About this book

Two common types of user interfaces in statistical computing are the command line interface (CLI) and the graphical user interface (GUI). The usual CLI consists of a textual console where the user types a sequence of commands at a prompt and the output of the commands is printed to the console as text. The `R` console is an example of a CLI. A GUI is the primary means of interacting with desktop environments, like Windows and Mac OS, and statistical software like JMP. GUIs are contained within windows, and resources, such as documents, are represented by graphical icons. User controls are packed into hierarchical drop-down menus, buttons, sliders, etc. The user manipulates the windows, icons and menus with a pointer device, such as a mouse.

The R language, like its predecessor S, is designed for interactive use through a command line interface (CLI), and the CLI remains the primary interface to R. However, the graphical user interface (GUI) has emerged as an effective alternative, depending on the specific task and the target audience. With respect to GUIs, we see R users falling into three main target audiences: those who are familiar with programming `R`, those who are still learning how to program, and those who have no interest in programming.

On some platforms, such as Windows and Mac OS, `R` has graphical front-ends that provide a CLI through a text console control. Similar examples include the web-based RStudio^{TM} IDE, the Java-based JGR and the RKWard GUI for the Linux KDE desktop. Although these interfaces are GUIs, they are still very much in essence CLIs, in that the primary mode of interacting with `R` is the same. Thus, these GUIs appeal mostly to those who are comfortable with `R` programming.

A separate set of GUIs target the second group of users, those learning the `R` language. Since this group includes many students, these GUIs are often designed to teach general statistical concepts in addition to `R`. A CLI component is usually present in the interface, though it is deemphasized by the surrounding GUI, which is analogous to a set of "training wheels" on a bicycle. An example of such a GUI is `R` Commander, which provides a menu- and dialog-driven interface to a wide range of `R`'s functionality and supports plugins.

The third group of users, those who only require `R` for certain tasks and do not wish to learn the language, are targeted by task-specific GUIs. These interfaces usually do not contain a command line, as the limited scope of the task does not require it. If a task-specific GUI fits a task particularly well, it may even appeal to an experienced user. There are many examples of task-specific GUIs in `R`. Many GUIs assist in exploratory data analysis, including `exploRase`, `limmaGUI`, `playwith`, `latticist`, and Rattle. Other GUIs are aimed at teaching statistics, e.g., `teachingDemos`. There are a few tools to automatically generate a GUI that invokes a particular R function, such as the `fgui` package and the `guiDlgFunction` function from the `svDialogs` package.

All of these examples are within the scope of this book. We set out to show that, for many purposes, adding a graphical interface to one's work is not terribly sophisticated nor time-consuming. This book does not attempt to cover the development of GUIs that require knowledge of another programming language, although several such projects exist. One example is programming a Java/Swing GUI through `rJava`, a native interface between `R` and Java. It is also possible to extend the `RKWard` GUI using a mixture of XML and Javascript, and the `biocep` GUI supports Java extensions. Our focus is instead on programming GUIs with the `R` language.

The bulk of this text covers four different packages for writing GUIs in `R`. The `gWidgets` package is covered first. This provides a common programming interface over several `R` packages that implement low-level, native interfaces to GUI toolkits. The `gWidgets` interface is much simpler – and less powerful – than the native toolkits, so is useful for a programmer who does not wish to invest too much time into perfecting a GUI. There are a few other packages that provide a high-level `R` interface to a toolkit such as `rpanel` or `svDialogs`, but we focus on `gWidgets`, as it is the most general.

The next three parts introduce the native interfaces upon which `gWidgets` is built. These offer fuller and more direct control of the underlying toolkit and thus are well suited the development of GUIs that require special features or performance characteristics. The first of these is the `RGtk2` package which provides a link between `R` and the cross-platform GTK+ library. GTK+ is mature, feature rich and leveraged by several widely used projects.

Another mature and feature-rich toolkit is Qt, an open-source C++ library from Nokia. The `R` package `qtbase` provides a native interface from `R` to Qt. As Qt is implemented in C++, it is designed around the ability to create classes that extend the Qt classes. `qtbase` supports this from within `R`, although such object oriented concepts may be unfamiliar to many `R` users.

Finally, we discuss the `tcltk` package, which interfaces with the Tk libraries. Although not as modern as GTK+ nor Qt, these libraries come pre-installed with the Windows binary, thus avoiding installation issues for the average end-user. The bindings to Tk were the first ones to appear for `R` and most of the GUI projects above, notably `Rcmdr`, use this toolkit.

These four main parts are preceded by an introductory chapter on GUIs.

This text is written with the belief that much can be learned by studying examples. There are examples woven through the primary text, as well as stand-alone demonstrations of simple yet reasonably complete applications. The scope of this text is limited to features that are of most interest to statisticians aiming to provide a practical interface to functionality implemented in R. Thus, not every dusty corner of the toolkits will be covered. For the `tcltk`, `RGtk2` and `qtbase` packages, the underlying toolkits have well documented APIs.

The package `ProgGUIInR` accompanies this text. It includes the complete code for all the examples. In order to save space, some examples in the text have code that is not shown. The package provides the functions `browsegWidgetsFiles`, `browseRGtk2Files`, `browseQtFiles` and `browseTclTkFiles` for browsing the examples from the respective chapters.

The authors would like to thanks the following people for their helpful comments made regarding draft versions of this book; Richie Cotton, Erich Neuwirth, Jason Crowley, and Tengfei Yin.