Chapter 5
An Introduction to Basic Plotting Tools

We have demonstrated the use of R tools for importing data, manipulating data, extracting subsets of data, and making simple calculations, such as mean, variance, standard deviation, and the like. In this chapter, we introduce basic graph plotting tools. If you are interested in only simple graphing, this chapter will suffice; however, to construct more sophisticated graphs, or to add more complicated embellishments such as tick marks, or specialized fonts and font sizes, to basic graphs, you will need the more advanced plotting techniques presented in Chapters 7 and 8.

A discussion of elementary plotting tools may seem out of place at this stage, rather than being included in the sections on graphing beginning with Chapter 7. However, when teaching the material presented in this book, we became aware that, after discussing the relatively pedestrian material of the first four sections, the course participants were eagerly awaiting the lively, more visual, and easier, plotting tools. Therefore, we present a first encounter with graphing here, which allows the presentation of the more complex subjects in the next chapter with the aid of active tools such as the `plot` function.

5.1 The `plot` Function

This section uses the vegetation data introduced in Chapter 4. Recall that these are grassland data from a monitoring program conducted in two temperate communities in Yellowstone National Park and the National Bison Range, USA. To quantify biodiversity, species richness was calculated. In a statistical analysis, we may want to model richness as a function of `BARESOIL` (or any of the other soil and climate variables). Suppose we want to make a plot of species richness versus the substrate variable "exposed soil," denoted by `BARESOIL`. The R commands to create such a graph is

```
> setwd("c:/RBook/")
> Veg <- read.table(file = "Vegetation2.txt",
                    header = TRUE)
> plot(Veg$BARESOIL, Veg$R)
```

A.F. Zuur et al., *A Beginner's Guide to R*, Use R,
DOI 10.1007/978-0-387-93837-0_5, © Springer Science+Business Media, LLC 2009

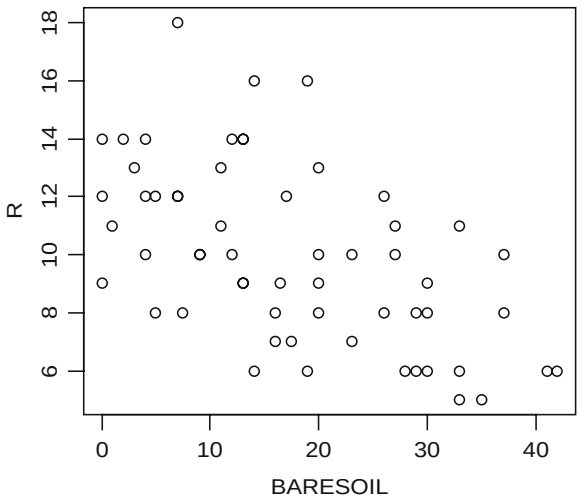

Fig. 5.1 Graph showing a scatterplot of `BARESOIL` versus species richness for the vegetation data. To import a graph from R into Microsoft Word, right-click on the graph in R to copy and paste it into Word, or save it as a metafile (recommended, as it produces a good quality graph), bitmap, or postscript file. This will also work with non-Windows operating systems. Importing the latter two formats into Word is more complicated. If the R console window (Fig. 1.5) is maximised, you may not see the panel with the graph. To access it, press Control-Tab, or minimise the R console

The resulting graph is presented in Fig. 5.1. The first argument of the `plot` command is displayed on the horizontal axis with the second argument along the vertical axis. Richness, in this case, is the response, or dependent, variable, and `BARESOIL` is the explanatory, or independent, variable. It is conventional to plot the response variable along the vertical axis and the explanatory variable along the horizontal axis. Be aware that for some statistical functions in R, you must specify the response variable first, followed by the explanatory variables. You would not be the first to accidentally type

```
> plot(Veg$R, Veg$BARESOIL)
```

and discover that the order should have been reversed. Alternatively, you can use

```
> plot(x = Veg$BARESOIL, y = Veg$R)
```

to avoid confusion over which variables will be plotted on the x-axis (horizontal) and which on the y-axis (vertical).

