APPENDIX 2

Main plotting functions and graphical parameters in R

1 A scatter plot with text labels

A scatter plot represents individual observations (rows) as points in a two-dimensional space delimited by the *x*- and *y*-axis. For illustration, we will use a small sample from the dataset `ELP` (English Lexicon Project) data:

```
> library(Rling)
> data(ELP)
> data <- ELP[31:40, ]
> data
    Word          Length  SUBTLWF  POS  Mean_RT
31  ordinances    10      0.16     NN   796.59
32  ovary         5       0.51     NN   694.13
33  reared        6       0.41     VB   736.19
34  nudist        6       0.65     NN   752.58
35  exportation   11      0.02     NN   1121.00
36  unhurt        6       0.10     JJ   759.00
37  jackpot       7       3.71     NN   598.18
38  medieval      8       2.96     JJ   752.97
39  hangover      8       3.90     NN   616.55
40  quintet       7       0.51     NN   881.25
```

First, we will plot the points (stimuli) according to their log-transformed corpus frequency (the *x*-axis) and mean reaction time (the *y*-axis) and next add the text labels (the words):

```
> plot(log(data$SUBTLWF), data$Mean_RT, pch = 16, cex = 1.2, main
= "Corpus frequency and mean reaction times", sub = "Data from
English Lexicon Project", xlim = c(-5, 2), ylim = c(500, 1200), xlab
= "Log-transformed corpus frequency", ylab = "Mean reaction time,
ms", cex.lab = 0.8, cex.axis = 0.8)
```

The result is shown in Figure A1. Some comments are due:

- `log(data$SUBTLWF)` are the *x*-coordinates of plotted points.
- `data$Mean_RT` are the *y*-coordinates of plotted points.
- `pch = 16` determines the type of plotted points, by default `pch = 1`.
- `cex = 1.2` specifies the size of plotted text and symbols, by default `cex = 1`.
- `main = "Corpus…"` gives the plot a title.

- `sub = "data..."` provides a subtitle of the plot.
- `xlim = c(-5, 2)` determines the range of *x*-values on the plot.
- `ylim = c(500, 1200)` determines the range of *y*-values on the plot.
- `xlab = "Log..."` provides a text label for the *x*-axis.
- `ylab = "Mean..."` provides a text label for the *y*-axis.
- `cex.lab = 0.8` specifies the size of axis marks, relative to the current setting of `cex`.
- `cex.axis = 0.8` specifies the size of axis labels, relative to the current setting of `cex`.

```
> text(log(data$SUBTLWF), data$Mean_RT, labels = data$Word, adj =
c(1.2, 0), cex = 0.7, font = 4)
```

- `log(data$SUBTLWF)` provides the *x*-coordinates of the text labels.
- `data$Mean_RT` contains the *y*-coordinates of the text labels.
- `labels = data$Word` specifies the text labels.
- `adj = c(1.2, 0)` are adjustments for text labels. The first number is the horizontal adjustment (negative: right, positive: left), and the second one is the vertical adjustment (negative: up, positive: down). The adjustments are added to the *x* and *y*-values provided in the first two arguments.
- `cex = 0.7` determines the font size of text labels (1 by default).
- `font = 4` specifies he font type for text: '1' for plain text (the default), '2' for bold face, '3' for italic, '4' for bold italic (here).

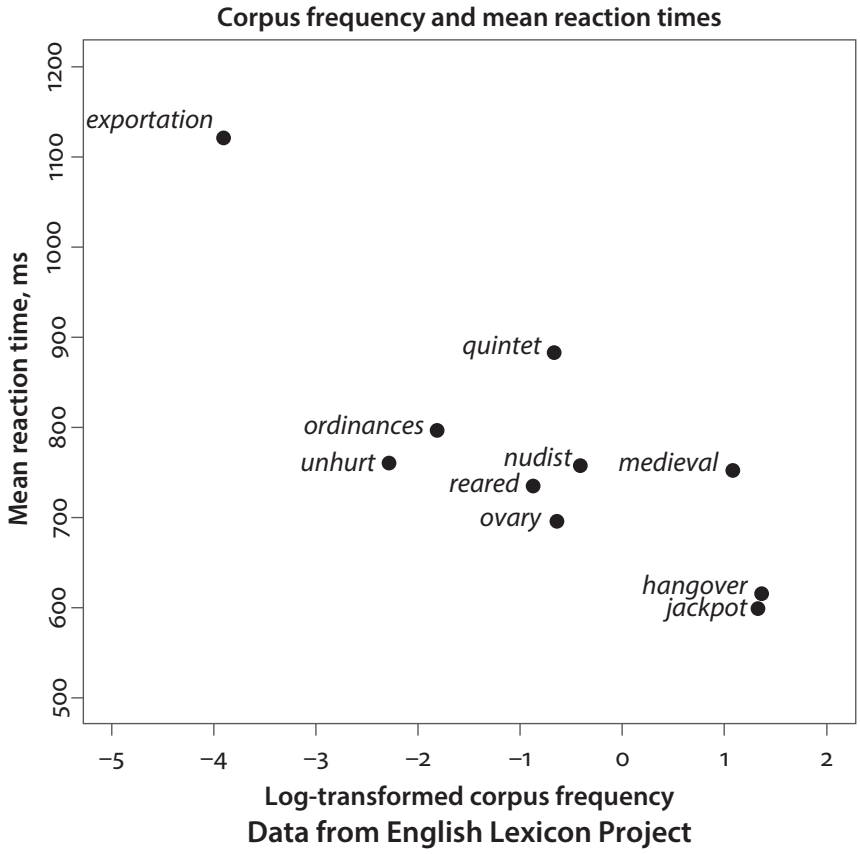

Figure A1. A scatter plot of experimental lexical data with points and text labels

