Session Summary

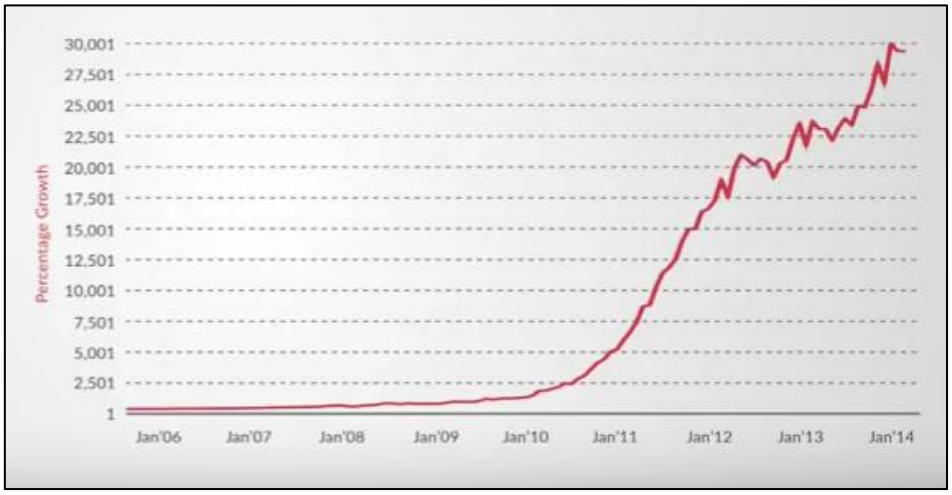
Data Visualisation - How to Make Data Presentable!

One picture is worth a thousand words. This is the power of visualisation. What message cannot be conveyed through a large set of texts and tabular data can easily be presented through visuals. Graphs can be used to understand complex information. In this module, you saw how different kinds of graphs and charts are used to represent different kinds of information and each type have their own utility and limitations. You also saw to plot these graphs using the R base package and also by using a special package called ggplot2.

Different Types of Charts/Graphs

In this session, we discussed the different types of graphs and their major area of applications:

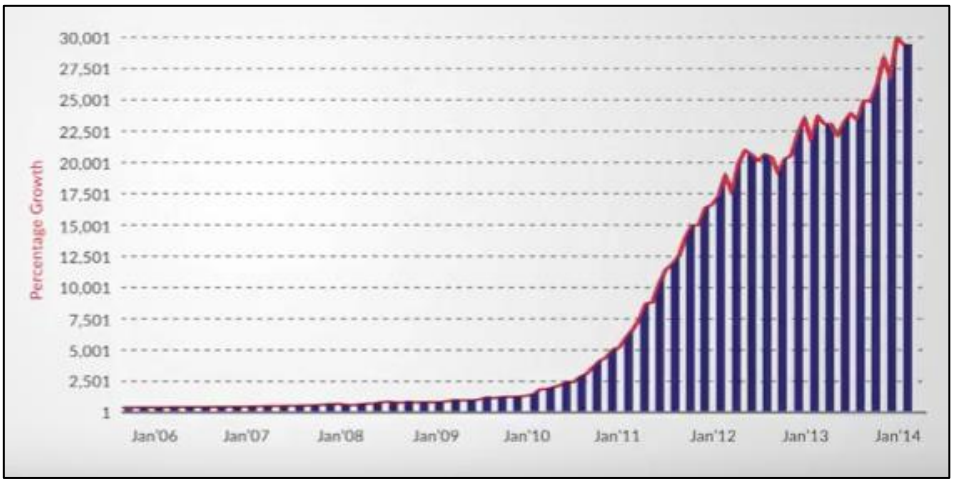
1. Line Charts



A line chart plots the points as a series of points connected by straight lines. Line charts are extensively used to display time series data.

Line graphs show the trends of data and can also be used to extrapolate the data. They are also very useful for plotting multiple variables against the same independent variable to show how all these variables vary. But remember that the scales of all these variables should be comparable.

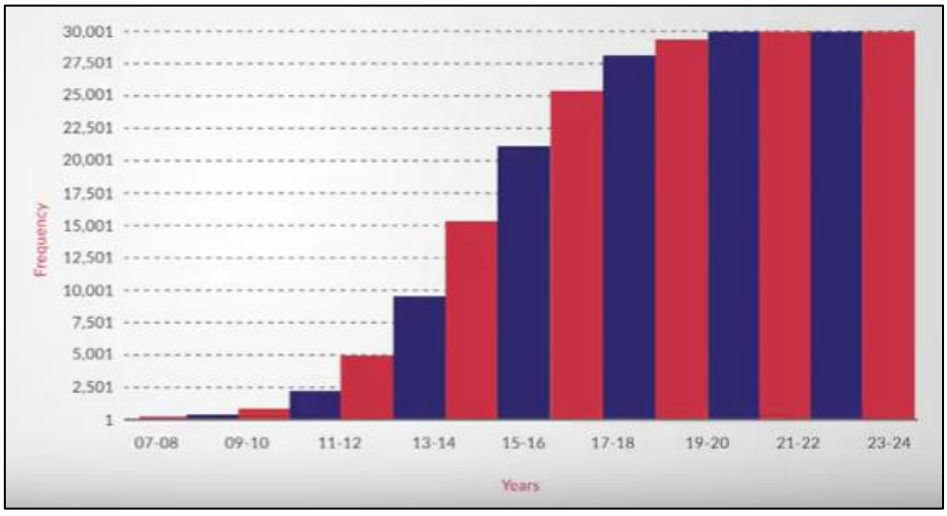
2. Bar Charts



A bar chart shows how the dependent variables vary with changes in categorical independent variables.

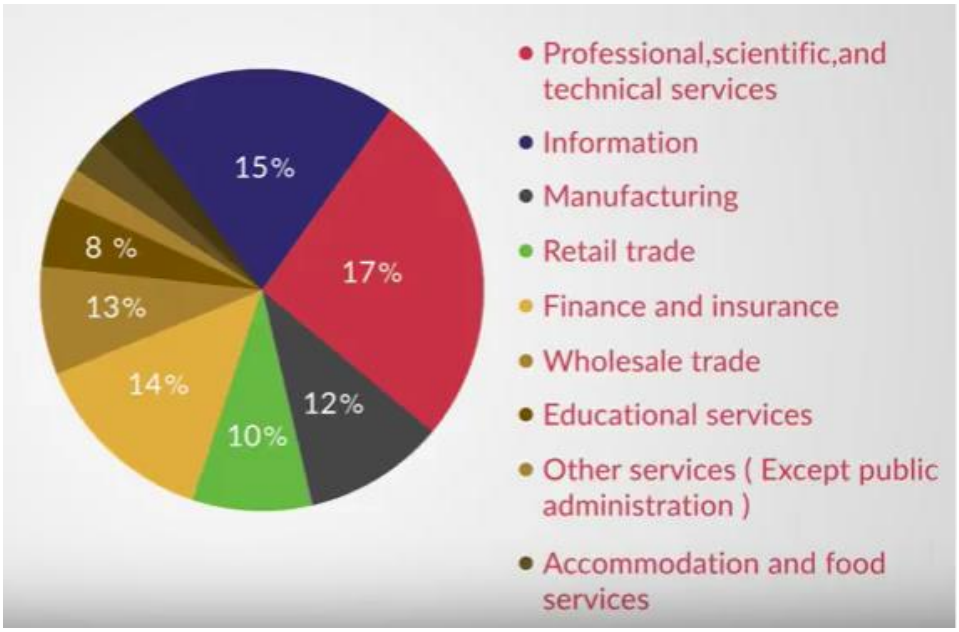
The values of the dependent variables are plotted as bars of different heights of equal thickness. The horizontal axis plots the independent categorical variable.

3. Histograms



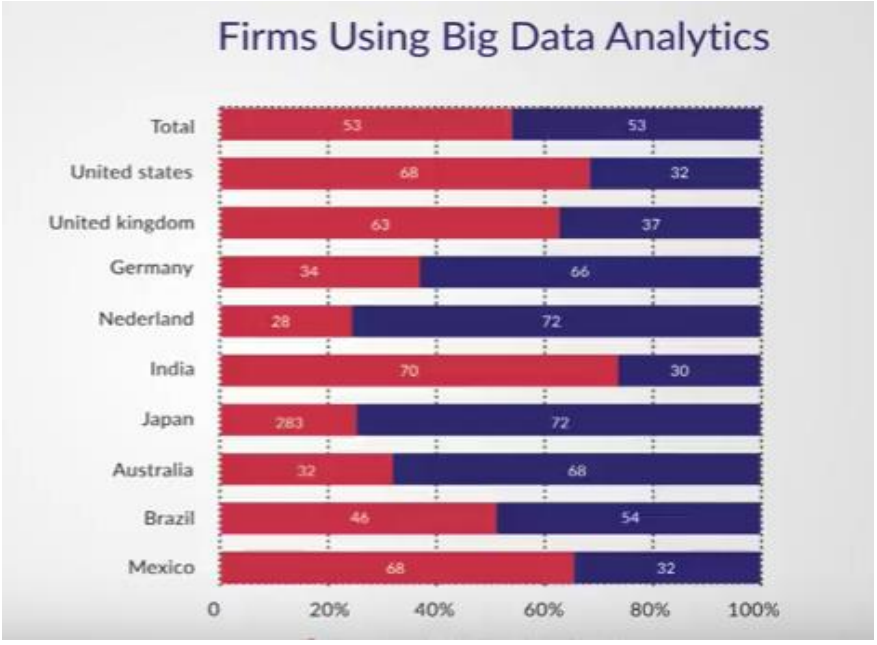
A frequency histogram shows the frequency distribution of a continuous variable. The data is split into intervals (or bins) and the number of times the data appears in the specific bin corresponds to the height of the vertical bar.

