STAT 2670: Elementary Statistics with R

Fall 2023

Jerome Goddard II

2023-08-18

Contents

2 Exploring Data with Tables and Graphs

2.1 Frequency distributions for organizing and summarizing data

A frequency distribution is a summary table or graph that shows the count or frequency of each unique value or category in a dataset, providing a clear picture of how data is distributed across different values or groups.

2.1.1 Frequency distributions

The R command `table()` will generate a frequency distribution for any data set. Let's analyze example test scores from a fictional math class. Notice the first row of the output is the data name, the second row is the actual data, and the third row contains the number of times each data value appears.

```r
# Load test data into a variable names scores
scores = c(95, 90, 85, 85, 87, 74, 75, 64, 85, 84, 87, 15, 20, 75, 75, 90, 75)

# Create a frequency table for the scores data
table(scores)
```

```
## scores
## 15 20 64 74 75 84 85 87 90 95
##  1  1  1  1  4  1  3  2  2  1
```

2.1.2 Relative frequency distributions

Relative frequency distributions give similar information as a frequency distribution except they use percentages. Let's examine the same `scores` data set defined above. This code will give relative frequency rounded to the nearest whole number. Notice in the output that the second row is the actual data and the third row contains the relative frequencies (rounded to two decimal places).

```r
# Create a relative frequency table for the scores data
rftable = table(scores)/length(scores)
round(rftable, digits = 2)
```

```
## scores
##   15   20   64   74   75   84   85   87   90   95
## 0.06 0.06 0.06 0.06 0.24 0.06 0.18 0.12 0.12 0.06
```

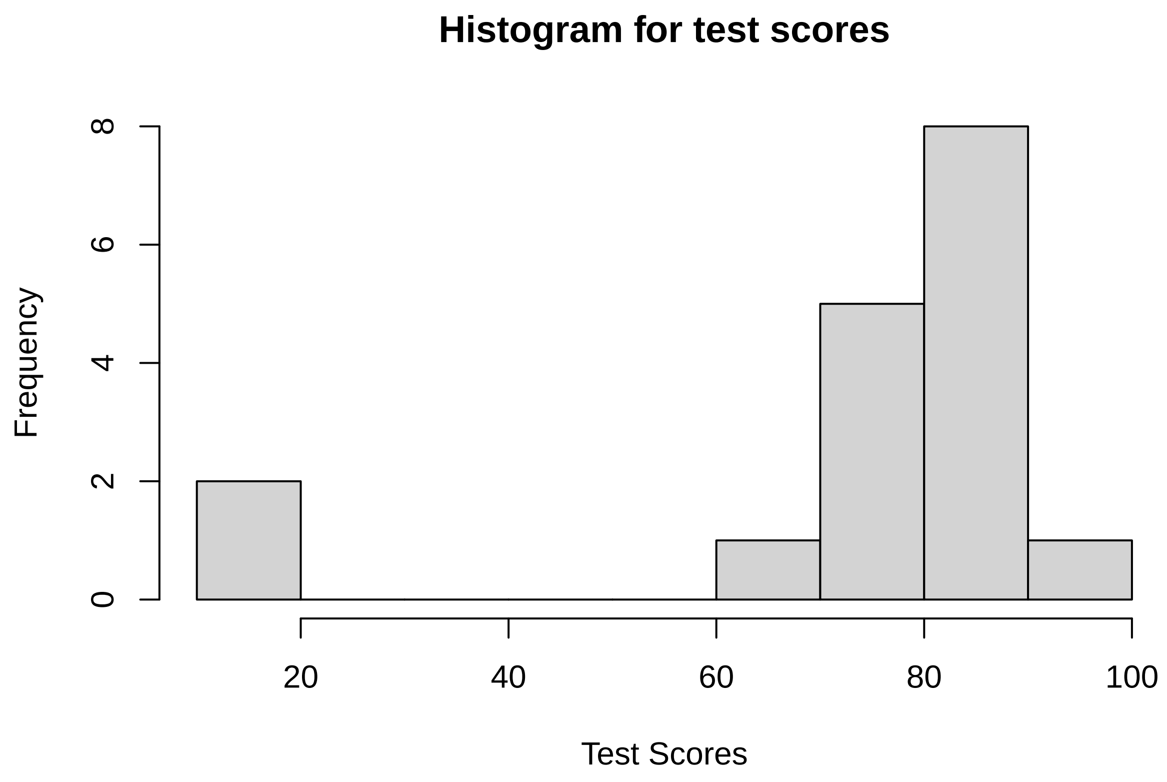
2.2 Histograms

A histogram is a bar chart that shows how often different values occur in a dataset.

2.2.1 Histogram

The command `hist()` will generate a histogram for any data. Here is an example using our `scores` data from above. Notice the x-axis represents the actual scores and the y-axis shows the frequency of the data points. We will use the following command options: 1) `main` allows the title to be specified, 2) `xlab` sets the x-axis label, and 3) `ylab` sets the y-axis label.

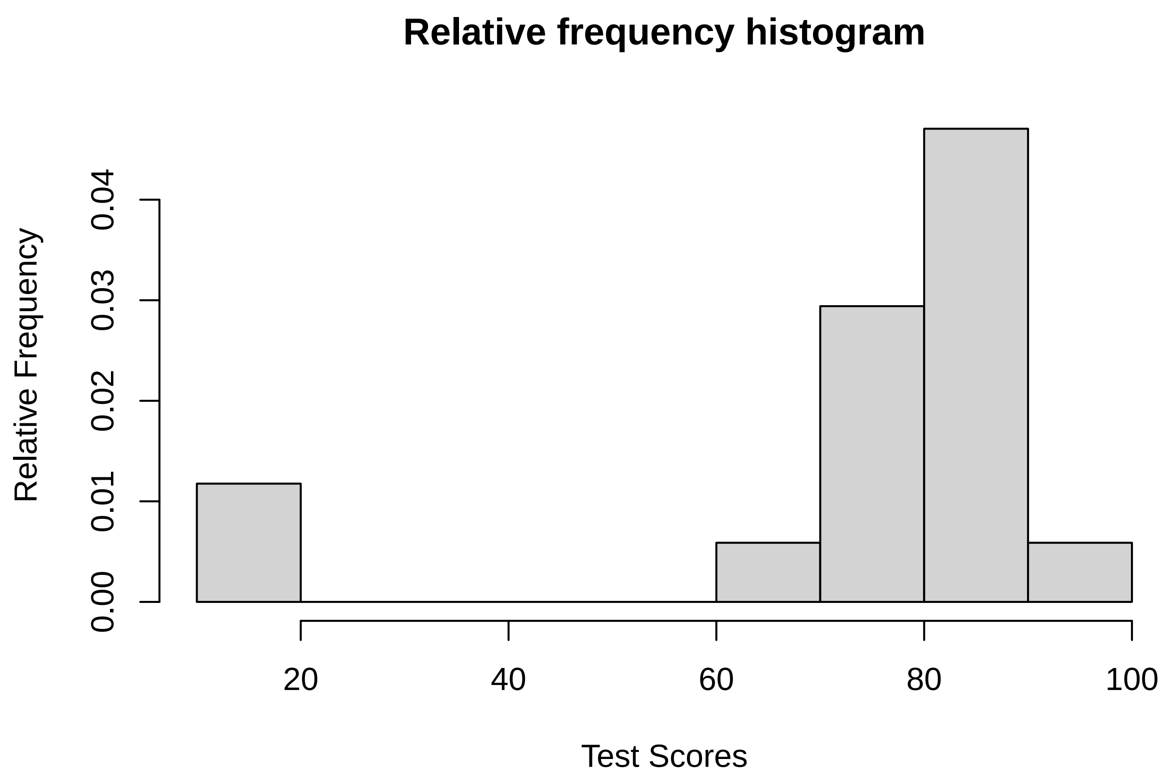
```r
# Create a histogram and customize the axis labels and title main is the
# Plot title, xlab is the x-axis label, & ylab is the y-axis label
hist(scores, main = "Histogram for test scores", xlab = "Test Scores", ylab = "Frequency")
```



2.2.2 Relative frequency histogram

A relative histogram is a bar chart that displays the proportion or percentage of values in different bins within a dataset, providing a relative view of the data distribution.

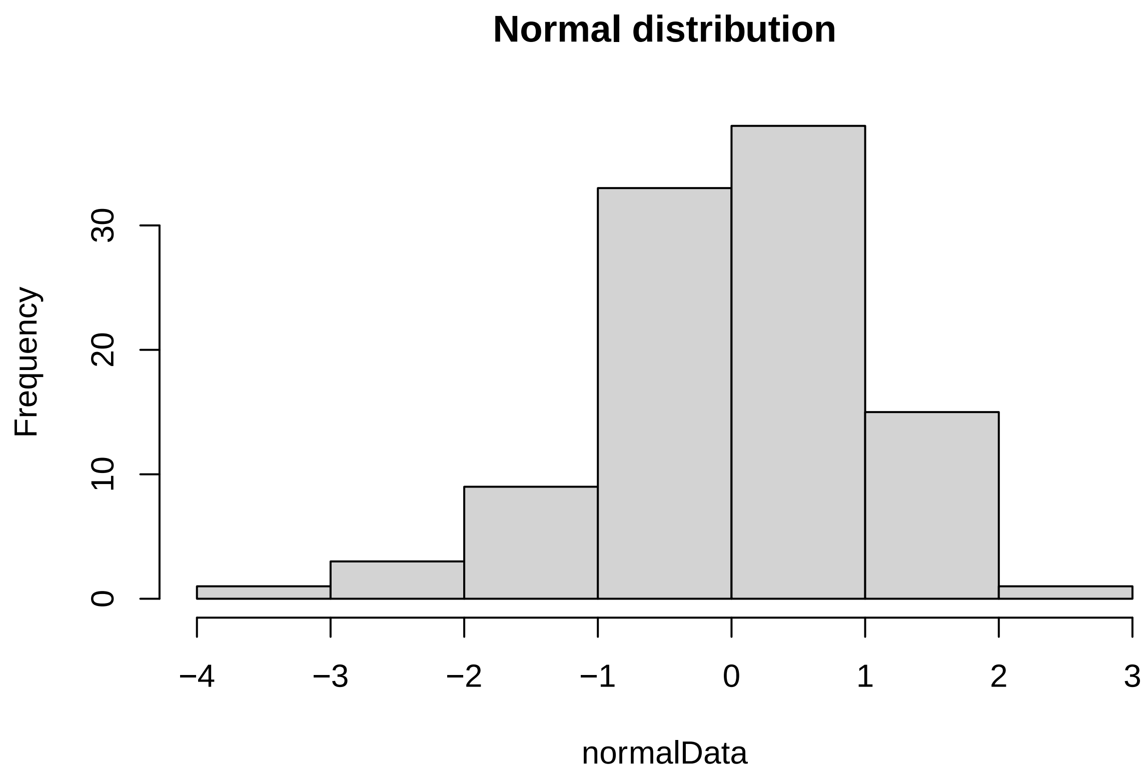
```r
# Using freq = FALSE in hist() will create a relative frequency histogram
hist(scores, freq = FALSE, main = "Relative frequency histogram", xlab = "Test Scores",
    ylab = "Relative Frequency")
```



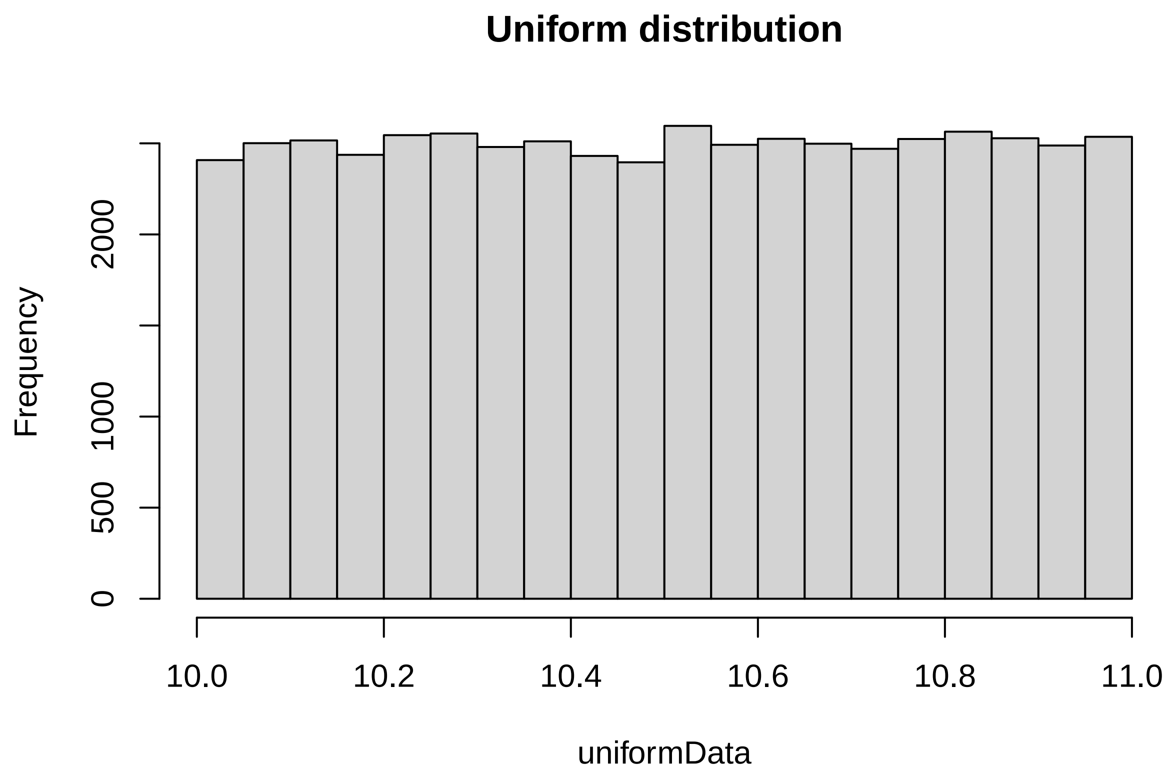
2.2.3 Common distributions

Normal distributions are bell-shaped and symmetrical, uniform distributions have constant probabilities across a range, skewed right distributions are characterized by a long tail on the right side, and skewed left distributions have a long tail on the left side, each exhibiting distinct patterns of data distribution. We will use the `hist()` command to explore each of these common distributions in the code below.

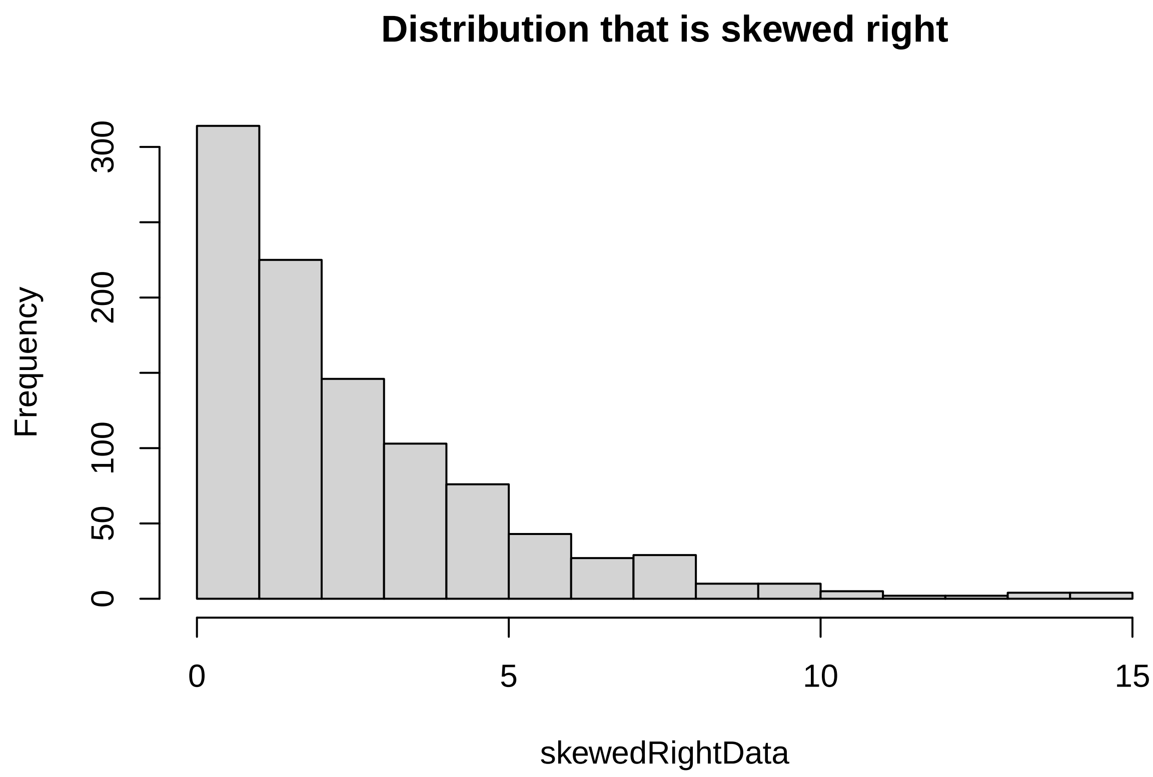
```r
# Sample normal distribution
normalData = rnorm(100)

# Sample uniform distribution using the command runif
uniformData = runif(50000, min = 10, max = 11)

# Sample of a distribution that is skewed right
skewedRightData = rexp(1000, 0.4)

# Sample of a distribution that is skewed left
skewedLeftData = 1 - rexp(1000, 0.2)

# Create histogram of normal data
hist(normalData, main = "Normal distribution")
```



