Homework 1

Instructions. This homework is due at the beginning of lab, Wednesday, September 6th. You can edit this file and copy/paste into it by opening it in Libreoffice on Linux, PDFfiller Google App, Acrobat, and others (I can only confirm Libreoffice). Bioinformaticians are especially agile at using the internet to gather and share information, so if you do not remember or see the answer in class materials, feel free to use the web, but do not plagiarize it. Also, like anything you can obtain from the internet, information about bioinformatics may be wrong or only partially correct. Be sure to check the source of the information to help judge its quality.

1. The FASTA format is a common format for sequence data used in bioinformatics. Please answer these questions about this format. The first three are true/false questions.
   (a) (5 points) **F** FASTA files are binary files.
   (b) (5 points) **F** The sequence block should contain no more than 80 nucleotides per line.
   (c) (5 points) **T** The header or descriptor for each sequence can contain any printable ASCII character.

   > **Solution:** Truth is, there appears to be no technical specification for the FASTA format. Every program that handles FASTA files makes assumptions about the file format, but they differ in those assumptions. Many people recommend that sequence block lines should contain no more than 80 characters, but this is not required and was so stated twice in class. I stated in class that the descriptor line must start with a '>' and end with a newline. Most programs appear to assume the descriptor contains only printable ASCII, but I could not confirm whether it is an actual requirement. Thus, there is no right answer for Part c, so all answers earn 3 points, and thinkers could earn 5 points.

   (d) (5 points) The sequences in a FASTA file are encoded using IUPAC (now a link) symbols. In class we mentioned the 15mer (15 base pair oligomer) YYYYYYYYYYYYYYY. What nucleotides does the IUPAC code Y represent? C **or** T **or** U.

   > **Solution:** Four points if you forgot U or just put "pyrimidines". I didn't say "what *type* of nucleotides", rather "what nucleotides". Nitpicky perhaps, but a programmer needs to know the concrete list of possibilities, and bugs result if you are not precise.

   (e) The newline character(s) that occur(s) at the end of each line (except perhaps the last) in a text file are encoded differently on different operating systems. Sometimes these differences can confuse a novice bioinformatician. In the `hpc-class` directory `/ptmp/bcbio444/hw1`, you will find an executable program called `seq_name` that reads FASTA files and reports the name of the nth sequence, the first by default.

i. (10 points) To run this command, you need to log onto `hpc-class`, type the full path of the program and give it two arguments, the name of a text file and an (optional) index of the sequence name sought. For example, the following command outputs the name of the third sequence in the FASTA file `permissive.fa`.
`/ptmp/bcbio444/hw1/seq_name /ptmp/bcbio444/permissive.fa 3`
Use this command to report the name of the 153rd sequence in the FASTA file `/ptmp/bcbio444/hw1/REF_2010_env_DNA.fasta`. Record the output below:

Solution: Using \ to break long lines, here is the command I issued and the result:

```
[kdorman@hpc-class ~]$ /ptmp/bcbio444/hw1/seq_name \
/ptmp/bcbio444/hw1/REF_2010_env_DNA.fasta 153
' is the name of sequence 153 in file \
'/ptmp/bcbio444/hw1/REF_2010_env_DNA.fasta'.
```

Push your understanding of UNIX just a bit more! Everything you type at the command line consists of a *command* followed by options and arguments and possibly more commands after ';', '|', or '>'. So `/ptmp/bcbio444/hw1/seq_name` is the *command* in this example, and `/ptmp/bcbio444/hw1/REF_2010_env_DNA.fasta` and `153` are the arguments.

File `/ptmp/bcbio444/hw1/seq_name` is an *executable* file (see its permissions: `ls -l /ptmp/bcbio444/hw1/seq_name`) that you execute by typing its name at the command prompt. It is a binary file, not a text file, and cannot be read by (most) humans (try: `head /ptmp/bcbio444/hw1/seq_name`). Because this file was compiled on my Linux machine, some Linux machines, including `hpc-class`, can read and execute the file, but Windows and Macintosh machines cannot. This is why Linux is protected from many Windows viruses: the Linux machines simply cannot execute the virus software!

