Practical 4:
Somatic mutation calling and ESE motif analysis
Foundations of Computational Biology and Bioinformatics

Due before 11:59 PM, March 6, 2019. Email to fcbb2homework@gmail.com

Total points: 300

Problems overview

- `Q1:` Uniform distribution of null p-values (*50 pts*)
- `Q2:` Simplified GATK somatic mutation caller (*125 pts*)
- `Q3:` Modeling exonic splicing enhancer (ESE) binding sites in mRNA (*125 pts*).

Detailed Problem Descriptions

Q1: Uniform distribution of null p-values *(50 pts.)*.

P-values under the null are uniformly distributed. Show that it is true for the following case. Write an R script (binomial.R) that samples 1000 values from a binomial distribution with θ=(p=0.5,n=500), computes their p-values, and plots a histogram of the p-values.
Please set the pseudo random number generator seed to 101: set.seed(101)
HINT: use the CDF in R to compute the p-values.

```
Usage:  Rscript binomial.R
```

$$p(X) = \binom{n}{X} p^{X}(1-p)^{n-X}$$

What you will turn in:

- binomial.R
- A plot of the histogram

Q2: Simplified GATK mutation calling *(125 pts.)*.

- Download BAM files and BAM indices from class website:
  - cancer.bam
  - cancer.bam.bai
  - normal.bam
  - normal.bam.bai
- Download and install python-dev on your Ubuntu system
- From synaptic package manager download and install `python3-pip`
- From terminal command line enter `sudo pip3 install pysam`
  - If pip is having trouble installing pysam then uninstall your pip (`sudo apt-get remove python3-pip`). Then run the following commands, and try to install pysam again.
    * `wget https://bootstrap.pypa.io/get-pip.py`
    * `python3 get-pip.py`
- Check that you have the correct version of pysam `python3 -c 'import pysam; print(pysam.__version__)'`. Should return $\geq$ 0.10.0
- Look at example code to pull a pileup from a bamfile `http://pysam.readthedocs.org/en/latest/api.html`

Write a short program (mutation_caller.py) that does the following:

- Iterate over the positions in the normal.bam and cancer.bam files
- At each position, read in the pileup from normal.bam and cancer.bam
- If coverage is less than 20 in either the normal.bam or cancer.bam skip that position and report to standard output "Insufficient coverage at position XXX". *(30 pts.)*
- Otherwise, use the equations on slides 41–49 of Lecture 4 to:
  - Compute the likelihood of each of the ten possible diploid genotypes `{AA,CC,GG,TT,AC,AG,AT,CG,CT,GT}` in the normal pileup
  - For this exercise, set the `e` parameter to 0.1, rather than calculating it based on the base calling quality score at each position

- Identify the genotype $\hat{G}$ that maximizes this likelihood
- If the (log) likelihood $Log(P(D|\hat{G})) < -50$, skip the position and report to standard out "Position XXX has ambiguous genotype". *(30 pts.)*
- Using the pileup from the cancer.bam at the same position, compute the (log) likelihood of the bases in the cancer at that position,

$$Log(P(D_{tumor}|\hat{G})) \tag{1}$$

given $\hat{G}$ (the most likely normal genotype at that position).

- If $Log(P(D_{tumor}|\hat{G})) < -75$ print a statement to standard output that you have detected a candidate somatic mutation at that position: "Position XXX has a candidate somatic mutation (Log-likelihood=YYY)". Where YYY is the log-likelihood for the tumor data *(60 pts.)*
- HINT: somatic mutations are somewhat rare events so you should not expect to find too many positions that support a call of possible candidate somatic mutation.

- Your code must handle* input parameters specified in the assignment. We recommend that you use the `argparse` module to handle* command line options in conventional syntax (GNU/POSIX) *(5 pts.)*.
- Your python script starts with a Python docstring that identifies your JHED ID, the practical number, and a brief description of the file contents. *(15 pts.)*

Your assignment will be evaluated on code functionality and robustness: Your program must run on the Ubuntu environment we have specified for you without crashing on the command line.

We will be stress-testing this assignment by running your code with several different edge cases, unexpected inputs etc. When encountering an edge case, your code should catch the exception raised by python and exit gracefully with a friendly message.

```
Usage:  python3 mutation_caller.py -n normal.bam -c cancer.bam
```

Q3: Modeling exonic splicing enhancer (ESE) binding sites in mRNA (*125 pts*).

