Journal of Statistical Software

January 2007, Volume 17, Issue 9. http://www.jstatsoft.org/

rpanel: Simple Interactive Controls for R Functions Using the tcltk Package

Adrian Bowman
University of Glasgow

Ewan Crawford
University of Glasgow

Gavin Alexander
University of Glasgow

Richard W. Bowman
University of Cambridge

Abstract

In a variety of settings it is extremely helpful to be able to apply R functions through buttons, sliders and other types of graphical control. This is particularly true in plotting activities where immediate communication between such controls and a graphical display allows the user to interact with a plot in a very effective manner. The **tcltk** package provides extensive tools for this and the aim of the **rpanel** package is to provide simple and well documented functions which make these facilities as accessible as possible. In addition, the operations which form the basis of communication within **tcltk** are managed in a way which allows users to write functions with a more standard form of parameter passing. This paper describes the basic design of the software and illustrates it on a variety of examples of interactive control of graphics. The **tkrplot** system is used to allow plots to be integrated with controls into a single panel. An example of the use of a graphical image, and the ability to interact with this, is also discussed.

Keywords: dynamic graphics, graphical user interface, interactive plots, R, **tcltk**.

1. Introduction

The arrival of facilities for interactive and dynamic graphical displays was a significant milestone in statistical computing. However, only a relatively small number of tools, such as spinning three-dimensional plots, have found their way into some standard packages. Experimentation with new dynamic graphical methods has largely been limited to those who have the ability to programme in appropriate languages. For example, there are many examples of Java applets which use dynamic graphics to excellent effect, thanks to the considerable efforts of the authors. However, the creation of displays of this type has been limited to a relatively

small subset of the statistics community who can programme in the required languages. A notable exception as a development platform is the XLISP-STAT system developed by Tierney (1990), which has very flexible facilities for creating dynamic graphics and more general graphical user interfaces. This is illustrated further in the Arc system described by Cook and Weisberg (1999). However, development in XLISP-STAT requires some knowledge of the Lisp language.

The advent of R (R Development Core Team 2006), and its S-based predecessors, as a standard and universal statistical computing environment, together with its facilities for communication with other software, has changed this picture. The origins of the **rpanel** project lie in a wish to create R functions to provide button and other graphical user interface (GUI) controls for the dynamic graphical tools available in the **iplots** system described by Urbanek and Theus (2003). The intention was to provide users with the ability to write R scripts to create new graphical displays of their own from the **iplots** building blocks, without requiring knowledge of the Java language in which that system is written. However, the more general **JGR** system (`http://stats.math.uni-augsburg.de/JGR/`), which provides a GUI operating environment for R, now incorporates **iplots** within it.

In fact, a very general set of tools for interactive control was provided at an early stage by the **tcltk** package, created by Dalgaard (2001). This represented a significant development by providing a link from R to the extensive GUI control facilities of the Tcl/Tk system (`http://www.tcl.tk/`). This is becoming increasingly popular as a means of controlling R functions. For example, Fox (2005) describes the substantial R Commander (**Rcmdr**) system which allows users to control R functions through an extensive menu and dialogue system, based on the **tcltk** package. This offers a style of control which is similar to those provided by other packages which are designed for the application of relatively straightforward statistical methods. A further example is provided in the **tkWidgets** package of Zhang (2006), which provides tools for the interactive exploration of file pathnames, R packages and objects.

In the case of dynamic graphics, more continuous control is required so that, for example, the altered position of a slider can immediately affect the displayed graph. The Tcl/Tk system is also able to provide this, as illustrated by the `tkdensity` example provided with the **tcltk** package and developed by Mächler (2005) in the **sfsmisc** package. While graphics based on Java and other languages are generally object-oriented in their construction, a convenient approach with Tcl/Tk is simply to redraw the entire plot according to the values of the parameters set by the relevant controls. With the speed of modern machines, this rather simple device is sufficient to create an effective dynamic display, unless the graphics are particularly complex to construct. An alternative set of GUI facilities is available in the Gtk toolkit, available in R through the **RGtk** package of Temple Lang (2004). The **iSPlot** package of Whalen (2006) uses this route to create facilities for linking and interacting with plots. Temple Lang and Lawrence (2006) have also produced an **RGtk2** package.

The **tcltk** package provides access to a very extensive range of powerful facilities provided by the Tcl/Tk system. A primary aim of the present paper is to describe a suite of R functions, referred to as the **rpanel** package, which are intended to make access to a limited but very useful range of these facilities as convenient as possible for those who have reasonable familiarity with R but are less confident in embarking on a general exploration of the Tcl/Tk system. In particular, facilities are provided to create control widgets through single function calls. The **rpanel** package provides documentation for these so that reference to Tcl/Tk documentation is not required. The operations which are the means of communication in

tcltk are also managed by **rpanel** so that the user need not be aware of them. A standard, flexible form of parameter passing is also used to communicate with the plotting functions written by the user. Among other advantages, this allows users to call a particular function repeatedly, to produce simultaneous and independent copies of the same panel and associated graphics, without causing control difficulties.

Part of the aim of the paper is also to outline the range of uses to which these facilities can be put. This includes situations in data analysis where interactive control gives important and convenient insight, such as in dynamic graphics. A special, but important, case arises in teaching, where concepts and ideas can be expressed graphically. Animation in particular can communicate some concepts in a much more effective manner than static plots. Interaction with more general images is also a potentially rich facility. All of these aspects are illustrated in the paper. The basic ideas and design of the package are described and illustrated in Section 2 with further, more extensive, examples in Section 3. Some technical issues are outlined in Section 4. The ability to incorporate R graphics into the control panel, using the **tkrplot** system (Tierney 2005), is discussed in Section 5, along with an exercise involving data collection in a teaching setting which highlights the potential for interacting with more general images. Some final discussion is given in Section 6.

The **rpanel** package is available at http://CRAN.R-project.org/. A collection of example scripts is available at http://www.stats.gla.ac.uk/~adrian/rpanel/.

