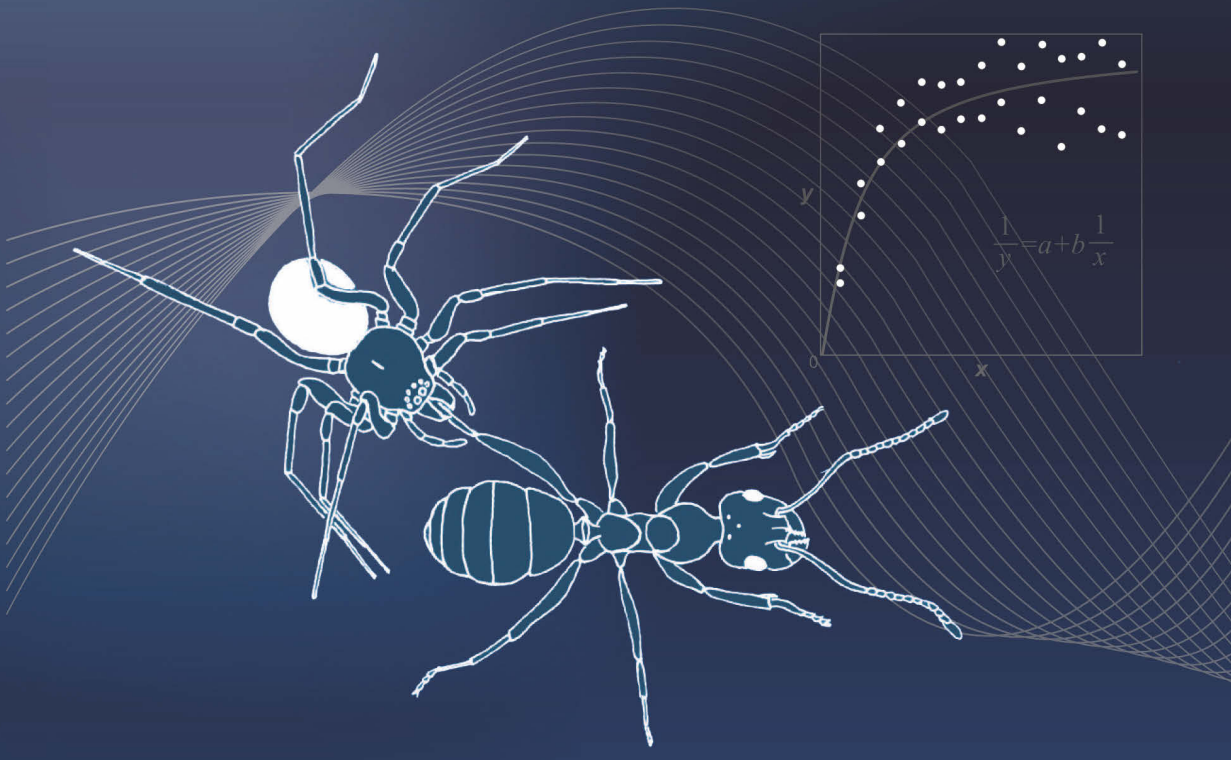
MODERN ANALYSIS OF BIOLOGICAL DATA

GENERALIZED LINEAR MODELS IN R



STANO PEKÁR
MAREK BRABEC

muni PRESS

MODERN ANALYSIS OF BIOLOGICAL DATA
GENERALIZED LINEAR MODELS IN R

STANO PEKÁR, MAREK BRABEC

muni
PRESS

MODERN ANALYSIS OF BIOLOGICAL DATA

GENERALIZED LINEAR MODELS IN R

STANO PEKÁR
MAREK BRABEC

Masaryk University, Brno 2016

http://www.muni.cz/press/books/pekar_en

Pekár S. & Brabec M. 2016. Modern Analysis of Biological Data:
Generalized Linear Models in R. Masaryk University Press, Brno.

This book was supported by Masaryk University Project No. MUNI/FR/1304/2014.

ISBN 978-80-210-8106-2 (online : pdf)
ISBN 978-80-210-8019-5 (Paperback)

CONTENTS

FOREWORD

This book is meant especially for students and scholars of biology, i.e. biologists who work in natural science, at agricultural, veterinary, pharmaceutical and medical faculties or at research institutes of a similar orientation. It has been written for people who have only a basic knowledge of statistics (for example, people who have attended only a Basic statistics/Biostatistics course) but who need to correctly analyse the data resulting from their observations or experiments.

The generally negative attitude of biologists towards mathematics is well known. It is precisely why we have tried to write the book in a relatively simple style – with minimal mathematical requirements. Sometimes, the task turned out to be easy, other times not that easy, and sometimes it became almost impossible. That is why there are still some mathematical equations in almost all chapters of the book (even though they are used in a simplified form in order to be more apprehensible to less experienced readers). Despite this fact, the book includes much less mathematical and statistical theories than it is common for standard statistical literature.

