

# A critique of R and S-PLUS

Eric Blair

28 February 2004

A few publishers have shown interest in my book on statistical programming. One asked me why I'm using C programming language instead of the more popular R. I told her that R broke my heart, and that I could write a thousand word essay on R's problems. Though, it seems I've blown clear through that word limit here. I've decided to focus on the non-technical discussion of why stats packages in general have made the practice of science worse off, and then tacked on a few S-specific notes at the end for the programmers.

**Stats packages encourage data snooping and data mining** In olden days, the idea behind doing statistics was that you'd develop some theory about the world based on your observations, and then you'd gather data with the explicit intent of testing the theory you'd come up with. The premise was that people are eminently creative, and will make sense of whatever noise gets thrown at them, so hypothesis testing is thus built around rejecting spurious theories, rather than proving anything. [The wording is 'we reject' or 'we fail to reject'.]

There are two problems with this ideal in the real world. The first is that data gathering is expensive, and an efficient division of labor requires us to have people who work full time on producing large, general data sets, such as everything you ever wanted to know about a corner of the Black Sea, or the Current Population Survey (CPS) which asks a number of households a few hundred questions every month. The second part is that all the easy hypotheses you'd develop by thinking hard and watching stuff have been hypothesized, and we're now looking for more detailed information that can only come from looking closely at the general data sets there.

This draws us very far from the ideal: before, we were looking at the world at large and then gathering data to test the hunches we develop therefrom, and now we're looking at the general-purpose data sets and then using the same general-purpose data sets to test the impressions they'd just given us. The data can no longer be used to weed out the spurious hypotheses—and using next year's CPS to test hypotheses based on this year's CPS is still not going to weed out nearly as much as it should. But going out and gathering a purpose-built data set built around your hypothesis is hopelessly expensive, especially since every last grantmaking body you talk to is going to ask you, 'Why aren't you just using the CPS?'

This is a big problem, because there's a lot of data in the CPS, and it can easily be proven (I have an amusing cite from Freedman available on request [update: see below]) that data mining from a set that large will always be fruitful. If somebody hands you a

hundred uncorrelated, random variables, you can easily find five of those variables that explain another of the random variables with very high significance, nice t-statistics, a huge  $R^2$ , et cetera—but by construction, it was all just noise. The CPS is not noise, of course, but when our statistical tests do as well with noise as with real data, then something in the overall method is wrong.

Hypothesis testing after snooping around the data can't be used to falsify spurious guesses, which puts science up a creek without a proverbial paddle. But I can offer one constructive suggestion: stop depending so much on interactive stats packages. These packages let you graph your data in a hundred ways, compare every variable with every other, and run regressions on the same data. That is, they encourage data snooping and data mining.

Now, there are situations where people are not being ultraskeptical scientists, but are just trying to come up with some interesting facts for the boss, or a few new ideas. In this case, data mining is not such a bad thing, provided you bear in mind that there's still a good chance that your observations are pure luck. But it's essential to not confound these methods with those of hypothesis testing if hypothesis testing is to have any legitimacy.

A representative line from the *SPSS 8.0 Guide to Data Analysis*, a textbook a friend of mine used for a general statistics class, published by SPSS Corp. “You should always plot the data first and then think about appropriate methods for summarizing them.” [p 133] The author (Marija Norusis) is clearly more interested in summarizing data than testing hypotheses, which is cool, but why does she then segue into the chapters on testing hypotheses without any note (that I could find) that the techniques are for different and basically contradictory goals?

As for R and S-Plus, their data presentation is carefully designed to facilitate data mining. Here is a truly interesting interview with the guys who came up with Trellis graphics, in which they explain the great pains they went through to make patterns pop out of the visual displays of data, and the psychological foundations upon which this stuff is based. Again, this is great for any of a number of applications, but not hypothesis testing. They dance by the issues I discuss here by saying that the graphic display “allows you to make a good guess about an initial model to fit and then to diagnose how well it fits the data” but then chide the interviewer for saying that you might as well run explicit tests for goodness of fit of every variable against every other. [There are valid reasons for graphing data in a hypothesis-testing context, such as making sure that your results aren't driven by outliers, and that the data's range is basically sane and doesn't indicate errors by surveyors or instruments. But these are things you do *after* you've decided on what your regression will be about, and don't require nearly the visualization power that stats packages provide and encourage the user to use.] Of course, this is true of *all* existing statistics packages; I single out S because of what's to follow, but they all suck for this reason, and as a whole, stats packages have given statistics a bad name. As Professor NS of Saint Louis, MO, advises: never trust a regression run after 1975.