2. The design of the software

Dalgaard (2001) and Mächler (2005) used the creation of a control panel for a density estimate as a simple but striking example of the use of **tcltk**. A similar example is used here to illustrate the aims of **rpanel**. The data refer to the percentages of aluminium oxide found in samples from a tephra layer, resulting from a volcanic eruption in Iceland around 3500 years ago. The geological background to the data is given by Dugmore, Larsen, Newton, and Sugden (1992). In the code below it is assumed that the data are stored in the vector `Al2O3` and, for convenience, the variable `r` is set to the span of the data, `diff(range(Al2O3))`.

```
density.draw <- function(panel) {
   plot(density(panel$y, panel$sp), main = "Density estimate")
   panel
   }
density.panel <- rp.control("Density estimation", y = Al2O3, sp = r / 8)
rp.slider(density.panel, sp, r / 50, r / 4, density.draw, "Bandwidth")
```

The first part of the code defines an 'action' function called `density.draw` which produces the desired graph and is available to be called by subsequent controls. In this case, the function is essentially one line, to draw a density estimate. The input to the function is a list object which contains all the information required. For reasons discussed in Section 4, all action functions must return the list object. Some functions may amend the list components but that is not necessary in this simple example.

The `rp.control` function creates a panel window on the screen, with an optional title defined by the `title` argument. The output of the function is a string variable, called `density.panel` on this occasion, giving the name of a list object which contains information on the current

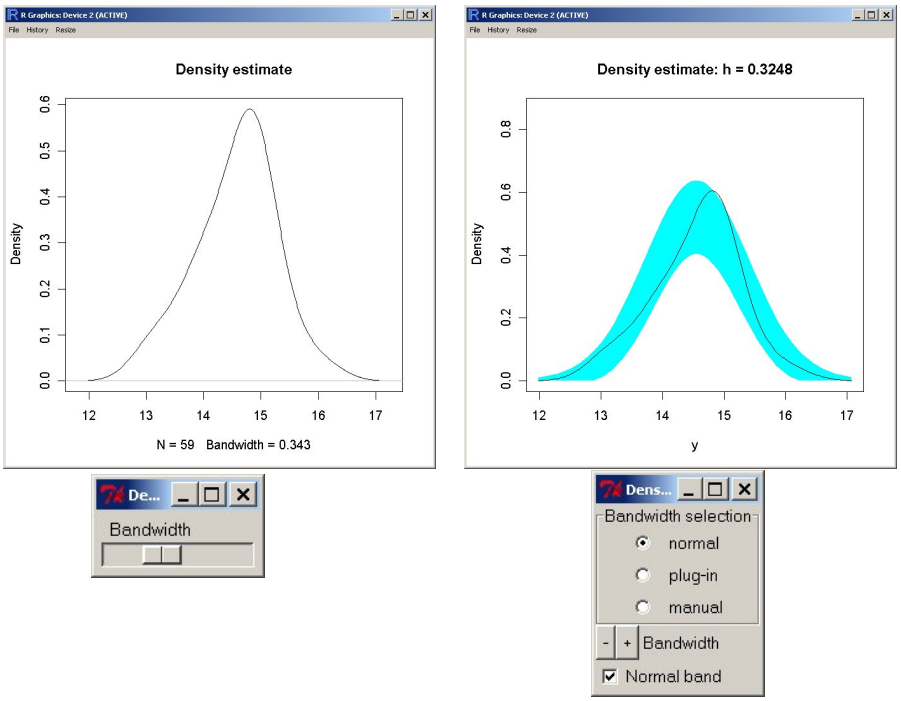

Figure 1: The left hand graph show an R graphics window with a plot of a density estimate whose bandwidth is controlled interactively by the slider below. The right hand graph shows an R graphics window containing a plot of a density estimate, with a reference band for normality superimposed. The bandwidth and the superimposition of the reference band is controlled interactively by the radio buttons, buttons and checkbox in the panel below.

state of the panel. The list object itself is held in a special R environment. Any additional arguments of `rp.control`, such as the data `y` and the smoothing parameter `sp` in this example, are simply passed directly into components of the panel object, so that this information can be easily accessed when later actions are requested.

The `rp.slider` function adds a slider to the panel window. The first argument identifies the panel object to which the slider is to be added. The second argument gives the name of the component of the panel object which is to be created and subsequently controlled by the slider. The following two arguments define the start and end points of the range of the slider. The `action` argument then gives the name of a function which will be called when the slider position is changed. The final argument allows a title to be added to the slider.

The left hand side of Figure 1 illustrates the panel and R graphics windows which are created by this script. The ability to alter the degree of smoothing applied in the construction of the density estimate by moving the slider gives the user a very convenient method of interaction. It is one of the main aims of **rpanel** that control facilities such as these can be created with a very small number of function calls, using a standard style of R programming. The functions involved are intended to be accessible to users who have a reasonable acquaintance with R but who wish to avoid the need for more extensive programming or reference to Tcl/Tk documentation. In later sections of the paper, some further modes of use of the **rpanel** system, which involve the use of graphics integrated into the panel, are also discussed.

The script discussed above made use of a single slider. The commands below illustrate how

a wider range of controls can be used to extend the options which can be created. The **sm** package described by Bowman and Azzalini (1997) is used here, through the functions `sm.density`, `hnorm` and `hsj`, to provide more extensive smoothing techniques.

```
density.panel <- rp.control("Density estimation", x = Al2O3, xlab = "x",
                    ylab = "Density", model = "none", sp = hnorm(Al2O3))
rp.radiogroup(density.panel, sp.method,
                    c("normal", "plug-in", "manual"),
                    title = "Bandwidth selection", action = density.draw)
rp.doublebutton(density.panel, sp, 1.02, log = TRUE,
                    range = c(diff(range(Al2O3)) / 50, NA),
                    title = "Bandwidth", action = density.draw)
rp.checkbox(density.panel, model,
                    title = "Normal band", action = density.draw)
```

The `rp.radiogroup` function provides options for the choice of the smoothing parameter. When this is to be set manually, the `rp.doublebutton` function provides a pair of buttons to alter the value of the specified panel component by small increments. This form of control, which was used to good effect in the XLISP-STAT system developed by Tierney (1990), is particularly appropriate when it is helpful to be able to locate accurately particular values of the parameter in question and when there is no natural constraint on its range. It is used here for the smoothing parameter, on a logarithmic scale. The `rp.checkbox` function allows a flag to be set through the named logical component. Here this is used to control whether a reference band for normality is to be added to the plot. Reference bands are discussed in Bowman and Azzalini (1997) and implemented in the associated **sm** package.