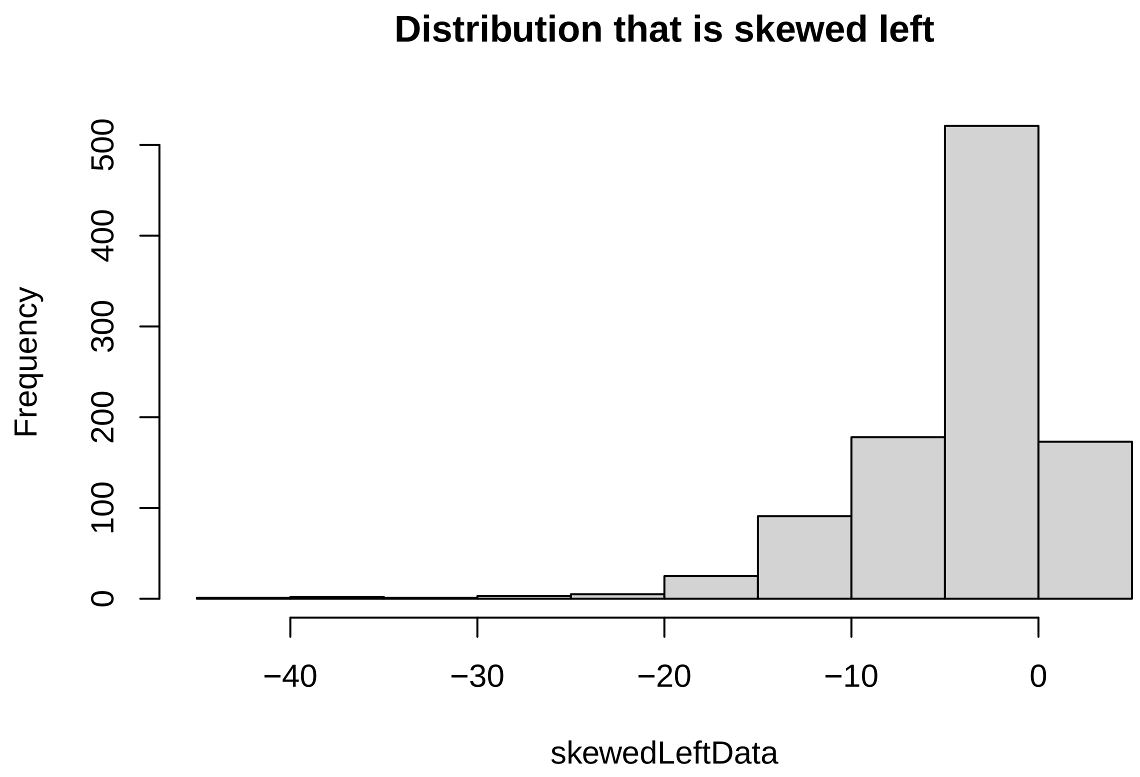
```
# Create histogram of uniform data
hist(uniformData, main = "Uniform distribution")
```



```r
# Create histogram of skewed right data
hist(skewedRightData, main = "Distribution that is skewed right")
```



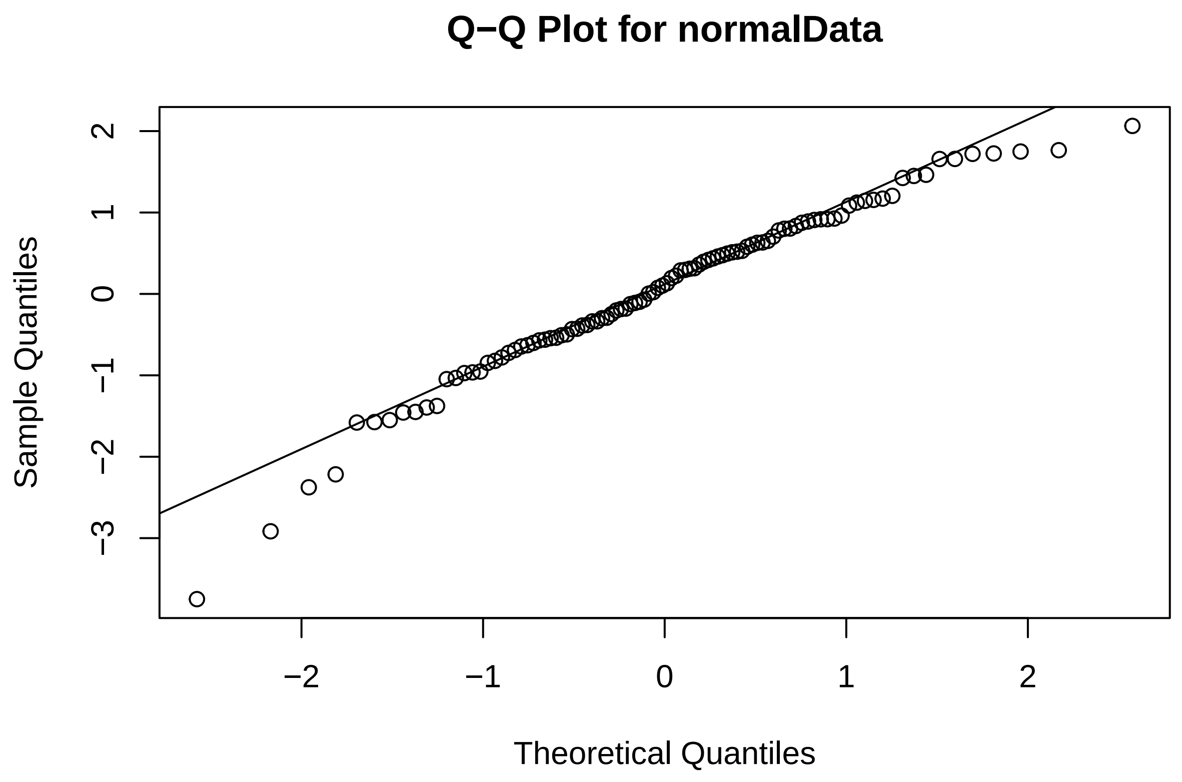
```r
# Create histogram of skewed left data
hist(skewedLeftData, main = "Distribution that is skewed left")
```



2.2.4 Normal quantile plots

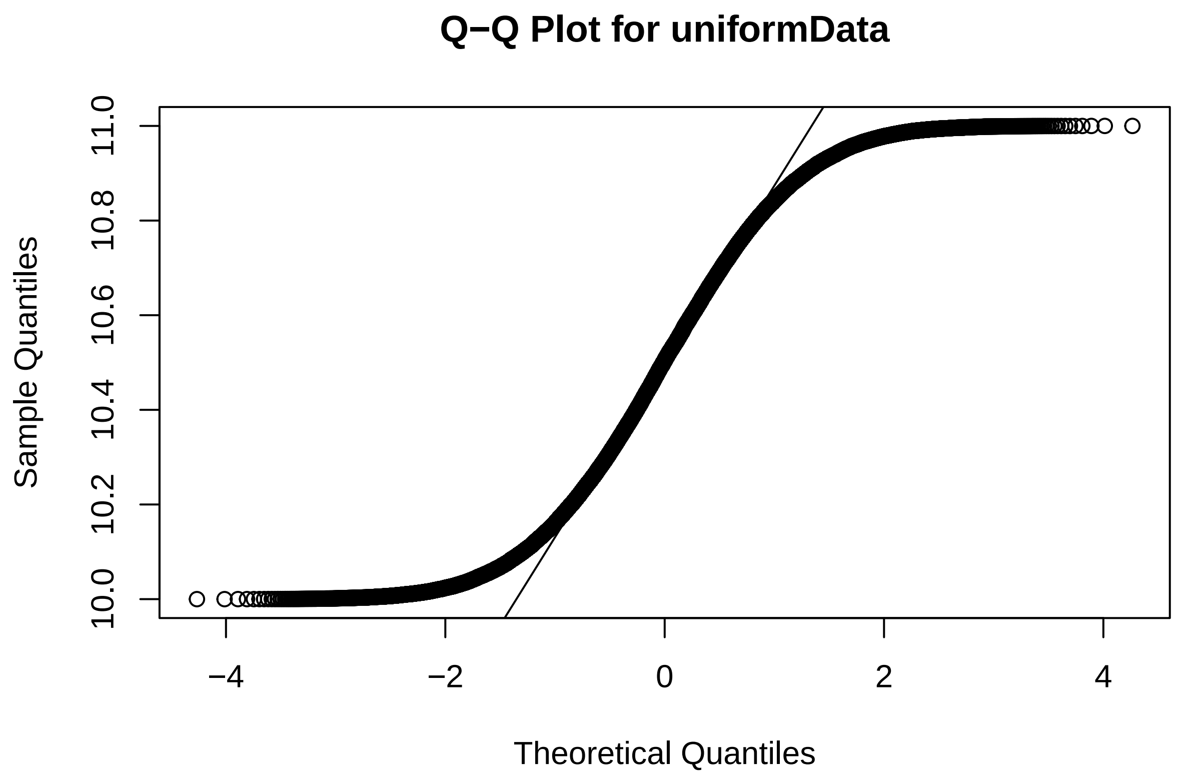
A normal quantile plot, also known as a Q-Q plot, is a graphical tool used to assess whether a dataset follows a normal distribution by comparing its quantiles (ordered values) to the quantiles of a theoretical normal distribution; if the points closely follow a straight line, the data is approximately normal. Let's use the commands `qqnorm()` and `qqline()` to visually test which data set is most likely a sample from a normal distribution.

```r
# Test normalData from above
qqnorm(normalData, main = "Q-Q Plot for normalData")
qqline(normalData)
```



Notice that the `normalData` Q-Q plot shows the points close to the Q-Q line over the entire x-axis.

```r
# Test uniformData from above
qqnorm(uniformData, main = "Q-Q Plot for uniformData")
qqline(uniformData)
```

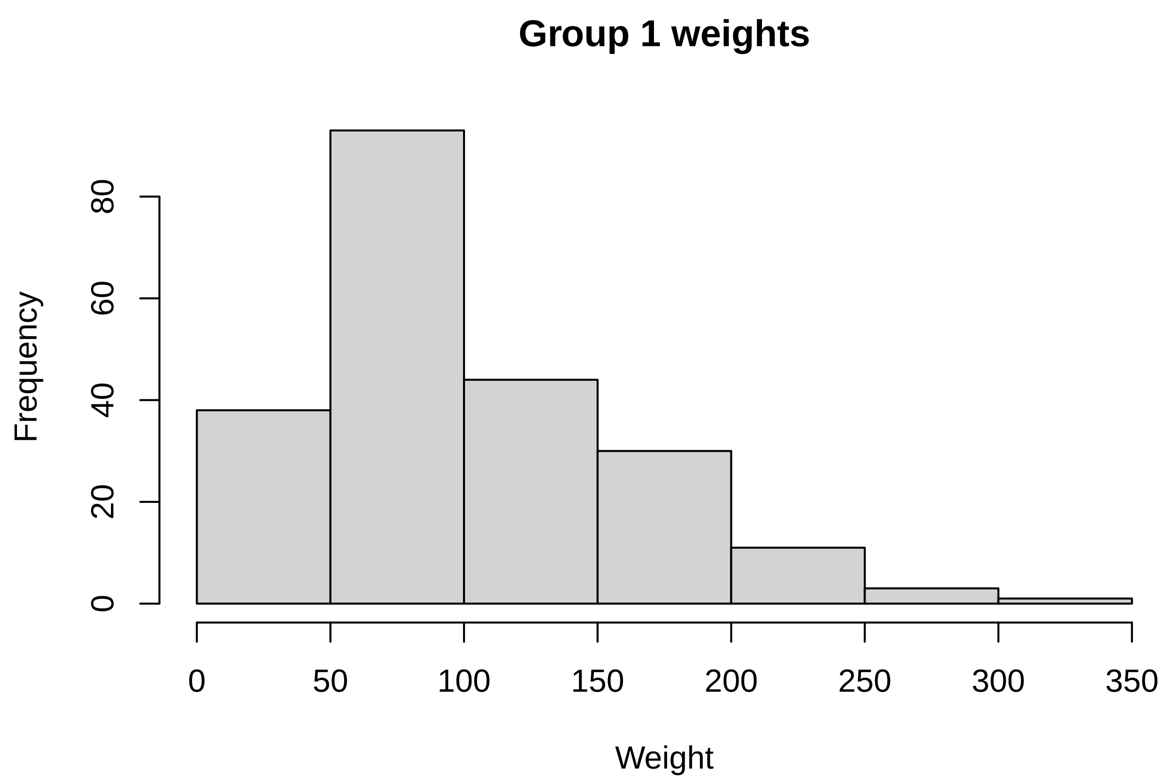


For the `uniformData` dataset, the Q-Q plot shows good agreement between points and line in the center (around 0) but not on either left or right of the x-axis.

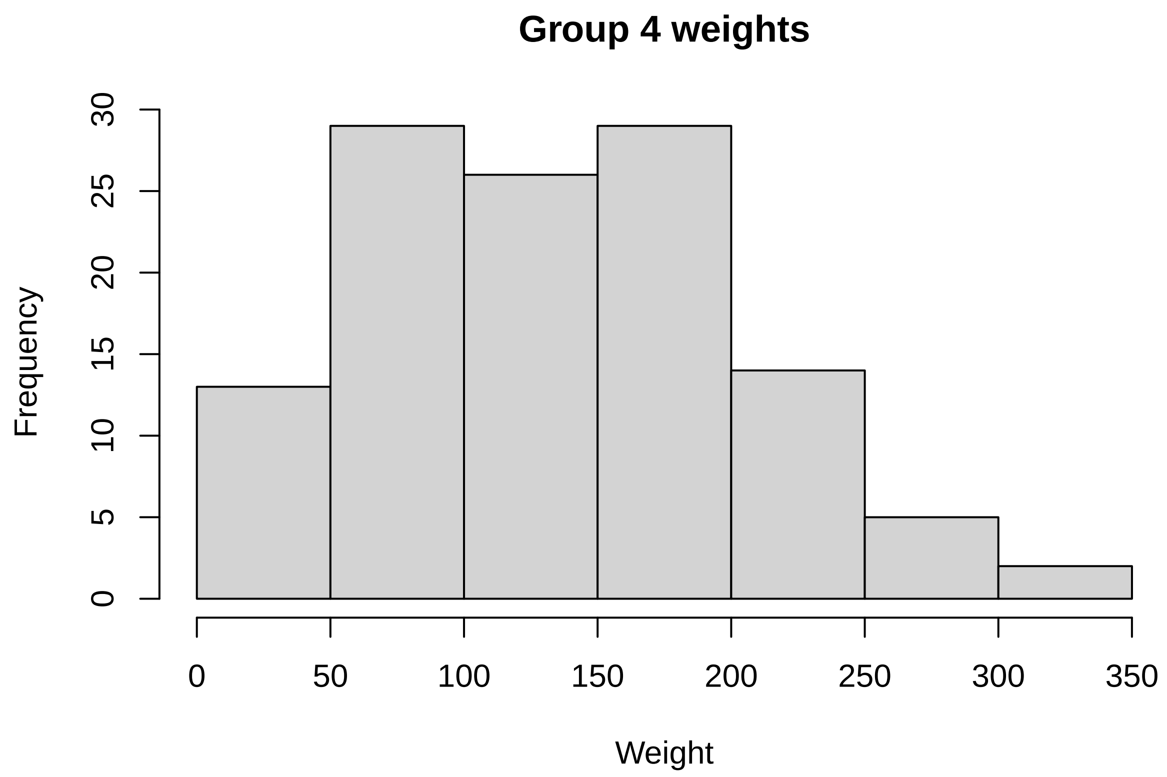
2.2.5 Let's put it all together!

In the built-in R dataset `ChickWeight`, weights are taken from several groups of chickens that were fed various diets. We are asked to use both histogram and Q-Q plots to determine if weights from group 1 and 4 are approximately normal, uniform, skewed left, or skewed right.

```r
# Load data from the built-in dataset into a variable named ChickWeight
data("ChickWeight")

# Extract all weights from group 1
group1Weights = ChickWeight[ChickWeight$Diet == 1, 1]

# Extract all weights from group 4
group4Weights = ChickWeight[ChickWeight$Diet == 4, 1]

# Create a histogram of weights from group 1
hist(group1Weights, main = "Group 1 weights", xlab = "Weight", ylab = "Frequency")
```

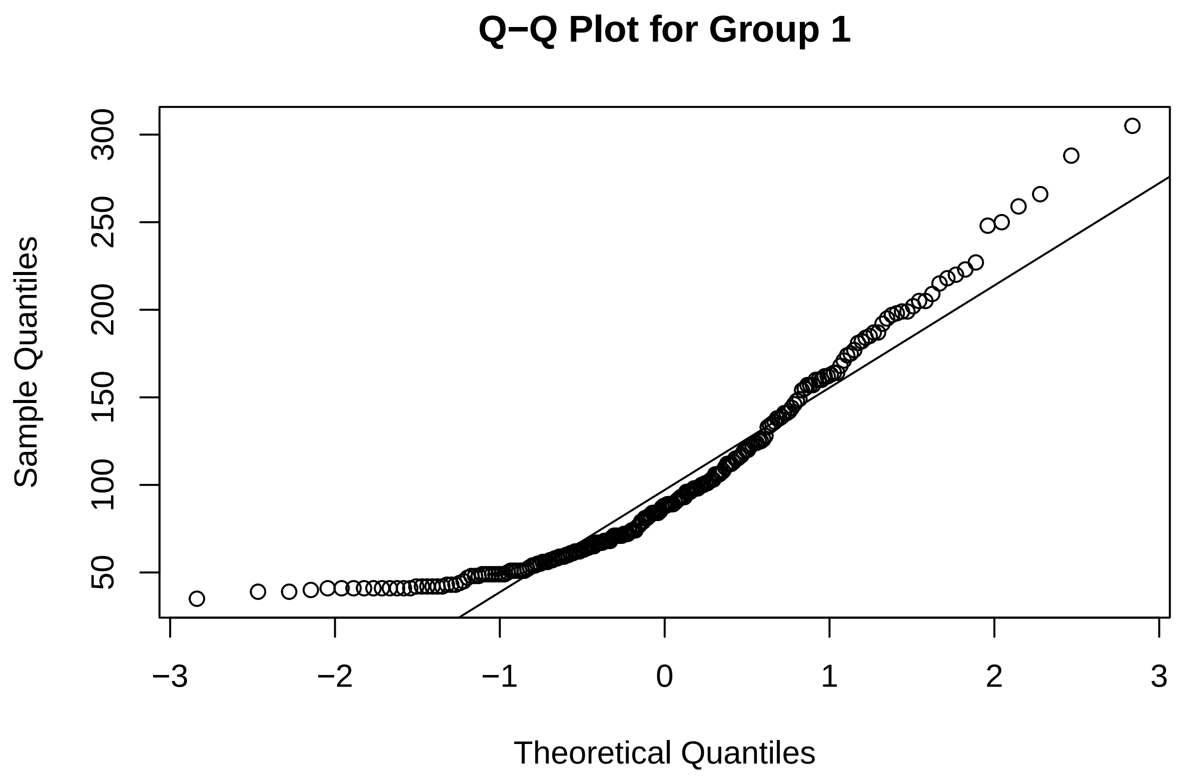


```r
# Create a histogram of weights from group 4
hist(group4Weights, main = "Group 4 weights", xlab = "Weight", ylab = "Frequency")
```

