
Statistical Cartoons in R

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



Introduction

The title of this article might encourage the reader to look forward to some humorous drawings, along the lines of Mickey Mouse or Charlie Brown. Sadly, the abilities of this author fall woefully short of such a task. There is a much older use of the word 'cartoon', referring to the prototypes or trial drawings of masters such as Michelangelo, in preparation for the finished work to follow. Again this raises the artistic expectations absurdly high. However, this article does in fact have some associations with both of these meanings of the word. The aim is to provide tools which make it easy for lecturers to produce drawings to explain ideas and for students to explore data and concepts. In that sense, students are introduced to sketches of the real thing. In addition, the ability to animate drawings easily is a particular aim and raising a smile in the classroom or lab is certainly welcome. The word 'cartoon' therefore seemed particularly appropriate.

There is of course, in statistics and in the mathematical sciences more widely, a long tradition of good use of graphics in teaching. Animation became feasible with the arrival of flexible computer graphics. In the 1980's, I had the pleasure of teaming up with Derek Robinson to investigate what the BBC microcomputer could do in this respect [1,2,3]. Although the technology seems very old-fashioned now, the accessibility to graphics which that machine provided makes those of us of a certain age and interest go misty-eyed with nostalgia. Ever since, it seems, we've been in search of the 'lost chord' of a system which allows graphical animations to be constructed easily, without an intricate knowledge of computing or the need to learn a new language.

The arrival and widespread adoption of R [4], as a standard part of the computing environment within the statistics community, has also changed the scene radically by providing easy access to a huge variety of modern statistical tools together with a very flexible computing environment. In addition, R has become increasingly good at providing tools for communicating with some other systems. One example is the `tcltk` package which is a standard part of R and which allows access to the *Tcl/Tk* system for creating buttons, sliders and other graphical controls. Spurred on by the availability of these tools, a group in the Department of Statistics at the University of Glasgow has been exploring how they might be packaged in a form which makes it particularly easy for lecturers who have some familiarity with R, but neither time nor inclination to learn other computing systems, to produce useful animated graphs for both teaching and research. Detailed descriptions of the system, called `rpanel`, are available elsewhere [5,6]. The aim of this article is to give some simple examples, particularly from the viewpoint of teaching.

The idea

In introducing elementary distributions, it is natural to plot them. Some simple R code for the Poisson distribution is given below:

```
plot(c(0, n), c(0, ylim),
     type = "n", xlab = "x",
     ylab = "Probability")
segments(0:n, rep(0, n),
         0:n, dpois(0:n, lambda))
title(paste("Poisson distribution",
           "with mean", lambda))
```

Here the parameters `n`, `ylim` and `lambda` control the ranges of the x- and y-axes and the mean of the distribution. To get a feel for the nature of the probability model this represents, it would be convenient to plot the distribution for a variety of different settings of `lambda`. Repeated execution of the code above is not hard but, in a lecture room setting in particular, the availability of a slider would be very useful.

The code below shows how this can be achieved. The same plotting code is placed inside a function, called here `poisson.draw`. That function takes a single argument, which is a list containing any information needed to draw the plot. The use of the `with` construction provides a useful way of avoiding the need to use a `panel$` prefix when

referring to the information in the list. An animation can then be created by two further function calls. One, to the `rpanel` function `rp.control`, simply sets up the list where information is to be found and launches a control panel. The other call, to `rp.slider`, adds a slider to the panel. When the slider is moved, the value of `lambda` is altered and the `poisson.draw` function called to redraw the graph. By this simple device, an animation is produced.

```
poisson.draw <- function(panel) {
  with(panel, {
    plot(c(0, n), c(0, ylim),
         type = "n", xlab = "x",
         ylab = "Probability")
    segments(0:n, rep(0, n),
            0:n, dpois(0:n, lambda))
    title(paste("Poisson distribution",
              "with mean", lambda))
  })
  panel
}

panel <- rp.control("Poisson distribution",
  n = 30, lambda = 3, ylim = 0.5)
rp.slider(panel, lambda, 1, 30,
  poisson.draw)
```

