Using R to Teach Econometrics

Jeff Racine*
Department of Economics
University of South Florida
Tampa, Florida, U.S.A., 33620

Rob Hyndman
Department of Econometrics & Business Statistics
Monash University
VIC 3800, Melbourne, Australia

October 30, 2001

Abstract

R, an open-source programming environment for data analysis and graphics, has in only a decade grown to become a de-facto standard for statistical analysis against which many popular commercial programs may be measured. The use of R for the teaching of econometric methods is appealing. It provides cutting-edge statistical methods which are, by R's open-source nature, available immediately. The software is stable, available at no cost, and exists for a number of platforms, including various flavors of Unix and Linux, Windows (9x/NT/2000), and the MacOS. Manuals are also available for download at no cost, and there is extensive on-line information for the novice user. This review focuses on using R for teaching econometrics. Since R is an extremely powerful environment, this review should also be of interest to researchers.

1 R: an overview

There is a wide variety of programs for conducting statistical analysis. Many are powerful commercial products, such as Gauss, Matlab, Minitab, SAS, Shazam, Stata, SPSS, S-PLUS, and TSP. Unfortunately, commercial software can be costly, sometimes prohibitively so, can involve relatively long development cycles, and rarely features experimental or cutting-edge statistical methods. Fortunately, there now exists an open source alternative to commercial statistical programs that is available for a variety of computing platforms including Windows, MacOS, MacOS X, FreeBSD, NetBSD, Linux, Irix, Solaris, OSF/1, AIX and HPUX. This software has been developed by a core team of leading statisticians and programmers. The software is called 'R' and is available under the GNU Public License[1]. To obtain the software, simply go to the site `http://www.r-project.org` and follow the download instructions.

The syntax of the R language is very similar to that of the S language which underlies S-PLUS. In fact, you can think of R as 'GNU S'. The main differences that users familiar with S will notice are (1) objects are not saved as separate files in a directory but are stored internally; and (2) the packages available in R are not the same as the libraries available in S-PLUS. However, for most purposes, programming in R is almost identical to using the command syntax of S-PLUS, and some

*The authors would like to express our gratitude to James MacKinnon and Douglas Bates for their encouragement and support. The usual caveat applies.

[1]Readers who are unfamiliar with open source software or the GNU project should visit `http://www.gnu.org`.

recent books attempt to cover both packages. Two of the best full-length books on using R (and S-PLUS) are Venables & Ripley (1999, 2000).

Cribari-Neto & Zarkos (1999) review a beta version of R. Their review focuses on language syntax, Monte Carlo experiments, and speed improvements relative to S-PLUS. They conclude by stating that R "...has the potential to become a useful tool ... for teaching in the econometrics community." Since we attempt to avoid repetition of material in their article, we urge readers to look at it. As of this writing, the current version of R is 1.3.1.

2 Using R for teaching

Unlike many commercial programs, which must first be purchased in order to evaluate them, you can evaluate R with minimal effort and with no financial outlay required. Why might a reader consider using R rather than popular commercial software? We briefly list a few other reasons.

Pros:

- Free. Students can have copies at home.
- Portable. Once students invest in learning this program, they can take it with them and install it again wherever they may end up working.
- Versatile. The software exists for more platforms than virtually any existing commercial program.
- General. A very large number of statistical/econometric tools are available, so the software could be used for many (maybe all) subjects.
- Cutting-edge. It includes the very latest methods.
- Programmable. It is easy for students to program new methods or develop modifications of existing methods.
- Matrix language. The R language handles vectors and matrices directly (as do Gauss, Matlab and Ox). This makes programming much simpler for students and reinforces the matrix notation used in class.
- Object-oriented language. Students may take a little time to adjust to the object-oriented way of thinking, but it simplifies things greatly. For example, the `plot` function and the `summary` function can be used on all types of data and fitted models.
- Great graphics.
- Relatively fast.

Cons:

- Command-driven. Although the command line, which is similar to that of the `bash` shell, is extremely powerful and easy to use, some students accustomed to drag-and-drop menu programs may find R awkward to use.

- Missing functions. Some desirable functions have not yet been written (e.g., there is no function for testing Granger causality). Of course, it is easy to add such functions yourself.

