Lab 3_visualizations

Netasha

2025-04-28

The most common graphical techniques

These visual methods help identify important patterns and insights. Here are some of the most commonly used graphical techniques:

1. Boxplot
2. Histogram (also Density)
3. Pareto Chart
4. Pie Chart
5. Bar Chart (OG, Stacked, & Grouped)
6. Scatter Plot
7. Run Chart

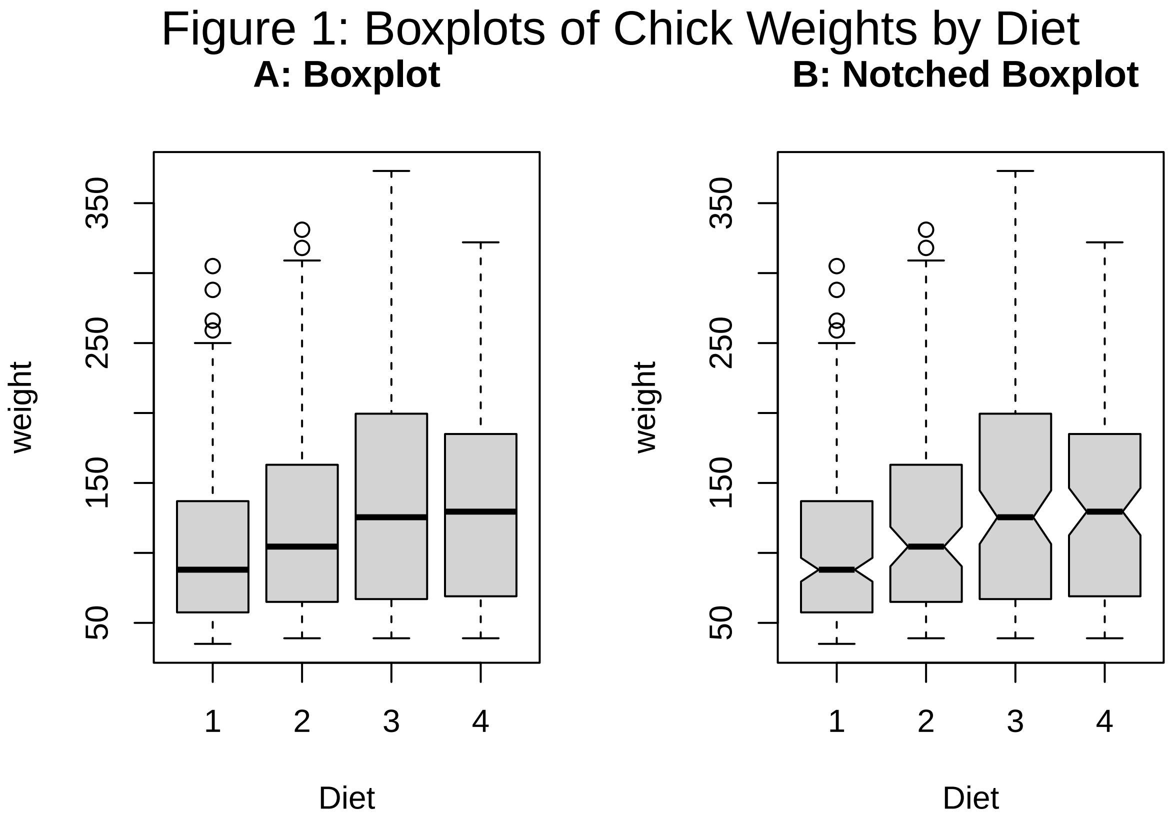
Boxplot

```
#![](Boxplot.png)
#We are going to use the ACSWR package which has the information of baby chickens and their diet. The C
#install.packages("ACSWR")
library(ACSWR)
head(ChickWeight)
```

```
##   weight Time Chick Diet
## 1     42    0     1    1
## 2     51    2     1    1
## 3     59    4     1    1
## 4     64    6     1    1
## 5     76    8     1    1
## 6     93   10     1    1
```

```
#![](babychick.png)
# We can use the par function function from the graphics function here to sect up a vector of the form
par(mfrow=c(1, 2))

#Use the boxplot function from the  graphics package. Since our outcome is weight and it is continuous
boxplot(weight ~ Diet, data=ChickWeight)
title("A: Boxplot")

#if notch is TRUE, a notch is drawn in each side of the boxes. If the notches of two plots do not overl
boxplot(weight ~ Diet, data=ChickWeight, notch=TRUE)
title("B: Notched Boxplot ")
```

```r
mtext("Figure 1: Boxplots of Chick Weights by Diet", outer=TRUE, cex=1.5, line=-1.5)
```



What do you notice?

-We have identified a few outliers in the weights of chicks that received Diet 1 and Diet 2.

- The right boxplots show that there is an overlapping between notches of Diet 3 and Diet 4. Thus, we can conclude with 95% confidence level that the true median are not differ for the last two diets.

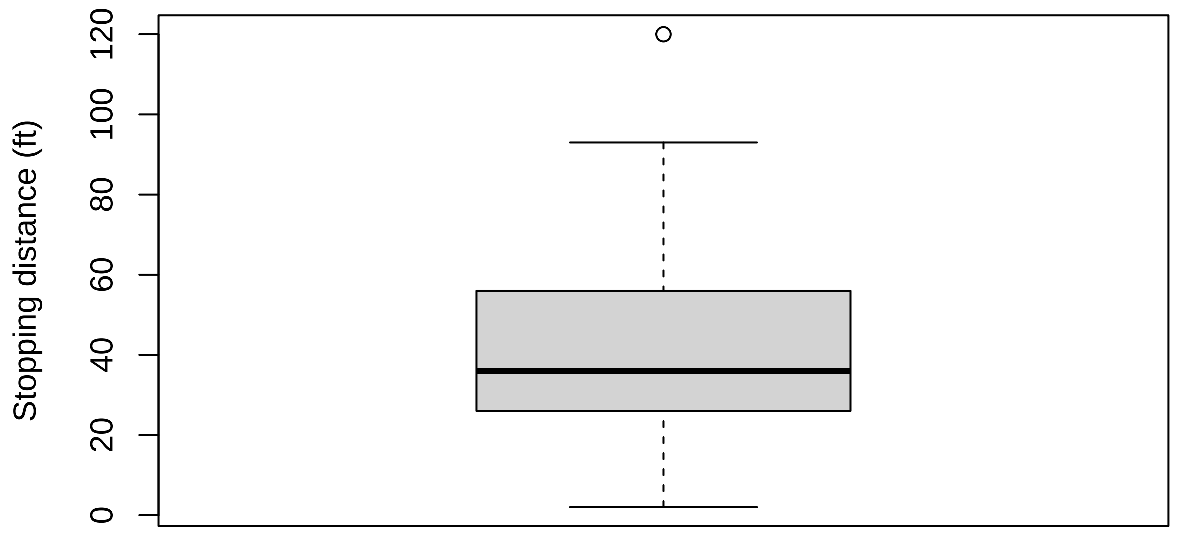
Boxplot Statistics

- The statistics of boxplot as well as the identified outliers can be obtained by boxplot.stats() function from grDevices package.

```r
#install.packages("grDevices")
library(grDevices)
```