You will complete the task of developing, implementing and evaluating the performance of a zero-order Markov chain model to discriminate between experimentally determined ("trusted") ESE binding sites and decoys. You will use your `trainer.py` code and the same sequences from the Burge lab used in Practical 3 Exercise 1. Burge lab sequences are here: `http://genes.mit.edu/burgelab/rescue-ese/ESE.txt`.
First you will write a script called `permute_ese.py` For each ESE sequence, your script will read it in and permute the nucleotides. You will write the permuted sequences to an output file named permuted_ese.txt.

```
Usage:  python3 permute_ese.py -f input_seq_file.txt {o permuted_ese.txt
```

The program you will write for this homework `evaluator.py` will evaluate the model's performance on your decoy set.

Please modify `trainer.py` so that it does not output any row or column names, only numbers in tab delimited format, where the rows representing each nucleotide appear in the following order: A,G,C and T.

You will produce the permuted decoy sequences, and evaluate model performance on those sequences.
`evaluator.py` will take three inputs:

1. trained model
2. file of trusted ESEs (NOT from the training set. WHY?)
3. file of decoys

It will use the model to score the trusted ESEs and the decoys. The score is the log probability of the sequence, given the model $S = \sum_{i=1}^{l} \log_2 p\{x_i\}$. Finally, it will output text to standard output that quantifies your model's performance (described below). Command line options MUST be as follows.

```
Usage:  python3 evaluator.py -m output_model_file.txt -t ese_test.txt -d permuted_ese.txt
```

Output of evaluator.py: Your program will assess the performance of the model by its ability to discriminate between trusted ESEs and decoys. It will also provide false discovery rates. To do this, you will compute the total number of true positives (correctly classified trusted ESEs), false positives (incorrectly classified decoys), true negatives (correctly classified decoys) and false negatives (incorrectly classified trusted ESEs) at a series of score thresholds (use all possible scores as a threshold). You will figure out how to compute the false discovery rate at each threshold from these counts. Example of output with descriptive header and five rows:

```
#thresh TP FP TN FN   FDR
-12.204 88 99 1 0     ?
-11.961 88 98 2 0     ?
-11.883 88 97 3 0     ?
-11.794 88 96 4 0     ?
-11.706 88 95 5 0     ?
```

For your convenience, we have randomly split the experimentally validated ESEs that you used in the practical for Week 3 to provide a training set and a test set: 150 ESEs for training and 88 for test. Please find them on the class schedule page (ese_training.txt and ese_test.txt).

NOTE: create the decoy sequences based on the test set of ESE sequences (88 decoy saved as permuted_ese.txt).

Your assignment will be evaluated on code functionality and robustness: Your program must run on the Ubuntu environment we have specified for you without crashing on the command line.

- Your python script must start with a Python docstring that identifies your JHED ID, the practical number, and a brief description of the file contents. *(15 pts.)*
- The evaluator.py must correctly calculate the log probability of each RNA sequence, and the TP, FP, TN, FN, and FDR in the specified format *(90 pts.)*.
- Your code must handle* input parameters specified in the assignment. We recommend that you use the `argparse` module to handle* command line options in conventional syntax (GNU/POSIX) *(15 pts.)*.
- Your code must handle* the following situations: missing input files, and length of (any) input sequences inconsistent with your trained model *(15 pts.)*.
- The scores produced by the evaluator should be finite, so you need to deal with the case where $p\{x_i\} = 0$ (undefined log) is present in your model. You may use a pseudocount of 1 for this purpose *(15 pts.)*.

***Handle = process input and intercept unexpected inputs with Python's try/except mechanism.**

Submitting the Assignment

To submit, you will put all your scripts and all associated input files into a directory named p04_your_JHED_ID. To compress it into a single .tar.gz file use:

```
tar -zcvf p04_your_JHED_ID.tar.gz p04_your_JHED_ID
```

Email to fcbb2homework@gmail.com

Make sure to turn in your source code for the the python scripts and **all** associated input data to the scripts. If input data necessary to run your code is not included in your submission, your work will be returned to you by the TA and you will be assessed late penalties if applicable.