4. Pie Charts



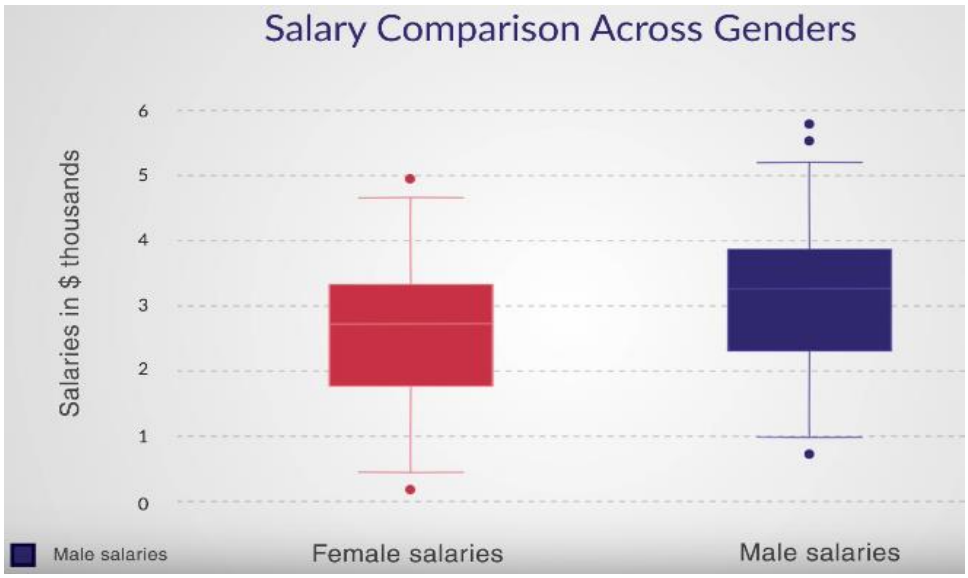
A pie chart is useful for representing proportions of data out of a whole. The size of the slices represent the percentage, with higher percentage categories representing larger slices of the pie.

5. Stack Bar Charts



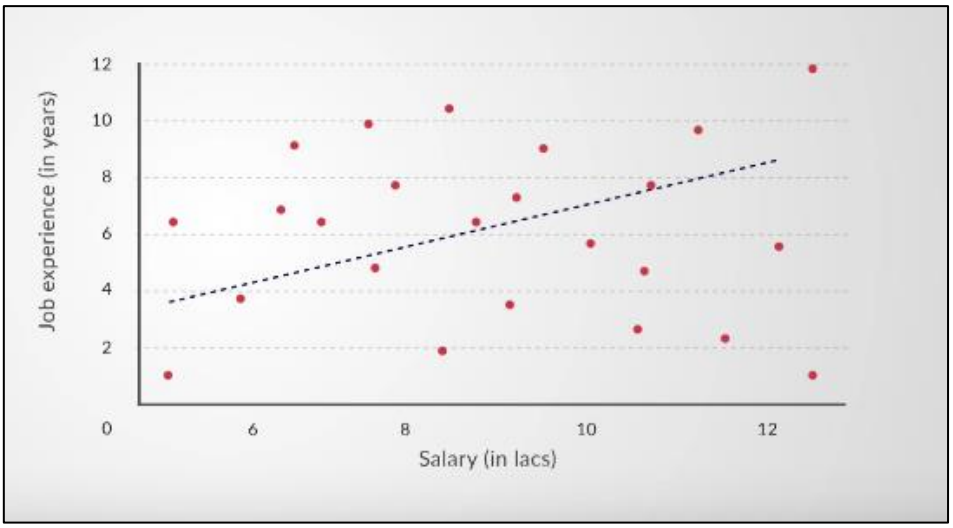
A stacked bar chart is also a good method to compare proportions. Each bar is of the same height, with various segments of the bar showing the share of various categories. It can be used to compare the proportions of different categories across another categorical variable.

6. Box Plots



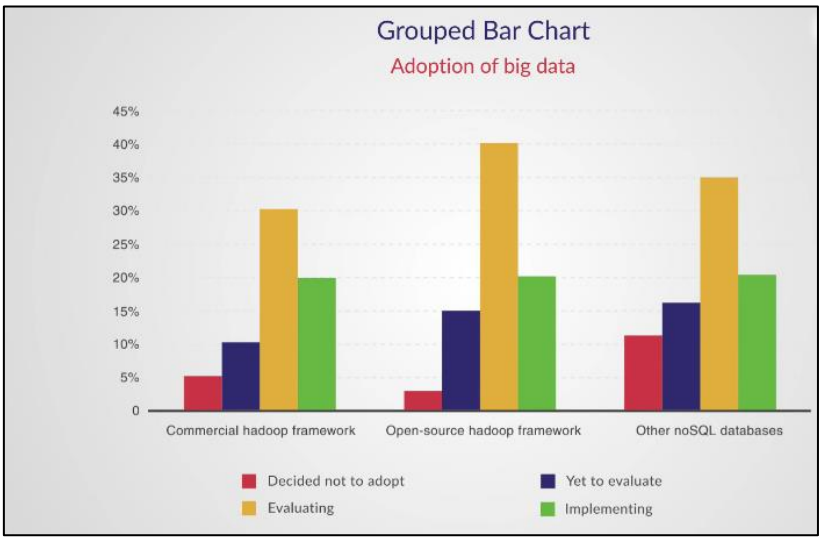
Box and whisker plot shows the spread of the data. It demonstrates the range of data along with the median. It is useful for looking at the distribution of data across various categories.

7. Scatter Plots



A scatter plot shows how one continuous dependent variable varies with another continuous independent variable. It plots points, where the x axis shows the value of independent variable, while the y axis shows the dependent variable. The position of each point represents its x and y value. These plots are very useful in studying trends and distribution in data, identifying clusters, outliers and even predicting values based on trends.

8. Grouped Bar Charts



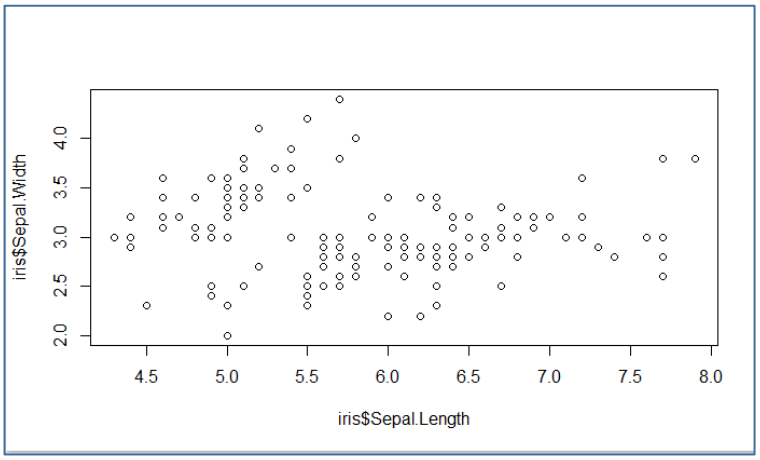
Grouped bar charts show information about different subgroups for a main category. Each subgroup is shown by a separate bar, which is coloured differently for each subgroup.

Basic Plotting: The Base Package

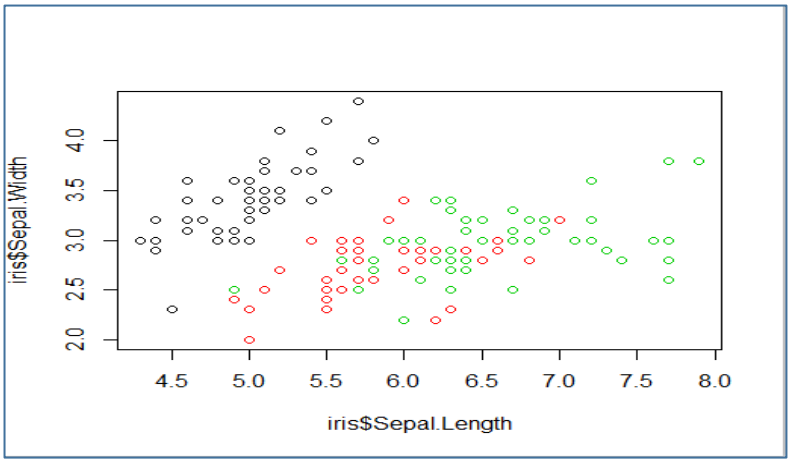
In this session, you saw how to make some basic plots using the R-base functions e.g. plot(), hist() or boxplot().

You typically have a two-dimensional space to create visualisations. We explored the following commands in R.

