CHAPTER 3

Descriptive statistics for quantitative variables

What you will learn from this chapter:

This chapter shows how to compute basic descriptive statistics for a quantitative variable. You will learn the most popular measures of central tendency (the mean, the median and the mode) and dispersion (variance, standard deviation, range, IQR, median absolute deviation). The chapter will also demonstrate how to produce different graphs (box-and-whisker plots, histograms, density plots, Q–Q plots, line charts), which visualize univariate distributions and help one determine whether a variable is normally distributed. From the case studies you will learn how to analyse the distribution of word lengths in a sample, to detect suspicious values in subjects' reaction times in a lexical decision task, and to correct some problems with the shape of a distribution.

3.1 Analysing the distribution of word lengths: Basic descriptive statistics

3.1.1 The data

To be able to reproduce the code in this section, you will need two add-on packages: `Rling` and `modeest`. See Chapter 2 (Section 2.2) to find out how to install the former. To install the latter, you should do the following:

```
> install.packages("modeest")
```

When you have both packages installed, you should load them:

```
> library(Rling); library(modeest)
```

The first case study analyses the distribution of word lengths (in letters) in a sample of words. The dataset that we will use in this case study is `ldt` from `Rling`. If you want to work with a dataset from an add-on package, it is necessary to load it into the workspace:

```
> data(ldt)
```

The dataset contains 100 randomly selected words from the English Lexicon Project data (Balota et al. 2007). The English Lexicon Project provides behavioural and descriptive data

from hundreds of native speakers for over forty thousand words in American English, as well as for non-words. The dataset contains three quantitative variables:

- *Length*: word length in letters;
- *Freq*: word frequency according to the Hyperspace Analogue to Language frequency norms (Lund & Burgess 1996), based on the HAL corpus, which consists of about 131 million words gathered from Usenet newsgroups, also known as Google groups;
- *Mean_RT*: average reaction times in a lexical decision task, in milliseconds. A lexical decision task is an experiment where subjects are asked to classify stimuli as words or non-words.

In R data frames, individual observations or items are usually represented as rows, and variables as columns. You can access the first six words and their values by using the function `head()`:

```
> head(ldt)
              Length  Freq  Mean_RT
marveled       8      131    819.19
persuaders    10       82    977.63
midmost        7        0    908.22
crutch         6      592    766.30
resuspension  12        2   1125.42
efflorescent  12        9    948.33
```

To learn more about the dataset and its structure, one can use the function `str()`:

```
> str(ldt)
'data.frame': 100 obs. of 3 variables:
$ Length: int 8 10 7 6 12 12 3 11 11 5 …
$ Freq: int 131 82 0 592 2 9 14013 15 48 290 …
$ Mean_RT: num 819 978 908 766 1125 …
```

The function returns the type of R object (data frame), the number of observations (100) and variables (3), as well as the class of each variable and its first values. The abbreviation `int` stands for a numeric vector with integers only (numbers like 1, 2, 3, 4, 100…). Word length in letters and word frequency, as discrete variables, contain only integers. The abbreviation `num` designates a numeric vector with non-integers (numbers like 1.2, 5.84, 9.946 etc.). R may also store integers as `num` objects, but this should not be a matter of concern.

In this case study we will analyse the distribution of word lengths. When analysing a quantitative variable, it is necessary to estimate the central tendency and dispersion. The

measures of central tendency tell us which values tend to be the most typical. The measures of dispersion show how variable the data are.

3.1.2 Measures of central tendency

Which values are the most typical of a distribution? This question can be answered in different ways. Perhaps the most popular measure of central tendency is the **mean**, or **average**. For example, one can speak about average income in a country, average number of children in a family, or average life expectancy. In R, the mean is computed straightforwardly with the help of `mean()`. The average word length in the sample is obtained as follows:

```
> mean(ldt$Length) # equivalent to mean(…, na.rm = FALSE), i.e.
by default, NAs are not removed. If there is a missing value, the
function returns NA. See box 'Handling Missing Data' below.
[1] 8.23
```

So, on average a word in the sample contains 8.23 letters.

How to attach and detach a dataset

If you do not want to type the name of your dataset every time you refer to the variables that it contains, you can simply attach it, so that R will be able to access the objects in the dataset directly:

```
> attach(ldt)
> mean(Length)
[1] 8.23
```

If you do not want R to access the objects directly (for example, if you have another dataset with similar variable names), you should detach the dataset:

```
> detach(ldt)
> mean(Length)
Error in mean(Length): object 'Length' not found
```

Another useful measure of central tendency is the **median**. This notion requires some explanation. Let us first sort all values in the vector of word lengths in increasing order:

```
> sort(ldt$Length) #equivalent to sort(…, decreasing = FALSE)

[1] 3 3 4 4 4 4 4 5 5 5 5 5 5 5 6 6 6 6 6 6 6 6 6
[23] 6 6 6 6 6 7 7 7 7 7 7 7 7 7 7 7 7 8 8 8 8 8
[45] 8 8 8 8 8 8 8 8 8 8 8 9 9 9 9 9 9 9 9 9 9 9
[67] 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 11 11 11 11 11 11
[89] 11 11 11 11 11 12 12 12 13 14 14 15
```

The median is the value that is found in the middle of the ordered values. So, if we have 100 data points, the first half will include the first fifty values, whereas the values ranked 51 to 100 will constitute the second half. The middle point is thus found between the values with the ranks 50 and 51. Both of them equal eight. The median is then the sum of eight and eight divided by two, that is, eight. If we had 99 observations, the median would be the 50th value. If we had 101 observations, the median would be the 51st value. In R, the median can be obtained as follows:

```
> median(ldt$Length) #equivalent to median(…, na.rm = FALSE)
[1] 8
```

In some situations the median gives a better idea of the most typical value than the mean. The problem with the latter is that it is easily influenced by outliers, i.e. scores with unusually high or low values. For example, if twenty employees in a company have net salaries of €2000 a month, and the CEO's salary is €50000, the mean salary will be €4286, and the median will be €2000. The median gives a more realistic idea of the salaries in the company than the mean because the CEO's salary is exceptional.

The median can also be defined in terms of **percentiles** and **quartiles**. They represent values below which a given percentage or fraction of observations in a distribution falls. The median is a value below which we can find 50%. In other words, the median is the 50th percentile. We can also obtain the 33rd percentile, 67th percentile, 99th percentile, and so on. If we cut the data into four quarters, we will obtain quartiles. The point that separates the first quarter, or 25% of observations, from the rest, is called the first quartile, or the 25th percentile. Its counterpart is the third quartile, which separates the first three quarters, or 75% of observations, from the remaining 25%. It can also be called the 75th percentile. The median is then the second quartile, because it separates the two first quarters from the remaining two.