The book is mainly built on examples of real data analyses. They are presented from the very beginning to the end, from a description and determination of objectives and assumptions to study conclusions. They thus simulate (even though in a simplified way) the procedure usually used when preparing a paper for a scientific journal. We believe that practical experience with data analyses is irreplaceable. Because of the anticipated biology-oriented readers, we selected examples from the areas of ecology, ethology, toxicology, physiology, zoology and agricultural production. All these data were analysed during various previous research projects. They have been adjusted in order to suit the pedagogical intentions of this book. For example, the original long and complex Latin names of species have been replaced with a generic short name (e.g., specA).

Finally, we would like to thank all our colleagues without whose help this book would never have been written. First of all, we would like to thank to Vojtěch Jarošík (in memoriam), for introducing GLM to the first author of the book during his studies at the university, thus igniting his interest in statistics generally; Alois Honěk for many consultations, and our colleagues from the Crop Research Institute in Prague-Ruzyně and the students of the Faculty of Science of the Masaryk University in Brno for inspiring comments to the original text of the book. Finally, we would also like to thank the following colleagues of ours who have kindly let us use their (though adjusted) data for presenting examples in this book: T. Bilde, A. Honěk, J. Hubert, D. Chmelař, J. Lipavský, M. Řezáč, P. Saska and V. Stejskal.

We welcome any comments regarding the text and content of the book. Please direct them to the following email addresses: pekar@sci.muni.cz and/or mbrabec@cs.cas.cz.

December 2015

Stano Pekár
Marek Brabec

1 INTRODUCTION

Let us start with a demonstration of two standard situations. The first one took place after a thesis defence when a student complained to another student: "Supposedly I used the wrong statistical test." Other situations can occur, for example, in a hallway of a research institute where a biologist reproaches a colleague for the way he/she presented his/her results: "The data should be analysed more properly." Both situations have one thing in common – a desperate reference to the statistics. Indeed, statistical data analysis forms an integral part of scientific publications in many biological (and other) fields, thus accompanying bio-logists throughout their careers. In some fields, such as taxonomy, statistical analyses may play just a marginal role. For other fields, such as ecology or physiology, it is often almost a cornerstone of many new findings. In these fields, shortcomings in statistical analysis can have catastrophic consequences. It can easily happen that a report on a good experimental study (e.g., in the form of a scientific paper) will not be complete without a statistical analysis, in which case the results of the study can become completely useless and all the previous effort of its authors can thus be wasted.

The only way to prevent such disasters is to strive to understand the statistics or, at least, to find somebody who understands it. Obviously, conducting practical data analyses are much easier today than ever before. This is due to the development of personal computers and subsequent developments in computational algorithms, which have literally meant a revolution for data analyses. Fifty years ago, even a simple statistical analysis, using a calculator and a pen, would take several hours and sometimes even days (while, at the same time, it was not easy to avoid calculation errors etc.). Today, using computers, even a relatively complicated analysis can take less than a minute or even just a few milliseconds. Preparing a plot is often easier and faster than, for example, preparing a coffee. However, technical improvements have led to increased demands for using adequate statistical methods. While simpler statistical methods were preferred in the past, despite the fact they were not the most suitable, today the emphasis is put on using methods that correspond in the relevant aspects to the actual data at hand as closely as possible. This is because the computing requirements do not represent an insurmountable obstacle any more. Unlike the former "universal" simple procedures, application of statistics today utilises such methods and models that realistically consider the important characteristics of the data and of the studied processes. Very often this means that a given method or a statistical model can and should be adjusted to the real situation at hand. In short, models should be adjusted to the data and not vice versa!

Nevertheless, it is obviously not always easy to comply with this requirement. In fact, creative and useful practical analyses need theoretical knowledge about numerous models and methods as a pre-requisite. Moreover, certain experience with practical data analyses, with

the application of various models on real data, with model building as well as estimation procedures, with conducting appropriate tests, etc. is also necessary. It is clear that one cannot become an experienced data analyst only from reading books. To get such experience, you have to put in some work and, most of all, some thinking. Guidance and examples can assist you in this process.

This book attempts to help exactly along these lines by presenting examples of particular analyses, including the specification of the problem of interest, development of a statistical model and formulation of possible conclusions. The book is not, and does not even aspire to be, a manual for selecting the "best" method for a particular data analysis (it is our strong opinion that, for various reasons, such a manual cannot be ever made). Instead, the book tries to demonstrate how to think about particular statistical models, how to use them and also how not to use them – to point out many of the problems and errors that can occur (and do indeed occur in practical analyses and even in published papers). We demonstrate various general approaches on particular (hopefully biologist-engaging) examples and we also present their detailed implementation in the R language, thus allowing everybody to try them for themselves using the data provided in the book as well as their own datasets of a similar nature.

