Uniform model forms and microsimulations

Ben Klemens

25 June 2013

This is the first of an N-part series whose broad story arc will be about applying statistical tools—notably Bayesian updating and parametric hypothesis tests—to microsimulations and agent-based models. I will post every three days.

I wrote a paper[1] (PDF) on many of the topics I'll cover. That paper focuses on designing and using black box models as the basis for a modeling system. It may sound trivial, but I wound up writing two dozen single-spaced pages on the implications. The paper is still a draft and I expect to present some small subset of it at a workshop at SAMSI[2]. So my self-interested goal in posting is to get your comments and suggestions.

The first few episodes of this blog series will be an informal discussion of the problems and benefits of fitting models into uniform black boxes. On the problem side, is it reasonable to fit 'nonparametric' models into parameter-centric form? What happens when we have a stochastic likelihood function? On the benefit side, we're going to have methods to measure the variance of the parameters of any given model.

Once we've surmounted the technical details, then we can put the sort of simulations people don't think of as models into the model box, and use the above black box tools directly to calculate parameter variances or assign priors to ABM inputs and produce posterior distributions for those inputs.

Models as black boxes By way of introduction to this little series, here is a question I have often pondered: how have our general systems for statistical and scientific computing progressed past FORTRAN '77? F77 has great matrix facilities built in to the language; is Turing Complete, so you can write anything in it; and has good libraries for random number generation, optimization, linear algebra, regressions, or statistical distributions. That list basically summarizes the capabilities of the typical scientific computing language today. [Graphics and a nice REPL are splendid, but that's about the environment, not the language itself.]

I'd be interested to hear your take on my F77 question, but my own response is in the form of Apophenia[3], the library of stats functions I've been working on since 2005. It is built on two basic structures. The `apop_data` mashes together vector, matrix, and text elements, and handles text and row/column names, but like matrices in Stata or Matlab, or data frames in R, it's just a few conveniences beyond a FORTRAN array.

[1] `http://ben.klemens.org/pdfs/klemens-modelcats.pdf`

[2] `http://www.samsi.info/programs/2013-14-program-computational-methods-social-sciences-cmss`

[3] `http://apophenia.info`

The `apop_model` structure describes a statistical model. Given a model, we can estimate it with data, make random draws from it, transform it or combine it with other models to produce new models, or send it to functions like optimizers or bootstrappers. Having a structure like this in FORTRAN '77 is on the edge of possible. [I picked F77 for my question because it doesn't have a grammar for structures (aka objects). All of the more modern languages do, and yet so few use them to represent anything in the statistical domain beyond data sets.]

As this series will demonstrate with running examples, there are a lot of benefits to wrapping all the details of a model into a single object, which can be referred to by a single name when interrogating and dissecting it, and to the typical programmer, this sort of encapsulation is natural. I've had a variant of this conversation with several people:

Me: I think statistical models should be first-class objects in our languages. They should have a consistent interface, whether they're linear regressions or Bayesian posteriors, and we should have lots of functions written around that single form, like `estimate(data, model)` or `boostrap(data, model)` or `posterior_model = update(data, pri`

Imaginary Friend: That's completely obvious. Say something interesting.

Me: Do you have any examples of computing platforms that are built around a general model object like that?

IF: Well, not exactly. But you could implement this in R in, like, an hour.

Me: Could you point me to an R package that does this?

IF: Not exactly, no. But I'll bet nobody's done it because it's just so obvious.

Given so little precedent for model structures—especially ones that can accommodate a microsimulation, the next few episodes will cover some of the less-obvious details about making this obvious concept a reality. Then we can get to transformations and applications.

[Reader, if you have examples of a system built on a generalized model object, I'd love to hear it, so I can expand my lit review. As for R packages, I already know about Rapophenia[4].]

[4] `http://r-forge.r-project.org/projects/rapophenia/`