Now we can try to plot the words that belong to different parts of speech (nouns vs. non-nouns) as different symbols. We will also use different colours for their text labels. First, it is necessary to get the indices of all observations that represent nouns:

```
> NN <- which(data$POS == "NN")
> NN
[1] 1 2 4 5 7 9 10
```

These are the rows which contain nouns. Next, we create an empty plot without any points with the help of `type = "n"`. The axes and titles remain the same.

```
> plot(log(data$SUBTLWF), data$Mean_RT, type = "n", main = "Corpus frequency and mean reaction times", sub = "Data from English Lexicon Project", xlim = c(-5, 2), ylim = c(500, 1200), xlab = "Log-transformed corpus frequency", ylab = "Mean reaction time, ms", cex.lab = 0.8, cex.axis = 0.8)
```

Now we can plot the points, first, the ones that correspond to the nouns, and next those that correspond to other parts of speech:

```
> points(log(data$SUBTLWF)[NN], data$Mean_RT[NN], cex = 1.2)
> points(log(data$SUBTLWF)[-NN], data$Mean_RT[-NN], cex = 1.2, pch = 2, col = "grey40")
```

Finally, we add text labels, which differ in the font effects and colour (`font = 2` is for italic).

```
> text(log(data$SUBTLWF)[NN], data$Mean_RT[NN], labels = data$Word[NN], adj = c(1.2, 0), cex = 0.7, font = 4)
> text(log(data$SUBTLWF)[-NN], data$Mean_RT[-NN], labels = data$Word[-NN], adj = c(1.2, 0), cex = 0.7, font = 2, col = "grey40")
```

To add a legend, one can do the following:

```
> legend("topright", pch = c(1, 2), col = c("black", "grey40"), legend = c("Nouns", "Other POS"), cex = 0.8, bty = "n")
```

Comments:

- `"bottomright"` specifies the position of the legend. The other options are `"top"`, `"bottom"`, `"left"`, `"right"`, `"center"`, `"topleft"`, `"topright"` and `"bottomleft"`. Alternatively, one can use coordinates, e.g. `c(1, 1)`.
- `pch = c(1, 2)` specifies the symbols that should be plotted.
- `col = c("black", "grey40")` determines the colours of the symbols.
- `legend = c("Nouns", "Other POS")` is the text of the legend.
- `bty = "n"` says that no box should be added around the legend. By default, there is a box.

The result is shown in Figure A2.

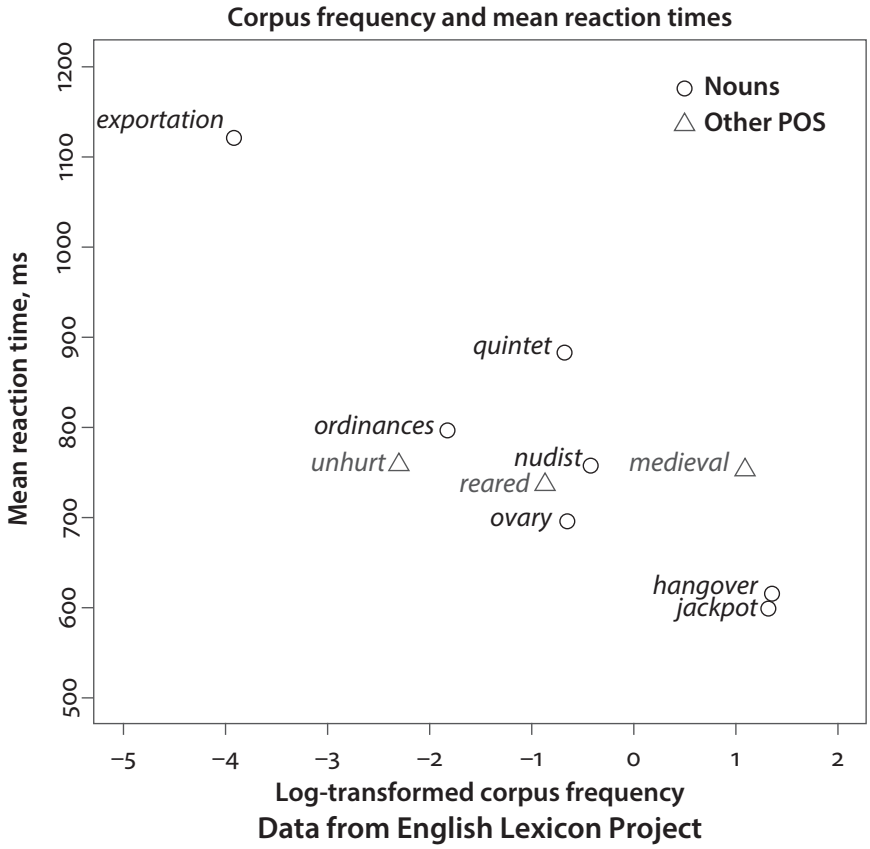

Figure A2. A scatter plot of experimental lexical data with different symbols and colours