Most UNIX commands can be specified using their full pathname. For example instead of `ls`, you can type `/usr/bin/ls` for the same result:

```
[kdorman@hpc-class ~]$ ls f*
fastqc.sbatch  fastqc.sbatch.orig
[kdorman@hpc-class ~]$ /usr/bin/ls f*
fastqc.sbatch  fastqc.sbatch.orig
```

Aha, so the executable program `ls` is located in the directory `/usr/bin`? Let's check:

```
[kdorman@hpc-class hw1]$ ls -l /usr/bin/ls
-rwxr-xr-x. 1 root root 117656 Jun 30  2016 /usr/bin/ls
```

There it is, owned by root and with the executable permissions set for owner, group, and the rest of us, like you and me.

When you type a command at the UNIX prompt, the computer will look for that command in a set of predefined directories. You can see a list of those directories by displaying the *environment variable* `PATH`. For example, on my account, I get

```
[kdorman@hpc-class ~]$ echo $PATH
/usr/lib64/qt-3.3/bin:/usr/local/bin:/usr/bin:/usr/local/sbin\
:/usr/sbin:/home/kdorman/bin
```

which is a list of colon-separated directories where executable programs are sought. While the `ls` command *is* in one of these directories, the `seq_name` command provided to you for this problem *is not*. To run it, you must give the full pathname: `/ptmp/bcbio444/hw1/seq_name`. Otherwise, `hpc-class` will say:

```
[kdorman@hpc-class ~]$ seq_name \
/ptmp/bcbio444/hw1/REF_2010_env_DNA.fasta 153
-bash: seq_name: command not found
```

If you copy it into your local directory (or `cd /ptmp/bcbio444/hw1`), then you can access it via a relative pathname, although you must still identify the directory it is in (UNIX refuses to search your current directory for executables *unless* `.` is in `PATH`):

```
[kdorman@hpc-class ~]$ cp -r /ptmp/bcbio444/hw1 .
[kdorman@hpc-class ~]$ cd hw1
[kdorman@hpc-class hw1]$ ./seq_name
Could not open file '(null)'.  Please provide a filename as\
 first argument.
```

Since I provide no argument, it complains when I use it this way, but it doesn't complain about "command not found" anymore. This one does because the path is not specified:

```
[kdorman@hpc-class hw1]$ seq_name
-bash: seq_name: command not found
```

By the way, here is another way to access `seq_name` by a relative path name:

```
[kdorman@hpc-class hw1]$ pwd
/home/kdorman/hw1
[kdorman@hpc-class hw1]$ ../../../ptmp/bcbio444/hw1/seq_name
Could not open file '(null)'.  Please provide a filename as\
 first argument.
```

but now I am probably driving you nuts with details.

It does not work because the file was created on a Windows machine. You can detect this fact by opening the file in `vim`. When you first open it, read the last line where `vim` tells you the name of the file just opened and some other information about the file. You will see `[dos]` if the file was created on the Windows OS. (You will see `[noeol]` if there is no newline after the last line, which can also confuse bioinformaticians and bioinformatics software.)

ii. (10 points) To fix the file, copy it to your own directory, and learn how to run the command `dos2unix`, which converts the newlines in the file. Verify that the converted file works with the program `seq_name`. Report all the commands you ran to answer Parts i–ii (the command `history` may help you remember, but clean it up to just include the necessary commands).

Solution: I will explain each of the commands I ran and their purpose. First I make a local copy of the fasta file: I used tab completion because that is one awful name for a file!

```
[kdorman@hpc-class hw1]$ cp\
 /ptmp/bcbio444/hw1/REF_2010_env_DNA.fasta .
```

Then, I read about how to use the dos2unix command.

```
[kdorman@hpc-class hw1]$ man dos2unix
```

There are a lot of options, but by default it just converts the newlines on the file I provide as an argument, which is what I need.

```
[kdorman@hpc-class hw1]$ dos2unix REF_2010_env_DNA.fasta
dos2unix: converting file REF_2010_env_DNA.fasta to Unix format ...
```

It seems to have worked: no complaints. Let's try again. I ask `seq_name` to work on the local copy of the fasta file:

```
[kdorman@hpc-class hw1]$ /ptmp/bcbio444/hw1/seq_name\
 REF_2010_env_DNA.fasta 153
'Ref.46_BF.BR.07.07BR_FPS625.HM026456' is the name of sequence\
 153 in file 'REF_2010_env_DNA.fasta'.
```

So, I conclude the 153rd sequence is called '`Ref.46_BF.BR.07.07BR_FPS625.HM026456`'.

iii. (5 points) What one-liner UNIX command could you use to determine the name of the 153rd sequence.

