Fundamentals of Biostatics: Pre-course Computational Module

The goals of this module are:

1) Install the Anaconda distribution of Python and Jupyter Notebook.
2) Learn to do basic tasks in Jupyter Notebook

This short module will show you how to install and use the software you need for this course. We will write code in the Python programming language. We will write and run our Python code in the Jupyter Notebook application (or sometimes called IPython Notebook).

It is important to install this software and become a little familiar with it before class starts. Follow these instructions to install the software on the laptop you will bring to class. You can get help installing this software at the pre-class workshop at 10:15 on September 27, in Holden auditorium. At the workshop, we will also do some warm-up programming in Python and practice using the Jupyter Notebook, so that everyone is ready to jump right in on the first day of class.

Part 1: Install the Anaconda Python Distribution

Anaconda is a free, pre-packaged collection of important software for running Python. It includes Python itself, the Jupyter Notebook application, and specialized software tools for specific tasks like performing statistical analyses and making plots. These software tools are conveniently collected into a free “distribution” called Anaconda, making it easy to install everything you need.

To download and install the Anaconda distribution, follow the simple, step-by-step instructions here: https://docs.continuum.io/anaconda/install

There are separate instructions for Windows, Mac OS X, and Linux. (For Mac OS X, choose the graphical install link, unless you are comfortable working from the command line.)

IMPORTANT NOTE: The instructions include a link to the downloads page for the Anaconda installers (https://www.continuum.io/downloads). On this page, be sure to choose Python 3.5, not 2.7 or 3.4.

Once you've installed the software, move on to Part 2. We will verify that it's working correctly and practice using Python in the Jupyter Notebook. If you have trouble installing the software, come to the September 27 pre-class workshop and we can help you.
Part 2: Launching Jupyter Notebook

During the course, you will run Python programs and perform data analyses using an application called Jupyter Notebook. Jupyter Notebook is essentially an interface that you will use to run Python code, much like you use a web browser to access files and software on web servers. In fact, Jupyter Notebook runs inside of your web browser (though you will use it to access programs and data files on your local hard drive, not the web).

In Part 2, you will learn how to start up Jupyter Notebook and navigate within it. You’ll create a class folder, where you will store your files for this class.

Step 1: Launch Jupyter Notebook

There are two ways to launch Jupyter Notebook – the easy way, and the slightly less easy way.

Easy way: Double-click the Anaconda Navigator icon that was created by the Anaconda software installer (which is likely on your Desktop):

The Navigator application will open a window that looks like this:

To launch a Jupyter Notebook session, click the blue ‘Launch’ button on the Jupyter Notebook panel. Jupyter Notebook will open in your default web browser. It may take a few moments; be patient.

Slightly less easy way: If you’re comfortable working from the command line, you may prefer to launch Jupyter Notebook from there. First, open a terminal window. (On a Mac, launch the Terminal app in your Application/Utilities folder. You can also type Terminal in the Spotlight search field to quickly find and launch Terminal. In Windows, select Start -> Run and type cmd in the field.)
From the command line, type `jupyter notebook` and hit return. A Notebook session will launch in your default web browser.

**Step 2: Create your class folder**

Jupyter Notebook will open in your browser window and present you with a page that looks something like this:

![Jupyter Notebook Interface](image)

This may look like a web page, but it's not – you're using your browser to view folders on your hard drive, not something on the web. But inside your browser, Jupyter behaves like a web page. Each folder name is a link, including the blue folder names in the gray header bar. The folder at the top of the list with two periods next to it is a link to move up one level from your current folder. Click these links to navigate around your hard drive's filesystem. Go ahead and try it out – in this class, never be afraid to explore!

Let's create a class folder for your files for this class, including data files and Notebook (.ipynb) files. (More on .ipynb files later.) Within your Jupyter browser window, click the blue links to navigate to wherever you plan to keep your class folder. Then create this folder by clicking on the **New** drop-down menu in upper right corner, and select folder. Name your folder ‘biostats.’

You’re now ready to write some code.

**Part 3: Get to know Jupyter Notebook**

Python is a programming language that you use in two ways. First, you can use it to write **scripts** – complete programs, saved as text files that you run again and again, each time you need them. Second, Python can be used **interactively**, much like a graphic calculator: you type a command, and Python immediately gives you an answer. Running scripts is a good way to automate tasks that you do routinely, such as processing data files or performing the same analysis over and over. Python’s interactive mode is more useful for exploratory data analysis.

In this course we will primarily use Python interactively, and we will use Jupyter Notebook as our way of interacting with Python. Jupyter Notebook has several powerful features that make it useful for performing data analysis with Python. A major advantage of Jupyter Notebook is that it also serves as a lab notebook for computational work, by making it easy to organize, annotate, and save your code and data analyses – which is important for making your computational work reproducible by yourself and others.
Here we will run through some basic operations in Jupyter Notebook. If you are interested in getting more practice before class, there are several good tutorials on the web. Finally, at our pre-class session on September 27th, we’ll start learning some Python basics using Notebook.