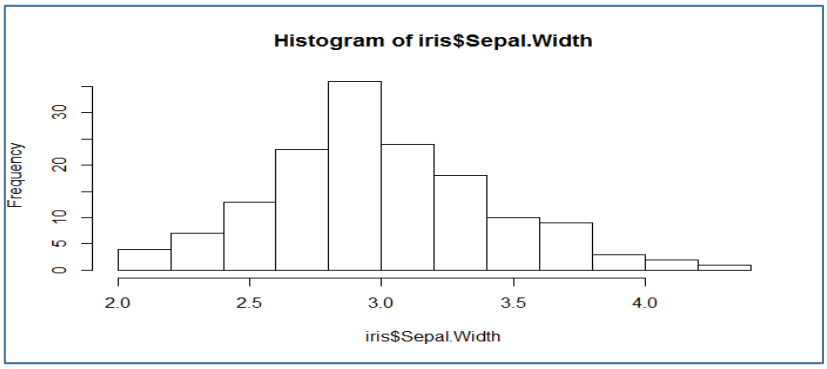
- Plotting sepal width versus sepal length
  >plot(iris$Sepal.Length, iris$Sepal.Width)



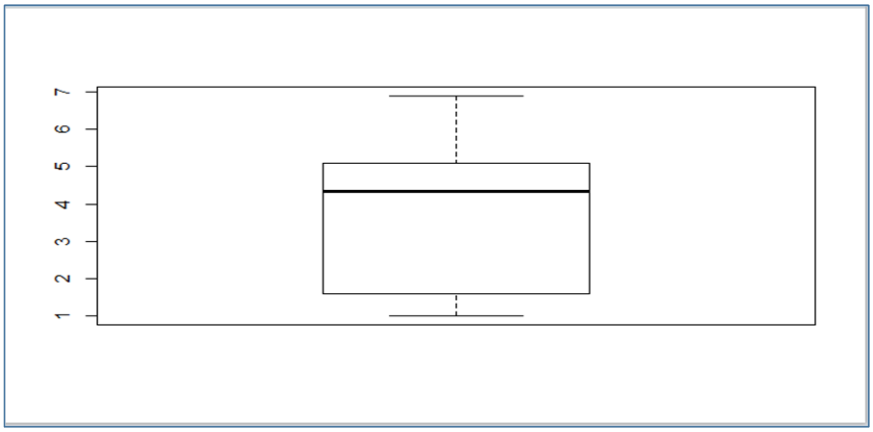
- Plotting sepal width versus sepal plotting and using colours to represent species
  >plot(iris$Sepal.Length, iris$Sepal.Width, col = iris$Species)



- Plotting histograms for sepal width using the hist() function
  >hist(iris$Sepal.Width)



- Plotting boxplot for petal length using the boxplot() function
  >boxplot(iris$Petal.Length)



Limitations of the Base Package

You learnt that there are a lot of limitations in using the base package, such as:

- Plots are not updated automatically with the data
- Plots are images, like jpeg, not objects; so it is hard to manipulate them after plotting
- You have to name the axes and legends manually

Thus, you need to use other library functions e.g. ggplot for creating better graphics and charts.

ggplot2: The Grammar of Graphics

The Grammar of Graphics: ggplot2

The three important graphical elements of ggplot2 are:

1. **Data** is the data sets you want to plot from – data frames, matrices, etc.
2. **Aesthetics** are scales where you map the data – axes, colour, fill, line width, line shape, etc.
3. **Geometries** are the types of plots – point (scatter), line, histogram, bar charts, etc.

Plotting Using ggplot Function

Let us look at the structure of ggplot command:

The Structure of Command: ggplot2

- ggplot(mtcars, aes(x = cyl, y = mpg)) + geom_point()

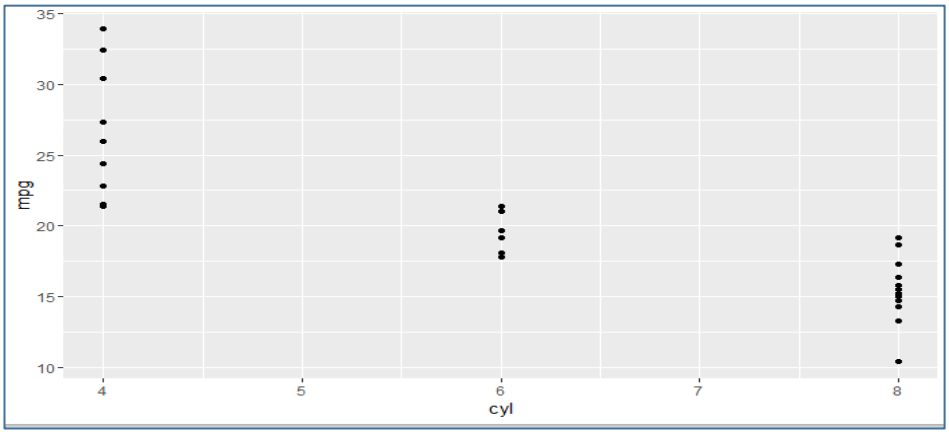
Data: mtcars data frame

Aesthetics: x and y axes

Geometry: Point

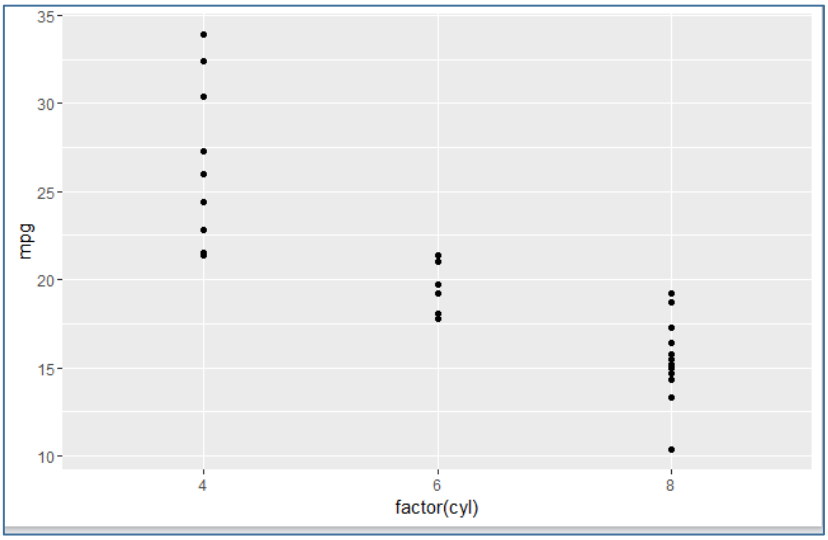
1. The first argument is the data layer. Here, mtcars is a data frame.
2. Aesthetics, the second layer, are put inside the aes() function – x, y, size, colour, etc.
3. The third layer, geometry, is simply added to the ggplot as a layer. This layer is used to specify what type of plot you want: points, bars, lines, etc.

The above command gives the scatter plot for “miles per gallon” and “cylinders”.



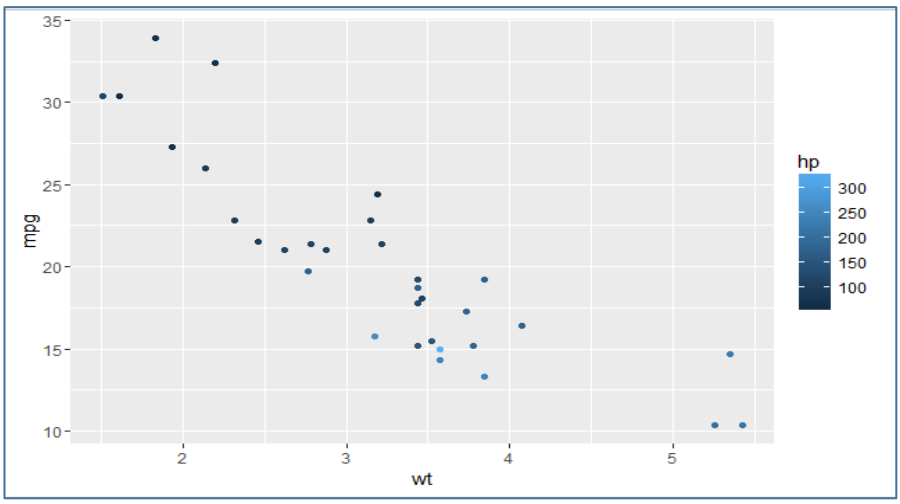
- You know that cylinder is not really a numeric variable, because intermediate values of cylinder such as 4.2, 5, 7, etc. is not possible. Hence, you would define “cyl” as a factor using factor(cyl) instead of only “cyl” in the x-axis. You can see that the x-axis changes to discrete values 4, 6 and 8.