2 Line chart

A line chart shows one or more lines. It is especially useful in diachronic studies. We will compare the normalized frequencies of the nouns *nerd* and *geek* in the Corpus of Contemporary American English in five periods from 1990 to 2012. The matrix with the frequencies can be created as follows:

```
> data <- rbind(c(1.74, 1.81, 2.36, 2.43, 3.06), c(0.77, 1.89, 3.72,
3.88, 3.56))
> rownames(data) <- c("nerd", "geek")
> colnames(data) <- c("1990-1994", "1995-1999", "2000-2004", "2005-
2009", "2010-2012")
> data
      1990-1994  1995-1999  2000-2004  2005-2009  2010-2012
nerd  1.74       1.81       2.36       2.43       3.06
geek  0.77       1.89       3.72       3.88       3.56
```

First, we plot the frequencies of *nerd*. The *x*-axis corresponds to the time periods, whereas the *y*-axis shows the normalized frequencies per million words.

```
> plot(data[1,], type = "o", pch = 2, ylim = range(0, data[1, ],
data[2, ]), axes = FALSE, xlab = "Periods", ylab = "Frequency per
million words", main = "nerd and geek in COCA")
```

Comments:

- `data[1,]` gives the frequencies of *nerd*.
- `type = "o"` ('overplotted') will produce both lines and points.
- `pch = 2` specifies the type of points (empty triangles).
- `ylim = range(0, data[1, ], data[2, ])` gives the range of *y*, from 0 (the minimum value) to the maximum frequency from the frequencies of *nerd* and *geek*.
- `axes = FALSE` suppresses the plotting of axes.
- `xlab = "Periods"` specifies the text label for the *x*-axis.
- `ylab = "Frequency per million words"` specifies the text label for the *y*-axis.
- `main = "nerd and geek in COCA"` provides the title of the plot.

Next, we draw the axes: first, the horizontal, and then the vertical one:

```
> axis(1, at = 1:5, labels = colnames(data))
> axis(2)
```

This will add a box around the plot:

```
> box()
```

Now it is time to add the second line with the frequencies of *geek*:

```
> lines(data[2,], type = "o", lty = 2, pch = 0, col = "darkgrey")
```

- `lty = 2` specifies the line type (dashed). The other options are '1' (solid, default), '3' (dotted), '4' (dot-dash), '5' (long-dash), and '6' (two-dash).
- `pch = 0` specifies the points (empty squares).
- `col = "darkgrey"` specifies the colour of the line and the points.

Finally, we add a legend:

```
> legend("bottomright", lty = c(1, 2), col = c("black", "darkgrey"),
pch = c(2, 0), legend = c("nerd", "geek"))
```

See comments to the legend in a scatter plot. Figure A3 displays the results.

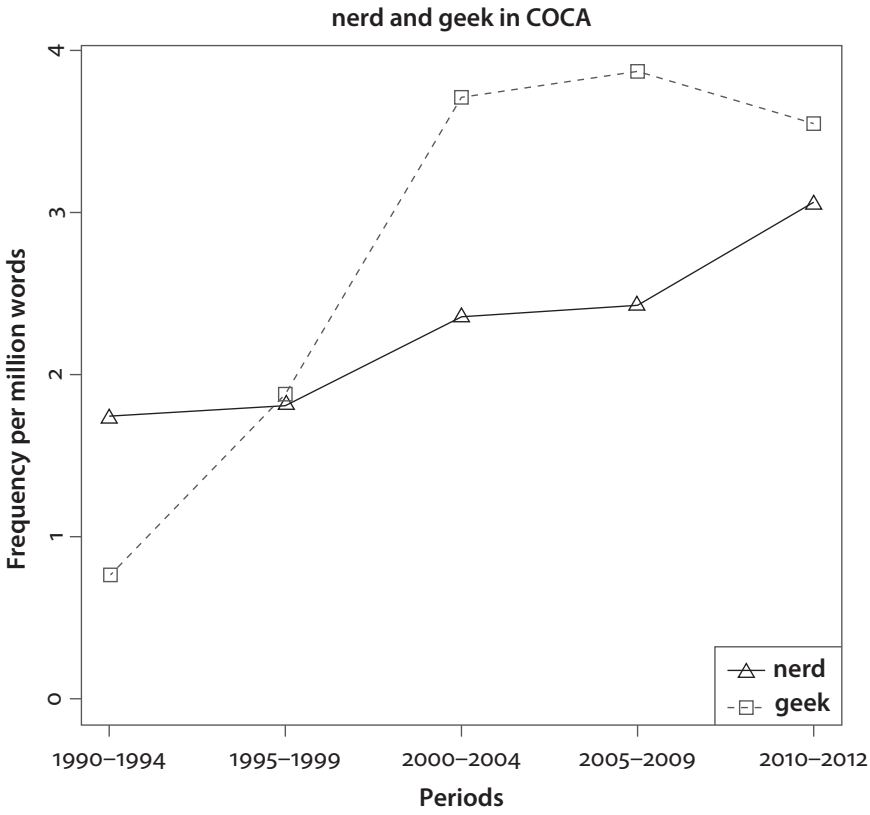

Figure A3. The frequencies of *nerd* and *geek* in COCA

3 Bar plots

Bar plots show numeric values as bars of different height. You can plot a single vector, or several. They can be stacked or juxtaposed. We will plot the frequencies of eleven basic colour terms in four registers of the Corpus of Contemporary American English (the data set `colreg` in `Rling`). The data look as follows:

```
> library(Rling)
> 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
```

