```
symbolbox
 Draw a box filled with symbols
textbox
 Display justified text in an optional box
thigmophobe.labels
 Place labels away from the nearest other point
triax.plot
 Display a triangle (three axis) plot
```

The proliferation of packages and functions for R has led to many similar functions in different packages. The `radial.plot` family, including `radial.plot`, `polar.plot` and `clock24.plot` has relatives including `rosavent` in **climatol**, (Guijarro, 2004) `plot.circular` in **circular** (Agostinelli, 2005) and `rose.diag` in **CircStats** (Agostinelli, 2005). The presence of similar functions in different packages that give the user more choices leads to the ensuing difficulty of finding the functions that afford the choices. There have been several attempts to minimize the latter effect. My own favorite is arbitrary text searching such as that implemented by Professor Jonathon Baron's R Site Search facility (Baron, 2006).

A parting glance at style

plotrix has a strong collaborative influence, and it is hoped that this will continue. The programming style leans toward the explicit rather than being highly condensed or efficient, and many R users have contributed not only suggestions but segments of code that have been incorporated. Those experienced in the more arcane techniques of illustration have offered valuable comments on improving certain functions. The overall utility of **plotrix** is largely a product of this expert feedback. A possible extension of **plotrix** would be to add compatibility with the more flexible **grid** package plotting functions.

The aim of **plotrix** is not to erode the high standards of R but to allow its users to perform the more mundane tasks of their calling with less effort. The consumer who accepts that R can produce the comforting pictures to which he or she is accustomed may well be seduced into the more challenging illustrations of data. It is hoped that **plotrix** and similar packages will provide functions that allow the new user to demonstrate the basic capabilities of R as rapidly as any other statistical software and the experienced user to generate the common varieties of data illustration more conveniently.

Bibliography

C. Agostinelli. *circular: Circular statistics.* URL `http://cran.r-project.org/src/contrib/Descriptions/CircStats.html`

C. Agostinelli. *CircStats: Circular statistics.* URL `http://cran.r-project.org/src/contrib/Descriptions/circular.html`

J. Baron. *R Site Search* URL `http://finzi.psych.upenn.edu/search.html`

K.W. Deininger. *Measuring Income Inequality Database.* URL `http://siteresources.worldbank.org/INTRES/Resources/469232-1107449512766/648083-1108140788422/A_New_Dataset_Measuring_Income_Inequality.zip`

J.A. Guijarro. *Climatol: some tools for climatology.* URL `http://cran.r-project.org/src/contrib/Descriptions/climatol.html`

J. Lemon. *plotrix: Various plotting functions.* URL `http://cran.r-project.org/src/contrib/Descriptions/plotrix.html`

F. Ng. *The Division of Wealth* URL `http://www.progressiveschoolhouse.org/wealth/wealthdivisions.htm`

S. Penel. *ade4: Multivariate data analysis and graphical display.* URL `http://cran.r-project.org/src/contrib/Descriptions/ade4.html`

Jim Lemon
bitwrit software, Gladesville, Australia
`jim@bitwrit.com.au`

rpanel: making graphs move with tcltk

by Adrian Bowman, Ewan Crawford, and Richard Bowman

The command line interface of R provides a flexibility and precision which is very well suited to many settings. Alternatively, at the introductory level, where point-and-click interfaces can be useful, *gui* interfaces such as **R Commander** described by Fox (2005) are available. However, in between these two modes of use there is sometimes a very useful role for interactive control of R operations, particularly where graphics are involved. The **tcltk** package of Dalgaard (2001), which provides a link from R to the *Tcl/Tk* system, helpfully provides a very extensive set of tools for this purpose and this system has featured regularly in the pages of *R News*.