  > ggplot(mtcars, aes(x = factor(cyl), y = mpg)) + geom_point()

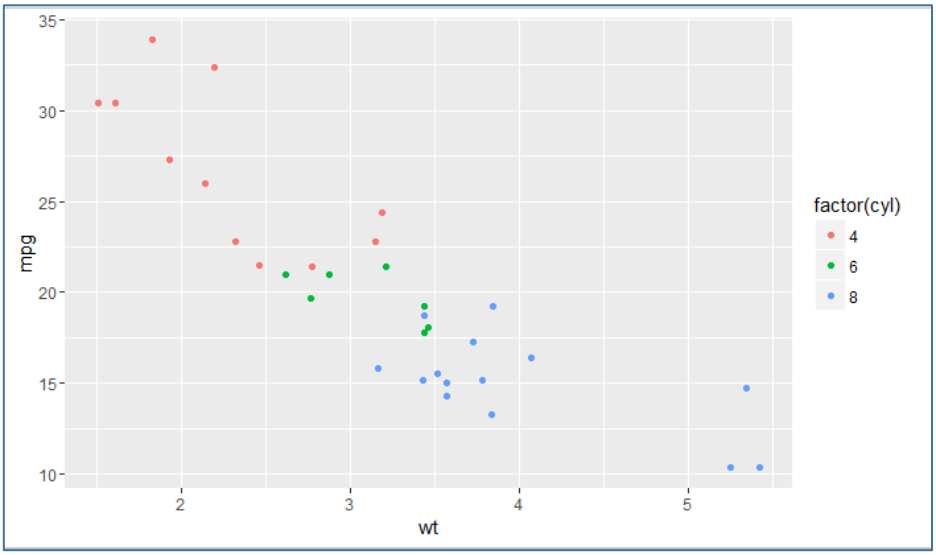


- Now, plotting the scatter plot for mpg versus weight
  > ggplot(mtcars, aes(x = wt, y = mpg)) + geom_point()

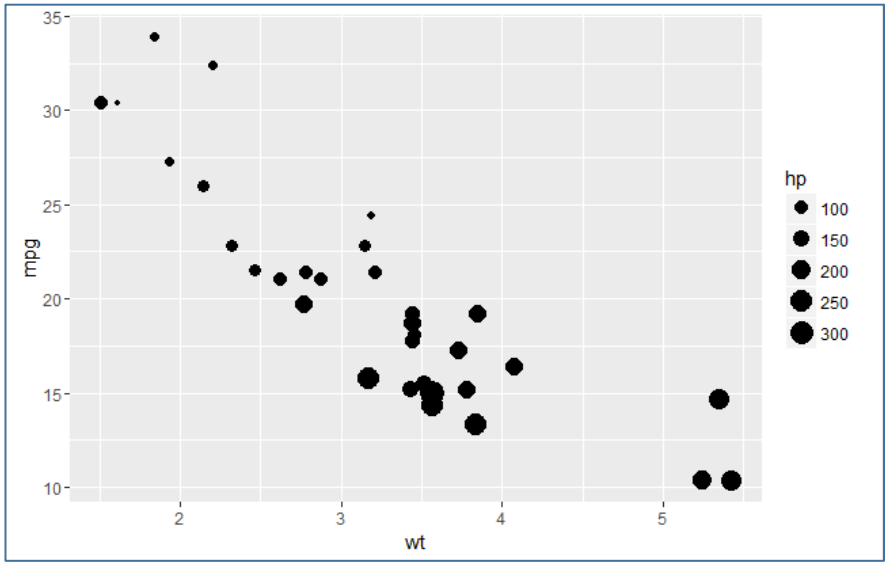
- Adding colour aesthetic to the above command to show an additional information about horsepower
  > ggplot(mtcars, aes(x = wt, y = mpg, col = hp)) + geom_point()



- Plotting mpg vs wt and using different colours to denote the number of cylinders
  >ggplot(mtcars, aes(x = wt,y = mpg, col = factor(cyl))) + geom_point()

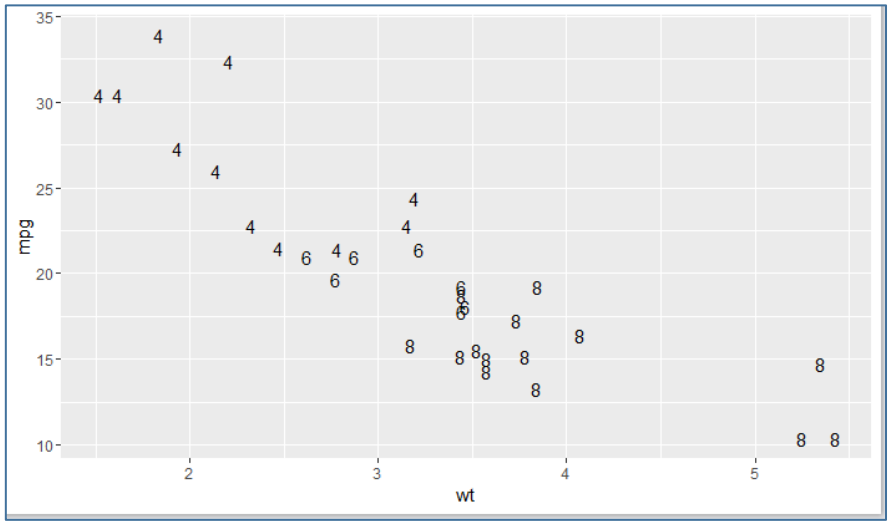


- Now, adding size aesthetic to show different horsepower
  > ggplot(mtcars, aes(x = wt, y = mpg, size = hp)) + geom_point()



- We can also add text as graphic to the scatter plot using the command “geom_text”
- Now, we used **geom_text** to display the number of cylinders. aes(label = cyl) tells R to label the points according to its *“cyl” value* on the graph

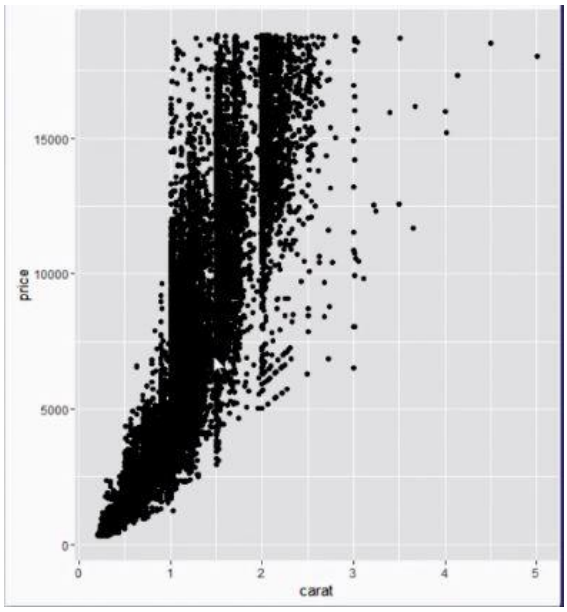
> ggplot(mtcars, aes(x = wt, y = mpg, label = cyl)) + geom_text(aes(label=cyl))



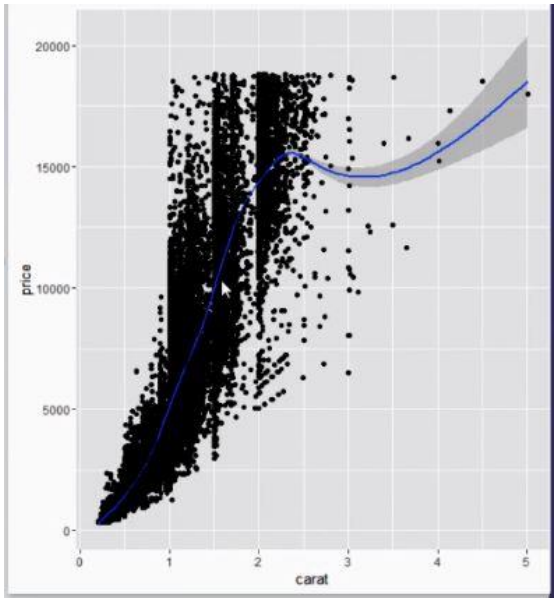
Plotting Using ggplot

In the previous sessions, you learnt how to plot some basic plots such as box plots, histograms and scatter plots. You also installed advanced plotting package called ggplot2 package and looked at some advanced techniques in plotting. You applied all these techniques on not so big data sets, so in this session you learnt how to visualise data sets which are relatively big.

