CS544 Module 3

1. Module 3 Study Guide and Deliverables

Readings:	Lecture material
Assessments:	Quiz 3 due ...
Assignments:	Assignment 3 due ...

2. Univariate Data

2.1. Types of Data

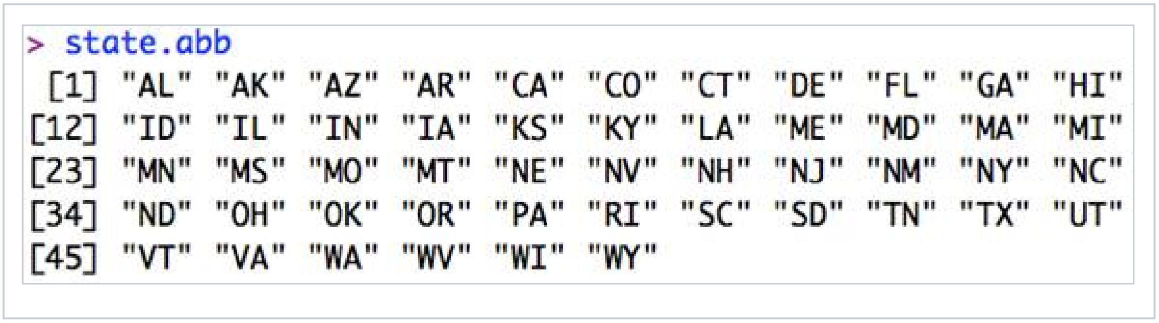
The data that we analyze can be classified into two types:

- Qualitative data, associated with a property or a quality (e.g., gender, eye color, grade, rank, etc.)
- Quantitative data, associated with a numeric measurement (e.g., temperature, age, weight, income, etc.)

The qualitative data, also known as categorical data, can further be classified as:

- Nominal data, where there is no ranking among the distinct categories (e.g., male or female for gender, democrat, republican, or independent for party affiliation, etc.)
- Ordinal data, where there is an implicit ranking among the distinct categories (e.g., sergeant, lieutenant, captain, or chief for the police rank, A, B, C, D, or F for class grade, poor, good, or excellent for customer satisfaction, etc.)

An example of nominal categorical data is the list of states as shown below. The `state.abb` dataset provided by *R* lists the state abbreviations in alphabetical order.



```
> state.abb
 [1] "AL" "AK" "AZ" "AR" "CA" "CO" "CT" "DE" "FL" "GA" "HI"
[12] "ID" "IL" "IN" "IA" "KS" "KY" "LA" "ME" "MD" "MA" "MI"
[23] "MN" "MS" "MO" "MT" "NE" "NV" "NH" "NJ" "NM" "NY" "NC"
[34] "ND" "OH" "OK" "OR" "PA" "RI" "SC" "SD" "TN" "TX" "UT"
[45] "VT" "VA" "WA" "WV" "WI" "WY"
```

The quantitative data, also known as numerical data, can further be classified as:

- Interval data, where the values are ordered and the difference in the values is meaningful (e.g., temperature in Fahrenheit. If the morning temperature is 60 and the afternoon temperature is 70, it is 10 degrees warmer. If the average temperatures in May and December are 90 and 45, respectively, we cannot infer that May is twice as warm as December). We can only add or subtract with interval data—we cannot multiply or divide.

- Ratio data, where the values are ordered and can be compared using addition, subtraction, multiplication, or division (e.g., age in years. We can compare twice as old, two years older, etc.)

2.2. Example Datasets for *R*

Some of the datasets are in the basic `datasets` package. Other datasets are in the `UsingR` package. For working with the `UsingR` datasets, the library is first loaded as follows.

```
> library(UsingR)
```

2.3. Univariate Data

Univariate datasets deal with data contained in a single variable. Even if the dataset consists of the data for multiple variables, univariate analysis examines one variable at a time. The data can be categorical data or numerical data. Summary descriptions and visual representations of the data are presented in the following sections.

The problems in this module require the use of the following packages:

UsingR

Use the following commands to load the 'BushApproval' data set into memory. The following problems will use this data set. Feel free to explore the data and data types present before proceeding.

library(UsingR)

data <- BushApproval

Test Yourself 3.1

Assign the 'date' column to a new variable. Show the R commands.

x <- data$date

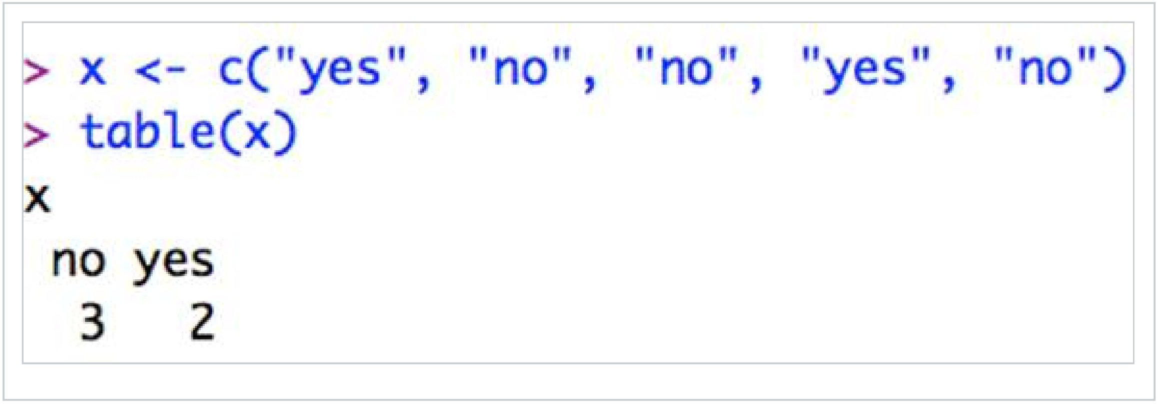
Test Yourself 3.2

What kind of data is date?

Quantitative and Interval data

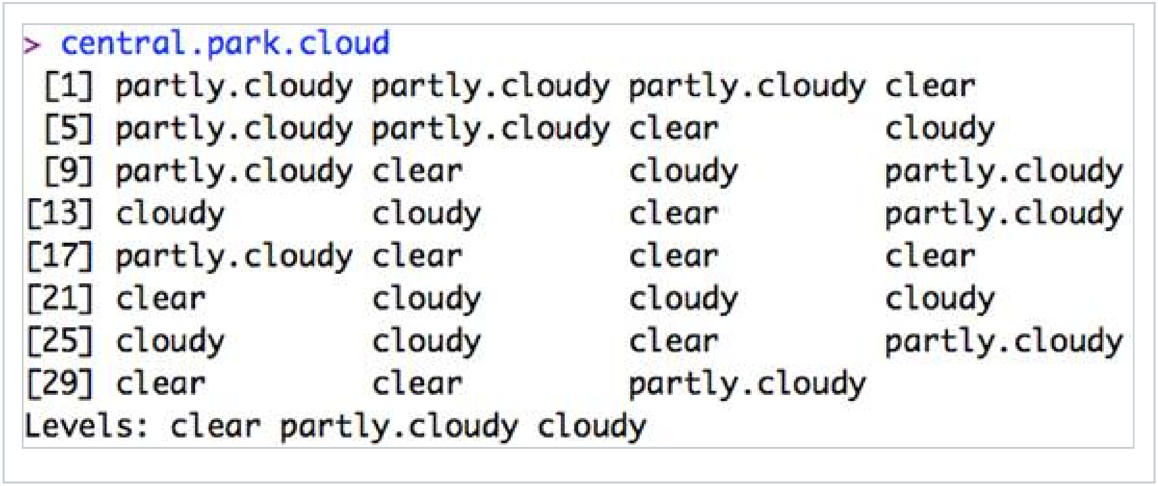
2.4. Categorical Data

Tables are frequently used for summarizing the categorical data for non-visual representations. Given a vector of values, x, the function `table(x)` tabulates the frequencies for all the unique values in the data vector x. The function can be applied for both categorical data and numerical data.



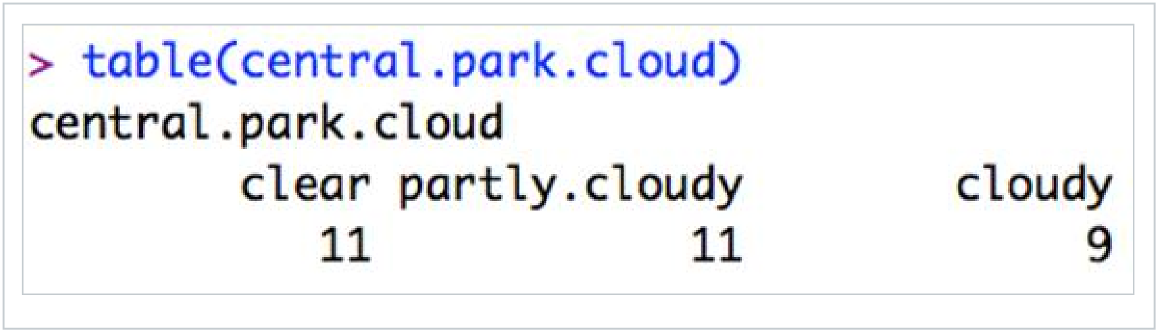
```
> x <- c("yes", "no", "no", "yes", "no")
> table(x)
x
 no yes
  3   2
```

The dataset `central.park.cloud` (from the `UsingR` package) contains the information about the type of day (`clear`, `partly.cloudy`, or `cloudy`) for Central Park, NY, during the month of May 2003. The dataset is shown below.



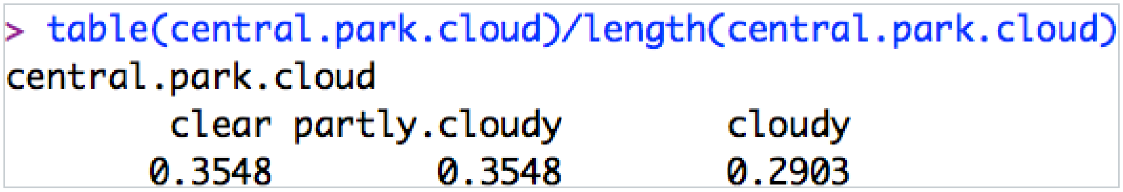
```
> central.park.cloud
 [1] partly.cloudy partly.cloudy partly.cloudy clear
 [5] partly.cloudy partly.cloudy clear         cloudy
 [9] partly.cloudy clear         cloudy        partly.cloudy
[13] cloudy        cloudy        clear         partly.cloudy
[17] partly.cloudy clear         clear         clear
[21] clear         cloudy        cloudy        cloudy
[25] cloudy        cloudy        clear         partly.cloudy
[29] clear         clear         partly.cloudy
Levels: clear partly.cloudy cloudy
```

The `table` function provides a more meaningful summary, showing the frequencies of the types of days.



```
> table(central.park.cloud)
central.park.cloud
        clear partly.cloudy        cloudy
           11            11             9
```

The proportion of each of these categories with respect to the total size of the dataset can be calculated as follows.



```
> table(central.park.cloud)/length(central.park.cloud)
central.park.cloud
        clear partly.cloudy        cloudy
       0.3548        0.3548        0.2903
```

Test Yourself 3.3

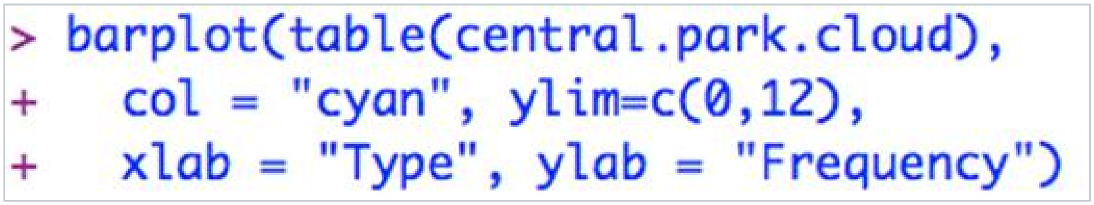
Which column represents categorical data? Show the R commands needed to see the levels of categorical data.

The 'who' column has categorical data.

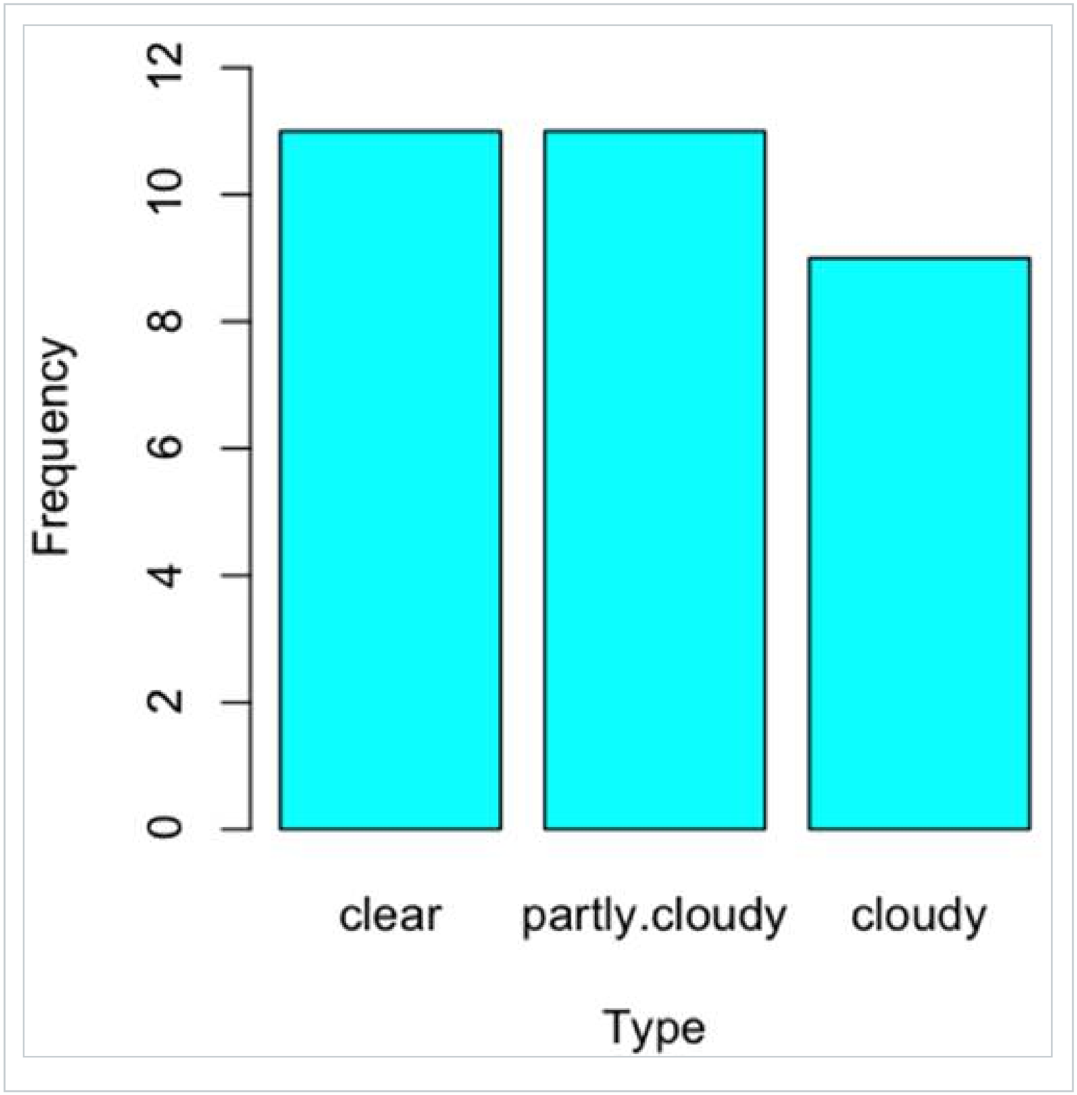
```
x <- data$who
levels(x)
```

2.5. Graphical Representations of Categorical Data

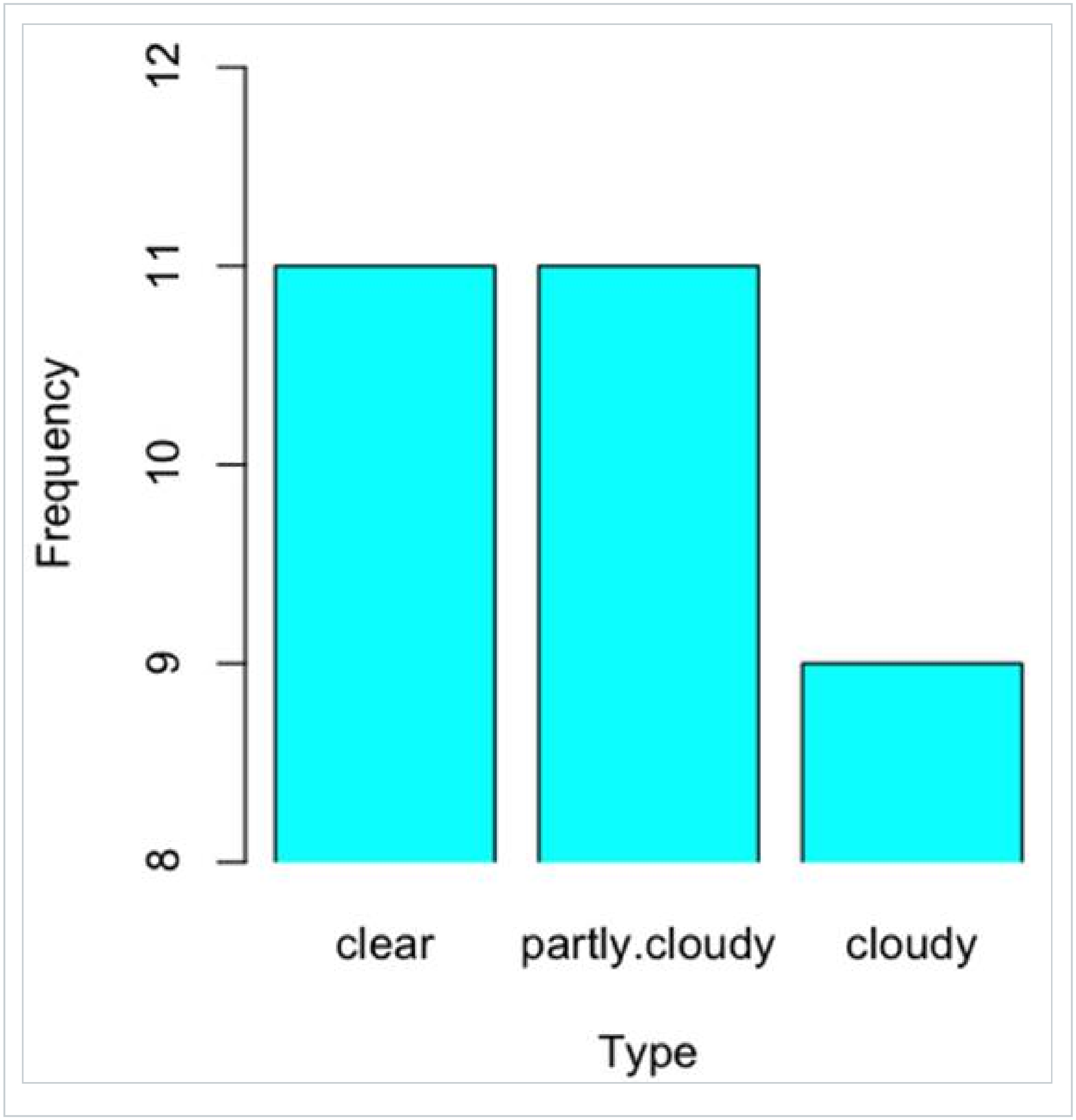
The frequencies of the various categories can be graphically represented using a barplot. By default, the bars are drawn vertically with the first data bar at the left. The names of the categories are used as the labels for the bars.



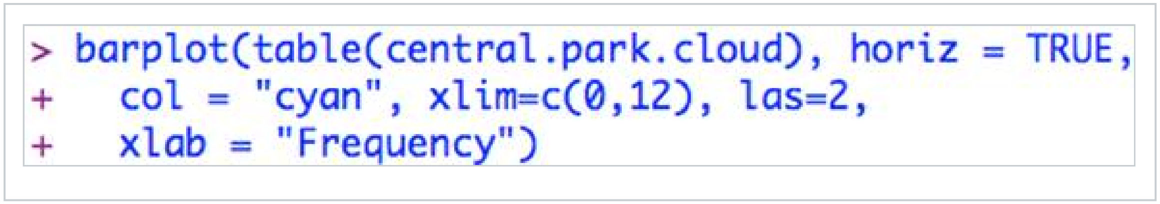
```
> barplot(table(central.park.cloud),
+   col = "cyan", ylim=c(0,12),
+   xlab = "Type", ylab = "Frequency")
```



A barplot could be misleading if the ranges of the frequencies are not properly shown. The same data would produce the following chart if the y-axis were limited to the range 8 to 12.

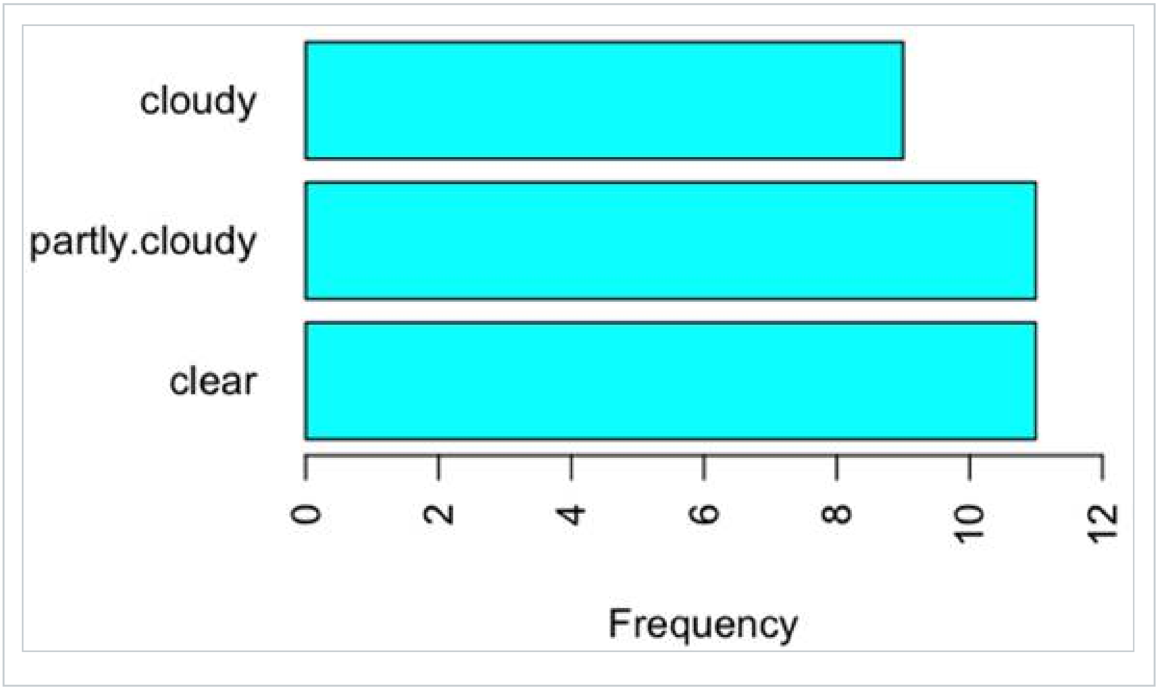


The barplot can be created with the bars shown horizontally; the option is specified as shown below.

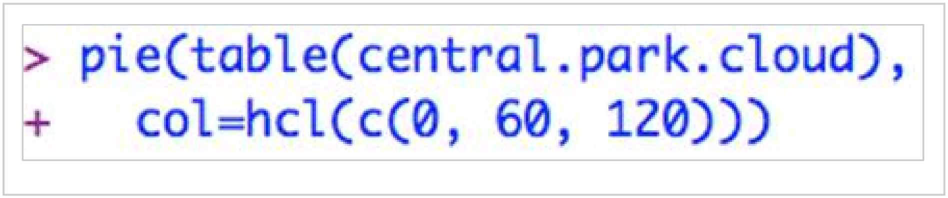


```
> barplot(table(central.park.cloud), horiz = TRUE,
+   col = "cyan", xlim=c(0,12), las=2,
+   xlab = "Frequency")
```

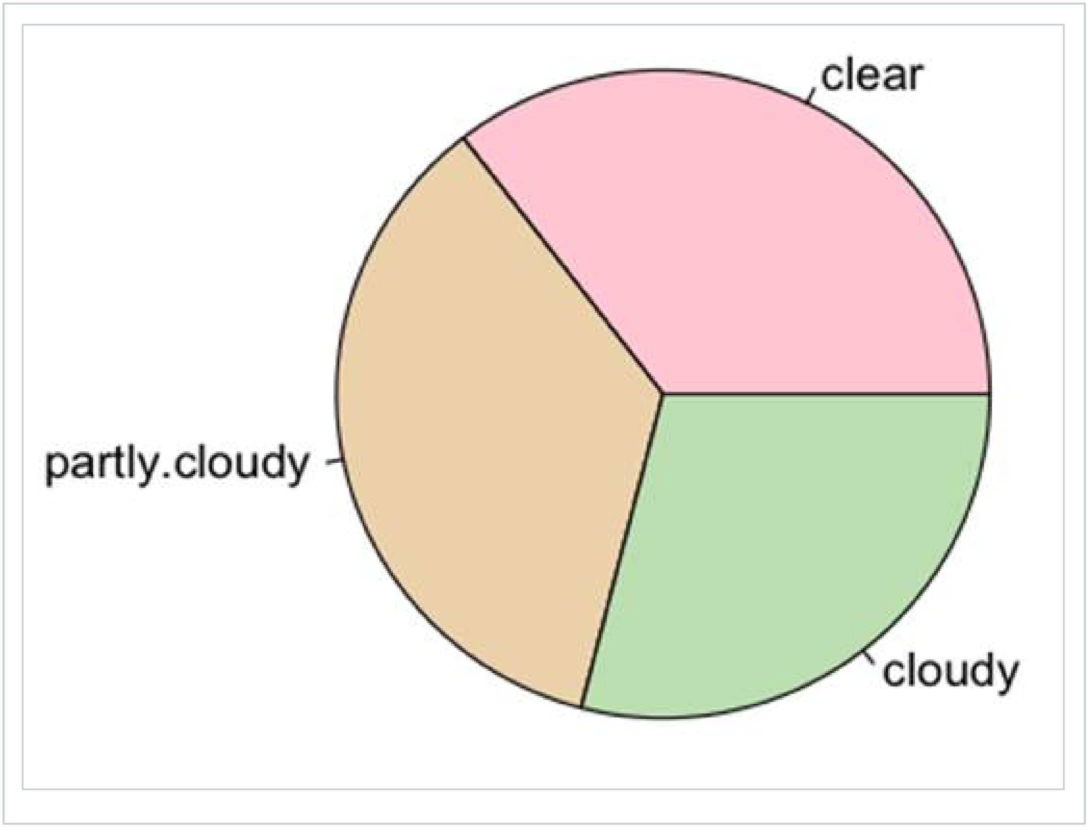
In this scenario, the data for the first bar is at the bottom of the plot.



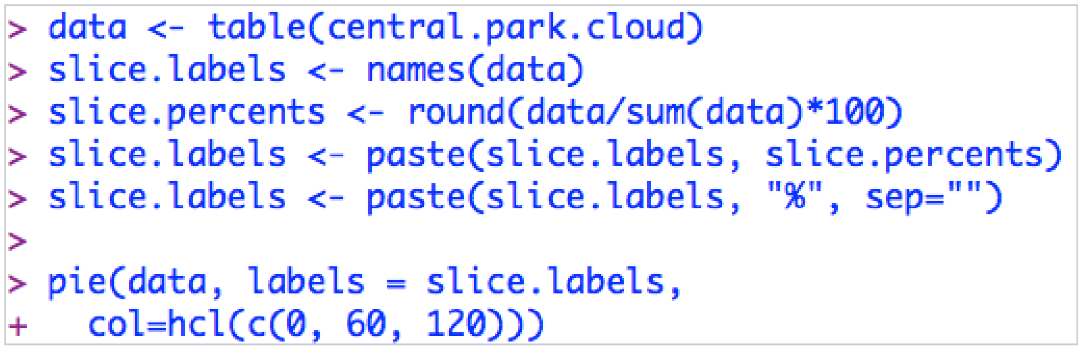
A pie chart of the data is useful for comparing the proportions of each of the categories.



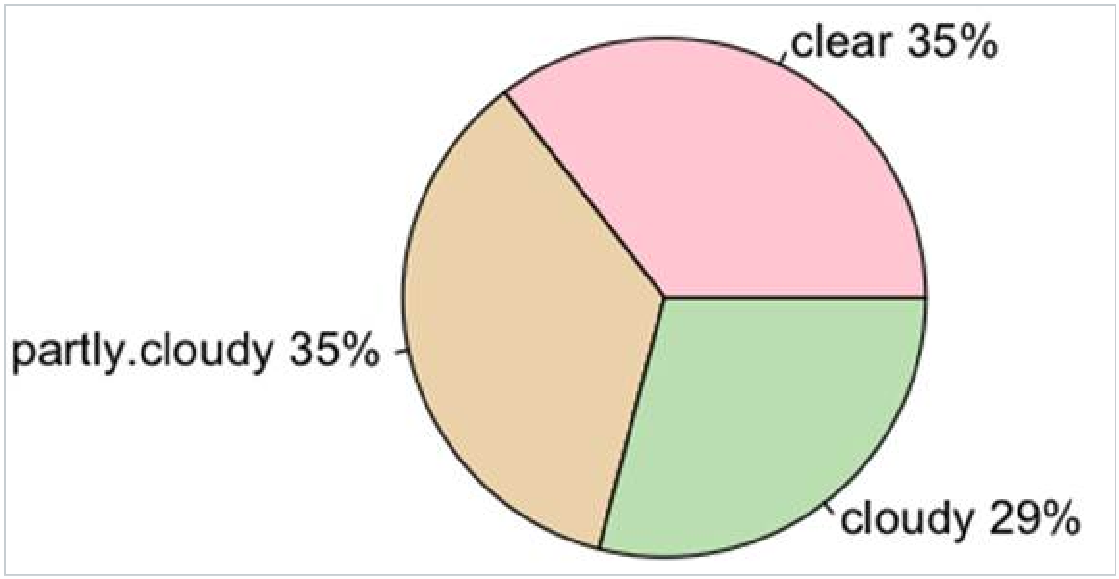
```
> pie(table(central.park.cloud),
+   col=hcl(c(0, 60, 120)))
```



The labels for the pie chart can be customized as shown below. The proportion of each category is computed and appended to the respective name.



```
> data <- table(central.park.cloud)
> slice.labels <- names(data)
> slice.percents <- round(data/sum(data)*100)
> slice.labels <- paste(slice.labels, slice.percents)
> slice.labels <- paste(slice.labels, "%", sep="")
>
> pie(data, labels = slice.labels,
+   col=hcl(c(0, 60, 120)))
```



Test Yourself 3.4

Create a bar chart using the data from the 'who' column. Label the columns appropriately. Show the R commands.

```
y <- data$who
barchart(y, xlab='frequency', ylab='who')
```

Test Yourself 3.5

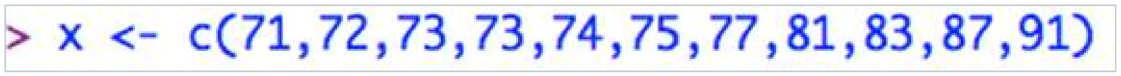
Create a pie chart of the data from the 'who' column. Label the slices appropriately. Show the R commands.

```
y <-table(data$who)
pie(y)
```

2.6. Numerical Data

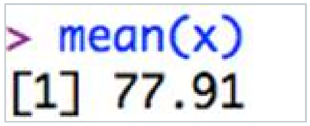
For the variables that hold numerical data, the common summaries of the variables are described using the measures of center and spread. The common measures of central tendency are the mean, the median, and the mode.

Consider the following dataset showing the grades of 11 students in an exam. The data is already in the sorted order.



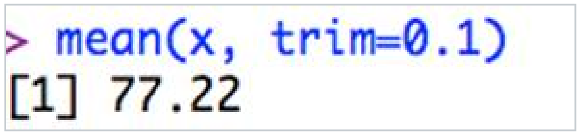
```
> x <- c(71,72,73,73,74,75,77,81,83,87,91)
```

The mean of the above variable is computed as shown below.



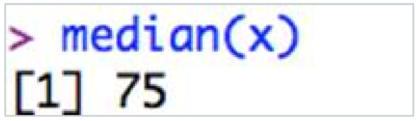
```
> mean(x)
[1] 77.91
```

In scenarios where the ends of the data contain extremely low and extremely high values, the mean of the trimmed dataset can be computed as shown below. In the following case, 10% of the data is trimmed from both the ends when computing the mean for illustration only.



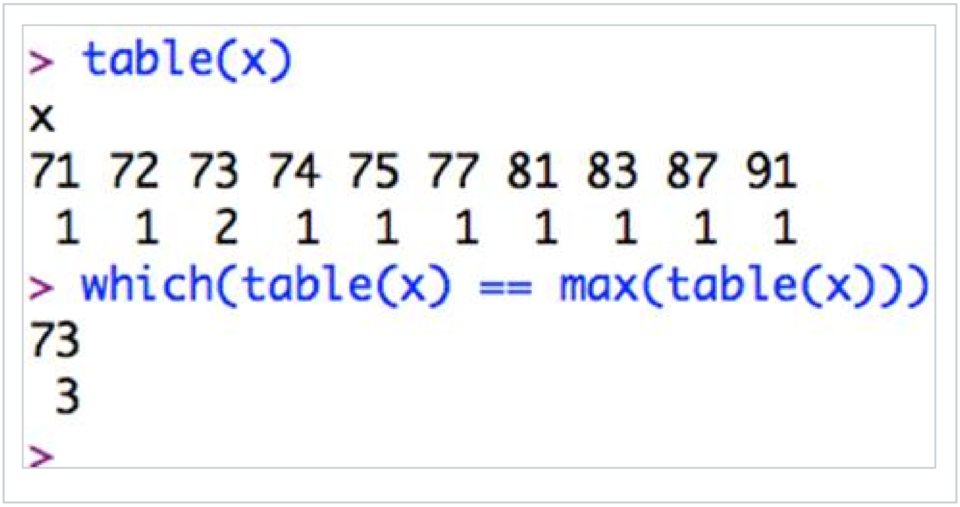
```
> mean(x, trim=0.1)
[1] 77.22
```

The median of the variable is computed as shown below.



```
> median(x)
[1] 75
```

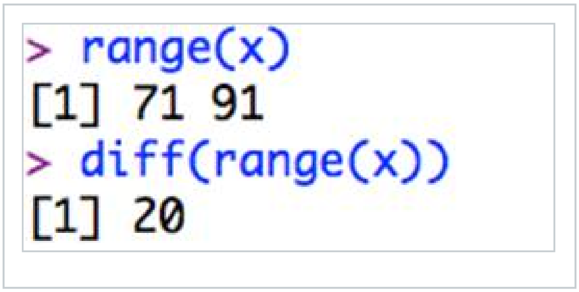
The mode of the variable is computed indirectly as there is no direct function in *R*. The `table` function shows the frequencies of the unique values in the given data. The maximum value is the mode.



```
> table(x)
x
71 72 73 74 75 77 81 83 87 91
 1  1  2  1  1  1  1  1  1  1
> which(table(x) == max(table(x)))
73
 3
>
```

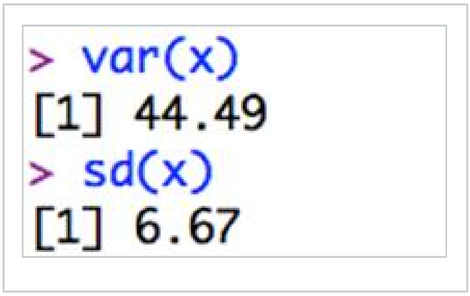
The value 73 occurs twice, which is the maximum occurrence. The second value shown by the `which` function is the index of the value in the tabular representation.

The common measures of spread are the range, the variance, the standard deviation, and the interquartile range. The range of the variable is the difference between the maximum and the minimum value. The function, `range`, returns the values as a pair.