The generic term for percentiles and quartiles is **quantiles.** They are usually expressed as fractions from 0 to 1. For example, the 0.5 quantile corresponds to the median, or 50th percentile, or second quartile. The correspondences between different terms are summarized in Table 3.1. Computation of quantiles in R is straightforward. For example, one can obtain the 0.25 quantile, or the first quartile, or the 25th percentile, as follows:

```
> quantile(ldt$Length, 0.25)
25%
6
```

Another way to obtain the median, or the second quartile, or the 50th percentile, or the 0.5 quantile, is as follows:

```
> quantile(ldt$Length, 0.5)
50%
8
```

Finally, one can compute the third quartile, or the 75th percentile, or the 0.75 quantile:

```
> quantile(ldt$Length, 0.75)
75%
10
```

Quantiles, especially quartiles, will become crucial when one has to interpret graphical representations of distributions, such as box plots (see Section 3.1.4).

Table 3.1 Correspondences between different types of quantiles

Quartiles	Percentiles	Quantiles	Median	Meaning
First quartile	25th percentile	0.25	–	¼ of all ranked scores are below this value
Second quartile	50th percentile	0.5	median	½ of all ranked scores are found below this value
Third quartile	75th percentile	0.75	–	¾ of all ranked scores are found below this value

A third measure of central tendency is the **mode**. It is the most 'popular' value in the data. To find the mode, one can tabulate all possible values with the help of `table()`:

```
> table(ldt$Length)

3 4 5 6 7 8 9 10 11 12 13 14 15
2 5 7 13 12 16 11 16 11 3 1 2 1
```

The function displays all word lengths (the upper row) and how many times each length occurs in the data (the bottom row). One can see that the maximum frequency is 16, which occurs twice: when the length is equal to eight and ten letters. The distribution therefore has two modes, eight and ten. Such distributions are called bimodal.

This method of obtaining the mode, however, is not always very practical. When the number of values is large or when the data are continuous, one can use the function `mlv()` in the package `modeest`. If the data are continuous, the function will find the value that

corresponds to the peak of the probability distribution (see Chapter 1). In this example it returns the same values, eight and ten:

```
> mlv(ldt$Length)
Mode (most frequent value): 8 10
Bickel's modal skewness: -0.17
Call: mlv.integer(x = ldt$Length)
```

More measures of central tendency

The list of central tendency measures is by no means restricted by the mean, the median and the mode. One can also compute the harmonic mean, the trimmed mean, the crude mode, and many more. See Huck (2009: 4–6).

A useful command to get most of the above-mentioned statistics at once is `summary()`, which also includes the minimum and the maximum:

```
> summary(ldt$Length)
   Min. 1st Qu.  Median   Mean 3rd Qu.  Max.
   3.00    6.00    8.00   8.23   10.00   15.00
```

The function for obtaining the maximum value only is `max()`. If you need the minimum value, you can use `min()`:

```
> max(ldt$Length)
[1] 15
> min(ldt$Length)
[1] 3
```

Handling missing data

In real life, we often have to deal with missing data. Consider an integer vector *x*, where the last value is missing ('NA'):

```
> x # do not run the code in this box
[1] 1 2 3 4 5 6 7 8 9 10 NA
```

To test if your data contain missing values, you can do the following:

```
> is.na(x)
[1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
TRUE
```

If there are missing values, this can produce NA's in the output of many standard functions, for example:

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

For some functions, you can fix this by adding `na.rm = TRUE` to make R ignore all missing values:

```
> mean(x, na.rm = TRUE)
[1] 5.5
```

To create a new vector without missing values, you can do the following:

```
> x1 <- x[!is.na(x)]
> x1
[1] 1 2 3 4 5 6 7 8 9 10
```

For data frames, it is also possible to remove all observations with at least one missing value with the help of `data[complete.cases(data),]`.

Sometimes removal of observations with missing data points is undesirable. An alternative is to assign a specific value instead of a missing one:

```
> x2 <- x #creates a new vector identical to x
> x2[is.na(x2)] <- 0
> x2
[1] 1 2 3 4 5 6 7 8 9 10 0
```

This approach will work for a matrix, as well.

Note that some functions remove observations with missing values automatically, so it is recommended to check the help page of the relevant function to be sure that the default option is what you really need.

3.1.3 Measures of dispersion

Although the measures of central tendency provide useful information, they do not show the entire picture. There is an old statistical joke that illustrates this idea, 'If your head is in the freezer, and your feet are in the oven, on average you're quite comfortable'. The mean, the median and the mode cannot say anything about how much variation we have in our data. For this purpose, we need measures of dispersion.

The simplest measure of dispersion is the **range**. It represents the difference between the maximum and minimum values. In our case, the range of word lengths equals 15 – 3 = 12. To obtain the minimum and maximum lengths simultaneously, one can use the following command:

```
> range(ldt$Length)
[1] 3 15
```

Probably the most popular measures of dispersion are variance and standard deviation. Mathematically speaking, **variance** is the sum of squared deviations of the observations from the mean divided by the number of observations minus 1.

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

where n is the number of observations, x_1, x_2, … x_n are individual scores and $\bar{x}$ is the mean. From this follows that the further the individual cases are on average from the mean, the larger the variance. Note that the deviations (i.e. differences between the individual scores and the mean) are squared. There is a simple explanation to that. If an individual score is greater than the mean, the deviation will be positive. If it is smaller than the mean, the deviation will be negative. As a result, all deviations will sum up to zero, which is not particularly informative. The solution is to get rid of the sign by squaring all deviations. This operation also gives more weight to large deviations. To compute variance, one can use the function `var()`:

```
> var(ldt$Length) # equivalent to var(…, na.rm = FALSE)
[1] 6.259697
```

However, it is more conventional to report the **standard deviation**, which is the square root of variance. By taking the square root, we come back to the original units:

```
> sqrt(var(ldt$Length))
[1] 2.501939
```

The standard deviation can also be computed directly with the help of `sd()`:

```
> sd(ldt$Length)
[1] 2.501939
```

Another measure of dispersion is the **interquartile range (IQR)**, which represents the difference between the third (75%) and the first (25%) quartiles. In our sample, the IQR is 10 – 6 = 4. Alternatively, it can be computed as follows:

```
> IQR(ldt$Length)
[1] 4
```

The IQR can be regarded as a more robust measure of dispersion than variance and standard deviation. When individual deviations are squared, the impact of outliers, i.e. observations that strongly deviate from the mean, becomes even stronger.

Yet another measure is the **median absolute deviation** (MAD). To compute it, one needs to put all absolute deviations from the median in ascending order and then select the median value of these deviations. Since this measure is the median of deviations from the median, it is extremely robust. It can be computed in R as follows:

```
> mad(ldt$Length, constant = 1)
[1] 2
```

Standard deviation should be used for normally distributed data; otherwise, report IQR or MAD.

Obtaining statistics for rows or columns in a matrix

We have discussed how one can compute basic descriptive statistics for one numeric vector. But how to obtain mean reaction times, standard deviations, etc., for many columns or rows in a dataset at once? For this purpose, one can use `apply()`. For example, if you want to compute the mean for every row in your data (a matrix or a data frame with only quantitative variables), you can use the following command:

```
> apply(data, 1, mean) # do not run
```

The number `1` tells R to apply the function (`mean`) to every *row*. If you want to compute the standard deviation for each column, you can type in the following:

```
> apply(data, 2, sd) # do not run
```

The number `2` says that you want to apply the function `sd` to every *column* of data. The argument `na.rm = TRUE` should be added if you have missing values in the matrix and want R to ignore them.

Unfortunately, some studies report statistics of the central tendency (e.g. the mean or the median), but fail to report dispersion statistics. This can lead to misinterpretation. Consider two countries with a similar average income per capita. In one country the variance and standard deviation are relatively small because the finances are distributed fairly, whereas in the other they are very large because of several billionaires and many extremely poor people. Although the means are identical, the life in these two countries will differ dramatically.

More measures of dispersion

There exist many other measures of dispersion. For example, Gries (2013: 120–125) discusses such measures as relative entropy, average deviation and variation coefficient.

3.2 Bad times, good times: Visualization of a distribution and detection of outliers

The focus of this case study is reaction times in a lexical decision task. They are available as the variable `Mean_RT` from the data frame `ldt`, which was introduced in the previous case study. The dataset is available in the `Rling` package:

```
> install.packages("ggplot2") # if you have not done so previously
> library(Rling); library(ggplot2)
> data(ldt)
```

Note that we deal with *mean* reaction times averaged across many subjects who performed the task.

It is important to know how a variable is distributed and what the distribution looks like because it will influence the choice of appropriate statistical tests. There are several types of graphs that allow one to visualize a distribution. A popular way of visualizing a univariate continuous distribution is a **histogram**. To create a histogram, the entire range of values is first divided into several intervals. Next, rectangles are drawn for every interval. The height of every rectangle is proportional to the observed frequency of the values in the interval. You can create a histogram of the mean reaction times as follows:

```
> hist(ldt$Mean_RT, main = "Histogram of mean reaction times", xlab
= "reaction times, ms")
```

The result is displayed in Figure 3.1 (left). The first argument is a numeric vector of interest. It is followed by two graphical parameters that specify the title of the plot (`main`) and the label of the horizontal axis (`xlab`). Note that all text labels in R should be enclosed in quotation marks. The quotation marks may be single or double. Make sure you use the same type of quotation marks in the beginning and at the end of your expression. More graphical parameters for standard plots are available on the help page of `par()`.

How many bins do you need?

The representation of a distribution by a histogram depends on the number of bins (rectangles). One can experiment with different number of bins in order to select an optimal representation by using the argument `breaks`, e.g. `hist(v, breaks = seq(min(v), max(v), length = 10))`, where `v` is a numeric vector. This argument specifies the position of the breakpoints between the rectangles, including the outward boundaries (one can also try using `breaks = 10`, but this does not always return the exact number of bins). In this example, you will have ten breakpoints and nine rectangles.

There exist different methods of determining the optimal number of bins. One of them is implemented by default when you run `hist()`. Alternatively, you can use the following formula: number of bins = $1 + 3.32 \times \log_{10} n$, where n is the number of data points (Gries 2013: 113). In our example with mean reaction times, the number of bins can be computed as follows:

```
> nbins <- 1 + 3.32*log10(length(ldt$Mean_RT))
> nbins
[1] 7.64
```

A **density plot** shows the ordered numerical values of a variable x on the horizontal axis, and the probability density of x on the vertical axis (see Chapter 1). The result is smoothed, so that one can see the general shape of the distribution. The plots can be created with the help of the simple `plot()` function and embedded `density()` function. The `xlab` argument allows one to specify the text labels for the x-axis. The result is shown in Figure 3.1 (centre).

```
> plot(density(ldt$Mean_RT), main = "Density plot of mean reaction
times", xlab = "reaction times, ms")
```

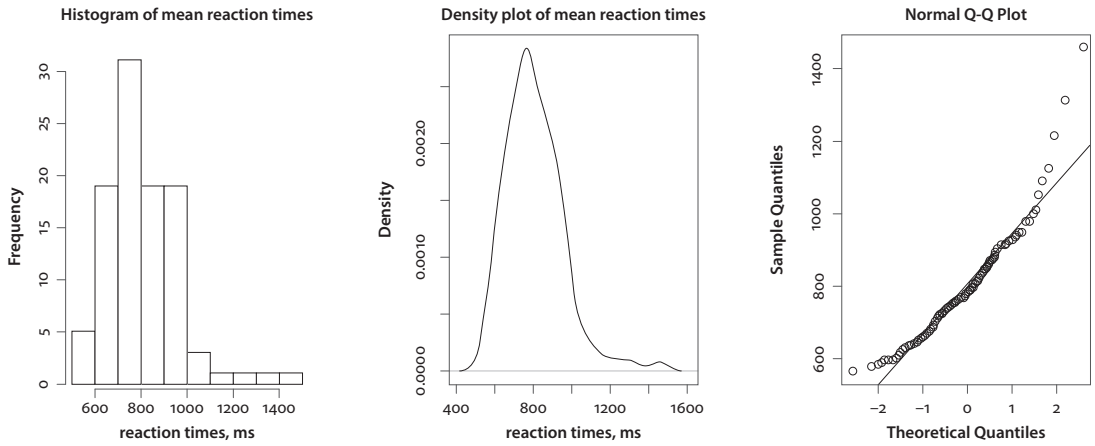


Figure 3.1. Histogram, density plot and Q–Q plot of mean reaction times

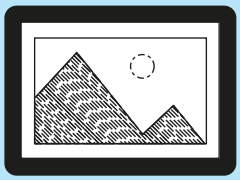
A very useful graph for diagnostics of normality is the **Q–Q (quantile-quantile) plot**. It represents the ordered sample values (the *y*-axis) against the quantiles of the standard normal distribution, which will be introduced below (the *x*-axis). First, one creates a plot, and then adds a straight line to it. The closer the points to the line, the more similar the observed data are to the normal distribution.

```
> qqnorm(ldt$Mean_RT)
> qqline(ldt$Mean_RT)
```

What can one infer from the plots? The histogram and the density plot reveal a long 'tail' on the right. This means that there may be one or several untypically large scores on the higher end of the distribution. Some words took the subjects very long to recognize. These words will be identified below. Such distributions with 'heavy' tails on the right are called **positively skewed**. The observations with untypically large values usually inflate the mean, so that it will become greater than the median.[1] This is what one can observe in the data, where the mean is 808.3, and the median is 784.9:

```
> summary(ldt$Mean_RT)
   Min.   1st Qu.  Median   Mean   3rd Qu.  Max.
   564.2  713.1    784.9    808.3  905.2    1459.0
```

Conversely, a distribution with a heavy left tail is called **negatively skewed**. The mean of such a distribution tends to be smaller than the median.



How to create a density plot, a histogram and a Q–Q plot with the help of ggplot2

From this chapter on, the book will provide code for advanced versions of the most important plots. These versions are created with the help of the package `ggplot2`. The code is very different from the standard plotting functions in R, and consists of several blocks: `ggplot()`, where you should specify the data frame, `geom_...()`, which specifies the kind of plot, `aes()`, which allows one to control 'aesthetics', such as colours, shapes, sizes, etc., and some others. To create a density plot, a histogram and a Q–Q plot of the reaction times with the help of `ggplot2`, you can use the following code:

(*Continued*)

1. But see Huck (2009: 25–28), who discusses exceptions from this rule.

```
> binsize <- diff(range(ldt$Mean_RT))/nbins # computes binwidth
for n bins. Here, we'll use nbins that was computed above.
> ggplot(ldt, aes(x = Mean_RT)) + geom_histogram(binwidth =
binsize, fill = "white", colour = "black") # creates a histogram
> ggplot(ldt, aes(x = Mean_RT)) + geom_line(stat = "density")
#creates a density plot
> ggplot(ldt, aes(sample = Mean_RT)) + stat_qq() # creates a Q-Q
plot
```

Unfortunately, there seems to be no easy way of adding a Q–Q line to the Q–Q plot. Figure 3.1a shows the three plots.

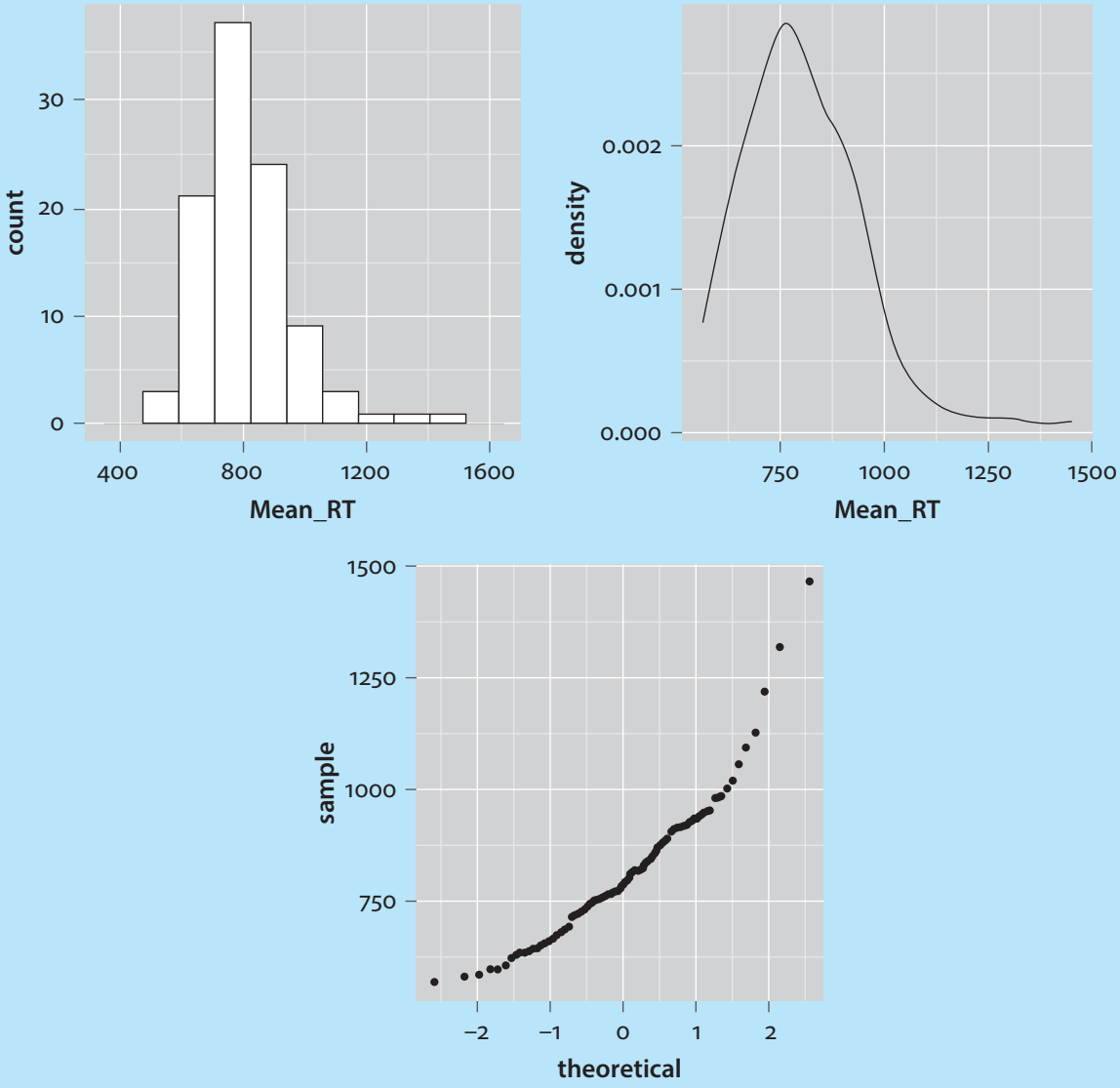


Figure 3.1a. A `qqplot2` version of the diagnostic plots in Figure 3.1

In order to choose an appropriate test, it is often important to know whether the data come from a normal distribution or not. Consider Figure 3.2. It shows a histogram, a density plot and a Q–Q plot of variable *x*. The vector contains 100 values that were randomly