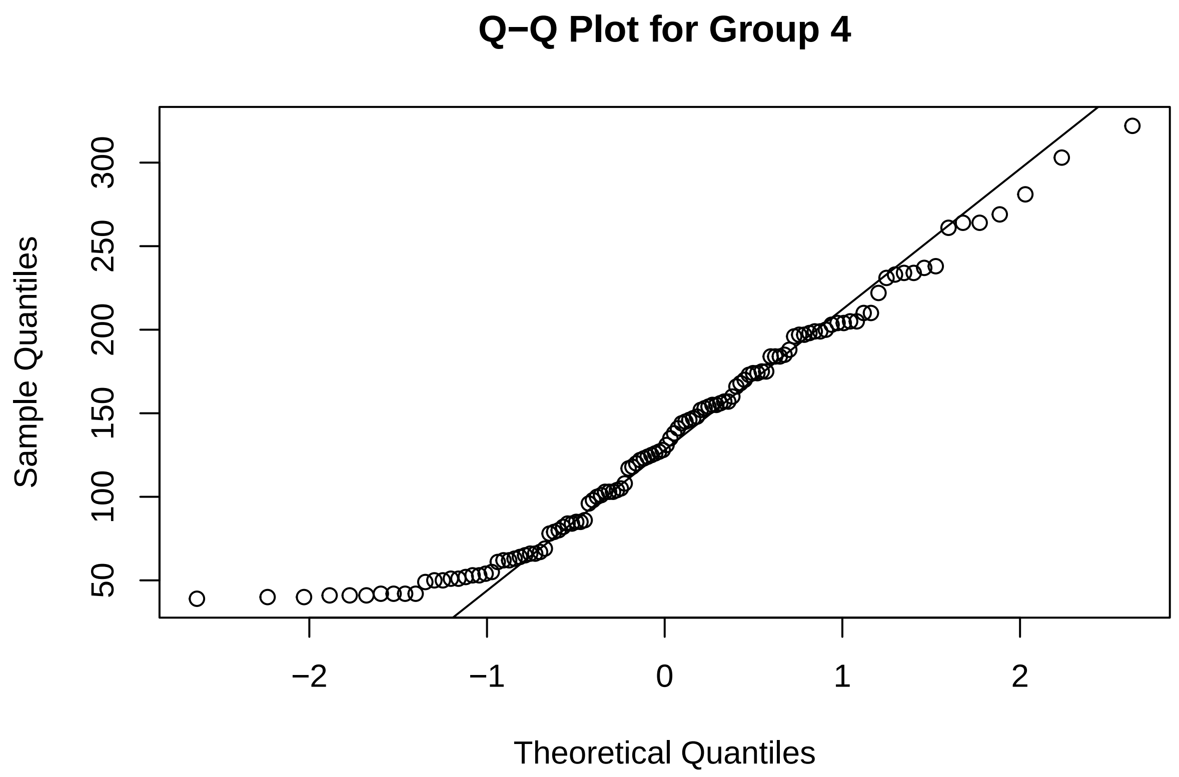


Is the group 1 distribution approximately normal or would a different distribution be a better fit? What about group 4? Now, let's confirm our results using Q-Q plots.

```r
# Test group1Weights from above
qqnorm(group1Weights, main = "Q-Q Plot for Group 1")
qqline(group1Weights)
```



```r
# Test group4Weights from above
qqnorm(group4Weights, main = "Q-Q Plot for Group 4")
qqline(group4Weights)
```



Does the Q-Q plot confirm your guess from our visual inspection? Which group is closer to a normal distribution?

2.3 Graphs that enlighten and graphs that deceive

R has many commands to illustrate data revealing hidden patterns that could be otherwise missed. We will explore several of these commands using three different datasets:

(A) **Chicken Weights:** Same data used in Section 2.2: two different groups of chickens fed with different feed.

(B) **Airline Passengers:** A time series of the number of airline passengers in the US by month.

(C) **US Personal Expenditure** Average personal expenditures for adults in the US from 1960.

This block of code will load this data.

```
# Chicken weights: Load data from the built-in dataset into a variable
# named ChickWeight
data("ChickWeight")

# Extract all weights from group 1
group1Weights = ChickWeight[ChickWeight$Diet == 1, 1]

# Extract all weights from group 4
group4Weights = ChickWeight[ChickWeight$Diet == 4, 1]

# Airline passengers: Load from the built-in dataset.  This will create a
# variable named AirPassengers containing the time series.
```

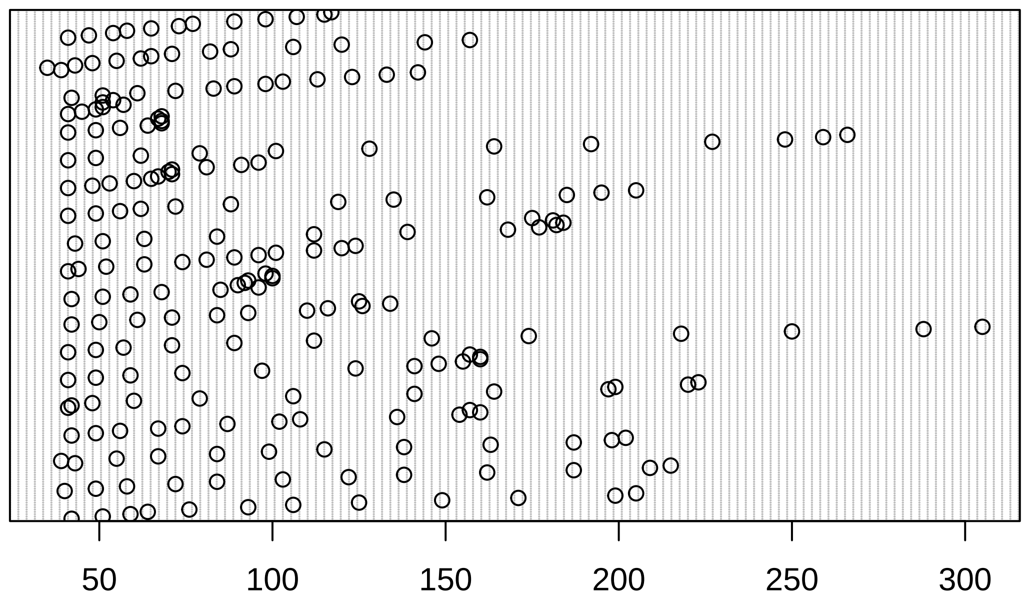
```r
data("AirPassengers")

# Personal expenditure: Load from the built-in dataset.  This will create
# a variable named USPersonalExpenditure containing the data.
data("USPersonalExpenditure")

# We now extract only information from 1940
expenditures1940 = USPersonalExpenditure[1:5]

# We now extract only information from 1960
expenditures1960 = USPersonalExpenditure[21:25]

# Define categories for expenditure data
cats = c("Food and Tobacco", "Household Operation", "Medical and Health", "Personal Care",
    "Private Education")

# Define category names from cats above
names(expenditures1940) = cats
names(expenditures1960) = cats
```

2.3.1 Dotplot

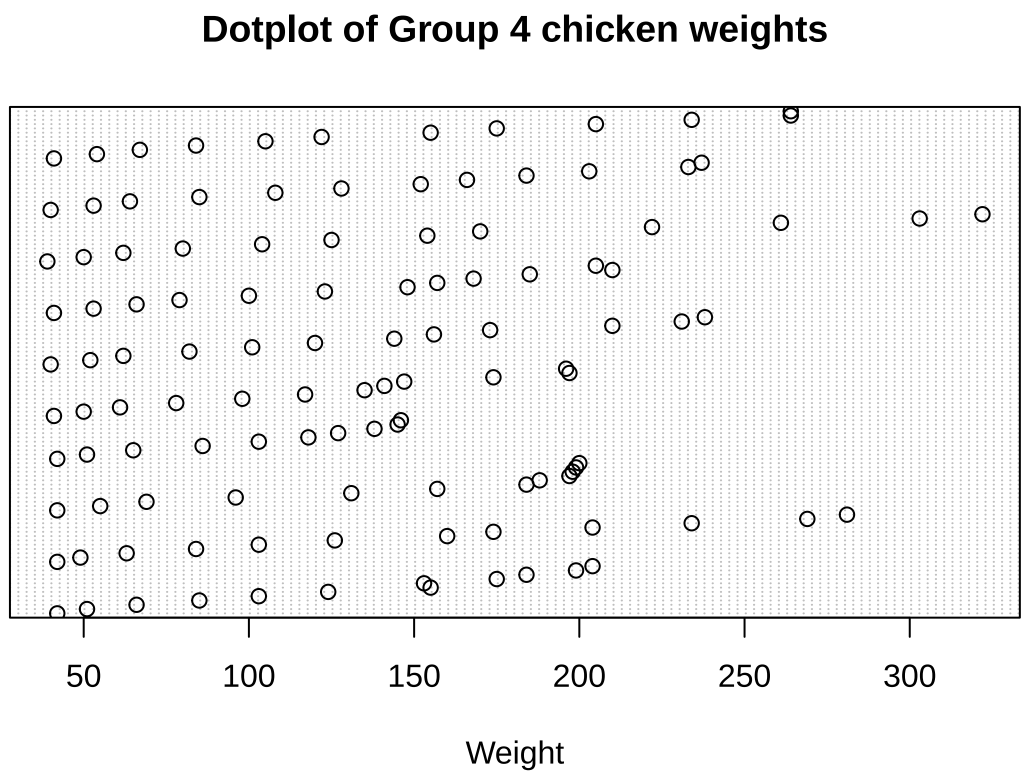
A dotplot is a simple graphical representation of data in which each data point is shown as a dot above its corresponding value on a number line, helping to visualize the distribution and identify patterns in a dataset. With our data previously loaded from the previous run, let's create a dotplot of the data. First for weights of both groups of chickens.

```r
# Dotplot for group 1 chickens
dotchart(group1Weights, main = "Dotplot of Group 1 chicken weights", xlab = "Weight")
```

Dotplot of Group 1 chicken weights



```
# Dotplot for group 4 chickens
dotchart(group4Weights, main = "Dotplot of Group 4 chicken weights", xlab = "Weight")
```



2.3.2 Stem plot

A stem plot, also known as a stem-and-leaf plot (or just stemplot), is a graphical representation of data where each data point is split into a "stem" (the leading digit or digits) and "leaves" (the trailing digits) to display the individual values in a dataset while preserving their relative order, making it easier to see the distribution and identify key data points. Let's create a stemplot for our chicken weight data from above.

```r
# Stemplot of group 1 weights
stem(group1Weights)
```

```
##
##   The decimal point is 1 digit(s) to the right of the |
##
##    2 | 599
##    4 | 011111111112222223334578889999999901111112344556667788999
##    6 | 001122233445557777888801111122234446799
##    8 | 112344445788999901233366678889
##   10 | 0011233666780222355679
##   12 | 00234455683456889
##   14 | 112468945777
##   16 | 0002234481457
##   18 | 124577257899
##   20 | 255958
##   22 | 037
##   24 | 809
##   26 | 6
##   28 | 8
##   30 | 5
```

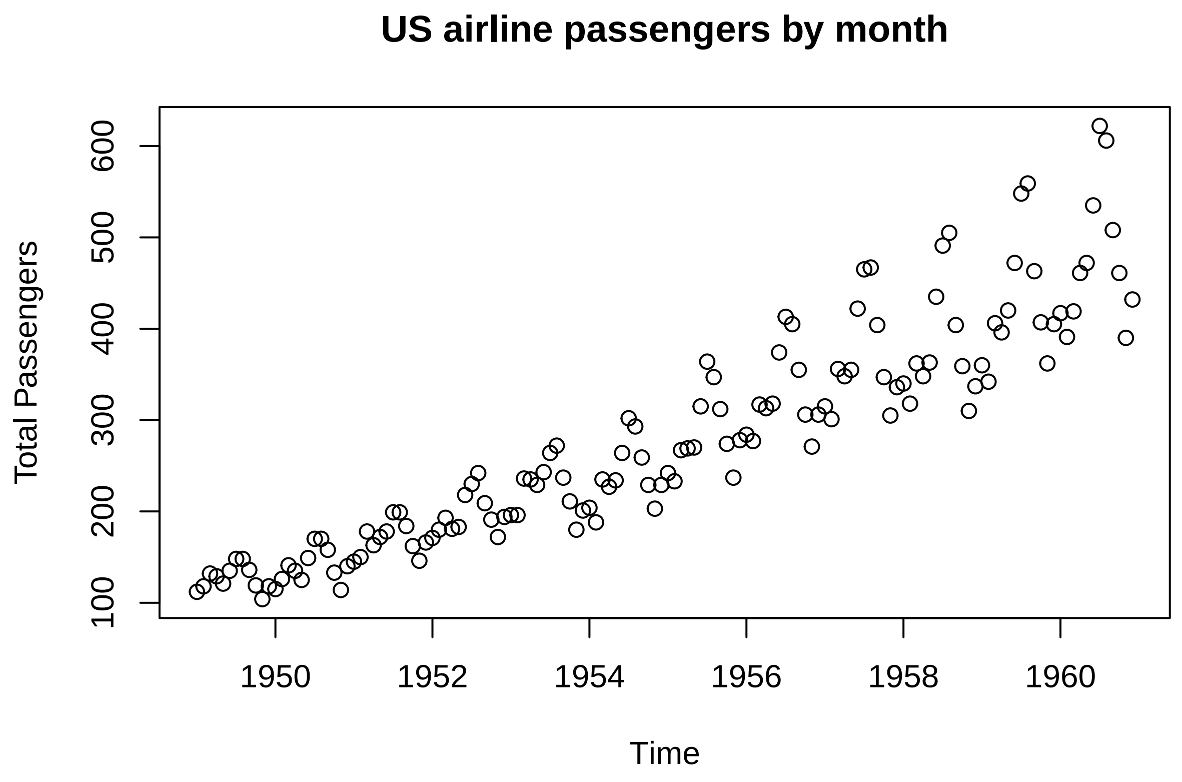
```r
# Stemplot of group 4 weights
stem(group4Weights)
```

```
##
##   The decimal point is 1 digit(s) to the right of the |
##
##    2 | 9
##    4 | 0011122229001123345
##    6 | 122345667989
##    8 | 024455668
##   10 | 0133345878
##   12 | 02345678158
##   14 | 14567823455677
##   16 | 068034455
##   18 | 44458677899
##   20 | 03445500
##   22 | 2134478
##   24 |
##   26 | 1449
##   28 | 1
##   30 | 3
##   32 | 2
```

2.3.3 Scatter Plot

A scatter plot is a graphical representation that displays individual data points on a two-dimensional plane, with one variable on the x-axis and another on the y-axis, allowing you to visualize the relationship, pattern, or correlation between the two variables. Let's create a scatter plot using the R command `plot()` for the US airline passengers by month using our data from above.

```r
# Time series plot of AirPassengers
plot(AirPassengers, main = "US airline passengers by month", xlab = "Time",
    ylab = "Total Passengers", type = "p")
```

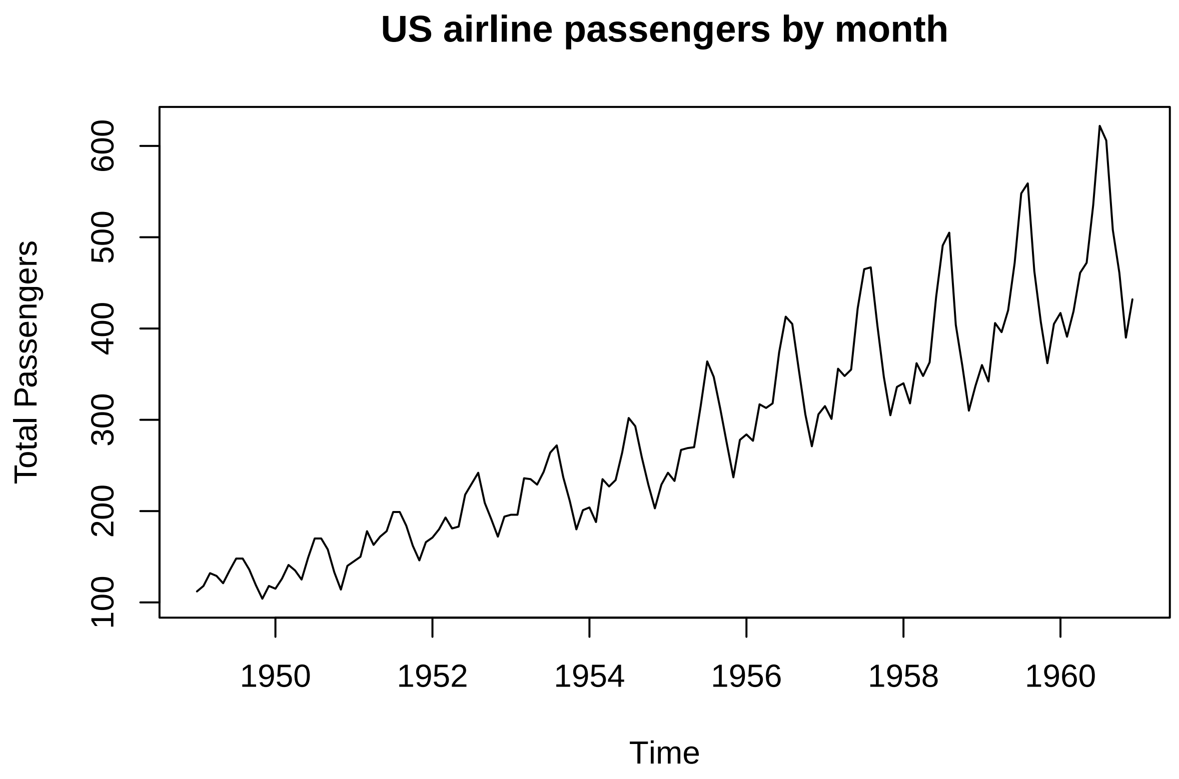


Notice the overall increasing trend of the data.

