RData for our data?

published on 27 Mar 2014

data, **R**

There's a blog post by Francis Smart over at **Economics by Simulation** advocating that .RData files become the standard data exchange format within research.

For those people who aren't R nerds: .RData is a serialised, R-specific file format that can be used as containers for a (potentially unlimited) number of R objects - functions, lists, data frames, you name it. Because of this, it's used for error handling (R dumps your workspace into an .RData file if you encounter somethin that unexpectedly ends a session) and data exchange between R programmers.

Smart's argument for it to become the standard for data transfers between researchers full stop is:
1. .RData files are a heck of a lot smaller than the equivalent TSVs or CSVs;
2. .RData files are a heck of a lot quicker to *load* into R than the equivalent TSVs or CSVs;
3. Because .RData files are object-based, rather than data-based, you can include not only the dataset but also functions dedicated to exploring or

expanding on it.

These are pretty good reasons, and I'd add another obvious one, which is ".RData files don't contain data, they contain R objects that contain data".

If I'm exchanging data with a coworker (say, **Aaron**) or an external researcher, and it's structured as a list of vectors, I can send it over *as* a list of vectos, and it'll be properly structured when imported. If I'm sending over a data frame, I can send it over *as* a data frame, and the included variables will have the right type. Conversely, if I'm sending a list over in a TSV...well, I can't. I have to unlist it and turn it into a single vector, which causes a loss of all of the breakpoints. If I'm sending a data frame over as a TSV, I can absolutely store it as a TSV - but the type of each variable when it's read back in will either need to be manually specified, or will be down to the whims of the read.delim() function and the options the user at the other end has set on their REPL.

Despite all of this - the speed, the size, the increased structural nuance - I *don't* think .RData should be a data transfer standard.

Don't get me wrong, it works perfectly well for projects with Aaron and I involved, because Aaron and I both use R. But an .RData file is no use for, say, **Dario**, who uses Python, or **Erik**, who uses Perl; they have to write R to read it in, turn it into something they can write back out, and then read it into their language of choice. Smart brings up the examples of **Stata, SPSS and Excel** as formats R can write to, but even when that is supported, you still run into having to read it into R in the first place, and **R isn't the only statistics environment out there**.

More importantly: at least in the chunk of the research and data community I tentatively occupy, we don't provide and exchange open datasets just for other researchers. We do it because being transparent to the *wider* community - data consumers, management, the general public - is part of the job description. .RData files are small and structured, yes, but they also impose an artificial barrie to accessibility. We can't be expecting data consumers to learn part of a programming language to access our datasets. TSVs and CSVs might be larger and slower, but they're also *universally accessible* - you can plug a TSV into any stats or spreadsheet program or language made in the last decade, and it goes "oh, a TSV. I know what to do with those". And if you're looking at things manually? Hey, guess what: TSVs are human-readable. .RData files are a block o seemingly-meaningless four character strings.

So, it's a nice idea; it would work for people sending me datasets. But that's the only place where it works - sending datasets to a subset of the researcher population. For the rest of those researches, and more importantly, for the data consumers, .RData files are pretty much opaque.

Share this post