1. **Create a new notebook:**
   
   When you launch Jupyter Notebook (Part 2), a new browser window opens the Jupyter dashboard. The dashboard shows you the files in the current folder (under the Files tab, top left, which opens by default).

   ![Jupyter Dashboard](image)

   You can’t write or run code from the dashboard – we run code from a Notebook. To create a new notebook, click on the New button (top right), and select “Python” under Notebooks. (If the menu shows more than one Python option, choose Python 3.) A notebook will open in a new browser window.

2. **Name your notebook:**

   In the notebook window, you’ll see that the default name is “Untitled.” Click on “Untitled” to open a dialog box. Then rename your notebook “Primer.”

   ![Untitled Notebook](image)

3. **Using the notebook:**

   A notebook consists of a sequence of **cells**. The first cell of your new notebook is the blank box below the menu (with the `In [ ]:` on the left side). You will type either python commands or regular text into these cells, depending on what type of cell it is. We will use two types of cells:

   - **Code cells:** Here you type Python code, which is then run when you hit `shift enter`.

   - **Markdown cells:** These cells are for text to organize and comment your code. You can create headers, descriptions, etc., just like you would in your lab notebook.

   Let’s start out by writing a header for the first part of our notebook. Click on the drop-down menu that says “Code”, in the toolbar at the top of the notebook window, and select “Markdown”:
Now click inside the blank cell and type: # **Python Warmup**. Hit *shift* and *enter*, and your notebook will now look like this:

To edit the header, just double-click on it to re-open the cell. Try typing ## **Python Warmup** (followed by *shift enter*) to see how the header changes.

After you finish entering text, a new code cell appears below. You can see that it is a code cell because the drop-down menu above says “code”. We’ll next try a few exercises in code cells.

**Math**: Doing basic math is much like doing it on a calculator (with some important exceptions which we’ll discuss during the course). In the next cell type `10 + 6` and hit *shift enter*. Python will show an output line with the answer.

**Print**: Often you’ll want Python to print some result to your screen – the value of a variable, the name of a gene, or an error message. To do this, we use the `print()` function. To see how this works, type `print('Hello World')` into a cell and hit *shift enter*.

Let’s try another example. Assign a DNA sequence to a variable named “DNA” by typing `DNA = 'ACGTTTGACCAGATCGA'` into the next cell. Then start a new line by hitting *enter* (not *shift enter*). Finally, type `print(DNA)`. Python will print the value of DNA to your screen:
Make a plot: There are some powerful graphic functions available for Python. Here we will make a simple graph of a sine function.

To do this, we must first load the software tools needed to make plots. These come from a package called matplotlib. Type the following commands. (Hit enter after the first line to start a new line, then hit shift enter after the second line to execute the commands.)

```python
%matplotlib inline
import matplotlib.pyplot as plt
```

The first command allows plots to be displayed in your browser window. The second line loads functions for making plots.

With our plotting tools loaded, let's plot some fake data. Type the following lines, then hit shift enter:

```python
drug_concentration = [0,1,2,4,8]
response = [0.2,15.1,39.8,90.5,100.2]
```

These commands create lists of data values and assign those lists to variables. To plot the data and create axis labels, enter the following commands, followed by shift enter:

```python
plt.plot(drug_concentration, response)
plt.xlabel("Drug (µM)")
plt.ylabel("Response")
```

Pro tip: You don’t have to type out the full names of the variables – type the first few letters, then hit tab and see what happens. This is called tab completion.

Hit shift enter and you should see your plot (it may take a moment to appear):
If you would like to go back and change any of your code, it's easy – just click in the cell you want to change, and begin typing.

**Save and quit:** Everything in your Notebook session is saved as a notebook file with a .ipynb extension. To save, select “Save and Checkpoint” from the Jupyter file menu. Then select “Close and Halt” from the file menu.

The notebook browser window will close, and you’ll return to the dashboard. You’ll notice in the dashboard that you now have a new file in your course folder called “Primer.ipynb.” All of your work is saved here. During your next session, you can re-open the notebook and start right where you left off, or rerun your previous code. You can now close the dashboard browser window.

**Important:** Closing your Jupyter Notebook browser windows is not enough to shut down! You also have to shut down the Jupyter server that was running behind the scenes. When you launched the Notebook application, you may have noticed that a terminal window opened. Close this window to completely shut down Jupyter:

Congratulations, you’ve completed your first Jupyter Notebook session! If it seems a little overwhelming, don’t worry – you’ll get plenty of practice with notebooks in class, beginning with our pre-class lecture on Sep. 27th.

**Where to learn more:** If you’d like to explore Python and Jupyter Notebook before class, here are a few helpful links:

