Task 1:

In [ ]:
```
# Installing the required packages and importing the required Libraries to perform the given task
install.packages("tidyverse")
install.packages("data.table")
install.packages("dplyr")
install.packages("modeest")
library(modeest)
library(tidyr)
library(ggplot2)
library(dplyr)
library(stringr)
```

In [ ]:
```
tidyr::who # Importing the who dataset
```

In [3]:
```
# 1.
who1 <- who %>% pivot_longer(
    cols = starts_with("new"),   # All the Columns that starts with "new"
    names_to = "Key",            # New column named "Key" for all the columns gathered
    values_to = "Cases",         # New Column named "Cases" for all values
    values_drop_na = TRUE        # Drops rows having missing values
    )
```

By using the above code, all the columns starting with "new" prefix is gathered as a value into new column "Key" and their respective values is gathered into column "Cases" with any missing values being droped from the dataset.

In [4]:
```
# 2.
who1$Key <- str_replace(who1$Key,'newrel','new_rel') #Replaces the String "newrel" with "new_rel"
who2 <- who1
```

In [5]:
```
# 3.
who3 <- who2 %>% separate(Key, c("new", "type", "sexage"), sep = "_")
```

The code above uses the separate function to split the value of a column(i.e. "Key") and respective rows into three values in three different columns(i.e. "new", "type", "sexage") whenever it encounters a Underscore in the value of column "Key" for a respective row in a data frame. "%>%" known as the "pipe" operator will provide a value or an expression's outcome from the left side of the operator to the following function call or expression on the right side of the operator.

In [6]:
```
# 4.
who4 <- who3 %>% separate(sexage, into = c("sex", "age"), sep =1)
```

The above code uses separate function to split the column "sexage" into two columns "sex" and "age" by interpretting the integer "1" as a position to split at from the far-left of the strings.

In [7]:
```
# 5.
head(who4, 5)  #Prints the first 5 rows from the dataset who4
tail(who4, 5)  #Prints the last 5 rows from the dataset who4
```

A tibble: 5 × 9

country	iso2	iso3	year	new	type	sex	age	Cases
<chr>	<chr>	<chr>	<int>	<chr>	<chr>	<chr>	<chr>	<int>
Afghanistan	AF	AFG	1997	new	sp	m	014	0
Afghanistan	AF	AFG	1997	new	sp	m	1524	10
Afghanistan	AF	AFG	1997	new	sp	m	2534	6
Afghanistan	AF	AFG	1997	new	sp	m	3544	3
Afghanistan	AF	AFG	1997	new	sp	m	4554	5

A tibble: 5 × 9

country	iso2	iso3	year	new	type	sex	age	Cases
<chr>	<chr>	<chr>	<int>	<chr>	<chr>	<chr>	<chr>	<int>
Zimbabwe	ZW	ZWE	2013	new	rel	f	2534	4649
Zimbabwe	ZW	ZWE	2013	new	rel	f	3544	3526
Zimbabwe	ZW	ZWE	2013	new	rel	f	4554	1453
Zimbabwe	ZW	ZWE	2013	new	rel	f	5564	811
Zimbabwe	ZW	ZWE	2013	new	rel	f	65	725

In [8]:
```
# 6.
# Create the csv file and saved it in my local directory
write.csv(who4,"who4.csv", row.names = FALSE)
```

Task 2:

In [9]:
```
data("Nile") # loads the Nile data set in R
```

In [10]:
```
# 1.
Mean = mean(Nile)                    # Computes the mean value
paste("Mean = ", Mean)

Median = median(Nile)                # Computes the median value
paste("Median = ",Median)

Mode = mfv(Nile)                     # Computes the mode value
cat("Mode = ",Mode,"\n")

Variance =  var(Nile)                # Computes the Variance
paste("Variance = ",round(Variance,2))

SD =  sd(Nile)                       # Computes the Standard Deviation
paste("Standard Deviation = ",round(SD, 2))
```

'Mean = 919.35'

'Median = 893.5'

```
Mode =  845 1020 1100 1160
```

'Variance = 28637.95'

'Standard Deviation = 169.23'

