Computational Biology
Tufts University Bio 35
Fall 2021

Assignment 1: Navigating the file system and running `Python`

need to update a few things, including Windows instructions and how we will log in to cluster

Welcome

In this course we will be programming in `Python 3`. The first assignment will serve as an introduction to basic coding necessities. It is an ungraded assignment (you will get two bonus points just for successfully turning it in), but it is critical that each element of this assignment is clear to you! Every step of the course will be otherwise greatly frustrating. Please get in touch with a peer, the TA, or the Instructor before the end of the second week of the course if some aspect of this assignment isn't working for you.

The first section of this assignment will concern our ability to navigate the file system and use the command line. Most of our projects will make use of `Jupyter notebooks`, which will mean that we will not need the command line interface very often for completing assignments. We include this material because you may sometimes wish to manipulate files on your own computer and navigating the file system is a basic skill you will need to translate the course material to another context.

The second part will consist of our first foray into Python programming. For this we will use `Jupyter notebooks`.

1 Installation of course software

1.1 Installing Python3

If you don't have Python3, you will want to install it by following the links under "Downloading and installing Python" here: https://sites.tufts.edu/datalab/python/setup. There are various ways to install Python, but we suggest following the instructions for Anaconda and using the conda package manager, which will allow you to install some useful software packages for the course.

1.2 Installing other required software

We will also be using several other pieces of software in class, including `numpy` and `stdpopsim`. You can install both using conda.

1. `numpy`: https://numpy.org/install/
2. `stdpopsim`: https://stdpopsim.readthedocs.io/en/latest/installation.html
3. `msprime`: https://msprime.readthedocs.io/en/stable/installation.html
4. `SLiM`: https://messerlab.org/slim/. Please note that SLiM might be challenging to install on Windows, but it is worth trying to install it from source (please ask for help if you haven't done this kind of thing before). We will have workarounds for Windows users available when we use SLiM, which will not be in the first few course sessions. With any of these steps, if it is taking more than a minute or two to figure out how to install, please come talk to us. It could be that a simple change will make it much easier, or it could be that something to do with your machine/operating system is interfering with proper installation.

If you are having trouble installing either of these packages please let the TA or instructor know!

2 Navigating Unix and Windows

2.1 Overview

You're probably familiar with navigating files on your computer through a graphical interface, such as the Finder application on Mac or File Explorer on a Windows PC. For coding, we may sometimes need another interface: the command line. The command line is a text-based interface that we will access through a "shell". There are various

different shells you might use, with `bash`, `csh`, and `zsh` being a few common ones. These shells are just programs that allow us to interact with the operating system on the computer.

The instructions in this assignment will assume that you have a unix-based machine (*i.e.*, a machine running linux or Mac OS), but you can also use a PC running Windows. In general you can write Python code on a Windows machine or a unix-based machine. From a practical perspective, the main difference between unix and Windows will be the specific commands we use to access the file system. In this lab we will provide examples for unix, but you can use your PC by substituting the unix commnads for the appropriate Windows commands. This will have no effect on the `Python` that you will write for this course, but will make a difference in how you navigate between files on your computer.

There are several very important commands that we will need to know, but the most critical are for listing files, changing directories, removing files, and moving files. In unix, these commands are:

1. `ls`: list contents of directory. A directory is really just a folder on your computer, and this command will just list the files on your computer.
2. `cd`: change directory. We will need to access files in multiple different directories in this course.
3. `rm`: remove. This command allows you to remove files from your file system if they are taking up too much space or no longer needed. You can remove entire directories with"`rm -r`", but you will want to be *very* careful with this command. You can not only lose data, but also damage your computer if you remove the wrong files.
4. `mv`: move. This will allow us to move files and also rename them.

In Windows, the equivalent commands are:

1. `dir`: list contents of directory.
2. `cd`: change directory.
3. `del`: remove.
4. `move`: move. This will allow us to move files and also rename them.

See here for more Windows commands.

2.2 What if I have a Windows laptop, but I want to use unix commands?

There are various ways to use unix-based commands on your Windows PC, including 1) dual booting another operating system on your laptop, 2) remotely logging in to a unix machine, or 3) download special programs for windows that provide a unix-like interface. We will not specifically support any of these options in the course, but feel free to contact the instructor/TA if you wish to learn more about these options.

2.3 Navigating on your machine

Download the folder for this assignment if you haven't already. Note the location to which it is downloaded. If you are on a Mac or Windows PC it will likely fall into the `Downloads` folder. You should be able to unzip the directory simply by clicking on it in the Finder (if you're on a Mac) or insert Windows instructions. If you're having trouble ask a peer, the TA, or the instructor.

Open a new terminal window. On a Mac, you can do this by typing "terminal" into spotlight. On a Windows machine you should use the File Explorer to find the command prompt (for more instructions try clicking here; ignore the steps about using `java`, just the first few steps under "What is Command Prompt?" are what we need).

