1 Introduction

I've been working as an analytics developer for nearly two years now, and wanted to share the most helpful things I've learned in my transition from academia.

First, there are a dizzying number of tools and techniques to learn. The general framework I want to cover is:

1. Setting up and navigating a development environment from the command line (I use OSX and Ubuntu)
2. Scripting with python and R,
3. Saving and sharing work with git and github, and
4. Storing and accessing data via databases and sql

This will be opinionated, in that I'll typically present what I do as the right way to do things. Obviously there are other ways of doing things.

2 Before You Start

We'll be moving mostly without a mouse. It is terrible at first, but pays dividends. If you are using Ubuntu you can get to a terminal with `ctrl+alt+T`. On OSX, use `cmd + space` to open spotlight, type in "terminal", then hit enter. I'll give some pointers to the most important commands to know in the terminal. Some important ones to start:

1. `$ ls` lists the files in the current directory
2. `$ cd directory` changes your directory to `directory` (if it exists).
3. The up/down keys scroll through recent commands
4. `tab` will autocomplete when it can. Sometimes you need to hit it twice and it will give you options
5. `ctrl+r` Gives you reverse search in the terminal, letting you type in a few letters. Hitting `ctrl+r` more scrolls backwards in these options, and `ctrl+g` cancel your search.

3 Setting up and navigating a development environment from the command line

If you can use an Ubuntu box, it will be much easier to set everything up, but it isn't terrible on OS X, either. In general, the better you can find your question on Stack Overflow, the better you'll do longterm, so check there early and often if you run into trouble.

We'll be using a few different terminals, and when you are meant to actively type something, I will try to copy the prompt: bash (my terminal) uses `$`, Python uses `>>>` and R uses `>`. Code with no prefix will be a `message printed back to you`.

Go to the appropriate section to set up your own environment for general purpose processing.

3.1 Ubuntu

We'll be using Ubuntu 12.04. On Ubuntu, you get the Advanced Packaging Tool, which we invoke with `$ apt-get`. We'll install a few other utilities off the bat:

Get git Start with `$ sudo apt-get install git`. Typing `$ git` should now output something reasonable. Bad news if you see `-bash: git: command not found`, and you'll have to do some research to fix this.

Get python Already built in – I'd just use theirs.

Get R We want 3.0.2, and for this we need to add their repository before we can `apt-get` R. Check out this page for good information.

3.2 OSX

Get Homebrew. Instead of `apt-get`, we'll use Homebrew. OS X ships with ruby, so you should be able to copy/paste the command at the Homebrew website with no problem.

Get git Now we can run `$ brew install git`. Typing `$ git` should now output something reasonable. Bad news if you see `-bash: git: command not found`, and you'll have to do some research to fix this.

Get python Python is already built into OS X (open your terminal and type `$ python` to start hacking away), but you want a different version of Python. Luckily, this is nice `$ brew install python`. This will install Python 2.7.6, which is what we'll be using. There are many discussions about Python 2 vs Python 3. We follow the advice of this one, and use Python2 while writing code that will be Python3 compatible. The main differences are that division is no longer integer division (in python2, `>>> 3/4 = 0`, in python3 `>>> 3/4 = 0.75` but `>>> 3//4 = 0`), more objects are generators, and the `print` function is invoked like a function.Python 3.4 was just released, but `$ brew install python3` will still install Python 3.3.3. Use at your own (minimal) risk.

Get R As usual, `$ brew install R` installs a good version of R (I am on 3.0.2).

3.3 Python Interpreter Hello World

Open your interpreter, type `$ python` and then `>>> print("Hello, World!")`.

That's a bit too easy. Let's instead run

```
>>> x = "Hello, World!"
>>> print(x)
Hello, World!
>>> print(x + x)
Hello, World!Hello, World!
>>> print(5 * x)
Hello, World!Hello, World!Hello, World!Hello, World!Hello, World!
>>> print(5 * (x + "\\n"))
Hello, World!
Hello, World!
Hello, World!
Hello, World!
Hello, World!
```

Notice that you can assign strings to a variable, and manipulate strings using addition in multiplication (and it works in a sane fashion). The `n` string is a newline. The other useful character to know for now is `t`, which is a tab. To quit the console, `ctrl+d` or `>>> quit()`.

3.4 R Interpreter Hello World