```
summary(data)
       x                    y
 Min.   :-1.90483   Min.   :-2.16545
 1st Qu.:-0.66321   1st Qu.:-0.71451
 Median : 0.09367   Median :-0.03797
 Mean   : 0.02522   Mean   :-0.02153
 3rd Qu.: 0.65414   3rd Qu.: 0.55738
 Max.   : 2.18471   Max.   : 1.70199
```

A useful way to detect patterns and anomalies in the data is through the exploratory data analysis with visualization. Visualization gives a succinct, holistic view of the data that may be difficult to grasp from the numbers and summaries alone. Variables *x* and *y* of the data frame `data` can instead be visualized in a scatterplot (Figure 3-5), which easily depicts the relationship between two variables. An important facet of the initial data exploration, visualization assesses data cleanliness and suggests potentially important relationships in the data prior to the model planning and building phases.

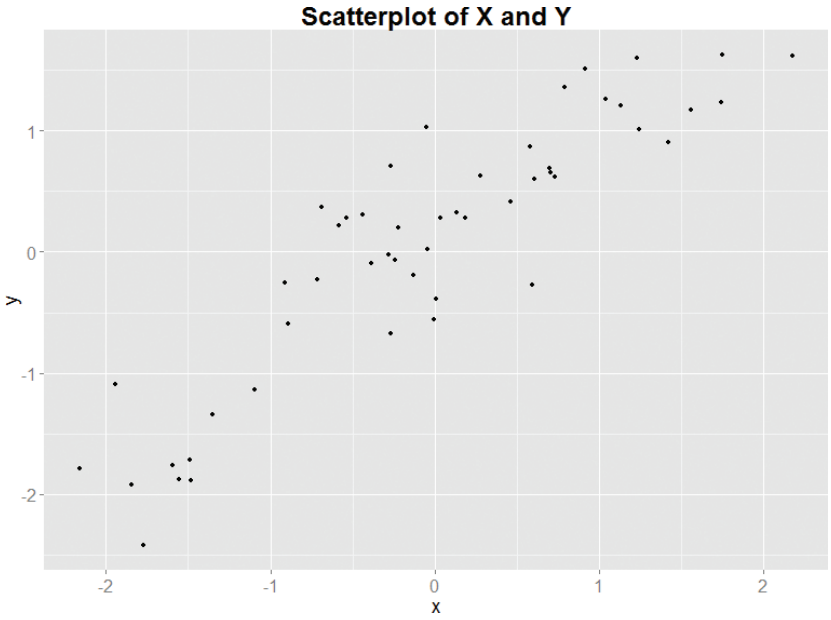

FIGURE 3-5 *A scatterplot can easily show if x and y share a relation*

The code to generate `data` as well as Figure 3-5 is shown next.

```
x <- rnorm(50)
y <- x + rnorm(50, mean=0, sd=0.5)

data <- as.data.frame(cbind(x, y))
```

```
summary(data)

library(ggplot2)
ggplot(data, aes(x=x, y=y)) +
  geom_point(size=2) +
  ggtitle("Scatterplot of X and Y") +
  theme(axis.text=element_text(size=12),
        axis.title = element_text(size=14),
        plot.title = element_text(size=20, face="bold"))
```

Exploratory data analysis [9] is a data analysis approach to reveal the important characteristics of a dataset, mainly through visualization. This section discusses how to use some basic visualization techniques and the plotting feature in R to perform exploratory data analysis.

3.2.1 Visualization Before Analysis

To illustrate the importance of visualizing data, consider Anscombe's quartet. Anscombe's quartet consists of four datasets, as shown in Figure 3-6. It was constructed by statistician Francis Anscombe [10] in 1973 to demonstrate the importance of graphs in statistical analyses.

# 1		# 2		# 3		# 4	
x	y	x	y	x	y	x	y
4	4.26	4	3.10	4	5.39	8	5.25
5	5.68	5	4.74	5	5.73	8	5.56
6	7.24	6	6.13	6	6.08	8	5.76
7	4.82	7	7.26	7	6.42	8	6.58
8	6.95	8	8.14	8	6.77	8	6.89
9	8.81	9	8.77	9	7.11	8	7.04
10	8.04	10	9.14	10	7.46	8	7.71
11	8.33	11	9.26	11	7.81	8	7.91
12	10.84	12	9.13	12	8.15	8	8.47
13	7.58	13	8.74	13	12.74	8	8.84
14	9.96	14	8.10	14	8.84	19	12.50

FIGURE 3-6 *Anscombe's quartet*

The four datasets in Anscombe's quartet have nearly identical statistical properties, as shown in Table 3-3.

TABLE 3-3 *Statistical Properties of Anscombe's Quartet*

Statistical Property	Value
Mean of *x*	9
Variance of *y*	11
Mean of *y*	7.50 (to 2 decimal points)

Variance of *y*	4.12 or 4.13 (to 2 decimal points)
Correlations between *x* and *y*	0.816
Linear regression line	$y = 3.00 + 0.50x$ (to 2 decimal points)

Based on the nearly identical statistical properties across each dataset, one might conclude that these four datasets are quite similar. However, the scatterplots in Figure 3-7 tell a different story. Each dataset is plotted as a scatterplot, and the fitted lines are the result of applying linear regression models. The estimated regression line fits Dataset 1 reasonably well. Dataset 2 is definitely nonlinear. Dataset 3 exhibits a linear trend, with one apparent outlier at $x = 13$. For Dataset 4, the regression line fits the dataset quite well. However, with only points at two x values, it is not possible to determine that the linearity assumption is proper.

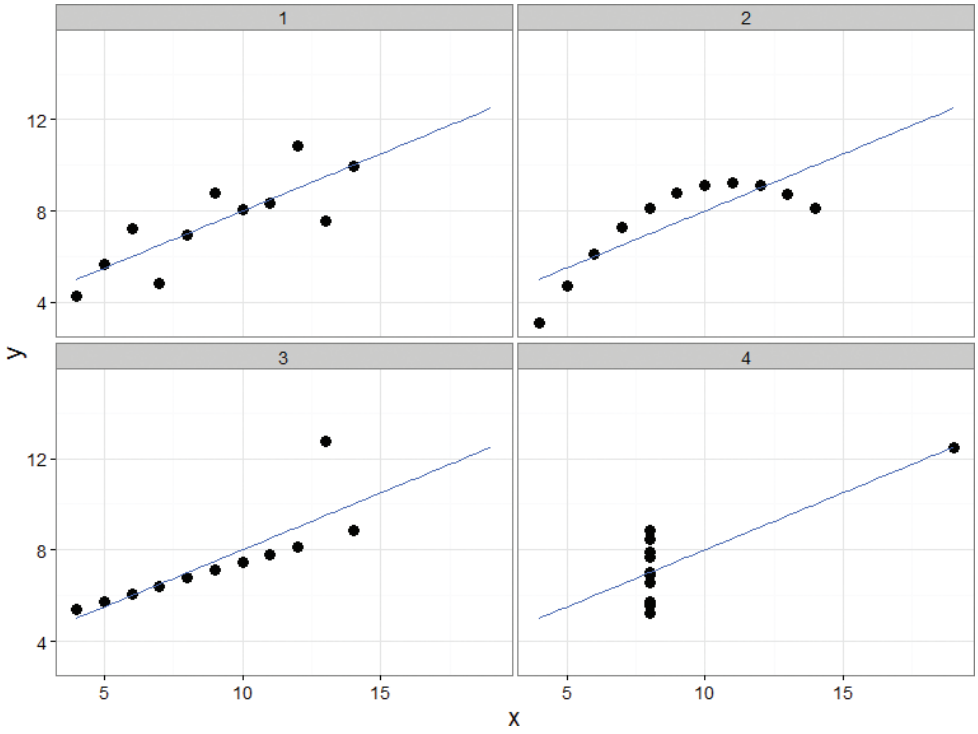

FIGURE 3-7 *Anscombe's quartet visualized as scatterplots*

The R code for generating Figure 3-7 is shown next. It requires the R package `ggplot2` [11], which can be installed simply by running the command `install.packages("ggplot2")`. The `anscombe`

dataset for the plot is included in the standard R distribution. Enter `data()` for a list of datasets included in the R base distribution. Enter `data(`***DatasetName***`)` to make a dataset available in the current workspace.

In the code that follows, variable *levels* is created using the `gl()` function, which generates factors of four levels (1, 2, 3, and 4), each repeating 11 times. Variable *mydata* is created using the `with(`***data***`, ***expression***`) function, which evaluates an ***expression*** in an environment constructed from ***data***. In this example, the ***data*** is the `anscombe` dataset, which includes eight attributes: *x1*, *x2*, *x3*, *x4*, *y1*, *y2*, *y3*, and *y4*. The ***expression*** part in the code creates a data frame from the `anscombe` dataset, and it only includes three attributes: *x*, *y*, and the group each data point belongs to (*mygroup*).

```
install.packages("ggplot2") # not required if package has been installed

data(anscombe)    # load the anscombe dataset into the current workspace
anscombe
   x1 x2 x3 x4    y1   y2    y3    y4
1  10 10 10  8  8.04 9.14  7.46  6.58
2   8  8  8  8  6.95 8.14  6.77  5.76
3  13 13 13  8  7.58 8.74 12.74  7.71
4   9  9  9  8  8.81 8.77  7.11  8.84
5  11 11 11  8  8.33 9.26  7.81  8.47
6  14 14 14  8  9.96 8.10  8.84  7.04
7   6  6  6  8  7.24 6.13  6.08  5.25
8   4  4  4 19  4.26 3.10  5.39 12.50
9  12 12 12  8 10.84 9.13  8.15  5.56
10  7  7  7  8  4.82 7.26  6.42  7.91
11  5  5  5  8  5.68 4.74  5.73  6.89

nrow(anscombe)    # number of rows
[1] 11

# generates levels to indicate which group each data point belongs to
levels <- gl(4, nrow(anscombe))
levels
[1] 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3
[34] 4 4 4 4 4 4 4 4 4 4 4
Levels: 1 2 3 4

# Group anscombe into a data frame
mydata <- with(anscombe, data.frame(x=c(x1,x2,x3,x4), y=c(y1,y2,y3,y4),
               mygroup=levels))

mydata
    x     y mygroup
1  10  8.04       1
2   8  6.95       1
3  13  7.58       1
4   9  8.81       1
...
```

```
41 19 12.50       4
42  8  5.56       4
43  8  7.91       4
44  8  6.89       4

# Make scatterplots using the ggplot2 package
library(ggplot2)
theme_set(theme_bw())    # set plot color theme

# create the four plots of Figure 3-7
ggplot(mydata, aes(x,y)) +
      geom_point(size=4) +
      geom_smooth(method="lm", fill=NA, fullrange=TRUE) +
      facet_wrap(~mygroup)
```

3.2.2 Dirty Data

This section addresses how dirty data can be detected in the data exploration phase with visualizations. In general, analysts should look for anomalies, verify the data with domain knowledge, and decide the most appropriate approach to clean the data.

Consider a scenario in which a bank is conducting data analyses of its account holders to gauge customer retention. Figure 3-8 shows the age distribution of the account holders.

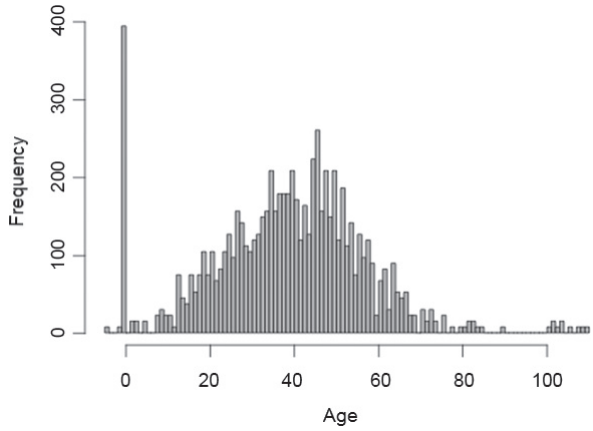

FIGURE 3-8 *Age distribution of bank account holders*

If the age data is in a vector called `age`, the graph can be created with the following R script:

```
hist(age, breaks=100, main="Age Distribution of Account Holders",
     xlab="Age", ylab="Frequency", col="gray")
```

The figure shows that the median age of the account holders is around 40. A few accounts with account holder age less than 10 are unusual but plausible. These could be custodial accounts or college savings accounts set up by the parents of young children. These accounts should be retained for future analyses.

However, the left side of the graph shows a huge spike of customers who are zero years old or have negative ages. This is likely to be evidence of ***missing data***. One possible explanation is that the null age values could have been replaced by 0 or negative values during the data input. Such an occurrence may be caused by entering age in a text box that only allows numbers and does not accept empty values. Or it might be caused by transferring data among several systems that have different definitions for null values (such as NULL, NA, 0, –1, or –2). Therefore, ***data cleansing*** needs to be performed over the accounts with abnormal age values. Analysts should take a closer look at the records to decide if the missing data should be eliminated or if an appropriate age value can be determined using other available information for each of the accounts.

In R, the `is.na()` function provides tests for missing values. The following example creates a vector *x* where the fourth value is not available (`NA`). The `is.na()` function returns `TRUE` at each `NA` value and `FALSE` otherwise.

```
x <- c(1, 2, 3, NA, 4)
is.na(x)
[1] FALSE FALSE FALSE TRUE FALSE
```

Some arithmetic functions, such as `mean()`, applied to data containing missing values can yield an `NA` result. To prevent this, set the `na.rm` parameter to `TRUE` to remove the missing value during the function's execution.

```
mean(x)
[1] NA
mean(x, na.rm=TRUE)
[1] 2.5
```

The `na.exclude()` function returns the object with incomplete cases removed.

```
DF <- data.frame(x = c(1, 2, 3), y = c(10, 20, NA))
DF
  x  y
1 1 10
2 2 20
3 3 NA

DF1 <- na.exclude(DF)
DF1
  x  y
1 1 10
2 2 20
```

Account holders older than 100 may be due to bad data caused by typos. Another possibility is that these accounts may have been passed down to the heirs of the original account holders without being updated. In this case, one needs to further examine the data and conduct data cleansing if necessary. The dirty data could be simply removed or filtered out with an age threshold for future analyses. If removing records is not an option, the analysts can look for patterns within the data and develop a set of heuristics to attack the problem of dirty data. For example, wrong age values could be replaced with ***approximation*** based on the nearest neighbor—the record that is the most similar to the record in question based on analyzing the differences in all the other variables besides age.

Figure 3-9 presents another example of dirty data. The distribution shown here corresponds to the age of mortgages in a bank's home loan portfolio. The mortgage age is calculated by subtracting the origination date of the loan from the current date. The vertical axis corresponds to the number of mortgages at each mortgage age.

FIGURE 3-9 *Distribution of mortgage in years since origination from a bank's home loan portfolio*

If the data is in a vector called `mortgage`, Figure 3-9 can be produced by the following R script.

```
hist(mortgage, breaks=10, xlab="Mortgage Age", col="gray",
     main="Portfolio Distribution, Years Since Origination")
```

Figure 3-9 shows that the loans are no more than 10 years old, and these 10-year-old loans have a disproportionate frequency compared to the rest of the population. One possible explanation is that the 10-year-old loans do not only include loans originated 10 years ago, but also those originated earlier than that. In other words, the 10 in the *x*-axis actually means ≥ 10. This sometimes happens when data is ported from one system to another or because the data provider decided, for some reason, not to distinguish loans that are more than 10 years old. Analysts need to study the data further and decide the most appropriate way to perform data cleansing.

Data analysts should perform sanity checks against domain knowledge and decide if the dirty data needs to be eliminated. Consider the task to find out the probability of mortgage loan default. If the past observations suggest that most defaults occur before about the 4th year and 10-year-old mortgages rarely default, it may be safe to eliminate the dirty data and assume that the defaulted loans are less than 10 years old. For other analyses, it may become necessary to track down the source and find out the true origination dates.

Dirty data can occur due to acts of omission. In the `sales` data used at the beginning of this chapter, it was seen that the minimum number of orders was 1 and the minimum annual sales amount was $30.02. Thus, there is a strong possibility that the provided dataset did not include the sales data on all customers, just the customers who purchased something during the past year.

3.2.3 Visualizing a Single Variable

Using visual representations of data is a hallmark of exploratory data analyses: letting the data speak to its audience rather than imposing an interpretation on the data *a priori*. Sections 3.2.3 and 3.2.4 examine ways of displaying data to help explain the underlying distributions of a single variable or the relationships of two or more variables.

R has many functions available to examine a single variable. Some of these functions are listed in Table 3-4.

TABLE 3-4 *Example Functions for Visualizing a Single Variable*

Function	Purpose
`plot(data)`	Scatterplot where x is the index and y is the value; suitable for low-volume data
`barplot(data)`	Barplot with vertical or horizontal bars
`dotchart(data)`	Cleveland dot plot [12]
`hist(data)`	Histogram
`plot(density(data))`	Density plot (a continuous histogram)
`stem(data)`	Stem-and-leaf plot
`rug(data)`	Add a rug representation (1-d plot) of the data to an existing plot

Dotchart and Barplot

Dotchart and barplot portray continuous values with labels from a discrete variable. A dotchart can be created in R with the function `dotchart(x, label=...)`, where *x* is a numeric vector and *label* is a vector of categorical labels for *x*. A barplot can be created with the `barplot(height)` function, where *height* represents a vector or matrix. Figure 3-10 shows (a) a dotchart and (b) a barplot based on the `mtcars` dataset, which includes the fuel consumption and 10 aspects of automobile design and performance of 32 automobiles. This dataset comes with the standard R distribution.

The plots in Figure 3-10 can be produced with the following R code.

```
data(mtcars)
dotchart(mtcars$mpg,labels=row.names(mtcars),cex=.7,
         main="Miles Per Gallon (MPG) of Car Models",
         xlab="MPG")
barplot(table(mtcars$cyl), main="Distribution of Car Cylinder Counts",
          xlab="Number of Cylinders")
```

Histogram and Density Plot

Figure 3-11(a) includes a histogram of household income. The histogram shows a clear concentration of low household incomes on the left and the long tail of the higher incomes on the right.

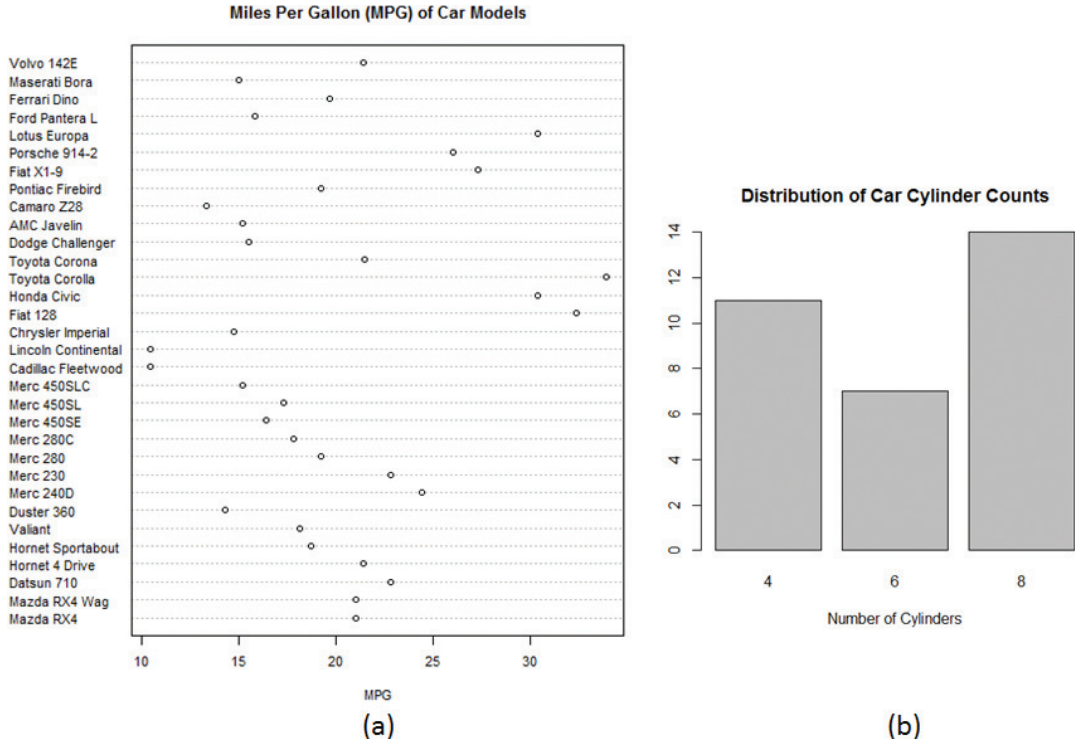

Figure 3-10 *(a) Dotchart on the miles per gallon of cars and (b) Barplot on the distribution of car cylinder counts*

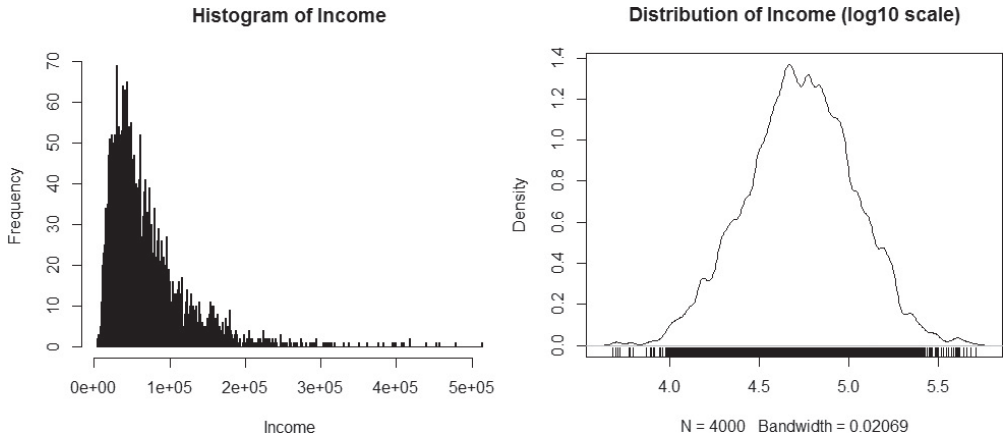

Figure 3-11 *(a) Histogram and (b) Density plot of household income*

Figure 3-11(b) shows a density plot of the logarithm of household income values, which emphasizes the distribution. The income distribution is concentrated in the center portion of the graph. The code to generate the two plots in Figure 3-11 is provided next. The `rug()` function creates a one-dimensional density plot on the bottom of the graph to emphasize the distribution of the observation.

```
# randomly generate 4000 observations from the log normal distribution
income <- rlnorm(4000, meanlog = 4, sdlog = 0.7)
summary(income)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  4.301  33.720  54.970  70.320  88.800 659.800
income <- 1000*income
summary(income)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   4301   33720   54970   70320   88800  659800
# plot the histogram
hist(income, breaks=500, xlab="Income", main="Histogram of Income")
# density plot
plot(density(log10(income), adjust=0.5),
     main="Distribution of Income (log10 scale)")
# add rug to the density plot
rug(log10(income))
```

In the data preparation phase of the Data Analytics Lifecycle, the data range and distribution can be obtained. If the data is skewed, viewing the logarithm of the data (if it's all positive) can help detect structures that might otherwise be overlooked in a graph with a regular, nonlogarithmic scale.

When preparing the data, one should look for signs of dirty data, as explained in the previous section. Examining if the data is unimodal or multimodal will give an idea of how many distinct populations with different behavior patterns might be mixed into the overall population. Many modeling techniques assume that the data follows a normal distribution. Therefore, it is important to know if the available dataset can match that assumption before applying any of those modeling techniques.

Consider a density plot of diamond prices (in USD). Figure 3-12(a) contains two density plots for premium and ideal cuts of diamonds. The group of premium cuts is shown in red, and the group of ideal cuts is shown in blue. The range of diamond prices is wide—in this case ranging from around $300 to almost $20,000. Extreme values are typical of monetary data such as income, customer value, tax liabilities, and bank account sizes.

Figure 3-12(b) shows more detail of the diamond prices than Figure 3-12(a) by taking the logarithm. The two humps in the premium cut represent two distinct groups of diamond prices: One group centers around $\log_{10}$ price = 2.9 (where the price is about $794), and the other centers around $\log_{10}$ price = 3.7 (where the price is about $5,012). The ideal cut contains three humps, centering around 2.9, 3.3, and 3.7 respectively.

The R script to generate the plots in Figure 3-12 is shown next. The *diamonds* dataset comes with the `ggplot2` package.

```
library("ggplot2")
data(diamonds)    # load the diamonds dataset from ggplot2

# Only keep the premium and ideal cuts of diamonds
```

```
niceDiamonds <- diamonds[diamonds$cut=="Premium" |
                         diamonds$cut=="Ideal",]

summary(niceDiamonds$cut)
     Fair      Good Very Good   Premium     Ideal
        0         0         0     13791     21551

# plot density plot of diamond prices
ggplot(niceDiamonds, aes(x=price, fill=cut)) +
  geom_density(alpha = .3, color=NA)

# plot density plot of the log10 of diamond prices
ggplot(niceDiamonds, aes(x=log10(price), fill=cut)) +
  geom_density(alpha = .3, color=NA)
```

As an alternative to `ggplot2`, the `lattice` package provides a function called `densityplot()` for making simple density plots.

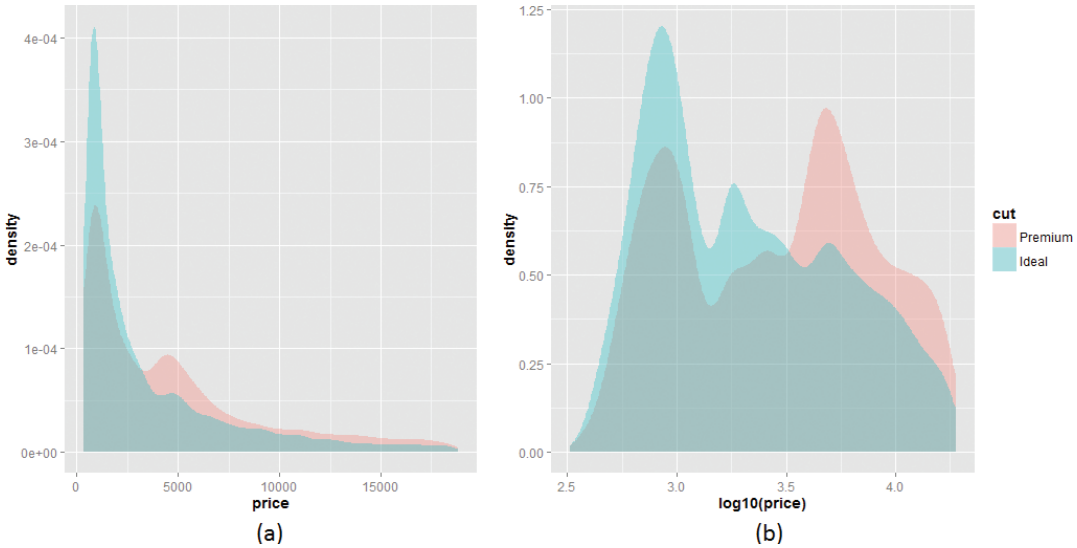

FIGURE 3-12 *Density plots of (a) diamond prices and (b) the logarithm of diamond prices*

3.2.4 Examining Multiple Variables

A scatterplot (shown previously in Figure 3-1 and Figure 3-5) is a simple and widely used visualization for finding the relationship among multiple variables. A scatterplot can represent data with up to five variables using x-axis, y-axis, size, color, and shape. But usually only two to four variables are portrayed in a scatterplot to minimize confusion. When examining a scatterplot, one needs to pay close attention

to the possible relationship between the variables. If the functional relationship between the variables is somewhat pronounced, the data may roughly lie along a straight line, a parabola, or an exponential curve. If variable y is related exponentially to x, then the plot of x versus `log(y)` is approximately linear. If the plot looks more like a cluster without a pattern, the corresponding variables may have a weak relationship.

The scatterplot in Figure 3-13 portrays the relationship of two variables: x and y. The red line shown on the graph is the fitted line from the linear regression. Linear regression will be revisited in Chapter 6, "Advanced Analytical Theory and Methods: Regression." Figure 3-13 shows that the regression line does not fit the data well. This is a case in which linear regression cannot model the relationship between the variables. Alternative methods such as the `loess()` function can be used to fit a nonlinear line to the data. The blue curve shown on the graph represents the LOESS curve, which fits the data better than linear regression.