Since regression is a very powerful instrument, used very often in biological studies, this book is almost exclusively dedicated to regression models. It assists in solving important questions of the following type: how does variable y depend on variable x or how can we predict the value of a new observation when we know the value of one or several so-called explanatory variables. However, we will talk about regression in a somewhat wider context compared to what you know from basic statistics courses. Namely, we will use the concept of the so-called Generalized Linear Models (GLM). Their name explicitly emphasises that they are generalized linear regression models – that is, generalisations of models that many people know under the term of general linear model. The GLM generalization is very useful from a practical point of view since it allows correct analyses of various data that cannot be processed using a standard (linear) regression without (very) rough approximations and (crude) simplifications. GLM represents a relatively large class of models with a unified statistical theory and very universal computational implementations. It is a class by the means of which you can analyse a wide range of data, such as weight measurements, concentration, number of individuals, or relative frequency of a phenomenon (i.e. various continuous unrestricted, continuous positive, discrete positive data).

From a formal point of view, we will use only univariate GLM models (or statistical methods based on them), and only those that are suitable exclusively for independent measurements (observations). In order to analyse data with some kind of correlation (for example, in the case of repeated measurements on the same individuals), somewhat different and more complex models and methods need to be used. We address these in another book (Pekár & Brabec 2012).

As we have already stated, the objective of this book is to assist practical users of statistics in the process of formulating and using statistical models – and thus also in selecting suit-

able methods for analyses of various data types. At the same time, we would like to motivate them to seek assistance from a professional statistician if they find that the complexity of the model, study design and/or data complexity exceed their abilities and experience with statistical modelling. Things are not always as easy as they may appear at first. The same data can be analysed using various methods, depending, for example, on the objectives of a particular analyst. In fact, data include a large amount of information but we are often interested only in extracting just some part of it at a given moment. That is why the method selected depends on what we regard as salient features of the data (and what we are willing to leave out), as well as what (and for what purposes) we want to extract from the observations/measurements. Nevertheless, it is often the case that multiple procedures (with different characteristics) can be used for analysing even a single aspect. In this book, we will mostly present only one procedure without denying the existence of other approaches. Otherwise, the book would become significantly longer.

We have applied a similar approach when describing the R language and discussing the use of particular objects. The R language is, in fact, very rich. Like in any other programming language environment, the same result can be often obtained in several ways. There is not enough room here to list them all (if you are interested, you can consult the corresponding manuals, e.g. Zoonekynd 2007). We chose those that we consider to be the easiest and most practical for a beginner.

1.1 How to read the book

The text of this book combines examples of selected statistical methods and descriptions of the R language as a statistical environment. Both are then illustrated on practical data analyses.

How to read this book? It depends on your knowledge and experience with statistical analyses as well as with the R environment. Those of you who have not done any (or almost any) data analysis since they attended their basic statistics/biostatistics course and who have never worked with R before should read the book from the beginning to the end. You who already know the environment R (at least the basics) and have been using regression in your work, can read the book somewhat non-systematically – picking the chapter that deals with data and/or methods you are currently interested in.

The most important part of the first chapter is the section 1.2, which defines variables. You certainly should not skip this part even if you believe that you are clear about distinguishing among variable types. This is because our definitions may differ from the definitions you may have encountered in other books and/or courses.

Chapter 2 describes the installation and use of the R software, which we will be using in this book for various data analyses. If you have never heard about R (or you have heard about it but never actually worked with it), this chapter is here especially for you. Apart from the installation of the software, you will also learn both some general principles for working in the R environment and important commands used repeatedly for various data manipulations later

in this book. Moreover, the chapter explains how to enter data into the R environment. All of the above is absolutely essential for being able to use the software successfully, not only for practicing the examples of this book, but also for practical data analyses of your own.

Chapter 3 shows some of the methods of so-called exploratory data analysis. This analysis means calculation of the basic statistical characteristics of the examined data sets as well as their informal comparison using plots, tables and other instruments. You will find there a general description of several useful R commands, by means of which you can create the majority of plots, with which we will work in later chapters.

The next four chapters (Chapters 4-8) are oriented rather more generally and even include some theory. Do not be misled and do not skip them thinking that they are not important from a practical point of view! In fact, they are some of the most important ones. This is because you can use their content for data analyses even if you use different software or, for example, for general considerations related to the strategy of statistical analyses. Particularly, Chapter 4 addresses the issue of how to work with statistical models of the type that will be used later in the book. Model formulation (traditionally in the form of a mathematical formula) forms a steppingstone, on which statistical analysis is based. In this chapter, we will talk about regression models and we will discuss what GLM actually is.

Chapter 5 presents the first example of a concrete and non-trivial data analysis. The presented example is not as simple as motivational examples usually are. It is more demanding, which allows us to discuss more of the aspects you will encounter when working with other examples (and in practical analyses of your own). The analysis is described in detail and commented abundantly (with references to the basic rules, which analysts should observe). In addition to the previous, some R outputs are described and interpreted and important definitions are stated (for example, the definition of contrasts).

