CHAPTER 7

More on frequencies and reaction times

Linear regression

What you will learn from this chapter

After the previous chapter has introduced some basic elements of regression analysis, this chapter will provide a more thorough discussion of linear regression. This method enables one to model and explain the relationships between one or more explanatory variables at any level of measurement, on the one hand, and one ratio- or interval-scaled response variable, on the other hand. In addition, one can investigate interactions between explanatory variables. You will learn how to fit a multiple linear regression model, to perform its diagnostics and to interpret the results. You will also learn how to carry out non-parametric linear regression with the help of bootstrap. The case study investigates the relationship between reaction times in a lexical decision task, and such factors as word length, corpus frequency and part of speech of lexical stimuli. In contrast with the previous case studies, all these factors are tested here simultaneously in a multiple linear regression model.

7.1 The basic principles of linear regression analysis

Regression analysis is a method that allows one to explain and model the relationship between the response (dependent) variable, on the one hand, and one or more explanatory (independent) variables, on the other hand. When a model contains one explanatory variable, one speaks about **simple regression** analysis. When the number of explanatory variables is two or more, that is a case of **multiple regression**. In linear regression analysis, which will be discussed in this chapter, explanatory variables can be on any scale of measurement, from categorical to ratio-scaled, while the response should be on the interval or ratio scale. For example, one can model the relationship between a test score in a language test, on the one hand, and the method of teaching, number of hours spent on preparation, level of motivation and other possible factors, on the other hand. Multiple regression enables one to measure the individual impact of each variable in the model while controlling for the other variables. It is also possible to model interactions between the explanatory variables.

The previous chapter discussed correlation analysis, which measures the relationship between two quantitative or ordinal variables x and y. If x is regarded as an explanatory variable and y as a response variable, correlation analysis can be thought of as an instance

of simple linear regression. A regression line was added to the scatter plots in order to visualize the relationship between *x* and *y* (see Figure 6.1 in Chapter 6). However, we did not discuss in detail how this line was constructed. In fact, the position and orientation of a regression line can be described by the following formula:

$$\hat{y} = b_0 + bx$$

where

- $\hat{y}$ corresponds to the fitted values of the response variable *y*;
- b_0 is the intercept, i.e. the value of *y* when the line crosses the vertical axis. In other words, this is the predicted value of the response when *x* is equal to zero;
- *b* is the coefficient that determines the slope of the regression line;
- *x* is the explanatory variable;

Consider Figure 7.1 with different regression lines. In the top left graph, the intercept is zero, and the coefficient is 1. Thus, $y = 0 + 1x$, or simply $y = x$. The line crosses the *y*-axis at zero, and with every unit of *x*, *y* increases by one unit, as well. On the top right graph, the intercept is again zero, but the coefficient is 3. Again, the line crosses the *y*-axis at $y = 0$, but this time, *y* increases by three units as *x* increases by one unit. The slope of the regression line is therefore much steeper. On the bottom left plot, the intercept is 2, and the coefficient is 1. This means that the regression line crosses the *y*-axis at $y = 2$, and as *x*

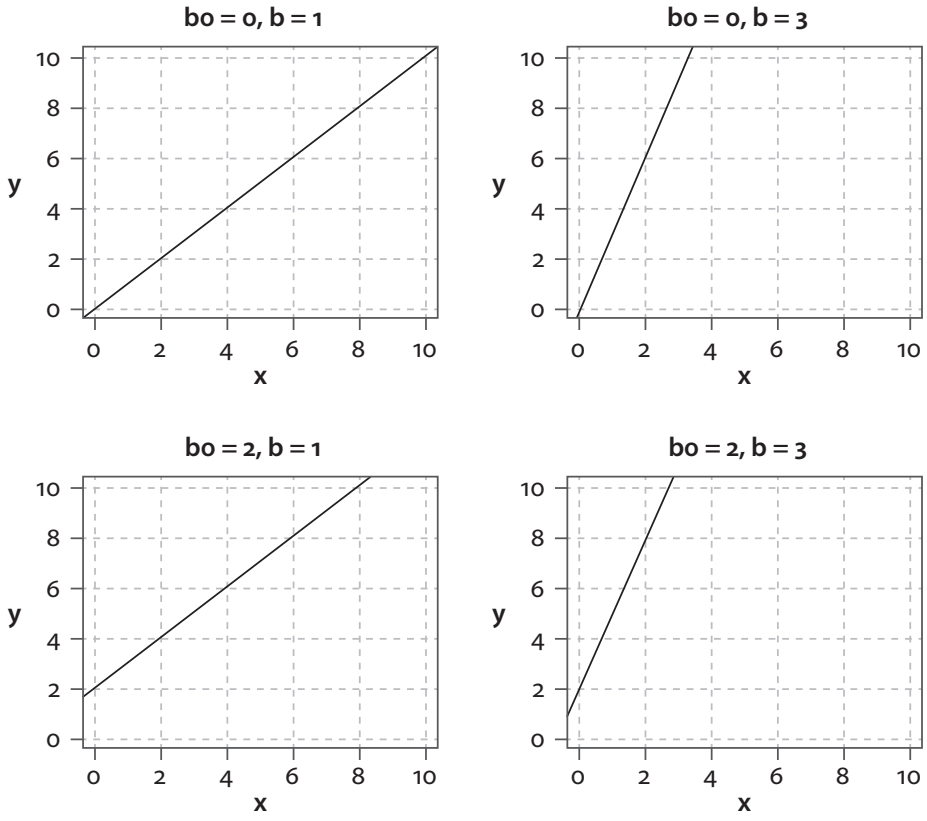


Figure 7.1. Simple regression: lines with different intercepts and slopes

increases by one unit, y increases by one unit, too. Finally, the bottom right plot displays a line with the intercept of 2 and the coefficient of 3. That is, the line crosses the y-axis at 2, and every increase of x by one unit leads to an increase of y by three units.

These plots show the relationships between two variables, an explanatory variable x and a response y. In case of two explanatory variables, one can visualize the relationships with the help of a regression plane in a three-dimensional space, and if the number of explanatory variables is large, one would have to create a hyperplane in a multidimensional space. The principle, however, and the form of the equation remain the same. The only difference is the presence of new terms in the equation:

$$\hat{y} = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \ldots$$

So far, we have focused on the regression line (or hyperplane), which describes the fitted values of the response variable. But this line should be drawn through a cloud of points that represent real observations with actual values of y. In fact, there is one more crucial element that should be added to the equation, the **error**. This is the difference between the fitted and observed values of y. Thus, the actual values of y can be described as follows:

$$y = \hat{y} + \varepsilon$$

or

$$y = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \ldots + \varepsilon$$

where y is the actual value of the response variable, and ε is the error, or residual term. The differences between the fitted and observed values of each individual observation are called **residuals**. This notion was introduced in the previous chapter. An observed value of y for a given observation is therefore the sum of its fitted value, which can be computed from the intercept, coefficients and the values of the explanatory variables, and the residual.

Why bother with multiple regression?

Why fit a multiple regression model instead of fitting a few simple ones? The main reason is that multiple regression allows one to estimate the effect of each individual independent variable in your model **while controlling for the other independent variables**. Imagine that you investigate the relationship between age and gender of young children (explanatory variables) and their vocabulary size (the response). Suppose you have two expectations. First, the vocabulary size grows with age. Second, you expect girls to have

(*Continued*)

a greater vocabulary than boys at the same age. However, if the boys in your sample are on average older that the girls, this may result in a smaller gender difference than there actually is. And the other way round: if the boys in the sample are on average younger than the girls, then the gender difference might be greater than it is in the reality. You need multiple regression to control for such confounding effects.

The aim of regression modelling is to fit the line in such a way that the residuals are as small as possible (this is called the least squares approach). The smaller the sum of squared residuals, the better the model fits the data. The goodness of fit of a linear regression model is usually measured with the help of the R^2 statistic, which ranges from 0 to 1. It is worth mentioning that in usual cases the R^2-value of a simple linear regression model with explanatory variable x and response y is in fact the squared Pearson correlation coefficient r that measures the relationships between x and y.