2.3.4 Time-series Graph

A time series is a sequence of data points collected or recorded at successive points in time, typically at evenly spaced intervals, and a time series graph visually represents this data over time, allowing us to observe trends, patterns, and changes in the data's behavior. Let's use the R command `ts_plot()` to plot the total US airline passengers by month using our data from above.

```
# Time series plot of AirPassengers
ts.plot(AirPassengers, main = "US airline passengers by month", xlab = "Time",
    ylab = "Total Passengers")
```

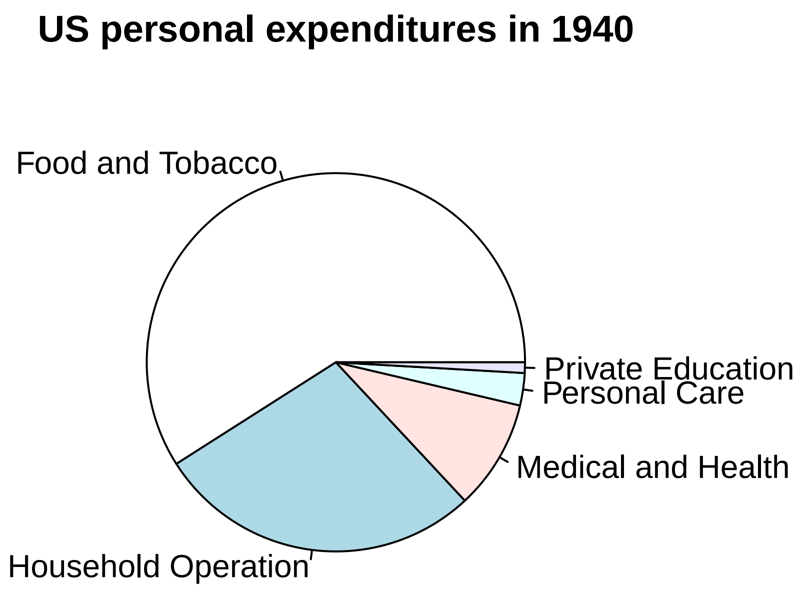


The time series graph shows several interesting phenomena: 1) airline travel is seasonal with the same basic pattern repeated each year and 2) the overall trend is increasing.

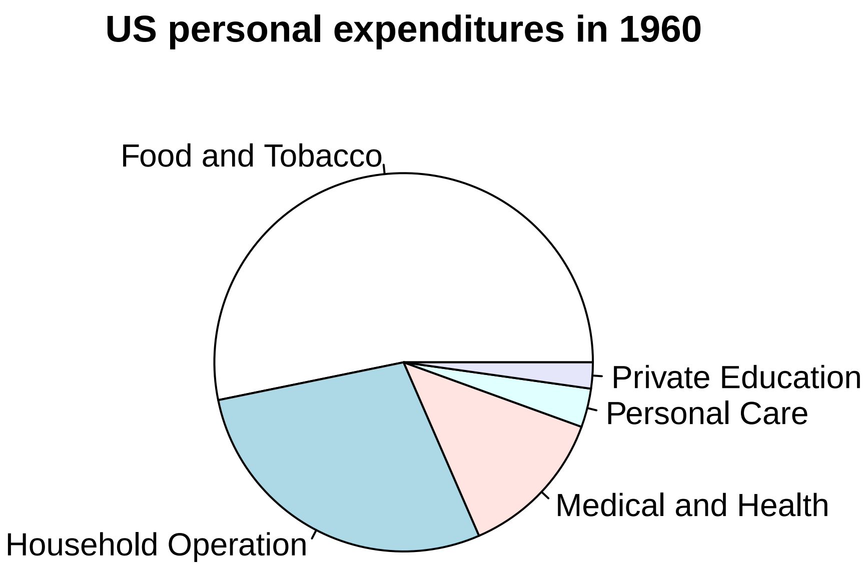
2.3.5 Pie Chart

A pie chart is a circular graph that visually represents data as slices, with each slice showing the proportion or percentage of different categories in the whole dataset. Let's use a pie chart to visualize the difference between average personal expenditure in the US in 1940 vs 1960 using `USPeronalExpenditure` defined above.

```r
# Pie chart of 1940 expenditures: labels allows us to name the categories
# as defined in cats above
pie(expenditures1940, main = "US personal expenditures in 1940")
```



```
# Pie chart of 1960 expenditures: labels allows us to name the categories
# as defined in cats above
pie(expenditures1960, main = "US personal expenditures in 1960")
```



2.3.6 Pareto Chart

A Pareto chart is a specialized bar chart that displays data in descending order of frequency or importance, highlighting the most significant factors or categories, making it a visual tool for prioritization and decision-making. Let's use the `expenditures1940` and `expenditures1960` data from above to illustrate the usefulness of a Pareto chart.

The first time you run this code, you will need to install the following package. After this initial run, you can skip running this code:

```
# Installs the package 'qcc'.  ONLY RUN THIS CODE ONCE!
install.packages("qcc")
```

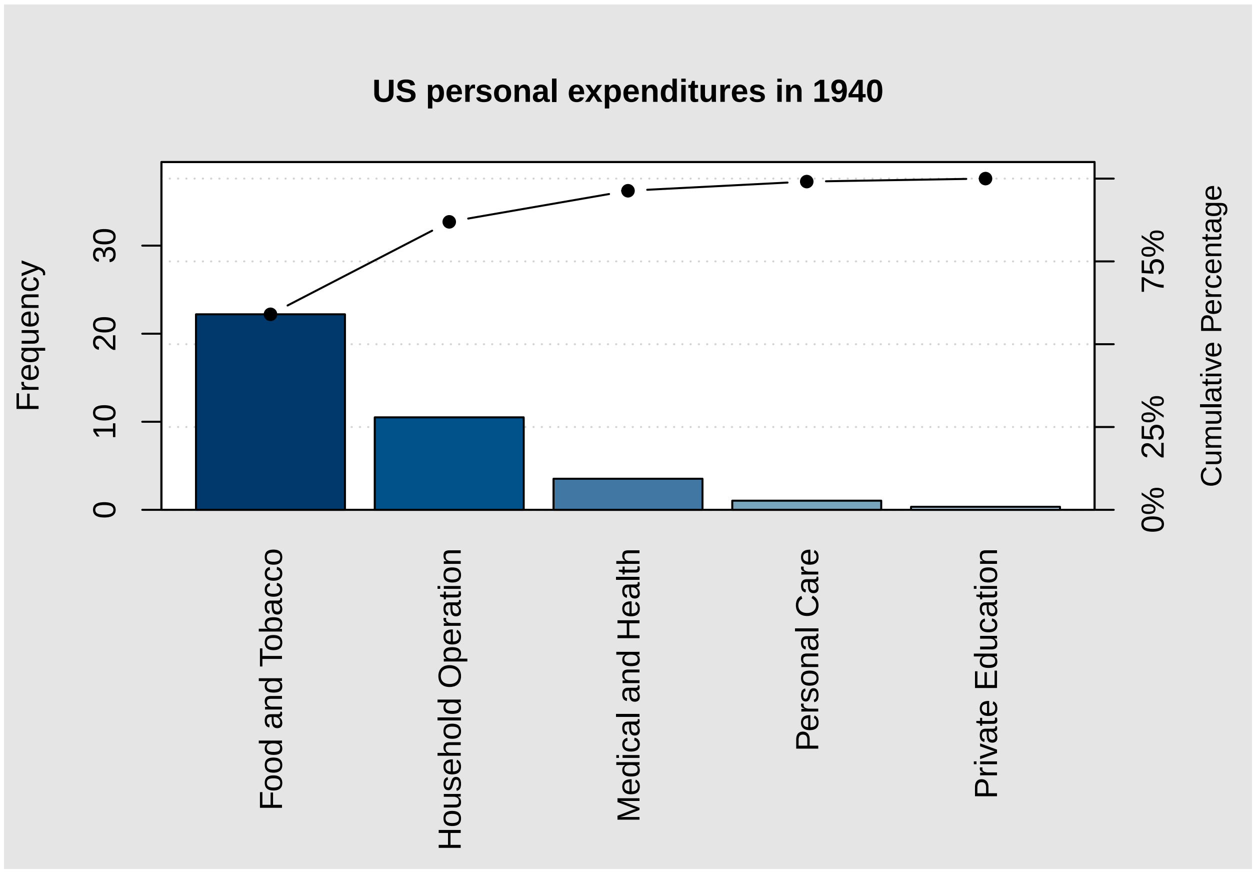
Now, let's create Pareto charts for the 1940 and 1960 expenditure data.

```
# Load 'qcc' package
library(qcc)
```

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

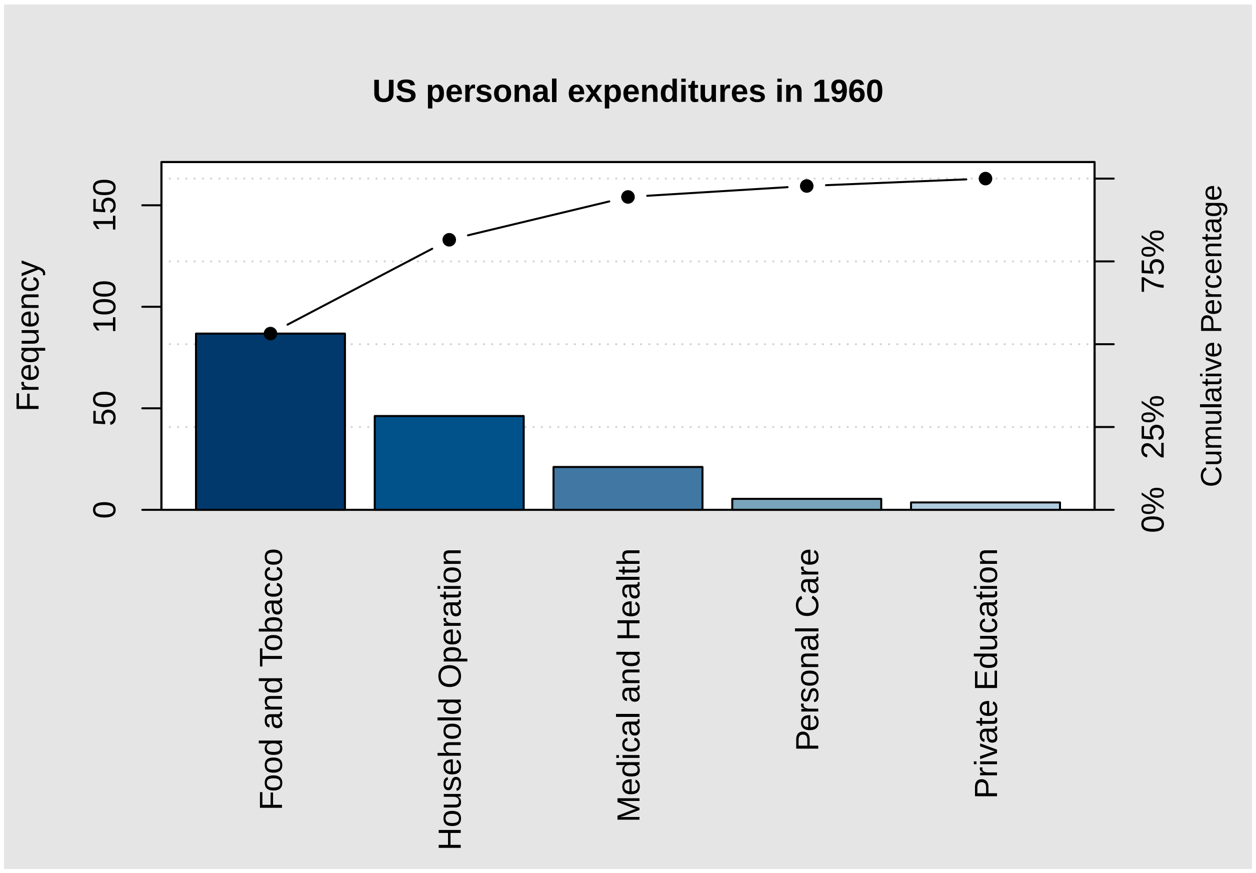
```
## Type 'citation("qcc")' for citing this R package in publications.
```

```
# Create the Pareto chart for 1940 data
pareto.chart(expenditures1940, xlab = "", ylab = "Frequency", main = "US personal expenditures in 1940")
```



```
##
## Pareto chart analysis for expenditures1940
##                        Frequency   Cum.Freq.  Percentage Cum.Percent.
##   Food and Tobacco     22.2000000  22.2000000  59.0252852   59.0252852
##   Household Operation  10.5000000  32.7000000  27.9173646   86.9426498
##   Medical and Health    3.5300000  36.2300000   9.3855521   96.3282019
##   Personal Care         1.0400000  37.2700000   2.7651485   99.0933503
##   Private Education     0.3410000  37.6110000   0.9066497  100.0000000
```

```
# Create the Pareto chart for 1960 data
pareto.chart(expenditures1960, xlab = "", ylab = "Frequency", main = "US personal expenditures in 1960")
```



```
##
## Pareto chart analysis for expenditures1960
##                        Frequency  Cum.Freq. Percentage Cum.Percent.
##   Food and Tobacco     86.800000  86.800000  53.205835    53.205835
##   Household Operation  46.200000 133.000000  28.319235    81.525070
##   Medical and Health   21.100000 154.100000  12.933677    94.458747
##   Personal Care         5.400000 159.500000   3.310040    97.768788
##   Private Education     3.640000 163.140000   2.231212   100.000000
```

2.3.7 Let's put it all together!

Using the built-in dataset for quarterly profits of the company Johnson & Johnson, load the data and view it using this code.

```
# Johnson & Johnson Profits: Load data from the built-in dataset into a
# variable named JohnsonJohnson
data("JohnsonJohnson")

JohnsonJohnson
```

```
##      Qtr1 Qtr2 Qtr3 Qtr4
## 1960 0.71 0.63 0.85 0.44
## 1961 0.61 0.69 0.92 0.55
## 1962 0.72 0.77 0.92 0.60
## 1963 0.83 0.80 1.00 0.77
## 1964 0.92 1.00 1.24 1.00
## 1965 1.16 1.30 1.45 1.25
## 1966  1.26  1.38  1.86  1.56
## 1967  1.53  1.59  1.83  1.86
## 1968  1.53  2.07  2.34  2.25
## 1969  2.16  2.43  2.70  2.25
## 1970  2.79  3.42  3.69  3.60
## 1971  3.60  4.32  4.32  4.05
## 1972  4.86  5.04  5.04  4.41
## 1973  5.58  5.85  6.57  5.31
## 1974  6.03  6.39  6.93  5.85
## 1975  6.93  7.74  7.83  6.12
## 1976  7.74  8.91  8.28  6.84
## 1977  9.54 10.26  9.54  8.73
## 1978 11.88 12.06 12.15  8.91
## 1979 14.04 12.96 14.85  9.99
## 1980 16.20 14.67 16.02 11.61
```

Now, select the best plot from those illustrated above and plot this data. Hint: this looks like a time series to me…

2.4 Scatter plots, correlation, and regression

Correlation quantifies the strength and direction of the relationship between two variables, helping assess how they move together (or in opposite directions). Any potential such relationship can be visualized using a scatter plot as introduced in Section 2.3.

2.4.1 Linear correlation

Linear correlation measures the strength and direction of the linear relationship between two variables, often represented by the correlation coefficient (r). The p-value associated with this coefficient assesses the statistical significance of the correlation, helping determine whether the observed relationship is likely due to chance or represents a real association. Let' consider the built-in dataset `mtcars` which contains several aspects and performance of several 1973 - 1974 model cars. This code loads the dataset and displays several of its entries.

```
# mtcars: Load data from the built-in dataset into a variable named mtcars
data("mtcars")

mtcars
```

```
##                      mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4           21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag       21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710          22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive      21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout   18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
## Valiant             18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
## Duster 360          14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
## Merc 240D           24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
## Merc 230            22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
## Merc 280            19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
## Merc 280C           17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
## Merc 450SE          16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
## Merc 450SL          17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
## Merc 450SLC         15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
## Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
## Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4
## Chrysler Imperial   14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
## Fiat 128            32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
## Honda Civic         30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
## Toyota Corolla      33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
## Toyota Corona       21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
## Dodge Challenger    15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2
## AMC Javelin         15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2
## Camaro Z28          13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
## Pontiac Firebird    19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
## Fiat X1-9           27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
## Porsche 914-2       26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
## Lotus Europa        30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
## Ford Pantera L      15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
## Ferrari Dino        19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
## Maserati Bora       15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
## Volvo 142E          21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
```

Let's see if there is a linear relationship between miles per gallon (MPG) and the engine horse powerr (HP) using the R command `cor.test()` and storing the **linear correlation coefficient** (r) and **P-value** in the variable `mpgvshp`. Notice that `mtcars$mpg` extracts just the column of MPG from the dataset and similarly for `mtcars$hp`. The r-value can be found by calling `mpgvshp$estimate`, whereas, the P-value can be found by calling `mpgvshp$p.value`. Finally, the critical r-value range is found using the `mpgvshp$conf.int` command.

```
# Calculate the correlation between MPG and HP
mpgvshp = cor.test(mtcars$mpg, mtcars$hp)

mpgvshp
```

```
##
##  Pearson's product-moment correlation
##
## data:  mtcars$mpg and mtcars$hp
## t = -6.7424, df = 30, p-value = 1.788e-07
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.8852686 -0.5860994
## sample estimates:
##        cor
## -0.7761684
```

```
# Let's view the r- and P-values and critical r-value range
cat("r:", mpgvshp$estimate, "\n")
```

```
## r: -0.7761684
```

```
cat("P-value:", mpgvshp$p.value, "\n")
```

```
## P-value: 1.787835e-07
```

```
cat("Critical r-value range: (", mpgvshp$conf.int[1], ", ", mpgvshp$conf.int[2],
    ")")
```

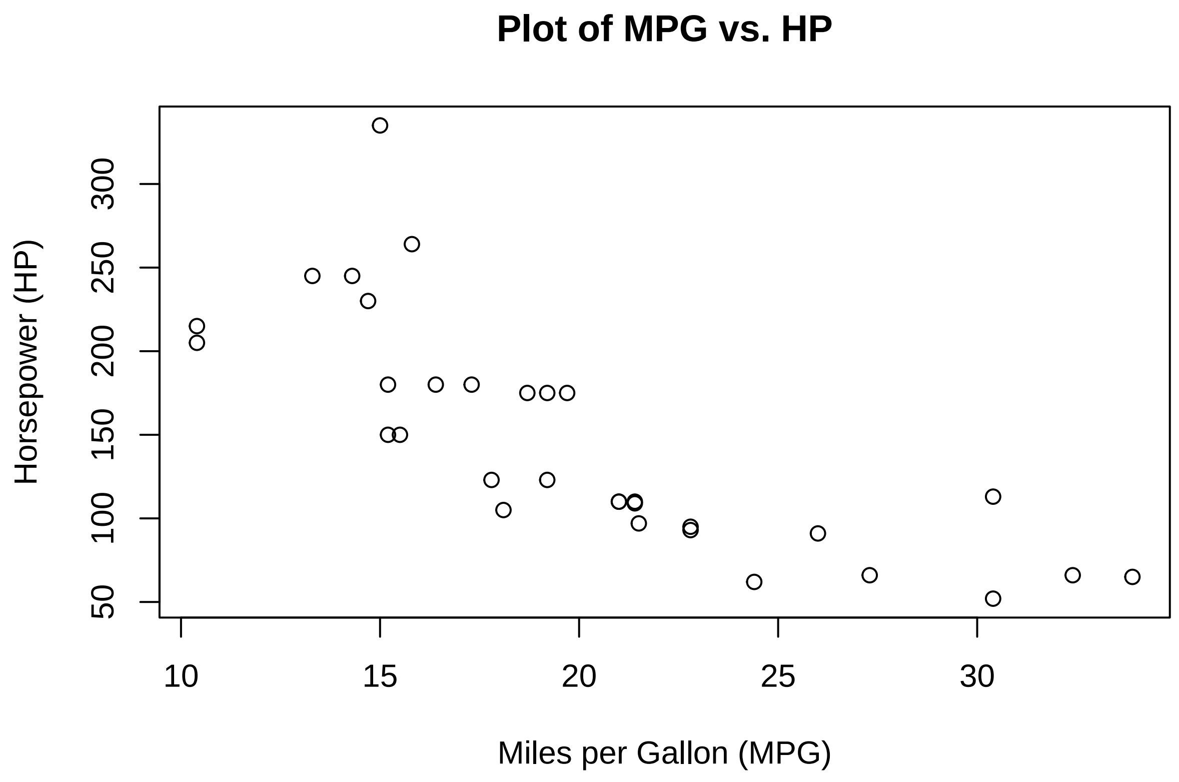
```
## Critical r-value range: ( -0.8852686 ,  -0.5860994 )
```

A negative r-value indicates that if a linear relationship is present then the relationship is negative, i.e., increasing the MPG decreases the HP. Having the absolute value of the r-value close to one indicates a linear relationship. Notice that our r-value falls within the critical r-value range supporting the conclusion that a linear relationship is present.