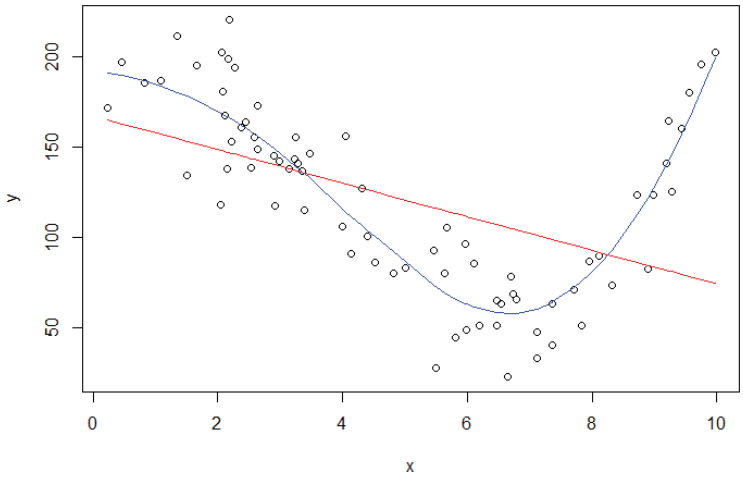

FIGURE 3-13 *Examining two variables with regression*

The R code to produce Figure 3-13 is as follows. The `runif(75,0,10)` generates 75 numbers between 0 to 10 with random deviates, and the numbers conform to the uniform distribution. The `rnorm(75,0,20)` generates 75 numbers that conform to the normal distribution, with the mean equal to 0 and the standard deviation equal to 20. The `points()` function is a generic function that draws a sequence of points at the specified coordinates. Parameter `type="l"` tells the function to draw a solid line. The `col` parameter sets the color of the line, where 2 represents the red color and 4 represents the blue color.

```
# 75 numbers between 0 and 10 of uniform distribution
x <- runif(75, 0, 10)

x <- sort(x)
y <- 200 + x^3 - 10 * x^2 + x + rnorm(75, 0, 20)


lr <- lm(y ~ x)        # linear regression
poly <- loess(y ~ x)   # LOESS
```

```
fit <- predict(poly)  # fit a nonlinear line

plot(x,y)

# draw the fitted line for the linear regression
points(x, lr$coefficients[1] + lr$coefficients[2] * x,
       type = "l", col = 2)

# draw the fitted line with LOESS
points(x, fit, type = "l", col = 4)
```

Dotchart and Barplot

Dotchart and barplot from the previous section can visualize multiple variables. Both of them use color as an additional dimension for visualizing the data.

For the same `mtcars` dataset, Figure 3-14 shows a dotchart that groups vehicle cylinders at the y-axis and uses colors to distinguish different cylinders. The vehicles are sorted according to their MPG values. The code to generate Figure 3-14 is shown next.

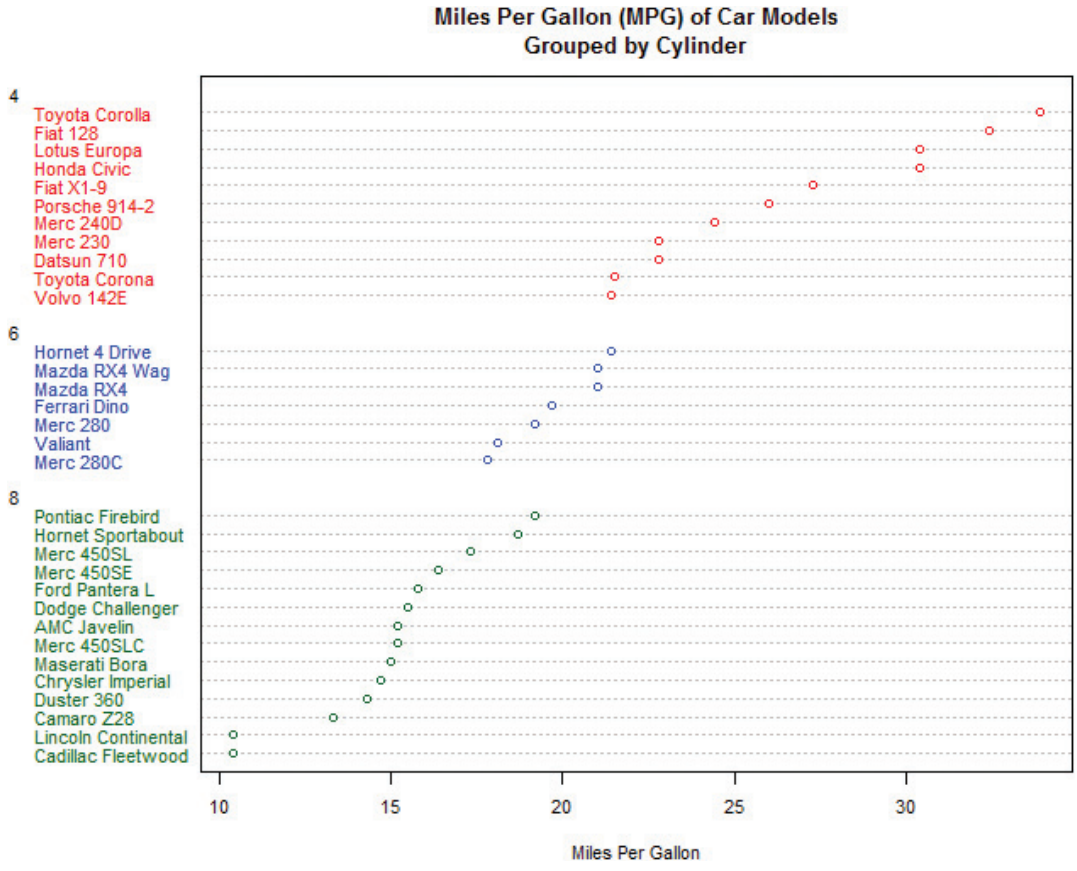

FIGURE 3-14 *Dotplot to visualize multiple variables*

```
# sort by mpg
cars <- mtcars[order(mtcars$mpg),]

# grouping variable must be a factor
cars$cyl <- factor(cars$cyl)

cars$color[cars$cyl==4] <- "red"
cars$color[cars$cyl==6] <- "blue"
cars$color[cars$cyl==8] <- "darkgreen"

dotchart(cars$mpg, labels=row.names(cars), cex=.7, groups= cars$cyl,
       main="Miles Per Gallon (MPG) of Car Models\nGrouped by Cylinder",
       xlab="Miles Per Gallon", color=cars$color, gcolor="black")
```

The barplot in Figure 3-15 visualizes the distribution of car cylinder counts and number of gears. The x-axis represents the number of cylinders, and the color represents the number of gears. The code to generate Figure 3-15 is shown next.

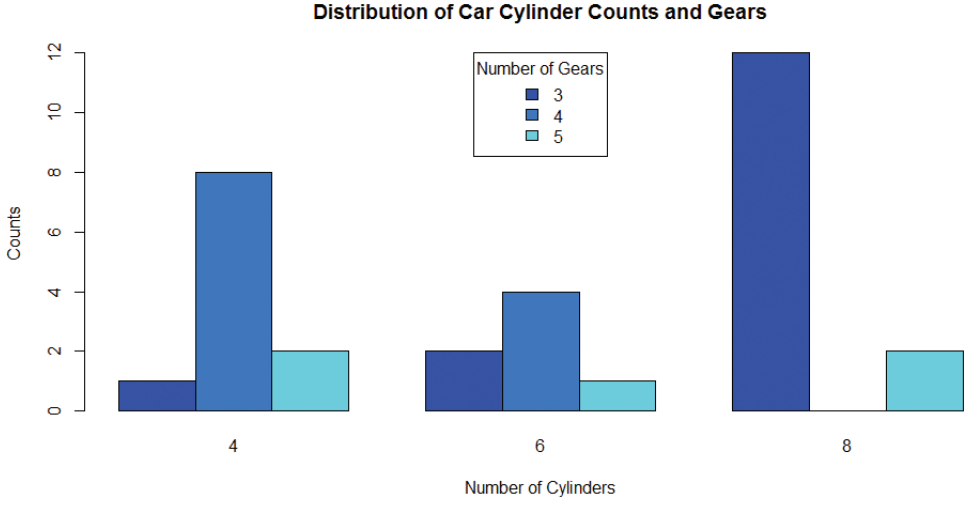

FIGURE 3-15 *Barplot to visualize multiple variables*

```
counts <- table(mtcars$gear, mtcars$cyl)
barplot(counts, main="Distribution of Car Cylinder Counts and Gears",
        xlab="Number of Cylinders", ylab="Counts",
        col=c("#0000FFFF", "#0080FFFF", "#00FFFFFF"),
        legend = rownames(counts), beside=TRUE,
        args.legend = list(x="top", title = "Number of Gears"))
```

Box-and-Whisker Plot

Box-and-whisker plots show the distribution of a continuous variable for each value of a discrete variable. The box-and-whisker plot in Figure 3-16 visualizes mean household incomes as a function of region in the United States. The first digit of the U.S. postal ("ZIP") code corresponds to a geographical region in the United States. In Figure 3-16, each data point corresponds to the mean household income from a particular zip code. The horizontal axis represents the first digit of a zip code, ranging from 0 to 9, where 0 corresponds to the northeast region of the United States (such as Maine, Vermont, and Massachusetts), and 9 corresponds to the southwest region (such as California and Hawaii). The vertical axis represents the logarithm of mean household incomes. The logarithm is taken to better visualize the distribution of the mean household incomes.

FIGURE 3-16 *A box-and-whisker plot of mean household income and geographical region*

In this figure, the scatterplot is displayed beneath the box-and-whisker plot, with some jittering for the overlap points so that each line of points widens into a strip. The "box" of the box-and-whisker shows the range that contains the central 50% of the data, and the line inside the box is the location of the median value. The upper and lower hinges of the boxes correspond to the first and third quartiles of the data. The upper whisker extends from the hinge to the highest value that is within 1.5 * IQR of the hinge. The lower whisker extends from the hinge to the lowest value within 1.5 * IQR of the hinge. IQR is the inter-quartile range, as discussed in Section 3.1.4. The points outside the whiskers can be considered possible outliers.

The graph shows how household income varies by region. The highest median incomes are in region 0 and region 9. Region 0 is slightly higher, but the boxes for the two regions overlap enough that the difference between the two regions probably is not significant. The lowest household incomes tend to be in region 7, which includes states such as Louisiana, Arkansas, and Oklahoma.

Assuming a data frame called *DF* contains two columns (*MeanHouseholdIncome* and *Zip1*), the following R script uses the `ggplot2` library [11] to plot a graph that is similar to Figure 3-16.

```
library(ggplot2)
# plot the jittered scatterplot w/ boxplot
# color-code points with zip codes
# the outlier.size=0 prevents the boxplot from plotting the outlier
ggplot(data=DF, aes(x=as.factor(Zip1), y=log10(MeanHouseholdIncome))) +
  geom_point(aes(color=factor(Zip1)), alpha=0.2, position="jitter") +
  geom_boxplot(outlier.size=0, alpha=0.1) +
  guides(colour=FALSE) +
  ggtitle ("Mean Household Income by Zip Code")
```

Alternatively, one can create a simple box-and-whisker plot with the `boxplot()` function provided by the R base package.

Hexbinplot for Large Datasets

This chapter has shown that scatterplot as a popular visualization can visualize data containing one or more variables. But one should be careful about using it on high-volume data. If there is too much data, the structure of the data may become difficult to see in a scatterplot. Consider a case to compare the logarithm of household income against the years of education, as shown in Figure 3-17. The cluster in the scatterplot on the left (a) suggests a somewhat linear relationship of the two variables. However, one cannot really see the structure of how the data is distributed inside the cluster. This is a Big Data type of problem. Millions or billions of data points would require different approaches for exploration, visualization, and analysis.

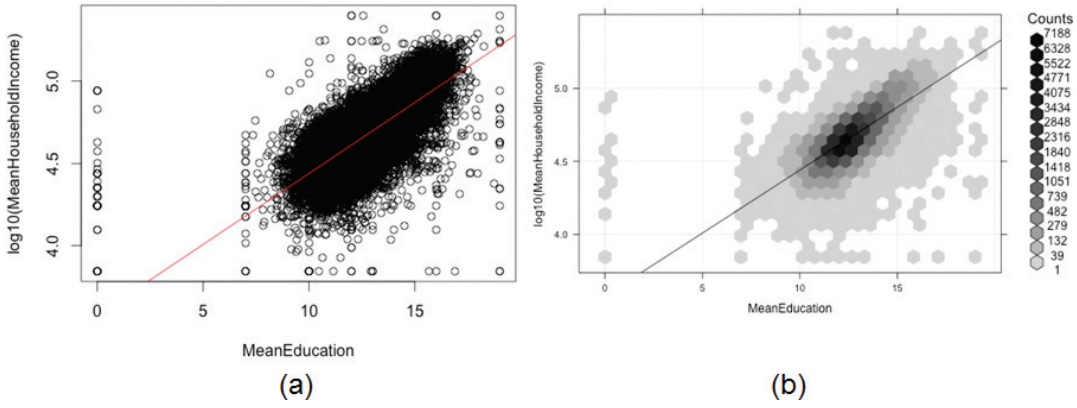

Figure 3-17 *(a) Scatterplot and (b) Hexbinplot of household income against years of education*

Although color and transparency can be used in a scatterplot to address this issue, a hexbinplot is sometimes a better alternative. A hexbinplot combines the ideas of scatterplot and histogram. Similar to a scatterplot, a hexbinplot visualizes data in the *x*-axis and *y*-axis. Data is placed into hexbins, and the third dimension uses shading to represent the concentration of data in each hexbin.

In Figure 3-17(b), the same data is plotted using a hexbinplot. The hexbinplot shows that the data is more densely clustered in a streak that runs through the center of the cluster, roughly along the regression line. The biggest concentration is around 12 years of education, extending to about 15 years.

In Figure 3-17, note the outlier data at `MeanEducation=0`. These data points may correspond to some missing data that needs further cleansing.

Assuming the two variables *`MeanHouseholdIncome`* and *`MeanEducation`* are from a data frame named *`zcta`*, the scatterplot of Figure 3-17(a) is plotted by the following R code.

```
# plot the data points
plot(log10(MeanHouseholdIncome) ~ MeanEducation, data=zcta)
# add a straight fitted line of the linear regression
abline(lm(log10(MeanHouseholdIncome) ~ MeanEducation, data=zcta), col='red')
```

Using the *`zcta`* data frame, the hexbinplot of Figure 3-17(b) is plotted by the following R code. Running the code requires the use of the `hexbin` package, which can be installed by running `install.packages("hexbin")`.

```
library(hexbin)
# "g" adds the grid, "r" adds the regression line
# sqrt transform on the count gives more dynamic range to the shading
# inv provides the inverse transformation function of trans
hexbinplot(log10(MeanHouseholdIncome) ~ MeanEducation,
  data=zcta, trans = sqrt, inv = function(x) x^2, type=c("g", "r"))
```

Scatterplot Matrix

A scatterplot matrix shows many scatterplots in a compact, side-by-side fashion. The scatterplot matrix, therefore, can visually represent multiple attributes of a dataset to explore their relationships, magnify differences, and disclose hidden patterns.

Fisher's *`iris`* dataset [13] includes the measurements in centimeters of the sepal length, sepal width, petal length, and petal width for 50 flowers from three species of iris. The three species are *setosa*, *versicolor*, and *virginica*. The iris dataset comes with the standard R distribution.

In Figure 3-18, all the variables of Fisher's iris dataset (sepal length, sepal width, petal length, and petal width) are compared in a scatterplot matrix. The three different colors represent three species of iris flowers. The scatterplot matrix in Figure 3-18 allows its viewers to compare the differences across the iris species for any pairs of attributes.

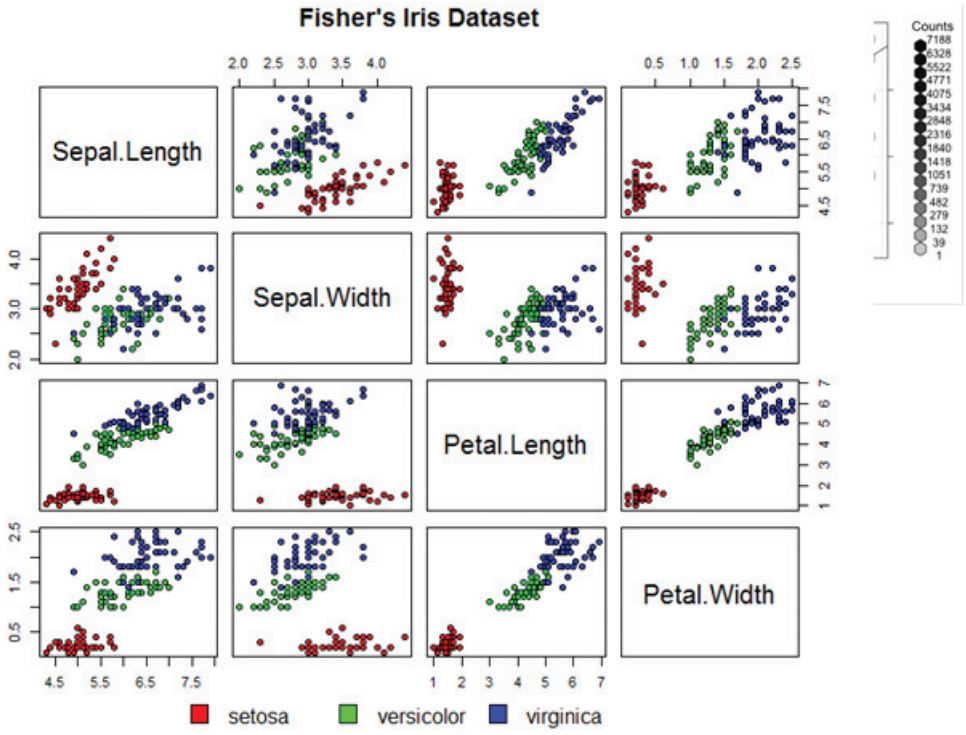

FIGURE 3-18 *Scatterplot matrix of Fisher's [13] iris dataset*

Consider the scatterplot from the first row and third column of Figure 3-18, where sepal length is compared against petal length. The horizontal axis is the petal length, and the vertical axis is the sepal length. The scatterplot shows that *versicolor* and *virginica* share similar sepal and petal lengths, although the latter has longer petals. The petal lengths of all *setosa* are about the same, and the petal lengths are remarkably shorter than the other two species. The scatterplot shows that for *versicolor* and *virginica*, sepal length grows linearly with the petal length.

The R code for generating the scatterplot matrix is provided next.

```
# define the colors
colors <- c("red", "green", "blue")

# draw the plot matrix
pairs(iris[1:4], main = "Fisher's Iris Dataset",
      pch = 21, bg = colors[unclass(iris$Species)] )

# set graphical parameter to clip plotting to the figure region
par(xpd = TRUE)

# add legend
legend(0.2, 0.02, horiz = TRUE, as.vector(unique(iris$Species)),
       fill = colors, bty = "n")
```

The vector *colors* defines the color scheme for the plot. It could be changed to something like `colors <- c("gray50", "white", "black")` to make the scatterplots grayscale.

Analyzing a Variable over Time

Visualizing a variable over time is the same as visualizing any pair of variables, but in this case the goal is to identify time-specific patterns.

Figure 3-19 plots the monthly total numbers of international airline passengers (in thousands) from January 1940 to December 1960. Enter `plot(AirPassengers)` in the R console to obtain a similar graph. The plot shows that, for each year, a large peak occurs mid-year around July and August, and a small peak happens around the end of the year, possibly due to the holidays. Such a phenomenon is referred to as a ***seasonality effect***.

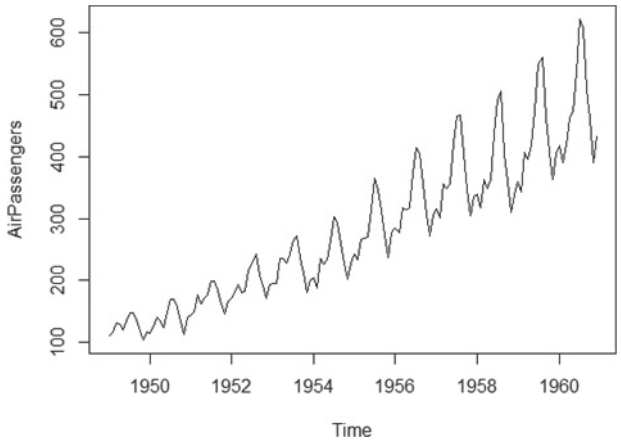

FIGURE 3-19 *Airline passenger counts from 1949 to 1960*

Additionally, the overall trend is that the number of air passengers steadily increased from 1949 to 1960. Chapter 8, "Advanced Analytical Theory and Methods: Time Series Analysis," discusses the analysis of such datasets in greater detail.

3.2.5 Data Exploration Versus Presentation

Using visualization for data exploration is different from presenting results to stakeholders. Not every type of plot is suitable for all audiences. Most of the plots presented earlier try to detail the data as clearly as possible for data scientists to identify structures and relationships. These graphs are more technical in nature and are better suited to technical audiences such as data scientists. Nontechnical stakeholders, however, generally prefer simple, clear graphics that focus on the message rather than the data.

Figure 3-20 shows the density plot on the distribution of account values from a bank. The data has been converted to the $\log_{10}$ scale. The plot includes a rug on the bottom to show the distribution of the variable. This graph is more suitable for data scientists and business analysts because it provides information that

can be relevant to the downstream analysis. The graph shows that the transformed account values follow an approximate normal distribution, in the range from $100 to $10,000,000. The median account value is approximately $30,000 ($10^{4.5}$), with the majority of the accounts between $1,000 ($10^3$) and $1,000,000 ($10^6$).

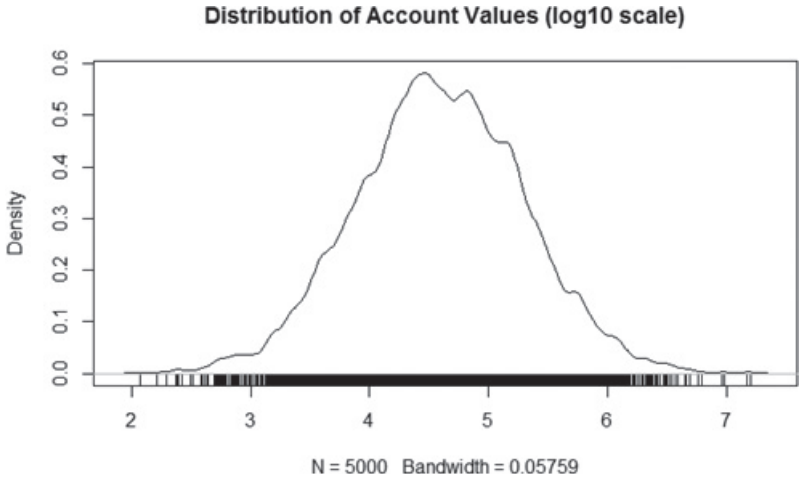

FIGURE 3-20 *Density plots are better to show to data scientists*

Density plots are fairly technical, and they contain so much information that they would be difficult to explain to less technical stakeholders. For example, it would be challenging to explain why the account values are in the $\log_{10}$ scale, and such information is not relevant to stakeholders. The same message can be conveyed by partitioning the data into log-like bins and presenting it as a histogram. As can be seen in Figure 3-21, the bulk of the accounts are in the $1,000–1,000,000 range, with the peak concentration in the $10–50K range, extending to $500K. This portrayal gives the stakeholders a better sense of the customer base than the density plot shown in Figure 3-20.

Note that the bin sizes should be carefully chosen to avoid distortion of the data. In this example, the bins in Figure 3-21 are chosen based on observations from the density plot in Figure 3-20. Without the density plot, the peak concentration might be just due to the somewhat arbitrary appearing choices for the bin sizes.

This simple example addresses the different needs of two groups of audience: analysts and stakeholders. Chapter 12, "The Endgame, or Putting It All Together," further discusses the best practices of delivering presentations to these two groups.

Following is the R code to generate the plots in Figure 3-20 and Figure 3-21.

```
# Generate random log normal income data
income = rlnorm(5000, meanlog=log(40000), sdlog=log(5))

# Part I: Create the density plot
plot(density(log10(income), adjust=0.5),
     main="Distribution of Account Values (log10 scale)")
# Add rug to the density plot
```

```
rug(log10(income))

# Part II: Make the histogram
# Create "log-like bins"
breaks = c(0, 1000, 5000, 10000, 50000, 100000, 5e5, 1e6, 2e7)
# Create bins and label the data
bins = cut(income, breaks, include.lowest=T,
           labels = c("< 1K", "1-5K", "5-10K", "10-50K",
                      "50-100K", "100-500K", "500K-1M", "> 1M"))
# Plot the bins
plot(bins, main = "Distribution of Account Values",
     xlab = "Account value ($ USD)",
     ylab = "Number of Accounts", col="blue")
```

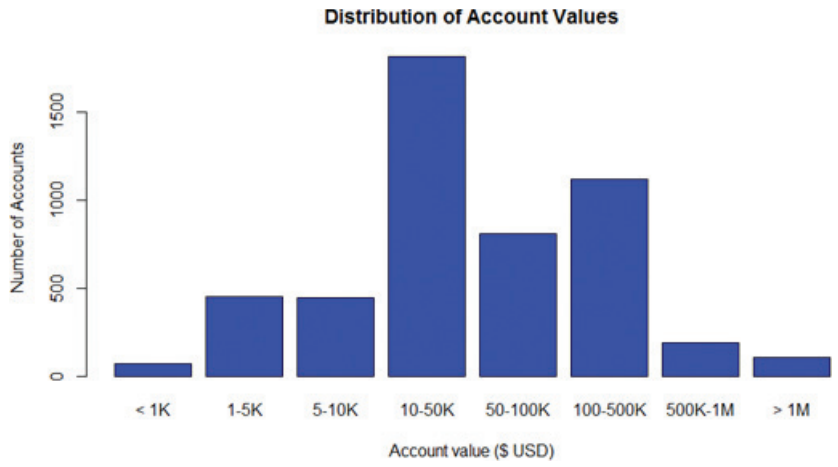

FIGURE 3-21 *Histograms are better to show to stakeholders*

3.3 Statistical Methods for Evaluation

Visualization is useful for data exploration and presentation, but statistics is crucial because it may exist throughout the entire Data Analytics Lifecycle. Statistical techniques are used during the initial data exploration and data preparation, model building, evaluation of the final models, and assessment of how the new models improve the situation when deployed in the field. In particular, statistics can help answer the following questions for data analytics:

- Model Building and Planning
  - What are the best input variables for the model?
  - Can the model predict the outcome given the input?

- Model Evaluation
  - Is the model accurate?
  - Does the model perform better than an obvious guess?
  - Does the model perform better than another candidate model?
- Model Deployment
  - Is the prediction sound?
  - Does the model have the desired effect (such as reducing the cost)?

This section discusses some useful statistical tools that may answer these questions.

3.3.1 Hypothesis Testing

When comparing populations, such as testing or evaluating the difference of the means from two samples of data (Figure 3-22), a common technique to assess the difference or the significance of the difference is ***hypothesis testing***.

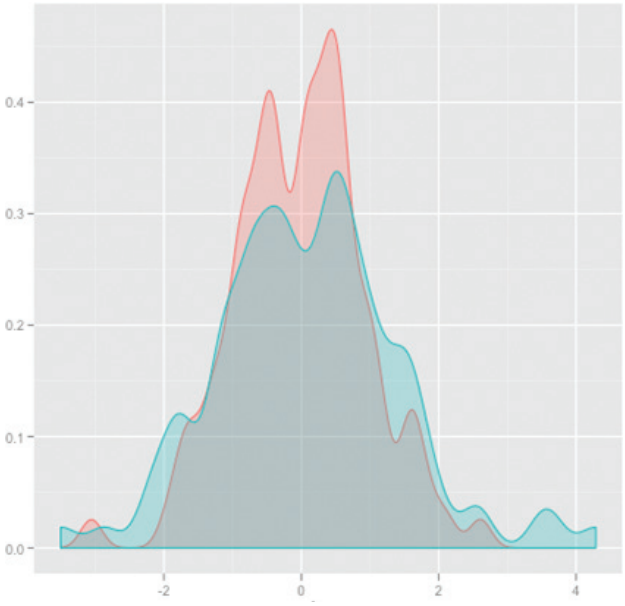

FIGURE 3-22 *Distributions of two samples of data*

The basic concept of hypothesis testing is to form an assertion and test it with data. When performing hypothesis tests, the common assumption is that there is no difference between two samples. This assumption is used as the default position for building the test or conducting a scientific experiment. Statisticians refer to this as the ***null hypothesis*** (H_0). The ***alternative hypothesis*** (H_A) is that there is a

difference between two samples. For example, if the task is to identify the effect of drug A compared to drug B on patients, the null hypothesis and alternative hypothesis would be this.

- H_0: Drug A and drug B have the same effect on patients.
- H_A: Drug A has a greater effect than drug B on patients.

If the task is to identify whether advertising Campaign C is effective on reducing customer churn, the null hypothesis and alternative hypothesis would be as follows.

- H_0: Campaign C does not reduce customer churn better than the current campaign method.
- H_A: Campaign C does reduce customer churn better than the current campaign.

It is important to state the null hypothesis and alternative hypothesis, because misstating them is likely to undermine the subsequent steps of the hypothesis testing process. A hypothesis test leads to either rejecting the null hypothesis in favor of the alternative or not rejecting the null hypothesis.

Table 3-5 includes some examples of null and alternative hypotheses that should be answered during the analytic lifecycle.

Table 3-5 *Example Null Hypotheses and Alternative Hypotheses*

Application	Null Hypothesis	Alternative Hypothesis
Accuracy Forecast	Model X *does not predict* better than the existing model.	Model X *predicts* better than the existing model.
Recommendation Engine	Algorithm Y *does not produce* better recommendations than the current algorithm being used.	Algorithm Y *produces* better recommendations than the current algorithm being used.
Regression Modeling	This variable *does not affect* the outcome because its coefficient is *zero*.	This variable *affects* outcome because its coefficient is not *zero*.

Once a model is built over the training data, it needs to be evaluated over the testing data to see if the proposed model predicts better than the existing model currently being used. The null hypothesis is that the proposed model does not predict better than the existing model. The alternative hypothesis is that the proposed model indeed predicts better than the existing model. In accuracy forecast, the null model could be that the sales of the next month are the same as the prior month. The hypothesis test needs to evaluate if the proposed model provides a better prediction. Take a recommendation engine as an example. The null hypothesis could be that the new algorithm does not produce better recommendations than the current algorithm being deployed. The alternative hypothesis is that the new algorithm produces better recommendations than the old algorithm.