7.2 Putting several explanatory variables together: Predicting reaction times in a lexical decision task

7.2.1 Data and hypotheses

To reproduce the code in this case study, you will need quite a few add-on packages that should be installed and loaded before you begin.

```
> install.packages(c("leaps", "car", "rms", "visreg", "boot"))
>  library(Rling);  library(leaps);  library(car);  library(rms);
library(visreg); library(boot)
```

The data are available as the dataset `ELP` in `Rling`. This is a small sample of data from the English Lexicon Project (Balota et al. 2007).[1] The dataset contains several variables for 880 randomly selected English nouns, adjectives and verbs. The first factor, *Word*, specifies the lexical stimuli. The integer numeric vector *Length* displays the word lengths in letters. *POS* is the part of speech. It is a categorical variable represented by a factor with three levels: 'JJ' (adjectives), 'NN' (common and proper nouns) and 'VB' (verbs). The numeric vector `SUBTLWF` contains normalized word frequencies in a corpus of film subtitles (per million words). This type of data has been found to be a useful source of lexical norms in recent psycholinguistic studies (Keuleers et al. 2012). Finally, the quantitative variable `Mean_RT` shows the mean reaction times in the lexical decision task (in milliseconds). This variable

1. The experimental data are freely available from http://elexicon.wustl.edu/WordStart.asp (last access 30.06.2014).

will be regarded as the response, and the previous ones (excluding *Word*) will be analysed as the explanatory variables.

```
> data(ELP)
> str(ELP)
'data.frame': 880 obs. of 5 variables:
$ Word: Factor w/ 880 levels "abbreviation",..: 631 747 200 773 821
134 845 140 94 354 …
$ Length: int 7 10 10 8 6 5 5 8 8 6 …
$ SUBTLWF: num 0.96 4.24 0.04 1.49 1.06 3.33 0.1 0.06 0.43 5.41 …
$ POS: Factor w/ 3 levels "JJ","NN","VB": 2 2 3 2 2 2 3 2 2 2 …
$ Mean_RT: num 791 693 960 771 882 …
```

Note that your data for linear regression should have the following format:

- the data are optimally a data frame object in R;
- the rows are individual items or subjects, and the columns are variables;
- the response variable should be a numeric vector;
- the explanatory variables can be numeric vectors (quantitative variables) or factors (categorical variables). The number of factor levels should not be too large: twenty different values are likely to result in data sparseness. If the dataset contains many correlated (associated) variables that present very similar information, try to choose the ones that are the most relevant, or reduce the number of variables by using different exploratory dimensionality reduction techniques, such as Principal Components Analysis (Chapter 18) and Multiple Correspondence Analysis (Chapter 19);
- if you prepare your data in a spreadsheet, make sure that there are no empty cells. It is possible to mark the missing data as 'NA', but you should be aware that the observations with at least one 'NA' value in the response or explanatory variables will be excluded automatically by most algorithms that fit regression models. There exist some solutions to this problem, e.g. one can impute the missing information, but this is beyond the scope of this textbook. For more information, see Harrell (2001: Ch. 3).

Is your sample large enough?

Having too few observations and too many explanatory variables can result in overfitting, which makes your model useless if you try it on a new dataset. However, opinions

(*Continued*)

vary with regard to the exact minimal number of observations. Many statisticians are skeptical about any rules of thumb because the amount of variance in the data, the number of outliers and the size of an effect that one wants to detect can vary greatly. Still, there is a popular formula, according to which the minimum sample size *n* equals 10–15 observations per each regression coefficient that you want to estimate (including the intercept). For example, if we have two quantitative variables and one categorical variable with three levels, we will have five coefficients in the model, including the intercept (see below for an explanation of how categorical explanatory variables can be dealt with in R). Therefore, we will need at least 50–75 observations to fit the model, ignoring possible interactions at the moment. With 880 observations, we are quite safe. Still, it is always useful to check if your model overfits the data with the help of the bootstrap test presented in Section 7.2.8.

As was shown in Chapter 3, corpus frequencies tend to be distributed in a very peculiar way that is known as the Zipfian distribution, which is to say such data contain very few high-frequency items, and very many low-frequency items. Frequency data are often log-transformed to overcome this skewness (see Section 3.3 in Chapter 3 on the effect of a log-transformation of frequencies). Thus, we will use the log-transformed version of the variable `log(SUBTLWF)` in the analyses presented below.

7.2.2 The `lm()` function and interpretation of its output

The main function for linear regression is `lm()`. Let us fit our first model with three explanatory variables. To specify the model, one should use the *formula* interface, where the response variable is on the left side from the tilde sign, and the explanatory variables are on the right. The explanatory variables are separated by the plus sign, which means that we are interested in the additive effect of all these variables on the response variable. To model non-additive effects of interacting variables, one has to make some changes in the formula, as will be shown below. The results can be accessed with the help of the `summary()` function.

```
> m <- lm(Mean_RT ~ Length + log(SUBTLWF) + POS, data = ELP)
> summary(m)

Call:
lm(formula = Mean_RT ~ Length + log(SUBTLWF) + POS, data = ELP)

Residuals:
Min      1Q      Median  3Q      Max
-213.70  -62.55  -9.71   53.87   389.00
```

```
Coefficients:
              Estimate Std.   Error   t value   Pr(>|t|)
(Intercept)   622.466        14.191   43.864    < 2e-16 ***
Length        19.555         1.433   13.645    < 2e-16 ***
log(SUBTLWF)  -29.288        1.784   -16.420   < 2e-16 ***
POSNN         -6.115         8.506   -0.719    0.47238
POSVB         -29.184        10.154  -2.874    0.00415 **
--
Signif. codes: 0 '***'  0.001 '**'   0.01 '*'  0.05 '.' 0.1 ' ' 1

Residual standard error: 93.29 on 875 degrees of freedom
Multiple R-squared: 0.4565,      Adjusted R-squared: 0.454
F-statistic: 183.7 on 4 and 875 DF, p-value: < 2.2e-16
```

The summary contains a lot of information. First, it repeats the formula. The next line shows the distribution of residuals. Ideally, those should be normally distributed and centre around zero. This assumption will be discussed in more detail in Section 7.2.5.

Under the information about the residuals, one can see a table of regression coefficients. The first column shows the estimates, which specify the intercept and the slopes of the regression line. The *p*-values in the rightmost column are based on the *t*-statistics (see the second column from the right) and show how confident we can be that the values are different from zero. A *p*-value less than 0.05 suggests that the null hypothesis of no effect can be rejected. In addition, the table displays the standard errors of estimated coefficients.

The estimated coefficients for quantitative and categorical explanatory variables are displayed differently and should be interpreted differently. The coefficient of *Length* is approximately 19.6. It means that for every additional letter in a lexical stimulus the average reaction time increases by 19.6 milliseconds. Thus, the longer a word, the slower the response. The estimate of *log*(*SUBTLWF*) is approximately –29.3. This means that the average reaction time decreases by 29.3 with every additional unit of *log*(*SUBTLWF*). In other words, the more frequent a word, the faster the response. The categorical variable *POS*, however, is displayed twice, as `POSNN` and `POSVB`. The corresponding estimated coefficients show the change in mean reaction times in comparison with the reference level, *POS* = JJ. Both estimates are negative. Therefore, nouns and verbs are on average recognized faster than adjectives. Note, however, that only the second coefficient ('VB') has the *p*-value smaller than 0.05. This suggests a lack of statistically significant difference between mean reaction times of nouns and adjectives, other factors being controlled for.

Categorical explanatory variables in regression analysis

The default way of treating categorical variables in R is by using the so-called treatment contrasts. That is, the reference level serves as the basis of comparison for all other levels. If an explanatory variable *YourVar* is binary with levels 'A' and 'B', the algorithm will compare 'B' with 'A', i.e. the default reference level (unless you choose 'B' as the reference level). The corresponding estimated coefficient will show by how much the observations with value 'B' differ on average from those with value 'A'. If a categorical variable *X* contains levels 'A', 'B' and 'C', then the algorithm will compare 'B' and 'C' with the reference level 'A'. As a result, you will see only two terms in the model, *YourVar* = 'B' and *YourVar* = 'C', and their coefficients. Four levels will produce three terms, and so on. By default, R chooses the category that comes first alphabetically as the reference level, but you can choose another one by using `factor(YourVar, levels = c("C", "A", "B"))` or `relevel(YourVar, ref = "C")`, as was shown in Chapter 4.

Another popular option is sum contrasts, when the intercept is an unweighted average of the fitted values for all levels of *YourVar* ('A', 'B' and 'C'). The average is called unweighted because it does not take into account that the levels may have different frequencies. In that case, the coefficient of *YourVar* = 'A' shows how much one should add to the intercept to compute the estimated mean value for the observations in the category 'A', and the coefficient of *YourVar* = 'B' shows how much one should add to the intercept to compute the estimated mean value for the category 'B'. The last category, 'C', will be omitted. Its estimated average can be computed by subtracting both coefficients of 'A' and 'B' from the intercept. See a detailed discussion in Gries (2013: Sections 5.2.1 and 5.2.2).

Finally, you may want to consider using Helmert coding for ordinal variables, which compares the second level with the first, the third level with the average of the first and the second levels, and so on.

To use the alternative contrasts in your model, you can do the following (3 is given as an example of the number of levels):

```
# do not run; the code below is provided only as an example
>contrasts(YourVar) <- contr.sum(3) # for sum contrasts
>contrasts(YourVar) <- contr.helmert(3)# for Helmert contrasts
>contrasts(YourVar) <- contr.treatment(3)# to return to treatment
contrasts (the default)
```

As has been explained above, the intercept is the predicted value of *y* at the point where the regression line crosses the *y*-axis. In case of multiple regression, the intercept is the expected value of the response variable for a situation when all quantitative explanatory variables equal zero, and all categorical variables are at their reference levels (provided the default treatment coding is used). In our case, this is the reaction time for a word when its length is zero characters, the log-transformed frequency is zero (this corresponds to the untransformed frequency 1), and the word is an adjective. Although zero-length words do not exist, the algorithm simply extrapolates to the length of zero.