A P-value of less than **0.05** suggests that the sample results are *not* likely to occur merely by chance when there is no linear correlation. Thus, a small P-value such as the one we received here supports a conclusion that there is a linear correlation between MPG and HP.

Now, let's use a scatter plot to visualize the relationship.

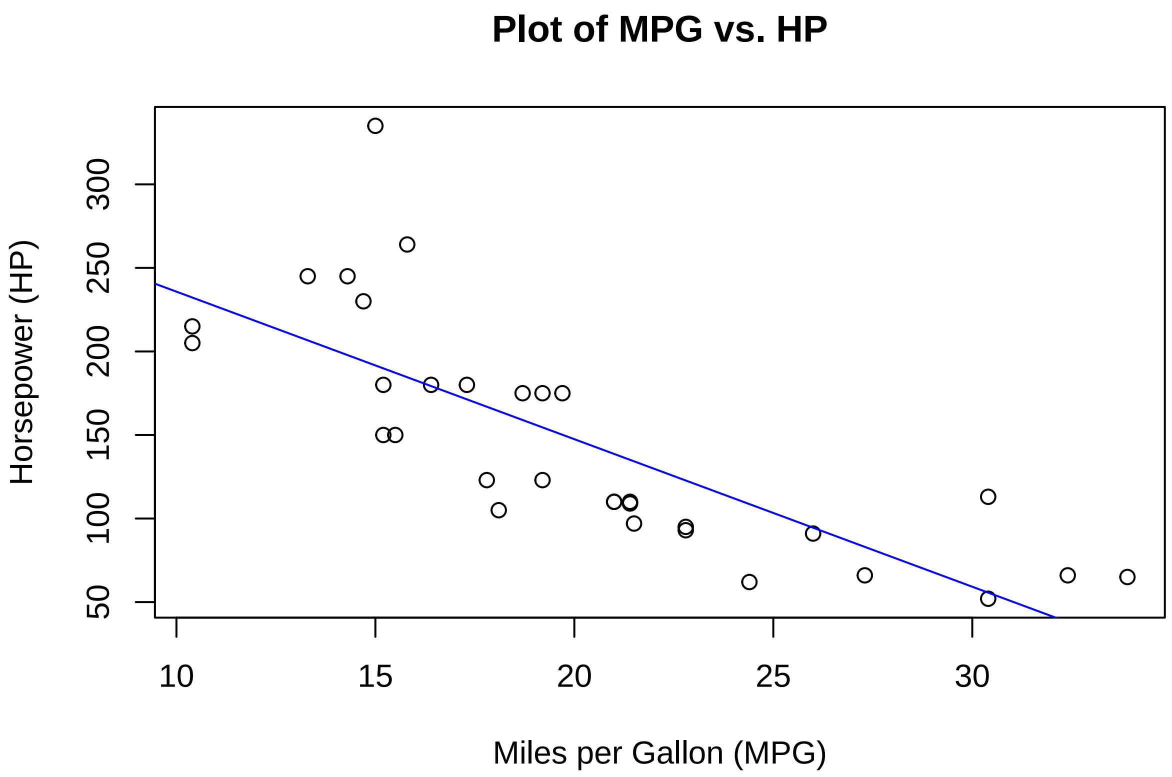
```r
# Create a scatter plot to visualize the relationship
plot(mtcars$mpg, mtcars$hp, xlab = "Miles per Gallon (MPG)", ylab = "Horsepower (HP)",
    main = "Plot of MPG vs. HP")
```



2.4.2 Regression line

Regression analyzes and models the relationship between variables, allowing us to predict one variable based on the values of others. Let's return to our MPG vs HP example. We will use the R command `lm()` to create a linear model (or linear regression) for this data. We then use our scatter plot created previously to plot the model prediction alongside the actual data points. In this case, the R command `abline()` adds the regression line stored in `model` with the color being specified by the attribute `col`.

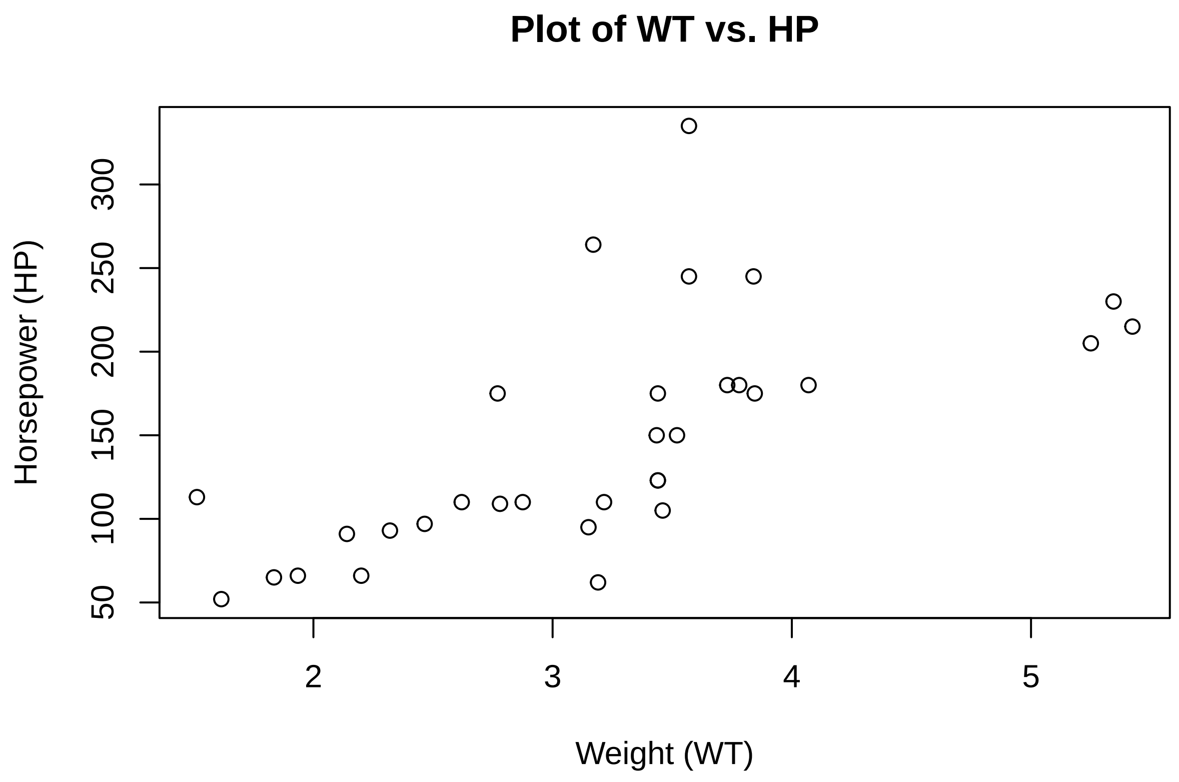
```r
# Create a linear regression model
model = lm(hp ~ mpg, data = mtcars)

# Create a scatter plot to visualize the relationship
plot(mtcars$mpg, mtcars$hp, xlab = "Miles per Gallon (MPG)", ylab = "Horsepower (HP)",
    main = "Plot of MPG vs. HP")

# Add the regression line to the plot
abline(model, col = "blue")
```



2.4.3 Let's put it all together!

Using the same `mtcars` dataset, use what you have learned above to determine if there is a linear correlation between the weight of a car in the set versus the engine's horse power. The following code will walk you through the process. We begin with a visualization of the data using a scatter plot.

```r
# Create a scatter plot to visualize the relationship
plot(mtcars$wt, mtcars$hp, xlab = "Weight (WT)", ylab = "Horsepower (HP)", main = "Plot of WT vs. HP")
```



Now, let's determine if there is a linear relationship between car weight `mtcars$wt` and engine horsepower `mtcars$hp`.

```
# Calculate the correlation between MPG and HP
wtvshp = cor.test(mtcars$wt, mtcars$hp)

wtvshp
```

```
##
##  Pearson's product-moment correlation
##
## data:  mtcars$wt and mtcars$hp
## t = 4.7957, df = 30, p-value = 4.146e-05
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.4025113 0.8192573
## sample estimates:
##       cor
## 0.6587479
```

```
# Let's view the r- and P-values and critical r-value range
cat("r:", wtvshp$estimate, "\n")
```

```
## r: 0.6587479
```

```
cat("P-value:", wtvshp$p.value, "\n")
```

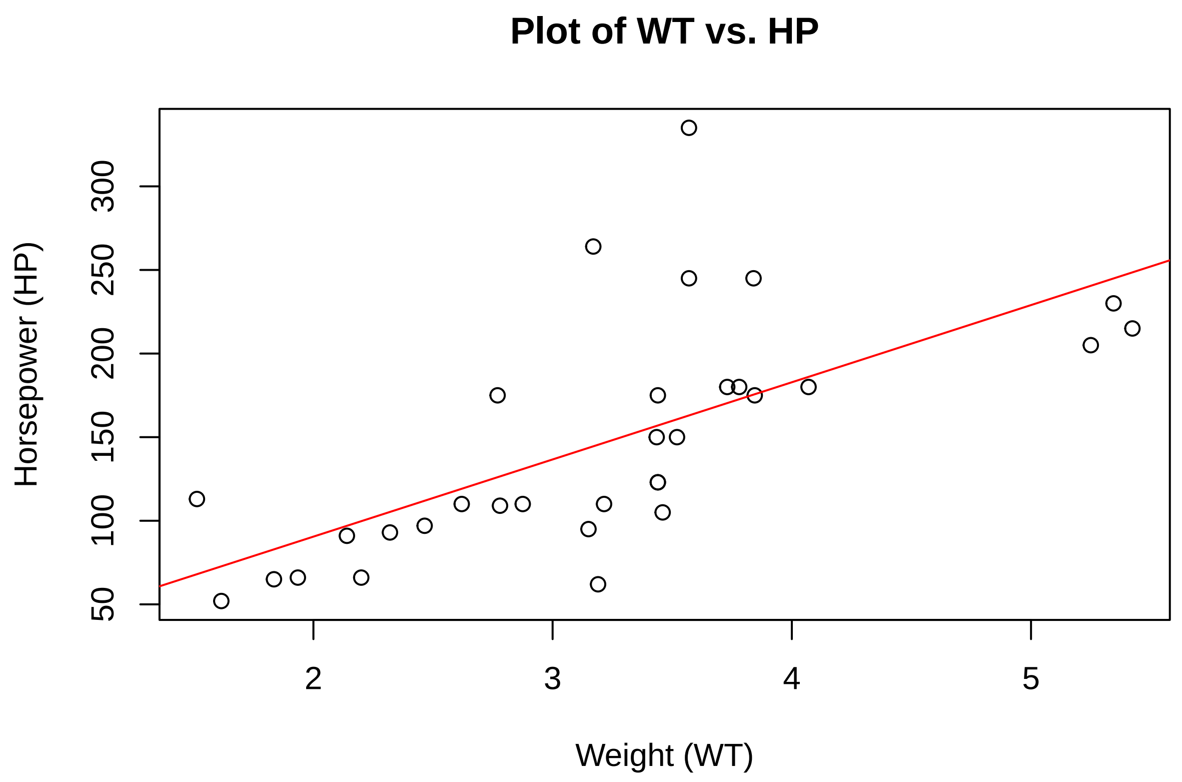
```
## P-value: 4.145827e-05
```

```r
cat("Critical r-value range: (", wtvshp$conf.int[1], ", ", wtvshp$conf.int[2],
    ")")
```

```
## Critical r-value range: ( 0.4025113 ,  0.8192573 )
```

What can we conclude about a possible linear relationship between car weight and horsepower? Is this relationship supported? Finally, let's visualize the regression line and data together.

```r
# Create a linear regression model
model2 <- lm(hp ~ wt, data = mtcars)

# Create a scatter plot to visualize the relationship
plot(mtcars$wt, mtcars$hp, xlab = "Weight (WT)", ylab = "Horsepower (HP)", main = "Plot of WT vs. HP")

# Add the regression line to the plot
abline(model2, col = "red")
```



What about causation? Does having a heavier car make it have higher or lower horsepower?

3 Describing, Exploring, and Comparing Data

3.1 Measures of center

Measures of center, such as the mean and median, provide a central value that summarizes a dataset, helping to understand its typical or central tendency, which is crucial for making data-driven decisions and drawing inferences.

3.1.1 Mean

The mean, also known as the average, is a measure of center in a dataset that calculates the sum of all values divided by the total number of values, providing a representative value for the dataset. We will employ the R command `mean()` to calculate the mean of several datasets. First, let's use the test scores from Section 2.1.1 which should be stored in `scores`.

```
# Calculate mean of scores and then store it in the variable meanScore
meanScore = mean(scores)

# print out the answer
cat("Mean test score is: ", meanScore, "\n")
```

```
## Mean test score is:  74.17647
```

The mean is very sensitive to outliers. Let's see what happens when we take the same `scores` list and add some really low grades to the list.

```
# Previous test scores with a several much lower scores added
scores2 = c(95, 90, 85, 85, 87, 74, 75, 64, 85, 84, 87, 15, 20, 75, 75, 90,
    75, 2, 1, 5, 3)

# Calculate mean of scores2 and then store it in the variable meanScore2
meanScore2 = mean(scores2)

# print out the answer
cat("Mean test score is from original is: ", meanScore, ", while from scores2 is: ",
    meanScore2)
```

```
## Mean test score is from original is:  74.17647 , while from scores2 is:  60.57143
```

This sensitivity to outliers is the notion of resistance. The mean is not a resistant measure of middle.

3.1.2 Median

The median is a measure of center in a dataset that represents the middle value when all values are ordered, and it is resistant to extreme outliers, making it a robust statistic for summarizing data. Let's return to the scores data and see the difference between mean and median of the two datasets `scores` and `scores2` using the R commands `median()`.

```
# Calculate median of scores and then store it in the variable medianScore
medianScore = median(scores)

# Calculate median of scores2 and then store it in the variable
# medianScore2
medianScore2 = median(scores2)

# print out the answer
cat("Mean test score from original is: ", meanScore, ", while from scores2 is: ",
    meanScore2, "\n\n")
```

```
## Mean test score from original is:  74.17647 , while from scores2 is:  60.57143
```

```
cat("Median test score from original is: ", medianScore, ", while from scores2 is: ",
    medianScore2, "\n")
```

```
## Median test score from original is:  84 , while from scores2 is:  75
```

3.1.3 Mode

The mode is a statistical measure that represents the value or values that occur most frequently in a dataset, making it a useful indicator of the most common observation(s); however, it is not necessarily resistant to outliers, meaning extreme values can heavily influence the mode. There is no bulit-in R command for mode, so we will have to employ the package `DescTools`.

The first time you run this code, you will need to install the following package. After this initial run, you can skip running this code:

```r
# Installs the package 'DescTools'.  ONLY RUN THIS CODE ONCE!
install.packages("DescTools")
```

Once this package is installed, then we can load the library `DescTools` in order to use the R command `Mode()`.

```r
# Load the DescTools package
library(DescTools)

# Calculate the mode of both scores and scores2 using the Mode() method

# Calculate Mode of scores and then store it in the variable modeScore
modeScore = Mode(scores)

# Calculate median of scores2 and then store it in the variable modeScore2
modeScore2 = Mode(scores2)

# print out the answer
cat("Mode test score from original is: ", modeScore, ", while from scores2 is: ",
    modeScore2, "\n")
```

```
## Mode test score from original is:  75 , while from scores2 is:  75
```

3.1.4 Midrange

The midrange is a measure of center in a dataset that represents the arithmetic mean of the maximum and minimum values, and it is not resistant to extreme outliers, making it sensitive to extreme values. There is no built-in R command for midrange, thus we will use the following code to calculate the midrange of our `scores` and `scores2` data.