Solution: I heard a collective, "What...?" What I thought was clear is that `seq_name` is *not* a UNIX command, but truth is the distinction between UNIX commands and other commands is not so clear. I was looking for a command like the ones we learned in Lab 1, but I have to give some credit for one-liners using `seq_name`, and bonus for those who remembered....

The answer without the sed would be fine, but I decided to remove the '>' since it is technically not part of the name.

```
[kdorman@hpc-class hw1]$ grep ">"\
```

```
 /ptmp/bcbio444/hw1/REF_2010_env_DNA.fasta | head -n 153\
 | tail -n 1 | sed 's/>//'
Ref.46_BF.BR.07.07BR_FPS625.HM026456
```

Some discussion on choice of programming languages. Since `seq_name` is the executable compiled from 67 lines of `C` source code, you can see why the UNIX command-line is so powerful (and why `C` is such a pain for input/ouput). Not only did 67 lines get reduced to one line of UNIX commands, but the UNIX commands are smart enough to handle the newline problem without complaining! Not to completely dis' `C`, my language of choice, though: you can't beat well-programmed `C` for speed and efficient memory use, well except with `Fortran`, the language of my teachers! Over time we are sacrificing time to run for time to program.

2. RNA-Seq is an extremely common procedure for generating transcriptomic data. This will not be the first time we talk about it. Read the Wikipedia entry (linked) through the Methods section. (For those of you interested, notice the section on Genomic Medicine.) For engineers who have not seen biology since high school, this page describes the basic biology in a very stepwise way, which should make you comfortable, but consider asking your biological colleagues for additional help. Also, you should know the The Central Dogma of information flow in molecular biology.

   In the following question, suppose you have a pair of tissue samples, normal and tumor, from 14 patients with small cell lung cancer (SCLC). You isolate the RNA from each tissue of each patient and perform RNA-Seq on the mRNA. For the purposes of this question, you are focused on a single gene that you think plays a role in SCLC. A bioinformatician prepares the data and gives you the number of RNA-Seq *reads* (fragments sequenced) of this gene in each of the samples. Please answer the following questions.

   (a) (10 points) Why is the RNA-Seq experiment an example of a random experiment?

   Solution: A random experiment is a "repeatable procedure whose outcome cannot be predicted beforehand." In this case, we cannot predict beforehand how many reads will come from the gene in question because their is randomness in the selection of 14 patients, there is randomness in the selection of cells to be included in the tumor and normal tissue samples, there is randomness in the operation of those cells themselves and how many mRNA copies they have of every gene, there is randomness in the mRNA that get isolated from the cells into the RNA-Seq library, and there is randomness in which mRNA get sequenced. There is even more randomness that was not covered in the readings, but each of those reads is read with errors that are random, and those errors affect whether the read can get assigned to the source gene, so there is even randomness introduced during the bioinformatics data processing. All

in all, these random processes interfere with our ability to predict the number of reads assigned to the gene of interest for each tissue sample. Thus, the RNA-Seq procedure described above is a random experiment, and one heck of a complicated random experiment!

Some confusion. Several of you thought that this was a random experiment because we cannot know before hand whether the gene is involved in cancer or not. Not true: in fact, we may have a very good idea whether a particular gene is involved in cancer or not. We may already have a load of evidence to suggest it is. This is not the source of randomness. The experimental procedure and life itself are the sources of randomness, and there would *still* be randomness even if we knew for sure that the gene was involved in cancer.

Some of you also simply stated that the result could not be known beforehand, but I wanted to know *why*.