The `plot` function does have a `data` argument, but we cannot use

```
> plot(BARESOIL, R, data = Veg)
Error in plot(BARESOIL, R, data = Veg): object
"BARESOIL" not found
```

This is unfortunate, and forces us to use Veg$ in the command (recall from Chapter 2 that we do not use the attach command). This is one of the reasons (if not the only one) that we chose the variable name Veg instead of the longer, Vegetation.It is also possible to use:

```
> plot(R ~ BARESOIL, data = Veg)
```

This does produce a graph (not shown here), but our objection against this notation is that in some functions, the R ~ BARESOIL notation is used to tell R that richness is modelled *as a function of* baresoil. Athough that may be the case here, not every scatterplot involves variables that have a cause–effect relationship.

The most common modifications to any graph are adding a title and *x*- and *y*-labels and setting the *x*- and *y*-limits, as with the graph in Fig. 5.1. This is accomplished by extending the plot command:

```
> plot(x = Veg$BARESOIL, y = Veg$R,
     xlab = "Exposed soil",
     ylab = "Species richness", main = "Scatter plot",
     xlim = c(0, 45), ylim = c(4, 19))
```

The resulting graph is presented in Fig. 5.2A. The four panels (A–D) were imported into the text document (Microsoft Word) by using the text editor to construct a 2-by-2 table with no borders. (Using R to create a graph with multiple panels is demonstrated in Chapter 8.)

The order in which xlab, ylab , main, xlim, and ylim are entered is irrelevant, but they must be in lowercase letters. The xlab and ylab options are used for labels and the main option for the title. The xlim and ylim options are used to specify the lower and upper limits on the axes. You can also use

```
xlim = c(min(Veg$BARESOIL), max(Veg$BARESOIL))
```

within the plot command, but, if there are missing values in the plotted variable, you should extend the min and max functions with the na.rm = TRUE option. This produces

```
xlim = c(min(Veg$BARESOIL, na.rm = TRUE),
         max(Veg$BARESOIL, na.rm = TRUE))
```

In Chapters 7 and 8, we demonstrate changing the style and size of the font used for the labels and title, and adding symbols, such as, ^{0}C, and so on.

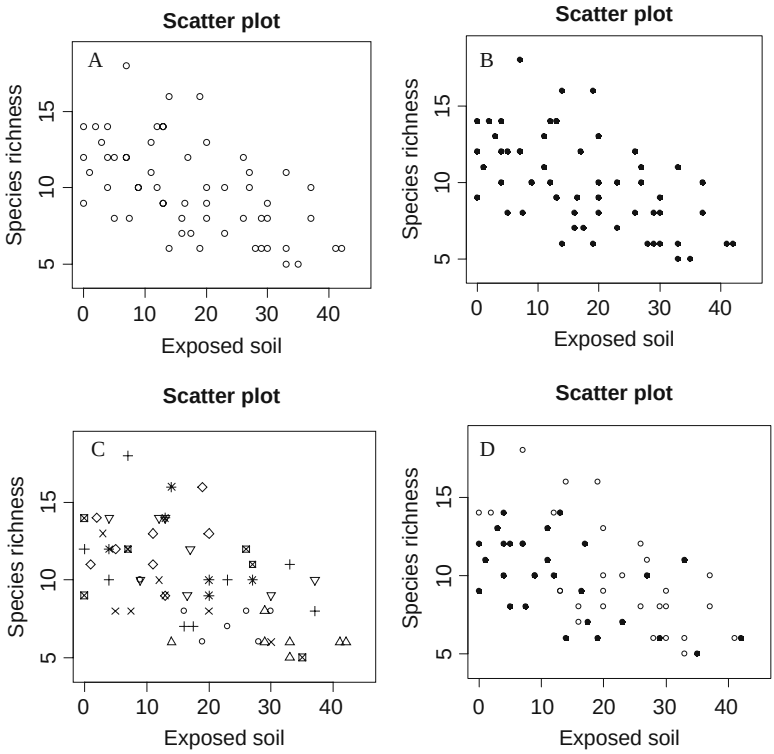

Fig. 5.2 Examples of various plotting options. **A**: Scatterplot of species richness versus exposed soil (`BARESOIL`). **B**: The same scatterplot as in A, with observations plotted using a *filled circle* (*or dot*). **C**: The same scatterplot as in A, with observations from each transect indicated by a different symbol. **D**: The same scatterplot as in A, but with observations measured before 1975 plotted as *open circles* and those after 1975 as *filled circles*

5.2 Symbols, Colours, and Sizes

During our courses, the most frequently asked questions concerning graphs are whether (1) the plotting symbols can be changed, (2) different colours can be used, and (3) the size of the plotting symbols can be varied conditional on the values of another variable. We discuss these three options in this section and leave the more sophisticated modifications such as altering tick marks, and adding subscripts and superscripts, among other things, for Chapters 7 and 8.