- We used the diamond data set. We started by plotting the carat value on the x axis and the price on the y axis with point geometry
  > ggplot(diamonds, aes(x=carat, y=price))+geom_point()

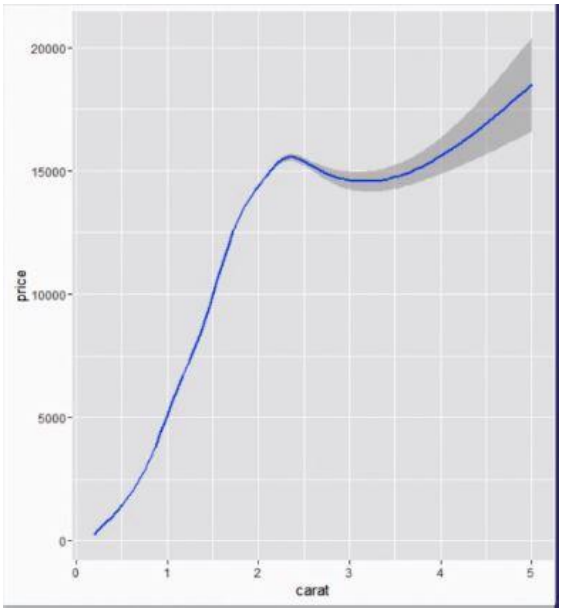


- Since there are so many points, it becomes difficult to understand what it trends in price as carat varies. Hence, plotting the carat value on the x axis and the price on the y axis with smoothening line and point geometry
  > ggplot(diamonds,aes(x=carat, y=price))+geom_point()+geom_smooth()



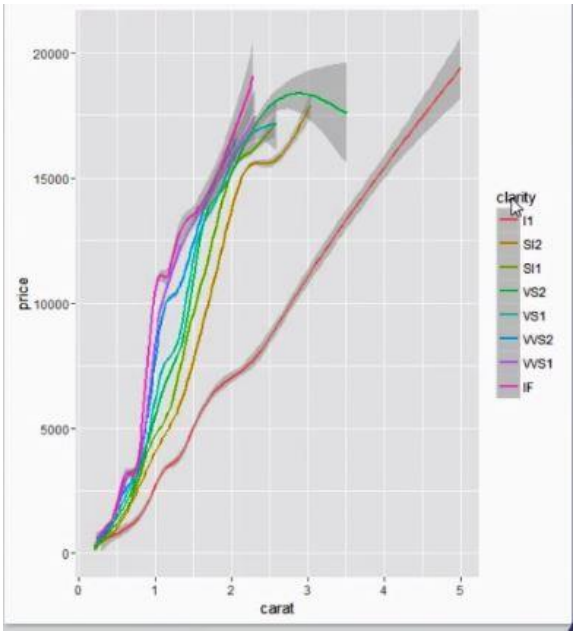
- You can remove the plotting of the carat value on the x axis and the price on the y axis with smoothening line

> ggplot(diamonds,aes(x=carat,y=price))+geom_smooth()



- You can add more dimensions to the plot area. Plotting the carat value on the x axis and the price on the y axis with smoothening line and colour clarity

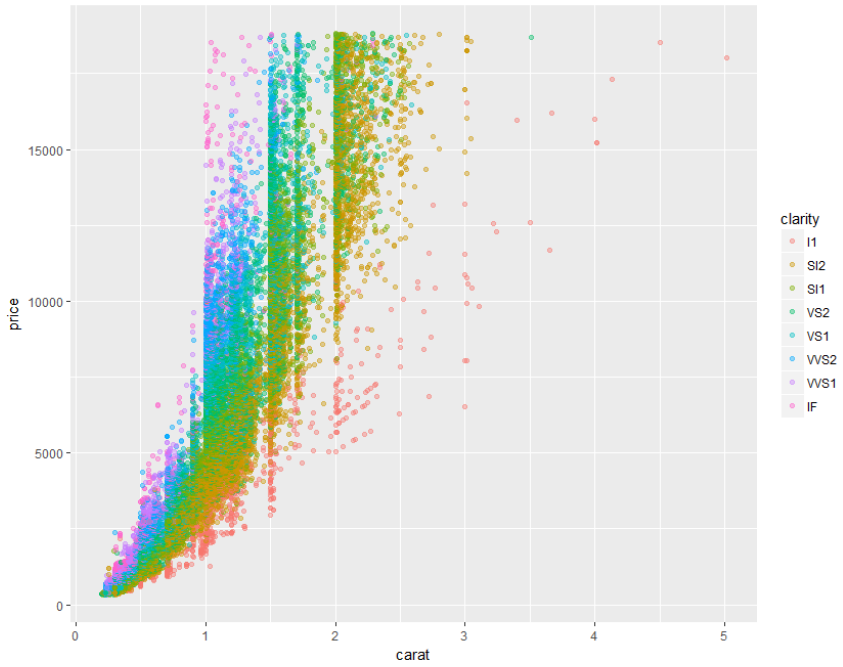
> ggplot(diamonds,aes(x=carat,y=price,col=clarity))+geom_smoot



- When there are too many points present in the plot area, some points might have the exact same coordinates in the xy plane. This proximity makes it difficult for the viewer to distinguish between those points. One way to avoid this problem is to play with the alpha parameter in R, which controls the transparency of the points.

Plotting the carat value on the x axis and the price on the y axis with point geometry and colour clarity with transparency.

> ggplot(diamonds, aes(x=carat, y=price,col=clarity))+geom_point(alpha=0.4)

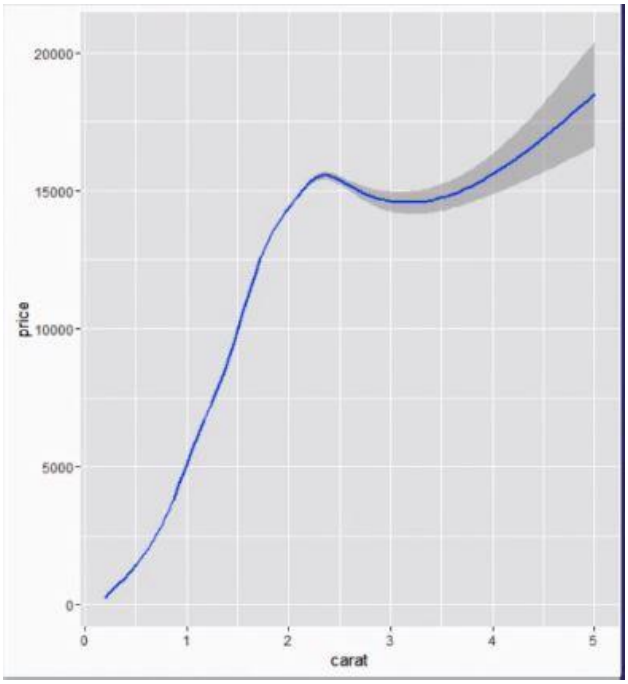


Visualisation as Object

You often create visuals but in case you have to modify them, you also have to re-create them. A great functionality of ggplot is that you can store visuals as objects and modify them. Thus, they can be saved for reuse and be built upon by adding more elements to it.

- Assigning plot to the variable
  > diamond_plot <-ggplot(diamonds, aes(x=carat, y=price))

- Adding additional attributes to the variable
  > diamond_plot + geom_smooth()

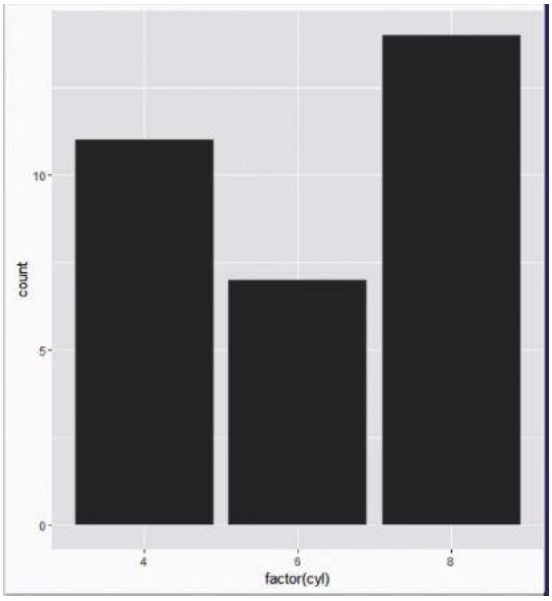


Plotting Using ggplot Function

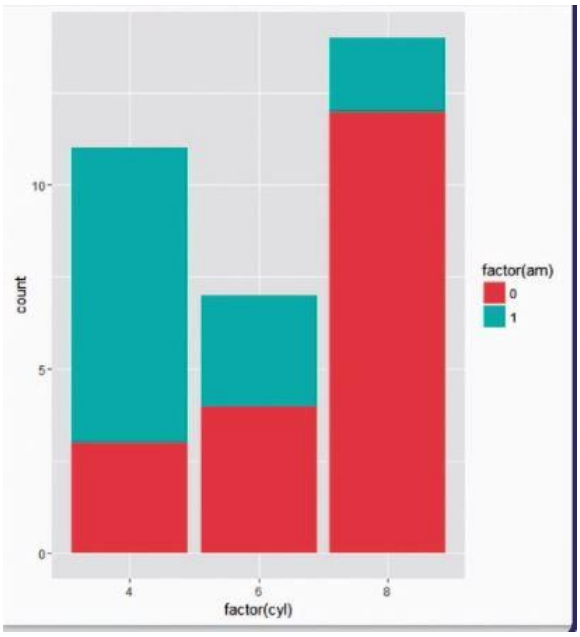
You created different types of bar graphs using ggplot on the mtcars data set.