(b) (10 points) For a single sample (tumor or normal) from a single patient, what is the sample space?

Solution: Since the outcome is *the number of reads* assigned to the gene of interest, the sample space is the counting numbers. Using typical mathematical notation, we have

$$\Omega = \{0\} \cup \mathbb{Z}^+,$$

where Ω is the sample space, $\cup$ is the union operator, and $\mathbb{Z}^+$ is the set of positive, non-zero integers.

Some confusion. Many of you thought the sample space was all possible collections of fragments of mRNA, which is more-or-less true if I had not focused your attention on the *number of RNA-Seq reads of this gene in each of the samples.* Also, since we do not observe the mRNA fragments directly, the sample space for you guys who didn't focus is all possible collections of *reads of* fragments of mRNA.

(c) Thirteen of the 14 patients show the same pattern: there are more reads for this gene in the tumor than the normal tissues.

i. If this gene has nothing to do with the cancer (this is a hypothesis, or model), what probability would you assign to the event that there are more reads in the tumor tissue for a single patient?

Solution: If this gene has nothing to do with cancer, then whether there is more or less mRNA of the gene of interest in the tumor tissue should be a completely random event. Let X_i be the number of reads in the tumor

tissue of the ith patient, and Y_i be the number of reads in the normal tissue of the ith patient, then under our model, the event of more reads in the tumor tissue for patient i is represented as $\{X_i > Y_i\}$ and its probability is

$$\Pr(X_i > Y_i) = 0.5.$$

Some confusion. Many of you *estimated* the probability from the data to be $\widehat{\Pr}(X_i > Y_i) = \frac{13}{14}$. I did not ask you to *estimate* the probability: I asked you to *hypothesize* a probability. Also, the gene may or may not have anything to do with cancer. If it has something to do with cancer, then this estimate is definitely *not* an estimate of the probability of the event when "the gene has nothing to do with the cancer." When the gene has something to do with cancer, it may be more or less transcribed in tumor vs. non-tumor cells. In fact, that signal, of higher or lower transcription, is exactly the signal we will detect to conclude that the gene *does* have something to do with cancer. Follow my logic?

ii. If patients are independent, what is the probability of the event that 13 of 14 patients have more reads in the tumor samples? Show your work.

Solution: Let Z be the number of patients for which event $\{X_i > Y_i\}$, $i = 1, 2, \ldots, 14$, is true. Because patients are assumed independent and the probability of success $\Pr(X_i > Y_i) = 0.5$ under our model, you may recognize Z to be a binomial random variable with $n = 14$ trials and probability of success 0.5. Therefore, under the no connection model and the assumption of independence of patients, we find

$$\Pr(Z = 13) = \binom{14}{13} 0.5^{14} = 8.5449219 \times 10^{-4},$$

given by the probability mass function of a Binomial$(14, 0.5)$ random variable.

You could also have gotten there as

$$
\begin{aligned}
\{Z = 13\} &= \{X_1 \le Y_1, X_2 > Y_2, \ldots, X_{14} > Y_{14}\} \\
&\quad \cup \{X_1 > Y_1, X_2 \le Y_2, \ldots, X_{14} > Y_{14}\} \\
&\quad \cup \cdots \cup \{X_1 > Y_1, X_2 > Y_2, \ldots, X_{13} > Y_{13}, X_{14} \le Y_{14}\} \\
\Pr(Z = 13) &= \Pr(X_1 \le Y_1, X_2 > Y_2, \ldots, X_{14} > Y_{14}) \\
&\quad + \Pr(X_1 > Y_1, X_2 \le Y_2, \ldots, X_{14} > Y_{14}) \\
&\quad + \cdots + \Pr(X_1 > Y_1, X_2 > Y_2, \ldots, X_{13} > Y_{13}, X_{14} \le Y_{14}) \\
&= \Pr(X_1 \le Y_1) \Pr(X_2 > Y_2) \cdots \Pr(X_{14} > Y_{14}) \\
&\quad + \Pr(X_1 > Y_1) \Pr(X_2 \le Y_2) \Pr(X_3 > Y_3) \cdots \Pr(X_{14} > Y_{14}) \\
&\quad + \cdots + \Pr(X_1 > Y_1) \cdots \Pr(X_{13} > Y_{13}) \Pr(X_{14} \le Y_{14}) \\
&= 0.5^{14} + 0.5^{14} + \cdots + 0.5^{14} = 14 \times 0.5^{14} = \binom{14}{13} 0.5^{14}.
\end{aligned}
$$

using the addition rule for probabilities of mutually exclusive events, the independence of the patients, and simplifying the algebra.