When evaluating a model, sometimes it needs to be determined if a given input variable improves the model. In regression analysis (Chapter 6), for example, this is the same as asking if the regression coefficient for a variable is zero. The null hypothesis is that the coefficient is zero, which means the variable does not have an impact on the outcome. The alternative hypothesis is that the coefficient is nonzero, which means the variable does have an impact on the outcome.

A common hypothesis test is to compare the means of two populations. Two such hypothesis tests are discussed in Section 3.3.2.

3.3.2 Difference of Means

Hypothesis testing is a common approach to draw inferences on whether or not the two populations, denoted *pop*1 and *pop*2, are different from each other. This section provides two hypothesis tests to compare the means of the respective populations based on samples randomly drawn from each population. Specifically, the two hypothesis tests in this section consider the following null and alternative hypotheses.

- H_0: $\mu_1 = \mu_2$
- H_A: $\mu_1 \neq \mu_2$

The μ_1 and μ_2 denote the population means of *pop*1 and *pop*2, respectively.

The basic testing approach is to compare the observed sample means, $\bar{X}_1$ and $\bar{X}_2$, corresponding to each population. If the values of $\bar{X}_1$ and $\bar{X}_2$ are approximately equal to each other, the distributions of $\bar{X}_1$ and $\bar{X}_2$ overlap substantially (Figure 3-23), and the null hypothesis is supported. A large observed difference between the sample means indicates that the null hypothesis should be rejected. Formally, the difference in means can be tested using Student's *t*-test or the Welch's *t*-test.

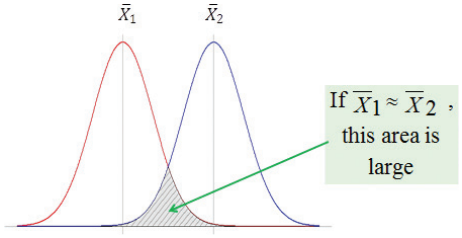

FIGURE 3-23 *Overlap of the two distributions is large if* $\bar{X}_1 \approx \bar{X}_2$

Student's t-test

Student's *t*-test assumes that distributions of the two populations have equal but unknown variances. Suppose n_1 and n_2 samples are randomly and independently selected from two populations, *pop*1 and *pop*2, respectively. If each population is normally distributed with the same mean ($\mu_1 = \mu_2$) and with the same variance, then *T* (the ***t-statistic***), given in Equation 3-1, follows a ***t-distribution*** with $n_1 + n_2 - 2$ ***degrees of freedom (df).***

$$T = \frac{\bar{X}_1 - \bar{X}_2}{S_p \sqrt{\frac{1}{n_1} + \frac{1}{n_2}}}$$

where

$$S_p^2 = \frac{(n_1 - 1)S_1^2 + (n_2 - 1)S_2^2}{n_1 + n_2 - 2} \qquad (3\text{-}1)$$

The shape of the *t*-distribution is similar to the normal distribution. In fact, as the degrees of freedom approaches 30 or more, the *t*-distribution is nearly identical to the normal distribution. Because the numerator of *T* is the difference of the sample means, if the observed value of *T* is far enough from zero such that the probability of observing such a value of *T* is unlikely, one would reject the null hypothesis that the population means are equal. Thus, for a small probability, say $\alpha = 0.05$, T^* is determined such that $P(|T| \geq T^*) = 0.05$. After the samples are collected and the observed value of *T* is calculated according to Equation 3-1, the null hypothesis ($\mu_1 = \mu_2$) is rejected if $|T| \geq T^*$.

In hypothesis testing, in general, the small probability, α, is known as the ***significance level*** of the test. The significance level of the test is the probability of rejecting the null hypothesis, when the null hypothesis is actually `TRUE`. In other words, for $\alpha = 0.05$, if the means from the two populations are truly equal, then in repeated random sampling, the observed magnitude of *T* would only exceed T^* 5% of the time.

In the following R code example, 10 observations are randomly selected from two normally distributed populations and assigned to the variables x and y. The two populations have a mean of 100 and 105, respectively, and a standard deviation equal to 5. Student's *t*-test is then conducted to determine if the obtained random samples support the rejection of the null hypothesis.

```
# generate random observations from the two populations
x <- rnorm(10, mean=100, sd=5)     # normal distribution centered at 100
y <- rnorm(20, mean=105, sd=5)     # normal distribution centered at 105

t.test(x, y, var.equal=TRUE)       # run the Student's t-test
Two Sample t-test

data:  x and y
t = -1.7828, df = 28, p-value = 0.08547
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
  -6.1611557  0.4271893
sample estimates:
  mean of x mean of y
102.2136  105.0806
```

From the R output, the observed value of *T* is $t = -1.7828$. The negative sign is due to the fact that the sample mean of x is less than the sample mean of y. Using the `qt()` function in R, a *T* value of 2.0484 corresponds to a 0.05 significance level.

```
# obtain t value for a two-sided test at a 0.05 significance level
qt(p=0.05/2, df=28, lower.tail= FALSE)
2.048407
```

Because the magnitude of the observed *T* statistic is less than the *T* value corresponding to the 0.05 significance level ($|-1.7828| < 2.0484$), the null hypothesis is not rejected. Because the alternative hypothesis is that the means are not equal ($\mu_1 \neq \mu_2$), the possibilities of both $\mu_1 > \mu_2$ and $\mu_1 < \mu_2$ need to be considered. This form of Student's *t*-test is known as a ***two-sided hypothesis test***, and it is necessary for the sum of the probabilities under both tails of the *t*-distribution to equal the significance level. It is customary to evenly

divide the significance level between both tails. So, $p = 0.05/2 = 0.025$ was used in the `qt()` function to obtain the appropriate *t*-value.

To simplify the comparison of the *t*-test results to the significance level, the R output includes a quantity known as the ***p-value***. In the preceding example, the *p*-value is 0.08547, which is the sum of $P(T \leq -1.7828)$ and $P(T \geq 1.7828)$. Figure 3-24 illustrates the *t*-statistic for the area under the tail of a *t*-distribution. The *-t* and *t* are the observed values of the *t*-statistic. In the R output, $t = 1.7828$. The left shaded area corresponds to the $P(T \leq -1.7828)$, and the right shaded area corresponds to the $P(T \geq 1.7828)$.

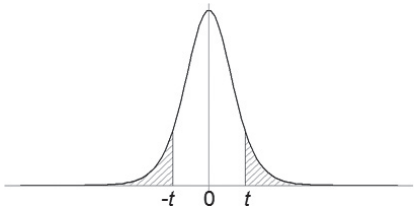

FIGURE 3-24 *Area under the tails (shaded) of a student's t-distribution*

In the R output, for a significance level of 0.05, the null hypothesis would not be rejected because the likelihood of a *T* value of magnitude 1.7828 or greater would occur at higher probability than 0.05. However, based on the *p*-value, if the significance level was chosen to be 0.10, instead of 0.05, the null hypothesis would be rejected. In general, the *p*-value offers the probability of observing such a sample result given the null hypothesis is `TRUE`.

A key assumption in using Student's *t*-test is that the population variances are equal. In the previous example, the `t.test()` function call includes `var.equal=TRUE` to specify that equality of the variances should be assumed. If that assumption is not appropriate, then Welch's *t*-test should be used.

Welch's t-test

When the equal population variance assumption is not justified in performing Student's *t*-test for the difference of means, Welch's *t*-test [14] can be used based on *T* expressed in Equation 3-2.

$$T_{welch} = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{S_1^2}{n_1} + \frac{S_2^2}{n_2}}} \tag{3-2}$$

where $\bar{X}_i$, S_i^2, and n_i correspond to the *i*-th sample mean, sample variance, and sample size. Notice that Welch's *t*-test uses the sample variance (S_i^2) for each population instead of the pooled sample variance.

In Welch's test, under the remaining assumptions of random samples from two normal populations with the same mean, the distribution of *T* is approximated by the *t*-distribution. The following R code performs the Welch's *t*-test on the same set of data analyzed in the earlier Student's *t*-test example.

```
t.test(x, y, var.equal=FALSE)       # run the Welch's t-test

Welch Two Sample t-test

data:  x and y
t = -1.6596, df = 15.118, p-value = 0.1176
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
  -6.546629  0.812663
sample estimates:
  mean of x mean of y
102.2136  105.0806
```

In this particular example of using Welch's *t*-test, the *p*-value is 0.1176, which is greater than the *p*-value of 0.08547 observed in the Student's *t*-test example. In this case, the null hypothesis would not be rejected at a 0.10 or 0.05 significance level.

It should be noted that the degrees of freedom calculation is not as straightforward as in the Student's *t*-test. In fact, the degrees of freedom calculation often results in a non-integer value, as in this example. The degrees of freedom for Welch's *t*-test is defined in Equation 3-3.

$$df = \frac{\left(\frac{S_1^2}{n_1} + \frac{S_2^2}{n_2}\right)^2}{\frac{\left(\frac{S_1^2}{n_1}\right)^2}{n_1 - 1} + \frac{\left(\frac{S_2^2}{n_2}\right)^2}{n_2 - 1}} \tag{3-3}$$

In both the Student's and Welch's *t*-test examples, the R output provides 95% confidence intervals on the difference of the means. In both examples, the confidence intervals straddle zero. Regardless of the result of the hypothesis test, the confidence interval provides an interval estimate of the difference of the population means, not just a point estimate.

A ***confidence interval*** is an interval estimate of a population parameter or characteristic based on sample data. A confidence interval is used to indicate the uncertainty of a point estimate. If $\overline{x}$ is the estimate of some unknown population mean μ, the confidence interval provides an idea of how close $\overline{x}$ is to the unknown μ. For example, a 95% confidence interval for a population mean straddles the `TRUE`, but unknown mean 95% of the time. Consider Figure 3-25 as an example. Assume the confidence level is 95%. If the task is to estimate the mean of an unknown value μ in a normal distribution with known standard deviation σ and the estimate based on n observations is x, then the interval $\overline{x} \pm \frac{2\sigma}{\sqrt{n}}$ straddles the unknown value of μ with about a 95% chance. If one takes 100 different samples and computes the 95% confidence interval for the mean, 95 of the 100 confidence intervals will be expected to straddle the population mean μ.

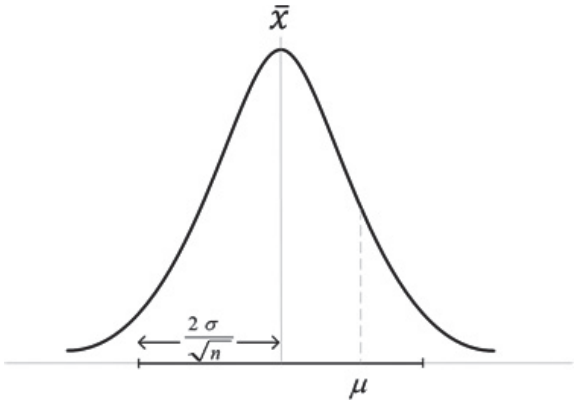

FIGURE 3-25 *A 95% confidence interval straddling the unknown population mean μ*

Confidence intervals appear again in Section 3.3.6 on ANOVA. Returning to the discussion of hypothesis testing, a key assumption in both the Student's and Welch's *t*-test is that the relevant population attribute is normally distributed. For non-normally distributed data, it is sometimes possible to transform the collected data to approximate a normal distribution. For example, taking the logarithm of a dataset can often transform skewed data to a dataset that is at least symmetric around its mean. However, if such transformations are ineffective, there are tests like the Wilcoxon rank-sum test that can be applied to see if two population distributions are different.

3.3.3 Wilcoxon Rank-Sum Test

A *t*-test represents a ***parametric test*** in that it makes assumptions about the population distributions from which the samples are drawn. If the populations cannot be assumed or transformed to follow a normal distribution, a ***nonparametric test*** can be used. The ***Wilcoxon rank-sum test*** [15] is a nonparametric hypothesis test that checks whether two populations are identically distributed. Assuming the two populations are identically distributed, one would expect that the ordering of any sampled observations would be evenly intermixed among themselves. For example, in ordering the observations, one would not expect to see a large number of observations from one population grouped together, especially at the beginning or the end of ordering.

Let the two populations again be `pop1` and `pop2`, with independently random samples of size n_1 and n_2 respectively. The total number of observations is then $N = n_1 + n_2$. The first step of the Wilcoxon test is to rank the set of observations from the two groups as if they came from one large group. The smallest observation receives a rank of 1, the second smallest observation receives a rank of 2, and so on with the largest observation being assigned the rank of *N*. Ties among the observations receive a rank equal to the average of the ranks they span. The test uses ranks instead of numerical outcomes to avoid specific assumptions about the shape of the distribution.

After ranking all the observations, the assigned ranks are summed for at least one population's sample. If the distribution of `pop1` is shifted to the right of the other distribution, the rank-sum corresponding to `pop1`'s sample should be larger than the rank-sum of `pop2`. The Wilcoxon rank-sum test determines the

significance of the observed rank-sums. The following R code performs the test on the same dataset used for the previous *t*-test.

```
wilcox.test(x, y, conf.int = TRUE)

Wilcoxon rank sum test

data:  x and y
W = 55, p-value = 0.04903
alternative hypothesis: true location shift is not equal to 0
95 percent confidence interval:
  -6.2596774 -0.1240618
sample estimates:
  difference in location
-3.417658
```

The `wilcox.test()` function ranks the observations, determines the respective rank-sums corresponding to each population's sample, and then determines the probability of such rank-sums of such magnitude being observed assuming that the population distributions are identical. In this example, the probability is given by the *p*-value of 0.04903. Thus, the null hypothesis would be rejected at a 0.05 significance level. The reader is cautioned against interpreting that one hypothesis test is clearly better than another test based solely on the examples given in this section.

Because the Wilcoxon test does not assume anything about the population distribution, it is generally considered more robust than the *t*-test. In other words, there are fewer assumptions to violate. However, when it is reasonable to assume that the data is normally distributed, Student's or Welch's *t*-test is an appropriate hypothesis test to consider.

3.3.4 Type I and Type II Errors

A hypothesis test may result in two types of errors, depending on whether the test accepts or rejects the null hypothesis. These two errors are known as type I and type II errors.

- A ***type I error*** is the rejection of the null hypothesis when the null hypothesis is `TRUE`. The probability of the type I error is denoted by the Greek letter α.
- A ***type II error*** is the acceptance of a null hypothesis when the null hypothesis is `FALSE`. The probability of the type II error is denoted by the Greek letter β.

Table 3-6 lists the four possible states of a hypothesis test, including the two types of errors.

TABLE 3-6 *Type I and Type II Error*

	H_0 is true	H_0 is false
H_0 is accepted	Correct outcome	*Type II Error*
H_0 is rejected	*Type I error*	Correct outcome

The significance level, as mentioned in the Student's *t*-test discussion, is equivalent to the type I error. For a significance level such as $\alpha = 0.05$, if the null hypothesis ($\mu_1 = \mu_2$) is `TRUE`, there is a 5% chance that the observed *T* value based on the sample data will be large enough to reject the null hypothesis. By selecting an appropriate significance level, the probability of committing a type I error can be defined before any data is collected or analyzed.

The probability of committing a Type II error is somewhat more difficult to determine. If two population means are truly not equal, the probability of committing a type II error will depend on how far apart the means truly are. To reduce the probability of a type II error to a reasonable level, it is often necessary to increase the sample size. This topic is addressed in the next section.

3.3.5 Power and Sample Size

The ***power*** of a test is the probability of correctly rejecting the null hypothesis. It is denoted by $1-\beta$, where β is the probability of a type II error. Because the power of a test improves as the sample size increases, power is used to determine the necessary sample size. In the difference of means, the power of a hypothesis test depends on the true difference of the population means. In other words, for a fixed significance level, a larger sample size is required to detect a smaller difference in the means. In general, the magnitude of the difference is known as the ***effect size***. As the sample size becomes larger, it is easier to detect a given effect size, δ, as illustrated in Figure 3-26.

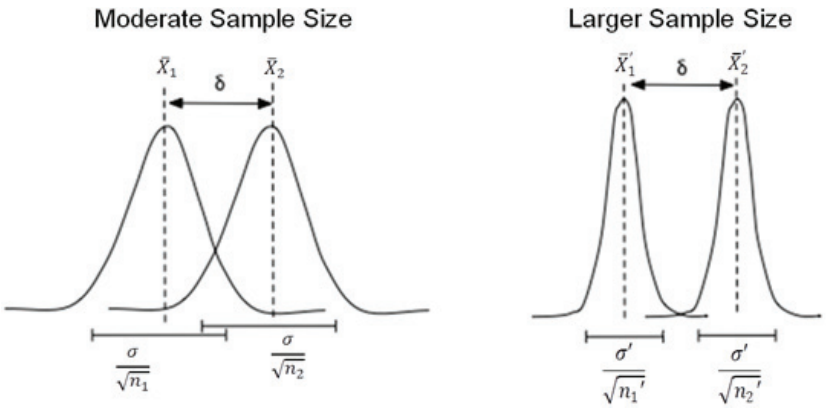

Figure 3-26 *A larger sample size better identifies a fixed effect size*

With a large enough sample size, almost any effect size can appear statistically significant. However, a very small effect size may be useless in a practical sense. It is important to consider an appropriate effect size for the problem at hand.

3.3.6 ANOVA

The *hypothesis tests* presented in the previous sections are good for analyzing means between two populations. But what if there are more than two populations? Consider an example of testing the impact of

nutrition and exercise on 60 candidates between age 18 and 50. The candidates are randomly split into six groups, each assigned with a different weight loss strategy, and the goal is to determine which strategy is the most effective.

- Group 1 only eats junk food.
- Group 2 only eats healthy food.
- Group 3 eats junk food and does cardio exercise every other day.
- Group 4 eats healthy food and does cardio exercise every other day.
- Group 5 eats junk food and does both cardio and strength training every other day.
- Group 6 eats healthy food and does both cardio and strength training every other day.

Multiple *t*-tests could be applied to each pair of weight loss strategies. In this example, the weight loss of Group 1 is compared with the weight loss of Group 2, 3, 4, 5, or 6. Similarly, the weight loss of Group 2 is compared with that of the next 4 groups. Therefore, a total of 15 *t*-tests would be performed.

However, multiple *t*-tests may not perform well on several populations for two reasons. First, because the number of *t*-tests increases as the number of groups increases, analysis using the multiple *t*-tests becomes cognitively more difficult. Second, by doing a greater number of analyses, the probability of committing at least one type I error somewhere in the analysis greatly increases.

Analysis of Variance (***ANOVA***) is designed to address these issues. ANOVA is a generalization of the hypothesis testing of the difference of two population means. ANOVA tests if any of the population means differ from the other population means. The null hypothesis of ANOVA is that all the population means are equal. The alternative hypothesis is that at least one pair of the population means is not equal. In other words,

- $\boldsymbol{H_0}$: $\mu_1 = \mu_2 = \ldots = \mu_n$
- $\boldsymbol{H_A}$: $\mu_i \neq \mu_j$ for at least one pair of i, j

As seen in Section 3.3.2, "Difference of Means," each population is assumed to be normally distributed with the same variance.

The first thing to calculate for the ANOVA is the test statistic. Essentially, the goal is to test whether the clusters formed by each population are more tightly grouped than the spread across all the populations.

Let the total number of populations be k. The total number of samples N is randomly split into the k groups. The number of samples in the i-th group is denoted as n_i, and the mean of the group is $\overline{X}_i$ where $i \in [1, k]$. The mean of all the samples is denoted as $\overline{X}_0$.

The ***between-groups mean sum of squares***, S_B^2, is an estimate of the ***between-groups variance***. It measures how the population means vary with respect to the grand mean, or the mean spread across all the populations. Formally, this is presented as shown in Equation 3-4.

$$S_B^2 = \frac{1}{k-1}\sum_{i=1}^{k} n_i \cdot (\overline{x}_i - \overline{x}_0)^2 \tag{3-4}$$

The ***within-group mean sum of squares***, S_W^2, is an estimate of the ***within-group variance***. It quantifies the spread of values within groups. Formally, this is presented as shown in Equation 3-5.

$$S_W^2 = \frac{1}{n-k}\sum_{i=1}^{k}\sum_{j=1}^{n_i}(x_{ij} - \overline{x}_i)^2 \quad (3\text{-}5)$$

If S_B^2 is much larger than S_W^2, then some of the population means are different from each other.

The *F*-test statistic is defined as the ratio of the between-groups mean sum of squares and the within-group mean sum of squares. Formally, this is presented as shown in Equation 3-6.

$$F = \frac{S_B^2}{S_W^2} \quad (3\text{-}6)$$

The *F*-test statistic in ANOVA can be thought of as a measure of how different the means are relative to the variability within each group. The larger the observed *F*-test statistic, the greater the likelihood that the differences between the means are due to something other than chance alone. The *F*-test statistic is used to test the hypothesis that the observed effects are not due to chance—that is, if the means are significantly different from one another.

Consider an example that every customer who visits a retail website gets one of two promotional offers or gets no promotion at all. The goal is to see if making the promotional offers makes a difference. ANOVA could be used, and the null hypothesis is that neither promotion makes a difference. The code that follows randomly generates a total of 500 observations of purchase sizes on three different offer options.

```
offers <- sample(c("offer1", "offer2", "nopromo"), size=500, replace=T)

# Simulated 500 observations of purchase sizes on the 3 offer options
purchasesize <- ifelse(offers=="offer1", rnorm(500, mean=80, sd=30),
                ifelse(offers=="offer2", rnorm(500, mean=85, sd=30),
                      rnorm(500, mean=40, sd=30)))

# create a data frame of offer option and purchase size
offertest <- data.frame(offer=as.factor(offers),
                        purchase_amt=purchasesize)
```

The summary of the *offertest* data frame shows that 170 `offer1`, 161 `offer2`, and 169 `nopromo` (no promotion) offers have been made. It also shows the range of purchase size (`purchase_amt`) for each of the three offer options.

```
# display a summary of offertest where offer="offer1"
summary(offertest[offertest$offer=="offer1",])
     offer      purchase_amt
 nopromo:  0   Min.   :  4.521
 offer1 :170   1st Qu.: 58.158
 offer2 :  0   Median : 76.944
               Mean   : 81.936
               3rd Qu.:104.959
               Max.   :180.507

# display a summary of offertest where offer="offer2"
summary(offertest[offertest$offer=="offer2",])
```

```
     offer      purchase_amt
 nopromo:  0   Min.   : 14.04
 offer1 :  0   1st Qu.: 69.46
 offer2 :161   Median : 90.20
               Mean   : 89.09
               3rd Qu.:107.48
               Max.   :154.33

# display a summary of offertest where offer="nopromo"
summary(offertest[offertest$offer=="nopromo",])
     offer      purchase_amt
 nopromo:169   Min.   :-27.00
 offer1 :  0   1st Qu.: 20.22
 offer2 :  0   Median : 42.44
               Mean   : 40.97
               3rd Qu.: 58.96
               Max.   :164.04
```

The `aov()` function performs the ANOVA on purchase size and offer options.

```
# fit ANOVA test
model <- aov(purchase_amt ~ offers, data=offertest)
```

The `summary()` function shows a summary of the model. The degrees of freedom for offers is 2, which corresponds to the $k-1$ in the denominator of Equation 3-4. The degrees of freedom for residuals is 497, which corresponds to the $n-k$ in the denominator of Equation 3-5.

```
summary(model)
             Df Sum Sq Mean Sq F value Pr(>F)
offers        2 225222  112611   130.6 <2e-16 ***
Residuals   497 428470     862
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The output also includes the S_B^2 (112,611), S_W^2 (862), the *F*-test statistic (130.6), and the *p*-value (< 2e–16). The *F*-test statistic is much greater than 1 with a *p*-value much less than 1. Thus, the null hypothesis that the means are equal should be rejected.

However, the result does not show whether `offer1` is different from `offer2`, which requires additional tests. The `TukeyHSD()` function implements Tukey's Honest Significant Difference (HSD) on all pair-wise tests for difference of means.

```
TukeyHSD(model)
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = purchase_amt ~ offers, data = offertest)

$offers
                    diff        lwr      upr     p adj
offer1-nopromo 40.961437 33.4638483 48.45903 0.0000000
```

```
offer2-nopromo 48.120286 40.5189446 55.72163 0.0000000
offer2-offer1   7.158849 -0.4315769 14.74928 0.0692895
```

The result includes *p*-values of pair-wise comparisons of the three offer options. The *p*-values for `offer1-nopromo` and `offer-nopromo` are equal to 0, smaller than the significance level 0.05. This suggests that both `offer1` and `offer2` are significantly different from `nopromo`. A *p*-value of 0.0692895 for `offer2` against `offer1` is greater than the significance level 0.05. This suggests that `offer2` is *not* significantly different from `offer1`.

Because only the influence of one factor (offers) was executed, the presented ANOVA is known as one-way ANOVA. If the goal is to analyze two factors, such as offers and day of week, that would be a two-way ANOVA [16]. If the goal is to model more than one outcome variable, then multivariate ANOVA (or MANOVA) could be used.

Summary

R is a popular package and programming language for data exploration, analytics, and visualization. As an introduction to R, this chapter covers the R GUI, data I/O, attribute and data types, and descriptive statistics. This chapter also discusses how to use R to perform exploratory data analysis, including the discovery of dirty data, visualization of one or more variables, and customization of visualization for different audiences. Finally, the chapter introduces some basic statistical methods. The first statistical method presented in the chapter is the hypothesis testing. The Student's *t*-test and Welch's *t*-test are included as two example hypothesis tests designed for testing the difference of means. Other statistical methods and tools presented in this chapter include confidence intervals, Wilcoxon rank-sum test, type I and II errors, effect size, and ANOVA.

Exercises

1. How many levels does *fdata* contain in the following R code?