Chapter 6 then goes deeper and deals with various systematic parts of a GLM model. It describes basic model classification based on the character of the explanatory variables. However, most importantly, it demonstrates in general terms how to translate a mathematical model into the R language and suggests how to interpret the model after the analysis is done.

Chapter 7 can be used as a simple "key", based on which readers (beginners) can try to decide which particular method to use (you will see yourself that, once you acquire more knowledge and experience, this decision will require even more thinking). We assume that the reader will be coming back to the book to use it as an aid for his/her own data analyses. If the reader remembers the general knowledge from Chapters 1–6, he/she can go straight to Chapter 7, which will direct him/her to a chapter where he/she will find an analysis of an example similar to his/her own; however, if you cannot decide which method to use, continue reading subsequent chapters in the order they are presented. Either way, we certainly recommend that you initially read Chapters 8–12 completely.

Chapters 8–12 are based on several examples, but they explain also theory, which you will encounter during these (and other) analyses. In general, the analyses of all examples from

these chapters follow a similar plan. The chapters are written in a way that makes them mutually independent (they can be read separately). As a result of this, you encounter similar problems repeatedly. When you pick a chapter, you should always read it as a whole. Do not try to follow it merely as a concrete report of a particular analysis.

1.2 Types of variables

Definitions and a detailed description of the characteristics of various variable types form the topic of a basic statistical course, which we certainly do not intend to repeat here. Let us just remind you of a few important facts that will be the most useful later in the book.

There are several possible viewpoints based on which variables can be classified. For our purposes, we will only need the following variable types:

Response variable: a (random) variable, the variability of which we attempt to explain by means of a statistical (regression) model. In other words, it is a variable that we want to model using a single explanatory variable or multiple explanatory variables. In this book we will exclusively address univariate models. That is, we will always model only one random response variable at a time (in contrast to multivariate models with several response variables considered in one model simultaneously). Depending on the particular GLM model type, our response variable will be either continuous or discrete, but always numeric (i.e. the values are numbers).

Explanatory variable: a variable by the means of which we explain the values of a response variable. Even in the case of univariate models, a response variable can be modelled by a single explanatory variable or by multiple explanatory variables. These can be either numeric or categorical (the values are characters and character chains that correspond to a given code marking groups/categories). Numerical variables can be continuous or discrete. We will mark continuous explanatory variables using lowercase letters, for example x. We will mark their value for the ith observation using index i (e.g. x_i). We will call continuous variables **covariates**. We will mark **categorical** variables using uppercase letters (for example A). a categorical variable can always take at least two different values, which we call **levels** (for example, "male" and "female"). Then the jth level of such a variable is marked with index j, for example as A_j. a categorical explanatory variable (with categories denoted by characters or numbers) will be also called a **factor**, as in the analysis of variance (ANOVA).

Weights represent a special case of variable. They determine relative weights of individual observations within a given data set. By default, functions in R use the same (unitary) weights for all observations. Externally entered weights are useful when a scheme with equal weights is not satisfactory and we need to change it. For example, weights can be based on known relative accuracy of observations (when data are averages of samples of different size, the sample sizes are often taken as weights). The weights must take non-negative values. Zero weights are possible, but somewhat extreme (they exclude the given measurement from the analysis), and we do not recommend using them (selected observations can be excluded from the analysis in a much more elegant manner).

1.3 Conventions

The book uses several font types. We use them for distinguishing the basic text of the book from software commands (and other key words) of the R language. For the names of commands and their arguments, we use the bold **`Courier New`** font. The names of objects that we create during analyses are typed in the normal `Courier New` font. The other text is typed in the Times New Roman font. Names of variables, parameter values and mathematical formulas are written in *italics*, while the names of factor levels are enclosed in quotation marks. The names of packages are underlined.

For transcribing everything that takes place in the command window of the R environment, we use the `Courier New` font of a smaller size. For better orientation, we differentiate between commands entered by the user, which we write in bold (for example, **`a<-1:5`**), and program responses in normal style (for example, `a`). To save space, some rows of output were cut off and substituted with three dots.

Plots made at the beginning of analyses were created using as few commands as possible and that is why they often do not have appropriate captions, legends, etc. Only plots made at the end of the analysis are closer to real presentation-quality and hence include various details (at the cost of longer commands).

The use of the natural logarithm (with base e) is prevalent in the statistics. We will thus label it simply by log. You need to be aware of the fact that this symbol can have a different meaning elsewhere (for example, in MS Excel it means the decadic logarithm – with base 10).