The action function `density.draw` is redefined as follows.

```
density.draw <- function(panel) {
   if (panel$sp.method == "normal")  panel$sp <- hnorm(panel$x)
   if (panel$sp.method == "plug-in") panel$sp <-   hsj(panel$x)
   with(panel, {
      if (model == TRUE) Model <- "normal" else Model <- "none"
      sm.density(x, sp, xlab = xlab, ylab = ylab, model = Model,
                  rugplot = FALSE)
      title(paste("Density estimate: h =", signif(sp, 4)))
      })
   panel
   }
```

This sets the `sp` component according to the selected method and constructs the plot, with a reference band if requested. The `with` construction is very useful to avoid having to include the panel object prefix when referring to components. However, where assignments are required to alter particular components these must take place outside the `with` block so that the effect is not simply local. The end result is shown on the right hand side of Figure 1.

3. Examples of interactive control of R graphics

There are very many examples of situations where interactive control can add very useful functionality to an analysis. In this section, three further examples are given. The first concerns the calculation of sample sizes required in the design of an experiment. It is a very simple matter to perform these calculations in R. The following code embodies this in a function which can then be called through a panel. (Note that only the principal tail calculation is performed as the contribution of the subsidiary tail is negligible for any reasonable power.) The R panel object contains all the necessary information in the form of the populations means `mu1`, `mu2`, the common standard deviation `sigma` and a vector `ngrid` which defines the candidate sample sizes of each group.

```
powerplot.pars <- function(powerplot) {
   se             <- powerplot$sigma * 2 / sqrt(powerplot$ngrid)
   powerplot$pow <- 1 - pnorm(2 - abs(powerplot$mu2 - powerplot$mu1) / se)
   powerplot.draw(powerplot)
   }
```

A function `powerplot.draw` is used to construct the plot from the current setting of the parameters in the R panel object. Some effort has been expended on this occasion to print the parameter values neatly in the title of the plot, so that the text positions do not move as the parameter values are changed. The plot also identifies the currently specified sample size through dotted lines. The considerable benefit of interaction in this setting is that the sensitivity of the power to the current settings can be easily explored. A further option is available through the checkbox, which allows the underlying populations to be plotted at the same time. This provides a very helpful graphical display of the separation between the underlying populations corresponding to the current parameter settings. The code to create the plots is straightforward, in principle, for R users and is encapsulated in R functions in standard form. However, the addition of simple interactive controls greatly enhances the functionality of the final code.

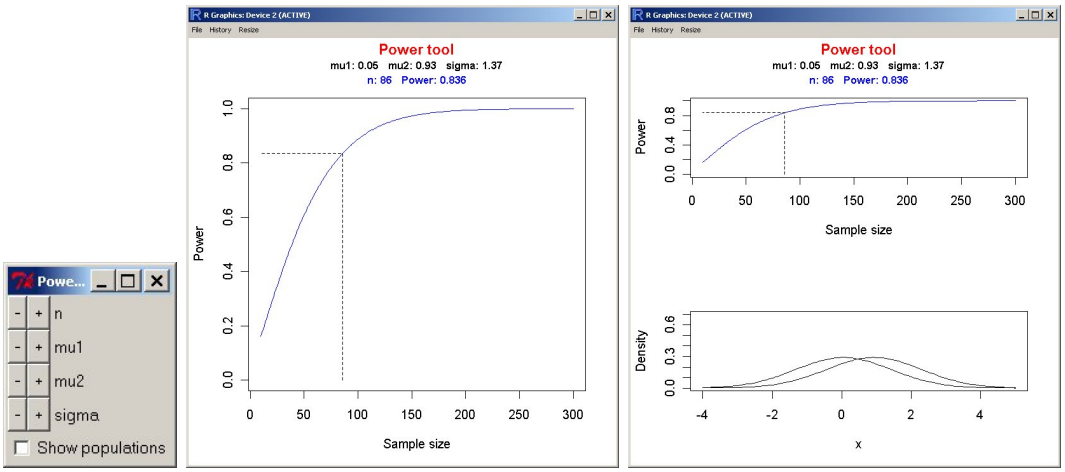

Figure 2: A panel to control plots of a power function for a two-sample t-test. The right hand plot shows the result when a display of the underlying populations is requested.

A second example is provided by a panel to control the display of three-dimensional data, using the **rgl** package written by Adler (2005). This provides access to a very useful subset of OpenGL (http://www.opengl.org/) facilities for the display of three-dimensional objects. The code below creates a panel which allows interactive control of an **rgl** display of regression data, involving one response and two explanatory variables. The data used here refer to the amount of giving in pounds per annum per member in different dioceses of the Church of England, related to the employment rate of the region and the proportion of the population who usually attend on Sundays. A full description of the data and an associated analysis are given by Pickering (1985). Graphical illustrations of the issues raised by the data are given by Bowman and Robinson (1990).

```
spin.panel <- rp.control("Spin plot", x = x, y = y, z = z, xlab = xlab,
                ylab = ylab, zlab = zlab, theta = -30, phi = 30,
                fov = 1, current.model = "None", smat = smat, fitted = fv)
rp.doublebutton(spin.panel, theta, -1,
                title = "Theta", action = rotate)
rp.doublebutton(spin.panel, phi, -1,
                title = "Phi",   action = rotate)
rp.radiogroup(spin.panel, model,
                c("None", "No effects", xlab, zlab, paste(xlab,"and",zlab)),
                title = "Model", action = model.fn)
rp.checkbox(spin.panel, residuals.showing,
                title = "Show residuals", action = residuals.fn)
```

The code to produce the three-dimensional plot scales the axes and creates a surrounding box, using straightforward R programming and calls to the **rgl** package. The panel controls allow the plot to be rotated smoothly, through the use of double-buttons and the associated `rotate` action. A plane to represent a fitted regression model can also be superimposed through the radio buttons and their associated action `model.fn`. The **rgl** functions allow this to be transparent, so that the data beneath remain visible. A checkbox controls whether the residuals are displayed as vertical lines joining the observations to the fitted plane, using the `residuals.fn` action.