- Using factor(cyl) and assigning it to cyl.am
  > cyl.am<-ggplot(mtcars, aes(x = factor(cyl))

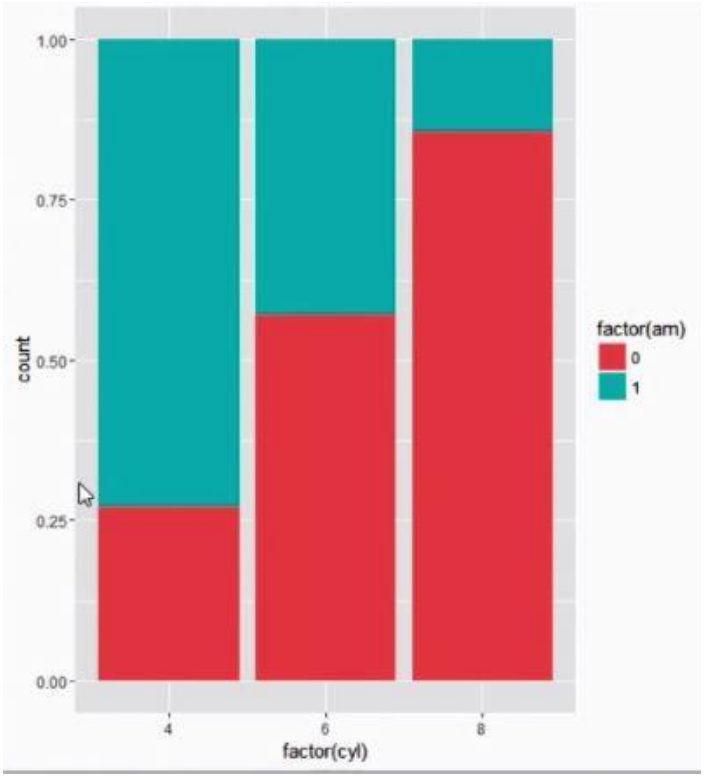
- Now, plotting cyl.am with bar geometry
  > cyl.am + geom_bar()



- Adding fill aesthetic to cyl.am and plotting stacked bar chart
  > cyl.am<-cyl.am + aes(fill=factor(am))
  > cyl.am + geom_bar()



- Plotting absolute bar chart. The height of each bar in this case is the same.
  > cyl.am + geom_bar(position="fill")



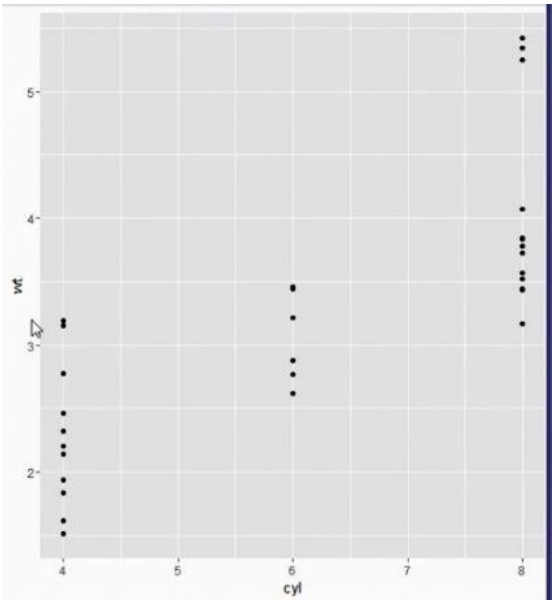
Factors Influencing Visualisation

- Factors Influencing Visualization
  - Aesthetics
    - Number of variables
  - Geometry
    - Types of variables

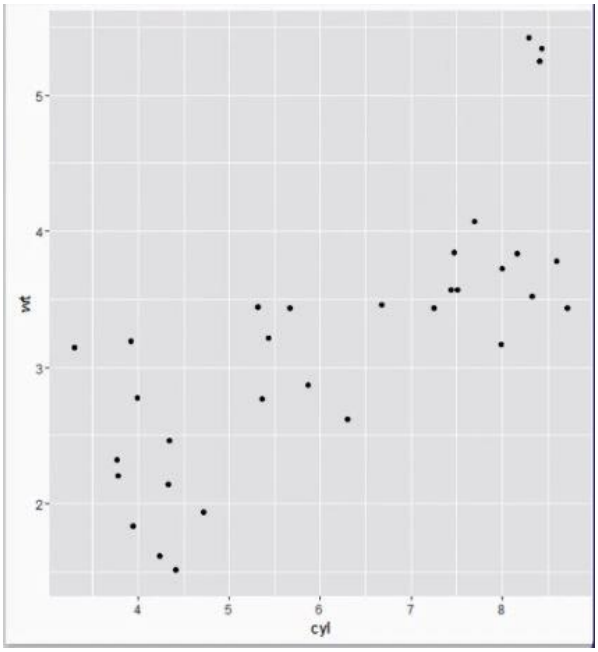
Jitter Plots

We previously discussed the issue of overlapping points when you plot too many points in the xy plane. You can avoid this issue by using jitter. Using jitter, you can scatter overlapping points in the neighbourhood to make them distinguishable. Also, it is advisable to use jitter for categorical variables and not quantitative variables.

- Now, plotting wt versus cyl
  > ggplot(mtcars, aes(x = cyl, y = wt)) + geom_point()

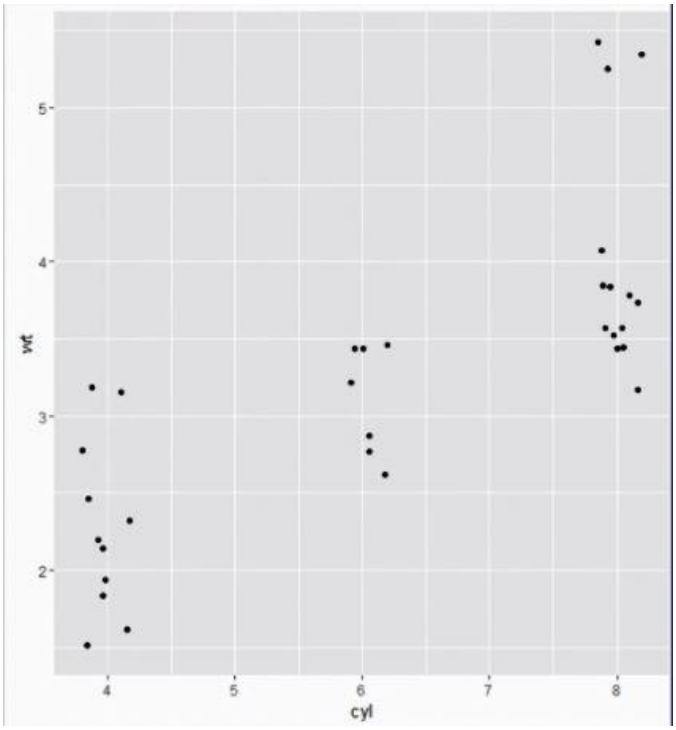


- Now, let's introduce jitter
  > ggplot(mtcars, aes(x = cyl, y = wt)) + geom_point(position="jitter)



As you can see, after usage of jitter, the points have been scattered around cylinder values 4, 6, and 8. Now, there is a way to control the spread that jitter introduces. Let's look at that.

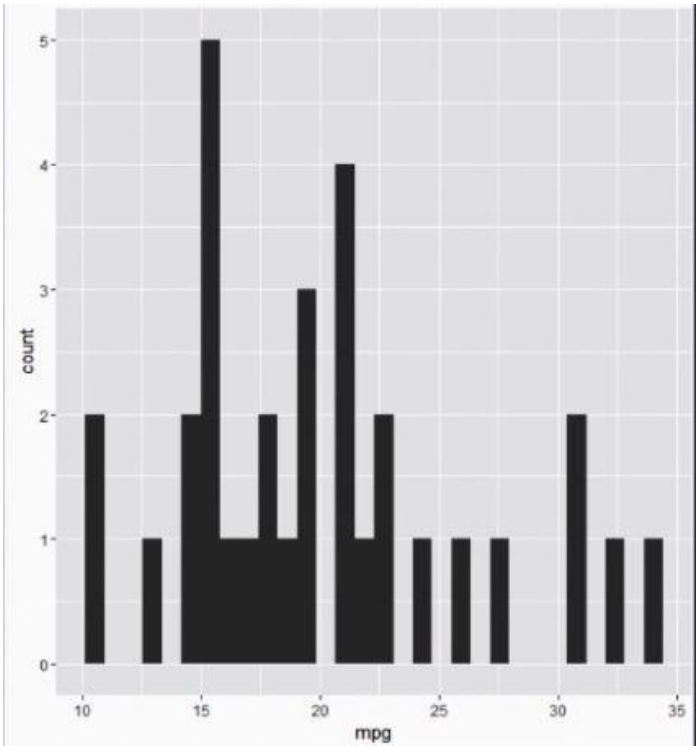
- Now, let's introduce jitter width
  > ggplot(mtcars, aes(x = cyl, y = wt)) + geom_point(position="jitter")
  > jitter_posn <- position_jitter(width=0.5)
  > ggplot(mtcars, aes(x = cyl, y = wt)) + geom_jitter(position=jitter_posn)



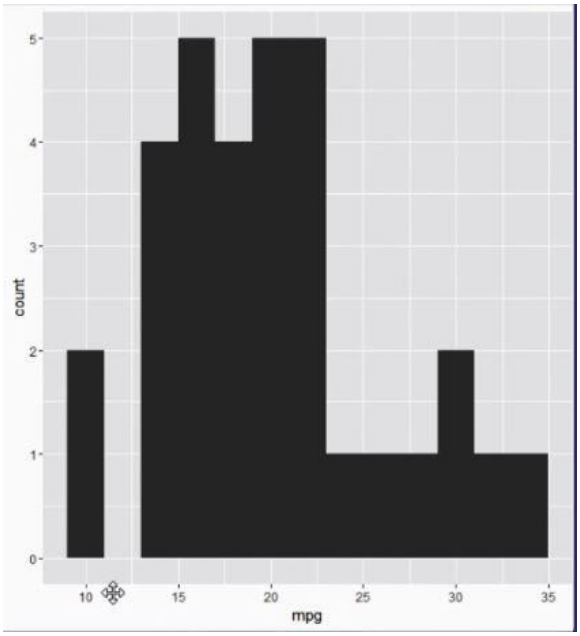
Histograms

You learnt different variations of bar charts. Next, you learnt the different variations of histograms as well, which can be plotted using ggplot.