The aim of the **rpanel** package is to provide a simple set of tools such as buttons, sliders, checkboxes and textentry boxes, each of which can be created by a single function call, with full documentation. The package essentially provides a set of wrappers for underlying **tcltk** operations which are intended to make access to these controls as simple as possible. In particular, the **tcltk** variables which are the basis of communication between the control panel and R are managed behind the scenes in a way that is transparent to the user. Some simple facilities for interacting with images are also provided.

Tools for interactive controls within R is a rapidly developing area, with some exciting prospects. The **iplots** package of Urbanek and Theus (2003) offers genuine interaction with Java-based plots, while the **RGtk2** package of Lang and Lawrence provides a set of gui building-blocks that is similar to **tcltk**. The main emphasis of the **rpanel** package is to provide a simple and easy-to-use set of controls which can enhance the standard R plots which form familiar territory to the majority of R users.

A simple example

A simple example is provided by the application of a Box-Cox power transformation in the context of a Q-Q plot for normality. Tierney (1990) used this as an illustration of the dynamic graphics available in the Lisp-Stat system. The R code below defines a power transformation function `bc.fn` and a simple plotting function `qq.draw` which applies the transformation and passes the result to `qqnorm`. The final two statements set up a control panel and a slider. Movement of the slider causes the 'action' function `qq.draw` to be called with the current setting of the variable `lambda`. This simply redraws the plot to create an animation effect.

```
bc.fn <- function(y, lambda) {
   if (abs(lambda) < 0.001) z <- log(y)
   else z <- (y^lambda - 1)/ lambda
   }
qq.draw <- function(panel) {
   z <- bc.fn(panel$y, panel$lambda)
   qqnorm(z, main = paste("lambda =",
          round(panel$lambda, 2)))
   panel
   }
panel <- rp.control(y = exp(rnorm(50)),
               lambda = 1)
rp.slider(panel, lambda, -2, 2, qq.draw)
```

An illustration is given in Figure 1, where the lower plot shows the untransformed data (`lambda` = 1) and the upper plot show the effect of a log transformation. Since the data were simulated as exponentially transformed normal random variables, this is the appropriate transformation back to normality. Here `qq.draw` has been modified in a simple way to add a histogram of the transformed data in a second panel of the plot. It would be easy to amend the code in other ways, such as drawing the plot in a different colour for values of `lambda` that lie inside the likelihood-based 95% confidence interval for the power transformation parameter.

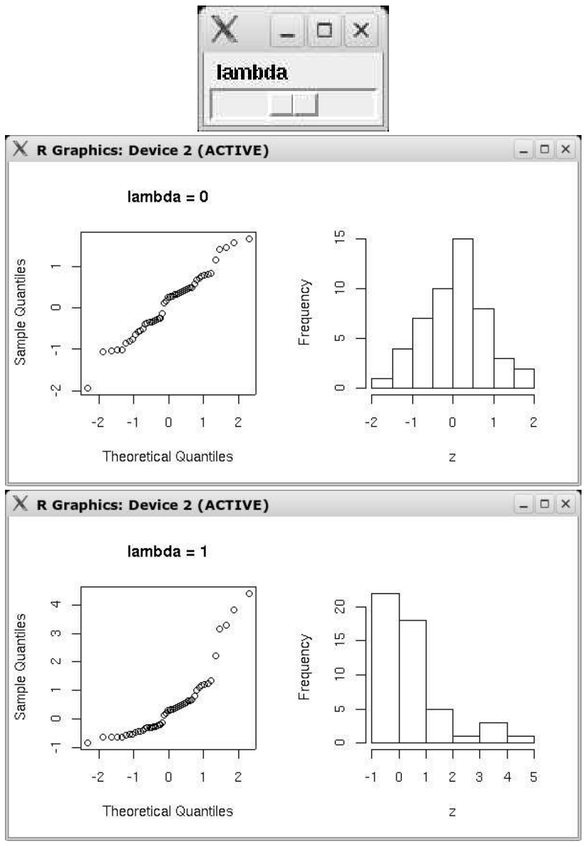


Figure 1: Slider control of a power transformation for a normal q-q plot.