A comparison of different models is very helpful in explaining their meaning in a teaching context. This particular example has special interest because the effect of attendance is not statistically significant until the effect of employment is included in the model. The graphical displays assist the discussion of the reasons for this, which are connected with the fact that the relationship of giving to employment is positive while the relationship with attendance is negative.

It would be relatively straightforward in principle to allow nonparametric surfaces to be added to the plot with bandwidth control along the lines of the simpler example of density estimation discussed in Section 2. The **rgl** package has special facilities for plotting surfaces and the examples provided there make good use of these.

A final example arises in the context of repeated measurements data. Figure 4 shows a plot of leutinising hormone in cows, split into two groups according to whether the calves were suckled or not and presented on a log scale. The data and background are given by Raz (1989). Here a panel allows easy movement between plots of the raw data and plots of means and standard errors using a function which has been written for the display of

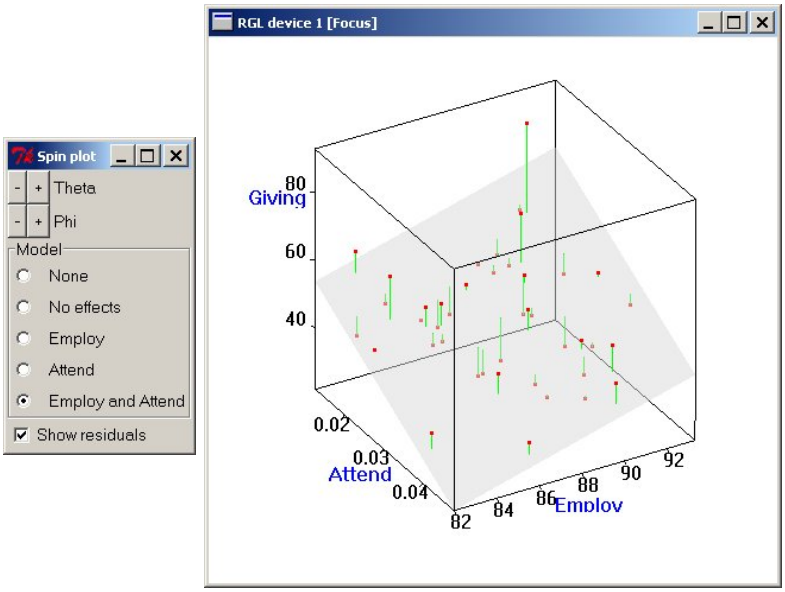

Figure 3: A plot of the Church of England data, with Giving as response and Employ and Attend as explanatory variables. The display of the regression plane and residuals is controlled by the panel.

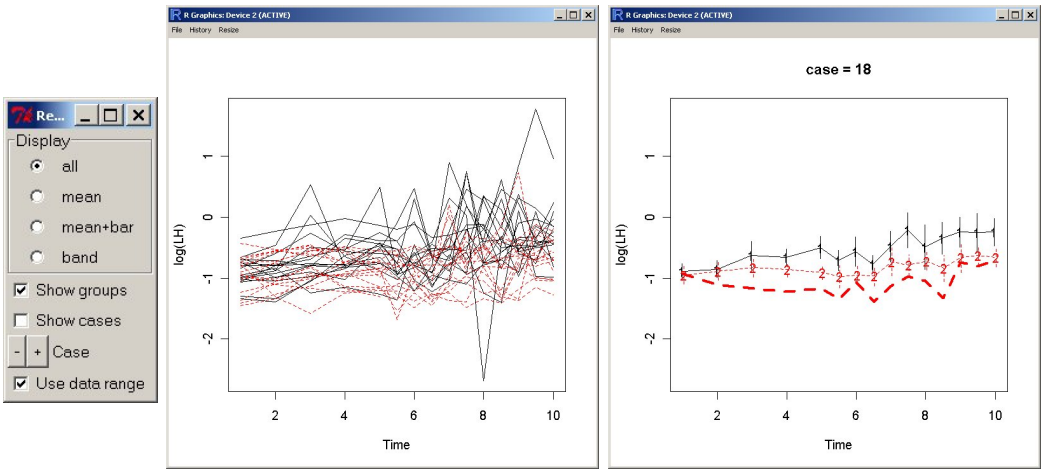

Figure 4: Plots of repeated measurements data on leutinising hormone in cows. The left hand plot shows all the data, with group membership identified by different colours and linestyles. The right hand plot shows means and bars representing ±2 standard errors. The control panel allows these, and other, views of the data to be displayed, as well as enabling the user to examine the shapes of individual profiles. A profile from group 2 is displayed in the right hand plot.

repeated measurement data. However, in addition, a double-button allows individual profiles to be superimposed and holding the button down creates an animation. This provides a very helpful means of identifying unusual profile shapes in a manner which is not possible by plotting all the data, due to the difficulty in identifying individual cases. In addition, the assumption of the additivity of random effects can be explored visually through assessing the extent to which the observed profiles are consistent with an additive shift of a common underlying pattern.

4. Technical issues

The **tcltk** package allows a nominated function to be called when a slider, button or other control widget is activated. In **rpanel** this facility is used to update the value of the panel component which corresponds to the variable of interest. This is done in a manner which relieves the user from the task of managing the variable handles and type-casting required by Tcl/Tk. In addition, the assignment

```
panel <- action(panel)
```

is evaluated, where `action` is the action function associated with the control widget. In the examples of Section 3, this simply redraws the graph with the current variable settings. By this simple mechanism, the panel object provides considerable flexibility in passing information to all relevant functions, simply by reading the current state of its components. The panel objects are held in a special R environment to avoid conflict with other R objects.