- Inconsistent syntax. Command syntax is not always consistent between packages which do similar things. For example, the function `locpoly` from the package `KernSmooth` is designed to fit a smooth nonparametric regression curve using local polynomials. The function `loess` from the package `modreg` does the same thing, but it is more robust to outliers. However, the syntax for specifying the dependent and independent variables is different. Students can find this confusing.

There is an important difference in philosophy between R and most other statistical packages. With most packages, a statistical analysis will lead to a large amount of output containing information about the estimation, diagnostic tests, etc. In R, a statistical analysis is normally done as a series of steps, with intermediate results being stored in objects. There is usually minimal output at each step, but the objects obtained at each step can be interrogated by R functions to obtain the information required.

At first, this may seem like a disadvantage, as students will not always be able to find the information they want from an analysis. However, it does teach students the interactive nature of data analysis (rather than seeing it as the application of "recipes"), and it avoids the confusion that students often experience in being confronted with pages of output that they cannot interpret.

3 Packages

`R` functionality is based around the concept of "packages". A package is a collection of functions to carry out certain tasks (rather like Gauss modules or Matlab toolboxes). For example, the `nls` package does nonlinear regression, the `ts` package contains a variety of time-series functions, and so on. The `base` packages are automatically available with a default installation. Contributed packages, on the other hand, need to be installed individually. One can obtain these packages by following the download link on the `R` home page[2].

By its very nature, `R` is a dynamic, evolving computing environment, and packages are continuing to be written at a rapid rate. Very often, the capabilities of two or more packages overlap. For example, both the `tseries` and `ts` packages provide procedures for fitting ARMA models.

Some of the packages that may be of interest to econometricians are listed below. Some of these are installed automatically when you install `R`, and the others can be downloaded and installed individually from the `R` web site.

Time series

- `ts` Time-series functions, including ARIMA modelling, regression with ARMA errors, Box-Pierce and Ljung-Box tests, ACF, PACF, CCF, spectral density estimation, Phillips-Perron unit root test, STL seasonal decomposition, and kernel smoothing.
- `tseries` More time series functions, including ARMA modelling, auto mutual information function, BDS test, GARCH modelling, Jarque-Bera test, ADF test for a unit

[2]The Windows version has a menu which allows the user to select available packages from a menu. The Unix versions require that one download the appropriate gzipped tar file, log in as root, and type "`R CMD INSTALL` *`packagename.tar.gz`*".

root, Phillips-Ouliaris cointegration test, KPSS test for stationarity, runs tests, and Teräsvirta's and White's neural network tests for nonlinearity.

- `dse` Dynamic Systems Estimation, including multivariate state space and ARMA and VAR models.
- `fracdiff` Fractionally differenced ARIMA(p, d, q) models.
- `strucchange` Structural change tests based on empirical fluctuation processes.

Regression

- `base`. The base package (automatically loaded) contains functions for inference using linear models and extensive regression diagnostics and plots.
- `lmtest` Breusch-Pagan test, CUSUM test, Durbin-Watson test, Goldfeld-Quandt test, Harvey-Collier test, Harrison-McCabe test, Ramsey's RESET test.
- `modreg` Modern Regression: Smoothing and Local Methods. Includes kernel regression, local polynomial regression, loess, projection pursuit regression, smoothing splines, and Friedman's "super smoother".
- `quantreg` Quantile regression as described in Koenker & Basset (1978).
- `nlme` Linear and nonlinear mixed effects models based on Pinheiro & Bates (2000).
- `nls` Nonlinear regression. Facilities are included for some specific models including biexponential model, first-order compartment model, four-parameter logistic model, Gompertz growth model, logistic model, Michaelis-Menten model, Weibull growth curve model.

Inference

- `boot` Bootstrap functions. Both parametric and nonparametric resampling are available.
- `coda` Output analysis and diagnostics for MCMC. A menu interface is provided. Many functions for diagnostics and plots are included.

Panel data

- `panel` A function to compute the maximum likelihood estimates of the transition parameters from panel data. The algorithm is discussed in Kalbfleisch & Lawless (1985) and Gentleman (1994).

Exploratory data analysis

- `base` The base package contains many graphics functions which are useful for EDA, including scatterplots, scatterplot matrices, boxplots, histograms, stem-and-leaf plots, qqplots, sunflower plots and so on.
- `eda` Exploratory Data Analysis based on Tukey (1977). Includes robust line fitting, median polishing, and median smoothing.

Distributions and tests

- `ctest` Classical Tests. All the usual tests including various t-tests, F-tests, rank-based tests, power calculations, tests for homogeneity of variances, χ^2 tests, tests of normality, and many more.

Computation

- `Matrix`. Provides a number of additional facilities for matrix arithmetic, including Schur decomposition, LU decomposition, Hilbert matrices, and various tests for symmetry and orthogonality.

Miscellaneous

- `mva` Classical Multivariate Analysis including principal component analysis, factor analysis, canonical correlations, multidimensional scaling and hierarchical clustering.
- `xtable` This package can be used to create LaTeX formatted tables.

4 Web sites

There is a growing number of web sites dedicated to providing information about `R` and code to be used with `R`. Some of the main sites are listed here.

- `http://www.R-project.org/`

  This is the `R` home page from which you can download the program itself and many `R` packages. There are also manuals, other links, and facilities for joining various `R` mailing lists.

- `http://maths.newcastle.edu.au/~rking/R/`

  Three `R` mailing lists are archived here.

- `http://lark.cc.ukans.edu/~pauljohn/R/statsRus.html`

  This site provides a large and excellent collection of `R` tips.

- `http://www.vanderbilt.edu/quantmetheval/r.htm`

  This is a compilation of `R` resources on the web.

In addition, some of the web sites which provide S-PLUS functions will be of interest to `R` users. Most code written for S-PLUS should work in `R` (although sometimes minor modification is required). Two sites of interest are given here.

- `http://lib.stat.cmu.edu/S/`

  This is the URL for the large library of S-PLUS code on statlib.

- `http://www.econ.uiuc.edu/~econ472/routines.html`

  This page is for an Applied Econometrics class at the University of Illinois at Urbana-Champaign. It contains S-PLUS functions for SUR, 2SLS, 3SLS, Breusch-Pagan test, ADF test, Granger causality test, Johansen's cointegration test, and some additional functions for panel estimation. They may require minor modification for use in `R`.

5 Getting started

Having installed and run R, you will find yourself at the > prompt. To quit the program, simply type `q()`. To get help, you can either enter a command proceeded by a question mark, as in `?help`, or type `help.start()` at the > prompt. The latter will spawn your web browser (it reads files from your hard drive, so you do not have to be connected to the Internet to use this feature).

You can enter commands interactively at the R prompt, or you can create a text file containing the commands and execute all commands in the file from the R prompt by typing `source ("commands.txt")`, where `commands.txt` is the text file containing your commands.

When you quit by entering the `q()` command, you will be asked whether or not you wish to save the current session. If you enter `Y`, then the next time you run R *in the same directory* it will load all of the objects created in the previous session. If you do so, typing the command `ls()` will list all of the objects. For this reason, it is wise to use different directories for different projects. To remove objects that have been loaded, you can use the command `rm(`*`objectname`*`)`.

5.1 Reading ASCII data

The two most common functions for reading data from disk files are the functions `read.table()` and `scan()`. The most portable format is simply ASCII column format with observations entered on separate rows and variables in separate columns. Variable names may be included in the first row. For instance, if you type `data <- read.table(file='`*`datafilename`*`', header=T)`, this will read the data from the file *datafilename* and use the first row as the variable names. If you do not have variable names in the first row, simply omit the `header=T` statement. Having read the data into a `data table`, you then "attach" the data object with the command `attach(`*`data`*`)`. This allows individual variables within the data set to be accessed directly.

The `scan()` command is used to read a single vector into R. For example, to read the data in *datafilename* into a vector called "datavec", issue the command `datavec <- scan(file='`*`datafilename`*`')`.

5.2 Importing other data formats

The `foreign` package allows you to read data created by different popular programs. To load it, simply type `library(foreign)` from within R. Supported formats include

- `read.dta` – Read Stata binary files
- `read.mtp` – Read a Minitab Portable Worksheet
- `read.spss` – Read an SPSS data file
- `read.xport` – Read a SAS XPORT format library
- `read.S` – Read an S-PLUS binary file

6 Examples

The best way to see how R works is to look at some code and examples for a number of popular econometric and statistical methods. That is what we do in this section. We will use data sets included with both the base and contributed packages.

6.1 Linear regression models and the lm() function

We start with the estimation of a linear regression model. By way of example, we use a data set found in the base package which can be loaded using the command `data(cars)`. For this example we regress braking distance (`dist`) on the car's velocity (`speed`).

We can estimate the linear model using the `lm()` function, and we can assign the result to an object of our choosing, say `carfit`. To regress `dist` on a constant and `speed`, we enter the command `carfit <- lm(dist ~ speed, cars)`. To view a summary of the fitted model we type `summary(carfit)`. Here is this sequence of commands and the program output:

```
> data(cars)
> carfit <- lm(dist ~ speed, data=cars)
> summary(carfit)
Residuals:
        Min      1Q  Median      3Q     Max
    -29.069  -9.525  -2.272   9.215  43.201
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -17.5791     6.7584  -2.601   0.0123 *
speed         3.9324     0.4155   9.464 1.49e-12 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
Residual standard error: 15.38 on 48 degrees of freedom
Multiple R-Squared: 0.6511, Adjusted R-squared: 0.6438
F-statistic: 89.57 on 1 and 48 DF,  p-value: 1.49e-12
```

The object that we named `carfit` now contains information that can be accessed. For instance, the residuals are retrieved from the object `carfit` using the accessor function `resid()` as in `resid(carfit)`, the coefficients using `coef(carfit)`, and the fitted values `fitted(carfit)`. We can see all stored values in such objects by typing `names(carfit)` and `names(summary(carfit))`. To print values stored in the former, we could use the accessor function or type `carfit$objectname`, while for those in the latter we could type `summary(carfit)$objectname`. In general, to display an object, we simply type its name.

To visualize the fitted line, we can use the following commands.

```
> plot(speed ~ dist, data=cars)
> abline(carfit)
```

The first command creates a graph with `speed` on the X axis and `dist` on the Y axis and plots the actual data. The second one adds the fitted line to the graph. If we did not want to include a constant, we might call this model '`carfitnocst`' and instead type `carfitnocst <- lm(dist ~ speed - 1)`. The effect of the '-1' term is to remove the constant from the model's formula.

To compute the covariance matrix $s^2(X'X)^{-1}$ for the model's parameters, we can retrieve `sigma` (s) and `cov.unscaled` ($(X'X)^{-1}$) and multiply the square of the former times the latter. For example,

```
> vcov <- summary(carfit)$sigma^2 * summary(carfit)$cov.unscaled
> vcov
             (Intercept)      speed
(Intercept)   45.676514 -2.6588234
speed         -2.658823  0.1726509
```

We can retrieve the $(1, 2)$ element of the covariance matrix by typing `vcov[1,2]`.

6.1.1 Transformation of variables in regression models

It is possible to transform one or more variables from within the `lm()` function. For example, if you wished to include `speed` and its square, you would need to use the `I()` operator to enclose the transformed variable `speed^2` as in `I(speed^2)`. To include this transformation, you might call this model '`carfitquad`' and would enter `carfitquad <- lm(dist ~ speed + I(speed^2))`. Logarithmic and trigonometric transformations are obtained in exactly the same manner.

6.1.2 Interaction of variables in regression models

In order to include interaction between *two variables*, you can use the colon operator. In order to include interaction among a group of variables, you can enclose the variables in parentheses and use the `^j` command, where j is an integer > 1 that dictates the order of the interactions.

6.1.3 Common F tests in regression models

To perform F tests, you can use the `anova()` function. For instance, if you wished to compute an F test for the joint significance of a subset of regressors, you can estimate the restricted model and the unrestricted model as outlined above, then compare the two using the command `anova(nameres, nameunres)` where '`nameres`' and '`nameunres`' are the names we have given the respective models. Consider the following example.

```
> carfitunres <- lm(dist ~ speed + I(speed^2))
> carfitres <- lm(dist ~ speed)
> anova(carfitres,carfitunres)
Analysis of Variance Table
Model 1: dist ~ speed
Model 2: dist ~ speed + I(speed^2)
  Res.Df     RSS Df Sum of Sq     F Pr(>F)
1     48 11353.5
2     47 10824.7  1     528.8 2.296 0.1364
```

6.1.4 Tests for serial correlation and heteroskedasticity in regression models

To perform these tests, you will need to install the contributed package `lmtest` and load this using the command `library(lmtest)`. To conduct the Breusch-Pagan and Durbin-Watson tests for an estimated model, you would proceed as follows.

```
> library(lmtest)
> bptest(carfit)
data:  form1 BP = 4.3257, df = 2, p-value = 0.115
> dwtest(carfit)
data: DW = 1.6762, p-value = NA
```

The accompanying manual states that "the null value depends on X, so the p-value is hardly tractable, so p-value is NA."