First, we will plot only the sum frequencies of every colour term, sorted in descending order. To do so, we first compute the sum frequencies and sort them:

```
> colreg_sums <- sort(rowSums(colreg), decreasing = TRUE)
> colreg_sums
black   white   red    blue   green  brown  yellow  gray   pink
orange  purple
161425  136438  72558  51601  49572  24839  24139   21156  15214
10667   7811
```

Now we can make our bar plot. We can easily plot the bars in the colours that correspond to the colour terms:

```
> barplot(colreg_sums, main = "Basic Colour Terms in COCA", ylab =
"frequency", las = 3, col = names(colreg_sums))
```

The result can be seen in Figure A4. Some comments are due:

- `colreg_sums` is a vector with the frequencies to plot.
- `main = "Basic Colour Terms in COCA"` provides the title.
- `ylab = "frequency"` names the *y*-axis.
- `las = 3` makes the annotation of both axes vertically oriented. The other options are '0' (always parallel to the axes, the default), '1' (always horizontal) and '2' (always perpendicular to the axes).
- `col = names(colreg_sums)` specifies the colours of the bars.

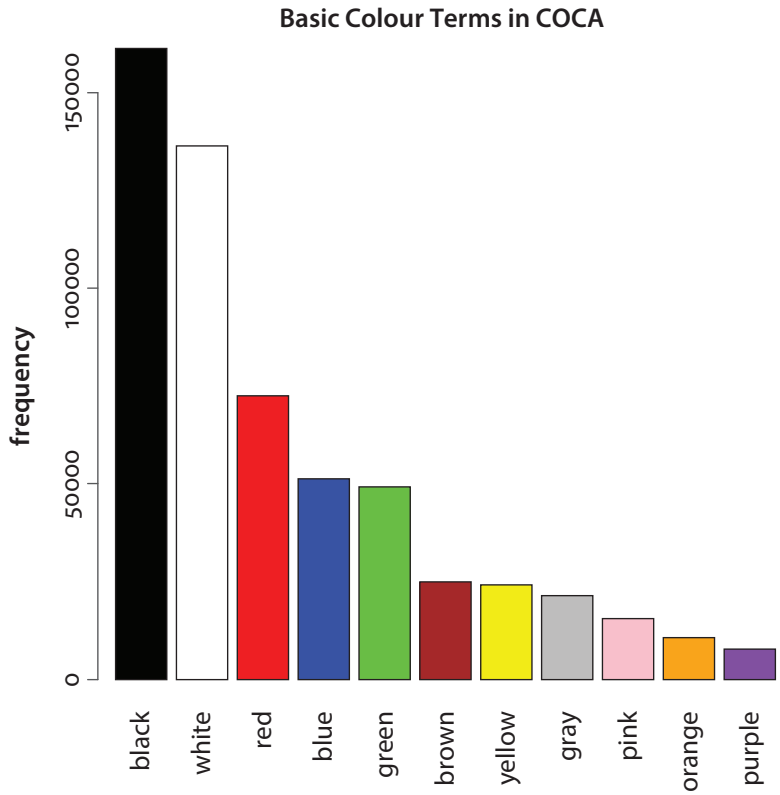

Figure A4. A bar plot of frequencies of Basic Colour Terms in COCA

We have just created a simple bar plot of only one vector. It is also possible to combine two and more vectors. Let us compare the frequencies of the colour terms in the spoken data, fiction and academic prose. First, we create a stacked bar plot with a legend. The bars are stacked by default. Note that we will have to subset and transpose the matrix `colreg`.

```
> barplot(t(colreg[, 1:3]), las = 3, main = "Frequencies of Basic
Colour Terms in COCA", ylab = "Frequency", col = c("black", "grey",
"white"))
```

- `t(colreg[, 1:3])` is the transposed version of the initial matrix with only three first columns ("spoken", "fiction" and "academic"). These columns become rows in the transposed version.
- `las = 3` makes the annotation of both axes vertically oriented. See the comments above.
- `main = "Frequencies of Basic Colour Terms in COCA"` provides the title of the plot.
- `ylab = "Frequency"` gives the name to the *y*-axis.
- `col = c("black", "grey", "white")` provides colours for the bars that relate to the registers.

```
> legend("top", fill = c("black", "grey", "white"), legend =
c("spoken", "fiction", "academic"))
```

- `"top"` specifies the position of the legend (at the top).
- `fill = c("black", "grey", "white")` provides the colours for the small filled squares.
- `legend = c("spoken", "fiction", "academic")` specifies the text of the legend.