Those who use the **tcltk** package will be entirely familiar with this kind of facility. It should be emphasised again that the principal aim of the **rpanel** package is simply to make access to these facilities as straightforward as possible for those who are less well placed to become familiar with the *Tcl/Tk* system. The user can then direct the majority of attention to the construction of the appropriate action functions. The `panel` object created by the `rp.control` function contains the data to be used by the action functions, including the variables associated with subsequent widgets (buttons, sliders, etc.). An advantage of this construction is that the underlying communication variables can be managed in a manner which allows the use of functions and other multiple instances of the same R code, without creating communication difficulties.

Note that it is important that action functions are defined before the **rpanel** functions which create the panel controls are called. This is because the action functions are attached to the panel object for later execution. It is also important that each action function returns the R panel object. This is a feature of the communication mechanism used by the package. It is clearly particularly relevant in situations where an action function may alter the contents of the panel object and so this needs to be made available for subsequent use.

Further examples

A full description of the **rpanel** package is provided in a paper which is available at `www.stats.gla.ac.uk/~adrian/rpanel/`. A set of illustration scripts, including the code for the examples of this article, can also be found there. One or two further illustrations of the package and its potential uses are given here.

The `persp` function in R allows three-dimensional perspective plots to be created, with the viewing angle controlled by the arguments `theta` and `phi`. It can be helpful to rotate the surface to examine features that are hidden by the current perspective. The code below creates an action function `volcano.plot` which simply draws the graph with the current values of `theta` and `phi`. The use of the `with` environment helpfully allows components of the `panel` object to be referred to directly. Two doublebuttons are then inserted into a control panel. When pressed, the relevant parameter is incremented or decremented by 10 degrees and the plotting function called. With a plotting task of this complexity, the refresh rate is not sufficiently fast to create a very smooth animation. Nonetheless, the interactive control is very useful in visual examination of the surface.

```
z <- 2 * volcano
x <- 10 * (1:nrow(z))
y <- 10 * (1:ncol(z))
volcano.plot <- function(panel) {
  with(panel, {
    persp(x, y, z, theta = theta, phi = phi,
      col = "green3", scale = FALSE,
      ltheta = -120, shade = 0.75,
      border = NA, box = FALSE)
    })
  panel
  }
volcano.panel <- rp.control(x = x, y = y,
         z = z, theta = 135, phi = 30)
rp.doublebutton(volcano.panel, theta, 10,
         action = volcano.plot)
rp.doublebutton(volcano.panel, phi,   10,
         action = volcano.plot)
rp.do(volcano.panel, volcano.plot)
```

