Rules for Writing Software

This document details the rules for coding assignments for the class. The rules are intended to make your codes more readable, reliable, reusable as well as make the results produced by the codes reproducible.

## **Handling dependent packages**

You must ensure that any packages (dependencies) required for your code to run are installed. In a Jupyter notebook, this is accomplished by having a code cell that invokes the python installer (pip) for each dependency. For example, pip install -q tellurium. For codes written as python models (with a .py file extension), you will need to create a requirements.txt file that can be used in the command pip install -r requirements.txt. (The latter is a consideration for graduate student projects.)

## **Use functions instead of scripts**

Whenever possible, use functions instead of scripts. This is because functions facilitate reuse, and functions are testable. Never use copy and paste for reuse. For jupyter notebooks, a function should be defined in a separate cell that contains just the function definition and tests for the function.

## **Names of variables and functions**

Use meaningful names for functions are variables. Function names should be verbs. For example, a function that calculates a fast Fourier transform might be named calcFFT. A bad name for this function would be the single letter f.

## **Use named constants**

A constant is a variable whose value never changes. Constants should have a name in all capital letters. For example, use PI, not 3.141592. Named constants should also be used for dataframes. For example, instead of df["mean"] use df[MEAN], where MEAN = "mean" appears elsewhere.

## **Document your functions**

After the function definition, you should have comment lines that specify:

1. What the function does
2. The data types of each input and output (including names of columns if an input is a dataframe)

## **Functions must have tests**

You must have at least one test for each function that shows that the major code paths work correctly. In a Jupyter notebook, the tests should follow the function definition in the code cell in which the function is defined. The test will use the python assert statement to evaluate a boolean condition that constitutes the test. For python modules, you will create a separate test file that uses the python unittest framework.