ECON 815 - Computational Methods for Economists

Fall 2019

Notes on Productivity Enhancing Software

Big Picture

- You want to be a productive researcher, which means spending less time on tasks that aren't helping you uncover new facts or answer import economic questions.
- To do this, we need to invest in some human capital and learn how to leverage software to maximize our research productivity.
- That's what this class is about.
- But in this first week, we'll focus on less on software with direct application to solving or estimating economic models and more on software that is helpful in general.
- What are these?
  1. TeX for typesetting a document.
  2. Text editors for writing code.
  3. Interacting with software through the command line
  4. Using Git for version control
  5. Using GitHub to work collaboratively
  6. Conda for managing Python packages.

TeX

- TeX is a typesetting system
- In short, you use a programming language to define the objects in your document
- This is distinct from a work processor like MS Word, where WYSIWYG
- Why use TeX over Word?
  1. Free
  2. More control - especially for formatting tables, figures, equations
  3. Faster typesetting for equations (once get past initial learning curve)
  4. BibTeX for bibliography
     - Makes it very easy to change formatting per journal requirements
     - Easy to grab bibliography info from IDEAS, http://www.jstor.org, or other places.
  5. Once learn syntax for symbols in TeX, you can use this in other applications such as MS Word's equation editor and Jupyter Notebooks.
  6. Aids in organizing large documents - like dissertations!
     - Can link multiple files into one document
     - Auto general table of contents, handle equation/table/figure numbering across all files
- To download the TeX software:
  - Unix/Linux: TeX Live

  - Mac: MacTex plus TeXShop frontend
  - Windows: proTeXt (comes with TeXStudio frontend)
- Useful software when using TeX:
  - Excel2Latex plug-in for Excel
    * Allows you to export an Excel table to TeX markup.
    * I often use as follows: Statistical software -> text table -> text table linked to Excel table that provides formatting -> Excel2LaTeX to bring table into TeX document.
    * Not fully automated, by often nice to see all results together in Excel without having to compile in TeX
  - A good text editor with TeX plug-ins (discussed below)

Text Editors

- A good text editor is critical.
- You'll use it to:
  - Write code (in various languages)
  - Write documents in TeX
  - Take notes
- What are key attributes of a good text editor?
  - Low overhead (i.e., loads quickly, not a memory hog)
  - Syntax highlighting
  - Dictionary/spell check integration
  - Plug-ins for:
    * Linters
    * Compiling TeX code
    * Other handy utilities...
- Recommendations:
  1. Atom (free)
  2. VS Code (should be installed along with Anaconda)

Using the command line

- It's often useful to interact with software though the command line
- Sometimes (e.g., on remote server) this will be your only option
- So you'll want to know how to do this.
- On a Mac: Applications -> Other -> Terminal
- On Windows: Click "Start" -> type `cmd` and press Enter
- There is a learning curve, but here's a quick reference for the handful of commands I use most often:
- I'll suggest you use the command line when using Git and running Python (noted below)

Table 1: Unix and DOS Commands

Command	Unix	DOS
Change directory	`cd <directory path>` `<(could be relative path)`	`cd`
List files in directory	`ls`	`dir`
Move up one level in directory structure	`cd ..`	`cd ..`
List current processes	`ps`	`tasklist`
Kill a running process	`kill <process id>`	`Taskkill /PID <process id> /F`
Connect to remote machine via secure shell	`ssh -p <port number> <user@hostname>`	`<path to PuTTY.exe>` `-ssh <username@host> <port number>`

Transfer files to a remote machine (via Secure Copy)	`scp [options]`
	 `<username1@source_host:directory1/filename1>`
	 `<username2@destination_host:directory2/filename2>`
		`pscp -scp [options]`
		 `<username1@source_host:directory1/filename1>`
		 `<username2@destination_host:directory2/filename2>`

Submit a batch script	`qsub <filename.sh>`	Unlikely. If so, see here

a) Windows has no built in `ssh` utility. Need to install PuTTY or other software to use `ssh`.
b) Windows has no built in `scp` utility, so you need to install PuTTY and PSCP

Git and GitHub

- Git is an open source version control system
- Git Hub is an online platform to help with collaboration on coding projects
  - As the name suggests, it builds upon Git
- Git and Github do have steep learning curves! But it's worth it.
- What does Git do?
  - Version control! Keeps track of changes to code - noting differences line by line
  - Allows you to "tag" versions of your code
    - E.g., you might tag a version of your code that you used to generate results used in a submitted article. In this way, you can always reproduce what you submitted to the journal, even if you go on to edit that code base, you use this tag to go back to the old state of the code.
  - You can create development branches while keeping your main code in tact - only merging in changes when you know they work
- What does GitHub do?
  - Acts as a remote server where you can store your code (so it's backed up in the cloud)
  - Allows you to collaborate with others - you all can submit changes to the same code base stored in a GitHub repository
  - Provides a web interface for your project
- Tips for you all using Git and GitHub:
  - There are GUI interfaces for Git, but I recommend learning to use Git through the command line.
    - Here are the most common commands you'll use often:
  - GitHub has student accounts that allow unlimited private repositories. See here
    - As much as I like the idea of open source projects, I think you might benefit from having repos private as you work in grad school (you can always make public later - e.g. after publish paper based on project)
  - You might not have as much use for GitHub in grad school as you typically do projects solo, so the collaboration aspect is not as helpful.

Table 2: Common Git Commands

Functionality	Git Command
See active branch and uncommitted changes for tracked files	`git status -uno`
Change branch	`git checkout <branch name>`
Create new branch and change to it	`git checkout -b <new branch name>`
List all branches	`git branch`
Track file or latest changes to file	`git add <filename>`
Commit changes to branch	`git commit -m "message describing changes"`
Push committed changes to remote branch	`git push origin <branch name>`
Merge changes from master into development branch	(change working branch to master, then...) `git merge <branch name>`
Merge changes from development branch into master	(change to development branch, then...) `git merge master`
List current tags	`git tag`
Create a new tag	`git tag -a v<version number> -m "message with new tag"`
Pull changes from remote repo onto local machine	`git fetch upstream`
Merge changes from remote into active local branch	`git merge upstream/<branch name>`
Clone a remote repository	`git clone <url to remote repo>`

Anaconda

- Anaconda is a distribution of Python
- This means that is installs Python as well as a set of Python packages together
- In addition, Anaconda includes:
  - Spyder, and IDE for Python (with interface similar to Matlab's IDE)
  - iPython
  - Jupyter Notebooks
  - The Conda package management system
    - This system makes it easy to update/install packages
    - You can also set environments so you can run your programs on multiple machines (or on your machine at separate times) and know that it is using the same version of Python and dependent packages.
- You'll want to download the most recent Anaconda which will install Python 3.6 and most of the packages will need
  - e.g., Numpy for numerical programming and SciPy for scientific computing