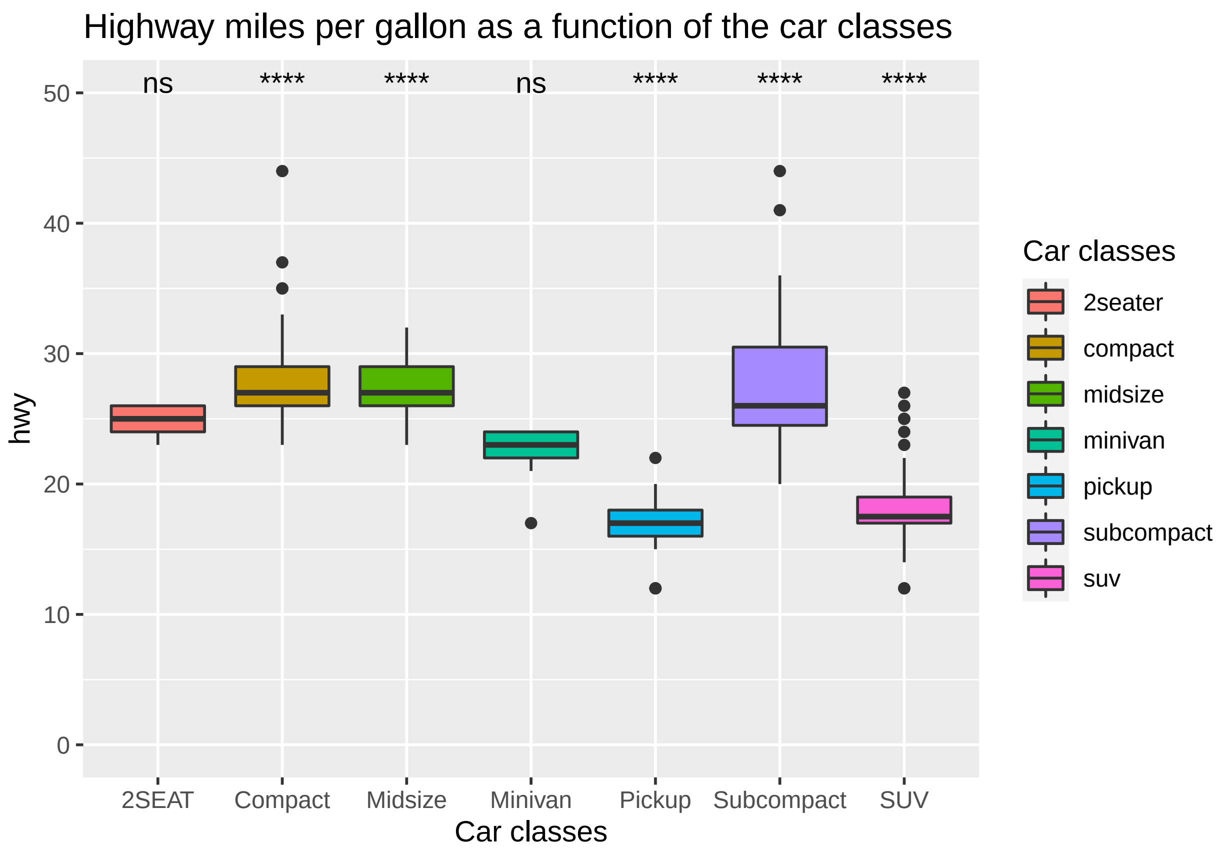
To create a boxplot with ggplot2, we use the geom_function *geom_boxplot()* as follow :

```
ggplot(mpg, aes(x=class, y=hwy))+
  geom_boxplot(aes(fill=class))+
  labs(x="Car classes",
       color="Fuel type",
       fill="Fuel type",
       title="Car classes per fuel type")+
  ylim(0, 50)+
  scale_x_discrete(labels=c("2seater"="2SEAT",
                            "compact"="Compact",
                            "midsize"="Midsize",
                            "minivan"="Minivan",
                            "pickup"="Pickup",
                            "subcompact"="Subcompact",
                            "suv"="SUV"))
```



If we want the significance level between each group, we can use the *stat_compare_means()* function as follow :

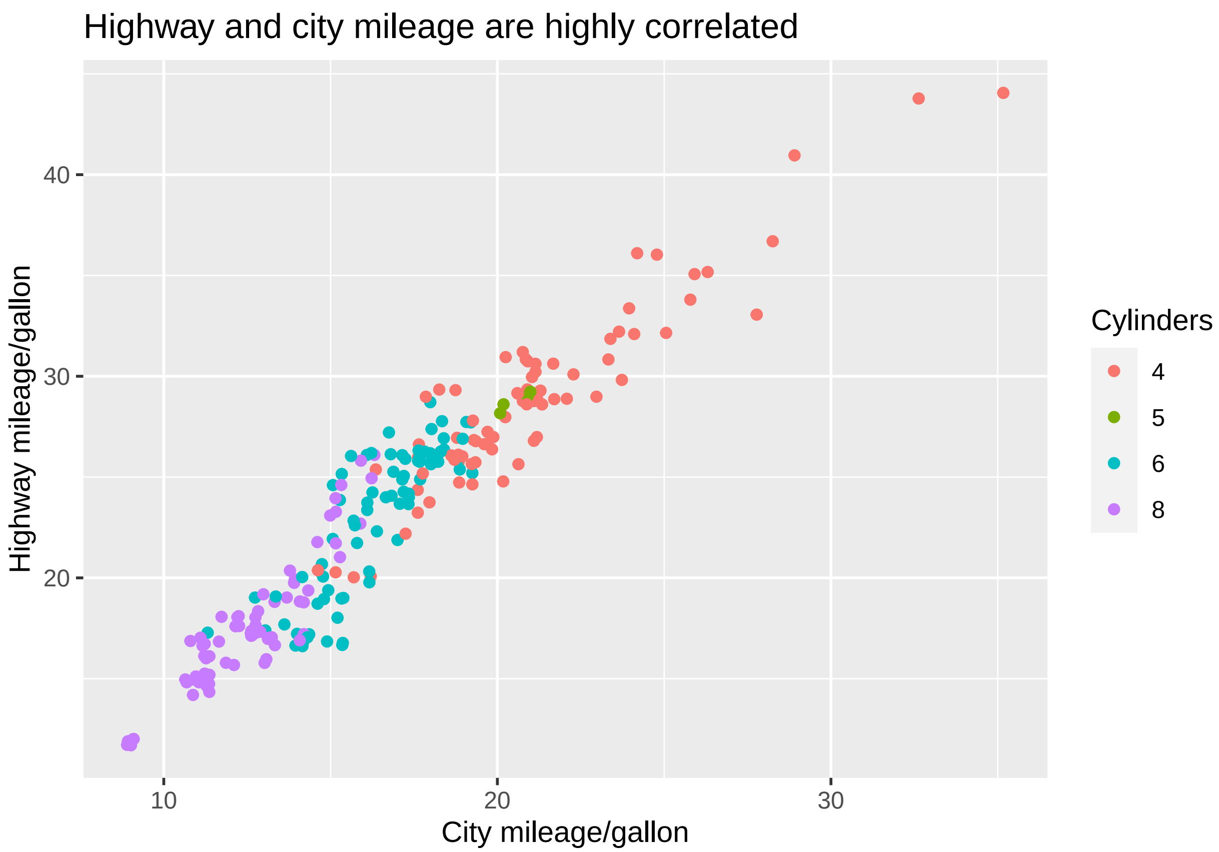
```
box<-ggplot(mpg, aes(x=class, y=hwy))+
  geom_boxplot(aes(fill=class))+
  stat_compare_means(aes(label = ..p.signif..),
                     method="wilcox.test",
                     label.x = 1.5,
                     label.y = 50,
                     ref.group = ".all.")+
  labs(x="Car classes",
       color="Car classes",
       fill="Car classes",
       title="Highway miles per gallon as a function of the car classes")+
  ylim(0, 50)+
  scale_x_discrete(labels=c("2seater"="2SEAT",
                            "compact"="Compact",
                            "midsize"="Midsize",
                            "minivan"="Minivan",
                            "pickup"="Pickup",
                            "subcompact"="Subcompact",
                            "suv"="SUV"))

box
```