In [11]:
```
# 2.
Min = min(Nile)            # Computes the minimum value
Max = max(Nile)            # Computes the maximum value
Range = Max - Min          # Computes the range
paste("Min = ", Min)
paste("Max = ", Max)
paste("Range = ", Range)
```

'Min = 456'

'Max = 1370'

'Range = 914'

In [12]:
```
# 3.
Interquartile_range = IQR(Nile)   # Calculating Interquartile range
Quartiles = quantile(Nile)        # Calculating Quartiles
paste("Interquartile Range = ",Interquartile_range)
paste("Quartiles = ")
print(Quartiles)
```

'Interquartile Range = 234'

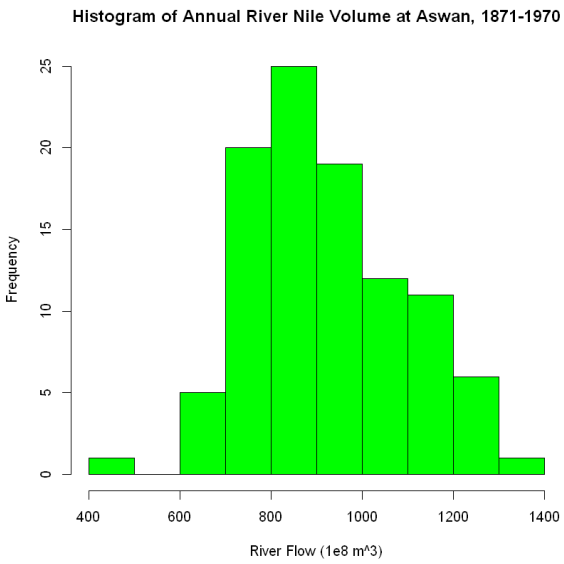
'Quartiles = '

```
    0%    25%    50%    75%   100%
 456.0  798.5  893.5 1032.5 1370.0
```

Difference between the third quartile (75th percentile) and the first quartile (25th percentile) is calculated by the interquartile range (IQR) function. IQR contains 50% of the observations and covers the middle of the distribution.

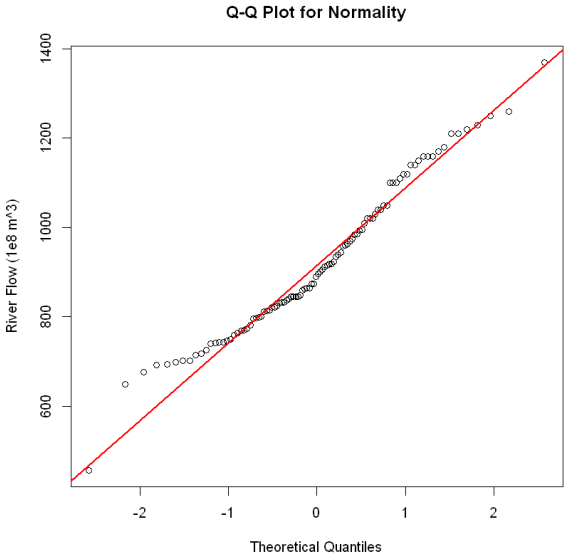
Quantile function shows the sample quantiles within the dataset. The first quartile shows the value that falls in the middle of the data set's smallest value and median, the second quartile (50th percentile) known as the data set's median, and the third quartile indicates the value that falls in the middle of the median and largest value in a data set.

In [13]:
```
# 4.
hist(Nile, main="Histogram of Annual River Nile Volume at Aswan, 1871-1970",
     xlab="River Flow (1e8 m^3)", ylab="Frequency", col="green")
```



The above histogram shows the distribution of the data for a given dataset. Above histogram plots the data related to Annual River Nile Volume at Aswan, 1871-1970. There are cells also known as bins with evenly spaced breaks, as seen above. In this instance, a cell's height is determined by how many observations fall into it. The frequency of the data is shown on the y-axis, while the annual river flow is shown on the x-axis. As noticed from the above histogram that the maximum no of observations related to annual flow of the river Nile has been in between 800(1e8 m^3) to 900(1e8 m^3).