```
data = c(1,2,2,3,1,2,3,3,1,2,3,3,1)
fdata = factor(data)
```

2. Two vectors, *v1* and *v2*, are created with the following R code:

```
v1 <- 1:5
v2 <- 6:2
```

   What are the results of `cbind(v1,v2)` and `rbind(v1,v2)`?

3. What R command(s) would you use to remove null values from a dataset?

4. What R command can be used to install an additional R package?

5. What R function is used to encode a vector as a category?

6. What is a rug plot used for in a density plot?

7. An online retailer wants to study the purchase behaviors of its customers. Figure 3-27 shows the density plot of the purchase sizes (in dollars). What would be your recommendation to enhance the plot to detect more structures that otherwise might be missed?

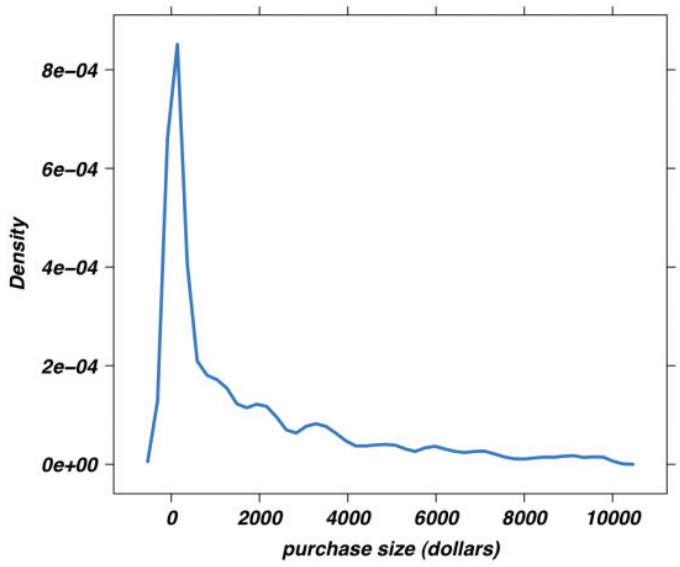

FIGURE 3-27 *Density plot of purchase size*

8. How many sections does a box-and-whisker divide the data into? What are these sections?
9. What attributes are correlated according to Figure 3-18? How would you describe their relationships?
10. What function can be used to fit a nonlinear line to the data?
11. If a graph of data is skewed and all the data is positive, what mathematical technique may be used to help detect structures that might otherwise be overlooked?
12. What is a type I error? What is a type II error? Is one always more serious than the other? Why?
13. Suppose everyone who visits a retail website gets one promotional offer or no promotion at all. We want to see if making a promotional offer makes a difference. What statistical method would you recommend for this analysis?
14. You are analyzing two normally distributed populations, and your null hypothesis is that the mean μ_1 of the first population is equal to the mean μ_2 of the second. Assume the significance level is set at 0.05. If the observed *p*-value is 4.33e-05, what will be your decision regarding the null hypothesis?

Bibliography

[1] The R Project for Statistical Computing, "R Licenses." [Online]. Available: `http://www.r-project.org/Licenses/`. [Accessed 10 December 2013].

[2] The R Project for Statistical Computing, "The Comprehensive R Archive Network." [Online]. Available: `http://cran.r-project.org/`. [Accessed 10 December 2013].

[3] J. Fox and M. Bouchet-Valat, "The R Commander: A Basic-Statistics GUI for R," CRAN. [Online]. Available: http://socserv.mcmaster.ca/jfox/Misc/Rcmdr/. [Accessed 11 December 2013].

[4] G. Williams, M. V. Culp, E. Cox, A. Nolan, D. White, D. Medri, and A. Waljee, "Rattle: Graphical User Interface for Data Mining in R," CRAN. [Online]. Available: http://cran.r-project.org/web/packages/rattle/index.html. [Accessed 12 December 2013].

[5] RStudio, "RStudio IDE" [Online]. Available: http://www.rstudio.com/ide/. [Accessed 11 December 2013].

[6] R Special Interest Group on Databases (R-SIG-DB), "DBI: R Database Interface." CRAN [Online]. Available: http://cran.r-project.org/web/packages/DBI/index.html. [Accessed 13 December 2013].

[7] B. Ripley, "RODBC: ODBC Database Access," CRAN. [Online]. Available: http://cran.r-project.org/web/packages/RODBC/index.html. [Accessed 13 December 2013].

[8] S. S. Stevens, "On the Theory of Scales of Measurement," *Science,* vol. 103, no. 2684, p. 677–680, 1946.

[9] D. C. Hoaglin, F. Mosteller, and J. W. Tukey, *Understanding Robust and Exploratory Data Analysis*, New York: Wiley, 1983.

[10] F. J. Anscombe, "Graphs in Statistical Analysis," *The American Statistician,* vol. 27, no. 1, pp. 17–21, 1973.

[11] H. Wickham, "ggplot2," 2013. [Online]. Available: http://ggplot2.org/. [Accessed 8 January 2014].

[12] W. S. Cleveland, *Visualizing Data*, Lafayette, IN: Hobart Press, 1993.

[13] R. A. Fisher, "The Use of Multiple Measurements in Taxonomic Problems," *Annals of Eugenics,* vol. 7, no. 2, pp. 179–188, 1936.

[14] B. L. Welch, "The Generalization of "Student's" Problem When Several Different Population Variances Are Involved," *Biometrika,* vol. 34, no. 1–2, pp. 28–35, 1947.

[15] F. Wilcoxon, "Individual Comparisons by Ranking Methods," *Biometrics Bulletin,* vol. 1, no. 6, pp. 80–83, 1945.

[16] J. J. Faraway, "Practical Regression and Anova Using R," July 2002. [Online]. Available: http://cran.r-project.org/doc/contrib/Faraway-PRA.pdf. [Accessed 22 January 2014].

4

Advanced Analytical Theory and Methods: Clustering

Key Concepts

Centroid
Clustering
K-means
Unsupervised
Within Sum of Squares

Building upon the introduction to R presented in Chapter 3, "Review of Basic Data Analytic Methods Using R," Chapter 4, "Advanced Analytical Theory and Methods: Clustering" through Chapter 9, "Advanced Analytical Theory and Methods: Text Analysis" describe several commonly used analytical methods that may be considered for the Model Planning and Execution phases (Phases 3 and 4) of the Data Analytics Lifecycle. This chapter considers clustering techniques and algorithms.

4.1 Overview of Clustering

In general, clustering is the use of ***unsupervised*** techniques for grouping similar objects. In machine learning, unsupervised refers to the problem of finding hidden structure within unlabeled data. Clustering techniques are unsupervised in the sense that the data scientist does not determine, in advance, the labels to apply to the clusters. The structure of the data describes the objects of interest and determines how best to group the objects. For example, based on customers' personal income, it is straightforward to divide the customers into three groups depending on arbitrarily selected values. The customers could be divided into three groups as follows:

- Earn less than $10,000
- Earn between $10,000 and $99,999
- Earn $100,000 or more

In this case, the income levels were chosen somewhat subjectively based on easy-to-communicate points of delineation. However, such groupings do not indicate a natural affinity of the customers within each group. In other words, there is no inherent reason to believe that the customer making $90,000 will behave any differently than the customer making $110,000. As additional dimensions are introduced by adding more variables about the customers, the task of finding meaningful groupings becomes more complex. For instance, suppose variables such as age, years of education, household size, and annual purchase expenditures were considered along with the personal income variable. What are the natural occurring groupings of customers? This is the type of question that clustering analysis can help answer.

Clustering is a method often used for exploratory analysis of the data. In clustering, there are no predictions made. Rather, clustering methods find the similarities between objects according to the object attributes and group the similar objects into clusters. Clustering techniques are utilized in marketing, economics, and various branches of science. A popular clustering method is k-means.

4.2 K-means

Given a collection of objects each with n measurable attributes, ***k-means*** [1] is an analytical technique that, for a chosen value of k, identifies k clusters of objects based on the objects' proximity to the center of the k groups. The center is determined as the arithmetic average (mean) of each cluster's n-dimensional vector of attributes. This section describes the algorithm to determine the k means as well as how best to apply this technique to several use cases. Figure 4-1 illustrates three clusters of objects with two attributes. Each object in the dataset is represented by a small dot color-coded to the closest large dot, the mean of the cluster.

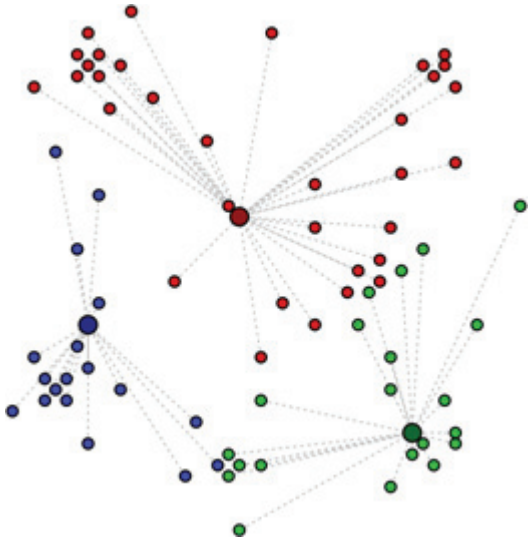

FIGURE 4-1 *Possible k-means clusters for k=3*

4.2.1 Use Cases

Clustering is often used as a lead-in to classification. Once the clusters are identified, labels can be applied to each cluster to classify each group based on its characteristics. Classification is covered in more detail in Chapter 7, "Advanced Analytical Theory and Methods: Classification." Clustering is primarily an exploratory technique to discover hidden structures of the data, possibly as a prelude to more focused analysis or decision processes. Some specific applications of k-means are image processing, medical, and customer segmentation.

Image Processing

Video is one example of the growing volumes of unstructured data being collected. Within each frame of a video, k-means analysis can be used to identify objects in the video. For each frame, the task is to determine which pixels are most similar to each other. The attributes of each pixel can include brightness, color, and location, the x and y coordinates in the frame. With security video images, for example, successive frames are examined to identify any changes to the clusters. These newly identified clusters may indicate unauthorized access to a facility.

Medical

Patient attributes such as age, height, weight, systolic and diastolic blood pressures, cholesterol level, and other attributes can identify naturally occurring clusters. These clusters could be used to target individuals for specific preventive measures or clinical trial participation. Clustering, in general, is useful in biology for the classification of plants and animals as well as in the field of human genetics.

Customer Segmentation

Marketing and sales groups use k-means to better identify customers who have similar behaviors and spending patterns. For example, a wireless provider may look at the following customer attributes: monthly bill, number of text messages, data volume consumed, minutes used during various daily periods, and years as a customer. The wireless company could then look at the naturally occurring clusters and consider tactics to increase sales or reduce the customer ***churn rate***, the proportion of customers who end their relationship with a particular company.

4.2.2 Overview of the Method

To illustrate the method to find k clusters from a collection of M objects with n attributes, the two-dimensional case (n = 2) is examined. It is much easier to visualize the k-means method in two dimensions. Later in the chapter, the two-dimension scenario is generalized to handle any number of attributes.

Because each object in this example has two attributes, it is useful to consider each object corresponding to the point (x_i, y_i), where x and y denote the two attributes and i = 1, 2 … M. For a given cluster of m points (m $\leq$M), the point that corresponds to the cluster's mean is called a ***centroid***. In mathematics, a centroid refers to a point that corresponds to the center of mass for an object.

The k-means algorithm to find k clusters can be described in the following four steps.

1. Choose the value of k and the k initial guesses for the centroids.

   In this example, k = 3, and the initial centroids are indicated by the points shaded in red, green, and blue in Figure 4-2.

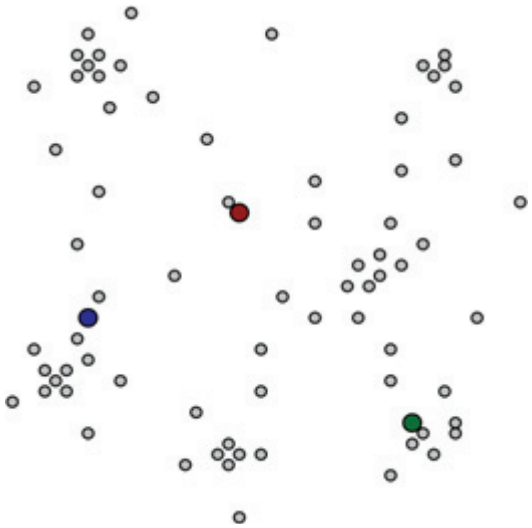

FIGURE 4-2 *Initial starting points for the centroids*

2. Compute the distance from each data point (x_i, y_i) to each centroid. Assign each point to the closest centroid. This association defines the first k clusters.

   In two dimensions, the distance, d, between any two points, (x_1, y_1) and (x_2, y_2), in the Cartesian plane is typically expressed by using the Euclidean distance measure provided in Equation 4-1.

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{4-1}$$

   In Figure 4-3, the points closest to a centroid are shaded the corresponding color.

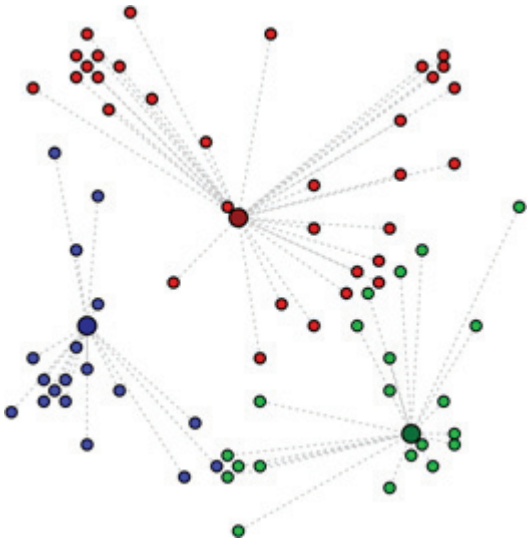

FIGURE 4-3 *Points are assigned to the closest centroid*

3. Compute the centroid, the center of mass, of each newly defined cluster from Step 2.

   In Figure 4-4, the computed centroids in Step 3 are the lightly shaded points of the corresponding color. In two dimensions, the centroid (x_C, y_C) of the m points in a k-means cluster is calculated as follows in Equation 4-2.

$$(x_C, y_C) = \left(\frac{\sum_{i=1}^{m} x_i}{m}, \frac{\sum_{i=1}^{m} y_i}{m} \right) \tag{4-2}$$

   Thus, (x_C, y_C) is the ordered pair of the arithmetic means of the coordinates of the m points in the cluster. In this step, a centroid is computed for each of the k clusters.

4. Repeat Steps 2 and 3 until the algorithm converges to an answer.

a. Assign each point to the closest centroid computed in Step 3.

b. Compute the centroid of newly defined clusters.

c. Repeat until the algorithm reaches the final answer.

Convergence is reached when the computed centroids do not change or the centroids and the assigned points oscillate back and forth from one iteration to the next. The latter case can occur when there are one or more points that are equal distances from the computed centroid.

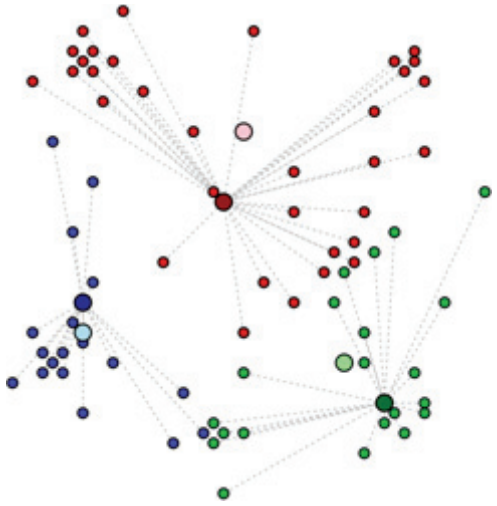

FIGURE 4-4 *Compute the mean of each cluster*

To generalize the prior algorithm to n dimensions, suppose there are M objects, where each object is described by n attributes or property values $(p_1, p_2, \ldots p_n)$. Then object i is described by $(p_{i1}, p_{i2}, \ldots p_{in})$ for $i = 1,2,\ldots$, M. In other words, there is a matrix with M rows corresponding to the M objects and n columns to store the attribute values. To expand the earlier process to find the k clusters from two dimensions to n dimensions, the following equations provide the formulas for calculating the distances and the locations of the centroids for n ≥ 1.

For a given point, $\boldsymbol{p}_i$, at $(p_{i1}, p_{i2}, \ldots p_{in})$ and a centroid, $\boldsymbol{q}$, located at $(q_1, q_2, \ldots q_n)$, the distance, $\boldsymbol{d}$, between $\boldsymbol{p}_i$ and $\boldsymbol{q}$, is expressed as shown in Equation 4-3.

$$d(p_i, q) = \sqrt{\sum_{j=1}^{n} (p_{ij} - q_j)^2} \tag{4-3}$$

The centroid, $\boldsymbol{q}$, of a cluster of m points, $(p_{i1}, p_{i2}, \ldots p_{in})$, is calculated as shown in Equation 4-4.

$$(q_1, q_2, \ldots, q_n) = \left(\frac{\sum_{i=1}^{m} p_{i1}}{m}, \frac{\sum_{i=1}^{m} p_{i2}}{m}, \ldots, \frac{\sum_{i=1}^{m} p_{in}}{m} \right) \tag{4-4}$$

4.2.3 Determining the Number of Clusters

With the preceding algorithm, k clusters can be identified in a given dataset, but what value of k should be selected? The value of k can be chosen based on a reasonable guess or some predefined requirement. However, even then, it would be good to know how much better or worse having k clusters versus k – 1 or k + 1 clusters would be in explaining the structure of the data. Next, a heuristic using the Within Sum of Squares (WSS) metric is examined to determine a reasonably optimal value of k. Using the distance function given in Equation 4-3, WSS is defined as shown in Equation 4-5.

$$WSS = \sum_{i=1}^{M} d(p_i, q^{(i)})^2 = \sum_{i=1}^{M} \sum_{j=1}^{n} \left(p_{ij} - q_j^{(i)}\right)^2 \quad (4\text{-}5)$$

In other words, WSS is the sum of the squares of the distances between each data point and the closest centroid. The term $q^{(i)}$ indicates the closest centroid that is associated with the ith point. If the points are relatively close to their respective centroids, the WSS is relatively small. Thus, if k + 1 clusters do not greatly reduce the value of WSS from the case with only k clusters, there may be little benefit to adding another cluster.

Using R to Perform a K-means Analysis

To illustrate how to use the WSS to determine an appropriate number, k, of clusters, the following example uses R to perform a k-means analysis. The task is to group 620 high school seniors based on their grades in three subject areas: English, mathematics, and science. The grades are averaged over their high school career and assume values from 0 to 100. The following R code establishes the necessary R libraries and imports the CSV file containing the grades.

```
library(plyr)
library(ggplot2)
library(cluster)
library(lattice)
library(graphics)
library(grid)
library(gridExtra)

#import the student grades
grade_input = as.data.frame(read.csv("c:/data/grades_km_input.csv"))
```

The following R code formats the grades for processing. The data file contains four columns. The first column holds a student identification (ID) number, and the other three columns are for the grades in the three subject areas. Because the student ID is not used in the clustering analysis, it is excluded from the k-means input matrix, *kmdata*.

```
kmdata_orig = as.matrix(grade_input[,c("Student","English", "Math","Science")])
kmdata <- kmdata_orig[,2:4]
```

```
kmdata[1:10,]

      English Math Science
 [1,]      99   96      97
 [2,]      99   96      97
 [3,]      98   97      97
 [4,]      95  100      95
 [5,]      95   96      96
 [6,]      96   97      96
 [7,]     100   96      97
 [8,]      95   98      98
 [9,]      98   96      96
[10,]      99   99      95
```

To determine an appropriate value for k, the k-means algorithm is used to identify clusters for k = 1, 2, …, 15. For each value of k, the WSS is calculated. If an additional cluster provides a better partitioning of the data points, the WSS should be markedly smaller than without the additional cluster.

The following R code loops through several k-means analyses for the number of centroids, `k`, varying from 1 to 15. For each k, the option `nstart=25` specifies that the k-means algorithm will be repeated 25 times, each starting with `k` random initial centroids. The corresponding value of WSS for each k-mean analysis is stored in the `wss` vector.

```
wss <- numeric(15)
for (k in 1:15) wss[k] <- sum(kmeans(kmdata, centers=k, nstart=25)$withinss)
```

Using the basic R plot function, each WSS is plotted against the respective number of centroids, 1 through 15. This plot is provided in Figure 4-5.

```
plot(1:15, wss, type="b", xlab="Number of Clusters", ylab="Within Sum of
Squares")
```

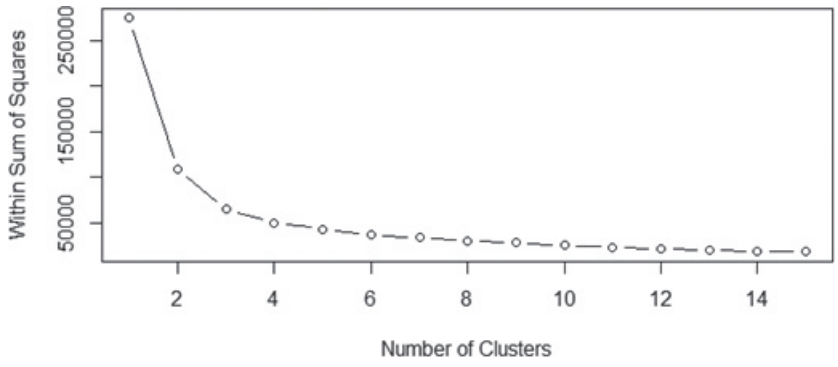

FIGURE 4-5 *WSS of the student grade data*

As can be seen, the WSS is greatly reduced when k increases from one to two. Another substantial reduction in WSS occurs at k = 3. However, the improvement in WSS is fairly linear for k > 3. Therefore, the k-means analysis will be conducted for k = 3. The process of identifying the appropriate value of k is referred to as finding the "elbow" of the WSS curve.

```
km = kmeans(kmdata,3, nstart=25)
km

K-means clustering with 3 clusters of sizes 158, 218, 244

Cluster means:
   English     Math  Science
1 97.21519 93.37342 94.86076
2 73.22018 64.62844 65.84862
3 85.84426 79.68033 81.50820

Clustering vector:
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
      1 1 1 1 1 1 1 1 1 1
 [41] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
      1 1 1 1 1 1 1 1 1 1
 [81] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
      1 1 1 1 1 1 1 1 1 1
[121] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
      3 3 3 3 3 3 3 3 3 3
[161] 3 3 3 3 3 3 1 3 3 3 3 3 3 3 3 3 3 1 1 3 3 1 3 3 3 1 3 3 3 3
      3 3 1 3 3 3 3 3 3 3
[201] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
      3 3 3 3 3 3 3 3 3 3
[241] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
      3 3 3 3 3 3 3 3 3 3
[281] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
      3 3 3 3 3 3 3 3 3 3
[321] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
      3 3 3 3 3 3 3 3 3 3
[361] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 3 2 3 2 3 3 3
      2 2 2 2 3 3 2 2 2 2
[401] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
      2 2 2 2 2 2 2 2 2 2
[441] 2 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2 3
      2 2 2 2 2 2 2 2 3 2
[481] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
      2 2 2 2 2 2 2 2 2 2
[521] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
      2 2 2 2 2 2 2 2 2 2
[561] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
      2 2 2 2 2 2 2 2 2 2
[601] 3 3 2 2 3 3 3 3 1 1 3 3 3 2 2 3 2 3 3 3

Within cluster sum of squares by cluster:
[1]  6692.589 34806.339 22984.131
 (between_SS / total_SS =  76.5 %)

Available components:

[1] "cluster"   "centers"  "totss"    "withinss"   "tot.withinss"
[6] "betweenss" "size"     "iter"     "ifault"
```

The displayed contents of the variable *km* include the following:

- The location of the cluster means
- A clustering vector that defines the membership of each student to a corresponding cluster 1, 2, or 3
- The WSS of each cluster
- A list of all the available k-means components

The reader can find details on these components and using k-means in R by employing the help facility.

The reader may have wondered whether the k-means results stored in *km* are equivalent to the WSS results obtained earlier in generating the plot in Figure 4-5. The following check verifies that the results are indeed equivalent.

```
c( wss[3] , sum(km$withinss) )
```

```
[1] 64483.06 64483.06
```

In determining the value of k, the data scientist should visualize the data and assigned clusters. In the following code, the `ggplot2` package is used to visualize the identified student clusters and centroids.

```
#prepare the student data and clustering results for plotting
df = as.data.frame(kmdata_orig[,2:4])
df$cluster = factor(km$cluster)
centers=as.data.frame(km$centers)

g1= ggplot(data=df, aes(x=English, y=Math, color=cluster )) +
  geom_point() + theme(legend.position="right") +
  geom_point(data=centers,
             aes(x=English,y=Math, color=as.factor(c(1,2,3))),
             size=10, alpha=.3, show_guide=FALSE)

g2 =ggplot(data=df, aes(x=English, y=Science, color=cluster )) +
  geom_point() +
  geom_point(data=centers,
             aes(x=English,y=Science, color=as.factor(c(1,2,3))),
             size=10, alpha=.3, show_guide=FALSE)

g3 = ggplot(data=df, aes(x=Math, y=Science, color=cluster )) +
  geom_point() +
  geom_point(data=centers,
             aes(x=Math,y=Science, color=as.factor(c(1,2,3))),
             size=10, alpha=.3, show_guide=FALSE)