The **rpanel** functions illustrated in the previous sections have been used with default settings for the layout of multiple widget controls. For simple panels, with only a small number of widgets, this is generally adequate. However, with larger numbers of widgets, or in settings where very particular widget placings are required, a more precise method of control is necessary. The **rpanel** package allows a degree of control through a `pos` parameter which can be set to `"left"`, `"right"`, `"top"` or `"bottom"`. The 'pack' layout manager in the Tk system is then used to control the layout of the specified widget. Alternatively, when `pos` is set to a numeric vector of length 4, Tk's 'place' layout manager is called to place the widget at a specified pixel position (first two elements) and with a specified width and height (second two elements). The following code, which shows only the first three of several statements used to construct controls for the display of distributions as a form of electronic tables, gives an illustration of this. Careful layout and alignment of widgets allows a very compact scheme with appropriate associations between relevant widgets.

```
tables.panel <- rp.control("Distributions", size = c(625, 150),
             xobs = 1, p = 0.05, distribution = "Normal",
             degf1 = 5, degf2 = 30)
rp.radiogroup(tables.panel, distribution,
             c("Normal", "t", "Chi-squared", "F"), title = "Distribution",
             action = tables.draw, pos = c(25, 15, 100, 120))
rp.doublebutton(tables.panel, degf1, 1, range = c(1, NA), title = "df1",
             action = tables.draw, pos = c(140, 10, 100, 50))
```

Earlier examples have described code which can be executed directly to produce desired panels and graphical effects. However, it is clearly advantageous to be able to produce functions

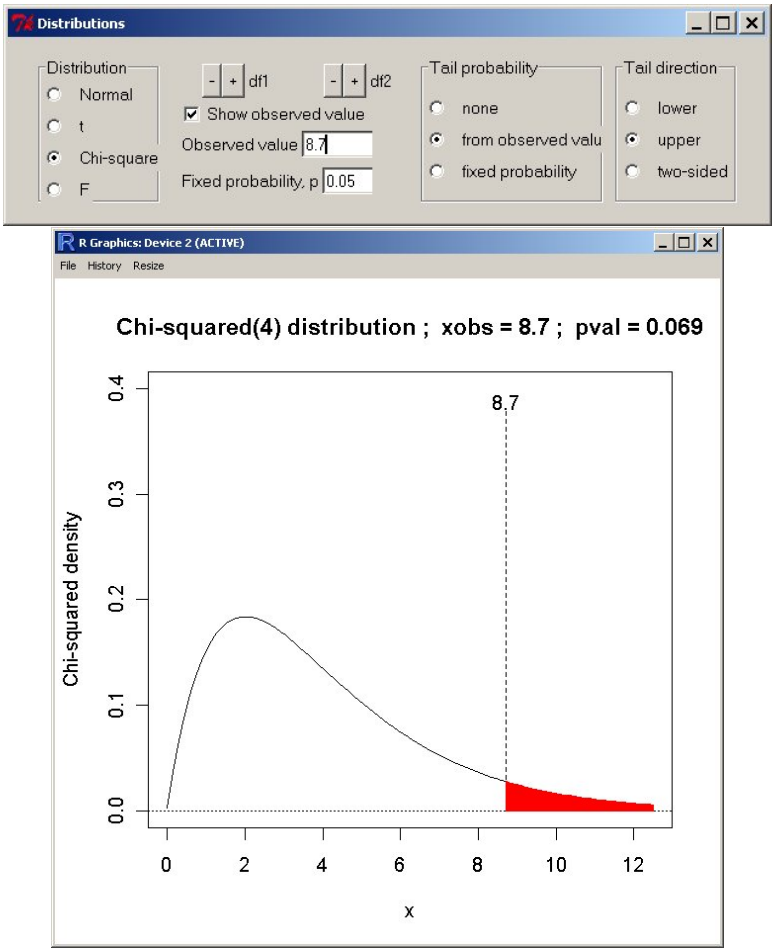

Figure 5: A panel control for electronic tables.

which contain **rpanel** calls but which have the flexibility of producing graphics based on data which are passed through arguments in the usual way. **rpanel** addresses the issues created by defining panels and action functions in environments other than the global one, for example within a function, by holding the master copy as a hidden object in the dedicated **rpanel** environment `.rpenv`. The following code illustrates this with two-dimensional density estimation, where both the graphical form of the display, and a multiplier of the normal-optimal smoothing parameter, can be controlled, as shown in Figure 6.

```
rp.density2d <- function(x, xlab = deparse(substitute(x))) {
   density.panel <- rp.control("Density estimation",
                       x = x, sp = hnorm(x), hmult = 1)
   rp.radiogroup(density.panel, display,
                       c("image", "persp", "slice"),
                       title = "Display", action = density2d.draw)
   rp.slider(density.panel, hmult, 0.5, 2,
                       density2d.draw, "Bandwidth multiplier", log = TRUE)
   }
```

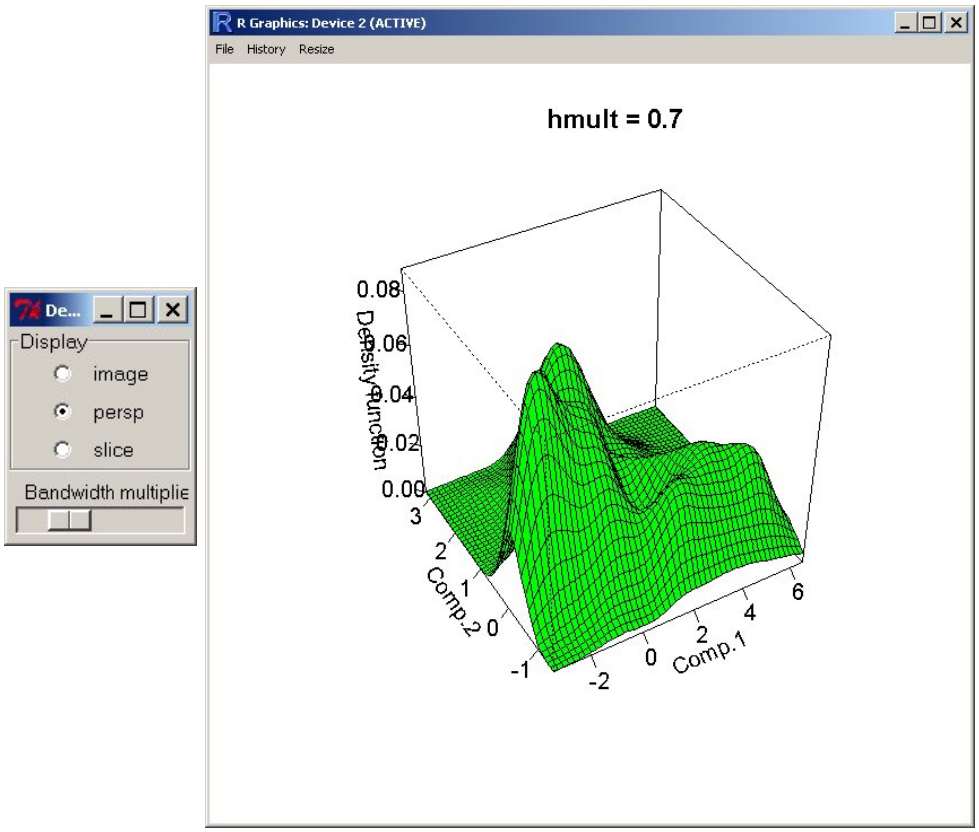

Figure 6: A panel control for a two-dimensional density estimate.

