Chapter 5

Exploring Data

As a data miner, we need to live and breathe our data. Even before we start building our data mining models, we can gain significant insights through exploring the data. Insights gained can deliver new discoveries to our clients—discoveries that can offer benefits early on in a data mining project. Through such insights and discoveries, we will increase our knowledge and understanding.

Through exploring our data, we can discover what the data looks like, its boundaries (the minimum and maximum values), its numeric characteristics (the average value), and how it is distributed (how spread out the data is). The data begins to tell us a story, and we need to build and understand that story for ourselves. By capturing that story, we can communicate it back to our clients.

This task of exploratory data analysis (often abbreviated as EDA) is a core activity in any data mining project. Exploring the data generally involves getting a basic understanding of a dataset through numerous variable summaries and visual plots.

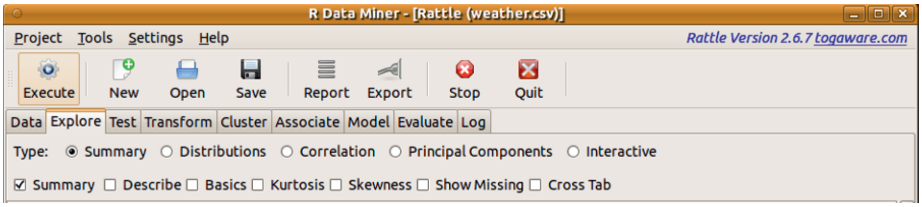
Through data exploration, we begin to understand the "lay of the land" just as a gold miner works to understand the terrain rather than blindly digging for gold randomly. Through this exploration, we will often identify problems with the data, including missing values, noise, erroneous data, and skewed distributions. This in turn will drive our choice of the most appropriate and, importantly, applicable tools for preparing and transforming our data and for mining. Some tools, for example, are limited in use when there is much missing data.

Rattle provides tools ranging from textual summaries to visually appealing graphical plots for identifying correlations between variables. The

Explore tab within Rattle provides a window into the tools for helping us understand our data, driving the many options available in R.

5.1 Summarising Data

Figure 5.1 shows the options available under Rattle's Explore tab. We begin our exploration of data with the basic Summary option, which provides a textual overview of the data. Whilst a picture may be worth a thousand words, textual summaries still play an important role in our understanding of data.


R Data Miner - [Rattle (weather.csv)]
Project Tools Settings Help
Rattle Version 2.6.7 togaware.com
Execute New Open Save Report Export Stop Quit
Data Explore Test Transform Cluster Associate Model Evaluate Log

Type: [✓ Summary] Distributions Correlation Principal Components Interactive

[✓ Summary] Describe Basics Kurtosis Skewness Show Missing Cross Tab

Figure 5.1: The Explore tab provides access to a variety of ways in which we start to understand our data.

Often, we deal with very large datasets, and some of the calculations and visualisations we perform will be computationally quite expensive. Thus it may be useful to summarise random subsets of the data instead. The Partition option of the Data tab is useful here. This uses `sample()` to generate a list of row numbers that we can then use to index the dataset.

The following example generates a 20% (i.e., 0.2 times the number of rows) random sample of our *weather* dataset. We use `nrow()` to obtain the number of rows in the sample (73.2) and `dim()` for information about the number of rows and columns in the data frame:

```
> library(rattle)
> dim(weather)

[1] 366  24

> set.seed(42)
> smpl <- sample(nrow(weather), 0.2*nrow(weather))
> dim(weather[smpl,])

[1] 73 24
```

For our *weather* dataset with only 366 observations, we clearly do not need to sample.

5.1.1 Basic Summaries

The simplest text-based statistical summary of a dataset is provided by `summary()`. This is always a useful starting point in reviewing our data. It provides a summary of each variable. Here we see summaries for a mixture of numeric and categoric variables:

```
> summary(weather[7:9])

    Sunshine       WindGustDir  WindGustSpeed
 Min.   : 0.00   NW     : 73   Min.   :13.0
 1st Qu.: 5.95   NNW    : 44   1st Qu.:31.0
 Median : 8.60   E      : 37   Median :39.0
 Mean   : 7.91   WNW    : 35   Mean   :39.8
 3rd Qu.:10.50   ENE    : 30   3rd Qu.:46.0
 Max.   :13.60   (Other):144   Max.   :98.0
 NA's   : 3.00   NA's   :  3   NA's   : 2.0
```

For the numeric variables, `summary()` will list the minimum and maximum values together with average values (the mean and median) and the first and third quartiles. The quartiles represent a partitioning of the values of the numeric variable into four equally sized sets. The first quartile includes 25% of the observations of this variable that have a value less than this first quartile. The third quartile is the same, but at the 75% mark. The median is actually also the second quartile, representing the 50% cutoff (i.e., the middle value).

Generally, if the mean and median are significantly different, then we would think that there are some observations of this variable that are quite a distance from the mean in one particular direction (i.e., some exceptionally large positive or negative values, generally called outliers, which we cover in Chapter 7). From the variables we see above, `Sunshine` has a relatively larger (although still small) gap between its mean and median, whilst the mean and median of `WindGustSpeed` are quite similar. `Sunshine` has more small observations than large observations, using our terms rather loosely.

The categoric variables will have listed for them the top few most frequent levels with their frequency counts and then aggregate the remainder under the `(Other)` label. Thus there are 73 observations with a `NW` wind gust, 44 with a `NNW` wind gust, and so on. We observe quite a predominance of these northwesterly wind gusts. For both types of listings, the count of any missing values (`NAs`) will be reported. A somewhat more detailed summary is obtained from `describe()`, provided by **Hmisc** (Harrell, 2010). To illustrate this we first load **Hmisc** into the library:

```
> library(Hmisc)
```

For numeric variables like `Sunshine` (which is variable number 7) `describe()` outputs two more deciles (10% and 90%) as well as two other percentiles (5% and 95%). The output continues with a list of the lowest few and highest few observations of the variable. The extra information is quite useful in building up our picture of the data.

```
> describe(weather[7])

weather[7]

 1  Variables      366  Observations
-----------------------------------------------------------
Sunshine
      n missing  unique    Mean     .05     .10     .25
    363       3     114   7.909    0.60    2.04    5.95

    .50     .75     .90     .95
   8.60   10.50   11.80   12.60

lowest :  0.0  0.1  0.2  0.3  0.4
highest: 13.1 13.2 13.3 13.5 13.6
-----------------------------------------------------------
```

For categoric variables like `WindGustDir` (which is variable number 8) `describe()` outputs the frequency count and the percentage this represents for each level. The information is split over as many lines as is required, as we see in the following code box.

```
> describe(weather[8])

weather[8]

 1  Variables      366  Observations
----------------------------------------------------------
WindGustDir
      n missing  unique
    363       3      16

           N NNE NE ENE  E ESE SE SSE  S SSW SW WSW  W
Frequency 21   8 16  30 37  23 12  12 22   5  3   2 20
%          6   2  4   8 10   6  3   3  6   1  1   1  6
          WNW NW NNW
Frequency  35 73  44
%          10 20  12
----------------------------------------------------------
```

5.1.2 Detailed Numeric Summaries

An even more detailed summary of the numeric data is provided by `basicStats()` from **fBasics** (Wuertz et al., 2010). Though intended for time series data, it provides useful statistics in general, as we see in the code box below.

Some of the same data that we have already seen is presented together with a little more. Here we see that the variable `Sunshine` is observed 366 times, of which 3 are missing (`NAs`). The minimum, maximum, quartiles, mean, and median are as before.

The statistics then go on to include the total sum of the amount of sunshine, the standard error of the mean, the lower and upper confidence limits on the true value of the mean (at a 95% level of confidence), the variance and standard deviation, and two measures of the shape of the distribution of the data: skewness and kurtosis (explained below).

The mean is stated as being 7.91. We can be 95% confident that the actual mean (the *true mean*) of the population, of which the data we have here is assumed to be a random sample, is somewhere between 7.55 and 8.27.

```
> library(fBasics)
> basicStats(weather$Sunshine)

             X..weather.Sunshine
nobs                    366.0000
NAs                       3.0000
Minimum                   0.0000
Maximum                  13.6000
1. Quartile               5.9500
3. Quartile              10.5000
Mean                      7.9094
Median                    8.6000
Sum                    2871.1000
SE Mean                   0.1827
LCL Mean                  7.5500
UCL Mean                  8.2687
Variance                 12.1210
Stdev                     3.4815
Skewness                 -0.7235
Kurtosis                 -0.2706
```

The standard deviation is a measure of how spread out (or how dispersed or how variable) the data is with respect to the mean. It is measured in the same units as the mean itself. We can read it to say that most observations (about 68% of them) are no more than this distance from the mean. That is, most days have 7.91 hours of sunshine, plus or minus 3.48 hours.

Our observation of the mean and standard deviation for the sunshine data needs to be understood in the context of other knowledge we glean about the variable. Consider again Figure 2.8 on page 34. An observation we might make there is that the distribution appears to be what we might call bimodal—that is it has two distinct scenarios. One is that of a cloudy day, and for such days the hours of sunshine will be quite small. The other is that of a sunny day, for which the hours of sunshine will cover the whole day. This observation might be more important to us in weather forecasting, than the interval around the mean. We might want to transform this variable into a binary variable to capture this observation. Transformations are covered in Chapter 7. The variance is the square of the standard deviation.

5.1.3 Distribution

In statistics, we often talk about how observations (i.e., the values of a variable) are distributed. By "distributed" we mean how many times each value of some variable might appear in a collection of data. For the variable `Sunshine`, for example, the distribution is concerned with how many days have 8 hours of sunshine, how many have 8.1 hours, and so on.

The concept is not quite that black and white, though. In fact, the distribution is often visualised as a smooth curve, as we might be familiar with from published articles that talk about a *normal* (or some other common) distribution. We often hear about the bell curve. This is a graph that plots a shape similar to that of musical bells. For our discussion here, it is useful to have a mental picture of such a bell curve, where the horizontal axis represents the possible values of the variable (the observations) and the vertical axis represents how often those values might occur.

5.1.4 Skewness

The skewness is a measure of how asymmetrically our data is distributed. The skewness indicates whether there is a long tail on one or the other side of the mean value of the data. Here we use `skewness()` from **Hmisc** to compare the distributions of a number of variables:

```
> skewness(weather[,c(7,9,12,13)], na.rm=TRUE)

     Sunshine WindGustSpeed  WindSpeed9am  WindSpeed3pm
      -0.7235        0.8361        1.3602        0.5913
```

A skewness of magnitude (i.e., ignoring whether it is positive or negative) greater than 1 represents quite an obvious extended spread of the data in one direction or the other. The direction of the spread is indicated by the sign of the skewness. A positive skewness indicates that the spread is more to the right side of the mean (i.e., above the mean) and is referred to as having a longer right tail. A negative skewness is the same but on the left side.

Many models and statistical tests are based on the assumption of a so-called bell curve distribution of the data, which describes a symmetric spread of data values around the mean. The greater the skewness, the greater the distortion to this spread of values. For a large skewness, the assumptions of the models and statistical tests will not hold, and so we need to be a little more careful in their use. The impact tends to be greater for traditional statistical approaches and less so for more recent approaches like decision trees.

5.1.5 Kurtosis

A companion for skewness is kurtosis, which is a measure of the nature of the peaks in the distribution of the data. Once again, we might picture the distribution of the data as having a shape that is something like that of a church bell (i.e., a bell curve). The kurtosis tells us how skinny or fat the bell is. **Hmisc** provides `kurtosis()`:

```
> kurtosis(weather[,c(7,9,12,13)], na.rm=TRUE)

     Sunshine WindGustSpeed  WindSpeed9am  WindSpeed3pm
      -0.2706        1.4761        1.4758        0.1963
```

A larger value for the kurtosis indicates that the distribution has a sharper peak, primarily because there are only a few values with more extreme values compared with the mean value. Thus, `WindSpeed9am` has a sharper peak and a smaller number of more extreme values than `WindSpeed3pm`. The lower kurtosis value indicates a flatter peak.

5.1.6 Missing Values

Missing values present challenges to data mining and modelling in general. There can be many reasons for missing values, including the fact that the data is hard to collect and so not always available (e.g., results of an expensive medical test), or that it is simply not recorded because it is in fact 0 (e.g., spouse income for a spouse who stays home to manage the family). Knowing why the data is missing is important in deciding how to deal with the missing value.

We can explore the nature of the missing data using `md.pattern()` from **mice** (van Buuren and Groothuis-Oudshoorn, 2011), as Rattle does when activating the Show Missing check button of the Summary option of the Explore tab. The results can help us understand any structure in the missing data and even why the data is missing:

```
> library(mice)

mice 2.8 2011-03-24

> md.pattern(weather[,7:10])

    WindGustSpeed Sunshine WindGustDir WindDir9am
329             1        1           1          1  0
  3             1        0           1          1  1
  1             1        1           0          1  1
 31             1        1           1          0  1
  2             0        1           0          1  2
                2        3           3         31 39
```

The table presents, for each variable, a pattern of missing values. Within the table, a `1` indicates a value is present, whereas a `0` indicates a value is missing.

The left column records the number of observations that match the corresponding pattern of missing values. There are 329 observations with no missing values over these four variables (each having a value of `1` within that row). The final column is the number of missing values within the pattern. In the case of the first row here, with no missing values, this is `0`.

The rows and columns are sorted in ascending order according to the amount of missing data. Thus, generally, the first row records the number of observations that have no missing values. In our example, the second row corresponds to a pattern of missing values for the variable `Sunshine`. There are NA hundred NA three observations that have just `Sunshine` missing (and there are three observations overall that have `Sunshine` missing based on the final row). This particular row's pattern has just a single variable missing, as indicated by the `1` in the final column.

The final row records the number of missing values over the whole dataset for each of the variables. For example, `WindGustSpeed` has two missing values. The total number of missing values over all observations and variables is noted at the bottom right (39 in this example).

In Section 7.4, we will discuss how we might deal with missing values through an approach called imputation.

5.2 Visualising Distributions

In the previous section, we purposely avoided any graphical presentation of our data. In fact, I rather expect you might have been frustrated that there was no picture there to help you visualise what we were describing. The absence of a picture was primarily to make the point that it can get a little tricky explaining ideas without the aid of pictures. In particular, our explanation of skewness and kurtosis was quite laboured, and reverted to painting a mental picture rather than presenting an actual picture. After reviewing this current section, go back to reconsider the discussion of skewness and kurtosis. Pictures really do play a significant role in understanding, and graphical presentations, for many, are more effective for communicating than tables of numbers.

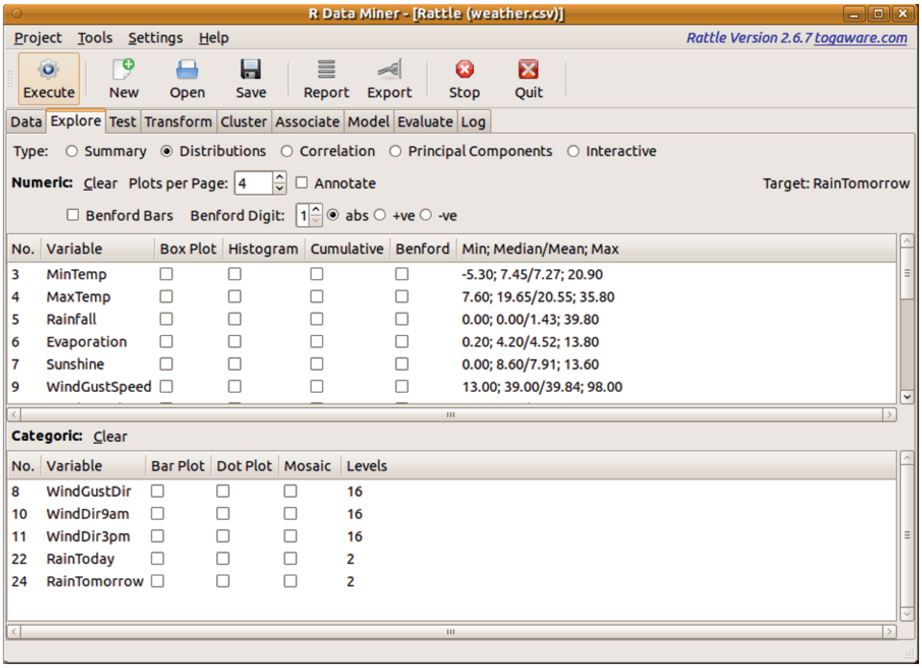
Graphical tools allow us to visually investigate the data's characteristics to help us understand it. Such an exploration of the data can clearly identify errors in the data or oddities about its collection. This will also guide our choice of options to transform variables in different ways and to select those variables of interest.

Visualising data has been an area of study within statistics for many years. A vast array of techniques have been developed for presenting data visually, and the topic is covered in great detail in many books, including Cleveland (1993) and Tufte (1985).

It is a good idea, then, early on in a data mining project, to review the distributions of the values of each of the variables in our dataset graphically. R provides a wealth of options for graphically presenting data. Indeed, R is one of the most capable data visualisation languages and allows us to program the visualisations. There are also many standard types of visualisations, and some of these are available through Rattle's Distributions option on the Explore tab (Figure 5.2).

Using Rattle's Distributions option, we can select specific variables of interest and display various distribution plots. Selecting many variables will of course lead to many plots being displayed, and so it may be useful to display multiple plots per page (i.e., per window). Rattle will do this for us automatically, controlled by our setting of the appropriate value for the number of plots per page within the interface. By default, four plots are displayed per page or window.

Figure 5.3 illustrates a sample of the variety of plots available. Clockwise from the top left plot, we have illustrated a **box plot**, a **histogram**, a **mosaic plot**, and a **cumulative function plot**. Because we have


R Data Miner - [Rattle (weather.csv)]
Project Tools Settings Help
Rattle Version 2.6.7 togaware.com
Execute New Open Save Report Export Stop Quit
Data Explore Test Transform Cluster Associate Model Evaluate Log

Type: ○ Summary [✓ Distributions] ○ Correlation ○ Principal Components ○ Interactive

Numeric: Clear Plots per Page: 4 ☐ Annotate Target: RainTomorrow

☐ Benford Bars Benford Digit: 1 [✓ abs] ○ +ve ○ -ve

No.	Variable	Box Plot	Histogram	Cumulative	Benford	Min; Median/Mean; Max
3	MinTemp	☐	☐	☐	☐	-5.30; 7.45/7.27; 20.90
4	MaxTemp	☐	☐	☐	☐	7.60; 19.65/20.55; 35.80
5	Rainfall	☐	☐	☐	☐	0.00; 0.00/1.43; 39.80
6	Evaporation	☐	☐	☐	☐	0.20; 4.20/4.52; 13.80
7	Sunshine	☐	☐	☐	☐	0.00; 8.60/7.91; 13.60
9	WindGustSpeed	☐	☐	☐	☐	13.00; 39.00/39.84; 98.00

Categoric: Clear

No.	Variable	Bar Plot	Dot Plot	Mosaic	Levels
8	WindGustDir	☐	☐	☐	16
10	WindDir9am	☐	☐	☐	16
11	WindDir3pm	☐	☐	☐	16
22	RainToday	☐	☐	☐	2
24	RainTomorrow	☐	☐	☐	2

Figure 5.2: The Explore tab's Distributions option provides convenient access to a variety of standard plots for the two primary variable types—numeric and categoric.

identified a target variable (`RainTomorrow`), the plots include the distributions for each subset of observations associated with each value (`No` and `Yes`) of the target variable. That is, the plots include a visualisation of the stratification of the data based on the different values of the target variable.

In brief, the box plot identifies the median and mean of the variable (`MinTemp`) and the spread from the first quartile to the third, and indicates the outliers. The histogram splits the range of values of the variable (`Sunshine`) into segments (hours in this case) and shows the number of observations in each segment. The mosaic plot shows the proportions of data split according to the target (`RainTomorrow`) and the chosen variable (`WindGustDir`, modified to have fewer levels in this case). The cumulative plot shows the percentage of observations below any particular value of the variable (`WindGustSpeed`).

Each of the plots available through Rattle is explained in more detail in the following sections.