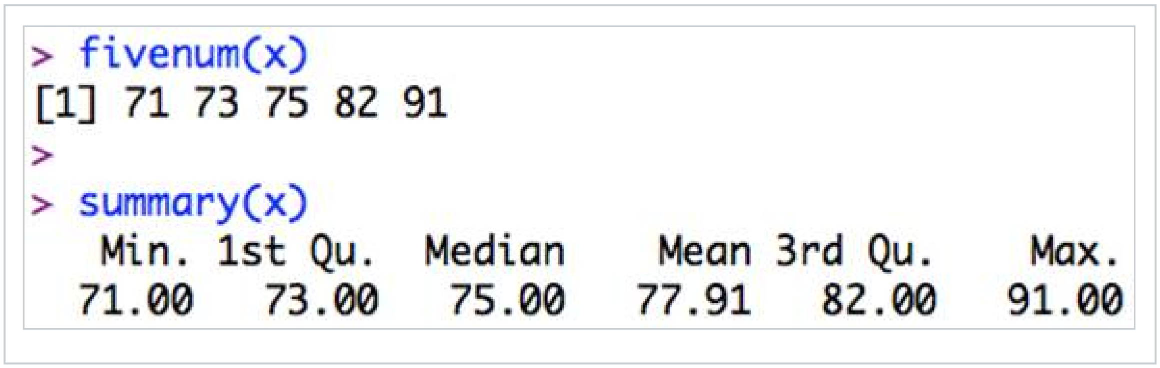
```
> range(x)
[1] 71 91
> diff(range(x))
[1] 20
```

The variance and the standard deviation of the variable are computed as shown below.



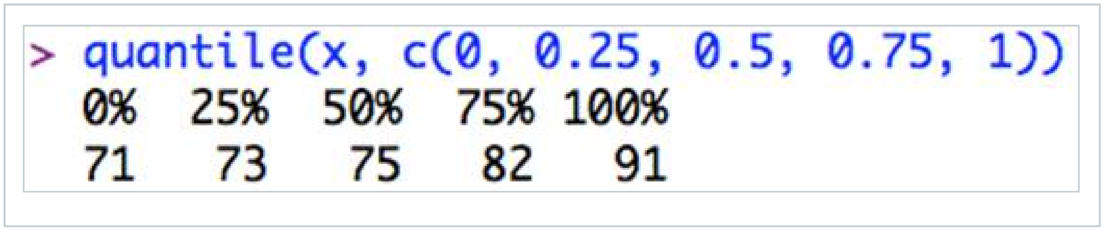
```
> var(x)
[1] 44.49
> sd(x)
[1] 6.67
```

The interquartile range represents the middle 50% of the data. This value is computed as the difference between the third quartile (75 percentile) and the first quartile (25 percentile). The function `fivenum` gives the five-number summary of the data (minimum, first quartile/lower hinge, median, third quartile/upper hinge, maximum). The `summary` function gives the mean in addition to the above five numbers.



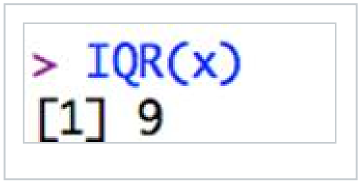
```
> fivenum(x)
[1] 71 73 75 82 91
>
> summary(x)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  71.00   73.00   75.00   77.91   82.00   91.00
```

The `quantile` function can be used to calculate explicitly the appropriate percentiles.



```
> quantile(x, c(0, 0.25, 0.5, 0.75, 1))
  0%  25%  50%  75% 100%
  71   73   75   82   91
```

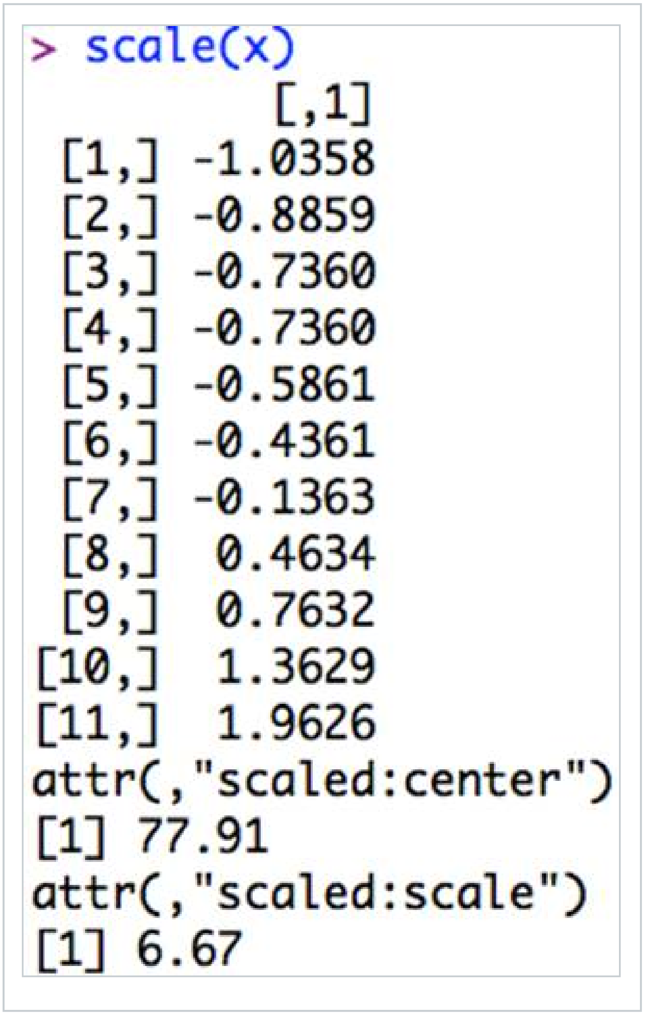
The `IQR` function can also be used to directly compute the interquartile range.



```
> IQR(x)
[1] 9
```

The above value is the difference between the third quartile, 82, and the first quartile, 73.

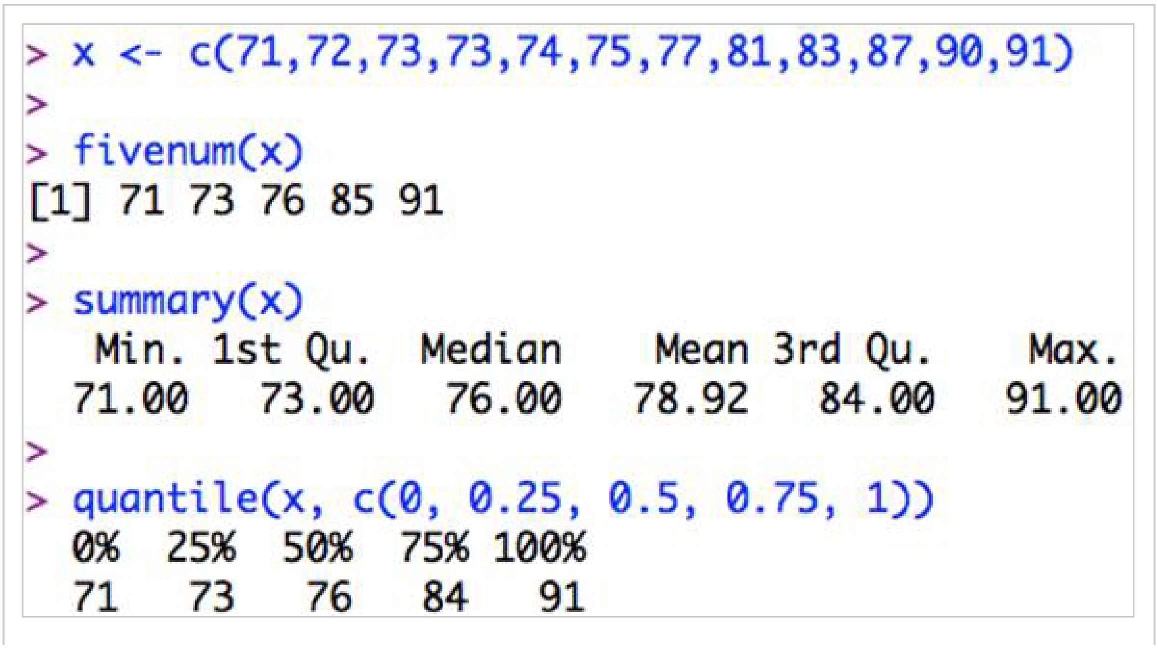
The z-scores of the data show the number of standard deviations the value is from the mean of the data set. The above data has a mean value 77.91 and a standard deviation of 6.67. The `scale` function computes the z-scores of the given data.



```
> scale(x)
         [,1]
 [1,] -1.0358
 [2,] -0.8859
 [3,] -0.7360
 [4,] -0.7360
 [5,] -0.5861
 [6,] -0.4361
 [7,] -0.1363
 [8,]  0.4634
 [9,]  0.7632
[10,]  1.3629
[11,]  1.9626
attr(,"scaled:center")
[1] 77.91
attr(,"scaled:scale")
[1] 6.67
```

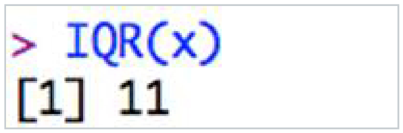
The z-scores of a given dataset have a mean of 0 and a standard deviation of 1.

In the above example, which had an odd number of values, the `fivenum` and `summary` functions produced the same results. However, these two functions behave differently when computing the quartiles for even number of values as shown below.



```
> x <- c(71,72,73,73,74,75,77,81,83,87,90,91)
>
> fivenum(x)
[1] 71 73 76 85 91
>
> summary(x)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  71.00   73.00   76.00   78.92   84.00   91.00 
>
> quantile(x, c(0, 0.25, 0.5, 0.75, 1))
  0%  25%  50%  75% 100% 
  71   73   76   84   91 
```

The quartiles shown in the `summary` function correspond to the values directly computed using the `quantile` function. The interquartile range function also uses these values.



```
> IQR(x)
[1] 11
```

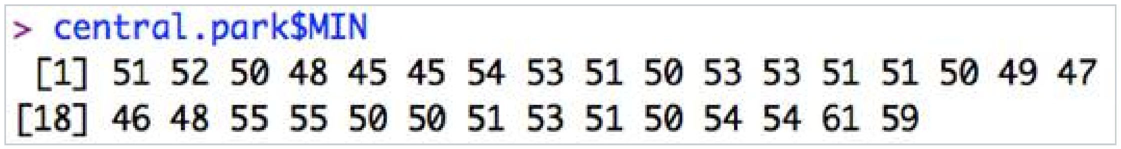
Test Yourself 3.6

Assign the 'approval' column to a new variable. What is the mean approval? What is the median approval? What is the variance of the approval? Produce a summary using the summary() function. Show all R commands.

```
x <- data$approval
mean(x)
median(x)
var(x)
summary(x)
```

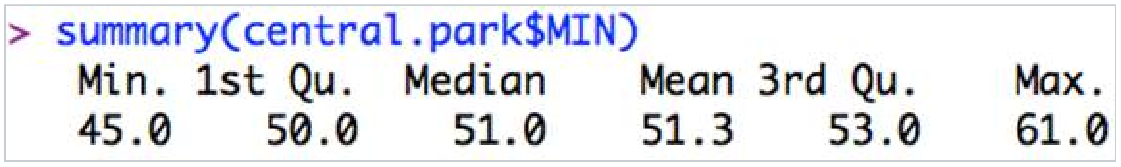
2.7. Graphical Representations of Numerical Data

The following dataset variable (`central.park` from the `UsingR` package) shows the minimum temperatures observed in Central Park (New York City) during May 2003.



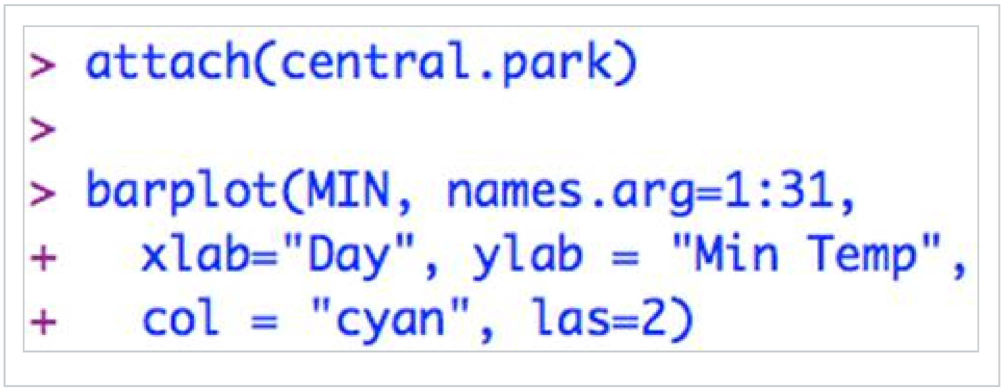
```
> central.park$MIN
 [1] 51 52 50 48 45 45 54 53 51 50 53 53 51 51 50 49 47
[18] 46 48 55 55 50 50 51 53 51 50 54 54 61 59
```

The summary function shows the various measures of the data.

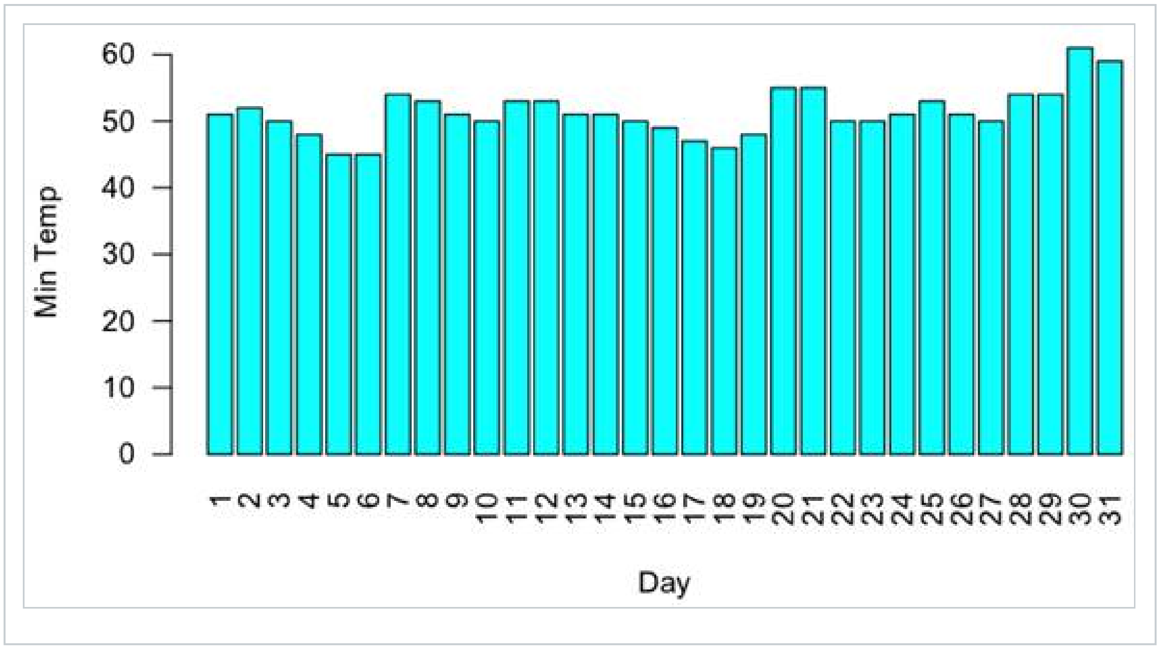


```
> summary(central.park$MIN)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   45.0    50.0    51.0    51.3    53.0    61.0
```

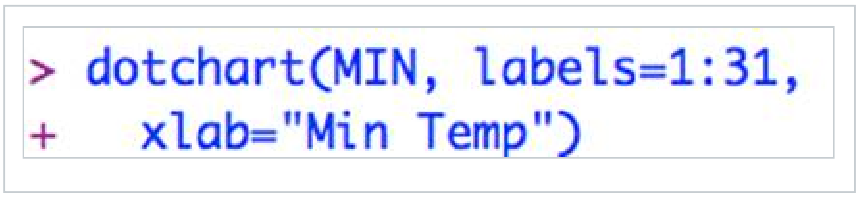
A barplot showing the daily minimum temperatures is plotted as shown below.



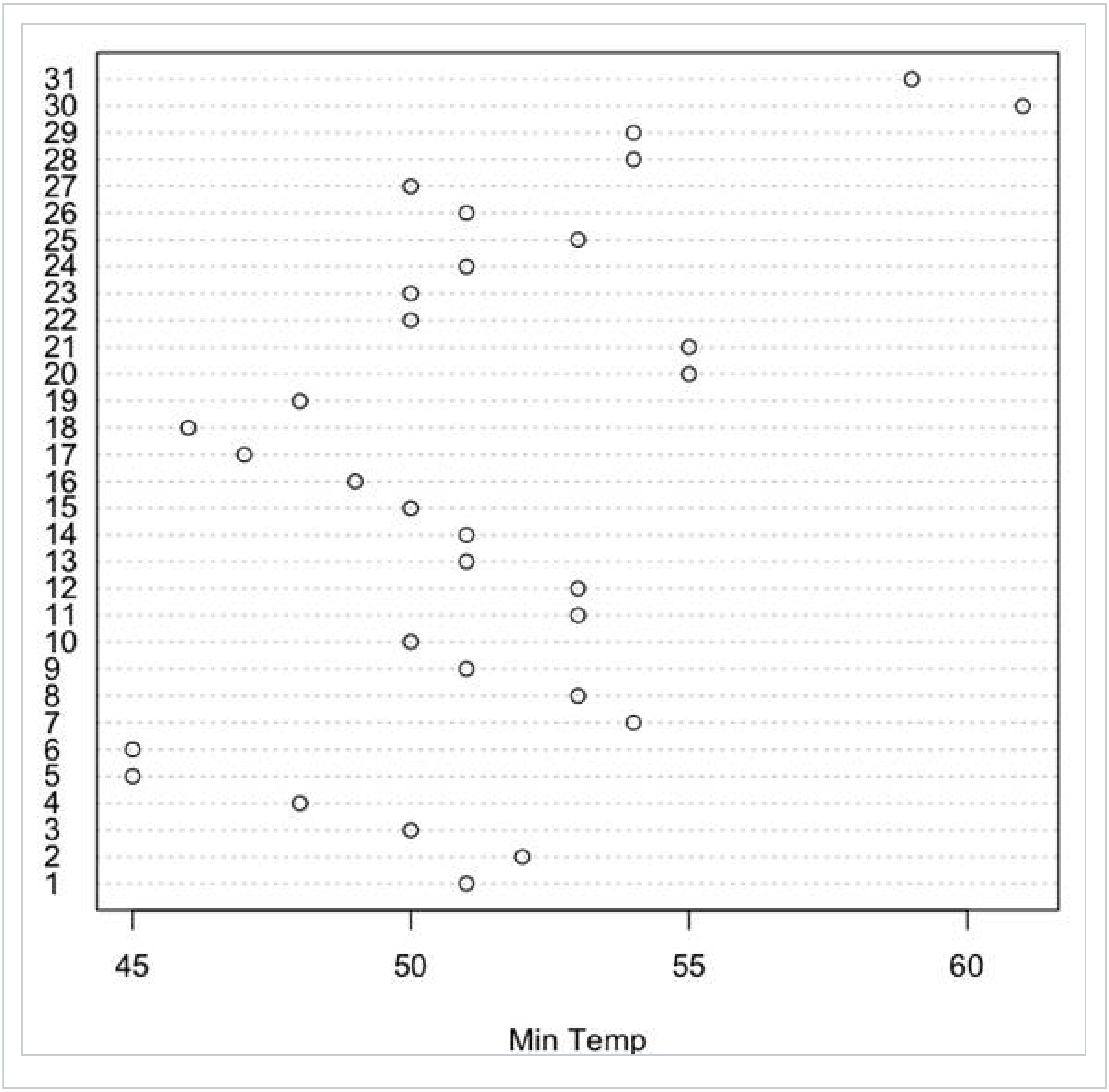
```
> attach(central.park)
>
> barplot(MIN, names.arg=1:31,
+   xlab="Day", ylab = "Min Temp",
+   col = "cyan", las=2)
```



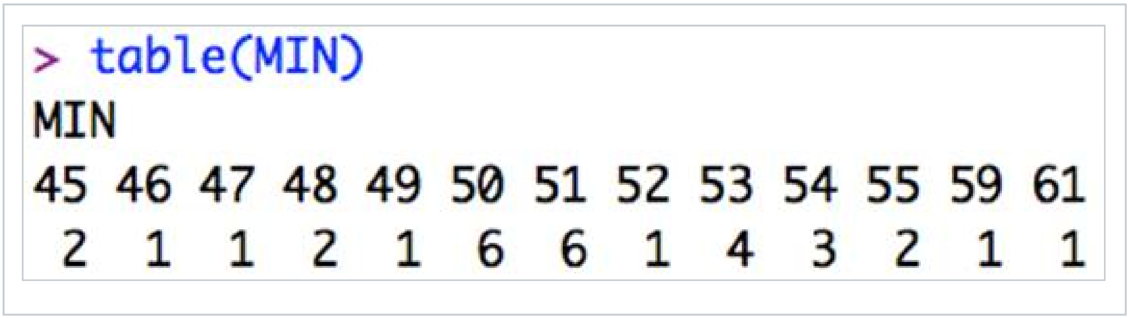
The `dotchart` is sometimes used as a substitute for the barplot. The values of the variable are shown as dots horizontally over the range of the data.



```
> dotchart(MIN, labels=1:31,
+   xlab="Min Temp")
```

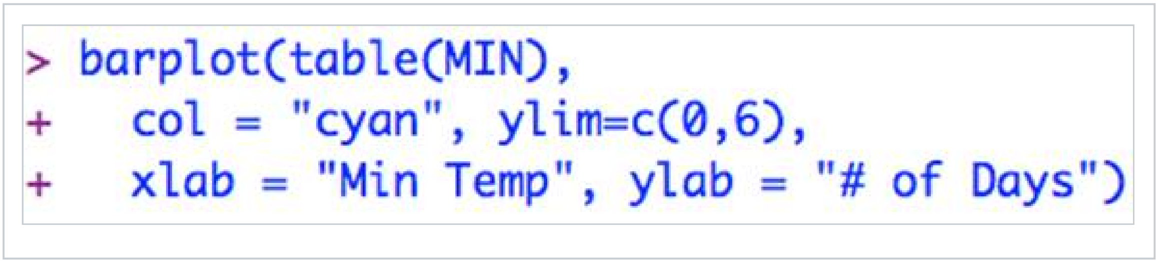


The `table` function may also be used with the numeric data for finding the frequencies of the unique values in the data.

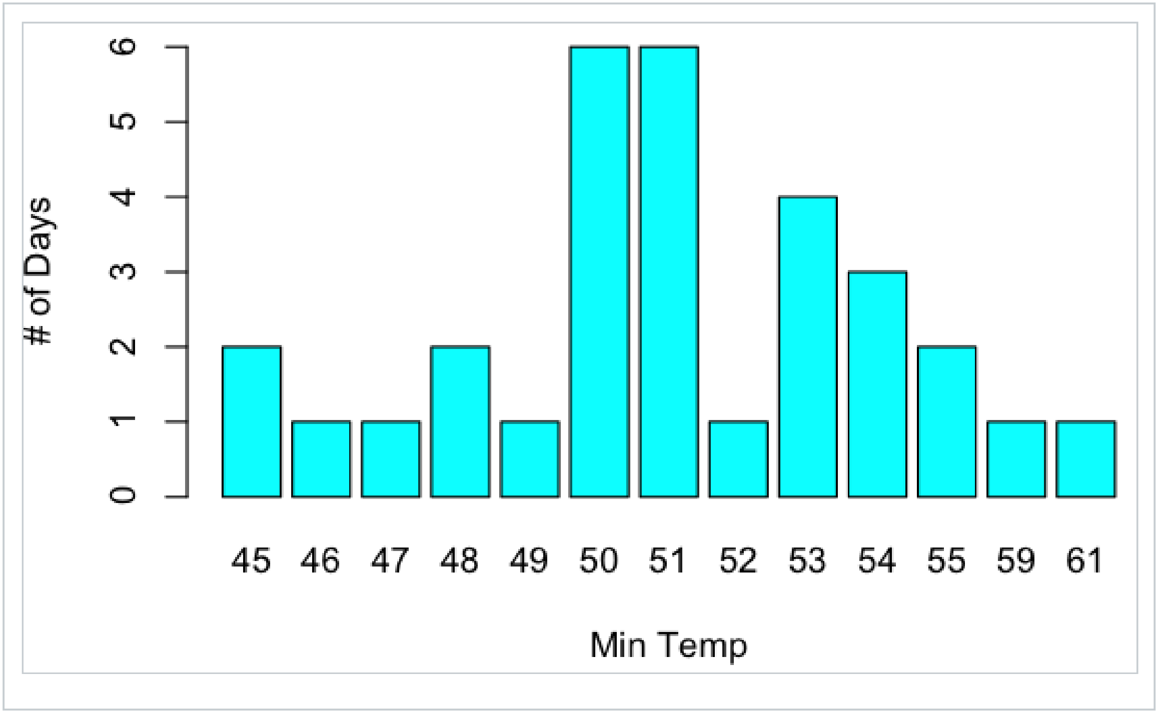


```
> table(MIN)
MIN
45 46 47 48 49 50 51 52 53 54 55 59 61
 2  1  1  2  1  6  6  1  4  3  2  1  1
```

A barplot showing these frequencies is seen below. The heights of the bars show how many days in the month had that particular minimum temperature.



```
> barplot(table(MIN),
+    col = "cyan", ylim=c(0,6),
+    xlab = "Min Temp", ylab = "# of Days")
```



Test Yourself 3.7

Produce a dot plot of the approvals. Show the R commands.

```
x <- data$approval
dotchart(x)
```

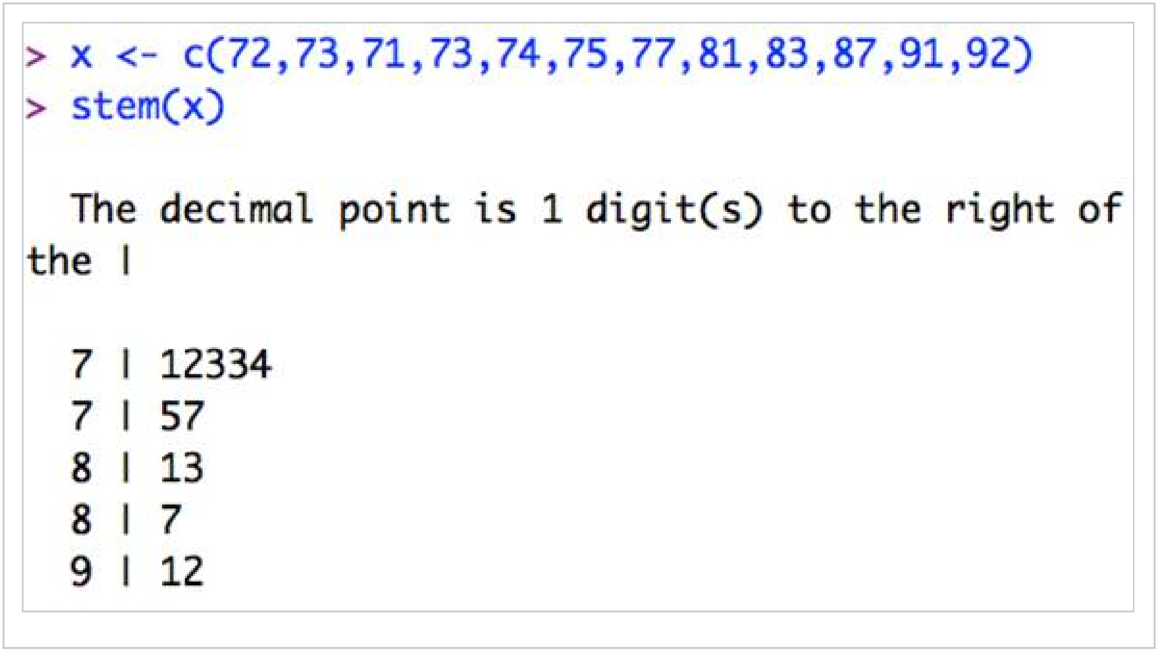
Test Yourself 3.8

Produce a barplot showing the frequency of the approvals. Show the R commands.

```
x <- table(data$approval)
barplot(x, ylab='Frequency', xlab='approval', ylim=c(0, 20))
```

2.8. Stem Plots

The stem and leaf plot is useful for small datasets as it shows the values and the shape of the distribution compactly. The stem is the number on the left of the bar. The number on the right of the bar is the leaf.



```
> x <- c(72,73,71,73,74,75,77,81,83,87,91,92)
> stem(x)

  The decimal point is 1 digit(s) to the right of
the |

  7 | 12334
  7 | 57
  8 | 13
  8 | 7
  9 | 12
```

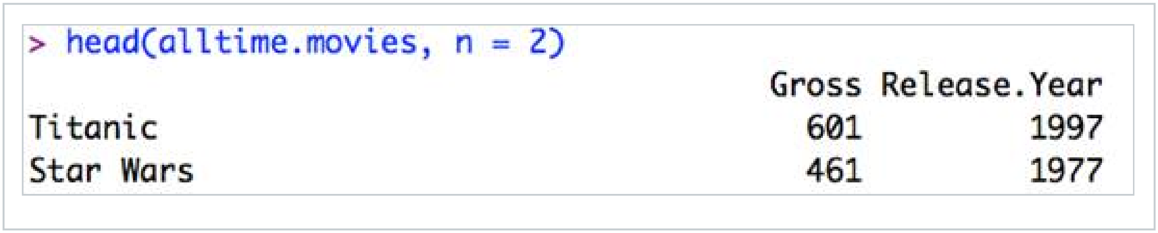
Test Yourself 3.9

Produce a stem plot of the approvals. The stem should be value in the tens place while the leaves represent the values in the ones place (e.g- 71 would be 7 I 1). Adjust the stem() function's scale and width arguments to get the desired plot. Show the R commands.

```
x <- data$approval
stem(x, scale=0.5)
```

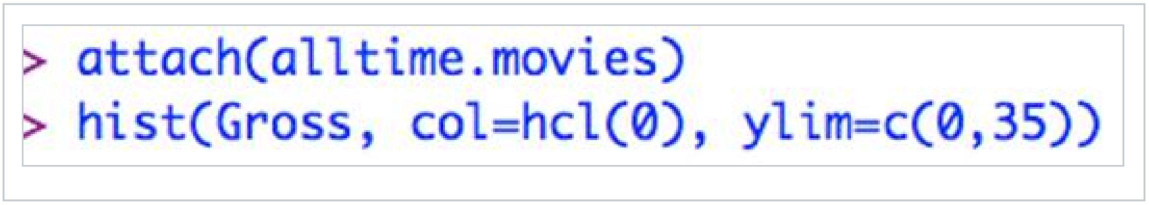
2.9. Histograms and Boxplots

The histogram shows the frequency of the data for a particular interval, whereas the barplot is used for a category. The `alltime.movies` dataset (from the `UsingR` package) shows the gross receipts (in millions) for the top 79 movies as of the year 2003. The data is already sorted in descending order.

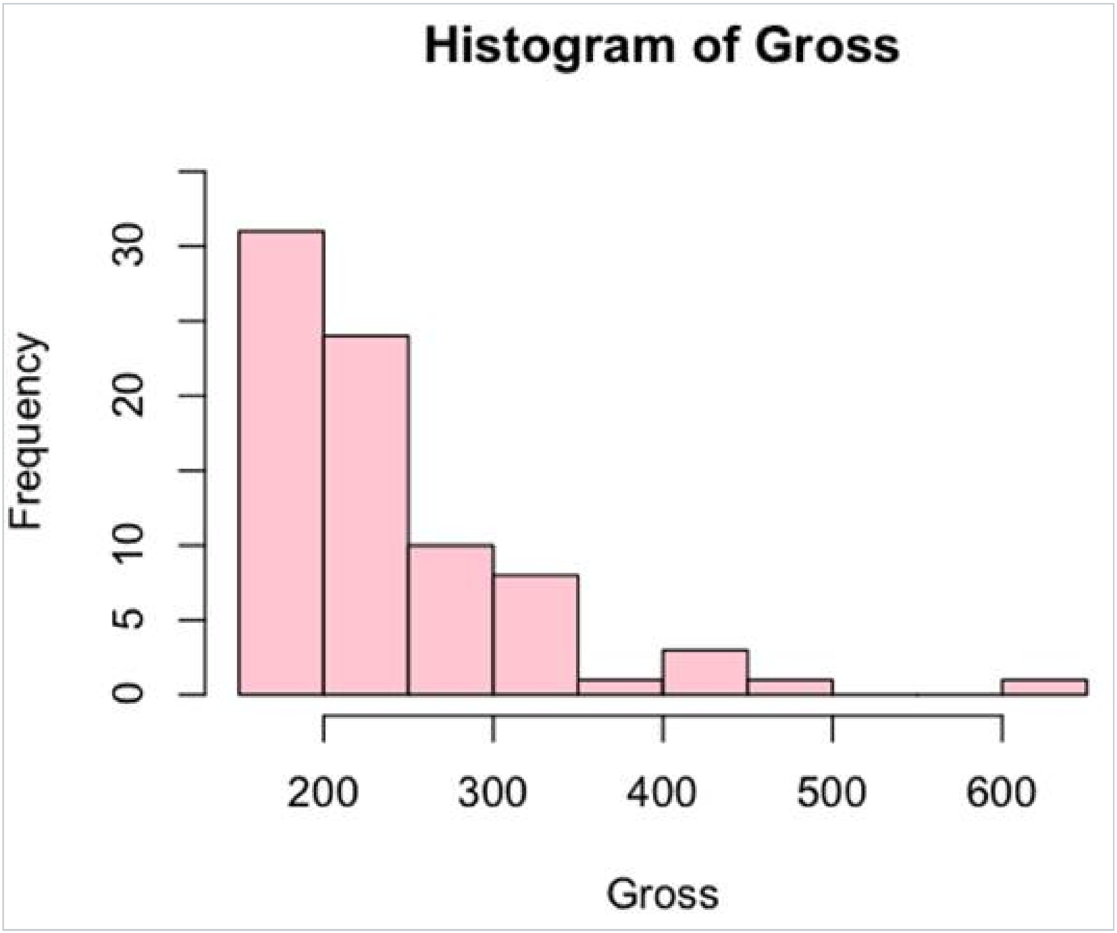


```
> head(alltime.movies, n = 2)
                                 Gross Release.Year
Titanic                            601         1997
Star Wars                          461         1977
```

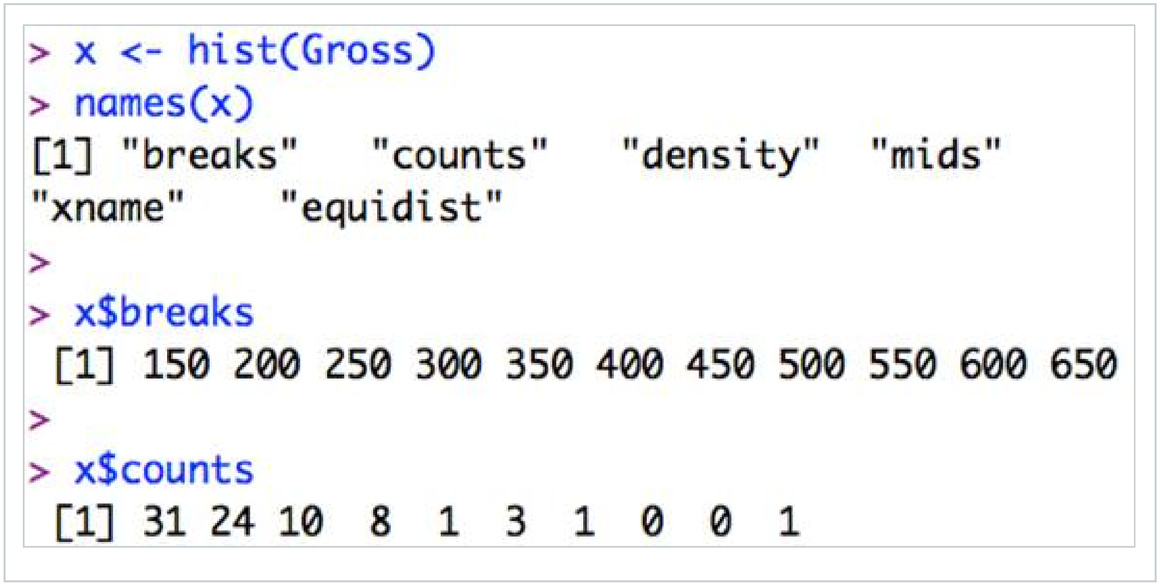
The histogram for the data is shown below.



```
> attach(alltime.movies)
> hist(Gross, col=hcl(0), ylim=c(0,35))
```

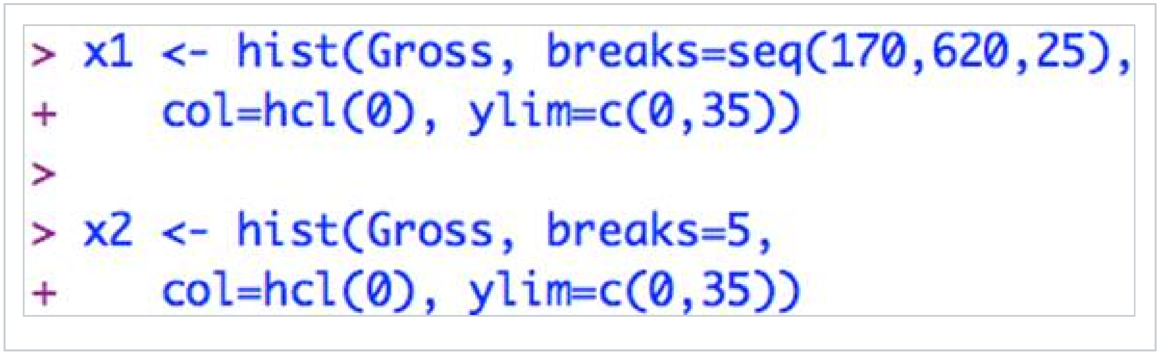


The above plot shows that 31 movies had gross receipts of less than 200 million dollars and one movie greater than 600 million dollars. The intervals (also known as bins or buckets) for the histogram bars can be examined by assigning to a variable as shown below. The `breaks` variable of the histogram shows the ranges for each interval. The `counts` variable shows the number of values that lie within each of these intervals. Note that the length of `counts` is one less than the length of `breaks`.



