CHAPTER 4

How to explore qualitative variables

proportions and their visualizations

What you will learn from this chapter:

This chapter demonstrates how to explore a categorical variable with the help of tables of counts and proportions. As in the previous chapter, graphs (pie charts, bar plots and dot charts) will play a very important role. You will also learn how to change values of a categorical variable. In addition, we will discuss how one can use Deviation of Proportions to measure dispersion of words in a corpus. This approach will be illustrated by a case study of the Basic Colour Terms in English.

4.1 Frequency tables, proportions and percentages

To reproduce the code in this and the following case studies, you will need two packages: `Rling` and `ggplot2`.

```
> install.packages("ggplot2") #unless you have already installed
the package
```

To be able to access the data and functions, you should load the packages:

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

This section investigates imaginary data with twenty simple clauses. Each of them is coded as intransitive, transitive or ditransitive. The data are available in the dataset `sent` in the package `Rling`. The structure of the dataset is the following:

```
> data(sent)
> str(sent)
'data.frame': 20 obs. of 2 variables:
$ clause: Factor w/ 3 levels "Ditr","Intrans",..: 3 3 1 3 2 2 3 …
$ subj: Factor w/ 4 levels "Abstr","Animal",..: 3 1 1 3 1 3 4 …
```

As one can see, the data frame contains twenty observations coded for two variables, *clause* and *subj*. The first variable, *clause*, is a factor with three levels: 'Ditr' (ditransitive), 'Intrans' (intransitive) and 'Trans' (transitive). The variable `subj` is a factor with four levels, which correspond to four semantic classes of the subject. Note that the function `summary()`, when applied to a factor, returns a vector with the frequencies of each level:

```
> summary(sent$clause)

  Ditr  Intrans  Trans
  2     10       8
```

Alternatively, one can use the function `table()` to see the frequencies:

```
> table(sent$clause)

  Ditr  Intrans  Trans
  2     10       8
```

For convenience, let us create a table with frequencies, which can be later used for various computations:

```
> sent.t <- table(sent$clause)
> sent.t

  Ditr  Intrans  Trans
  2     10       8
```

When analysing frequency data, it is often useful to compute proportions or percentages. Proportions are usually expressed as decimals that range from 0 to 1. To compute a proportion, one divides each value by the total frequency, which can be accessed with the help of `sum()`:

```
> sent.t/sum(sent.t)

  Ditr  Intrans  Trans
  0.1   0.5      0.4
```

Alternatively, one can use the function `prop.table()`, which turns raw frequencies into proportions:

```
> prop.table(sent.t)

  Ditr  Intrans  Trans
  0.1   0.5      0.4
```

Proportions of one category

If you only need the proportion of one category, for example, that of intransitive clauses, you can use the code below:

```
> mean(sent$clause == "Intrans")
[1] 0.5
```

How can this be possible? Let us look closer at the code. The expression in parentheses returns a vector of Boolean values (TRUE or FALSE):

```
> sent$clause == "Intrans"
[1] FALSE FALSE FALSE FALSE TRUE TRUE FALSE FALSE FALSE TRUE
[11] TRUE FALSE TRUE TRUE FALSE FALSE TRUE TRUE TRUE TRUE
```

In the numerical form, the values TRUE are represented by 1's, and FALSE is represented by 0's:

```
> as.numeric(sent$clause == "Intrans")
[1] 0 0 0 0 1 1 0 0 0 1 1 0 1 1 0 0 1 1 1 1
```

Since the function `mean()` coerces the Boolean values into 1's and 0's, the mean of this vector will be equal to the proportion of intransitive clauses: (0 + 0 + 0 + 0 + 1 + 1 + 0 + 0 + 0 + 1 + 1 + 0 + 1 + 1 + 0 + 0 + 1 + 1 + 1 + 1)/20 = 0.5.

One can also express proportions as percentages. To do so, one should multiply each proportion by 100, for example:

```
> prop.table(sent.t)*100

  Ditr   Intrans   Trans
  10     50        40
```

This means that intransitive clauses constitute 50% of the observations, transitive ones account for 40%, and ditransitive clauses represent 10% of the data.

Caution: Proportions and ratios (odds)

It is easy to confuse proportions with ratios. If we have two categories, *a* and *b*, the **proportion** of *a* in the data will be equal to *a*/(*a*+*b*). The minimum proportion is 0, and the

(*Continued*)

maximum is 1. If *a* and *b* have equal values, their proportions will be 0.5. In contrast, a **ratio (in probabilistic terms, odds)** of *a* to *b* can be expressed as *a*/*b*. For example, the ratio of intransitive to transitive clauses in our data is as follows:

```
> 10/8
[1] 1.25
```

And the other way round, the ratio of transitive to intransitive is as follows:

```
> 8/10
[1] 0.8
```

If *a* and *b* are equal, the ratio is 1. If $a > b$, the ratio is greater than 1. If $a < b$, the ratio will be in the range from 0 to 1.

An important related concept is **odds ratio**. It will be discussed in Chapter 9.

4.2 Visualization of categorical data

It is often useful to visualize frequencies and proportions. R offers many different tools for that purpose. This section will introduce three popular types of graphs: pie charts, bar plots and dot charts. We will continue to work with the same data frame `sent` and the table `sent.t` that were introduced in the previous section.

A **pie chart** displays proportions as sections of a circle (to be entirely correct, a disc). The greater a proportion, the greater the size of the section. A simple pie chart can be produced as follows:

```
> pie(sent.t)
```

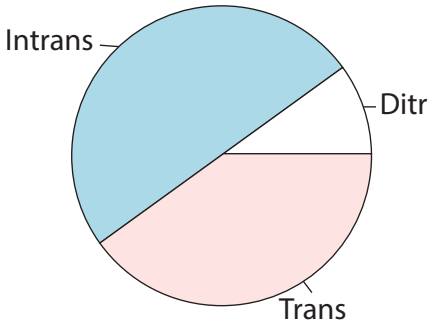


Figure 4.1. Pie chart of clause type proportions, simple version

The result is shown in Figure 4.1. However, it may be worth knowing how to create a customized version with a title, percentages as labels, a legend, and customized colours, as shown in Figure 4.2. To provide the labels, one has to compute the percentages first:

```
> sent_labels <- prop.table(sent.t)*100
> sent_labels

     Ditr Intrans   Trans
     10      50     40
```

Next, one should paste the numbers and the '%' symbol, which should immediately follow the numbers. The last argument, `sep = ""`, 'tells' R to insert nothing between the numbers and the '%' symbol (by default, R will add a space).

```
> sent_labels <- paste(sent_labels, "%", sep = "")
> sent_labels
[1] "10%" "50%" "40%"
```

The labels are ready. The next step is to make a customized list of colour names, which will be used to fill in the segments. We will use black for the first frequency value (ditransitives), 'grey40' (darker grey) for the second frequency value (intransitives), and 'grey80' (lighter grey) for the third frequency (transitives). The complete list of colours available in R can be accessed by typing in `colours() or colors()`.

```
> sent_colours <- c("black", "grey40", "grey80")
```

Now it is time to put all elements together and to produce an enhanced pie chart with the title, labels and customized colours:

```
> pie(sent.t, main = "Pie chart of clause types", labels = sent_
labels, col = sent_colours)
```

The final task is to add a legend at the right-hand side. The position of the legend is specified here by coordinates 1 (horizontal) and 0 (vertical). The coordinates usually have their origin (0, 0) at the bottom left corner and then increase in the right direction (horizontal) and to the top (vertical). In this pie chart, the horizontal 0 corresponds to the location of the pie centre. The text for the legend will correspond to the factor levels, and the customized colours will be used for the fillings:

```
> legend(1, 0, legend = levels(sent$clause), fill = sent_colours)
```

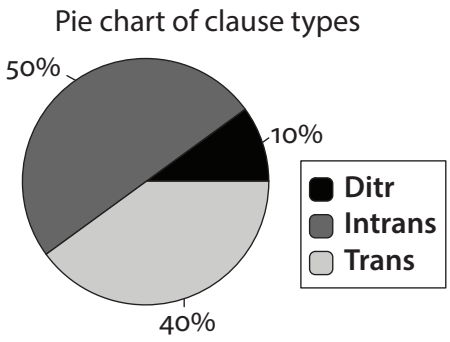
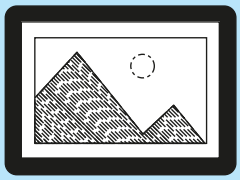