## Stripchart

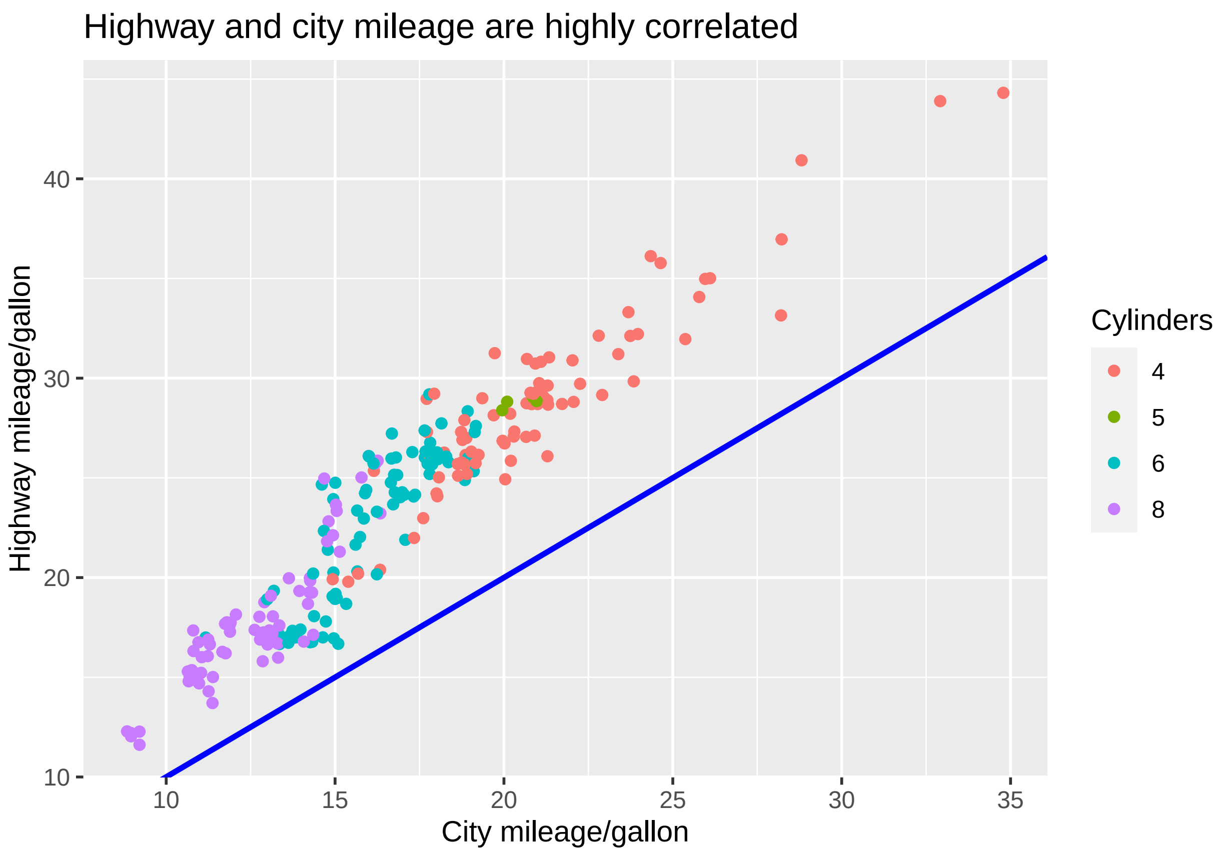
A stripchart or jitter is a plot that produces a one dimensional dot plot of the given data. We create it with the *geom_jitter()* function.

```
ggplot(mpg, aes(cty, hwy, color=factor(cyl)))+
  geom_jitter()+
  labs(
    x = "City mileage/gallon",
    y = "Highway mileage/gallon",
    colour = "Cylinders",
    title = "Highway and city mileage are highly correlated"
  )
```



We can map the tendency of the dots using the *abline()* function as follow :

```
jitt<-ggplot(mpg, aes(cty, hwy, color=factor(cyl)))+
  geom_jitter()+
  labs(
    x = "City mileage/gallon",
    y = "Highway mileage/gallon",
    colour = "Cylinders",
    title = "Highway and city mileage are highly correlated"
  )+
  geom_abline(colour="blue",
              size=1)
jitt
```

