Christopher Lang

STA9794 – February 24, 2017 – Homework C

# Question 1

Setting up a good developer toolchain is essential for developer speed, efficiency, and work correctness. Programming can be a complex art; the workstation must be setup properly, such as having the correct environmental variable set up, the correct compilers (if applicable), code testing suite, etc.

Essentially, programming has a multistep, recursive workflow. As you shouldn’t copy and paste the same code with slight modification vs. a looping construct, you should not need to constantly set up your computer just to test your work. Having a well thought out, developer workflow would ease the development process in a significant way, allowing the developer to deploy correct production ready work quickly and efficiently

# Question 2

I’ll start with my work computer, as it is my primary development environment for my data science work and would be the same in general. I heavily use both R and Python, but RStudio integrates most of the functionality needed. For Python I forgo IDEs and favor text editors

1. A simple notebook
   1. The notebook provides the quickest way to get ideas down and formulating it into a product. The primary goal is to define the problem, how it can be solved, and developing a implementation plan (plan of attack)
2. Sublime Text Editor
   1. Code is of course written in a text editor and the choice here is Sublime Text Editor. Its fast, customizable, and extendible with plugins
   2. One of the plugins I use is Anaconda IDE, providing autocompletion and syntax checking for Python
   3. When applicable, I also use Sublime built-in build functionality. With a keyboard shortcut, I can run the code within Sublime’s console. But generally, for larger and complex work, I use iPython in a terminal
3. Autogenerating key directories for project
   1. In data science, it is important to document your work but also keep the project portable and organized. I’d generally auto-generate all directories necessary, though I won’t use all of them
   2. For example, a data directory for storing relevant data
   3. A source folder to hold all source files
   4. A library folder to hold class and function definitions relevant to the project
   5. Etc.
4. iPython and ConEmu terminal
   1. Python provides an interactive interpreter which I use to test code. Pieces of code is essentially copied from the text editor and pasted into the command line to run
   2. iPython also provides other useful functionalities such as code timing and debugging
   3. This is not as efficient as using an IDE, but I haven’t found one for Python that I liked, preferring the “old school” method in this case
5. GIT with GitHub or GitLab
   1. Every time a working piece of code is implemented and integrated with the whole, the changes are committed and pushed to Github (Gitlab at work). To maintain sanity and organization, GIT branches are created
   2. A master branch is used for deployable code. It actually doesn’t have to be feature complete, it just can’t break
   3. A dev branch, used to add new features and or fix bugs found in master. There can be multiple dev branches, one for different features or bugs
   4. A trans branch, used to transfer current development code from one computer to another. Often, there are local commits I do not want pushed to Github or Gitlab just yet, but know that I need to transfer to another computer to work on it. This is the branch for that

Additional requirements are needed when working with CUNY’s HPCC. After the above:

1. Code transfer to Linux VM, work with WinSCP
   1. A special Linux server is setup to test the code to ensure it works on Linux. This is relevant as most of the work is done on a Windows machine. Every effort is made to ensure the Python code is written in a portable manner, but nothing beats testing it on the actual platform
   2. At work, this would be a Linux VM. For this class, I’ll have a Linux VM set up on my local computer. It would not be fast, but enough to check code correctness vs. performance. Chances are, a bash script would be needed to setup the environment, such as setting the right environment variables
   3. The code transfer would occur via Github. The Linux VM would have a text editor, iPython setup as well to allow for development on there as well. Changes made would be pushed to Github
   4. Sometimes there are data files I do not transfer to GIT due to size issues. WinSCP, with connection via SCP, would be used to transfer files
2. Code transfer to CUNY VM
   1. This step mirrors the last step. The primary goal here is to test parallelism; I would want to make sure the program is communicating with the various nodes in the cluster correctly
   2. This is also where scalability will be tested. Synthetic data of significant size will be generated for program speed test
3. Job submission bash scripts
   1. This has not been developed, but from class it’s clear that job submission is done via command line, which can be scripted in bash

Test on Linux environment similar to HPCC. Likely a local VM

**Commit Code**(GIT, Github)

**Test code HPCC**(iPython, WinSCP)

**Test code remote**(iPython, WinSCP)

Do this over and over until your happy

**Commit Code**(GIT, Github)

**Test code local**(iPython)

**Write code**(Sublime Text)

Simple notebook for getting thoughts down in writing

**Thoughts in writing**(simple notebook)