CHAPTER 5

Comparing two groups

t-test and Wilcoxon and Mann-Whitney tests for independent and dependent samples

What you will learn from this chapter:

Do language learners who are taught by an innovative method show better results than those who are taught traditionally? Do speakers of one language variety speak faster than speakers of another variety? Do people of one gender use more hedging constructions than people of another? In this chapter, you will learn how to make such comparisons using the parametric *t*-test and the non-parametric Wilcoxon and Mann-Whitney tests for dependent and independent samples. You will learn how to compute the standard error and confidence intervals for the mean. The case studies will involve differences between high- and low-frequency nouns with regard to the number of associations that they trigger and their abstractness/concreteness scores.

5.1 Comparing group means or medians: An overview of the tests

This chapter introduces the *t*-test and its non-parametric version, the Wilcoxon test, which is equivalent to the Matt-Whitney test, also known as the *U*-test. These tests compare the measures of central tendency in two groups. More specifically, the *t*-test can tell us whether the group means are different, and the non-parametric tests are usually regarded as tests of the differences between the group medians.

All these tests exist in two versions: dependent (paired) and independent (unpaired). A **dependent**, or **paired** test is performed when the observations in two groups are paired. That is, an observation from one group has a related observation in the second group. Such observations may come from the same subject or be related to the same experimental stimulus. For example, one can measure learners' proficiency in a language before and after they are taught by a new teaching method, and investigate if the method has had any effect on their performance. In this case, every learner will have two scores, before and after the experiment. If there is no such connection between the scores in two groups, the test is called **independent**, or **unpaired**. An example is a test of the differences in vocabulary size of two groups of learners, where every learner is tested only once and has only one score.

To choose between the *t*-test and the non-parametric options, one should check a number of assumptions, i.e. the criteria that the data should meet so that the test returns meaningful results. The assumptions of the dependent and independent tests are slightly different. The parametric *t*-test for **independent** samples has the following assumptions:

- *The samples have been randomly selected from the populations they represent.*
- *The observations are independent, both between the groups and within the groups.*
- *The variable should be at least interval-scaled.*
- *The data in both samples are normally distributed, and/or the sample sizes are greater than 30.*[1]
- *The variances of the samples should be homogeneous.* That is, the variances in both groups should be equal. However, the standard implementation of the *t*-test in R includes Welch's adjustment, which provides a correction for unequal variances (see Field et al. 2012: 373 for more detail). This is why we will not be concerned about this assumption in this chapter. In Chapter 8, we will discuss the Levene test and the Fligner-Killeen test, which can be used for comparing group variances.

The test assumptions of the *t*-test for **dependent** samples are as follows:

- *The subjects have been sampled randomly.*
- *The data are at least interval-scaled.*
- *The differences between the pairs of scores* (*not the scores themselves!*) *are normally distributed, and/or the sample size is greater than 30.*
- *The variances in the underlying populations that represent two groups or conditions are equal.* As in the previous case, we will rely on the built-in correction for heterogeneous variance, which is implemented in the default version of the *t*-test in R.

If these assumptions are not met, one should use the non-parametric Wilcoxon or Matt-Whitney tests. They have less strict assumptions:

- *Each sample has been drawn randomly from the population.*
- *The observations are independent, within each sample only* (the dependent test) *or both within each sample and between the samples* (the independent test).
- *The measurement scale is at least ordinal.*
- *The underlying distributions need not be normal, but they should be of a similar shape.*

---

1. Behind these two conditions is, in fact, one assumption, which requires that the sampling distribution should be normal. However, since we do not have access to this distribution we have to rely on the data at hand. See a definition of the sampling distribution in Section 5.2.4 and a detailed discussion in Urdan (2010: Ch. 6).

Finally, all above-mentioned tests exist in two versions: **one-tailed** and **two-tailed**. These notions were discussed in Chapter 1. Recall that a one-tailed test should be used when the alternative hypothesis is directional. For example, it contains expressions 'X is more than Y' or 'the greater X, the greater Y'. In contrast, if the hypothesis is non-directional, e.g. 'X does not equal Y', it is correct to use a two-tailed test. It is crucial that the choice between a one-tailed and two-tailed test should be made before the test statistic and the *p*-value are computed.

The remaining part of the chapter is organized as follows. Section 5.2 discusses the independent one-tailed *t*-test. Section 5.3 explains the non-parametric independent two-tailed Wilcoxon test. Section 5.4 demonstrates how one can perform the paired two-tailed *t*-test for dependent samples. Finally, Section 5.5 provides a short summary and suggestions on writing up the results of these tests.

5.2 Comparing the number of associations triggered by high- and low-frequency nouns with the help of the independent *t*-test

5.2.1 Data and hypothesis

To access the data and functions used in this case study you will need to have the following add-on packages installed:

```
> install.packages(c("ggplot2", "gplots"))
> library(Rling); library(ggplot2); library(gplots)
```

You will need two data frames, which are available in the `Rling` package under the names `pym_low` and `pym_high`. The data come from a well-known experimental study of 925 English nouns by Paivio et al. (1968). The subjects were asked to rate the nouns on concreteness, imagery and so-called meaningfulness. Concreteness is the most transparent parameter: concrete words are those that denote tangible objects, materials or persons that can be easily perceived with the senses. Imagery scores reflect how quickly and easily the word arouses a mental image defined as sensory experience, such as a mental picture or sound. The subjects were asked to rate concreteness and imagery on a scale from 1 (minimum) to 7 (maximum). Finally, meaningfulness represents the number of associations provided by the speakers in 30 seconds after they were shown the words. Paivio, Yuille and Madigan also used information about word length in syllables and letters.

```
> data(pym_high)
> data(pym_low)
```

The sample in `pym_low` contains 51 nouns sampled from the words with the frequency from 1 to 20, whereas `pym_high` contains 50 nouns with the frequency greater than 50.

These two samples were randomly produced by an online word list generator (Friendly 1996). The frequencies were taken from the Brown corpus (Kučera & Francis 1967), which was contemporary to the experiment, although it looks very small by the modern standards. The word list generator also returns the values for all variables of interest.

The data frames have similar structure. The observations are individual words, which are represented by the row names. Each dataset contains five variables. The first variable *syl* specifies the number of syllables for each word. The second one *let* is the number of letters. *Imag* is the average imagery score, which ranges from 1 (the lowest score) to 7 (the highest score). Likewise, *conc*, which is the average concreteness score, ranges from 1 to 7. The final variable *assoc* represents the average number of associations provided by the speakers for every word.