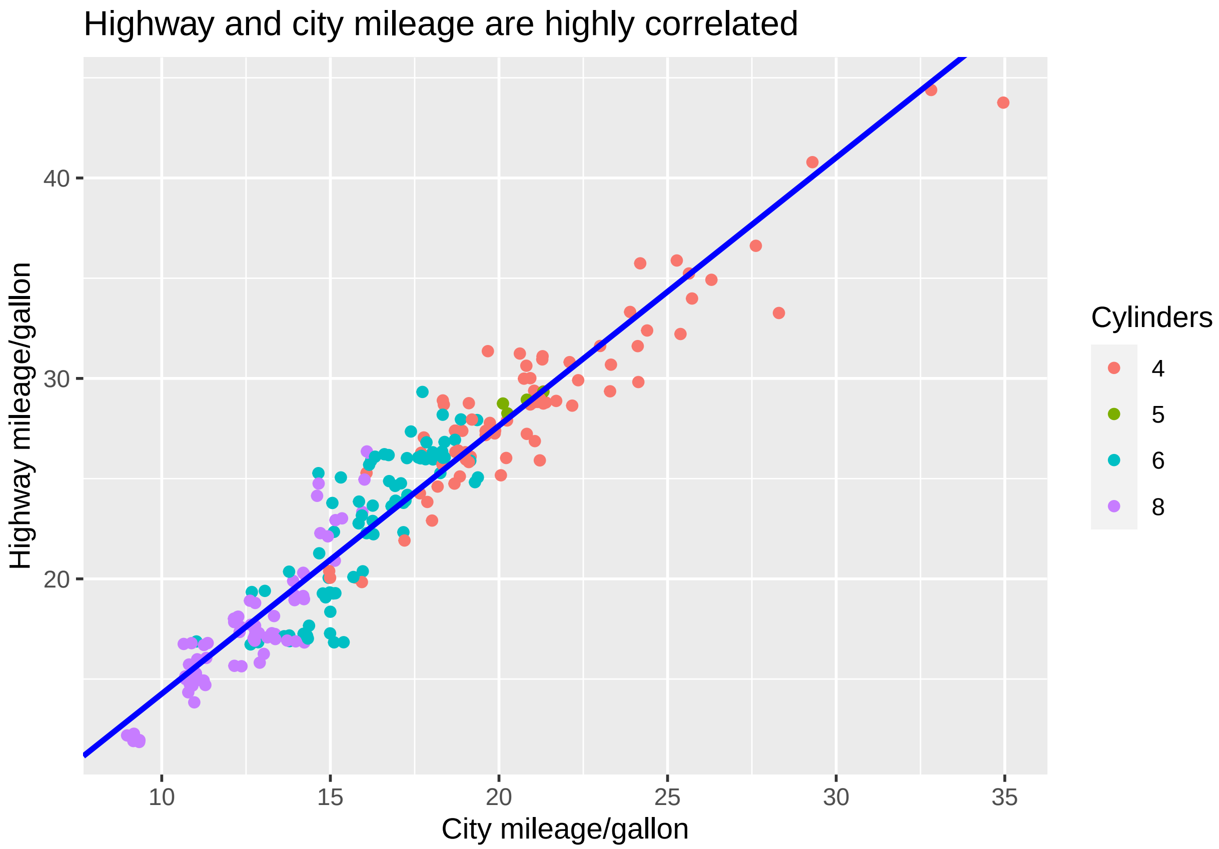


So, as you can see, the line is not mapping our dots at all. To do so, it's better to perform a linear regression to estimate the slope and intercept of the line.

```
reg<-lm(hwy~cty, data=mpg)
reg
```

```
##
## Call:
## lm(formula = hwy ~ cty, data = mpg)
##
## Coefficients:
## (Intercept)          cty
##       0.892        1.337
```

```
jitt<-ggplot(mpg, aes(cty, hwy, color=factor(cyl)))+
  geom_jitter()+
  labs(
    x = "City mileage/gallon",
    y = "Highway mileage/gallon",
    colour = "Cylinders",
    title = "Highway and city mileage are highly correlated"
  )+
  geom_abline(colour="blue",
              size=1,
              intercept = reg$coefficients[[1]],
              slope=reg$coefficients[[2]])
jitt
```

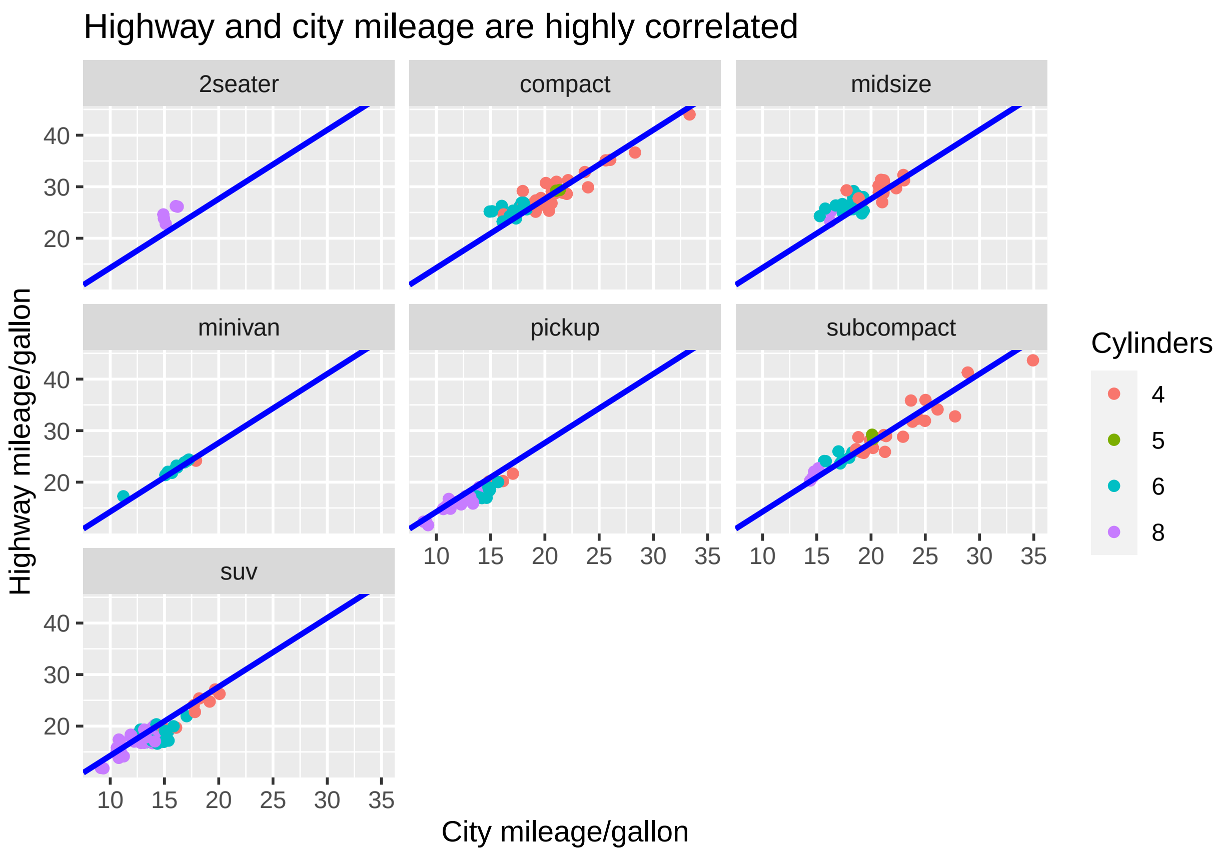


And now we have a very correlation between our $x$ and $y$ axes.