6.1.5 Prediction in regression models

We shall use the `predict()` function. Having estimated a model and called it, say, '`carfit`', we then use this model as an argument of `predict()`. If you use the option `interval="confidence"`, then this generates confidence intervals for mean predictions, while if you use the command `interval="prediction"`, this generates prediction intervals for individual predictions which are stored in the object we have called '`carpred`'. For this example, we print out the first prediction and its lower and upper prediction interval.

```
> carpred <- predict(carfit,se.fit=T,interval="prediction")
> carpred$fit[1,]
       fit       lwr      upr
1 -1.849460 -34.49984 30.80092
```

When desired, you can specify both an estimation data set and an evaluation data set using the commands `data=` and `newdata=` respectively.

6.2 Binary choice models and the `glm()` function

We can use the 'generalized linear model' function '`glm()`' to estimate binary choice models using the `link = probit` or `link = logit` options, while count models are estimated using the `family = poisson()` option. By way of example, we simulate some data from a latent variable specification and use this to estimate a probit model.

```
> x <- rnorm(100)
> y <- 1 + x + rnorm(100)
> y <- as.integer(y<0)
> probitfit <- glm(y ~ x, family = binomial(link = probit))
> summary(probitfit)
Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)   1.1031     0.2128   5.184 2.17e-07 ***
x             1.3707     0.2740   5.003 5.65e-07 ***
(Dispersion parameter for binomial family taken to be 1)
    Null deviance: 112.467  on 99  degrees of freedom
Residual deviance:  62.249  on 98  degrees of freedom
AIC: 66.249
Number of Fisher Scoring iterations: 5
```

Note that the above listing has been shortened slightly, in part by removing blank lines as have many of those that follow.

6.3 Selectivity models

Suppose that we wish to compute the inverse Mills ratio based upon a probit model, which might then be used to correct for selectivity bias. Using the same data and estimated probit model as above, we can compute $f(x_i'\beta)/F(x_i'\beta)$ in the following manner:

```
> index <- probitfit$coef[1] + probitfit$coef[2]*x
> invmills <- dnorm(index)/pnorm(index)
```

6.4 Least squares estimation of nonlinear models and the `nls()` function

To estimate nonlinear regression models, we first load the `nls` library, which is part of the base package, using `library(nls)` and then use the `nls()` function. We write the formula for the model and specify starting values for the model's parameters, which can be arguments in the `nls()` function, as the following example demonstrates.

```
> library(nls)
> carfitnls <- nls(dist ~ a + speed^b, start=list(a=1,b=1))
> summary(carfitnls)
Parameters:
  Estimate Std. Error t value Pr(>|t|)
a -3.35701    4.44957  -0.754    0.454
b  1.39114    0.02971  46.827   <2e-16 ***
Residual standard error: 15.11 on 48 degrees of freedom
```

6.5 Time series

Time series functions are provided by the `ts` and `tseries` packages.

6.5.1 Example: ARIMA model

As an example, we shall consider the monthly totals of car drivers in Great Britain killed or seriously injured between January 1969 to December 1984. Mandatory wearing of seatbelts was introduced in February 1983. We fit the seatbelt intervention using a dummy variable with an ARIMA error structure. We first plot the data, and then its seasonal difference.

```
> library(ts)
> data(UKDriverDeaths)
> plot(UKDriverDeaths)
> diff.1 <- diff(UKDriverDeaths,12)
> plot(diff.1)
```

It is not clear whether the seasonally differenced series is stationary, so we use the command `adf.test(diff.1)` to do an augmented Dickey-Fuller test. The test is significant, so we do no further differencing. The ACF and PACF of the differenced series can plotted as follows:

```
> par(mfrow=c(1,2))
> acf(diff.1,lag.max=40)
> pacf(diff.1,lag.max=40)
```

The `par` command sets up the graphing window so that both plots appear side-by-side.