5.2.1 Changing Plotting Characters

By default, the `plot` function uses open circles (open dots) as plotting characters, but characters can be selected from about 20 additional symbols. The plotting character is specified with the `pch` option in the `plot` function; its

default value is 1 (which is the open dot or circle). Figure 5.3 shows the symbols that can be obtained with the different values of `pch`. For solid points the command is `pch =16`. As an example, the following code produces Fig. 5.2B, in which we replaced the open dots with filled dots.

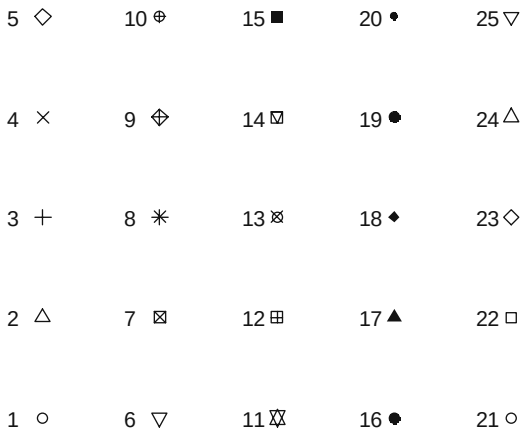

Fig. 5.3 Symbols that can be obtained with the `pch` option in the `plot` function. The number left of a symbol is the `pch` value (e.g., `pch = 16` gives •)

```
> plot(x = Veg$BARESOIL, y = Veg$R,
     xlab = "Exposed soil",
     ylab = "Species richness", main = "Scatter plot",
     xlim = c(0, 45), ylim = c(4, 19), pch = 16)
```

In Fig. 5.2A, B, all observations are represented by the same plotting symbol (the open circles in panel A were obtained with the default, `pch = 1`, and the closed circles in panel B with `pch = 16`).

The grassland data were measured over the course of several years in eight transects. It would be helpful to add this information to the graph in Fig. 5.2A. At this point, the flexibility of R begins to emerge. Suppose you want to use a different symbol for observations from each transect. To do this, use a numerical vector that has the same length as `BARESOIL` and richness `R` and contains the value 1 for all observations from transect 1, the value 2 for all observations from transect 2, and so on. Of course it is not necessary to use 1, 2, and so on. The values can be any valid `pch` number (Fig. 5.3). You only need to ensure that, in the new numerical vector, the values for observations within a single transect are the same and are different from those of the other transects. In this case you are lucky; the variable `Transect` is already coded with numbers 1 through 8 designating the eight transects. To see this, type

```
> Veg$Transect
 [1] 1 1 1 1 1 1 1 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3
[23] 4 4 4 4 4 4 4 4 5 5 5 5 5 5 5 5 6 6 6 6 6 6
[45] 6 6 7 7 7 7 7 7 8 8 8 8 8 8 8
```

Thus, there is no need to create a new vector; you can use the variable Transect (this will not work if Transect is defined as a factor, see below):

```
> plot(x = Veg$BARESOIL, y = Veg$R,
    xlab = "Exposed soil", ylab = "Species richness",
    main = "Scatter plot", xlim = c(0, 45),
    ylim = c(4, 19), pch = Veg$Transect)
```

The resulting graph is presented in Fig. 5.2C. It shows no clear transect effect. It is not a good graph, as there is too much information, but you have learned the basic process.

There are three potential problems with the pch = Transect approach:

1. If Transect had been coded as 0, 1, 2, and so on, the transect for which pch = 0 would not have been plotted.
2. If the variable Transect did not have the same length as BARESOIL and richness R, assume it was shorter; R would have repeated (iterated) the first elements in the vector used for the pch option, which would obviously produce a misleading plot. In our example, we do not have this problem, as BARESOIL, richness, and transect have the same length.
3. In Chapter 2, we recommended that categorical (or nominal) variables be defined as such in the data frame using the factor command. If you select a nominal variable as the argument for pch, R will give an error message. This error message is illustrated below:

```
> Veg$fTransect <- factor(Veg$Transect)
> plot(x = Veg$BARESOIL, y = Veg$R,
      xlab = "Exposed soil",
      ylab = "Species richness", main = "Scatter plot",
      xlim = c(0, 45), ylim = c(4, 19),
      pch = Veg$fTransect)
Error in plot.xy(xy, type, ...): invalid plotting symbol
```

On the first line of the R code above, we defined fTransect as a nominal variable inside the Veg data frame, and went on to use it as argument for the pch option. As you can see, R will not accept a factor as pch argument; it must be a numerical vector.

