The Introduction of R And ggplot2

You Zhou

June 10, 2022

Contents

What is R? **1**

Why Use R? **2**

R as a pocket calculator **2**

Logical comparison operators **3**

Data objects **3**
- Data types . . . 3
- Vectors . . . 4
- Exercise 1 . . . 5
- Lists . . . 5
- Data frame . . . 5
  - Add rows or columns . . . 6
- Exercise 2 . . . 8

Visualizing data with ggplot2 **8**
- Installation of ggplot2 . . . 8
- Usage . . . 8
  - Scatter plot . . . 9
  - Histogram . . . 12
  - Density plot . . . 13
  - Barplot . . . 14
  - Boxplot . . . 15
- Violin plot . . . 17
  - Customize ggplots . . . 18
- Exercise 3 . . . 22

What is R?

R is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS. The **R** can be downloaded from R Project. To download R, please choose your preferred CRAN mirror.

When you have downloaded and installed R, you can run R on your computer. Besides directly running it, we can also run R in an integrated working environment, e.g. **Rstudio**. The Rstudio is highly recommanded when you are working with R, and it can be downloaded from http://www.r-project.org.

Why Use R?

- It is a great resource for data analysis, data visualization, data science and machine learning
- It provides many statistical techniques (such as statistical tests, classification, clustering and data reduction)
- It is easy to draw graphs in R, like pie charts, histograms, box plot, scatter plot, etc++
- It works on different platforms (Windows, Mac, Linux)
- It is open-source and free
- It has a large community support
- It has many packages (libraries of functions) that can be used to solve different problems

R as a pocket calculator

It is possible to use R like a pocket calculator:

```
+ "plus" (addition)
- "minus" (subtraction)
* "times" (multiplication)
/ "divided by" (division)
^ "to the power of" (exponentiation)
sqrt() "square root" (taking the square root)
%% "Modulo" (modulo operation)
```

```
## addition
5+3
```

```
## [1] 8
```

```
## subtraction
5-3
```

```
## [1] 2
```

```
## multiplication
5*3
```

```
## [1] 15
```

```
## division
5/3
```

```
## [1] 1.666667
```

```
## exponentiation
5^3
```

```
## [1] 125
```

```
## taking the square root
sqrt(9)
```

```
## [1] 3
```

```
## Modulo
5%%3
```

```
## [1] 2
```

Logical comparison operators

```
    < for less than
    > for greater than
    <= for less than or equal to
    >= for greater than or equal to
    == for equal to each other
    != not equal to each other
```

```
## Some examples
5 > 3
```

```
## [1] TRUE
```

```
5 >= 6
```

```
## [1] FALSE
```

```
5 != 10
```

```
## [1] TRUE
```

Data objects

Data we want to work with are typed into R as data objects. For example, we can create an object `Grogu` and assign it the value `5` with the assignment operator `<-` or `=`.

```
Grogu <- 5 ## or Grogu = 5
Grogu
```

```
## [1] 5
```

It is possible to do calculations with the data in `Grogu`.

```
Grogu - 1
```

```
## [1] 4
```

Data types

Basic data types in R can be divided into the following types:

```
    numeric - (10.5, 55, 787)
    integer - (1L, 55L, 100L, where the letter "L" declares this as an integer)
    complex - (9 + 3i, where "i" is the imaginary part)
    character (a.k.a. string) - ("k", "R is exciting", "FALSE", "11.5")
    logical (a.k.a. boolean) - (TRUE or FALSE)
    We can use the class() function to check the data type of a variable:
```

```
## numeric
x <- 10.5
class(x)
```

```
## [1] "numeric"
```

```
## integer
x <- 1000L
class(x)
```

```
## [1] "integer"
```

```r
## complex
x <- 9i + 3
class(x)
```

```
## [1] "complex"
```

```r
## character/string
x <- "R is exciting"
class(x)
```

```
## [1] "character"
```

```r
## logical/boolean
x <- TRUE
class(x)
```

```
## [1] "logical"
```

Vectors

Vector is a basic data structure in R. It contains element of the same type. The data types can be logical, numeric, character, etc. To combine the multiple of items into a vector, use the `c()` function and separate the items by a comma (`,`).

```r
## assign elements to vectors
classmate <- c("John", "Lina", "Kristin")
numbers <- c(1, 2, 3)
## check the data type
class(numbers)
```

```
## [1] "numeric"
```

The single components of a vector can be accessed by squared brackets. For instance, to retrieve the second element of the vector **classmate**.

```r
classmate[2]
```

```
## [1] "Lina"
```

When the vector is composed of numeric elements, some basic operations can be done:

```r
numeric_vect <- c(1,2,1,10,50)

## compute the mean of the elements in the vector
mean(numeric_vect)
```

```
## [1] 12.8
```

```r
## compute the sum of the elements in the vector
sum(numeric_vect)
```

```
## [1] 64
```

```r
## compute the median of the elements in the vector
median(numeric_vect)
```

```
## [1] 2
```

```r
## add a value to all elements in the vector
numeric_vect + 1
```

```
## [1]  2  3  2 11 51
```

Exercise 1

Please combine the following elements into a vector, and find out its data type.

```
1       2       A       B
```

Lists

A list in R can *contain many different data types* inside it. A list is a collection of data which is ordered and changeable. To create a list, we can use the `list()` function.

```
alist <- list(1, c(2, 3), "A")
```

We can access the list items by referring to its index number, inside brackets. The first item has index 1, the second item has index 2, and so on.

```
alist[2]
```

```
## [[1]]
## [1] 2 3
```

For checking out the data type in a list, we can use the `str()` function.

```
str(alist)
```

```
## List of 3
##  $ : num 1
##  $ : num [1:2] 2 3
##  $ : chr "A"
```

Data frame

Data frames are data displayed in a format as a table. Data Frames can have different types of data inside it. While the first column can be character, the second and third can be numeric or logical. However, each column should have the same type of data. We can use the `data.frame()` function to create a data frame.

```
## create a data frame and assign it to a object df
df <- data.frame(
    Age = c(20, 21, 20),
    classmate = classmate,
    like_coffee = c(TRUE, TRUE, FALSE)
)
df
```

```
##   Age classmate like_coffee
## 1  20      John        TRUE
## 2  21      Lina        TRUE
## 3  20   Kristin       FALSE
```

Use the `summary()` function we can quickly summarize the data from a data frame.

```
summary(df)
```

```
##       Age         classmate         like_coffee
##  Min.   :20.00   Length:3           Mode :logical
##  1st Qu.:20.00   Class :character   FALSE:1
##  Median :20.00   Mode  :character   TRUE :2
##  Mean   :20.33
##  3rd Qu.:20.50
##  Max.   :21.00
```

For accessing the items in data frame, we have some different methods. We can use single brackets `[ ]`, double brackets `[[ ]]` or `$` to access columns from a data frame.

```
df[1] ## or df["Age"]
```

```
##   Age
## 1  20
## 2  21
## 3  20
```

```
df[["classmate"]]
```

```
## [1] "John"    "Lina"    "Kristin"
```

```
df$like_coffee
```

```
## [1]  TRUE  TRUE FALSE
```

We can also select the row or column in a data frame like this:

```
## access the row 1 in df
df[1,]
```

```
##   Age classmate like_coffee
## 1  20      John        TRUE
```

```
## access the column 1 in df
df[,1]
```

```
## [1] 20 21 20
```