See the result in Figure A5.

Another version of the bar plot is with bars next to one another. The code is the same, with the exception of one additional argument, `beside = TRUE`:

```
> barplot(t(colreg[, 1:3]), las = 3, main = "Frequencies of Basic
Colour Terms in COCA", ylab = "Frequency", col = c("black", "grey",
"white"), beside = TRUE)
> legend("top", fill = c("black", "grey", "white"), legend =
c("spoken", "fiction", "academic"))
```

See the result in Figure A6.

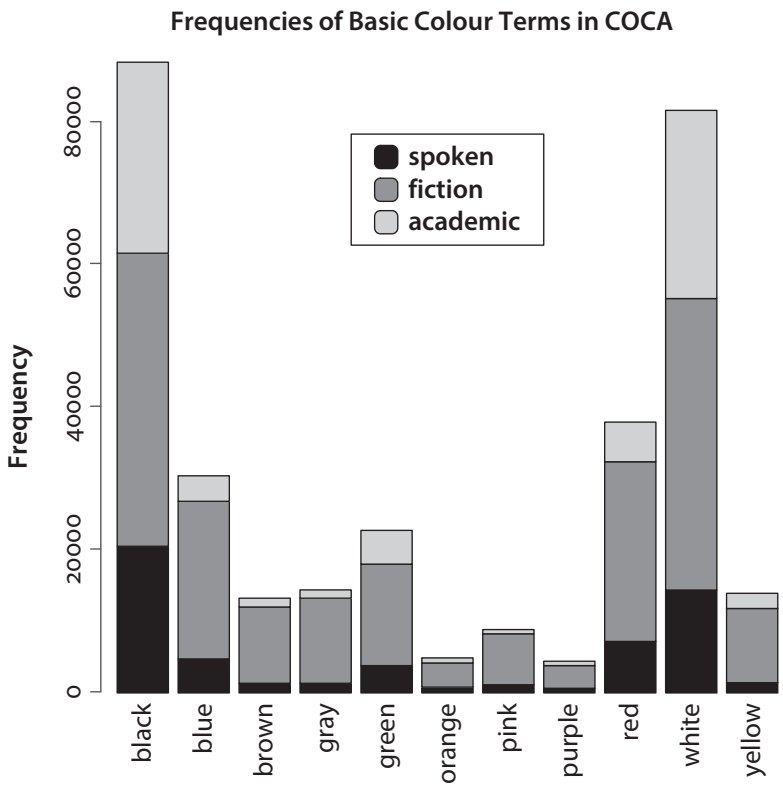

Figure A5. A bar plot of Basic Colour Terms in four registers of COCA. The bars are stacked

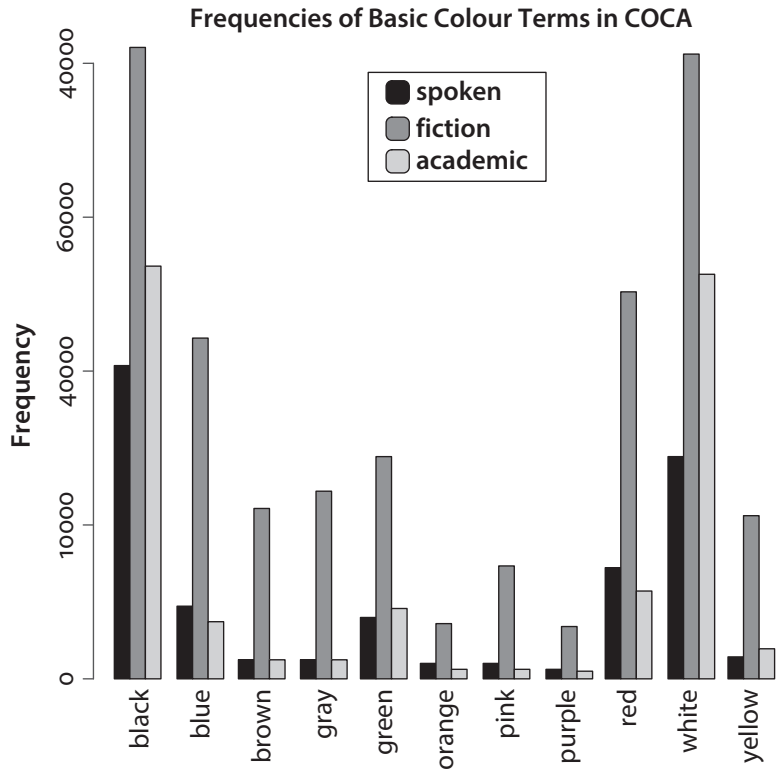

Figure A6. A bar plot of Basic Colour Terms in four registers of COCA. The bars are grouped (unstacked)

4 Dot charts

A dot chart shows basically the same information as a bar plot, but numeric values are displayed as dots at a certain distance from the origin. Below is a dot chart of the Basic Colour Term frequencies from the previous section. Note that a dot chart displays the first values in a vector at the bottom and the last values at the top, so we have to sort the vector again to put the more frequent terms at the top.

```
> dotchart(sort(colreg_sums), xlab = "frequency", main = "Basic
Colour Terms in COCA")
```

See the result in Figure A7.