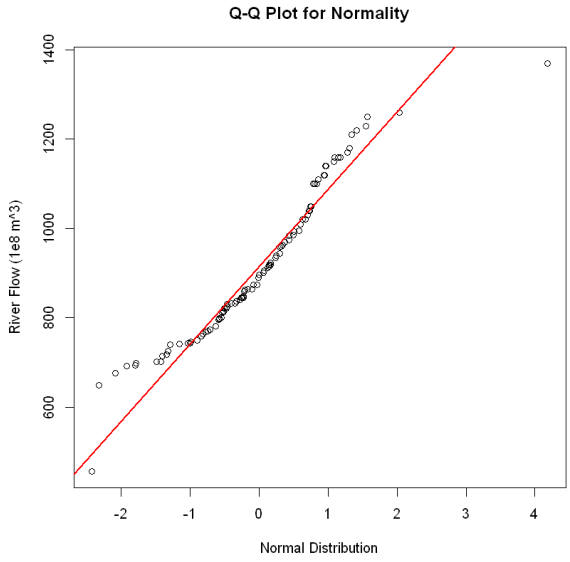
In [14]:
```
# 5.
# creating Q-Q plot to compare the dataset to a theoretical normal distribution
qqnorm(Nile, main = 'Q-Q Plot for Normality', ylab="River Flow (1e8 m^3)")
qqline(Nile, col = "red", lwd = 2)   # Adding reference line to the plot
```



qqnorm function creates a quantile-quantile plot. It uses a data vector as input and shows the sorted data against quantiles from a standard Normal distribution. In the above chart, the theoretical quantiles from the standard Normal distribution with a mean of 0 and a standard deviation of 1, are shown on the x-axis. y-axis indicates the annual river flow. The plotted points don't appear to fall along a straight line, indicating that the data is not distributed normally.

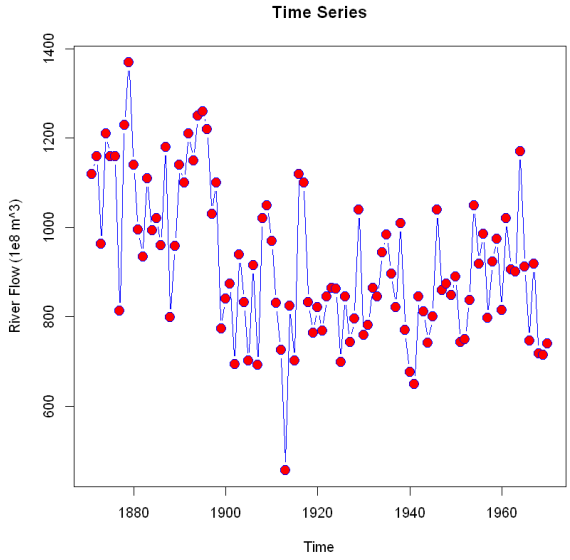
In [15]:
```
x <- rnorm(100) # Generating vector of 100 values following a normal distribution

# Creating Q-Q plot to compare the dataset to a random values that follow a normal distribution
qqplot(x, Nile, main = 'Q-Q Plot for Normality', xlab = "Normal Distribution", ylab="River Flow (1e8 m^3)")
qqline(Nile, col = "red", lwd = 2)
```



The above code creates a vector of 100 randomly selected values that follow a normal distribution and creates a Q-Q plot to compare the dataset to the selected random values to determine whether it follows a normal distribution or not. Once more, the Q-Q plot reveals points that do not fall in a straight line, which is compelling evidence that the dataset in question probably does not follow a normal distribution.

In [16]:
```
# 6.
plot(Nile,main = 'Time Series', ylab="River Flow (1e8 m^3)", type="b", col = "blue",pch = 21,
     bg = "red", cex = 1.5,lwd = 1 )
```