Lets Construct the Boxplot for car distance in cars data and obtain the boxplot statistics.

```r
#this is us creating a visual aid for the box plot
 boxplot(cars$dist,ylab="Stopping distance (ft)")
```



```
#this is use using the boxplot.stats function to get all of the statistics in our boxplot
boxplot.stats(cars$dist)
```

```
## $stats
## [1]  2 26 36 56 93
##
## $n
## [1] 50
##
## $conf
## [1] 29.29663 42.70337
##
## $out
## [1] 120
```

- stats*‘ : a vector of length 5, containing th the lower whisker, , the median, the and the extreme of the upper whisker.
- n : the number of non-NA observations in the sample.
- conf : the lower and upper extremes of the notch (if(do.conf)).
- out : the values of any data points which lie beyond the extremes of the whiskers (if(do.out)).

Histogram

- The histogram was invented by the eminent statistician Karl Pearson. It is one of the earliest types of graphical display.
- The basic idea is to plot a bar over an interval proportional to the frequency of the observations that lie in that interval.
- If the sample size is adequate and accurately represents the population, the histogram reflects the shape of the underlying uncertainty curve.
- Furthermore, the Pareto chart, stem-and-leaf plot, and a few others may be shown as special cases of the histogram.

Examples Example Generate a random sample of weights for 300 individuals, where:

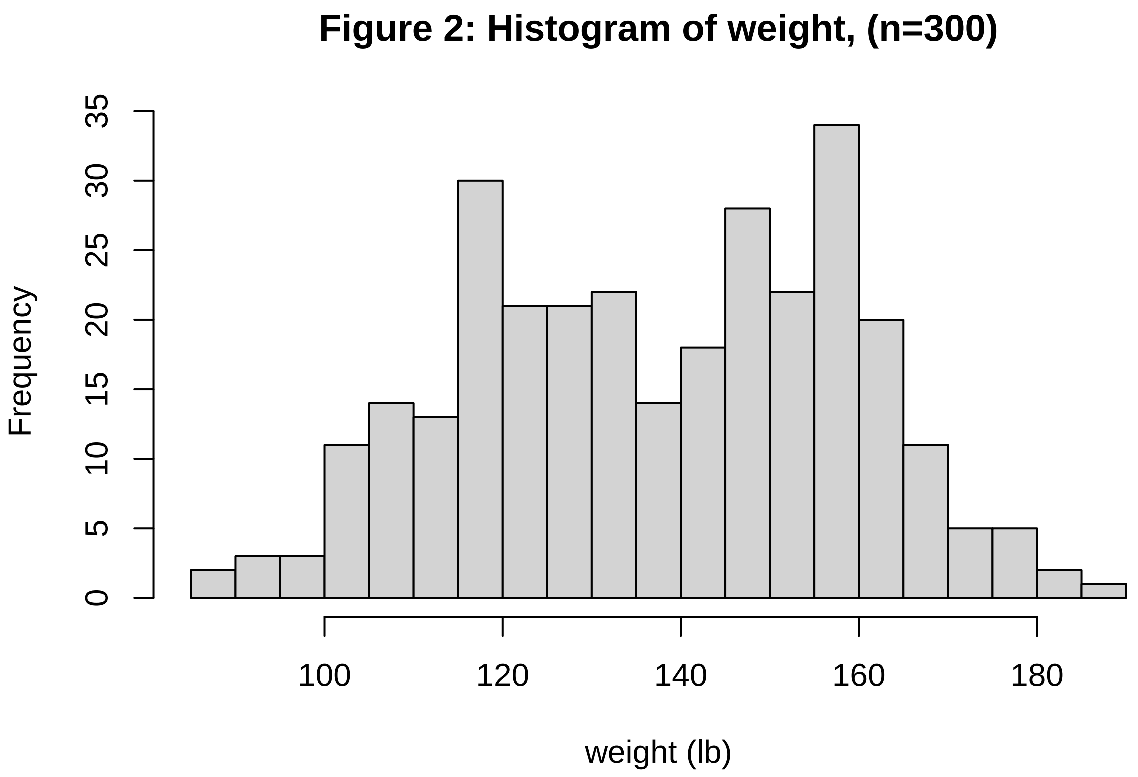
150 females have weights following a normal distribution with a mean of 120 lb and a standard deviation of 12 lb.

150 males have weights following a normal distribution with a mean of 155 lb and a standard deviation of 12 lb.

```r
set.seed(012024)
df <- data.frame(
  sex=factor(rep(c("F", "M"), each=150)),
  weight=round(c(rnorm(150, mean=120, sd=12), rnorm(150, mean=155, sd=12)))
  )

df[sample(nrow(df), 5), ]
```

```
##     sex weight
## 135   F    108
## 54    F    132
## 240   M    151
## 53    F    104
## 61    F    110
```

```r
# Basic histogram
hist(df$weight, breaks = 30, xlab="weight (lb)", main="Figure 2: Histogram of weight, (n=300)")
```

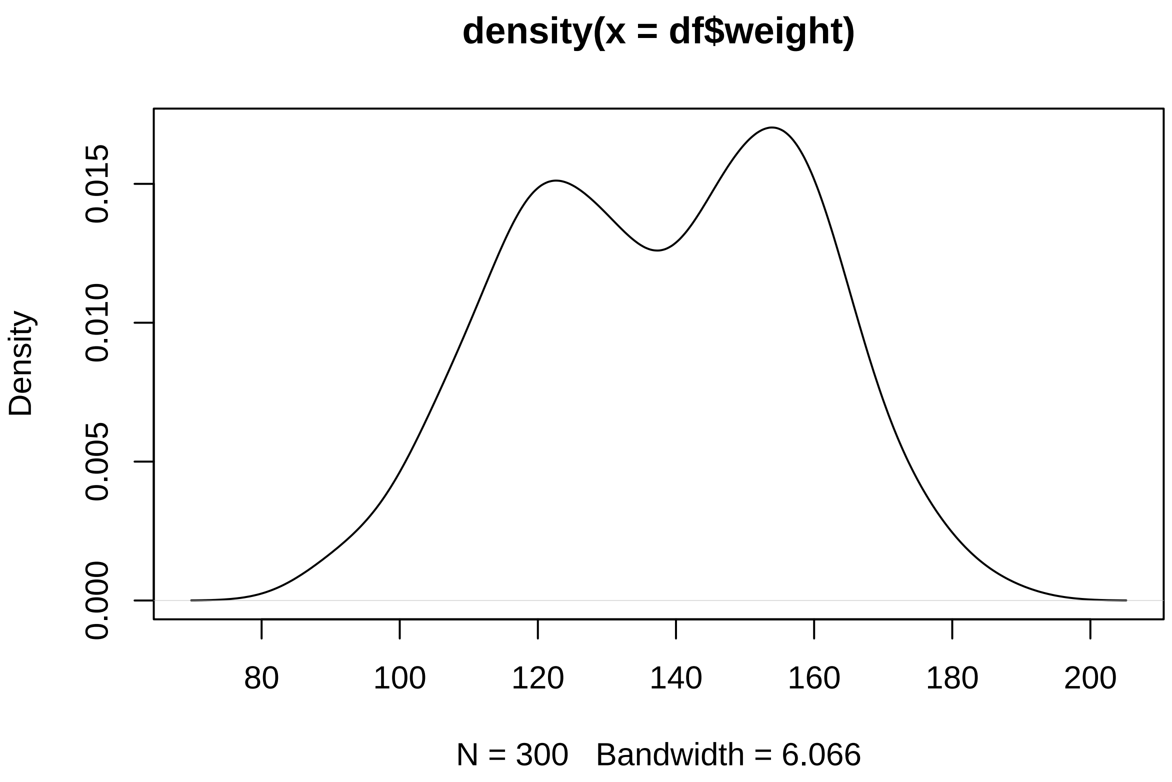


-What do we see?

- Histogram reveals a bimodal, symmetric distribution. The blue vertical line shows the mean of data.

This pattern indicates two distinct peaks in the data, corresponding to the different weight distributions for females and males.

```r
d <- density(df$weight) # returns the density data
plot(d) # plots the results
```



Pareto Chart

History

- The Pareto chart has been designed to address the implicit questions answered by the Pareto law.
- The common understanding of the Pareto law is that "majority resources" is consumed by a "minority users."
- The most common of the percentages is the 80–20 rule, implying that 80% of the effects come from 20% of the causes.

What is the Pareto Chart

- The Pareto chart gives very smart answers by answering "how much is owned by how many?"
- A Pareto chart is a bar graph. The lengths of the bars represent frequency or cost (time or money), and are arranged with longest bars on the left and the shortest to the right. In this way the chart visually depicts which situations are more significant.

When to use the Pareto chart?

- When analyzing data about the frequency of problems or causes in a process.
- When there are many problems or causes and you want to focus on the most significant.
- When analyzing broad causes by looking at their specific components.
- When communicating with others about your data.

Example

- The Pareto chart can be plotted using pareto.chart from the qccpackage in R.

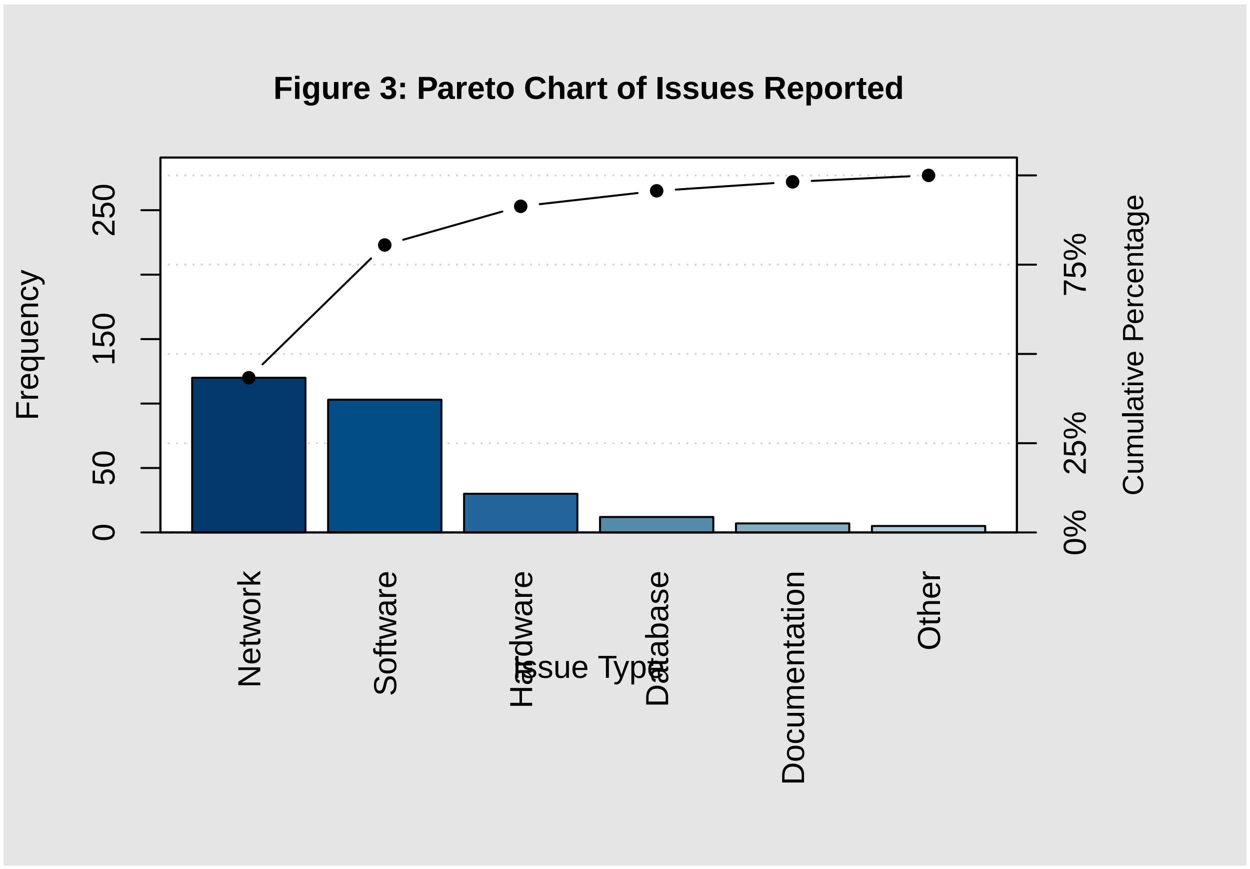
- The Pareto chart contains three axes on a two-dimensional plot only.
- Generally, causes/users are arranged along the horizontal axis.
- The bars are arranged in a decreasing order of the frequency. The left-hand side of the vertical axis denotes the frequencies.
- The cumulative frequency curve along the causes are then simultaneously plotted with the right-hand side of the vertical axis giving the cumulative counts.
- Thus, at each cause we know precisely its frequency and also the total issues up to that point of time

```
#Plot the Pareto chart for identifying the most common issues reported in the support center and unders
#install.packages("qcc")
library(qcc)
```

```
## Package 'qcc' version 2.7
```

```
## Type 'citation("qcc")' for citing this R package in publications.
```

```
# Shrink the margins
par(mar = c(5, 4, 4, 4))  # (bottom, left, top, right)

freq = c(120, 103, 30, 12, 7, 5)
names(freq)=c("Network", "Software", "Hardware", "Database", "Documentation", "Other")

# Create the Pareto chart
pareto.chart(freq,
             main = "Figure 3: Pareto Chart of Issues Reported",
             xlab = "Issue Type",
             ylab = "Frequency",
             ylab2 = "Cumulative Percentage")
```



```
##
## Pareto chart analysis for freq
##                    Frequency  Cum.Freq. Percentage Cum.Percent.
##   Network         120.000000 120.000000  43.321300    43.321300
##   Software        103.000000 223.000000  37.184116    80.505415
##   Hardware         30.000000 253.000000  10.830325    91.335740
##   Database         12.000000 265.000000   4.332130    95.667870
##   Documentation     7.000000 272.000000   2.527076    98.194946
##   Other             5.000000 277.000000   1.805054   100.000000
```