The first six rows of `pym_high` look as follows:

```
> head(pym_high)
        syl  let  imag  conc  assoc
time    1    4    4.13  2.47  7.00
life    1    4    4.07  2.96  6.78
home    1    4    6.50  6.25  6.88
church  1    6    6.63  6.59  7.52
mind    1    4    3.03  2.60  5.88
door    1    4    6.60  7.00  7.96
```

You can see the structure of the data frame with the help of `str()`:

```
> str(pym_high)
'data.frame': 50 obs. of 5 variables:
$ syl: int 1 1 1 1 1 1 2 1 5 2 …
$ let: int 4 4 4 6 4 4 7 4 10 8 …
$ imag: num 4.13 4.07 6.5 6.63 3.03 6.6 6.2 6.87 6.53 4.1 …
$ conc: num 2.47 2.96 6.25 6.59 2.6 7 6.38 6.83 5.87 3.63 …
$ assoc: num 7 6.78 6.88 7.52 5.88 7.96 7.28 5.12 7.2 5.8 …
```

One can see that the first two variables, *syl* and *let*, are integer vectors, since they contain only discrete values, and the other variables are non-integer numeric vectors, which contain mean scores averaged over many different subjects.

The main question of this and the following case study is whether there is a difference between the high- and low-frequency nouns with regard to the experimental scores. There is substantial evidence of frequency effects in language use, acquisition and change (see Diessel 2007 for an overview). The evidence suggests that language learners are sensitive to frequencies of words and expressions that they encounter.

An important contribution to study of frequency effects was made by Zipf (1935), who discovered several frequency-related regularities. One of them, known as Zipf's law, was discussed in Chapter 3. In addition, Zipf found that the length of a word is in inverse

relationship to its relative frequency (he called it the Law of Abbreviation).[2] Frequency also affects semantic functions of a word or construction. Frequent expressions have chances of being used in very diverse contexts, and therefore can acquire new grammatical and pragmatic functions. As a result, the higher the frequency of a word, the more semantically versatile it is. This correlation was pinpointed by Zipf (1949) in his Principle of Economic Versatility. From this follows that one can expect the high-frequency words to trigger on average more associations than the low-frequency ones, since the former are used in a greater number of diverse contexts. This will be the alternative hypothesis of the present study. The null hypothesis is that there is no difference between the frequency groups with regard to the number of associations. To see whether our expectations are supported by the data, we will first perform some exploratory analyses and visualizations, and next, will turn to the discussion of possible inferential statistics.

5.2.2 Descriptive statistics and visualizations

It is always recommended to begin statistical analyses by looking at the data distribution with the help of the methods introduced in Chapters 3 and 4. First, let us get the most important descriptive statistics by using the `summary()` function:

```
> summary(pym_high$assoc)
  Min.  1st Qu.  Median  Mean  3rd Qu.  Max.
  4.88  5.69     6.24    6.38  7.16     9.12
> summary(pym_low$assoc)
  Min.   1st Qu.  Median  Mean   3rd Qu.  Max.
  3.000  5.345   5.920   5.857  6.460    8.000
```

All statistics shown in the summary are higher in the high-frequency sample than in the low-frequency sample.

A useful visual tool for comparison of two and more groups is the box-and-whisker plot. Chapter 3 showed how to create a box plot for one sample. One can also put several boxes for two or more samples side by side in one graph and compare them (see Figure 5.1):

---

2. There exist two main explanations of the reduction effect, and they are not mutually exclusive. On the one hand, frequent use of a word results in automation of production processes, which has influence on articulation (Bybee 2001). On the other hand, according to Diessel (2007), the more frequent a word, the more distributional information about the word is available, and therefore the more predictable it is from the context. As a result, even when a word is strongly reduced, the hearer can still restore it from the context. So the speaker can afford to be 'lazy' without the risk of being misunderstood.

```
> boxplot(pym_high$assoc, pym_low$assoc, names = c("high", "low"),
main = "Box plots of average numbers of associations", xlab =
"Frequency group", ylab = "Average number of associations")
```

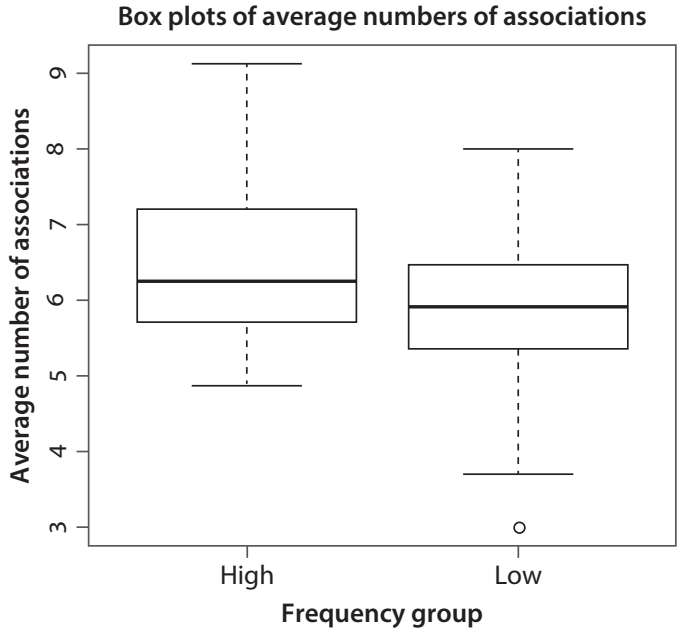
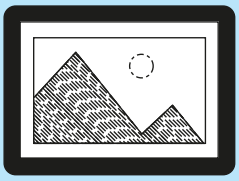


Figure 5.1. Box plots of average numbers of associations for the high- and low-frequency nouns



How to create a box plot of two or more groups with the help of ggplot2

To make a box plot of two vectors, one first has to combine the vectors with association scores for the high- and low-frequency nouns in one data frame, also creating a column that specifies the frequency group ('high' or 'low'). The function `rep()` repeats the same character strings.

```
> pym_assoc <- data.frame(assoc = c(pym_high$assoc, pym_
low$assoc), freq = c(rep("high", 50), rep("low", 51)))
> head(pym_assoc)
   assoc  freq
1  7.00   high
2  6.78   high
3  6.88   high
4  7.52   high
5  5.88   high
```

```
6  7.96    high
```

Now you can create the box plot (see Figure 5.1a):

```
> ggplot(pym_assoc, aes(x = freq, y = assoc)) + geom_boxplot() +
xlab("Frequency group") + ylab("Average number of associations")
```

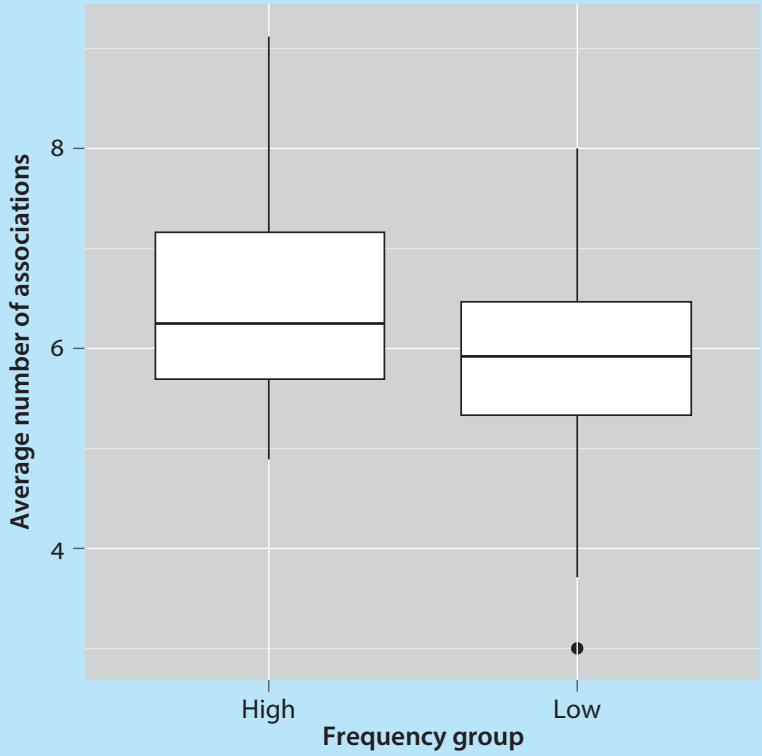


Figure 5.1a. A `ggplot2` version of the box plot in Figure 5.1

One can see from the plot that the scores of the low-frequency sample are distributed slightly more symmetrically around the median than those of the high-frequency sample. However, the low-frequency sample has an outlier with a very low score. It is represented by a dot. One can identify this observation by first retrieving its score, and then selecting the row with this score from the data frame:

```
> boxplot.stats(pym_low$assoc)$out
[1] 3
> pym_low[pym_low$assoc == 3, ]
           syl  let   imag   conc   assoc
criterion   4    9    1.83   1.93   3
```

The outlier is the Greek loan word *criterion*. This noun also has the lowest imagery score (1.83) and the third lowest concreteness score (1.93) (see the box *How to sort data frames* to find information about detecting the highest- and lowest-ranking scores in a data frame).

How to sort data frames

In order to sort data frames, you can use the function `order()` within the subsetting square brackets. For example, one can sort the observations (rows) by the imagery scores in ascending order as follows:

```
> pym_low[order(pym_low$imag),]
             syl  let  imag  conc  assoc
criterion    4    9    1.83  1.93  3.00
impropriety  5    11   1.87  2.08  3.75
allegory     4    8    2.13  2.56  4.48
hypothesis   4    10   2.40  2.25  5.36
gender       2    6    2.90  3.63  5.41
[output omitted]
```

To sort in descending order, one adds the minus sign before the name of the column:

```
> pym_low[order(-pym_low$imag),]
         syl  let  imag  conc  assoc
lip      1    3    6.57  6.93  5.32
priest   1    6    6.53  6.59  6.88
tower    2    5    6.53  6.96  6.42
potato   3    6    6.50  7.00  7.13
nail     1    4    6.50  6.96  6.08
meadow   2    6    6.43  6.69  8.00
[output omitted]
```

To sort by more than one column, for example, by the number of syllables and then by the number of letters, simply list all these columns in the order that you prefer:

```
> pym_low[order(pym_low$syl, pym_low$let),]
      syl  let  imag  conc  assoc
lip   1    3    6.57  6.93  5.32
rod   1    3    5.97  6.62  6.04
fur   1    3    6.23  6.69  7.36
deed  1    4    3.63  4.19  5.32
lump  1    4    5.63  6.20  5.44
[output omitted]
```

5.2.3 Choosing an appropriate test to compare the measures of central tendency in two groups

The descriptive statistics and the paired box plot support our hypothesis that high-frequency nouns trigger more associations than low-frequency nouns. However, it may well be possible that this difference is due to chance only. If this is so, the results cannot be extrapolated to the entire population. In other words, if other researchers replicated the analyses on new samples of low- and high-frequency nouns, the results could be different. This is why one needs inferential statistics. To decide on the type of test, one has to answer three questions, which were already mentioned in Section 5.1:

1) should one use the test for dependent (paired) or independent (unpaired) data?
2) is the *t*-test or its non-parametric version more appropriate?
3) should the test be one- or two-tailed?

Since there is no connection between the scores in the high- and low-frequency samples, there are no reasons to assume that the observations are dependent. Thus, an independent (unpaired) test should be used. The next question is whether the independent *t*-test or its non-parametric version is more appropriate. Let us check if the assumptions of the *t*-test are met. For convenience, the assumptions are repeated below.

- *The samples have been randomly selected from the populations they represent.* This assumption is met.
- *The observations are independent, both between the groups and within the groups.* Since we do not have good reasons to believe that the number of associations triggered by some words depends on the number of associations triggered by others, we can consider this assumption to be met.
- *The quantitative variable should be at least interval-scaled*. Since the data are on the ratio scale, this assumption is met, as well.
- *The data in both samples are normally distributed, and/or the sample sizes are greater than 30.* To find out whether the samples are normally distributed or not, one can use visualization tools (histograms, density plots and Q–Q plots) and the Shapiro-Wilk test of normality. All these diagnostic tools were introduced in Chapter 3. However, the number of observations in the samples is 50 and 51, so we do not need to worry about normality.

Finally, would it be correct to use a one- or two-tailed test? Recall that the alternative hypothesis is directional. The high-frequency nouns are expected to trigger more associations than the low-frequency ones, so we should prefer the one-tailed version.

To summarize, the data and alternative hypothesis suggest that we should use the independent one-tailed *t*-test. The test, as well as its dependent and two-tailed versions, is

implemented in the `t.test()` function. By default, the test is independent (`paired = FALSE`), but one can change that by typing in `paired = TRUE` if the samples are paired. The default test is also two-tailed (`alternative = "two.sided"`). If the alternative hypothesis is directional, one has two options. One can use `alternative = "greater"` if one expects that the first sample has a greater mean than the second sample, or `alternative = "less"` if one expects the first sample to have a smaller mean than the second sample. In our case, the code is as follows:

```
> t.test(pym_high$assoc, pym_low$assoc, alternative = "greater")

     Welch Two Sample t-test

data: pym_high$assoc and pym_low$assoc
t = 2.6717, df = 98.281, p-value = 0.004417
alternative hypothesis: true difference in means is greater than 0
95 percent confidence interval:
0.1977777     Inf
sample estimates:
mean of x mean of y
6.380000 5.857451
```

The most important line is the one that contains the *t*-statistic (2.6717), the degrees of freedom (98.281) and the *p*-value (0.004). Since the *p*-value is smaller than the significance level, we can reject the null hypothesis of no difference between the means. High-frequency words indeed trigger more associations than low-frequency words, and this difference is statistically significant. The algorithm also returns the 95% confidence interval of the difference in means. If a confidence interval includes 0, then it is possible that the difference between the means may be zero and therefore our result may be due to chance. In our example, the confidence interval ranges from approximately 0.198 to infinity. This is another indication that the difference in the number of associations triggered by high- and low-frequency words is significant. More information about the notion of confidence intervals will follow in the next subsection.

A column with groupings and formula interface

It is possible that your original data have a different format. Instead of two vectors, you may have two columns: one with numerical values and the other with information

about the group membership. An example is `pym_assoc`, the data frame, which was created in the `ggplot2` box in the previous subsection:

```
> head(pym_assoc)
     assoc  freq
1     7.00   high
2     6.78   high
3     6.88   high
4     7.52   high
5     5.88   high
6     7.96   high

> tail(pym_assoc)
     assoc  freq
96    5.40   low
97    6.72   low
98    6.00   low
99    4.92   low
100   3.75   low
101   6.00   low
```

The column *assoc* shows the average number of associations, and the column *freq* displays which of the two frequency groups every nouns belongs to. In this case, you can use the *t*-test (and many other tests) with the formula interface:

```
> t.test(pym_assoc$assoc ~ pym_assoc$freq, alternative =
"greater")
[output omitted]
```

Note that the first (reference) level of the factor `freq` ('high') is compared with the second level of the factor ('low'):

```
> levels(pym_assoc$freq)
[1] "high" "low"
```

By default, the reference level is the one that comes first alphabetically. This is why one should use `alternative = "greater"` if one expects the high-frequency nouns to trigger more associations that the low-frequency nouns. If the reference level were 'low', one should use `alternative = "less"`.

Note that it is perfectly possible to swap the first two arguments and use `alternative = "less"`. The results will be identical, except for the reverse order of the means and the negative sign of the *t*-statistic and the confidence interval boundaries:

```
> t.test(pym_low$assoc, pym_high$assoc, alternative = "less")

     Welch Two Sample t-test
```

```
data: pym_low$assoc and pym_high$assoc
t = -2.6717, df = 98.281, p-value = 0.004417
alternative hypothesis: true difference in means is less than 0
95 percent confidence interval:
     -Inf -0.1977777
sample estimates:
mean of x mean of y
5.857451 6.380000
```

To conclude, our analyses reveal that the average number of associations per word is greater for high-frequency nouns than for low-frequency ones. The results of the test give us grounds to believe that the difference is not due to chance.

How to report *p*-values correctly

A common mistake made by beginners is to report *p*-values with excessive precision, e.g. $p = 0.000123456$. It is not recommended to go beyond three decimal places, e.g. $p = 0.047$. If your *p*-value is smaller than 0.001, you should write $p < 0.001$. Check the style sheet of your publisher for more precise guidelines. Also, you should specify whether the test is two-tailed or one-tailed. This is absolutely necessary in case of a one-tailed test.

5.2.4 Confidence intervals and standard errors

As was shown above, the *t*-test returns **the 95% confidence interval** of the difference between the means, and one should check whether the confidence interval includes the zero. But what are confidence intervals? A 95% confidence interval means that if we repeated the estimation process again and again on different samples from the population, there would be 95% probability that the given confidence interval is one containing the true parameter value (here, the difference of means), of all constructed confidence intervals. One can also encounter 99% and 90% confidence intervals, which correspond to the probability of 99% and 90%, respectively.

Confidence intervals are also commonly applied for estimation of the mean. To compute a confidence interval around the mean, one needs two things: the mean and the

standard error (SE), which should not be confused with the standard deviation. The latter is a dispersion measure that describes the average deviation from the mean (see Chapter 3). In contrast, the standard error is the standard deviation of the sampling distribution of the statistic, e.g. the mean. A **sampling distribution** is the distribution of a statistic calculated for many samples drawn from the same population. The standard error of the mean can be calculated by dividing the standard deviation of the sample by the square root of the number of observations in the sample:

$$SE_{\bar{x}} = \frac{s}{\sqrt{n}}$$

where $\bar{x}$ is the sample mean, s is the standard deviation, and n is the number of observations in the sample.

The computation of confidence intervals around the mean depends on the sample size. If the sample size is large (according to a rule of thumb, greater than 30), the lower boundary of a 95% confidence interval is computed as the sample mean minus 1.96 times the SE. The upper boundary is the sample mean plus 1.96 times the SE. The constant 1.96 is the value which corresponds to the significance level of 0.05 in the standard normal distribution with the mean of 0 and the standard deviation of 1. It is also known as the z-distribution (recall z-scores, which were discussed in Chapter 3). In the z-distribution, 95% of z-scores will be located between –1.96 and 1.96. If you need a 99% confidence interval, the constant is 2.58. For a 90% confidence interval, the constant is 1.64. The general formula for the lower and upper boundaries of a confidence interval looks as follows:

$$CI = \bar{x} \pm z_{\frac{1-p}{2}} \times SE$$

where $\bar{x}$ is the sample mean, SE is the standard error, p is the probability that you want for your confidence interval (e.g. 0.95 for a 95% CI, 0.99 for a 99% CI, etc.), and z is the z-score, which can be computed for a 95% confidence interval as follows:

```
> qnorm((1 - 0.95)/2, lower.tail = FALSE)
[1] 1.959964
```

If the sample size is small (less than or equal to 30), the t-distribution is used instead. The problem with small samples is that the scores are more spread out. The general formula for computing the confidence intervals for small samples is as follows:

$$CI = \bar{x} \pm t_{\text{—}, df = n-1} \times SE$$

where $\bar{x}$ is the sample mean, SE is the standard error, p is the probability that you want for your confidence interval, and t is the t-score for the probability and the number of degrees

of freedom, which is equal to the sample size minus one. The *t*-score can be obtained as follows (the example is given for the number of associations triggered by the high-frequency nouns):

```
> qt((1 - 0.95)/2, df = length(pym_high$assoc) - 1, lower.tail =
FALSE)
[1] 2.009575
```

Let us now compute the standard error and the 95% confidence interval (CI) for the number of associations triggered by the high-frequency nouns. Since the sample is greater than 30, we will use the approach based on the *z*-distribution.

```
> se.high <- sd(pym_high$assoc)/sqrt(length(pym_high$assoc)) #SE
for the high-frequency nouns
> se.high
[1] 0.1315214
> ci.lower.high <- mean(pym_high$assoc) - 1.96*se.high # the lower
boundary of the 95% CI around the mean for the high-frequency
nouns
> ci.lower.high
[1] 6.122218
> ci.upper.high <- mean(pym_high$assoc) + 1.96*se.high # the upper
boundary of the 95% CI around the mean for the high-frequency nouns
> ci.upper.high
[1] 6.637782
```

Next, let us do the same for the low-frequency nouns:

```
> se.low <- sd(pym_low$assoc)/sqrt(length(pym_low$assoc)) #SE for
the low-frequency nouns
> se.low
[1] 0.1447616
> ci.lower.low <- mean(pym_low$assoc) - 1.96*se.low # the lower
boundary of the 95% CI around the mean for the low-frequency nouns
> ci.lower.low
[1] 5.573718
> ci.upper.low <- mean(pym_low$assoc) + 1.96*se.low # the upper
boundary of the 95% CI around the mean for the low-frequency nouns
> ci.upper.low
[1] 6.141184
```

Now we have the lower and upper boundaries of the confidence intervals. It is common to visualize them on bar plots, such as the one shown in Figure 5.2. To create the graph, we will create a vector with two means:

```
> means <- c(mean(pym_high$assoc), mean(pym_low$assoc))
> means
[1] 6.380000 5.857451
```

Next, create a vector with the 95% CI lower boundaries and a vector with the 95% CI upper boundaries. Be careful and keep the same order: first, the statistics of the high-frequency nouns, and then those of the low-frequency nouns:

```
> ci.lower <- c(ci.lower.high, ci.lower.low)
> ci.lower
[1] 6.122218 5.573718
> ci.upper <- c(ci.upper.high, ci.upper.low)
> ci.upper
[1] 6.637782 6.141184
```

Finally, we will use the function `barplot2()` from the package `gplots` to create a bar plot of means and their confidence intervals. The argument `plot.ci` should have the TRUE value, `ci.l` specifies the vector with the lower boundaries, and `ci.u` the vector with the upper boundaries.

```
> barplot2(means, plot.ci = TRUE, ci.l = ci.lower, ci.u = ci.upper,
main = "Bar plot with 95% confidence intervals", xlab = "Frequency
groups", ylab = "Average number of associations", names = c("High",
"Low"))
```

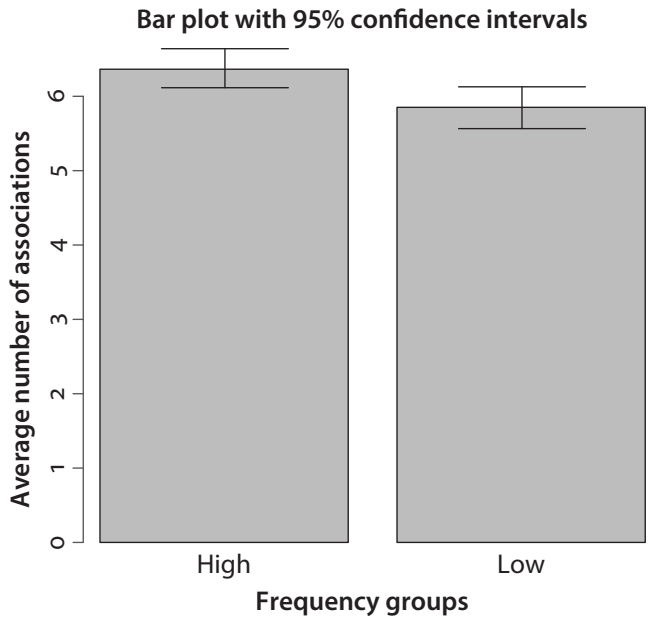
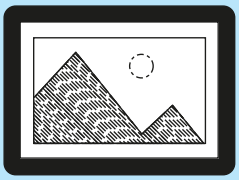


Figure 5.2. Bar plot with 95% confidence intervals around the means of the average number of associations triggered by high- and low-frequency nouns



How to create a bar plot with 95% confidence intervals with the help of `ggplot2`

To create a `ggplot2` version of a bar plot with 95% confidence intervals (Figure 5.2a), you should first create a small data frame that contains only the means, standard errors and group names for the high-frequency and low-frequency samples:

```
> assoc.df <- data.frame(group = c("High", "Low"), mean = means,
se = c(se.high, se.low))
> assoc.df
   group   mean       se
1  High    6.380000  0.1315214
2  Low     5.857451  0.1447616
```

To create the plot, you can use the following code:

```
> ggplot(assoc.df, aes(x = group, y = mean)) + geom_bar(stat =
"identity", fill = "lightblue", colour = "black") + xlab("Frequency
group") + ylab("Average number of associations") + geom_
errorbar(aes(ymin = mean - 1.96*se, ymax = mean + 1.96*se), width
= 0.2)
```

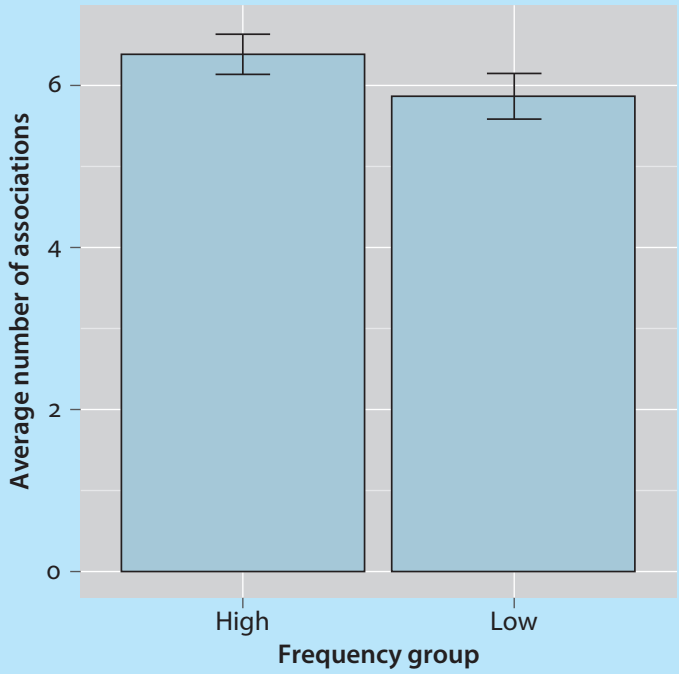


Figure 5.2a. A `ggplot2` version of the bar plot with 95% confidence intervals in Figure 5.2

Bar plots of sample means with 95% confidence intervals provide a lot of useful information. In addition to the average numbers of associations, they show how variable the data are and how much one can rely on the mean (or another statistic) as a measure of central tendency. The smaller the confidence interval, the less the error margin. Moreover, the overlap in 95% confidence intervals can provide an informal estimation of statistical significance. Non-overlapping 95% confidence intervals signal significant differences between samples at the significance level of 0.05. If the intervals overlap strongly, the difference is most probably not statistically significant. If there is a small overlap, as in our case, the means can still be significantly different. Recall that the *t*-test yielded a *p*-value of 0.004, so the difference is statistically significant. In case of overlap, only a statistical test can give a definite answer.

Note that confidence intervals around the mean are meaningful only if the mean is a good estimator of the central tendency, that is, your distribution is not skewed, there are no outliers, the samples are large, etc. If any of these conditions do not hold, you can either try to transform the data, as shown in Chapter 3, or use non-parametric bootstrap methods (see, for example, the package `simpleboot`).

Some common misconceptions about *p*-values, confidence intervals and hypothesis testing

Misconception 1: *p*-values show the probability that the null hypothesis is true.
Correction: *p*-values show the probability of observing the given and more extreme results if the null hypothesis were true.

Misconception 2: A 95% confidence interval means that the probability that this interval includes the population mean is 95%.
Correction: A 95% confidence interval means that this interval has a 95% probability of being one that contains the population mean.

Misconception 3: If your *p*-value is small, the difference is very large. If a *p*-value is large, the difference is very small.
Correction: If your *p*-value is small, this tells only about how confident you can be about the difference. Even a tiny difference can be statistically significant if the sample is very large. In other words, **statistical significance** does not tell you anything about the **effect size**.

(*Continued*)

Misconception 4: If your *p*-value is smaller than the significance level, you have **proven** that the null hypothesis is wrong.
Correction: If your *p*-value is smaller than the significance level, you can **reject** the null hypothesis (but you cannot **prove** that it is wrong!)

You can find other wide-spread misconceptions in Huck (2009).

5.3 Comparing concreteness scores of high- and low-frequency nouns with the help of a two-tailed Wilcoxon test

5.3.1 Data and hypotheses

To be able to work with the data and functions discussed in this case study you will need the following add-on packages:

```
> install.packages("ggplot2")
> library(Rling); library(ggplot2)
```

You will need the same datasets that were presented in the previous case study: `pym_high`, with 50 high-frequency nouns, and `pym_low`, with 51 low-frequency nouns. Both datasets can be found in the `Rling` package.

```
> data(pym_high)
> data(pym_low)
```

See the description of the datasets in the previous section. This time the variable of interest is *conc* (concreteness). It shows the average scores on the concreteness – abstractness scale given by the subjects on the scale from 1 to 7. The higher the score, the more concrete the noun. One can hypothesize that the high- and low-frequency nouns will on average have different concreteness scores. This hypothesis is non-directional because we do not have expectations which group will have higher scores than the other.

5.3.2 Descriptive statistics and visualizations: Strip charts and rug plots

As usual, let us first look at the main descriptive statistics. They demonstrate that the high-frequency nouns tend to have greater concreteness values that the low-frequency ones.

```
> summary(pym_high$conc)
Min.   1st Qu.  Median  Mean   3rd Qu.  Max.
1.830  3.202    5.850   5.074  6.590    7.000
> summary(pym_low$conc)
Min.   1st Qu.  Median  Mean   3rd Qu.  Max.
1.730  2.740    4.850   4.728  6.620    7.000
```

It is also instructive to look at the overall distributions, as was done in Chapter 3. The code below creates a Q–Q plot and a density plot of the concreteness scores in the high-frequency dataset (you can also create a histogram on your own):

```
> qqnorm(pym_high$conc, main = "Q-Q plot of concreteness scores")
> qqline(pym_high$conc)
> plot(density(pym_high$conc), main = "Density plot of concreteness
scores", xlab = "Concreteness")
```

The graphs are shown in Figure 5.3. They demonstrate that the distribution is not normal and that it is actually bimodal, having two local peaks. You can repeat the procedure to see that low-frequency nouns behave in a similar way.

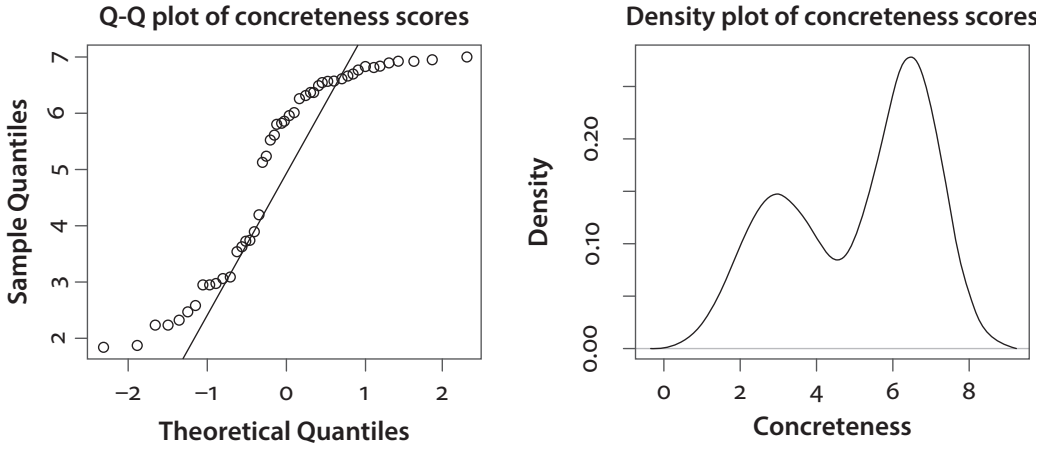


Figure 5.3. Concreteness scores of high-frequency nouns. Right: Q–Q plot; left: a density plot

Other useful alternatives to visualize distributions are a strip chart and a rug plot. The former can be created as follows:

```
> stripchart(list(pym_high$conc, pym_low$conc), main = "Distribution
of concreteness scores", group.names = c("high", "low"), method =
"jitter", xlim = c(1, 7))
```

The argument `group.names = c("high", "low")` provides labels for the two subplots, whereas `method = "jitter"` allows one to avoid overplotting of symbols. Adding some jitter is useful when your data contain many similar scores. Finally, `xlim = c(1, 7)` is added to display the full range of possible scores from 1 to 7.

It is also possible to add a 'rug' representation of each distribution to the plot. One has to specify the sides ('1' is bottom and '3' is top), where the rug representations should appear. You can see the result in Figure 5.4.

```
> rug(pym_high$conc, side = 1)
> rug(pym_low$conc, side = 3)
```

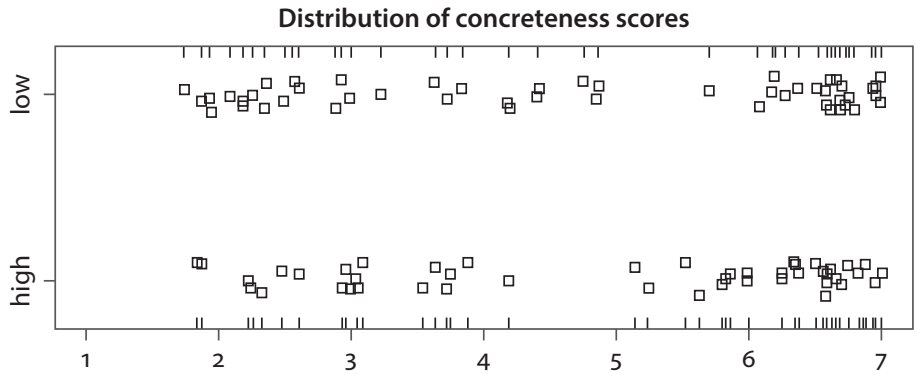
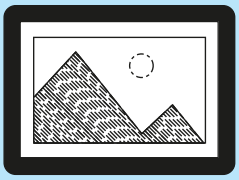


Figure 5.4. A strip chart with rug representations of concreteness scores of high- and low-frequency nouns



How to create a strip chart with the help of `ggplot2`

To represent two or more vectors with the help of a strip chart, you can use the following code (see the result in Figure 5.4a):

```
> pym_conc <- data.frame(conc = c(pym_high$conc, pym_low$conc),
freq = c(rep("high", 50), rep("low", 51)))
> ggplot(pym_conc, aes(x = freq, y = conc)) + geom_point(position
= position_jitter(width = 0.05), shape=0) + coord_flip() + labs(x
= "Frequency group", y = "Average concreteness score", ylim =
c(1, 7))
```

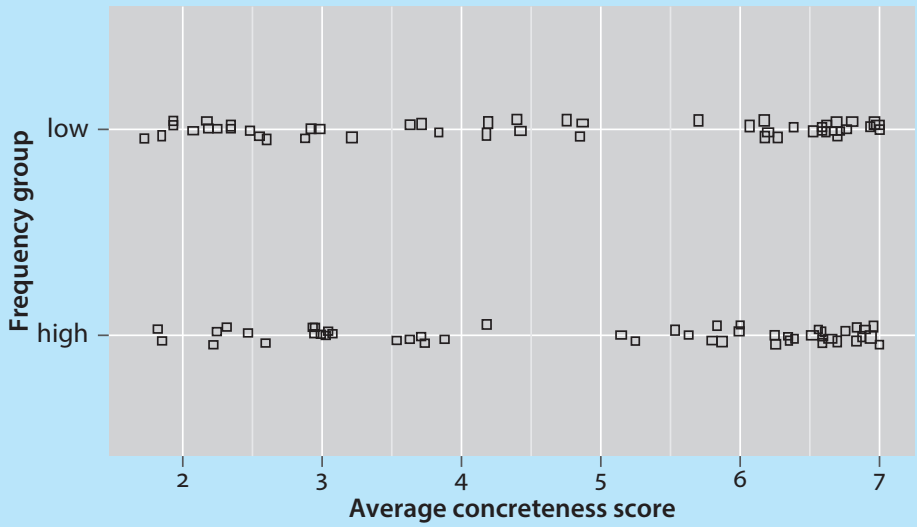


Figure 5.4a. A `ggplot2` version of the strip chart in Figure 5.4

What are the concrete and abstract nouns like? This is a good opportunity to practice in complex subsetting of data frames. Below is the code for retrieving all high-frequency nouns with concreteness scores greater than six, and their concreteness scores:

```
> pym_high[pym_high$conc > 6, 4, drop = FALSE]
           conc
home       6.25
church     6.59
door       7.00
college    6.38
girl       6.83
person     6.51
river      6.83
paper      6.89
earth      6.58
letter     6.94
poet       6.35
king       6.34
judge      6.25
bottle     6.94
valley     6.66
animal     6.75
forest     6.69
newspaper  6.56
cell       6.63
library    6.87
coast      6.59
stone      6.96
```

To perform this operation, one has to subset both rows and columns. The first expression in the square brackets specifies the rows to be selected. It tells R to select only the rows with *conc* scores greater than 6. The digit 4 after the first comma specifies the fourth column, *conc*. See Appendix 1 for more examples of subsetting. The final argument, `drop = FALSE`, is added in order to retain the row names. Without this argument, you will get only a sequence of numbers, as shown below.

```
> pym_high[pym_high$conc > 6, 4]
[1] 6.25 6.59 7.00 6.38 6.83 6.51 6.83 6.89 6.58 6.94 6.35 6.34 6.25
6.94 6.66 6.75 6.69 6.56 6.63 6.87 6.59 6.96
```

The smaller peak in the high-frequency sample (approximately between the scores 2 and 4) is represented by abstract nouns shown below. To retrieve them, you can use the following expression with `&`, which means that the score should be greater than two AND smaller than four:

```
> pym_high[pym_high$conc > 2&pym_high$conc < 4, 4, drop = FALSE]
           conc
time       2.47
life       2.96
mind       2.60
pressure   3.63
effort     2.22
hour       2.93
trouble    2.25
science    3.05
series     3.88
length     3.75
health     3.54
event      3.72
dream      3.03
duty       2.32
victory    2.95
silence    3.09
```

The words that are located in the 'valley' between those two peaks are as follows:

```
> pym_high[pym_high$conc > 4&pym_high$conc < 6, 4, drop = FALSE]
              conc
university    5.87
property      5.99
volume        5.14
lord          4.18
product       5.80
leader        5.83
master        5.53
contract      5.24
disease       5.63
```

Some of them are polysemous or vague, e.g. *property* (an object that one owns and an abstract feature), *volume* (a book and an abstract characteristic), *product* (a material object or abstract result), *contract* (an agreement or a written document that represents it) and *university* (a social institution or a building, or people who work there). It is also interesting to find nouns related to social status (*lord*, *leader* and *master*). The noun *disease* is a state with salient physical manifestations, which probably explains the in-between status of the word.

5.3.3 Inferential statistics: The two-tailed Wilcoxon test

After we have explored the data, it is time to perform an appropriate test. We have independent observations, so the test will be independent. The alternative hypothesis is

non-directional: we expect that the concreteness scores of the high-frequency nouns are different from the concreteness scores of the low-frequency nouns. Thus, the test should be two-tailed. Shall we use a parametric or non-parametric test? The previous section discussed the main assumptions of the parametric *t*-test for independent samples. There are some problems. First, the psychological scaling data are problematic. Even when numbers from 1 to 7 are used, subjects may treat the distances between the points at the ends of the scale differently than the distances between the points in the middle of the scale. Thus, the data are in fact quasi-interval, although it is quite common to treat them as interval, for instance, by computing averages, as it is done here.

It is also clear that the assumption of normality is not met: each group has two more or less distinct clusters that correspond to abstract and concrete nouns. The Shapiro-Wilk test supports this conclusion:

```
> shapiro.test(pym_high$conc)

      Shapiro-Wilk normality test

data: pym_high$conc
W = 0.8468, p-value = 1.269e-05

> shapiro.test(pym_low$conc)

      Shapiro-Wilk normality test

data: pym_low$conc
W = 0.8553, p-value = 1.827e-05
```

The very small *p*-values support our observation that both samples strongly depart from normality. Since the data are quasi-interval and markedly non-normal, with two clusters for abstract and concrete nouns, we will be cautious and use the non-parametric option, the Wilcoxon test, even though both sample sizes are greater than 30. The test is equivalent to the Mann-Whitney test, and the Mann-Whitney test is also sometimes called the *U*-test because of the name of the test statistic. The Mann-Whitney and Wilcoxon tests are called non-parametric because they do not assume a distribution of any particular shape. There is another important difference. Instead of actual scores, the Mann-Whitney and Wilcoxon tests use the ranks of the scores in each group. In the Mann-Whitney test, the test statistic *U* is computed by comparing all possible pairs of values, one from each group, and then giving these pairs the score 1 if the observation from the first group is greater than the one from the second group, and 0 is the first group observation is lower than that from the second group. The Wilcoxon test has a slightly different procedure: first, all observations from both groups are put together and ranked; next, the test statistic *W* is computed as the sum of the ranks in the smaller group. However, the results of the two tests are identical. Since these tests are based on ranks, they can be used with **ordinal** data. They are also regarded as tests of the difference in medians, rather than the difference in means.

As was mentioned in Section 5.1, the independent Wilcoxon and Mann-Whitney tests have a number of assumptions, as well:

- *Each sample has been drawn randomly from the population.*
- *The observations are independent, both between the two samples and within each sample.*
- *The measurement scale is at least ordinal.*
- *The underlying population distributions should be of a similar shape.*

Since all these assumptions are met, the Wilcoxon test can be performed. The two-tailed option and independent test are the default options, so they do not have to be specified. Instead, we will add `correct = FALSE` to cancel the continuity correction,[3] and `conf.int = TRUE` to obtain the 95% confidence intervals.

```
> wilcox.test(pym_high$conc, pym_low$conc, correct = FALSE, conf.
int = TRUE)

     Wilcoxon rank sum test

data: pym_high$conc and pym_low$conc
W = 1380, p-value = 0.4757
alternative hypothesis: true location shift is not equal to 0
95 percent confidence interval:
-0.2699598 0.7700643
sample estimates:
difference in location
             0.1599588
```

The *W*-statistic is 1380, and the *p*-value is much greater than 0.05. Therefore, we cannot reject the null hypothesis of no differences in the concreteness scores of the high- and low-frequency nouns. Even though the descriptive statistics were slightly greater for the former ones, these differences are not statistically significant.

5.4 Comparing associations produced by native and non-native speakers: The dependent one-tailed *t*-test

5.4.1 Creating simulation data

To reproduce the code in this case study, you will need only one add-on package, `Rling`.

3. According to some statisticians, the correction for continuity should be applied when non-parametric tests for continuous variables are run on discrete data. The correction decreases the risk of Type I error. See Sheskin (2011: 252).

```
> library(Rling)
```

The previous case studies in this chapter were based on independent samples. Therefore, unpaired tests were performed. A different situation is observed when one has paired observations. Imagine that you want to compare the average number of associations per word produced by native and non-native speakers. In that case, you would have a pair of scores for each word. Another example is language proficiency scores of a group of students before and after some innovative teaching technology was used in the class. There would be two scores per student. All these are examples of paired data, which require a dependent, or paired test.

To provide an illustration, we will take the scores from the high-frequency nouns in the dataset `pym_high`, which was discussed in the previous sections, and will assume that these are native speakers' scores. See Section 5.2 for more details about the dataset and the variables.

```
> data(pym_high)
```

The scores for non-native speakers will be generated automatically. First, we will create a vector with 50 numbers that represent the differences between the native speakers and non-native speakers. The function `rnorm()` will generate normally distributed data with the given number of observations (50), the mean (–1.35) and the standard deviation (1.27). The mean and the standard deviation are completely arbitrary. A negative mean means that the average number of associations produced by non-native speakers should be smaller. Note that your vector will look different because the numbers are generated randomly.

```
> diff <- rnorm(50, -1.35, 1.27)
> head(diff)
[1] -1.8013467 -2.1152391 -1.6954642 -0.8052109 -1.2967283
[6] -0.5255384
```

Next, the fictitious scores of the non-native speakers are created, which are equal to the native speakers' scores plus the difference:

```
> nn <- pym_high$assoc + diff
> head(nn)
[1] 5.198653 4.664761 5.184536 6.714789 4.583272 7.434462
```

Next, the scores are rounded up to two decimal places:

```
> nn <- round(nn, 2)
> head(nn)
[1] 5.20 4.66 5.18 6.71 4.58 7.43
```

The data are ready for the demonstration of the paired test.

5.4.2 Performing the dependent *t*-test

The assumptions of the dependent *t*-test, which were listed in Section 5.1, are very similar to those of the independent *t*-test. One important difference is that instead of testing the samples for normality, we test the **differences** between the pairs of scores. We already have the vector with differences. They should be normally distributed because this is the way how we have created them. Not surprisingly, the Shapiro-Wilk test returns a high *p*-value.

```
> shapiro.test(diff)

     Shapiro-Wilk normality test

data: diff
W = 0.9922, p-value = 0.9832
```

If you have real observed scores, the vector with differences can be obtained by subtracting the scores of one group (imaginary vector `groupA`) from the scores of the other group (imaginary vector `groupB`):

```
> diff <- groupA – groupB # do not run; only given as an example
```

Since all other assumptions hold, as well, we can use the parametric *t*-test. To perform a dependent, or paired test, one should add `paired = TRUE` to the list of arguments. Because the alternative hypothesis is directional (that is, native speakers produce more associations than non-native speakers) and the test is one-tailed, it is also necessary to add `alternative = "greater"`:

```
> t.test(pym_high$assoc, nn, alternative = "greater", paired =
TRUE)

     Paired t-test

data: pym_high$assoc and nn
t = 8.4935, df = 49, p-value = 1.698e-11
alternative hypothesis: true difference in means is greater than 0
95 percent confidence interval:
1.330853      Inf
sample estimates:
mean of the differences
             1.658162
```

The *t*-statistic is 8.494 with 49 degrees of freedom. The *p*-value is very small. This means that the null hypothesis of no difference can be rejected and the difference between the average number of associations produced by native and non-native speakers is statistically significant.

In case of violations of the test assumptions, one can use the paired version of the Wilcoxon (Mann-Whitney) test. The default unpaired version of this test was presented in Section 5.3. To perform a paired test, add the argument `paired = TRUE`.

5.5 Summary

This chapter has presented dependent and independent, parametric and non-parametric, one-tailed and two-tailed tests for investigating differences between two group means. The choice of the appropriate test requires a careful check of numerous assumptions, but you will be rewarded by statistically reliable results. You have also learnt about standard errors and confidence intervals around the mean. All these fundamental notions and skills will be useful in the following case studies.

Reporting the *t*-test and the Wilcoxon/Mann-Whitney test

When you write up the results of your independent or dependent *t*-test or its non-parametric equivalent, you should mention the test statistic (t or W), the degrees of freedom and the p-value. It is crucial to mention whether you performed an independent or dependent test, one-tailed or two-tailed. For example, one could write up the results of the first case study as follows: "An independent one-tailed *t*-test with Welch's correction was performed to compare the average number of associations produced by native speakers for two lists of randomly selected high- and low-frequency words. On average, high-frequency words triggered a significantly higher number of associations (M = 6.38, SE = 0.13) than low-frequency words (M = 5.86, SE = 0.14), $t_{(98.281)} = 2.67$, $p = 0.004$."

Another application of the *t*-test

In addition to comparing two dependent or independent samples, the *t*-test can be used on one sample only when we know the population mean and want to find out whether the sample mean M is significantly different from the population mean μ (see Baayen 2008: Section 4.1.2; Gries 2013: 205–209).