2 STATISTICAL SOFTWARE

There are many commercial as well as non-commercial programs available for data analyses. They widely differ by their quality, extent and price. Many inexperienced users work just with the basic software, for example, MS Excel. Some of the popular specialised programs include Statistica or JMP. And finally, there are large (and expensive) packages, such as SPSS or SAS. The difference among statistical software packages is not only in the number of implemented analysis types, but also in their flexibility, i.e. in the possibilities of user programming and other "customised" modifications and the automation of selected procedures. A simple analysis can, of course, be executed using any of the statistical software, however, more advanced methods are offered only by specialised packages.

2.1 The R environment

The good news is that one of the best statistical software packages for modern data analyses is available completely for free. It is called R (R Core Team 2015) and this book is based on its extensive possibilities. It actually covers a much wider area than is presented in this book.

The R programming environment is similar to the commercial program S-Plus (© Insightful Corporation), which used the S programming language; it was developed in the 1980's in the American AT&T Bell Laboratories (it has been improved several times since then). R uses the dialect of the S language in combination with the Scheme language, which means that, from the user's point of view, the overwhelming majority of the language for R and S-Plus are either the same or similar. R was created by New Zealanders Ihaka & Gentleman (1996). Today, it is managed by a group of people from around the globe, who call themselves the R Core Team. Thanks to that, the development of R is extremely dynamic. It is constantly evolving and self-improving.

On its own, R is not a user-friendly program of the MS Excel or Statistica type. You will not find nice, colourful windows, pull-down menus and clickable buttons, basically an environment where the main control instrument is the mouse. Be prepared that when you open R, you will only see an empty grey window (Fig. 2-1). R is mainly controlled by commands entered on a keyboard.

There are specialised environments available which can provide a standard user higher comfort when using R (for example, the Rstudio downloadable from http://www.rstudio.com/). As we are concentrating on the GLM models and data analysis here, we will not use them in this book.

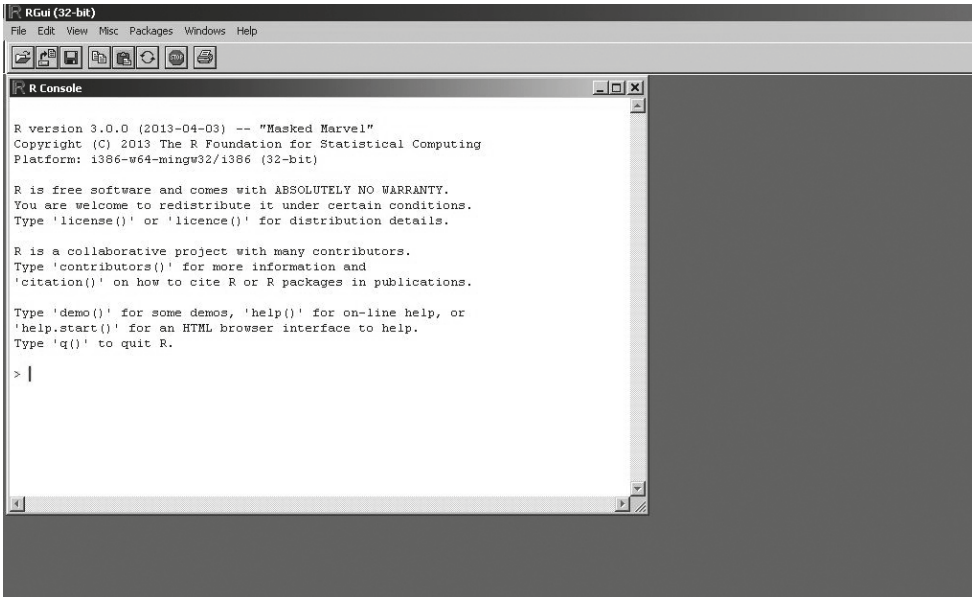


Fig. 2-1 Window of the R environment.

You may ask why we have chosen such "unfriendly" software. We have several reasons for it:

- R is one of the most extensive statistical packages that contain all essential types for modern analyses, which are continuously amended thanks to the continuous work of many people from around the world. Commercial statistical software is usually a closed system (it can be expanded only by purchasing yet another version), while R has been built as an open system. It means that new and additional methods can be added easily and for free at any time.
- Control of this program is not (for a non-statistician) that easy. However, that can be, paradoxically, an advantage because it forces its users to acquire certain knowledge about what they are doing and to think about it (at least) a little bit. Data analyses using commercial software can be somewhat "dangerous" since overly-friendly software allows people who have no idea what they are doing to perform an analysis by a sequence of more or less random clicks. Needless to say that the results of such analyses are often misleading or completely wrong.
- Yet another reason is that "friendly" statistical software will produce a huge volume of information, which they will spout out to you once the analysis is completed. It may not be easy for you to make sense of it. The philosophy of R is completely different – it will display only what the user asks for with the commands. This approach is based on the assumption that users will only use commands that they know something about or, alternatively, that they will look for in help pages or reference literature, thus being stimulated to study the given topic. By default, R outputs are typically quite modest in size. And, in fact, that

is good indeed. As you will see, it is easier to make general sense of the analysis this way. At any rate, specific details can be explicitly requested from the R environment later, if needed.
- One of the strengths of R lies in powerful modern graphics, focused on transparent and efficient data presentation.

