19 Elementary statistical testing with R

STEFAN TH. GRIES

1 Introduction

In Brian Joseph's final editorial as the editor of what many see as the flagship journal of the discipline, *Language*, he commented on recent developments in the field. One of the recent developments he has seen happening is the following:

> Linguistics has always had a numerical and mathematical side . . . but the use of quantitative methods, and, relatedly, formalizations and modeling, seems to be ever on the increase; rare is the paper that does not report on some statistical analysis of relevant data or offer some model of the problem at hand. (Joseph 2008: 687)

For several reasons, this appears to be a development for the better. First, it situates the field of linguistics more firmly in the domains of social sciences and cognitive science to which, I think, it belongs. Other fields in the social sciences and in cognitive science – psychology, sociology, computer science, to name but a few – have long recognized the power of quantitative methods for their respective fields of study, and since linguists deal with phenomena just as multifactorial and interrelated as scholars in these disciplines, it was time we also began to use the tools that have been so useful in neighboring disciplines.

Second, the quantitative study of phenomena affords us with a higher degree of comparability, objectivity, and replicability. Consider the following slightly edited statement:

> there is at best a very slight increase in the first four decades, then a small but clear increase in the next three decades, followed by a very large increase in the last two decades (this chapter, cf. below Section 2.4).

Such statements are uninformative as long as:

- notions such as 'very slight increase', 'small but clear increase', and 'very large increase' are not quantified: one researcher considers an increase in frequency from 20 observations to 40 'large' (because the frequency increases by 100%), another considers it small (because the frequency increases by only 20), and yet another considers it intermediate, but considers a change from 160 to 320

large (because, while this also involves an addition of 100% to the first number, it involves an addition of 160, not just 20) . . .;
- the division of the nine decades into (three) stages is simply assumed but not justified on the basis of operationalizable data;
- the lengths of the nine stages (first four decades, then three decades, then two decades) is simply stated but not justified on the basis of data.

Third, there is increasing evidence that much of the cognitive and/or linguistic system is statistical or probabilistic in nature, as opposed to based on clear-cut categorical rules. Many scholars in first language acquisition, cognitive linguistics, sociolinguistics, psycholinguistics etc. embrace, for instance, exemplar-based models of language acquisition, representation, processing, and change. It is certainly no coincidence that this theoretical development coincides with the methodological development pointed out by Joseph, and if one adopts a probabilistic theoretical perspective, then the choice of probabilistic – i.e. statistical – tools is only natural; cf. Bod, Hay, and Jannedy (2003) for an excellent overview of probabilistic linguistics.

For obvious limitations of space, this chapter cannot provide a full-fledged introduction to quantitative methods (cf. the references below). However, the main body of this chapter explains how to set up different kinds of quantitative data for statistical analysis and exemplifies three tests for the three most common statistical scenarios: the study of frequencies, of averages, and of correlations. Joseph himself is a historical linguist and it is therefore only fitting that this chapter illustrates these guidelines and statistical methods using examples from language variation and change.

2 Elementary statistical tests

In this section, I will first illustrate in what format data need to be stored for virtually all spreadsheet software applications and statistical tools and how to load them into R, the software that I will use throughout this chapter (cf. Section 2.1). Then, Section 2.2 will discuss how to perform simple statistical tests on frequency data, Section 2.3 will outline how the central tendencies of variables (i.e. averages) can be tested, and Section 2.4 will be concerned with tests for correlations. Each section will also exemplify briefly how to create useful graphs within R.

Limitations of space require two comments before we begin. First, I cannot discuss the very foundations of statistical thinking, the notions of (independent and dependent) variables, (null and alternative) hypotheses, Occam’s razor, etc., but must assume the reader is already familiar with these (or reads up on them in, say, Gries 2009: chapter 5, 2013: chapters 1, 5). Second, I can only discuss a few very elementary statistical tests here, although many more and more interesting things are of course possible. I hope, however, that this chapter will trigger the reader's interest in additional references and the exploration of the methods presented here.

2.1 Tabular data and how to load them into R

Trivially, before any statistical analysis of data can be undertaken, three steps are necessary. First, the data have to be gathered and organized in a suitable format. Second, they must be saved in a way that allows them to be imported into statistical software. Third, the data have to be loaded into some statistical software. The first subsection of this section deals with these three steps.

As for the first step, it is absolutely essential to store your data in a spreadsheet software application so that they can be easily evaluated both with that software and with statistical software. There are three main rules that need to be considered in the construction of a table of raw data:

(1) each data point, i.e. count or measurement of the dependent variable(s), is listed in a row on its own;

(2) every variable with respect to which each data point is described is recorded in a column on its own;

(3) the first row contains the names of all variables.

For example, Hundt and Smith (2009) discuss the frequencies of present perfects and simple pasts in British and American English in the 1960s and the 1990s. One of their overview tables, Table 2 (Appendix 2), is represented here as Table 19.1.

Table 19.1. *Table 2 (Appendix 2) from Hundt and Smith (2009), cross-tabulating tenses and corpora*

	LOB (BrE 1961)	FLOB (BrE 1991)	BROWN (AmE 1961)	FROWN (AmE 1992)	Totals
present perfect	4196	4073	3538	3499	15306
simple past	35821	35276	37223	36250	144570
Totals	40017	39349	40761	39749	159876

This is an excellent overview and may look like a good starting point, but this is in fact nearly always already the *result* of an evaluation rather than the *starting point* of the raw data table. Notably, Table 19.1 does not:

- have one row for each of the 159,876 data points, but it has only two rows;
- have one column for each of the three variables involved (*tense*: present perfect vs. simple past; *variety*: BrE vs. AmE; *time*: 1960s vs. 1990s), but it has one column for each combination of *variety* and *time*.