5.2.1.1 Use of a Vector for pch

The use of a vector forpch (and for the col and cex options discussed later) can be confusing.

The vegetation data were measured in 1958, 1962, 1967, 1974, 1981, 1989, 1994, and 2002. We arbitrarily selected an open circle to represent observations measured from 1958 to 1974 and a filled circle for those made after 1974. Obviously, the option pch = Veg$Time is out of the question, as it tries to

use eight different symbols, and, also, the pch value 1958 (or of any year) does not exist. We must create a new numerical vector of the same length as Veg $Time, using the value 1 when Time is 1958, 1962, 1967, and 1974 and 16 for the more recent years. The values 1 and 16 were chosen because we like open and filled circles as they show a greater contrast than other combinations. Here is the R code (you can also do this in one line with the ifelse command):

```
> Veg$Time2 <- Veg$Time
> Veg$Time2 [Veg$Time <= 1974] <- 1
> Veg$Time2 [Veg$Time > 1974] <- 16
> Veg$Time2
 [1]  1 1 1  1 16 16 16  1 1 1  1  6 16 16  1
[16]  1 1 1 16 16 16 16  1 1 1  1 16 16 16 16
[31]  1 1 1  1 16 16 16 16 1 1  1  1 16 16 16
[46] 16 1 1  1 16 16 16  1 1 1 16 16 16
```

The first command creates a new numerical vector of the same length as Veg $Time, and the following two commands allocate the values 1 and 16 to the proper places. The rest of the R code is easy; simply use Veg $Time2 as the pch option. The resulting graph is presented in Fig. 5.2D:

```
> plot(x = Veg$BARESOIL, y = Veg$R,
     xlab = "Exposed soil",
     ylab = "Species richness", main = "Scatter plot",
     xlim = c(0, 45), ylim = c(4, 19),
     pch = Veg$Time2)
```

In the text above, we mentioned that you should not use pch = Veg $Time as Time contains values that are not valid pch commands. The use of Veg $Time will result in

```
> plot(x = Veg$BARESOIL, y = Veg$R,
      xlab = "Exposed soil",
      ylab = "Species richness", main = "Scatter plot",
      xlim = c(0, 45), ylim = c(4, 19),
      pch = Veg$Time)

There were 50 or more warnings (use warnings() to see
the first 50)

> warnings()

Warning messages:
1: In plot.xy(xy, type, ...) : unimplemented pch value
' 1958'
```

```
2: In plot.xy(xy, type, ...) : unimplemented pch value
' 1962'
3: In plot.xy(xy, type, ...) : unimplemented pch value
' 1967'
4: In plot.xy(xy, type, ...) : unimplemented pch value
' 1974'
5: In plot.xy(xy, type, ...) : unimplemented pch value
' 1981'
....
```

We typed `warnings ()` as instructed by R. The warning message speaks for itself.

To learn more about the `pch` option, look at the help file of the function `points`, obtained with the `?points` command.

5.2.2 *Changing the Colour of Plotting Symbols*

The plotting option for changing colours is useful for graphics presented on a screen or in a report, but is less so for scientific publications, as these are most often printed in black and white. We recommend that you read Section 5.2.1 before reading this section, as the procedure for colour is the same as that for symbols.

To replace the black dots in Fig. 5.2 with red, use

```
> plot(x = Veg$BARESOIL, y = Veg$R,
     xlab = "Exposed soil",
     ylab = "Species richness", main = "Scatter plot",
     xlim = c(0, 45), ylim = c(4, 19),
     col = 2)
```

For green, use `col =3`. Run the following code to see the other available colours.

```
> x <- 1:8
> plot(x, col = x)
```

We do not present the results of the two commands as this book is without colour pages. In fact, there are considerably more colours available in R than these eight. Open the `par` help file with the `?par` command, and read the "Color Specification" section near the end. It directs you to the function `colors` (or `colours`), where, apparently, you can choose from hundreds.