Notice that 'action' functions such as `poisson.draw` must always return the panel object. This is required by the

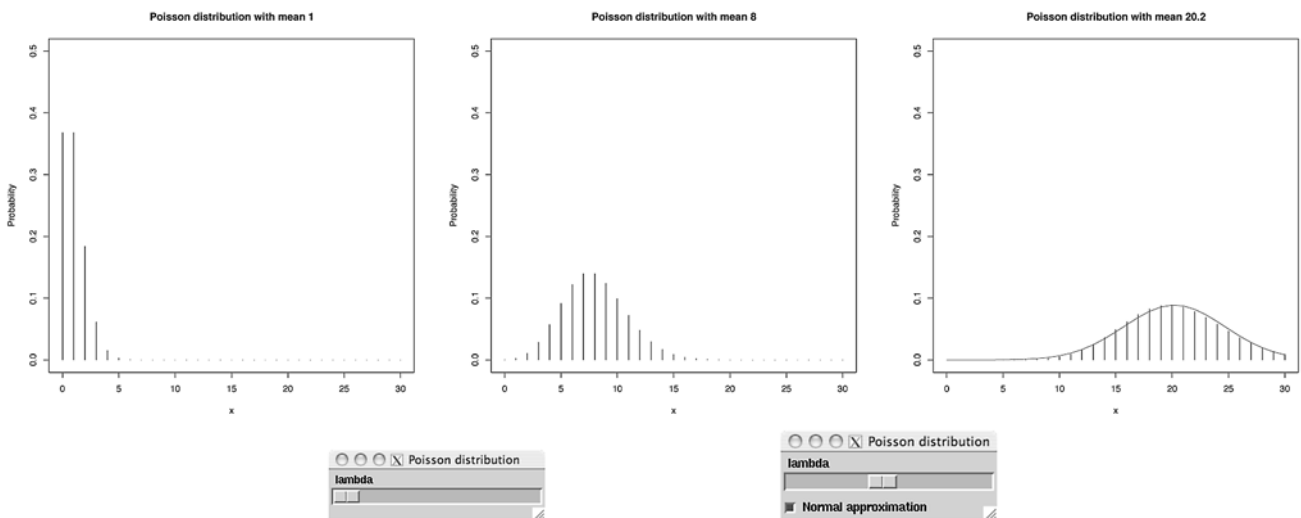


Fig 1 – Plots of a Poisson distribution, controlled by a slider

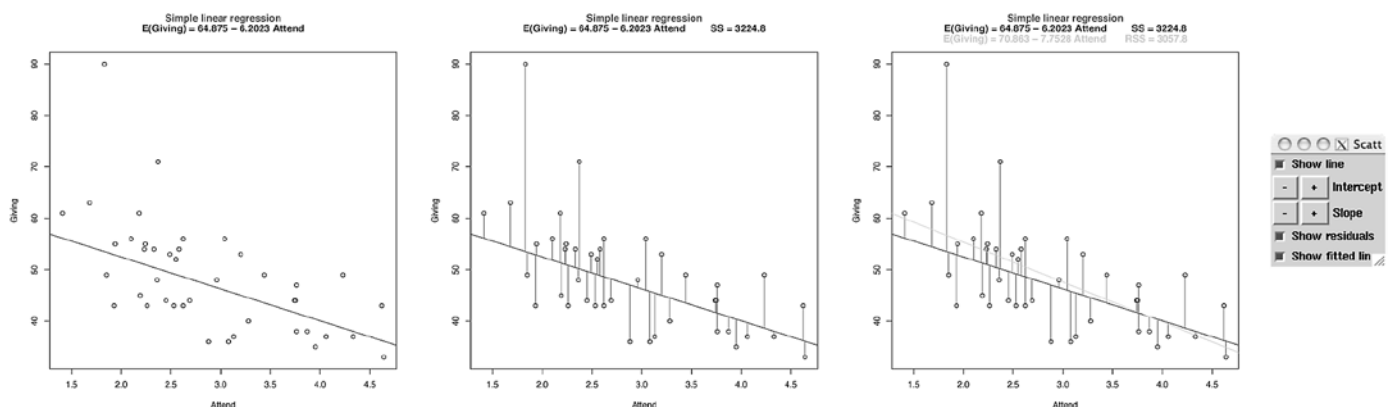


Fig 2 – An animated illustration of simple linear regression and the process of fitting this by least squares

behind-the-scenes communication mechanism which the `rpanel` package employs.

Fig 1 illustrates the slider, together with plots for different values of λ . The skewness for small λ , the approximately normal shape for larger ones, and the increasing spread caused by identity of the variance and the mean, are all clearly shown.

Since further ‘widgets’ can be added by calling additional functions, it is easy to enhance the panel. For example, a normal approximation could be added. The right hand plot and slider in Fig 1 show the effects of adding the following code to the action function:

```
if (normal.showing)
  lines(0:n, dnorm(0:n, lambda,
    sqrt(lambda)), col = "red")
```

and adding the following statement to the `rpanel` function calls.

```
rp.checkbox(panel, normal.showing,
  poisson.draw,
  title = "Normal approximation")
```

Clicking on the checkbox switches the setting of the logical variable `normal.showing` which controls whether the normal approximation is drawn by the action function.

Some other examples

A classic example of animation, used by many authors in the past, involves the control of the intercept and slope parameters of a simple linear regression. Fig 2 shows a panel with a checkbox which enables a regression line to be added to the plot, double-buttons to alter the values of the intercept and slope of the line, and further checkboxes to show the residuals and the least squares fitted line. In elementary teaching, this can be used to reinforce the meaning of the model parameters and the role of the sum-of-squares function in finding the least squares estimates.

A variety of other examples of animated plotting tools, to illustrate issues such as sampling variation, regression with two covariates, Box-Cox transformations, confidence intervals, density estimation and some more sophisticated topics, are available at:

<http://www.stats.gla.ac.uk/~adrian/rpanel> and in [5,6]. These two papers are also available on the web (references below).

Plotting log-likelihoods

The plotting of log-likelihood functions and surfaces offers a slightly more advanced topic where suitable graphics and animation may be helpful. Log-likelihoods involving one parameter can be plotted by standard methods. Animation can be of some value in displaying the fitted model as the parameter values are altered. However, with problems involving two parameters there is much more scope for interesting exploration and discussion. The `rgl` package [7]

provides access to very useful three-dimensional rendering tools within the *OpenGL* system, making three-dimensional displays relatively easy to construct. (The `rpanel` package contains a simple `rp.plot3d` function for three-dimensional scatter plots and the `rgl` package has the ability to add surfaces to a plot.)

A function `rp.loglik2` is available on the `rpanel` website. It is not reproduced here because quite a lot of detailed control has been implemented and this necessitates fairly lengthy code. However, the main point is that the panel control part of this code can be constructed again by simple calls to the relevant `rpanel` functions. It is the plotting code which becomes lengthier.

Fig 3 on the following page shows a log-likelihood surface for the mean and standard deviation of a normal distribution, using simulated standard normal data. The ability, provided by `rgl`, to rotate the plot with the mouse enhances the three-dimensional effect and allows the surface to be viewed from any orientation of interest. Two approaches to confidence regions can then be discussed. One, the Wilks approach, cuts the surface horizontally at a distance $\chi^2(2; 0.95)/2$ below the maximum, where $\chi^2(2; 0.95)$ denotes the 95th percentile of the χ^2_2 distribution. The other, the Wald approach, uses a quadratic approximation to the log-likelihood surface about its maximum. This quadratic surface is shown in the figure, with transparent colour on the original surface to enhance visibility. By viewing the surfaces from directly above, the confidence regions are displayed in a two-dimensional manner. (By a serendipitous feature of the `rgl` lighting scheme, the colour shading of the horizontal cutting plane simply masks the lower parts of the surfaces.) The relatively good agreement between the regions, and the enforced symmetry in the shape of the Wald region with respect to the standard deviation parameter, are both apparent. The control panel for this example is also shown at the foot of the figure.