Pie Chart

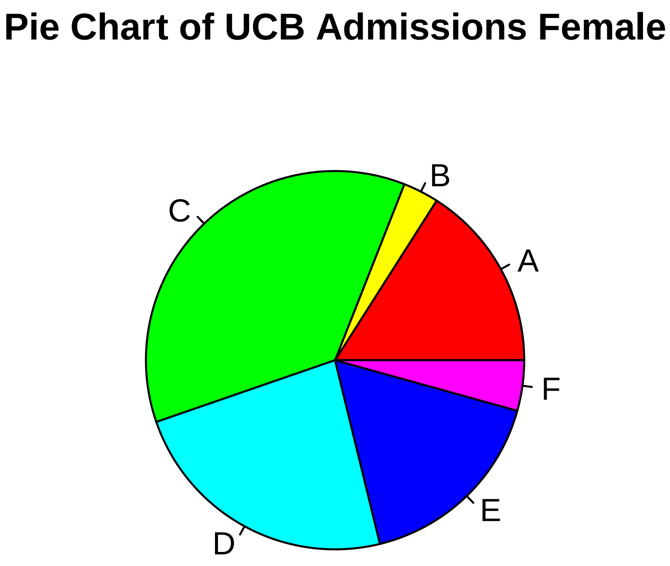
- A pie chart is a graphical representation technique that displays data in a circular-shaped graph.
- It is a composite static chart that works best with few variables.
- Pie charts are often used to represent sample data with data points belonging to a combination of different categories.
- The features of Pie charts are somewhat limited.

```
# The data is ready to use!
Tab<-UCBAdmissions[1,2,]
Tab
```

Example

```
##   A   B   C   D   E   F
##   89  17 202 131  94  24
```

```r
#use the pie function in graphics to produce a pie graph
pie(Tab,
    labels = names(Tab),                  # Add labels for each slice
    main = "Pie Chart of UCB Admissions Female",  # Title of the chart
    col = rainbow(length(Tab)))            # Use a rainbow color palette for the slices
```



- Note. NEVER USE A 3D ANYTHING!

Bar Chart

- A bar graph can be defined as a graphical representation of data, quantities, or numbers using bars or strips.
- They are used to compare and contrast different types of data, frequencies, or other measures of distinct categories of data.