The panel created by this function clearly cannot have a fixed name as this would create conflicts between repeated calls. A simple solution is provided by the automatic generation by `rp.control` of name of the form `.rpanelx`, where `x` denotes a random eight-digit integer. A check is also made that a panel of this name does not already exist.

An advantage of this mode of operation, and the fact that each panel contains its own data and controlling variables, is that repeated calls to the same function will create multiple panels which have no difficulty in running simultaneously. For example, it could be very helpful to generate two panels controlling density estimates on different data, or even density estimates involving different degrees of smoothing applied to the same data. Each panel can easily be associated with a different R graphics window by adding the value of `dev.cur()` to the panel object.

5. Incorporating graphics into a panel

The examples described above use the native plotting facilities of R which are displayed in a standard R graphics window. There are occasions when it would be helpful to create a single panel which incorporates the plot. For example, utilities such as the power function plotter described in Section 3 or the electronic tables described in Section 4 could be launched as separate applications which can be controlled by users who have no knowledge of R. The **tkrplot** package developed by Tierney (2005) provides a simple means of doing this, by incorporat-

ing R graphics into a Tcl/Tk panel. The following code illustrates this with nonparametric regression.

```
smooth.panel <- rp.control("Nonparametric regression", size = c(600, 450),
             x = x, y = y, xlab = "Covariate", ylab = "Response",
             h = diff(range(x)) / 8)
rp.doublebutton(smooth.panel, h, h.delta, pos = c(25, 25, 100, 30),
             range = c(diff(range(smooth.panel$x)) / 50, NA),
             title = "Bandwidth", action = replot.smooth)
rp.radiogroup(smooth.panel, model, pos = c(25, 75, 100, 100),
             c("none", "no effect", "linear"),
             title = "Reference", action = replot.smooth)
rp.tkrplot(smooth.panel, plot, plot.smooth, pos = c(150, 0, 500, 450))
```

The `plot.smooth` function uses standard R commands to draw the graph while `replot.smooth`, shown below, uses the **tkrreplot** function to place each new version of the graph within the panel.

```
replot.smooth <- function(panel) {
   rp.tkrreplot(panel, plot)
   panel
   }
```

There is a minor difficulty with this procedure in Microsoft Windows, where the focus continually returns to the R window. This is not a problem if the R window does not overlap with the **rpanel** window.

Integrated graphics are also potentially useful when running in batch mode, to create an independent graphical application. This requires R to be put into an infinite loop, so that the window remains visible and that is achieved by the function `rp.block(panel)`. Note that batch operation is not appropriate in Linux, where the `X11` driver cannot be activated in batch mode.

The ability to integrate panel controls and graphics is a very powerful one with considerable attractions from a design perspective. The creation of independent applications, with R running in the background, is also very appealing as this allows software for particular types of analysis to be used without knowledge of R itself. The development of sophisticated and flexible tools of this type is a great deal of work. However, the ability to 'package' simple applications and illustrations for use by others under GUI control, with integrated graphics through **tkrplot**, has extensive potential applications.

The ability to place an image within a panel raises interesting possibilities of other types of interaction. Bowman and McColl (1999) describe materials provided by the STEPS project which are built around a problem-solving approach to teaching. Bowman, Currall, and Lyall (1997) describe one particular module involving data collection on herring gulls which requires students to identify ways of reliably determining the sex of a bird. Software, written in a commercial authoring tool, is available to allow students to tackle the data collection activity.

The code below shows how the essential operations can be undertaken by creating a panel in R. This incorporates an image of a herring gull, through the `rp.image` function. A

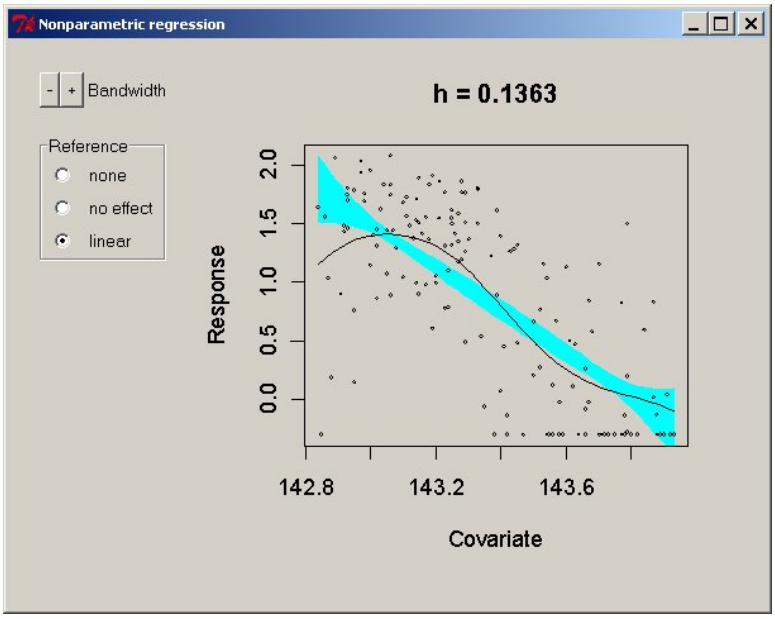

Figure 7: A panel for nonparametric regression which integrates the graphics with the controls, using `tkrplot`.