Figure 4.2. Pie chart of clause type proportions, enhanced version



How to make a pie chart with the help of `ggplot2`

To make a pie chart with `ggplot2`, you can use the following code:

```
> ggplot(sent, aes(x = factor(""), fill = clause)) + geom_bar()
+ coord_polar(theta = "y") + scale_x_discrete("") + scale_fill_
manual(values = c("black", "grey40", "grey80"))
```

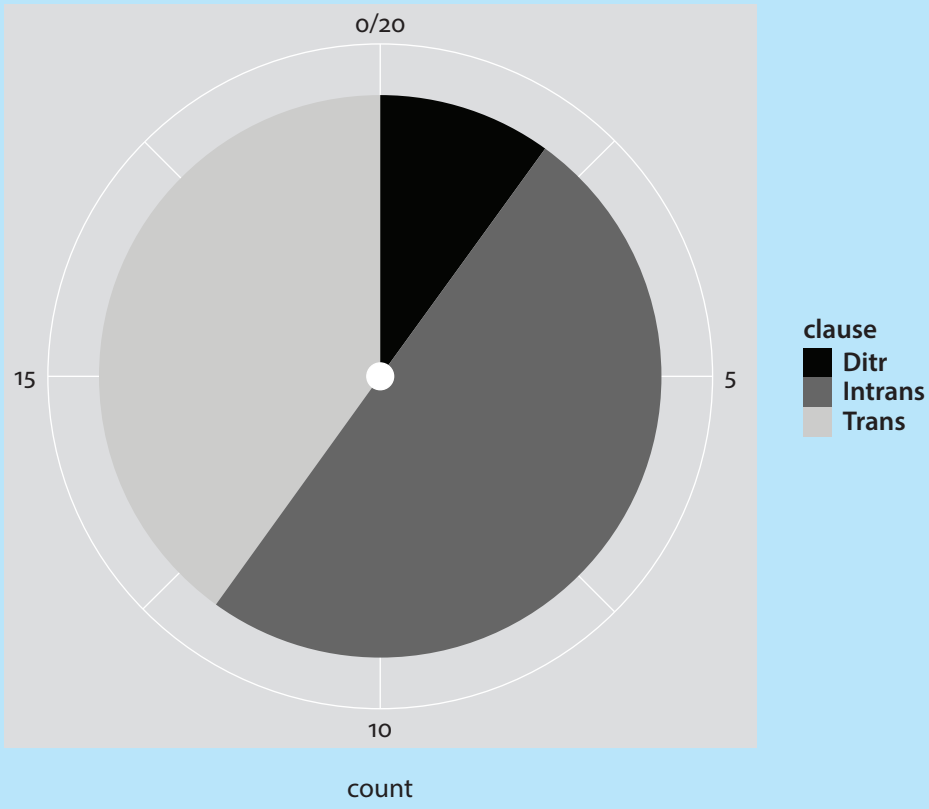


Figure 4.2a. A modified `ggplot2` version of the pie chart in Figure 4.2

The next graph is the **bar plot**, which represents quantities as vertical or horizontal bars. The code below can be used to reproduce the graph in Figure 4.3:

```
> barplot(sent.t, main = "Bar plot of clause types", col = "grey50",
cex.names = 1.2, xlab = "Clause type", ylab = "Frequency")
```

The argument `col` specifies a new shade of grey, whereas `cex.names = 1.2` is added to increase the size of the text labels in comparison with the default (1). Finally, `xlab` and `ylab` specify the labels of the axes.

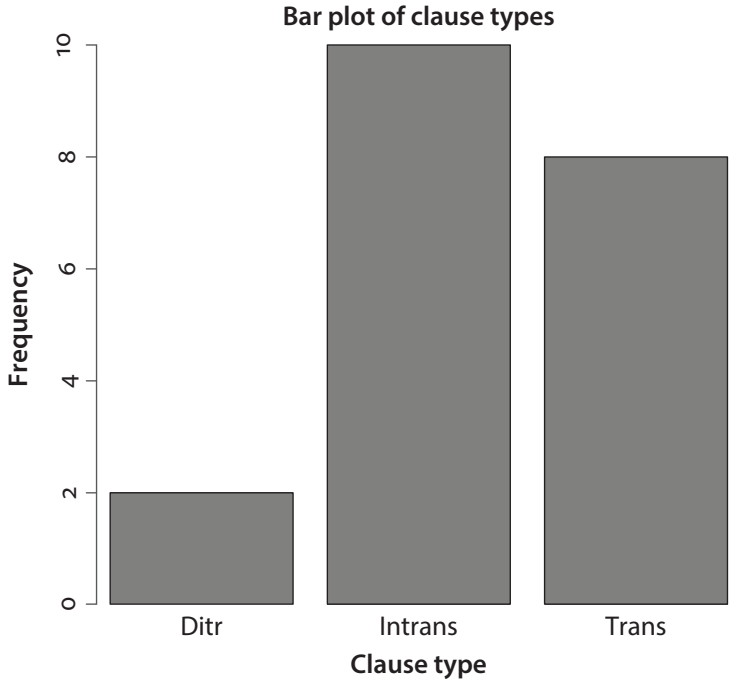
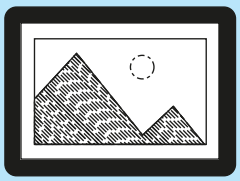