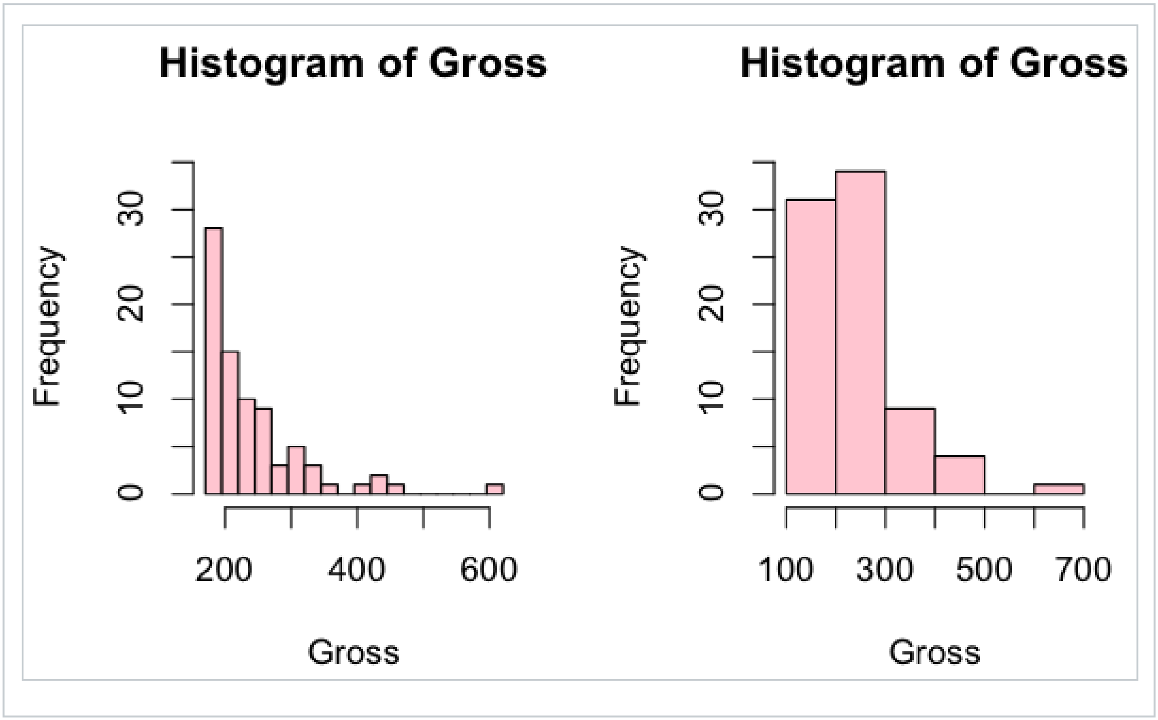
```
> x <- hist(Gross)
> names(x)
[1] "breaks"   "counts"   "density"  "mids"
"xname"    "equidist"
>
> x$breaks
 [1] 150 200 250 300 350 400 450 500 550 600 650
>
> x$counts
 [1] 31 24 10  8  1  3  1  0  0  1
```

The sequence of breaks can be explicitly specified for the histogram. Another option is to specify the number of breaks for the histogram. Both these options are shown below.

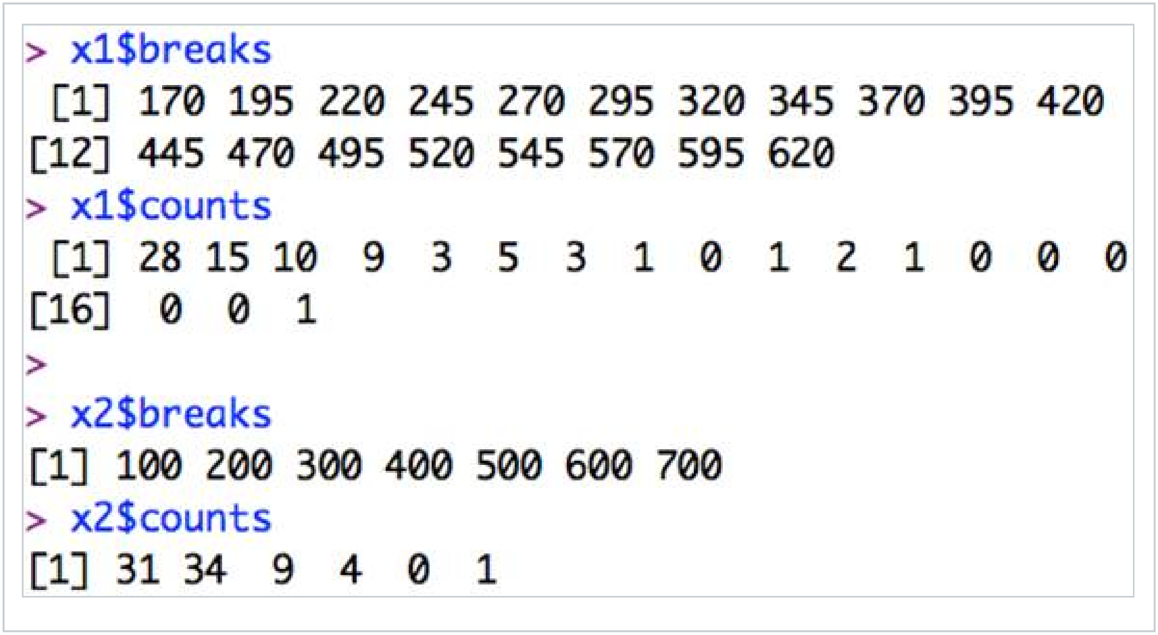


```
> x1 <- hist(Gross, breaks=seq(170,620,25),
+    col=hcl(0), ylim=c(0,35))
>
> x2 <- hist(Gross, breaks=5,
+    col=hcl(0), ylim=c(0,35))
```

The shape of the histogram differs depending upon how the intervals are established.

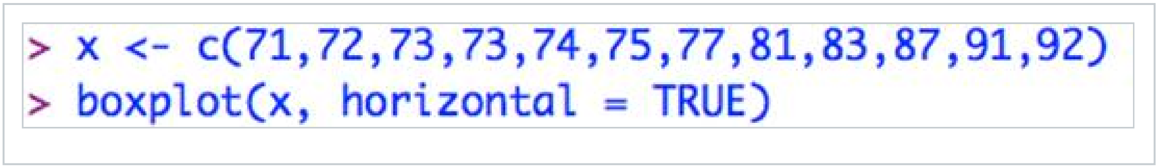


The breaks and the counts for the above two options are as follows.



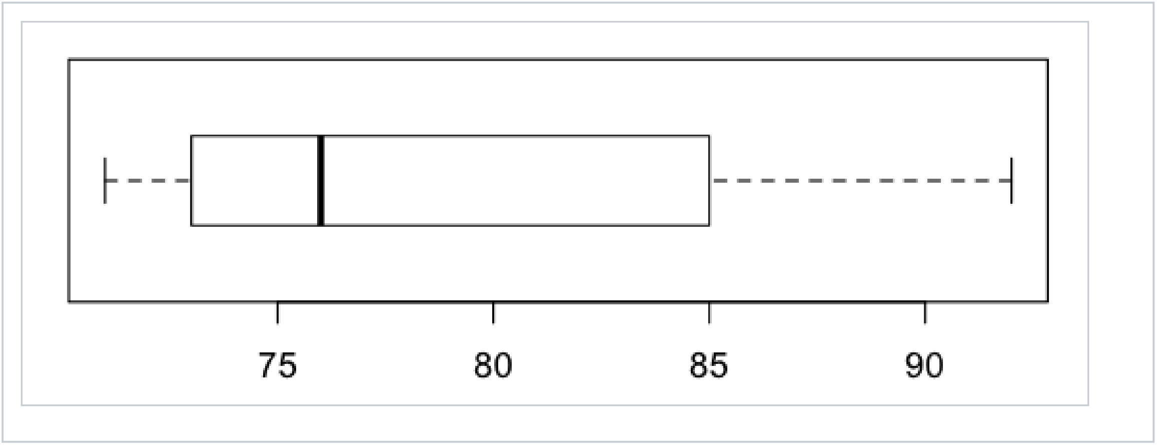
```
> x1$breaks
 [1] 170 195 220 245 270 295 320 345 370 395 420
[12] 445 470 495 520 545 570 595 620
> x1$counts
 [1] 28 15 10  9  3  5  3  1  0  1  2  1  0  0  0
[16]  0  0  1
>
> x2$breaks
[1] 100 200 300 400 500 600 700
> x2$counts
[1] 31 34  9  4  0  1
```

A boxplot (also known as box and whisker plot) is used to display the five-number summary of the data and allows the visual inspection of the center of the data, the spread of the data, and the skew of the data. The boxplot function, by default, shows the plot vertically. In the following example, the boxplot is drawn horizontally.

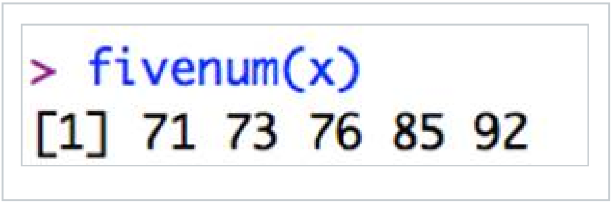


```
> x <- c(71,72,73,73,74,75,77,81,83,87,91,92)
> boxplot(x, horizontal = TRUE)
```

If there are no outliers in the data, the whiskers at the ends show the minimum and maximum values of the data. The left end of the box shows the lower hinge (or the first quartile) while the right end of the box shows the upper hinge (or the third quartile). The box is essentially the middle 50% of the data, the interquartile range. The thick line within the box shows the median of the data. The lower 50% of the data occurs to the left of the median, while the upper 50% of the data occurs to the right of the median. Similarly, the lower 25% of the data lies between the left whisker and the left edge of the box, while the upper 25% of the data lies between the right edge of the box and the right whisker.

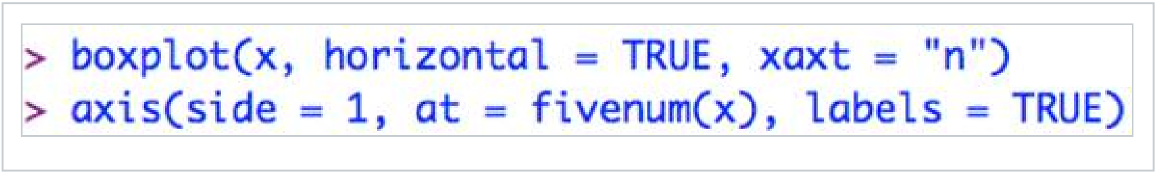


The five-number summary of the data is shown below.

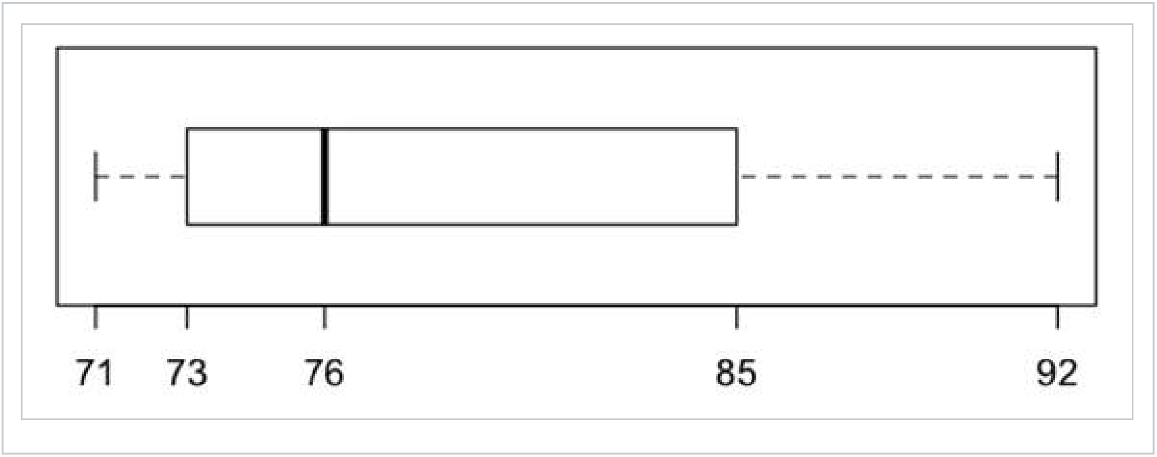


```
> fivenum(x)
[1] 71 73 76 85 92
```

The default labels for the x-axis can be suppressed and labeled with the five-number summary values as shown below.

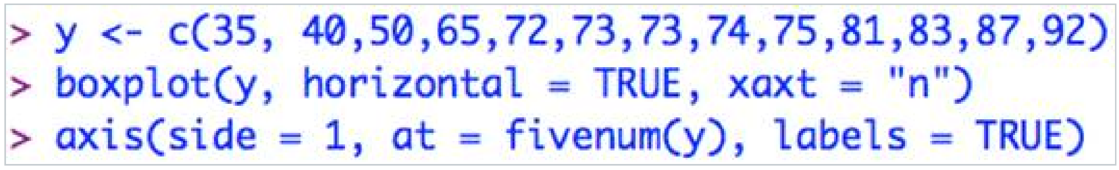


```
> boxplot(x, horizontal = TRUE, xaxt = "n")
> axis(side = 1, at = fivenum(x), labels = TRUE)
```

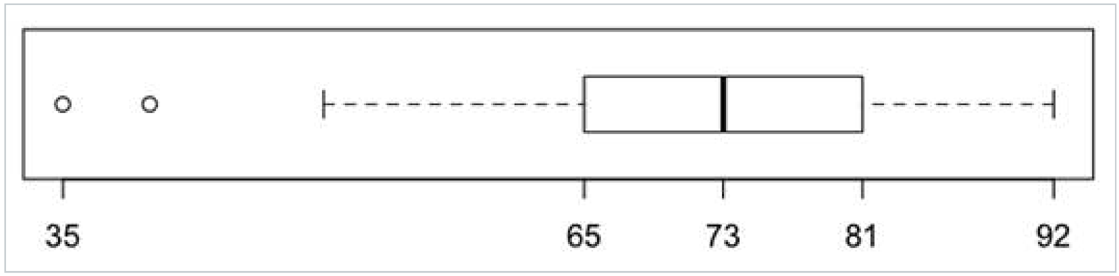


If there are outliers in the data, the boxplot is modified to show the outliers. Any data that is greater than $(3\text{rd quartile} + 1.5 \cdot \text{IQR})$ or less than $(1\text{st quartile} - 1.5 \cdot \text{IQR})$ is considered an outlier and is plotted using the circle dots as shown below. If there are outliers, the whiskers at those ends mark the data point that doesn't lie in the outlier range.

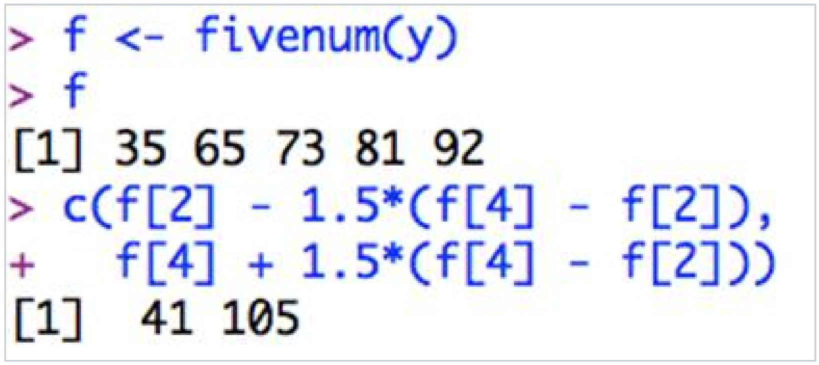
The following data has two outliers on the lower end of the data.



```
> y <- c(35, 40,50,65,72,73,73,74,75,81,83,87,92)
> boxplot(y, horizontal = TRUE, xaxt = "n")
> axis(side = 1, at = fivenum(y), labels = TRUE)
```



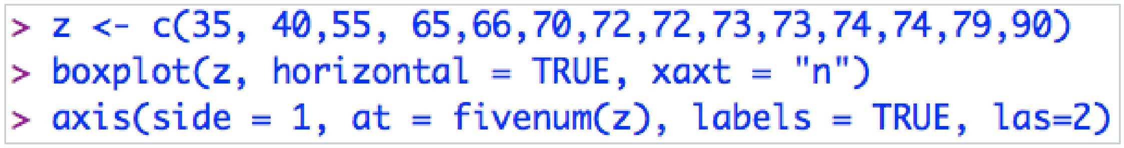
The lower and upper ends of the outlier ranges can be calculated as follows.



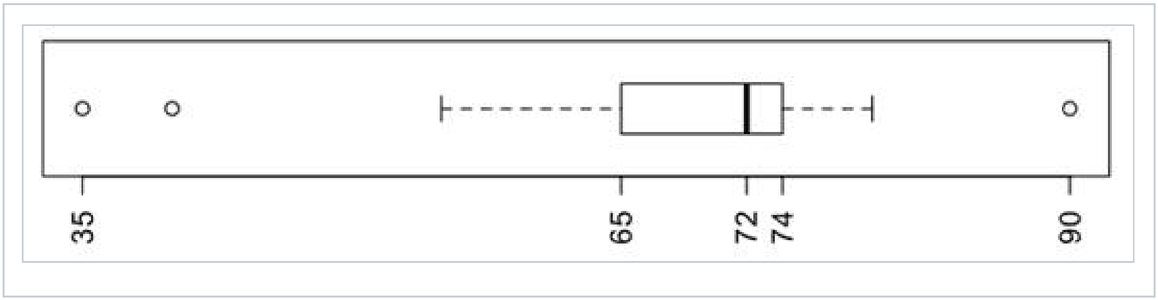
```
> f <- fivenum(y)
> f
[1] 35 65 73 81 92
> c(f[2] - 1.5*(f[4] - f[2]),
+    f[4] + 1.5*(f[4] - f[2]))
[1]  41 105
```

For the above dataset, any values that are less than 41 and greater than 105 are considered outliers. So, the values 35 and 40 are outliers. The whisker at the left is drawn at the next value 50.

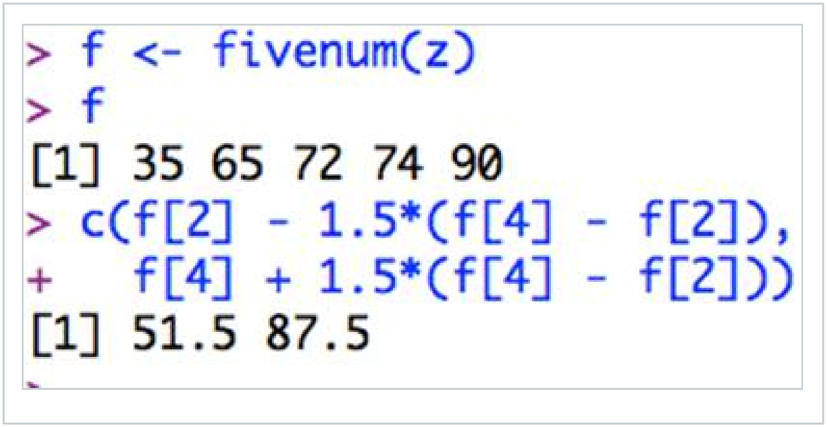
The following data has two outliers on the lower end and one outlier on the upper end of the data.



```
> z <- c(35, 40,55, 65,66,70,72,72,73,73,74,74,79,90)
> boxplot(z, horizontal = TRUE, xaxt = "n")
> axis(side = 1, at = fivenum(z), labels = TRUE, las=2)
```



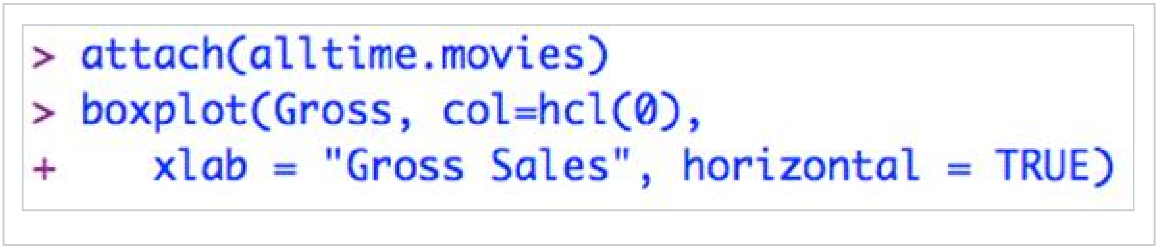
The lower and upper ends of the outlier ranges for the above data can be calculated as follows.



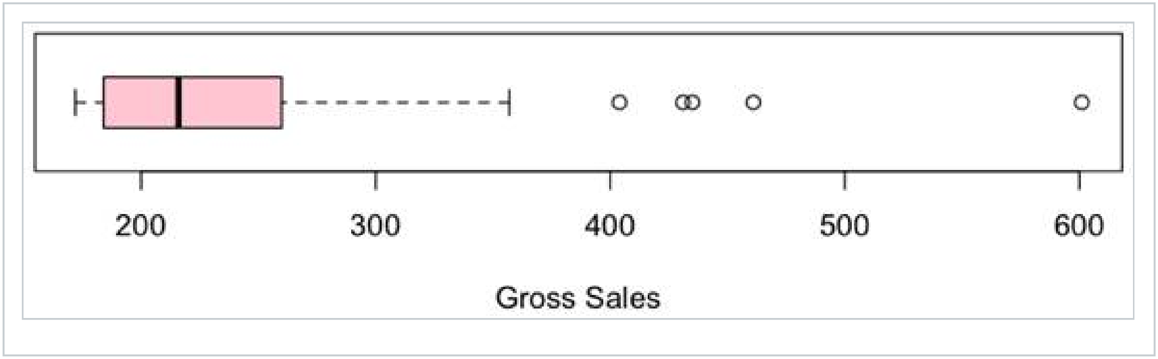
```
> f <- fivenum(z)
> f
[1] 35 65 72 74 90
> c(f[2] - 1.5*(f[4] - f[2]),
+   f[4] + 1.5*(f[4] - f[2]))
[1] 51.5 87.5
```

Hence, the values 35, 40, and 90 are marked as outliers. The whisker at the left is shown for the input value 55 whereas the whisker on the right corresponds to the input value 79.

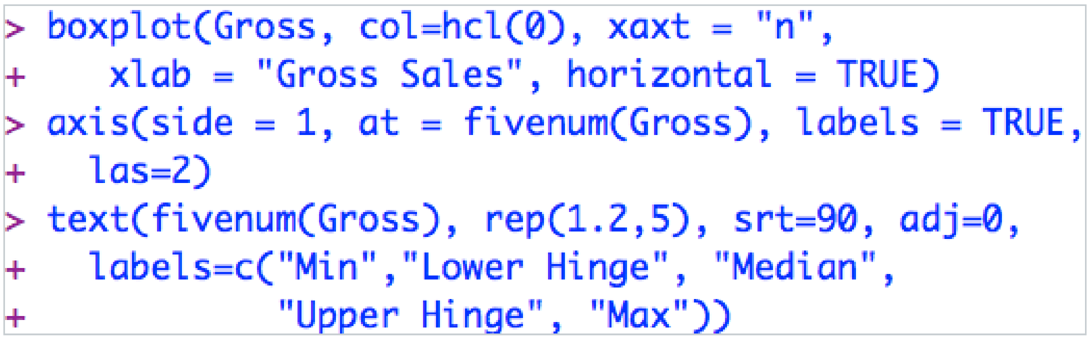
The boxplot for the gross receipts of the movies dataset is shown below.



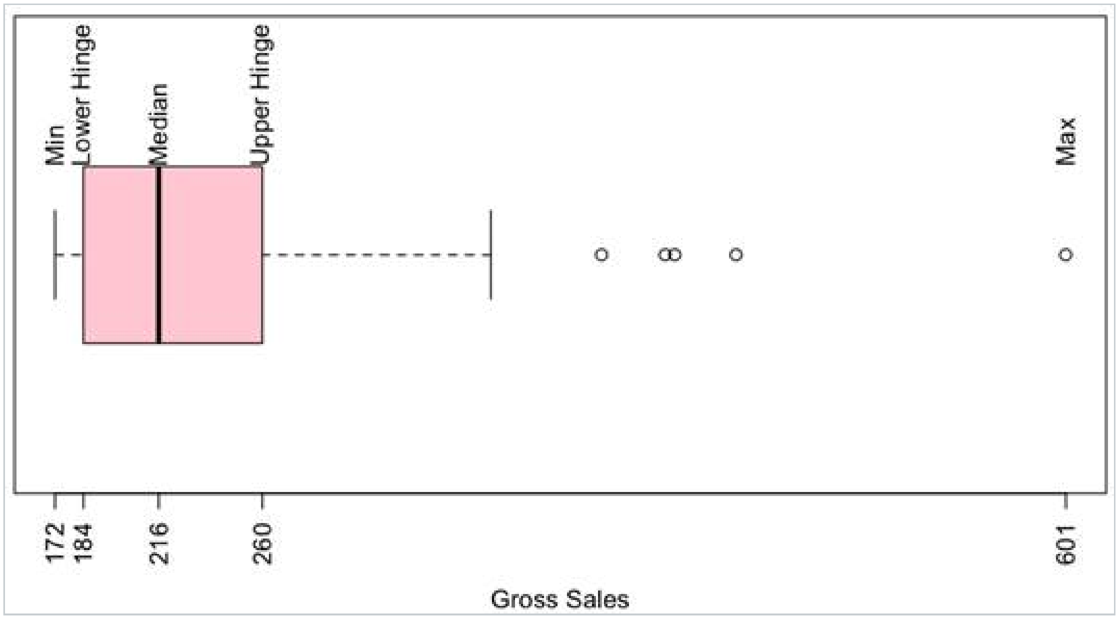
```
> attach(alltime.movies)
> boxplot(Gross, col=hcl(0),
+    xlab = "Gross Sales", horizontal = TRUE)
```



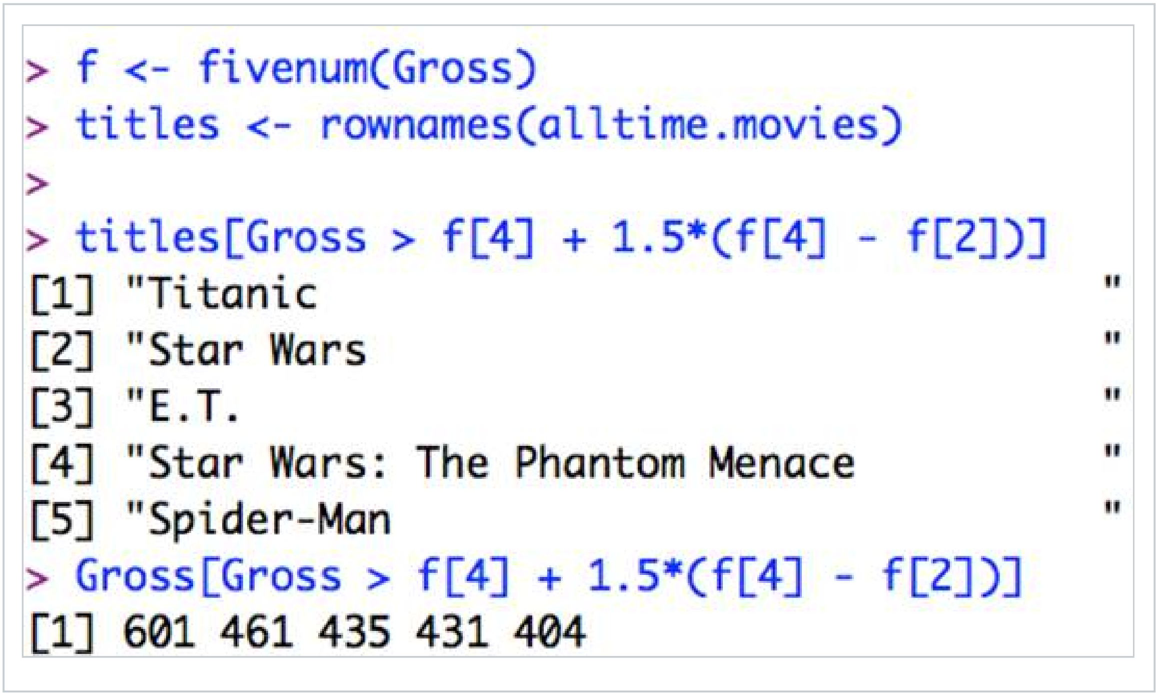
The five-number summary values along with the text labels can be added to the plot as shown below.



```
> boxplot(Gross, col=hcl(0), xaxt = "n",
+    xlab = "Gross Sales", horizontal = TRUE)
> axis(side = 1, at = fivenum(Gross), labels = TRUE,
+   las=2)
> text(fivenum(Gross), rep(1.2,5), srt=90, adj=0,
+   labels=c("Min","Lower Hinge", "Median",
+             "Upper Hinge", "Max"))
```



The boxplot shows that there are five outlier values at the upper end of the data. The titles of these movies and their gross receipts can be explicitly calculated as shown below.



```
> f <- fivenum(Gross)
> titles <- rownames(alltime.movies)
>
> titles[Gross > f[4] + 1.5*(f[4] - f[2])]
[1] "Titanic                                          "
[2] "Star Wars                                        "
[3] "E.T.                                             "
[4] "Star Wars: The Phantom Menace                    "
[5] "Spider-Man                                       "
> Gross[Gross > f[4] + 1.5*(f[4] - f[2])]
[1] 601 461 435 431 404
```

Test Yourself 3.10

Produce a boxplot of the approvals. Label the plot at the five number summary positions. Show the R commands.

```
x <- data$approvals
boxplot(x, yaxt='n')
axis(side=2, at=fivenum(x), labels=TRUE)
```

Test Yourself 3.11

Produce a histogram of the approvals. Make sure the range you select is appropriate and encompasses all approval values. Try 3 different bucket sizes. Show the R commands.

```
x <- data$approvals
hist(x, breaks=5)
hist(x, breaks=10)
hist(x, breaks=20)
```

Test Yourself 3.12

Add two values, 8 and 99 to the vector of approvals. Produce a boxplot. Are either of the new data points outliers? Show the R commands.

```
x <- data$approval
x <- c(x, 8, 99)
boxplot(x)
```

Yes, the two new data points are both outliers

3. Bivariate Data

3.1. Bivariate Data

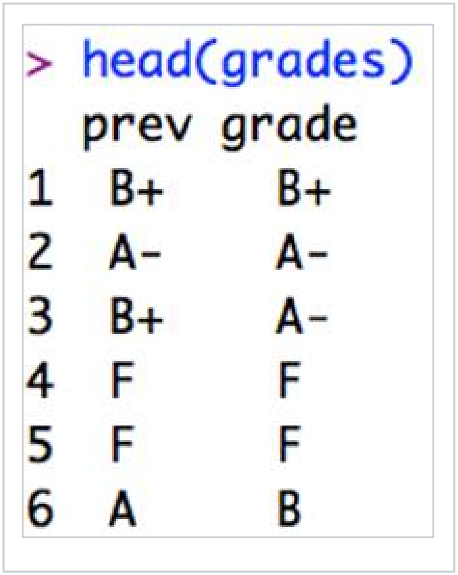
Bivariate datasets deal with data contained in two variables. Depending on the given data, the two variables may be related or unrelated. If the given data has the test scores of the students from two different sections taking the same test, the pairs of data are unrelated. If the same section of students takes two different tests, the data is related. In addition to summarizing data as categorical or numeric, the relationship between the data is also explored.

Contingency (two-way) tables are frequently used to summarize bivariate categorical data. For each possible pair of values for the two variables, the contingency table provides the number of occurrences for that pair. Relationships between the two variables can be examined by comparing the rows/columns of the contingency table.

3.2. Two-way Tables—Unsummarized Data

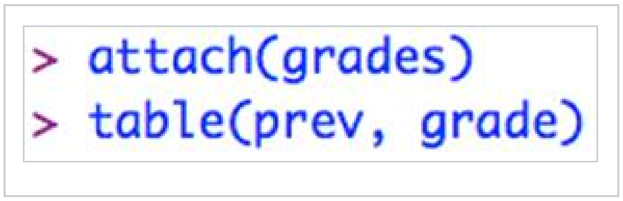
If the data for the two categorical variables is given in the unsummarized form (the actual values), the contingency table is created using the *R* function `table(x,y)`, where `x` and `y` are the two data vectors representing the two variables.

The `grades` dataset (from the `UsingR` package) is used in this example. The dataset provides the grades of 122 students for the current class (`grade`*)* and their previous class *(*`prev`*)*. The first six rows of the dataset are shown below.

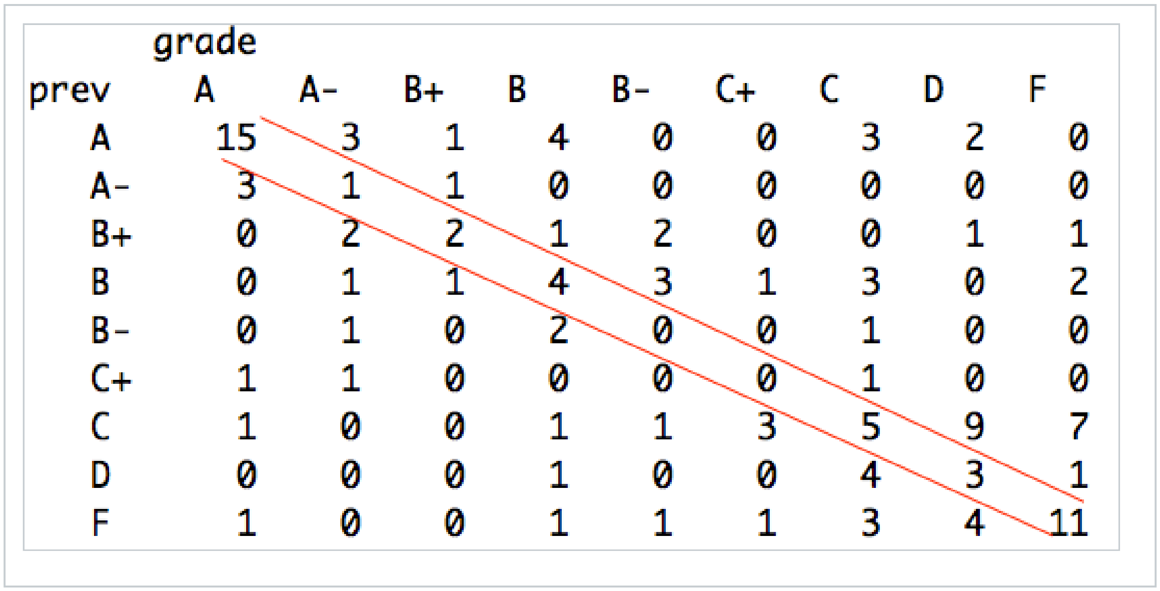


```
> head(grades)
  prev grade
1   B+    B+
2   A-    A-
3   B+    A-
4    F     F
5    F     F
6    A     B
```

The contingency table between the `prev` and `grade` variables is computed using the `table` function. The function can be called directly on the dataset, `table(grades)`, or explicitly by specifying the two variables as shown below.



```
> attach(grades)
> table(prev, grade)
```

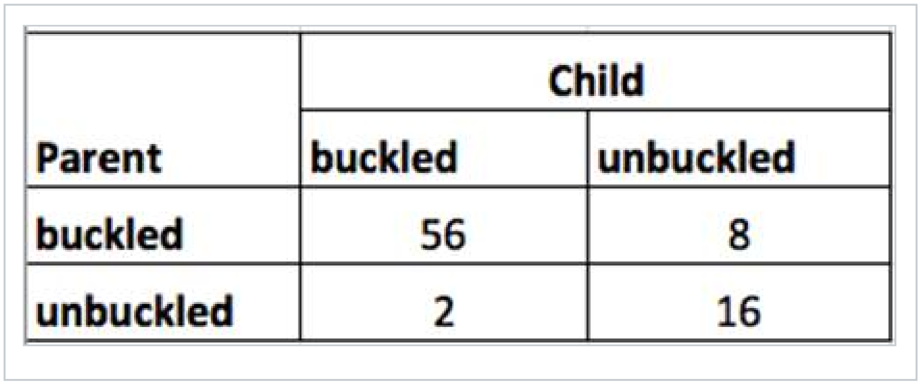


```
      grade
prev    A   A-  B+  B   B-  C+  C   D   F
   A    15  3   1   4   0   0   3   2   0
   A-   3   1   1   0   0   0   0   0   0
   B+   0   2   2   1   2   0   0   1   1
   B    0   1   1   4   3   1   3   0   2
   B-   0   1   0   2   0   0   1   0   0
   C+   1   1   0   0   0   0   1   0   0
   C    1   0   0   1   1   3   5   9   7
   D    0   0   0   1   0   0   4   3   1
   F    1   0   0   1   1   1   3   4  11
```

The above table shows that the current grades relate to the previous grades. Fifteen students who got *A* in the previous class also got *A* in the current class. Similarly, eleven students who got *F* in the previous class also got *F* in the current class. The lower left triangle shows the students who improved when the previous grade is compared to the current grade. The upper right triangle shows the number of students whose performance declined between the previous class and the current one.