5.2.2.1 Use of a Vector for `col`

You can also use a vector for the `col` option in the `plot` function. Suppose you want to plot the observations from 1958 to 1974 as black filled squares and the observations from 1981 to 2002 as red filled circles (shown here as light grey). In the previous section, you learned how to create filled squares and circles using the variable `Time2` with values 15 (square) and 16 (circle). Using two colours is based on similar R code. First, create a new variable of the same length as `BARESOIL` and richness `R`, which can be called `Col2` . For those observations from1958 to 1974, `Col2` takes the value 1 (= black) and, for the following years, 2 (= red). The R code is

```
> Veg$Time2 <- Veg$Time
> Veg$Time2 [Veg$Time <= 1974] <- 15
> Veg$Time2 [Veg$Time > 1974] <- 16
> Veg$Col2 <- Veg$Time
> Veg$Col2 [Veg$Time <= 1974] <- 1
> Veg$Col2 [Veg$Time > 1974] <- 2
> plot(x = Veg$BARESOIL, y = Veg$R,
     xlab = "Exposed soil",
     ylab = "Species richness", main = "Scatter plot",
     xlim = c(0, 45), ylim = c(4, 19),
     pch = Veg$Time2, col = Veg$Col2)
```

The resulting graph is presented in Fig. 5.4A. The problems that were outlined for the `pch` option also apply to the `col` option. If you use `col =0`, the observations will not appear in a graph having a white background; the vector with values for the colours should have the same length as `BARESOIL` and richness `R;` and you must use values that are linked to a colour in R.

Before expending a great deal of effort on producing colourful graphs, it may be worth considering that, in some populations, 8% of the male population is colourblind!

5.2.3 Altering the Size of Plotting Symbols

The size of the plotting symbols can be changed with the `cex` option, and again, this can be added as an argument to the `plot` command. The default value for `cex` is 1. Adding `cex =1.5` to the plot command produces a graph in which all points are 1.5 times the default size:

```
> plot(x = Veg$BARESOIL, y = Veg$R,
     xlab = "Exposed soil", ylab = "Species richness",
     main = "Scatter plot",
     xlim = c(0, 45), ylim = c(4, 19),
     pch = 16, cex = 1.5)
```

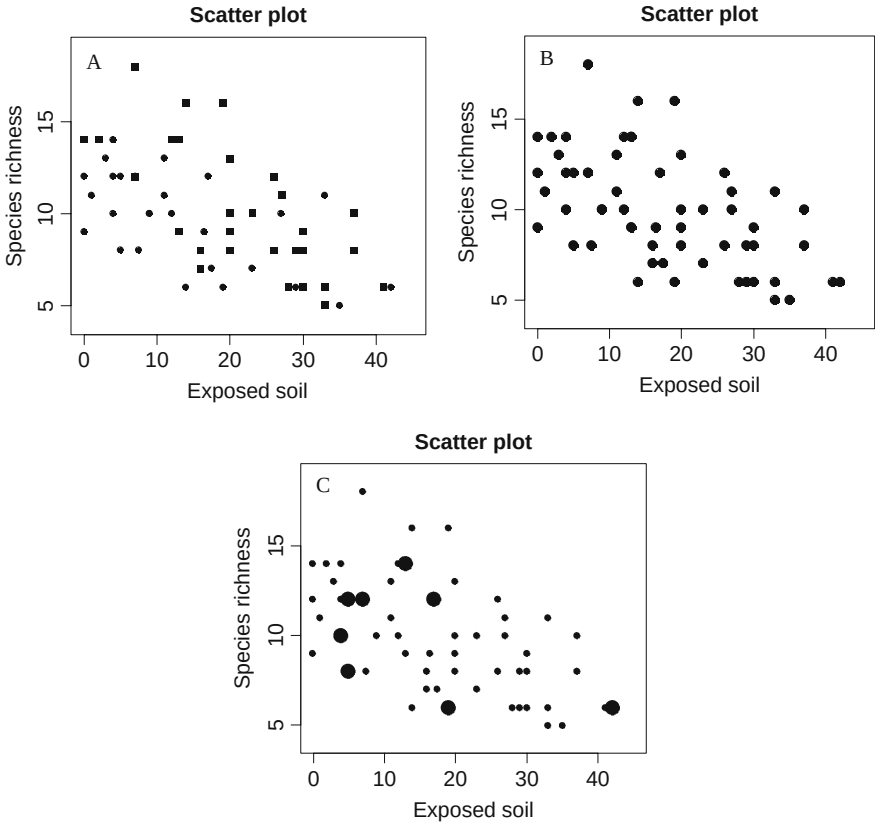

Fig. 5.4 Examples of various plot commands. **A**: Scatterplot of species richness versus BARESOIL. Observations from 1958 to 1974 are represented as filled squares in black and observations from 1981 to 2002 as *filled circles* in red. Colours were converted to greyscale in the printing process. **B**: The same scatterplot as in Fig. 5.2A, with all observations represented as black filled dots 1.5 times the size of the dots in Fig. 5.2A. **C**: The same scatterplot as in Fig. 5.2A with observations from 2002 represented by dots twice those of Fig. 5.2A

We used filled circles. The resulting graph is presented in Fig. 5.4B.