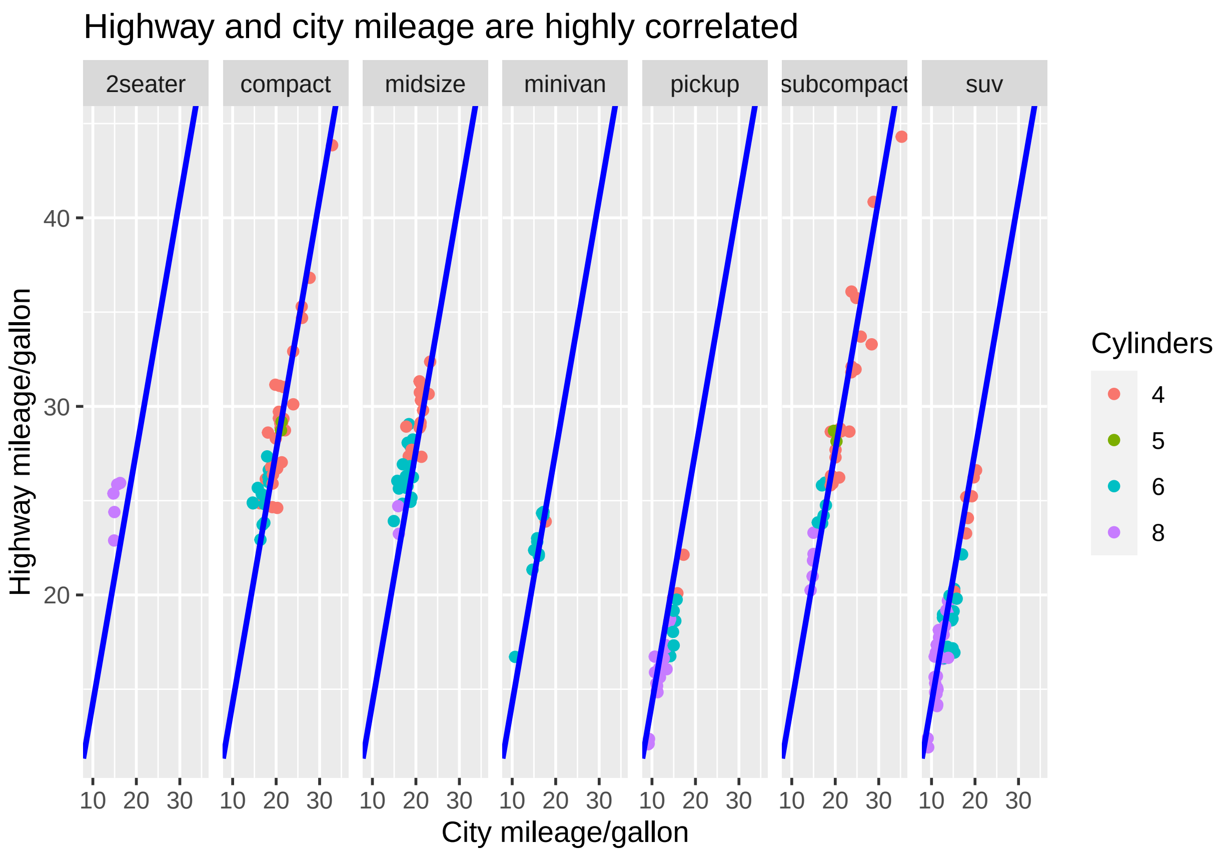
## Facetting

Using facetting, you can split your plot based on one or several variables :

```
jitt<-ggplot(mpg, aes(cty, hwy, color=factor(cyl)))+
  geom_jitter()+
  labs(
    x = "City mileage/gallon",
    y = "Highway mileage/gallon",
    colour = "Cylinders",
    title = "Highway and city mileage are highly correlated"
  )+
  facet_wrap(~class)+
  geom_abline(colour="blue",
              size=1,
              intercept = reg$coefficients[[1]],
              slope=reg$coefficients[[2]])
jitt
```

