Python & Data Science

Getting Started with Anaconda

Todd Young

August 21, 2020

DSE511

Table of Contents

Motivation

Orientation in Data Science

There are many directions that you go in data science. This is a broad, interdisciplinary field, and people pick up tools that are necessary for their work as they go. You are likely already using some of them.

Here are a few goals that Jacob and I have laid out for this course:

- Introducing you to tools that are broadly applicable
- Giving you a mental model of how to think about these tools
- Teaching best practices to work in this field
- Showing how we continue to learn and adapt as times change

Choices of Tools

There are tons of tools out there. We won't be able to cover them all, but we have chosen tools that are broadly useful, even outside of data science. One of the main choices we contended with was programming language we will be using to present this material, Python.

Why Python:

- The language is mature
- Tons of open source libraries
- It is useful in and out of data science
- Much of the state of the art in machine learning uses it

Anaconda

What is Anaconda?

Anaconda is a distribution of Python. This means a fresh install of the language and a large amount of pre-installed packages that we can make use of. It also includes a package manager and an environment manager called conda.

Two main things Anaconda gives us:

- package management
- environment management

Welcome to Python Dependency Hell!

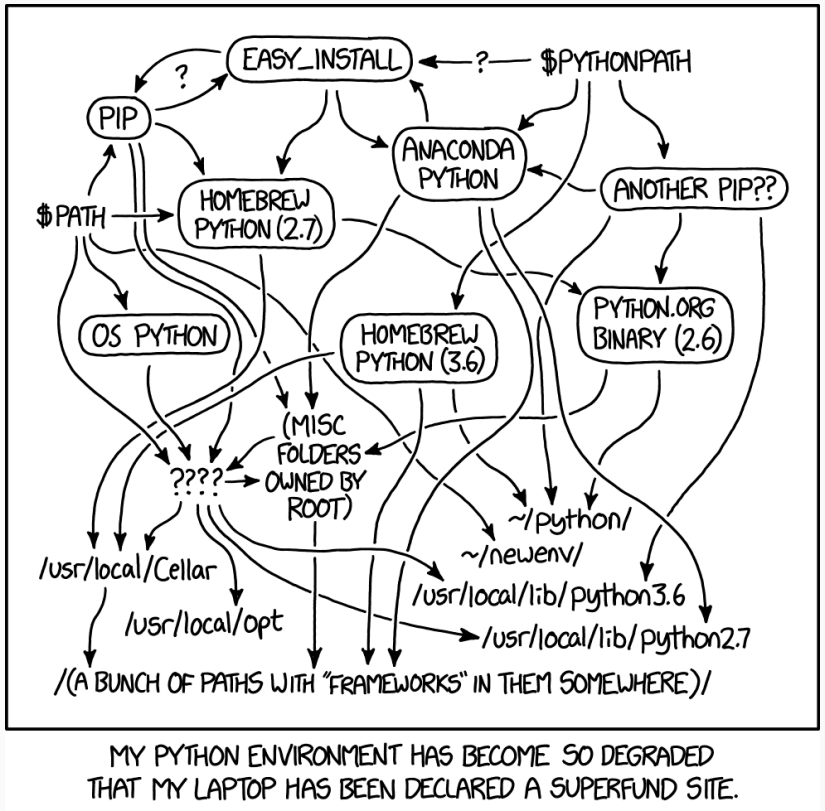

Figure 1: Software dependency management can sometimes be a nightmare of our own making. Over time, things can get hairy. **https://xkcd.com/1987/**

What about Pip?

There are other tools out there that can manage packages, pip being the most famous. What are the differences between these options?

- Anaconda handles environments
- Anaconda can install libraries that depend on other languages
- Anaconda does not require you have compilers to install its packages

But what about Pip!

No judgment. We're cool.

Pip is a great tool and is the officially supported tool by the Python Packaging Authority (which sounds serious). Pip installs packages from the Python Packaging Index, PyPi.

Sometimes packages aren't available through Anaconda. In those cases, pip is the way to go. Luckily, we can use pip in conjunction with Anaconda.

Garden of Forking Paths

I would be remiss if I didn't tell you about some of the alternatives out there. For the adventurous among us, who laugh at broken environments and who need to see what those other options are like...

You might consider:

- Pipenv
- Poetry
- Hatch
- Nix (I put this one in for Jacob)

Jacob and I both tend to be the sort that will try a tool just to see what it does. This is true of languages, tools, and rare Linux distributions. Being adventurous, trying things, and breaking things is a great way to learn.

But beware! There be dragons.

Installation

Graphical Installer:

https://www.anaconda.com/products/individual

Command Line Installer (Linux and MacOS):

Grab an install script from the Anaconda archive (designated by *Filename* in following *wget* command).

```
wget https://repo.anaconda.com/archive/<Filename>
bash <Filename>
```

Demo

Because live coding is always a good idea.

Where does Anaconda Live?

Linux:

/home/<your-username>/anaconda3

MacOS:

Graphical Installer: /opt/anaconda3

Command Line Installer: /Users/<your-username>/anaconda3

Windows:

C:\Users\<your-username>\Anaconda3\

Note:
When using the command line installer, Anaconda will ask you if you would like it to initialize Anaconda for you when you open your terminal. It will then add a bit of bash to your .bashrc. If you are not familiar with what this means, I would suggest saying yes to this option. If you are like me and don't want Anaconda littering your .bashrc (or if you are using a different shell, I can show you my work around).

Conda Commands

There are several useful commands when interacting with Anaconda. These allow you to

- Search the Anaconda package index
- Create new conda environments
- Install and update packages in conda environments

Checking Available Packages

List packages in a conda environment:

```
conda list
```

Search available packages from the Anaconda Package Index:

```
conda search
```

or

```
# <package-name> can be a partial name if you can't remember what it is called
conda search <package-name>
```

Installing Packages

Install a package:

```
conda install <package-name>
```

Optionally you can install a specific version of a package:

```
conda install <package-name>=<version>
```

Where <version> is a semver version number

Environments

What is an Environment?

An environment is a directory of specific packages and their dependencies. They allow us to isolate our collection of libraries so that we can account for functionality changes between different versions of packages.

When we first install Anaconda, we get a default environment called the *base* environment.

Anaconda's package and environment manager called *conda*. This is what we were interacting with in the earlier Conda Commands slides.

Conda Directory Structure

So environments are directories that contain Python packages. What is this directory structure?

/**anaconda**: the Anaconda root directory

/**pkgs**: decompressed Python packages

The following collections of directories determine a conda env:

/**bin**: binaries

/**include**: header files

/**lib**: library files

/**share**: architecture independent data