Introduction-to-EDA-in-R

Gian Carlo Sanfuego

2024-03-10

# Exploratory Data Analysis with R

In statistics, exploratory data analysis, EDA, for short, is an approach to analyzing data sets to summarize their main characteristics, often with visual methods. Although, a statistical model can be used or not, but when we talk about EDA, EDA primarily is for seeing what data can tell us beyond the formal modeling or hypothesis testing task.

Exploratory data analysis was promoted by John Tukey to encourage statisticians to explore the data and possibly formulate hypothesis that could lead to new data collection and experiment.

# Task One: Getting Started

In this task, you will learn set and check your current working directory

### 1.1: Set and get the working directory

# setwd(dir = "C:\\Users\\gianc\\OneDrive\\Desktop\\DATA ANALYSIS\\PROJECTS\\R\\INTRODUCTION TO EDA IN R")

### 1.2: Get the working directory

getwd()

## [1] "C:/Users/gianc/OneDrive/Desktop/DATA ANALYSIS/PROJECTS/R/INTRODUCTION TO EDA IN R"

### 1.3: Importing packages needed

library(e1071)  
library(ggplot2)  
library(graphics)  
library(lattice)  
require(stats)

# Task Two: Import data set and explore

In this task, we will import two data sets and explore them

### 2.1: Import the msleep.csv and mpg.csv data sets

df <- read.csv(file = "msleep.csv")  
df1 <- read.csv(file = "mpg.csv")

### 2.2: View and check the dimension of the data sets

# View(df)  
dim(df)

## [1] 83 12

# View(df1)  
dim(df1)

## [1] 234 12

### 2.3: Take a peek at df using the head and tail functions

head(df)

## X.1 X name genus vore order conservation  
## 1 1 1 Cheetah Acinonyx 1 Carnivora 4  
## 2 2 2 Owl monkey Aotus 4 Primates 4  
## 3 3 3 Mountain beaver Aplodontia 2 Rodentia 5  
## 4 4 4 Greater short-tailed shrew Blarina 4 Soricomorpha 4  
## 5 5 5 Cow Bos 2 Artiodactyla 2  
## 6 6 6 Three-toed sloth Bradypus 2 Pilosa 4  
## Total.Sleep.Time sleep\_rem awake brainwt bodywt  
## 1 12.1 1.87541 11.9 0.2815814 50.000  
## 2 17.0 1.80000 7.0 0.0155000 0.480  
## 3 14.4 2.40000 9.6 0.2815814 1.350  
## 4 14.9 2.30000 9.1 0.0002900 0.019  
## 5 4.0 0.70000 20.0 0.4230000 600.000  
## 6 14.4 2.20000 9.6 0.2815814 3.850

tail(df)

## X.1 X name genus vore order conservation  
## 78 78 78 Tenrec Tenrec 4 Afrosoricida 4  
## 79 79 79 Tree shrew Tupaia 4 Scandentia 4  
## 80 80 80 Bottle-nosed dolphin Tursiops 1 Cetacea 4  
## 81 81 81 Genet Genetta 1 Carnivora 4  
## 82 82 82 Arctic fox Vulpes 1 Carnivora 4  
## 83 83 83 Red fox Vulpes 1 Carnivora 4  
## Total.Sleep.Time sleep\_rem awake brainwt bodywt  
## 78 15.6 2.30000 8.4 0.0026000 0.900  
## 79 8.9 2.60000 15.1 0.0025000 0.104  
## 80 5.2 1.87541 18.8 0.2815814 173.330  
## 81 6.3 1.30000 17.7 0.0175000 2.000  
## 82 12.5 1.87541 11.5 0.0445000 3.380  
## 83 9.8 2.40000 14.2 0.0504000 4.230

### 2.4: Check the internal structure of the data frame

str(df)

## 'data.frame': 83 obs. of 12 variables:  
## $ X.1 : int 1 2 3 4 5 6 7 8 9 10 ...  
## $ X : int 1 2 3 4 5 6 7 8 9 10 ...  
## $ name : chr "Cheetah" "Owl monkey" "Mountain beaver" "Greater short-tailed shrew" ...  
## $ genus : chr "Acinonyx" "Aotus" "Aplodontia" "Blarina" ...  
## $ vore : int 1 4 2 4 2 2 1 2 1 2 ...  
## $ order : chr "Carnivora" "Primates" "Rodentia" "Soricomorpha" ...  
## $ conservation : int 4 4 5 4 2 4 6 4 2 4 ...  
## $ Total.Sleep.Time: num 12.1 17 14.4 14.9 4 14.4 8.7 7 10.1 3 ...  
## $ sleep\_rem : num 1.88 1.8 2.4 2.3 0.7 ...  
## $ awake : num 11.9 7 9.6 9.1 20 9.6 15.3 17 13.9 21 ...  
## $ brainwt : num 0.28158 0.0155 0.28158 0.00029 0.423 ...  
## $ bodywt : num 50 0.48 1.35 0.019 600 ...

What we have here are: - the name is a factor with 83 levels; - the genus is a factor; - Vore is an integer; - Order is a factor, that is a categorical variable with 19 levels; - we have total sleep time; - we have sleep\_rem; - awake time is a numeric variable; - brain weight is a numeric variable; - body width is also a numeric variable.

So, that gives us an idea of the structure of the different variables we have.

### 2.5: Count missing values in the variables

sum(is.na(df))

## [1] 0

It says the sum is zero which means that there is no missing value.