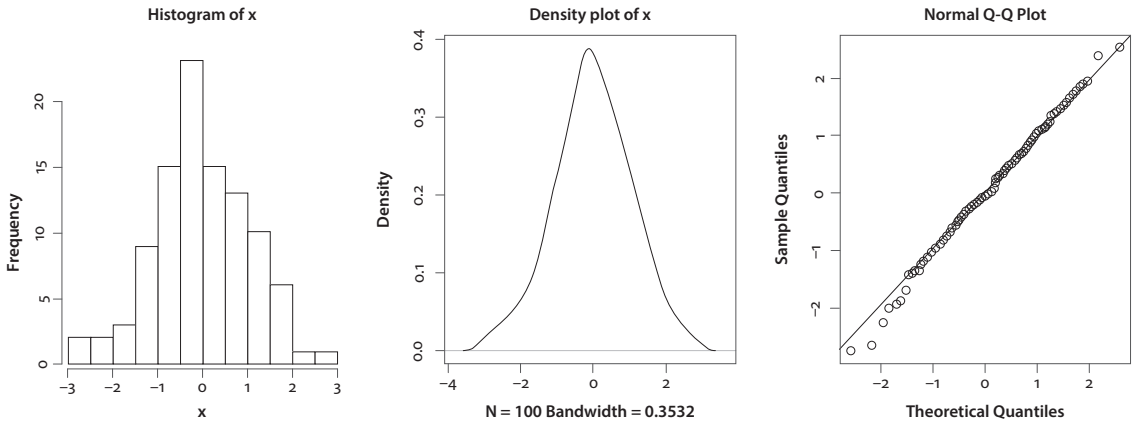


Figure 3.2. A histogram, density plot and Q–Q plot of a normally distributed variable

sampled from the standard normal distribution with the mean of 0 and standard deviation of 1. Normally distributed data are represented by a peculiar bell shape, which is observable on the histogram (left) and the density plot (centre). The closer the value of x to the mean (zero), the higher the probability density (see Chapter 1). Thus, the mean coincides with the mode. A normal distribution is also symmetric. From this follows that the mean also coincides with the median, which divides the distribution into two halves. The Q–Q plot shows that the points are very close to the line. The minor deviations at the ends can be disregarded.

The graphs in Figure 3.1 demonstrate that the mean reaction times are not normally distributed. In some situations, it may be difficult to decide whether the deviations are serious enough. There is a more formal way of testing whether a sample comes from a normal distribution, namely, the Shapiro-Wilk test for normality:

```
> shapiro.test(ldt$Mean_RT)
  Shapiro-Wilk normality test

data: ldt$Mean_RT
W = 0.9201, p-value = 1.418e-05
```

W is the Shapiro-Wilk test statistic. It is used to derive the *p*-value. Recall that if the *p*-value is greater than 0.05, the null hypothesis cannot be discarded. The null hypothesis of the Shapiro-Wilk test is that the sample comes from a normally distributed population. The alternative hypothesis is that the data deviate from normality. The *p*-value is displayed in the scientific notation, which is used to represent very small or very large numbers. The number 1.418e–5 is equal to 1.418 multiplied by 10 at the power of –5. To obtain a 'normal' notation, one can 'move' the decimal delimiter to the left by five decimal places: 0.00001418. Since the *p*-value is below 0.05, we can discard the null hypothesis of normality. In other words, we have enough reasons to believe that the distribution is non-normal, as one can see from the graphs. Note, however, that a visual inspection, especially examining a Q–Q plot, should be preferred to the Shapiro-Wilk test, since the latter is influenced by the sample size. If the number of observations is high, it is 'easier' for the algorithm to find deviations from normality than when the sample is small.

How to disable scientific notation

Scientific notation, e.g. 2.083e–10, can be disabled in R with the help of the following command:

```
> options(scipen = 999)
```

This command tells R to prefer fixed (normal) notation, unless it is 999 digits longer than the expression in scientific notation.

Thus, the mean reaction times are not distributed normally. There seems to be extremely long reaction times for some words. Observations with unusually high or low scores are called **outliers**. The presence of outliers may signal problems with data collection or design. There are different ways of identifying outliers. This section will discuss three criteria: relationship to the IQR (available in a box plot), *z*-scores and MAD-scores.

Let us first create a **box plot** (or, more precisely, a box-and-whisker plot) with the help of `boxplot()`. The first argument of the function is a vector, whereas the second argument is the graphical parameter which specifies the title of the plot.

```
> boxplot(ldt$Mean_RT, main = "Mean reaction times", ylab = "reaction
time in ms")
```

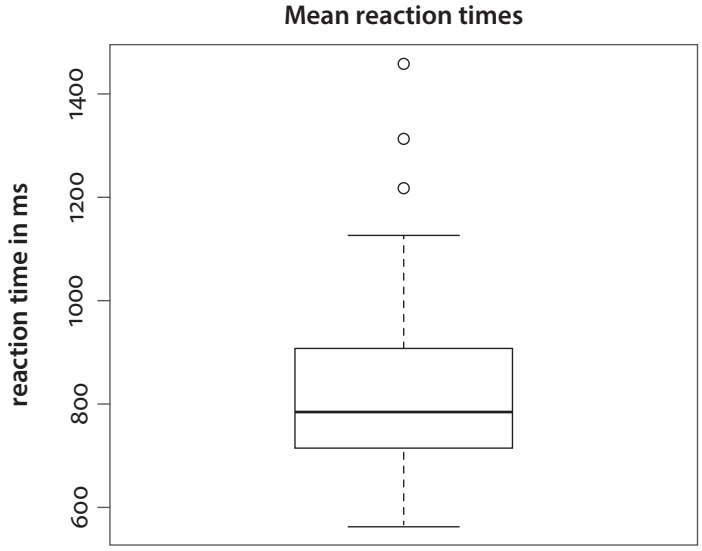
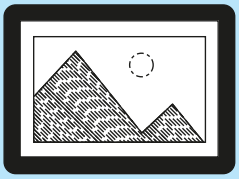


Figure 3.3. Box plot of mean reaction times in a lexical decision task



How to create a box plot with `ggplot2`

One can create a `ggplot2` version of the box plot in Figure 3.3 as follows (see the result in Figure 3.3a):

(*Continued*)

```
> ggplot(ldt, aes(x = 1, y = Mean_RT)) + geom_boxplot() + theme(axis.
title.x = element_blank()) + scale_x_continuous(breaks = NULL)
```