In practice, you will need the information about the intercept only when you want to compute the **fitted** or **predicted values** of the response variable for a specific observation.[2] This is the same as finding the *y*-value of a point on the regression line when you know the *x*-value. Let us illustrate the idea by computing the fitted reaction time for the first word in the list, *rackets*. To do so, one should simply insert the corresponding values of the independent variables to the regression equation and use the estimated coefficients from the `lm()` output above. For categorical variables, one should multiply a regression coefficient given in the table by 1 in case the value applies, or by zero if it does not apply:

$$\begin{aligned}\text{Mean\_RT}_1 &= \text{Intercept} + 7 \times \text{Length} + \log(0.96) \times \log(\text{SUBTLWF}) + 1 \times \text{POSNN} + 0 \times \text{POSVB} \\ &= 622.466 + 7 \times 19.555 + \log(0.96) \times (-29.288) + 1 \times (-6.115) + 0 \times (-29.184) \\ &= 754.43\end{aligned}$$

To compute the fitted values automatically, one can use `fitted(m)`, which will return a vector with fitted values of the response variable. For example, the first word has the following fitted mean reaction time:

```
> fitted(m)[1]
  1
  754.4299
```

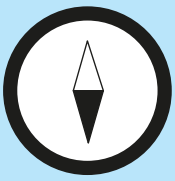
This number coincides with the result of the manual calculations above.

In previous chapters, we discussed the notion of confidence intervals around a statistic. One can obtain the 95% confidence intervals of estimates of regression coefficients as follows:

```
> confint(m) # equivalent to confint(…, level = 0.95)
                2.5 %       97.5 %
(Intercept)    594.61399   650.318207
Length         16.74194    22.367571
log(SUBTLWF)   -32.78872   -25.787038
POSNN          -22.80935   10.579145
POSVB          -49.11280   -9.254989
```

2. There exist models without an intercept, but this topic is beyond the scope of this introductory chapter.

Only one 95% confidence interval, that of `POSNN`, contains the zero. Recall that this regression coefficient also had a *p*-value greater than 0.05. Thus, the confidence intervals corroborate the conclusions made on the basis of the *p*-values from the table of coefficients.



One- and two-tailed tests of regression coefficients

By default, regression analysis involves two-tailed tests. The alternative hypothesis is that the coefficients are different from zero. Whether the difference is positive or negative is not important. However, it is possible to use a one-tailed test if the consequences of missing an effect in the opposite direction are negligible, or if observing the opposite effect is impossible. For example, a pharmaceutical company might want to test if their new drug performs worse than some standard. Obviously, if this is the case and the drug goes into production, the company will face serious legal and economic problems. However, it may be irrelevant whether the new drug performs better than the standard. For example, if the new drug is considerably cheaper, this alone is a strong reason for producing it. In that case, one can simply divide the *p*-value by two to obtain a *p*-value of a one-tailed test. Of course, this method should not be used for '*p*-hacking'.

The next line in the summary shows the standard error of the residuals, i.e. how much variation they display, and the degrees of freedom. These statistics are followed by R^2. As has been mentioned above, this is a standard measure of goodness of fit in linear regression. It shows the proportion of variation in the response variable that is due to variation in the explanatory variables. The coefficient ranges from 0 (a totally useless model; one could achieve the same success by always predicting the mean value of the response) to 1 (perfect fit; each individual value of the response is predicted with exact precision). In our case, the model explains 0.4565, or 45.65% of entire variance. This is not bad for a start, although there is still room for improvement. The adjusted R^2 (0.454) is a useful measure of the model's effectiveness because it penalizes the model for having too many variables that do not make a substantial contribution, as well as for having too few observations. A combination of those two factors is particularly dangerous. The adjusted R^2 provides a necessary correction. Note that the adjusted R^2 can be negative, unlike the original R^2 statistic.

The final line contains the *F*-ratio. It shows if the model is significant in general. In most cases, this means that at least one variable has an estimate that is significantly different from zero. The statistic is the ratio of the variance explained by the model and the residual variance. The *F*-ratio should be at least greater than 1. The *p*-value shows the probability of observing a given *F*-ratio if the null hypothesis were true (that is, the variables had no effect). Since the *p*-value is very low, we can conclude that the model is significant.

To summarize, the model yields interpretable results and has some explanatory power. But is this model good? Can we improve it? The remaining part of this chapter deals with these questions.

7.2.3 Selecting the explanatory variables

If one has more than one explanatory variable, a crucial question is which of them should enter the model, and which should not. According to the principle of parsimony, known as Occam's razor, a simpler model should be preferred to a more complex one if they have the same explanatory power. This is why an optimal model should contain only those variables that contribute substantially to explaining variance in the response variable. There exist different approaches to variable selection. One of them is stepwise selection, which adds or removes explanatory variables iteratively according to some criterion (*p*-values, R^2 scores, AIC values, etc.). Another approach is to try to retain all theoretically relevant factors in the model. Finally, a relatively new approach is the best subsets method. It simply tries all possible models with all variables, and then returns the best one according to some statistical criterion.

We have already fitted a model with all explanatory variables, and it has yielded interpretable results. For the purposes of illustration, however, we will do stepwise selection and use the best subsets method and compare the results. There are three ways of performing stepwise selection: forward, backward and bidirectional. To perform forward selection, one should begin with a null model without any explanatory variables. The program tries to add variables one by one, until no further improvement can be made. To perform backward selection, one begins with a model which contains all explanatory variables, and then the program tries to get rid of all irrelevant ones. Finally, the bidirectional approach begins like the forward approach, but every time a new variable is added to the model, the algorithm also tries to remove one redundant variable.

Let us begin with forward selection. First, we will create a null model with the intercept only by using the `step()` function in the forward direction. The statistical criterion is the Akaike Information Criterion (AIC), which penalizes a model if it has too many variables. It is similar in this respect to the adjusted R^2. Note that the smaller the AIC, the better the fit.

```
> m0 <- lm(Mean_RT ~ 1, data = ELP)
> m.fw <- step(m0, direction = "forward", scope = ~ Length +
log(SUBTLWF) + POS)
[output omitted]

> m.fw

Call:
lm(formula = Mean_RT ~ log(SUBTLWF) + Length + POS, data = ELP)

Coefficients:
(Intercept)  log(SUBTLWF)  Length  POSNN
622.466      -29.288       19.555  -6.115
POSVB
-29.184
```

The algorithm picks all three variables. For comparison, let us run backward stepwise selection. In that case, one starts with a maximal model and then uses the `step()` function in the backward direction.

```
> m.bw <- step(m, direction = "backward")
[output omitted]
> m.bw
Call:
lm(formula = Mean_RT ~ Length + log(SUBTLWF) + POS, data = ELP)

Coefficients:
(Intercept)  Length  log(SUBTLWF)  POSNN
622.466      19.555  -29.288       -6.115
POSVB
-29.184
```

Finally, we will run bidirectional selection. Since `both` is the default option, the argument that specifies the direction can be omitted:

```
> m.both <- step(m0, scope = ~ Length + log(SUBTLWF) + POS) #
equivalent to step(…, direction = "both")
[output omitted]

> m.both

Call:
lm(formula = Mean_RT ~ log(SUBTLWF) + Length + POS, data = ELP)

Coefficients:
(Intercept)  log(SUBTLWF)  Length  POSNN
622.466      -29.288       19.555  -6.115
POSVB
-29.184
```

The results of the forward, backward and bidirectional methods converge. Every explanatory variable in the initial model contributes to explaining variation in the response.

Finally, let us try the best subsets method, which is implemented in the function `regsubsets()` from the package `leaps`.

```
> subsets <- regsubsets(Mean_RT ~ Length + log(SUBTLWF) + POS, data
= ELP, nbest = 4)
```

The function `regubsets()` finds the best subsets of explanatory variables. The argument `nbest = 4` tells the algorithm to return up to four best models of each possible size (from one to four terms). It is the maximal number of estimates in the model, disregarding the intercept (recall that *POS* yields two estimates). One can obtain several goodness-of-fit statistics for each model, including the unadjusted and adjusted R^2:

```
> round(summary(subsets)$rsq, 3)
[1]  0.328  0.278  0.018  0.000  0.450  0.339  0.330  0.286  0.456
[10] 0.451  0.341  0.289  0.456
> round(summary(subsets)$adjr2, 3)
[1]  0.327  0.278  0.017 -0.001  0.449  0.338  0.328 0.284
[9]  0.454  0.449  0.339  0.287  0.454
```

To interpret these results, one needs to know which variables are in which model. One can obtain a table with this information as follows:

```
> summary(subsets)$which
(Intercept)  Length  log(SUBTLWF)  POSNN  POSVB
1  TRUE      FALSE   TRUE          FALSE  FALSE
1  TRUE      TRUE    FALSE         FALSE  FALSE
1  TRUE      FALSE   FALSE         FALSE  TRUE
1  TRUE      FALSE   FALSE         TRUE   FALSE
2  TRUE      TRUE    TRUE          FALSE  FALSE
2  TRUE      FALSE   TRUE          FALSE  TRUE
2  TRUE      FALSE   TRUE          TRUE   FALSE
2  TRUE      TRUE    FALSE         FALSE  TRUE
3  TRUE      TRUE    TRUE          FALSE  TRUE
3  TRUE      TRUE    TRUE          TRUE   FALSE
3  TRUE      FALSE   TRUE          TRUE   TRUE
3  TRUE      TRUE    FALSE         TRUE   TRUE
4  TRUE      TRUE    TRUE          TRUE   TRUE
```

If the value is TRUE, the term is in the model. If the value is FALSE, it is omitted. The last model with all three variables and four terms has the highest scores ($R^2 = 0.456$, adjusted $R^2 = 0.454$), although the ninth model without the term `POSNN` performs equally well. This means that one could conflate *POS* = 'NN' and *POS* = 'Adj' without a loss in the explanatory power of the model. Since there are no obvious theoretical reasons for such conflation, we will keep both levels as they are.