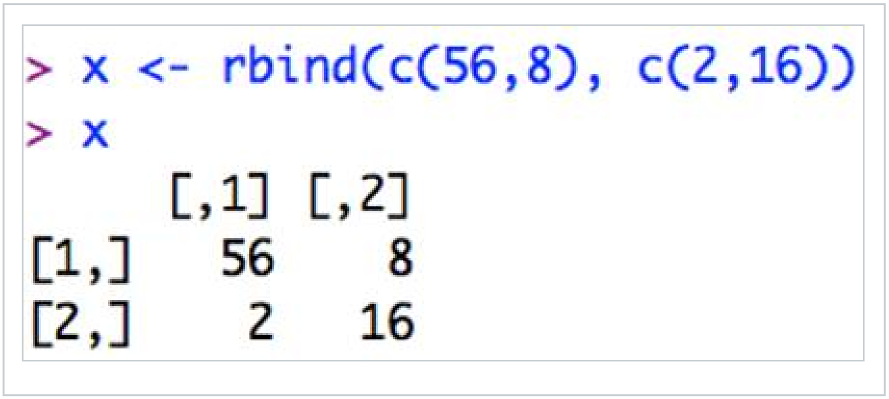
3.3. Two-way Tables—Summarized Data

If the raw data comparing two variables is available in summarized form, the data can be entered into *R* in various ways. The following table shows the results of a survey of 82 vehicles looking at seat belt use by parents and their children.



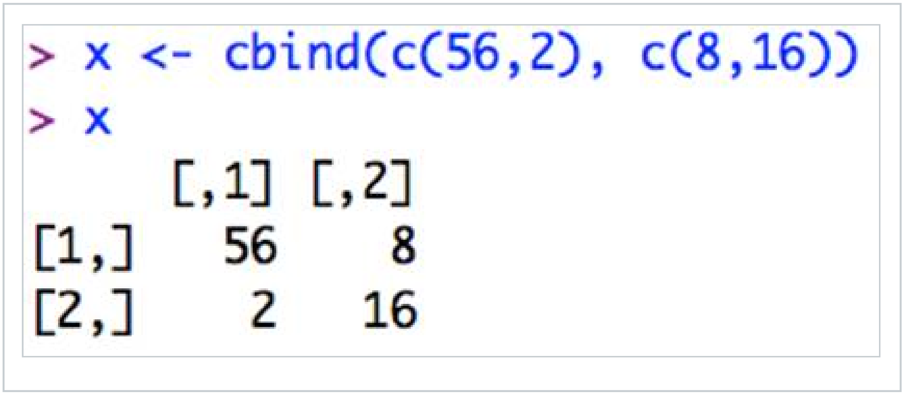
	Child	
Parent	**buckled**	**unbuckled**
buckled	56	8
unbuckled	2	16

The data can be entered using the `rbind` function, where each argument is a vector representing the corresponding row of the table.



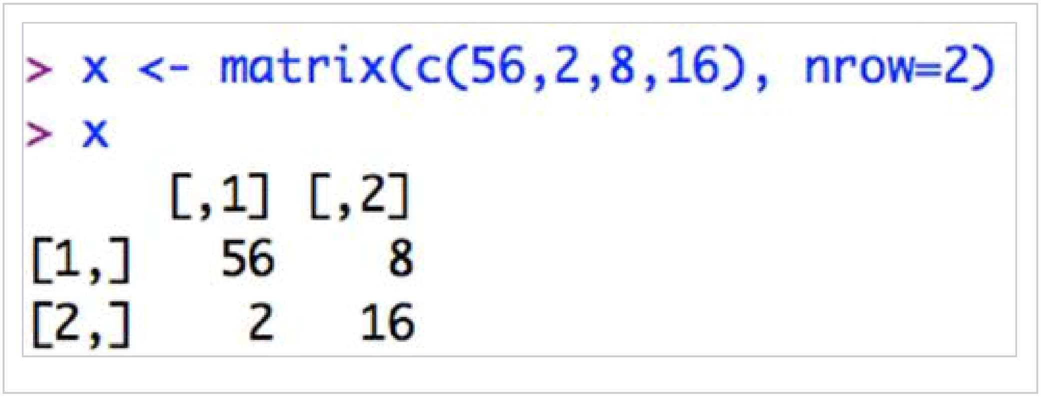
```
> x <- rbind(c(56,8), c(2,16))
> x
     [,1] [,2]
[1,]   56    8
[2,]    2   16
```

The data can also be entered using the `cbind` function, where each argument is a vector representing the corresponding column of the table.



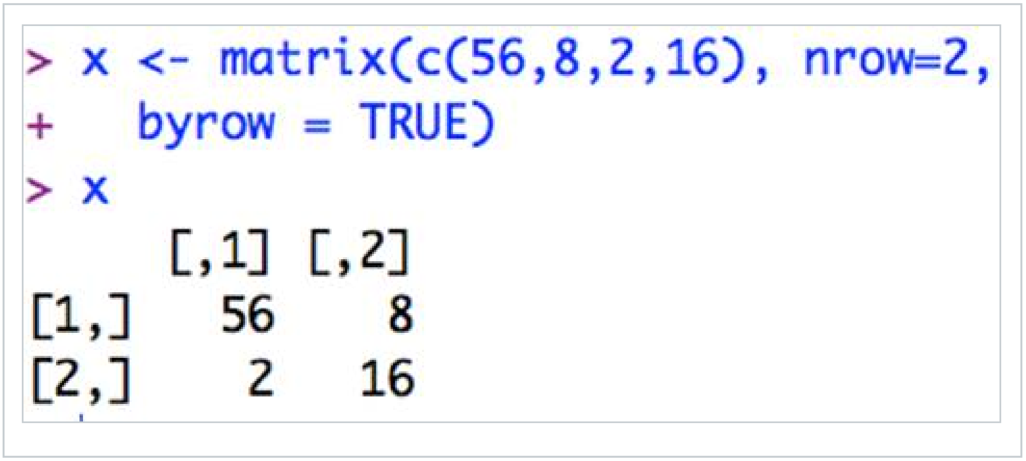
```
> x <- cbind(c(56,2), c(8,16))
> x
     [,1] [,2]
[1,]   56    8
[2,]    2   16
```

The matrix function can also be used by providing all the values as a single vector in the column-major order (default) and specifying either the number of rows or number of columns.



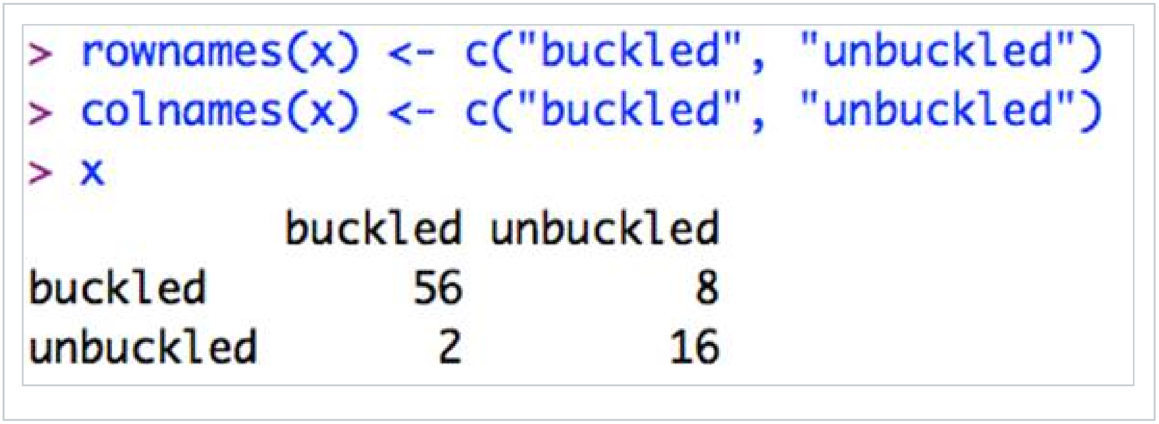
```
> x <- matrix(c(56,2,8,16), nrow=2)
> x
     [,1] [,2]
[1,]   56    8
[2,]    2   16
```

If the values are desired in the row-major order, the `byrow` option is specified as `TRUE`.

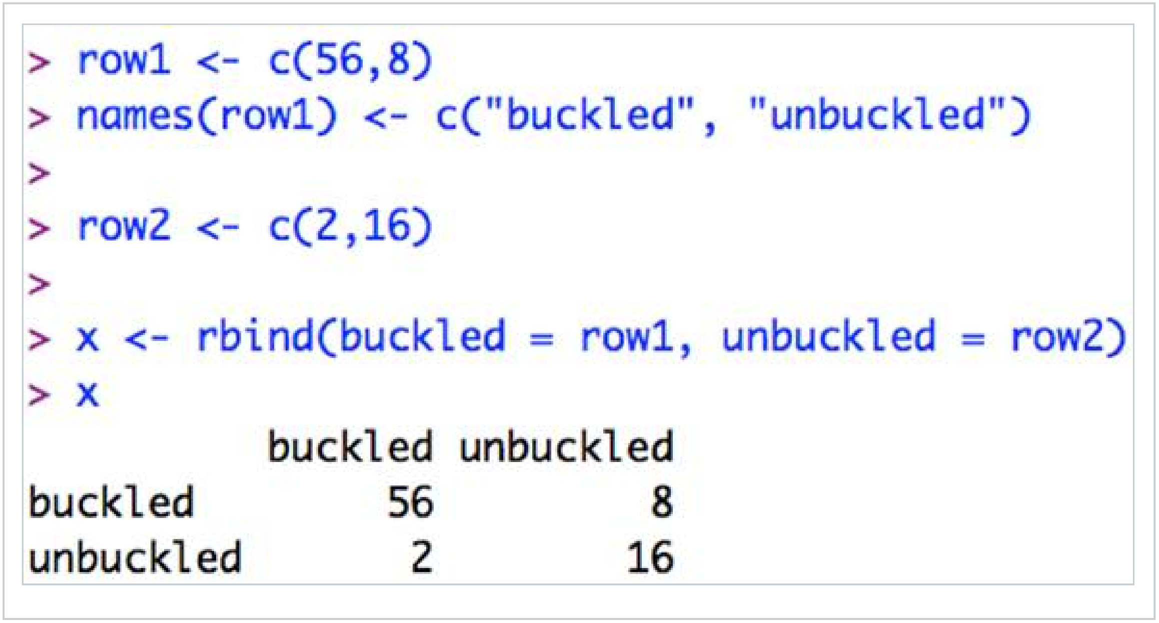


```
> x <- matrix(c(56,8,2,16), nrow=2,
+    byrow = TRUE)
> x
     [,1] [,2]
[1,]   56    8
[2,]    2   16
```

Once the table is created, names may optionally be assigned to the rows and columns of the table as follows.

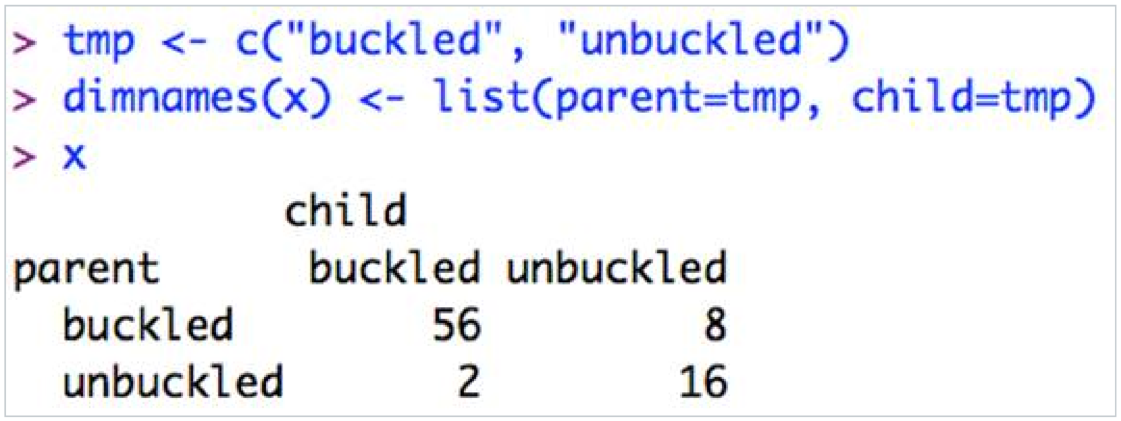


```
> rownames(x) <- c("buckled", "unbuckled")
> colnames(x) <- c("buckled", "unbuckled")
> x
          buckled unbuckled
buckled        56         8
unbuckled       2        16
```



```
> row1 <- c(56,8)
> names(row1) <- c("buckled", "unbuckled")
>
> row2 <- c(2,16)
>
> x <- rbind(buckled = row1, unbuckled = row2)
> x
          buckled unbuckled
buckled        56         8
unbuckled       2        16
```

The names for the variables can be specified using `dimnames` as follows.



```
> tmp <- c("buckled", "unbuckled")
> dimnames(x) <- list(parent=tmp, child=tmp)
> x
           child
parent      buckled unbuckled
  buckled        56         8
  unbuckled       2        16
```

Test Yourself 3.13

Consider the following data set representing a clinical trial on 10 patients. Each row represents a patient and the first column indicates their disease level before treatment while the second column represents the disease level after treatment. It should be noted that the disease level is a categorical variables and can take on the following values: "mild", "moderate", "severe", "terminal"

Before Treatment	After Treatment
Mild	Mild
Moderate	Mild
Mild	Mild
Moderate	Mild
Severe	Severe
Severe	Moderate
Terminal	Severe
Terminal	Terminal
Severe	Mild

Moderate	Moderate

Use whatever R commands needed to produce a two-way table representing the unsummarized data in the table above.

▶ Show Hint

```
pre <- c('mild', 'moderate', 'mild', 'moderate', 'severe', 'severe', 'term
post <- c('mild' ,'mild', 'mild', 'mild', 'severe', 'moderate', 'severe',
table(pre, post)
```

Values along the diagonal indicate the number of patients who say no changes in their disease status after treatment. Numbers in the upper diagonal represent patients whose status worsened, while those in the lower diagonal represent patients whose status improved.

Test Yourself 3.14

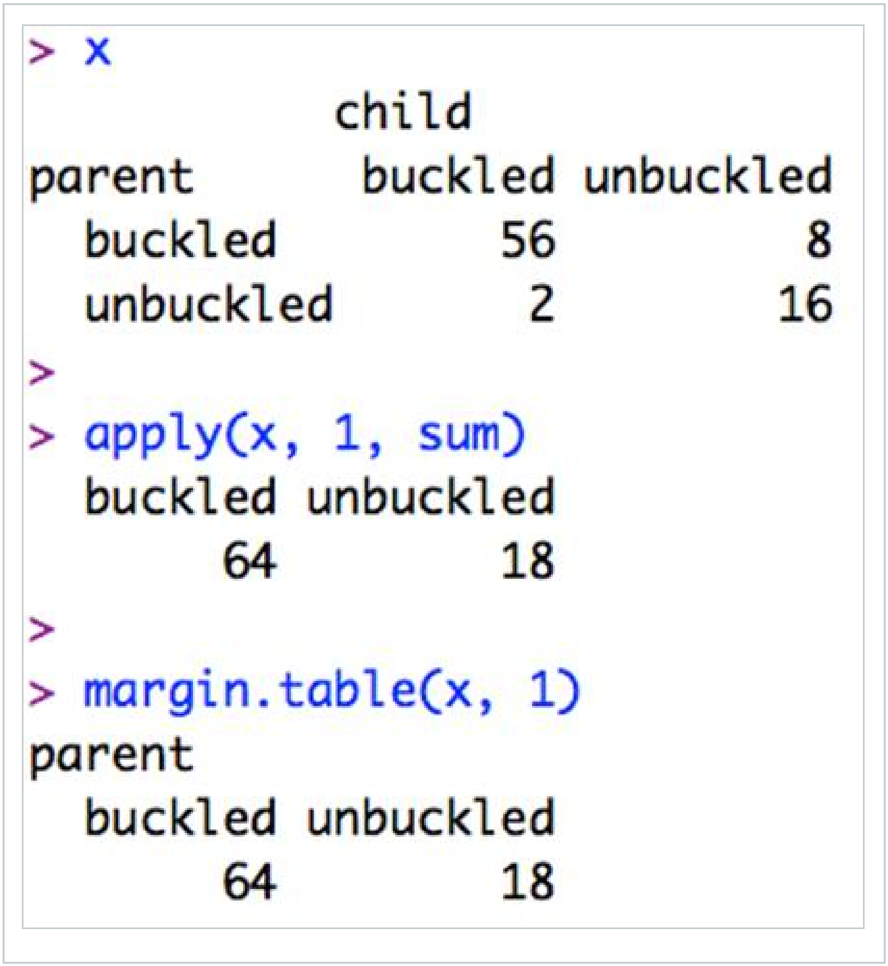
Reproduce the summarized table from the previous problem using the matrix() command. Name the rows and columns accordingly. Show the R commands.

```
data <- matrix(c(2, 0, 0, 0, 2, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1), nrow=4,
levels <- c('mild',  'moderate',  'severe',  'terminal')
dimnames(data) <- list(before.treatment=levels, after.treatment=levels)
```

3.4. Marginal Distributions of Two-way Tables

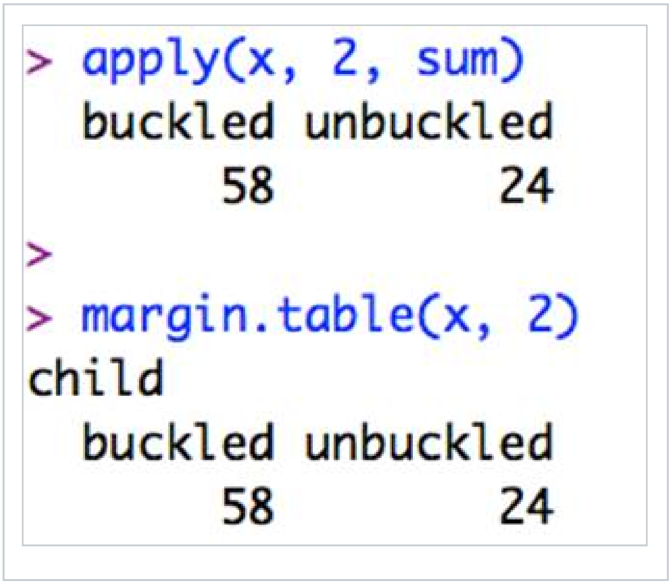
The distribution of each variable separately is the `marginal` distribution of that variable. In a two-way table, adding the rows or the columns gives the marginal distribution of the corresponding variable. The `apply` sum or the `margin.table` function can be used for this purpose.

For the seat belt dataset, the marginal distribution for the `parent` (rows in the table) is calculated as follows. The argument value 1 identifies the rows while the value 2 identifies the columns.



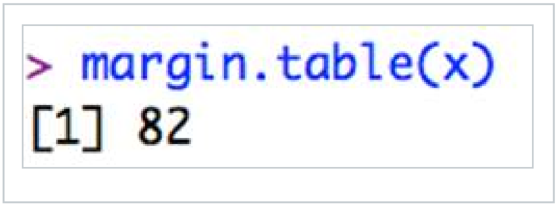
```
> x
           child
parent      buckled unbuckled
  buckled        56         8
  unbuckled       2        16
>
> apply(x, 1, sum)
  buckled unbuckled
       64        18
>
> margin.table(x, 1)
parent
  buckled unbuckled
       64        18
```

Similarly, the marginal distribution for the `child` variable (columns in the table) is computed as follows.



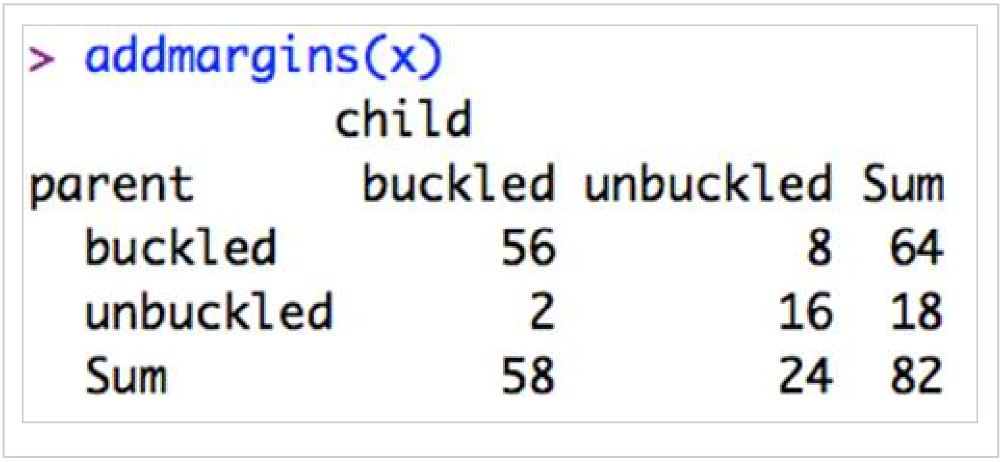
```
> apply(x, 2, sum)
  buckled unbuckled
       58        24
>
> margin.table(x, 2)
child
  buckled unbuckled
       58        24
```

If neither the row nor the column is specified, the function returns the sum of all the values in the table.



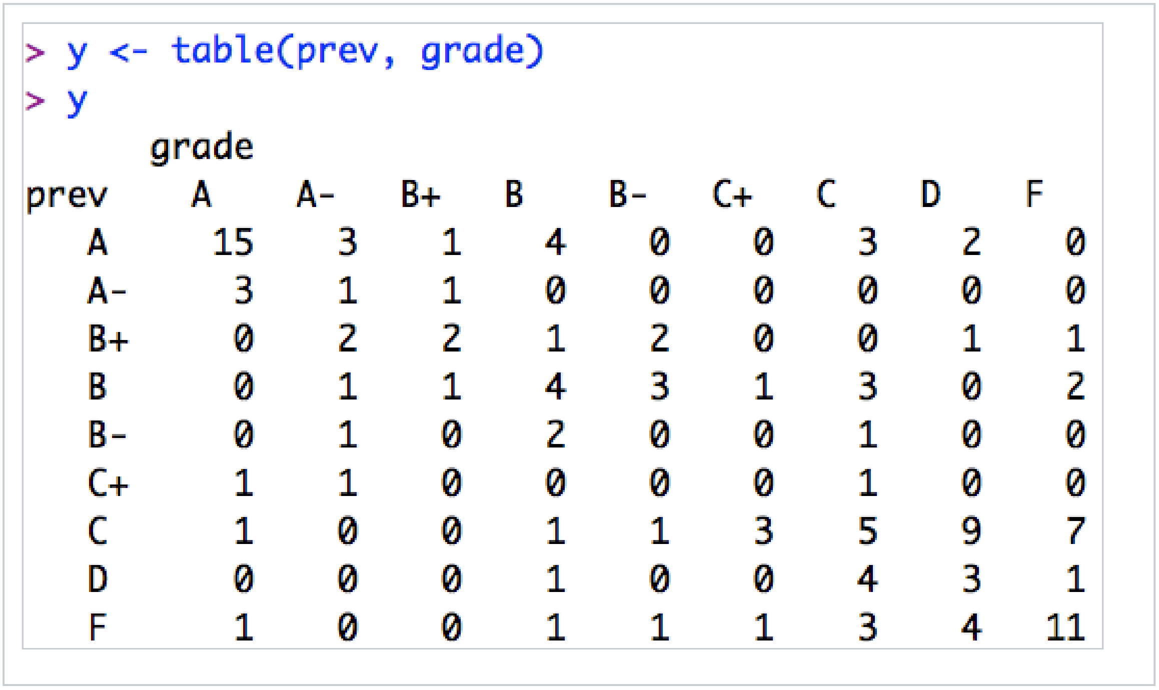
```
> margin.table(x)
[1] 82
```

The `addmargins` function extends the two-way table by adding the marginal distributions as shown below.

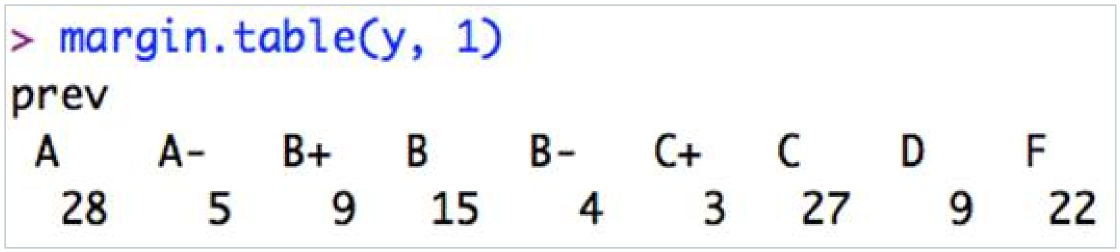


```
> addmargins(x)
           child
parent      buckled unbuckled Sum
  buckled        56         8  64
  unbuckled       2        16  18
  Sum            58        24  82
```

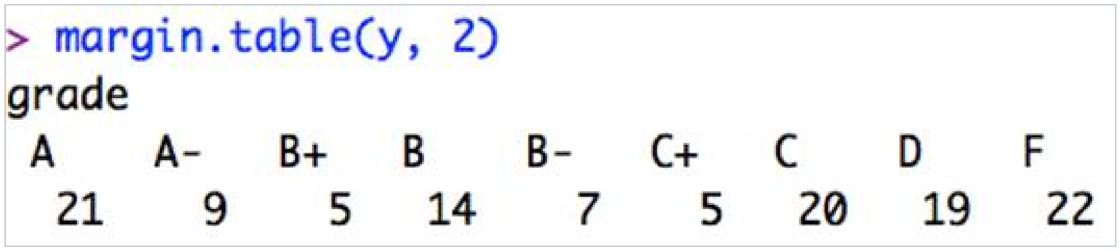
Similarly, for the grades dataset shown below, the marginal distributions of the previous grade (`prev`*)* and current grade (`grade`*)* variables can be computed.



```
> y <- table(prev, grade)
> y
      grade
prev    A  A- B+  B  B- C+  C  D  F
  A    15  3  1  4  0  0  3  2  0
  A-    3  1  1  0  0  0  0  0  0
  B+    0  2  2  1  2  0  0  1  1
  B     0  1  1  4  3  1  3  0  2
  B-    0  1  0  2  0  0  1  0  0
  C+    1  1  0  0  0  0  1  0  0
  C     1  0  0  1  1  3  5  9  7
  D     0  0  0  1  0  0  4  3  1
  F     1  0  0  1  1  1  3  4 11
```

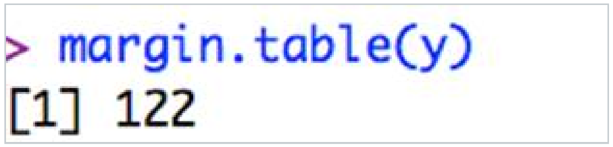


```
> margin.table(y, 1)
prev
 A     A-    B+    B     B-    C+    C     D     F 
  28     5     9    15     4     3    27     9    22 
```

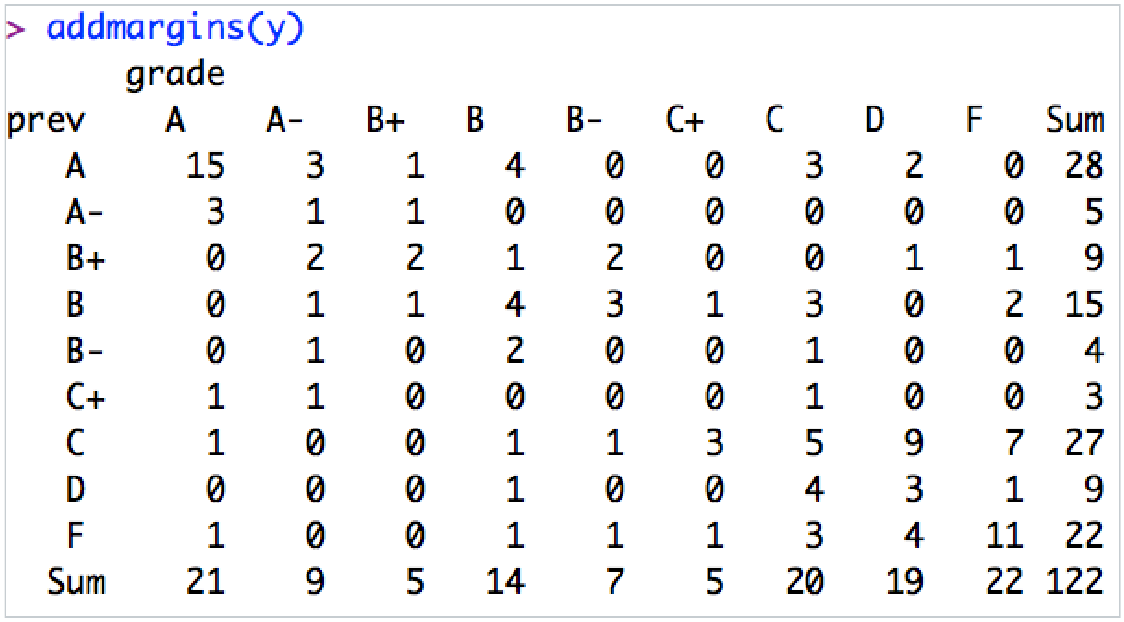


```
> margin.table(y, 2)
grade
 A     A-    B+    B     B-    C+    C     D     F 
  21     9     5    14     7     5    20    19    22 
```

From the above, 28 students had an *A* grade in the previous class while 21 students got an *A* grade in the current class. The number of *F* grades remained the same.



```
> margin.table(y)
[1] 122
```



```
> addmargins(y)
      grade
prev    A   A-  B+  B   B-  C+  C   D   F   Sum
   A    15   3   1   4   0   0   3   2   0  28
   A-    3   1   1   0   0   0   0   0   0   5
   B+    0   2   2   1   2   0   0   1   1   9
   B     0   1   1   4   3   1   3   0   2  15
   B-    0   1   0   2   0   0   1   0   0   4
   C+    1   1   0   0   0   0   1   0   0   3
   C     1   0   0   1   1   3   5   9   7  27
   D     0   0   0   1   0   0   4   3   1   9
   F     1   0   0   1   1   1   3   4  11  22
  Sum   21   9   5  14   7   5  20  19  22 122
```

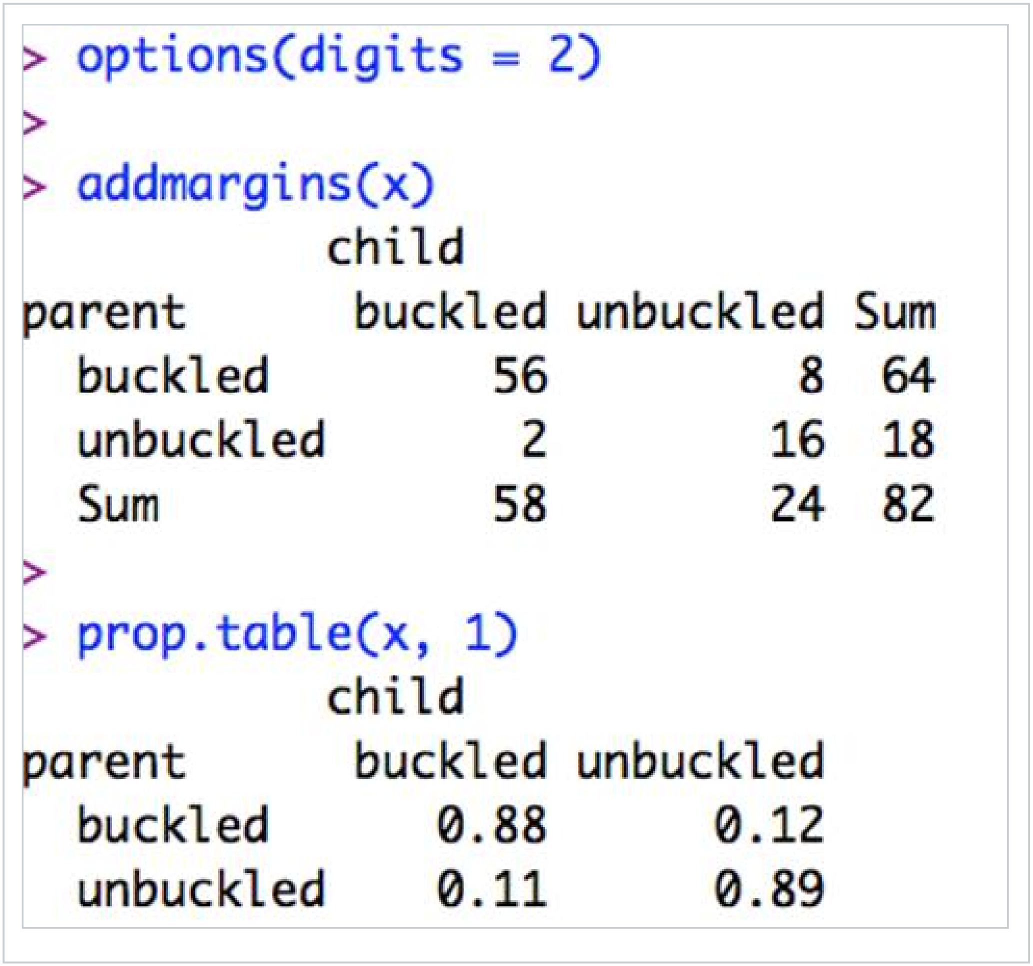
Test Yourself 3.15

Using the table generated in the previous problem, produce a marginal table by using the apply() and sum() functions. Repeat the task using the margin.table() function. Now add the margins to the original table. Show the R commands.

```
apply(data, 1, sum)
apply(data, 2, sum)
margin.table(data, 1)
margin.table(data, 2)
data.with.margins <- addmargins(data)
```

3.5. Conditional Distributions of Two-way Tables

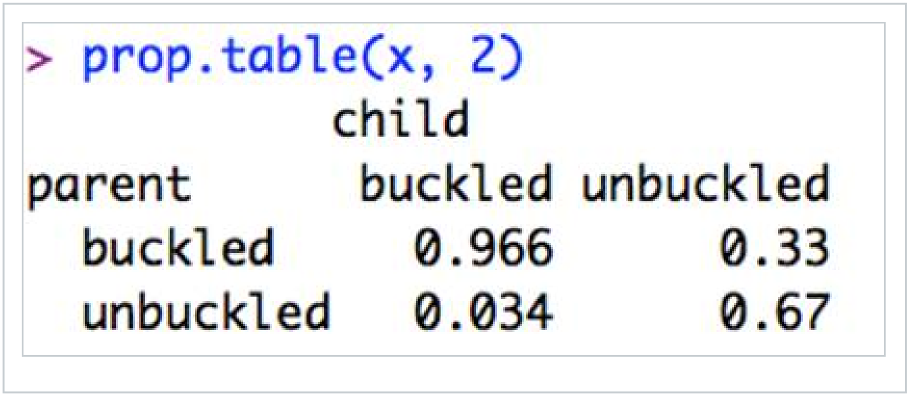
When comparing the rows (or columns) of the two-way table, having the values as proportions or percentages is more effective. The `prop.table` function computes the percentages with respect to the sum of the corresponding row (or column). With the argument value 1, the values of the rows add to 1 (100%). With the argument value 2, the values of the columns add to 1 (100%).



```
> options(digits = 2)
>
> addmargins(x)
           child
parent      buckled unbuckled Sum
  buckled        56         8  64
  unbuckled       2        16  18
  Sum            58        24  82
>
> prop.table(x, 1)
           child
parent      buckled unbuckled
  buckled      0.88      0.12
  unbuckled    0.11      0.89
```

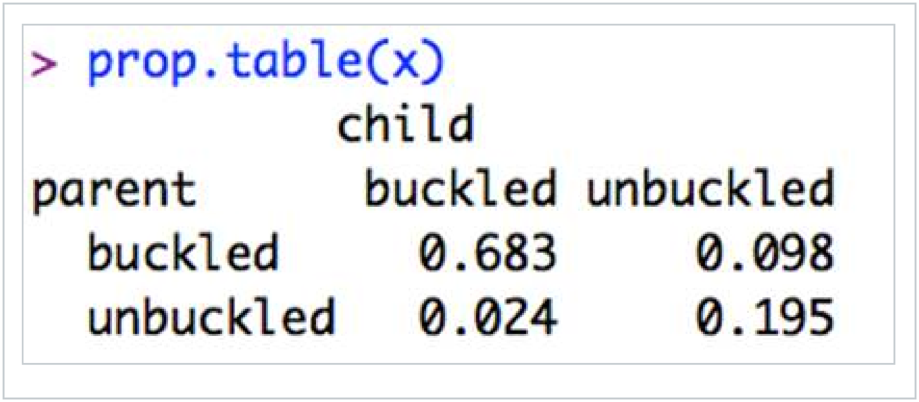
From the above table, 88% of the time children wear seat belts when their parents do. If the parent is not wearing a seat belt, 11% of the time the child wore the seat belt, while 89% of the time the child did not.