tmp = ggplot_gtable(ggplot_build(g1))
```

```
grid.arrange(arrangeGrob(g1 + theme(legend.position="none"),
                         g2 + theme(legend.position="none"),
                         g3 + theme(legend.position="none"),
                         main ="High School Student Cluster Analysis",
                         ncol=1))
```

The resulting plots are provided in Figure 4-6. The large circles represent the location of the cluster means provided earlier in the display of the *km* contents. The small dots represent the students corresponding to the appropriate cluster by assigned color: red, blue, or green. In general, the plots indicate the three clusters of students: the top academic students (red), the academically challenged students (green), and the other students (blue) who fall somewhere between those two groups. The plots also highlight which students may excel in one or two subject areas but struggle in other areas.

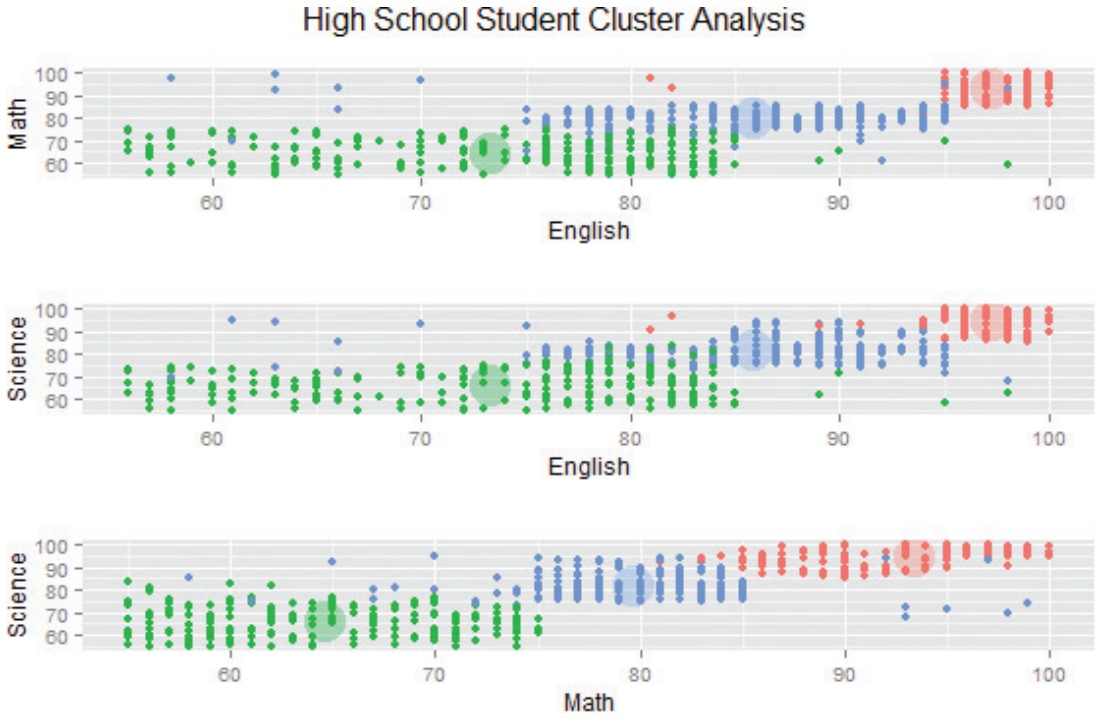

FIGURE 4-6 *Plots of the identified student clusters*

Assigning labels to the identified clusters is useful to communicate the results of an analysis. In a marketing context, it is common to label a group of customers as frequent shoppers or big spenders. Such designations are especially useful when communicating the clustering results to business users or executives. It is better to describe the marketing plan for big spenders rather than Cluster #1.

4.2.4 Diagnostics

The heuristic using WSS can provide at least several possible k values to consider. When the number of attributes is relatively small, a common approach to further refine the choice of k is to plot the data to determine how distinct the identified clusters are from each other. In general, the following questions should be considered.

- Are the clusters well separated from each other?
- Do any of the clusters have only a few points?
- Do any of the centroids appear to be too close to each other?

In the first case, ideally the plot would look like the one shown in Figure 4-7, when n = 2. The clusters are well defined, with considerable space between the four identified clusters. However, in other cases, such as Figure 4-8, the clusters may be close to each other, and the distinction may not be so obvious.

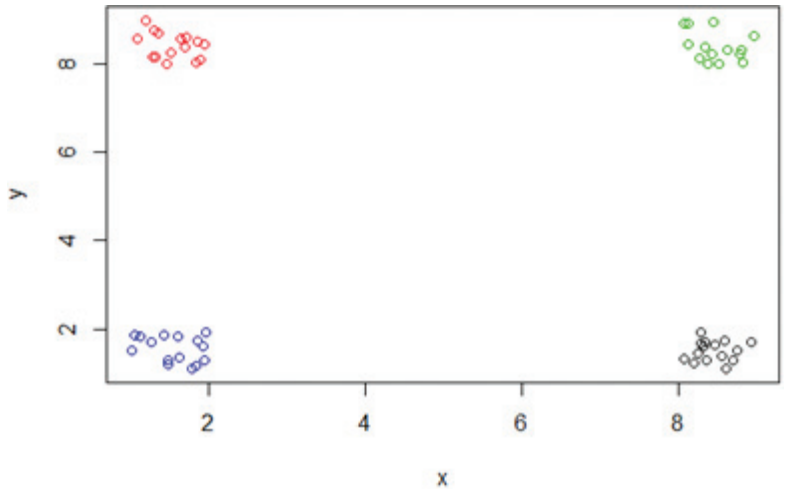

FIGURE 4-7 *Example of distinct clusters*

In such cases, it is important to apply some judgment on whether anything different will result by using more clusters. For example, Figure 4-9 uses six clusters to describe the same dataset as used in Figure 4-8. If using more clusters does not better distinguish the groups, it is almost certainly better to go with fewer clusters.

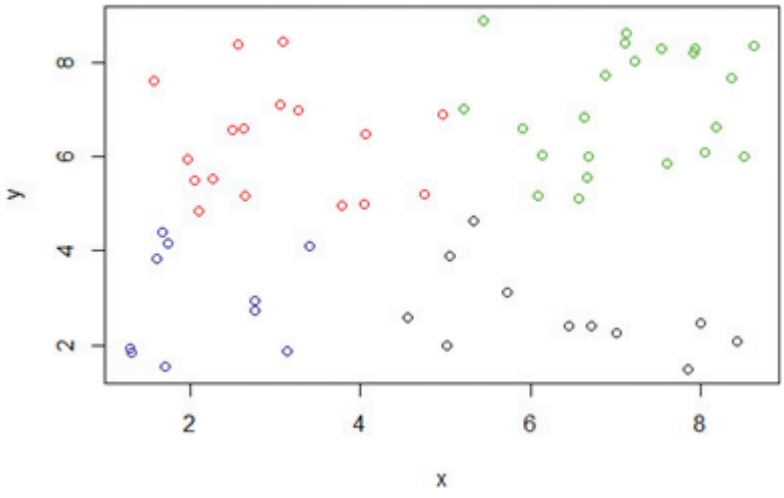

FIGURE 4-8 *Example of less obvious clusters*

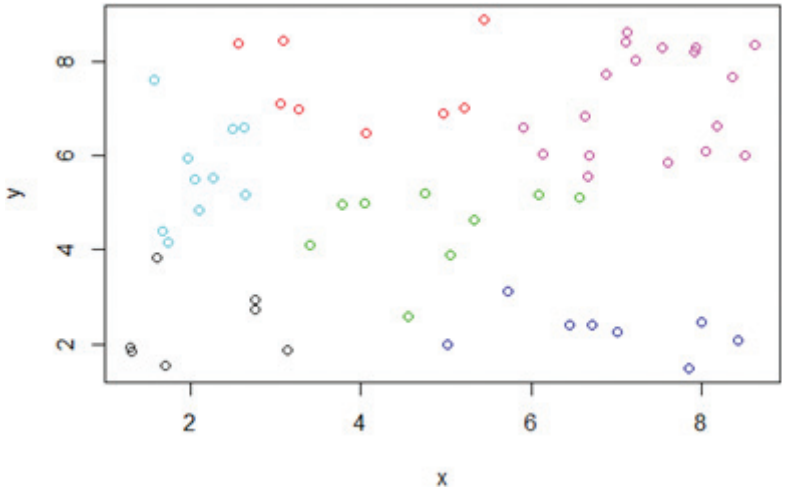

FIGURE 4-9 *Six clusters applied to the points from Figure 4-8*

4.2.5 Reasons to Choose and Cautions

K-means is a simple and straightforward method for defining clusters. Once clusters and their associated centroids are identified, it is easy to assign new objects (for example, new customers) to a cluster based on the object's distance from the closest centroid. Because the method is unsupervised, using k-means helps to eliminate subjectivity from the analysis.

Although k-means is considered an unsupervised method, there are still several decisions that the practitioner must make:

- What object attributes should be included in the analysis?
- What unit of measure (for example, miles or kilometers) should be used for each attribute?
- Do the attributes need to be rescaled so that one attribute does not have a disproportionate effect on the results?
- What other considerations might apply?

Object Attributes

Regarding which object attributes (for example, age and income) to use in the analysis, it is important to understand what attributes will be known at the time a new object will be assigned to a cluster. For example, information on existing customers' satisfaction or purchase frequency may be available, but such information may not be available for potential customers.

The Data Scientist may have a choice of a dozen or more attributes to use in the clustering analysis. Whenever possible and based on the data, it is best to reduce the number of attributes to the extent possible. Too many attributes can minimize the impact of the most important variables. Also, the use of several similar attributes can place too much importance on one type of attribute. For example, if five attributes related to personal wealth are included in a clustering analysis, the wealth attributes dominate the analysis and possibly mask the importance of other attributes, such as age.

When dealing with the problem of too many attributes, one useful approach is to identify any highly correlated attributes and use only one or two of the correlated attributes in the clustering analysis. As illustrated in Figure 4-10, a scatterplot matrix, as introduced in Chapter 3, is a useful tool to visualize the pair-wise relationships between the attributes.

The strongest relationship is observed to be between `Attribute3` and `Attribute7`. If the value of one of these two attributes is known, it appears that the value of the other attribute is known with near certainty. Other linear relationships are also identified in the plot. For example, consider the plot of `Attribute2` against `Attribute3`. If the value of `Attribute2` is known, there is still a wide range of possible values for `Attribute3`. Thus, greater consideration must be given prior to dropping one of these attributes from the clustering analysis.

Another option to reduce the number of attributes is to combine several attributes into one measure. For example, instead of using two attribute variables, one for Debt and one for Assets, a Debt to Asset ratio could be used. This option also addresses the problem when the magnitude of an attribute is not of real interest, but the relative magnitude is a more important measure.

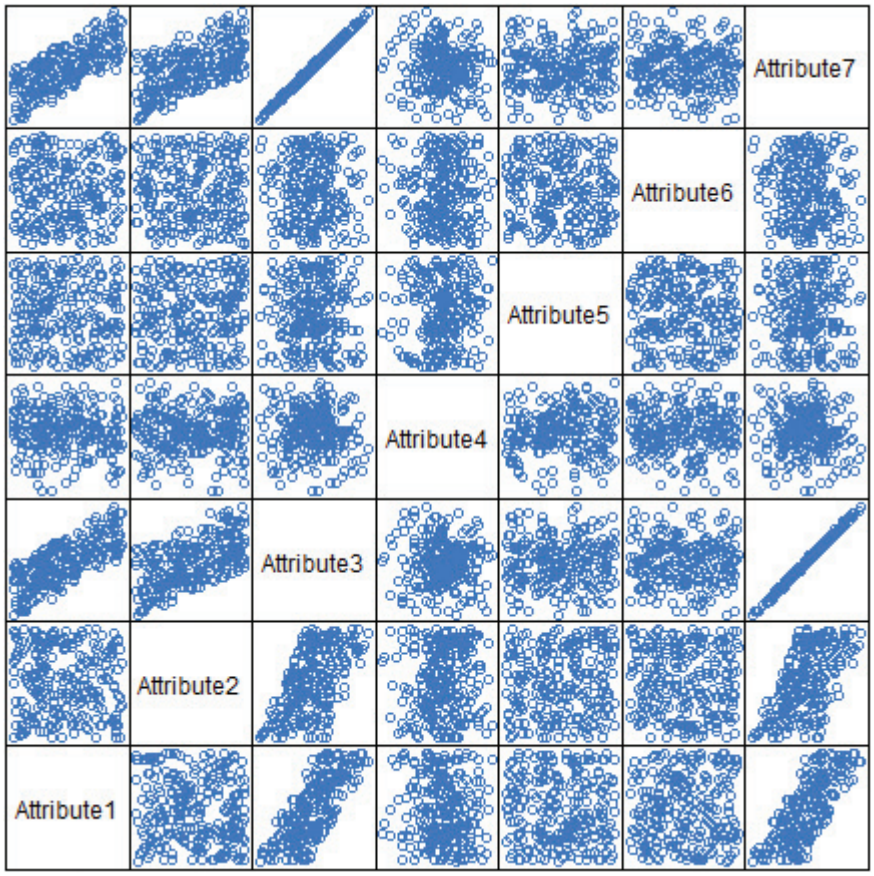

FIGURE 4-10 *Scatterplot matrix for seven attributes*

Units of Measure

From a computational perspective, the k-means algorithm is somewhat indifferent to the units of measure for a given attribute (for example, meters or centimeters for a patient's height). However, the algorithm will identify different clusters depending on the choice of the units of measure. For example, suppose that k-means is used to cluster patients based on age in years and height in centimeters. For k=2, Figure 4-11 illustrates the two clusters that would be determined for a given dataset.

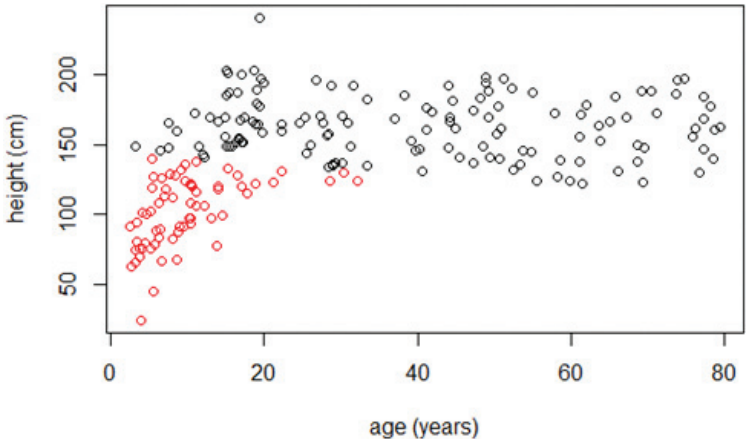

FIGURE 4-11 *Clusters with height expressed in centimeters*

But if the height was rescaled from centimeters to meters by dividing by 100, the resulting clusters would be slightly different, as illustrated in Figure 4-12.

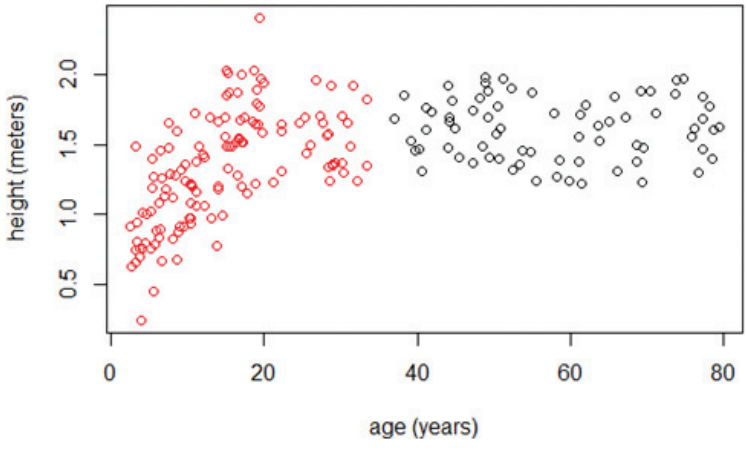

FIGURE 4-12 *Clusters with height expressed in meters*

When the height is expressed in meters, the magnitude of the ages dominates the distance calculation between two points. The height attribute provides only as much as the square between the difference of the maximum height and the minimum height or $(2.0-0)^2=4$ to the radicand, the number under the square root symbol in the distance formula given in Equation 4-3. Age can contribute as much as $(80-0)^2=6,400$ to the radicand when measuring the distance.

Rescaling

Attributes that are expressed in dollars are common in clustering analyses and can differ in magnitude from the other attributes. For example, if personal income is expressed in dollars and age is expressed in years, the income attribute, often exceeding $10,000, can easily dominate the distance calculation with ages typically less than 100 years.

Although some adjustments could be made by expressing the income in thousands of dollars (for example, 10 for $10,000), a more straightforward method is to divide each attribute by the attribute's standard deviation. The resulting attributes will each have a standard deviation equal to 1 and will be without units. Returning to the age and height example, the standard deviations are 23.1 years and 36.4 cm, respectively. Dividing each attribute value by the appropriate standard deviation and performing the k-means analysis yields the result shown in Figure 4-13.

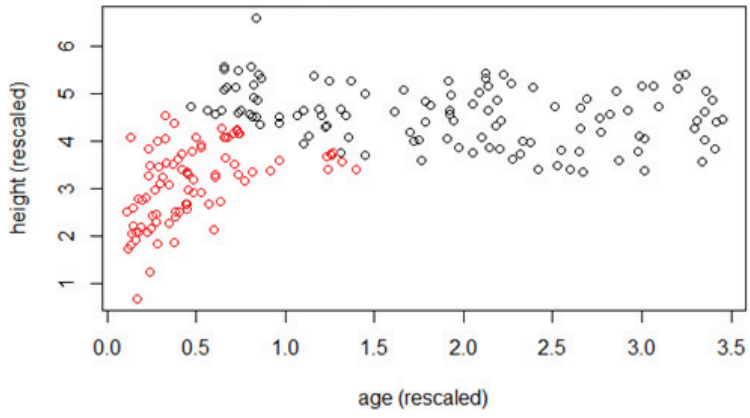

FIGURE 4-13 *Clusters with rescaled attributes*

With the rescaled attributes for age and height, the borders of the resulting clusters now fall somewhere between the two earlier clustering analyses. Such an occurrence is not surprising based on the magnitudes of the attributes of the previous clustering attempts. Some practitioners also subtract the means of the attributes to center the attributes around zero. However, this step is unnecessary because the distance formula is only sensitive to the scale of the attribute, not its location.

In many statistical analyses, it is common to transform typically skewed data, such as income, with long tails by taking the logarithm of the data. Such transformation can also be appied in k-means, but the Data Scientist needs to be aware of what effect this transformation will have. For example, if log_{10} of income expressed in dollars is used, the practitioner is essentially stating that, from a clustering perspective, $1,000 is as close to $10,000 as $10,000 is to $100,000 (*because* $log_{10}1{,}000 = 3$, $log_{10}10{,}000 = 4$, *and* $log_{10}100{,}000 = 5$). In many cases, the skewness of the data may be the reason to perform the clustering analysis in the first place.

Additional Considerations

The k-means algorithm is sensitive to the starting positions of the initial centroid. Thus, it is important to rerun the k-means analysis several times for a particular value of k to ensure the cluster results provide the overall minimum WSS. As seen earlier, this task is accomplished in R by using the `nstart` option in the `kmeans()` function call.

This chapter presented the use of the Euclidean distance function to assign the points to the closest centroids. Other possible function choices include the cosine similarity and the Manhattan distance functions. The cosine similarity function is often chosen to compare two documents based on the frequency of each word that appears in each of the documents [2]. For two points, p and q, at $(p_1, p_2, \ldots p_n)$ and $(q_1, q_2, \ldots q_n)$, respectively, the Manhattan distance, d_1, between p and q is expressed as shown in Equation 4-6.

$$d_1(p, q) = \sum_{j=1}^{n} |p_j - q_j| \tag{4-6}$$

The Manhattan distance function is analogous to the distance traveled by a car in a city, where the streets are laid out in a rectangular grid (such as city blocks). In Euclidean distance, the measurement is made in a straight line. Using Equation 4-6, the distance from (1, 1) to (4, 5) would be $|1 - 4| + |1 - 5| = 7$. From an optimization perspective, if there is a need to use the Manhattan distance for a clustering analysis, the median is a better choice for the centroid than use of the mean [2].

K-means clustering is applicable to objects that can be described by attributes that are numerical with a meaningful distance measure. From Chapter 3, interval and ratio attribute types can certainly be used. However, k-means does not handle categorical variables well. For example, suppose a clustering analysis is to be conducted on new car sales. Among other attributes, such as the sale price, the color of the car is considered important. Although one could assign numerical values to the color, such as red = 1, yellow = 2, and green = 3, it is not useful to consider that yellow is as close to red as yellow is to green from a clustering perspective. In such cases, it may be necessary to use an alternative clustering methodology. Such methods are described in the next section.

4.3 Additional Algorithms

The k-means clustering method is easily applied to numeric data where the concept of distance can naturally be applied. However, it may be necessary or desirable to use an alternative clustering algorithm. As discussed at the end of the previous section, k-means does not handle categorical data. In such cases, k-modes [3] is a commonly used method for clustering categorical data based on the number of differences in the respective components of the attributes. For example, if each object has four attributes, the distance from (a, b, e, d) to (d, d, d, d) is 3. In R, the function `kmode()` is implemented in the `klaR` package.

Because k-means and k-modes divide the entire dataset into distinct groups, both approaches are considered partitioning methods. A third partitioning method is known as Partitioning around Medoids (PAM) [4]. In general, a medoid is a representative object in a set of objects. In clustering, the ***medoids*** are the objects in each cluster that minimize the sum of the distances from the medoid to the other objects in the cluster. The advantage of using PAM is that the "center" of each cluster is an actual object in the dataset. PAM is implemented in R by the `pam()` function included in the `cluster` R package. The `fpc` R package includes a function `pamk()`, which uses the `pam()` function to find the optimal value for k.

Other clustering methods include hierarchical agglomerative clustering and density clustering methods. In hierarchical agglomerative clustering, each object is initially placed in its own cluster. The clusters are then combined with the most similar cluster. This process is repeated until one cluster, which includes all the objects, exists. The R `stats` package includes the `hclust()` function for performing hierarchical agglomerative clustering. In density-based clustering methods, the clusters are identified by the concentration of points. The `fpc` R package includes a function, `dbscan()`, to perform density-based clustering analysis. Density-based clustering can be useful to identify irregularly shaped clusters.

Summary

Clustering analysis groups similar objects based on the objects' attributes. Clustering is applied in areas such as marketing, economics, biology, and medicine. This chapter presented a detailed explanation of the k-means algorithm and its implementation in R. To use k-means properly, it is important to do the following:

- Properly scale the attribute values to prevent certain attributes from dominating the other attributes.
- Ensure that the concept of distance between the assigned values within an attribute is meaningful.
- Choose the number of clusters, k, such that the sum of the Within Sum of Squares (WSS) of the distances is reasonably minimized. A plot such as the example in Figure 4-5 can be helpful in this respect.

If k-means does not appear to be an appropriate clustering technique for a given dataset, then alternative techniques such as k-modes or PAM should be considered.

Once the clusters are identified, it is often useful to label these clusters in some descriptive way. Especially when dealing with upper management, these labels are useful to easily communicate the findings of the clustering analysis. In clustering, the labels are not preassigned to each object. The labels are subjectively assigned after the clusters have been identified. Chapter 7 considers several methods to perform the classification of objects with predetermined labels. Clustering can be used with other analytical techniques, such as regression. Linear regression and logistic regression are covered in Chapter 6, "Advanced Analytical Theory and Methods: Regression."

Exercises

1. Using the age and height clustering example in section 4.2.5, algebraically illustrate the impact on the measured distance when the height is expressed in meters rather than centimeters. Explain why different clusters will result depending on the choice of units for the patient's height.
2. Compare and contrast five clustering algorithms, assigned by the instructor or selected by the student.

3. Using the `ruspini` dataset provided with the `cluster` package in R, perform a k-means analysis. Document the findings and justify the choice of k. Hint: use `data(ruspini)` to load the dataset into the R workspace.

Bibliography

[1] J. MacQueen, "Some Methods for Classification and Analysis of Multivariate Observations," in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability,* Berkeley, CA, 1967.

[2] P.-N. Tan, V. Kumar, and M. Steinbach, *Introduction to Data Mining*, Upper Saddle River, NJ: Person, 2013.

[3] Z. Huang, "A Fast Clustering Algorithm to Cluster Very Large Categorical Data Sets in Data Mining," 1997. [Online]. Available: `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.134.83&rep=rep1&type=pdf`. [Accessed 13 March 2014].

[4] L. Kaufman and P. J. Rousseeuw, "Partitioning Around Medoids (Program PAM)," in *Finding Groups in Data: An Introduction to Cluster Analysis,* Hoboken, NJ, John Wiley & Sons, Inc, 2008, p. 68-125, Chapter 2.

5

Advanced Analytical Theory and Methods: Association Rules

Key Concepts

Association rules
Apriori algorithm
Support
Confidence
Lift
Leverage

This chapter discusses an unsupervised learning method called association rules. This is a descriptive, not predictive, method often used to discover interesting relationships hidden in a large dataset. The disclosed relationships can be represented as rules or frequent itemsets. Association rules are commonly used for mining transactions in databases.

Here are some possible questions that association rules can answer:

- Which products tend to be purchased together?
- Of those customers who are similar to this person, what products do they tend to buy?
- Of those customers who have purchased this product, what other similar products do they tend to view or purchase?

5.1 Overview

Figure 5-1 shows the general logic behind association rules. Given a large collection of transactions (depicted as three stacks of receipts in the figure), in which each transaction consists of one or more items, association rules go through the items being purchased to see what items are frequently bought together and to discover a list of rules that describe the purchasing behavior. The goal with association rules is to discover interesting relationships among the items. (The relationship occurs too frequently to be random and is meaningful from a business perspective, which may or may not be obvious.) The relationships that are interesting depend both on the business context and the nature of the algorithm being used for the discovery.

FIGURE 5-1 *The general logic behind association rules*

Each of the uncovered rules is in the form X → Y, meaning that when item X is observed, item Y is also observed. In this case, the left-hand side (LHS) of the rule is X, and the right-hand side (RHS) of the rule is Y.

Using association rules, patterns can be discovered from the data that allow the association rule algorithms to disclose rules of related product purchases. The uncovered rules are listed on the right side of

Figure 5-1. The first three rules suggest that when cereal is purchased, 90% of the time milk is purchased also. When bread is purchased, 40% of the time milk is purchased also. When milk is purchased, 23% of the time cereal is also purchased.

In the example of a retail store, association rules are used over transactions that consist of one or more items. In fact, because of their popularity in mining customer transactions, association rules are sometimes referred to as ***market basket analysis***. Each transaction can be viewed as the shopping basket of a customer that contains one or more items. This is also known as an itemset. The term ***itemset*** refers to a collection of items or individual entities that contain some kind of relationship. This could be a set of retail items purchased together in one transaction, a set of hyperlinks clicked on by one user in a single session, or a set of tasks done in one day. An itemset containing ***k*** items is called a ***k-itemset***. This chapter uses curly braces like $\{item\ 1, item\ 2, \ldots item\ k\}$ to denote a k-itemset. Computation of the association rules is typically based on itemsets.

The research of association rules started as early as the 1960s. Early research by Hájek et al. [1] introduced many of the key concepts and approaches of association rule learning, but it focused on the mathematical representation rather than the algorithm. The framework of association rule learning was brought into the database community by Agrawal et al. [2] in the early 1990s for discovering regularities between products in a large database of customer transactions recorded by point-of-sale systems in supermarkets. In later years, it expanded to web contexts, such as mining path traversal patterns [3] and usage patterns [4] to facilitate organization of web pages.

This chapter chooses Apriori as the main focus of the discussion of association rules. Apriori [5] is one of the earliest and the most fundamental algorithms for generating association rules. It pioneered the use of support for pruning the itemsets and controlling the exponential growth of candidate itemsets. Shorter candidate itemsets, which are known to be frequent itemsets, are combined and pruned to generate longer frequent itemsets. This approach eliminates the need for all possible itemsets to be enumerated within the algorithm, since the number of all possible itemsets can become exponentially large.

One major component of Apriori is support. Given an itemset L, the ***support*** [2] of L is the percentage of transactions that contain L. For example, if 80% of all transactions contain itemset $\{bread\}$, then the support of $\{bread\}$ is 0.8. Similarly, if 60% of all transactions contain itemset $\{bread, butter\}$, then the support of $\{bread, butter\}$ is 0.6.

A ***frequent itemset*** has items that appear together often enough. The term "often enough" is formally defined with a ***minimum support*** criterion. If the minimum support is set at 0.5, any itemset can be considered a frequent itemset if at least 50% of the transactions contain this itemset. In other words, the support of a frequent itemset should be greater than or equal to the minimum support. For the previous example, both $\{bread\}$ and $\{bread, butter\}$ are considered frequent itemsets at the minimum support 0.5. If the minimum support is 0.7, only $\{bread\}$ is considered a frequent itemset.

If an itemset is considered frequent, then any subset of the frequent itemset must also be frequent. This is referred to as the ***Apriori property*** (or ***downward closure property***). For example, if 60% of the transactions contain $\{bread, jam\}$, then at least 60% of all the transactions will contain $\{bread\}$ or $\{jam\}$. In other words, when the support of $\{bread, jam\}$ is 0.6, the support of $\{bread\}$ or $\{jam\}$ is at least 0.6. Figure 5-2 illustrates how the Apriori property works. If itemset $\{B, C, D\}$ is frequent, then all the subsets of this itemset, shaded, must also be frequent itemsets. The Apriori property provides the basis for the Apriori algorithm.

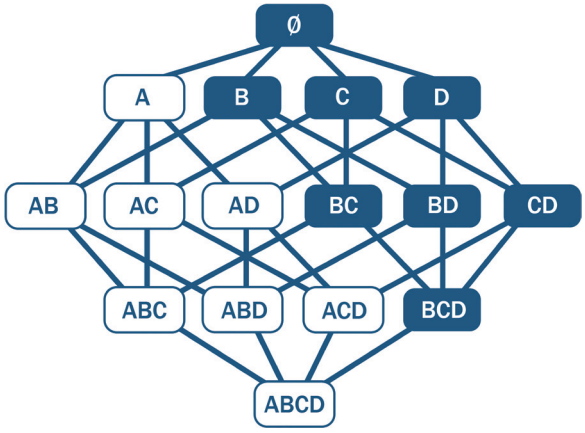

FIGURE 5-2 *Itemset {A,B,C,D} and its subsets*

5.2 Apriori Algorithm

The Apriori algorithm takes a bottom-up iterative approach to uncovering the frequent itemsets by first determining all the possible items (or 1-itemsets, for example `{bread}`, `{eggs}`, `{milk}`, …) and then identifying which among them are frequent.

Assuming the minimum support threshold (or the minimum support criterion) is set at 0.5, the algorithm identifies and retains those itemsets that appear in at least 50% of all transactions and discards (or "prunes away") the itemsets that have a support less than 0.5 or appear in fewer than 50% of the transactions. The word ***prune*** is used like it would be in gardening, where unwanted branches of a bush are clipped away.

In the next iteration of the Apriori algorithm, the identified frequent 1-itemsets are paired into 2-itemsets (for example, `{bread,eggs}`, `{bread,milk}`, `{eggs,milk}`, …) and again evaluated to identify the frequent 2-itemsets among them.

At each iteration, the algorithm checks whether the support criterion can be met; if it can, the algorithm grows the itemset, repeating the process until it runs out of support or until the itemsets reach a predefined length. The Apriori algorithm [5] is given next. Let variable C_k be the set of candidate k-itemsets and variable L_k be the set of k-itemsets that satisfy the minimum support. Given a transaction database D, a minimum support threshold δ, and an optional parameter N indicating the maximum length an itemset could reach, Apriori iteratively computes frequent itemsets L_{k+1} based on L_k.

```
1  Apriori (D, δ, N)
2    k ← 1
3    L_k ← {1-itemsets that satisfy minimum support δ}
4    while L_k ≠ ∅
5      if ∄N ∨ (∃N ∧ k < N)
```

```
6        C_{k+1} ← candidate itemsets generated from L_k
7        for each transaction t in database D do
8          increment the counts of C_{k+1} contained in t
9        L_{k+1} ← candidates in C_{k+1} that satisfy minimum support δ
10       k ← k+1
11   return ⋃_k L_k
```

The first step of the Apriori algorithm is to identify the frequent itemsets by starting with each item in the transactions that meets the predefined minimum support threshold δ. These itemsets are 1-itemsets denoted as L_1, as each 1-itemset contains only one item. Next, the algorithm grows the itemsets by joining L_1 onto itself to form new, grown 2-itemsets denoted as L_2 and determines the support of each 2-itemset in L_2. Those itemsets that do not meet the minimum support threshold δ are pruned away. The growing and pruning process is repeated until no itemsets meet the minimum support threshold. Optionally, a threshold N can be set up to specify the maximum number of items the itemset can reach or the maximum number of iterations of the algorithm. Once completed, output of the Apriori algorithm is the collection of all the frequent k-itemsets.

Next, a collection of candidate rules is formed based on the frequent itemsets uncovered in the iterative process described earlier. For example, a frequent itemset {milk, eggs} may suggest candidate rules {milk}→{eggs} and {eggs}→{milk}.

5.3 Evaluation of Candidate Rules

Frequent itemsets from the previous section can form candidate rules such as X implies Y (X → Y). This section discusses how measures such as confidence, lift, and leverage can help evaluate the appropriateness of these candidate rules.

Confidence [2] is defined as the measure of certainty or trustworthiness associated with each discovered rule. Mathematically, confidence is the percent of transactions that contain both X and Y out of all the transactions that contain X (see Equation 5-1).

$$Confidence(X \rightarrow Y) = \frac{Support(X \wedge Y)}{Support(X)} \tag{5-1}$$

For example, if {bread, eggs, milk} has a support of 0.15 and {bread, eggs} also has a support of 0.15, the confidence of rule {bread, eggs}→{milk} is 1, which means 100% of the time a customer buys bread and eggs, milk is bought as well. The rule is therefore correct for 100% of the transactions containing bread and eggs.

A relationship may be thought of as interesting when the algorithm identifies the relationship with a measure of confidence greater than or equal to a predefined threshold. This predefined threshold is called the ***minimum confidence***. A higher confidence indicates that the rule (X → Y) is more interesting or more trustworthy, based on the sample dataset.

So far, this chapter has talked about two common measures that the Apriori algorithm uses: support and confidence. All the rules can be ranked based on these two measures to filter out the uninteresting rules and retain the interesting ones.

Even though confidence can identify the interesting rules from all the candidate rules, it comes with a problem. Given rules in the form of $X \rightarrow Y$, confidence considers only the antecedent (X) and the co-occurrence of X and Y; it does not take the consequent of the rule (Y) into concern. Therefore, confidence cannot tell if a rule contains true implication of the relationship or if the rule is purely coincidental. X and Y can be statistically independent yet still receive a high confidence score. Other measures such as lift [6] and leverage [7] are designed to address this issue.

Lift measures how many times more often X and Y occur together than expected if they are statistically independent of each other. Lift is a measure [6] of how X and Y are really related rather than coincidentally happening together (see Equation 5-2).

$$Lift(X \rightarrow Y) = \frac{Support(X \wedge Y)}{Support(X) * Support(Y)} \tag{5-2}$$

Lift is 1 if X and Y are statistically independent of each other. In contrast, a lift of $X \rightarrow Y$ greater than 1 indicates that there is some usefulness to the rule. A larger value of lift suggests a greater strength of the association between X and Y.

Assuming 1,000 transactions, with $\{\texttt{milk}, \texttt{eggs}\}$ appearing in 300 of them, $\{\texttt{milk}\}$ appearing in 500, and $\{\texttt{eggs}\}$ appearing in 400, then $Lift(milk \rightarrow eggs) = 0.3/(0.5 * 0.4) = 1.5$. If $\{\texttt{bread}\}$ appears in 400 transactions and $\{\texttt{milk}, \texttt{bread}\}$ appears in 400, then $Lift(milk \rightarrow bread) = 0.4/(0.5 * 0.4) = 2$. Therefore it can be concluded that milk and bread have a stronger association than milk and eggs.

Leverage [7] is a similar notion, but instead of using a ratio, leverage uses the difference (see Equation 5-3). Leverage measures the difference in the probability of X and Y appearing together in the dataset compared to what would be expected if X and Y were statistically independent of each other.

$$Leverage(X \rightarrow Y) = Support(X \wedge Y) - Support(X) * Support(Y) \tag{5-3}$$

In theory, leverage is 0 when X and Y are statistically independent of each other. If X and Y have some kind of relationship, the leverage would be greater than zero. A larger leverage value indicates a stronger relationship between X and Y. For the previous example, $Leverage(milk \rightarrow eggs) = 0.3 - (0.5 * 0.4) = 0.1$ and $Leverage(milk \rightarrow bread) = 0.4 - (0.5 * 0.4) = 0.2$. It again confirms that milk and bread have a stronger association than milk and eggs.

Confidence is able to identify trustworthy rules, but it cannot tell whether a rule is coincidental. A high-confidence rule can sometimes be misleading because confidence does not consider support of the itemset in the rule consequent. Measures such as lift and leverage not only ensure interesting rules are identified but also filter out the coincidental rules.

This chapter has discussed four measures of significance and interestingness for association rules: support, confidence, lift, and leverage. These measures ensure the discovery of interesting and strong rules from sample datasets. Besides these four rules, there are other alternative measures, such as correlation [8], collective strength [9], conviction [6], and coverage [10]. Refer to the Bibliography to learn how these measures work.

5.4 Applications of Association Rules

The term ***market basket analysis*** refers to a specific implementation of association rules mining that many companies use for a variety of purposes, including these:

- Broad-scale approaches to better merchandising—what products should be included in or excluded from the inventory each month
- Cross-merchandising between products and high-margin or high-ticket items
- Physical or logical placement of product within related categories of products
- Promotional programs—multiple product purchase incentives managed through a loyalty card program

Besides market basket analysis, association rules are commonly used for recommender systems [11] and clickstream analysis [12].

Many online service providers such as Amazon and Netflix use recommender systems. Recommender systems can use association rules to discover related products or identify customers who have similar interests. For example, association rules may suggest that those customers who have bought product A have also bought product B, or those customers who have bought products A, B, and C are more similar to this customer. These findings provide opportunities for retailers to cross-sell their products.

Clickstream analysis refers to the analytics on data related to web browsing and user clicks, which is stored on the client or the server side. Web usage log files generated on web servers contain huge amounts of information, and association rules can potentially give useful knowledge to web usage data analysts. For example, association rules may suggest that website visitors who land on page X click on links A, B, and C much more often than links D, E, and F. This observation provides valuable insight on how to better personalize and recommend the content to site visitors.

The next section shows an example of grocery store transactions and demonstrates how to use R to perform association rule mining.

5.5 An Example: Transactions in a Grocery Store

An example illustrates the application of the Apriori algorithm to a relatively simple case that generalizes to those used in practice. Using R and the `arules` and `arulesViz` packages, this example shows how to use the Apriori algorithm to generate frequent itemsets and rules and to evaluate and visualize the rules.

The following commands install these two packages and import them into the current R workspace:

```
install.packages('arules')
install.packages('arulesViz')

library('arules')
library('arulesViz')
```

5.5.1 The Groceries Dataset

The example uses the *Groceries* dataset from the R `arules` package. The *Groceries* dataset is collected from 30 days of real-world point-of-sale transactions of a grocery store. The dataset contains 9,835 transactions, and the items are aggregated into 169 categories.