5.2.3.1 Use of a Vector for cex

As with the pch and col options, we demonstrate the use of a vector as the argument of the cex option. Suppose you want to plot BARESOIL against species richness using a large filled dot for observations made in 2002 and a smaller filled dot for all other observations. Begin by creating a new vector with values of 2 for observations made in 2002 and 1 for those from all other years. The values 1 and 2 are good starting points for finding, through trial and error, the optimal size difference. Try 3 and 1, 1.5 and 1, or 2 and 0.5, and so on, and decide which looks best.

```
> Veg$Cex2 <- Veg$Time
> Veg$Cex2[Veg$Time == 2002] <- 2
> Veg$Cex2[Veg$Time != 2002] <- 1
```

Using the vector `Cex2` , our code can easily be adjusted:

```
> plot(x = Veg$BARESOIL, y = Veg$R,
     xlab = "Exposed soil", ylab = "Species richness",
     main = "Scatter plot",
     xlim = c(0, 45), ylim = c(4, 19),
     pch = 16, cex = Veg$Cex2)
```

The resulting graph is presented in Fig. 5.4C. Altering the symbol size can also be accomplished by using `cex =1.5 * Veg$Cex2` or `cex =Veg $Cex2 /2`.

5.3 Adding a Smoothing Line

It is difficult to see a pattern in Fig. 5.1. The information that you want to impart to the viewer will become clearer if you add a smoothing curve[1] to aid in visualising the relationship between species richness and `BARESOIL`. The underlying principle of smoothing is not dealt with in this book, and we refer the interested reader to Hastie and Tibshirani (1990), Wood (2006), or Zuur et al. (2007).

The following code redraws the plot, applies the smoothing method, and superimposes the fitted smoothing curve over the plot, through the use of the `lines` command.

```
> plot(x = Veg$BARESOIL, y = Veg$R,
     xlab = "Exposed soil", ylab = "Species richness",
     main = "Scatter plot", xlim = c(0, 45),
     ylim = c(4, 19))
> M.Loess <- loess(R ~ BARESOIL, data = Veg)
> Fit <- fitted(M.Loess)
> lines(Veg$BARESOIL, Fit)
```

The resulting graph is presented in Fig. 5.5A. The command

[1] A smoothing curve is a line that follows the shape of the data. For our purposes, it is sufficient to know that a smoothing curve serves to capture the important patterns in, or features of, the data.

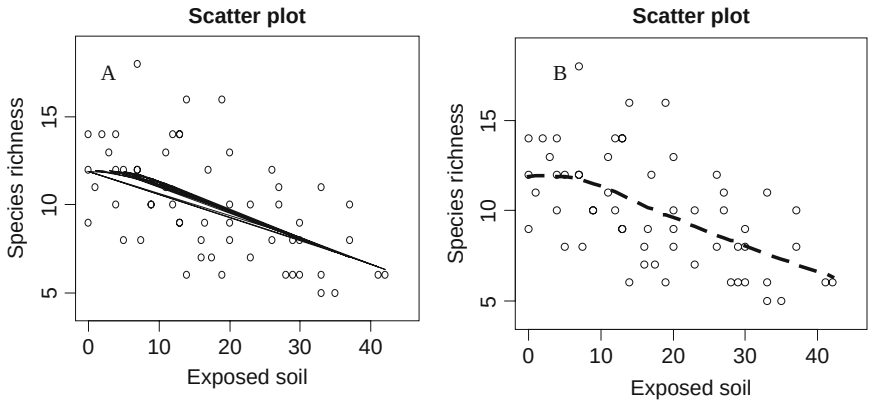

Fig. 5.5 **A**: The same scatterplot as in Fig. 5.2A, with a smoothing curve. Problems occur with the `lines` command because `BARESOIL` is not sorted from low to high. **B**: The same scatterplot as in Fig. 5.2A, but with a properly drawn smoothing curve

```
> M.Loess <- loess(R ~ BARESOIL, data = Veg)
```

is the step that applies the smoothing method, and its output is stored in the object `M.Loess` . To see what it comprises, type:

```
> M.Loess

Call:
loess(formula = R ~ BARESOIL, data = Veg)
Number of Observations: 58
Equivalent Number of Parameters: 4.53
Residual Standard Error: 2.63
```