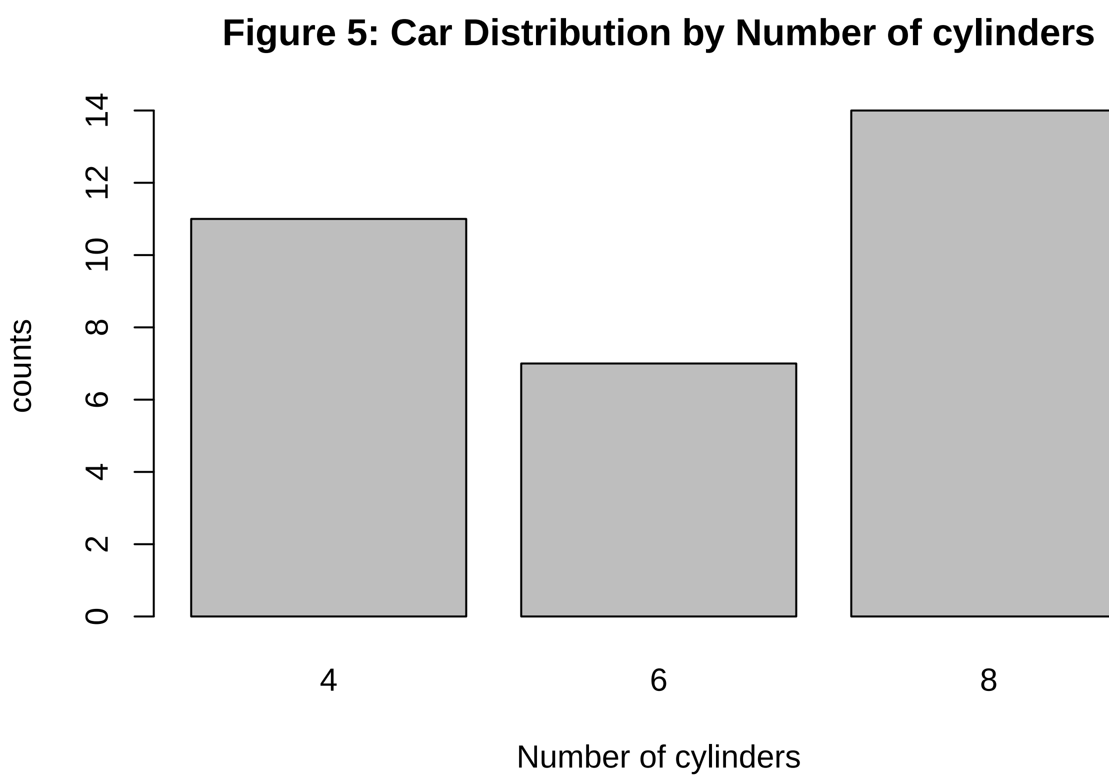
Example

- Example : Construct the bar plot of cyl type in mtcars dataset

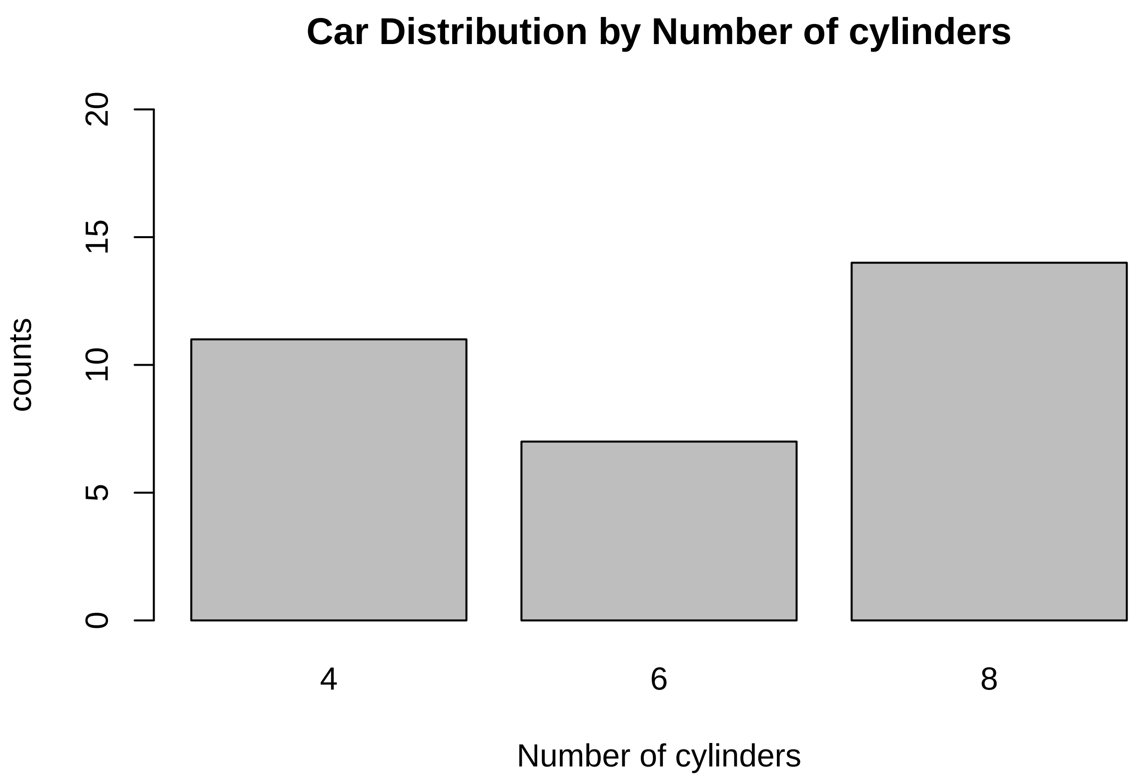
```r
# we are using the table function to create a table of the data. We want to get the frequency of cars t
(counts <- table(mtcars$ cyl))
```

```
##
##  4  6  8
## 11  7 14
```

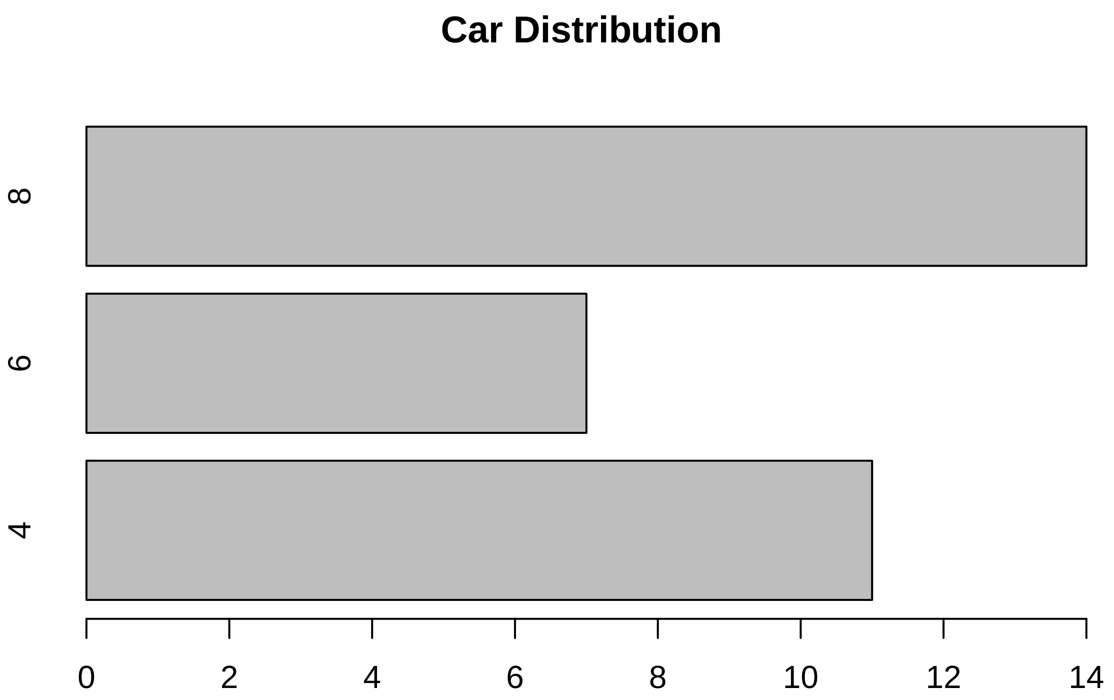
```r
#we use the barplot function from the graphics package
barplot(counts, main="Figure 5: Car Distribution by Number of cylinders",
   xlab="Number of cylinders", ylab="counts")
```



```r
#expand your y-axis limits by adding ylim
barplot(counts, main="Car Distribution by Number of cylinders",
   xlab="Number of cylinders", ylab="counts", ylim=c(0,20))
```



```
# change your Bar to a horizontal display using horix =TRUE
barplot(counts, main="Car Distribution", horiz=TRUE)
```

