Apophenia, a library for modeling

Ben Klemens

16 June 2015

Apophenia version 1.0[1] is out!

What I deem to be the official package announcement is this 3,000 word post at medium.com[2], which focuses much more on the *why* than the *what* or *how*. If you follow me on twitter[3] then you've already seen it; otherwise, I encourage you to click through.

This post is a little more on the *what*. You're reading this because you're involved in statistical or scientific computing, and want to know if Apophenia is worth working with. This post is primarily a series of bullet points that basically cover background, present, and future.

A summary Apohenia is a library of functions for data processing and modeling.

The PDF manual is over 230 pages, featuring dozens of base models and about 250 data and model manipulation functions. So if you're thinking of doing any sort of data analysis in C, there is probably already something there for you to not reinvent. You can start at the manual's Gentle Introduction[4] page and see if anything seems useful to you.

For data processing, it is based on an `apop_data` structure, which is a lot like an R data frame or a Python Pandas data frame, except it brings the operations you expect to be able to do with a data set to plain C, so you have predictable syntax and minimal overhead.

For modeling, it is based on an `apop_model` structure, which is different from anything I've seen in any other stats package. In Stats 101, the term *statistical model* is synonymous with Ordinary Least Squares and its variants, but the statistical world is much wider than that, and is getting wider every year. Apophenia starts with a broad model object, of which ordinary/weighted least squares is a single instance (`apop_ols`).

By assiging a format for a single model structure:

- We can give identical treatment to models across paradigms, like microsimulations, or probabilistic decision-tree models, or regressions.

[1]`http://apophenia.info`
[2]`https://medium.com/@b_k/on-bringing-stories-to-data-or-the-trouble-with-the-cubists-e3c6ff8f91fd`
[3]`https://twitter.com/b__k`
[4]`http://apophenia.info/gentle.html`

- We can have uniform functions like `apop_estimate` and `apop_model_entropy` that accommodate known models using known techniques and models not from the textbooks using computationally-intensive generic routines. Then you don't have to rewrite your code when you want to generalize from the Normal distributions you started with for convenience to something more nuanced.
- We can write down transformations of the form f:(model, model) $\to$ model.
  - Want a mixture of an empirical distribution built from observed data (a probability mass function, PMF) and a Normal distribution estimated using that data?

    ```
    apop_model *mix = apop_model_mixture(
                            apop_estimate(your_data, apop_pmf),
                            apop_estimate(your_data, apop_normal)
                          );
    ```
  - You want to fix a Normal(μ, σ) at $\sigma = 1$? It's a little verbose, because we first have to set the parameters with μ =NaN and $\sigma = 1$, then send that to the parameter-fixing function:

    ```
    apop_model *N1 = apop_model_fix_parameters(
                            apop_model_set_parameters(apop_normal, NAN,
                                 1));
    ```
  - You want to use your mixture of a PMF+Normal as a prior to the μ in your one-parameter Normal distribution? OK, sure:

    ```
    apop_model *posterior = apop_update(more_data,
                                 .prior=mix, .likelihood=N1);
    ```
  - You want to modify your agent-based model via a Jacobian [`apop_coordinate_transform`], then truncate it to data above zero [`apop_model_truncate`]? Why not—once your model is in the right form, those transformations know what to do.
- In short, we can treat models and their transformations as an algebraic system; see a paper I once wrote[5] for details.

What v1 means

- It means that this is reasonably reliable.
  - Can the United States Census Bureau rely on it for certain aspects of production on its largest survey (the ACS)? Yes, it can (and does).
  - Does it have a test bed that checks that for correct data-shunting and good numeric results in all sorts of situations? Yes: I could talk all day about how much the 186 scripts in the test base do.
  - Is it documented? Yes: the narrative online documentation is novella length, plus documentation for every function and model, plus the book from Princeton University Press described on the other tabs on this web site, plus the