Figure 4.3. Bar plot of clause type frequencies



How to make a bar plot with the help of `ggplot2`

To create a `ggplot2` version of a bar plot of counts, you can use the code below. To make the bars more visible on the grey background, one can make them white and add a black outline, as shown in Figure 4.3a.

```
> ggplot(sent, aes(x = clause)) + geom_bar(fill = "white", colour
= "black") + xlab("Clause type") + ylab("Frequency")
```

(*Continued*)

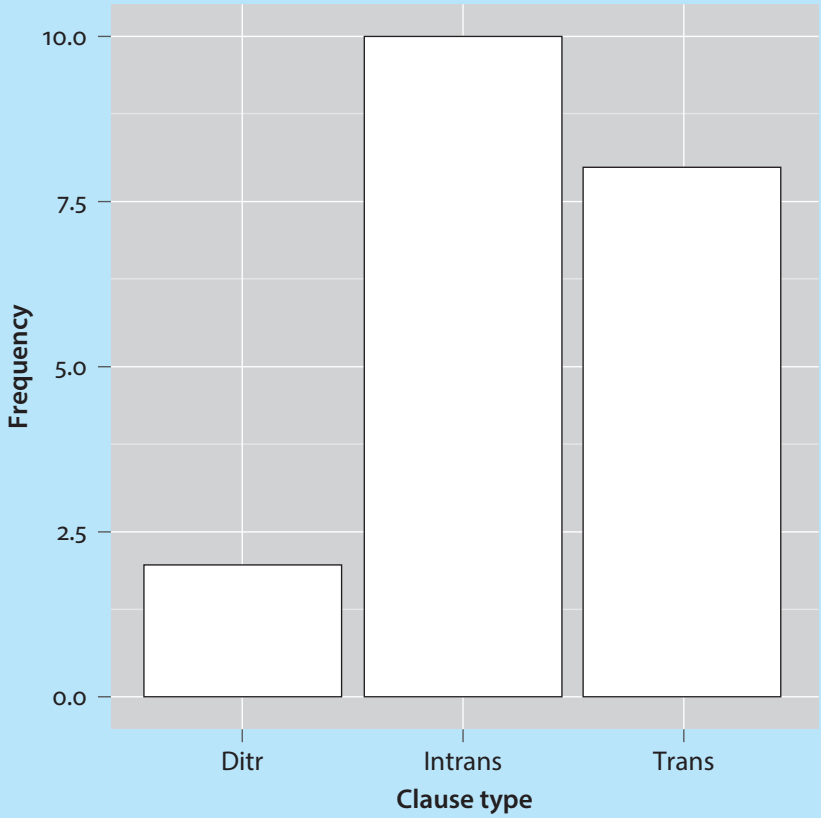


Figure 4.3a. A `ggplot2` version of the bar plot in Figure 4.3

How to edit factors

Editing factors in R is, unfortunately, not always easy. Let us first create a new factor, a copy of `sent$clause`, and use it to try different transformations:

```
> clause1 <- sent$clause
> head(clause1)
[1] Trans    Trans  Ditr   Trans  Intrans Intrans
Levels: Ditr Intrans Trans
```

Imagine now that the first observation contains an error. Instead of 'Intrans', the value should be 'Trans'. To correct this mistake, one can specify the index of the observation (`[1]`) and assign the correct value:

```
> clause1[1] <- "Intrans"
```

```
> head(clause1)
[1] Intrans Trans   Ditr   Trans  Intrans Intrans
Levels: Ditr Intrans Trans
```