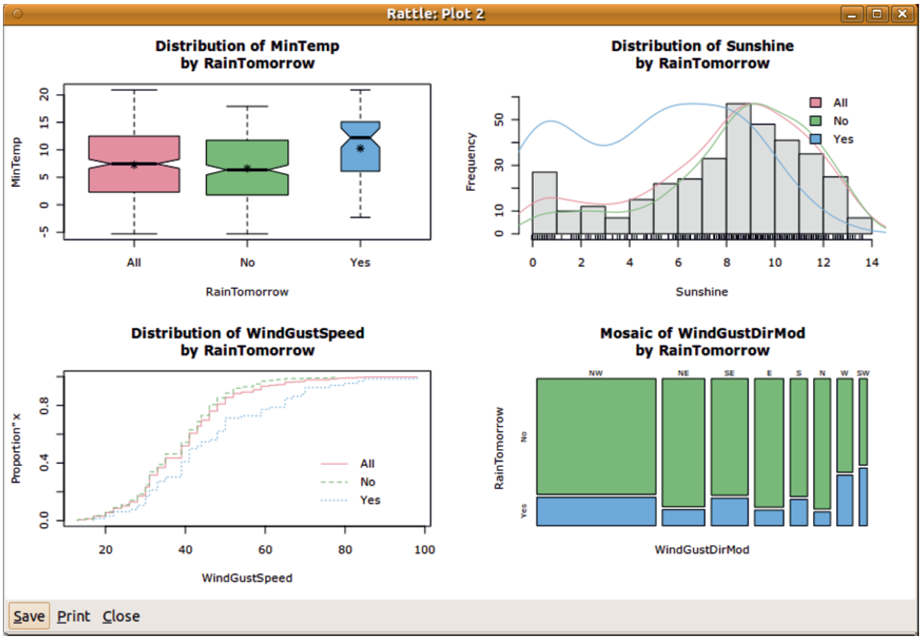


Figure 5.3: A sample of plots illustrates the different distributions and how they can be visualised.

5.2.1 Box Plot

A box plot (Tukey, 1977) (also known as a box-and-whisker plot) provides a graphical overview of how the observations of a variable are distributed. Rattle's box plot adds some additional information to the basic box plot provided by R.

A box plot is useful for quickly ascertaining the distribution of numeric data, covering some of the same statistics presented textually in Section 5.1.1. In particular, any skewness will be clearly visible.

When a target variable has been identified the box plot will also show the distribution of the observations of the chosen variable by the levels of the target variable. We see such a plot for the variable `Humidity3pm` in Figure 5.4, noting that `RainTomorrow` is the target variable. The width of each of the box plots also indicates the distribution of the values of the target variable. We see that there are quite a few more observations with `No` for `RainTomorrow` than with `Yes`.

The box plot (which is shown with Rattle's Annotate option active in Figure 5.4) presents a variety of statistics. The thicker horizontal

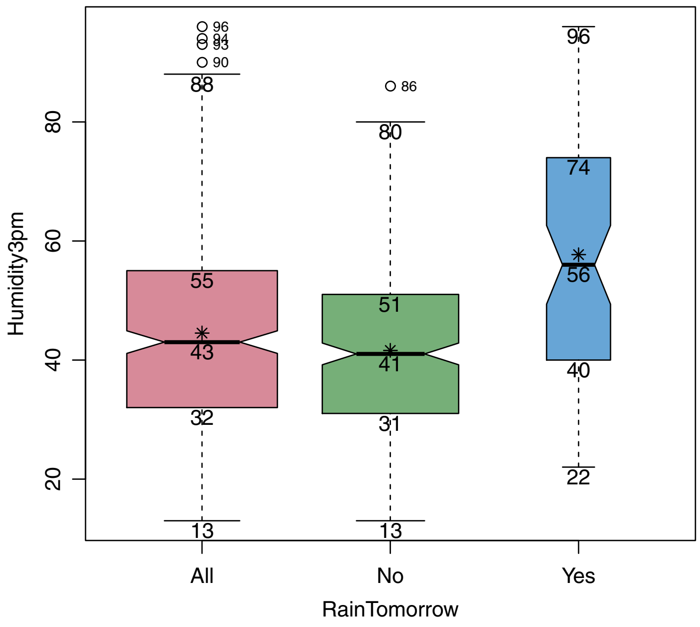


Figure 5.4: The Rattle box plot extends the default R box plots to provide a little more information by default and includes annotations if requested. The plot here is for the full dataset.

line within the box represents the median (also known as the second quartile or the 50th percentile). The leftmost box plot in Figure 5.4 (showing the distribution over all of the observations for `Humidity3pm`) has the median labelled as 43. The top and bottom extents of the box (55 and 32, respectively) identify the upper quartile (also known as the third quartile or the 75th percentile) and the lower quartile (the first quartile or the 25th percentile). The extent of the box is known as the interquartile range ($55 - 32 = 23$).

Dashed lines extend to the maximum and minimum data points, which are no more than 1.5 times the interquartile range from the median. We might expect most of the rest of the observations to be within this region. Outliers (points further than 1.5 times the interquartile range from the median) are then individually plotted (we can see a small number of outliers for the left two box plots, each being annotated with the actual value of the observation).

The notches in the box, around the median, indicate an approximate 95% confidence level for the differences between the medians (assuming independent observations, which may not be the case). Thus they are useful in comparing the distributions. In this instance, we can observe that the median of the values associated with the observations for which it rained tomorrow (i.e., the variable `RainTomorrow` has the value `Yes`) is significantly different (at the 95% level of confidence) from the median for those observations for which it did not rain tomorrow. It would appear that a higher humidity recorded at 3 pm is an indication that it might rain tomorrow.

The mean is also displayed as the asterisk in each of the boxes. A large gap between the median and the mean is another indication of a skewed distribution.

Rattle's Log tab records the sequence of commands used to draw the box plot and to annotate it. Basically, `boxplot()` (the basic plot), `points()` (to plot the means), and `text()` (to label the various points) are employed.

We can, as always, copy-and-paste these commands into the R Console to replicate the plot and to then manually modify the plot commands to suit any specific need. The automatically generated code is shown below, modified slightly for clarity.

The first step is to generate the data we wish to plot. The following example creates a single dataset with two columns, one being the observations of `Humidity3pm` and the other, identified by a variable called `grp`, the group to which the observation belongs. There are three groups, two corresponding to the two values of the target variable and the other covering all observations.

The use of `with()` allows the variables within the original dataset to be referenced without having to name the dataset each time. We combine three `data.frame()` objects row-wise, using `rbind()`, to generate the final dataset:

```
> ds <- with(crs$dataset[crs$train,],
     rbind(data.frame(dat=Humidity3pm,
                      grp="All"),
           data.frame(dat=Humidity3pm[RainTomorrow=="No"],
                      grp="No"),
           data.frame(dat=Humidity3pm[RainTomorrow=="Yes"],
                      grp="Yes")))
```

Now we display the boxplot(), grouping our data by the variable grp:

```
> bp <- boxplot(formula=dat ~ grp, data=ds,
                col=rainbow_hcl(3),
                xlab="RainTomorrow", ylab="Humidity3pm",
                notch=TRUE)
```

Notice that we assign to the variable bp the value returned by boxplot(). The function returns the data for the calculation needed to draw the box plot. By saving the result, we can make further use of it, as we do below, to annotate the plot.

We will also annotate the plot with the means. To do so, summaryBy() from **doBy** comes in handy. The use of points() together with pch= results in the asterisks we see in Figure 5.4.

```
> library(doBy)
> points(x=1:3, y=summaryBy(formula=dat ~ grp, data=ds,
                  FUN=mean, na.rm=TRUE)$dat.mean, pch=8)
```

Next, we add further text() annotations to identify the median and interquartile range:

```
> for (i in seq(ncol(bp$stats)))
  {
    text(x=i, y=bp$stats[,i] -
         0.02*(max(ds$dat, na.rm=TRUE) -
               min(ds$dat, na.rm=TRUE)),
         labels=bp$stats[,i])
  }
```

The outliers are then annotated using text(), but decreasing the font size using cex=:

```
> text(x=bp$group+0.1, y=bp$out, labels=bp$out, cex=0.6)
```

To round out our plot, we add a title() to include a main= and a sub= title. We format() the current date and time (Sys.time()) and include the current user (obtained from Sys.info()) in the titles:

```
> title(main="Distribution of Humidity3pm (sample)",
        sub=paste("Rattle",
          format(Sys.time(), "%Y-%b-%d %H:%M:%S"),
          Sys.info()["user"]))
```

A variation of the box plot is the box-percentile plot. This plot provides more information about the distribution of the values. We can see such a plot in Figure 5.5, which is generated using `bpplot()` of **Hmisc**. The following code will generate the plot (at the time of writing this book, box-percentile plots are not yet available in Rattle):

```
> library(Hmisc)
> h3 <- weather$Humidity3pm
> hn <- h3[weather$RainTomorrow=="No"]
> hy <- h3[weather$RainTomorrow=="Yes"]
> ds <- list(h3, hn, hy)
> bpplot(ds, name=c("All", "No", "Yes"),
         ylab="Humidity3pm", xlab="RainTomorrow")
```

The width within each box (they aren't quite boxes as such, but we get the idea) is determined to be proportional to the number of observations that are below (or above) that point. The median and the 25th and 75th percentiles are also shown.

5.2.2 Histogram

A histogram provides a quick and useful graphical view of the spread of the data. We can very quickly get a feel for the distribution of our data, including an idea of its skewness and kurtosis. Histograms are probably one of the more common ways of visually presenting data.

A histogram plot in Rattle includes three components, as we see in Figure 5.6. The first of these is obviously the vertical bars. The continuous data in the example here (the wind speed at 9 am) has been partitioned into ranges, and the frequency of each range is displayed as the bar. R automatically chooses both the partitioning and how the x-axis is labelled, showing x-axis points at 0, 10, 20, and so on. We might observe that the most frequent range of values is in the 4–6 partition.

The plot also includes a line plot showing the so-called density estimate. The density plot is a more accurate display of the actual (at least estimated true) distribution of the data (the values of `WindSpeed9am`).

