Programming Bootcamp 2015: Lab 7

Sarah Middleton

Last Updated: June 22, 2015

Name: ANSWER KEY

If you would like to receive points for your work, copy and paste your code into this document, save it, and send it to **sarahmid@mail.med.upenn.edu** with "Bootcamp Lab7" as the subject line **before *10PM* on Thursday, *5/25*** (NOTE DIFFERENT TIME/DAY!). This will be the final graded problem set! The winner(s) will be announced at the meeting on 5/26.

1 `os` and `glob` practice (6pts)

For this problem, use `horrible.fasta` and the fasta-reading function we created in lab6 (use the my solutions if you did not get to this yet).

(a) **(2 pt)** Write a script that takes two **command line arguments**: an input file name (assumed to be a fasta file) and an output folder name. Check if the input file exists; if it doesn't, print an error message. Then check if the output folder exists; if it doesn't, create it.

```
import sys
import os

# read in command line args
if len(sys.argv) == 3:
    inputFile = sys.argv[1]
```

(b) **(2 pt)** Change the script so that it also reads in the fasta file (if it exists) and prints each individual sequence to a separate file. The files should be named `<SEQID>.fasta`, where `<SEQID>` is the name of the sequence, and all files should be output to the specified output folder.

```
import sys
import os
import my_utils

# read in command line args
if len(sys.argv) == 3:
```

(c) **(2 pt)** Now use `glob` to get a list of all files in the output folder from part (b) that have a .fasta extension. For each file, print just the file name (not the file path) to the screen.

```
import sys
import glob
import os

# read in command line args
if len(sys.argv) == 2:
```

2 `optparse` practice (6pts)

Here you will create a program that generates a random dataset of sequences. This program will accept the following args/options:

OUTFILE (required) - file the sequences will be printed to
NUM_SEQS (optional; default = 1) - number of sequences to create
MIN_LEN (optional; default = 100) - minimum sequence length
MAX_LEN (optional; default = 100) - maximum sequence length
RNA (optional; a flag (true or false); default = False) - whether U or T is used
ID_PREFIX (optional; default = "seq") - Prefix to be added to beginning of each seq ID

The program should print the indicated number of randomly generated sequences, each with a length randomly chosen from the indicated range, to the indicated output file. If the –rna flag is used, use U's instead of T's. The output file should be in fasta format, and the sequence ID of each sequence should be the indicated prefix followed by a unique number (make sure each ID is unique; you can just use a counter for the number).

Run your script to create a large file of 10,000 random sequences of random length 500-5,000 nt. Call this file `fake.fasta` – it will be used in the next problem.

```
import optparse
import random
import my_utils

msg = "Usage: %prog OUTFILE [options]"
parser = optparse.OptionParser(usage=msg)
```

3 `time` practice (6pts)

(a) **(2 pt)** Is it faster to use the built-in function `str.replace()` or to loop through a string and replace characters manually? Compare the two by replacing all the T's in `fake.fasta` with U's using each method separately and comparing how long they take to run. Put both your code and your answer below.

```
import time
import my_utils

seqs = my_utils.read_fasta("fake.fasta")

# test manual replacement
```

(b) **(2 pt)** Is it faster to use the built-in function `str.count()` or to loop through a string and count characters manually? Compare the two by counting all the A's in `fake.fasta` using each method separately and comparing how long they take to run. Put both your code and answer below.

```
import time
import my_utils

seqs = my_utils.read_fasta("fake.fasta")

# test manual replacement
```

(c) **(2 pt)** Is it faster to use a list or a dictionary as a lookup table? Read in `fake.fasta`, ignoring everything but the header lines. Count the number of unique IDs (headers) using a dictionary and then a list, and compare how long each method takes to run. Put both your code and answer below.

```
import time
import my_utils

seqs = my_utils.read_fasta("fake.fasta")

# test list as lookup table
```

4 `subprocess` practice (6pts)

(a) **(2 pt)** Write a script that runs the command `ls -l`. Print the captured output to the screen.

```
# command-running function
# if verbose=True, prints output of the command to the screen
def run_command(command, verbose=False):
    import subprocess
    error = False
```

(b) **(4 pt)** Write a script that (1) creates a random dataset (by calling the script you created in question 2), (2) gets statistics about the sequences in the newly generated file (by calling the script you wrote in lab6, question 4b) and then (3) (optionally) launches an R script to graph the output (using the supplied graph_gc.r script). You will need R installed with ggplot to make the script work; if you don't have it then you can skip that last step for now. All scripts should be called using `subprocess.Popen()` (in other words, DO NOT copy/paste all the code from those other scripts into one script; the point is to practice using subprocess to launch existing scripts!).

The R script can be called as follows:

```
"Rscript graph_gc.r seq_info.txt"
```

where `seq_info.txt` can be replaced with the name of the output file of statistics (Length/GC) that you create in step(2).

```
import sys

# command-running function
# if verbose=True, prints output of the command to the screen
def run_command(command, verbose=False):
    import subprocess
```