Do not be surprised when you hear different opinions about the optimal strategy of regression modelling. For instance, some people use stepwise variable selection, when one adds new variables to the model sequentially (Hosmer & Lemeshow 2000), whereas others recommend beginning with a model which contains all variables of interest (Harrell 2001). The main problem with stepwise selection is that the usefulness of one variable is assessed on the basis of other variables that have been selected. This may be dangerous when potential explanatory variables are correlated. Field et al. (2012) draw the following humorous analogy between stepwise selection and getting dressed:

> If a stepwise regression method was selecting your clothes, it would decide what clothes you should wear, based on the clothes it has already selected. If, for example, it is a cold day, a stepwise selection method might choose a pair of trousers to put on first. But if you are wearing trousers already, it is difficult to get your underwear on: stepwise methods will decide that underwear does not fit, and you will therefore go without. (Field et al. 2012: 265)

However, if you really need to do stepwise selection, backward selection poses less risk of missing out important variables than forward selection. In any case, one should not rely blindly on automatic approaches. Your choice of variables should be informed by theory and previous research.

Other useful functions that help one estimate whether a variable is useful are `anova()` for model comparison, which will be introduced in the section about testing of interactions, and `drop1()`, which removes one variable from the model at a time and tests the difference. To test explanatory variables in a linear regression model, you should specify that you want to carry out the *F*-test.

```
> drop1(m, test = "F")
Single term deletions

Model:
Mean_RT ~ Length + log(SUBTLWF) + POS
             Df Sum of Sq  RSS      AIC F   value     Pr(>F)
⟨none⟩                     7614993 7987.8
Length       1  1620252    9235246  8155.6  186.1749  < 2e-16 ***
log(SUBTLWF) 1  2346341    9961335  8222.2  269.6061  < 2e-16 ***
POS          2  92194      7707187  7994.4  5.2968    0.00517 **
--
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The results show, not surprisingly, that all variables contribute significantly to the explanatory power of the model. This method is particularly useful when you have categorical variables with more than two values. In that case, it may not be obvious from the table of coefficients and the *p*-values whether the variable has any significant effect on the response, since not all possible pairwise comparisons are shown.

7.2.4 Checking for outliers and overly influential observations

As you already know from Chapter 3, outliers are observations that behave in a different way from the rest of the data. In regression analysis, these are observations with large discrepancies between the observed and fitted values of the response variable. They have unusually large residuals and lie far above or below the regression line. Their presence indicates a lack of fit. Another class of potentially problematic data is so-called influential observations. These points have unusually small or large values on an explanatory variable. They can – but not necessarily do – have a significant effect on the regression slopes, i.e. the estimates of regression coefficients. To identify both types of problematic cases, one can use the function `influencePlot()` in the `car` package. The argument `id.method = "identify"` will start an interactive session. In this session, you can identify the points on the plot by clicking on them. After you have identified all points of interest, press *Esc* or press the *Finish* button (in RStudio).[3] The IDs (the row numbers in our data frame) will be displayed on the plot (see Figure 7.2).

```
> influencePlot(m, id.method = "identify")
```

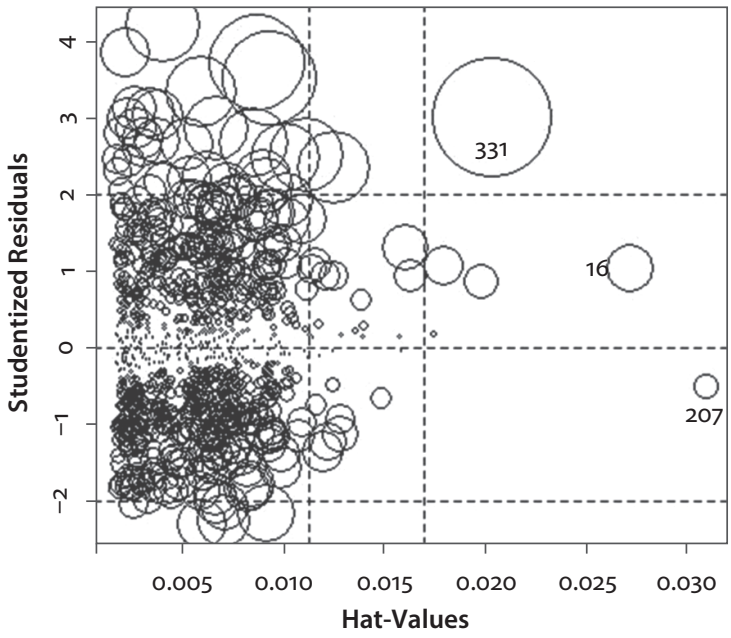


Figure 7.2. Influence plot with several identified points

The bubble plot in Figure 7.2 is based on three values for each observation in the corpus: hat-values, studentized residuals and Cook's distances. The meaning of the values is as follows:

3. There seems to be a problem with identification of points in the current version of the package. If you receive an error message after finishing an interaction session, ignore it.

- Studentized residuals (the *y*-axis) are normalized residuals, adjusted by their expected variability. They represent the discrepancy between the actual and fitted values. It is recommended to check observations with the values greater than 2 or smaller than –2 on the vertical axis.
- Hat-values, or leverage (the *x*-axis), indicate how much influence the observation can potentially have on the fitted values. The influence of a given observation is determined by the difference between the observation's value on an explanatory variable and the mean value of the variable. For leverage, there is no absolute threshold that needs to be controlled. The vertical lines are drawn through the points which correspond to two and three times the average hat-values.
- Cook's distances show the effect of removing an observation on the coefficients and fitted values. They are represented by the size of the bubbles.

As one can see in Figure 7.2, there is one point that has a relatively high leverage, a large residual and a very high Cook's distance score. The ID number of this observation is 331. The word is 'interdepartmental':

```
> ELP[331, ]
Word                    Length  SUBTLWF  POS  Mean_RT
331 interdepartmental   17      0.04     JJ   1324.57
```

This 17-letter word is the second longest stimulus with the longest reaction time in the dataset. If we fit a model without this observation, the coefficients of *Length*, POS = 'NN' and *POS* = 'VB' now have slightly smaller values:

```
> m1 <- lm(Mean_RT ~ Length + log(SUBTLWF) + POS, data = ELP[-331,])
> summary(m1)

Call:
lm(formula = Mean_RT ~ Length + log(SUBTLWF) + POS, data = ELP[-
331, ])

Residuals:
Min      1Q      Median  3Q      Max
-213.32  -63.04  -10.42  53.09   388.42

Coefficients:
              Estimate Std.   Error    t value   Pr(>|t|)
(Intercept)   624.932         14.151   44.163    < 2e-16 ***
Length        19.073          1.436    13.285    < 2e-16 ***
log(SUBTLWF)  -29.288         1.776    -16.495   < 2e-16 ***
POSNN         -4.629          8.482    -0.546    0.58540
POSVB         -27.891         10.117   -2.757    0.00596 **
--
```

```
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Residual standard error: 92.87 on 874 degrees of freedom
Multiple R-squared: 0.4506, Adjusted R-squared: 0.4481
F-statistic: 179.2 on 4 and 874 DF, p-value: < 2.2e-16
```

What should one do in such cases? Outliers and influential observations should be removed or recoded when they arise as a result of coding or sampling errors. If this is not the case and unusual points represent inherent variability in the data, as in this case, opinions differ. Some statisticians consider that removal of 'legitimate' outliers will make the data less representative of the whole population, while others believe it would be best to remove such points for statistical reasons. It is worth mentioning here that today, with the advance of non-parametric methods, such as bootstrap and permutation, the need for 'brushing up' the data in order to meet all assumptions has become less pressing. In case there are legitimate outliers and influential points it is worthwhile to perform non-parametric regression with bootstrap, which will be discussed in Section 7.2.8.

7.2.5 Checking the regression assumptions

Similar to all previously discussed tests, linear regression has a number of assumptions that should be met if one wants to obtain reliable and meaningful results:

Assumption 1. *The observations are independent from one another.*
Assumption 2. *The response variable is at least interval-scaled.*
Assumption 3. *The relationship between the dependent and independent variables is linear.* If not, a power transformation of the response or explanatory variables should be applied.
Assumption 4. *The errors vary constantly.* This is also called homoscedasticity of variance. In other words, the variability of the residuals does not increase or decrease with the explanatory variables or the response.
Assumption 5. *There is no strong linear dependence between explanatory variables.* Such dependence is also known as **multicollinearity**.
Assumption 6. *The residuals are not autocorrelated.*
Assumption 7. *The residuals are normally distributed, with a mean of 0.* This assumption becomes less important when the sample is large.

Let us examine whether our model meets these assumptions one by one.

Assumption 1. *The observations are independent from one another.*
This means that each value of the response variable should be independent from the values of other observations. In our case, this means that the reaction time needed for recognition of one lexical stimulus should not depend on the reaction time of another word in the dataset. This might happen, for example, if the data contained only words with the same small set of roots. This is clearly not the case. If this assumption is violated, one should use a mixed model with fixed and random effects (see Chapter 8).

Assumption 2. *The response variable is measured at least on the interval scale.*
The reaction time is a ratio-scaled variable. Although it would be unusual to observe reaction times of zero milliseconds, this number is still meaningful. One can also apply multiplication and say that 500 ms is twice as fast as 1000 ms.

Assumption 3. *The relationship between the dependent and independent quantitative variables is linear.*
The assumption of linearity was mentioned in Chapter 6, when the Pearson correlation coefficient was discussed. To detect non-linearity in a simple regression model with one explanatory variable *x*, a simple scatter plot of *x* and *y* should suffice (see Chapter 6). A convenient tool for detection of non-linearity in multiple regression analysis is the so-called component-residual, or partial-residual plot. It is available as `crPlot()` in the package `car`. The model contains two quantitative explanatory variables, *Length* and *log*(*SUBTLWF*). We will create two plots with these two variables side by side.

```
> par(mfrow = c(1, 2))
> crPlot(m, var = "Length")
> crPlot(m, var = "log(SUBTLWF)")
> par(mfrow = c(1, 1))
```

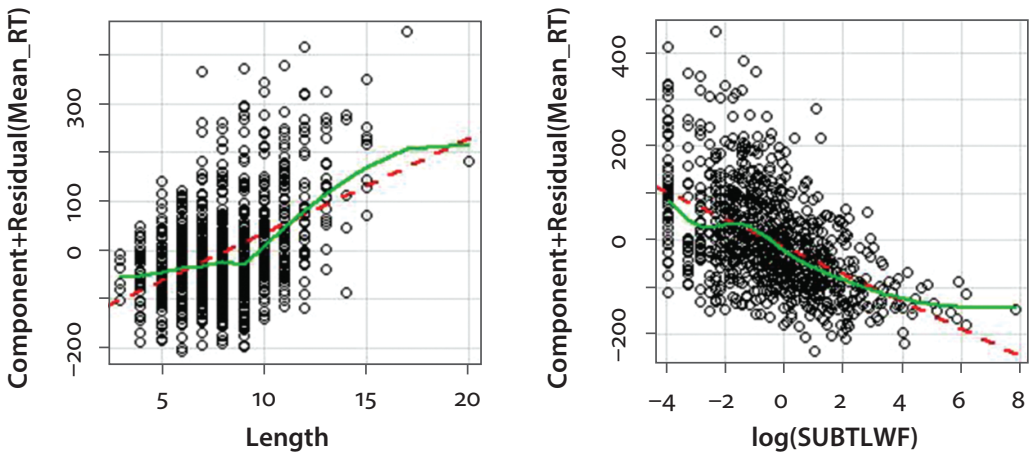


Figure 7.3. Component-residual plot with *Length* (left) and *log*(*SUBTLWF*) (right)

Both plots are displayed in Figure 7.3. The component-residual plot shows how the residuals plus the corresponding regression coefficient (the vertical axis) change depending on an explanatory variable (the horizontal axis), after taking into account the effect of all other explanatory variables. The dashed red lines represent the slopes of the explanatory variables based on the regression estimates, and the solid green lines reflect the main tendency in the cloud of data points. The solid lines deviate from the dashed lines, indicating some non-linearity. There is, however, a more serious problem, which is evident in the right-hand plot. Namely, the variance is not homogeneous: the dispersion of points seems to be decreasing with *x*. That brings us to the next assumption.

Assumption 4. *The errors vary constantly. There is no heteroscedasticity.*
Visual diagnostics of homoscedasticity is possible with the help of the plot of the residuals against fitted values. It is easily performed with the help of `plot.lm()`, which can be simply invoked by `plot()` when the first argument is an `lm` object. This function offers in fact several different plots, which can be selected by using `which`.

```
> plot(m, which = 1)
```

The result is displayed in Figure 7.4. The plot indicates that the greatest spread of residuals is observed where the fitted values ranges approximately from 800 to 950 milliseconds, and the smallest variability is when the words are recognized very quickly, with the predicted mean reaction times from 400 to 600 milliseconds. The pattern is not homoscedastic.

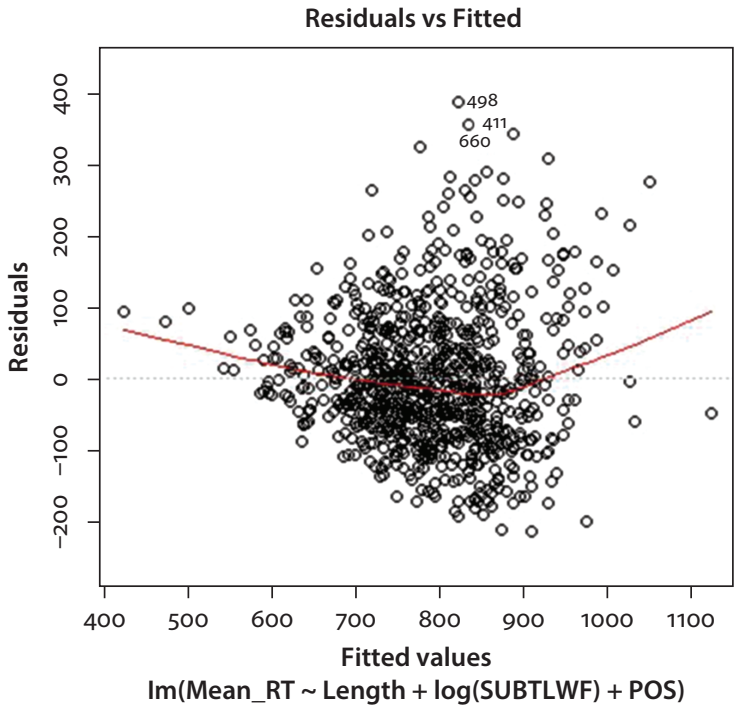


Figure 7.4. Residuals vs. fitted plot

One can also perform the non-constant variance test from the package `car`:

```
> ncvTest(m)
Non-constant Variance Score Test
Variance formula: ~ fitted.values
Chisquare = 79.67363       Df = 1 p = 4.41657e-19
```

This test has the null hypothesis that the error has constant variance with the response (fitted values). The *p*-value smaller than 0.05 tells us that we can reject the null hypothesis of constant error variance. All this signals a problem.

It is also necessary to test whether the error has constant variance depending on the values of quantitative explanatory variables. To test that, one can use the component-residual plots shown above, as well as the non-constant variance test, where you should specify the name of the explanatory variable:

```
> ncvTest(m, ~ Length)
Non-constant Variance Score Test
Variance formula: ~ Length
Chisquare = 42.39826       Df = 1 p = 7.445615e-11
> ncvTest(m, ~ log(SUBTLWF))
Non-constant Variance Score Test
Variance formula: ~ log(SUBTLWF)
Chisquare = 59.12633 Df = 1       p = 1.478672e-14
```

The small *p*-values in both tests are a clear indication of heteroscedasticity in the residuals, depending on the values of both variables.

In such situations, one can try power transformations of the response variable. Some popular power transformations, such as squaring (raising to the power of 2) or unsquaring (raising to the power of 0.5), were discussed in Section 3.3 of Chapter 3. Power transformations are often done by trial and error, and require some experience. Alternatively, one can find the optimal transformation with the help of the Box-Cox plot, which is based on the approach developed by George Box and Sir David Cox and implemented as `boxCox()` in the package `car`. The resulting plot is shown in Figure 7.5.