An intervention model with an ARIMA(1,0,1)(0,1,1)$_{12}$ error is fitted as follows.

```
> seatbelt <- as.matrix(c(rep(0,169),rep(1,23)))
> fit <- arima0(UKDriverDeaths,order=c(1,0,1),seasonal=list(order=c(0,1,1),period=12),
    xreg=seatbelt)
> print(fit)
Coefficients:
      ar1        ma1       sma1      xreg1
   0.9360    -0.6042    -0.8898  -346.9095
Approx standard errors:
    ar1      ma1     sma1    xreg1
 0.0550   0.1328   0.0117  85.9939
sigma^2 estimated as 16767:  log likelihood = -1139.96,  aic = 2287.93
```

The AIC is stored as `fit$aic`. Some diagnostics can be obtained using `arima0.diag(fit)`. In fact, this model has the smallest AIC of all models with one seasonal difference. Clearly, the seatbelt variable is significant ($t = -346.9/86.0 = -4.03$) and accounts for an average decrease of 346.9 deaths or serious injuries per month.

Predictions can be computed as follows:

```
> fcast <- predict(fit,n.ahead=12,newxreg=as.matrix(rep(1,12)))
```

The object `fcast` now contains point forecasts (`fcast$pred`) and forecast standard deviations (`fcast$se`). Point forecasts and forecast intervals can be plotted using `fcast`:

```
> plot(UKDriverDeaths,xlim=c(1969,1986),ylim=c(900,2600))
> lines(fcast$pred,col=2)
> lines(fcast$pred + 2*fcast$se,col=3)
> lines(fcast$pred - 2*fcast$se,col=3)
```

6.5.2 Example: GARCH model

Facilities for fitting ARCH and GARCH models are provided in the `tseries` package. Here we fit a GARCH(1,1) model to the log returns of the daily closing prices of the UK FTSE for 1991–1998.

```
> data(EuStockMarkets)
> ftse <- diff(log(EuStockMarkets))[,"FTSE"]
> ftse.garch <- garch(ftse, order=c(1,1))
> summary(ftse.garch)
Coefficient(s):
    Estimate  Std. Error  t value Pr(>|t|)
a0 8.722e-07   3.083e-07    2.829  0.00467 **
a1 4.532e-02   6.791e-03    6.674 2.49e-11 ***
b1 9.419e-01   1.018e-02   92.478  < 2e-16 ***
Diagnostic Tests:
        Jarque Bera Test
data:  Residuals
X-squared = 215.3627, df = 2, p-value = < 2.2e-16
```

```
        Box-Ljung test
data:  Squared.Residuals
X-squared = 0.1001, df = 1, p-value = 0.7517
```

The ARCH effects have been filtered, but conditional normality has been violated. A variety of diagnostic plots for the fitted model can be obtained using `plot(ftse.garch)`.

6.6 Maximum likelihood estimation

Maximum likelihood estimation is easily accomplished using the nonlinear optimization function `optim()`. For example, consider the loglikelihood of a logistic regression with one covariate:

$$l(\beta_0, \beta_1; \mathbf{y}) = \sum_{i=1}^{n} [y_i \log p_i + (1 - y_i) \log(1 - p_i)]$$

where $p_i = \ell(\beta_0 + \beta_1 x)$ and $\ell(u) = e^u/(1 + e^u)$ is the logistic function. We can write a function in `R` to compute the negative loglikelihood:

```
logitreg <- function(beta,x,y)
{   eta <- exp(beta[1] + beta[2]*x)
    p <- eta/(1+eta)
    return(-sum(ifelse(y,log(p),log(1-p)))) }
```

To demonstrate this function, consider the variable identifying which towns within Boston have a crime rate per capita in the top 20% of towns. We shall seek to model this variable as a function of the wealth of the locality, where wealth is measured using the median value of owner-occupied homes in the district. The Boston data are part of the MASS library, so we first need to load that library.

```
> library(MASS)
> data(Boston)
> crim <- as.integer(Boston$crim > quantile(Boston$crim,0.8))
> med.value <- Boston$medv
> mle <- optim(c(0,0),logitreg,x=med.value,y=crim,hessian=T)
```

The object `mle` contains information about the estimated coefficients. For example, `med$par` gives the estimates and `mle$hessian` contains the Hessian matrix. Therefore, to compute confidence intervals, we use

```
> se <- sqrt(diag(solve(mle$hessian)))
> cbind(lower=mle$par - 1.96*se, upper=mle$par + 1.96*se)
          lower      upper
[1,]  2.1449125  4.0137381
[2,] -0.2896639 -0.1842742
```