The way that raw data tables normally have to be arranged for statistical processing requires a reorganization of Table 19.1 into Table 19.2, which then fulfills the three above-mentioned criteria for raw data tables.

Once the data have been organized in this way, the second step before the statistical analysis is to save them so that they can be easily loaded into a statistics application. To that end, you should save the data into a format that makes them maximally readable by a wide variety of programs. The simplest way to do this is to save the data into a tab-separated file, i.e. a raw text file in which different columns are separated from each other with tabs. In Libre-Office Calc, one first chooses *File: Save As . . .*, then chooses 'Text CSV (.csv)' as the file type, and chooses {Tab} as the *Field delimiter*.[1]

Table 19.2. *Reorganization of Table 2 (Appendix 2) from Hundt and Smith (2009)*

TENSE	VARIETY	TIME
present_perfect	BrE	1960s
	4195 more rows like the one immediately above	
present_perfect	BrE	1990s
	4072 more rows like the one immediately above	
present_perfect	AmE	1960s
	3537 more rows like the one immediately above	
present_perfect	AmE	1990s
	3498 more rows like the one immediately above	
simple_past	BrE	1960s
	35820 more rows like the one immediately above	
simple_past	BrE	1990s
	35275 more rows like the one immediately above	
simple_past	AmE	1960s
	37223 more rows like the one immediately above	
simple_past	AmE	1990s
	36249 more rows like the one immediately above	

To perform the third step, i.e. to load the data into statistical software, you must first decide on which software to use. From my point of view, the best statistical package currently available is the open source software environment R (cf. R Development Core Team 2006–2013). The basic software as

[1] I recommend using only word characters (letters, numbers, and underscores) within such tables. While this is strictly speaking not necessary to guarantee proper data exchange between different programs – since most programs nowadays provide sophisticated import functions or wizards – it is my experience that 'simple works best.'

well as supplementary packages can be downloaded from www.r-project.org. While R does not feature a clickable graphical user interface (GUI) by default, such a GUI can be installed (cf. the appendix) and R is extremely powerful both in terms of the sizes of data sets it can handle and the number of procedures it allows the user to perform – indeed, since R is a programming language, it can do whatever a user is able to program. In addition, R's graphical facilities are unrivaled and since it is an open source project, it is freely available and has extremely fast bugfix-release times. For these and many other reasons, R is used increasingly widely in the scientific community, but also in linguistics in particular, and I will use it here, too.

When R is started, by default it only shows a fairly empty console and expects user input from the keyboard. Nearly all of the time, the input to R consists of what are called *functions* and *arguments*. The former are commands that tell R what to do; the latter are specifics for the commands, namely what to apply a function to (e.g. a value, the first row of a table, a complete table, etc.) or how to apply the function to it (e.g. what kind of logarithm to compute, a binary log, a natural log, etc.).[2] The simplest way to read a table with raw data from a tab-separated file created as above involves the function `read.table`, which, if the raw data table has been created as outlined above and in note 1, requires only three arguments:

- the argument `file`, which specifies the path to the file containing the data;
- the argument `header`, which can be set to `T` (or `TRUE`) or `F` (or `FALSE`), where `T/TRUE` means 'the first row contains the variable names', and `F/FALSE` means the opposite;
- the argument `sep`, which specifies the character that separates columns from each other and which should therefore be set to a tabstop, "\t".

Thus, to import a raw data table from an input file <C:/Temp/example1.txt> and store that table in a so-called *data frame* (called `data.table`) in R, you enter the following line of code (where the "<–" tells R to store something in the data structure to the left of the 'arrow' and where "¶" means 'press ENTER'):

```
data.table<–read.table("file=C:/Temp/example1.txt",
     header=TRUE, sep="\t")¶
```

[2] The general logic of functions and arguments and different kinds of data structures – while essential to working with spreadsheet software such as LibreOffice Calc as well as programming languages such as R – will not be discussed here in detail; cf. the recommended further readings for more information.

To check whether the data have been read in correctly, it is always useful to look at the structure of the imported data first, using the function `str`, which provides all the column names together with some information on what the columns contain, namely their kind of data (integer numbers, character strings as factors, etc.) as well as the first few values. If you had read in a file of the kind shown in Table 19.2, then this is what the output would look like:

```
str(data.table)¶
'data.frame':   159876 obs. of 3 variables:
$ TENSE   : Factor w/ 2 levels "present_perfect",..: 1 1 1 1 1 1 1 1 1 1...
$ VARIETY: Factor w/ 2 levels "AmE","BrE": 2 2 2 2 2 2 2 2 2 2...
$ TIME    : Factor w/ 2 levels "1960s","1990s": 1 1 1 1 1 1 1 1 1 1...
```

The simplest way to be able to access all values of a column at the same time by just using the column name involves the function `attach`, which requires the data frame's name as its only argument and typically does not return any output:

```
attach(data.table)¶
```

While the above is the typical way to input data into R, when the data set in question is small, another way is sometimes simpler, namely entering the data oneself. For example, if you have collected the lengths of five indirect objects and of five direct objects in terms of number of phonemes and want to quickly compare their mean lengths, it is maybe not necessary to create a tab-delimited input file – you can just enter the data into R and assign them to a data structure, a so-called *vector*, using the function `c`, which concatenates the elements provided as arguments (numbers or character strings) into a vector.

```
indir.objects<-c(1, 2, 3, 4, 5)
dir.objects<-c(3, 5, 4, 4, 3)
```

Then, computing the means is easy and returns 3 and 3.8 for indirect and direct objects respectively:

```
mean(indir.objects)¶
[1] 3
mean(dir.objects)¶
[1] 3.8
```

Once the data are available in either the tabular data frame or the unidimensional vector format, it often takes only a very small amount of R code – usually only one line – to run statistical analyses or produce quite revealing graphs, some of which will be exemplified in the three following sections.

2.2 Two-dimensional frequency data

The first application to be discussed here involves two-dimensional frequency tables, i.e. tables such as Table 19.2, and their evaluation. As an example, I will discuss fictitious data that bear on the question to what degree, if any, the syntactic form of a response to a question is determined by the syntactic form of the question. Let us assume researchers asked subjects altogether 200 instances of two types of questions in Dutch, whose English glosses are listed in (4) and (5).

(4) Of whom is this cap? [prepositional question type]

(5) Whose cap is this? [non-prepositional question type]

The researchers then recorded subjects' answers to these questions and counted how many times the answer was a prepositional or non-prepositional one. One main point of such a study could be to find whether prepositional and non-prepositional questions trigger prepositional and non-prepositional responses respectively.[3] Let us assume the frequencies listed in Table 19.3 were obtained.

Table 19.3. *Fictitious frequencies obtained in a question-answer experiment*

	Question: with prep.	Question: without prep.	Totals
Answer: with prep.	98	64	162
Answer: without prep.	2	36	38
Totals	100	100	200

First, the data need to be entered into R. With small two-dimensional tables like these, it is easy to enter them directly rather than prepare the above type of raw data table to read in. In the following line, the function `matrix` creates a two-dimensional matrix of the four values, which are listed column-wise and arranged into `ncol=2` columns.

```
data.matrix<-matrix(c(98, 2, 64, 36), ncol=2)¶
```

Typically, it is useful to also provide the matrix with row and column names because this facilitates the subsequent interpretation of statistics and graphs. The following line provides row names (`ANSWER=...`) and column names (`QUESTION=...`) to the matrix and outputs it so one can check it.

[3] This example is modeled after Levelt and Kelter (1982).

```
attr(data.matrix, "dimnames")<-list(ANSWER=c("+ prep", "– prep"),
      QUESTION=c("+ prep", "– prep"))¶
data.matrix¶
        QUESTION
ANSWER   + prep – prep
  + prep     98     64
  – prep      2     36
```

If you want to see the row and column totals, too, this is how they can be obtained:

```
addmargins(data.matrix)¶
        QUESTION
ANSWER   + prep – prep  Sum
  + prep     98     64  162
  – prep      2     36   38
  Sum       100    100  200
```

Obviously, when the question involves a preposition, the answer nearly always does, too, whereas if the question does not involve a preposition, then the answer still contains a preposition more often than not, but the effect is much less extreme. The question arises whether this difference – 98:2 vs. 64:36 – is large enough to be significant, a question which is addressed by the chi-square test for independence. This test requires that all observations are independent of each other and that 80+% of the expected frequencies are larger than 5.

We assume for now that the 200 responses are completely independent of each other (and will check the expected frequencies shortly). You can then use the function `chisq.test`, which in the standard form to be discussed here requires the matrix to be tested (`data.matrix`) and an argument `correct`, which can be set to `TRUE` or `FALSE` depending on whether you want to use a correction for continuity, which we do not want here (because the sample size is greater than 60). For reasons that will become clear shortly, it is best to not just compute the test but also assign the result of the test to another data structure:

```
data.matrix.test<-chisq.test(data.matrix, correct=FALSE)¶
data.matrix.test¶
      Pearson's Chi-squared test
data: data.matrix
X-squared = 37.5569, df = 1, p-value = 8.879e-10
```

The test shows that there is a highly significant effect:[4] there is definitely a correlation between the questions and the answers. The question is what this

[4] The choice of words 'highly significant' is based on the following, frequently-used classification: $p<0.001$: 'highly significant'; $0.001\leq p<0.01$: 'very significant'; $0.01\leq p<0.05$: 'significant'.

correlation looks like and whether the expected frequencies are large enough to allow the chi-square test in the first place. As for the latter, the chi-square test in R does not just compute the above output but also some additional information such as the expected frequencies, i.e. the frequencies one would expect to find if questions and answers were *not* related. These can be obtained by requesting them from the data structure `data.matrix.test`:

```
data.matrix.test$exp¶
      QUESTION
ANSWER   + prep − prep
  + prep     81    81
  − prep     19    19
```

This table not only shows that the expected frequencies are large enough to allow the chi-square test. They also show what the effect looks like: we observed:

- more prepositional answers after prepositional questions than expected (98>81);
- fewer prepositional answers after preposition-less questions than expected (64<81);
- fewer preposition-less answers after prepositional questions than expected (2>19);
- more preposition-less answers after preposition-less questions than expected (36>19).

Since this piecemeal comparison of observed and expected frequencies is somewhat tedious, it is usually easier to inspect the so-called Pearson residuals. Pearson residuals can be computed for each cell in a table; they are positive and negative when a cell's frequency is larger or smaller than expected respectively, and the more they deviate from 0, the stronger the effect. From a purely exploratory perspective, Pearson residuals smaller than -3.841 or greater than 3.841 are particularly noteworthy.

```
data.matrix.test$res¶
      QUESTION
ANSWER       + prep      − prep
  + prep   1.888889  −1.888889
  − prep  −3.900067   3.900067
```

The findings are the same as above, but they are easier to identify than from the comparisons of observed and expected frequencies, and we also now see that the effects for the prepositionless answers are somewhat more pronounced.

A graphical representation that makes this even more obvious is the so-called association plot, which is shown in Figure 19.1: black boxes on top of the dashed lines and grey boxes below the dashed lines represent cell frequencies that are larger and smaller than expected respectively; the heights of the boxes are proportional to the above residuals and the widths are proportional to the square roots of the expected frequencies.

```
assocplot(t(data.matrix))¶
```

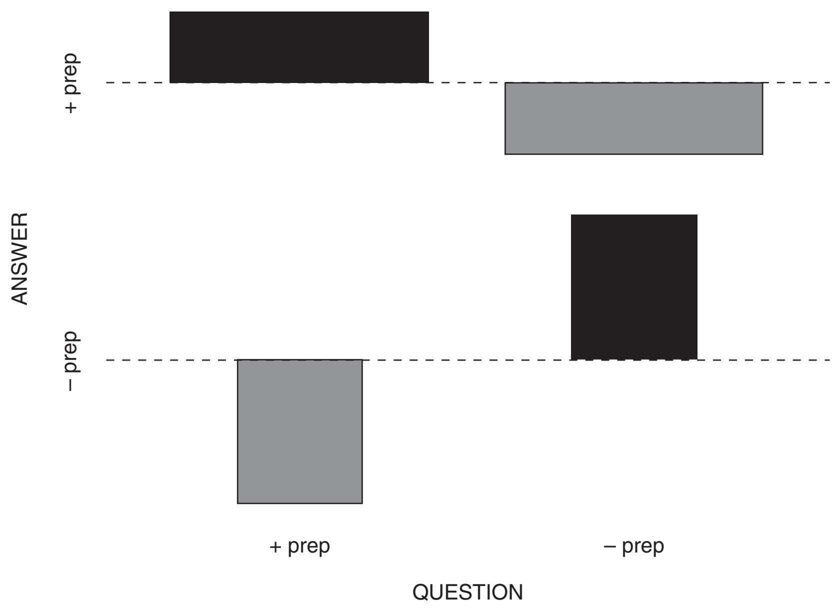


Figure 19.1. *Association plot for the relation between question and answer syntax*

The only thing that remains to be done is to quantify the size of the effect. Since chi-square values are correlated with sample sizes, one cannot readily use chi-square to identify effect sizes or compare them across different studies. Instead, one can use a correlation coefficient called Cramer's V, which falls between 0 and 1, and the larger the value, the stronger the correlation. Cramer's V is computed as shown in (6).

(6) $$\text{Cramer's } V = \sqrt{\frac{\chi^2}{n \cdot \left(min(n_{rows}, n_{columns}) - 1\right)}}$$

If we compute Cramer's V for the present data set, we obtain the result in (7).

(7) $$\text{Cramer's } V = \sqrt{\frac{37.5569}{200 \cdot \left(min(2,2) - 1\right)}} = \sqrt{\frac{37.5569}{200}} = 0.433$$

In R:

```
sqrt(data.matrix.test$stat/(sum(data.matrix)*min(dim(data.
     matrix)-1)))¶
X-squared
0.4333408
```

Thus, the data show an effect such that the type of question determines the type of answer: prepositional and prepositionless questions yield higher frequencies of occurrence of prepositional and prepositionless answers respectively; the effect is highly significant (p<0.0001) and intermediately strong (Cramer's V=0.433).[5]

2.3 Differences between central tendencies of variables

Often, the statistic of interest is not just the observed frequency of some phenomenon, but the central tendency of some phenomenon, i.e. what is commonly referred to as the average. At the risk of simplifying somewhat, we can say that there are two main averages for numeric data, the arithmetic mean and the median. Consider the following vector of numbers:

```
x<-c(0,0,0,1,1,1,2,2,5)¶
```

The arithmetic mean of the numbers in `x` is the quotient of the sum of the values in `x` (12) divided by the number of elements of `x` (9), i.e. $1^1/_3$. The median, by contrast, is the value you get when you sort the numbers according to their size and pick the one in the middle, i.e. 1:[6]

```
mean(x)¶
[1] 1.333333
median(x)¶
[1] 1
```

[5] The choice of words 'intermediately strong' is based on the following, frequently-used classification: 0.1≤effect size<0.3: 'small effect'; 0.3≤effect size<0.5: 'medium effect'; effect size≥0.5: 'large effect'.

[6] If the vector has an equal number of elements, the median is the arithmetic mean of the two middle values.

A frequent scenario is, then, that one wants to compare the central tendencies of two vectors to see whether they are significantly different from each other. Consider the case where you have collected the lengths of subordinate clauses in words in 10 samples of spoken and 10 samples of written data and obtained the following results:

(8) spoken: 11, 10, 9, 6, 8, 9 11, 7, 8, 6

(9) written: 13, 14, 12, 13, 11, 14, 7, 10, 12, 12

These data were stored in a tab-separated text file <C:/Temp/subcl_lengths.txt> as shown in Table 19.4, which you can load as discussed above in Section 2.1.

```
data.table<-read.table("C:/Temp/subcl_lengths.txt",
      header=TRUE, sep="\t")¶
str(data.table)¶
'data.frame':   20 obs. of 3 variables:
$ CASE : int 1 2 3 4 5 6 7 8 9 10 ...
$ MODE : Factor w/ 2 levels "spoken","written": 1 1 1 1 1 1 1 1 1 1 ...
$ LENGTH: num 11 10 9 6 8 9 11 7 8 6 ...
attach(data.table)¶
```

Table 19.4. *Lengths of subordinate clauses in two samples (of spoken and written data)*

CASE	MODE	LENGTH
1	spoken	11
2	spoken	10
. . .	. . .	. . .
19	written	12
20	written	12

Once the data have been loaded, the best approach is often to explore them graphically. One immensely informative plot for summarizing numeric variables is the so-called boxplot. The corresponding R function, `boxplot`, takes two arguments: a formula in which a dependent variable (here, the length of the subordinate clause) precedes the tilde and the independent variable (here, the mode) follows it, and the argument `notch=TRUE`, which creates notches whose function will be explained shortly. The result is shown (in slightly modified form) in Figure 19.2.

```
boxplot(LENGTH~MODE, notch=TRUE)¶
```

This plot provides a great deal of information:

- the thick horizontal lines correspond to the medians;
- the upper and lower horizontal lines indicate the central 50% of the data around the median (approximately the 2nd and 3rd quartiles);
- the upper and lower end of the whiskers extend to the most extreme data point which is no more than 1.5 times the height of the box away from the box;
- values outside of the range of the whiskers are marked individually as small circles;
- the notches on the sides of the boxes provide an approximate 95% confidence interval for the difference of the medians: if they overlap, then the medians are most likely not significantly different.

Of course, we also want to know the exact medians. These can be computed with the following line of code, which basically means 'apply the function

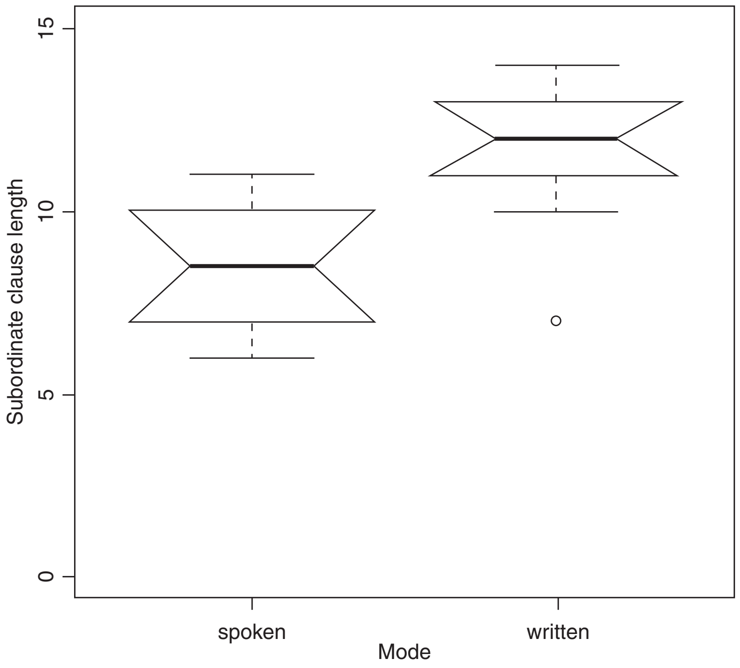


Figure 19.2. *Boxplot for the relation between subordinate clause length and mode*

`median` to the data you get when you split the values of `LENGTH` into the groups resulting from `MODE`':

```
tapply(LENGTH, MODE, median)¶
 spoken written
    8.5    12.0
```

And since one should always provide a measure of dispersion for measures of central tendency, we apply the same type of code to retrieve a simple measure of dispersion for the lengths in the spoken and the written data, the interquartile ranges, which indicate the spread of the central 50% around the medians:

```
tapply(LENGTH, MODE, IQR)¶
 spoken written
   2.50    1.75
```

From Figure 19.2, it already seems as if the differences between the two modes will be significant since the medians are fairly far apart and the notches do not overlap. In spite of this, the data must, of course, be tested, and the test that would normally be used for such data is the *t*-test for independent samples (esp. since, unlike in most cases, the data do not differ significantly from a normal distribution).[7] On the other hand, since the sample sizes are very small and linguistic data will often be non-normal, I will instead discuss a test that is slightly less powerful, but that can be used regardless of whether the data are normally distributed or not, the *U*-test (sometimes also called the two-sample Wilcoxon test). This test only requires that the observations are all independent of each other and its function in R is `wilcox.test`. It is best used with four arguments:

- a formula of the same kind as used for `boxplot`: dependent variable ~ independent variable;
- the argument `paired`, which can be set to `TRUE` or `FALSE`, where `TRUE` means the values of the two groups form meaningful pairs, and `FALSE` means the opposite. Since in this case the length of any one subordinate clause in speaking is not related to that of any one subordinate clause from a written file, we set `paired=FALSE`;
- the argument `correct`, which can be set to `TRUE` or `FALSE` depending on whether you want to apply a correction for continuity or not as is sometimes recommended for small sample sizes. In the interest of comparability of the test with most other statistical software, we set this to `FALSE` for now;
- the argument `exact`, which can be set to `TRUE` or `FALSE` depending on whether you want R to compute an exact test (for small sample sizes) or not. Again, in the interest of comparability, we set this to `FALSE`.

[7] At the risk of considerable simplification, a distribution of values is normal if, on the whole, most of its values do not differ much from the overall mean and if the differences of values from the overall mean are equally much positive and negative. In graphical terms, normal distributions can usually be identified by a bell-shaped curve/histogram.

With these settings, a *U*-test yields the following results:

```
wilcox.test(LENGTH~MODE, paired=FALSE, correct=FALSE,
      exact=FALSE)¶
      Wilcoxon rank sum test
data:  LENGTH by MODE
W = 11, p-value = 0.003028
alternative hypothesis: true location shift is not equal to 0
```

The last line can be ignored since it only summarizes which statistical hypothesis was tested, namely that the two distributions are not identical (such that their difference would be 0). All the findings show that the lengths of subordinate clauses are longer but less diverse in writing (cf. the interquartile ranges), and that the difference between the clause lengths in the two modes of 3.5 words is very significant ($p \approx 0.003$).

2.4 Correlations between numeric variables

The final method to be discussed here involves the correlation between two variables that are numeric in nature, such as lengths (of XPs), reaction times (in milliseconds), time (in years), numbers of nodes in a phrase structure tree, etc. By computing a correlation coefficient, which usually falls between −1 and +1, one tries to answer the following questions:

- is there a relationship between a variable *x* and a variable *y* such that, on the whole, one can say 'the more *x*, the more *y*' and/or 'the less *x*, the less *y*' or, on the other hand, 'the more *x*, the less *y*' and/or 'the less *x*, the more *y*'? If the relationship is of the former type, then the correlation coefficient will be >0; if the relationship is of the latter type, then the correlation coefficient will be <0; if there is no relationship between *x* and *y*, the correlation coefficient will be ≈0;
- how strong is this relationship? The more the correlation coefficient differs from 0, the stronger the correlation;
- is the correlation statistically significant?

For example, consider the case where one wants to determine whether the frequencies of two lexical items undergo a temporal trend such that, on the whole, they increase or decrease over time. The two lexical items to be considered are *in* and *just because*, and the corpus to be investigated is Mark Davies's TIME corpus (http://corpus.byu.edu/time), a corpus containing 100 million words of text of American English from 1923 to the present, as found in TIME magazine.

As a first step, the data have to be entered into R, and this is a case where they can be easily entered into R in the vector format with `c`. We create one vector for the time periods (using the decades as reference points),

```
times<-c(1920, 1930, 1940, 1950, 1960, 1970, 1980, 1990,2000)¶
```

and we create one vector for each lexical item that contains their relative frequencies per 10,000 words in the same order as the vector `times` contains the decades. That is, the relative frequency of *in* in the 1920s is $^{188.7}/_{10,000}$, the relative frequency of *in* in the 1930s is $^{174.8}/_{10,000}$, etc.:

```
lex.in<-c(188.7, 174.8, 196.2, 211.1, 221.2, 200.5, 194.3,
      185.8, 192.5)¶
lex.jb<-c(0.005, 0.004, 0.005, 0.004, 0.010, 0.009, 0.013,
      0.029, 0.039)¶
```

Again, it is usually best to first explore the data graphically. If one has two numeric vectors like here, one can use the function `plot` with a formula where again the dependent variable (the frequency of the lexical item) precedes the tilde and the independent variable (time) follows it. In addition, we can provide the argument type, which specifies the type of plot we want: “p” for points only, “l” for lines only, “b” for both, “h” for histograms/bar charts, etc.

```
plot(lex.in~times, type=“b”)¶
```

Especially with data sets larger than the present one, it is often also useful to immediately add a smoother, which is a line that tries to summarize the way the points pattern within the coordinate system. Unlike a linear regression line, which is (too) often used on such occasions, such smoothing lines do not have to be straight but can be curved and are thus often better at identifying curvature and nonlinear trends in the data.

```
lines(lowess(lex.in~times))¶
```

We can then do the same for *just because*, and Figure 19.3 shows slightly prettified versions of the graphs we obtain.

```
plot(lex.jb~times, type="b")¶
lines(lowess(lex.jb~times))¶
```

As with the boxplot, this is another instance where a good graph helps us to analyse our data correctly and already very strongly suggests the outcome of the study. The frequencies of *in* fluctuate across time without a clear pattern such that, for instance, the relative frequency of *in* in the last two decades is approximately the same as that in the first. On the other hand, the frequencies of *just because* exhibit a clear trend such that they clearly increase over time. It is important to note, however, that the growth trend is not linear in the sense that there is at best a very slight increase in the first four decades, then a small but clear increase in the next three decades (from 0.0045 to 0.011 per 10,000 words), followed by a very large increase in the last two decades (from 0.011 to 0.034 per 10,000 words).

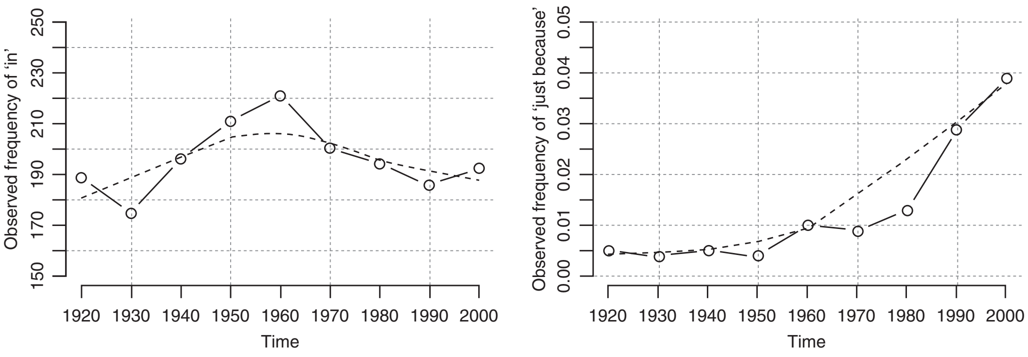


Figure 19.3. *Line plots and smoothers for the normalized frequencies of* in *and* just because *over time*

To determine whether the observed patterns are significantly correlated with time or not, one can compute a correlation coefficient. The probably most frequently used one is Pearson's product-moment correlation *r* (which is related to linear regressions). However, just as a linear regression is often not the best way to inspect data for trends, Pearson's *r* is often not ideal either. This is because Pearson's *r* requires that the vectors/variables that are correlated are interval-scaled, do not contain influential outliers, and are bivariately normally distributed, and linguistic data often violate one or more of these assumptions. It is therefore often better to use a measure that, while a little less powerful, is also less sensitive to potentially problematic distributions. One such measure is Kendall's *tau* τ, which can be computed in R very easily. The necessary function is `cor.test`, which takes three arguments: the two vectors to be correlated and the argument `method`, which is set to "kendall". For the entire time span from 1920 to 2000, this is the result for *in*:

```
cor.test(times, lex.in, method="kendall")¶
        Kendall's rank correlation tau
data:  times and lex.in
T = 18, p-value = 1
alternative hypothesis: true tau is not equal to 0
sample estimates:
tau
  0
```

There is no correlation whatsoever (τ=0) and the result is completely insignificant (p=1). (Note that a restriction of the analysis to the period up to 1960 would yield a high positive correlation, and restriction to the period from 1960 a high negative correlation. Thus, as Figure 19.3 indicates, the result of such analyses always depends on the choice of the period investigated, and it is indispensable to first inspect diachronic data visually before subjecting them to statistical analysis. Hilpert and Gries (2009) discuss techniques such as Variability-based Neighbor Clustering or regression with breakpoints, which can help analysts to discover structure in temporal data.)

The results for *just because*, on the other hand, are very different:[8] there is a high positive correlation (τ=0.743), which is very significant ($p\approx$0.006). The higher the value for time (i.e. the more recent the corpus data), the higher the relative frequency of *just because*, and this correlation is very unlikely to arise just by chance.

```
cor.test(times, lex.jb, method="kendall")¶
       Kendall's rank correlation tau
data:  times and lex.jb
z = 2.7406, p-value = 0.006132
alternative hypothesis: true tau is not equal to 0
sample estimates:
      tau
0.7431605
```

In sum, the overall relative frequency of *in* does not change over time, but the frequency of *just because* does so quite markedly. While the above observations do not exhaust the range of methods that can be applied to the present and similar kinds of data, they already provide a good assessment of whether there is a trend or not, whether it is significant or not, and to some degree at least what its internal structure looks like.

[8] I am omitting a warning about exact p-values and ties here, which informs the user that, because the lex.jb values are not all different from each other, the estimated p-value might not be perfectly accurate; for the present discussion, this is not relevant; cf. Hilpert and Gries (2009) for the exact p-value.

3 Concluding remarks

As mentioned at the outset of this chapter, rigorous quantitative analyses are not yet as frequent in linguistics as they could be, but they are on the rise. This chapter has only discussed a few simple tests, and, of course, linguistic data are often much more complex: For example, this chapter has only dealt with monofactorial tests, i.e. tests involving one independent and one dependent variable. However, more advanced scenarios may involve more independent and more dependent variables. This opens up a whole host of interesting research possibilities, but also requires more sophisticated methods to check one's models. Also, the chapter has not discussed cases where interactions – combinations of independent variables that have unexpected effects on dependent variables – arise and how to deal with these. Finally, the section on correlations has only mentioned correlation coefficients that are typically used for linear trends, but has not dealt with other kinds of regression models.

As the methodological landscape in linguistics is changing, it is important for the progress within our field(s) that we learn how to handle the kinds of complex and multifaceted scenarios linguistic data pose. I hope that this chapter has provided a first overview of what is possible and has whetted the reader's appetite to apply more methods from this exciting domain to linguistic data.

Elementary statistical testing with R

Pros and potentials	Cons and caveats
• statistics package R is freely available • methods allow testing for statistical significance and graphic visualization of distributions • statistical techniques enable users to see contingencies and patterns that remain implicit without them	• quantitative methods and studies must be complemented by qualitative interpretation and validation • statistical comparisons must make linguistic sense • user needs some expertise to determine which tests and graphical displays are appropriate

Further reading

Baayen, R. Harald 2008. *Analyzing linguistic data: a practical introduction to statistics using R*. Cambridge University Press.

Crawley, Michael 2012. *The R book*. 2nd edn. Chichester: John Wiley.

Fox, John 2005. 'The R commander: a basic statistics graphical user interface to R', *Journal of Statistical Software* 14(9): 1–42.

Gries, Stefan Th. 2009. *Quantitative corpus linguistics with R: a practical introduction*. London and New York: Routledge, Taylor & Francis Group.

Gries, Stefan Th. 2013. *Statistics for linguistics with R: a practical introduction*. 2nd edn. Berlin and New York: DeGruyter Mouton.