To find out the number of rows and columns, we can use the function `nrow()` and `ncol()` or `dim()` function.

```
## checking how many rows are there in df
nrow(df)
```

```
## [1] 3
```

```
## checking how many columns are there in df
ncol(df)
```

```
## [1] 3
```

```
## checking the dimension of df
dim(df)
```

```
## [1] 3 3
```

```
## we can check the data type in the data frame with str()
str(df)
```

```
## 'data.frame':    3 obs. of  3 variables:
##  $ Age        : num  20 21 20
##  $ classmate  : chr  "John" "Lina" "Kristin"
##  $ like_coffee: logi  TRUE TRUE FALSE
```

Add rows or columns

For adding rows, we can use the `rbind()` function or directly assign the elements to it.

```
## rbind function combines rows
new_df <- rbind(df, c(19, "Nana", TRUE))
df[4,] <- c(19, "Nana", TRUE)
new_df
```

```
##   Age classmate like_coffee
## 1  20      John        TRUE
## 2  21      Lina        TRUE
## 3  20   Kristin       FALSE
## 4  19      Nana        TRUE
```

```
df
```

```
##   Age classmate like_coffee
## 1  20      John        TRUE
## 2  21      Lina        TRUE
## 3  20   Kristin       FALSE
## 4  19      Nana        TRUE
```

For adding columns, we can use `cbind()` function, `$` or `[]`.

```
df$height <- c(180, 170, 165, 162)
df <- cbind(df, weight = c(75, 60, 59, 50))
df[,"like_star_wars"] <- c(TRUE, TRUE, TRUE, FALSE)

df
```

```
##   Age classmate like_coffee height weight like_star_wars
## 1  20      John        TRUE    180     75           TRUE
## 2  21      Lina        TRUE    170     60           TRUE
## 3  20   Kristin       FALSE    165     59           TRUE
## 4  19      Nana        TRUE    162     50          FALSE
```

If you would like to remove a column or row, you can use the `[]` function.

```
# Remove the first row and column
df_new <- df[-1, -1]
df_new
```

```
##   classmate like_coffee height weight like_star_wars
## 2      Lina        TRUE    170     60           TRUE
## 3   Kristin       FALSE    165     59           TRUE
## 4      Nana        TRUE    162     50          FALSE
```

The data frame can be subset specific criteria:

```
## select rows with Age < 20
df[df$Age < 20, ]
```

```
##   Age classmate like_coffee height weight like_star_wars
## 4  19      Nana        TRUE    162     50          FALSE
```

```
## select the coffee lovers
subset(df, like_coffee == TRUE)
```

```
##   Age classmate like_coffee height weight like_star_wars
## 1  20      John        TRUE    180     75           TRUE
## 2  21      Lina        TRUE    170     60           TRUE
## 4  19      Nana        TRUE    162     50          FALSE
```

```
## select the coffee lovers who has an Age <= 20
df[df$Age <= 20 & df$like_coffee == TRUE, ]
```

```
##   Age classmate like_coffee height weight like_star_wars
```

```
## 1  20      John        TRUE    180     75           TRUE
## 4  19      Nana        TRUE    162     50          FALSE
```

```
## select the height >= 180 or name Kristin
df[df$classmate == "Kristin" | df$height >= 180, ]
```

```
##   Age classmate like_coffee height weight like_star_wars
## 1  20      John        TRUE    180     75           TRUE
## 3  20   Kristin       FALSE    165     59           TRUE
```

Exercise 2

Create a two columns data frame with these two vectors and select the rows with the `order` value >= 15.

```
order <- c( 6, 14, 21, 1, 3, 18, 16, 2, 12, 5,
            10, 19, 25, 9, 15, 11, 20, 7, 24, 13,
            22, 23, 17, 4, 8)
ABC <- c("F", "N", "V", "A", "C", "S", "P", "B", "L", "E",
         "J", "T", "Z", "I", "O", "K", "U", "G", "Y", "M",
         "W", "X", "R", "D", "H")
```

Visualizing data with ggplot2

`ggplot2` is a package for creating the advanced graphics in R. You provide the data, tell ggplot2 how to map variables to aesthetics, what graphical primitives to use, and it takes care of the details.

Installation of ggplot2

For installing packages, we can use the `install.packages()` function.

```
## install ggplot2
install.packages("ggplot2")
```

After the package installation, we need to load the package into our R environment, so we can use all the functions inside the package.

```
## load the ggplot2 package
library(ggplot2)
## Or we can use ggplot2:: to use the function within the ggplot2 package
## i.e. ggplot2::ggplot()
```

Usage

The function `ggplot()` offers an easy way to produce complex graphics. It requires two main arguments:
1) data: the data set to be plotted, usually a `data.frame`
2) mapping: aesthetic mappings provided by `aes()` function

Components like points, lines, bars etc. can be added to a plot with `geom_*()` functions following the symbol `+`. For a comprehensive list of `geom_*()` functions see https://ggplot2.tidyverse.org/reference/.

Besides, the cheetsheet of `ggplot2` can be downloaded as follow: https://github.com/rstudio/cheatsheets/blob/main/data-visualization-2.1.pdf. **HISS**~

For showing how the `ggplot()` works, we will use the ggplot2 embeded data set `mpg` as a test data set. This data is about technical spec of cars. Here are some explanations for the attributions in this data set:
1) manufacturer
2) model > model name
3) displ > engine displacement, in litres or size of engine

4) year > year of manufacture
5) cyl > number of cylinders
6) trans > type of transmission
7) drv > f = front-wheel drive, r = rear wheel drive, 4 = 4 wheel drive
8) cty > city miles per gallon
9) hwy > highway miles per gallon or efficiency
10) fl > fuel type
11) class > "type" of car

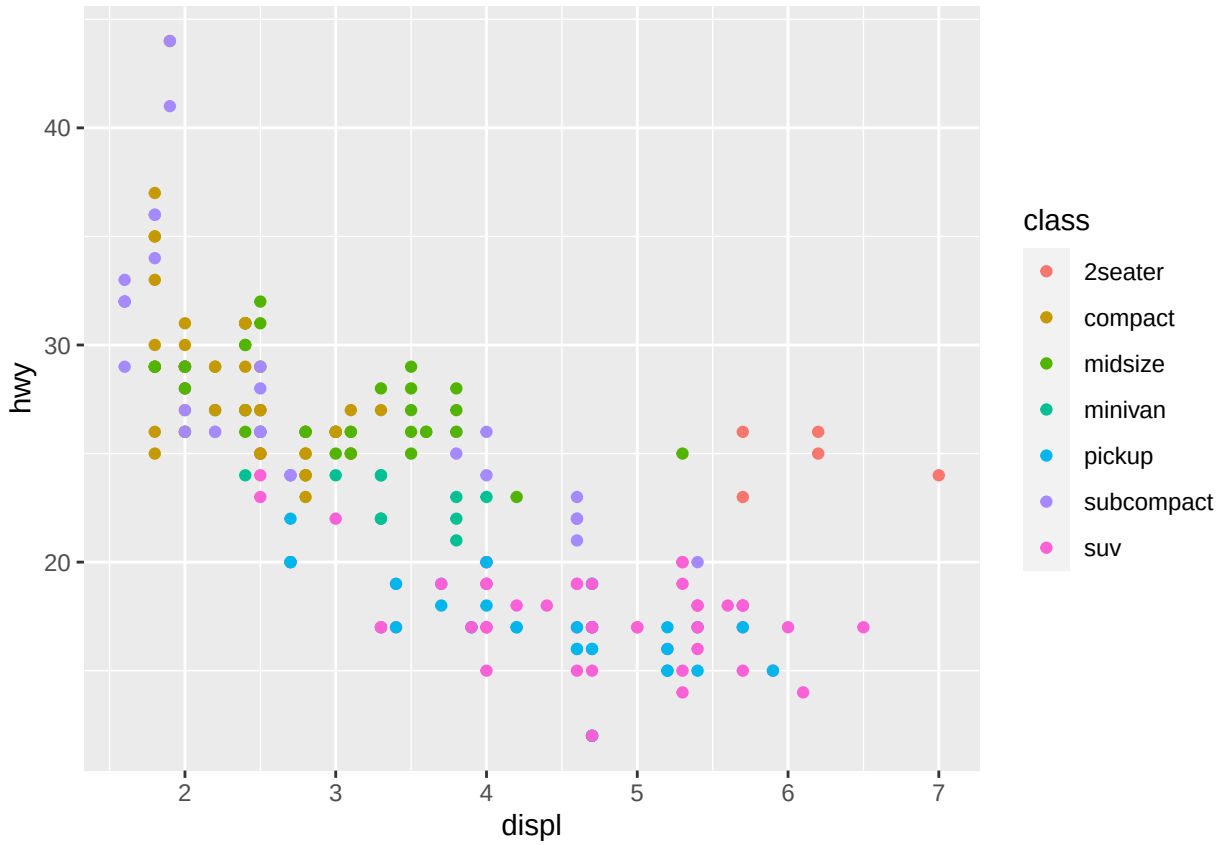
```
mpg
```

```
## # 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
```