It is often useful for students to visualize the likelihood surface near the maximum. This can be done as follows.

```
> a _ seq(-10,15,l=50)
> b _ seq(-1,0.2,l=50)
> L <- matrix(NA,50,50)
> for(i in 1:50)
>     for(j in 1:50)
>         L[i,j] <- -logitreg(c(a[i],b[j]), med.value, crim)
> par(mfrow=c(1,2))
> contour(a,b,L)
> persp(a,b,L)
```

The latter two commands give a contour plot and a wire-frame perspective plot, respectively. These show clearly that the likelihood has a narrow ridge near the optimum, which is why the confidence intervals of the two parameters have such different widths.

Of course, in the case of logistic regression, there is a built-in function (`glm`) which estimates the parameters and provides a variety of other information for inference and diagnostics. However, it is often useful for teaching purposes for students to see how the solutions can be calculated using methods that are more generally applicable.

6.7 Programming estimators

The matrix capabilities in R's base installation are sufficient for programming many estimators. Common functions needed for matrix manipulation are the `solve()` function needed for matrix inversion, the `t()` function needed for matrix transposition, and the %*% function needed for matrix multiplication.

As an example, we read in a hypothetical ASCII data set, where Y is in column 1 and X in the remaining columns, attach a column of 1s to X, and then compute the unrestricted OLS estimator $\hat{\beta} = (X'X)^{-1}X'Y$ and the F ratio based upon the restriction matrix R and equality vector r for the restrictions $\beta_2 = \beta_3 = 0$. For comparison purposes, we also use the `lm()` command (linear model) and the `anova()` command to conduct an F test.

```
data <- read.table("data.dat",header=F)
attach(data)
Y <- as.matrix(data[,1])
X <- as.matrix(cbind(1, data[,2:ncol(data)]))
XTX <- t(X) %*% X
XTXINV <- solve(XTX)
XTY <- t(X) %*% Y
betahat <- XTXINV %*% XTY
Xbetahat <- X %*% betahat
sigsq <- as.vector(t(Y - Xbetahat) %*% (Y - Xbetahat) / (nrow(X) - ncol(X)))
VCOV <- sigsq * XTXINV
J <- 2
R <- matrix(0:0, J, ncol(X))
r <- matrix(0:0,J,1)
R[1,2] <- 1
R[2,3] <- 1
Rbetahat <- R %*% betahat
F <- t(Rbetahat - r) %*% solve(R %*% XTXINV %*% t(R)) %*% (Rbetahat - r) / (J * sigsq)
# Compare with the lm() command (linear regression model)
```

```
fitlm <- lm(Y ~ X[,2:ncol(X)])
fitreslm <- lm(Y ~ X[,4:ncol(X)])
anova(fitlm, fitreslm)
```

7 Conclusion

R is a powerful and well-written open-source statistical software package. The examples provided in this review merely scratch the surface of its extensive capabilities. We encourage readers to browse the extensive documentation and to visit the web sites listed in Section 4 to see examples of what R can do. All code and output files used in this review can be obtained from the JAE Data Archive (www.econ.queensu.ca/jae/).

References

Cribari-Neto, F. & Zarkos, S. G. (1999), 'R: yet another econometric programming environment', *Journal of Applied Econometrics* **14**(3), 319–329.

Gentleman, R. (1994), The use of covariate information in multi-state Markov models, Technical Report 13, Department of Statistics, University of Auckland.

Kalbfleisch & Lawless (1985), 'The analysis of panel data under a Markov assumption', *Journal of the American Statistical Association* **80**, 863–871.

Koenker & Basset (1978), 'Regression quantiles', *Econometrica* **46**, 33–50.

Pinheiro, J. & Bates, D. (2000), *Mixed-effects models in S and S-PLUS*, Springer, New York.

Tukey, J. (1977), *Exploratory data analysis*, Addison-Wesley, Reading.

Venables, W. & Ripley, B. (1999), *Modern applied statistics with S-Plus*, 3rd edn, Springer-Verlag, New York.

Venables, W. & Ripley, B. (2000), *S programming*, Springer-Verlag, New York.