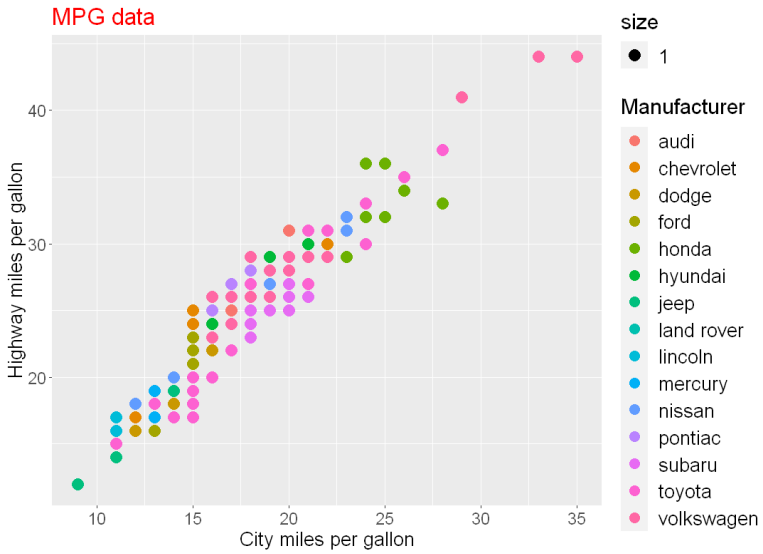
The plot function shows the resulting time series plot of Nile dataset. As can be seen, the plot highlights a number of intriguing characteristics. Firstly, an unusually high flow was recorded around 1880. Secondly, an exceptionally low flow was observed shortly after 1910. The plot also demonstrates the general decreasing trend, which shows that from 1910 to 1970, less water flowed than it did from 1870 to 1910.

Task 3:

In [17]:
```
mpg = ggplot2::mpg
```

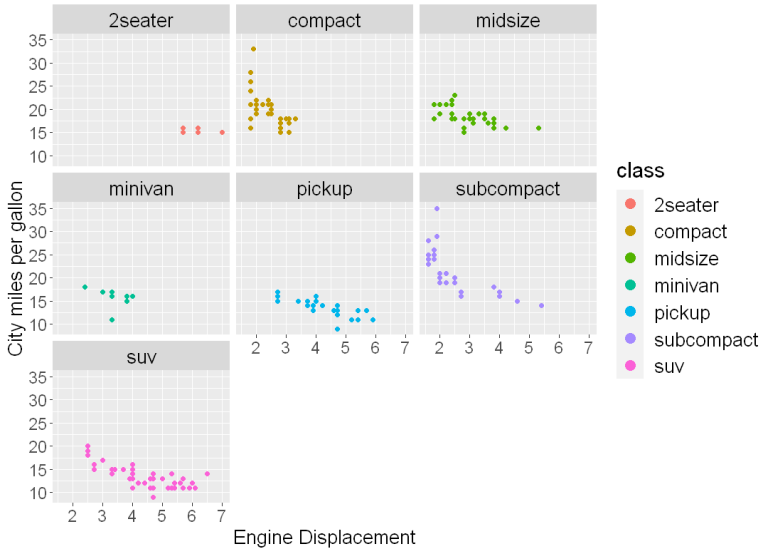
In [18]:
```
# 1.
options(repr.plot.width = 9, repr.plot.height =6.5) # changes the plot size

ggplot(data = mpg) + geom_point(mapping = aes(x = cty, y = hwy, color = manufacturer, size = 1)) +
labs(x = "City miles per gallon",y = "Highway miles per gallon", title = "MPG data",color = "Manufacturer") +
theme(plot.title = element_text(size = 20, color = "red"),legend.key.size = unit(0.8, 'cm'),
legend.title = element_text(size=18), legend.text = element_text(size=16),
axis.title.x = element_text(size = 16), axis.title.y = element_text(size = 16),
axis.title = element_text(size = 16), axis.text = element_text(size = 14)) +
guides(color = guide_legend(override.aes = list(size = 4)))
```



Above scatter plot indicates the points based on the Highway mpg(Miles per gallon) and the city mpg(Miles per gallon) for a respective manufacturers. x-axis represents the city mpg and y-axis represents the highway mpg data. The scatter plot shows that, when compared to other manufacturers, Volkswagen gets the highest mpg in both the city and the highway (35 and 44 respectively).

