Bio5075 Fundamentals of Biostatics: Pre-course Computational Primer

In this class, we learn some basic programming skills necessary for statistical computation. For all class programming activities and homework, we will use the Python programming language within an application called Jupyter Notebook (also called iPython Notebook). Jupyter Notebook runs inside your web browser and makes it easy to write, run and document your code. (For the instructors, it also makes distributing, submitting, and grading homework much easier.) You can think of Jupyter Notebook as the computational equivalent of a lab notebook – an organized record of what you did, why you did it, and what the results were.

Before our class starts, you must have Jupyter Notebook installed and working on the laptop you will bring to class. (If you don’t have a laptop that you can bring to class, don’t worry – we can provide you with one. Please email the instructors at bio5075-admin@lists.genetics.wustl.edu.) Use this primer to install the software and make sure it’s running. If you have trouble, come to our pre-class help session on August TBD, and we will help you troubleshoot any issues. We will use this software on the first day of class, so it is critical that you complete this primer beforehand.

This self-guided primer will show you how to:

1) Install the Anaconda distribution of Python and Jupyter Notebook.
2) Launch Jupyter Notebook and move around your filesystem.
3) Do basic tasks in Jupyter Notebook.

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 a pile of specialized software packages 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

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 3.6, not 2.7. (See the figure on the next page.)

There are separate tabs for Windows, Mac OS X, and Linux – make sure you choose the installer for your operating system. (For Mac OS X, choose the graphical install link, unless you are comfortable working from the command line.)
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 pre-class workshop and we can help you.

**Part 2: Launching Jupyter Notebook**

During the course, we will use Jupyter Notebook to write code and analyze data. Jupyter Notebook is essentially an interface that you will use to interact with data files and Python software libraries, 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).

This part of the primer, you will learn how to start up Jupyter Notebook and navigate to different directories or folders on your hard drive. 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 (using your mouse), and the slightly less easy way (using your keyboard).

**Easy way**: Double-click the Anaconda Navigator icon that was created by the Anaconda software installer, which may be on your Desktop:
If you don’t see this icon on your Desktop, it might be in your home folder or the root folder for your hard drive, inside a folder called **anaconda**. (You can find this folder using the search function in Windows Explorer or Spotlight on a Mac). Once you find it, drag the icon onto your Desktop and keep it there for easy access.

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

To launch a Jupyter Notebook session, click the blue **Launch** button in the **Jupyter Notebook** panel. Jupyter Notebook will open in your default web browser. If you have an older laptop, this may take a few moments; be patient. Once the browser window opens, you’re ready to go!

**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.

(To 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.)

Then, from the command line, type **jupyter notebook** and hit return. A Notebook session will launch in your default web browser.

**NOTE:** When you launch **jupyter notebook** from the command line, be sure to be in your home directory. Once **jupyter notebook** is open, you unfortunately can’t navigate to folders higher than the folder from which the notebook software was launched.

**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:
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. Nonetheless, Jupyter Notebook behaves like a web page. Each folder name is a link, including all of 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 blue links to move 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 to hold all your files for this class, which will include 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’ or something similar.

You’re now ready to write some code. For more on installing launching Jupyter Notebook, here is the official beginner’s guide.

**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 that are saved as text files that you run again and again, each time you need them. Second, Python can be used **interactively**, much like a graphing 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 use Python interactively, and we will use Jupyter Notebook as our means of interacting with Python. Jupyter Notebook has several powerful features that make it useful for performing data analysis with Python. One major advantage of Jupyter Notebook is that it serves essentially as a lab notebook for computational work. Jupyter’s notebook format makes 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 Jupyter Notebook. If you are interested in getting more practice before class, check out some of the tutorials on the web, such as this longer one, and this short one.
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 Notebook Dashboard]

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.” This will create a file called “Primer.ipynb” in your class folder.

![Untitled Notebook]

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 – this will display the input line number). 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 the keys **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. You type text, then press **shift-enter** to exit the cell.

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. Then press both shift and enter. Your notebook will now look like this:

![Jupyter Notebook Image]

To edit the header, just double-click on it, and the cell will re-open. Try it – add another # to the text in the cell, to get ## Python Warmup (followed by shift enter). What happened to the header formatting?

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”, and it has In [ ]: on the left. 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. Below your code cell, python will return an output line with the answer. (Notice what happens to the In [ ]: on the left.)

**Print:** When you write code to do an analysis, 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 the print function 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 = ‘ACGTGACCAGATCGA’ (or some random sequence) into the next cell. Then start a new line by hitting enter (not shift enter – we still have more code to type in this cell). Finally, type print(DNA). Now hit shift enter, and python will print the value of the variable 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 python package called matplotlib. Type the following two commands. (Hit enter after the first line, type the second line, then hit shift enter to execute the commands.)

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

The first command loads the software to allow plots to be displayed in your Jupyter Notebook. The second line loads functions for making plots. Note that no output is displayed after running these commands. When the commands are complete, a new blank cell appears below.

With our plotting tools loaded, let’s plot some fake data. Type the following lines into the new cell, 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 inside any 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, or click on the button with the floppy disk icon. 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. (Try re-opening 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. Press Control-C to shut down the Jupyter server, then close your terminal window.

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. If this seems easy, then don’t worry – the class will get harder!

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


---

```python
In [3]: DNA = 'ACGTTCGACAGATCGA'
    print(DNA)
ACGTTCGACAGATCGA

In [4]: %matplotlib inline
   import matplotlib.pyplot as plt

In [5]: drug_concentration = [0,1,2,4,8]
   response = [0.2,15.1,39.4,90.5,100.2]

In [6]: plt.plot(drug_concentration, response)
   plt.xlabel('Drug (µM)')
   plt.ylabel('Response')

Out[6]: <matplotlib.text.Text at 0x10b5c0d6d8>
```