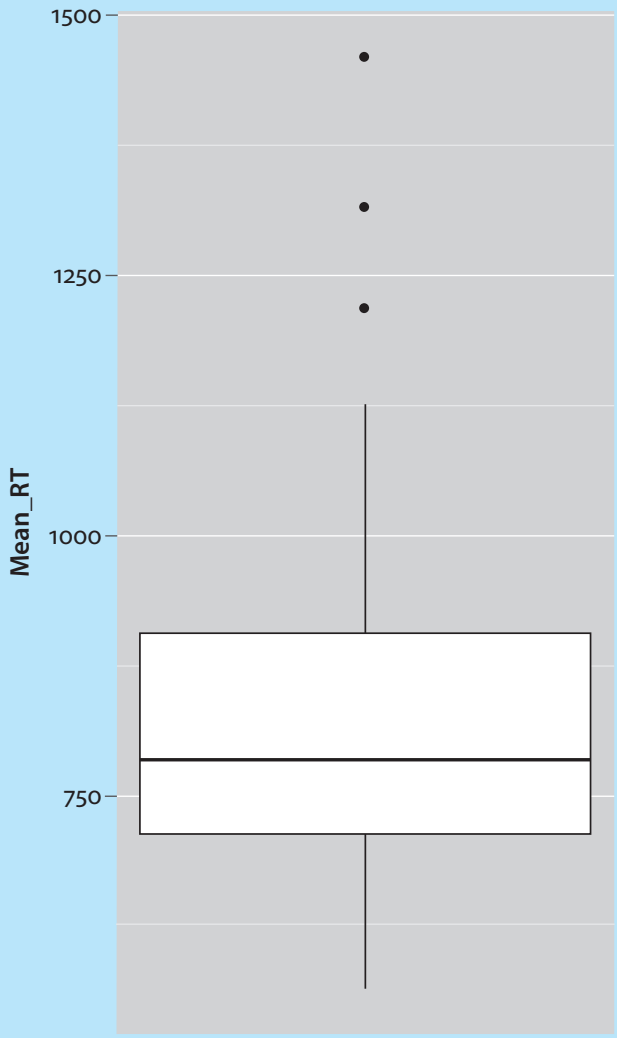


Figure 3.3a. A `ggplot2` version of box plot in Figure 3.3

Note that the *x*-axis has neither a label nor breaks, since these are irrelevant.

A box plot shows the statistics that should be already familiar to the reader from the previous sections. The thick line inside the box corresponds to the median. The lower and upper boundaries of the boxes are called hinges. They show the first and third quartiles, respectively. The size of the box corresponds to the IQR, which represents the difference between the third (upper) and the first (lower) quartiles. The 'whiskers' (the dashed lines) are never longer than 1.5 times the IQR. All observations beyond that value are represented as dots and can be considered outliers. To find out which words these values represent, one can use the following command:

```
> ldt[ldt$Mean_RT > 1200, ]
                 Length  Freq  Mean_RT
dessertspoon     12       11   1314.33
acquisitiveness  15        4   1216.81
diacritical      11      162   1458.75
```

You can also obtain the extreme values by typing in the following:

```
> boxplot.stats(ldt$Mean_RT)$out
[1] 1314.33 1216.81 1458.75
```

These words were on average identified slower. One can consider the unconventional spelling of the first word, length and morphological complexity of the second word, and terminological character of the third word as possible explanations of the longer reaction times.

Another way of detecting outliers, which is probably the most popular one, is by using *z*-scores, also known as standard scores. To compute a *z*-score from an individual score, or, in other words, to standardize the latter, one can subtract the mean from the score, and divide the result by the standard deviation:

$$z = \frac{x_i - \overline{x}}{s}$$

where x_i is an individual score, $\bar{x}$ is the mean, and s is the standard deviation.

To transform the reaction times into *z*-scores, you can use the `scale()` function in the base distribution. However, here we will use the function `normalize()` from `Rling` because it also allows one to compute MAD-scores, i.e. scores based on median absolute deviations. This measure of dispersion was introduced in Section 3.1.3. A further discussion will be provided below. First, however, we will compute the *z*-scores of the mean reaction times. The use of the function is straightforward:

```
> normalize(ldt$Mean_RT)
  [1] 0.07138544 1.10554648 0.65249726 -0.27383532 2.07019341
[output omitted]
```

The scores are now centred around zero. There are different rules of thumb as to which *z*-scores should be treated as suspicious. The most popular absolute values are 2 (not very conservative), 2.5 (moderately conservative) and 3 (very conservative). The higher the cut-off point, the fewer observations have the chance of being detected as outliers. Let us find the points whose absolute *z*-scores are equal to or greater than 2.5. Absolute values are non-negative values regardless of the sign. For example, 3 is the absolute value of both 3 and –3. Absolute values can be obtained with the help of `abs()`.

```
> ldt[abs(normalize(ldt$Mean_RT)) >= 2.5,]
                 Length  Freq  Mean_RT
dessertspoon     12       11   1314.33
```

```
acquisitiveness  15        4   1216.81
diacritical      11      162   1458.75
```

This expression returns the observations whose absolute *z*-scores are greater than or equal to 2.5. The outliers detected with the help of the *z*-scores method are identical to the ones we found with the help of the box plot.

Although *z*-scores are arguably the most popular method, using them to detect outliers sounds somewhat paradoxical, since *z*-scores are based on the mean and the standard deviation, which are sensitive to the presence of outliers. A more robust approach is based on MAD-scores (Leys et al. 2013). One can obtain the MAD-score outliers as follows:

```
> ldt[abs(normalize(ldt$Mean_RT, method = "mad")) >= 2.5,]
                 Length  Freq  Mean_RT
dessertspoon     12       11   1314.33
acquisitiveness  15        4   1216.81
diacritical      11      162   1458.75
```

Note that this method is acceptable only for the data that look normally distributed if you ignore the outliers. If the overall shape of the distribution represented on a diagnostic plot is different from the bell curve, one is advised to examine the outliers outside of the 1.5 × IQR region on the box plot.

After the 'suspects' have been detected, one should decide on what to do with them. First, it is necessary to check whether there are any coding errors and correct them, if possible. If all your data are correct, Field et al. (2012: 190–191) suggest three options: removal of the case(s), data transformation and assigning of new scores. Let us examine them one by one.

1. Remove the case(s). This should be done only in very special cases, for example, if there is a sampling mistake and the observation actually comes from a different population than your target population. For instance, you have got a non-native speaker in an experiment where you want to test only native speakers' performance. To create a new dataset without the outliers, you can take the following steps. First, you need to identify the row IDs of the outliers. For example, if you want to remove all outliers with absolute MAD-scores equal to or greater than 2.5, you can first obtain the row numbers of the suspicious observations with the help of `which()` containing the expression that has been used for identification of the outliers:

```
> outliers <- which(abs(normalize(ldt$Mean_RT, method = "mad")) >=
2.5)
> outliers
[1] 29 70 100
```

Thus, the outliers' row IDs are 29, 70 and 100. If you want to remove the outliers, the next step is as follows:

```
> ldt_remove <- ldt[-outliers,] # or ldt[-c(29, 70, 100),]
> dim(ldt_remove)
[1] 97 3
```

The function `dim()` returns the dimensions of the data: the number of rows and the number of columns. Its output shows that the data frame has now only 97 rows, instead of the original 100.

