Greengraph Report

David Wise

January 11, 2016

1 Introduction

This report gives some supplemental details on the assignment to package the Greengraph software for release over github. Along with the basic Greengraph classes, taken from the UCL software development library, I have included License, Citation and Readme files as well as a series of tests for the greengraph functions.

2 Entry Point

The software can be installed by downloading the GitHub repository and navigating to the setup.py file, created using the 'setuptools' library and running it with the command:

```
sudo python setup.py install
```

A command line entry point has been created for the software using the 'argparse' library and which allows users to implement the software form the command line using the following form:

```
greengraph -b ‘Start’ -t ‘End’ -s ‘# of steps’ -o ‘output.png’
```

Running this command will output a graph showing the proportion of greenspace between two geographical locations, the location parameters are required whilst number of steps and output file are optional. If these are not provided then the program will default to 10 steps and to displaying the graph via the Matplotlib show() function.

3 Difficulties

During the course of this project I encountered several difficulties, mostly centred around my inexperience with programming and particularly with the creation of the test functions, which made up the bulk of my work. The creation of the setup and command files took less time despite my relative lack of familiarity with the process.

One technical challenge came from the use of the Geopy library which interacts with Google's online servers in order to download satellite imagery, calls to these are limited by Google, a feature that made testing occasionally frustrating. To avoid this I downloaded a picture which I then called in my test functions using mocks, allowing me to avoid repeated calls to the internet. This was also helpful in the creation of tests that did not interact with the internet.
The use of the mocking functions were the source of the greatest difficulty and frustration to me, as I could never quite get a handle on their exact function. Whilst I was able to get them working well when testing the Map class - for example to feed my placeholder image to the various functions - I found it much harder to mediate the interaction between the Greengraph class and the Map class using them. As a result I was unable to get one of my test functions (greenbetween) to avoid the internet.

4 Advantages and Costs of Package Preparation

The preparation of software in this packaged form obviously comes with both positives and negatives. In the case of scientific software, releasing it for public use comes with the obvious cost of preparation time, whereby one needs to go to the trouble of producing tests, setup and command files, and producing some documentation of the software's use. All this before the inevitable requirement to patch and update the software as it becomes widely used. These costs will, however, likely be offset by the fact that such a process will almost certainly subject the software to far more rigorous testing than it would have experienced had it never left one computer. The need to gain a fuller understanding of the software in order to help others use it, as well as the inevitable bugs that others will find in it, will result in higher quality software in the long run.
The use of software libraries is in general to be encouraged, most tasks that you could wish to complete have already had software written to accomplish them. Instead of writing your own, likely poorer, code it is a much better use of your time to use pre-existing libraries and to help with their ongoing development. Package indexes such as PyPI provide sources for software that must conform to certain minimum standards making the use of such software that much simpler and more reliable. Using package managers in addition will simplify installation. These do come at the cost that they are often both language and operating system dependent, requiring software to be adapted so that it will work with multiple platforms.

5 Building a Community

I have taken the first step in building a community of users for Greengraph by making it available on GitHub for installation. In order to further progress there are several steps I ought to take. The first would be to test out my installation on multiple systems and to make sure it works on the most common

OS's, ensuring that the majority of potential users will be able to do so without significant headaches. Second I would look for people for whom my software might be useful and encourage them to have a look at it - perhaps passing it on if the like it. This will build a core group who will subject my work to some testing. Most importantly I need to make sure someone listens and responds to any feedback from these users, efficiently fixing any bugs they find. Doing this will reassure people that this is a project that will have ongoing support and which they can invest their time into without fear that it will be dropped before long.