However, the task becomes more complex when one wants to add a new category, which does not correspond to any existing factor level. Let us try to change the value of the second observation from 'Trans' to 'Copula':

```
> clause1[2] <- "Copula"
Warning message:
In '[<-.factor'('*tmp*', 2, value = "Copula"):
invalid factor level, NA generated
> head(clause1)
[1] Intrans ⟨NA⟩     Ditr   Trans  Intrans Intrans
Levels: Ditr Intrans Trans
```

Our actions produce a warning message and a missing value. In this situation, one first has to add the new level to the list of factor levels:

```
> clause1 <- factor(clause1, levels = c(levels(clause1), "Copula"))
> summary(clause1)
  Ditr   Intrans  Trans   Copula   NA's
  2      11       6       0        1
```

Now it is possible to assign the new value:

```
> clause1[2] <- "Copula"
> head(clause1)
[1] Intrans Copula Ditr Trans Intrans Intrans
Levels: Ditr Intrans Trans Copula
```

Sometimes it is necessary to conflate two or more levels in one new level. Imagine you want to replace two old categories, 'Copula' and 'Intrans' with one some new value, e.g. 'NonTrans'. In that case, one has to add the new level first, and only after that can one replace all values 'Copula' and 'Intrans' with the new value:

```
> clause1 <- factor(clause1, levels = c(levels(clause1),
"NonTrans"))
> clause1[clause1=="Intrans"|clause1=="Copula"] <- "NonTrans"
> summary(clause1)
  Ditr   Intrans  Trans  Copula  NonTrans
  2      0        6      0       12
```

(*Continued*)

The sign '|' tells R to replace any value that is equal either to 'Intrans' or 'Copula'. Now these two levels have zero frequencies. To remove them, one can simply use `factor()`. This will drop the levels that do not occur:

```
> clause1 <- factor(clause1)
> summary(clause1)
  Ditr  Trans   NonTrans
  2     6       12
```

One often needs to reorder the levels of a factor. By default, the levels of a factor are ordered alphabetically. New levels (here, 'NonTrans') are appended at the end of the list:

```
> levels(clause1)
[1] "Ditr"    "Trans" "NonTrans"
```

Imagine that you want to change the order of the levels for some subsequent analyses. In that case, you can again use `factor()` and specify the preferred order of levels:

```
> clause1 <- factor(clause1, levels = c("Trans", "NonTrans",
"Ditr"))
> levels(clause1)
[1] "Trans"   "NonTrans" "Ditr"
```

Another useful command is `relevel()`, which reorders a factor so that the level specified by `ref` comes first and the other ones are moved down. For example, one can make 'NonTrans' the first level, which is also called the **reference level**:

```
> clause1 <- relevel(clause1, ref = "NonTrans")
> levels(clause1)
[1] "NonTrans" "Trans"     "Ditr"
```

This function is particularly useful when one wants to investigate categorical variables in regression analysis (see Chapters 7 and 12).

The final type of graph discussed in this section is **Cleveland's dot chart** (Figure 4.4). Although dot charts may be more useful in the situations when there are many different scores, let us create one here as an illustration:

```
> dotchart(sent.t, main = "Dot chart of clause types", xlab =
"Frequency", ylab = "Clause type", lcolor = "black", pch = 16, xlim
= c(0, 12))
[warning message omitted]
```

The argument `lcolor` specifies the colour of the horizontal lines, whereas `pch` specifies the type of points. The argument `xlim` provides the limits, i.e. the minimum and the maximum values, for the horizontal axis. See Appendix 2 for more options. There is also a warning message because we use a table instead of a numeric vector or a matrix, but this is not dangerous.

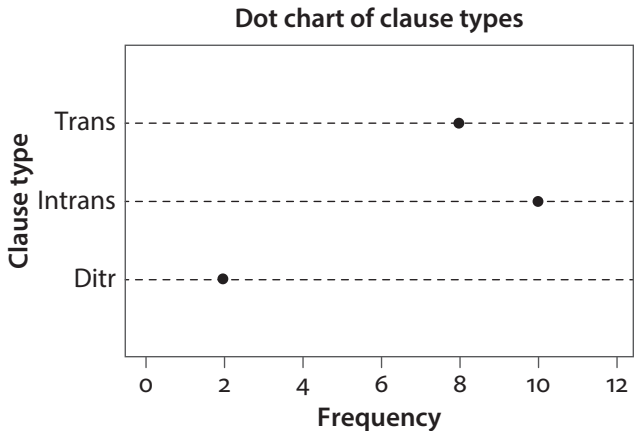
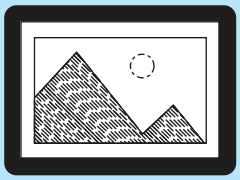