Similarly, the proportions along the columns are shown below.



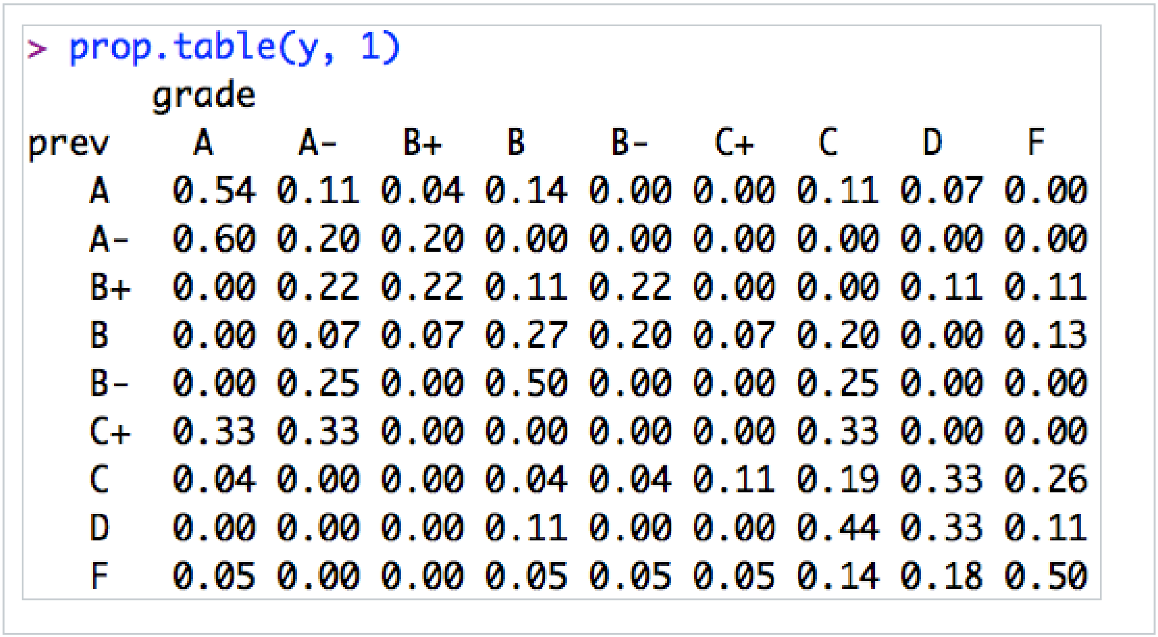
```
> prop.table(x, 2)
           child
parent      buckled unbuckled
  buckled     0.966      0.33
  unbuckled   0.034      0.67
```

If the row or column is not specified, the `prop.table` function computes the relative percentages with respect to the entire table.



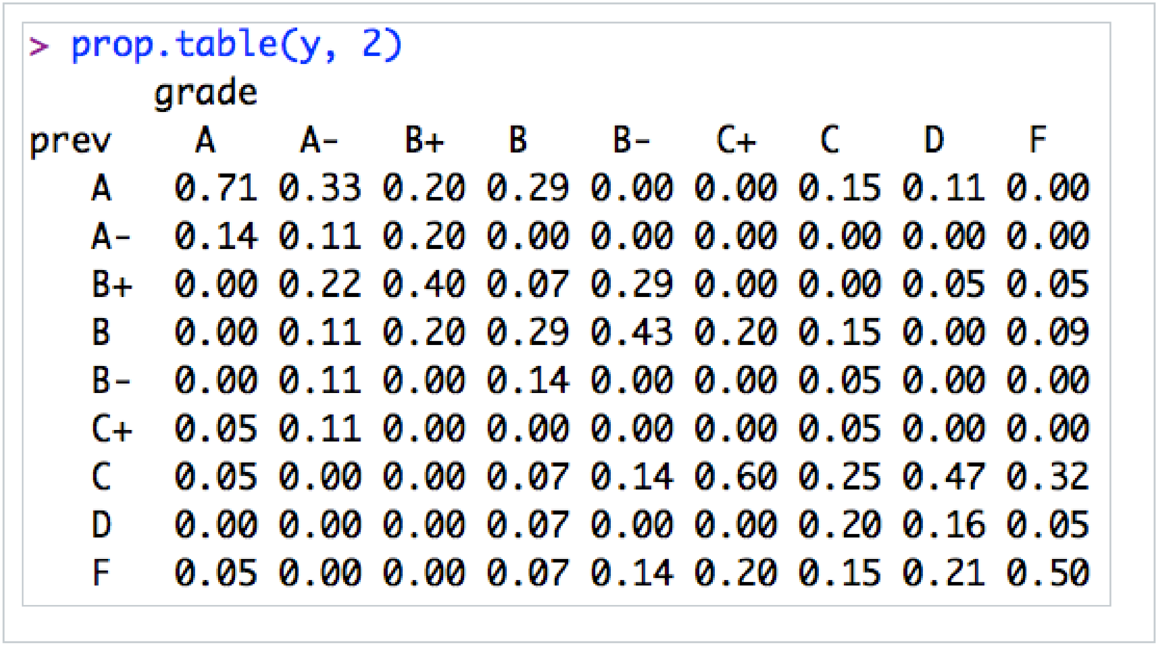
```
> prop.table(x)
           child
parent      buckled unbuckled
  buckled     0.683     0.098
  unbuckled   0.024     0.195
```

The proportions for the grades dataset along the rows are shown below. For each row, the percentages show the effect of the previous grade on the current grade.



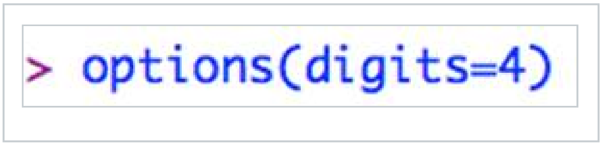
```
> prop.table(y, 1)
     grade
prev   A    A-   B+   B    B-   C+   C    D    F
  A    0.54 0.11 0.04 0.14 0.00 0.00 0.11 0.07 0.00
  A-   0.60 0.20 0.20 0.00 0.00 0.00 0.00 0.00 0.00
  B+   0.00 0.22 0.22 0.11 0.22 0.00 0.00 0.11 0.11
  B    0.00 0.07 0.07 0.27 0.20 0.07 0.20 0.00 0.13
  B-   0.00 0.25 0.00 0.50 0.00 0.00 0.25 0.00 0.00
  C+   0.33 0.33 0.00 0.00 0.00 0.00 0.33 0.00 0.00
  C    0.04 0.00 0.00 0.04 0.04 0.11 0.19 0.33 0.26
  D    0.00 0.00 0.00 0.11 0.00 0.00 0.44 0.33 0.11
  F    0.05 0.00 0.00 0.05 0.05 0.05 0.14 0.18 0.50
```

The proportions for the grades dataset along the columns are shown below. For each column, the percentages show the effect of the current grade on the previous grade.



```
> prop.table(y, 2)
     grade
prev     A    A-   B+   B    B-   C+   C    D    F
   A  0.71 0.33 0.20 0.29 0.00 0.00 0.15 0.11 0.00
   A- 0.14 0.11 0.20 0.00 0.00 0.00 0.00 0.00 0.00
   B+ 0.00 0.22 0.40 0.07 0.29 0.00 0.00 0.05 0.05
   B  0.00 0.11 0.20 0.29 0.43 0.20 0.15 0.00 0.09
   B- 0.00 0.11 0.00 0.14 0.00 0.00 0.05 0.00 0.00
   C+ 0.05 0.11 0.00 0.00 0.00 0.00 0.05 0.00 0.00
   C  0.05 0.00 0.00 0.07 0.14 0.60 0.25 0.47 0.32
   D  0.00 0.00 0.00 0.07 0.00 0.00 0.20 0.16 0.05
   F  0.05 0.00 0.00 0.07 0.14 0.20 0.15 0.21 0.50
```

The options function was used to set the number of digits in the decimal part for display purpose.



```
> options(digits=4)
```

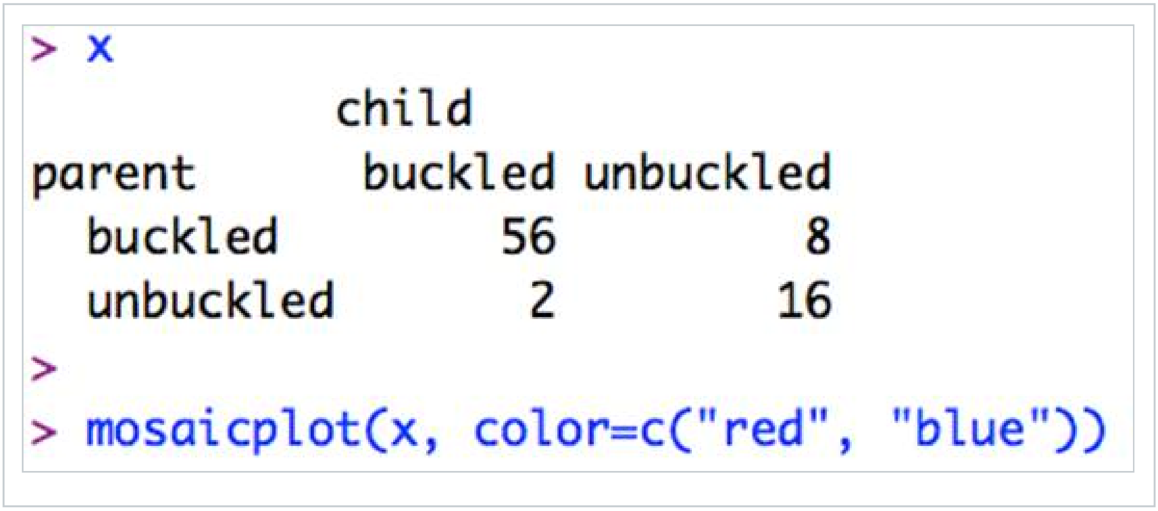
Test Yourself 3.16

Using the table generated in the previous problem, use prop.table() to show the table values as proportions of the marginal sums, as well as the total sum. Show the R commands.

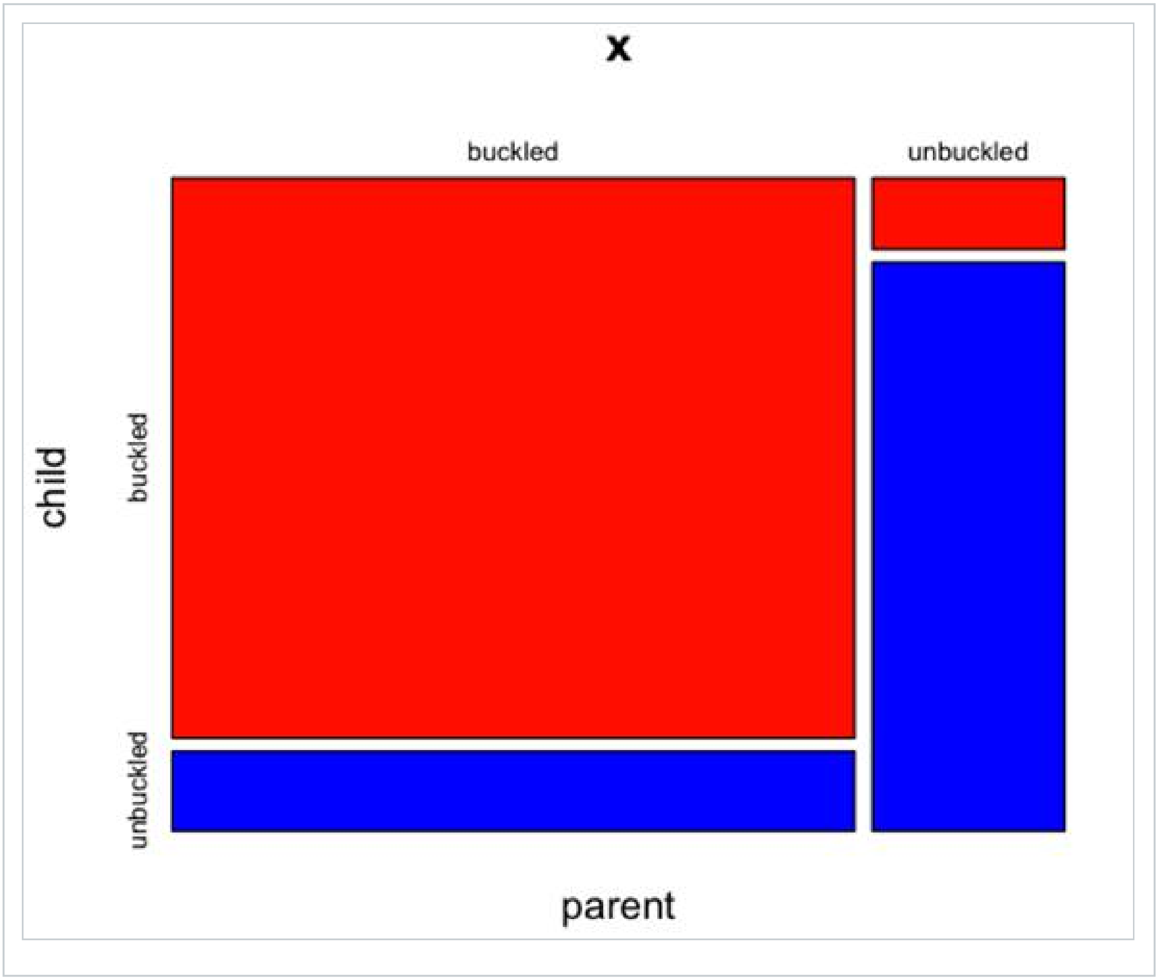
```
prop.table(data, 1)
prop.table(data ,2)
prop.table(data)
```

3.6. Graphical Summarization of Two-way Tables

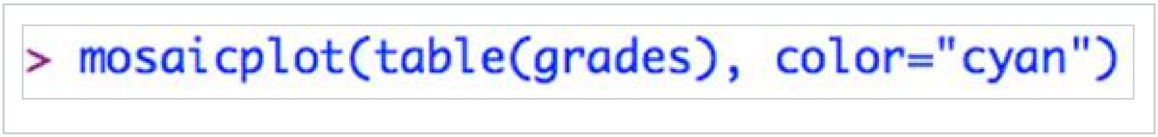
The mosaic plot is one option for creating a graphical representation of the contingency table. The counts in the table are represented by rectangles. The mosaic plot of the seat belt data can be plotted in *R* as shown below.



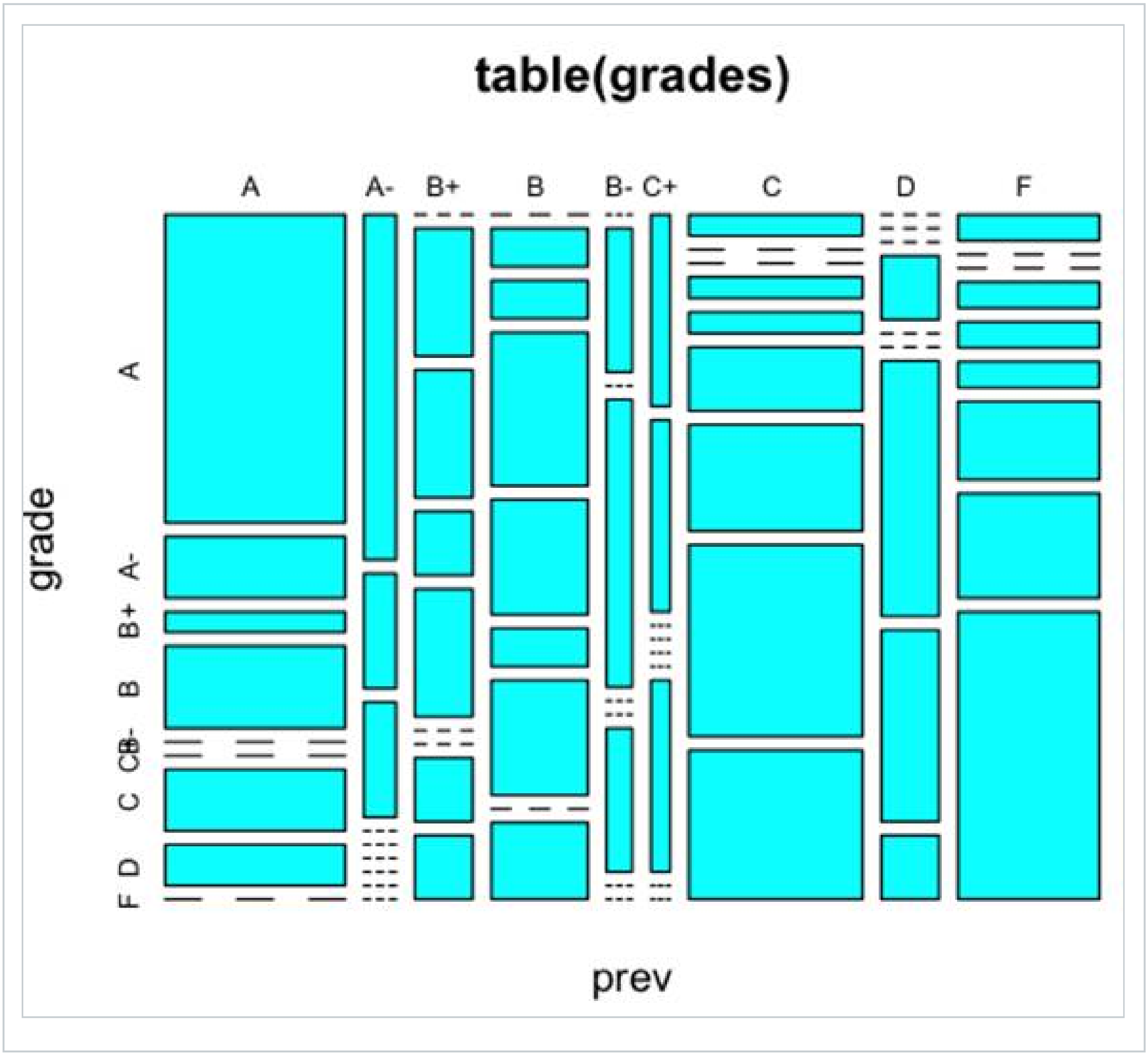
```
> x
           child
parent      buckled unbuckled
  buckled        56         8
  unbuckled       2        16
>
> mosaicplot(x, color=c("red", "blue"))
```



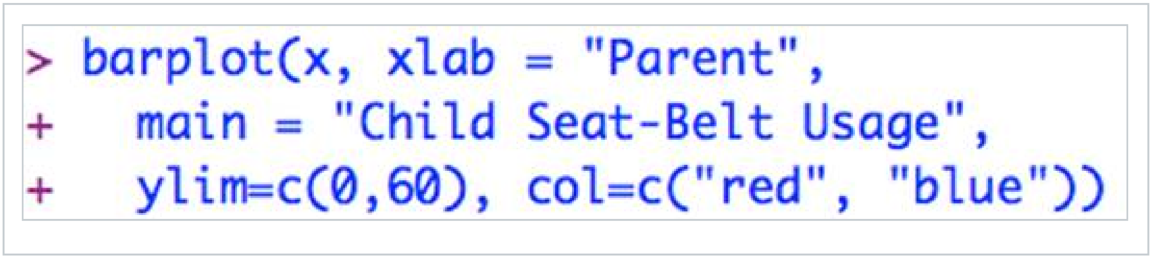
Similarly, the mosaic plot of the grade data is shown below.

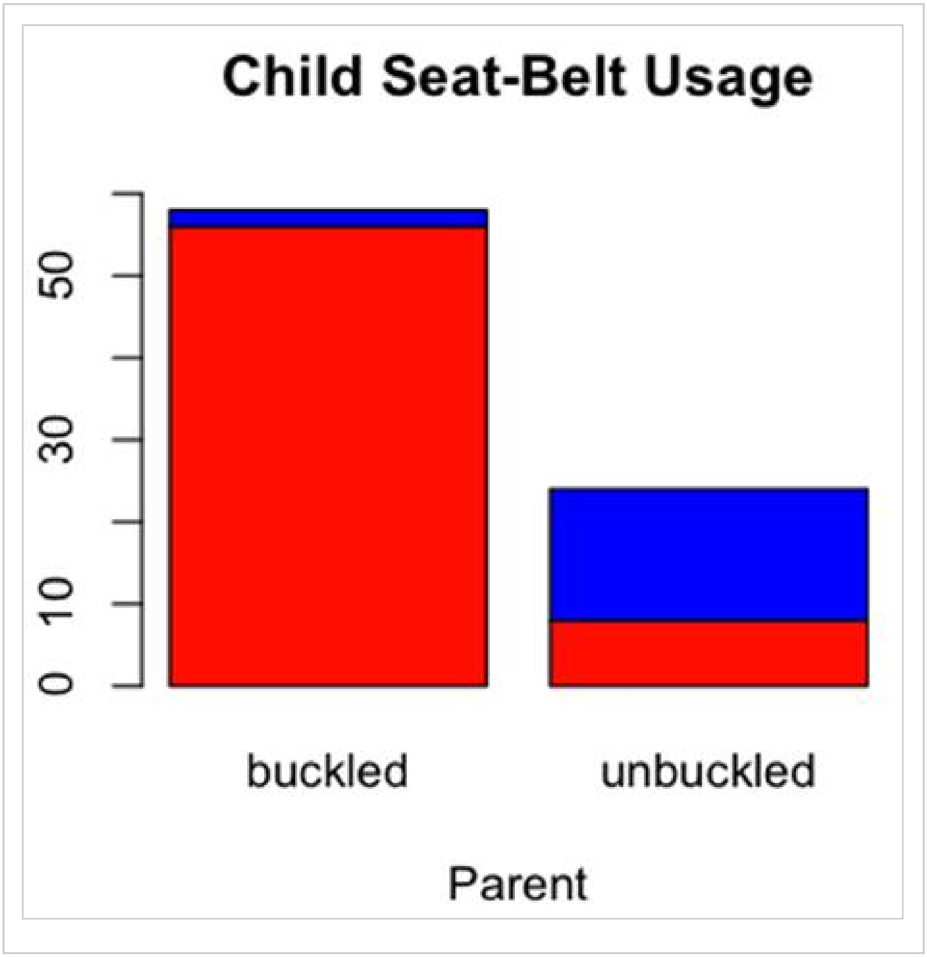


```
> mosaicplot(table(grades), color="cyan")
```

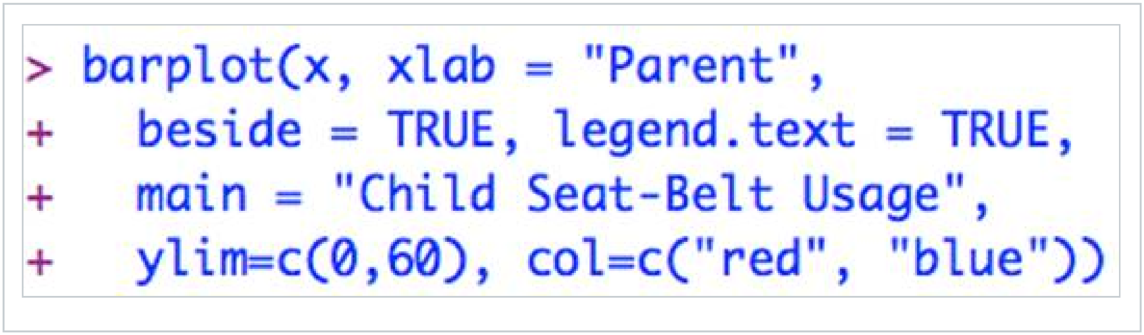


The bar plot can also be used for the graphical presentation of the two-way data. The first argument is the two-way table. The first variable of the table is used for the categories of the bar plot. The bars for each level are stacked by default and the heights show the proportions of the second variable. In the following example, the variable `x` stores the seat belt dataset.

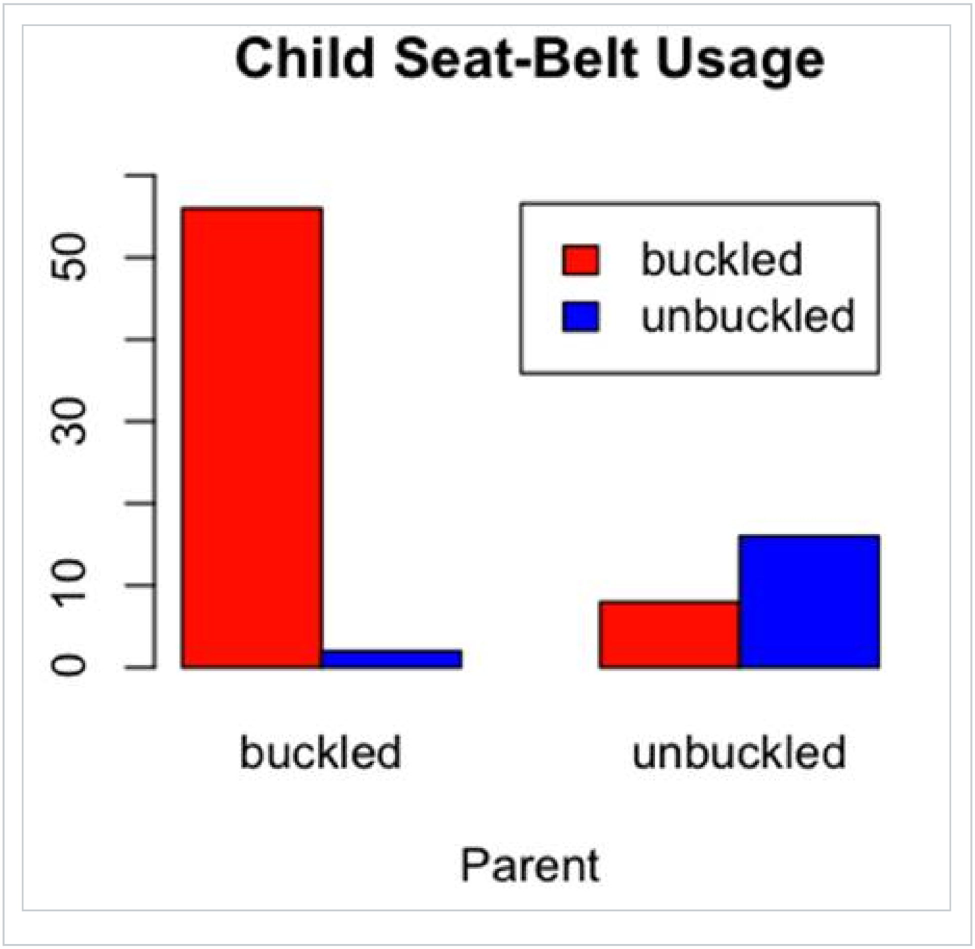

```
> barplot(x, xlab = "Parent",
+   main = "Child Seat-Belt Usage",
+   ylim=c(0,60), col=c("red", "blue"))
```



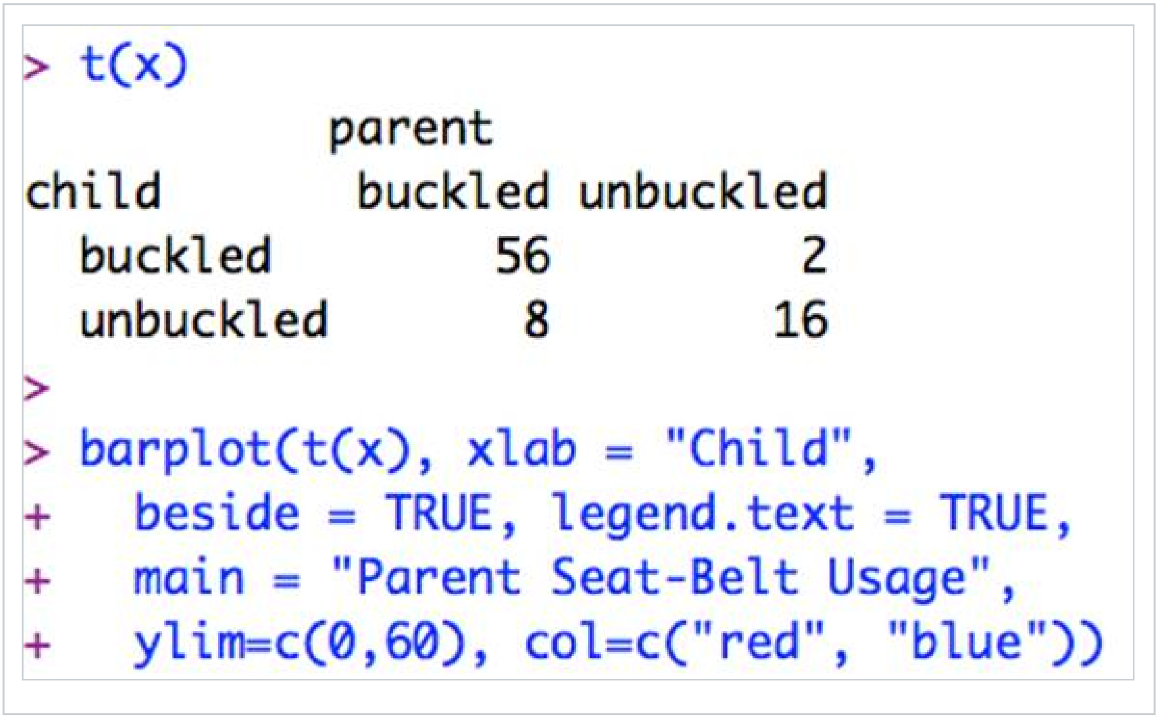
If the bars for the second variable are to be plotted side by side, the option `beside=TRUE` is specified. A legend can also be added to the plot as shown.



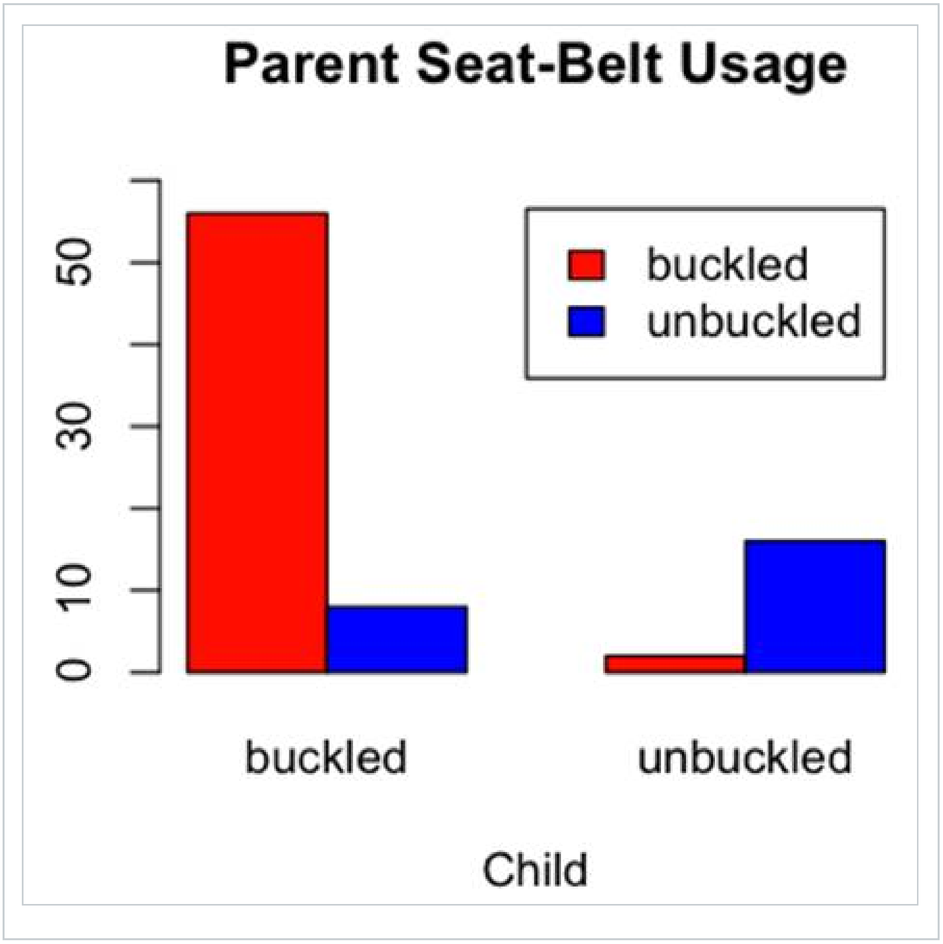
```
> barplot(x, xlab = "Parent",
+   beside = TRUE, legend.text = TRUE,
+   main = "Child Seat-Belt Usage",
+   ylim=c(0,60), col=c("red", "blue"))
```



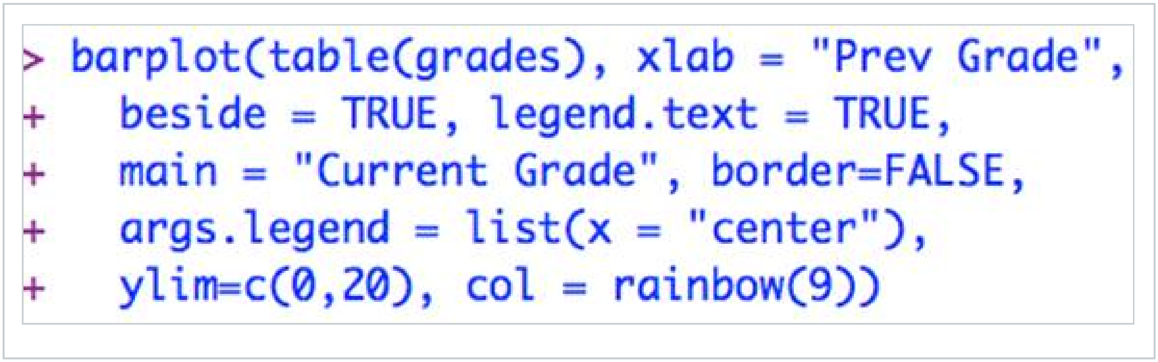
The above plots show the distribution of the `child` variable. However, if the `parent` distribution is desired, the table is first transposed and then the plot is done as shown below.