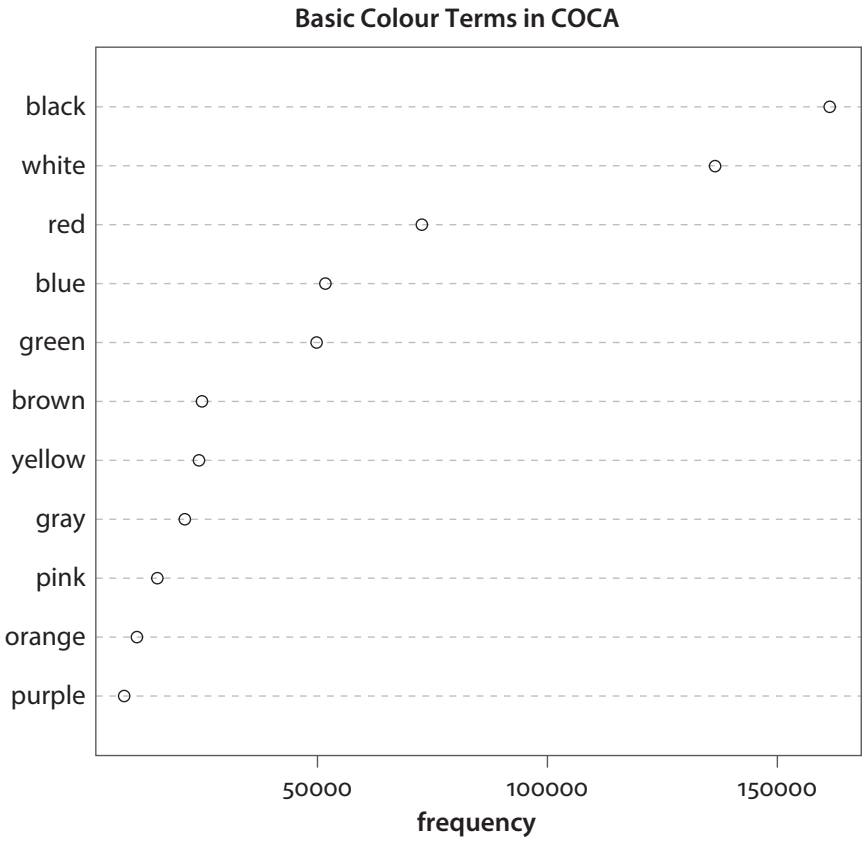

Figure A7. A dot chart of Basic Colour Terms in COCA

5 Pie charts

A pie chart displays the proportions of different categories as pieces of a pie. The larger a piece, the greater the proportion. We create a pie chart of frequencies of *although* in different registers of COCA. The frequencies are as follows:

```
> although <- c(9749, 13134, 25233, 20184, 52837)
> although
[1] 9749 13134 25233 20184 52837
```

The five frequencies relate to five registers in the following order: spoken, fiction, magazines, newspapers and academic. To make a pie chart, we will need to define the colours of the pie pieces:

```
> although_col <- c("black", "grey40", "grey70", "grey90", "white")
```

It is also useful to add the percentages of each register to the plot. First, we compute the percentages and then turn them into a character vector by pasting '%'.

```
> although_percent <- round(prop.table(although)*100, 1)
> although_percent
[1] 8.0 10.8 20.8 16.7 43.6
> although_percent <- paste(although_percent, "%", sep = "")
> although_percent
[1] "8%"     "10.8%" "20.8%" "16.7%" "43.6%"
```

Now we are ready to make our pie chart:

```
> pie(although, col = although_col, labels = although_percent, main
= "although in COCA")
> legend(c(1, 1), fill = although_col, legend = c("spoken", "fiction",
"magazine", "newspaper", "academic"))
```

See the previous plots and help pages for more details. The result can be found in Figure A8.