Figure 4.4. A dot chart of clause type frequencies



How to make a dot chart with the help of `ggplot2`

To make a `ggplot2` version of the dot chart in Figure 4.4, you can use the following code:

```
> ggplot(sent, aes(x = clause)) + geom_point(stat = "bin", size
= 5) + xlab("Clause type") + ylab("Frequency") + coord_flip() +
ylim(0, 12) + theme_bw() + theme(panel.grid.major.x = element_
blank(), panel.grid.major.y = element_line(colour = "grey60",
linetype = "dashed"))
```

The result is shown in Fugure 4.4a.

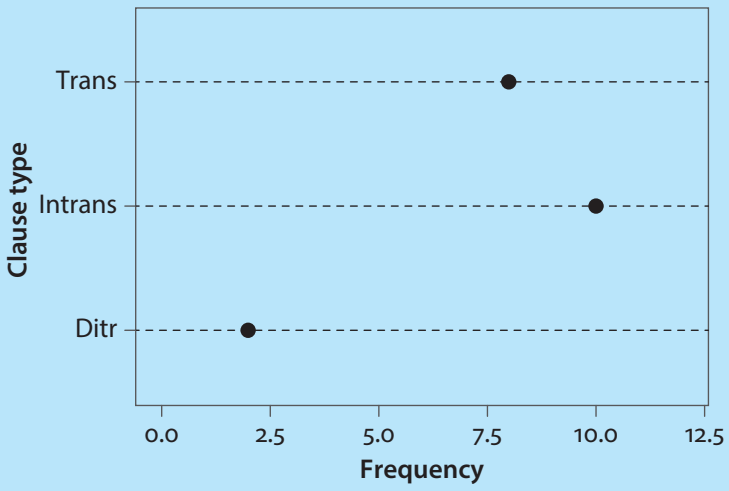
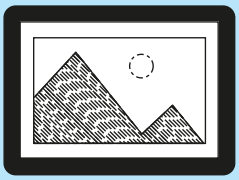


Figure 4.4a. A `ggplot2` version of the dot chart in Figure 4.4

(*Continued*)



Some do's and dont's for producing graphs

- avoid using 3D-effects, fancy graphical patterns, flashy colours and other distractors; follow the rule 'less is more' and let your data speak for themselves
- provide clear and informative labels
- do not use tricks, such as manipulating the scale to make differences more or less visible
- do not use red and green symbols on the same graph, as up to 5% of your audience may be colour-blind people

There is an opinion that pie charts are not optimal for representation of quantitative differences because humans are not good at estimating and comparing angles (Gries 2013: 109), so do not use pie charts when the differences between numerical values are too subtle.

4.3 Basic Colour Terms: Deviations of Proportions in subcorpora

4.3.1 The data and hypothesis

To reproduce the code in this case study, you will need only the companion package, `Rling`.

```
> library(Rling)
```

The aim of this case study is to pinpoint usage constraints on a universal conceptual and linguistic phenomenon, namely, the Basic Colour Terms (BCT). The research of colour categories began with a seminal study by Berlin and Kay (1969), who suggested a universal hierarchy in the development of colour terms across languages of the world. This hierarchy looks as follows:

white < red < green < blue < brown < purple pink grey orange black yellow

The hierarchy constrains the inventory of colour terms in languages. If a language has a colour category, it will also have all terms that are found on the left from this term. For instance, if a language contains a term for blue, it will also have words for white, black, red, green and yellow. It has been assumed for a long time that the basic colour categories are universal and are grounded in the neurophychological mechanisms of perception that are shared by most humans (e.g. Rosch Heider & Olivier 1972). This evidence was important in the theoretical battle over the Whorfian hypothesis, which says that different languages cut the conceptual space in arbitrary ways, and the universalist paradigm promoted by Generative Linguistics. Today most researchers would agree that neurophysiological, psychological and cultural factors simultaneously play a role in the way we learn, process and use colour categories (see an overview in Anishchanka 2013).