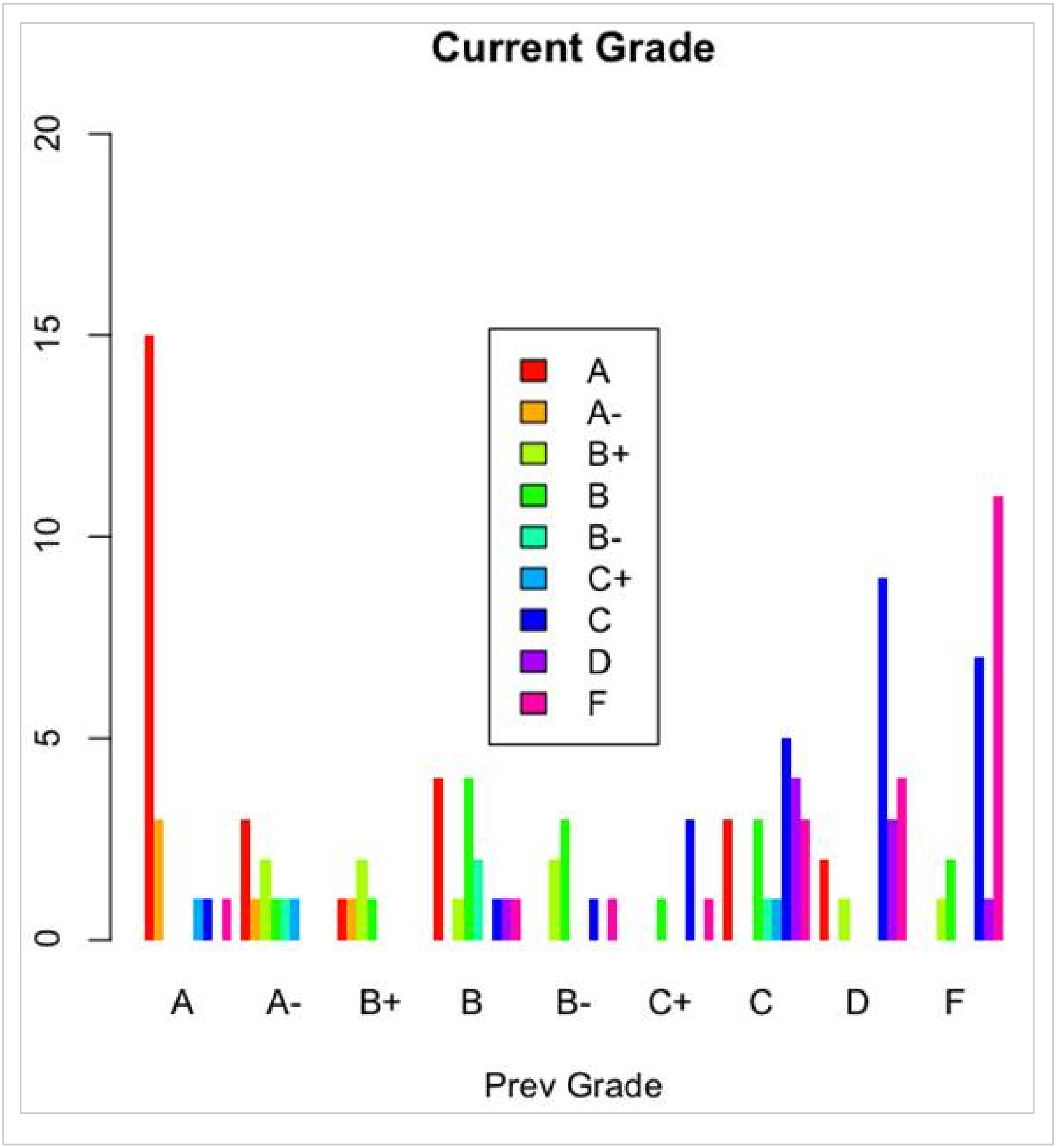
```
> t(x)
           parent
child       buckled unbuckled
  buckled        56         2
  unbuckled       8        16
>
> barplot(t(x), xlab = "Child",
+   beside = TRUE, legend.text = TRUE,
+   main = "Parent Seat-Belt Usage",
+   ylim=c(0,60), col=c("red", "blue"))
```



Similarly, the bar plot for the grades dataset is shown below. The colors for the nine categories of grades can be explicitly specified, or the `rainbow` function can be used to create a vector of the required number of colors.



```
> barplot(table(grades), xlab = "Prev Grade",
+   beside = TRUE, legend.text = TRUE,
+   main = "Current Grade", border=FALSE,
+   args.legend = list(x = "center"),
+   ylim=c(0,20), col = rainbow(9))
```



Test Yourself 3.17

Before continuing, let's replace our table with some different values. Replace the contents of the table produced in the previous problems with the values seen below:

10	1	0	0
6	9	0	0
1	3	5	1
0	1	8	4

You may do this however you like but a recommendation is to assign the rows or columns of the matrix to these new values so as to maintain the dimension names and labels of your data. Show the R commands.

```
data[1, ] <- c(10, 1, 0, 0)
data[2, ] <- c(6, 9, 0, 0)
data[3, ] <- c(1, 3, 5, 1)
data[4, ] <- c(0, 1, 8, 4)
```

Test Yourself 3.18

Now that we have our new table, produce a mosaic plot of the table, use 4 different colors. Show the R commands.

```
mosaicplot(data, color=c('green', 'yellow', 'red', 'black'))
```

Test Yourself 3.19

Show a barplot of the data where the x-axis indicates the patient's disease status before treatment. You may use a stacked barplot or set beside=TRUE. In any case, you must label your axes and include a legend. Show the R commands.

```
barplot(data, xlab='Status.After.Treatment', beside=TRUE, main='Patient di
```

Test Yourself 3.20

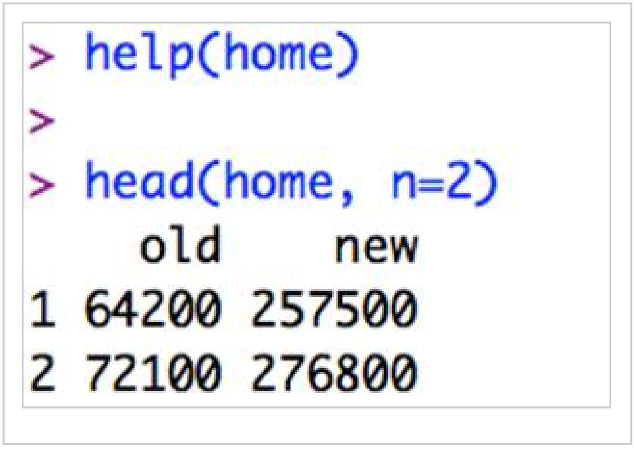
Load the iris data set using data('iris'). Produce a scatter plot of the Sepal Length vs. the Sepal Width. Label the axes. Show the R commands.

```
data("iris")
plot(iris$Sepal.Length, iris$Sepal.Width)
```

3.7. Relationships in Numeric Data

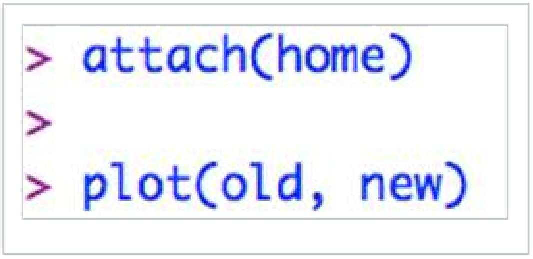
The two-way table allows us to investigate the relationship between categorical variables. In this section, the relationship between numerical variables is explored. The bivariate data of two numeric variables X and Y has a natural pairing, $(x_1, y_1), (x_2, y_2), \dots (x_n, y_n)$. In some cases, the variables are related and in other cases, no relation exists between them. The scatterplot is one option to explore the relation between the two numeric variables. The `home` dataset (from the

UsingR package) shows the old and the new prices of 15 homes from Maplewood, NJ. The home prices were assessed first in 1970 and then again in 2000, after 30 years. The first two rows of the dataset are shown below.

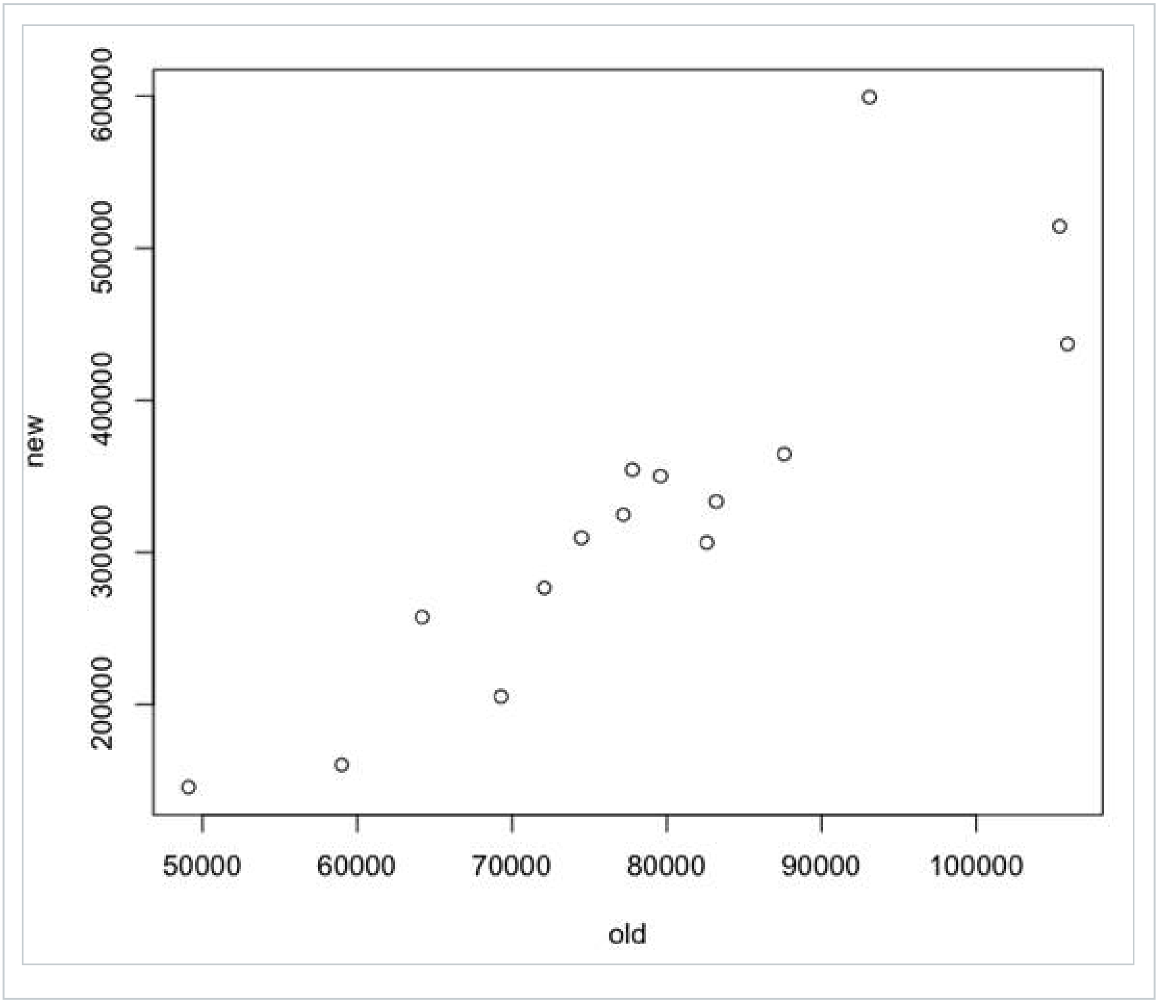


```
> help(home)
>
> head(home, n=2)
    old    new
1 64200 257500
2 72100 276800
```

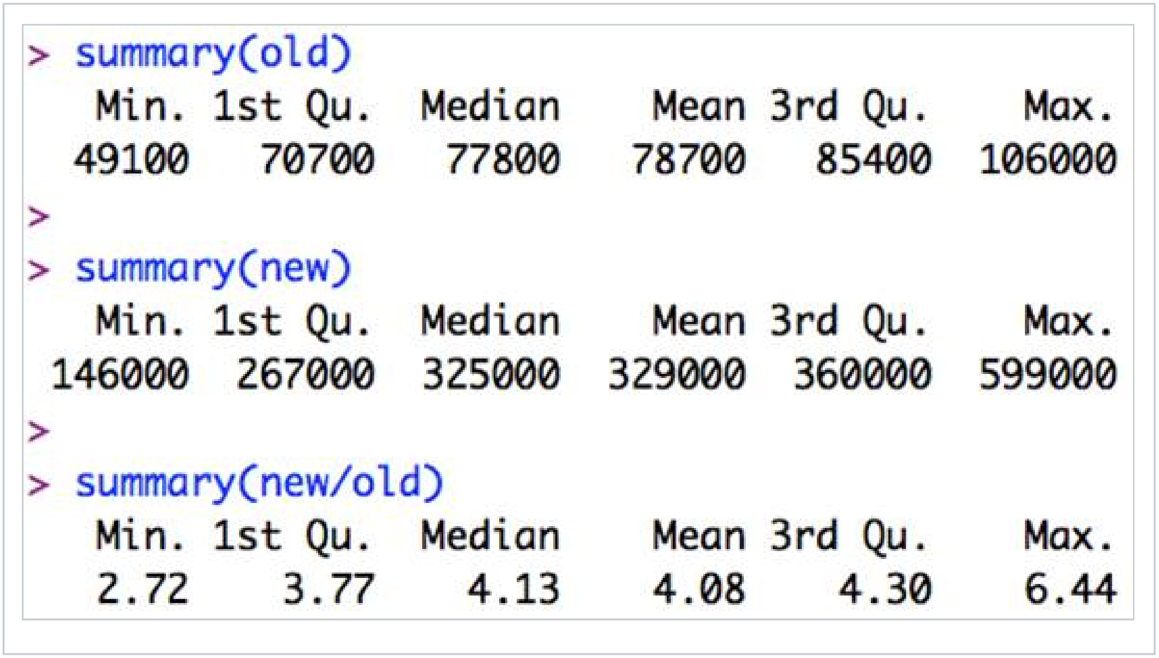
A scatterplot is drawn using the `plot` function. The `old` home prices are plotted against the `new` home prices.



```
> attach(home)
>
> plot(old, new)
```

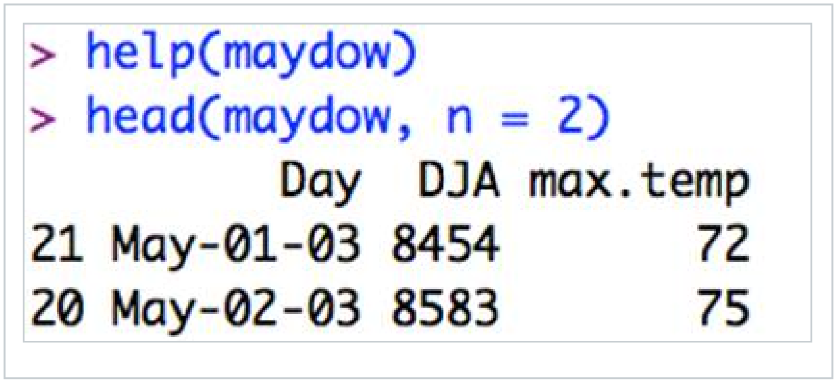


With a few exceptions, the data falls along a straight line showing a linear increase in the home prices. The five-number summary of the old prices and the new prices are shown below. The ratio of the two shows that some houses doubled in value, while others increased to six times the original value. On the average, the prices increased by four times.



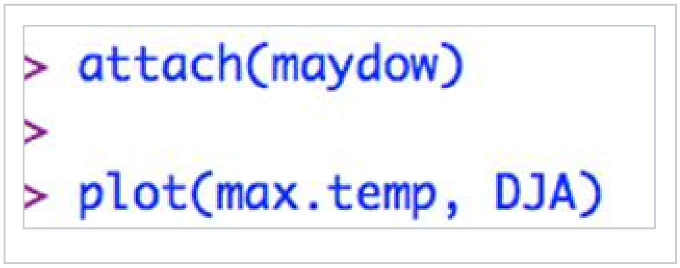
```
> summary(old)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  49100   70700   77800   78700   85400  106000
>
> summary(new)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 146000  267000  325000  329000  360000  599000
>
> summary(new/old)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   2.72    3.77    4.13    4.08    4.30    6.44
```

The next example shows the Dow Jones dataset and the maximum daily temperature in New York City for the month of May 2003.

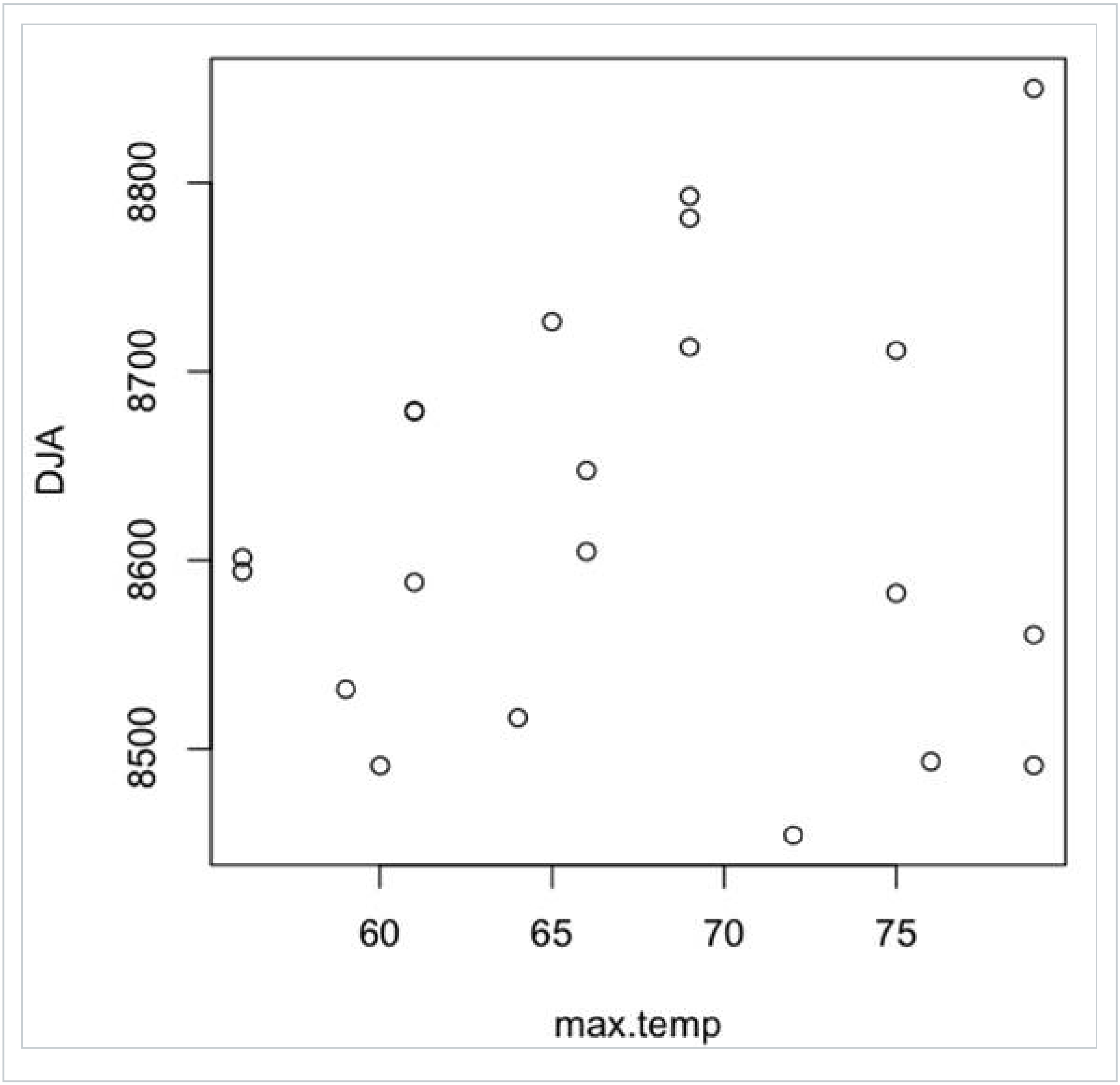


```
> help(maydow)
> head(maydow, n = 2)
         Day  DJA max.temp
21 May-01-03 8454       72
20 May-02-03 8583       75
```

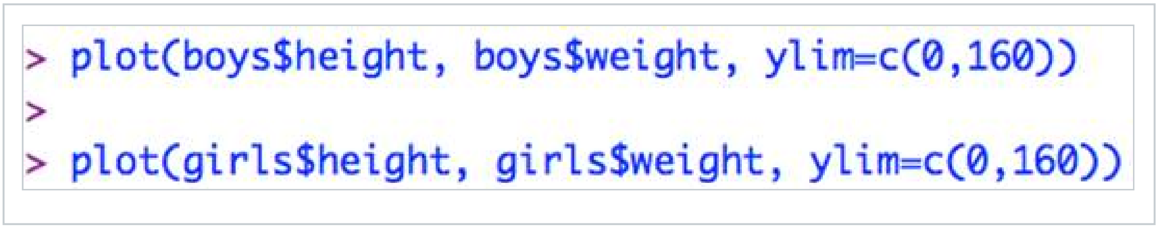
A scatter plot of the data shows that no relation exists between the two variables.



```
> attach(maydow)
>
> plot(max.temp, DJA)
```

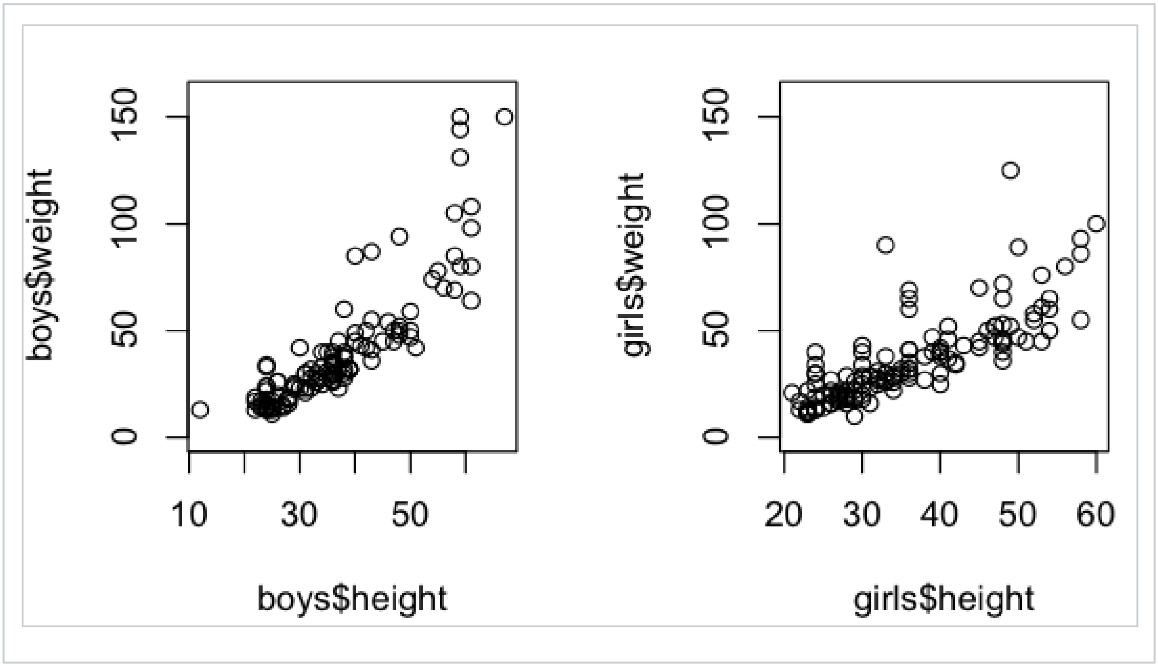


The next dataset (`kid.weights`) contains the weight and height measurement of 250 children. First, we will plot the height versus the weight separately for the boys and girls.

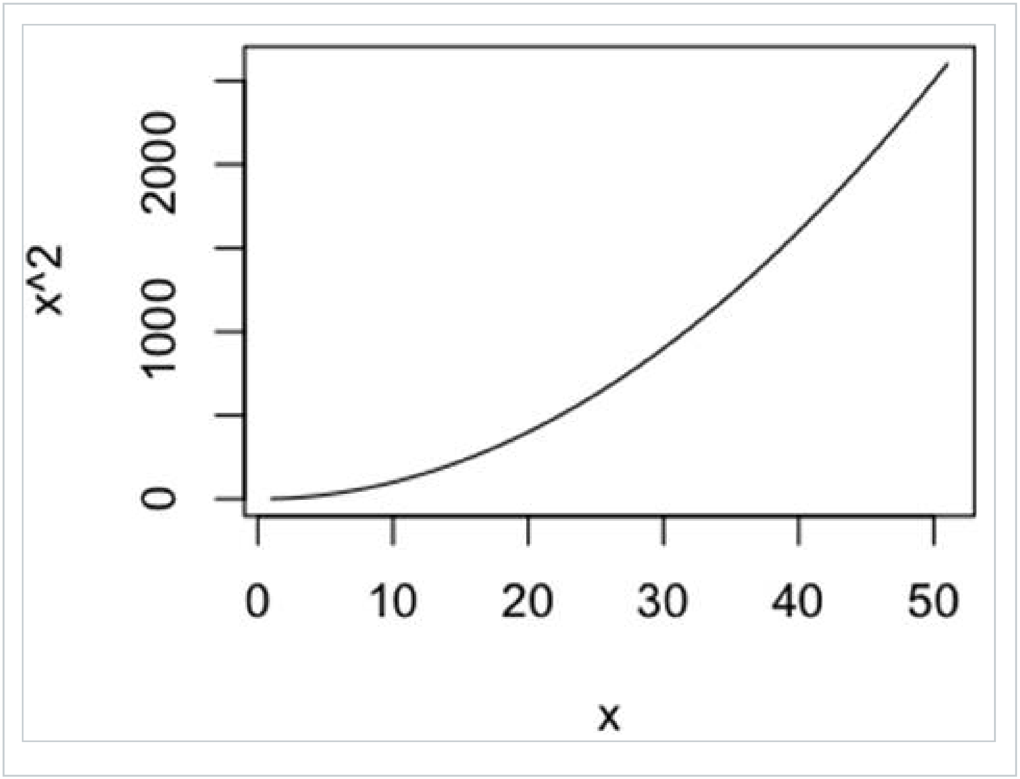


```
> plot(boys$height, boys$weight, ylim=c(0,160))
>
> plot(girls$height, girls$weight, ylim=c(0,160))
```

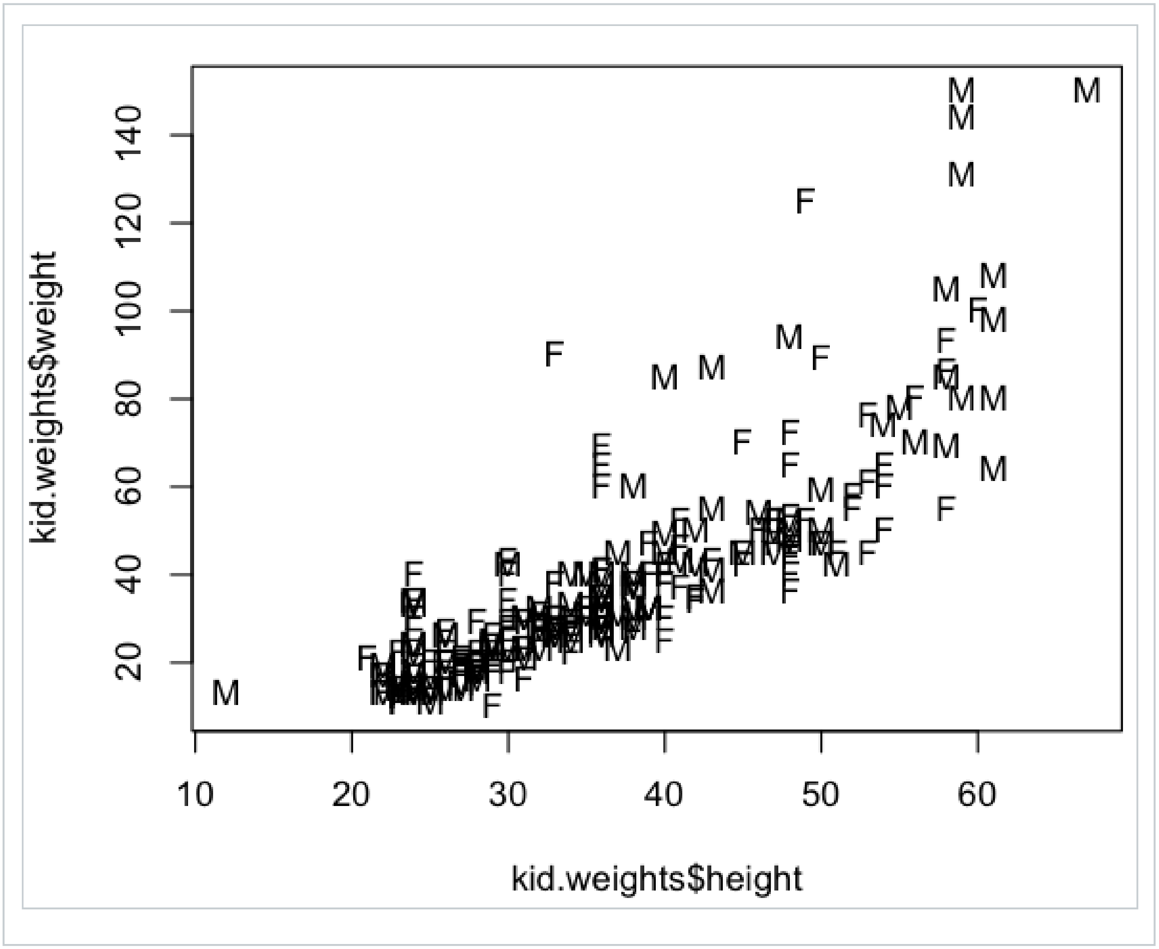
A side-by-side comparison of the two plots is shown below.



The figure suggests that the weight and the height are in a squared relationship. A simple plot of $y = x^2$ is shown below.



The entire data about the boys and the girls together can be plotted in the same figure. The plotting character option can be used to show which data belongs to the boys (`gender = "M"`) and to the girls (`gender = "F"`) in the dataset.

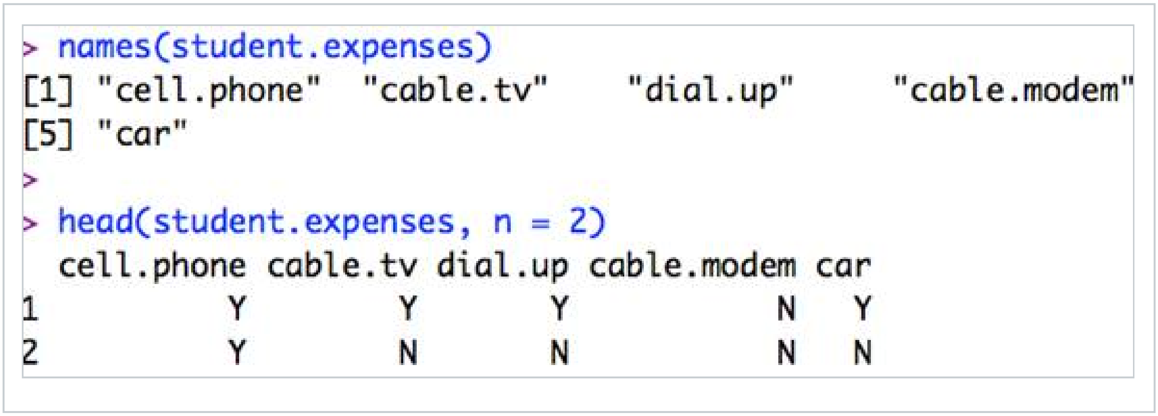


4. Multivariate Data

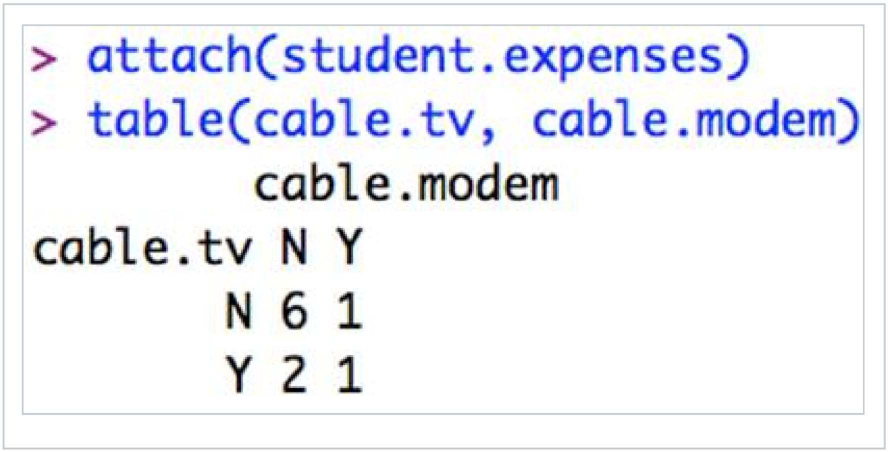
4.1. Multivariate Data

Multivariate datasets have data populated with more than two variables. The data can either be in the summarized form or in the original form. For categorical data, the `table` function may be used to summarize the data given in the original form. But, the `table` function shows only the relationship between two variables at once. For examining three or more variables, more than one table is used to show the relationships.

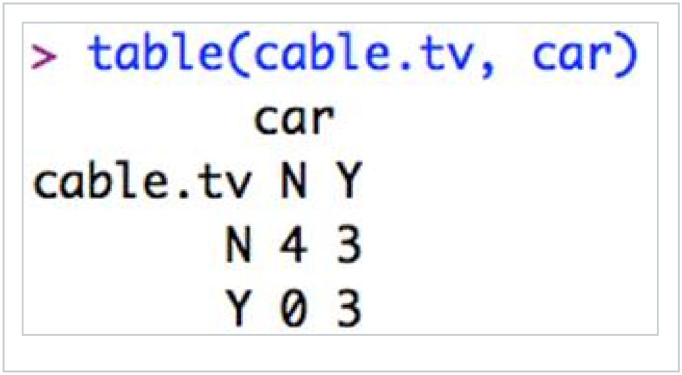
The `student.expenses` dataset (from the `UsingR` package) shows the survey results for ten students asked whether they incurred any expenses in the specified five categories. The categories and the first two rows of the data are shown below.


```
> names(student.expenses)
[1] "cell.phone"  "cable.tv"    "dial.up"     "cable.modem"
[5] "car"
>
> head(student.expenses, n = 2)
  cell.phone cable.tv dial.up cable.modem car
1          Y        Y       Y           N   Y
2          Y        N       N           N   N
```

Examining the relationship between two variables using the `table` function is straightforward. The following code shows the relationship between having `cable TV` and `cable modem`.