```
> boxCox(m)
```

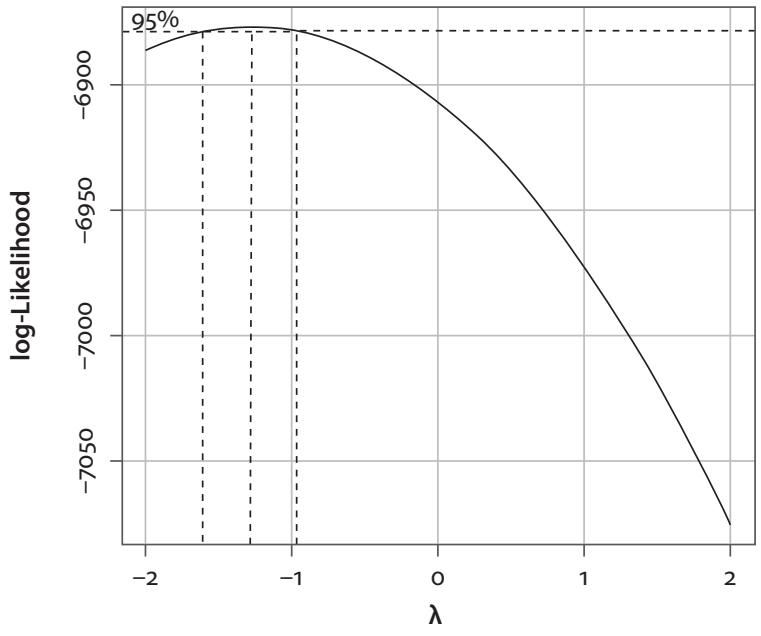


Figure 7.5. A Box–Cox plot

The plot displays two values. The horizontal axis represents the transformation power of the response variable. Note that 0 does not correspond to raising a number to the power

of 0 (which will result in 1 for any number). Instead, it means taking the natural logarithm. The vertical axis represents the log-likelihood of the corresponding power transformation. The optimal transformation is the one with the highest y-value. The dashed lines mark the 95% confidence region. If necessary, the range of possible transformations can be extended, for example, from –3 to 3:

```
> boxCox(m, lambda = seq(-3, 3, 1/10)) # output not shown
```

Figure 7.5 shows that the optimal transformation is approximately $y^{-1.3}$. It is equal to 1 divided by $y^{1.3}$.[4]

```
> m.trans <- lm(Mean_RT^-1.3 ~ Length + log(SUBTLWF) + POS, data =
ELP)
> ncvTest(m.trans)
Non-constant Variance Score Test
Variance formula: ~ fitted.values
Chisquare = 0.9711398      Df = 1 p = 0.3243961
```

The high p-value allows us to believe that the problem of heteroscedastic error variance is resolved. The readers are encouraged to repeat the tests with the two quantitative explanatory variables and see that there is no evidence of heteroscedasticity in those cases, as well. One can also re-create the component-residual plots and the residuals vs. fitted values plot to see that adding the transformation improves the model.

As was mentioned in Section 3.3 of Chapter 3, transformation gives neater data at the cost of interpretability of results. An alternative is to use non-parametric regression, which is described in Section 7.2.8.

Assumption 5. *There is no multicollinearity.*

Multicollinearity is a phenomenon that can be observed when some variables relate to the same underlying causal effect. Imagine that you want to predict whether a postgraduate student will obtain a Ph.D. or not on the basis of such factors as the university grades, the number of books read per month and the difficulty level of the courses that he or she has taken at college. All these variables might reflect one underlying factor, namely, learning motivation. When combined in one model, very strongly correlated variables tend to have unstable estimates and large standard errors. This means that the estimates of correlated variables are no longer reliable, even if the model has sufficient predictive power (i.e. a high R^2). Multicollinearity can be detected with the help of `vif()` in the `car` package.[5]

---

4. When applying transformations to predictors, you should use the function *I()*, which tells R to interpret the operator in its arithmetical sense, rather as a regression formula expression. An example is $y \sim I(x^2)$. However, expressions like $y \sim log(x)$ or $y \sim sqrt(x)$ are perfectly legal.

5. A function under the same name can be found in the `rms` package (see an example in Chapter 12). It does not matter which one you use. If two functions from different loaded add-on packages

The function returns the Variance Inflation Factors (VIF-scores), which help us estimate how much collinearity is associated with each regression term. The generalized VIF-scores are shown in the first column, whereas the third column displays the same scores corrected for the degrees of freedom.

```
> car::vif(m.trans)
               GVIF      Df  GVIF^(1/(2*Df))
Length         1.151054  1   1.072872
log(SUBTLWF)   1.150140  1   1.072446
POS            1.026925  2   1.006664
```

There exist various rules of thumb, some of them stricter (VIF-scores should not exceed 5) and others less so (VIF-scores should not be greater than 10). In our case, both uncorrected and corrected VIF-scores are very small. Thus, multicollinearity is not a matter of concern.

To illustrate the effects of multicollinearity, let us create a vector of word length that is identical to *Length*, with the exception of the first three values, which will be changed to 8 letters with the help of the function `rep()`, i.e. repeat.

```
> Length1 <- ELP$Length
> Length1[1:3] <- rep(8, 3)
> head(ELP$Length)
[1] 7 10 10 8 6 5
> head(Length1)
[1] 8 8 8 8 6 5
```

The vectors differ only in the first three values. The remaining scores are absolutely identical. Let us now add the new term *Length1* to the model:

```
> m.test <- lm(Mean_RT^-1.3 ~ Length + log(SUBTLWF) + POS + Length1,
data = ELP)
```

```
> car:: vif(m.test)
               GVIF          Df  GVIF^(1/(2*Df))
Length         543.443758    1   23.311880
log(SUBTLWF)   1.150140      1   1.072446
POS            1.028782      2   1.007119
Length1        543.518917    1   23.313492
```

The VIF-scores of both *Length* and *Length1* are huge. This is an indication that something is really wrong. Let us examine the table of coefficients of the new model:

---

have the same name, e.g. `vif()` in `car` and `rms`, one of them will be masked, and R will give a warning message. To use a masked function, you should specify the package where it comes from, e.g. `car::vif(m)`.

```
> summary(m.test)

Call:
lm(formula = Mean_RT^-1.3 ~ Length + log(SUBTLWF) + POS + Length1,
data = ELP)

Residuals:
Min          1Q          Median      3Q          Max
-7.313e-05  -1.681e-05  -2.530e-07  1.646e-05  6.679e-05

Coefficients:
             Estimate     Std. Error  t value   Pr(>|t|)
(Intercept)  2.200e-04    3.724e-06   59.081    < 2e-16 ***
Length       -3.031e-06   8.166e-06   -0.371    0.71066
log(SUBTLWF) 9.059e-06    4.678e-07   19.367    < 2e-16 ***
POSNN        1.017e-06    2.231e-06   0.456     0.64871
POSVB        7.614e-06    2.665e-06   2.857     0.00437 **
Length1      -1.835e-06   8.173e-06   -0.225    0.82239
--

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.446e-05 on 874 degrees of freedom
Multiple R-squared: 0.4951, Adjusted R-squared: 0.4922
F-statistic: 171.4 on 5 and 874 DF, p-value: < 2.2e-16
```

The table of coefficients shows that the *p*-values of the estimates of *Length* and *Length1* are very large. The estimates of the correlated variables are not reliable any more. In situations of multicollinearity, one can either try to get rid of one of the correlated variables, or, if all correlated variables are equally theoretically important, try to use a dimensionality-reduction technique, such as Principal Components Analysis (for quantitative variables) or Multiple Correspondence Analysis (for categorical variables). See more information in Chapters 18 and 19, respectively.

Assumption 6. *There should be no autocorrelation between the residuals.*
As was discussed in the previous chapter, autocorrelated residuals tend to emerge when we have time series. We would not expect them here. The test is performed solely for illustration:

```
> durbinWatsonTest(m.trans)
lag  Autocorrelation  D-W Statistic  p-value
1    -0.001031322     2.000717       0.932
Alternative  hypothesis: rho != 0
```

Since the *p*-value is greater than 0.05, and the *D–W* statistic is very close to 2, we can conclude that there is no autocorrelation.

Assumption 7. *The residuals should be normally distributed, with the mean of 0.*
One can use the Shapiro-Wilk normality test to check is there are any violations of normality:

```
> shapiro.test(residuals(m.trans))
  Shapiro-Wilk normality test
data: residuals(m.trans)
W = 0.9983, p-value = 0.5434
```

Since the *p*-value is greater than 0.05, the null hypothesis of normality cannot be rejected. One can also inspect a Q–Q plot shown in Figure 7.6, which is one of the plots produced by `plot.lm()`. The Q–Q plot (`which = 2`) does not display deviations from non-normality.

```
> plot(m.trans, which = 2)
```

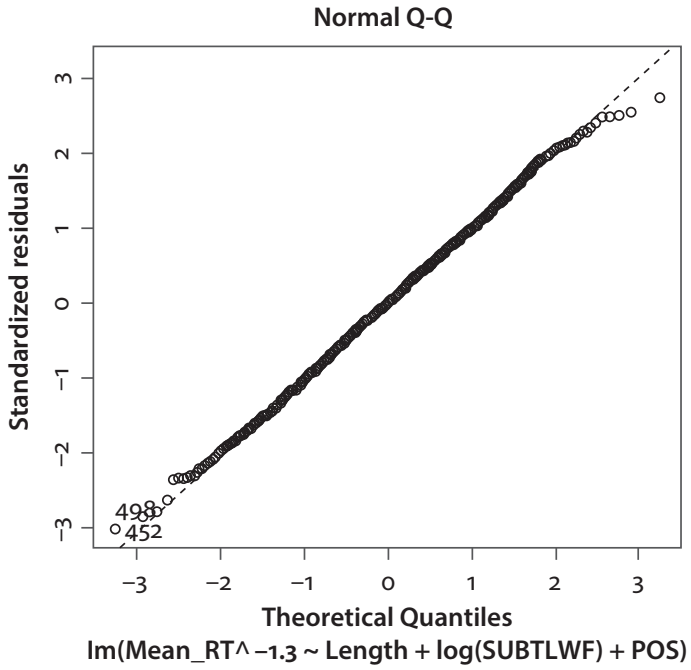


Figure 7.6. Q–Q plot of residuals

7.2.6 Testing and interpreting interactions

Interactions are observed when the effect of one explanatory variable on the outcome depends on the value of another variable. The effect can be reinforced, weakened, or even reversed. For instance, consider caviar and vanilla ice-cream. As separate courses, they are delicious, but to put them together would be a culinary blunder. One can also say that the joint effect is not additive.

R supports two ways of designating an interaction between two or more variables, which are equivalent:

(1) `Response ~ Var1*Var2`

(2) `Response ~ Var1 + Var2 + Var1:Var2`

Let us test whether the effect of word length varies for different parts of speech. To do so, we will fit a model with an interaction between *Length* and *POS*, using the first type of notation:

```
> m.int <- lm(Mean_RT^-1.3 ~ Length*POS + log(SUBTLWF), data = ELP)
```

One can use ANOVA (Analysis of Variance) to compare two models, one with main effects only, and the other with the interaction term:

```
> anova(m.trans, m.int) # equivalent to anova(…, test = "F")
Analysis of Variance Table
Model 1: Mean_RT^-1.3 ~ Length + log(SUBTLWF) + POS
Model 2: Mean_RT^-1.3 ~ Length *POS + log(SUBTLWF)
Res.Df  RSS Df  Sum of Sq  F Pr(>F)
1  875  5.2314e-07
2  873  5.1931e-07 2 3.8336e-09 3.2223 0.04034 *
--
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

More on different kinds of ANOVA will follow in Chapter 8, but this particular use of ANOVA allows one to test the null hypothesis that the models are not significantly different. Since the *p*-value smaller than 0.05, the interaction term is significant. Note that the second model should contain all explanatory variables in the first model.

How can this interaction be interpreted? The best option is to make a visual representation. The package `visreg` offers many useful options for visualization of interactions. To explore the interaction between *Length* and *POS*, one can use the following code:

```
> visreg(m.int, xvar = "Length", by = "POS")
```

where the first argument is the model that contains the interaction, `xvar` specifies the explanatory variable that will be represented by the *x*-axis of the interaction plot, and `by` introduces the variable that will be used to divide the plot into cross-sections. The resulting plot is shown in Figure 7.7. It is crucial that the lines are not exactly parallel. The length of nouns has a smaller effect on the reaction times than the length of verbs and adjectives. Although short verbs and adjectives have a slightly higher *y*-value than nouns of the same length, very long verbs and adjectives have a lower *y*-value than very long nouns. Note that the transformed *Mean_RT* is close to the inverse of the original scores. That is, the smaller the original reaction times, the greater the transformed ones, and vice versa. Thus, the greater the *y*-value on the plot, the faster the word is recognized.

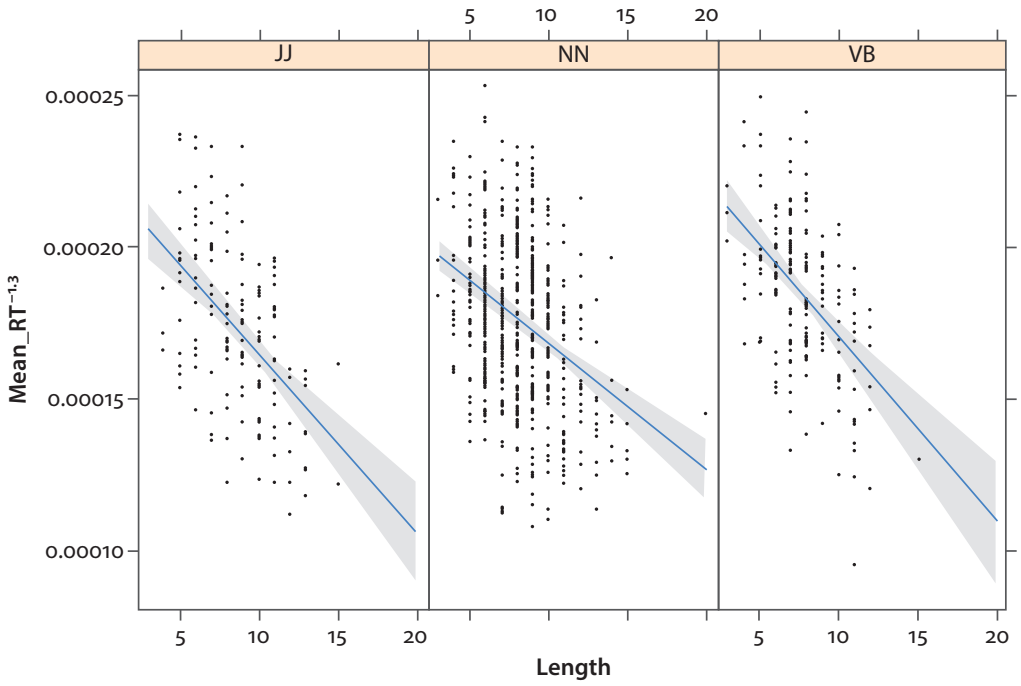


Figure 7.7. Interaction of *POS* and *Length*

The interaction is mild. For all parts of speech, the effect of word length is similar. To show what a very strong interaction may look like, consider Figure 7.8. It shows two interacting explanatory variables: a binary variable with values 'A' and 'B' and *numVar*, a numerical variable ranging from 1 to 50. The dependent variable is called *response*. The interaction is very strong because the effect of *numVar* is opposite for different levels of the nominal variable, 'A' and 'B'. As *numVar* increases, the response variable increases when the binary variable has value 'A', and decreases with it has value 'B'. Such interactions are called cross-over.

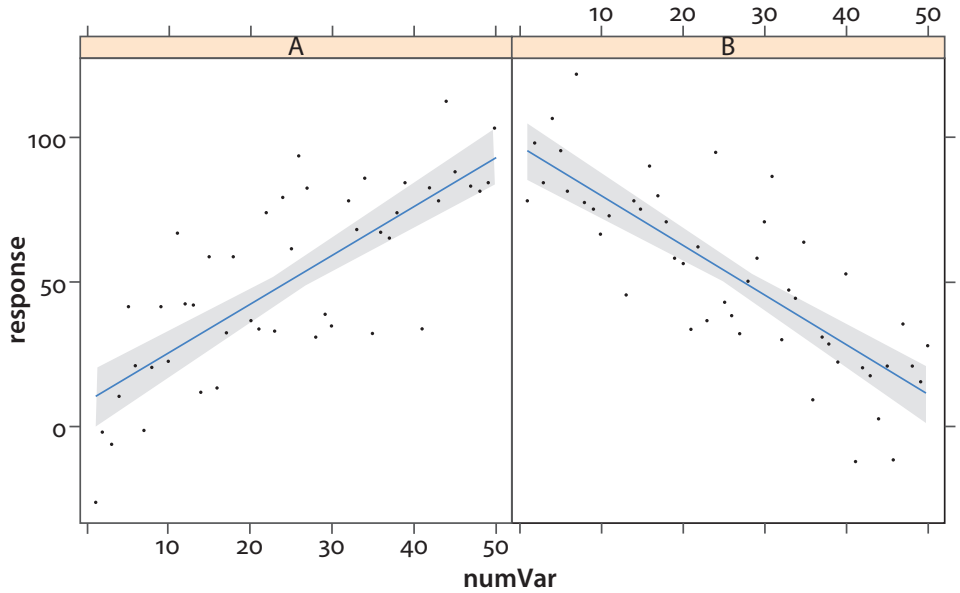


Figure 7.8. Interaction plot of a binary and continuous independent variables, which displays a cross-over interaction

If you have a look at the summary of the model with the interaction, you will find two new terms in the table of coefficients: `Length:POSNN` and `Length:POSVB`.

```
> summary(m.int)
Call:
lm(formula = Mean_RT^-1.3 ~ Length + log(SUBTLWF) + POS +
  Length:POS, data = ELP)
Residuals:
Min          1Q          Median       3Q          Max
-6.901e-05   -1.643e-05   -1.510e-07   1.650e-05   6.835e-05

Coefficients:
               Estimate Std.  Error       t value  Pr(>|t|)
(Intercept)    2.285e-04      7.046e-06   32.423   < 2e-16 ***
Length         -5.825e-06     7.770e-07   -7.497   1.6e-13 ***
log(SUBTLWF)   9.061e-06      4.665e-07   19.424   < 2e-16 ***
POSNN          -1.324e-05     8.014e-06   -1.652   0.0989.
POSVB          9.297e-06      9.758e-06   0.953    0.3410
Length:POSNN   1.670e-06      8.909e-07   1.875    0.0612.
Length:POSVB   -3.348e-07     1.136e-06   -0.295   0.7683
--

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.439e-05 on 873 degrees of freedom
Multiple R-squared: 0.4987,    Adjusted R-squared: 0.4953
F-statistic: 144.8 on 6 and 873 DF, p-value: < 2.2e-16
```

It is important to understand that the interpretation of all terms involved in the interaction is now different. The coefficient of *Length* does not correspond to the effect of length for all data, but only for the observations with the reference level of the interacting term *POS* = 'JJ'. The coefficient of `POSNN` is now the effect of *Length* for *POS* = 'NN' for words with the length of 0 ms, which exists only hypothetically. The same holds for `POSVB`. The interaction term `Length:POSNN` shows the difference between the change in the transformed reaction times of nouns and the change in the transformed reaction times of the reference level, adjectives, for every additional unit of word length. For `Length:POSVB` the interpretation is analogous.