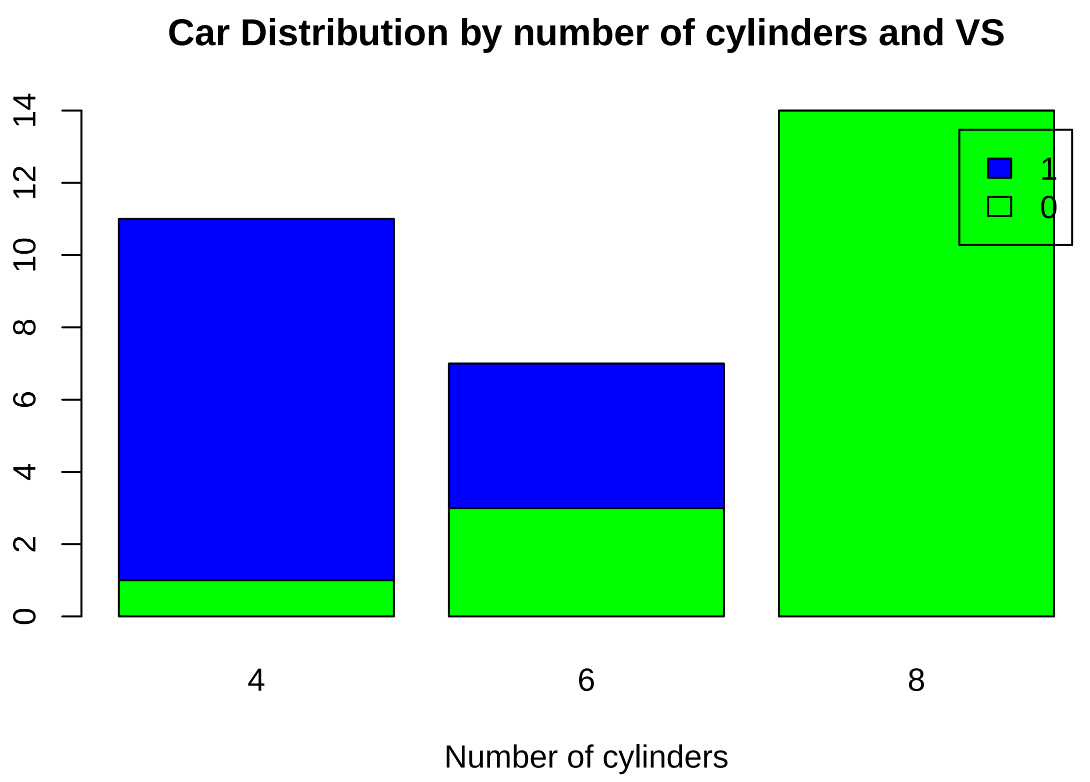


#### Bar Charts with more than one variable - Construct a stacked bar plot of cyl (cylinders) grouped by vs (engine shape) in the mtcars dataset.

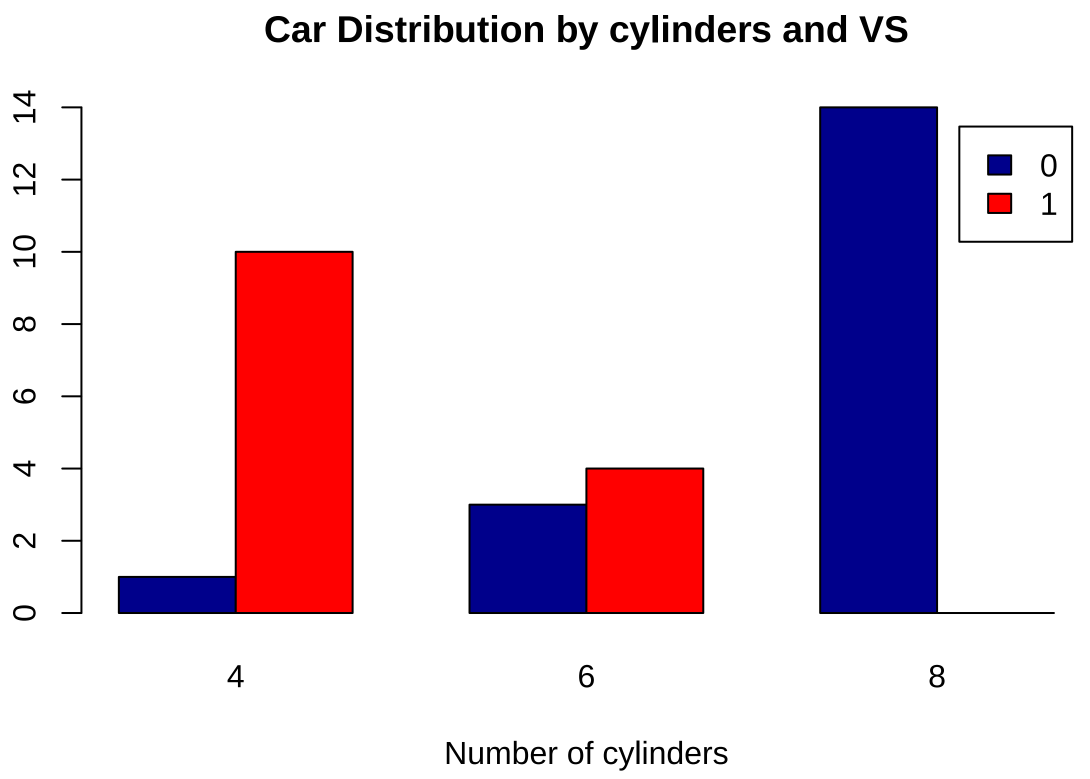
```
#combine engine shape and cylinders into one table
# 0 = v-shaped (This layout allows more cylinders to fit in a compact space, providing better power-to-
(counts <- table(mtcars$vs, mtcars$cyl))
```

```
##
##       4  6  8
##   0   1  3 14
##   1  10  4  0
```

```
#we are using the bar plot to plot both shape and number of cylinders
barplot(counts, main="Car Distribution by number of cylinders and VS",
  xlab="Number of cylinders", col=c("green", "blue"),
 legend.text=rownames(counts) )
```



```r
barplot(counts, main="Car Distribution by cylinders and VS",
  xlab="Number of cylinders", col=c("darkblue","red"),
  legend = rownames(counts), beside=TRUE)
```



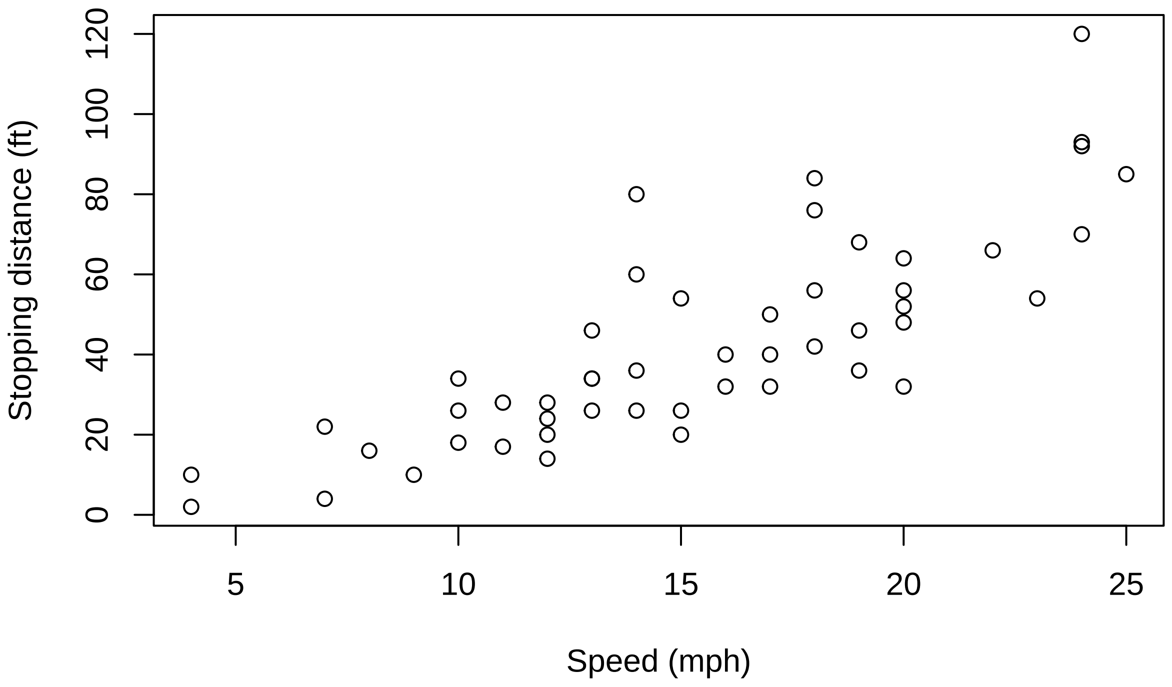
Scatter plots

- Whenever we have paired data and suspect a relationship between the variables, it is natural to plot them against each other using a scatter plot (or x-y plot).

```
#Figure out the relationship between the speed(mph) and stopping distance dist in ft.

data(cars)
# we can create a scatter plot using the plot function #make sure to list both variables consecutively.
plot(cars$speed, cars$dist,data=cars, ylab= "Stopping distance (ft)",xlab="Speed (mph)")
```

Example

```
## Warning in plot.window(...): "data" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "data" is not a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "data" is not a
## graphical parameter
## Warning in axis(side = side, at = at, labels = labels, ...): "data" is not a
## graphical parameter

## Warning in box(...): "data" is not a graphical parameter

## Warning in title(...): "data" is not a graphical parameter
```



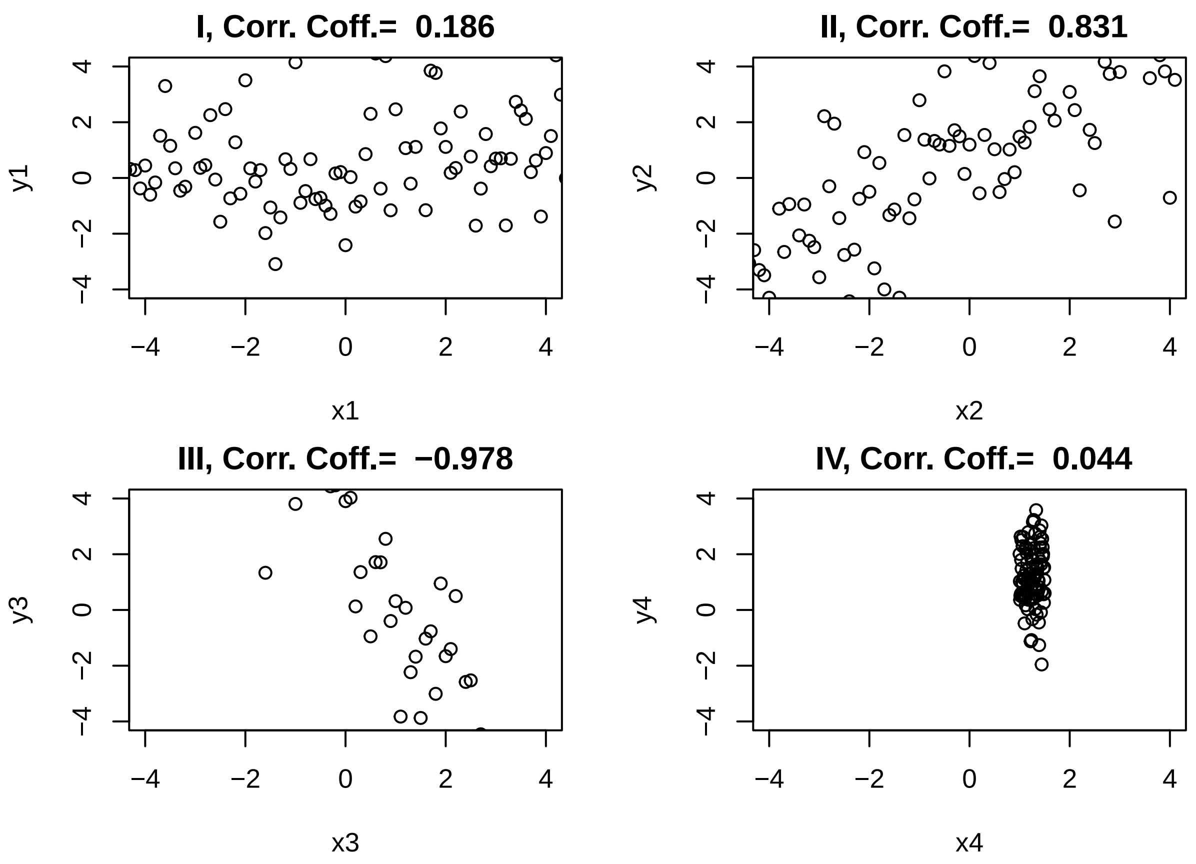
What does the plot suggests? - The display suggests a strong direct correlation between the two variables.

What should you do next? - find the Pearson's product moment correlation using cor.testfunction

```
(cor.test(cars$speed, cars$dist))
```

```
##
##  Pearson's product-moment correlation
##
## data:  cars$speed and cars$dist
## t = 9.464, df = 48, p-value = 1.49e-12
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.6816422 0.8862036
## sample estimates:
##       cor
## 0.8068949
```

```
# Shrink the margins
par(mar = c(4, 4, 2, 2))  # (bottom, left, top, right)

set.seed(1234)
x1=seq(-5,5,0.1)
y1=-0.05*x1+1.2+rnorm(length(x1),0,2)

x2=seq(-5,5,0.1)
y2=x2+1+rnorm(length(x2),0,2)

x3=seq(-5,5,0.1)
y3=-3*x3+3+rnorm(length(x3),0,2)

x4=seq(1,1.5,0.005)
y4=x4+rnorm(length(x4),0,1)

par(mfrow=c(2,2))
plot(x1,y1,main= paste("I, Corr. Coff.= ", round(cor.test(x1,y1)$estimate,3)),xlim=c(-4,4),ylim=c(-4,4)
plot(x2,y2,main=paste(" II, Corr. Coff.= ", round(cor.test(x2,y2)$estimate,3)),xlim=c(-4,4),ylim=c(-4,4
plot(x3,y3,main=paste("III, Corr. Coff.= ", round(cor.test(x3,y3)$estimate,3)),xlim=c(-4,4),ylim=c(-4,4
plot(x4,y4,main=paste(" IV, Corr. Coff.= ", round(cor.test(x4,y4)$estimate,3)),xlim=c(-4,4),ylim=c(-4,4
```



Run Chart

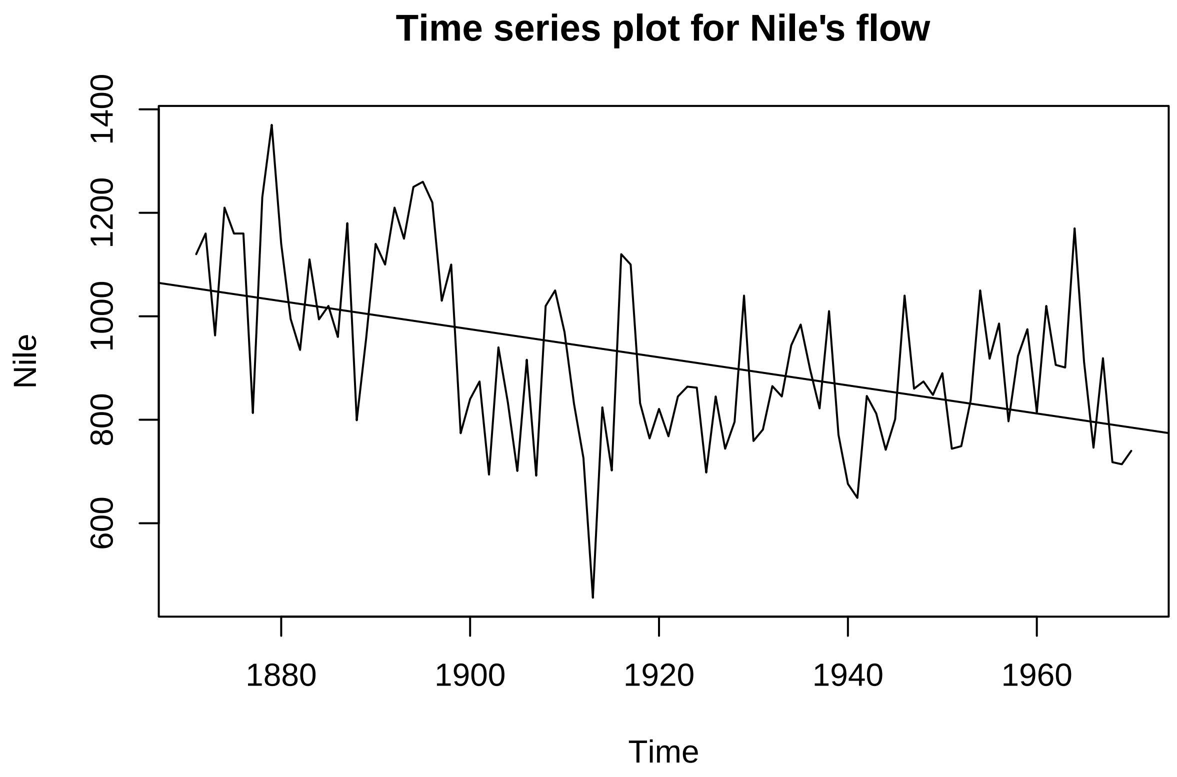
- A Run Chart is a time series plot (also known as a run-sequence plot) that illustrates shifts and trends in data over time. It can display both continuous and discrete data and typically includes a median or average line.
- Run charts can be plotted in R using the plot.ts function, which is commonly used in time-series analysis.
- Plot the measurements of the annual flow of the river Nile at Aswan (formerly Assuan), 1871–1970, in $10^8 m^3$ as given in Nile dataset, and add the linear trending line.

```
data(Nile)
# Show the first 10 values of the data
Nile[1:10]
```

Example

```
##  [1] 1120 1160  963 1210 1160 1160  813 1230 1370 1140
```

```
# Plot the Nile dataset as a time series
plot.ts(Nile)

# Add a title to the plot
title("Time series plot for Nile's flow")

# Add a regression line (best-fit straight line) over the time series
abline(reg = lm(Nile ~ time(Nile)))
```



-The run chart shows that there is a slight decreasing trend in annual flow measurements .

- It is evident that there is a noticeable change point around 1898, with a significant drop in Nile flow occurring after 1910, reaching its minimum value in 1913.
- This shift could be attributed to a combination of climatic changes, such as reduced rainfall in the Ethiopian highlands, and other environmental factors affecting the river's flow.