`click.capture` function, which has three arguments `(panel, x, y)`, allows the pixel position of a point on the image which has been clicked to be captured. Standard R code can then be used to identify whether this position is close to one of the marked landmarks on the bird. This allows students to nominate two landmarks to identify a length measurement. When the *Collect data* button is pressed, the `collect.data` function either provides suitable feedback on why this measurement is not an appropriate one or adds the measurement from a database of male and female birds to a dataframe. The function `rp.gulls` has built this idea into a freestanding application which includes buttons to plot and explore the data which have been collected. However, an alternative mode of use is to allow students direct access to the data, which can then be analysed by them in R, or indeed any other statistical computing environment. These facilities therefore provide, in relatively simple R code which could be written by many R users, the essential operations of an activity which is usually associated with a much more sophisticated software environment.

```
gulls.panel <- rp.control("STEPS:  The Birds and the Bees",
                gulls.all = gulls.all,
                lmk.names = c("Wing tip", "Tail feathers", "Wing joint",
                              "Bottom of bill", "Tip of beak", "Top of bill",
                              "Top of head", "Back of head"),
                lmks.x = c( 25,  40, 218, 417, 449, 436, 362, 330),
                lmks.y = c(134, 167, 183,  79,  78,  52,  11,  23),
                lmk1 = NA, lmk2 = NA)
rp.image(gulls.panel, "gulllmks.gif",
                id = "gulls.image", action = click.capture)
rp.button(gulls.panel,
                title = "Collect data", action = collect.data)
```

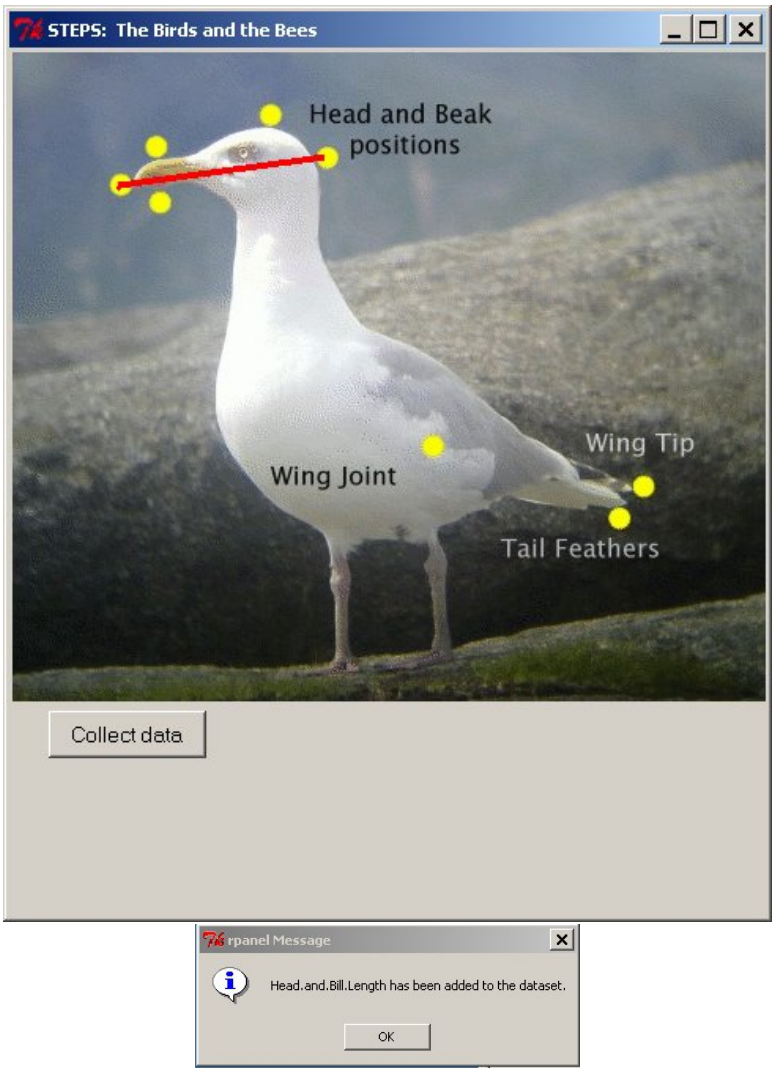

Figure 8: This panel gives students the opportunity to engage in data collection by selecting landmarks to define length measurements which may be able to distinguish male and female herring gulls.

6. Discussion

The examples of the paper have provided a wide range of illustrations where graphical interaction can be used to very good effect. The aim of the **rpanel** package is to make these types of interaction as accessible as possible to R users, by providing a simple 'front end' to the more general and flexible **tcltk** package. Programmers who are familiar with the Tcl/Tk system are likely to continue to use the **tcltk** package directly, to take full advantage of the facilities which that system provides. However, the **rpanel** package will enable and encourage a wider range of R users to experiment in creating new interactive graphs of their own.

We plan to develop the **rpanel** package further, to extend the facilities for the creation of control widgets. Recent additions include the functions `rp.listbox` and `rp.menu` which allows action functions to be called through listbox and menu items respectively. However,

it should be emphasised that the package does not provide a means by which all types of dynamic graphics can be constructed. For example, current methods of communication with R plots do not allow linking and brushing, for which the Java-based **iplots** package of Urbanek and Theus (2003) is required. However, the ability to rapidly redraw plots under GUI control has great potential. Further **rpanel** examples are available in the *R News* article by Bowman, Crawford, and Bowman (2006) and at http://www.stats.gla.ac.uk/~adrian/rpanel/.

Acknowledgements

Part of the project was pursued under the auspices of the Mathematics, Statistics and OR Network of the Higher Education Academy whose funding is gratefully acknowledged. We are also grateful for numerous helpful discussions with Simon Urbanek who motivated our interest in this topic and who has been the source of very helpful advice. James Wettenhall's collection of **tcltk** examples at http://bioinf.wehi.edu.au/~wettenhall/RTclTkExamples/ was also very useful. Finally, we would like to acknowledge the very helpful comments and suggestions of an associate editor and two referees.

References

Adler D (2005). **rgl**: *3D Visualization Device System (OpenGL)*. R package version 0.65, URL http://CRAN.R-project.org/.

Bowman A, Azzalini A (1997). *Applied Smoothing Techniques for Data Analysis*. Oxford University Press, Oxford.

Bowman A, Crawford E, Bowman R (2006). "**rpanel**: Making Graphs Move with **tcltk**." *R News*, **6**(4). URL http://www.R-project.org/.

Bowman A, Currall J, Lyall R (1997). "The Birds and the Bees: Interactive Graphics and Problem Solving in the Teaching of Statistics." *Statistics and Computing*, **7**, 237–246.