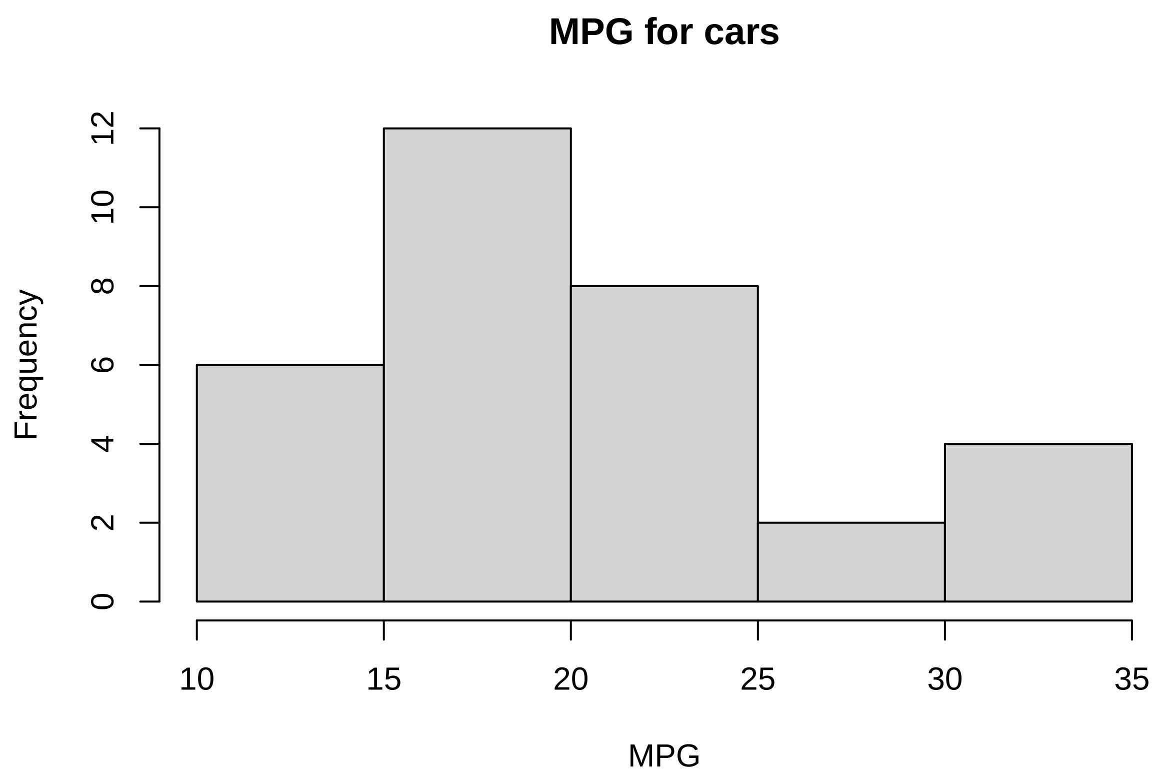
```r
# Calculate miderange of scores and then store it in the variable
# midrangeScore
midrangeScore = (max(scores) - min(scores))/2

# Calculate midrange of scores2 and then store it in the variable
# midrangeScore2
midrangeScore2 = (max(scores2) - min(scores2))/2

# print out the answer
cat("Midrange test score from original is: ", midrangeScore, ", while from scores2 is: ",
    midrangeScore2, "\n")
```

```
## Midrange test score from original is:  40 , while from scores2 is:  47
```

3.1.5 Let's put it all togeher!

Consider the built-in dataset `mtcars` which contains several aspects and performance of several 1973 - 1974 model cars which we studied in Section 2.4. We will calculate mean, meidan, mode, and midrange of the miles per gallon of tthe cars in the dataset. using the R commands illustrated in the previous sections, as well as compute the so-called 5-number summary using the R command `summary()`. First, let's plot a histogram of the data.

```r
# Extract the MPG data and store it into the variable carsMPG
carsMPG = mtcars$mpg

# Generate a histogram of the MPG data from mtcars
hist(carsMPG, main = "MPG for cars", xlab = "MPG")
```



```r
# Calculate mean of MPG data and then store it in the variable meanMPG
meanMPG = round(mean(carsMPG), digits = 2)

# Calculate median of MPG data and then store it in the variable medianMPG
medianMPG = median(carsMPG)

# Calculate Mode of scores and then store it in the variable modeMPG
modeMPG = Mode(carsMPG)

# Calculate miderange of scores and then store it in the variable
# midrangeMPG
midrangeMPG = (max(carsMPG) - min(carsMPG))/2

# print out the answer
cat("Mean \t Median \t \t \t  Mode \t \t \t Midrange \n")
```

```
## Mean      Median                Mode           Midrange
```

```
cat(meanMPG, " \t ", medianMPG, " \t ", modeMPG, " \t ", midrangeMPG, "\n\n")
```

```
## 20.09      19.2        10.4 15.2 19.2 21 21.4 22.8 30.4        11.75
```

```
# Give the 5-number summary for MPG data
cat("5-Number Summary \n")
```

```
## 5-Number Summary
```

```
summary(carsMPG)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   10.40   15.43   19.20   20.09   22.80   33.90
```

Notice that there are 7 elements in the mode. That's because there are 7 most frequent elements, each which appear twice. Which of these central measures best describes what you visually see as the "center" of data using the histogram? What does it "mean" that the mean and median are close to each other? Does the 5-number summary give us any additional information regarding the measure of "center" in the data?

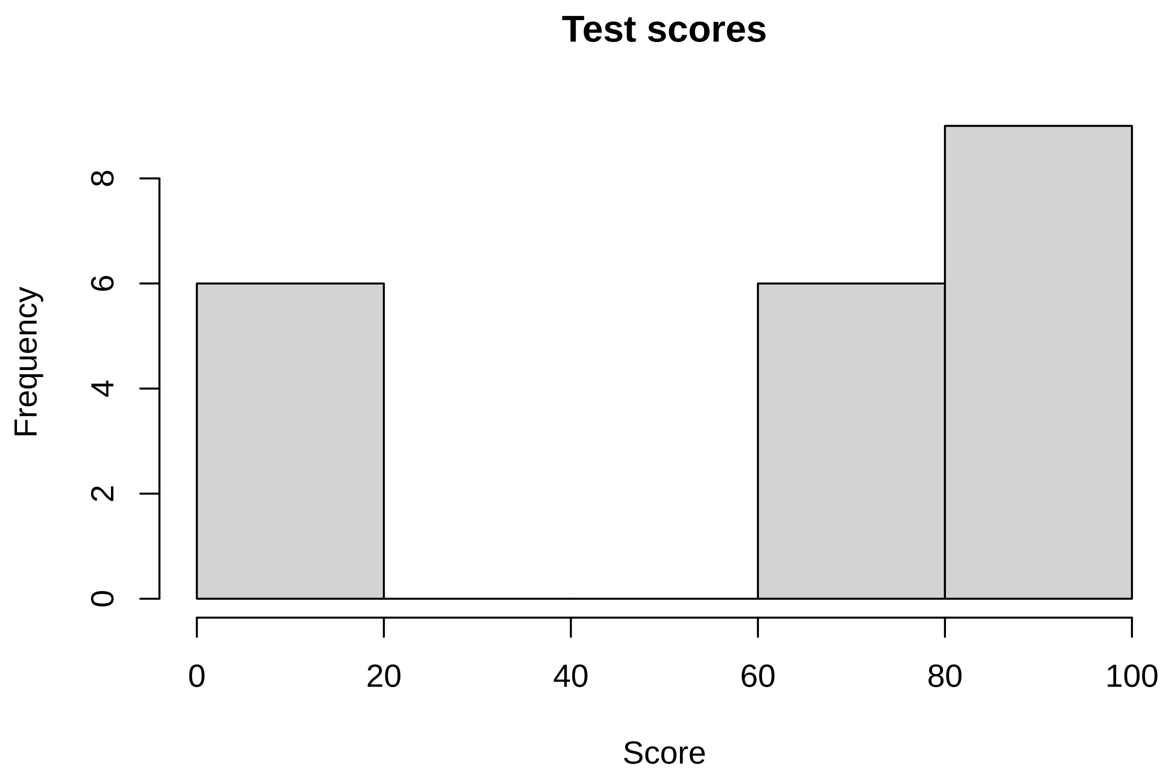
3.2 Measures of variation

Measures of variation, such as the range, variance, and standard deviation, provide insights into the spread or dispersion of data points within a dataset, helping us understand how much individual values deviate from the central tendency measures like the mean or median. These measures are essential because they quantify the degree of variability in data, allowing us to assess data quality, make more accurate predictions, and draw meaningful conclusions in statistical analysis.

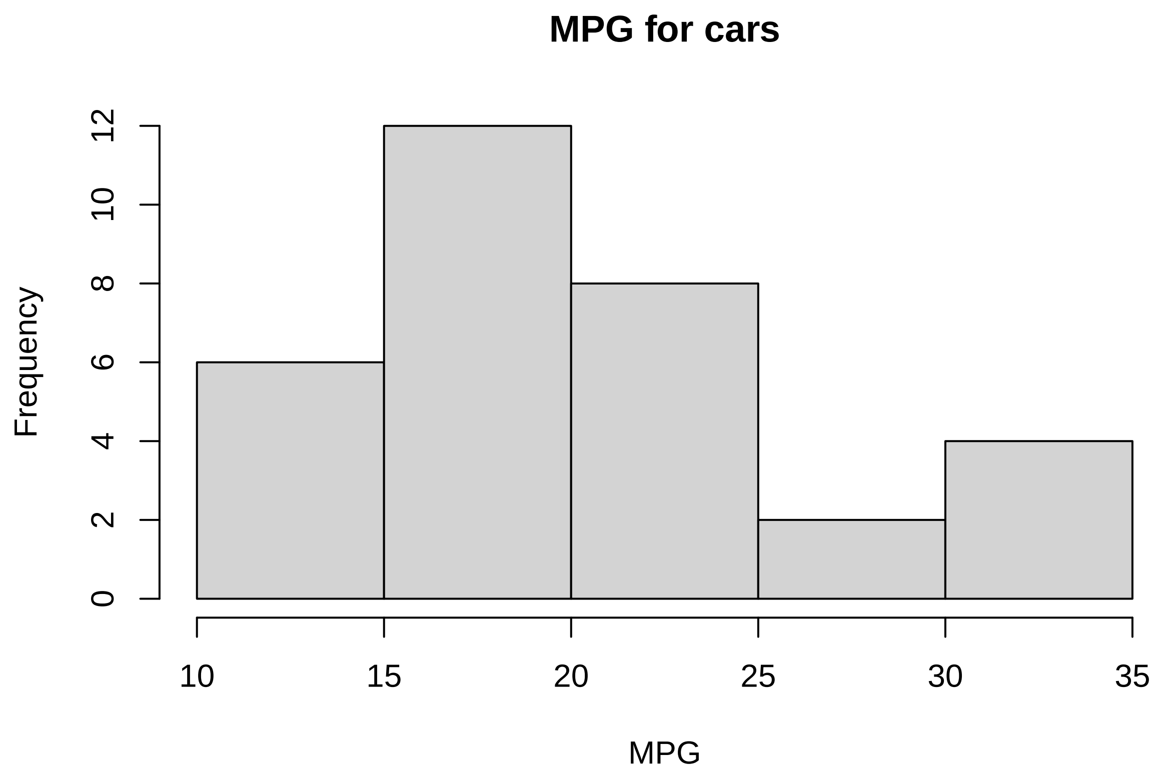
3.2.1 Visualizing variation

Histograms can visually represent the variation in a dataset by displaying the distribution of values across different bins or intervals, highlighting the frequency and pattern of data points, and revealing the shape and spread of the distribution. Let's compare histograms for our `scores2` and `carsMPG` datasets.

```
# Generate a histogram of the MPG data from scores2
hist(scores2, main = "Test scores", xlab = "Score")
```



```r
# Generate a histogram of the MPG data from mtcars
hist(carsMPG, main = "MPG for cars", xlab = "MPG")
```



3.2.2 Range

The range is a measure of variation that represents the difference between the maximum and minimum values in a dataset, but it is not resistant to outliers, meaning extreme values can substantially affect the range. Let's compare the ranges of our `carsMPG` and `scores2` datasets using the R command `range()`.

```
# Calculate range of scores2 and then store it in the variable rangeScore2
rangeScore2 = range(scores2)

# Calculate range of carsMPG and then store it in the variable rangeMPG
rangeMPG = range(carsMPG)

# print out the answer
cat("Range for test scores from scores2 is: (", rangeScore2[1], ", ", rangeScore2[2],
    ") \n")
```

```
## Range for test scores from scores2 is: ( 1 ,  95 )
```

```
cat("Range for MPG from carsMPG is: (", rangeMPG[1], ", ", rangeMPG[2], ") \n")
```

```
## Range for MPG from carsMPG is: ( 10.4 ,  33.9 )
```

3.2.3 Standard deviation

Standard deviation is a measure of the dispersion or spread of data points in a dataset, with a higher value indicating greater variability, and it's calculated differently for **populations** (σ) and **samples** (s), where the **sample** standard deviation (s) is often used for practical data analysis. However, standard deviation is not resistant to extreme outliers, making it sensitive to the influence of extreme values on its magnitude. There is a built-in R command for **sample** standard deviation, but no such command for **population** standard deviation. Recall our test scores dataset `scores2`. Since this data represents the entire population (every student in the class), we will calculate **population** standard deviation for that dataset. However, the MPG data in `carsMPG` is only a sample of all the cars on the market in 1973 - 1974. Thus, we will employ the R command `sd()` to calculate **sample** standard deviation.

```r
# Calculate population SD of scores2 and then store it in the variable
# popSDScore2
popSDScore2 = sqrt(var(scores2) * (length(scores2) - 1)/length(scores2))

# Calculate sample SD of carsMPG and then store it in the variable
# samSDMPG
samSDMPG = sd(carsMPG)

# Print out the answer
cat("Population standard deviation for test scores from scores2 is: ", popSDScore2,
    "\n\n")
```

```
## Population standard deviation for test scores from scores2 is:  34.35331
```

```r
cat("Sample standard deviation for MPG from carsMPG is: ", samSDMPG, " \n")
```

```
## Sample standard deviation for MPG from carsMPG is:  6.026948
```

3.2.4 Variance

Variance measures the average of the squared differences between each data point and the mean of a dataset, providing a measure of data dispersion, but it is not resistant to extreme outliers, making it sensitive to the influence of extreme values on its magnitude. Variance is calculated differently for **populations** (σ^2) and **samples** (s^2), with the **sample** variance (s^2) being used for practical data analysis to account for bias when working with a subset of a larger population. Let's compare **population** variance for our `scores2` dataset and **sample** variance for our `carsMPG` dataset. As with standard deviation, although there is a built-in R command for **sample** variance, there is not a built-in command for **population** variance, so we will have to improvise.

```r
# Calculate population variance of scores2 and then store it in the
# variable popVarScore2
popVarScore2 = var(scores2) * (length(scores2) - 1)/length(scores2)

# Calculate sample variance of carsMPG and then store it in the variable
# samSDMPG
samVarMPG = var(carsMPG)

# Print out the answer
cat("Population variance for test scores from scores2 is: ", popVarScore2, "\n\n")
```

```
## Population variance for test scores from scores2 is:  1180.15
```

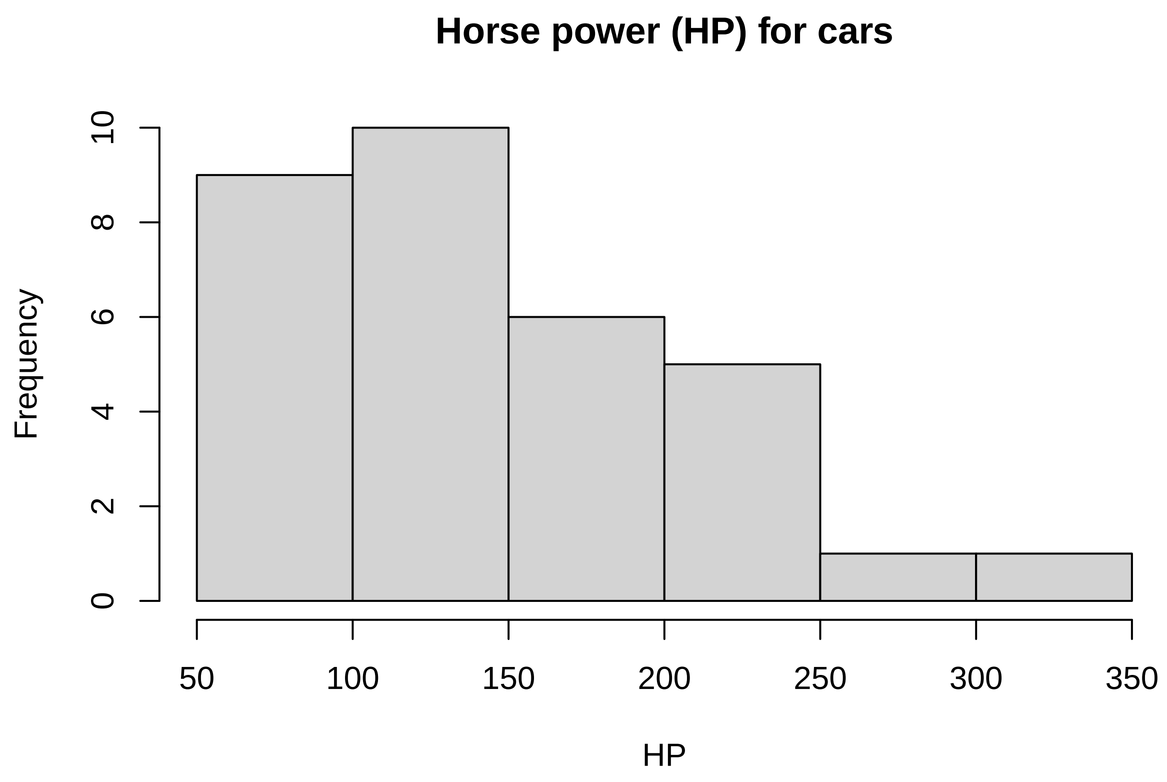
```r
cat("Sample variance for MPG from carsMPG is: ", samVarMPG, " \n")
```

```
## Sample variance for MPG from carsMPG is:  36.3241
```

3.2.5 Let's put it all togeher!

Consider the built-in dataset `mtcars` which contains several aspects and performance of several 1973 - 1974 model cars which we studied in Section 2.4. We will first calculate mean and median of the horse power (HP) of the cars in the dataset. To calculate measures of variation, we note that since this is just a **sample** of all possible cars on the market during 1973 - 1974, we will employ **sample** variance and standard deviation using the R commands illustrated in the previous sections, along with a histogram to visually explore the data.

```r
# Extract the HP data and store it into the variable carsHP
carsHP = mtcars$hp

# Generate a histogram of the HP data from mtcars
hist(carsHP, main = "Horse power (HP) for cars", xlab = "HP")
```



```r
# Calculate mean of HP data and then store it in the variable meanHP
meanHP = round(mean(carsHP), digits = 2)

# Calculate median of HP data and then store it in the variable medianHP
medianHP = median(carsHP)

# Calculate variance of HP data and then store it in the variable varHP
varHP = var(carsHP)

# Calculate standard deviation of HP and then store it in the variable
# midrangeHP
sdHP = sd(carsHP)

# print out the answer
cat("Mean \t Median \t  variance \t Standard Deviation \n")
```

```
## Mean      Median       variance   Standard Deviation
```

```r
cat(meanHP, " \t ", medianHP, " \t ", varHP, " \t ", sdHP, "\n\n")
```

```
## 146.69      123      4700.867         68.56287
```

Now, compare the MPG and HP data from the `mtcars` dataset. For MPG, we calculated a standard deviation around *36* and for HP of around *69*. Does this mean that the MPG data is less spread out that the HP data? Is your answer to this question consistent with the histograms we produced? Can we compare standard deviations from two totally different datasets in a meaningful way?