Taken together, R is certainly not a hostile program. You will see while reading this book and while analysing your own data that, quite to the contrary, R control is relatively easy and, most of all, very efficient. To make your learning easier, we have described and explained the way to call various important procedures (and their outputs) in detail.

As we have already stated above, R is an environment used for handling objects. Objects can be data, results of their analyses (or results of various intermediate computations and manipulations), and handling means mathematical and statistical computations, manipulations, and constructions of tables and plots, etc. In reality, R is more than just a statistical package: it is a powerful programming language (for object programming), similar to C, C+ or Java. In order to successfully use R, you need to, among other things, learn its basic control commands. Making you familiar with these basic commands is the main objective of this chapter. It is the basic R philosophy that you can learn more details later when you need them. However, first of all, you need to install R on your computer.

2.2 Installation and use of R

The R installation file can be found on the Internet at http://www.r-project.org/. You can get it via the Download dialogue box where, upon clicking on CRAN, a list of servers, from which you can actually download the file, is displayed. Upon selecting a server, a window comes up where you choose the platform on which you will work with R: Linux, Mac OS or MS Windows. All calculations in this book have been done in MS Windows (use of R in alternative platforms is quite similar). Apart from the latest version of the installation file of R, the base folder also includes several info files. What we need now is the installation file with EXE extension. New versions of the software are uploaded in a relatively regular manner every few months. At the time of writing this book, the latest available version was 3.0.0. The name of the installation file thus was **R-3.0.0-win.exe**. This version incorporates 29 basic packages (i.e. libraries that implement various statistical methods and other computations). There are several other packages that include additional methods and other useful commands (at the time of writing this book, there were more than a thousand of them). You can see them by clicking on Packages on the taskbar on the left. A list of packages with a brief description of included functions is then displayed. If you are interested in any of them, you can download them (of course for free and in unlimited number). We will talk about how to do it later.

Once you have successfully installed the software, an icon called **R 3.0.0** will appear on the desktop. Upon launching the program, the main (large) window opens with a command (smaller) window in it – this is the R Console (Fig. 2-1). The upper taskbar of the main window includes several basic commands accessible either via a pull-down menu or using a button.

We will describe the functions of the most important ones. In the File menu, you can find basic options, such as **Display file(s)...** for displaying files in the R-3.0.0 folder. This is a standard, pre-set working folder. It is quite convenient to store your data in this folder because then you do not have to specify the full path when you want to read/save them. To keep things organised, we have saved all data that we will work with in a folder called MABD (abbreviation of the name of the book), located in folder C:\Program Files\R\R-3.0.0. If you want to do the same, you have to first download the data files from http://www.muni.cz/press/books/pekar_en and save them in the folder called MABD. Then you can change the working directory via the **Change dir...** option from inside of the launched R program (using the File option from the main menu).

By selecting **Load Workspace...** (again using the **File** option from the main menu), you can upload your previous work, provided you saved it in the end of your previous session using the **Save Workspace...** option. These options are especially useful when you want to continue your analysis after an interruption, for example, on the next day. You can just upload what you did earlier and you thus do not have to do everything from the scratch again. Nevertheless, be careful! Only the commands and (result) objects are saved (uploaded), not outputs and graphics. This means that if you want to see an output from some of the previous commands, you have to call and launch it again (provided you have not already copied the text output into, for example, a text file using Copy and Paste).

The **Edit** menu includes the copy and inserts functions in the same way we are used to from other programs. Furthermore, the menu offers an option for clearing the console window: **Clear console**, and an option for opening the spreadsheet editor: **Data editor...**, and an option for modifying the program appearance: **GUI preferences...** Here, you can change the size of the windows, font type, background and font colour, etc.

The **Misc** menu includes the useful **Stop current computation** option for suspending a given computation (for example, when the computation "freezes" for some reason, or when you realise that you have entered an input incorrectly, etc.). This can be done even faster using a shortcut key – by pressing the ESC key or by clicking on the STOP button. The **Remove all objects** option is used for removing all currently defined or uploaded objects.

The **Packages** menu offers options for working with additional packages, for example those that contain additional functions. Packages (with the exception of a few basic ones) are not automatically loaded into memory upon launching the program so that they are not occupying it with functions that we do not need at a given moment. When you want to use a function included in a certain package, you need to load the package by selecting the **Load package...** option. However, if the required package is not included in the library of your computer, you have to install it first. We will practice it by installing the sciplot package, which we will use for creating some plots. If you are connected to the Internet, click on the **Install package(s)...** option and select a server from the displayed list; from this, you will download the package. When you do that, your computer connects to the given server and displays a list of all currently available packages. Select the one called sciplot (its installation on your computer is then executed automatically). If you are not connected to the Internet, you have to first download

the zip form of the package from the above address separately and install the package from the zip file manually afterwards. When doing so, you need to be aware of the version of R that you are using since packages can be specific for different versions. Install the zip file into your library using the **Install package(s) from local zip files...** option. Do not forget that in order to activate functions of an already installed package, you need to always load it for each session.

The **Windows** menu includes options for switching between the command and graphical windows. The most useful is probably the **Tile** option, which allows you to see two windows next to each other.

Finally, the **Help** menu offers many useful options. The **Console** option will explain to you how to control the console window. Commands are entered into the console following prompt (>). The prompt is normally red, while program answers are (typically) displayed in blue. Commands can be typed by the user line by line (hitting "Enter" after each complete command), or on the same line, separated by semicolons. No additional spaces are necessary between commands that are separated by a semicolon. Previous commands can be recalled one after another by pressing the up arrow key and back by pressing the down arrow key. Using the arrow keys makes the work much easier. For example, if you make a mistake entering a command, you can recall it by pressing the up arrow key and correct it. Using the left and right arrow keys, you can move within the currently edited line.

You can learn many useful facts about R when using the **FAQ on R** or **Manuals in (PDF)** options. The **R functions (text)...** option is very useful; it displays a detailed description of the function, name of which you enter. Here you can find a description of what the given function can be used for, a list of all its arguments, their legal values, references to literature (related to the method, on which the function is based), and a few examples. This command works only for functions from the package(s) that are currently uploaded. Descriptions of all functions (including those that are included in the installed packages but not uploaded) can be found by using the **Html help** option. If you select this option, pages with some basic information about R are displayed, including the functions included in the pre-installed packages. If you do not know the name of a given function but you know what it is supposed to do, you can try to find it using the **Search help** option. For example, if we want to find a function that computes the Shapiro-Wilk test, type the key word „shapiro". The program then searches all installed packages and displays the functions that include the searched word in their names or descriptions. The name of the package, in which the given function can be found, is displayed in the brackets behind the name of the function. In our case, the function is called **`shapiro.test`** and it is in the stats package.

2.3 Basic operations

The possibilities of R are really wide-ranging – the program includes hundreds and hundreds of different commands. Of course we cannot mention and explain all of them here. We will focus only on the most basic ones and on those you will encounter in this book most often. Some others will be introduced within the context of examples later in the book. A

brief list of the most important commands called the "R Reference Card" can be found at the following address: http://cran.r-project.org/doc/contrib/Short-refcard.pdf. It is basically a four-page sheet. You can also read more about R, in the book by one of the R Core Team experts, Dalgaard (2008).

Let us try some operations, starting with simple manipulations and proceeding to more complicated procedures. Firstly, we use the environment as a scientific calculator. Just type on the command line `2+5`, press ENTER and the software produces a result on a new line (which begins with `[1]`). Basic mathematical operators are: addition (**+**), subtraction (**-**), multiplication (*****), division (**/**), and power (**^**). Logic operators have the following form: less than (**<**), greater than (**>**), equal (**==**), not equal (**!=**), less than or equal (**<=**), greater than or equal (**>=**). They do not produce a number but a logical value: either **TRUE** (abbreviated **T**), or **FALSE** (abbreviated **F**).

Names of mathematical functions are in R very intuitive and thus easily memorable, for example, absolute value (**abs**), logarithm with base of *e* (**log**), logarithm with base of 2 (**log2**), logarithm with base of 10 (**log10**), exponent (**exp**), sine (**sin**), cosine (**cos**), tangent (**tan**), arc sine (**asin**), arc cosine (**acos**), arc tangent (**atan**), sum of several numbers (**sum**), product of several numbers (**prod**). The command for square root is **sqrt**. For other roots it is necessary to type them as powers (the power operator **^** is general and allows even negative and as well as non-integer arguments when they are mathematically correct). These simple functions are called by their name, followed by a number in parentheses, as you can see in the following examples:

```
> 3*2
[1] 6
> 3^4
[1] 81
> sqrt(9)
[1] 3
> 8^(1/3)
[1] 2
> 3==4
[1] FALSE
> log(10)
[1] 2.302585
> log10(10)
[1] 1
> exp(2)
[1] 7.389056
> prod(2,3,4)
[1] 24
```

Answers to these commands were only displayed on the screen. You cannot directly work with them any further. You can copy them and move them to the current command line using Copy and Paste. This can be inconvenient, especially if we want to conduct further operations with the result. In that case, you need to save the result into an object (and subsequently use its name in parentheses when using it as an argument of some other function). The simplest object is a scalar, i.e. a vector with a single element. Creating a scalar

is easy. Choose its name, for example `a`, type an arrow (composed of 'less than' and a dash, **`<-`**) or **`=`** behind it, and enter the value you want to enter in it, for example `8`. All of this is meant as an assignment of the value to the object `a`. You can easily display the content of a vector by simply typing its name on a new line or on the same line behind a semicolon. To create vectors of length larger than 1, you have to use the **`c`** (concatenate) function. This function will bind all entered values (given as arguments, within parentheses and separated by commas) into a vector in the order entered:

```
> a<-8; a
[1] 8
> b<-c(2,1/2,95); b
[1]  2.0  0.5 95.0
```

All names in R are case sensitive – including keywords (thus by typing `b` and `B` you are calling *different* objects). Object names can be almost arbitrary. Nevertheless, it is a good practice to remember some restrictions. Names must not begin with a number or a special symbol (such as comma or dot, etc.). Some names are better not to be used to avoid confusion. This applies to standard R commands and functions which have a predefined function, for example **`break`**, **`c`**, **`C`**, **`D`**, **`diff`**, **`else`**, **`F`**, **`FALSE`**, **`for`**, **`function`**, **`I`**, **`if`**, **`in`**, **`Inf`**, **`mean`**, **`NA`**, **`NaN`**, **`next`**, **`NULL`**, **`pi`**, **`q`**, **`range`**, **`rank`**, **`repeat`**, **`s`**, **`sd`**, **`t`**, **`tree`**, **`TRUE`**, **`T`**, **`var`** and **`while`**. If you use them then the predefined function will be masked (replaced with the new object). This can cause a number of nasty problems that are difficult to discover. You had best avoid using names that collide with the names of R functions (unless your intention is to replace the standard R function by a version of your own).

Values of vector components (and hence of vectors themselves) can be of different types: numeric, e.g. `c(1.5,20,-3.1)`, logical, e.g. `c(TRUE,FALSE,TRUE)`, or character. Characters are always enclosed in quotation marks, e.g. `c("blue","red","green")`. A vector should always include values of the same type. If there are various types, e.g. numbers and characters, the vector will be automatically classified by R as the most general appropriate type (i.e. character). Character vectors cannot be used for mathematical operations. This is useful to remember when searching for errors and reasons why something does not work.

Numeric vectors can be created by entering every number or using a colon, which requests creation of an integer sequence of values from:to.

```
> y<-1:11; y
 [1]  1  2  3  4  5  6  7  8  9 10 11
```

You can also create the same sequence in a different way – using the **`seq`** command. This command is not as simple as the previous command since it can do more – it can create a sequence that does not consist of integers only. To make it work, you need to specify the **arguments** that you need for the given objective. Particularly, you need to specify the initial value (**`from`**), final value (**`to`**) and the step size (**`by`**). It is typical for functions in R to include multiple arguments that have their own name and pre-defined positions (there are also functions with arguments that do not have names. It lets you to specify the arguments either using their names in any order or without names but in the appropriate order. The

latter option is more economical from the writing perspective and that is why we prefer to use it here. Nevertheless, this option has a significant implicit danger. The order of arguments is not quite codified and it can thus theoretically change with the development process of the R language. That is why, if you are using a different version of R than this one, it is absolutely necessary that you inspect the order of individual arguments for all of the used functions. Alternatively, you need to learn how to type commands with the names of arguments.

Now let us go back to the **seq** command and let us create a sequence from 1 to 2 with a step of 0.1:

```
> x<-seq(from=1,to=2,by=0.1); x
 [1] 1.0 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9 2.0
```

Try to play with the arguments of this command – skip their names and change their order – in order to see how dramatically the result can change.

You can select elements from such a vector. The easiest way is to enter the element positions in square brackets. One number can be typed directly as an argument. Multiple numbers have to be connected together, either as a sequence using a colon, or as a vector using **c**. This means that if we want to select the third element, followed by the third through fifth, and finally sixth, eighth, and ninth elements from vector `x`, we will do so in the following manner:

```
> x[3]
[1] 1.2
> x1<-x[3:5]; x1
[1] 1.2 1.3 1.4
> x2<-x[c(6,8,9)]; x2
[1] 1.5 1.7 1.8
```

We can make our selection not only based on a given position but also based on a different (more complex) condition. If you want to find out which elements within a vector are greater than a certain value, use the **which** command, which will return the sequence of numbers of the elements that meet the given condition:

```
> which(x>1.5)
[1]  7  8  9 10 11
```

Vectors can be used in many more operations. We can join them to longer vectors (e.g. join `x1` and `x2` to create `x12`), or alter them using mathematical operators and functions.

```
> x12<-c(x1,x2); x12
[1] 1.2 1.3 1.4 1.5 1.7 1.8
```

Many mathematical functions are "vectorised" in R, i.e. written in a way such that the function is automatically applied to all elements of the given vector (and the result is returned as a vector). We can demonstrate it on, for example, calculating the third power (for the 11 numbers saved in `y`):