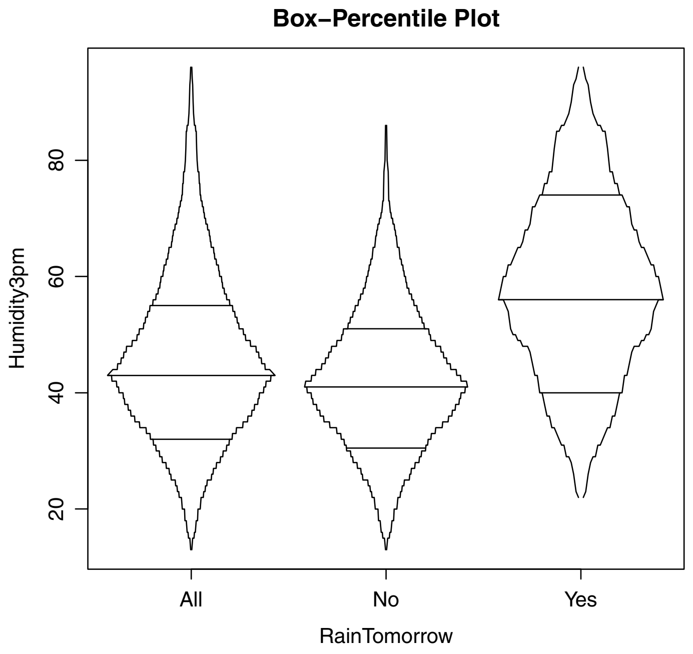


Figure 5.5: A box-percentile plot provides some more information about the distribution.

It allows us to see that rather than values in the range 4–6 occurring frequently, in fact it is "6" itself that occurs most frequently.

The third element of the plot is the so-called *rug* along the bottom of the plot. The rug is a single-dimensional plot of the data along the number line. It is useful in seeing exactly where data points actually lie. For large collections of data with a relatively even spread of values, the rug ends up being quite black. From Figure 5.6, we can make some observations about the data. First, it is clear that the measure of wind speed is actually an integer. Presumably, in the source data, it is rounded to the nearest integer. We can also observe that some values are not represented at all in the dataset. In particular, we can see that 0, 2, 4, 6, 7, and 9 are represented in the data but 1, 3, 5, and 8 are not.

The distribution of the values for `WindSpeed9am` is also clearly skewed, having a longer tail to the right than to the left. Recall from Section 5.1.4 that `WindSpeed9am` had a skewness of 1.36. Similarly, the kurtosis measure was 1.48, indicating a bit of a narrower peak.

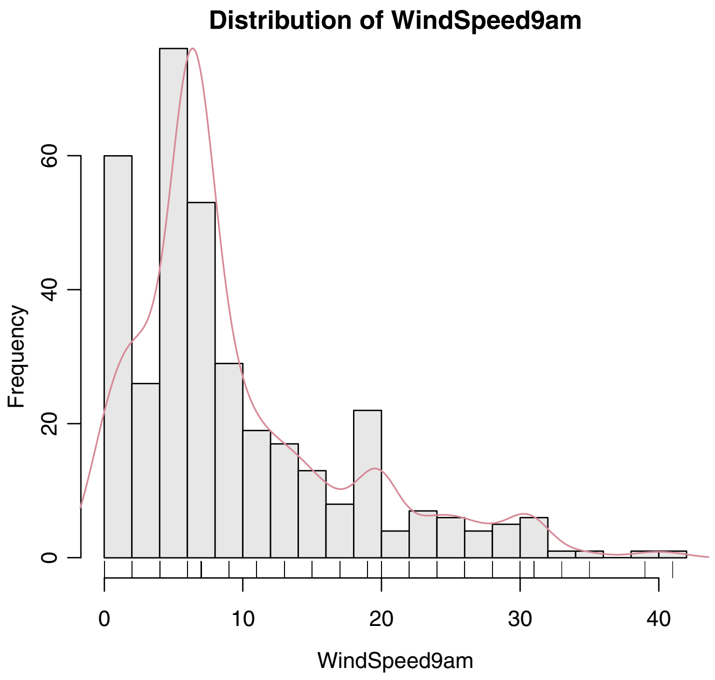


Figure 5.6: The Rattle histogram extends the default R histogram plots with a density plot and a rug plot.

We can compare `WindSpeed9am` with `Sunshine`, as in Figure 5.7. The corresponding skewness and kurtosis for `Sunshine` are −0.72 and −0.27, respectively. That is, `Sunshine` has a smaller and negative skew, and a smaller kurtosis and hence a more spread-out peak.

5.2.3 Cumulative Distribution Plot

Another popular plot for communicating the distribution of the values of a variable is the cumulative distribution plot. A cumulative distribution plot displays the proportion of the data that has a value that is less than or equal to the value shown on the x-axis.

Figure 5.8 shows a cumulative distribution plot for two variables, `WindSpeed9am` and `Sunshine`. Each chart includes three cumulative plots: one line is drawn for all the data and one line for each of the values of the target variable.

We can see again that these two variables have quite different distri-

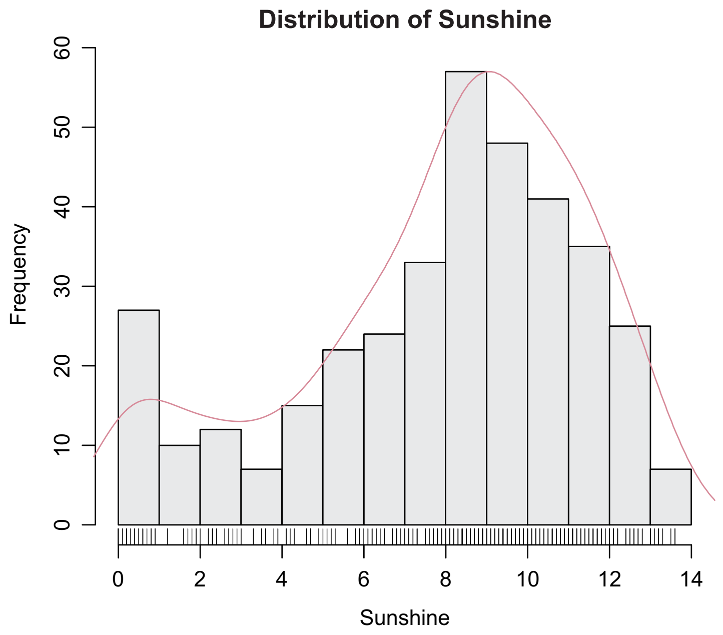
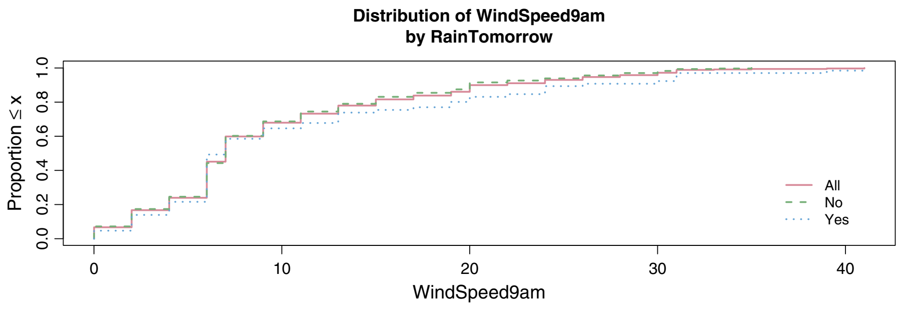


Figure 5.7: The Rattle histogram for `Sunshine` for comparison with `WindSpeed9am`.

butions. The plot for `WindSpeed9am` indicates that the wind speed at 9 am is usually at the lower end of the scale (e.g., less than 10), but there are a few days with quite extreme wind speeds at 9 am (i.e., outliers). For `Sunshine` there is a lot more data around the middle, which is typical of a more normal type of distribution. There is quite a spread of values between 6 and 10.

The `Sunshine` plot is also interesting. We can see quite an obvious difference between the two lines that represent `All` of the observations and just those with a `No` (i.e., observations for which there is no rain tomorrow) and the line that represents the `Yes` observations. It would appear that lower values of `Sunshine` today are associated with observations for which it rains tomorrow.

The `Ecdf()` command of **Hmisc** provides a simple interface for producing cumulative distribution plots. The code to generate the `Sunshine` plot is presented below.



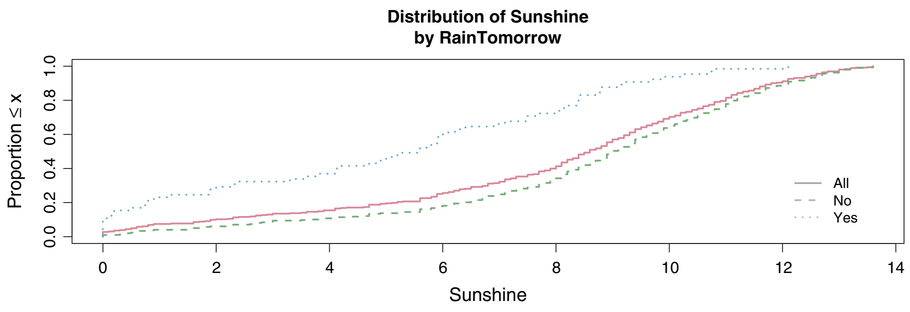


Figure 5.8: Cumulative distribution plots for `WindSpeed9am` and `Sunshine`.

```
> library(rattle)
> library(Hmisc)
> su <- weather$Sunshine
> sn <- su[weather$RainTomorrow=="No"]
> sy <- su[weather$RainTomorrow=="Yes"]
> Ecdf(su, col="#E495A5", xlab="Sunshine", subtitles=FALSE)
> Ecdf(sn, col="#86B875", lty=2, add=TRUE, subtitles=FALSE)
> Ecdf(sy, col="#7DB0DD", lty=3, add=TRUE, subtitles=FALSE)
```

We can add a legend and a title to the plot:

```
> legend("bottomright", c("All","No","Yes"), bty="n",
         col=c("#E495A5", "#86B875", "#7DB0DD"),
         lty=1:3, inset=c(0.05,0.05))
> title(main=paste("Distribution of Sunshine (sample)",
          "by RainTomorrow", sep="\n"),
        sub=paste("Rattle", format(Sys.time(),
          "%Y-%b-%d %H:%M:%S")))
```

5.2.4 Benford's Law

The use of Benford's law has proven to be effective in identifying oddities in data. It has been used for case selection in fraud detection, particularly in accounting data (Durtschi et al., 2004), where the value of a variable for a group of related observations might be identified as not conforming to Benford's law even though other groups do.

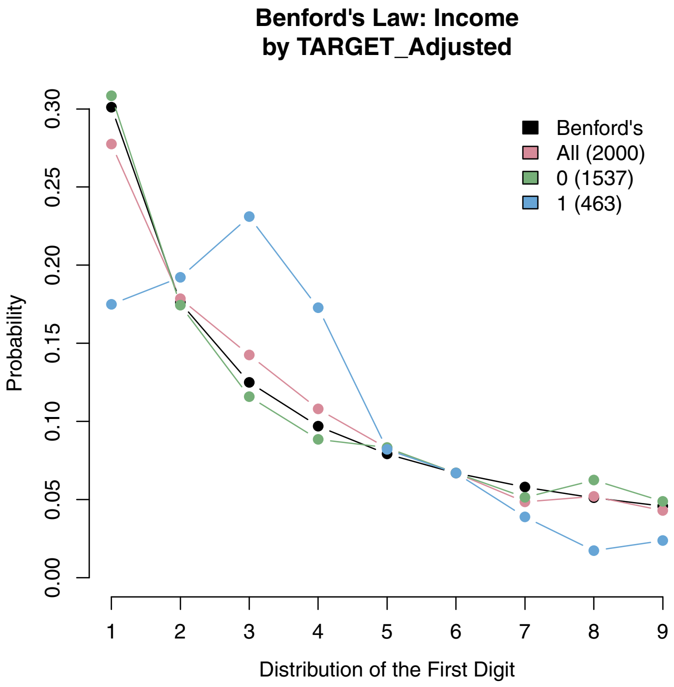


Figure 5.9: A Benford's law plot of the variable `Income` from the *audit* dataset, particularly showing nonconformance for the population of known noncompliant clients.

Benford's law relates to the frequency of occurrence of the first digit in a collection of numbers. These numbers might be the dollar income earned by individuals across a population of taxpayers or the height of buildings in a city. The law generally applies when several orders of magnitude (e.g., 10, 100, and 1000) are recorded in the observations.

The law states that the digit "1" appears as the first digit of the numbers some 30% of the time. That is, for income, numbers like \$13,245 and \$162,385 (having an initial digit of "1") will appear about 30% of the time in our population. On the other hand, the digit "9" (for example, as in $94,251) appears as the first digit less than 5% of the time. Other digits have frequencies between these two, as we can see from the black line in Figure 5.9.

This rather startling observation is certainly found, empirically, to hold in many collections of numbers, such as bank account balances, tax refunds, stock prices, death rates, lengths of rivers, and potential fraud in elections. It is observed to hold for processes that are described by what are called power laws, which are common in nature. By plotting a collection of numbers against the expectation as based on Benford's law, we are able to quickly see any odd behaviour in the data.

Benford's law is not valid for all collections of numbers. For example, peoples' ages would not be expected to follow Benford's law, nor would telephone numbers. So we do need to use caution in relying just on Benford's law to identify cases of interest.

We can illustrate Benford's law using the *audit* dataset from **rattle**. Rattle provides a convenient mechanism for generating a plot to visualise Benford's law, and we illustrate this with the variable `Income` in Figure 5.9.

The darker line corresponds to Benford's law, and we note that the lines corresponding to `All` and `0` follow the expected first-digit distribution proposed by Benford's law. However, the line corresponding to `1` (i.e., clients who had to have their claims adjusted) clearly deviates from the proposed distribution. This might indicate that these numbers have been made up by someone or that there is some other process happening that affects this population of numbers.

5.2.5 Bar Plot

The plots we have discussed so far work for numeric data. We now consider plots that work for categoric data. These include the bar plot, dot plot, and mosaic plot.

A bar plot, much like a histogram, uses vertical bars to show counts of the number of observations of each of the possible values of the categoric variable. There are many ways to graph a bar plot. In Rattle, the default is to list the possible values along the x-axis, leaving the y-axis for the frequency or count. When a categoric target variable is active within Rattle, additional bars will be drawn for each value, corresponding to the different values of the target. Figure 5.10 shows a typical bar plot. The sample bar plot also shows that by default Rattle will sort the bars from the largest to the smallest.

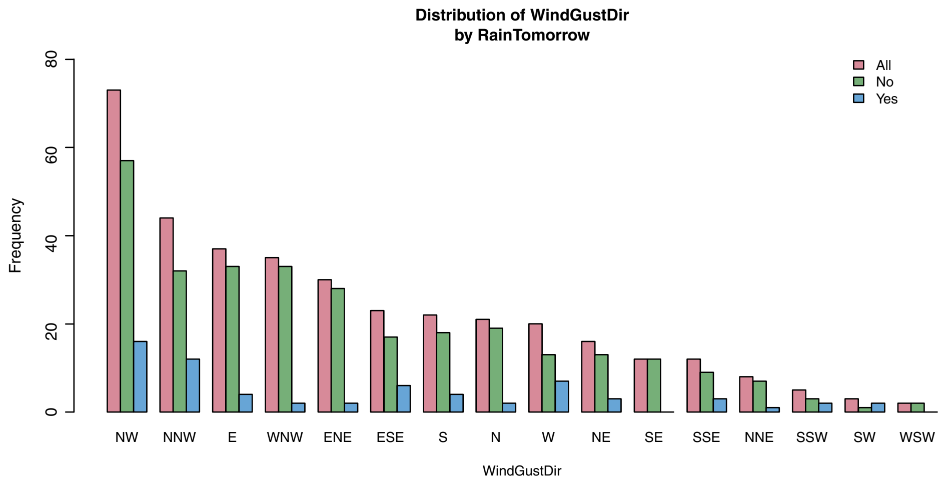


Figure 5.10: A bar plot for the categoric variable `WindGustDir` (modified) from the *weather* dataset.

5.2.6 Dot Plot

A dot plot illustrates much the same information as a bar plot but uses a different approach. The bars are replaced by dots that show the height, thus not filling in as much of the graphic. A dotted line replaces the extent of the bar, and by default in Rattle the plots are horizontal rather than vertical. Once again, the categoric values are ordered to produce the plot in Figure 5.11.

For the dot plot, we illustrate the distribution of observations over all of the values of the categoric variable `WindGustDir`. With the horizontal plot it is more feasible to list all of the values than for the vertical bar plots. Of course, the bar plots could be drawn horizontally for the same reason.

Both the bar plot and dot plot are useful in understanding how the observations are distributed across the different categoric values. One thing to look for when a target variable is identified is any specific variation in the distributions between the target values. In Figure 5.11, for example, we can see that the distributions of the "Yes" and "No" observa-

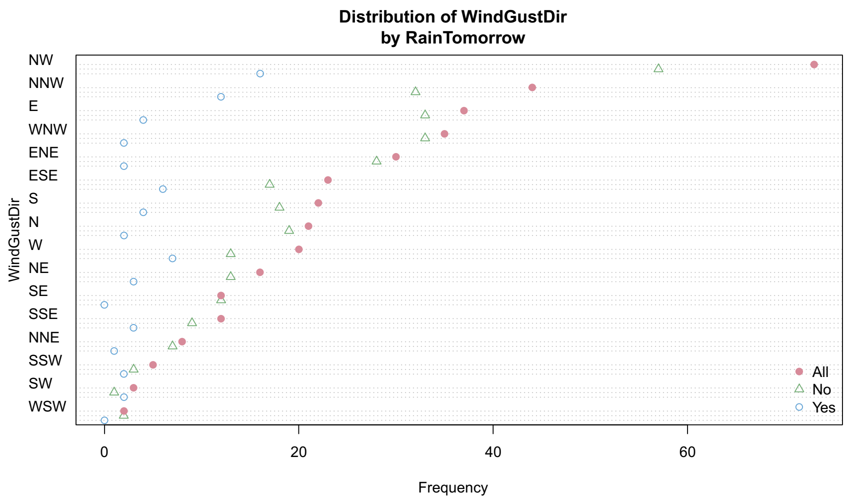


Figure 5.11: A dot plot for the categoric variable `WindGustDir` (original) from the *weather* dataset.

tions are quite different from the overall distributions. Such observations may merely be interesting or might lead to useful questions about the data. In our data here, we do need to recall that the majority of days have no rain. Thus the `No` distribution of values for `WindGustDir` follows the distribution of all observations quite closely. We can see a few deviations, suggesting that these wind directions have an influence on the target variable.

5.2.7 Mosaic Plot

A mosaic plot is an effective way to visualise the distribution of the values of one variable over the different values of another variable, looking for any structure or relationship between those variables. In Rattle, this second variable is usually the target variable (e.g., `RainTomorrow` in our *weather* dataset), as in Figure 5.12.

The mosaic plot again provides insights into how the data is distributed over the values of a second variable. The area of each bar is proportional to the number of observations having a particular value for the variable `WindGustDir`. Once again, the values of `WindGustDir` are ordered according to their frequency of occurrence. The value `NW` is observed most frequently. The split between the values of the target

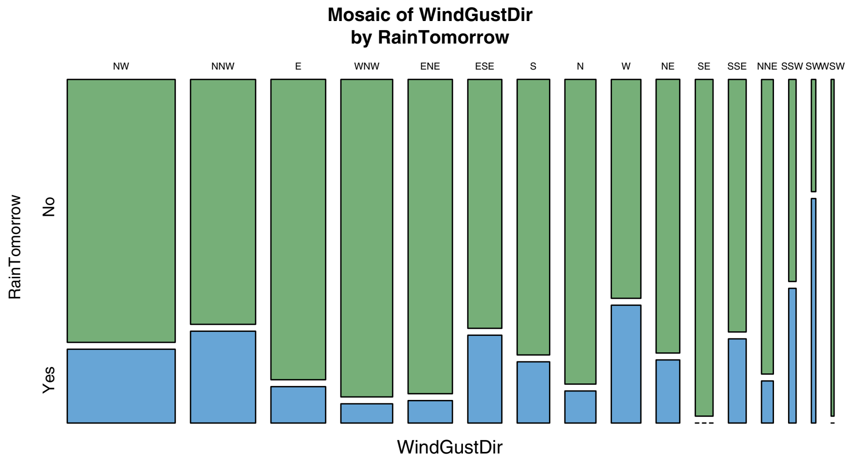


Figure 5.12: A mosaic plot for the categoric variable `WindGustDir` (original) from the *weather* dataset.

variable `RainTomorrow` is similarly proportional to their frequency.

Once again, we see that a wind gust of west has a high proportion of days for which `RainTomorrow` is true. Something that we missed in reviewing the bar and the dot plots is that a `SW` wind gust has the highest proportions of days where it rains tomorrow, followed by `SSW`.

It is arguable, though, that it is harder to see the overall distribution of the wind gust directions in a mosaic plot compared with the bar and dot plots. Mosaic plots are thus generally used in combination with other plots, and they are particularly good for comparing two or more variables at the same time.

5.2.8 Pairs and Scatter Plots

The bar and dot plots are basically single-variable (i.e., univariate) plots. In our plots, we have been including a second variable, the target. Moving on from considering the distribution of a single variable at a time, we can compare variables pairwise. Such a plot is called a **scatter plot**.

Generally we have multiple variables that we might wish to compare pairwise using multiple scatter plots. Such a plot then becomes a **scatter plot matrix**. The `pairs()` command in R can be used to generate a matrix of scatter plots. In fact, the function can be fine-tuned to not only display pairwise scatter plots but also to include histograms and a pairwise measure of the correlation between variables (correlations are discussed in Section 5.3).

For this added functionality we need two support functions that are a little more complex and that we won't explain in detail:

```
> panel.hist <- function(x, ...)
  {
    usr <- par("usr"); on.exit(par(usr))
    par(usr=c(usr[1:2], 0, 1.5) )
    h <- hist(x, plot=FALSE)
    breaks <- h$breaks; nB <- length(breaks)
    y <- h$counts; y <- y/max(y)
    rect(breaks[-nB], 0, breaks[-1], y, col="grey90", ...)
  }
> panel.cor <- function(x, y, digits=2, prefix="",
                        cex.cor, ...)
  {
    usr <- par("usr"); on.exit(par(usr))
    par(usr = c(0, 1, 0, 1))
    r <- (cor(x, y, use="complete"))
    txt <- format(c(r, 0.123456789), digits=digits)[1]
    txt <- paste(prefix, txt, sep="")
    if(missing(cex.cor)) cex.cor <- 0.8/strwidth(txt)
    text(0.5, 0.5, txt)
  }
```

We can then generate the plot with:

```
> vars <- c(5, 7, 8, 9, 15, 24)
> pairs(weather[vars],
        diag.panel=panel.hist,
        upper.panel=panel.smooth,
        lower.panel=panel.cor)
```

There are two additional commands defined here, `panel.hist()` and `panel.cor()`, provided as the arguments `diag.panel` and `lower.panel` to `pairs()`. These two commands are not provided by R directly. Their definitions can be obtained from the help page for `pairs()` and pasted into the R Console.

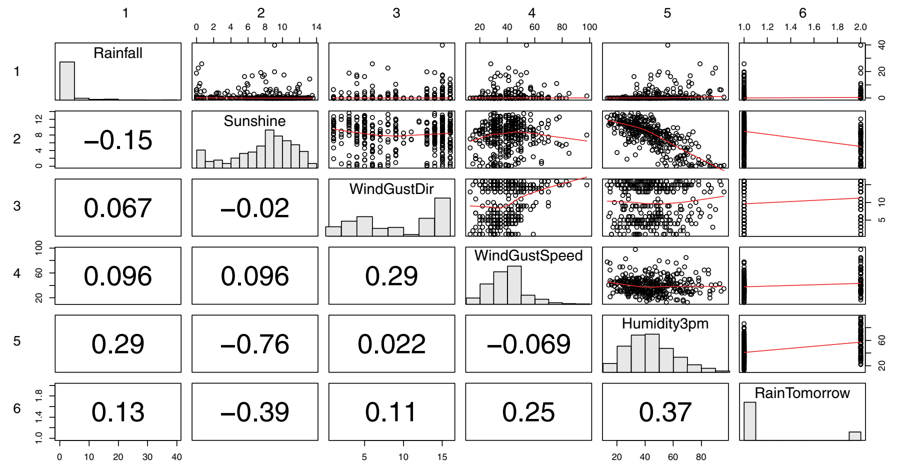


Figure 5.13: A pairs plot with a scatter plot matrix displayed in the upper panel, histograms in the diagonal, and a measure of correlation in the lower panel.

Tip: *Rattle can generate a pairs plot including the scatter plots, histograms, and correlations in the one plot (Figure 5.13). To do so, we go to the Distributions option of the Explore tab and ensure that no plot types are selected for any variable (the default). Then click the Execute button.*

Notice that we have only included six variables in the pairs plot. Any more than this and the plot becomes somewhat crowded. In generating a pairs plot, Rattle will randomly subset the total number of variables available down to just six variables. In fact, each time the Execute button is clicked, a different randomly selected collection of variables will be displayed. This is a useful exercise to explore for interesting pairwise relationships among our variables. If we are keen to do so, we can generate plots with more than six variables quite simply by copying the command from Rattle's Log tab (which will be similar to `pairs()`, shown above) and pasting it into the R Console.

Let's explore the pairs plot in a little more detail. The diagonal contains a histogram for the numeric variables and a bar plot (also a histogram) for the categoric variables. The top right plots (i.e., those plots above the diagonal) are pairwise scatter plots, which plot the observations of just two variables at a time. The corresponding variables are identified from the diagonal.

The top left scatter plot, which appears in the first row and the second column, has `Rainfall` on the y-axis and `Sunshine` on the x-axis. We can see quite a predominance of days (observations) with what looks like no rainfall at all, with fewer observations having some rainfall. There does not appear to be any particular relationship between the amount of rain and the hours of sunshine, although there are some days with higher rainfall when there is less sunshine. Note, though, that an outlier for rainfall (at about 40 mm of rain) appears on a day with about 9 hours of sunshine. We might decide to explore this apparent anomaly to assure ourselves that there was no measurement or data error that led to this observation.

An interesting scatter plot to examine is that in row 2 and column 5. This plot has `Sunshine` on the y-axis and `Humidity3pm` on the x-axis. The solid red lines that are drawn on the plot are a result of `panel.smooth()` being provided as the value for the `upper.panel` argument in the call to `pairs()`. The line provides a hint of any trend in the relationship between the two variables. For this particular scatter plot, we can see some structure in that higher levels of humidity at 3 pm are observed with lower hours of sunshine. One or two of the other scatter plots show other, but less pronounced and hence probably less significant, relationships.

The lower part of our scatter plot matrix contains numbers between -1 and 1. These are measures of the correlation between two variables. Pearson's correlation coefficient is used. We can see that `Rainfall` and `Humidity3pm` (see the number in row 5, column 1) have a small positive correlation of 0.29. That is not a great deal of correlation. If we square the correlation value to obtain 0.0841, we can interpret this as indicating that some 8% of the variation is related. There is perhaps some basis to expect that when we observe higher rainfall we might also observe higher humidity at 3 pm.

There is even stronger correlation between the variables `Sunshine` and `Humidity3pm` (row 5, column 2) measured at -0.76. The negative sign indicates a negative correlation of strength 0.76. Squaring this number leads us to observe that some 58% of the variation is related. Thus, observations of more sunshine do tend to occur with observations of less humidity at 3 pm, as we have already noted. We will come back to correlations shortly.

5.2.9 Plots with Groups

Extending the idea of comparing variables, we can usefully plot, for example, a box plot of one variable but with the observations split into groups that are defined by another variable. We have already seen this with the target variable being the one by which we group our observations, as in Figure 5.4. Simply through selecting another variable as the target, we can explore many different relationships quite effectively.

Consider, for example, the distribution of the observations of the variable `Sunshine`. We might choose `Cloud9am` as the target variable (in Rattle's Data tab) and then request a box plot from the Explore tab. The result will be as in Figure 5.14. Note that `Cloud9am` is actually a numeric variable, but we are effectively using it here as a categoric variable. This is okay since it has only nine numeric values and those are used here to group together different observations (days) having a common value for the variable.

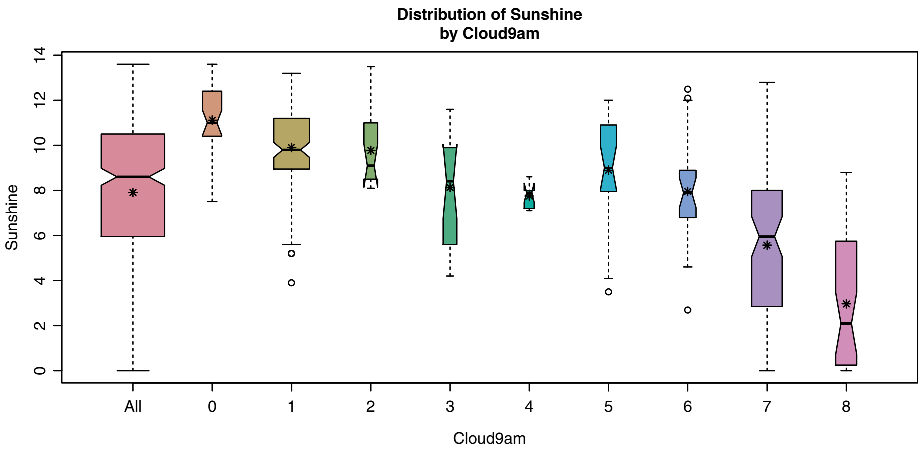


Figure 5.14: Data displayed through the Distributions tab is grouped using the target variable values to define the groups. Selecting alternative targets will group the data differently.

The leftmost box plot shows the distribution of the observations of `Sunshine` over the whole dataset. The remaining nine box plots then collect together the observations of `Sunshine` for each of the nine possible values of `Cloud9am`. Recall that `Cloud9am` is measured in something called oktas. An okta of 0 indicates no cloud coverage, 1 indicates one-eighth of the sky is covered, and so on up to 8, indicating that the sky is completely covered in clouds.

The relationship that we see in Figure 5.14 then makes some sense. There is a clear downward trend in the box plots for the amount of sunshine as we progressively have more cloud coverage. Some groups are quite distinct: compare groups 6, 7, and 8. They have different medians, with their notches clearly not overlapping.

The plot also illustrates some minor idiosyncrasies of the box plot. The box plots for groups 2 and 4 appear a little different. Each has an odd arrangement at the end of the quartile on one side of the box. This occurs when the notch is calculated to be larger than the portion of the box on one side of the median.

5.3 Correlation Analysis

We have seen from many of the plots in the sections above, particularly those plots with more than a single variable, that we often end up identifying some kind of relationship or correlation between the observations of two variables. The relationship we saw between `Sunshine` and `Humidity3pm` in Figure 5.13 is one such example.

A correlation coefficient is a measure of the degree of relationship between two variables—it is usually a number between −1 and 1. The magnitude represents the strength of the correlation and the sign represents the direction of the correlation. A high degree of correlation (closer to 1 or −1) indicates that the two variables are very highly correlated, either positively or negatively. A high positive correlation indicates that observations with a high value for one variable will also tend to have a high value for the second variable. A high negative correlation indicates that observations with a high value for one variable will also tend to have a lower value of the second variable. Correlations of 1 (or −1) indicate that the two variables are essentially identical, except perhaps for scale (i.e., one variable is just a multiple of the other).

5.3.1 Correlation Plot

From our previous exploration of the *weather* dataset, we noted a moderate (negative) correlation between `Sunshine` and `Humidity3pm`. Generally, days with a higher level of sunshine have a lower level of humidity at 3 pm and vice versa.

Variables that are very strongly correlated are probably not independent. That is, they have some close relationship. The relationship could be causal in that an increase in one has some physical impact on the other. But such evidence for this needs to be ascertained separately. Nonetheless, having correlated variables as input to some algorithms may misguide the data mining. Thus it is important to recognise this.

R can be used to quite easily generate a matrix of correlations between variables. The `cor()` command will calculate and list the Pearson correlation between variables:

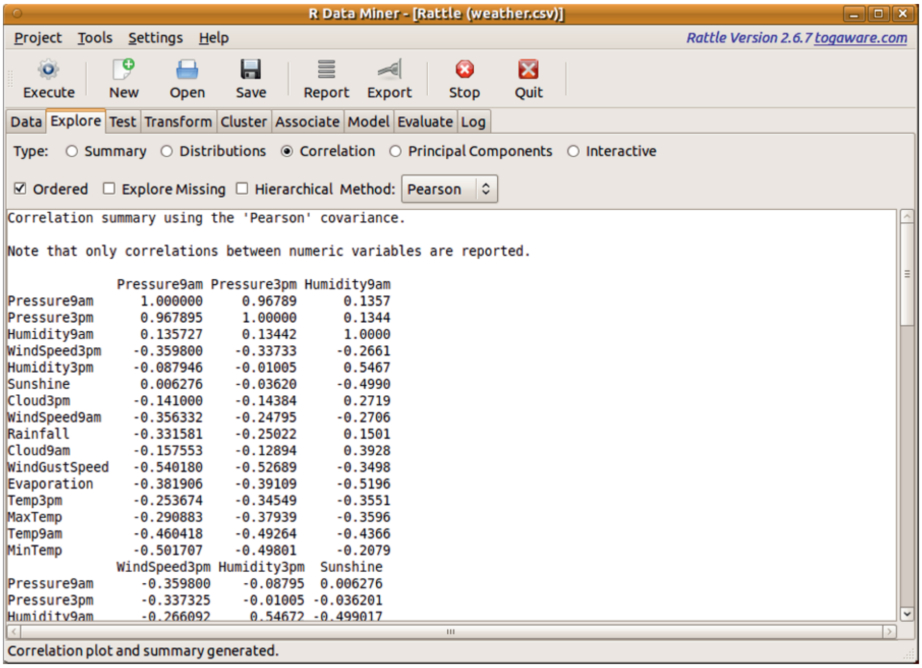
```
> vars <- c(5, 6, 7, 9, 15)
> cor(weather[vars], use="pairwise", method="pearson")

               Rainfall Evaporation Sunshine
Rainfall       1.000000   -0.007293 -0.15099
Evaporation   -0.007293    1.000000  0.31803
Sunshine      -0.150990    0.318025  1.00000
WindGustSpeed  0.096190    0.288477  0.09584
Humidity3pm    0.289013   -0.391780 -0.75943
              WindGustSpeed Humidity3pm
Rainfall            0.09619     0.28901
Evaporation         0.28848    -0.39178
Sunshine            0.09584    -0.75943
WindGustSpeed       1.00000    -0.06944
Humidity3pm        -0.06944     1.00000
```

We can compare these numbers with those in Figure 5.13. They should agree. Note that each variable is, of course, perfectly correlated with itself, and that the matrix here is symmetrical about the diagonal (i.e., the measure of the correlation between `Rainfall` and `Sunshine` is the same as that between `Sunshine` and `Rainfall`).

We have to work a little hard to find patterns in the matrix of correlation values expressed in this way. Rattle provides access to a graphical plot of the correlations between variables in our dataset. The Correlation option of the Explore tab provides a number of choices for correlation plots (Figure 5.15). Simply clicking the Execute button will cause the default correlation plot to be displayed (Figure 5.16).

The first thing we might notice about this correlation plot is that only the numeric variables appear. Rattle only computes correlations between numeric variables. The second thing to note about the plot is


R Data Miner - [Rattle (weather.csv)]
Project Tools Settings Help
Rattle Version 2.6.7 togaware.com
Execute New Open Save Report Export Stop Quit
Data Explore Test Transform Cluster Associate Model Evaluate Log

Type: Summary Distributions [✓ Correlation] Principal Components Interactive

[✓ Ordered] Explore Missing Hierarchical Method: Pearson

```
Correlation summary using the 'Pearson' covariance.

Note that only correlations between numeric variables are reported.

              Pressure9am Pressure3pm Humidity9am
Pressure9am      1.000000     0.96789      0.1357
Pressure3pm      0.967895     1.00000      0.1344
Humidity9am      0.135727     0.13442      1.0000
WindSpeed3pm    -0.359800    -0.33733     -0.2661
Humidity3pm     -0.087946    -0.01005      0.5467
Sunshine         0.006276    -0.03620     -0.4990
Cloud3pm        -0.141000    -0.14384      0.2719
WindSpeed9am    -0.356332    -0.24795     -0.2706
Rainfall        -0.331581    -0.25022      0.1501
Cloud9am        -0.157553    -0.12894      0.3928
WindGustSpeed   -0.540180    -0.52689     -0.3498
Evaporation     -0.381906    -0.39109     -0.5196
Temp3pm         -0.253674    -0.34549     -0.3551
MaxTemp         -0.290883    -0.37939     -0.3596
Temp9am         -0.460418    -0.49264     -0.4366
MinTemp         -0.501707    -0.49801     -0.2079
              WindSpeed3pm Humidity3pm  Sunshine
Pressure9am      -0.359800    -0.08795  0.006276
Pressure3pm      -0.337325    -0.01005 -0.036201
Humidity9am      -0.266092     0.54672 -0.499017
```

Correlation plot and summary generated.

Figure 5.15: The Explore tab's Correlation option provides access to plots that visualise correlations between pairs of variables.

that it is symmetric about the diagonal, as is the numeric correlation matrix we saw above—the correlation between two variables is the same, irrespective of the order in which we view them. The third thing to note is that the order of the variables does not correspond to the order in the dataset but to the order of the strength of any correlations, from the least to the greatest. This is done to achieve a more pleasing graphic but can also lead to further insight with groupings of similar correlations. This is controlled through the Ordered check button.

We can understand the degree of any correlation between two variables by both the shape and the colour of the graphic elements. Any variable is, of course, perfectly correlated with itself, and this is reflected as the straight lines on the diagonal of the plot.

A perfect circle, on the other hand, indicates that there is no (or very little) correlation between the variables. This appears to be the case, for example, for the correlation between `Sunshine` and `Pressure9am`. In fact, there is a correlation, just an extremely weak one (0.006), as we see in Figure 5.15.

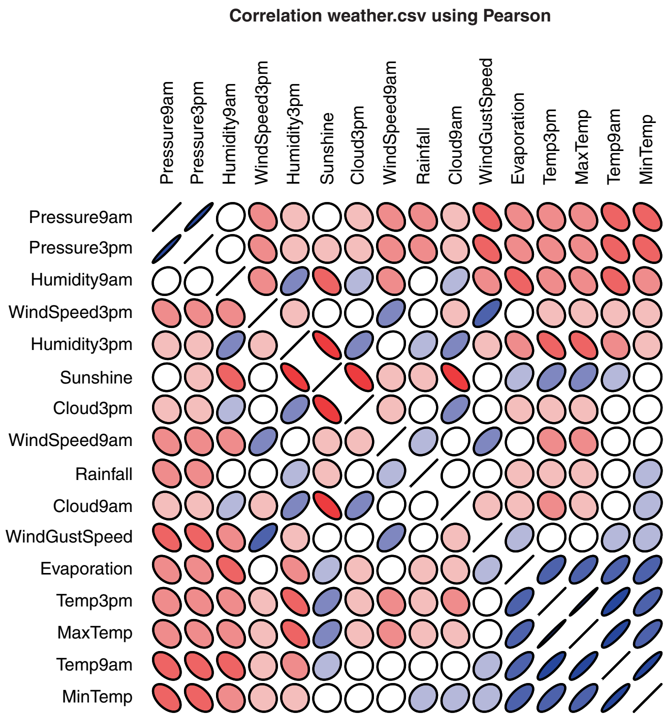


Figure 5.16: The correlation plot graphically displays different degrees of correlation pairwise between variables.

The circles turn into straight lines, by degrees, as the strength of correlation between the two variables increases. Thus we can see that there is some moderate correlation between `Humidity9am` and `Humidity3pm`, represented as the squashed circle (i.e., an ellipse shape). The more squashed (i.e., the more like a straight line), the higher the degree of correlation, as in the correlation between `MinTemp` and `Temp9am`. Notice that, intuitively, all of the observations of correlations make some sense.

The direction of the ellipse indicates whether the correlation is positive or negative. The correlations we noted above were in the positive direction. We can see, for example, our previously observed negative correlation between `Sunshine` and `Humidity3pm`.

The colours used to shade the ellipses give another, if redundant, clue to the strength of the correlation. The intensity of the colour is maximal (black) for a perfect correlation and minimal (white) if there is no correlation. Shades of red are used for negative correlations and blue for positive correlations.

5.3.2 Missing Value Correlations

An interesting and useful twist on the concept of correlation analysis is the concept of correlation amongst missing values in our data. In many datasets, it is often constructive to understand the nature of missing data. We often find commonality amongst observations with a missing value for one variable having missing values for other variables. A correlation plot can effectively highlight such structure in our datasets.

The correlation between missing values can be explored by clicking the Explore Missing check box. To understand missing values fully, we have also turned off the partitioning of the dataset on the Data tab so that all of the data is considered for the plot. The resulting plot is shown in Figure 5.17.

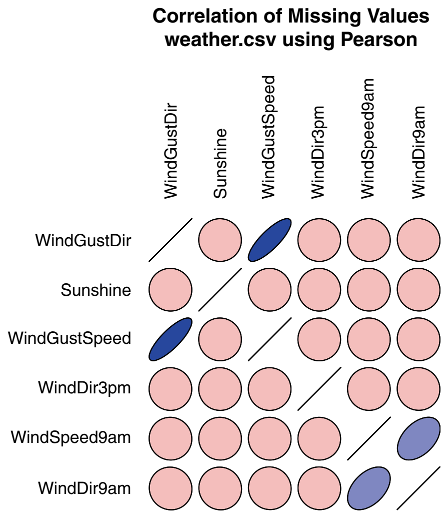


Figure 5.17: The missing values correlation plot showing correlations between missing values of variables.

We notice immediately that only six variables are included in this correlation plot. Rattle has identified that the other variables have no missing values, and so there is no point including them in the plot. We also notice that a categoric variable, `WindGustDir`, is included in the plot even though it was not included in the usual correlation plot. We can obtain a correlation for categoric variables since we only measure the absence or presence of a value, which is easily interpreted as numeric.

The graphic shows us that `WindGustSpeed` and `WindGustDir` are quite highly correlated with respect to missing values. That is, when the variable `WindGustSpeed` has a missing value, `WindGustDir` also tends to have a missing value, and vice versa. The actual correlation is 0.81489 (which can be read from the Rattle text view window). There is also a weak correlation between `WindDir9am` and `WindSpeed9am` (0.36427).

On the other hand, there is no (in fact, very little at −0.0079) correlation between `Sunshine` and `WindGustSpeed`, or any other variable, with regard to missing values.

It is important to note that the correlations showing missing values may be based on very small samples, and this information is included in the text view of the Rattle window. For example, in this case we can see in Figure 5.18 that there are only 21 missing observations for `WindDir9am` and only two or three for the other variables. This corresponds to approximately 8% and 1% of the observations, respectively, having missing values for these variables. This is too little to draw too many conclusions from.

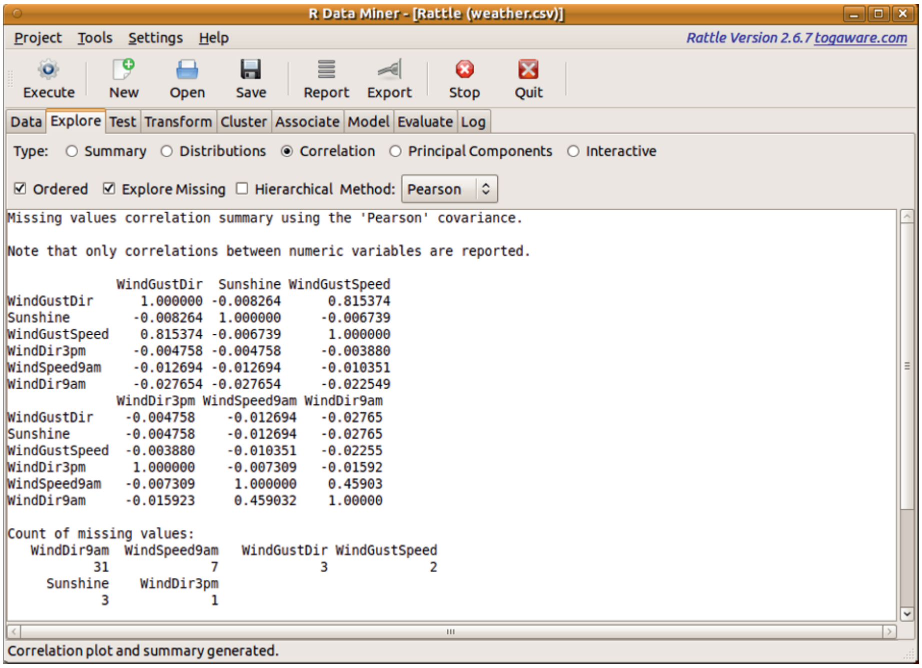
5.3.3 Hierarchical Correlation

Another useful option provided by Rattle is the hierarchical correlation plot (Figure 5.19). The plot provides an overview of the correlation between variables using a tree-like structure known as a dendrogram.

The plot lists the variables in the right column. The variables are then linked together in the dendrogram according to how well they are correlated. The x-axis is a measure of the height within the dendrogram, ranging from 0 to 3. The heights (i.e., lengths of the lines within the dendrogram) give an indication of the level of correlation between variables, with shorter heights indicating stronger correlations.

Very quickly we can observe that `Temp3pm` and `MaxTemp` are quite closely correlated (in fact, they have a correlation of 0.99). Similarly, `Cloud3pm` and `Cloud9am` are moderately correlated (0.51). The group of variables `Temp9am`, `MinTemp`, `Evaporation`, `Temp3pm`, and `MaxTemp`, unsurprisingly, have some higher level of correlation amongst themselves than they do with other variables.

A number of R functions are used together to generate the plot we see in Figure 5.19. We take the opportunity to review the R code to gain


R Data Miner - [Rattle (weather.csv)]
Project Tools Settings Help
Rattle Version 2.6.7 togaware.com
Execute New Open Save Report Export Stop Quit
Data Explore Test Transform Cluster Associate Model Evaluate Log

Type: ○ Summary ○ Distributions [✓ Correlation] ○ Principal Components ○ Interactive

[✓ Ordered] [✓ Explore Missing] ☐ Hierarchical Method: Pearson

```
Missing values correlation summary using the 'Pearson' covariance.

Note that only correlations between numeric variables are reported.

              WindGustDir  Sunshine WindGustSpeed
WindGustDir      1.000000 -0.008264      0.815374
Sunshine        -0.008264  1.000000     -0.006739
WindGustSpeed    0.815374 -0.006739      1.000000
WindDir3pm      -0.004758 -0.004758     -0.003880
WindSpeed9am    -0.012694 -0.012694     -0.010351
WindDir9am      -0.027654 -0.027654     -0.022549
              WindDir3pm WindSpeed9am WindDir9am
WindGustDir    -0.004758    -0.012694   -0.02765
Sunshine       -0.004758    -0.012694   -0.02765
WindGustSpeed  -0.003880    -0.010351   -0.02255
WindDir3pm      1.000000    -0.007309   -0.01592
WindSpeed9am   -0.007309     1.000000    0.45903
WindDir9am     -0.015923     0.459032    1.00000

Count of missing values:
   WindDir9am  WindSpeed9am   WindGustDir WindGustSpeed
           31             7             3             2
     Sunshine    WindDir3pm
            3             1
```

Correlation plot and summary generated.

Figure 5.18: The Rattle window displays the underlying data used for the missing observations correlation plot.

a little more understanding of working directly with R. Rattle's Log tab will again provide the steps, which include generating the correlations for the numeric variables using `cor()`:

```
> numerics <- c(3:7, 9, 12:21)
> cc <- cor(weather[numerics],
            use="pairwise",
            method="pearson")
```

We then generate a hierarchical clustering of the correlations. This can be done using `hclust()` (cluster analysis is detailed in Chapter 9):

```
> hc <- hclust(dist(cc), method="average")
```

A dendrogram, the graph structure that we see in the plot for Figure 5.19, can then be constructed using `as.dendrogram()`:

```
> dn <- as.dendrogram(hc)
```

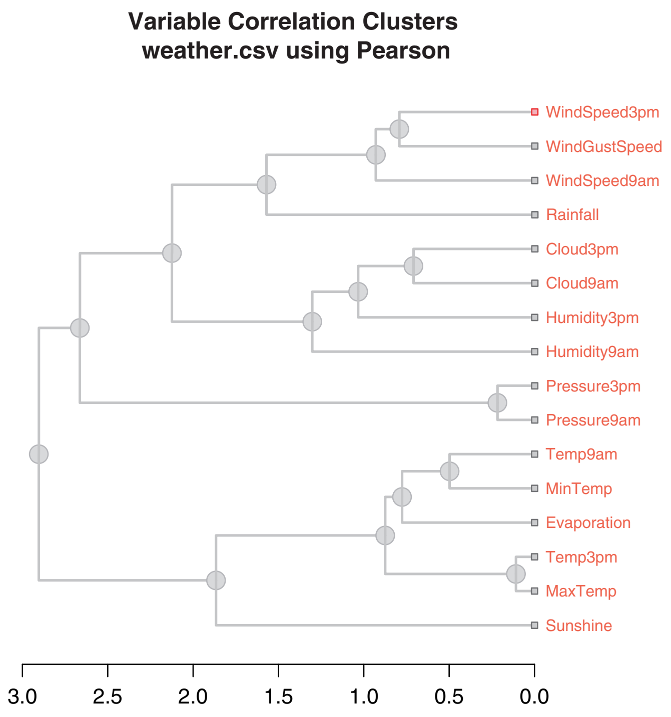


Figure 5.19: The Hierarchical option displays the correlation between variables using a dendrogram.

The actual plot is drawn using `plot()`:

```
> plot(dn, horiz = TRUE)
```

5.4 Command Summary

This chapter has referenced the following R packages, commands, functions, and datasets:

audit	dataset	Used to illustrate Benford's law.
`basicStats()`	command	Detailed statistics of data.
`bpplot()`	command	Box-percentile plot.

`describe()`	command	Detailed data summary.
`cor()`	function	Correlation between variables.
`Ecdf()`	command	Produce cumulative distribution plot.
fBasics	package	More comprehensive basic statistics.
`hclust()`	function	A hierarchical clustering algorithm.
Hmisc	package	Additional basic statistics and plots.
`kurtosis()`	function	A measure of distribution peakiness.
`md.pattern()`	command	Table of patterns of missing values.
mice	package	Missing data analysis.
`pairs()`	command	Matrix of pairwise scatter plots.
`panel.hist()`	command	Draw histograms within a pairs plot.
`panel.cor()`	command	Correlations within a pairs plot.
`panel.smooth()`	command	Add smooth line to pairs plot.
`sample()`	function	Select a random sample of a dataset.
`skewness()`	function	A measure of distribution skew.
`summary()`	command	Basic dataset statistics.
weather	dataset	Sample dataset from **rattle**.