The setup, 6 July 2013

Ben Klemens

Inspired by Mr.s Apiolaza[1] and Lumley[2], here's my technical setup. It's pretty different from what I understand to be the norm, and I can't imagine a blog entry in under three pages, so I have some added background. Next time, it's back to the regularly-scheduled computational epistemology.

Who am I and what do I do? I work at a large government bureaucracy doing large-scale computing on data sets the size of the United States population. I've written a few books on tech policy and computing technique. I'm very interested in better social science modeling.

What hardware do I use? If you asked me about my setup when I was a poor college student in London, I would have shown you a box with three floppies, including all my work, a basic TeX installation, and sundry other utilities. When I got to a computer lab (remember those?), I'd spend ten minutes copying setup files from the floppies and then get to work. I eventually upgraded to a five-disk box, and added a C compiler and text-mode spreadsheet program.

I can still do this: if you give me a computer with no operating system, I can make myself entirely at home rather quickly. This isn't hypothetical: I am a klutz and break pretty much anything you give me. I think I have to do a reinstall from scratch maybe every year and a half (not to mention the new instances of virtual machines in The Cloud, which also start as a *tabula rasa*).

I asked Hadley Wickham[3] what he saw for the future of computing, and one of his points was that we currently bring the data to the computer that does the analysis, but in the future we'll bring the analysis to the data, because it's easier to copy over some scripts than to mirror a many-terabyte, constantly-updated dataset. At work, due to bureaucratic restrictions, I already often have to bring my analysis to a data set that has to remain encased in a certain metal box.

Writing—code or plain text—is interesting for the imaginary structures that we produce. The hardware is just the palimpsest we use to get there. In recent memory, I haven't paid more than about $250 for a laptop, because if I pay more I'll feel bad when I break it. That's what I paid for the Dell just-above-a-netbook that I am writing this on. In the past, I usually bought refurbished or off-lease netbooks off of eBay.

[1] `http://www.quantumforest.com/2012/04/my-setup/`
[2] `http://notstatschat.tumblr.com/post/52445567432/my-setup`
[3] `http://had.co.nz`

So that's the hardware that I use to develop high-performance computing software. Because I am comfortable using the tools I develop on a disposable netbook, I know that people on top-of-the-line rigs are having a great experience, and that a user who was lucky to score a four year old hand-me-down can also comfortably use my software.

I use an external ergonomic keyboard every chance I get. I also have an external monitor, so when at home I don't even look at the laptop itself. At work, I ssh in to the same laptop, so the physical hardware is again largely irrelevant. I replaced the spinning-metal hard disk with an SSD, and expect to keep doing that until cheap laptops come from the factory with SSDs.

Other hardware: my telephone is a Palm. It has a terminal application so I can ssh in to the Dell, and I can use it for tethering, because the phone companies are too busy trying to restrict Android and iOS users to bother with me and the other four remaining Palm users. It has a hardware keyboard and plays mp3s and takes photographs just fine. I have a large-format ebook reader, because much of academia is about PDFs, and it's nice to look at something that isn't an LCD screen where possible. I accept that both of these things will break, and don't seriously depend on them (beyond the basic telecommunications part of the telephone).

I have a big pile of backup drives. They have saved me many times over, both from hardware failure and my own mistakes.

What software do I use? What's on the screen follows somewhat similar principles of being as portable as possible.

First, if I need a license or some other sort of permission to use the software, then it's not portable. It's that simple.

My bureaucracy asked me to produce an Application Inventory, so I already have a list of what I use for work; here it is with some modifications for our current purposes.

- Content:
  - Things specified in the IEEE POSIX standard[4]. These tools aren't very modern, but they are so stable and reliable—I trudged up the learning curve almost twenty years ago (I recall riding the Circle Line all night, reading a UNIX textbook from the library) and everything I learned then still works. Key elements of POSIX:
    - vi, because I have very much internalized the keymap. I often leave jjjjjs in word processor documents.
    - The shell. I switched to zsh as my shell, which has better interactive conveniences; it handles the vi keymap better.
    - make, which is a nice way to organize ad hoc shell scripts.
    - A C compiler (gcc or clang) and its attendant libraries.
    - sed, awk, m4. Increasingly, when I see a nail, I use m4 as the hammer, and take comfort in knowing that it is prevalent even though nobody knows they have it.

[4] `http://pubs.opengroup.org/onlinepubs/009695399/`

  - TeX and friends. I sometimes wonder what we'll all be using fifteen years from now; I don't think it'll be LaTeX.
  - Python, primarily for dealing with systems whose API is a Python package. I keep meaning to use Python more, but then end up just writing in C because it's just as easy.
  - R, increasingly only for graphics. R packages written by the public are especially unreliable in the sense that I describe here: scripts from three years ago often have little quirks that make them unrunnable on today's R installation. We spend a lot of time at work finding workarounds for these little things.
  - Gnuplot and Graphviz [less and less frequently; see R].
  - A database library. I've been relying on SQLite. I take comfort in knowing that you can build an SQLite interpreter into your program by including one C file. Readers of this blog know that my livelihood depends on the GNU Scientific Library, SQLite, and Apophenia.
  - Git is increasingly an essential part of my workflow, and is my preferred method of collaboration. It has such a strong user base that I'm confident that it'll be around in fifteen years.
  - I do backups with rsync and a simple linking trick. It's a two-line script; ask me about it if you're curious.
  - Having read a few of these make-use-of posts, there's always a point where the author says 'and I use an office suite so I can collaborate with others.' I don't put too much effort into this fight either, and use OpenOffice as needed, but I find that it's easy to amicably get around this. Typically, I send people plain text or a text list of revisions, and they can fold things into their documents better than I can.

- Interface
  - My window manager is Icewm, which I've been using for about fifteen years, so I have all the keybindings internalized. The purpose of a window manager is to get out of the way, and I get annoyed by always-present menu bars. Icewm even lets me eliminate window title bars—do you really need a label to know which window is your browser?
  - A terminal. At home, Gnome-terminal; at work, Cygwin's mintty.
  - tmux[5], a terminal multiplexer. Given that I have the terminal full-screen most of the time, it's basically my window manager. I spend most of my days with a file (TeX, C, R script, ...) on the left side, and a compiler/debugger/interpreter on the right; I've found this setup to be all the IDE I need. And then I have another file/compiler pair, yet another file/compiler pair, my text-mode IM client, my always-open to-do list, my text-mode email client (mutt) all within at most four keystrokes. Plus, I can call up the same session on my screen at work, the laptop, the telephone, or that

[5] `http://tmux.sourceforge.net/`

external monitor.

- A browser. I switch around depending on what seems the least intrusive this season. Right now it's Opera, but I might drift back to Firefox soon.
- A PDF viewer. On Linux, it's Evince. At work, the one from Adobe.
- If I'm in front of a computer, I have headphones on. My mp3 player of the moment is a slightly hacked version of cmus. You can use xbindkeys to get the play/pause/skip buttons to work with most music players, so I rarely actually look at it. My only nonobvious requirement is that I can write the current track to last.fm[6].

[6] `http://www.last.fm/user/lasttotheparty`