2. Use a transformation. This method should be applied when your data are very skewed (asymmetric). It is described in the next section.
3. Assign a different score, e.g. the mean plus or minus two standard deviations: `mean(data) + 2*sd(data)` for right-tailed, positively skewed data, or `mean(data) - 2*sd(data)` for left-tailed, negatively skewed data. For example, if you want to assign the mean plus two standard deviations from the mean, you can first compute the score:

```
> mean(ldt$Mean_RT) + 2*sd(ldt$Mean_RT)
[1] 1114.666
```

Next, you can create an identical data frame under a different name and replace the values in the outlier rows and the third column (reaction times):

```
> ldt_new <- ldt
> ldt_new[outliers, 3] <- 1114.666
```

Let us check the result:

```
> ldt_new[outliers,]
                 Length  Freq  Mean_RT
dessertspoon     12       11   1114.666
acquisitiveness  15        4   1114.666
diacritical      11      162   1114.666
```

How to compute *z*-scores and MAD-scores for rows or columns in a matrix?

The function `normalize()` from `Rling` can also compute *z*-scores and MAD-scores for all rows or columns in a matrix. For example, if you want to compute the MAD-scores for each column, you can use the following command:

(*Continued*)

```
> normalize(data, method = "mad", by = "col")
```

To compute the *z*-scores for each row, you can use the following:

```
> normalize(data, method = "z-scores", by = "row")
```

or simply

```
> normalize(data)
```

The *z*-score method and standardization by row are the default options.

3.3 Zipf's law and word frequency: Transformation of quantitative variables

To reproduce the code in this case study, you will need two packages: `Rling` and `ggplot2`. If you have not installed the latter yet, you can do it as follows:

```
> install.packages("ggplot2")
```

Next, you will need to load the packages:

```
> library(Rling); library(ggplot2)
```

The dataset for this case study is again `ldt` from `Rling`. You should load it into you current workspace if you have not done so already.

```
> data(ldt)
```

The third variable in the dataset is `Freq`, which we have not examined yet, shows the word frequency in a large corpus. Let us first obtain the basic statistics:

```
> summary(ldt$Freq)
   Min. 1st Qu. Median  Mean   3rd Qu. Max.
   0.0  53.5    310.5   3350.0 2103.0  75080.0
```

As one can see, the frequencies have a very broad range, from zero to 750,80. Let us explore the distribution visually by plotting the values in descending order, from high-frequency words to low-frequency ones (the function `sort()` with the option `decreasing = TRUE`):

```
> plot(sort(ldt$Freq, decreasing = TRUE), type = "b", main = "Zipf's
law", ylab = "Word frequency")
```

The result is shown in Figure 3.4. The position of data points is determined by their values on the *x*- and *y*-axes, which represent two numeric vectors. If there is only one vector, as in our case, then the scores are represented by the vertical axis, and the horizontal axis shows

the scores' indices (in our case, ranks). This graph is called a line chart because it displays data points connected by lines. By default, the function `plot()` usually creates a scatter plot with points only. If one adds `type = "b"`, the points will be connected by lines. Adding `type = "l"` will display only lines.

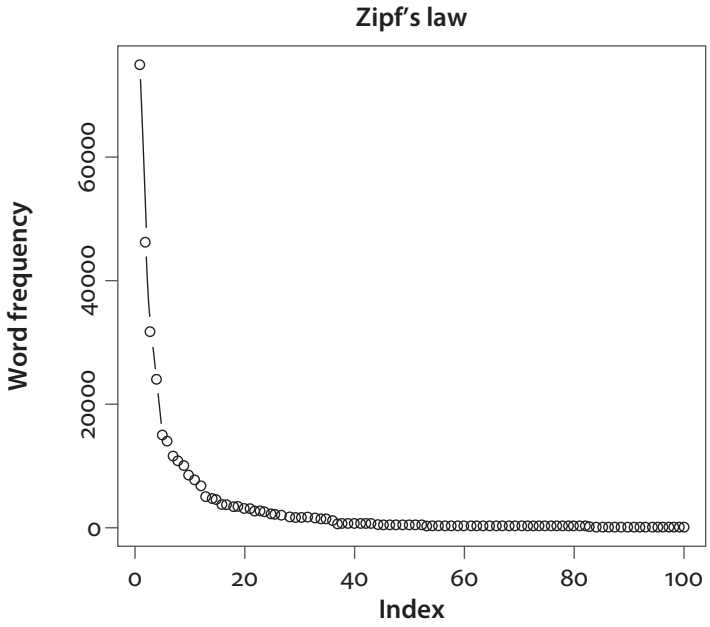
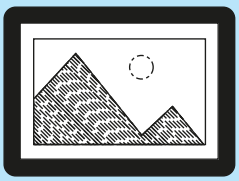


Figure 3.4. Line chart with word frequencies, displayed in descending order



Line chart with ggplot2

To create a `ggplot2` version of Figure 3.4, you can use the following code:

```
> ggplot(ldt, aes(x = 1:nrow(ldt), y = sort(Freq, decreasing
= TRUE))) + geom_line() + geom_point() + xlab("Index") +
ylab("Frequency")
```

(*Continued*)