I hope you see the connection to the binomial random variable, but if the second set of derivations does not make sense to you, then please let me know. There is more review we need to do. Also, for those of you who did *see the connection to the binomial random variable, many of you need to brush up on your knowledge of the binomial random variable!*

3. The file `/ptmp/bcbio444/hw1/REF_2010_env_DNA.fasta` mentioned in question 1 has parsable sequence descriptors, where the accession number is the word fragment after the final dot. Read this page on `sed` regular expressions (regexps) and this page of regexp examples.

   (a) Use `sed` on the command line to extract the accession numbers from the file. One solution uses these regexp elements: `.`, `*`, `\(`*`regexp`*`\)`, `\`*`digit`*, `\`*`char`*. Record the command below.

Solution: This is the way I would do it:

```
[kdorman@hpc-class hw1]$ grep ">"\
 /ptmp/bcbio444/hw1/REF_2010_env_DNA.fasta\
 | sed 's/.*\.\(.*\)/\1/'
```

because I didn't know about the use of command option `-n` combined with flag `p`, but here is a pure `sed` way, thanks to Audrey, who I did not understand when she first told me:

```
[kdorman@hpc-class hw1]$ sed -n 's/.*\.\(.*\)/\1/p'\
 /ptmp/bcbio444/hw1/REF_2010_env_DNA.fasta
```

Learn more! There are some helpful ways to try out regexps on the web, for example `https://www.regexpal.com/`, though keep in mind that sed processes regexp's slightly differently from other regexp parsers. For example, sed needs you to escape the grouping operators: `\(` and `\)`, but most do not and can process (and) directly, as in `.*\.(.*)`. In any case, let me explain how the `sed` regexp `.*\.\(.*\)` is working, using the FASTA descriptor line: `>Ref.A1.AU.03.PS1044_Day0.DQ676872`

```
.*     : matches any character 0 or more times, so it matches
       : ">Ref.A1.Au.03.PS1004_Day0"
\.     : matches the last dot "." (by greedy rules -- see below)
\(.*\) : .* matches any character 0 or more times, so it matches
       : "DQ676872", and \(\) stores the resulting match in \1
```

Finally, the string stored in \1 is used to replace the whole line when there is a match. Notice, the sequence lines partially match regexp `.*\.\(.*\)`, the whole sequence gets slurped up by the `.*`, but because there is no dot '.', the match fails thereafter. With the `-n` option, these non-matching lines are ignored. With the `p` flag, the result of substitution on the matching lines *is* printed.

Why doesn't `.*` *match the entire line* `>Ref.A1.AU.03.PS1044_Day0.DQ676872`? The match is *greedy* but not rudely so. It takes all possible matches, but then gives back characters in order to facilitate matching the rest of the regexp, so initially `.*` matches `>Ref.A1.AU.03.PS1044_Day0.DQ676872`, but when the regexp then needs to match `\.`, it gives back `.DQ676872` so that `\.` can be matched. To see even more of this idea, type in (slowly) this regexp `.*\.PS(.*)` on the string `>Ref.A1.AU.03.PS1044_Day0.DQ676872` at page `https://www.regexpal.com/`. You will see it greedily grab, then progressively give back until the pattern is matched and `\1` holds `1044_Day0.DQ676872` (hover your mouse over the match in the Test String to see the value of group #1, *i.e.* `\1`). For more information about how the number of matches is determined, read `http://www.rexegg.com/regex-quantifiers.html`.