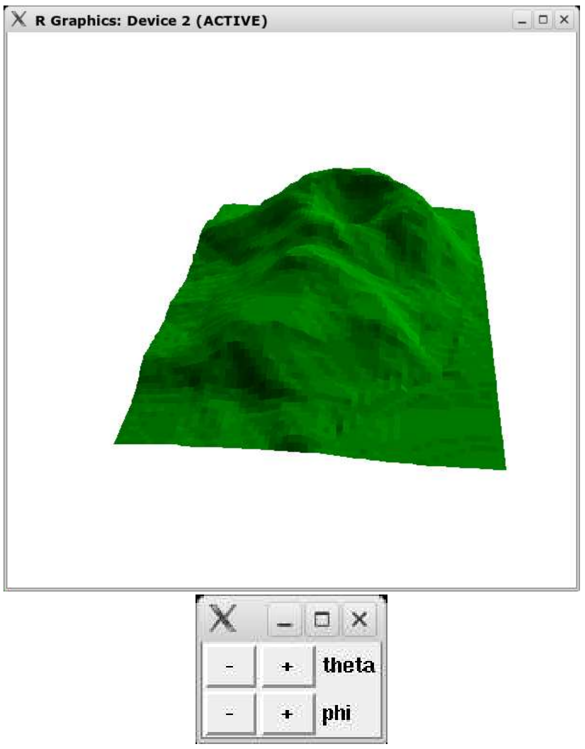
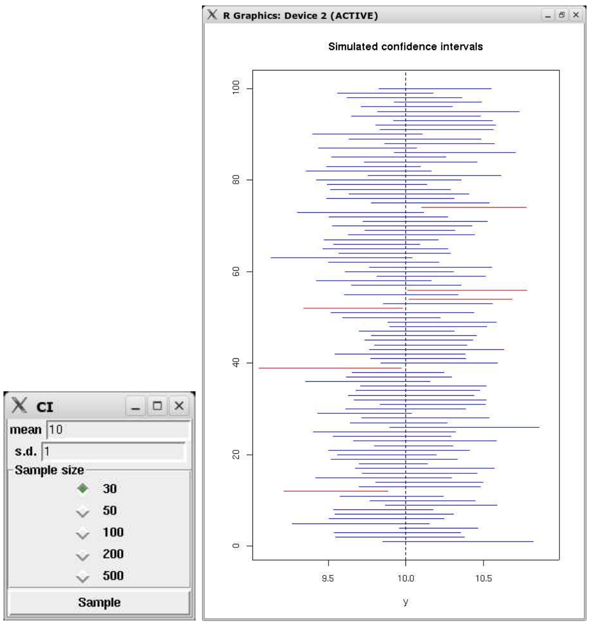


Figure 2: Doublebutton control of the viewing angles of a persp plot.

A further example involves the simulation of confidence intervals. This can be a very useful exercise in a teaching context, to communicate the meaning of 'confidence' through the long-run proportion of constructed intervals that contain the true value of the parameter of interest. Figure 3 shows a control panel which contains text-entry boxes to supply the mean and standard deviation of the normal population from which the data are sampled, radiobuttons to specify the sample size and a button to create a set of confidence intervals. The code to create the controls is straightforward and is shown below. The code for the action function is not shown but involves simple plotting procedures which are very familiar to the majority of R users. The teacher is therefore able to create the plot and its controls to suit personal preferences. Here the intervals that miss the true value of the parameter are coloured red.

There would be no difficulty in turning the code for this example into a function by adding the necessary wrapping statements. Repeated calls of the function would create independent control panels with their own communication mechanisms. It would clearly be neater in that case to add an `x11()` statement to launch a new graphics window for each call of the function. The use of `dev.cur` and `dev.set` would then allow the appropriate graphics window to be activated by each control panel, using an identifier contained in the R panel object. The binomial example on the `rpanel` web site gives an example of this.

```
ci.panel <- rp.control("CI", mu = 10,
   sigma = 1,
   sample.sizes = c("30","50","100","200",
   "500"))
rp.textentry(ci.panel, mu,
   title = "mean", action = ci.plot)
rp.textentry(ci.panel, sigma,
   title = "s.d.", action = ci.plot)
rp.radiogroup(ci.panel, ssize,
   c(30, 50, 100, 200, 500),
   title = "Sample size",
   action = ci.plot)
rp.button(ci.panel, title = "Sample",
   action = ci.plot)
```


CI

mean 10

s.d. 1

Sample size: [✓ 30] 50 100 200 500

Sample

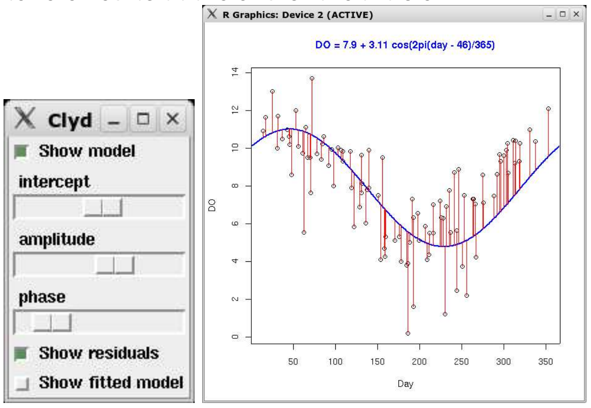
Figure 3: Panel control of simulated confidence intervals.

