Do what I mean

Ben Klemens

4 August 2013

This interstitial post gives some discussion of the design goals of Apophenia. I'll refer to it a few times when I get to things that some systems do automatically that Apophenia insists that users do manually.

The key point is the difference between interactive and script-oriented design. It's increasingly the case that people expect to be able to use one language both interactively and as an unsupervised and reliable moving part in a bigger machine, but the two goals are in some sense at odds. As an interactive user, I want the machine to do what I mean (DWIM), not force me to explicitly enumerate tædium and minutiæ; as a reliable component, I don't want the system to make guesses at what should happen, because sometimes those guesses will be wrong.

Why your R script broke At work, people take me as an authority on R, even though there are many people in the building far better with it than I. So I get to hear a lot of complaints about R scripts that don't work. Passing on the package management, version mismatches, and other sort of environment issues, here are the top three reasons a script worked yesterday but didn't today:

- What had been numeric data yesterday was now being treated as text data (and turned to *factors*, a numerically-indexed list of categories).
- A text variable was converted to factors without the user's knowledge, and something funny happened with the factors.
- A script written to take matrix input got a $1 \times N$ matrix, which R cast to a vector.

All three of these are situations where the system did something for the user without the user's knowledge, allegedly to help the user out, and what it did was too situation-specific to generalize. To give you a clearer image of the first two cases, here is a typical story:

- Monday: Data comes in. The OBS column includes observations $\in \{1, 2, 3, 4\}$, which are matched to the OBS2 column in a second data set, also $\in \{1, 2, 3, 4\}$. Analyst writes the script; it runs and produces good results.
- Tuesday: Data comes in. The first value of OBS is missing, and so is marked with `xx`. R reads the column as text. The script breaks. Analyst fixes it, forcing the column to always be text factors [1=1, 2=2, 3=3, 4=4, 5=xx]. It works again.

- Wednesday: Data comes in. The OBS column has no missing data, but today, by chance, there were no observations with value 3. The factors are now shifted [1=1, 2=2, 3=4, 4=xx], and the joinup to the second data set falls apart.

My point here is not to kvetch, and please do not send me defensive emails explaining to me the secret trick that my coworkers missed to making automatic and implicit factor conversions reliable. My point is to show that R was designed for interactive use first, and uploading to the server for batch processing second. The auto-casting DWIM features are a clear indication of those priorities, and really exactly what you want if you have a single data set in front of you and can check and correct any missteps as they happen. [And if you think I'm reading intent into details of syntax, get a copy of Becker, Chambers, and Wilks's *The New S Language*, and read about how much the designers of S (of which R is an implementation) pushed interactivity and reacting to output. The thought of running S headless only gets a half-page discussion on p 57: "There are occasions when interaction with S is unnecessary, e.g., when a large computation is to be carried out from a source file."]

Apophenia's intent is to allow kickass narrative model development (if you've been reading along, you know what I mean by that now) and to provide a statistics and data processing facility for the open source stack. You should be able to write a statistic-calculating function, post it on GitHub, and be reasonably confident that people with data nothing like yours don't have to bend over backward to use the function. Replicable processing is the priority, but we develop those reliable programs interactively, so the UI has to not hurt.

Thanks for reading. Tomorrow I'll post the logit regression I'd promised, and now you know why it will have a line explicitly casting a text variable to numeric values.