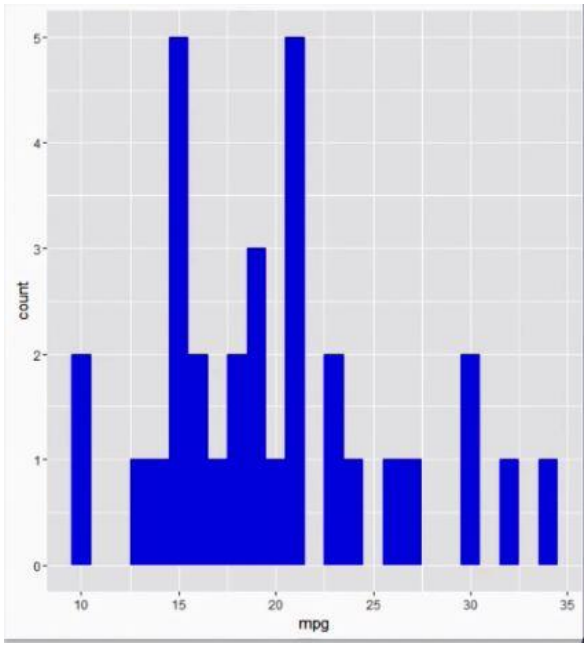
- Now, plotting histogram on mpg
  > ggplot(mtcars, aes(x = mpg)) + geom_histogram()



- Now, changing bin width to 2
  > ggplot(mtcars, aes(x = mpg)) + geom_histogram(binwidth=2)



- Now, changing colour
  >myBlue<-"blue"
  > ggplot(mtcars, aes(x = mpg)) + geom_histogram(binwidth=1,fill=myBlue)

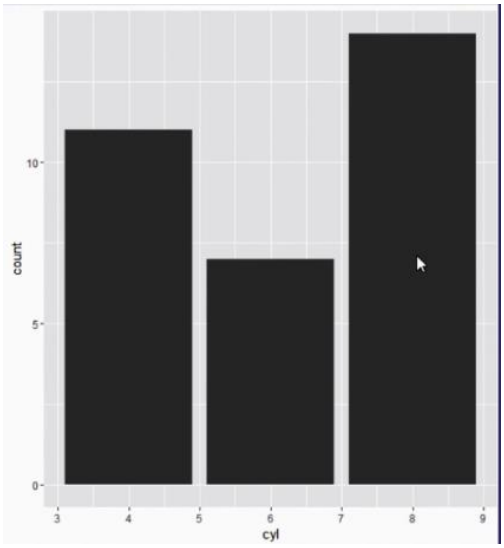


Note that bin width and colour are attributes not aesthetics and are being used in the geometric layer.

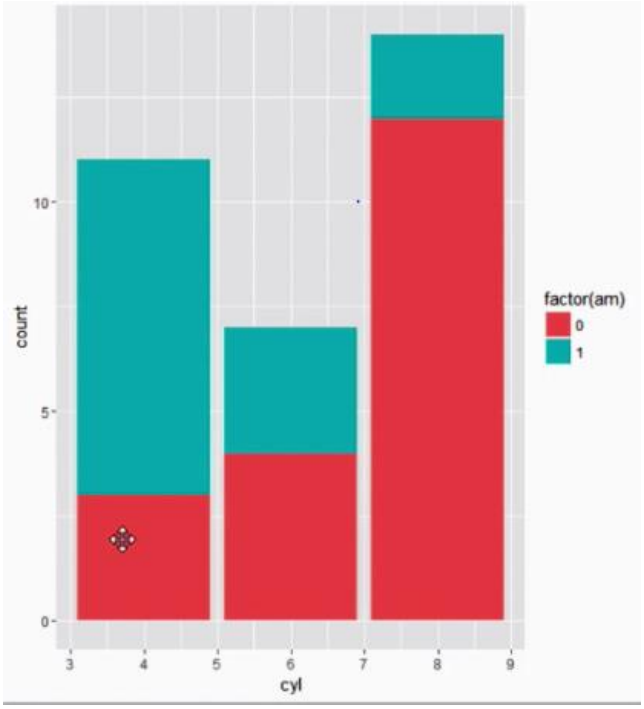
Bar Charts

We then revisited some aspects of bar charts.

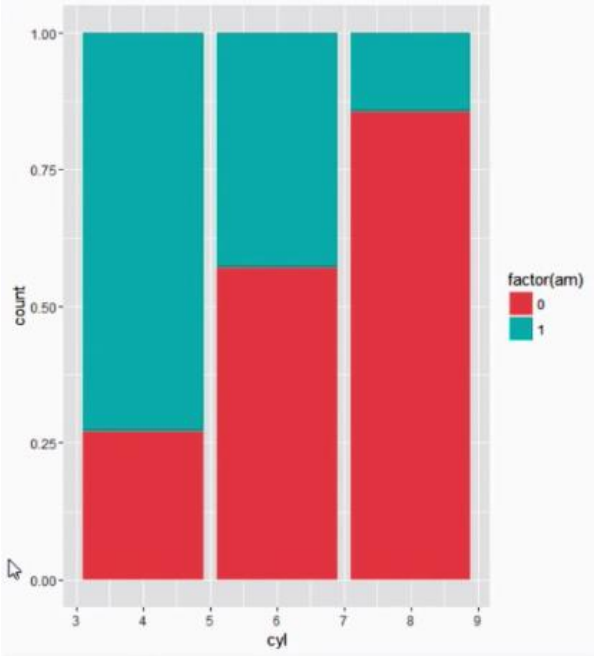
- Plotting bar chart on cyl
  >ggplot(mtcars, aes(x = cyl)) + geom_bar()
  >ggplot(mtcars, aes(x = cyl,fill=am)) + geom_bar(position="Stack")



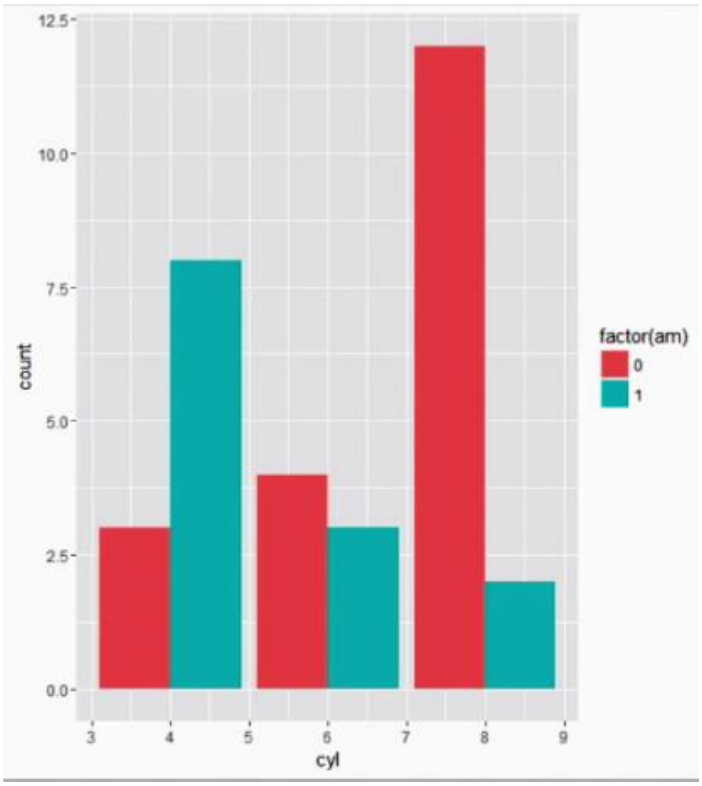
- Now, changing position argument to stack and adding factor(am)
  >ggplot(mtcars, aes(x = cyl,fill=factor(am))) + geom_bar(position="Stack")



- Now, changing position argument to fill it covers entire length of y axis
  >ggplot(mtcars, aes(x = cyl,fill=factor(am))) + geom_bar(position="fill")



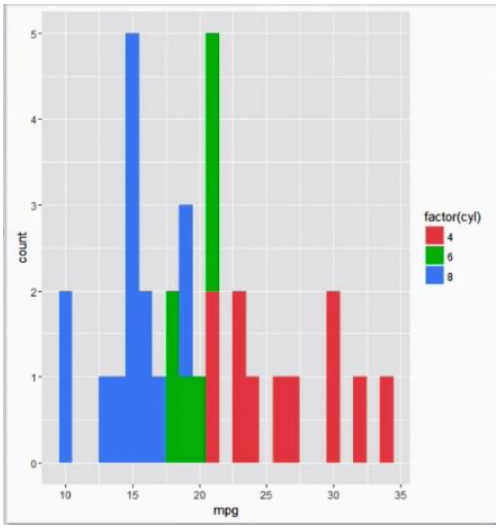
- Now, changing position argument to dodge and it makes the bars appear beside each other
  >ggplot(mtcars, aes(x = cyl,fill=factor(am))) + geom_bar(position="dodge")



Frequency Histograms

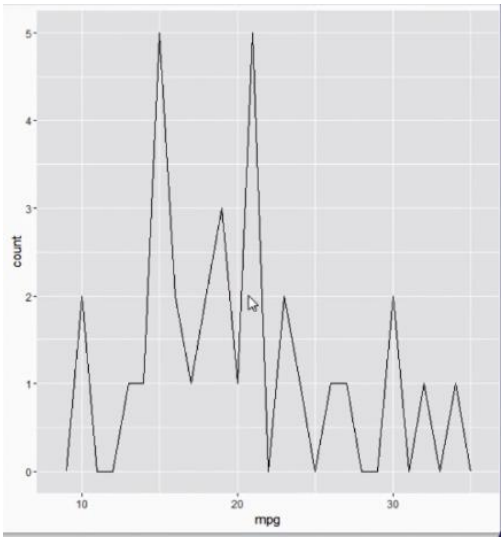
In the previous session, we discussed the difference between bar charts and histograms. We created some histograms too. Let's go back and take a look.