The following code shows how these figures can be produced:

```
x <- rnorm(25, 0, 1)
loglik.fn <- function(par, data)
  sum(dnorm(data, par[1], par[2],
    log = TRUE))
rp.loglik2(c(-1, 1), c(0.5, 2),
  loglik.fn, x)
```

The function `rp.loglik2` requires as input arguments the plotting range of parameter 1, the plotting range of parameter 2, a function to define the log-likelihood and the data which is used by this function. Notice that the log-likelihood function is very conveniently defined computationally, in a one-line statement, rather than algebraically. (This very useful approach was inherited from Simon Wood, who was my predecessor in teaching a third year inference course in the University of Glasgow.)

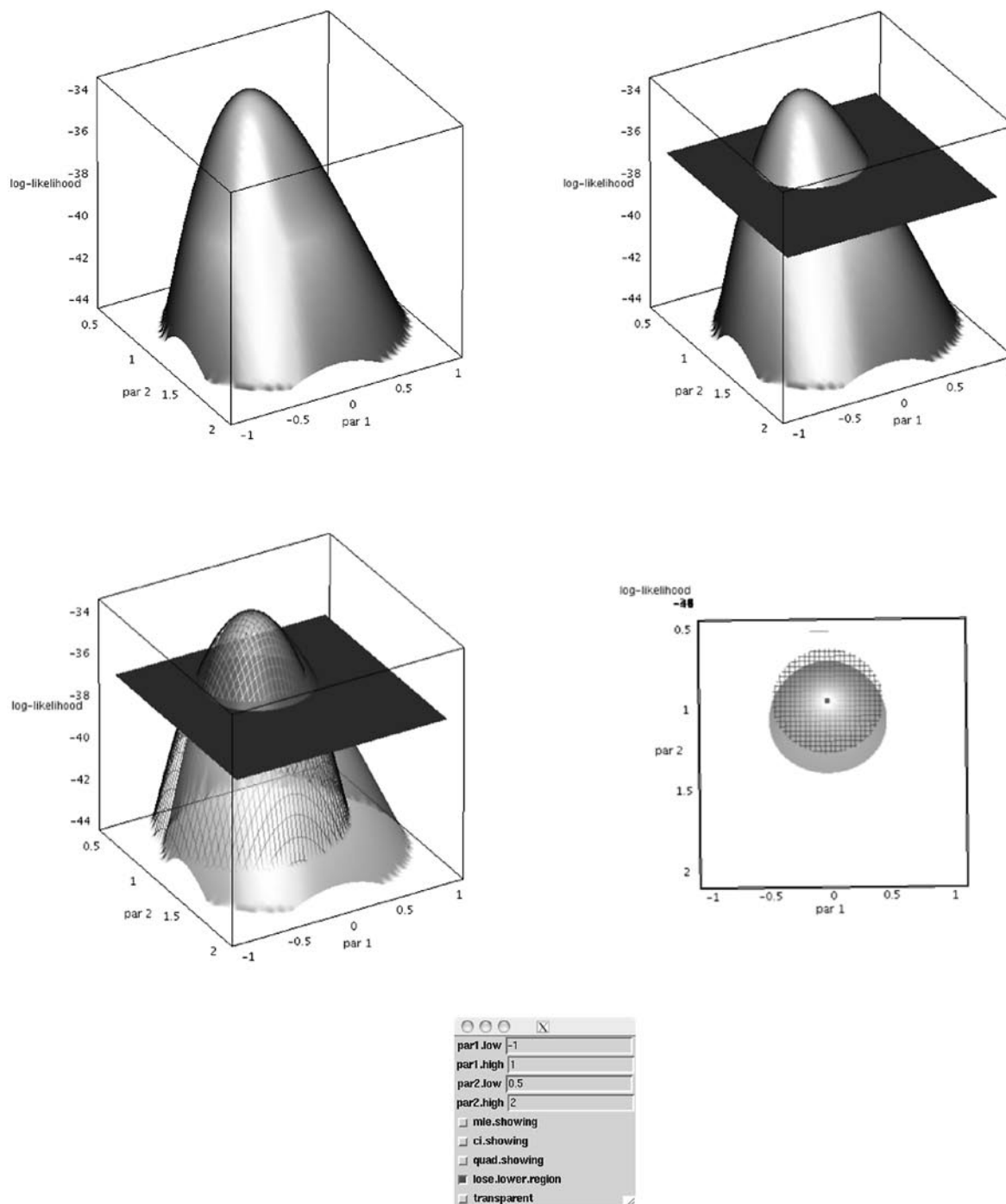


Fig 3 – Plots of a normal log likelihood surface

A more interesting example is provided by a sample of data on the failure times of air-conditioning equipment, given in [8]. Fig 4 shows the log-likelihood surfaces which arise when the Gamma distribution is parameterised in terms of a shape parameter together with either a rate or scale (reciprocal of rate) parameter. The reciprocal transformation has a marked effect on the shape of the log-likelihood surface and on the corresponding approximate confidence regions. There is therefore much that can be usefully discussed, using the pictures as a focus.

Conclusions

This article has outlined some simple ways in which animation can be added to R plots relatively easily, with potentially beneficial effects in the context of

teaching and learning. The aim is to provide simple and accessible tools for control panels, while the user has complete freedom over the construction of the graphs. As mentioned above, a variety of further examples are available on the `rpanel` web site. Suggestions and contributions for others are very welcome and should be sent to Adrian Bowman (adrian@stats.gla.ac.uk) or Ewan Crawford (ewan@stats.gla.ac.uk).

One feature of the `rpanel` package which has not been mentioned is the ability to interact in a simple way with images. This has been used to recreate a teaching module based on the sexing of herring gulls, which had previously been written with the aid of a sophisticated authoring tool. A discussion of this specific example is given in [6].