Scatter plot

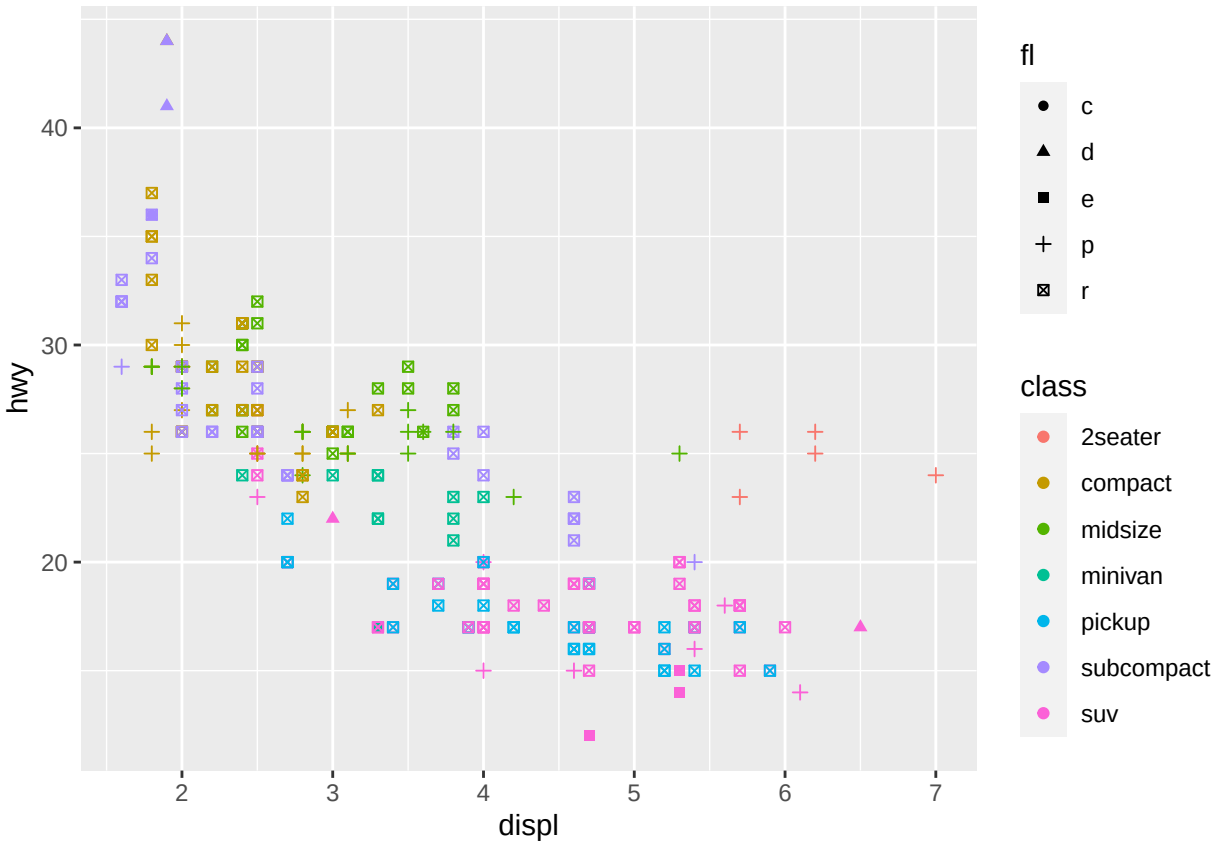
Here is an example for making a scatter plot.

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



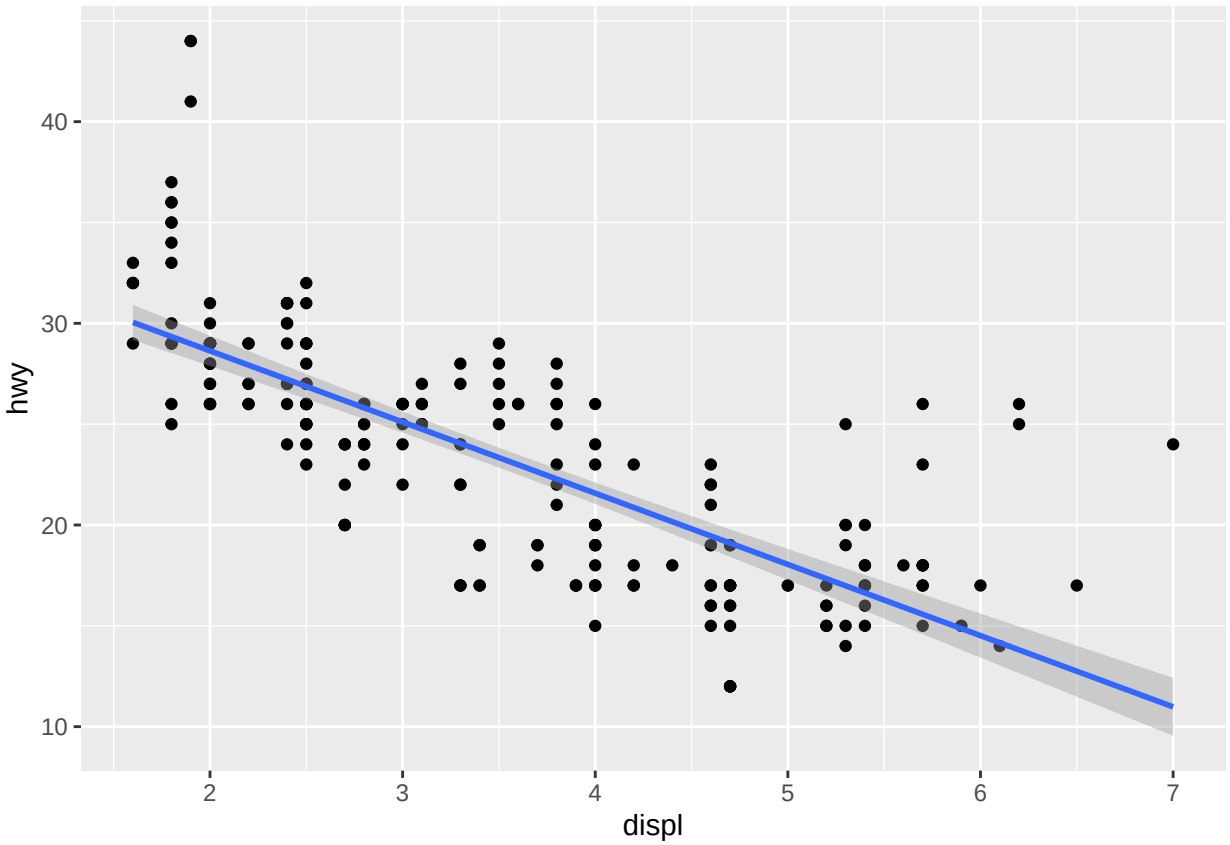
Shape the shape by the fuel type (fl).

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



Add regression line to all the points.

```
ggplot(data = mpg, aes(x = displ, y = hwy)) +
  geom_point() +
  geom_smooth(method='lm',formula=y~x)
```

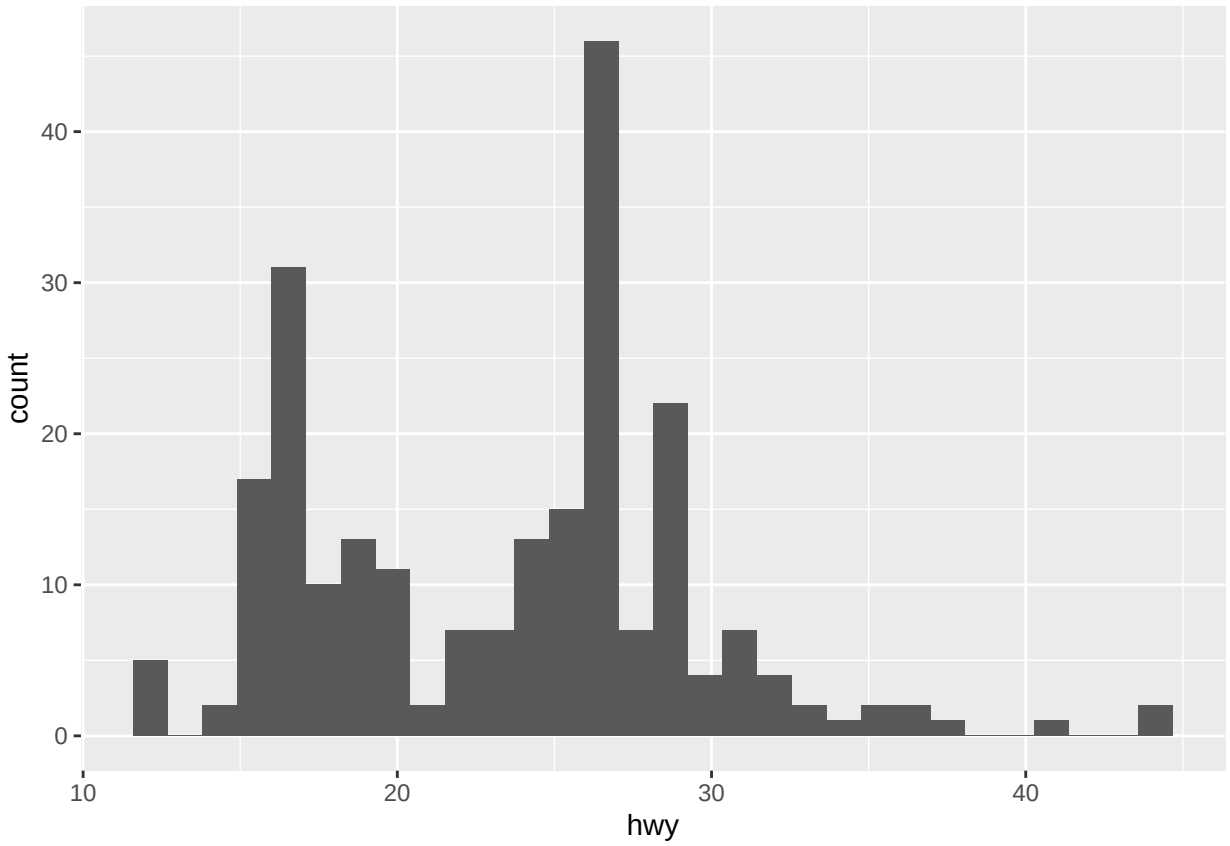


As you can see up there, the gramma of `ggplot2` is simple. You can just input the data set, select `aesthetic` then add the plot that you want to make layer by layer with `+`.

Histogram

A histogram is an approximate representation of the distribution of numerical data.

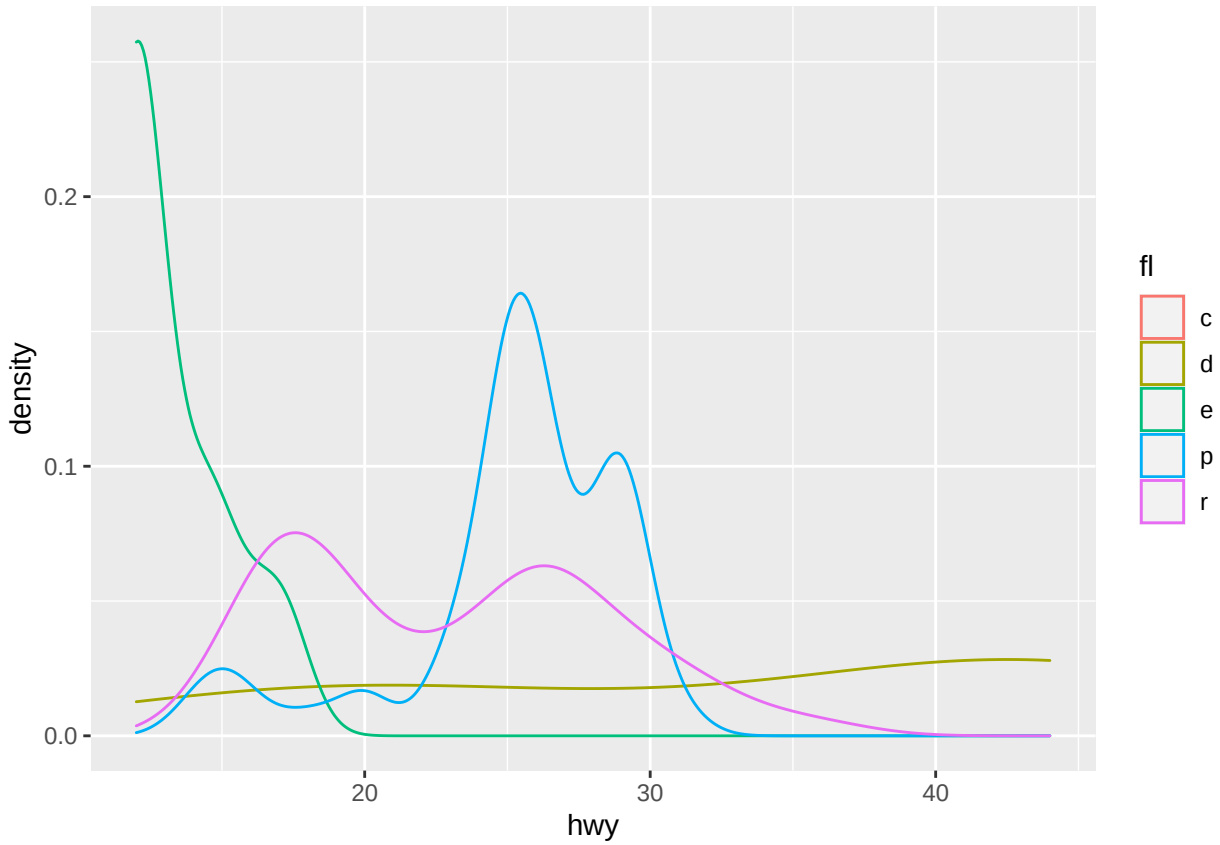
```
## here we use histogram to show the distribution of highway miles per gallon or efficiency
ggplot(data=mpg,aes(x=hwy)) + geom_histogram()
```



Density plot

A density plot shows the distribution of numerical data.

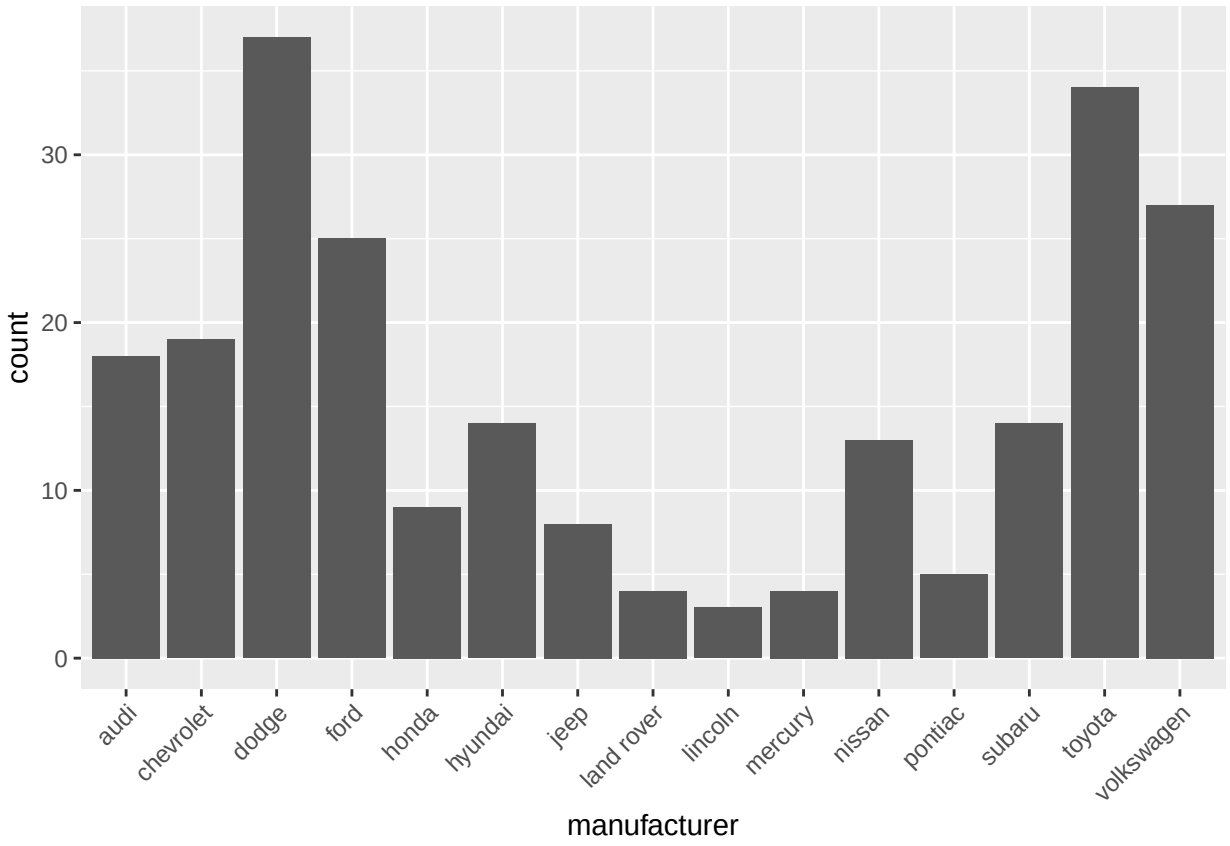
```
ggplot(data=mpg, aes(x=hwy, color = fl)) + geom_density()
```



Barplot

The barplot can be used to compare the numbers among different groups.

```
ggplot(data = mpg, aes(x = manufacturer)) +
    geom_bar() +
    theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust=1)) ## this command adjust the angle
```



Boxplot

In descriptive statistics, a boxplot is a method for graphically demonstrating the locality, spread and skewness groups of numerical data through their quartiles.

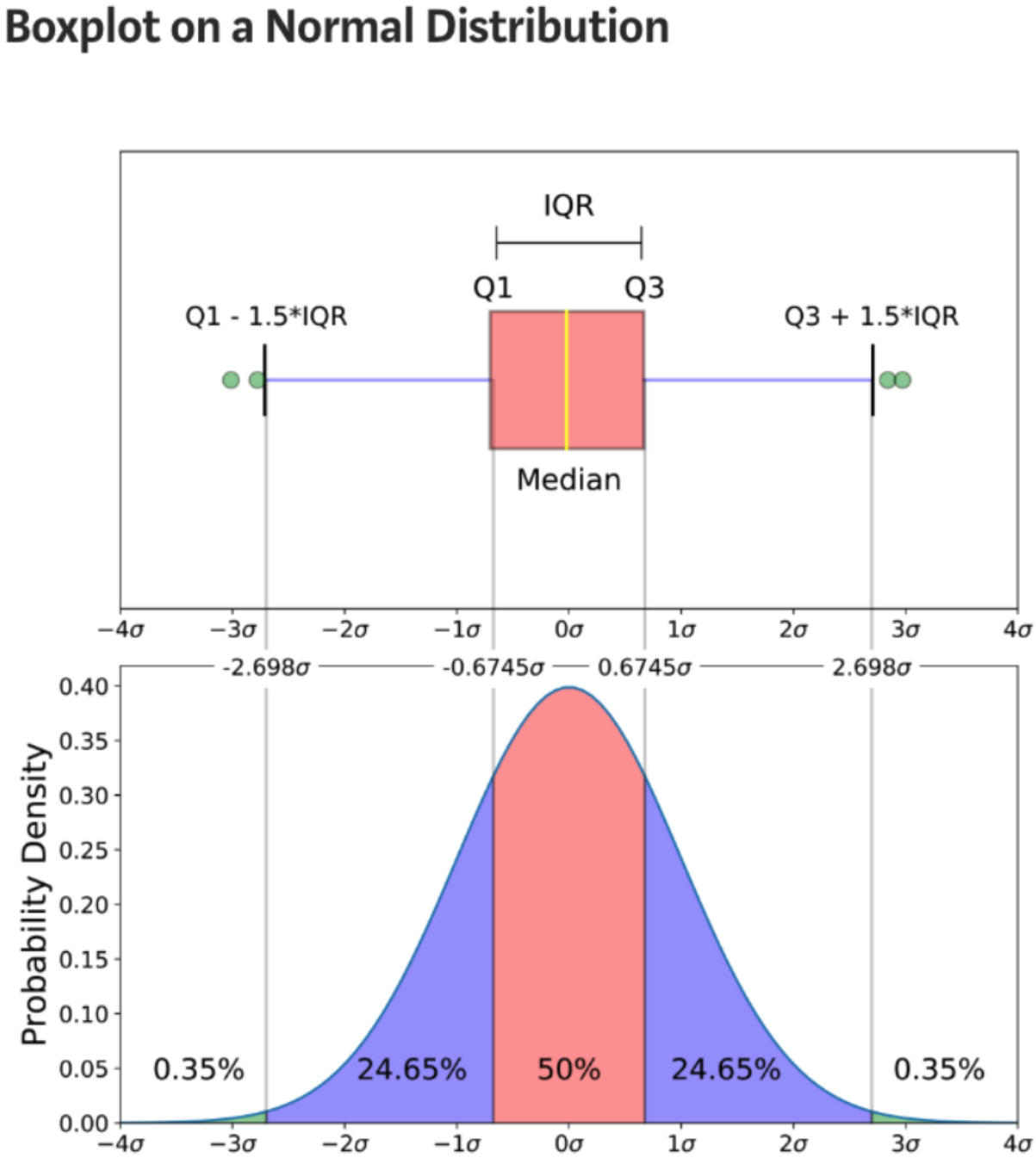
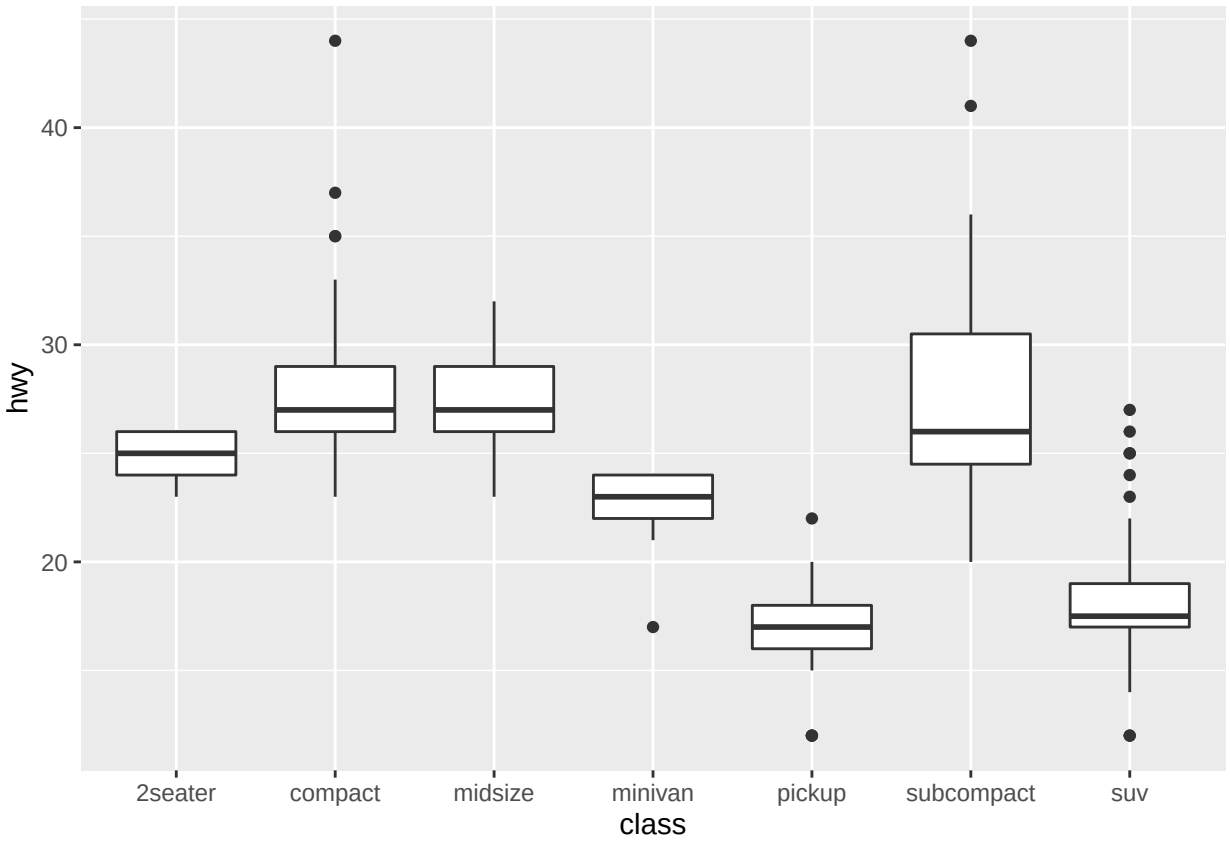


Figure 1: The explanation of boxplot

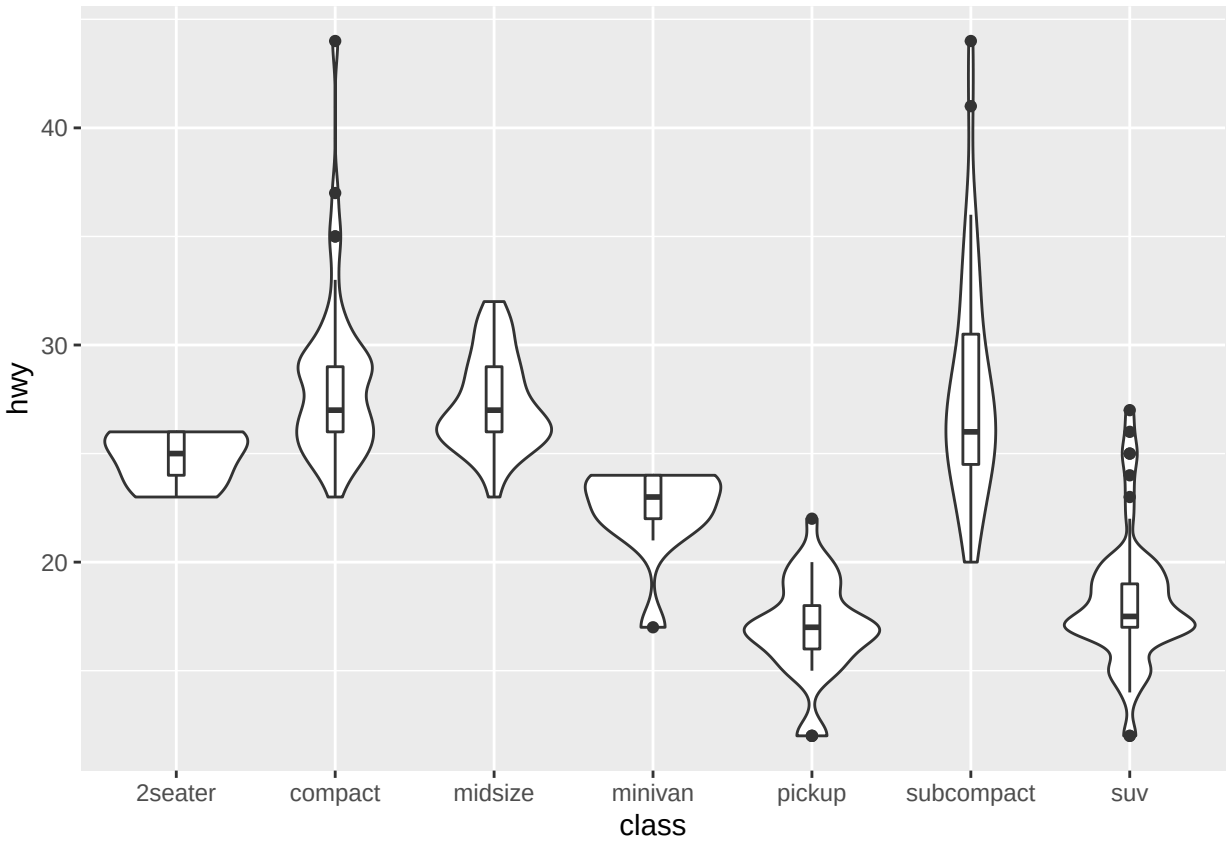
```
## the hwy
ggplot(mpg, aes(x = class, y  = hwy)) +
    geom_boxplot()
```



Violin plot

Another graphics useful to observe the distribution of data is a violin plot, a sort density plot that is rotated and placed on each side, to show the distribution shape of the data.

```
ggplot(mpg, aes(x = class, y  = hwy)) +
  geom_violin()  +
  geom_boxplot(width = 0.1)
```

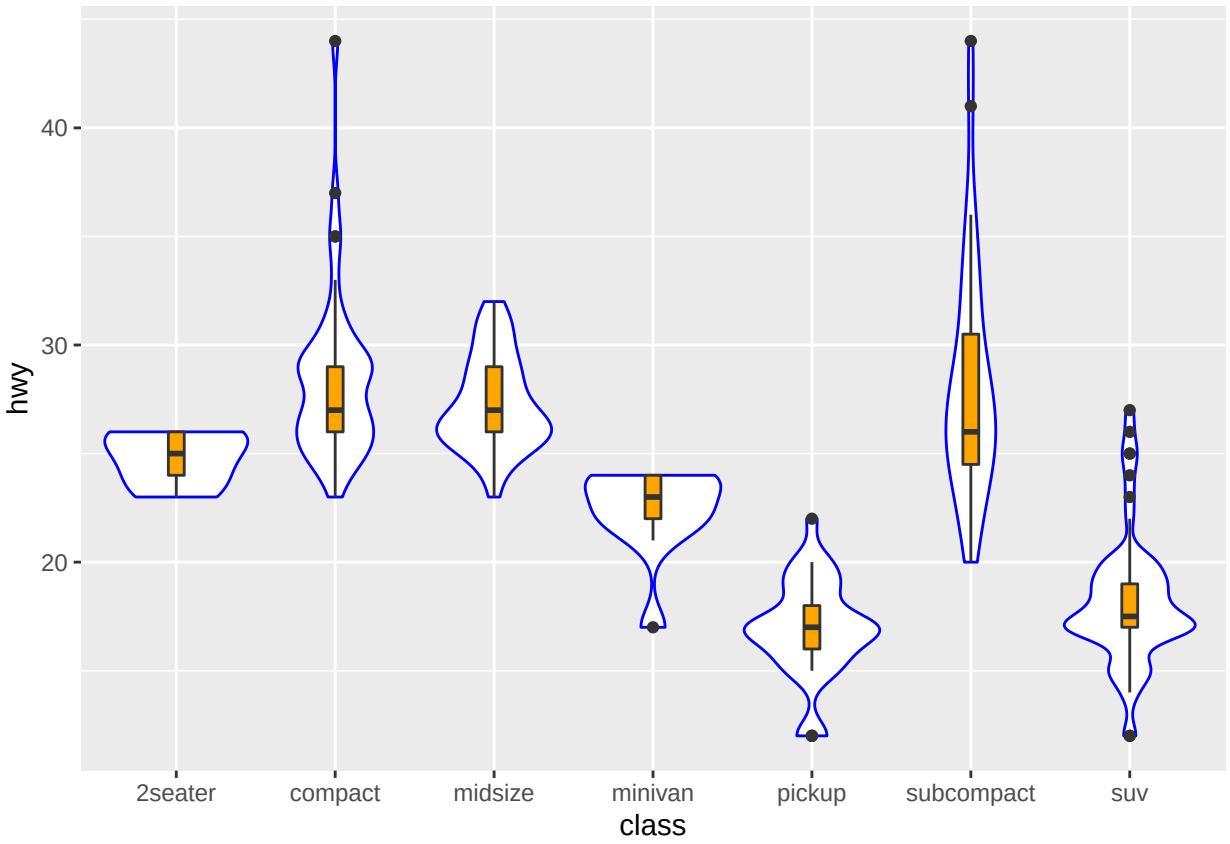


Customize ggplots

Besides make easy plots with the defaults, we can also customize almost everything in the plot, e.g. the font size, color, text, labels and so on.

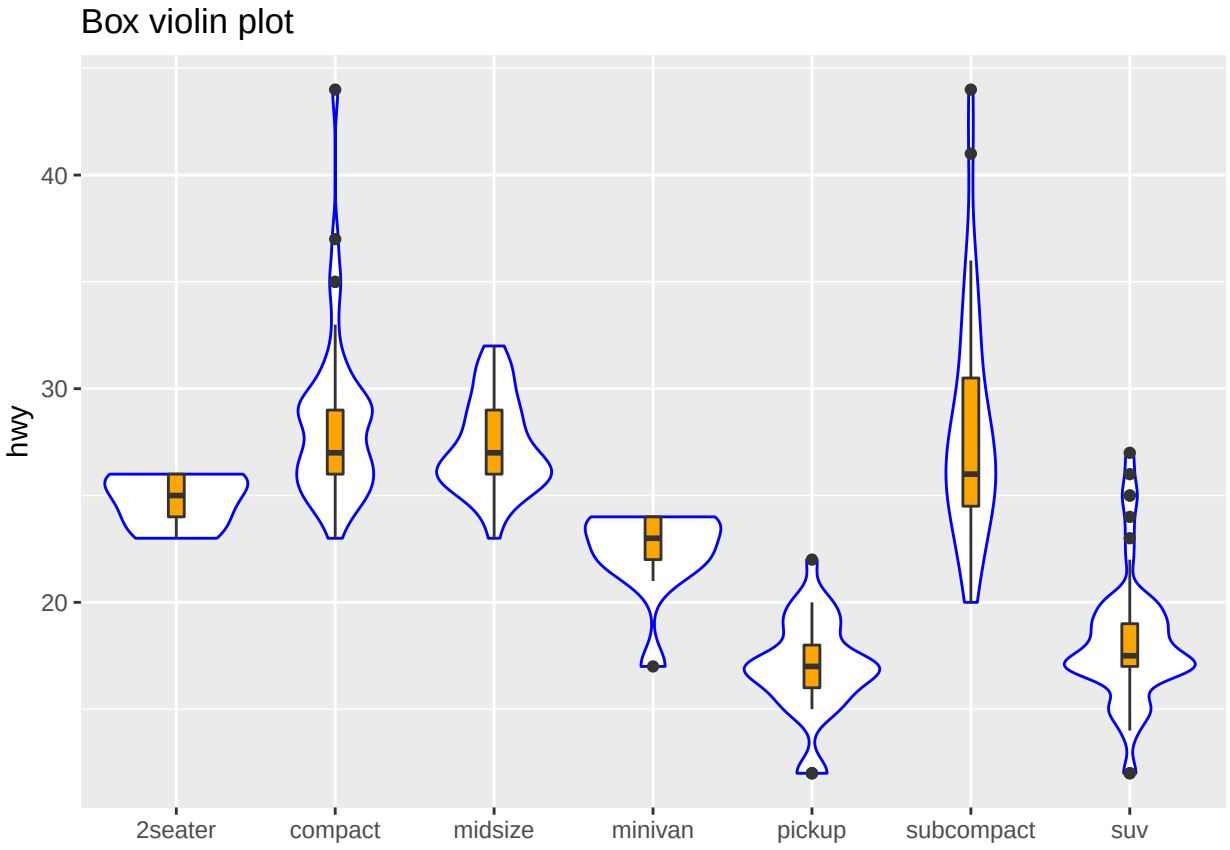
To explore the customize plot, let's first try to change the color in our very last plot.

```
## we can also assign the plot in an object for the later usage
p1 <- ggplot(mpg, aes(x = class, y  = hwy)) +
    geom_violin(color = "blue")  +
    geom_boxplot(width = 0.1, fill = "orange")
p1
```



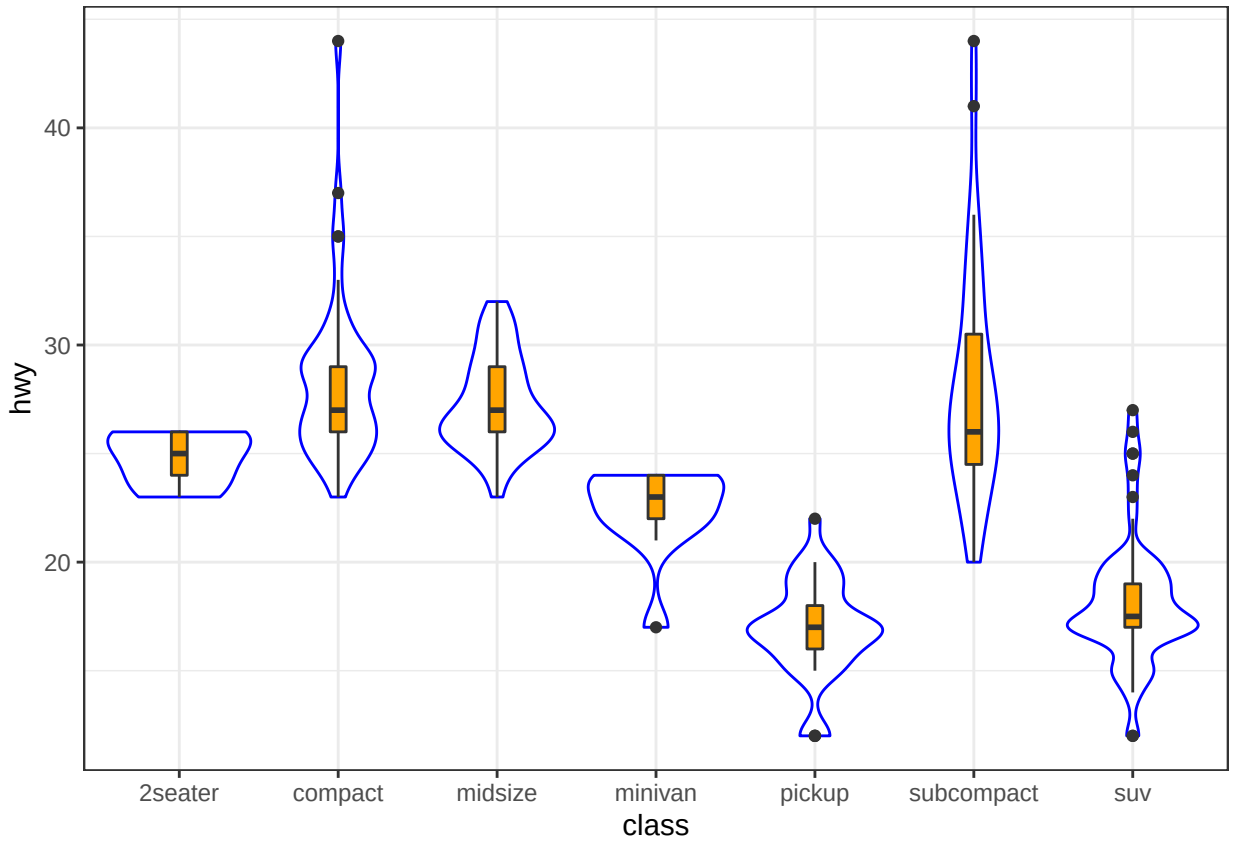
Now let's try to add a title to the plot and remove the label on x-axis.

```
p1 + labs(title = "Box violin plot",
          x = NULL)
```

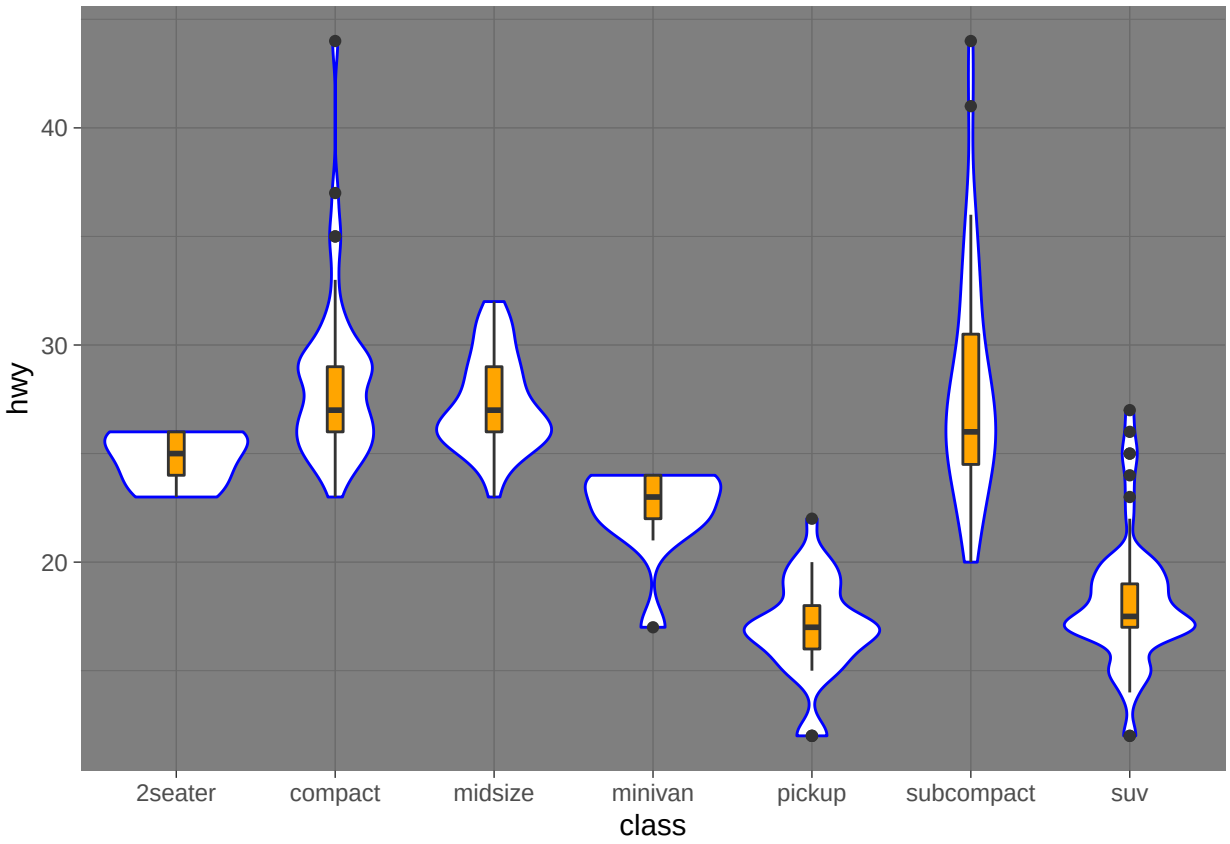


Besides manually change all the parameters and try to make the plot looks nicer, in the `ggplot2` package, they actually offer different themes which allow us to quickly change the style of the plot.

```
p1 + theme_bw()
```




```
p1 + theme_dark()
```



If you are interested in select a nice theme for your future plots, you can check out the following link https://ggplot2.tidyverse.org/reference/ggtheme.html.

Exercise 3

1) Use a bar chart to find out the which `audi model` has the highest frequency in the `mpg` data set.

2) Use a scatter plot to find out whether there is a correlation between `cty` and `hwy`.

3) Please try to answer the question: which manufacture produce the low energy consumption (`cty`) `midsize` cars?