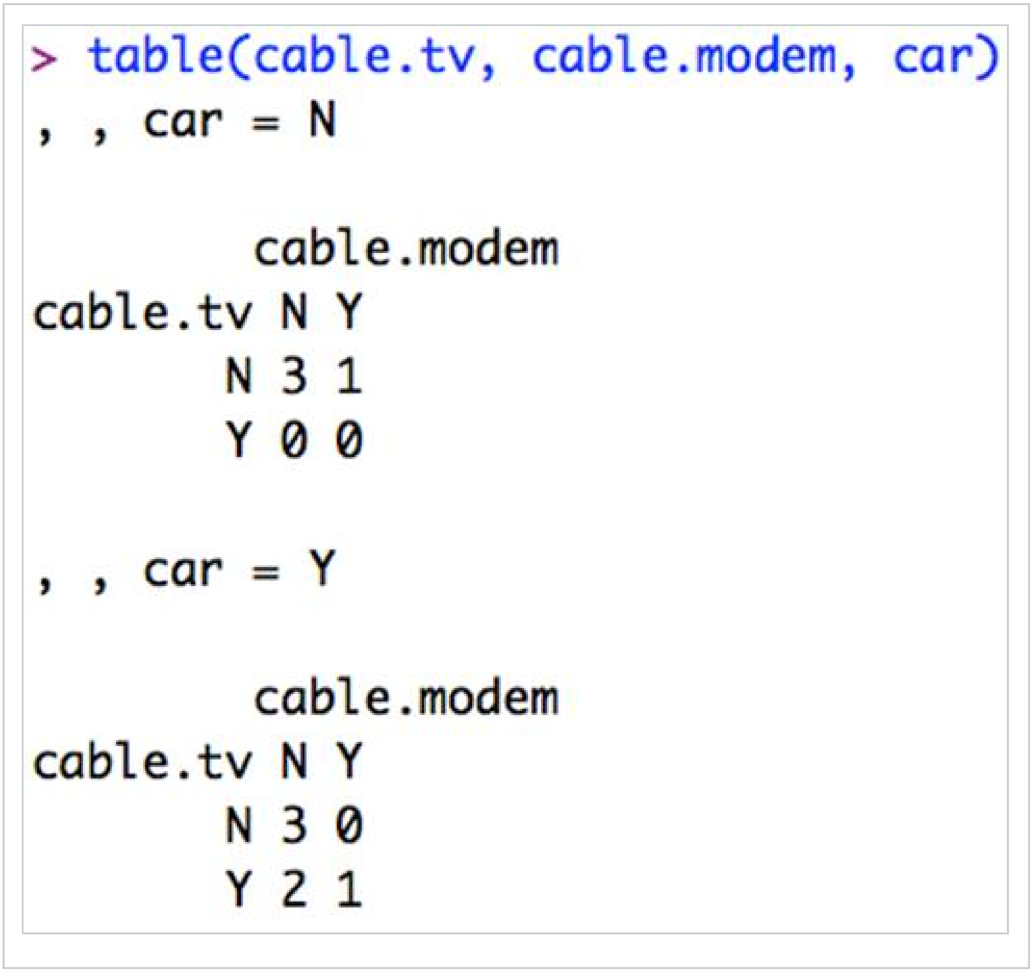

```
> attach(student.expenses)
> table(cable.tv, cable.modem)
        cable.modem
cable.tv N Y
       N 6 1
       Y 2 1
```

Similarly, the relationship between having `cable TV` and `car` is shown below.


```
> table(cable.tv, car)
        car
cable.tv N Y
       N 4 3
       Y 0 3
```

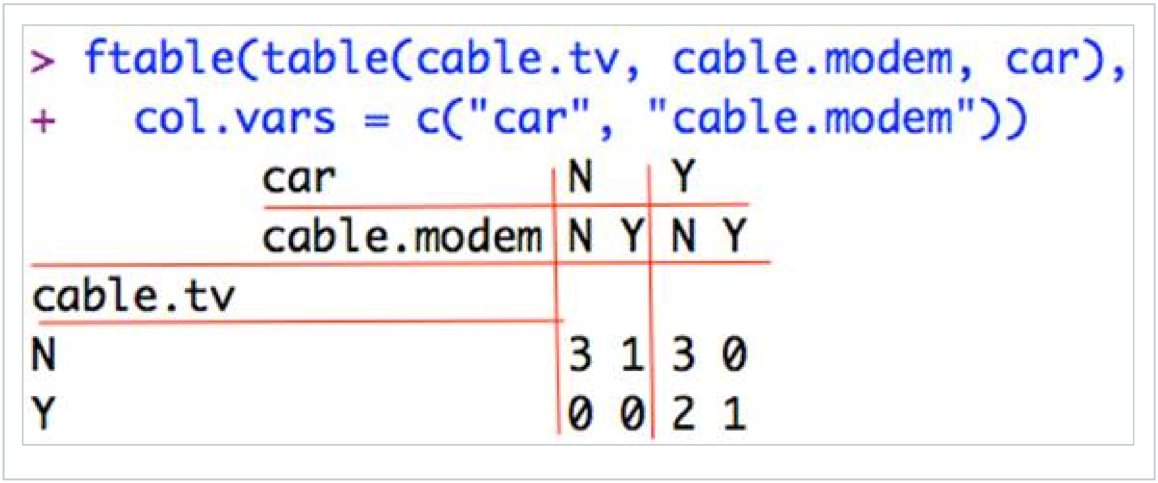
Three-way or (n-way) contingency tables show the relationships among three (or n variables using multiple two-way tables. The `table` function is invoked with the desired variables. The first two variables are fixed in the table output while the number of values for the extra variables determines the number of tables produced.

In the following code, the effect of having a `car` is investigated with respect to the `cable TV` and the `cable modem`. Since the `car` variable has two values, two tables are shown one with the `car` variable equal to N and the other with the car variable equal to Y.



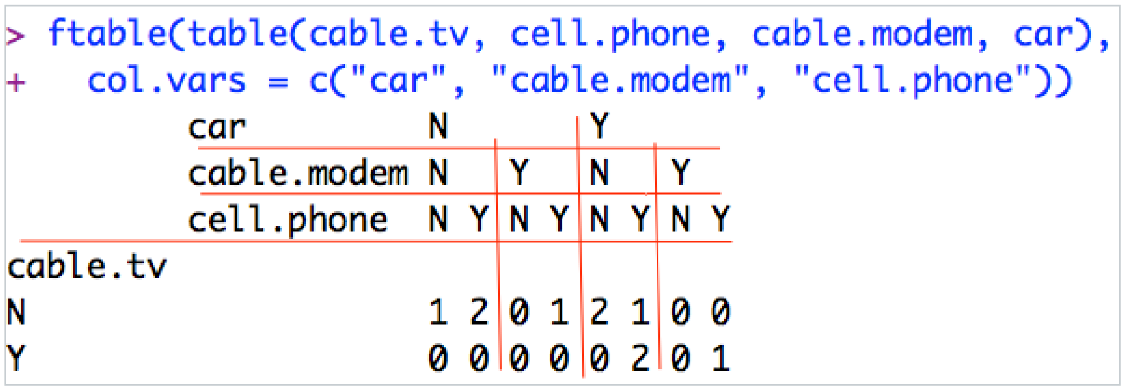
```
> table(cable.tv, cable.modem, car)
, , car = N

        cable.modem
cable.tv N Y
       N 3 1
       Y 0 0

, , car = Y

        cable.modem
cable.tv N Y
       N 3 0
       Y 2 1
```

The above output can be flattened using the `ftable` function. The flatten table function is invoked on the table by specifying the desired columns using the `col.vars` argument.



```
> ftable(table(cable.tv, cable.modem, car),
+   col.vars = c("car", "cable.modem"))
         car         N   Y
         cable.modem N Y N Y
cable.tv
N                    3 1 3 0
Y                    0 0 2 1
```

Similarly, the flattened table with four variables is shown below.



```
> ftable(table(cable.tv, cell.phone, cable.modem, car),
+   col.vars = c("car", "cable.modem", "cell.phone"))
          car          N       Y
          cable.modem  N   Y   N   Y
          cell.phone   N Y N Y N Y N Y
cable.tv
N                      1 2 0 1 2 1 0 0
Y                      0 0 0 0 0 2 0 1
```

The red lines are superimposed for clarity

Test Yourself 3.21

Load the motor trends data set using data('mtcars'). How many variables does each entry contain?

11

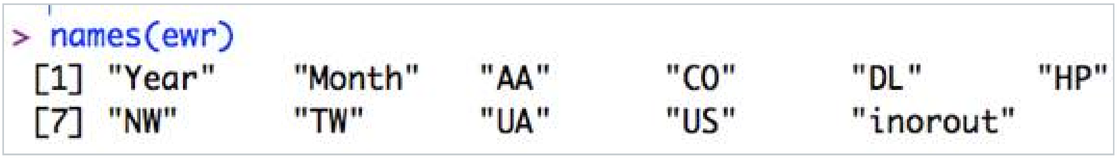
Test Yourself 3.22

Using the mtcars data set, slice the data frame such that the resulting data frame contains only the columns 'cyl', 'disp', 'hp', 'gear'. Make sure the resulting data frame has the same row names as that of mtcars. Using mtcars and the table() function to produce two-way tables of the 'cyl' vs the other three variables above. Show all R commands.

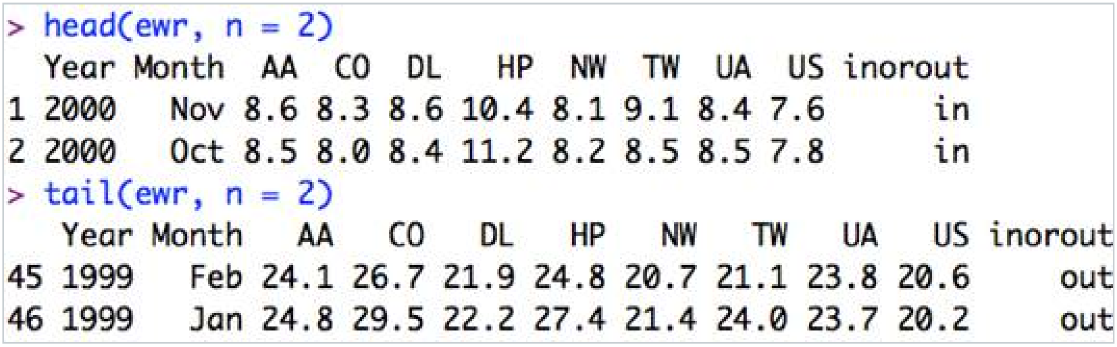
```
attach(mtcars)
data <- cbind(cyl, disp, hp, gear)
row.names(data) <- row.names(mtcars)
table(cyl, disp)
table(cyl, hp)
table(cyl, gear)
```

4.2. Independent Samples

In the `student.expenses` dataset, the variables are dependent for the particular student. However, if the dataset has several variables of the same type, the data can be compared using their means and spreads through box plots. The `ewr` dataset (from the `UsingR` package) has the monthly average taxi in and taxi out times of 8 different airlines at the Newark (EWR) airport during the months from January 1999 to November 2000.

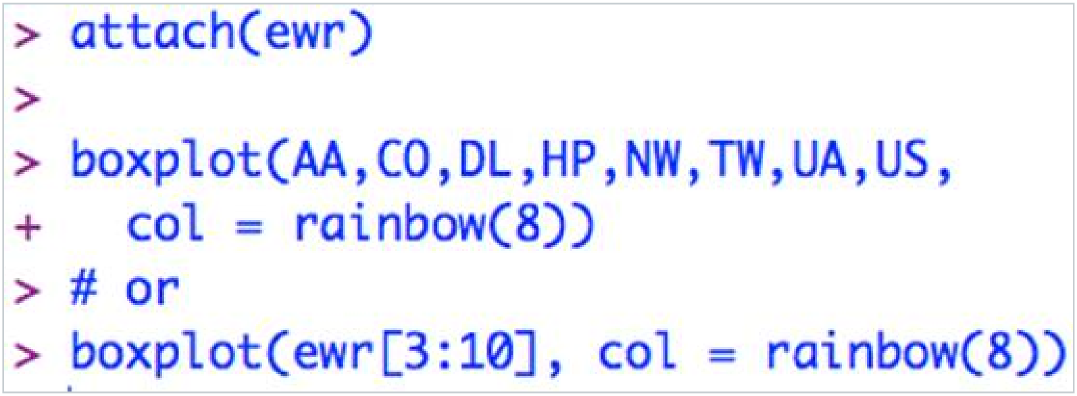


```
> names(ewr)
 [1] "Year"    "Month"   "AA"      "CO"      "DL"      "HP"
 [7] "NW"      "TW"      "UA"      "US"      "inorout"
```

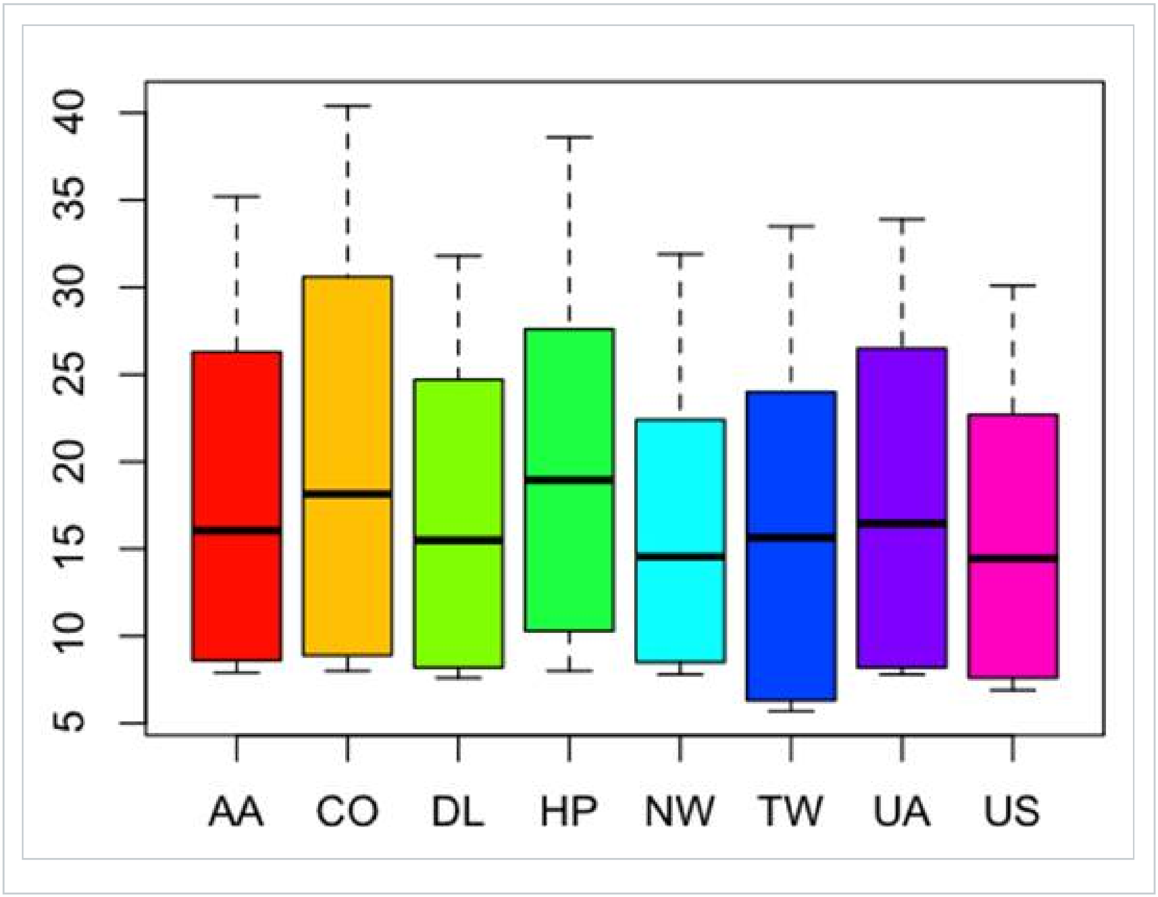


```
> head(ewr, n = 2)
  Year Month  AA  CO  DL   HP  NW  TW  UA  US inorout
1 2000   Nov 8.6 8.3 8.6 10.4 8.1 9.1 8.4 7.6      in
2 2000   Oct 8.5 8.0 8.4 11.2 8.2 8.5 8.5 7.8      in
> tail(ewr, n = 2)
   Year Month   AA   CO   DL   HP   NW   TW   UA   US inorout
45 1999   Feb 24.1 26.7 21.9 24.8 20.7 21.1 23.8 20.6     out
46 1999   Jan 24.8 29.5 22.2 27.4 21.4 24.0 23.7 20.2     out
```

A box plot of the average times of the given airlines shows the differences in the performances of the airlines. The data for both the taxi in and taxi out times is plotted as shown below.

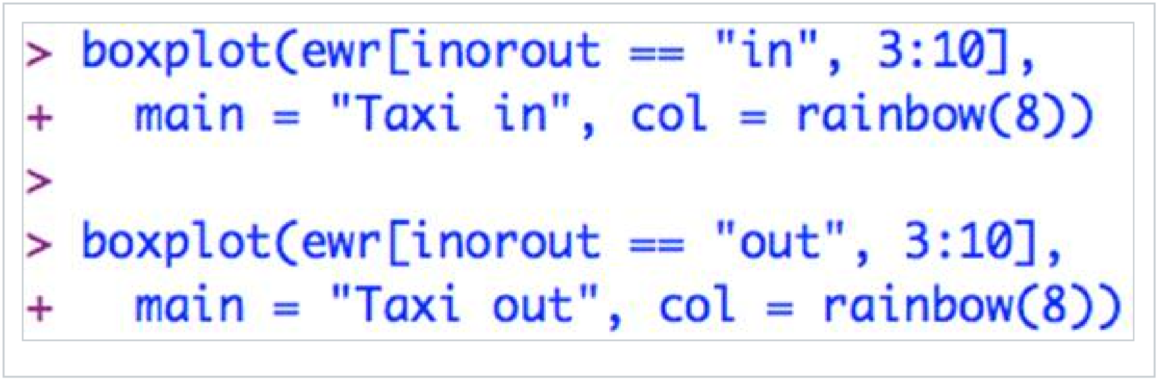


```
> attach(ewr)
>
> boxplot(AA,CO,DL,HP,NW,TW,UA,US,
+   col = rainbow(8))
> # or
> boxplot(ewr[3:10], col = rainbow(8))
```

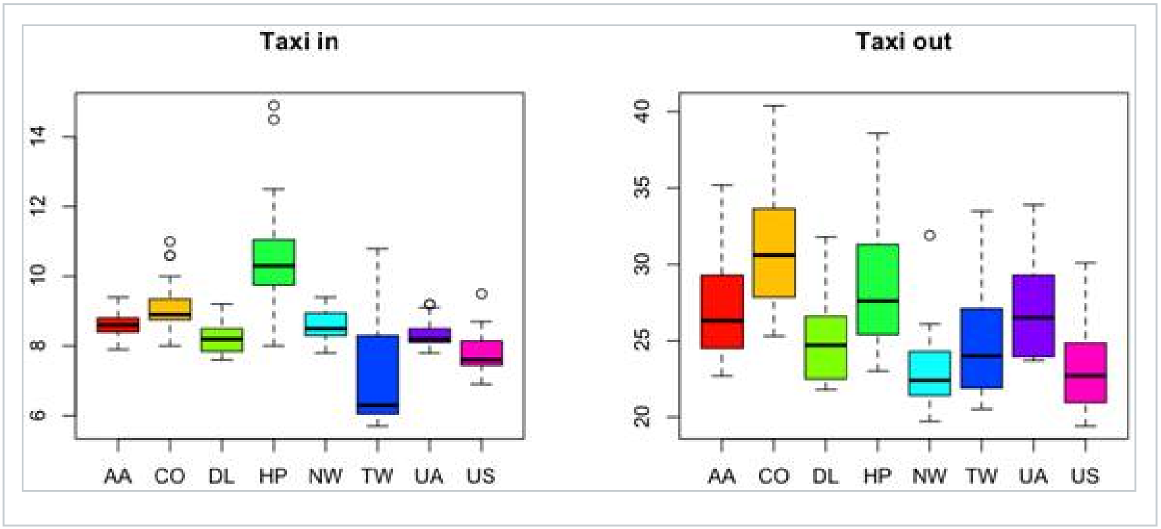


From the above figure, the second airline has the worst performance. The fourth airline has the largest median.

The given data has both the taxi in and taxi out times. The taxi in time is relatively smaller than the taxi out time. Investigating these two times separately provides more insight into the data. Filtering the data based on the `inorout` variable, the following plot shows the taxi in times and taxi out times of the given airlines side by side.



```
> boxplot(ewr[inorout == "in", 3:10],
+   main = "Taxi in", col = rainbow(8))
>
> boxplot(ewr[inorout == "out", 3:10],
+   main = "Taxi out", col = rainbow(8))
```



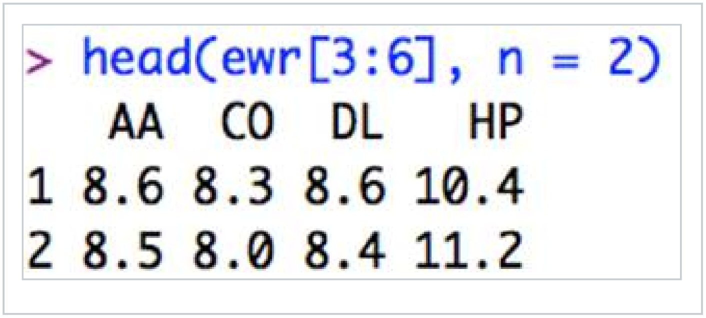
Test Yourself 3.23

Using the iris data set, produce boxplots of the sepal width, sepal length, petal width, and petal length. Slice the iris data set so it contains only flowers of the ‘virginica’ species. Produce boxplots of the virginica sepal width, sepal length, petal width, and petal length. Show the R commands.

```
data("iris")
attach(iris)
boxplot(Sepal.Width, Sepal.Length, Petal.Width, Petal.Length)
data <- iris[species == ‘virginica’, ]
boxplot(data$Sepal.Width, data$Sepal.Length, data$Petal.Width, data$Petal.
```

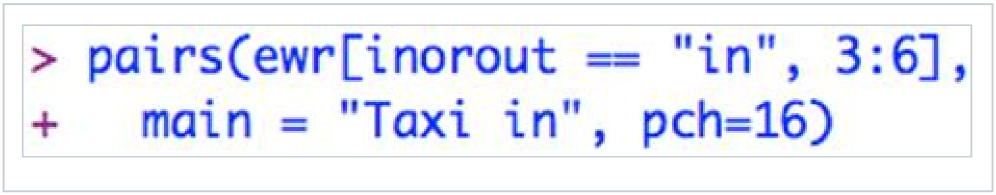
4.3. Relationship Between Pairs of Variables

The scatterplot matrix shows the pair-wise relationships between the given variables using a scatter plot for each pair. For the airline `ewr` dataset, consider the first four airlines (columns 3 to 6).

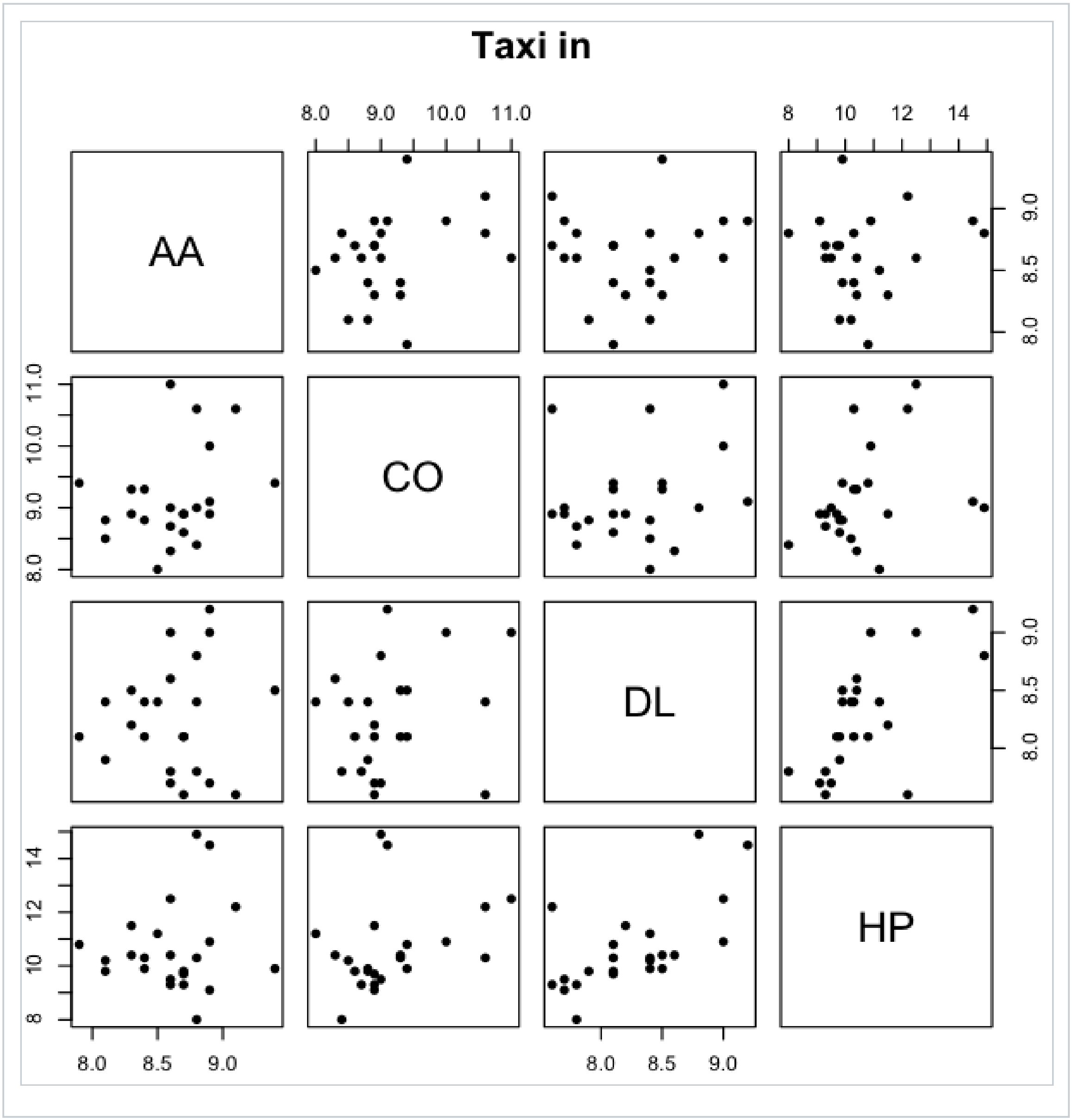


```
> head(ewr[3:6], n = 2)
   AA  CO  DL   HP
1 8.6 8.3 8.6 10.4
2 8.5 8.0 8.4 11.2
```

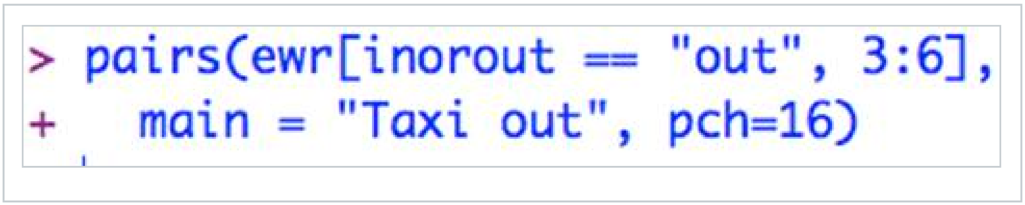
The taxi in times of these airlines are plotted pair-wise as shown below.



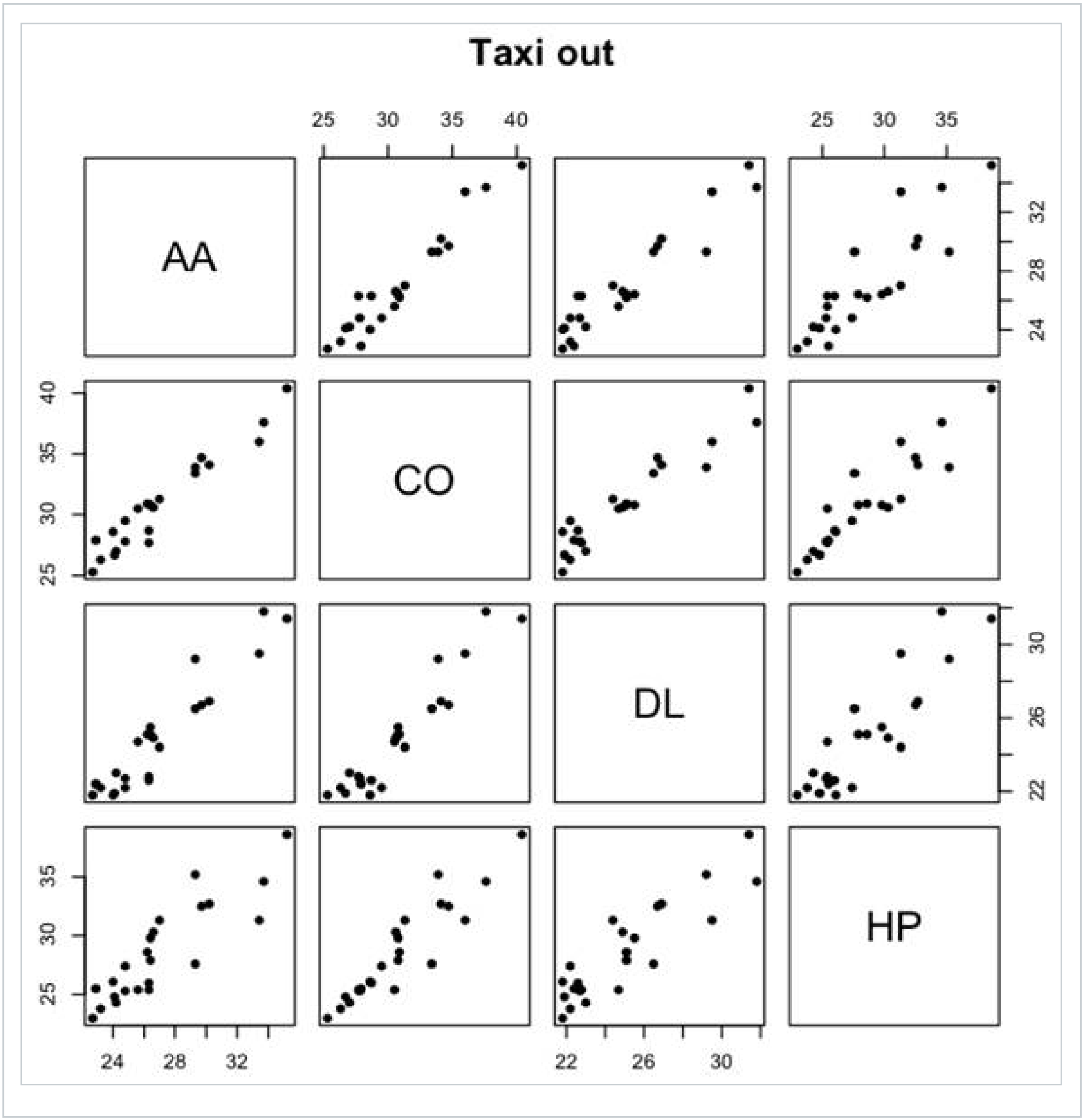
```
> pairs(ewr[inorout == "in", 3:6],
+   main = "Taxi in", pch=16)
```



Similarly, the pair-wise comparison of the taxi out times of the first four airlines is shown below.



```
> pairs(ewr[inorout == "out", 3:6],
+   main = "Taxi out", pch=16)
```



While the taxi in plot does not show any obvious relationship, examining the taxi out plot shows a linear correlation among the timings.

Test Yourself 3.24

Using the mtcars data set, produce a scatter plot of the the 'mpg' vs the 'hp'. Briefly describe any trends you see. Show the R commands.

```
attach(mtcars)
plot(mpg, hp)
```

In general, the mpg and hp are inversely correlated. As hp increases, mpg tends to decrease.

Test Yourself 3.25

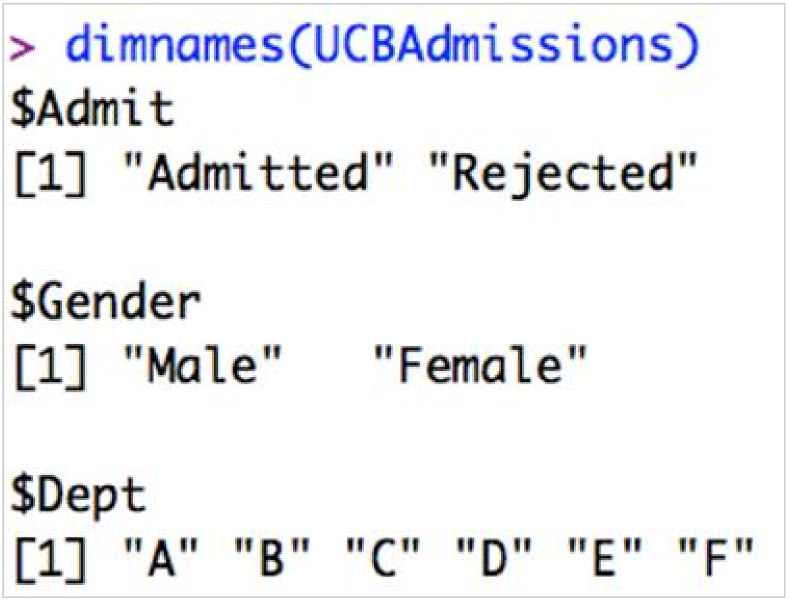
Use the pairs() function to plot the Sepal Width, Length, Petal Width, Length, of the virginica species. Exclude the 'Species' column. Show the R commands.

```
pairs(data[ , 1:4])
```

4.4. Summarized Data

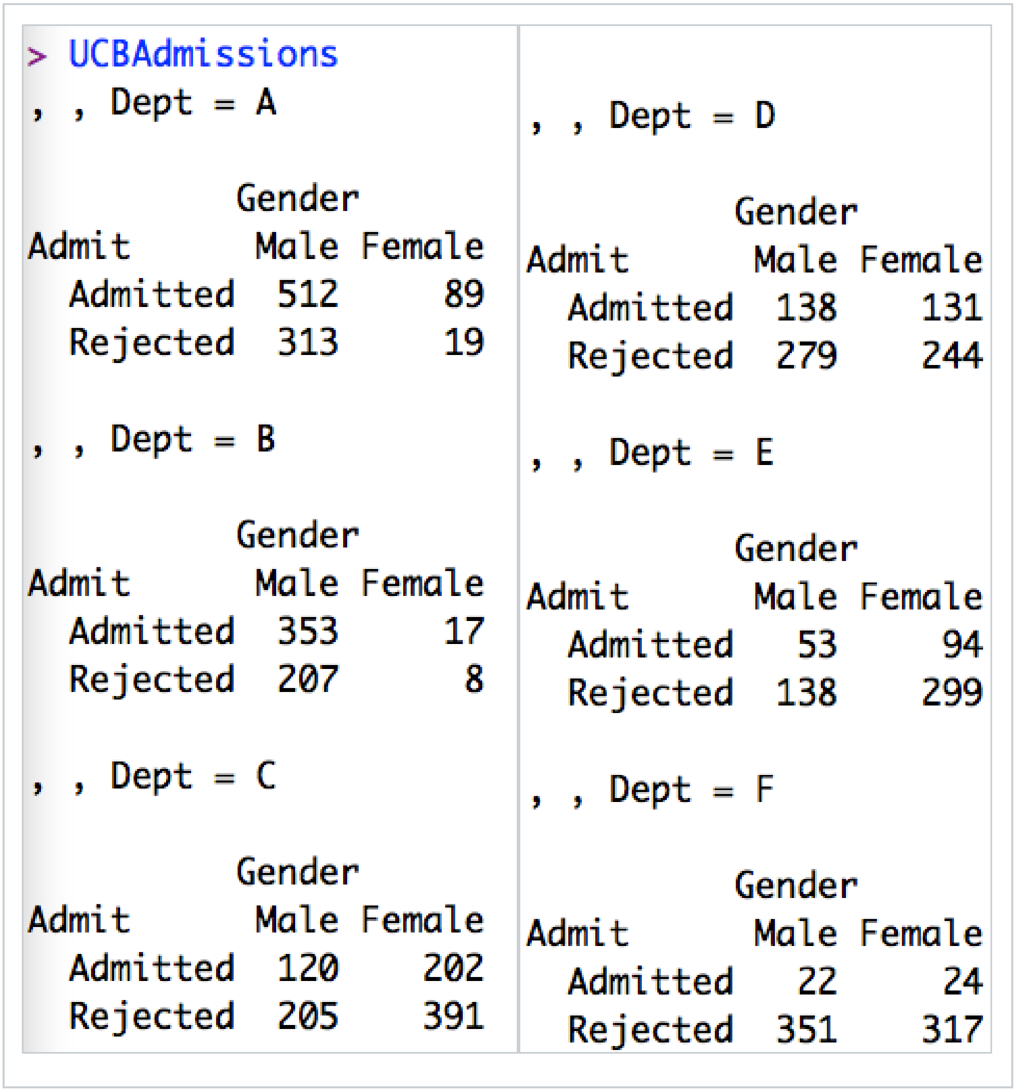
The `UCBAdmissions` dataset contains the summarized data of admissions and rejections of male and female applicants who applied to the top six graduate school departments at UC Berkeley in the year 1973. The data is in a $2 \times 2 \times 6$ table.

The dimension names are `Admit`, `Gender`, and `Dept`. The categorical values for these dimensions are shown below.



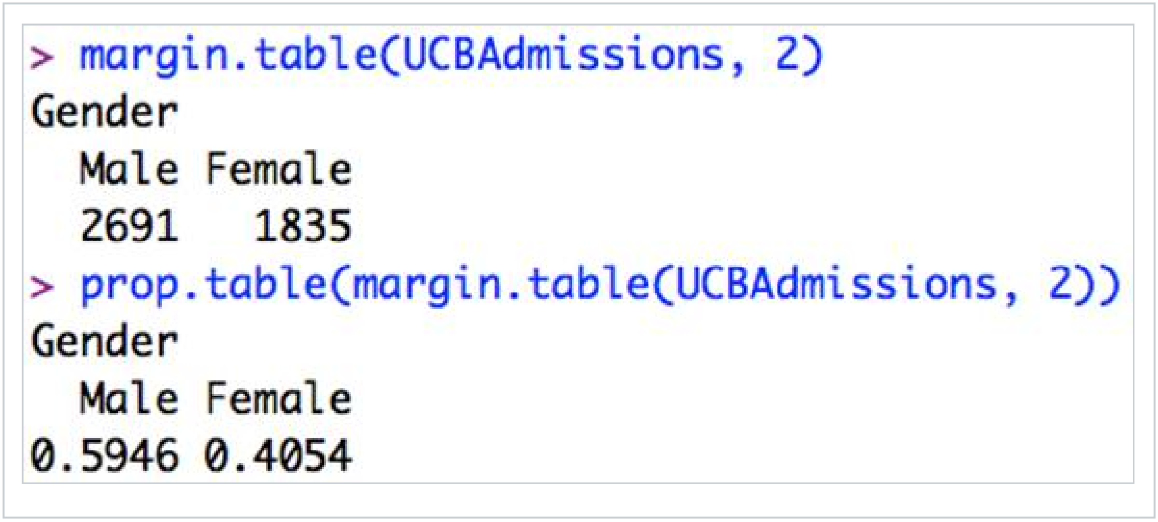
```
> dimnames(UCBAdmissions)
$Admit
[1] "Admitted" "Rejected"

$Gender
[1] "Male"   "Female"

$Dept
[1] "A" "B" "C" "D" "E" "F"
```

The data contains the summarized information about 4526 applicants. The two-way tables for the six departments are shown below.