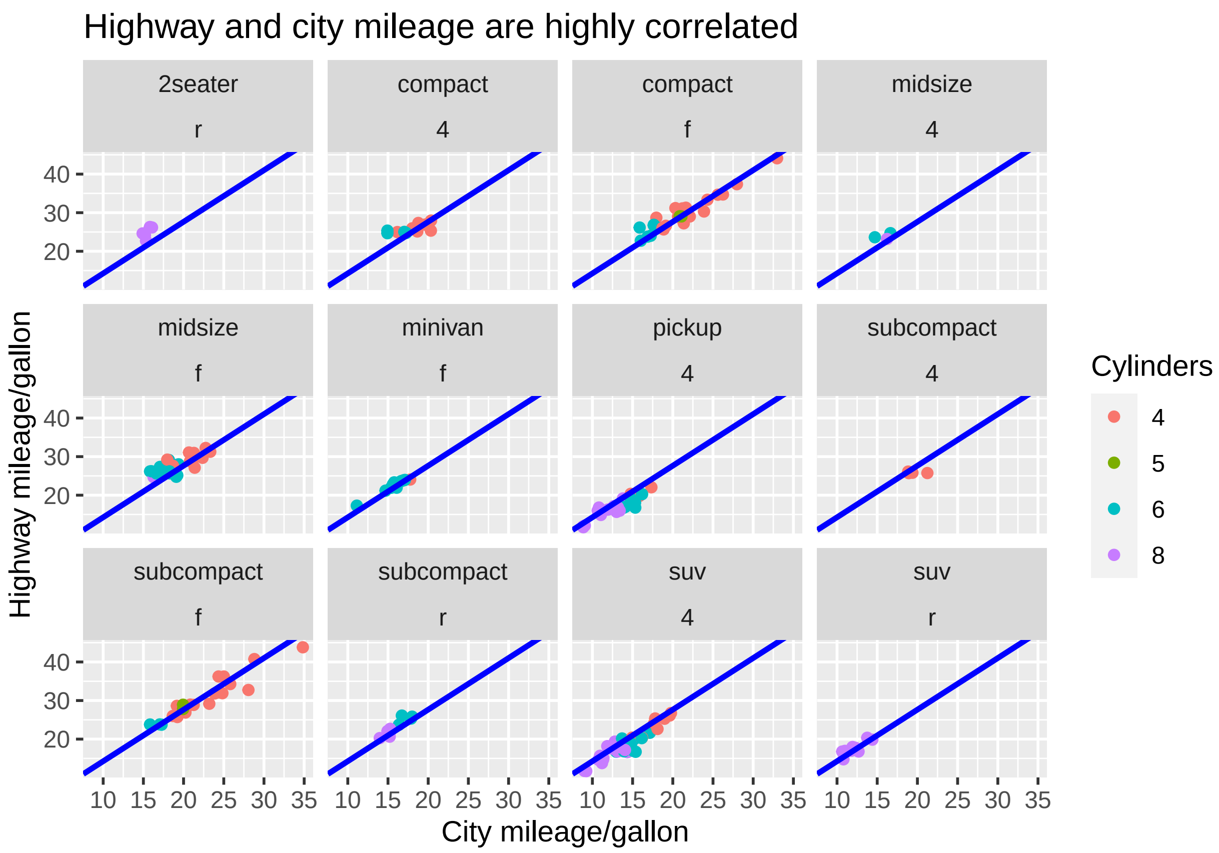


```
jitt<-ggplot(mpg, aes(cty, hwy, color=factor(cyl)))+
  geom_jitter()+
  labs(
    x = "City mileage/gallon",
    y = "Highway mileage/gallon",
    colour = "Cylinders",
    title = "Highway and city mileage are highly correlated"
  )+
  facet_grid(~class)+
  geom_abline(colour="blue",
              size=1,
              intercept = reg$coefficients[[1]],
              slope=reg$coefficients[[2]])
jitt
```



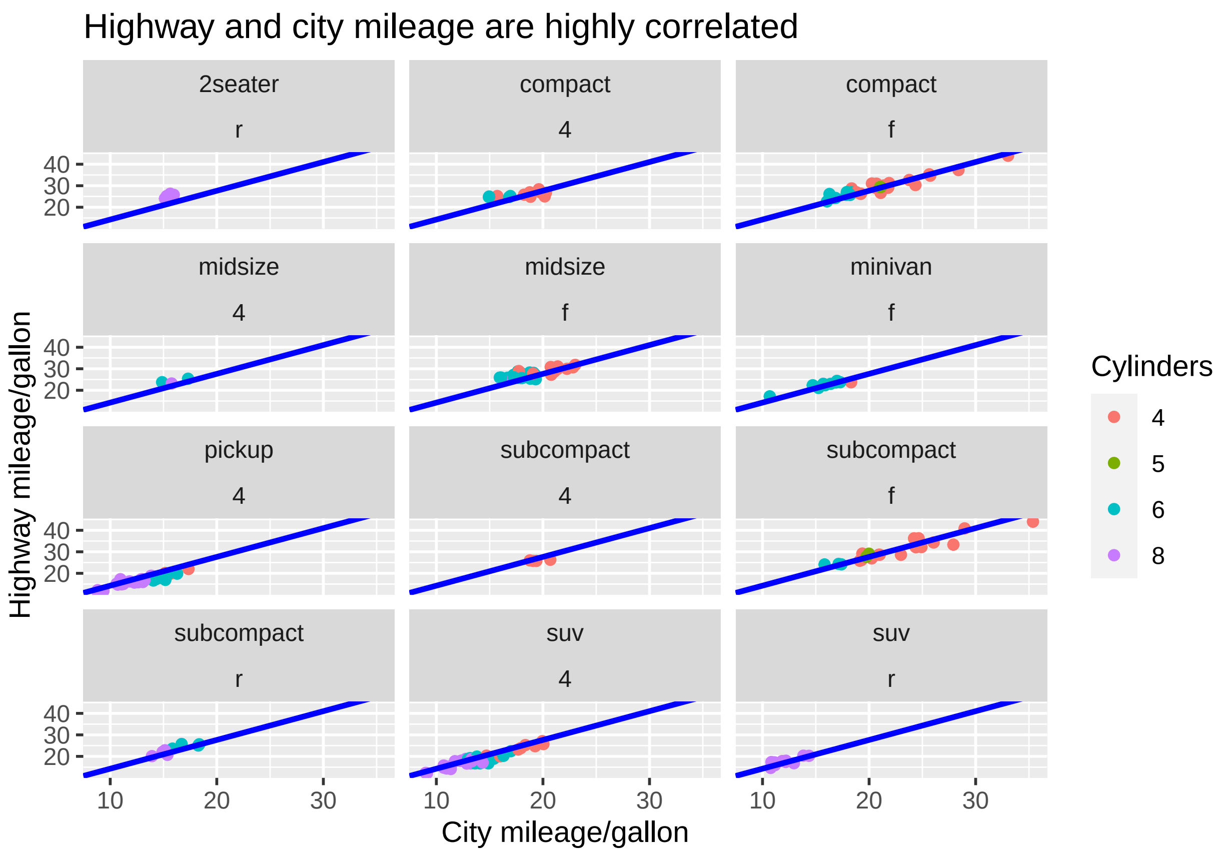
Splitting by several variables :

```
jitt<-ggplot(mpg, aes(cty, hwy, color=factor(cyl)))+
  geom_jitter()+
  labs(
    x = "City mileage/gallon",
    y = "Highway mileage/gallon",
    colour = "Cylinders",
    title = "Highway and city mileage are highly correlated"
  )+
  facet_wrap(~class+drv)+
  geom_abline(colour="blue",
              size=1,
              intercept = reg$coefficients[[1]],
              slope=reg$coefficients[[2]])
jitt
```