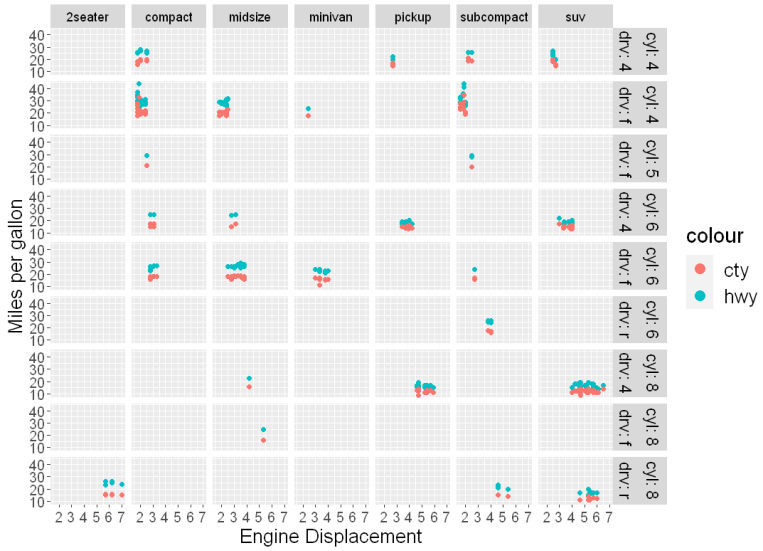
In [19]:
```
# 2.
mpg_displ <- ggplot(mpg, aes(displ,cty,color=class)) +
theme(legend.key.size = unit(0.8, 'cm'), legend.title = element_text(size=18),
legend.text = element_text(size=16), axis.title.x = element_text(size = 16),
axis.title.y = element_text(size = 16), axis.title = element_text(size = 16),
axis.text = element_text(size = 14), strip.text.x = element_text(size = 16,
colour = "black", angle = 360)) + guides(color = guide_legend(override.aes = list(size = 4)))
mpg_displ + facet_wrap(~ class, , scales = "fixed") +
geom_point() + labs(x = "Engine Displacement",y = "City miles per gallon")
```



The displacement range is shown on the x-axis in the above plot, while the city mpg is shown on the y-axis. Additionally, the data is classified using the facet wrap function according to several vehicle types. The subplots created by facets each show a different subset of the data in the main plot. The above scatter plots in general shows how city mpg(Miles per gallon) decreases with increase in engine dispacement of vehicles(size of engine). It can also be observed from the above plot that compared to other vehicle types, "SUV" and "pickup" have the lowest mpg ratings, having City mpg of 9 and Engine Displacement of 4.7.

In [20]:
```
# 3.
ggplot(data=mpg)+
geom_point(aes(x = displ , y = hwy,color = "hwy")) +
geom_point( aes(x = displ, y = cty,color = "cty")) +
facet_grid(cyl ~ drv~ class, labeller = labeller(.rows = label_both) ) +
labs(x = "Engine Displacement",y = "Miles per gallon") +
theme(legend.key.size = unit(0.8, 'cm'),legend.title = element_text(size=18),
legend.text = element_text(size=16),axis.title.x = element_text(size = 16),
axis.title.y = element_text(size = 16), axis.title = element_text(size = 16),
axis.text = element_text(size = 12),
strip.text.x = element_text(size = 11, colour = "black", angle = 360),
```

```
strip.text.y = element_text(size = 14, colour = "black", angle = 270)) +
guides(color = guide_legend(override.aes = list(size = 4)))
```



The above scatter plot indicates the points based on the Highway mpg(Miles per gallon) and the city mpg(Miles per gallon) against the Engine Displacement for different vehicle types. The plot is further divided into categories based on the number of cylinders and the type of drive. The x-axis represents the displacement range and y -axis represents the mpg performance. Orange and the blue points in the plot indicates the city mpg and highway mpg respectively based on Engine Displacement for different vehicle types,considering the no. of cylinders and the drive type as well.

It can be observed from the above plot that Compact, subcompact vehicle type with 4 no of cylinders and front wheel drive having engine displacement of 1.9 each has the highest mpg in both highway and city. It can also be observed from the plot that for a higher litter engine having a good highway mpg, best option is to choose the 2seater with 8 no of cylinders and rear-wheel drive having a 7 diplacement range and highway performace around 20-25 mpg.