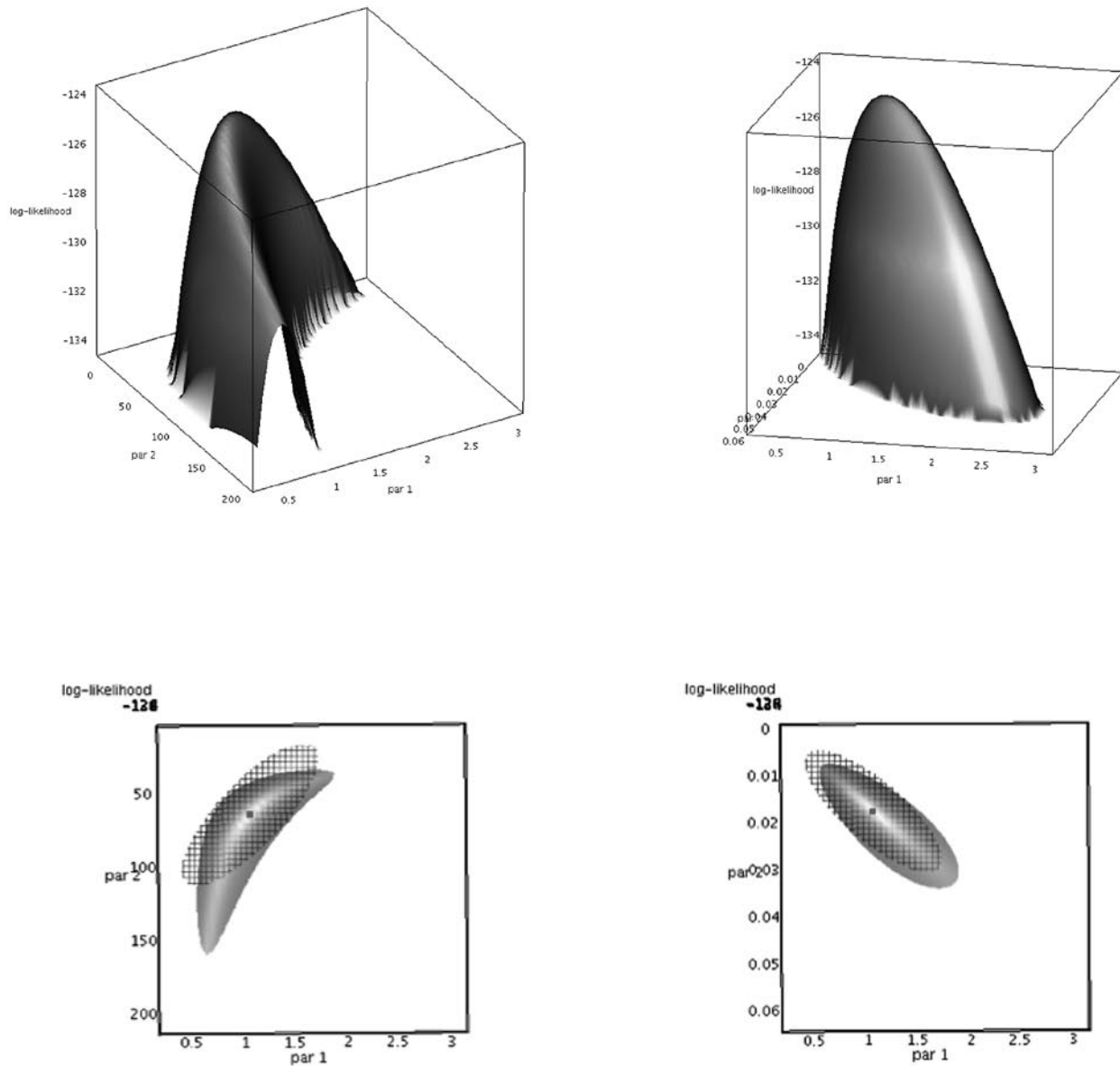


Fig 4 – Plots of a gamma log likelihood surface

We hope to develop `rpanel` further and major changes will be signalled in future editions of the MSOR Connections newsletter.

References

1. Robinson, D.R. and Bowman, A.W. (1986). *Introduction to Probability: a Computer Illustrated Text*. Adam Hilger, Bristol.
2. Bowman, A.W. and Robinson, D.R. (1987). *Introduction to Statistics: a Computer Illustrated Text*. Adam Hilger, Bristol.
3. Bowman, A.W. and Robinson, D.R. (1990). *Introduction to Regression and Analysis of Variance: a Computer Illustrated Text*. Adam Hilger, Bristol. 213pp.
4. R Development Core Team (2006). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3900051070. More info on the R Project for Statistical Computing is available via: <http://www.r-project.org> [Accessed 12 Oct 2007].
5. Bowman, A.W., Crawford, E. and Bowman, R.W. (2006). `rpanel`: making graphs move with `tk`. *R Newsletter*, Vol. 6/4, October 2006. Available via: http://cran.r-project.org/doc/Rnews/Rnews_2006-4.pdf [Accessed 12 Oct 2007].
6. Bowman, A.W., Crawford, E., Alexander, G. and Bowman, R.W. (2007). `rpanel`: simple interactive controls for R functions using the `tk` package. *Journal of Statistical Software*, 17, issue 9. Available via: <http://www.jstatsoft.org/v17/i09/> [Accessed 12 Oct 2007].
7. Adler, D. and Murdoch, D. (2006). `rgl`: 3D visualization device system (OpenGL). R package version 0.68. Available via: <http://www.r-project.org/> [Accessed 12 Oct 2007].
8. Cox, D.R. & Snell, E.J. (1980). *Applied Statistics: principles and examples*. Chapman & Hall/CRC, London.