```
data(Groceries)
Groceries
transactions in sparse format with
 9835 transactions (rows) and
 169 items (columns)
```

The summary shows that the most frequent items in the dataset include items such as whole milk, other vegetables, rolls/buns, soda, and yogurt. These items are purchased more often than the others.

```
summary(Groceries)
transactions as itemMatrix in sparse format with
 9835 rows (elements/itemsets/transactions) and
 169 columns (items) and a density of 0.02609146

most frequent items:
      whole milk other vegetables       rolls/buns             soda
            2513             1903             1809             1715
          yogurt          (Other)
            1372            34055

element (itemset/transaction) length distribution:
sizes
   1    2    3    4    5    6    7    8    9   10   11   12   13   14
2159 1643 1299 1005  855  645  545  438  350  246  182  117   78   77
  15   16   17   18   19   20   21   22   23   24   26   27   28   29
  55   46   29   14   14    9   11    4    6    1    1    1    1    3
  32
   1

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   2.000   3.000   4.409   6.000  32.000

includes extended item information - examples:
       labels  level2           level1
1 frankfurter sausage meet and sausage
2     sausage sausage meet and sausage
3  liver loaf sausage meet and sausage
```

The class of the dataset is `transactions`, as defined by the `arules` package. The `transactions` class contains three slots:

- **`transactionInfo`:** A data frame with vectors of the same length as the number of transactions
- **`itemInfo`:** A data frame to store item labels
- **`data`:** A binary incidence matrix that indicates which item labels appear in every transaction

```
class(Groceries)
[1] "transactions"
attr(,"package")
[1] "arules"
```

For the *Groceries* dataset, the `transactionInfo` is not being used. Enter `Groceries@itemInfo` to display all 169 grocery labels as well as their categories. The following command displays only the first 20 grocery labels. Each grocery label is mapped to two levels of categories—`level2` and `level1`—where `level1` is a superset of `level2`. For example, grocery label `sausage` belongs to the `sausage` category in `level2`, and it is part of the `meat and sausage` category in `level1`. (Note that "`meet`" in `level1` is a typo in the dataset.)

```
Groceries@itemInfo[1:20,]
              labels     level2              level1
1        frankfurter    sausage     meet and sausage
2            sausage    sausage     meet and sausage
3         liver loaf    sausage     meet and sausage
4                ham    sausage     meet and sausage
5               meat    sausage     meet and sausage
6  finished products    sausage     meet and sausage
7    organic sausage    sausage     meet and sausage
8            chicken    poultry     meet and sausage
9             turkey    poultry     meet and sausage
10              pork       pork     meet and sausage
11              beef       beef     meet and sausage
12    hamburger meat       beef     meet and sausage
13              fish       fish     meet and sausage
14      citrus fruit      fruit fruit and vegetables
15    tropical fruit      fruit fruit and vegetables
16         pip fruit      fruit fruit and vegetables
17            grapes      fruit fruit and vegetables
18           berries      fruit fruit and vegetables
19       nuts/prunes      fruit fruit and vegetables
20   root vegetables vegetables fruit and vegetables
```

The following code displays the 10th to 20th transactions of the `Groceries` dataset. The `[10:20]` can be changed to `[1:9835]` to display all the transactions.

```
apply(Groceries@data[,10:20], 2,
      function(r) paste(Groceries@itemInfo[r,"labels"], collapse=", ")
      )
```

Each row in the output shows a transaction that includes one or more products, and each transaction corresponds to everything in a customer's shopping cart. For example, in the first transaction, a customer has purchased whole milk and cereals.

```
 [1] "whole milk, cereals"
 [2] "tropical fruit, other vegetables, white bread, bottled water,
chocolate"
 [3] "citrus fruit, tropical fruit, whole milk, butter, curd, yogurt,
flour, bottled water, dishes"
 [4] "beef"
```

```
 [5] "frankfurter, rolls/buns, soda"
 [6] "chicken, tropical fruit"
 [7] "butter, sugar, fruit/vegetable juice, newspapers"
 [8] "fruit/vegetable juice"
 [9] "packaged fruit/vegetables"
[10] "chocolate"
[11] "specialty bar"
```

The next section shows how to generate frequent itemsets from the *Groceries* dataset.

5.5.2 Frequent Itemset Generation

The `apriori()` function from the `arule` package implements the Apriori algorithm to create frequent itemsets. Note that, by default, the `apriori()` function executes all the iterations at once. However, to illustrate how the Apriori algorithm works, the code examples in this section manually set the parameters of the `apriori()` function to simulate each iteration of the algorithm.

Assume that the minimum support threshold is set to 0.02 based on management discretion. Because the dataset contains 9,853 transactions, an itemset should appear at least 198 times to be considered a frequent itemset. The first iteration of the Apriori algorithm computes the support of each product in the dataset and retains those products that satisfy the minimum support. The following code identifies 59 frequent 1-itemsets that satisfy the minimum support. The parameters of `apriori()` specify the minimum and maximum lengths of the itemsets, the minimum support threshold, and the target indicating the type of association mined.

```
itemsets <- apriori(Groceries, parameter=list(minlen=1, maxlen=1,
                      support=0.02, target="frequent itemsets"))

parameter specification:
 confidence minval smax arem  aval originalSupport support minlen
        0.8    0.1    1 none FALSE            TRUE    0.02      1
 maxlen            target   ext
      1 frequent itemsets FALSE

algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

apriori - find association rules with the apriori algorithm
version 4.21 (2004.05.09)        (c) 1996-2004   Christian Borgelt
set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
sorting and recoding items ... [59 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 done [0.00s].
writing ... [59 set(s)] done [0.00s].
creating S4 object  ... done [0.00s].
```

The summary of the itemsets shows that the support of 1-itemsets ranges from 0.02105 to 0.25552. Because the maximum support of the 1-itemsets in the dataset is only 0.25552, to enable the discovery of interesting rules, the minimum support threshold should not be set too close to that number.

```
summary(itemsets)
set of 59 itemsets

most frequent items:
frankfurter     sausage         ham        meat     chicken
          1           1           1           1           1
    (Other)
         54

element (itemset/transaction) length distribution:sizes
 1
59

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      1       1       1       1       1       1

summary of quality measures:
    support
 Min.   :0.02105
 1st Qu.:0.03015
 Median :0.04809
 Mean   :0.06200
 3rd Qu.:0.07666
 Max.   :0.25552

includes transaction ID lists: FALSE

mining info:
      data ntransactions support confidence
 Groceries          9835    0.02          1
```

The following code uses the `inspect()` function to display the top 10 frequent 1-itemsets sorted by their support. Of all the transaction records, the 59 1-itemsets such as {`whole milk`}, {`other vegetables`}, {`rolls/buns`}, {`soda`}, and {`yogurt`} all satisfy the minimum support. Therefore, they are called frequent 1-itemsets.

```
inspect(head(sort(itemsets, by = "support"), 10))
   items                      support
1  {whole milk}           0.25551601
2  {other vegetables}     0.19349263
3  {rolls/buns}           0.18393493
4  {soda}                 0.17437722
5  {yogurt}               0.13950178
6  {bottled water}        0.11052364
```

```
7  {root vegetables}       0.10899847
8  {tropical fruit}        0.10493137
9  {shopping bags}         0.09852567
10 {sausage}               0.09395018
```

In the next iteration, the list of frequent 1-itemsets is joined onto itself to form all possible candidate 2-itemsets. For example, 1-itemsets {whole milk} and {soda} would be joined to become a 2-itemset {whole milk,soda}. The algorithm computes the support of each candidate 2-itemset and retains those that satisfy the minimum support. The output that follows shows that 61 frequent 2-itemsets have been identified.

```
itemsets <- apriori(Groceries, parameter=list(minlen=2, maxlen=2,
                      support=0.02, target="frequent itemsets"))

parameter specification:
 confidence minval smax arem  aval originalSupport support minlen
        0.8    0.1    1 none FALSE            TRUE    0.02      2
 maxlen            target   ext
      2 frequent itemsets FALSE

algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

apriori - find association rules with the apriori algorithm
version 4.21 (2004.05.09)        (c) 1996-2004   Christian Borgelt
set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
sorting and recoding items ... [59 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 done [0.00s].
writing ... [61 set(s)] done [0.00s].
creating S4 object  ... done [0.00s].
```

The summary of the itemsets shows that the support of 2-itemsets ranges from 0.02003 to 0.07483.

```
summary(itemsets)
set of 61 itemsets

most frequent items:
      whole milk other vegetables           yogurt       rolls/buns
              25               17                9                9
            soda          (Other)
               9               53

element (itemset/transaction) length distribution:sizes
 2
61
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      2       2       2       2       2       2

summary of quality measures:
    support
 Min.   :0.02003
 1st Qu.:0.02227
 Median :0.02613
 Mean   :0.02951
 3rd Qu.:0.03223
 Max.   :0.07483

includes transaction ID lists: FALSE

mining info:
      data ntransactions support confidence
 Groceries          9835    0.02          1
```

The top 10 most frequent 2-itemsets are displayed next, sorted by their support. Notice that whole milk appears six times in the top 10 2-itemsets ranked by support. As seen earlier, {whole milk} has the highest support among all the 1-itemsets. These top 10 2-itemsets with the highest support may not be interesting; this highlights the limitations of using support alone.

```
inspect(head(sort(itemsets, by ="support"),10))
   items                  support
1  {other vegetables,
    whole milk}       0.07483477
2  {whole milk,
    rolls/buns}       0.05663447
3  {whole milk,
    yogurt}           0.05602440
4  {root vegetables,
    whole milk}       0.04890696
5  {root vegetables,
    other vegetables} 0.04738180
6  {other vegetables,
    yogurt}           0.04341637
7  {other vegetables,
    rolls/buns}       0.04260295
8  {tropical fruit,
    whole milk}       0.04229792
9  {whole milk,
    soda}             0.04006101
10 {rolls/buns,
    soda}             0.03833249
```

Next, the list of frequent 2-itemsets is joined onto itself to form candidate 3-itemsets. For example {other vegetables,whole milk} and {whole milk,rolls/buns} would be joined as {other vegetables,whole milk,rolls/buns}. The algorithm retains those itemsets

that satisfy the minimum support. The following output shows that only two frequent 3-itemsets have been identified.

```
itemsets <- apriori(Groceries, parameter=list(minlen=3, maxlen=3,
                      support=0.02, target="frequent itemsets"))

parameter specification:
 confidence minval smax arem  aval originalSupport support minlen
        0.8    0.1    1 none FALSE            TRUE    0.02      3
 maxlen            target   ext
      3 frequent itemsets FALSE

algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

apriori - find association rules with the apriori algorithm
version 4.21 (2004.05.09)        (c) 1996-2004   Christian Borgelt
set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
sorting and recoding items ... [59 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 done [0.00s].
writing ... [2 set(s)] done [0.00s].
creating S4 object  ... done [0.00s].
```

The 3-itemsets are displayed next:

```
inspect(sort(itemsets, by ="support"))
  items                 support
1 {root vegetables,
   other vegetables,
   whole milk}       0.02318251
2 {other vegetables,
   whole milk,
   yogurt}           0.02226741
```

In the next iteration, there is only one candidate 4-itemset `{root vegetables,other vegetables,whole milk,yogurt}`, and its support is below 0.02. No frequent 4-itemsets have been found, and the algorithm converges.

```
itemsets <- apriori(Groceries, parameter=list(minlen=4, maxlen=4,
                      support=0.02, target="frequent itemsets"))

parameter specification:
 confidence minval smax arem  aval originalSupport support minlen
        0.8    0.1    1 none FALSE            TRUE    0.02      4
 maxlen            target   ext
      4 frequent itemsets FALSE
```

```
algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

apriori - find association rules with the apriori algorithm
version 4.21 (2004.05.09)        (c) 1996-2004   Christian Borgelt
set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
sorting and recoding items ... [59 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 done [0.00s].
writing ... [0 set(s)] done [0.00s].
creating S4 object  ... done [0.00s].
```

The previous steps simulate the Apriori algorithm at each iteration. For the *Groceries* dataset, the iterations run out of support when $k = 4$. Therefore, the frequent itemsets contain 59 frequent 1-itemsets, 61 frequent 2-itemsets, and 2 frequent 3-itemsets.

When the `maxlen` parameter is not set, the algorithm continues each iteration until it runs out of support or until k reaches the default `maxlen=10`. As shown in the code output that follows, 122 frequent itemsets have been identified. This matches the total number of 59 frequent 1-itemsets, 61 frequent 2-itemsets, and 2 frequent 3-itemsets.

```
itemsets <- apriori(Groceries, parameter=list(minlen=1, support=0.02,
                                        target="frequent itemsets"))

parameter specification:
 confidence minval smax arem  aval originalSupport support minlen
        0.8    0.1    1 none FALSE            TRUE    0.02      1
 maxlen            target   ext
     10 frequent itemsets FALSE

algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

apriori - find association rules with the apriori algorithm
version 4.21 (2004.05.09)        (c) 1996-2004   Christian Borgelt
set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
sorting and recoding items ... [59 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 done [0.00s].
writing ... [122 set(s)] done [0.00s].
creating S4 object  ... done [0.00s].
```

Note that the results are assessed based on the specific business context of the exercise using the specific dataset. If the dataset changes or a different minimum support threshold is chosen, the Apriori algorithm must run each iteration again to retrieve the updated frequent itemsets.

5.5.3 Rule Generation and Visualization

The `apriori()` function can also be used to generate rules. Assume that the minimum support threshold is now set to a lower value 0.001, and the minimum confidence threshold is set to 0.6. A lower minimum support threshold allows more rules to show up. The following code creates 2,918 rules from all the transactions in the *Groceries* dataset that satisfy both the minimum support and the minimum confidence.

```
rules <- apriori(Groceries, parameter=list(support=0.001,
                       confidence=0.6, target = "rules"))

parameter specification:
 confidence minval smax arem  aval originalSupport support minlen
        0.6    0.1    1 none FALSE            TRUE   0.001      1
 maxlen target   ext
     10  rules FALSE

algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

apriori - find association rules with the apriori algorithm
version 4.21 (2004.05.09)        (c) 1996-2004   Christian Borgelt
set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
sorting and recoding items ... [157 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 4 5 6 done [0.01s].
writing ... [2918 rule(s)] done [0.00s].
creating S4 object  ... done [0.01s].
```

The summary of the rules shows the number of rules and ranges of the support, confidence, and lift.

```
summary(rules)
set of 2918 rules

rule length distribution (lhs + rhs):sizes
   2    3    4    5    6
   3  490 1765  626   34

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  2.000   4.000   4.000   4.068   4.000   6.000

summary of quality measures:
    support           confidence          lift
 Min.   :0.001017   Min.   :0.6000   Min.   : 2.348
 1st Qu.:0.001118   1st Qu.:0.6316   1st Qu.: 2.668
 Median :0.001220   Median :0.6818   Median : 3.168
 Mean   :0.001480   Mean   :0.7028   Mean   : 3.450
 3rd Qu.:0.001525   3rd Qu.:0.7500   3rd Qu.: 3.692
 Max.   :0.009354   Max.   :1.0000   Max.   :18.996
```

```
mining info:
      data ntransactions support confidence
 Groceries          9835   0.001        0.6
```

Enter `plot(rules)` to display the scatterplot of the 2,918 rules (Figure 5-3), where the horizontal axis is the support, the vertical axis is the confidence, and the shading is the lift. The scatterplot shows that, of the 2,918 rules generated from the *Groceries* dataset, the highest lift occurs at a low support and a low confidence.

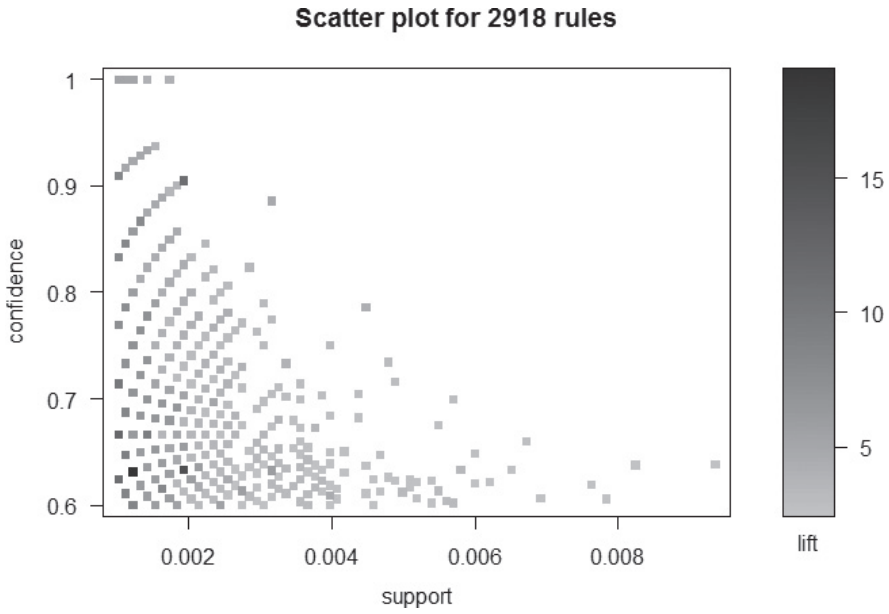

Figure 5-3 *Scatterplot of the 2,918 rules with minimum support 0.001 and minimum confidence 0.6*

Entering `plot(rules@quality)` displays a scatterplot matrix (Figure 5-4) to compare the support, confidence, and lift of the 2,918 rules.

Figure 5-4 shows that lift is proportional to confidence and illustrates several linear groupings. As indicated by Equation 5-2 and Equation 5-3, $Lift = Confidence / Support(Y)$. Therefore, when the support of Y remains the same, lift is proportional to confidence, and the slope of the linear trend is the reciprocal of $Support(Y)$. The following code shows that, of the 2,918 rules, there are only 18 different values for $\frac{1}{Support(Y)}$, and the majority occurs at slopes 3.91, 5.17, 7.17, 9.17, and 9.53. This matches the slopes shown in the third row and second column of Figure 5-4, where the x-axis is the confidence and the y-axis is the lift.

```
# compute the 1/Support(Y)
slope <- sort(round(rules@quality$lift / rules@quality$confidence, 2))
# Display the number of times each slope appears in the dataset
unlist(lapply(split(slope,f=slope),length))
```

```
 3.91  5.17  5.44  5.73  7.17  9.05  9.17  9.53 10.64 12.08
 1585   940    12     7   188     1   102    55      1      4
12.42 13.22 13.83 13.95 18.05 23.76 26.44 30.08
1     5     2     9     3     1     1     1
```

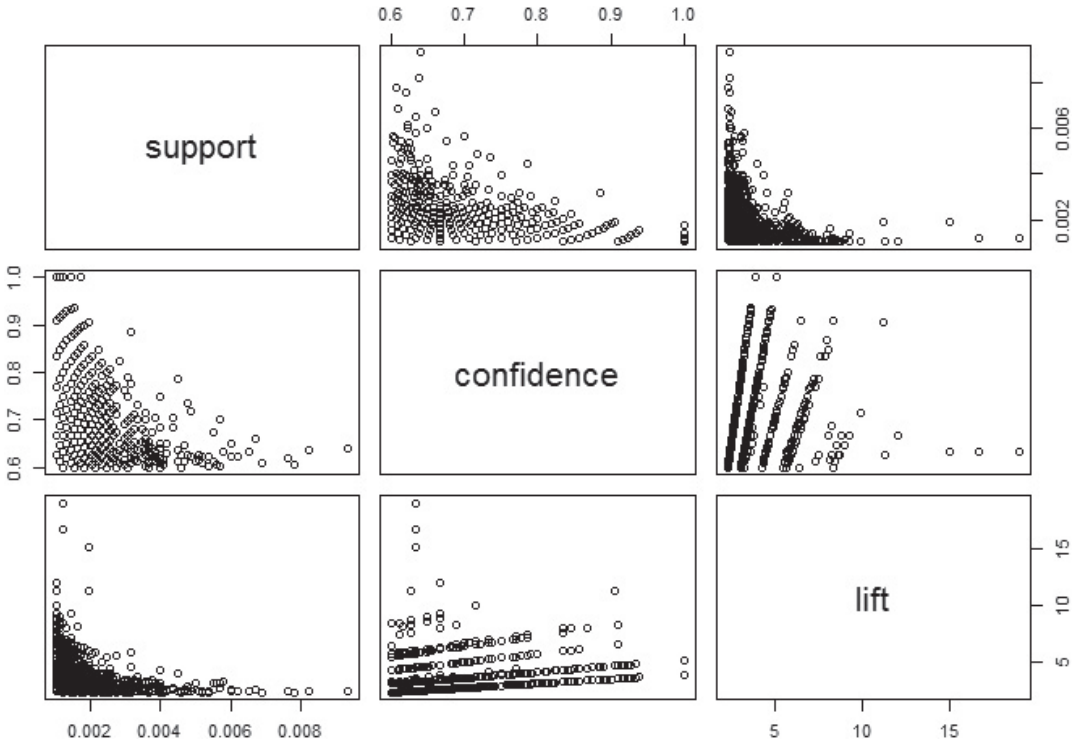

FIGURE 5-4 *Scatterplot matrix on the support, confidence, and lift of the 2,918 rules*

The `inspect()` function can display content of the rules generated previously. The following code shows the top ten rules sorted by the lift. Rule {`Instant food products,soda`}→{`hamburger meat`} has the highest lift of 18.995654.

```
inspect(head(sort(rules, by="lift"), 10))
   lhs                          rhs
support  confidence   lift
1  {Instant food products,
    soda}                  => {hamburger meat}
0.001220132  0.6315789 18.995654
2  {soda,
    popcorn}               => {salty snack}
0.001220132  0.6315789 16.697793
3  {ham,
    processed cheese}      => {white bread}
0.001931876  0.6333333 15.045491
```

```
4  {tropical fruit,
    other vegetables,
    yogurt,
    white bread}           => {butter}
0.001016777  0.6666667 12.030581
5  {hamburger meat,
    yogurt,
    whipped/sour cream}    => {butter}
0.001016777  0.6250000 11.278670
6  {tropical fruit,
    other vegetables,
    whole milk,
    yogurt,
    domestic eggs}         => {butter}
0.001016777  0.6250000 11.278670
7  {liquor,
    red/blush wine}        => {bottled beer}
0.001931876  0.9047619 11.235269
8  {other vegetables,
    butter,
    sugar}                 => {whipped/sour cream}
0.001016777  0.7142857  9.964539
9  {whole milk,
    butter,
    hard cheese}           => {whipped/sour cream}
0.001423488  0.6666667  9.300236
10 {tropical fruit,
    other vegetables,
    butter,
     fruit/vegetable juice} => {whipped/sour cream}
0.001016777  0.6666667  9.300236
```

The following code fetches a total of 127 rules whose confidence is above 0.9:

```
confidentRules <- rules[quality(rules)$confidence > 0.9]
confidentRules
set of 127 rules
```

The next command produces a matrix-based visualization (Figure 5-5) of the LHS versus the RHS of the rules. The legend on the right is a color matrix indicating the lift and the confidence to which each square in the main matrix corresponds.

```
plot(confidentRules, method="matrix", measure=c("lift", "confidence"),
     control=list(reorder=TRUE))
```

As the previous `plot()` command runs, the R console would simultaneously display a distinct list of the LHS and RHS from the 127 rules. A segment of the output is shown here:

```
Itemsets in Antecedent (LHS)
  [1] "{citrus fruit,other vegetables,soda,fruit/vegetable juice}"
  [2] "{tropical fruit,other vegetables,whole milk,yogurt,oil}"
```

```
  [3] "{tropical fruit,butter,whipped/sour cream,fruit/vegetable
juice}"
  [4] "{tropical fruit,grapes,whole milk,yogurt}"
  [5] "{ham,tropical fruit,pip fruit,whole milk}"
...
[124] "{liquor,red/blush wine}"
Itemsets in Consequent (RHS)
[1] "{whole milk}"       "{yogurt}"           "{root vegetables}"
[4] "{bottled beer}"     "{other vegetables}"
```

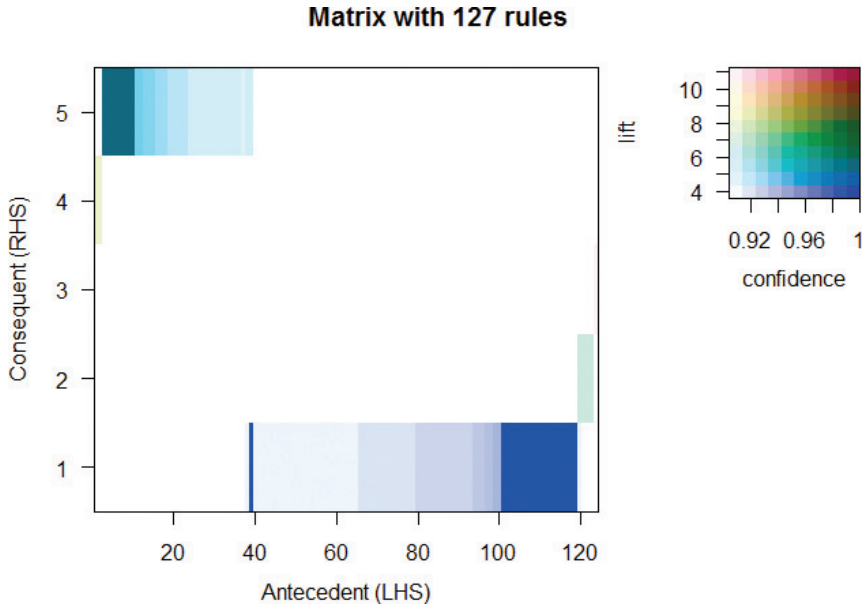

FIGURE 5-5 *Matrix-based visualization of LHS and RHS, colored by lift and confidence*

The following code provides a visualization of the top five rules with the highest lift. The plot is shown in Figure 5-6. In the graph, the arrow always points from an item on the LHS to an item on the RHS. For example, the arrows that connect ham, processed cheese, and white bread suggest rule `{ham,processed cheese}→{white bread}`. The legend on the top right of the graph shows that the size of a circle indicates the support of the rules ranging from 0.001 to 0.002. The color (or shade) represents the lift, which ranges from 11.279 to 18.996. The rule with the highest lift is `{Instant food products,soda} → {hamburger meat}`.