2.3.1 Finding a directory and listing its contents

Once you've got your terminal open, you will need to navigate to the folder's location on your machine. Type "`pwd`" into the terminal window.

Example 0.0: Unix

```
[yourprompt]$ pwd
/Users/uricchio
[yourprompt]$
```

By typing "`pwd`", you are asking the computer to return the "present working directory", which is the directory that you are currently inside. In the example above, my terminal opened inside my home directory. The location of this directory is "`/Users/uricchio`". `Users` is a directory that contains the directory `uricchio`, and would contain the directories for any other users on my machine. In addition, be aware that the "/" at the very beginning of the address has a special meaning. It indicates the "root directory", *i.e.* the highest level directory that contains all the other directories. If we want to navigate to the root directory, we can type

Example 0.1: Unix

```
[yourprompt]$ cd /
[yourprompt]$ ls
Applications System         Volumes  cores        etc         opt Library        Users bin
```

We just navigated to the root directory, and listed its contents, which include the "Users" folder that contains my home directory. The location of a folder is also known as its "path" or "filepath". If the folder is inside your Downloads, its location is likely "`~/Downloads/Assgn1DataAndScripts`", where the "`~`" is a special symbol that denotes your home directory. In other words, on my machine typing "`ls ~`" would have the same meaning as typing "`ls `". Let's navigate to the assignment folder and start poking around.

Example 0.3: Unix

```
[yourprompt]$ cd ~/Downloads/Assgn1DataAndScripts
[yourprompt]$ ls
data scripts
[yourprompt]$ cd scripts
[yourprompt]$
```

Now we are in the assignment directory that we downloaded from the course website. Inside we can see the folders that we will need for this assignment, including scripts (*e.g.*, Python programs that we will execute as part of the course) and data (*e.g.*, genetic data we obtain from organisms that we will analyze with various scripts). The final step above (`cd scripts`) will put us into the scripts directory.

2.3.2 Executing programs

There are multiple ways to execute Python scripts from the command line. A simple way is to type "`python <scriptname> <argument 1> <argument 2> ... <argument n>...`", where `<argument 1>` indicates an input variable expected by the script. Typing "python" at the beginning invokes the python interpreter, which will turn our script into the machine code needed to run our program. For example, we might run the script "`first.py`" in our scripts directory. To do so we would type the following:

Example 0.2: Unix

```
[yourprompt]$ python first.py
hello!
```

This is a simple script that takes no arguments, and prints out a single word (hello!). If your script didn't work, it's possible that 1) you don't have python installed on your machine or 2) python2 is installed by default. It's easy to check. Type "`python -V`" at your command prompt. If you get an error that says something like "command not found", that means that you don't have Python. If this command returns something like "Python 2.7.3", that means that you have Python 2 on your machine. You will likely want to have Python 3 installed. If your machine already has Python 3 you can skip the next section.

2.4 Putting it all together

Now we will attempt to put everything together and run a simple simulation. For this step, we will use the software `msprime` to simulate genetic variation sampled from *Homo sapiens* under a model of human demographic history (*i.e., and evolutionary model for migrations and relationships between human populations*).

To do this, we will run a Python script that loads a simulation package called `stdpopsim`. The script is in the "scripts" folder that you downloaded for this assignment. Use the command "cd" to navigate into the "scripts" directory if you are not there already. The run the script by writing the following command at the command prompt

Example 0.3: Unix

```
[yourprompt]$ python simHuman.py
```

The script may take a minute to run. At the end it should print out a number somewhere in the neighborhood of 95,000. If this works, then you are done! If not, please let the TA or instructor know.

simHuman.py script

```
# This is Python

import stdpopsim
import numpy

species = stdpopsim.get_species("HomSap") # picking a species from the catalog
contig = species.get_contig("chr22")  # default is a flat genetic map

model = species.get_demographic_model("OutOfAfrica_3G09") # a model of  demographic history

samples = model.get_samples(10, 10, 10) # 10 individuals per population
engine = stdpopsim.get_engine("msprime") # selecting a simulation program
ts = engine.simulate(model, contig, samples) # running the simulation

print(ts.num_sites) # the simulation results are stored in the variable "ts".
#We are printing out the number of sites that were simulated
```

simHuman2.py script

```
# This is Python

import stdpopsim
import numpy

species = stdpopsim.get_species("HomSap")
model = species.get_demographic_model("Africa_1T12")
model.get_demography_debugger().print_history()

contig = species.get_contig("chr22", genetic_map="HapMapII_GRCh37")
samples = model.get_samples(200)
engine = stdpopsim.get_engine("msprime")
ts = engine.simulate(model, contig, samples)

print (ts.num_sites)
```