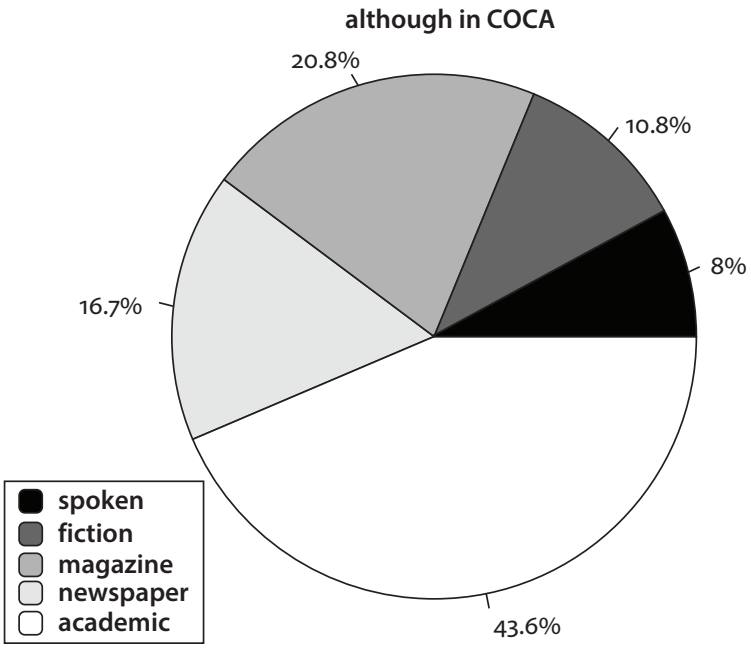

Figure A8. A pie chart of the distribution of *although* in the COCA registers

6 Some additional tips

– how to choose colours in R
To see the list of colours in R, type `colors()`. You will see a list with 657 colours that you can select for your plot. You can use colour names from the list, indices, hexadecimal or RGB values (see `help(rgb)`). When you want to use many colours, it may be convenient to use one of the ready-made palettes, such as `rainbow`, `heat`, `terrain`, `gray`, etc. See `help(rainbow)` for more details. Below is an illustration of how one can make the pie chart from the previous section colourful by using the rainbow palette:

```
> pie(although, col = rainbow(5), labels = although_percent,
main = "although in COCA")
> legend("bottomleft", fill = rainbow(5), legend = c("spoken",
"fiction", "magazine", "newspaper", "academic"))
```

`rainbow(5)` specifies the palette and the number of colours on the palette.
The result is shown in Figure A9.

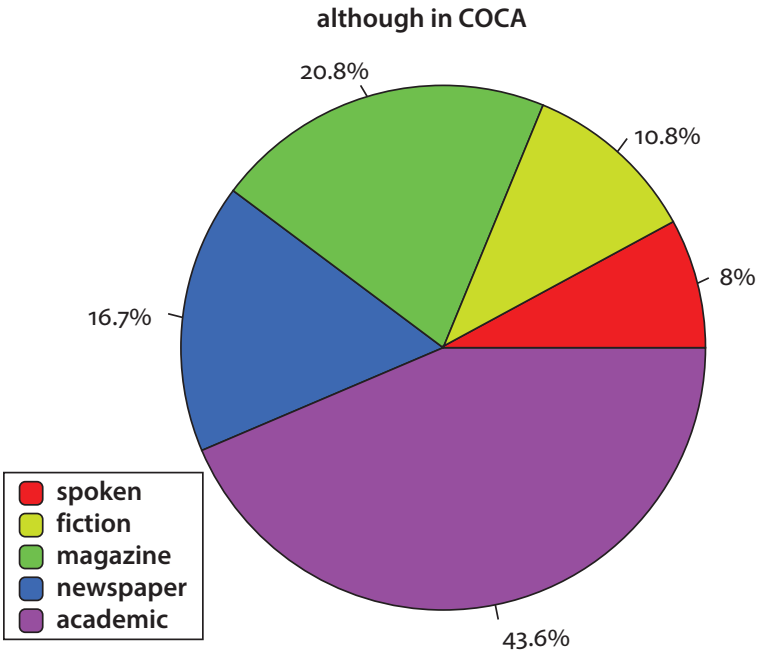

Figure A9. A pie chart with `rainbow` colours

– how to plot two or more graphs next to one another

```
> x <- 1:3
> par(mfrow = c(1, 2))
> barplot(sort(x, decreasing = TRUE))
> barplot(x)
```

This will allow two graphs to be plotted next to each other. See Figure A10.

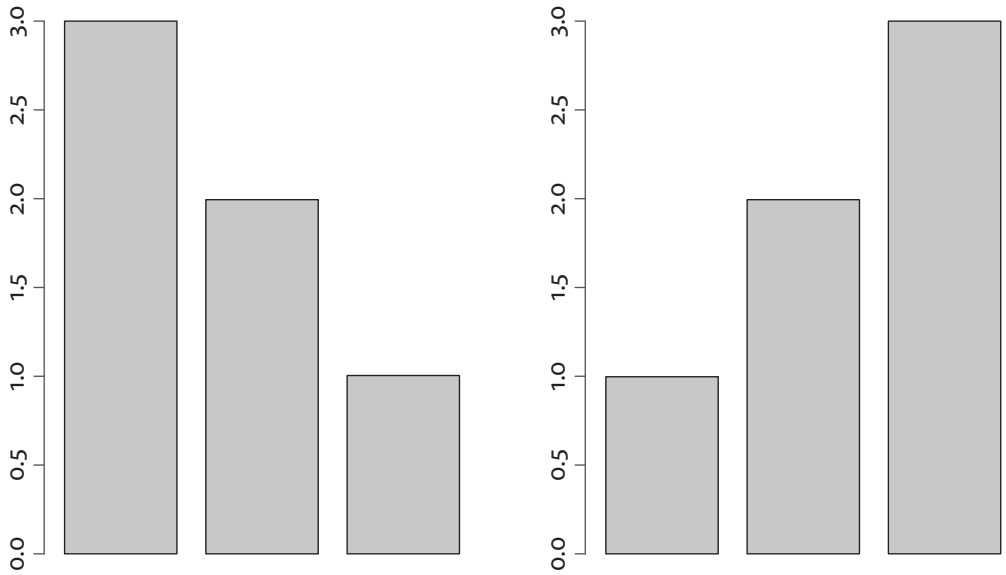

Figure A10. Two bar plots next to each another

```
> par(mfrow = c(5, 10))
> barplot(x)
… [repeat 49 times]
```

This will create fifty plots arranged in five rows and ten columns (see Figure A11).