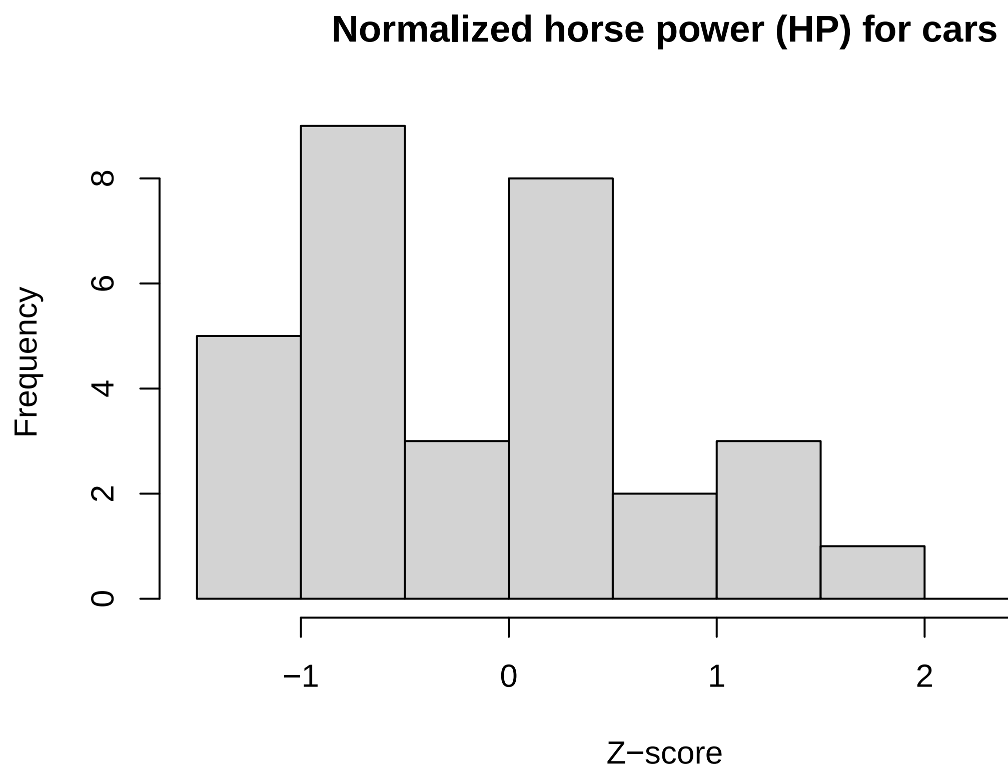
3.3 Measures of relative standing and boxplots

Measures of relative standing, such as percentiles and quartiles, provide information about where specific data points fall within a dataset, offering insights into the relative position of values. Boxplots are graphical representations that display the distribution of data, highlighting the median, quartiles, and potential outliers, making them valuable tools for comparing different datasets by visually assessing their central tendency, spread, and skewness.

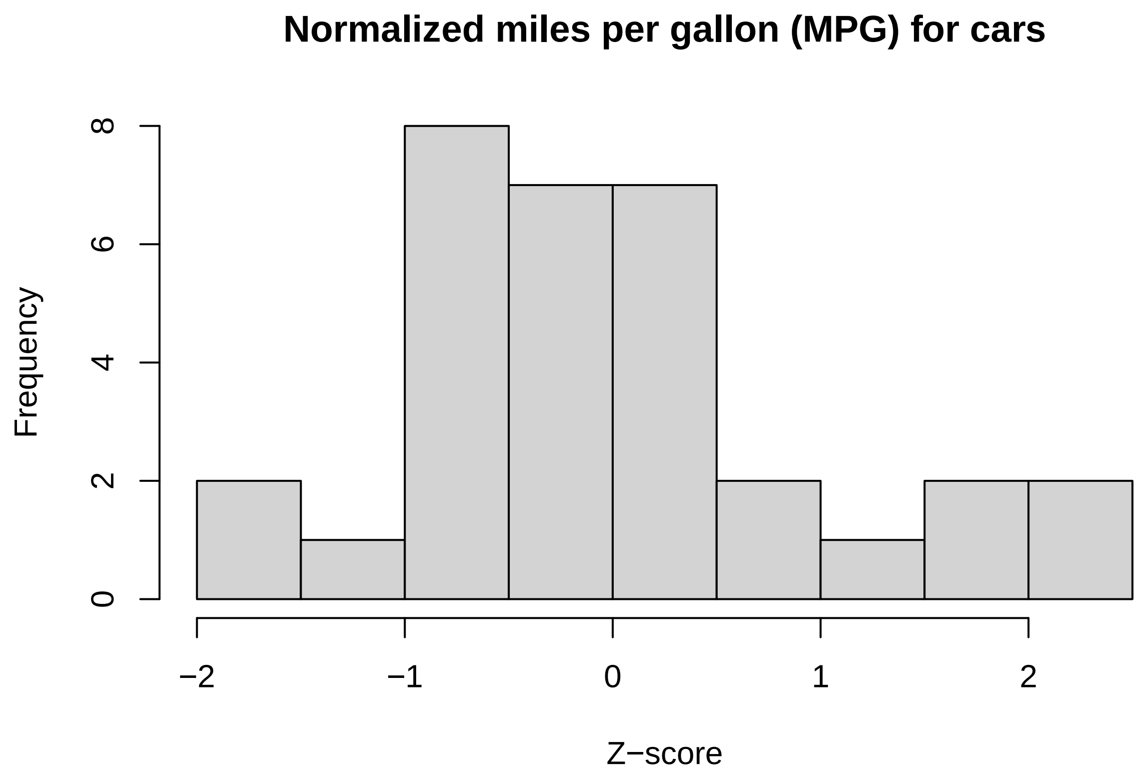
3.3.1 z-Scores

Z-scores, also known as standard scores, standardize individual data points by expressing how many standard deviations they are from the mean, enabling meaningful comparisons and assessments of data points' relative positions within a distribution, regardless of the original scale of the data. Z-scores are valuable for identifying outliers, understanding data distributions, and making statistical inferences, as they provide a common framework for measuring deviations from the mean across different datasets. Let's explore z-scores using the built-in dataset `mtcars` which contains several aspects and performance of several 1973 - 1974 model cars which we studied in the last section. Particularly, let's employ the built-in R command `scale()` to convert our dataset to z-scores which can be plotted in a histogram. Once the two datasets (MPG and HP) are normalized, we will be able to get a better picture of their spread away from the respective means.

```r
# Transform the MPG data to z-scores and store the new data in zcarsHP
zcarsHP = scale(carsHP)

# Transform the MPG data to z-scores and store the new data in zcarsHP
zcarsMPG = scale(carsMPG)

# Generate a histogram of the transformed HP data from mtcars
hist(zcarsHP, main = "Normalized horse power (HP) for cars", xlab = "Z-score")
```



```
# Generate a histogram of the transformed MPG data from mtcars
hist(zcarsMPG, main = "Normalized miles per gallon (MPG) for cars", xlab = "Z-score")
```



Visually, the normalized MPG data is more concentrated around the transformed mean of 0, while the HP data is much more spread out.

Any data point that has a z-score of less than -2 or higher than 2 is considered to be significantly lower or higher, respectively. Let's view our transformed data sets MPG and HP to identify data points that are significantly higher.

```
# Find MPG data points with z-scores higher than 2
outliersMPG = carsMPG[zcarsMPG > 2]

# Find HP data points with z-scores higher than 2
outliersHP = carsHP[zcarsHP > 2]

# Print the data points with z-scores higher than 2
cat("MPG Data with z-scores higher than 2:", outliersMPG, "\n")
```

```
## MPG Data with z-scores higher than 2: 32.4 33.9
```

```
cat("HP Data with z-scores higher than 2:", outliersHP, "\n")
```

```
## HP Data with z-scores higher than 2: 335
```

3.3.2 Percentiles

Percentiles are statistical measures that divide a dataset into 100 equal parts, helping identify values below which a certain percentage of the data falls and enabling comparisons of data points in a ranked order. Let's use the built-in R command `quantile()` with the MPG data from the previous example to compute the 10th, 50th, and 90th percentiles for that dataset.

```r
# Compute 10th, 50th, and 90th percentiles for the MPG dataset
percentiles = c(0.1, 0.5, 0.9)
percentilesMPG = quantile(carsMPG, probs = percentiles)

# Print the data points with z-scores higher than 2
percentilesMPG
```

```
##   10%   50%   90%
## 14.34 19.20 30.09
```

Notice that both of our significantly larger MPG values (i.e., 32.4 and 33.9) both fall above the 90th percentile of the dataset.

3.3.3 Quartiles & the 5-number summary

Quartiles are statistical measures that divide a dataset into four equal parts, with three quartiles (Q1, Q2, Q3) providing insights into the data's spread and central tendencies; they are resistant to outliers, making them robust tools for summarizing data. The 5-number summary is a set of five statistics (minimum, Q1, median, Q3, maximum) that provide a concise description of a dataset's central tendencies and spread. Keeping with our MPG dataset, we will employ the R command `summary()` to give the 5-number summary (which will include Q1, Q2 (also known as the median), & Q3).

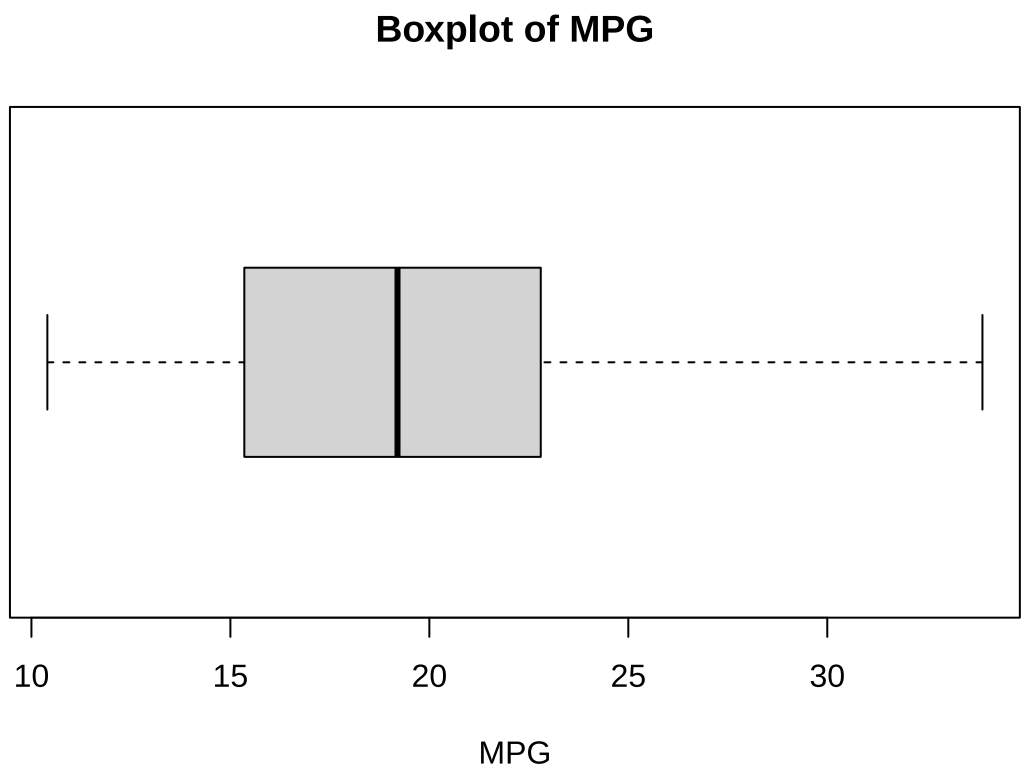
```r
# Compute 5-number summary for MPG data and store it in fiveMPG
fiveMPG = summary(carsMPG)

# Display the 5-number summary
fiveMPG
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   10.40   15.43   19.20   20.09   22.80   33.90
```

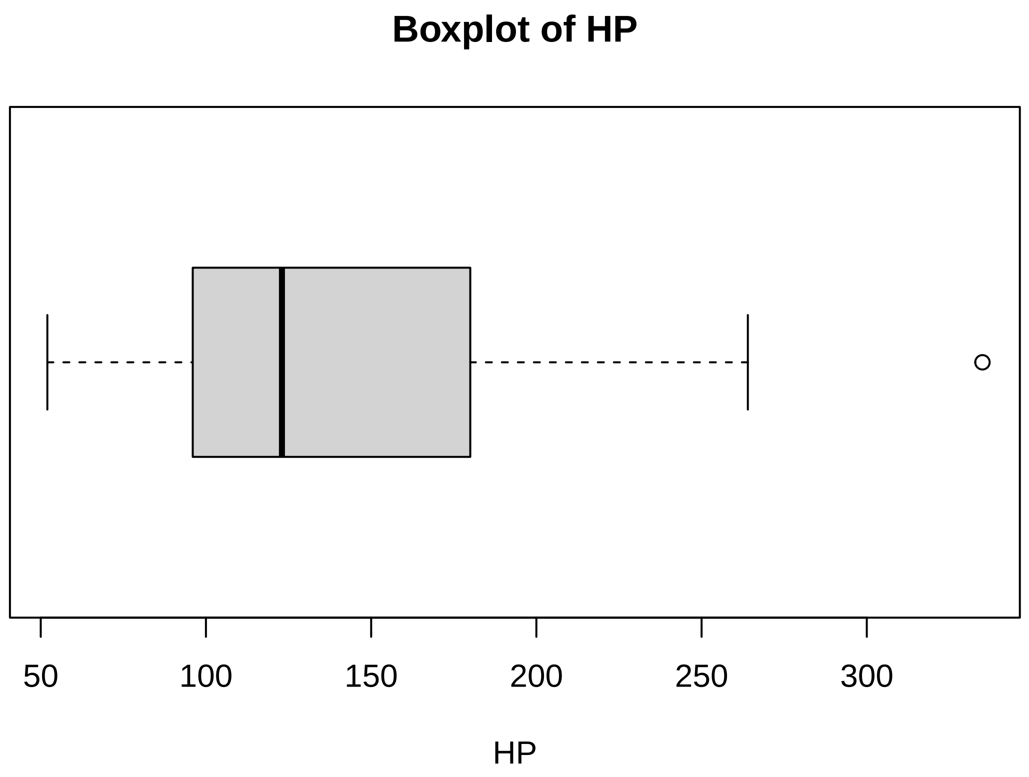
3.3.4 Boxplot

A boxplot, also known as a box-and-whisker plot, is a graphical representation of the five-number summary, displaying the median, quartiles, and potential outliers in a dataset, making it a valuable tool for visualizing the distribution and spread of data. We will employ the R command `boxplot()` to compare the MPG and HP datasets from previous examples. This R command actually creates a modified boxplot by default. Recall the only difference between a regular boxplot and a modified box plot is that data which falls outside of the interquartile range is denoted as an outlier and plotted as an individual point on the graph.

```r
# Generate boxplot for MPG
boxplot(carsMPG, main = "Boxplot of MPG", horizontal = TRUE, xlab = "MPG")
```

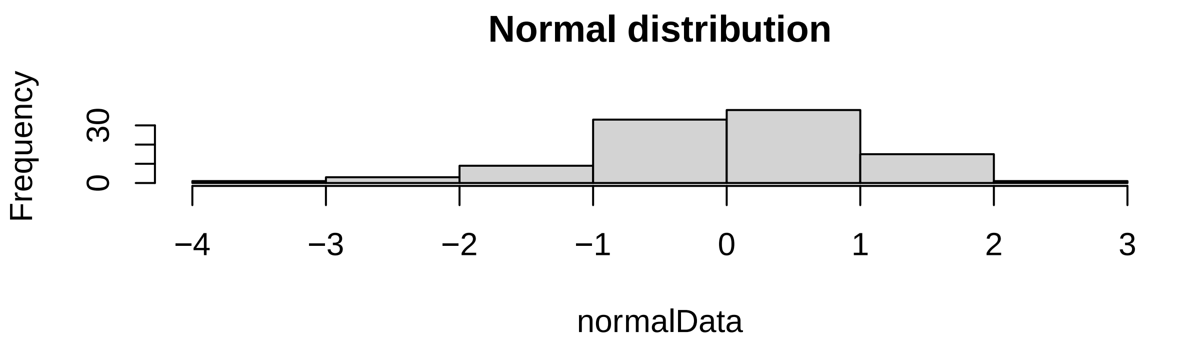


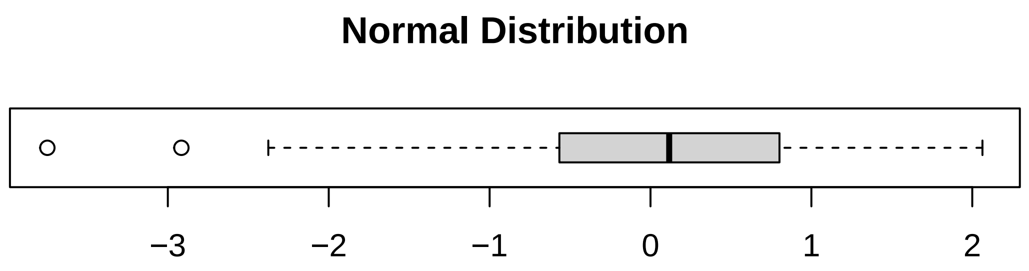
```r
# Generate boxplot for HP
boxplot(carsHP, main = "Boxplot of HP", horizontal = TRUE, xlab = "HP")
```



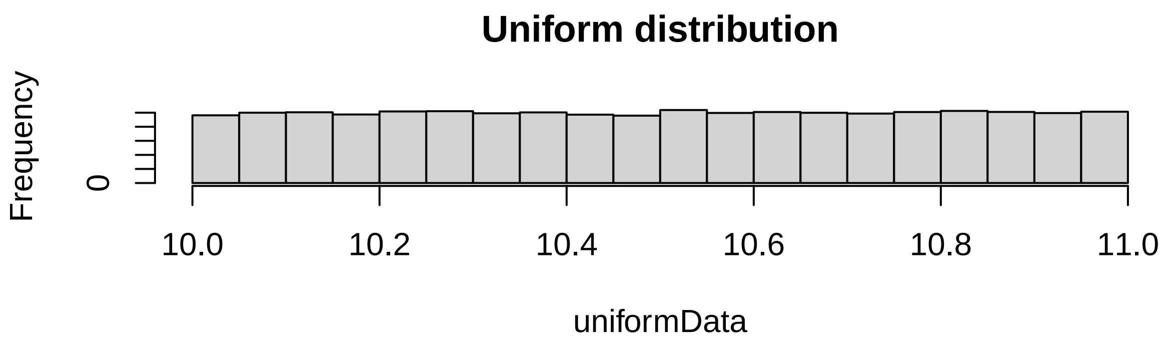
Let’s also compare the boxplots of each of the four datasets for which we explored normal, skewed right, skewed left, and uniform distributions.