You can evene choose the number of rows and columns to display on the plot :

```
jitt<-ggplot(mpg, aes(cty, hwy, color=factor(cyl)))+
  geom_jitter()+
  labs(
    x = "City mileage/gallon",
    y = "Highway mileage/gallon",
    colour = "Cylinders",
    title = "Highway and city mileage are highly correlated"
  )+
  facet_wrap(~class+drv, nrow=4, ncol=3)+
  geom_abline(colour="blue",
              size=1,
              intercept = reg$coefficients[[1]],
              slope=reg$coefficients[[2]])
jitt
```



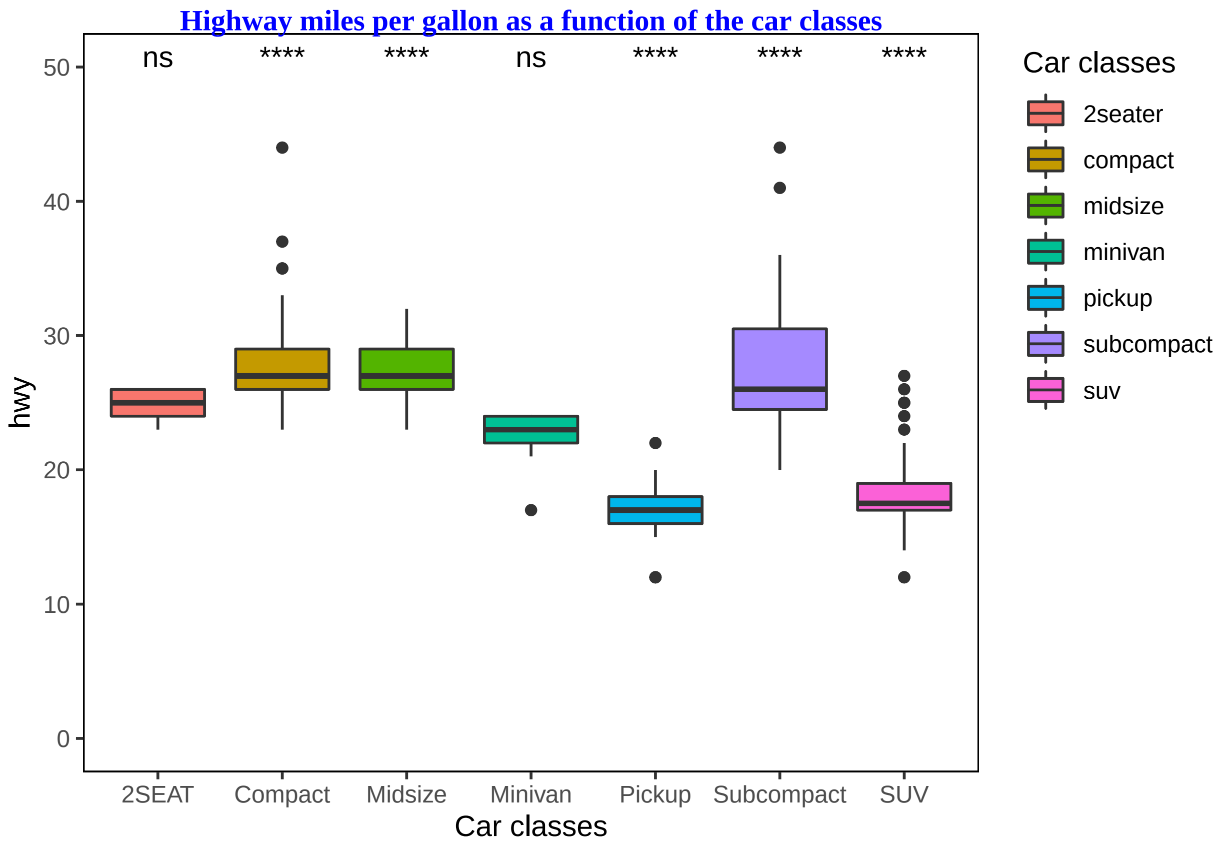
## Theme creation

There exist several themes in ggplot2, black and white, gray, void, minimal, classic, light and so on. But what if we want our own theme, with the colors we want, a grid or not, and a specific font ? We must create our own theme to do that. Here we will use the black and white theme as a base theme and the function *%+replace%* which replaces the entire element in a theme and if any element of a theme is not specified in the new theme, it will not be present in the resulting theme.