The dataset that will be used in this case study is called `colreg`. It contains the frequencies of eleven BCT in four registers from the Corpus of Contemporary American English, or COCA (Davies 2008 –).[1] Only the adjectival uses of the colour terms were counted.

```
> data(colreg)
> colreg
        spoken  fiction  academic  press
black   20335   41118    26892     73080
blue    4693    22093    3605      21210
brown   1185    10914    1201      11539
gray    1168    12140    1289      6559
green   3860    14398    4477      26837
orange  931     3496     474       5766
pink    962     7312     584       6356
purple  613     3366     429       3403
red     7230    25111    5621      34596
white   14474   40745    26336     54883
yellow  1349    10553    1855      10382
```

We will focus on the difference between primary and secondary BCT. The primary BCT are black, white, red, green, yellow and blue. The secondary ones are brown, purple, pink, orange and grey. It is believed that the primary colours are directly determined by neural responses, whereas the secondary ones are generated by additional cognitive operations on these neural responses (Kay & McDaniel 1978). This means that they can also be more prone to cultural constraints in their use. Thus, we can expect that the register biases will be greater for the secondary BCT than for the primary BCT, whereas the primary BCT will

1. The frequencies are available at http://www.wordandphrase.info/frequencyList.asp (last access 11.06.2015).

be more evenly distributed. These biases will be measured with the help of Gries' (2008) Deviations of Proportions.

4.3.2 Deviation of Proportions as a measure of dispersion

In the previous chapter you learnt a few popular measures of dispersion: range, variance, standard variation, IQR and median absolute deviation. In corpus linguistics, one also speaks about dispersion of words in a corpus, which serves as a measure of familiarity, or entrenchment of a word, in addition to corpus frequency. For example, one can use the number of texts in a corpus where a word occurs. If a word occurs in very many texts, it is highly entrenched. If it occurs in only one or two texts, it is less spread, even though it may have a high frequency due to a large number of occurrences in those few texts. Thus, information about dispersion provides a useful addition to corpus frequency.

Counting the number of texts where a word occurs works well if the number of texts is very large (hundreds or thousands) and the texts are of comparable size. However, in our case all colours are found in all registers, and the total numbers of tokens in each register are different. This is a normal situation: more often than not the sizes of texts that constitute a corpus vary substantially. Presumably, a word will have more chances to occur in a larger text than in a smaller one. How can one take this into account? A convenient and intuitive measure of variation is the so-called Deviation of Proportions, or DP (Gries 2008; Lijffijt & Gries 2012). This measure compares the actual proportion of occurrences of a word in a given text or group of texts (as compared to the other texts or groups) with the expected occurrences of the word given the size of the text. To compute it, we need the total sizes of the texts (registers). This information can be found in the COCA documentation. On the basis of this information, we can create a vector with the total number of words in each of the four registers:

```
> freqreg <- c(95385672, 90344134, 91044778, 187245672)
> freqreg
[1] 95385672 90344134 91044778 187245672
```

To obtain the proportions of each register in the corpus, one can use the function `prop.table()`. These proportions are called **expected proportions** because they represent the probability of a word to occur in a particular register by mere chance. The vector of expected proportions for all colour terms (or any other words that occur in the corpus) looks as follows:

```
> exp_prop <- prop.table(freqreg)
> exp_prop
[1] 0.2055636 0.1946987 0.1962086 0.4035291
```

The press has the greatest proportion (about 0.40) because it is the largest subcorpus, whereas the three other registers have nearly equal proportions. To compute a DP, we will

also need the observed proportions for each colour term. Let us begin with *black*. The frequencies of this adjective in the four registers are as follows:

```
> colreg[1,]
          spoken   fiction   academic   press
  black   20335    41118     26892      73080
```

These frequencies should be transformed into the so-called **observed proportions** with the help of `prop.table()`:

```
> black_obs <- prop.table(colreg[1,])
> black_obs
  spoken  fiction     academic   press
  black   0.1259718  0.2547189  0.1665913  0.452718
```

Now we are ready to compute the DP of *black*. DP is the sum of absolute differences between the observed and expected proportions of a word or a group of words, as in this case, divided by two. The function `abs()` returns the absolute value, regardless of the sign (plus or minus).

```
> DP_black <- sum(abs(black_obs - exp_prop))/2
> DP_black
[1] 0.1092091
```

It can be useful to normalize the DPs by dividing a DP by one minus the smallest proportion in the expected values, as suggested by Lijffijt & Gries (2012). In that case, the distribution of DP values will range from 0 to 1.

```
> DP_black_norm <- DP_black/(1 - min(exp_prop))
> DP_black_norm
[1] 0.1356127
```

The normalized DP can be interpreted as follows. If the value is close to 0, then the word is distributed across *n* subcorpora as one should expect given the sizes of the subcorpora. A DP close to 1 indicates that the word has a strong preference for some subcorpora, and strongly 'disfavours' others. The value 0.136 indicates that *black* is quite evenly distributed.