Note that the layout will be reproduced until you change it explicitly (e.g. `par(mfrow = c(1, 1))` or close the graphical device, which will restore the default settings.

– how to use the arguments `pch` and `cex`
  The former argument specifies the type of point symbol. The top row in Figure A12 shows possible values of `pch` from `pch = 0` to `pch = 20` and the symbols that they produce. The argument `cex` specifies the size of symbols and text labels. The default value is 1. The bottom row in Figure A12 shows a few examples of different values from `cex = 1` to `cex = 5`. Note that you can use decimals, as well, e.g. `cex = 0.8`.

7 The structure of `ggplot2` functions

Creating graphs with `ggplot2` is different from using other graphical utilities in R. The package is based on Leland Wilkinson's grammar of graphics. This grammar is quite complex, and this section only provides a basic introduction into the most common options. In `ggplot2`, a plot is built up layer by layer. These layers are as follows:

Figure A11. Fifty bar plots arranged in five rows and ten columns

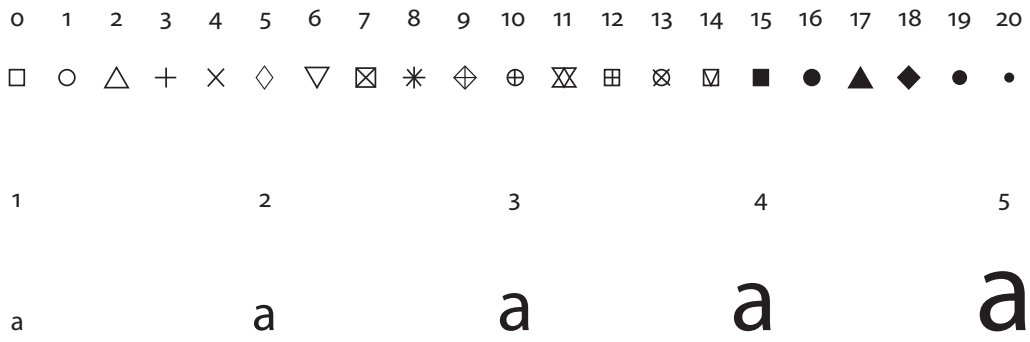

Figure A12. Top: different values of `pch`. Bottom: different values of `cex`

- `ggplot()` initiates a new plot. Here you normally declare the data frame and specify the values of *x* and *y* (when necessary) by using `ggplot(df, aes(x = Var1, y = Var2))`. The package also offers `qplot()`, a quick plot whose syntax resembles the basic R plotting functions.
- 'geoms', the layer that specifies the type of the plot. Some of the most popular geoms are as follows:
  - `geom_bar()` creates bar plots
  - `geom_boxplot()` creates box-and-whisker plots
  - `geom_errorbar()` plots error bars
  - `geom_histogram()` creates histograms
  - `geom_density()` creates density plots
  - `geom_line()` draws lines that connect ordered *x*-values
  - `geom_point()` creates scatter plots
  - `geom_text()` adds text annotations

  Here you can specify additional graphical parameters, e.g. `geom_bar(fill = "white", colour = "black", width = 0.5)` will result in a bar plot with white bars with black outline and the width of 0.5 units. One can also override the default way the plot statistics are computed, e.g. `geom_bar(stat = "identity")` instead of the default `stat = "bin"` will use the bar heights provided by the user. In this case, the heights should be specified as the *y*-variable in `ggplot(df, aes(x = Var1, y = Var2))`.
- statistical transformations, which should be specified if they are different from the default ones in the geoms.
  - `stat_bin()` is similar to `table()`. It counts the number of observations in bins (e.g. in each factor level). This is the default statistics for `geom_histogram()`. Here you can specify the bin width, e.g. `stat_bin(binwidth = 0.2)`.
  - `stat_boxplot()` summarizes the data for a box-and-whisker plot. This is the default transformation for `geom_boxplot()`.

  - `stat_density()` computes 1D kernel density estimates, which are used in `geom_density()`.
  - `stat_identity()` makes no transformations and plots the data as is.
- 'aesthetics' `aes()` specifies the mapping of variables to different parts of the plot, e.g. `aes(x = Var1 and y = Var2)`
- coordinate systems
  - `coord_flip()` flips the *x*- and *y*-axis
  - `coord_polar()` uses the polar coordinate system, which is useful for creation of pie charts
- position adjustments
  - `position_dodge()` is used for creation of plots with grouped (unstacked) bars
  - `position_jitter()` is used to add some jitter to avoid overplotting
- title and axes
  - `ggtitle("Title")` provides a title for the plot
  - `labs(x = "Label of x", y = "Label of y")` provides the labels for *x* and *y*
  - `xlim()`, `ylim()` specify the limits of the axes, as in conventional R plots
- `ggsave()` saves the last displayed plot (by default). You need to supply the file name, e.g. `ggsave(file = "YourDirectory/YourPlot.png)`.

These elements are concatenated, e.g. `ggplot(df, aes(x = Var1)) + geom_bar() + coord_flip()`. See the examples in the book. For more details, see Chang (2013), Wickham (2009) and http://docs.ggplot2.org/.