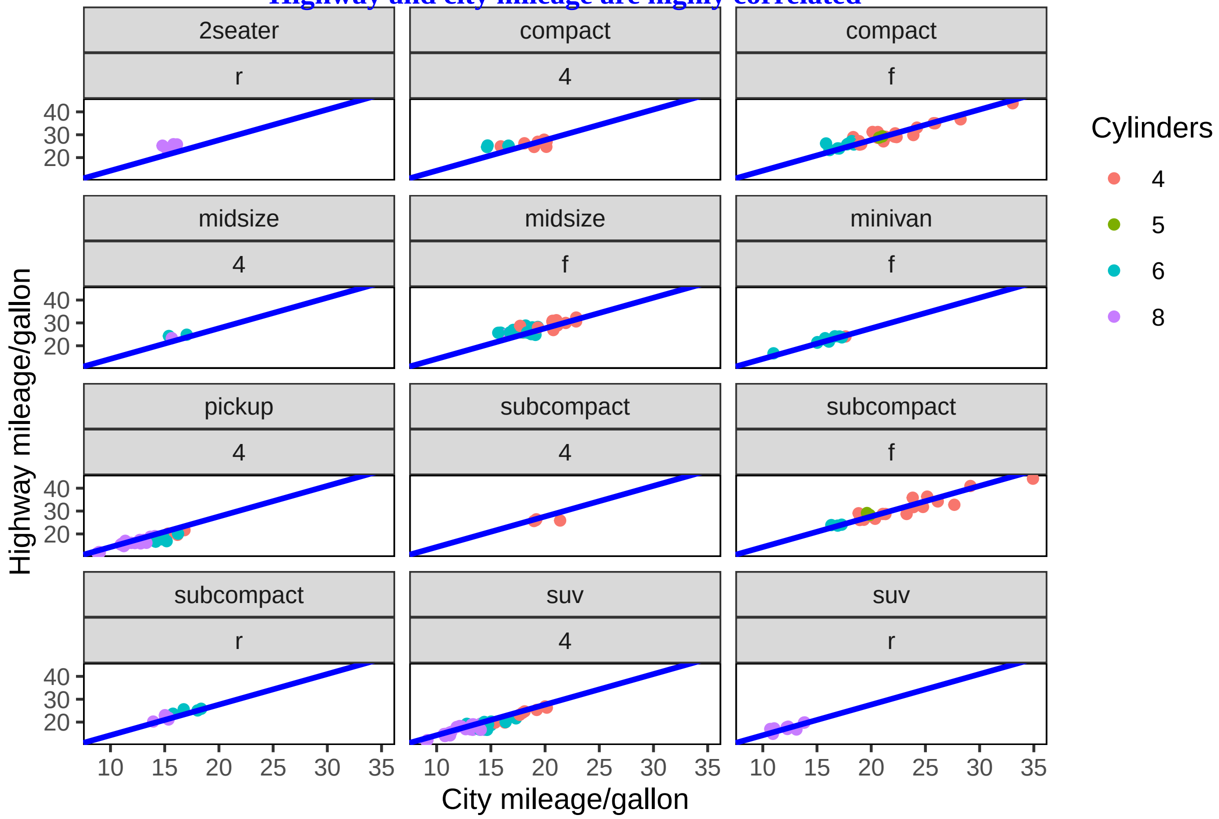
```
my_theme<-function(){
  theme_bw() %+replace%
  theme(plot.title = element_text(hjust = 0.5,
                                  face="bold",
                                  colour="blue",
                                  family = "Times"), #centers the title
        panel.border=element_blank(),
        panel.grid.major = element_blank(), #to remove the grid
        panel.grid.minor = element_blank(), #to remove the grid
        panel.background = element_rect(fill="white"), #to fill the background with a white color
        axis.line = element_blank(),
        legend.justification = c(0, 1),
        legend.position="right") #"left","top", "right", "bottom" or coordinate c(x, y) with x and y be
}
```

To apply the theme to your own plot, just add the newly created theme as if you were adding a new layer.

```
box+my_theme()
```



```
jitt+my_theme()
```


Highway and city mileage are highly correlated