Again in a teaching setting, it can be very helpful to use graphics to communicate the form of a model, the meaning of its parameters and the process of fitting the model to observed data. Figure 4 shows data on dissolved oxygen (DO) from the River Clyde in Scotland, measured at a particular sampling station over a period of several years. (The data were kindly provided by Dr. Brian Miller of the Scottish Environment Protection Agency.) The data are plotted against day of the year and a clear seasonal pattern is evident. A simple model for this is based on a shifted and scaled cosine curve as

$$y_i = \alpha + \beta \cos((x_i - \gamma)2\pi/366) + \varepsilon_i.$$

In order to illustrate the meaning of this model to students, it is helpful to create a control panel which allows the values of the intercept (α), scale (β) and phase shift (γ) to be altered by sliders. The intuitive meaning of each parameter is then clear. A checkbox has been used to allow the residuals to be displayed, leading to the concept of least squares fitting. A further checkbox allows the fitted model to be shown. (Expansion of the *cos* term in the model shows that it is in fact a linear model, but that is not the central issue in the illustration.)

The code for this example has been omitted. However, it involves a simple plotting function whose display is controlled by the logical flags associated with the checkboxes and the values of the parameters set by the sliders. Each widget in the control panel is created by a single call to the appropriate **rpanel** function. The full code is available on the web site referred to at the end of the article.


Clyd

[✓ Show model]

intercept

amplitude

phase

[✓ Show residuals]

Show fitted model

R Graphics: Device 2 (ACTIVE)
DO = 7.9 + 3.11 cos(2pi(day - 46)/365)
DO
0
2
4
6
8
10
12
14
50
100
150
200
250
300
350
Day

Figure 4: Panel control of the parameters of a cosine regression model.

Using images

A more unusual example involves interaction with images. Figure 5 shows a map of the Clyde estuary with a number of sampling points superimposed. (A more detailed high quality map could be used but this rather simple version has been used to avoid copyright issues.) Underneath the map a doublebutton has been inserted to allow movement along the sampling stations. The current station is marked by a yellow cross and the data from that location are plotted in the R graphics window. This allows the changing pattern of overall level and seasonal effect to be examined in an animated manner.

However, it is also possible to click on the map to identify a sampling station and plot the corresponding data. This gives a greater degree of interactive control. The code used for this is shown below. The `clyde.plot` function marks the nominated sampling station and produces the R plot of the corresponding data. The `click.action` function reacts to a click on the map image and checks whether this is close to a sampling station, in which case this is made the current station. The `clyde.plot` function is then called. The `rp.image` function inserts the map image into the control panel while the `rp.clearlines` and `rp.line` functions are used to mark the current sampling station. The detailed use of these functions can be seen from the package documentation.

```
clyde.plot <- function(panel) {
  with(panel, {
    rp.clearlines(panel, clyde.image)
      rp.line(panel, clyde.image,
              station.x[station] - 10,
              station.y[station] - 10,
              station.x[station] + 10,
              station.y[station] + 10,
              width = 4, color = "yellow")
      rp.line(panel, clyde.image,
              station.x[station] + 10,
              station.y[station] - 10,
              station.x[station] - 10,
              station.y[station] + 10,
              width = 4, color = "yellow")
    ind <- (Station == station)
    plot(DO[ind] ~ Day[ind], ylim = range(DO),
         xlab = "Day", ylab = "DO")
    })
  panel
  }
click.action <- function(panel, x, y) {
  x <- as.numeric(x)
  y <- as.numeric(y)
  distance <- sqrt((x - panel$station.x)^2 +
                   (y - panel$station.y)^2)
  if (min(distance) <= 25) {
    panel$station <-
            which(distance == min(distance))
    clyde.plot(panel)
    }
  panel
  }
clyde.panel <- rp.control("Clyde",
  station = 1,
  station.x = c(913, 849, 791, 743, 695, 660,
           600, 555, 485, 407, 333, 249, 167),
  station.y = c(550, 522, 502, 467, 432, 392,
           362, 312, 306, 294, 274, 249, 218),
  Day = Day, Station = Station, DO = DO)
rp.image(clyde.panel, "clyde.gif",
  id = "clyde.image", action = click.action)
rp.doublebutton(clyde.panel, station, 1,
  action = clyde.plot, range = c(1, 13))
rp.do(clyde.panel, clyde.plot)
```

Discussion

The interactive control of R graphics, or indeed R functions more generally, has considerable potential in a variety of settings. One of the main uses is the creation of animation effects through instantaneous redrawing when control widgets are activated. There are many potential applications of this in the graphical exploration of data, in teaching and in the creation of stand-alone tools for particular applications which can be used by those without knowledge of R.

All of the facilities required for these tasks are already available in the **tcltk** package. The aim of the **rpanel** package is to make a useful subset of these as widely available as possible by providing functions to create control widgets in single calls, with full documentation. In addition, the communication mechanism allows functions that create control panels to be written in the usual way. R graphics can also be incorporated into a control panel using the **tkrplot** package of Tierney.

A more extensive description of the package is available at

```
www.stats.gla.ac.uk/~adrian/rpanel/
```

where a collection of further examples is also located.

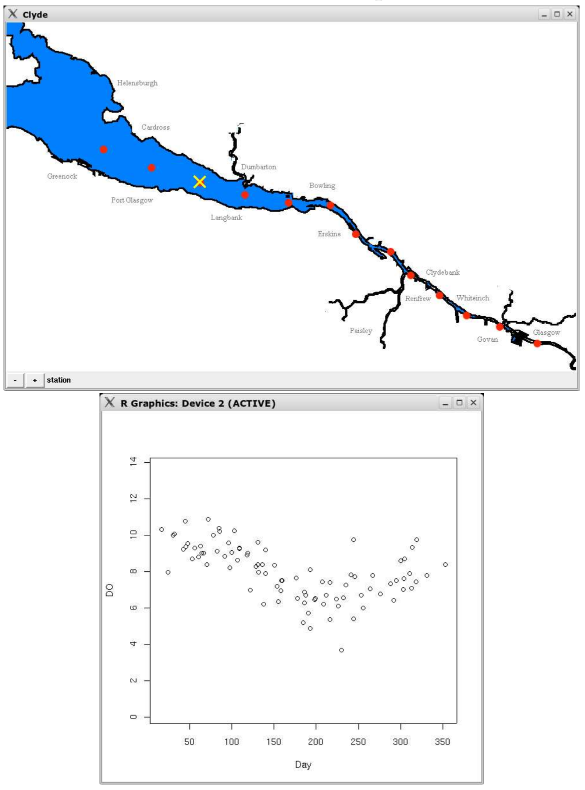


Figure 5: Use of a map to identify the station for which data should be plotted (the marked positions of the sampling locations are approximate, for the purposes of illustration).

Bibliography

P. Dalgaard. The R-tcl/tk interface. In *Proceedings of the 2nd International Workshop on Distributed Statistical Computing (DSC 2001)*, eds. K.Hornik, F.Leisch. Vienna, March 2001. ISSN: 1609-395X, 2001.

J. Fox. The R commander: a basic-statistics graphical user interface to R. *Journal of Statistical Software*, 14, 2005. URL http://www.jstatsoft.org/.

D. T. Lang and M. Lawrence. **RGtk2**: R bindings for GTK 2.0. URL http://cran.r-project.org/.

L. Tierney. *Lisp-stat: an object-oriented environment for statistical computing and dynamic graphics*. Wiley: New York, 1990.

L. Tierney. **tkrplot**: simple mechanism for placing R graphics in a tk widget. URL http://cran.r-project.org/.

S. Urbanek and M. Theus. **iPlots**: high interaction graphics for R. In *Proceedings of the 3rd International Workshop on Distributed Statistical Computing (DSC 2003)*, eds. K.Hornik, F.Leisch, A.Zeileis. Vienna, March 2003. ISSN: 1609-395X, 2003.

Adrian Bowman
Department of Statistics, University of Glasgow
adrian@stats.gla.ac.uk

Ewan Crawford
Department of Statistics, University of Glasgow
ewan@stats.gla.ac.uk

Richard Bowman
Churchill College, University of Cambridge
rwb27@cam.ac.uk

R's role in the climate change debate

by Matthew Pocernich

A contentious issue within the climate research community is the validity of a sequence of reconstructions of global surface temperatures which exhibit a hockey stick-like appearance. These graphs show relatively modest hemispheric temperature variations over the past millennium followed by a sharp increase over the 20th century (See Figure 1). Climate data collected instrumentally is largely limited to the last 150 years. For information prior to this, scientists rely on proxy sources such as tree rings, ice core samples and other types of information. The reconstructions created by Mann et al. (1998) and Mann et al. (1999) have attracted particular attention due to their prominence in the last International Panel on Climate Change reports (IPCC, 2001). Various criticisms against the Mann et al. reconstructions have been voiced over the years, but most recently, articles beginning with McIntyre and McKitrick (2003) questioned whether principal component analysis (PCA) was correctly used to summarize many sources of proxy data. The contention was that the implementation of PCA using a common centering convention limited to the overlap with the instrumental data rather than using a full-length centering could potentially produce a spurious offset, even in random data. This criticism has been transformed into a much broader challenge regarding the existence of climate change.

The National Research Council (NRC) established a committee to re-evaluate Mann's work in light of the subsequent criticisms. On June 22nd, a press conference was held to announce the findings of the report and answer questions. The Chair of the committee, Dr. Gerald North, made several remarks relating sharing code and data (Board on Atmospheric Sciences and Climate, 2006).

> Our view is that all research benefits from full and open access to published datasets and that a clear explanation of analytical methods is mandatory. Peers should have access to the information needed to reproduce published results, so that increased confidence in the outcome of the study can be generated inside and outside the scientific community. Other committees and organizations have produced an extensive body of literature on the importance of open access to scientific data and on the related guidelines for data archiving and data access (e.g., NRC 1995). Paleoclimate research would benefit if individual researchers, professional societies, journal editors, and funding agencies continued to improve their efforts to ensure that these existing open access practices are followed.

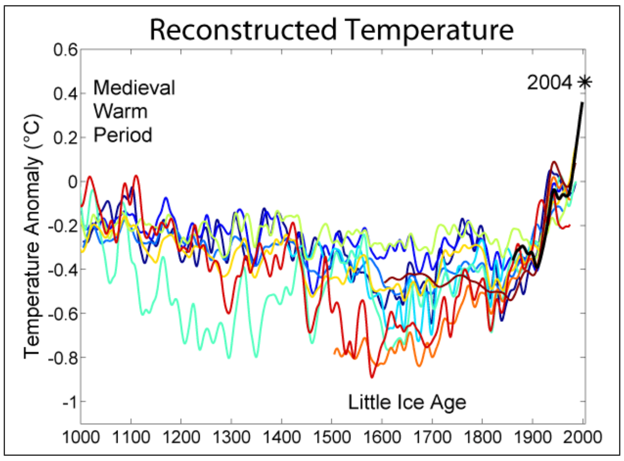


Figure 1: Reconstructed temperatures for the last millennium. Wahl and Ammann (2006)

The statistical and scientific details of this debate are not the point of this note, but rather the ways in which R has had a role in the dialog. A figure in the NRC report was created using R to illustrate how spurious trends could result from the use of princi-