# Multiple imputation's setting

Ben Klemens

24 October 2010

This is really part II of the last entry #036. M.I. or one of its friends is really essential for honest analysis. If you have missing data, you have some model for filling it in, and it's better to measure the variability added by the missing data model than to just ignore it and pretend the values you filled in are correct with certainty.

So, then, ¿why aren't these techniques absolutely everywhere missing data is found? People in some fields, like your survey jockeys, are entirely familiar with this problem, and would never fill in a value without properly specifying that model. Other fields and the systems that support them expect you to reinvent the tools as needed.

There's a multiple imputation function for Apophenia, which basically does the last step or two of the multiple imputation process for you: given a series of fill-ins, find the statistic for each, and apply the within/across variance formula. It was a bear to write, and not because the math is all that hard. In fact, Apophenia is built from the ground up around the use of models in the sort of plug-in format, so if there's any stats system out there where writing a function to find a statistic using a parent model crossed with a fill-in model, it'd be this one. But look at how much has to be specified: parent model, possibly one model for each column of missing data, a statistic that uses all of those models at once, and the base data in a format that the first three items know how to work with. All of these—especially the aggregation of several models for each variable into a unified missing-data story—have to be specified and tested by the user before calling the multi-impute function.

As above, Apophenia has a standardized model object that can be sent to functions and thrown around internally without much fuss; to the best of my knowledge, it is currently unique in that respect, so other systems need to come up with a fussier means of specifying how the multi-impute function is to make its draws and aggregate everything together.

There's an R package named *mi* which solves the problem by requiring the user to use specific set of models, based on a Bayesian framework preferred by some of the pioneering works in multiple imputation, and as per the last episode, the combination of models can easily make use of Bayesian model-combining techniques. To use the package, you pick from a short list of named models, and away you go.

Quick—if you start with a Dirichlet prior with parameters $[\alpha_1, \alpha_2]$ and a Multinomial likelihood function $[\beta_1, \beta_2]$, what will the posterior look like? OK, time's up: it's a Dirichlet distribution with parameters $[\alpha_1 + \beta_1, \alpha_2 + \beta_2]$. So using this specific form dodges a computational bullet, so it's the sort of thing that is the focus of the `mi` package. I'm being a little unfair with this example, because the package allows much more flexibility than just this simplest of model combinations, but it's also a far cry from accepting any type of input model crossed with any type of fill-in model. R is Turing Complete, so you can do it, but expect to start from near zero and brush up on your S4 object syntax.

By which I mean to say that crossing one model against another is basically the limit of what we can organize with the tools we have today, which is a little sad.

**Presentation** But let's say that you're one of those people who assumes away the problem of organizing two models (one of which is a compound model for several variables) plus a statistic-calculating function plus a data set as just trivial and to be assumed away; then you still have (1) the problem of having users understand that these are the inputs they are to provide. Remember that most users got an education in a traditional statistics course that taught a series of plug-and-play finalized techniques and procedures. If all you know is OLS, then estimating an aggregate of OLS and missing data generation process is mindblowing.

Then, (2), there's the output problem. There are a number of possible outputs: most verbose would be to (A) actually report the several imputations for every data point, or you could (B) report the variance of each imputed value, or you could (C) report the larger variance of the final statistic and not bother with the internal workings.

Option (A) is the most voluminous data, and has been advocated by a decent number of people, especially for the case where there are separate people on the data-gathering side and the data-analysis. The gathering side could give ten filled-in data sets to the analysis side, and leave the analyst side to calculate the statistic of its choosing and trust that it will apply the simple total across/within variance equation correctly.

Option (B) is a middle-ground that gets difficult. We want to know how well the imputed values are fitting in, especially when the imputations are more complex than a simple multivariate Normal. This is where good data visualization comes in. We've had literature on the problem of presenting too many data points for decades now, and we have established means of coping. But in this case, the data is both voluminous and complex: each point has a span around it, and data may or may not be missing on different dimensions, meaning that that different confidence blobs may have different dimension. This is something we're still working on.

The `mi` package focuses on this, providing a heap of plots of the imputations for use in diagnosing problems. The authors do a fine job of giving good views of what is fundamentally too much information, but it's still a lot to digest, and would be hard to throw into a journal article with only a few sentences of explanation.

Option (C) is certainly the easiest to the consumer, because everything has finally been summarized to a single variance, and the reader doesn't have to care about whether the main cause was within-imputation or across-imputation variance (though that'd be easy enough to report as well). Now your only problem is to explain to readers that your variance is larger than the next guy's because you took into account problems that the other guy didn't.

The problems for all of these options at these various levels of aggregation are real but surmountable. In each case, the problem is in education: the end-user, whether an analyst or a package user or a reader, needs to understand that we're combining a parent model with a data generation model, what a good or bad data generation model will look like, and how to fit this combination model into a world where most models are just a single surface model. That is, multiple imputation is also at the edge of what a typical statistics education can accommodate.