***Jupyter Notebooks and Numpy***

**Jupyter Notebooks**

*Why Jupyter?*

* Jupyter Notebooks allow us to the DS by combining notes, code, and graphics.
* Most importantly, combining these features allows others to read notebooks + understand motives behind each step + why decisions were mad 🡪 simply good collaboration + simply good science.
* If someone else wants to examine results or build on findings, they can see exactly how you conducted the research.
* Replication + inspection of other's methods has been a core element of the scientific process for centuries, but the advent of computational tools has caused discussion about how to best ensure such practices when conducting science using computing.
* Jupyter Notebooks aims to do just that by containing not just notes + process, but results as well.
* Jupyter supports Julia, Python, + R, 3 of the most popular programming languages for conducting data science today, along with other languages.

*Getting Started*

* **Dashboard** lists the contents of the folder you launched Jupyter notebook from.
* Can navigate around your file system through this, just like any other file explorer interface.
* The file system is where the **Jupyter notebook server** is running (could even be on another machine)
* The **notebook *application*** = a web application that creates the interface in the browser + executes Python + other languages' code.
* The ***notebook*** file is a file format w/ **ipynb** extension that saves code, images, + text in a single, easy-to-share doc.
* Executing a cell is executing code in a Python process called the **kernel**
* There is a kernel running in the background for each open notebook (browser tab).
* The Jupyter notebook web app talks to the kernel to have it load data + execute code.
* Can do any type of Python code in the notebook
* As simple as importing a module, printing a variable, or can be creating sophisticated analysis or graphs
* **Markdown** = simple text markup designed w/ simplicity, like a Word editor, w/ a bit of formatting beautification, as well as scientific editing
* Markdown cells have support for HTML + other text formatting languages like LaTeX
* Most think of markdowns as code commenting/documenting code, but we can think of it more like writing a scientific/white paper.
* You can create very sophisticated editing w/ markdowns, such as create equations using **LaTeX** equations.
* **LateX** is a document editing language which has a syntax for many text components for scientific writing, including writing equations (integrals, powers, exponentials, etc).
* LateX is great when implementing code from a scientific paper or an algorithm you'd like to try out.
* You can first write an elegant equation in LateX + then implement that in a Python function in a code cell right after that, which makes it a little easier to understand for readers.
* Although Python itself provides ways to use UNIX shell commands, Jupyter Notebooks provides an easier + more interactive way to use UNIX commands via an exclamation mark before commands
* Jupyter will use your default shell to execute these commands, so any adjustments needed to execute these commands should be based on the OS the Jupyter environment is set up on.

**Numpy**

*Why Numpy?*

* Offers # of key feature for scientific computing, most notably support for **multi-dimensional arrays**, (useful for representing vectors + matrices), as well as a # of operations to perform on matrices
* Operations include linear algebra (add, subtract, multiply matrices) + optimized statistical operations
* Tricky to make dimensions in vectors + matrices align properly, but Numpy takes a lot the work out of this by supporting **broadcasting** = makes this process easier + code easier to read
* Numpy is also fast enough for production-level code, so we don’t need to optimize it further
* But if needed, can work w/ o[optimized compile code form libraries w/in Fortran, C, and C++
* 3 reasons to do DS w/ Numpy in Python
* Fast (10x faster than lists)
* Numpy arrays are fixed in size to enable the speed (lists can change in size) + fixed to 1 data type (lists can have multiple)
* By restricting Numpy arrays like this, they’re much more space-efficient + opens up a range of memory + computational optimizations
* Functionality
* Average a vector in a matric, multiply matrices, subset matric based on index or values
* Even using many of these operations in Pandas
* Many packages in Python rely on Numpy
* Pandas is built on Numpy w/ higher-level functionality