### 2.6: Check the column names for the df data frame

names(df)

## [1] "X.1" "X" "name" "genus"   
## [5] "vore" "order" "conservation" "Total.Sleep.Time"  
## [9] "sleep\_rem" "awake" "brainwt" "bodywt"

That gives us the names of different variables in this df data frame. But you’ll notice that there is this ‘X.1’ and X variable, which is just like a number. We don’t really need that. So, we want to first drop the first two columns of this df data frame.

### 2.7: Drop the first two columns of df

df <- df[, -c(1,2)]  
# The comma after the bracket is like "I'm saying that I want all the rows". Then the -c is like I'm saying "but I want to take out the first and the second column."

Let’s confirm if these two variables have been dropped.

dim(df)

## [1] 83 10

# Task Three: Plot a categorical Variable

In this task, we will learn how to plot a categorical variable. A categorical variable has values that you can put into a countable number of distinct groups based on their characteristics. For example, gender is a categorical variable or a factor. And this has two categories; it’s either a male or female on a more generic note.

### 3.1: Let us check the internal structure the df1 data frame

str(df1)

## 'data.frame': 234 obs. of 12 variables:  
## $ X : int 1 2 3 4 5 6 7 8 9 10 ...  
## $ manufacturer: chr "audi" "audi" "audi" "audi" ...  
## $ model : chr "a4" "a4" "a4" "a4" ...  
## $ displ : num 1.8 1.8 2 2 2.8 2.8 3.1 1.8 1.8 2 ...  
## $ year : int 1999 1999 2008 2008 1999 1999 2008 1999 1999 2008 ...  
## $ cyl : int 4 4 4 4 6 6 6 4 4 4 ...  
## $ trans : chr "auto(l5)" "manual(m5)" "manual(m6)" "auto(av)" ...  
## $ drv : chr "f" "f" "f" "f" ...  
## $ cty : int 18 21 20 21 16 18 18 18 16 20 ...  
## $ hwy : int 29 29 31 30 26 26 27 26 25 28 ...  
## $ fl : chr "p" "p" "p" "p" ...  
## $ class : chr "compact" "compact" "compact" "compact" ...

We can see the different variables. We can see this “class” variable which is the class of the car. It has seven levels. We’ll be using that for this particular task. We will use this class variable from the df1, which is a categorical or qualitative variable, but first we want to assign this to a name that will be easy for us to move on.

### 3.2: Assign class from df1 to a new variable called c1

c1 <- df1$class  
c1

## [1] "compact" "compact" "compact" "compact" "compact"   
## [6] "compact" "compact" "compact" "compact" "compact"   
## [11] "compact" "compact" "compact" "compact" "compact"   
## [16] "midsize" "midsize" "midsize" "suv" "suv"   
## [21] "suv" "suv" "suv" "2seater" "2seater"   
## [26] "2seater" "2seater" "2seater" "suv" "suv"   
## [31] "suv" "suv" "midsize" "midsize" "midsize"   
## [36] "midsize" "midsize" "minivan" "minivan" "minivan"   
## [41] "minivan" "minivan" "minivan" "minivan" "minivan"   
## [46] "minivan" "minivan" "minivan" "pickup" "pickup"   
## [51] "pickup" "pickup" "pickup" "pickup" "pickup"   
## [56] "pickup" "pickup" "suv" "suv" "suv"   
## [61] "suv" "suv" "suv" "suv" "pickup"   
## [66] "pickup" "pickup" "pickup" "pickup" "pickup"   
## [71] "pickup" "pickup" "pickup" "pickup" "suv"   
## [76] "suv" "suv" "suv" "suv" "suv"   
## [81] "suv" "suv" "suv" "pickup" "pickup"   
## [86] "pickup" "pickup" "pickup" "pickup" "pickup"   
## [91] "subcompact" "subcompact" "subcompact" "subcompact" "subcompact"  
## [96] "subcompact" "subcompact" "subcompact" "subcompact" "subcompact"  
## [101] "subcompact" "subcompact" "subcompact" "subcompact" "subcompact"  
## [106] "subcompact" "subcompact" "subcompact" "midsize" "midsize"   
## [111] "midsize" "midsize" "midsize" "midsize" "midsize"   
## [116] "subcompact" "subcompact" "subcompact" "subcompact" "subcompact"  
## [121] "subcompact" "subcompact" "suv" "suv" "suv"   
## [126] "suv" "suv" "suv" "suv" "suv"   
## [131] "suv" "suv" "suv" "suv" "suv"   
## [136] "suv" "suv" "suv" "suv" "suv"   
## [141] "suv" "compact" "compact" "midsize" "midsize"   
## [146] "midsize" "midsize" "midsize" "midsize" "midsize"   
## [151] "suv" "suv" "suv" "suv" "midsize"   
## [156] "midsize" "midsize" "midsize" "midsize" "suv"   
## [161] "suv" "suv" "suv" "suv" "suv"   
## [166] "subcompact" "subcompact" "subcompact" "subcompact" "compact"   
## [171] "compact" "compact" "compact" "suv" "suv"   
## [176] "suv" "suv" "suv" "suv" "midsize"   
## [181] "midsize" "midsize" "midsize" "midsize" "midsize"   
## [186] "midsize" "compact" "compact" "compact" "compact"   
## [191] "compact" "compact" "compact" "compact" "compact"   
## [196] "compact" "compact" "compact" "suv" "suv"   
## [201] "pickup" "pickup" "pickup" "pickup" "pickup"   
## [206] "pickup" "pickup" "compact" "compact" "compact"   
## [211] "compact" "compact" "compact" "compact" "compact"   
## [216] "compact" "compact" "compact" "compact" "compact"   
## [221] "compact" "subcompact" "subcompact" "subcompact" "subcompact"  
## [226] "subcompact" "subcompact" "midsize" "midsize" "midsize"   
## [231] "midsize" "midsize" "midsize" "midsize"

So this gives us the different seven levels, 2 seaters, compact, midsize, minivan, pick up, sub compact and suv. Next is we want to get the frequency of the feature, that is the class feature.

### 3.3: Get the frequency of the feature “class”

c1\_freq <- table(c1)  
c1\_freq

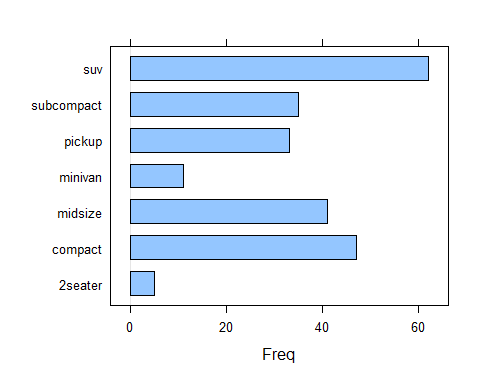
## c1  
## 2seater compact midsize minivan pickup subcompact suv   
## 5 47 41 11 33 35 62

So, we can see, 2 seater has five of them, compact 47, and so on.

### 3.4: Plot a categorical variable on a bar chart

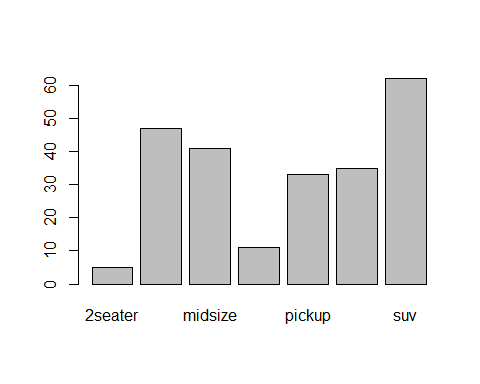
### Using the plot function

plot(barchart(c1))



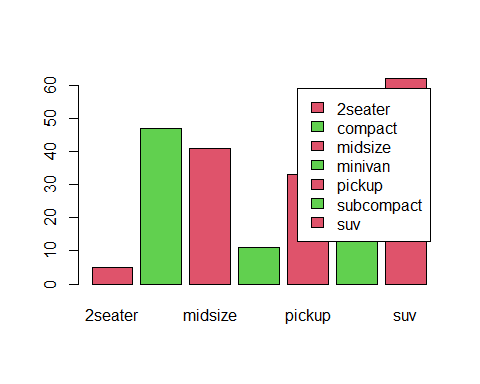
### Plotting tabular data

barplot(c1\_freq, beside = TRUE)



### Add legend and colour

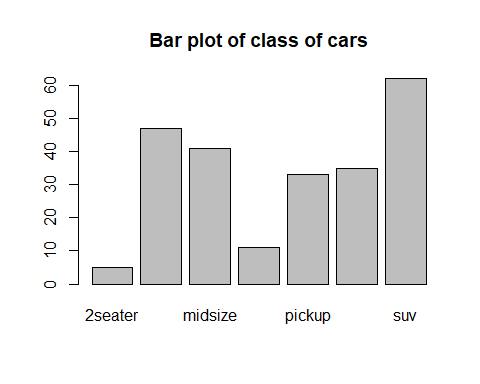
barplot(c1\_freq, beside = TRUE,  
 legend.text = TRUE,  
 col=c(2,3)) # I want to give it two colors, si I'll put my c variable, 2 comma 3.



We can see that this gives different colors, because we specified two colors there. We can also play around with colors by changing it. We see, when we specify that legend.text, what we are trying to tell R is that, make the boxes to be filled with specific colors and let it appear beside the legend text.

### Add a title

barplot(c1\_freq, main = "Bar plot of class of cars")

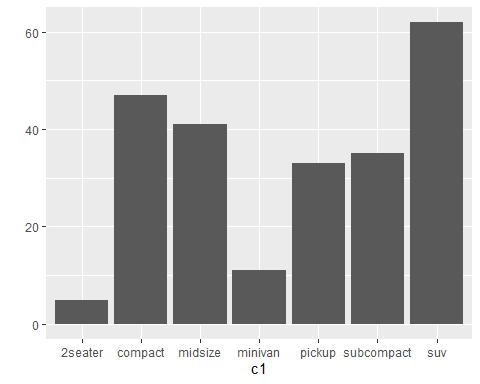


The next function we want to use is the qplot function. The qplot function from the ggplot2, which can be used to provide high quality graphs with little effort.

### Using the qplot function

qplot(c1, data = df1)

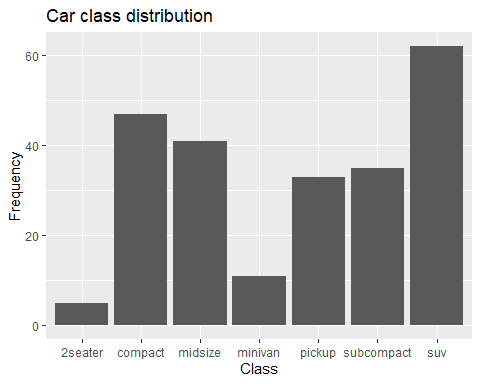
## Warning: `qplot()` was deprecated in ggplot2 3.4.0.  
## This warning is displayed once every 8 hours.  
## Call `lifecycle::last\_lifecycle\_warnings()` to see where this warning was  
## generated.



We can see that this renders a better plot. You can see the different car classes and their frequencies represented by bars.

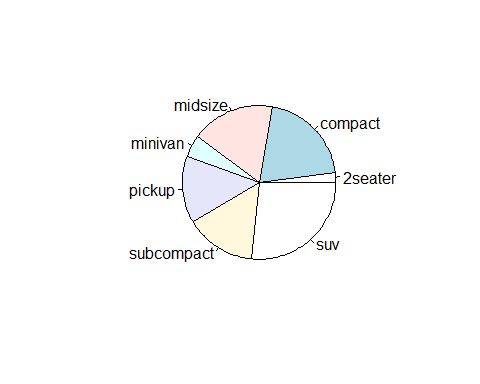
### Add a title and change label names

qplot(c1, data = df1,  
 main = "Car class distribution",  
 xlab = "Class",  
 ylab = "Frequency")



### 3.5: Plot a pie chart

pie(c1\_freq)



We can see the different classes are now put in a circle using different degrees.

# Task Four: Plotting a Numeric Variable

In this task, we will plot different plots for a quantitative variable. A numerical variable or continuous variable is one that may take any value within a finite or infinite interval. When we talk about the height of a person, the weight of a person, temperature, the salary, blood glucose, number of cars that come in, these are numeric variables.

In this task, we will consider some interesting plots for univariate, that is one numeric variable. Before we go on to the plots, let us check some descriptive statistics for the total sleep time variable from the df data frame.

# Plots for univariate variables

### 4.1: Descriptive statistics in R

### Assign a single variable to a new variable

sleeptime <- df$Total.Sleep.Time

Let us now check some descriptives for it.

summary(sleeptime)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 1.90 7.85 10.10 10.43 13.75 19.90

fivenum(sleeptime)

## [1] 1.90 7.85 10.10 13.75 19.90

var(sleeptime)

## [1] 19.80568

sd(sleeptime)

## [1] 4.450357

That gives us the minimum value, the first quartile, the median, the mean, the third quartile and the maximum. We can also go on to check the five number sum.

So, this also gives us the five number sum, which is similar to the summary, except that we don’t have the inclusion of the mean for the fivenum sum. We can also check the variance of sleeptime.

The variance is 19.80. That’s to check the variability. Let’s check also the standard deviation(sd). And sd, is like the square root of the variance used to check the distance between each of the data points from the mean.

The standard deviation is 4.45. So, we’ve been able to check some descriptive of our sleeptime variable.

### 4.2: Stem and Leaf

A stem and leaf display or the stem and leaf plot is a device for presenting quantitative or numerical data in a graphical format similar to the histogram. This helps to assist in visualizing the shape of a distribution. This evolved from Auther Bowley’s work in the early 1990’s, and are useful for exploratory data analysis.

### Use stem() command

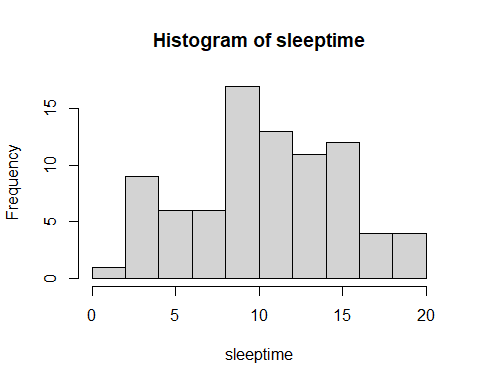
stem(sleeptime)

##   
## The decimal point is at the |  
##   
## 0 | 9  
## 2 | 79013589  
## 4 | 0423346  
## 6 | 23307  
## 8 | 03446779114456788  
## 10 | 01113346900135  
## 12 | 15555880578  
## 14 | 234456996889  
## 16 | 604  
## 18 | 01479

If you look at it, there’s this kind of shape that these figures may form. We can also read more on the sleep– on the stem and leaf display, but we can easily project the shape of the distribution and what we have here will be similar to the histogram that we will plot next.

### 4.3: Histogram

hist(sleeptime)



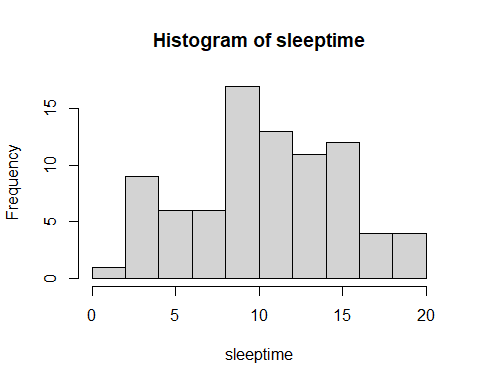
We can see that the histogram have been plotted. Although this looks very basic, we’ll still use some more interesting functions to represent better histogram here.

We know that a histogram is a graphical display of data using bars of different heights. In a histogram each bar groups numbers into ranges. A histogram displays the shape and the spread of a continuous sample data. It was first introduced by the man Karl Pearson, who was an English mathematician and Biostatistician. He has been credited with establishing the field of mathematical statistics.

So, from histogram we have here, we will see that although the sleep time variable is not completely normally distributed but it’s roughly or approximately normally distributed. Let’s see how we can set the bin or the breaks because a basic histogram has a predefined number of bins or that we can set the number of bins that is the number of bars that we want.

### Set number of bins to 10

hist(sleeptime, breaks = 10)  
hist(sleeptime, 10)

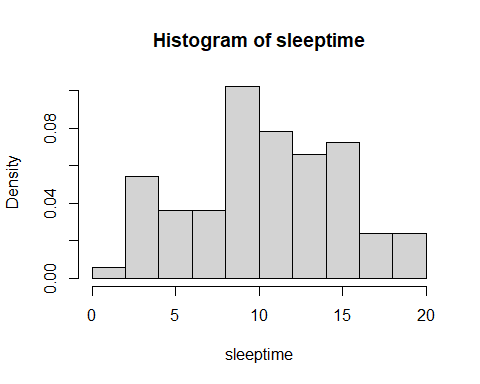


Now we see 10 bars in our histogram above.

It is also possible to use instead of frequency on the y-axis like we have here, we can also use a proportion or probability.

### Using proportions or probabilities

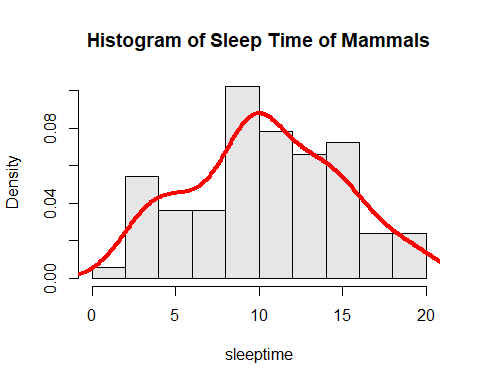
hist(sleeptime, probability = TRUE)



You will see that the frequency in y-axis has changed to density.

### Add a colour and a title. Superimpose a normal curve

hist(sleeptime, probability = TRUE,  
 col = gray(0.9),  
 main = "Histogram of Sleep Time of Mammals") # The 0.9 here is just tell us the opacity, how deep should the grey # look like. Then the main is just to give the title  
lines(density(sleeptime), col = "red", lw = 4)



You can see that the title is Histogram of Sleep Time of mammals. You can also change the x lab if you want. The grey color has been specified.

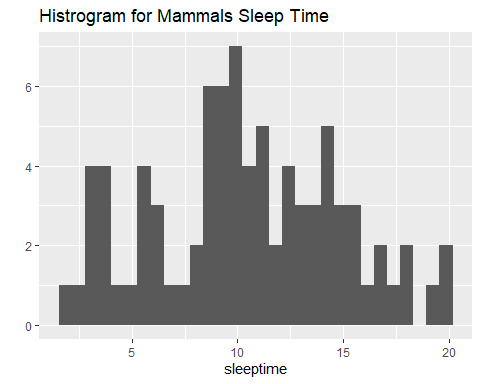
Now let’s try to superimpose a normal curve to better see the shape of the variable.

And we will see that there’s this red line here. And we see that the density line has been, the normal curve, has been plotted on top of the histogram and we can see that this looks like a normal distribution, although not entirely normal. Next let’s try to use the qplot function.

### Using the qplot function

qplot(sleeptime, data = df,  
 main = "Histrogram for Mammals Sleep Time")

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



Here we can see the same chart but with different rendering on the bins.

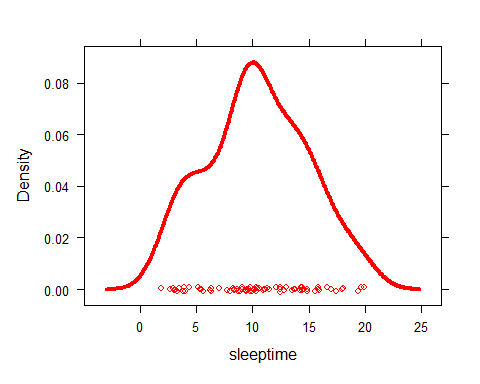
Next, let’s consider the density plot. We already know that a density plot visualizes the distribution of data over a continuous interval or time period.

This chart is a variation of the histogram. It has an advantage that help us better look at the distribution shape, and this will not be affected by the number of bins which is the bars used in a typical histogram.

### 4.4: Density plot

### Use the densityplot function

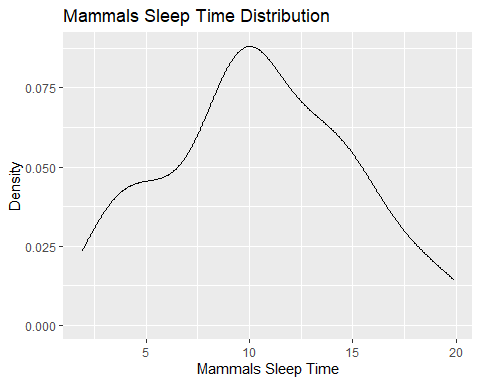
densityplot(sleeptime, lw = 4, col = "red")



Here we will just see only density plot. Now, let’s use the qplot function to also render the destiny plot.

### Using the qplot function

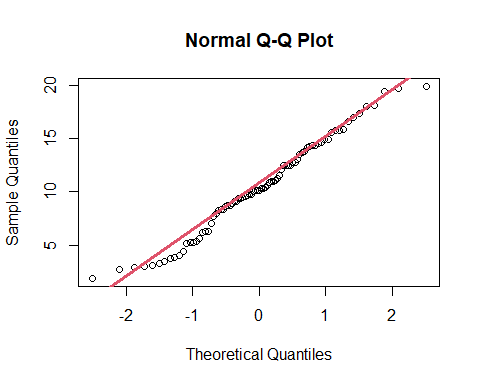
qplot(sleeptime, data = df, geom = "density",  
 main = "Mammals Sleep Time Distribution",  
 xlab = "Mammals Sleep Time",  
 ylab = "Density") # If I just use qplot without # specifying the geometry for a numeric variable, it # plots an histogram only.



And this renders well. We can see the x label, y label, and then the title correctly given. Before we go on to the box plot, which is the last plot for this task, let us look at some density related plot. The qqplot.

### 4.5: Density related plots

qqnorm(sleeptime)  
qqline(sleeptime, col = 2, lw = 3)



# qqplot means "quantile quantile plot". What it does is that it kind of plots the numeric variable on a line, # and this shows the distribution.

And we will see now that this looks normally distributed. For a typical normal distribution, we see all the points from -2 (x-axis) will be directly on the red line. But because we have little of the points in -1 dispersed a little from the line, then it looks like it is approximately normal. But a typical normal distribution will have all the points on the red line.

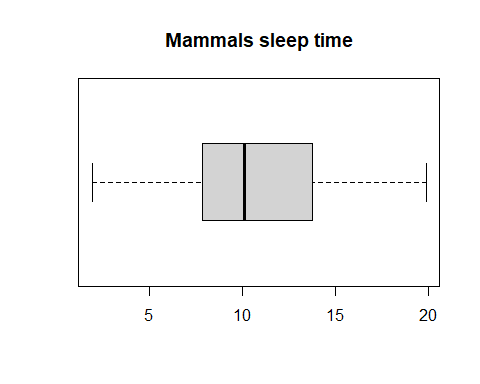
The box plot is another plot for numeric variable. According to Wikipedia, in descriptive statistics, a box plot is a method for graphically depicting groups of numerical data through their quantiles. Box plot may also have lines extending from the box indicating the variability outside the upper and lower quartiles, hence the term box and whisker plots or box and whisker diagram.

The box and whisker plot was first introduced in 1970 by John Tukey.

Now we will plot two box plots for mammal sleep time and mammal’s awake time. We’ll also help to see the distribution of the variables.

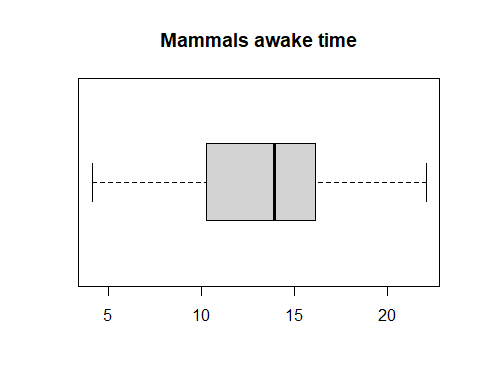
### 4.6: Box plot

boxplot(sleeptime, main = "Mammals sleep time", horizontal = TRUE)



In the sleeptime boxplot, we will see that the line inside the box is not in the middle. This means that it is kind of skewed a bit. The reason why we specified the horizontal to be true, is so that it will present it in horizontal format.

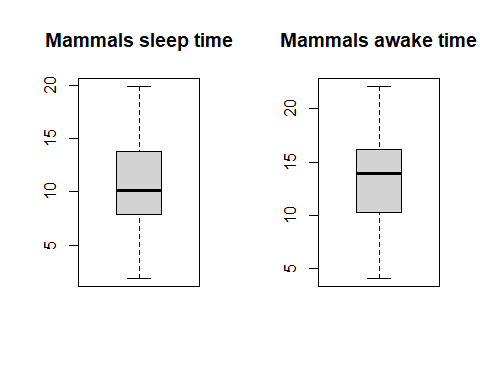
boxplot(df$awake, main = "Mammals awake time", horizontal = TRUE)



Lastly, we want to put these two plots, these two different plots on the same page- like to combine plots. R makes it easy to combine multiple plots into one over all graph. It does this using the par function. We can include the option of mfrow to specify the number of columns we want and number of rows. Let’s see how that works here.

### Plotting multiple plots

par(mfrow = c(1,2))  
boxplot(sleeptime, main = "Mammals sleep time")  
boxplot(df$awake, main = "Mammals awake time")



Notice, both distributions are skewed.

And we can see that the two plots are now rendered to one. We can easily see that this variable sleep time and awake time are skewed. The sleeptime is skewed to the left. The awaketime is skewed to the right.

They are not normally distributed because for example, the line inside the box in the sleeptime, which represents the median, is not at the center of the box.

Ang lastly, there are no visible outliers for these two variables.

### 4.7: Quick clue on resetting the par

If we do not reset the “par” to (1,1), It will keep every plot we make on one page, two graphs on one page. So, we can use “par(mfrow=c(1,1))” or “dev.off()” to just take it off. However, if we use the dev.off, it will take off all the plots we have made before. Thus, for this case, we’ll just reset the par to (1,1). As we move on, we’ll see on how to save or export our plots.

par(mfrow=c(1,1))

or

dev.off()

## null device   
## 1

# Task Five: Plotting a Numerical and Categorical Variable

In this task, we will continue to learn how to perform EDA in R by plotting a numeric and categorical variable.

We will use the qplot function majorly to visualize a numeric and a categorical variable.

So the first plot, we want to consider is the dot plot, which is a statistical chart that uses circles to plot different points. This is also similar to the histogram.

We’ll be performing this on our df1 variable.

### 5.1: Dot plot

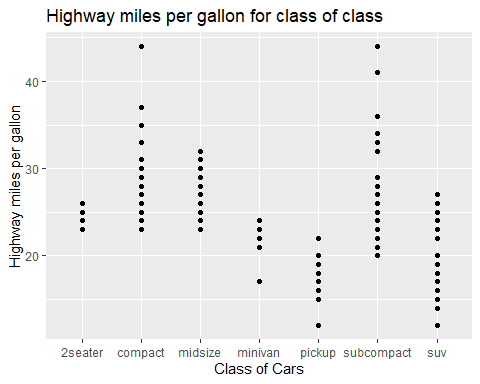
Let’s plot the class, which is a categorical variable. But first, let’s see the structure of df1.

str(df1)

## 'data.frame': 234 obs. of 12 variables:  
## $ X : int 1 2 3 4 5 6 7 8 9 10 ...  
## $ manufacturer: chr "audi" "audi" "audi" "audi" ...  
## $ model : chr "a4" "a4" "a4" "a4" ...  
## $ displ : num 1.8 1.8 2 2 2.8 2.8 3.1 1.8 1.8 2 ...  
## $ year : int 1999 1999 2008 2008 1999 1999 2008 1999 1999 2008 ...  
## $ cyl : int 4 4 4 4 6 6 6 4 4 4 ...  
## $ trans : chr "auto(l5)" "manual(m5)" "manual(m6)" "auto(av)" ...  
## $ drv : chr "f" "f" "f" "f" ...  
## $ cty : int 18 21 20 21 16 18 18 18 16 20 ...  
## $ hwy : int 29 29 31 30 26 26 27 26 25 28 ...  
## $ fl : chr "p" "p" "p" "p" ...  
## $ class : chr "compact" "compact" "compact" "compact" ...

Here we see that there is this hwy, which is the highway miles per gallon for each of these cars. So, for each of the class, like different classes of cars, we want to see the highway miles per gallon that they that they use.

qplot(class, hwy, data = df1,  
 main = "Highway miles per gallon for class of class",  
 xlab = "Class of Cars",  
 ylab = "Highway miles per gallon")

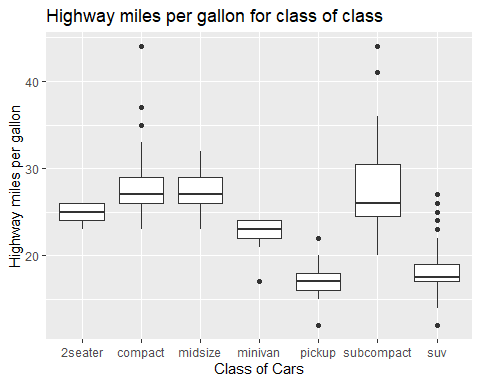


This is a boxplot that plotted in dots. wW can see for the different cars class, we can see the highway miles gallon. The sub compact looks to be higher point.

And now, let’s check the box plots.

### 5.2: Box plot

qplot(class, hwy, data = df1, geom = "boxplot",  
 main = "Highway miles per gallon for class of class",  
 xlab = "Class of Cars",  
 ylab = "Highway miles per gallon")

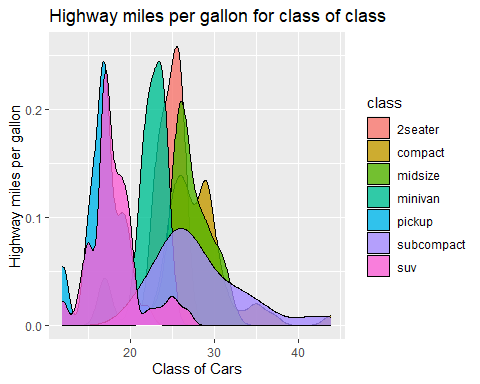


Now, we can see that this gives a box plot for the different class of cars. For compact, there are some outliers. Same with suv there are also some outliers,

### 5.3: Colored Density plot

Lastly, we want to plot a color density plot for the different classes of cars. Don’t forget, we are still doing numeric and categorical.

den <- qplot(hwy, fill=class, data = df1, geom = "density", alpha = I(0.8),   
 main = "Highway miles per gallon for class of class",  
 xlab = "Class of Cars",  
 ylab = "Highway miles per gallon")  
# The alpha is just to say the opacity of the different plots # that we have. If I put zero there, alpha will return a very transparent plot. # If I put one, it will be completely opaque. # So, we will put capital later I then we can specify 0.8 just to show opacity.  
  
den



Now we see that for 2 seater, that’s why we filled it with the class, we can see the density plot for the different classes that we have here.

# Task Six: Plotting two variables

In this task, we will learn how to plot two categorical or two numerical variables in R.

So for the categorical variables we will be using the drv with three levels 4, f, and r– that is for forward and rear wheel and the rest. Then class, which is of seven levels. Then for the numerical variable, we will be using cty and hwy. cty is the city miles per gallon. hwy is the highway miles per gallon.

So let’s start with the categorical variable. We want to check the frequency first.

# Plotting two categorical variables

### 6.1: Check the frequency of two categorical variables

table(df1$class, df1$drv)

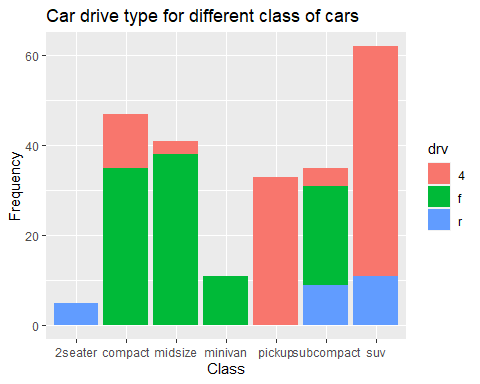
##   
## 4 f r  
## 2seater 0 0 5  
## compact 12 35 0  
## midsize 3 38 0  
## minivan 0 11 0  
## pickup 33 0 0  
## subcompact 4 22 9  
## suv 51 0 11

The table shows that for 2 seaters: - 4 is zero, - f is zero, - r is five. So the table shows the frequency for each of the classes and the drive type, whether it is forward wheel or rear wheel.

Now, let’s see how to make a colored bar plot using the qplot function for two categorical variables.

### 6.2: Colored bar plot for two categorical variables

qplot(class, fill = drv, data = df1, geom = "bar",  
 main = "Car drive type for different class of cars",  
 xlab = "Class",   
 ylab = "Frequency")



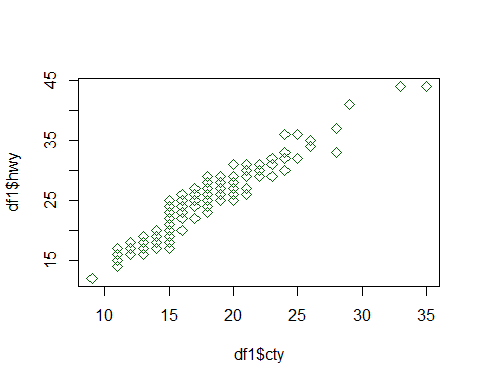
So, we can see for the different drive type, we have the frequency. We remember that 4 and f for 2 seater were zero. So, we can see that, that’s why only the blue shows that’s rear to be five. We can see that only the subcompact had the complete class frequency for each. So sub compact has rear which is in blue. We can see the color for the number 4 on the top of subcompact. Another observation is Minivan does not have front wheel and rear wheel. It has just 4 which means it has just forward- forward wheel as green. So that makes sense, right? So we’ve seen how to do two categorical variables. Let’s do for two numeric variables using the scatter plot.

# Plotting two numerical variables

### 6.3: Scatter plot using the plot function

Let us add something to this particular code here, which we call pch. The pch is used to change the nature of the plots– that’s the points.

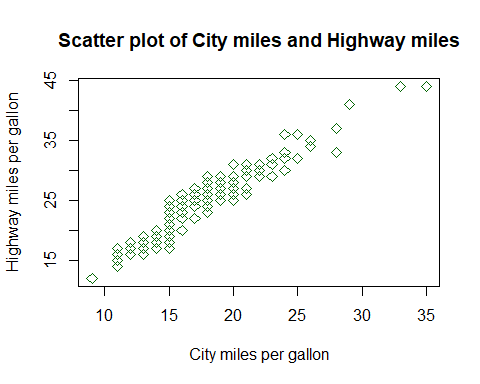
plot(df1$cty, df1$hwy,  
 col = "darkgreen", pch = 5)



# So if I say pch and put five or nine, I'm trying to change circle to something else.

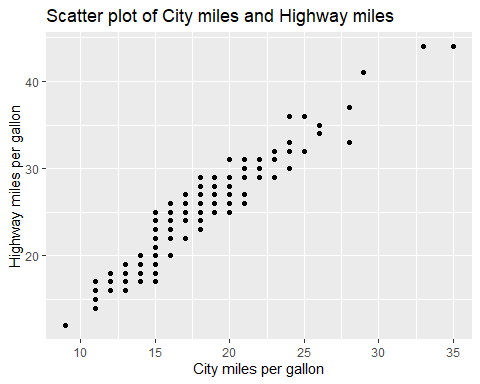
### 6.4: Add a title, and give appropriate label names

plot(df1$cty, df1$hwy,  
 col = "darkgreen", pch = 5,  
 main = "Scatter plot of City miles and Highway miles",  
 xlab = "City miles per gallon",  
 ylab = "Highway miles per gallon")



### 6.5: Scatter plot using the qplot function

qplot(cty, hwy, data = df1,  
 main = "Scatter plot of City miles and Highway miles",  
 xlab = "City miles per gallon",  
 ylab = "Highway miles per gallon")



In the code above, you notice we removed the color and the pch because that is not compatible with the qplot. So we can see using the qplot, it renders the same plot- the scatter plot, but in a more beautiful manner.

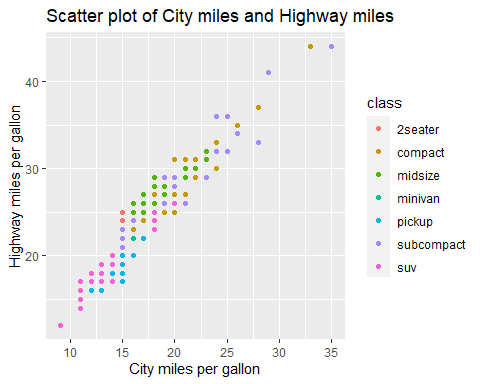
# Task Seven: Plotting three variables - Wrap up

In this task, we will learn how to plot three variables and how to save a plot in R

### 7.1: Colored Dot plot

Now we want to plot a colored dot plot or scatter plot using the class to– filling with the class of the cars. But this time I want to add color to it using the class variable, which is now the third variable.

qplot(cty, hwy, color = class, data = df1,  
 main = "Scatter plot of City miles and Highway miles",  
 xlab = "City miles per gallon",  
 ylab = "Highway miles per gallon")



Here we see that for the different classes, we’ve used different colors to represent different classes. We can now see which one particularly is the class. It still gives a linear trend or linear relationship but now, we’re checking the different classes of cars based on their highway miles per gallon and their city miles per gallon.

That’s how to check plot three variables in R.

### 7.2: Save last plot as an image

ggsave("Colored-dot-plot.jpeg", width = 9, height = 6)

Now, if we go to our working directory, the saved image can be found there.

### 7.3: Save any plot from global environment as an image

ggsave("Colored-dot-plot2.jpeg", den, width = 9, height = 6)

If we go again to our working directory, the saved image can be found there.