

R for Political Methodologists

Simon Jackman
Stanford University

Nine years ago I contributed an essay to *The Political Methodologist* comparing GAUSS and Spplus (Jackman, 1994). Spplus was the new kid on the block at that point; in the early 1990s, I could literally count on one hand the number of political scientists using Spplus.

What a difference a decade makes. Spplus made serious inroads into quantitative social science (and other fields) through the 1990s. And now its free competitor, R, is widely used in political science methods classes, and is now the package-of-choice in the advanced classes at the ICPSR Quantitative Methods Summer School (see the accompanying article in this issue of TPM). To clarify: Spplus is a commercial product, a superset of the S language for data analysis developed at Bell Labs. R is largely a “GNU S”, developed by some of the same people who developed S, plus a large group of public-spirited statisticians and programmers (many of whom had contributed libraries to Spplus). Basically, the goal of the R project was (and remains) to take the S language to the masses, using many features of S as the foundation of an open-source and freely-available statistics package.

In this short essay I give a brief overview of R as a tool for advanced researchers and methodologists. As I did in 1994, let me stress at the outset that comparing software for the TPM readership is asking for trouble: it is curious how zealous or protective some of us get about the tools of our trade. But the trade and the tools evolve, and getting emotionally invested in a piece of software is just not worth it (and frankly, reflects something more serious than a methodological problem, but I digress). I’ve used lots of different programs thus far, and imagine that I’ll keep doing so over the years ahead. And at any given time point, it’s extremely rare that I find one package does it all. This is especially true for methodologists (we’re more or less expected to know our way around lots of types of data and models, and to keep current with new approaches), and so while R does many things for me, it doesn’t do quite everything: every so often I’ll use SAS for getting a massive data set in shape for analysis (although I’ve been able to do most of my data management in R in recent years); when I want to implement a Markov chain Monte Carlo approach (aka “Bayesian simulation”) I use WinBUGS, or, for my work on roll call data with Josh Clinton and Doug Rivers, I use my own C program, ideal (see <http://jackman.stanford.edu/ideal>).

At the same time that Spplus and R have been making headway into the social sciences, so too has STATA. STATA has the strength of being fairly easy to use, with many models and options from econometrics implemented and added with successive releases. For instance, for panel data, STATA sports an impressive array of models, options and post-estimation

commands (the `xt...` family of commands). In general, the kinds of things we encounter in intermediate to advanced econometrics texts are more likely to have been implemented in STATA than in R: e.g., a Cochrane-Orcutt or Prais-Winsten correction for auto-correlated residuals in a linear regression, Durbin's h statistic, Newey-West standard errors, Heckman's two-stage selectivity estimator. But frankly, it is surprising is how much econometrics is in R across various user-contributed libraries, given how little econometrics there was in Splus, at least in 1994: for instance, in the `strucchange` library one can find Huber-White heteroskedasticity-consistent standard errors, recursive residuals and Chow tests; in the `lmtest` library there are many tests of iid disturbances and linear functional form from the econometric literature (e.g., Breush-Pagan and Goldfeld-Quandt tests of heteroskedasticity, and the Durbin-Watson and Breush-Godfrey tests for autoregressive disturbances); in the `tseries` library there are ARCH and GARCH models, and numerous tests of stationarity, normality, and non-linearity; tests of general linear hypotheses are supported in the aptly named `gregmisc` library; the `systemfit` library implements systems of equations methods (SUR, two and three stage least squares; see also the `sem` library); the Kalman filter and ARIMA modeling can found in the `ts` library, along with many other smoothers and filters for time series. Since many political scientists learn data analysis via econometrics, and they will be pleased to know R supports many of the models and specification tests from that literature.

Since R is modeled on S, it shares many of the strong points of Splus that I wrote about nine years ago: some object-orientated design features, a strong emphasis on graphics and visualizing data, and a steady flow of innovation (both computational and statistical) from the applied statistics community. For example, consider the following R code fragment, with comments:

```
plot(y ~ x)           ## scatterplot
identify(x,y)         ## interact with scatterplot, click to identify points
reg1 <- lm(y ~ x)     ## reg1 is a linear model object
abline(reg1)          ## overlay regression line on scatterplot
plot(reg1)            ## plot knows what to do with reg1 (diagnostic plots)
```

The `plot()` command is generic, and sees that it has been passed (1) a formula, in the first call to `plot()`, and so produces a scatterplot; (2) a fitted regression in the second call, or more precisely, an object of class `lm`, and so hands off to `plot.lm()` which produces a series of diagnostic plots (partial residual plots, residuals against fitted, and influence statistics). Many other classes of R objects have a `plot` method, including data frames and matrices (all pairwise scatterplots), discrete or character variables (barplots). My current

favorite is from the `MCMCpack` library contributed by (political science’s own) Andrew Martin and Kevin Quinn: when passed an object of class `mcmc`, `plot` will produce posterior density plots and diagnostic traces for parameters estimated via Markov chain Monte Carlo. Another useful generic command is `summary()`. In short, the goal of this object-oriented approach to program design is to reduce the workload for the user: one command does lots of different things, with the software “smart enough” to figure out what the command means in different contexts.