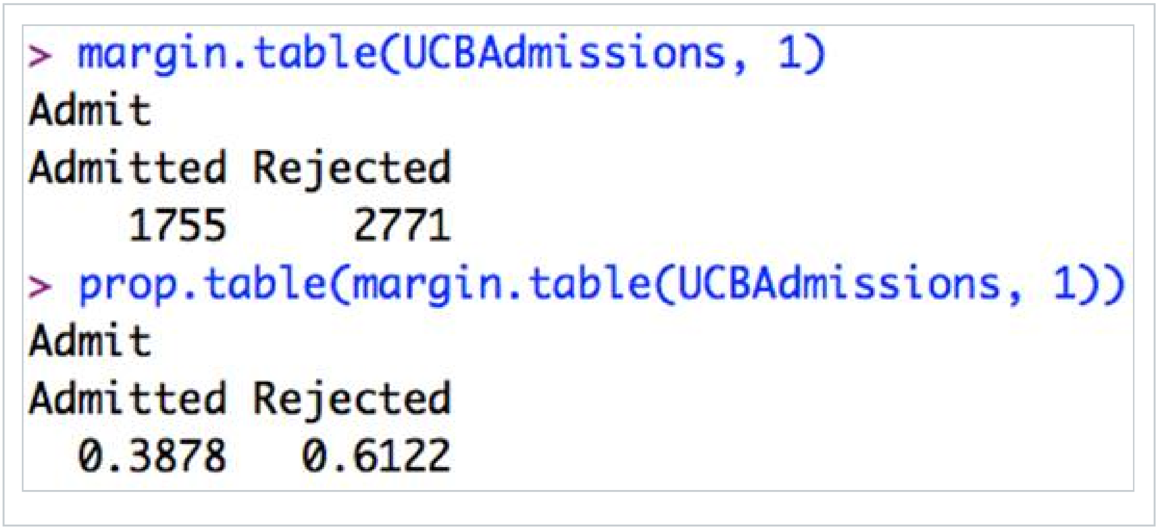
```
> UCBAdmissions
, , Dept = A

          Gender
Admit      Male Female
  Admitted  512     89
  Rejected  313     19

, , Dept = B

          Gender
Admit      Male Female
  Admitted  353     17
  Rejected  207      8

, , Dept = C

          Gender
Admit      Male Female
  Admitted  120    202
  Rejected  205    391

, , Dept = D

          Gender
Admit      Male Female
  Admitted  138    131
  Rejected  279    244

, , Dept = E

          Gender
Admit      Male Female
  Admitted   53     94
  Rejected  138    299

, , Dept = F

          Gender
Admit      Male Female
  Admitted   22     24
  Rejected  351    317
```

The number and proportion of the total male and female applications (the second dimension) can be computed as follows.



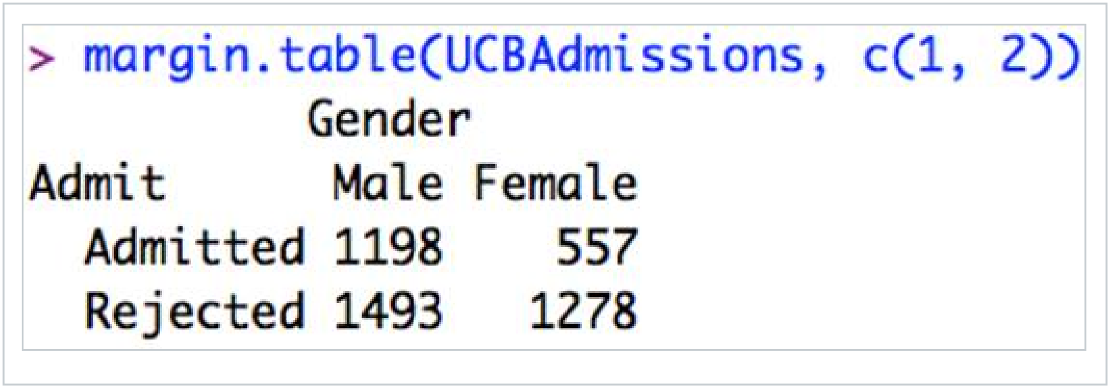
```
> margin.table(UCBAdmissions, 2)
Gender
  Male Female
  2691   1835
> prop.table(margin.table(UCBAdmissions, 2))
Gender
  Male Female
0.5946 0.4054
```

Similarly, the number and proportion of the total admitted and rejected applications (the dimension `1`) are as follows.



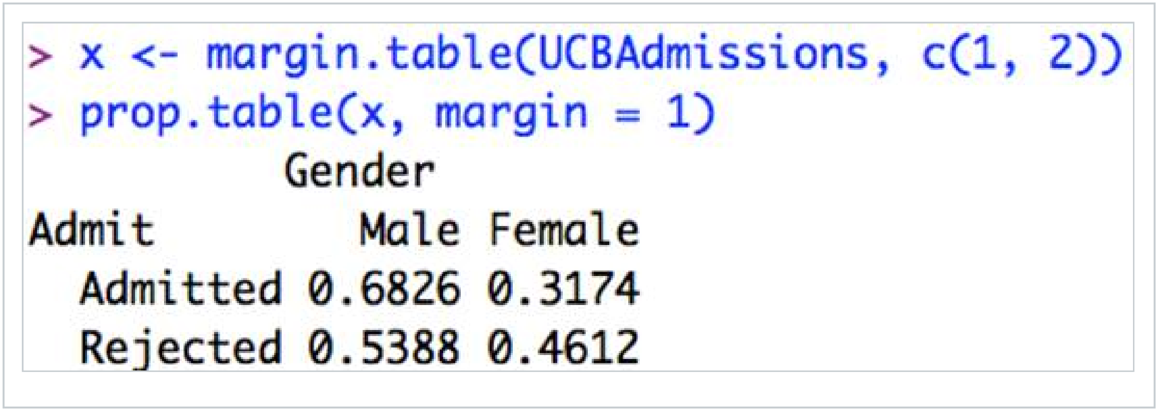
```
> margin.table(UCBAdmissions, 1)
Admit
Admitted Rejected
    1755     2771
> prop.table(margin.table(UCBAdmissions, 1))
Admit
Admitted Rejected
  0.3878   0.6122
```

The marginal distribution of the data over the first two dimensions (adding over all the departments) is shown below.



```
> margin.table(UCBAdmissions, c(1, 2))
          Gender
Admit      Male Female
  Admitted 1198    557
  Rejected 1493   1278
```

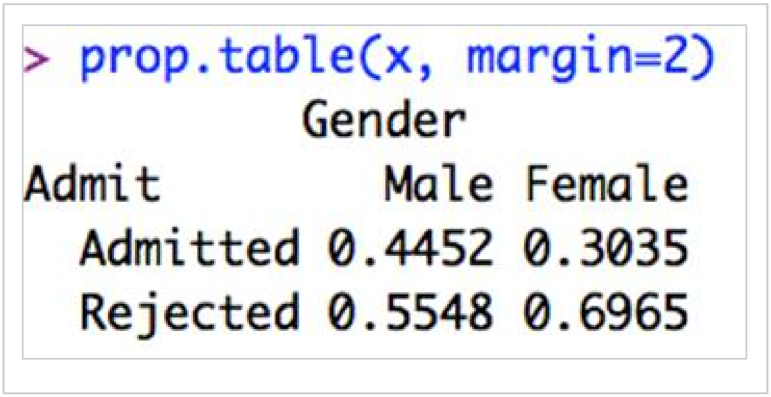
The proportional distribution of the `Admit` and `Gender` dimensions over the margin for rows (`margin = 1`) is:



```
> x <- margin.table(UCBAdmissions, c(1, 2))
> prop.table(x, margin = 1)
          Gender
Admit        Male Female
  Admitted 0.6826 0.3174
  Rejected 0.5388 0.4612
```

The above results show that among the admitted applicants, 68% are male while 32% are female. Among the rejected applicants, 54% are male while 46% are female.

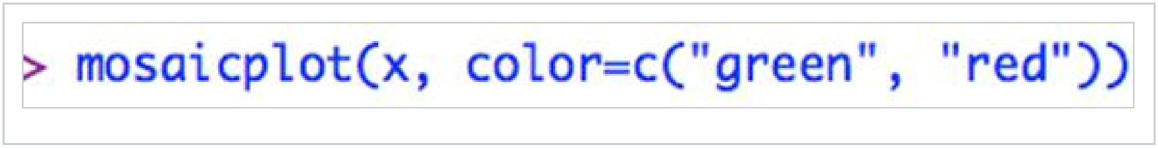
The proportional distribution of the `Admit` and `Gender` dimensions over the margin for columns (`margin = 2`) is:



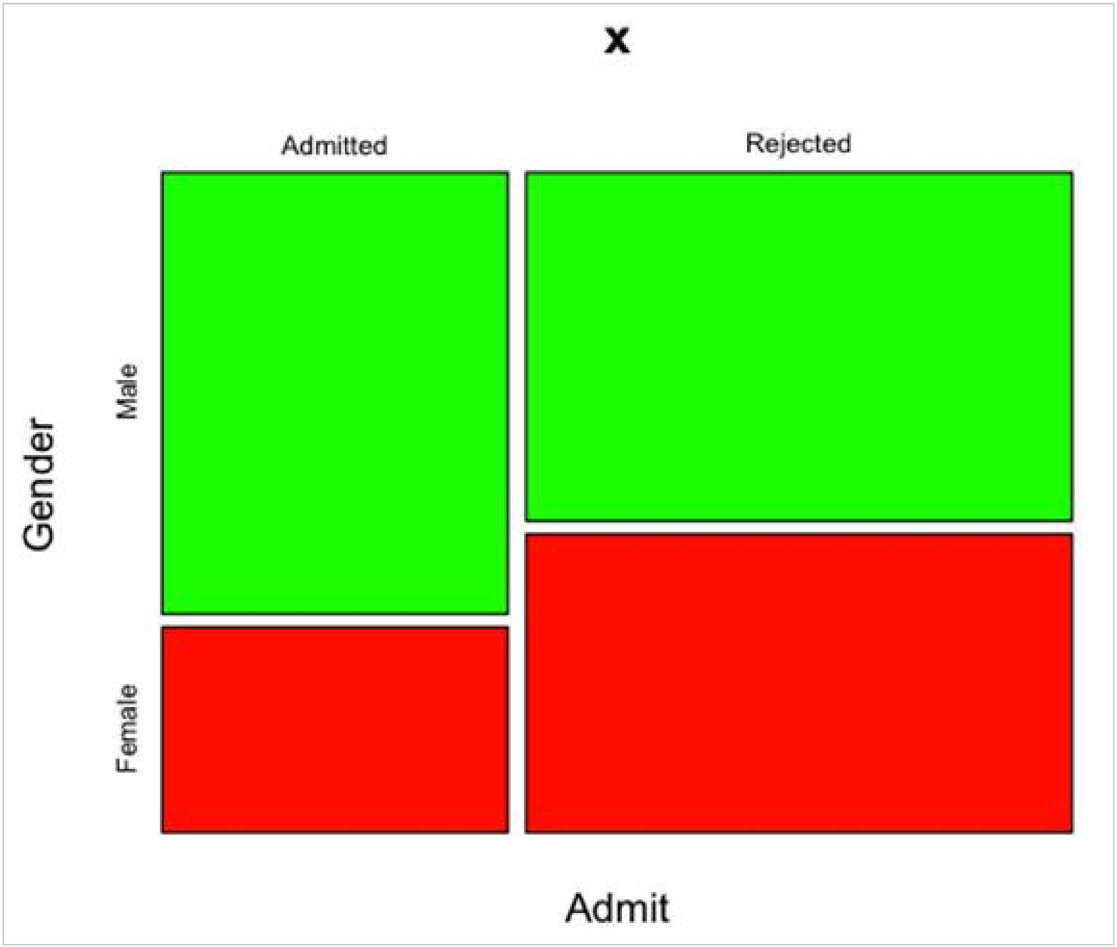
```
> prop.table(x, margin=2)
          Gender
Admit        Male Female
  Admitted 0.4452 0.3035
  Rejected 0.5548 0.6965
```

The above results show that the males have a 45% chance of getting admitted while the females have only a 30% chance of getting admitted.

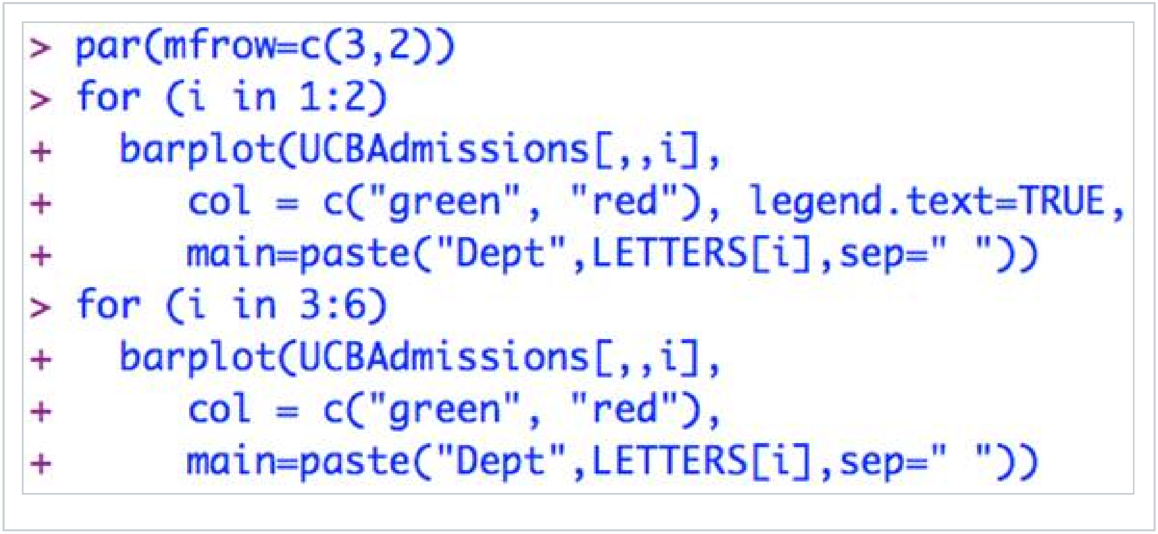
A mosaic plot of the above marginal distribution for `Admit` versus `Gender` is shown below.



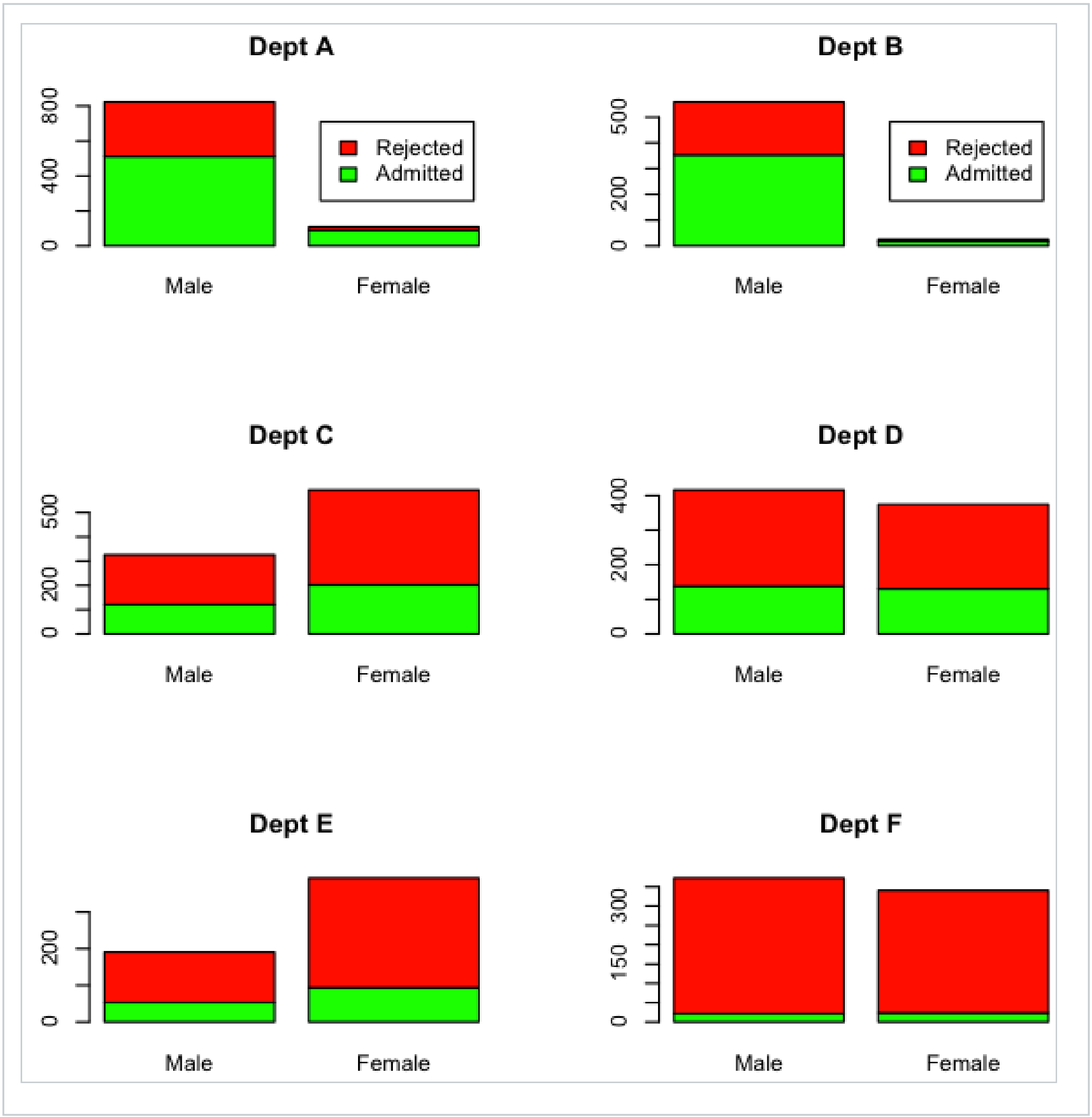
```
> mosaicplot(x, color=c("green", "red"))
```



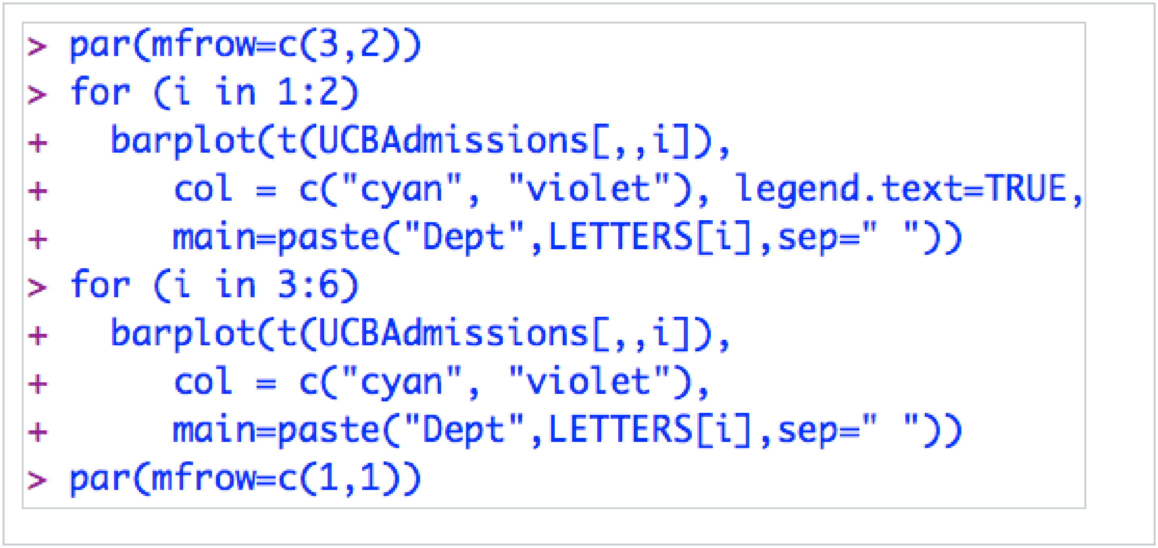
A bar plot showing the information for each department is computed as shown below. For the male and the female applicants, the number of accepted and rejected applicants is shown for each department. The first two departments are drawn separately to show the legend.



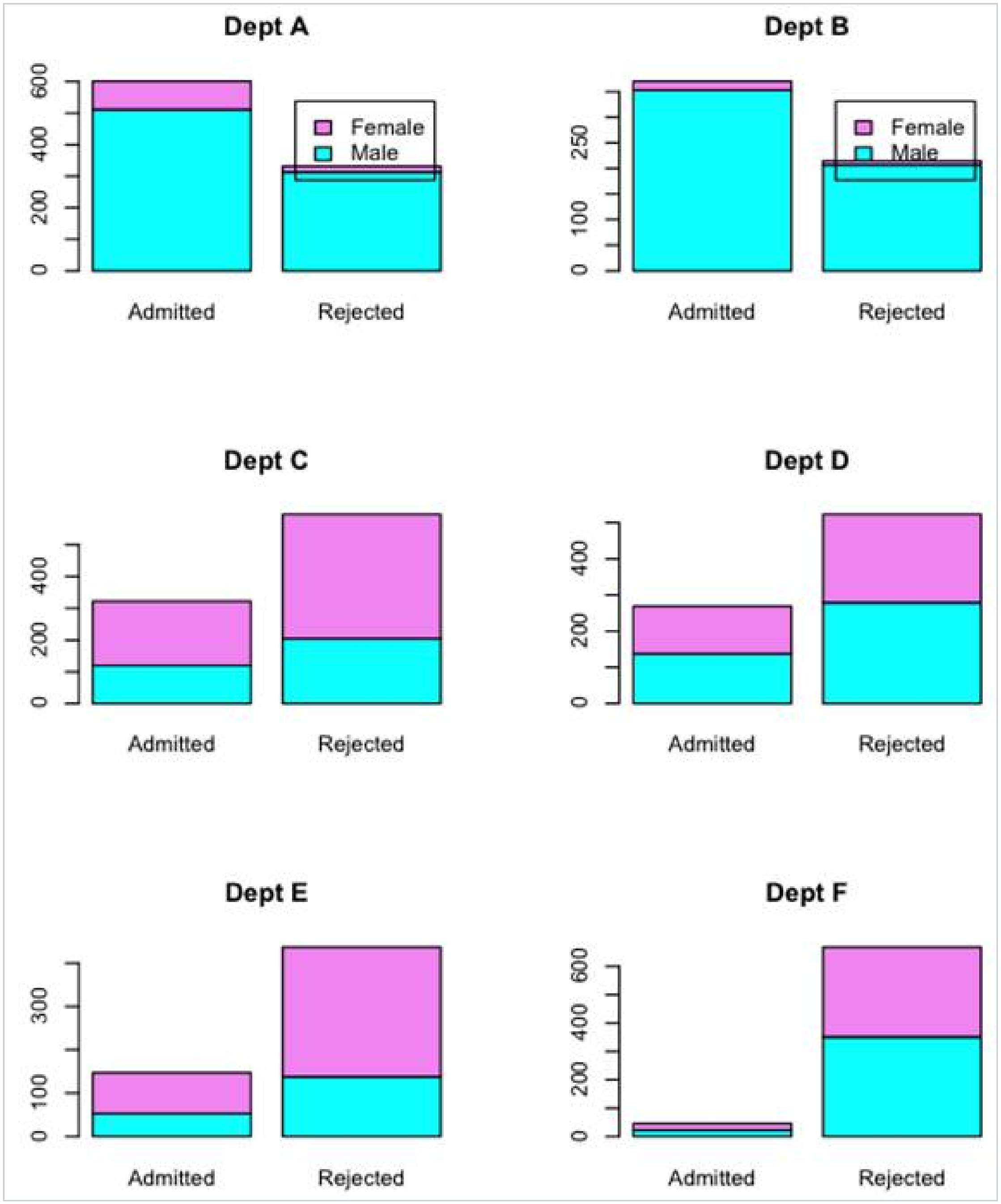
```
> par(mfrow=c(3,2))
> for (i in 1:2)
+   barplot(UCBAdmissions[,,i],
+       col = c("green", "red"), legend.text=TRUE,
+       main=paste("Dept",LETTERS[i],sep=" "))
> for (i in 3:6)
+   barplot(UCBAdmissions[,,i],
+       col = c("green", "red"),
+       main=paste("Dept",LETTERS[i],sep=" "))
```



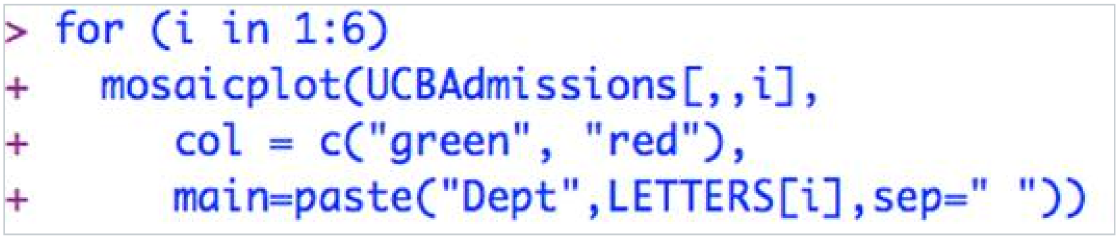
A plot of the transpose of each table provides the information about the number of the male and the female applicants among the accepted and rejected applicants for each department. The transposed data is plotted as shown below.



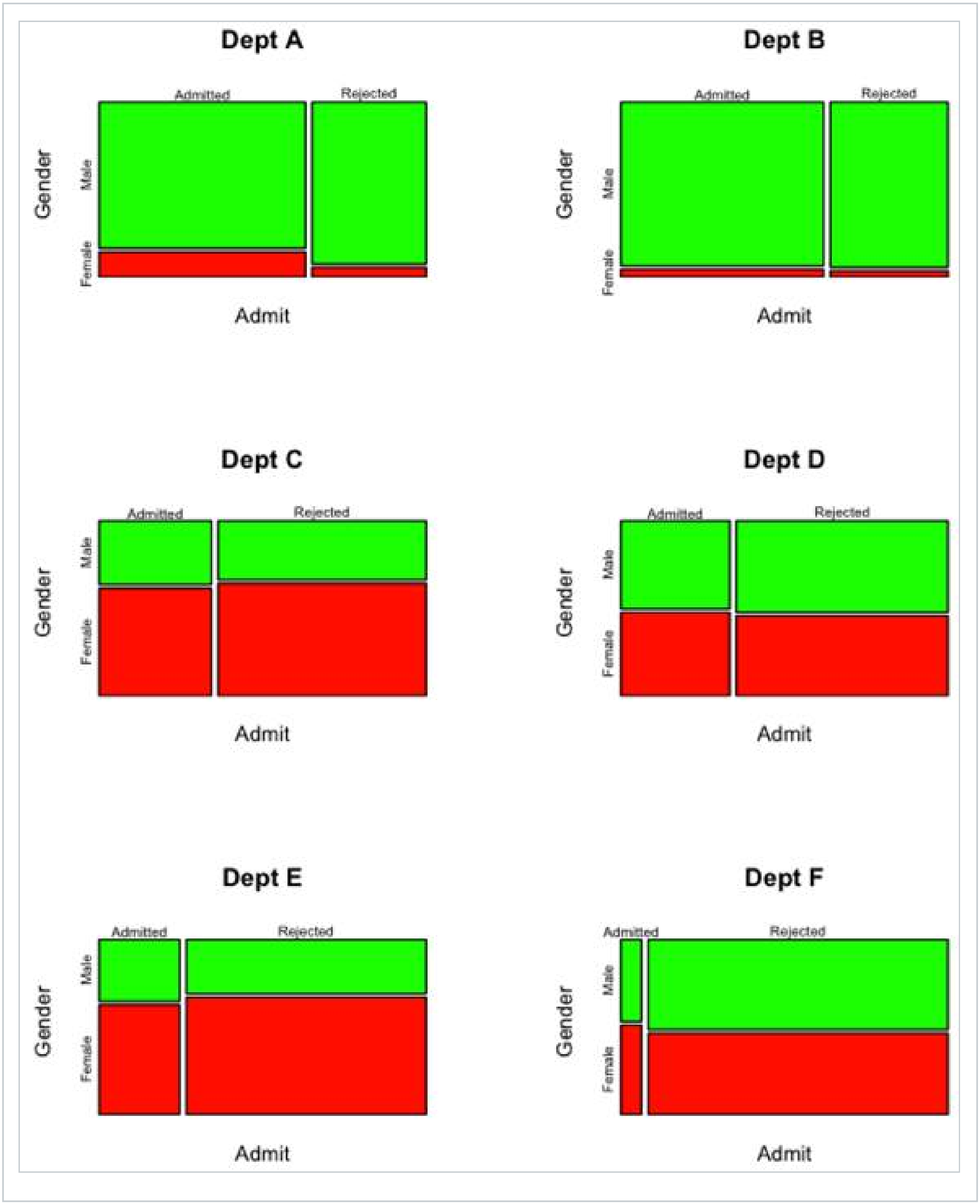
```
> par(mfrow=c(3,2))
> for (i in 1:2)
+   barplot(t(UCBAdmissions[,,i]),
+      col = c("cyan", "violet"), legend.text=TRUE,
+      main=paste("Dept",LETTERS[i],sep=" "))
> for (i in 3:6)
+   barplot(t(UCBAdmissions[,,i]),
+      col = c("cyan", "violet"),
+      main=paste("Dept",LETTERS[i],sep=" "))
> par(mfrow=c(1,1))
```



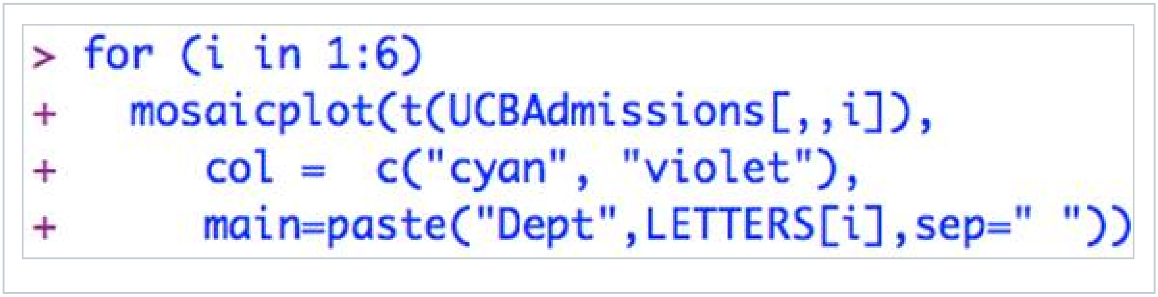
The mosaic plot showing the same information for each department is computed as shown below.



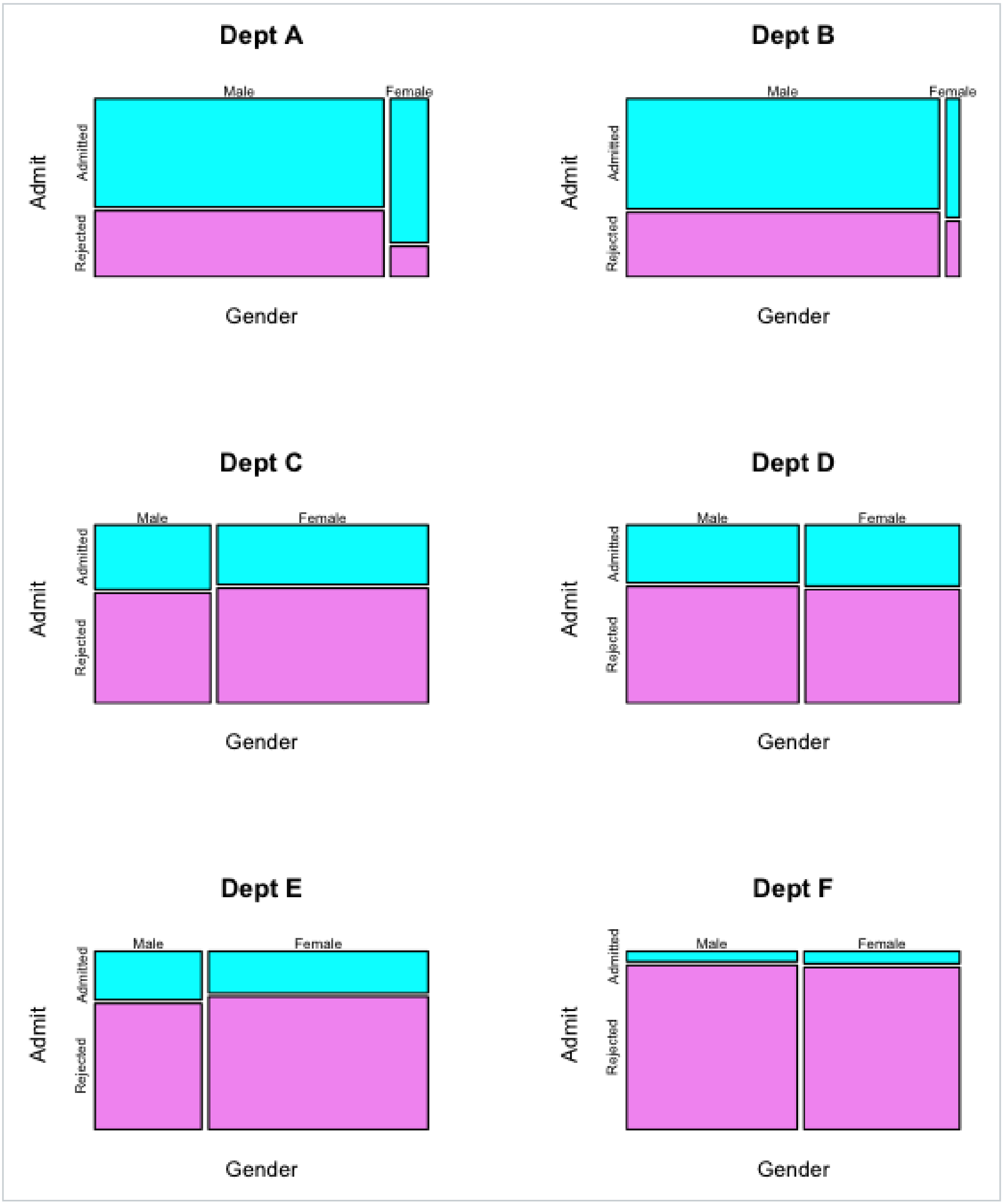
```
> for (i in 1:6)
+   mosaicplot(UCBAdmissions[,,i],
+      col = c("green", "red"),
+      main=paste("Dept",LETTERS[i],sep=" "))
```



The mosaic plots of the transpose of the above table provide a different picture of the data.



```
> for (i in 1:6)
+   mosaicplot(t(UCBAdmissions[,,i]),
+       col =  c("cyan", "violet"),
+       main=paste("Dept",LETTERS[i],sep=" "))
```



Test Yourself 3.26 (Multipart)

Load the titanic data set using data("Titanic"). The following questions will use this data set. (Click each step to reveal its solution.)

▶ **What are the dimensions of the summarized Titanic table?**

```
dim(Titanic)
4 x 2 x 2 x 2
```

▶ **What are the names of the dimensions of the summarized Titanic table?**

```
dimnames(Titanic)
Class, Sex, Age, Survived
```

▶ **Produce margin tables for the Class and Age variables separately, and then a joint margin table of both variables. Show the R commands.**

```
margin.table(Titanic, 1)
margin.table(Titanic, 3)
margin.table(Titanic, c(1, 3))
```

▶ **Produce a mosaic plot of the Titanic Class and Surivived variables. Show the R commands.**

```
mosaicplot(margin.table(Titanic, c(1, 4)))
```

▶ **Produce barplots showing the number of passengers that survived broken down by class. Include a legend and label the axes. Try to produce all four plots in one window. Use the par(mfrow) function to do so. Show the R commands.**

```
par(mfrow=c(2, 2))
class <- rownames(Titanic)
for(i in 1:4) {
        barplot(Titanic[i,,,2], legend.text=TRUE, main=paste("Class", class[i
}
```