- Now, plotting a frequency histogram
  > ggplot(mtcars, aes(x = mpg,fill=factor(cyl))) + geom_histogram(binwidth=1)



Frequency Polygon

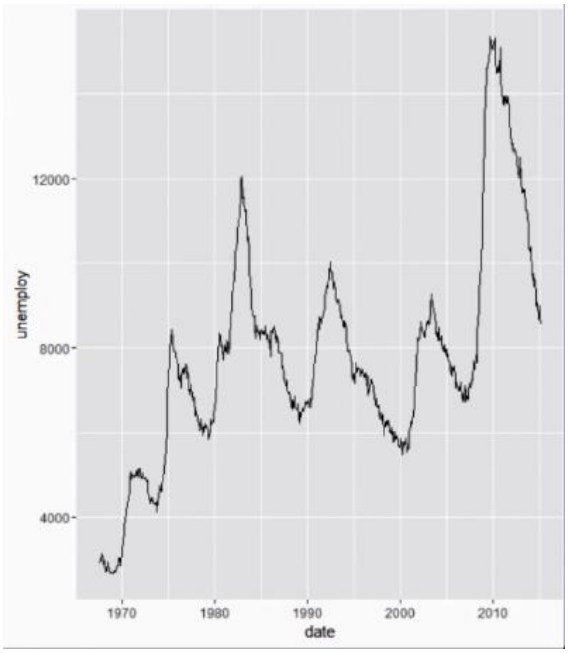
- Frequency polygons are made on top of histograms. Now, plotting a frequency polygon
  > ggplot(mtcars, aes(x = mpg,col=cyl)) + geom_freqpoly(binwidth=1)



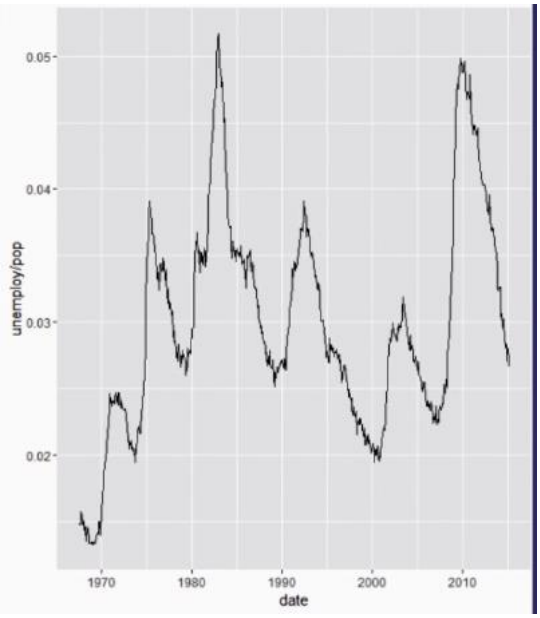
Plotting Economics Data Set using Line Graph

You then looked at another inbuilt data called “economics” which contains details regarding the US economy.

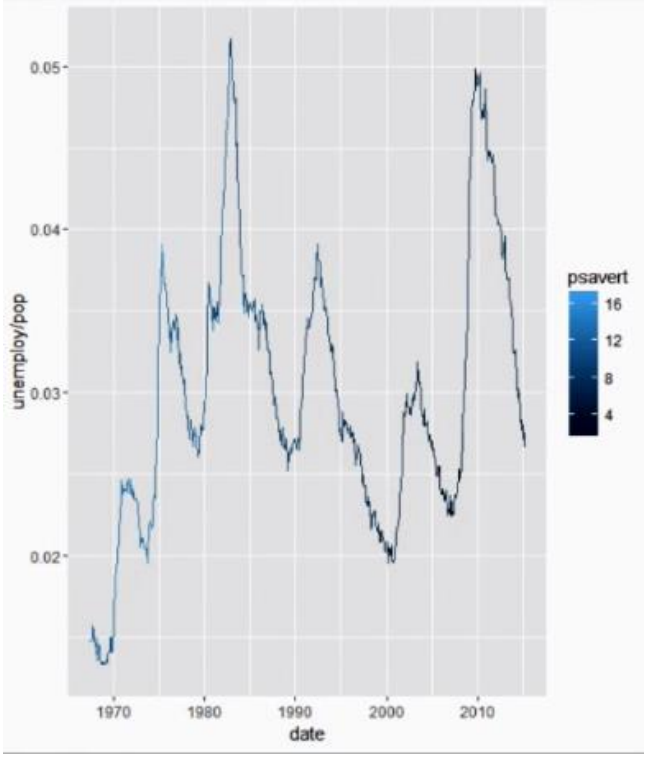
- Starting with basics and plotting a line graph
  > ggplot(economics, aes(x = date,y=unemploy)) + geom_line()



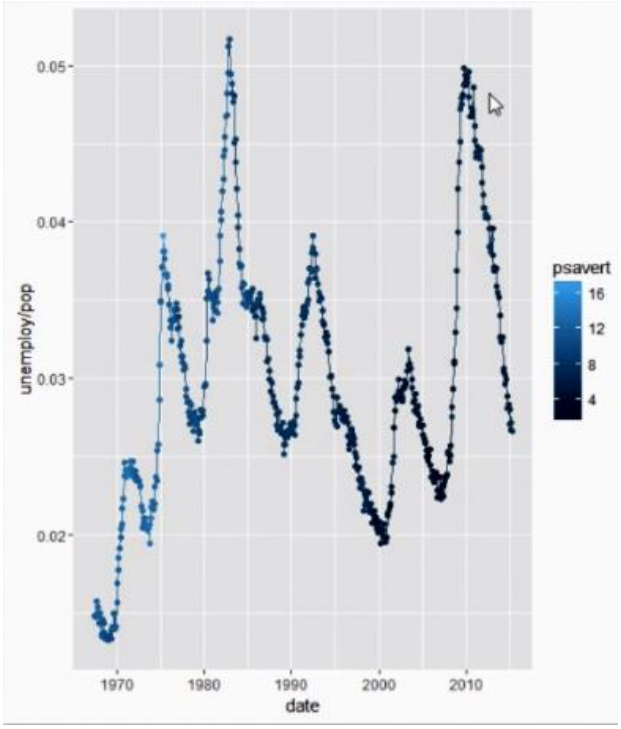
- Now, plotting a line graph by changing y as fraction of unemployment divided by population
  > ggplot(economics, aes(x = date,y=unemploy/pop)) + geom_line()



- Now, plotting a line graph by mapping colour to average savings rate
  > ggplot(economics, aes(x = date,y=unemploy/pop,col=psavert)) + geom_line()



- Now, plotting a line graph by adding geom_point layer
  > ggplot(economics, aes(x = date,y=unemploy/pop,col=psavert)) + geom_line()+geom_point()

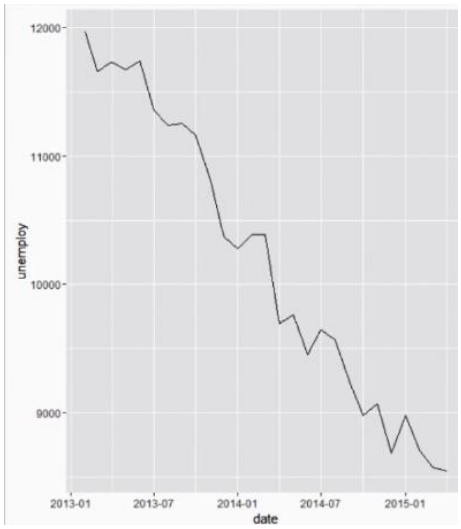


Plotting a Sample

In many cases, you need to visualise samples of the main data set. Let's take a look at how it is done.

- Creating subsets of Dataframe
  >recent <-economics[economics$date > as.date("2013-01-01"),]
  >str(recent)

- Plotting date on x axis and unemployment on y axis
  >ggplot(recent,aes(date,unemploy))+geom_line()



- Plotting date on x axis and unemployment on y axis as step graph
  >ggplot(recent,aes(date,unemploy))+geom_step()