```
highLiftRules <- head(sort(rules, by="lift"), 5)
plot(highLiftRules, method="graph", control=list(type="items"))
```

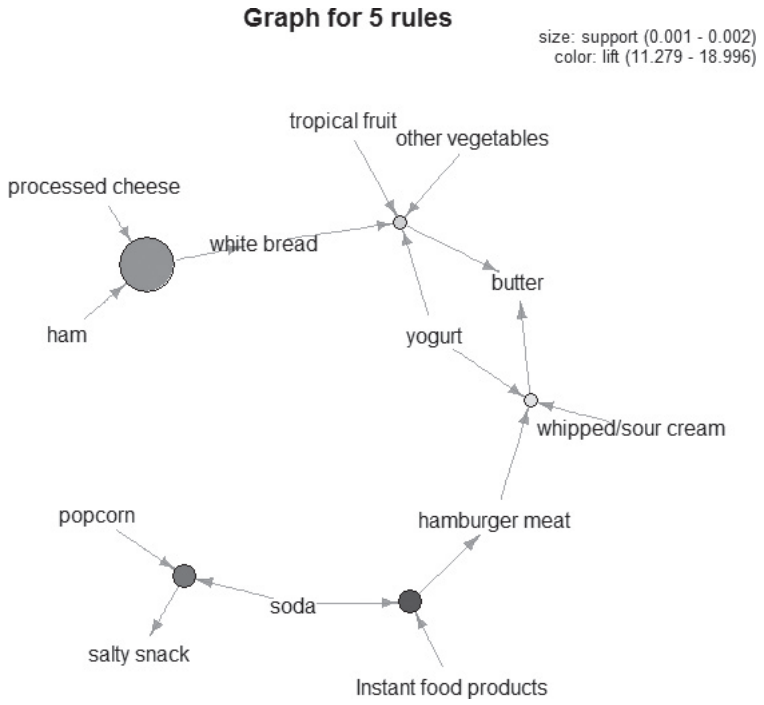

FIGURE 5-6 *Graph visualization of the top five rules sorted by lift*

5.6 Validation and Testing

After gathering the output rules, it may become necessary to use one or more methods to validate the results in the business context for the sample dataset. The first approach can be established through statistical measures such as confidence, lift, and leverage. Rules that involve mutually independent items or cover few transactions are considered uninteresting because they may capture spurious relationships.

As mentioned in Section 5.3, confidence measures the chance that X and Y appear together in relation to the chance X appears. Confidence can be used to identify the interestingness of the rules.

Lift and leverage both compare the support of X and Y against their individual support. While mining data with association rules, some rules generated could be purely coincidental. For example, if 95% of customers buy X and 90% of customers buy Y, then X and Y would occur together at least 85% of the time, even if there is no relationship between the two. Measures like lift and leverage ensure that interesting rules are identified rather than coincidental ones.

Another set of criteria can be established through subjective arguments. Even with a high confidence, a rule may be considered subjectively uninteresting unless it reveals any unexpected profitable actions. For example, rules like $\{\texttt{paper}\}\rightarrow\{\texttt{pencil}\}$ may not be subjectively interesting or meaningful despite high support and confidence values. In contrast, a rule like $\{\texttt{diaper}\}\rightarrow\{\texttt{beer}\}$ that satisfies both minimum support and minimum confidence can be considered subjectively interesting because this rule is

unexpected and may suggest a cross-sell opportunity for the retailer. This incorporation of subjective knowledge into the evaluation of rules can be a difficult task, and it requires collaboration with domain experts. As seen in Chapter 2, "Data Analytics Lifecycle," the domain experts may serve as the business users or the business intelligence analysts as part of the Data Science team. In Phase 5, the team can communicate the results and decide if it is appropriate to operationalize them.

5.7 Diagnostics

Although the Apriori algorithm is easy to understand and implement, some of the rules generated are uninteresting or practically useless. Additionally, some of the rules may be generated due to coincidental relationships between the variables. Measures like confidence, lift, and leverage should be used along with human insights to address this problem.

Another problem with association rules is that, in Phase 3 and 4 of the Data Analytics Lifecycle (Chapter 2), the team must specify the minimum support prior to the model execution, which may lead to too many or too few rules. In related research, a variant of the algorithm [13] can use a predefined target range for the number of rules so that the algorithm can adjust the minimum support accordingly.

Section 5.2 presented the Apriori algorithm, which is one of the earliest and the most fundamental algorithms for generating association rules. The Apriori algorithm reduces the computational workload by only examining itemsets that meet the specified minimum threshold. However, depending on the size of the dataset, the Apriori algorithm can be computationally expensive. For each level of support, the algorithm requires a scan of the entire database to obtain the result. Accordingly, as the database grows, it takes more time to compute in each run. Here are some approaches to improve Apriori's efficiency:

- **Partitioning:** Any itemset that is potentially frequent in a transaction database must be frequent in at least one of the partitions of the transaction database.
- **Sampling:** This extracts a subset of the data with a lower support threshold and uses the subset to perform association rule mining.
- **Transaction reduction:** A transaction that does not contain frequent ***k***-itemsets is useless in subsequent scans and therefore can be ignored.
- **Hash-based itemset counting:** If the corresponding hashing bucket count of a ***k***-itemset is below a certain threshold, the ***k***-itemset cannot be frequent.
- **Dynamic itemset counting:** Only add new candidate itemsets when all of their subsets are estimated to be frequent.

Summary

As an unsupervised analysis technique that uncovers relationships among items, association rules find many uses in activities, including market basket analysis, clickstream analysis, and recommendation engines. Although association rules are not used to predict outcomes or behaviors, they are good at identifying "interesting" relationships within items from a large dataset. Quite often, the disclosed relationships that the association rules suggest do not seem obvious; they, therefore, provide valuable insights for institutions to improve their business operations.

The Apriori algorithm is one of the earliest and most fundamental algorithms for association rules. This chapter used a grocery store example to walk through the steps of Apriori and generate frequent ***k***-itemsets and useful rules for downstream analysis and visualization. A few measures such as support, confidence, lift, and leverage were discussed. These measures together help identify the interesting rules and eliminate the coincidental rules. Finally, the chapter discussed some pros and cons of the Apriori algorithm and highlighted a few methods to improve its efficiency.

Exercises

1. What is the Apriori property?

2. Following is a list of five transactions that include items A, B, C, and D:

   - T1 : { A,B,C }
   - T2 : { A,C }
   - T3 : { B,C }
   - T4 : { A,D }
   - T5 : { A,C,D }

   Which itemsets satisfy the minimum support of 0.5? (Hint: An itemset may include more than one item.)

3. How are interesting rules identified? How are interesting rules distinguished from coincidental rules?

4. A local retailer has a database that stores 10,000 transactions of last summer. After analyzing the data, a data science team has identified the following statistics:

   - {battery} appears in 6,000 transactions.
   - {sunscreen} appears in 5,000 transactions.
   - {sandals} appears in 4,000 transactions.
   - {bowls} appears in 2,000 transactions.
   - {battery,sunscreen} appears in 1,500 transactions.
   - {battery,sandals} appears in 1,000 transactions.
   - {battery,bowls} appears in 250 transactions.
   - {battery,sunscreen,sandals} appears in 600 transactions.

   Answer the following questions:

   a. What are the support values of the preceding itemsets?

   b. Assuming the minimum support is 0.05, which itemsets are considered frequent?

   c. What are the confidence values of {battery}→{sunscreen} and {battery,sunscreen}→{sandals}? Which of the two rules is more interesting?

   d. List all the candidate rules that can be formed from the statistics. Which rules are considered interesting at the minimum confidence 0.25? Out of these interesting rules, which rule is considered the most useful (that is, least coincidental)?

Bibliography

[1] P. Hájek, I. Havel, and M. Chytil, "The GUHA Method of Automatic Hypotheses Determination," *Computing,* vol. 1, no. 4, pp. 293–308, 1966.

[2] R. Agrawal, T. Imieliński, and A. Swami, "Mining Association Rules Between Sets of Items in Large Databases," *SIGMOD '93 Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data,* pp. 207–216, 1993.

[3] M.-S. Chen, J. S. Park, and P. Yu, "Efficient Data Mining for Path Traversal Patterns," *IEEE Transactions on Knowledge and Data Engineering,* vol. 10, no. 2, pp. 209–221, 1998.

[4] R. Cooley, B. Mobasher, and J. Srivastava, "Web Mining: Information and Pattern Discovery on the World Wide Web," *Proceedings of the 9th IEEE International Conference on Tools with Artificial Intelligence,* pp. 558–567, 1997.

[5] R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules in Large Databases," in *Proceedings of the 20th International Conference on Very Large Data Bases*, San Francisco, CA, USA, 1994.

[6] S. Brin, R. Motwani, J. D. Ullman, and S. Tsur, "Dynamic Itemset Counting and Implication Rules for Market Basket Data," *SIGMOD,* vol. 26, no. 2, pp. 255–264, 1997.

[7] G. Piatetsky-Shapiro, "Discovery, Analysis and Presentation of Strong Rules," *Knowledge Discovery in Databases,* pp. 229–248, 1991.

[8] S. Brin, R. Motwani, and C. Silverstein, "Beyond Market Baskets: Generalizing Association Rules to Correlations," *Proceedings of the ACM SIGMOD/PODS '97 Joint Conference,* vol. 26, no. 2, pp. 265–276, 1997.

[9] C. C. Aggarwal and P. S. Yu, "A New Framework for Itemset Generation," in *Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS '98)*, Seattle, Washington, USA, 1998.

[10] M. Hahsler, "A Comparison of Commonly Used Interest Measures for Association Rules," 9 March 2011. [Online]. Available: `http://michael.hahsler.net/research/association_rules/measures.html`. [Accessed 4 March 2014].

[11] W. Lin, S. A. Alvarez, and C. Ruiz, "Efficient Adaptive-Support Association Rule Mining for Recommender Systems," *Data Mining and Knowledge Discovery,* vol. 6, no. 1, pp. 83–105, 2002.

[12] B. Mobasher, H. Dai, T. Luo, and M. Nakagawa, "Effective Personalization Based on Association Rule Discovery from Web Usage Data," in ***ACM***, 2011.

[13] W. Lin, S. A. Alvarez, and C. Ruiz, "Collaborative Recommendation via Adaptive Association Rule Mining," in *Proceedings of the International Workshop on Web Mining for E-Commerce (WEBKDD)*, Boston, MA, 2000.

6

Advanced Analytical Theory and Methods: Regression

Key Concepts

Categorical Variable
Linear Regression
Logistic Regression
Ordinary Least Squares (OLS)
Receiver Operating Characteristic (ROC) Curve
Residuals

In general, regression analysis attempts to explain the influence that a set of variables has on the outcome of another variable of interest. Often, the outcome variable is called a ***dependent variable*** because the outcome depends on the other variables. These additional variables are sometimes called the ***input variables*** or the ***independent variables***. Regression analysis is useful for answering the following kinds of questions:

- What is a person's expected income?
- What is the probability that an applicant will default on a loan?

Linear regression is a useful tool for answering the first question, and logistic regression is a popular method for addressing the second. This chapter examines these two regression techniques and explains when one technique is more appropriate than the other.

Regression analysis is a useful explanatory tool that can identify the input variables that have the greatest statistical influence on the outcome. With such knowledge and insight, environmental changes can be attempted to produce more favorable values of the input variables. For example, if it is found that the reading level of 10-year-old students is an excellent predictor of the students' success in high school and a factor in their attending college, then additional emphasis on reading can be considered, implemented, and evaluated to improve students' reading levels at a younger age.

6.1 Linear Regression

Linear regression is an analytical technique used to model the relationship between several input variables and a continuous outcome variable. A key assumption is that the relationship between an input variable and the outcome variable is linear. Although this assumption may appear restrictive, it is often possible to properly transform the input or outcome variables to achieve a linear relationship between the modified input and outcome variables. Possible transformations will be covered in more detail later in the chapter.

The physical sciences have well-known linear models, such as Ohm's Law, which states that the electrical current flowing through a resistive circuit is linearly proportional to the voltage applied to the circuit. Such a model is considered deterministic in the sense that if the input values are known, the value of the outcome variable is precisely determined. A linear regression model is a probabilistic one that accounts for the randomness that can affect any particular outcome. Based on known input values, a linear regression model provides the expected value of the outcome variable based on the values of the input variables, but some uncertainty may remain in predicting any particular outcome. Thus, linear regression models are useful in physical and social science applications where there may be considerable variation in a particular outcome based on a given set of input values. After presenting possible linear regression use cases, the foundations of linear regression modeling are provided.

6.1.1 Use Cases

Linear regression is often used in business, government, and other scenarios. Some common practical applications of linear regression in the real world include the following:

- **Real estate:** A simple linear regression analysis can be used to model residential home prices as a function of the home's living area. Such a model helps set or evaluate the list price of a home on the market. The model could be further improved by including other input variables such as number of bathrooms, number of bedrooms, lot size, school district rankings, crime statistics, and property taxes.

- **Demand forecasting:** Businesses and governments can use linear regression models to predict demand for goods and services. For example, restaurant chains can appropriately prepare for the predicted type and quantity of food that customers will consume based upon the weather, the day of the week, whether an item is offered as a special, the time of day, and the reservation volume. Similar models can be built to predict retail sales, emergency room visits, and ambulance dispatches.
- **Medical:** A linear regression model can be used to analyze the effect of a proposed radiation treatment on reducing tumor sizes. Input variables might include duration of a single radiation treatment, frequency of radiation treatment, and patient attributes such as age or weight.

6.1.2 Model Description

As the name of this technique suggests, the linear regression model assumes that there is a linear relationship between the input variables and the outcome variable. This relationship can be expressed as shown in Equation 6-1.

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_{p-1} x_{p-1} + \varepsilon \tag{6-1}$$

where:

y is the outcome variable

x_j are the input variables, for $j = 1, 2, \ldots, p - 1$

β_0 is the value of y when each x_j equals zero

β_j is the change in y based on a unit change in x_j, for $j = 1, 2, \ldots, p - 1$

ε is a random error term that represents the difference in the linear model and a particular observed value for y

Suppose it is desired to build a linear regression model that estimates a person's annual income as a function of two variables—age and education—both expressed in years. In this case, income is the outcome variable, and the input variables are age and education. Although it may be an over generalization, such a model seems intuitively correct in the sense that people's income should increase as their skill set and experience expand with age. Also, the employment opportunities and starting salaries would be expected to be greater for those who have attained more education.

However, it is also obvious that there is considerable variation in income levels for a group of people with identical ages and years of education. This variation is represented by ε in the model. So, in this example, the model would be expressed as shown in Equation 6-2.

$$Income = \beta_0 + \beta_1 Age + \beta_2 Education + \varepsilon \tag{6-2}$$

In the linear model, the β_js represent the unknown p parameters. The estimates for these unknown parameters are chosen so that, on average, the model provides a reasonable estimate of a person's income based on age and education. In other words, the fitted model should minimize the overall error between the linear model and the actual observations. Ordinary Least Squares (OLS) is a common technique to estimate the parameters.

To illustrate how OLS works, suppose there is only one input variable, x, for an outcome variable y. Furthermore, n observations of (x, y) are obtained and plotted in Figure 6-1.

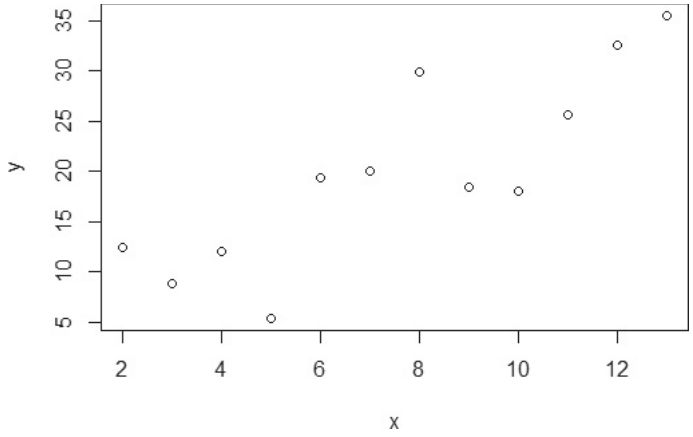

FIGURE 6-1 *Scatterplot of y versus x*

The goal is to find the line that best approximates the relationship between the outcome variable and the input variables. With OLS, the objective is to find the line through these points that minimizes the sum of the squares of the difference between each point and the line in the vertical direction. In other words, find the values of β_0 and β_1 such that the summation shown in Equation 6-3 is minimized.

$$\sum_{i=1}^{n} [y_i - (\beta_0 + \beta_1 x_i)]^2 \tag{6-3}$$

The n individual distances to be squared and then summed are illustrated in Figure 6-2. The vertical lines represent the distance between each observed y value and the line $y = \beta_0 + \beta_1 x$.

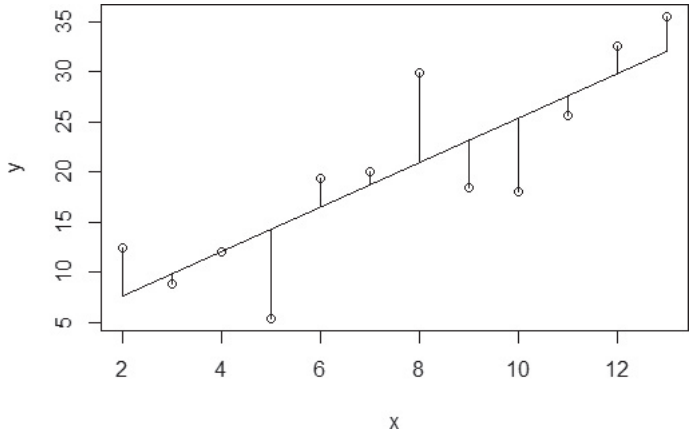

FIGURE 6-2 *Scatterplot of y versus x with vertical distances from the observed points to a fitted line*

In Figure 3-7 of Chapter 3, "Review of Basic Data Analytic Methods Using R," the Anscombe's Quartet example used OLS to fit the linear regression line to each of the four datasets. OLS for multiple input variables is a straightforward extension of the one input variable case provided in Equation 6-3.

The preceding discussion provided the approach to find the best linear fit to a set of observations. However, by making some additional assumptions on the error term, it is possible to provide further capabilities in utilizing the linear regression model. In general, these assumptions are almost always made, so the following model, built upon the earlier described model, is simply called the linear regression model.

Linear Regression Model (with Normally Distributed Errors)

In the previous model description, there were no assumptions made about the error term; no additional assumptions were necessary for OLS to provide estimates of the model parameters. However, in most linear regression analyses, it is common to assume that the error term is a normally distributed random variable with mean equal to zero and constant variance. Thus, the linear regression model is expressed as shown in Equation 6-4.

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 \ldots + \beta_{p-1} x_{p-1} + \varepsilon \tag{6-4}$$

where:

y is the outcome variable

x_j are the input variables, for j = 1, 2,…, p – 1

β_0 is the value of y when each x_j equals zero

β_j is the change in y based on a unit change in x_j, for j = 1, 2,…, p – 1

$\varepsilon \sim N(0, \sigma^2)$ and the εs are independent of each other

This additional assumption yields the following result about the expected value of y, E(y) for given $(x_1, x_2, \ldots x_{p-1})$:

$$\begin{aligned} E(y) &= E(\beta_0 + \beta_1 x_1 + \beta_2 x_2 \ldots + \beta_{p-1} x_{p-1} + \varepsilon) \\ &= \beta_0 + \beta_1 x_1 + \beta_2 x_2 \ldots + \beta_{p-1} x_{p-1} + E(\varepsilon) \\ &= \beta_0 + \beta_1 x_1 + \beta_2 x_2 \ldots + \beta_{p-1} x_{p-1} \end{aligned}$$

Because β_j and x_j are constants, the E(y) is the value of the linear regression model for the given $(x_1, x_2, \ldots x_{p-1})$. Furthermore, the variance of y, V(y), for given $(x_1, x_2, \ldots x_{p-1})$ is this:

$$\begin{aligned} V(y) &= V(\beta_0 + \beta_1 x_1 + \beta_2 x_2 \ldots + \beta_{p-1} x_{p-1} + \varepsilon) \\ &= 0 + V(\varepsilon) = \sigma^2 \end{aligned}$$

Thus, for a given $(x_1, x_2, \ldots x_{p-1})$, y is normally distributed with mean $\beta_0 + \beta_1 x_1 + \beta_2 x_2 \ldots + \beta_{p-1} x_{p-1}$ and variance σ^2. For a regression model with just one input variable, Figure 6-3 illustrates the normality assumption on the error terms and the effect on the outcome variable, y, for a given value of x.

For $x = 8$, one would expect to observe a value of y near 20, but a value of y from 15 to 25 would appear possible based on the illustrated normal distribution. Thus, the regression model estimates the expected value of y for the given value of x. Additionally, the normality assumption on the error term provides some useful properties that can be utilized in performing hypothesis testing on the linear regression model and

providing confidence intervals on the parameters and the mean of y given $(x_1, x_2, \ldots x_{p-1})$. The application of these statistical techniques is demonstrated by applying R to the earlier linear regression model on income.

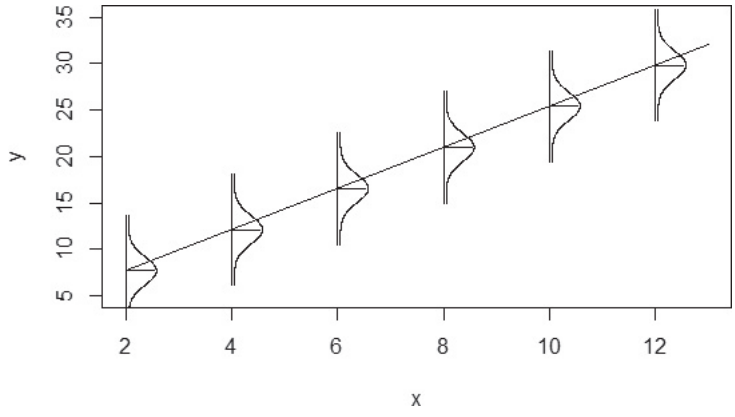

FIGURE 6-3 *Normal distribution about y for a given value of x*

Example in R

Returning to the *Income* example, in addition to the variables age and education, the person's gender, female or male, is considered an input variable. The following code reads a comma-separated-value (CSV) file of 1,500 people's incomes, ages, years of education, and gender. The first 10 rows are displayed:

```
income_input = as.data.frame(  read.csv("c:/data/income.csv")   )
income_input[1:10,]
```

```
   ID Income Age Education Gender
1   1    113  69        12      1
2   2     91  52        18      0
3   3    121  65        14      0
4   4     81  58        12      0
5   5     68  31        16      1
6   6     92  51        15      1
7   7     75  53        15      0
8   8     76  56        13      0
9   9     56  42        15      1
10 10     53  33        11      1
```

Each person in the sample has been assigned an identification number, *ID*. *Income* is expressed in thousands of dollars. (For example, 113 denotes $113,000.) As described earlier, *Age* and *Education* are expressed in years. For *Gender*, a 0 denotes female and a 1 denotes male. A summary of the imported data reveals that the incomes vary from $14,000 to $134,000. The ages are between 18 and 70 years. The education experience for each person varies from a minimum of 10 years to a maximum of 20 years.

```
summary(income_input)
```

```
       ID              Income           Age          Education
 Min.   :   1.0   Min.   : 14.00   Min.   :18.00   Min.   :10.00
```

```
1st Qu.: 375.8   1st Qu.: 62.00   1st Qu.:30.00   1st Qu.:12.00
Median : 750.5   Median : 76.00   Median :44.00   Median :15.00
Mean   : 750.5   Mean   : 75.99   Mean   :43.58   Mean   :14.68
3rd Qu.:1125.2   3rd Qu.: 91.00   3rd Qu.:57.00   3rd Qu.:16.00
Max.   :1500.0   Max.   :134.00   Max.   :70.00   Max.   :20.00

    Gender
Min.   :0.00
1st Qu.:0.00
Median :0.00
Mean   :0.49
3rd Qu.:1.00
Max.   :1.00
```

As described in Chapter 3, a scatterplot matrix is an informative tool to view the pair-wise relationships of the variables. The basic assumption of a linear regression model is that there is a linear relationship between the outcome variable and the input variables. Using the `lattice` package in R, the scatterplot matrix in Figure 6-4 is generated with the following R code:

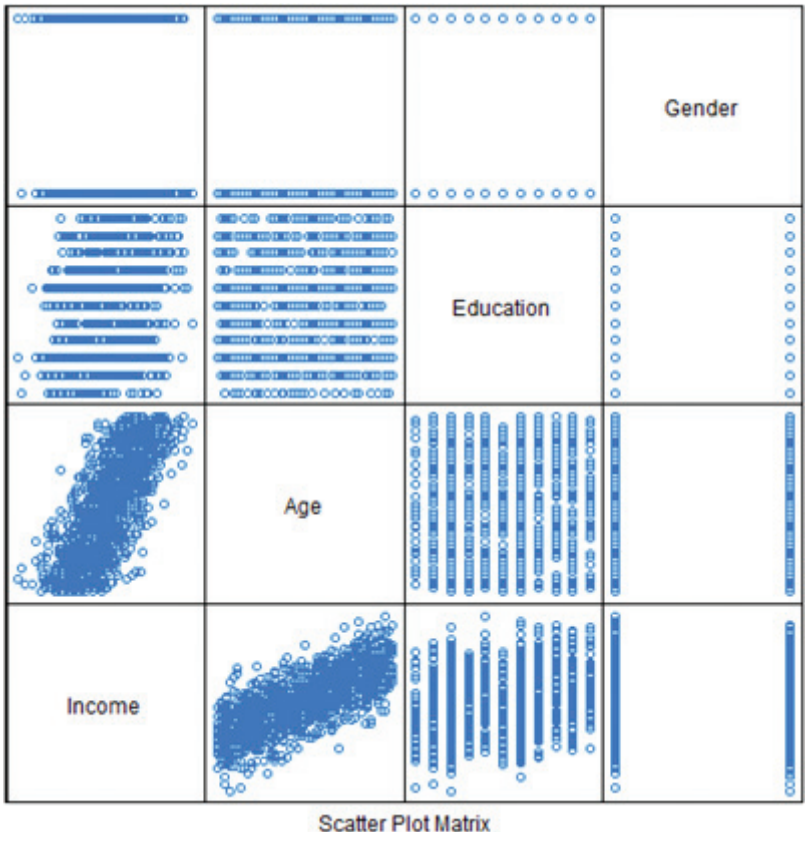

FIGURE 6-4 *Scatterplot matrix of the variables*

```
library(lattice)

splom(~income_input[c(2:5)],  groups=NULL, data=income_input,
      axis.line.tck = 0,
      axis.text.alpha = 0)
```

Because the dependent variable is typically plotted along the y-axis, examine the set of scatterplots along the bottom of the matrix. A strong positive linear trend is observed for *Income* as a function of *Age*. Against *Education*, a slight positive trend may exist, but the trend is not quite as obvious as is the case with the *Age* variable. Lastly, there is no observed effect on *Income* based on *Gender*.

With this qualitative understanding of the relationships between *Income* and the input variables, it seems reasonable to quantitatively evaluate the linear relationships of these variables. Utilizing the normality assumption applied to the error term, the proposed linear regression model is shown in Equation 6-5.

$$Income = \beta_0 + \beta_1 Age + \beta_2 Education + \beta_3 Gender + \varepsilon \tag{6-5}$$

Using the linear model function, `lm()`, in R, the income model can be applied to the data as follows:

```
results <- lm(Income~Age + Education + Gender, income_input)
summary(results)

Call:
lm(formula = Income ~ Age + Education + Gender, data = income_input)

Residuals:
    Min      1Q  Median      3Q     Max
-37.340  -8.101   0.139   7.885  37.271

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  7.26299    1.95575   3.714 0.000212 ***
Age          0.99520    0.02057  48.373  < 2e-16 ***
Education    1.75788    0.11581  15.179  < 2e-16 ***
Gender      -0.93433    0.62388  -1.498 0.134443
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 12.07 on 1496 degrees of freedom
Multiple R-squared:  0.6364,  Adjusted R-squared:  0.6357
F-statistic:   873 on 3 and 1496 DF,  p-value: < 2.2e-16
```

The intercept term, β_0, is implicitly included in the model. The `lm()` function performs the parameter estimation for the parameters β_j (j = 0, 1, 2, 3) using ordinary least squares and provides several useful calculations and results that are stored in the variable called *results* in this example.

After the stated call to `lm()`, a few statistics on the residuals are displayed in the output. The residuals are the observed values of the error term for each of the n observations and are defined for i = 1, 2, …n, as shown in Equation 6-6.

$$e_i = y_i - (b_0 + b_1 x_{i,1} + b_2 x_{i,2} \ldots + b_{p-1} x_{i,p-1}) \tag{6-6}$$

where b_j denotes the estimate for parameter β_j for $j = 0, 1, 2, \ldots p - 1$

From the R output, the residuals vary from approximately –37 to +37, with a median close to 0. Recall that the residuals are assumed to be normally distributed with a mean near zero and a constant variance. The normality assumption is examined more carefully later.

The output provides details about the coefficients. The column *Estimate* provides the OLS estimates of the coefficients in the fitted linear regression model. In general, the *(Intercept)* corresponds to the estimated response variable when all the input variables equal zero. In this example, the intercept corresponds to an estimated income of $7,263 for a newborn female with no education. It is important to note that the available dataset does not include such a person. The minimum age and education in the dataset are 18 and 10 years, respectively. Thus, misleading results may be obtained when using a linear regression model to estimate outcomes for input values not representative within the dataset used to train the model.

The coefficient for *Age* is approximately equal to one. This coefficient is interpreted as follows: For every one unit increase in a person's age, the person's income is expected to increase by $995. Similarly, for every unit increase in a person's years of education, the person's income is expected to increase by about $1,758.

Interpreting the *Gender* coefficient is slightly different. When *Gender* is equal to zero, the *Gender* coefficient contributes nothing to the estimate of the expected income. When *Gender* is equal to one, the expected *Income* is decreased by about $934.

Because the coefficient values are only estimates based on the observed incomes in the sample, there is some uncertainty or sampling error for the coefficient estimates. The *Std. Error* column next to the coefficients provides the sampling error associated with each coefficient and can be used to perform a hypothesis test, using the *t*-distribution, to determine if each coefficient is statistically different from zero. In other words, if a coefficient is not statistically different from zero, the coefficient and the associated variable in the model should be excluded from the model. In this example, the associated hypothesis tests' p-values, `Pr(>|t|)`, are very small for the *Intercept*, *Age*, and *Education* parameters. As seen in Chapter 3, a small p-value corresponds to a small probability that such a large *t* value would be observed under the assumptions of the null hypothesis. In this case, for a given j = 0, 1, 2, . . ., p – 1, the null and alternate hypotheses follow:

$$H_0 : \beta_j = 0 \quad \text{versus} \quad H_A : \beta_j \neq 0$$

For small p-values, as is the case for the *Intercept*, *Age*, and *Education* parameters, the null hypothesis would be rejected. For the *Gender* parameter, the corresponding p-value is fairly large at 0.13. In other words, at a 90% confidence level, the null hypothesis would not be rejected. So, dropping the variable *Gender* from the linear regression model should be considered. The following R code provides the modified model results:

```
results2 <- lm(Income ~ Age + Education, income_input)
summary(results2)

Call:
lm(formula = Income ~ Age + Education, data = income_input)

Residuals:
    Min      1Q  Median      3Q     Max
-36.889  -7.892   0.185   8.200  37.740

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
```

```
(Intercept)  6.75822    1.92728   3.507 0.000467 ***
Age          0.99603    0.02057  48.412  < 2e-16 ***
Education    1.75860    0.11586  15.179  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 12.08 on 1497 degrees of freedom
Multiple R-squared:  0.6359,  Adjusted R-squared:  0.6354
F-statistic:  1307 on 2 and 1497 DF,  p-value: < 2.2e-16
```

Dropping the `Gender` variable from the model resulted in a minimal change to the estimates of the remaining parameters and their statistical significances.

The last part of the displayed results provides some summary statistics and tests on the linear regression model. The ***residual standard error*** is the standard deviation of the observed residuals. This value, along with the associated degrees of freedom, can be used to examine the variance of the assumed normally distributed error terms. R-squared (R^2) is a commonly reported metric that measures the variation in the data that is explained by the regression model. Possible values of R^2 vary from 0 to 1, with values closer to 1 indicating that the model is better at explaining the data than values closer to 0. An R^2 of exactly 1 indicates that the model explains perfectly the observed data (all the residuals are equal to 0). In general, the R^2 value can be increased by adding more variables to the model. However, just adding more variables to explain a given dataset but not to improve the explanatory nature of the model is known as ***overfitting***. To address the possibility of overfitting the data, the adjusted R^2 accounts for the number of parameters included in the linear regression model.

The F-statistic provides a method for testing the entire regression model. In the previous *t*-tests, individual tests were conducted to determine the statistical significance of each parameter. The provided F-statistic and corresponding p-value enable the analyst to test the following hypotheses:

$$H_0 : \beta_1 = \beta_2 = \ldots = \beta_{p-1} = 0 \quad \text{versus} \quad H_A : \beta_j \neq 0$$
$$\text{for at least one } j = 1, 2, \ldots, p-1$$

In this example, the p-value of 2.2e – 16 is small, which indicates that the null hypothesis should be rejected.

Categorical Variables

In the previous example, the variable `Gender` was a simple binary variable that indicated whether a person is female or male. In general, these variables are known as ***categorical variables***. To illustrate how to use categorical variables properly, suppose it was decided in the earlier `Income` example to include an additional variable, `State`, to represent the U.S. state where the person resides. Similar to the use of the `Gender` variable, one possible, but incorrect, approach would be to include a `State` variable that would take a value of 0 for Alabama, 1 for Alaska, 2 for Arizona, and so on. The problem with this approach is that such a numeric assignment based on an alphabetical ordering of the states does not provide a meaningful measure of the difference in the states. For example, is it useful or proper to consider Arizona to be one unit greater than Alaska and two units greater that Alabama?

In regression, a proper way to implement a categorical variable that can take on *m* different values is to add *m-1* binary variables to the regression model. To illustrate with the *Income* example, a binary variable for each of 49 states, excluding Wyoming (arbitrarily chosen as the last of 50 states in an alphabetically sorted list), could be added to the model.