Let us now repeat the same procedure and compute the normalized DP score for *gray* (with the American English spelling):

```
> gray_obs <- prop.table(colreg[4,])
> gray_obs
       spoken       fiction      academic     press
gray   0.05520892   0.5738325   0.06092834   0.3100303
> DP_gray <- sum(abs(gray_obs - exp_prop))/2
> DP_gray
```

```
[1] 0.3791338
> DP_gray_norm <- DP_gray/(1 - min(exp_prop))
> DP_gray_norm
[1] 0.4707974
```

The register bias of *gray* is thus much greater than that of *black*.

This study focuses on the aggregate differences between the primary and secondary BCT. To obtain and compare the aggregate DP scores, one should first compute the sum frequencies of the primary and secondary colour terms. For convenience, let us create two subsets of the initial data, one with primary terms, and the other with the secondary terms. The subsetting is done by listing the row numbers of the corresponding terms (see more on subsetting in Appendix 1).

```
> primcol <- colreg[c(1, 2, 5, 9:11),]
> primcol
       spoken  fiction  academic  press
black  20335   41118   26892     73080
blue   4693    22093   3605      21210
green  3860    14398   4477      26837
red    7230    25111   5621      34596
white  14474   40745   26336     54883
yellow 1349    10553   1855      10382
> seccol <- colreg[-c(1, 2, 5, 9:11),]
> seccol
       spoken  fiction  academic  press
brown  1185    10914   1201      11539
gray   1168    12140   1289      6559
orange 931     3496    474       5766
pink   962     7312    584       6356
purple 613     3366    429       3403
```

The next step is to compute the column sums with the help of a special function `colSums()`.

```
> primcol_sums <- colSums(primcol)
> primcol_sums
spoken  fiction  academic  press
51941   154018  68786     220988
> seccol_sums <- colSums(seccol)
> seccol_sums
spoken  fiction  academic  press
4859    37228   3977      33623
```

Now one can obtain the observed proportions with the help of `prop.table()`:

```
> primcol_obs <- prop.table(primcol_sums)
> primcol_obs
```

```
spoken      fiction      academic    press
0.1047762   0.3106874    0.1387561   0.4457803
> seccol_obs <- prop.table(seccol_sums)
> seccol_obs
spoken      fiction      academic    press
0.06097607  0.46717783   0.04990776  0.42193833
```

The final step is to compute the DP values and their normalized versions for the primary and secondary terms.

```
> DP_primcol <- sum(abs(primcol_obs - exp_prop))/2 #simple DPs for
primary BCT
> DP_primcol
[1] 0.1582399
> DP_seccol <- sum(abs(seccol_obs - exp_prop))/2 #simple DPs for
secondary BCT
> DP_seccol
[1] 0.2908884
> DP_primcol_norm <- DP_primcol/(1 - min(exp_prop)) #normalized DPs
for primary BCT
> DP_primcol_norm
[1] 0.1964978
> DP_seccol_norm <- DP_seccol/(1 - min(exp_prop)) #normalized DPs
for secondary BCT
> DP_seccol_norm
[1] 0.3612168
```

The DPs show that the secondary terms are less evenly distributed in the corpus than the primary ones. This means that cultural factors have a stronger influence on the use of the secondary BCT than on the use of the primary BCT, in accordance with the theory-based expectations.

4.4 Summary

This chapter has presented different statistical and graphical tools for exploratory analysis of univariate categorical data. The key notions were counts (frequencies) and proportions. A case study of the Basic Colour Terms has also demonstrated how proportions can be used to measure dispersion of a word in a corpus. In addition, you have learnt how to edit factors, e.g. introduce new levels and remove unnecessary ones. The simple exploratory techniques described in this and previous chapters play a very important part in statistical analysis, as you will see in the next part of the book, which is dedicated to inferential statistics and hypothesis testing.