Next to its price, one of the key reasons to use R over `Splus` is the ease of writing functions. R has much more permissive scoping rules than `Splus` (*lexical* scoping versus *static* scoping in `Splus`, meaning that one need not worry about whether variables used by a function have been declared “local” or “global” or whatever). This amplifies one of the great strengths of packages like `Splus` and R: *user-extensibility* or “writing your own programs”. Many quantitatively-inclined political scientists will never have to extend the functionality of their statistical software, but that is probably less so for readers of *TPM*. Indeed, if your notion of data analysis runs to more than estimating coefficients and *t*-statistics, or if a write-up is more than cutting-and-pasting tables of estimated parameters, noting which have more “stars” (asterisks) than others, then from time-to-time you’ll find yourself programming, if only a little. Big substantive payoffs often come from computing “auxiliary quantities of interest”¹, but knowing just what those quantities might be in any given application is hard to know in advance, meaning that pre-programmed functions are only going to get you so far. Moreover, we should be driving the software, not the other way around: the methodological frontier should not be the drop-down menu of program *P*: or, as I said in 1994 (if a little too earnestly):

once methodological problems start being perceived or even defined in terms of what one’s favorite software does well, then the software has stopped being a tool, and has become a crutch, and at worse a shackle.

The implication is that easy programming and flexibility is key for a serious statistical computing environment.

To demonstrate function writing in R, suppose we want to compute the area under the receiver-operator characteristic (ROC) curve, a goodness of fit measure for binary classifiers (including, as a special case, logit and probit models), and closely related to other measures of association such as Somer’s *D*, Gamma, and τ_a . These goodness-of-fit measures are already present in R via Frank Harrell’s `Design` library, but let us proceed for purposes of exposition.

¹Note the enthusiasm for this type of add-on functionality King, Tomz and Wittenberg (2000) provided to STATA users).

Say we have a model that produces $\hat{p}_i = \widehat{\Pr}(y_i = 1)$ and we classify $\hat{y}_i = 1 \iff \hat{p}_i > k$ and $\hat{y}_i = 0$ otherwise, for some threshold $k \in [0, 1]$. The ROC curve plots the TPF (true positive fraction, $\text{TPF} = 1 - \text{FNF}$), versus the FPF (false positive fraction) as k varies over $[0, 1]$. The resulting function is defined on the unit square, and the area under the ROC curve C is interpreted as a measure of the classification success of the model. A value of $C = .5$ indicates random predictions and a value of $C = 1$ indicates perfect prediction. This is a useful and widespread measure of classification success and one that we'd like to be able to compute any time we fit a binary response model, suggesting that coding it up as a function is worthwhile. Suppose we've fitted a binary response model in R using the usual `glm` command, for generalized linear models, e.g.,

```
logit1 <- glm(y ~ x, family=binomial)
```

producing an object of class `glm`. The following function will compute C , the area under the ROC curve:

```
rocarea <- function(glmobj){
  p <- predict(glmobj,type="response")
  y <- glmobj$y
  n <- length(y)
  ones <- y==1
  n1 <- sum(ones)
  n0 <- n - n1
  C <- (mean(rank(p)[ones]) - (n1+1)/2)/n0
  C
}
```

exploiting the fact that

$$C = \frac{n_1^{-1} \sum_{\{i:y_i=1\}} r_i - \frac{n_1+1}{2}}{n - n_1}$$

where n_1 is the number of observations with $y_i = 1$, $r_i \in \{1, \dots, n\}$ are the ranks of the predicted probabilities, and n is the sample size. To use the function we'd type

```
rocarea(logit1)
```

and the answer would appear on screen.

R is also much better about memory usage than Splus; iteration and looping (essential to simulation-based estimation and inference) was and remains infeasible in Splus, but it is much more plausible to use R for simulation. The general purpose optimizer in R also

improves over that in `Splus`; the user need only supply a function to compute the objective function (e.g., a log-likelihood) and the `optim()` function will not only find optimal parameter values, but will also compute a Hessian matrix (minus the inverse of the Hessian is the usual estimate of the variance-covariance matrix of MLEs) without the user supplying functions for first and second derivatives; the optimizers in `Splus` would not provide a Hessian without functions to evaluate at least the first derivative of the objective function. `R` also has the very nice property that its binary data files (`.RData` files) can be automatically read across all hardware and operating systems: e.g., `R` for Windows reads the files created by `R` for whatever else (various Unixes, linux, and my own favorite, Mac OS/X).

On the other hand, `Splus` has a nice GUI, at least for its Windows product; indeed, the GUI is one of the ways `Splus` has been providing “value-added” over its `S` engine, fighting back against its free competitor, `R`. There are a number of projects underway for GUIs for `R`, `Splus` also has a nice set of hooks into `Excel` (permitting easy data import/export), whereas `R-to/from-Excel` is more circuitous. The `foreign` package in `R` imports/exports data sets from `Stata`, `SPSS` and `SAS`, among others. But in short, much of the “Plus” part of `Splus` that made `Splus` famous came from the public domain, and has been ported to `R` (e.g., generalized linear model, survival analysis, time-series, multivariate analysis, classification, bootstrapping, various smoothers and density estimators, multiple imputation for missing data and so on); almost everything I used to do in `Splus` I can do in `R`.

In summary, if you are attracted to `Splus` because of its fundamental strengths (easy user-extensibility, object-oriented design philosophy, interacting with data via visualization, and its widespread use among statisticians) and you’re not afraid of the command-line (a CLI, not a GUI), then you’ll like `R` and may already be using it. Otherwise, consider downloading the base distribution of `R` from <http://lib.stat.cmu.edu/R/CRAN> and see for yourself; the price is hard to beat.

References

- Jackman, Simon. 1994. “GAUSS and SPlus: a comparison.” *The Political Methodologist* 6:8--13.
- King, Gary, Michael Tomz and Jason Wittenberg. 2000. “Making the Most of Statistical Analysis: Improving Interpretation and Presentation.” *American Journal of Political Science* 44:341--355.