```
results3 <- lm(Income~Age + Education,
               + Alabama,
               + Alaska,
               + Arizona,
               .
               .
               .
               + WestVirginia,
               + Wisconsin,
               income_input)
```

The input file would have 49 columns added for these variables representing each of the first 49 states. If a person was from Alabama, the Alabama variable would be equal to 1, and the other 48 variables would be set to 0. This process would be applied for the other state variables. So, a person from Wyoming, the one state not explicitly stated in the model, would be identified by setting all 49 state variables equal to 0. In this representation, Wyoming would be considered the reference case, and the regression coefficients of the other state variables would represent the difference in income between Wyoming and a particular state.

Confidence Intervals on the Parameters

Once an acceptable linear regression model is developed, it is often helpful to use it to draw some inferences about the model and the population from which the observations were drawn. Earlier, we saw that *t*-tests could be used to perform hypothesis tests on the individual model parameters, β_j, j = 0, 1, …, p – 1. Alternatively, these *t*-tests could be expressed in terms of confidence intervals on the parameters. R simplifies the computation of confidence intervals on the parameters with the use of the `confint()` function. From the *Income* example, the following R command provides 95% confidence intervals on the intercept and the coefficients for the two variables, *Age* and *Education*.

```
confint(results2, level = .95)
```

```
                2.5 %    97.5 %
(Intercept) 2.9777598 10.538690
Age         0.9556771  1.036392
Education   1.5313393  1.985862
```

Based on the data, the earlier estimated value of the *Education* coefficient was 1.76. Using `confint()`, the corresponding 95% confidence interval is (1.53, 1.99), which provides the amount of uncertainty in the estimate. In other words, in repeated random sampling, the computed confidence interval straddles the true but unknown coefficient 95% of the time. As expected from the earlier *t*-test results, none of these confidence intervals straddles zero.

Confidence Interval on the Expected Outcome

In addition to obtaining confidence intervals on the model parameters, it is often desirable to obtain a confidence interval on the expected outcome. In the *Income* example, the fitted linear regression provides the expected income for a given *Age* and *Education*. However, that particular point estimate does not provide information on the amount of uncertainty in that estimate. Using the `predict()` function in R, a confidence interval on the expected outcome can be obtained for a given set of input variable values.

In this illustration, a data frame is built containing a specific age and education value. Using this set of input variable values, the `predict()` function provides a 95% confidence interval on the expected *Income* for a 41-year-old person with 12 years of education.

```
Age <- 41
Education <- 12
new_pt <- data.frame(Age, Education)

conf_int_pt <- predict(results2,new_pt,level=.95,interval="confidence")
conf_int_pt
```

```
       fit      lwr      upr
1 68.69884 67.83102 69.56667
```

For this set of input values, the expected income is $68,699 with a 95% confidence interval of ($67,831, $69,567).

Prediction Interval on a Particular Outcome

The previous confidence interval was relatively close (+/– approximately $900) to the fitted value. However, this confidence interval should not be considered as representing the uncertainty in estimating a particular person's income. The `predict()` function in R also provides the ability to calculate upper and lower bounds on a particular outcome. Such bounds provide what are referred to as ***prediction intervals***. Returning to the *Income* example, in R the 95% prediction interval on the *Income* for a 41-year-old person with 12 years of education is obtained as follows:

```
pred_int_pt <- predict(results2,new_pt,level=.95,interval="prediction")
pred_int_pt
```

```
       fit      lwr      upr
1 68.69884 44.98867 92.40902
```

Again, the expected income is $68,699. However, the 95% prediction interval is ($44,988, $92,409). If the reason for this much wider interval is not obvious, recall that in Figure 6-3, for a particular input variable value, the expected outcome falls on the regression line, but the individual observations are normally distributed about the expected outcome. The confidence interval applies to the expected outcome that falls on the regression line, but the prediction interval applies to an outcome that may appear anywhere within the normal distribution.

Thus, in linear regression, confidence intervals are used to draw inferences on the population's expected outcome, and prediction intervals are used to draw inferences on the next possible outcome.

6.1.3 Diagnostics

The use of hypothesis tests, confidence intervals, and prediction intervals is dependent on the model assumptions being true. The following discussion provides some tools and techniques that can be used to validate a fitted linear regression model.

Evaluating the Linearity Assumption

A major assumption in linear regression modeling is that the relationship between the input variables and the outcome variable is linear. The most fundamental way to evaluate such a relationship is to plot the outcome variable against each input variable. In the `Income` example, such scatterplots were generated in Figure 6-4. If the relationship between `Age` and `Income` is represented as illustrated in Figure 6-5, a linear model would not apply. In such a case, it is often useful to do any of the following:

- Transform the outcome variable.
- Transform the input variables.
- Add extra input variables or terms to the regression model.

Common transformations include taking square roots or the logarithm of the variables. Another option is to create a new input variable such as the age squared and add it to the linear regression model to fit a quadratic relationship between an input variable and the outcome.

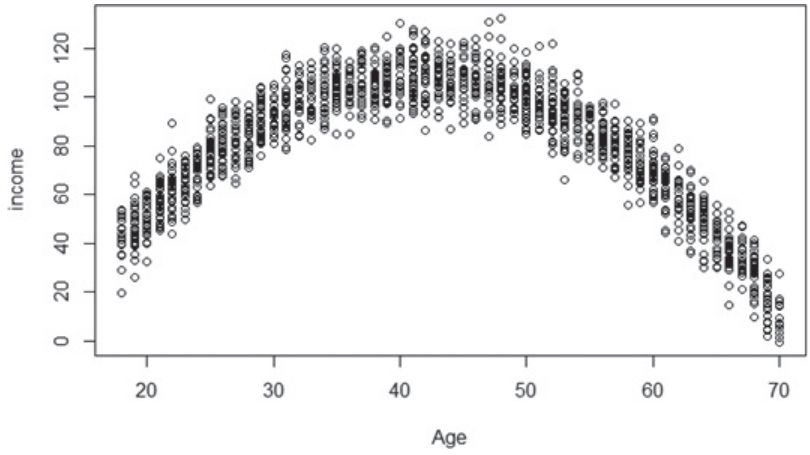

FIGURE 6-5 *Income as a quadratic function of Age*

Additional use of transformations will be considered when evaluating the residuals.

Evaluating the Residuals

As stated previously, it is assumed that the error terms in the linear regression model are normally distributed with a mean of zero and a constant variance. If this assumption does not hold, the various inferences that were made with the hypothesis tests, confidence intervals, and prediction intervals are suspect.

To check for constant variance across all y values along the regression line, use a simple plot of the residuals against the fitted outcome values. Recall that the residuals are the difference between the observed

outcome variables and the fitted value based on the OLS parameter estimates. Because of the importance of examining the residuals, the `lm()` function in R automatically calculates and stores the fitted values and the residuals, in the components *`fitted.values`* and *`residuals`* in the output of the `lm()` function. Using the Income regression model output stored in *`results2`*, Figure 6-6 was generated with the following R code:

```
with(results2, {
plot(fitted.values, residuals,ylim=c(-40,40) )
         points(c(min(fitted.values),max(fitted.values)),
                c(0,0), type = "l")})
```

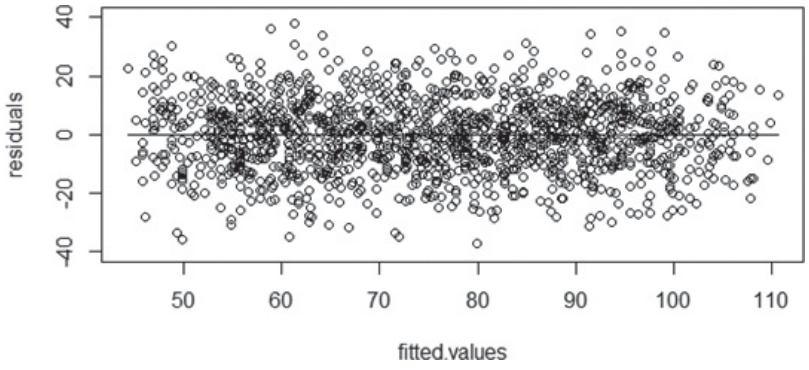

FIGURE 6-6 *Residual plot indicating constant variance*

The plot in Figure 6-6 indicates that regardless of income value along the fitted linear regression model, the residuals are observed somewhat evenly on both sides of the reference zero line, and the spread of the residuals is fairly constant from one fitted value to the next. Such a plot would support the mean of zero and the constant variance assumptions on the error terms.

If the residual plot appeared like any of those in Figures 6-7 through 6-10, then some of the earlier discussed transformations or possible input variable additions should be considered and attempted. Figure 6-7 illustrates the existence of a nonlinear trend in the residuals. Figure 6-8 illustrates that the residuals are not centered on zero. Figure 6-9 indicates a linear trend in the residuals across the various outcomes along the linear regression model. This plot may indicate a missing variable or term from the regression model. Figure 6-10 provides an example in which the variance of the error terms is not a constant but increases along the fitted linear regression model.

Evaluating the Normality Assumption

The residual plots are useful for confirming that the residuals were centered on zero and have a constant variance. However, the normality assumption still has to be validated. As shown in Figure 6-11, the following R code provides a histogram plot of the residuals from `results2`, the output from the *`Income`* example:

```
hist(results2$residuals, main="")
```

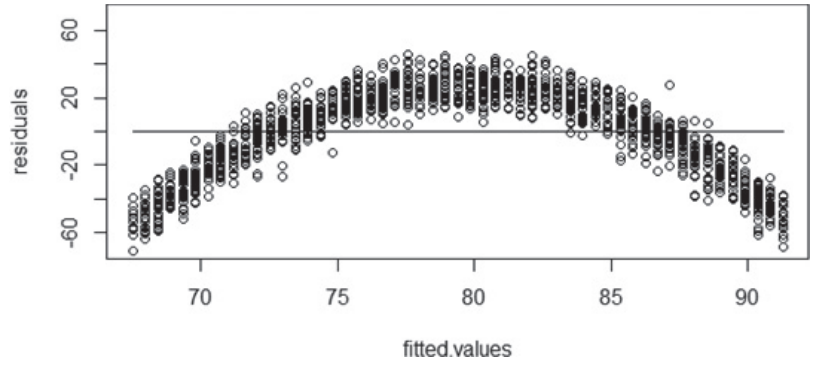

FIGURE 6-7 *Residuals with a nonlinear trend*

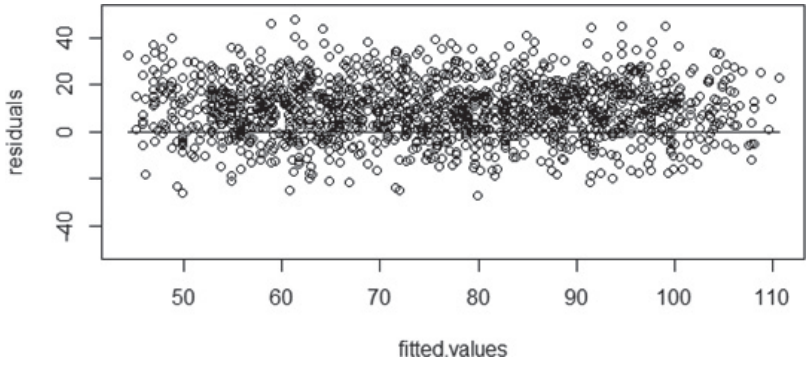

FIGURE 6-8 *Residuals not centered on the zero line*

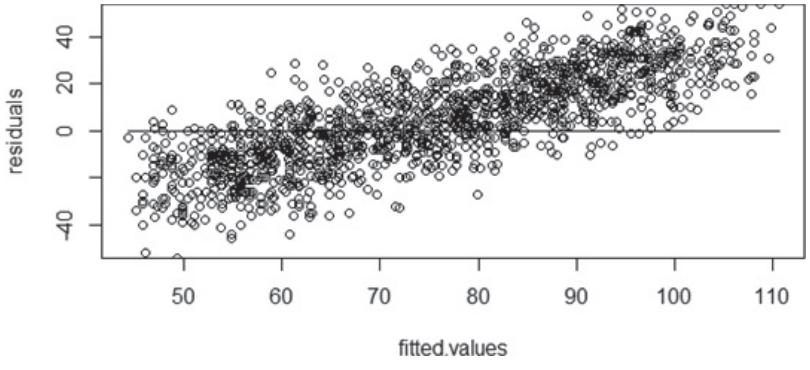

FIGURE 6-9 *Residuals with a linear trend*

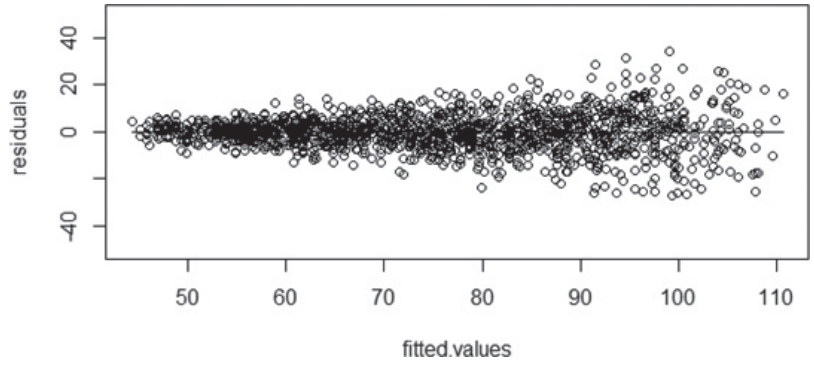

FIGURE 6-10 *Residuals with nonconstant variance*

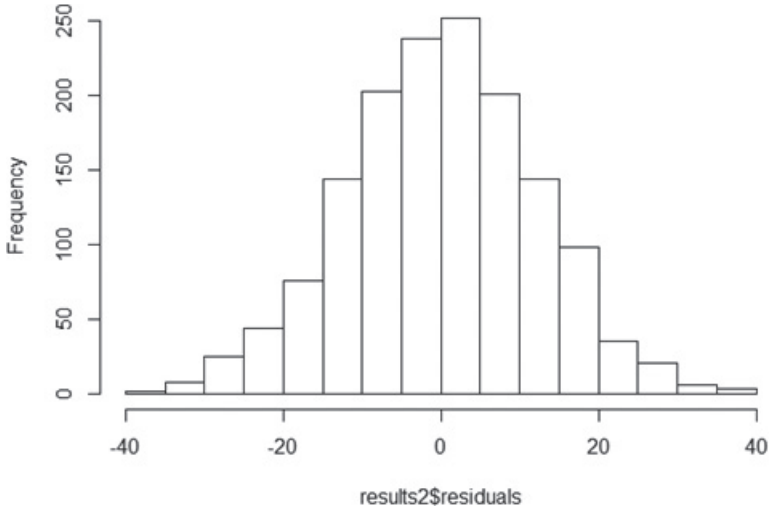

FIGURE 6-11 *Histogram of normally distributed residuals*

From the histogram, it is seen that the residuals are centered on zero and appear to be symmetric about zero, as one would expect for a normally distributed random variable. Another option is to examine a Q-Q plot that compares the observed data against the quantiles (Q) of the assumed distribution. In R, the following code generates the Q-Q plot shown in Figure 6-12 for the residuals from the *Income* example and provides the line that the points should follow for values from a normal distribution.

```
qqnorm(results2$residuals, ylab="Residuals", main="")
qqline(results2$residuals)
```

A Q-Q plot as provided in Figure 6-13 would indicate that additional refinement of the model is required to achieve normally distributed error terms.

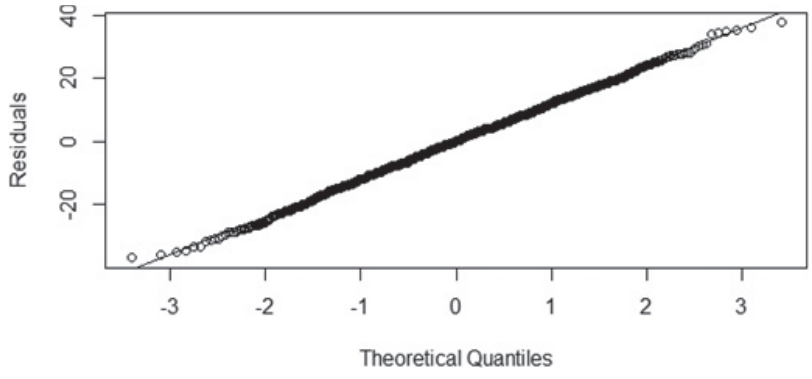

FIGURE 6-12 *Q-Q plot of normally distributed residuals*

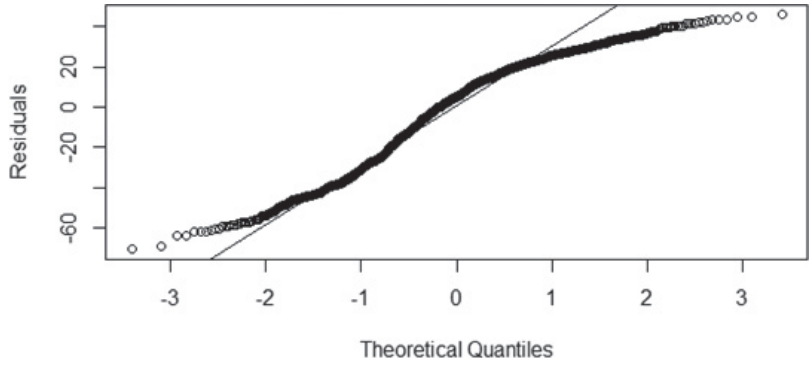

FIGURE 6-13 *Q-Q plot of non-normally distributed residuals*

N-Fold Cross-Validation

To prevent overfitting a given dataset, a common practice is to randomly split the entire dataset into a training set and a testing set. Once the model is developed on the training set, the model is evaluated against the testing set. When there is not enough data to create training and testing sets, an N-fold cross-validation technique may be helpful to compare one fitted model against another. In N-fold cross-validation, the following occurs:

- The entire dataset is randomly split into N datasets of approximately equal size.
- A model is trained against N – 1 of these datasets and tested against the remaining dataset. A measure of the model error is obtained.
- This process is repeated a total of N times across the various combinations of N datasets taken N – 1 at a time. Recall:

$$\binom{N}{N-1} = N$$

- The observed N model errors are averaged over the N folds.

The averaged error from one model is compared against the averaged error from another model. This technique can also help determine whether adding more variables to an existing model is beneficial or possibly overfitting the data.

Other Diagnostic Considerations

Although a fitted linear regression model conforms with the preceding diagnostic criteria, it is possible to improve the model by including additional input variables not yet considered. In the previous `Income` example, only three possible input variables—`Age`, `Education`, and `Gender`—were considered. Dozens of other additional input variables such as `Housing` or `Marital_Status` may improve the fitted model. It is important to consider all possible input variables early in the analytic process.

As mentioned earlier, in reviewing the R output from fitting a linear regression model, the adjusted R^2 applies a penalty to the R^2 value based on the number of parameters added to the model. Because the R^2 value will always move closer to one as more variables are added to an existing regression model, the adjusted R^2 value may actually decrease after adding more variables.

The residual plots should be examined for any ***outliers***, observed points that are markedly different from the majority of the points. Outliers can result from bad data collection, data processing errors, or an actual rare occurrence. In the `Income` example, suppose that an individual with an income of a million dollars was included in the dataset. Such an observation could affect the fitted regression model, as seen in one of the examples of Anscombe's Quartet.

Finally, the magnitudes and signs of the estimated parameters should be examined to see if they make sense. For example, suppose a negative coefficient for the Education variable in the `Income` example was obtained. Because it is natural to assume that more years of education lead to higher incomes, either something very unexpected has been discovered, or there is some issue with the model, how the data was collected, or some other factor. In either case, further investigation is warranted.

6.2 Logistic Regression

In linear regression modeling, the outcome variable is a continuous variable. As seen in the earlier `Income` example, linear regression can be used to model the relationship between age and education to income. Suppose a person's actual income was not of interest, but rather whether someone was wealthy or poor. In such a case, when the outcome variable is categorical in nature, logistic regression can be used to predict the likelihood of an outcome based on the input variables. Although logistic regression can be applied to an outcome variable that represents multiple values, the following discussion examines the case in which the outcome variable represents two values such as true/false, pass/fail, or yes/no.

For example, a logistic regression model can be built to determine if a person will or will not purchase a new automobile in the next 12 months. The training set could include input variables for a person's age, income, and gender as well as the age of an existing automobile. The training set would also include the outcome variable on whether the person purchased a new automobile over a 12-month period. The logistic regression model provides the likelihood or probability of a person making a purchase in the next 12 months. After examining a few more use cases for logistic regression, the remaining portion of this chapter examines how to build and evaluate a logistic regression model.

6.2.1 Use Cases

The logistic regression model is applied to a variety of situations in both the public and the private sector. Some common ways that the logistic regression model is used include the following:

- **Medical:** Develop a model to determine the likelihood of a patient's successful response to a specific medical treatment or procedure. Input variables could include age, weight, blood pressure, and cholesterol levels.
- **Finance:** Using a loan applicant's credit history and the details on the loan, determine the probability that an applicant will default on the loan. Based on the prediction, the loan can be approved or denied, or the terms can be modified.
- **Marketing:** Determine a wireless customer's probability of switching carriers (known as churning) based on age, number of family members on the plan, months remaining on the existing contract, and social network contacts. With such insight, target the high-probability customers with appropriate offers to prevent churn.
- **Engineering:** Based on operating conditions and various diagnostic measurements, determine the probability of a mechanical part experiencing a malfunction or failure. With this probability estimate, schedule the appropriate preventive maintenance activity.

6.2.2 Model Description

Logistic regression is based on the logistic function $f(y)$, as given in Equation 6-7.

$$f(y) = \frac{e^y}{1+e^y} \quad \text{for } -\infty < y < \infty \tag{6-7}$$

Note that as $y \to \infty$, $f(y) \to 1$, and as $y \to -\infty$, $f(y) \to 0$. So, as Figure 6-14 illustrates, the value of the logistic function $f(y)$ varies from 0 to 1 as y increases.

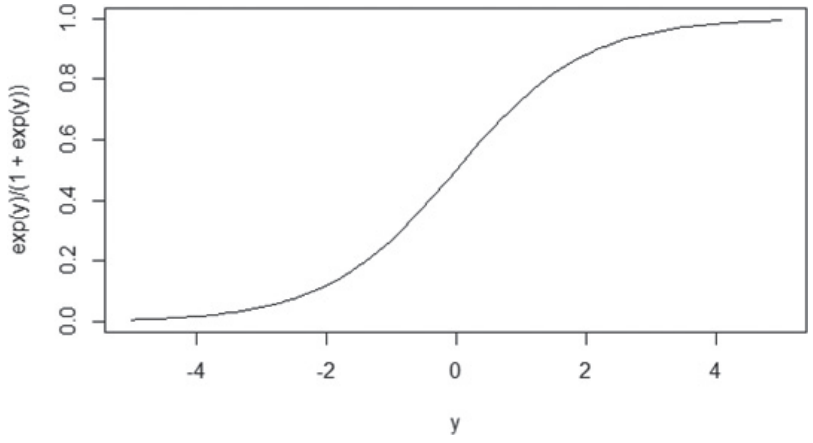

FIGURE 6-14 *The logistic function*

Because the range of $f(y)$ is (0, 1), the logistic function appears to be an appropriate function to model the probability of a particular outcome occurring. As the value of y increases, the probability of the outcome occurring increases. In any proposed model, to predict the likelihood of an outcome, y needs to be a function of the input variables. In logistic regression, y is expressed as a linear function of the input variables. In other words, the formula shown in Equation 6-8 applies.

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 \ldots + \beta_{p-1} x_{p-1} \tag{6-8}$$

Then, based on the input variables $x_1, x_2, \ldots, x_{p-1}$, the probability of an event is shown in Equation 6-9.

$$p(x_1, x_2, \ldots, x_{p-1}) = f(y) = \frac{e^y}{1+e^y} \quad \text{for} -\infty < y < \infty \tag{6-9}$$

Equation 6-8 is comparable to Equation 6-1 used in linear regression modeling. However, one difference is that the values of y are not directly observed. Only the value of $f(y)$ in terms of success or failure (typically expressed as 1 or 0, respectively) is observed.

Using p to denote $f(y)$, Equation 6-9 can be rewritten in the form provided in Equation 6-10.

$$ln\left(\frac{p}{1-p}\right) = y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 \ldots + \beta_p x_{p-1} \tag{6-10}$$

The quantity $ln\left(\frac{p}{1-p}\right)$, in Equation 6-10 is known as the log odds ratio, or the logit of p. Techniques such as Maximum Likelihood Estimation (MLE) are used to estimate the model parameters. MLE determines the values of the model parameters that maximize the chances of observing the given dataset. However, the specifics of implementing MLE are beyond the scope of this book.

The following example helps to clarify the logistic regression model. The mechanics of using R to fit a logistic regression model are covered in the next section on evaluating the fitted model. In this section, the discussion focuses on interpreting the fitted model.

Customer Churn Example

A wireless telecommunications company wants to estimate the probability that a customer will churn (switch to a different company) in the next six months. With a reasonably accurate prediction of a person's likelihood of churning, the sales and marketing groups can attempt to retain the customer by offering various incentives. Data on 8,000 current and prior customers was obtained. The variables collected for each customer follow:

- `Age` (years)
- `Married` (true/false)
- `Duration` as a customer (years)
- `Churned_contacts` (count)—Number of the customer's contacts that have churned (count)
- `Churned` (true/false)—Whether the customer churned

After analyzing the data and fitting a logistic regression model, `Age` and `Churned_contacts` were selected as the best predictor variables. Equation 6-11 provides the estimated model parameters.

$$y = 3.50 - 0.16 * Age + 0.38 * Churned\_contacts \tag{6-11}$$