Here are some other valid solutions:

```
sed -n 's/.*\.\([a-zA-Z]\{1,2\}[0-9]*\)/\1/p'\
 /ptmp/bcbio444/hw1/REF_2010_env_DNA.fasta      # -SVW
cat /ptmp/bcbio444/hw1/REF_2010_env_DNA.fasta\
 | sed -n '/[A-Z]......[0-9]/p'\
 | sed -e 's/[[:punct:]]*.*[[:punct:]]//g'      # -MCo
sed -e '/>/!d; s/.*\.//g'\
 /ptmp/bcbio444/hw1/REF_2010_env_DNA.fasta      # -SVM and -PA and -MCh
sed -n 's/.*\.\([A-Z]\+[0-9]\+\)/\1/p'\
 /ptmp/bcbio444/hw1/REF_2010_env_DNA.fasta      # -SJ
```

Look at what you can do! It looks like random letters on the page, but it works.

Yes, I may ask for regexps on the exam that are like this homework question.

(b) Look up the accession numbers using Batch Entrez at NCBI and record the source species of these sequences. Are they all from the same species?

Solution: First, sorry for the multiple typos in the question (now corrected).

I ran the command from Part a and captured the output in a file, which I then pulled from the server.

```
[kdorman@hpc-class ~]$ sed -n 's/.*\.\(.*\)/\1/p'\
 /ptmp/bcbio444/hw1/REF_2010_env_DNA.fasta > acns.txt
[kdorman@hpc-class ~]$ exit
# back on my local computer, I secure copied the file from hpc-class
[kdorman@bhagi ~]$ scp hpc-class.its.iastate.edu:acns.txt .
Password:
acns.txt                                  100% 1564     1.2MB/s   00:00
```

(You could also have copy-and-pasted from the terminal to a local file.)

I googled Batch Entrez and found this website `https://www.ncbi.nlm.nih.gov/sites/batchentrez`. I uploaded the file `acns.txt` from my local home directory.

When Batch Entrez had finished, it said it found 158 sequences, which is the number of sequences in the fasta file (just double-checking–always good to do):

```
[kdorman@hpc-class ~]$ grep ">"\
 /ptmp/bcbio444/hw1/REF_2010_env_DNA.fasta | wc
    158     158    6361
```

Command `wc` counts the number of lines, words, and characters in a file or, in this case, in the content piped in. When I clicked on the link Batch Entrez provided (note: did not work in the Mozilla Firefox browser!) and asked it to display 200 entries per page, I could visually verify that all sequences were HIV-1.

More information about secure copy. David has now mentioned this in lab, but here it is written out. Windows users, see `https://www.it.iastate.edu/howtos/sftp` for a graphical `scp` client sanctioned by ISU; it connects much like `putty` but is used to drag and drop files between `hpc-class` (or any remote computer) and your local computer. For Mac and UNIX users, `scp` works much like `cp` except that it works across computer boundaries. To give an absolute address of a file that includes the computer, you need to specify the name of the remote computer, `hpc-class.its.iastate.edu`, followed by a colon (:) and then the full path and name of the file on the filesystem or the relative path with respect to your *home directory*. You do not need to name the computer you are on. Let's interpret the command I gave above (forgive the tiny font):

```
#                 \/-the command     the colon-\/       \/-the relative path on my local computer
[kdorman@bhagi ~]$ scp hpc-class.its.iastate.edu:acns.txt .
#                      ^name of remote computer  ^path of the remote file, relative to my home dir
```

Note, if my username was different on the remote server, then I would have to specify my remote username in the command

```
[kdorman@bhagi ~]$ scp kdorman@hpc-class.its.iastate.edu:acns.txt .
```

Some advice. Help me understand *how* you achieved an answer. A superb example was:

> 3(a) I tried a lot of different sed commands and I honestly couldn't get anything to print out.
>
> 3(b) One example accession number is 'DQ676872' (which I found by using seq_name), from the first sequence I added that string to a txt file and uploaded that to the nucleotide option in Batch Entrez.
>
> I got that it was from HIV, as was the second one I tested ('AB253421').

This earned more points than:

> 3(a) [No answer]
>
> 3(b) Yes, they're all from the HIV-1 virus.

I want to know whether you can think, and I have no idea whether there was any thinking at all involved in the second version.