**S is call-by-value only** On to the computer-geek part, whose primary lesson is that you shouldn't tie the user's hands unless you have a really good reason. R and S-Plus are based on the S language, developed at Bell Labs. This is the same lab that gave us

B and C, so I guess the name is not particularly surprising. Anyway, the decision was made by the designers to not include a call-by-reference mechanism in the language.

By way of explanation for the people who are still reading but not familiar with this, it's all about calling a function like `product(A, B)`. One approach (call-by-value) is to treat the function like a black box. The program throws copies of A and B into the black box named 'product', which mangle these copies in any manner that seems expedient and then pops a product out the other end of the black box. The original versions of A and B don't change, since you only put copies into the black box.

The other option (call-by-reference) is to put A and B themselves into the box, and then the program returns the product, as before, and also potentially modified versions of A and B. The black box metaphor sort of breaks down, and your function becomes a component that is intermingled with the rest of the program.

There are loads of difficulties with call-by-reference; have a look at this little page, for example, for a long list of considerations. But to explicitly disallow it is to shoot oneself in the foot. If we had a language which allowed only call-by-reference, then we'd have no problem implementing a call-by-value function: just write `local_A=A` as the first line of the function and don't use A for anything after that. On the other hand, if your language allows only call-by-value, there is no way to implement a call-by-reference function, which is a pain. There are ways in S that involve unpleasant convolutions of the language, overuse of the global name space, et cetera, but as workarounds go, they are especially ornery.

The implications of this are many. First and foremost, the language is slow. Even with tricks to only copy when absolutely necessary (which add lots of code and fragility in and of themselves), the language is still a few hundred times slower than equivalent stuff written in C.

[7 April 2006: Not to be a total theorist about it, I did a in entry #172. I would call the results impressive.]

Second, it makes good coding more difficult, since you can't just move code in your main function out of the way into a subfunction when it makes things more readable. In many (but not all) cases, you'll need to do a rewrite as you move.

Third, it dictates how the program is written: *everything* winds up being an assignment. Some people write like this anyway, and it makes for very nice looking code when everything is lined up neatly. But some things don't lend themselves to being assignments, and there's no alternative available. To give a concrete, trivial, and painful example, I sorely miss being able to just write `var++` to add one to `var`. In R, I need `var<-var +1`, which is OK for such an easy case, but `var[2,other_var*7-2 == var[3,]]<- var[2,other_var*7-2 == var[3,]] +1` requires an editor with a good cut and paste function and gives me a little more redundant code to keep synced up. In other languages, I'd write an `increment(x)` function which would unroll to `x <- x+1`, but in S, such a simple desire is impossible.

[Update: You'll see in the comments below that since I wrote this, R gained a limited call-by-reference capability.]

**S has no forced typing** One of S's main delightful features over the other stats packages is that it allows for arbitrarily complex data types, notably data frames which gather lists of different types of data. Also, everything in S is implicitly typed, based

upon what S thinks you mean. These two features are painfully contradictory. My main problem with this is with building a list of lists: say the first element of your list is  $L[1]$ , and then your second is  $L[2]$ . Unfortunately, that first line there told S that L is an array of integers, not an array of lists of integers, so the second assignment will produce a complaint.

This is a simple case, but things get much worse when dealing with real data structures. And yes, there are ways around it, and yes, some of them even work. But in the end, they're all apologies for the fact that there's no way to just declare  $L$  to be a list of lists and not let S coerce it to a list of integers. Since I came to S because I was so enamored of its complex data structures, the fact that so much coercion goes on behind your back—and often can't be prevented—makes the lack of declared types a deal-breaker for me.

I could go on with a few other minor details that don't make for such a coherent essay, but I think these are already sufficient to tell us that S is a bad idea for the sort of work I talk about: hypothesis testing and simulation. If you have a data set which you just want to interrogate, you're not doing any heavy-duty programming, and you're not publishing in an academic journal, then by all means, R will work for you, and even has some nice features over other stats packages. But beyond that, you're better off with any of a number of options.

[2 Feb 2006: Somebody did actually request the Freedman article, so since I finally looked it up, here's the citation:

```
@article{freedman:screening, author = {David A Freedman}, title = {A Note on Screening Regression Equations}, journal = {The American Statistician}, volume = 37, number = 2, month = may, year = 1983, pages = {152-155}, url = {Jstor} }
```