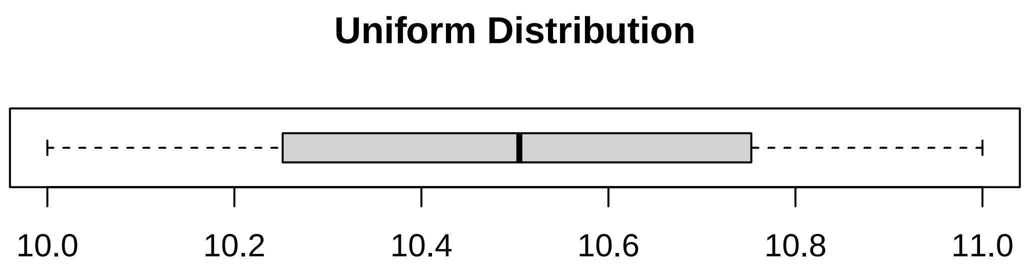
```
# Create histogram/boxplot of normal data
par(mfrow = c(2, 1))
hist(normalData, main = "Normal distribution")
boxplot(normalData, main = "Normal Distribution", horizontal = TRUE)
```



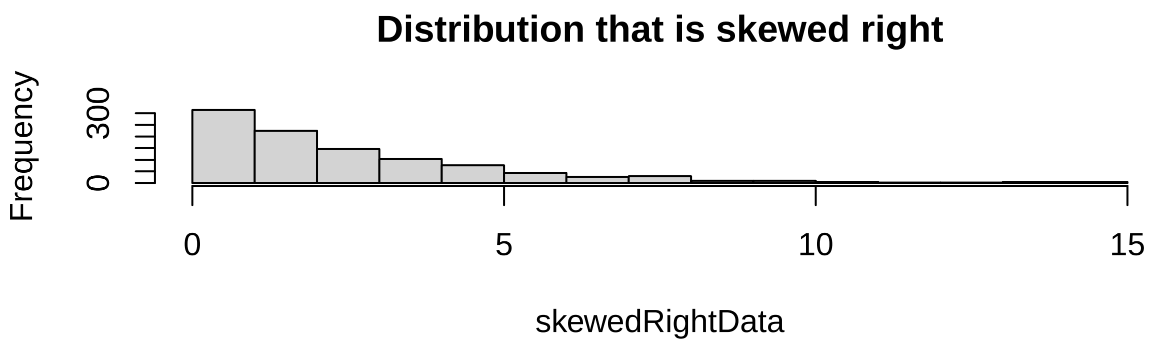


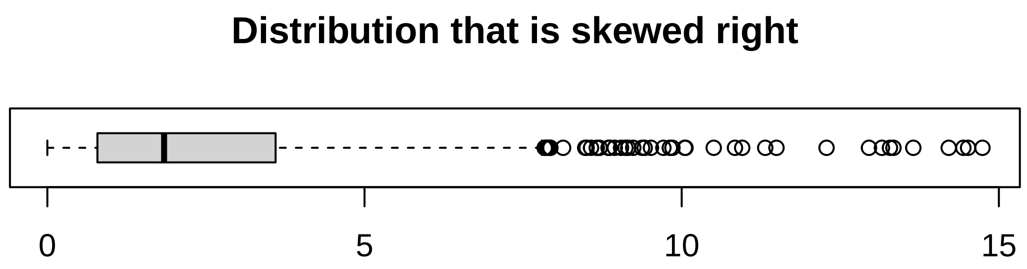
```
# Create histogram/boxplot of uniform data
par(mfrow = c(2, 1))
hist(uniformData, main = "Uniform distribution")
boxplot(uniformData, main = "Uniform Distribution", horizontal = TRUE)
```



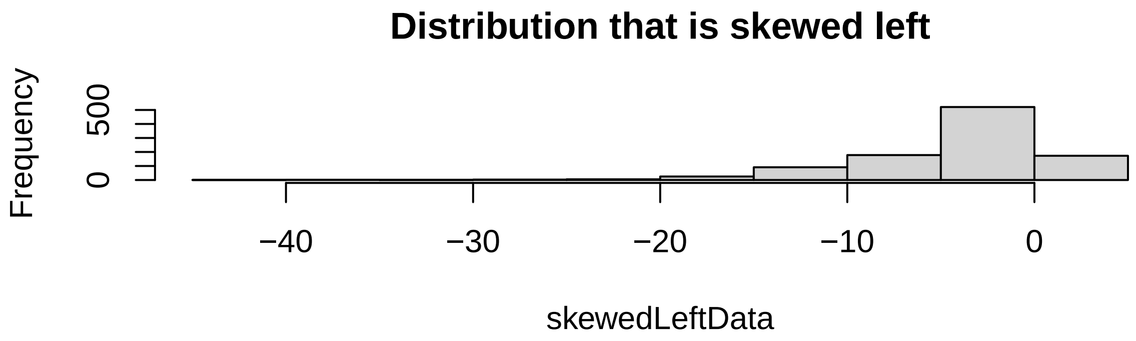


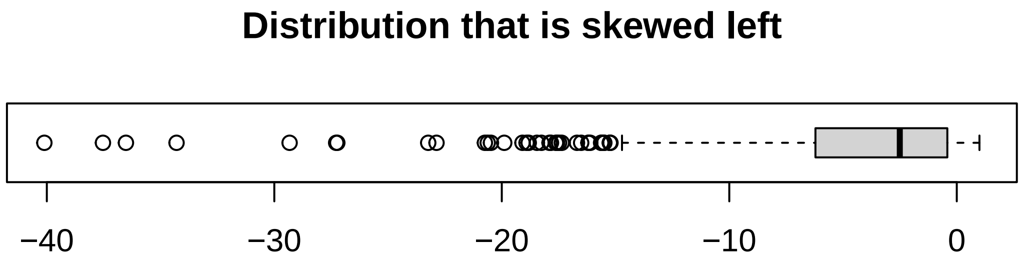
```r
# Create histogram/boxplot of skewed right data
par(mfrow = c(2, 1))
hist(skewedRightData, main = "Distribution that is skewed right")
boxplot(skewedRightData, main = "Distribution that is skewed right", horizontal = TRUE)
```





```r
# Create histogram/boxplot of skewed left data
par(mfrow = c(2, 1))
hist(skewedLeftData, main = "Distribution that is skewed left")
boxplot(skewedLeftData, main = "Distribution that is skewed left", horizontal = TRUE)
```



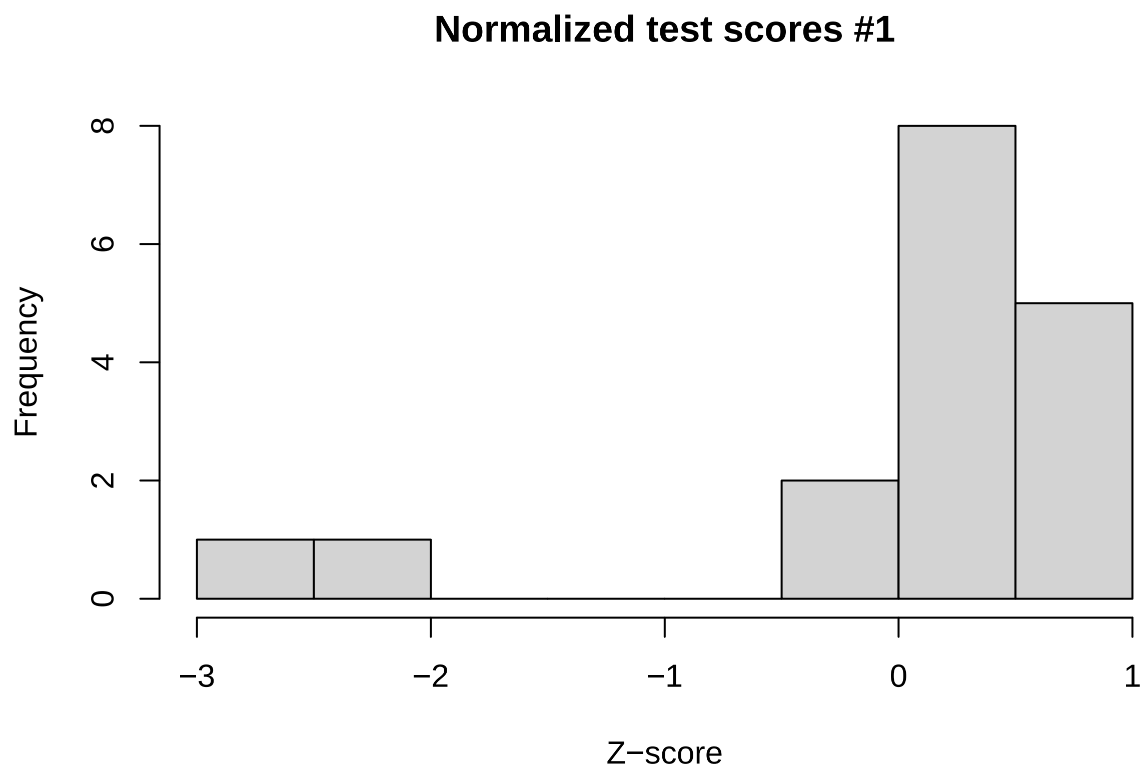


Notice there are a lot of outliers shown on the skewed left & right data. These points are what is causing the long tails on both histograms.

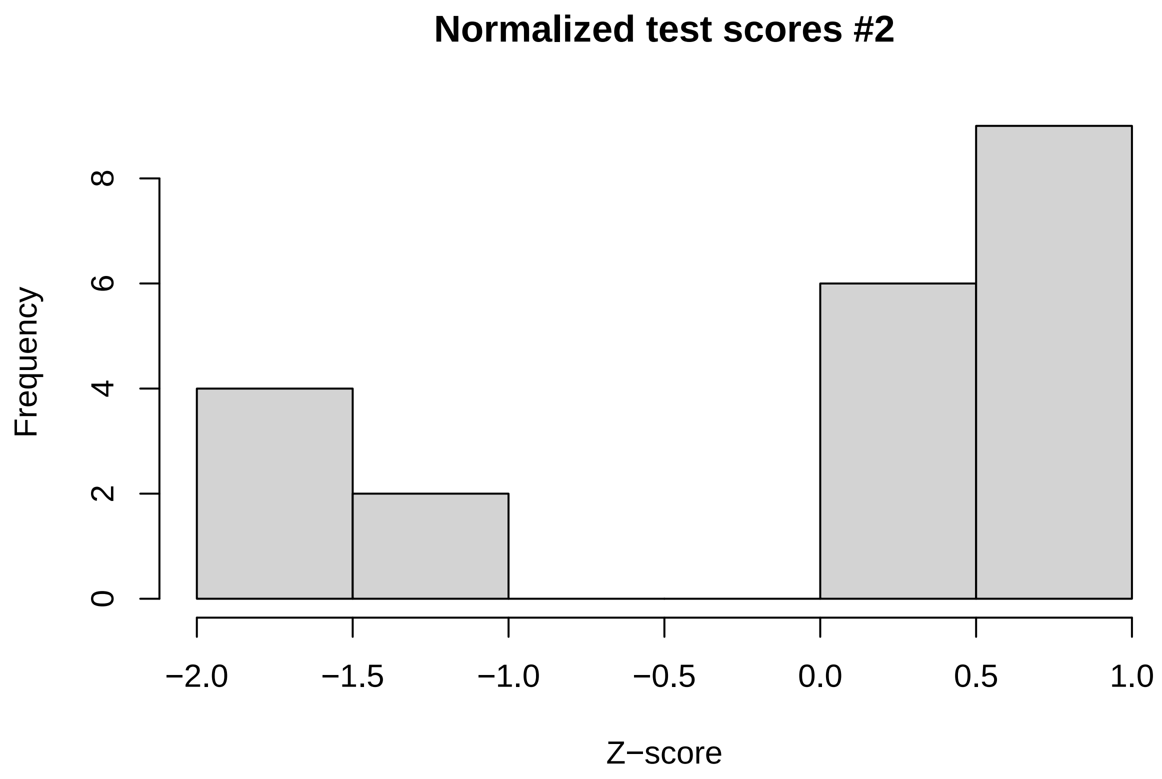
3.3.5 Let's put it all together!

We will use everything we have learned so far in this section to explore the differences between our two test score datasets, i.e., `scores` and `scores2`. These are fictional collections of test scores with `scores2` containing several more extremely low test scores than `scores`. Our first task is to transform the datasets to z-scores and visualize the scaled datasets with a histrogram.

```r
# Transform the scores data to z-scores and store the new data in zscores
zscores = scale(scores)

# Transform the scorres2 data to z-scores and store the new data in
# zscores2
zscores2 = scale(scores2)

# Generate a histogram of the transformed from scores
hist(zscores, main = "Normalized test scores #1", xlab = "Z-score")
```



```r
# Generate a histogram of the transformed from scores2
hist(zscores2, main = "Normalized test scores #2", xlab = "Z-score")
```



Out of the two fictional classes, are there any test scores that are significantly high or low? What can we conclude about those scores? Now, let's compute the 5-number summary for each group of test scores.

```r
# Compute 5-number summary for scores
cat("scores: \n \n")
```

```
## scores:
##
```

```r
summary(scores)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   15.00   75.00   84.00   74.18   87.00   95.00
```

```r
# Compute 5-number summary for scores2
cat("\nscores2: \n \n")
```

```
##
## scores2:
##
```

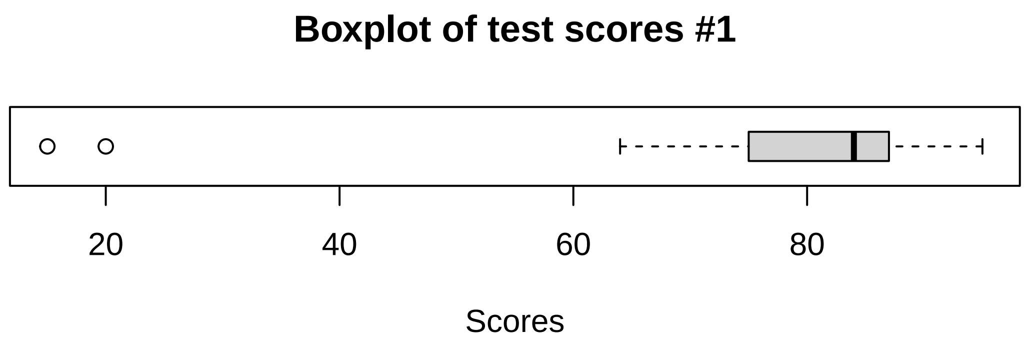
```r
summary(scores2)
```

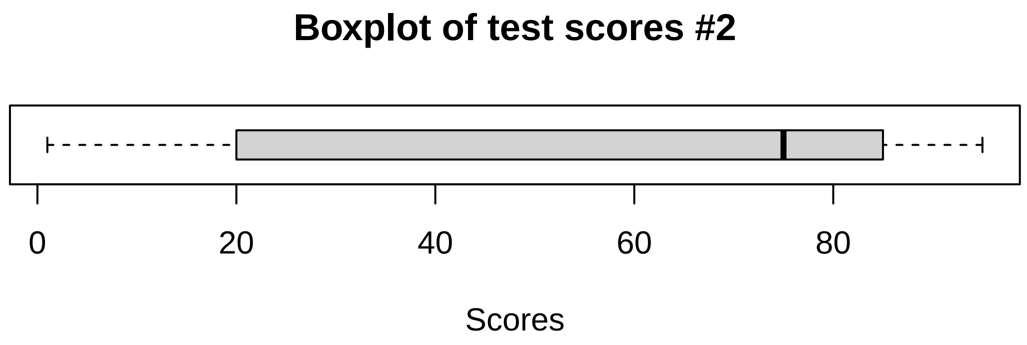
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00   20.00   75.00   60.57   85.00   95.00
```

Finally, let's create boxplots for both datasets and show them on the same plot window for comparison.

```r
# Generate boxplot for both
par(mfrow = c(2, 1))
```

```r
boxplot(scores, main = "Boxplot of test scores #1", horizontal = TRUE, xlab = "Scores")
boxplot(scores2, main = "Boxplot of test scores #2", horizontal = TRUE, xlab = "Scores")
```





What conclusions can we draw regarding the two datasets? If these were two real classes, how would the boxplots help the teacher understand grade performance for the entire class?