That is not very useful. `M.Loess` contains a great deal of information which can be extracted through the use of special functions. Knowing the proper functions and how to apply them brings us into the realm of statistics; the interested reader is referred to the help files of `resid`, `summary`, or `fitted` (and, obviously, `loess`).

The notation `R` ~`BARESOIL` means that the species richness `R` is modelled as a function of `BARESOIL`. The `loess` function allows for various options, such as the amount of smoothing, which is not discussed here as it brings us even further into statistical territory. As long as we do not impose further specifications on the `loess` function, R will use the default settings, which are perfect for our purpose: drawing a smoothing curve.

The output from the `loess` function, `M.Loess` , is used as input into the function, `fitted`. As the name suggests, this function extracts the fitted values, and we allocate it to the variable `Fit`. The last command,

```
> lines(Veg$BARESOIL, Fit)
```

superimposes a line onto the plot that captures the main pattern in the data and transfers it onto the graph. The first argument goes along the x-axis and the second along the y-axis. The resulting plot is given in Fig. 5.5A. However, the smoothed curve is not what we expected, as the lines form a spaghetti pattern (multiple lines). This is because the `lines` command connects points that are sequential in the first argument.

There are two options for solving this problem. We can sort `BARESOIL` from small to high values and permute the second argument in the `lines` command accordingly, or, alternatively, we can determine the order of the values in `BARESOIL`, and rearrange the values of both vectors in the `lines` command. The second option is used below, and the results are given in Fig. 5.5B. Here is the R code.

```
> plot(x =  Veg$BARESOIL, y = Veg$R,
     xlab = "Exposed soil",
     ylab = "Species richness", main = "Scatter plot",
     xlim = c(0, 45), ylim = c(4, 19))
> M.Loess <- loess(R ~ BARESOIL, data = Veg)
> Fit <- fitted(M.Loess)
> Ord1 <- order(Veg$BARESOIL)
> lines(Veg$BARESOIL[Ord1], Fit[Ord1],
        lwd = 3, lty = 2)
```

The `order` command determines the order of the elements in `BARESOIL` , and allows rearranging of the values from low to high in the `lines` command. This is a little trick that you only need to see once, and you will use it many times thereafter. We also added two more options to the `lines` command, `lwd` and `lty`, indicating line width and line type. These are further discussed in Chapter 7, but to see their effect, change the numbers and note the change in the graph. Within the `lines` command, the `col` option can also be used to change the colour, but obviously the `pch` option will have no effect.

The smoothing function seems to indicate that there is a negative effect of `BARESOIL` on species richness.

5.4 Which R Functions Did We Learn?

Table 5.1 shows the R functions that were introduced in this chapter.

5.5 Exercises

Exercise 1. Use of the `plot` function using terrestrial ecology data. In Chapter 16 of Zuur et al. (2009), a study is presented analysing numbers of amphibians

Table 5.1 R functions introduced in this chapter

Function	Purpose	Example
`plot`	Plots *y* versus *x*	`plot(y, x, xlab="X label", xlim=c(0, 1), pch=1, main="Main", ylim=c(0, 2), ylab="Y label", col=1)`
`lines`	Adds lines to an existing graph	`lines(x, y, lwd=3, lty=1, col=1)`
`order`	Determines the order of the data	`order(x)`
`loess`	Applies LOESS smoothing	`M<-loess(y~x)`
`fitted`	Obtains fitted values	`fitted(M)`

killed along a road in Portugal using generalised additive mixed modelling techniques. In this exercise, we use the `plot` command to visualise a segment of the data. Open the file *Amphibian_road_Kills.xls*, prepare a spreadsheet, and import the data into R.

The variable, TOT_N, is the number of dead animals at a sampling site, OLIVE is the number of olive groves at a sampling site, and D Park is the distance from each sampling point to the nearby natural park. Create a plot of TOT_N versus D_park. Use appropriate labels. Add a smoothing curve. Make the same plot again, but use points that are proportional to the value of OLIVE (this may show whether there is an OLIVE effect).