Bowman A, McColl J (eds.) (1999). *Statistics and Problem Solving*. Arnold, London.

Bowman A, Robinson D (1990). *Introduction to Regression and Analysis of Variance: A Computer Illustrated Text*. IOP, Bristol.

Cook R, Weisberg S (1999). *Applied Regression Including Computing and Graphics*. Wiley, New York.

Dalgaard P (2001). "The R-Tcl/Tk Interface." In K Hornik, F Leisch (eds.), "Proceedings of the 2nd International Workshop on Distributed Statistical Computing, March 15-17, 2001, Technische Universität Wien, Vienna, Austria," ISSN 1609-395X, URL http://www.ci.tuwien.ac.at/Conferences/DSC-2001/Proceedings/.

Dugmore A, Larsen G, Newton A, Sugden D (1992). "Geochemical Stability of Finegrained Silicic Tephra in Iceland and Scotland." *Journal of Quaternary Science*, **7**, 173–183.

Fox J (2005). "The R Commander: A Basic-Statistics Graphical User Interface to R." *Journal of Statistical Software*, **14**(9). URL http://www.jstatsoft.org/v14/i09/.

Mächler M (2005). *The* `tkdensity` *Function*. In the R package **sfsmisc**: Utilities from Seminar für Statistik, ETH, Zürich, URL http://CRAN.R-project.org/.

Pickering J (1985). "An Analysis of Giving in the Church of England." *Applied Economics*, **17**, 619–632.

Raz J (1989). "Analysis of Repeated Measurements Using Nonparametric Smoothers and Randomization Tests." *Biometrics*, **54**, 851–871.

R Development Core Team (2006). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org/.

Temple Lang D (2004). **RGtk**: *Initial R Bindings for Gtk*. URL http://www.omegahat.org/RGtk/.

Temple Lang D, Lawrence M (2006). **RGtk2**: *R Bindings for Gtk 2.0*. R package version 2.8.6, URL http://www.ggobi.org/rgtk2/.

Tierney L (1990). *Lisp-Stat: an Object-oriented Environment for Statistical Computing and Dynamic Graphics*. Wiley, New York.

Tierney L (2005). **tkrplot**: *Simple Mechanism for Placing R Graphics in a Tk Widget*. R package version 0.0-10, URL http://CRAN.R-project.org/.

Urbanek S, Theus M (2003). "iPlots – High Interaction Graphics for R." In K Hornik, F Leisch, A Zeileis (eds.), "Proceedings of the 3rd International Workshop on Distributed Statistical Computing, March 20-22, 2003, Technische Universität Wien, Vienna, Austria," ISSN 1609-395X, URL http://www.ci.tuwien.ac.at/Conferences/DSC-2003/Proceedings/.

Whalen E (2006). **iSPlot**: *Linking Plots*. URL http://www.bioconductor.org/packages/1.9/bioc/html/iSPlot.html.

Zhang J (2006). **tkWidgets**: *R based Tk Widgets*. URL http://www.bioconductor.org/packages/1.9/bioc/html/tkWidgets.html.

A. A list of the main rpanel functions

The principal functions available for the construction of panels are listed below.

`rp.control`	Creates a panel object and initialises its content
`rp.slider`	Adds a slider control for a nominated variable and determines the function to be called when the slider is moved
`rp.button`	Adds a button to the panel and identifies the function to be called when the button is pressed
`rp.doublebutton`	Adds a double-button to the panel and identifies the associated variable and the function to be called when a button is pressed
`rp.checkbox`	Adds a checkbox to the panel and identifies the associated logical variable and the function to be called when the box is checked
`rp.radiogroup`	Adds a set of radiobuttons to the panel and identifies the associated character variable and the function to be called when a button is pressed
`rp.listbox`	Adds a scrollable list of items and identifies the associated variable and the function to be called when an item is selected
`rp.menu`	Adds a scrollable list of items and identifies the associated variable and the function to be called when an item is selected
`rp.textentry`	Adds a box through which text can be entered and assigned to a nominated variable and identifies the function to be called when this happens
`rp.tkrplot`	Calls the `tkrplot` function of Tierney (2005) to place R graphics inside a panel
`rp.tkrreplot`	Calls the `tkrreplot` function of Tierney (2005) to update the R graphics inside a panel
`rp.messagebox`	Displays a modal pop-up box with specified text
`rp.image`	Adds an image to the panel
`rp.line`	Adds a line to a panel image
`rp.deleteline`	Removes an individual line from a panel image
`rp.clearlines`	Removes all lines from a panel image
`rp.block`	Prevents R from terminating, when running in batch mode, until the panel is closed
`rp.do`	Invokes an action function for a panel.

The following functions are available in the **rpanel** package to provide specific interactive facilities. They therefore also provide examples of the use of **rpanel** functions in creating interactive material.

A collection of other scripts, of interest mostly in a teaching context, is available at http://www.stats.gla.ac.uk/~adrian/rpanel/. The list below shows those available at the time of publication of the paper.

`rp.power`	The power of a two-sample t-test as a function of population parameters and sample size
`rp.rmplot`	Repeated measurements data displays
`rp.tables`	Graphical statistical tables
`rp.ancova`	Interactive display of analysis of covariance models
`rp.plot3d`	Three-dimensional scatterplot, using the `rgl` package
`rp.regression2`	Interactive display of regression models with two explanatory variables
`rp.gulls`	A teaching module on identifying the sex of herring gulls

Power transformations and a q-q plot
Plotting binomial distributions
Simulated confidence intervals
Simulations of correlated data
Rotation of a persp plot
Density estimation (1d) (requires the **sm** package)
Density estimation (2d) (requires the **sm** package)
Exponential likelihood
Simple regression
Nonparametric regression (1d) (requires the **sm** package)
Basis functions for nonparametric regression (requires the **splines** package)
Cosine regression
Interacting with a map

Interested readers are invited to submit further examples for posting.

Affiliation:

Adrian Bowman
Department of Statistics
University of Glasgow
Glasgow G12 8QQ, United Kingdom
E-mail: adrian@stats.gla.ac.uk
URL: http://www.stats.gla.ac.uk/~adrian/

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/
Volume 17, Issue 9 *Submitted:* 2006-08-29
January 2007 *Accepted:* 2007-01-01