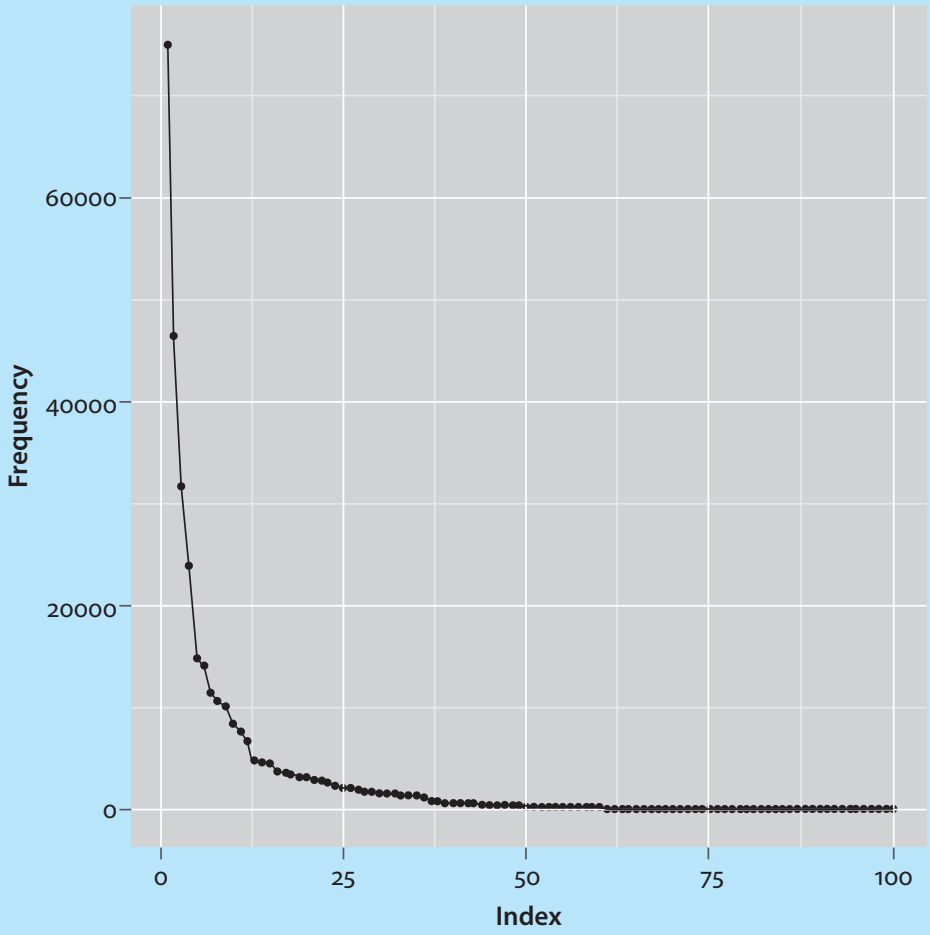


Figure 3.4a. A `ggplot2` version of the line chart in Figure 3.4

The title of the plot is 'Zipf's law' because it illustrates an empirical law of distribution of word frequencies in a text or corpus, which was first formulated by Zipf (1935). According to this law, the corpus frequency of any word is inversely proportional to its rank in the frequency list. In other words, the first word on the frequency list is twice as frequent as the second most frequent word, three times as frequent as the third most frequent word, etc. As a result, a corpus of natural language normally contains very few words with very high frequencies, which account for the larger part of the corpus, and very many words with low frequencies. This is why the majority of words that one can find in a corpus occur there only once. They are called *hapax legomena* (*hapax legomenon* in the singular).

Most frequency data are distributed in a similar fashion. The histogram, density plot and Q–Q plot in Figure 3.5 demonstrate that this distribution has a very peculiar shape. The plots were created with the help of the following code:

```
> par(mfrow = c(1, 3)) # allows to create several plots in one device: the first number (1) stands for the number of rows, and the second (3) for the number of columns
> hist(ldt$Freq, main = "Histogram of word frequencies", xlab = "Word frequency in a corpus", ylab = "Relative frequency in the sample")
```

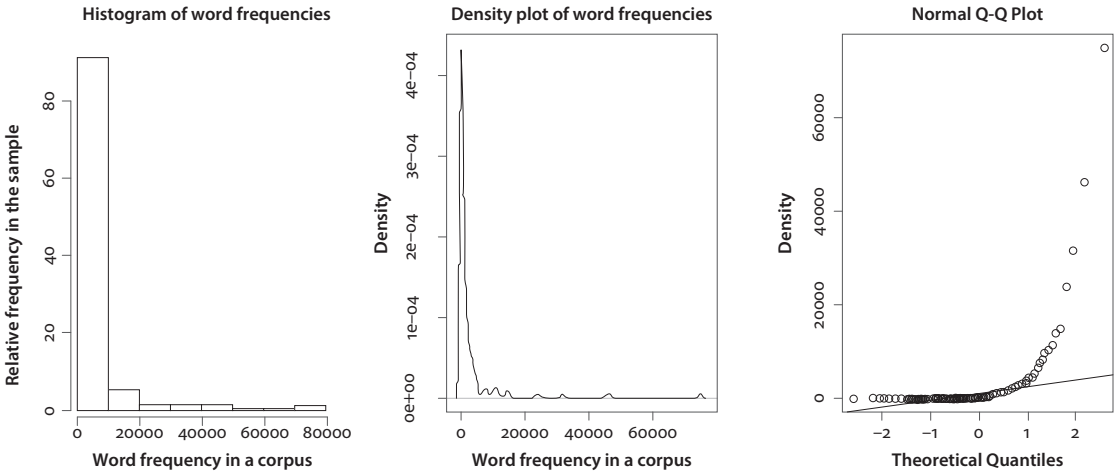


Figure 3.5. A histogram, density plot and Q–Q plot of word frequencies

```
> plot(density(ldt$Freq), main = "Density plot of word frequencies",
xlab = "Word frequency in a corpus")
> qqnorm(ldt$Freq)
> qqline(ldt$Freq)
```

Frequency data are usually logarithmically transformed. However, some frequencies are equal to zero. Since the natural logarithm of zero `log(0)` is minus infinity, this causes a problem:

```
> log(0)
[1] -Inf
```

To solve this problem, one can add 1 to each frequency score and compute the logarithms of the new scores. In R, this transformation can be done by using the function `log1p()`. Let us visualize the transformed data:

```
> hist(log1p(ldt$Freq), main = "Histogram of log-frequencies", xlab
= "Log-transformed word frequency", ylab = "Relative frequency in
the sample")
> plot(density(log1p(ldt$Freq)), main = "Density plot of log-
frequencies", xlab = "Log-transformed word frequency in a corpus")
> qqnorm(log1p(ldt$Freq))
> qqline(log1p(ldt$Freq))
```

As shown in Figure 3.6, the distribution of the log-transformed frequencies is now very similar to normal. The same conclusion can be made after using the Shapiro-Wilk test of normality:

```
> shapiro.test(log1p(ldt$Freq))

  Shapiro-Wilk normality test

data: log1p(ldt$Freq)
W = 0.9814, p-value = 0.1699
```

Other popular power transformations include the square root transformation `sqrt(yourVector)`, which can be helpful for moderate positive skews, the square transformation `(yourVector)^2` for negatively skewed data, and the reciprocal transformation, e.g. `1/(yourVector + 1)`, which can help in cases of distributions that look like the letter J or its mirror image. More details can be found in Field et al. (2012: 191–201) and Sheskin (2011: 483–488). The best way is to try different transformations and visualize them with the help of the plots that we have just discussed.

Applying power transformations has its costs and benefits. On the one hand, you can use many traditional well-understood parametric methods with normally distributed data. These tests will be introduced in the following chapters. On the other hand, overly sophisticated transformations make the results of statistical tests less interpretable. Moreover, there has been increasing interest in non-parametric methods, especially the ones

Figure 3.6. A histogram, density plot and Q–Q plot of log-transformed word frequencies

based on resampling, such as bootstrap and permutation. These approaches, which will be introduced in the following chapters, allow one to test statistical hypotheses on non-normal data.

3.4 Summary

This chapter has shown many basic operations that can help you explore univariate quantitative data. Data exploration is the first step in any statistical analysis. Its importance cannot be overestimated. This chapter has introduced the basic descriptive statistics (measures of central tendency and dispersion) and various graphical tools that enable one to investigate how the data are distributed. Several popular approaches to detecting outliers have been discussed, as well as some transformations that can be used in order to make the data more normal. The next chapter will demonstrate how basic diagnostics can be performed for qualitative, or categorical variables.