Hilpert, Martin and Gries, Stefan Th. 2009. 'Assessing frequency changes in multi-stage diachronic corpora: applications for historical corpus linguistics and the study of language acquisition', *Literary and Linguistic Computing* 34(4): 385–401.

Johnson, Keith 2008. *Quantitative methods in linguistics*. Malden, MA and Oxford: Blackwell.

Sheskin, David 2011. *Handbook of parametric and non-parametric statistical procedures*. 5th edn. Boca Raton, FL: Chapman and Hall.

Spector, Phil 2008. *Data manipulation with R*. Berlin and New York: Springer.

Useful online resources

The website of R: www.r-project.org

The CRAN task views: http://cran.r-project.org/web/views

The ling-r-lang-L mailing list: https://mailman.ucsd.edu/mailman/listinfo/ling-r-lang-l

The Statistics for Linguists with R newsgroup: http://groups.google.com/group/statforling-with-r

An electronic textbook for statistics: www.statsoft.com/textbook/stathome.html

Appendix

The R commander

While R does not by default feature a clickable GUI, some applications provide such a GUI. The best-known of these is probably the R commander (cf. Fox 2005). The two figures below illustrate how, once the data have been read into R, the U-test from Section 2.3 is computed with the R commander.

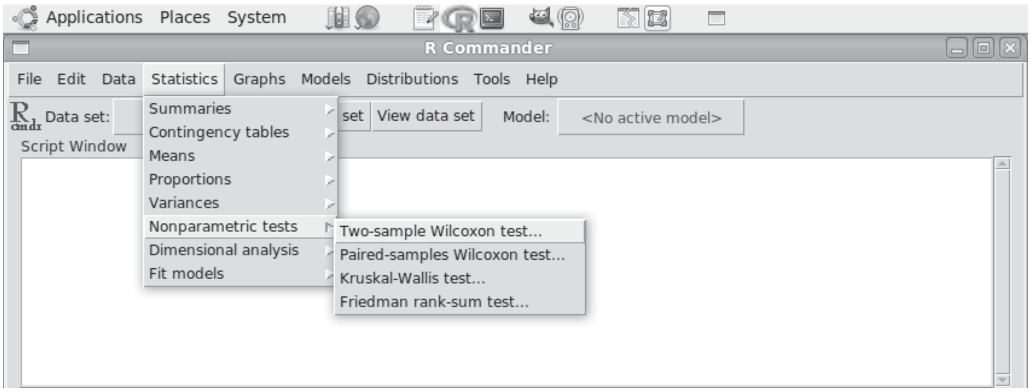
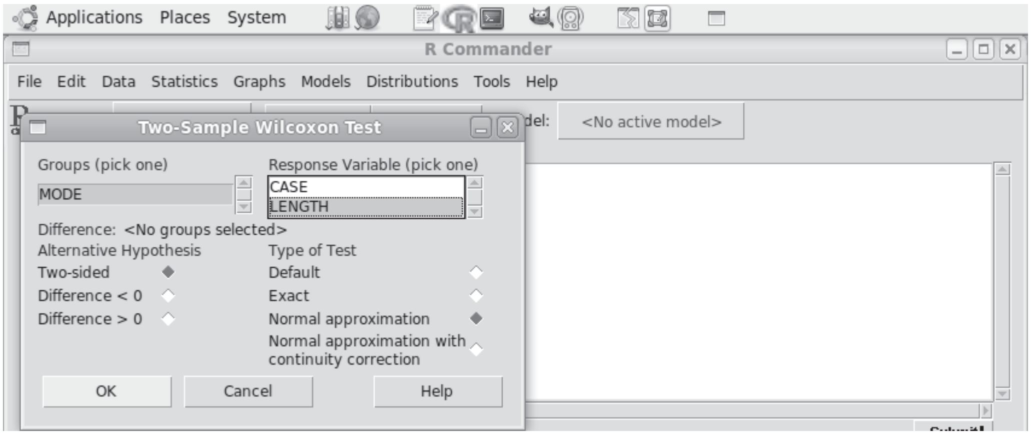


Figure 19.4. *Choosing a* U*-test / two-sample Wilcoxon test in the R commander*


Applications Places System
R Commander
File Edit Data Statistics Graphs Models Distributions Tools Help
<No active model>
OK
Cancel
Help

Two-Sample Wilcoxon Test

Groups (pick one): MODE

Response Variable (pick one): CASE, LENGTH

Difference: <No groups selected>

Alternative Hypothesis		Type of Test	
Two-sided	◆	Default	◇
Difference < 0	◇	Exact	◇
Difference > 0	◇	Normal approximation	◆
		Normal approximation with continuity correction	◇

Figure 19.5. *Performing a* U-*test / two-sample Wilcoxon test in the R commander*

The reader may wonder why not all statistical tests in the present chapter were introduced using the R commander. The main reason is that, while the R commander is without doubt a great tool, it is nevertheless incomplete and limited to what the package's maintainer included in it. Since old functions are improved and new functions/packages are developed all the time, the R commander provides access to 'only' the (admittedly considerable number of) functions included by the maintainer. Furthermore, the R commander is a useful tool in that it always outputs the code that resulted from the user's choices. In my opinion, it is too easy to become overly dependent on it and never get to see the real power that R provides as a programming language. I therefore strongly encourage the reader to nearly always use the command line; in the long run, this policy will pay off.