[5]`http://arxiv.org/abs/1502.02614`

above-linked Census working paper. There's a lot to cover, but an effort has been made to cover it.
  - Are there still bugs? Absolutely, but by calling this v1, I contend that they're relatively isolated.
  - Is it idiot-proof? Nope. For example, finding the optimum in a 20-dimensional space is still a fundamentally hard problem, and the software won't stop you from doing one optimization run with default parameters and reporting the output as gospel truth. I know somebody somewhere will write me an angry letter about how software that does not produce 100% verifiably correct results is garbage; I will invite that future correspondent to stick with the `apop_normal` and `apop_ols` models, which work just fine (and the OLS estimator even checks for singular matrices). Meanwhile, it is easy to write models that don't even have proven properties such as consistency (can we prove that as draw count $\to \infty$, parameter estimate variance $\to 0$?). I am hoping that Apophenia will help a smart model author determine whether the model is or is not consistent, rather than just printing `error: problem too hard` and exiting.

- It means that it does enough to be useful. A stats library will never be feature-complete, but as per the series of blog posts starting in June 2013 (entry #146) and, well, the majority of what I've done for the last decade, it provides real avenues for exploration and an efficient path for many of the things a professional economist/statistician faces.
- It means I'm no longer making compatibility-breaking changes. A lot of new facilities, including the named/optional arguments setup, vtables for special handling of certain models, a decent error-handling macro, better subsetting macros, and the `apop_map` facilities (see previously (entry #165)) meant that features implemented earlier merited reconsideration, but we're through all that now.
- It's a part of Debian! See the setup page[6] for instructions on how to get it from the Debian Stretch repository. It got there via a ton of testing (and a lot of help from Jerome Benoit on the Debian Scientific team), so we know it runs on a lot more than just my own personal box.

From here, the code base is in a good position to evolve:

- The core is designed to facilitate incremental improvements: we can add a new model, or a new optimization method, or another means of estimating the variance of an existing model, or make the K-L divergence function smarter, or add a new option to an existing function, and we've made that one corner of the system better without requiring other changes or work by the user. The intent is that from here on out, every time the user downloads a new version of Apophenia, the interface stays the same but that the results get better and are delivered faster, and new models and options appear.
- That means there are a lot of avenues for you and/or your students to contribute.
- Did I mention that you'll find bugs? Report them and we'll still fix them.

[6]`http://apophenia.info/setup.html`

- It's safe to write wrappers around the core. I wrote an entire textbook[7] to combat the perception that C is a scary monster, but if the user doesn't come to the 19,000-line mountain of code that is Apophenia, we've got to bring the mountain to the user.
  - For R, there's Rapophenia[8]
  - For Julia, I presented version 0.01 of a Julia wrapper (entry #173).
  - There's a perl wrapper[9].
  - The esteemed Josh Tauberer threw together a zeroth draft of a Python wrapper[10].

By the way, Apophenia is free, both as in beer and as in speech. I forget to mention this because it is so obvious to me that software—especially in a research context—should be free, but there are people for whom this isn't so obvious, so there you have it.

A request I haven't done much to promote Apophenia. A friend who got an MFA from an art school says that she had a teacher who pushed that you should spend 50% of your time producing art, and 50% of your time selling your art.

I know I'm behind on the promotion, so, please: blog it, tweet it, post it on Instagram, make a wikipage for it[11], invite me to give talks at your department. People will always reinvent already-extant code, but they should at least know that they're doing so.

And my final request: try it! Apophenia doesn't look like every other stats package, and may require seeing modeling from a different perspective, but that just may prove to be a good thing.

[7] http://www.amazon.com/exec/obidos/redirect?link_code=ur2&camp=1789&tag=caltechdivini-20&creative=9325&path=tg/detail/-/1491903899/qid=1120157199/sr=8-1/ref=pd_bbs_ur_1

[8] https://r-forge.r-project.org/projects/rapophenia/

[9] https://github.com/swuecho/apophenia-perl

[10] https://github.com/JoshData/apophenia-python

[11] https://en.wikipedia.org/wiki/Apophenia_(software)