More about interactions

Examples of interactions between categorical variables will be discussed in the next chapter. See also Chapter 12. If you have multiple or high-order interactions, you might

(*Continued*)

wish to use additional techniques to disentangle such complex relationships, e.g. conditional inference trees and random forests (see Chapter 14).

7.2.7 Checking for overfitting

Overfitting is a serious problem in regression modelling. If you take another sample, an overfitted model will perform poorly on the new data. Such models have little, if any, scientific value. Overfitting usually occurs in situations 'small *n*, large *p*', where *n* is the number of cases, and *p* is the number of explanatory variables. Ideally, every fitted model should be tested on new data (this is called cross-validation). However, this is not always possible because new data are costly. Instead, one can use resampling methods, such as bootstrapping. In a nutshell, bootstrapping means that we use one sample to both carry out an analysis and validate its results, as if one tried to pull him- or herself up by bootstraps, similar to Baron Munchhausen, who pulled himself out of a swamp by his pigtail. Although such deeds may sound like impossible tasks, bootstrapping is a very popular procedure in statistics. Its basic principle is simple: a new dataset is sampled from the original data randomly with replacement (that is, the same observation can be used more than once), and then the statistical analysis is performed again. One usually logs some statistics of how similar the new result is to the original one. When one repeats the procedure many times, one can obtain reliable estimates of the accuracy of a model.

To perform bootstrap validation, we will use the `validate()` function in the `rms` package. To be able to use this function, one has to refit the model first with the help of the `ols()` function, which is equivalent to `lm()`. It is also necessary to add `x = TRUE` and `y = TRUE`.

```
> m.val <- ols(Mean_RT^-1.3 ~ Length*POS + log(SUBTLWF), data = ELP,
x = TRUE, y = TRUE)
> validate(m.val, bw = TRUE, B = 200)
             Backwards Step-down - Original Model

No Factors Deleted

Factors in Final Model

[1] Length    POS     SUBTLWF       Length *POS
Loading required package: tcltk
           index.orig  training  test    optimism  index.corrected  n
R-square   0.4987      0.4996    0.4943  0.0054    0.4934           200
MSE        0.0000      0.0000    0.0000  0.0000    0.0000           200
g          0.0000      0.0000    0.0000  0.0000    0.0000           200
Intercept  0.0000      0.0000    0.0000  0.0000    0.0000           200
Slope      1.0000      1.0000    0.9966  0.0034    0.9966           200
```

```
Factors Retained in Backwards Elimination

Length   POS   SUBTLWF   Length * POS
*        *     *         *
*        *     *         *
*        *     *         *
[output omitted]

Frequencies of Numbers of Factors Retained

2  3   4
4  49  147
```

The algorithm does a resampling validation of the regression model. The procedure is repeated many times. In our case, `B = 200` tells R to make 200 bootstrap runs. The argument `bw = TRUE` chooses backward stepwise variable elimination. The output shows how many times the variables were retained in the model. In our case, all variables were retained in 147 resamplings from 200. Note that your results will be slightly different every time you run the procedure.

However, the most important information about potential overfitting is contained in the column named `Optimism`. It displays the differences between the training and test statistics shown in the corresponding columns. The training statistics are averaged over all regression models based on bootstrapped data, whereas the test statistics are obtained when the bootstrap models are used to predict the transformed reaction times on the full dataset. See Baayen (2008: 212–214) for more details. Large optimism values indicate overfitting. As a rule of thumb, the slope optimism (i.e. the optimism in the estimation of the regression coefficients of the explanatory variables) should not exceed 0.05. In case of overfitting, a possible solution is adding a penalty to the model, which will make the estimates shrink (see Baayen 2008: 224–227). Overfitting may also result from ignoring dependence between observations, such as individual variation between subjects. In this case, fitting a mixed model with fixed and random effects may help. This method will be briefly introduced in the next chapter. In our example, the optimism in the slopes is less than 0.01, and the optimism in R^2 is around 1%: 0.0054/0.4987 = 0.01. Thus, there are no signs of overfitting.

7.2.8 Non-parametric regression: Bootstrap

When one or more assumptions of linear regression have been violated, one can use regression based on bootstrapping, which was introduced in the previous subsection. Its advantage is that it does not require all those assumptions to be met in order to return results that can be trusted. We have already used bootstrapping to check whether there is evidence that the model overfits the data. In this section, you will learn how to obtain non-parametric confidence intervals with the help of bootstrap to see whether the explanatory variables will have similar effect sizes if you take another sample. The procedure we will follow is described in Field et al. (2012: 299–300). We will use the function `boot()` from the package `boot`. The first step is to create a function that will return regression coefficients after each bootstrap:

```
> bootcoef <- function(formula, data, indices)
{
   d <- data[indices,]
   model <- lm(formula, data=d)
   return(coef(model))
}
```

Next, we will test the model where the assumption of homoscedasticity was violated. Recall that we had to use a power transformation of the response variable. This improved the model, but the results are much more difficult to interpret now. For better comparability with the model discussed in detail previously, we will perform the bootstrap procedure for the model without the interaction. The bootstrap will involve 2000 resamplings.

```
> m.boot <- boot(formula = Mean_RT ~ Length + log(SUBTLWF) + POS,
data = ELP, statistic = bootcoef, R = 2000)
```

Now we can compute the 95% confidence intervals for each of our estimates. The confidence intervals for the intercept, *Length*, *POS*, *log*(*SUBTLWF*), *POS* = 'NN' and *POS* = 'VB' can be retrieved as follows:

```
> boot.ci(m.boot, type = "bca", index = 1) # Intercept
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 2000 bootstrap replicates

CALL:
boot.ci(boot.out = m.boot, type = "bca", index = 1)

Intervals:
Level  BCa
95%    (596.1, 649.3)
Calculations and Intervals on Original Scale

> boot.ci(m.boot, type = "bca", index = 2) # Length
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 2000 bootstrap replicates

CALL:
boot.ci(boot.out = m.boot, type = "bca", index = 2)

Intervals:
Level  BCa
95%    (16.72, 22.58)
Calculations and Intervals on Original Scale
```

The reader is encouraged to repeat the procedure for the three remaining terms. Note that the results will be slightly different every time you run the procedure. The output is quite

verbose, but the confidence intervals can be easily found on the second line from the bottom, in parentheses. The confidence intervals are very similar to the confidence intervals that were computed in Section 7.2.2.

The bootstrap method also allows for computation of the 95% confidence interval of any other statistic in the model. For instance, one can compute the interval of the model's R^2 as follows:

```
> bootR2 <- function(formula, data, indices)
{
  d <- data[indices,]
  model <- lm(formula, data=d)
  return(summary(model)$r.squared)
}
```

```
> m.boot.R2 <- boot(formula = Mean_RT ~ Length + log(SUBTLWF) + POS,
data = ELP, statistic = bootR2,  R = 2000)
> boot.ci(m.boot.R2, type = "bca")
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 2000 bootstrap replicates

CALL:
boot.ci(boot.out = m.boot.R2, type = "bca")

Intervals:
Level  BCa
95%    (0.4096, 0.5033)
Calculations and Intervals on Original Scale
```

The original statistic lies in the middle of the confidence interval based on the bootstrap. Therefore, the results of our initial model are stable and reliable.

7.3 Summary

In this chapter we have discussed linear regression, including modelling strategies, diagnostics, assumptions, interactions and non-parametric tests with the help of bootstrapping. We have found significant effects of frequency, word length and part of speech on the time needed by speakers to recognize a word. Of course, many other factors may be relevant, as well (cf. Balota et al. 2007), although even the current small model has a reasonable predictive power. There is one caveat, however. The model, which is based on average reaction times, does not take into account individual variation between the subjects, which may have considerable effects on the results. See Chapter 8, Section 8.3 for more information about mixed Generalized Linear Models.

How to report your linear regression model

There are no strict rules for reporting the results, but you should provide the following information: (a) the table with estimated regression coefficients (including the intercept), their standard errors and p-values; (b) the goodness-of-fit statistic (R^2). Additionally, you can also report the 95% confidence intervals for your estimates. You can also use the asterisk system to indicate the significance of coefficients in the table (*** for $p < 0.001$, ** for $0.001 \leq p < 0.01$, * for $0.01 \leq p < 0.05$, "." for $0.5 \leq p < 0.1$), e.g. 12.17*** or –5.49*.

More about regression

Regression modelling is a very complex and broad topic. This chapter only provides the basic strategies of fitting and interpreting linear regression models. For more detailed information, see, for example, Faraway (2009) and Fox (2008). Regression can also be performed on non-numeric response variables. Such extensions are called Generalized Linear Models. For example, logistic regression is an extension that deals with binary, multinomial and ordinal response variables. Logistic regression will be discussed in Chapters 12 and 13. Another popular extension is the Poisson regression, where one regresses on count data. It is also known as log-linear regression, especially when the counts are represented in contingency tables. See Gries (2013: 324–327) for a brief illustration of Poisson regression models and Field et al. (2012: 829–852) for a discussion of log-linear analysis.