CS 273a Homework 3

Due by 11:59 PM PST 3/1/23. Late day policies are on the course website http://cs173a.stanford.edu.

Instructions on how to complete this problem set:

- Know the homework collaboration and honor code policies on the course website.
    - We report honor code violations. Please don't test us. The consequences are very unpleasant for those caught.
- Ask all questions by posting on Ed; we are happy to answer questions to better understand the question or the material, but we will not provide feedback whether a specific answer is correct.
- Read class materials (your class notes and TA primers) carefully. They may include hints.
- There is a coding aspect to this homework, as well as a written section. Submit both to Gradescope.
    - Your code is graded solely on autograder correctness; there are no hidden tests.
    - Your writeup will be graded by correctness and answers that demonstrate thoughtful engagement with the questions and course material.
- Data files for the homework can be found under DropBox: https://www.dropbox.com/sh/wgpsg4wjn4ij5ua/AAA8Fqcbt0RniYRmSDFYF3gka?dl=0
- All plots should be clearly labelled with axes labels.

I) Transcription factor binding site prediction

Description

Recall from the lecture that transcription factors (TF) are proteins that bind to DNA to promote/enhance/reduce (coding or non-coding gene) transcription. TFs typically recognize and bind specific DNA sequences (i.e., motifs) in the genome. The sites that contain these motifs are known as TF binding sites (TFBS).

Several experimental methods allow us to sample the genomic sequences a given transcription factor binds to. (If you are interested, the most common ones to read on are ChIP-seq, protein binding microarrays or PBM, and high throughput- or HT-SELEX). Because these measurements are done in one or a handful of cellular contexts, we want to learn from them a general model of what sequences the TF preferentially binds and then go predict these TFBS genome wide. And that's exactly what you'll do in this part.

We'll use a slightly simplified version of the most common (but by no means only) model for binding site prediction, known as a position weight matrix (PWM). PWMs contain information content scores built from position frequency matrix (PFM) and position probability matrix (PPM). We will only build PFM and PPM, but you can see how information content score is calculated in this nice review.

Walk through toy example

We will walk you through a toy example of binding site prediction. Read it carefully because you will then need to code it up! But first it's story time:

Consider newly discovered TF X. Your colleagues in the wet lab have discovered that X binds sequence motifs of length 5. They've also recovered 6 (we said toy) different sequences X likes to bind to (these have also been provided for you in the Python skeleton file):

```
5'-> ACCCG ->3'
     ACCCT
     ACGCA
     ATGCT
     CCGAT
     TCGCT
```

(Step 1) First we'll build a PFM for X based on these 6 sequences. This is what it looks like:

	0	1	2	3	4
A	4	0	0	1	1
T	1	1	0	0	4
C	1	5	2	5	0
G	0	0	4	0	1

The length of the PFM is the length of our motif (5 here). And the rows denote the 4 nucleotides. Entry (i,j) denotes the number of times nucleotide i appears in column j in the 6 sequences available to us. For example, 4 of the 6 sequences start with an A. That's why entry (A,0) contains 4, and that's also why each column sums to 6.

(Step 2) We plan, below, to convert each column to a probability distribution and score each possible 5-mer (i.e., subsequence length 5) in the genome according to the probability we will read off this matrix. Problem is our matrix contains 0s. So, we will predict, for example, that it is impossible to see X bound to any sequence that starts with G. Because we've never seen it do it. But, dude, we've only seen 6 sequences that it likes, how sure are we that there aren't many others, including ones that start with G? To avoid any and all zeros we will add a pseudo count of 0.5 to every cell in our matrix. In our example:

	0	1	2	3	4
A	4.5	0.5	0.5	1.5	1.5
T	1.5	1.5	0.5	0.5	4.5
C	1.5	5.5	2.5	5.5	0.5
G	0.5	0.5	4.5	0.5	1.5

Note that now every column sums to 6+4*0.5=8, and there are no zeroes.

(Step 3) Next we'll turn each column to a probability distribution by dividing each entry by the sum of its column (8 here):

	0	1	2	3	4
A	0.5625	0.0625	0.0625	0.1875	0.1875
T	0.1875	0.1875	0.0625	0.0625	0.5625
C	0.1875	0.6875	0.3125	0.6875	0.0625
G	0.0625	0.0625	0.5625	0.0625	0.1875

Now each column sums to 1, and we have ourselves a PPM.

(Step 4) The most likely sequence bound by TF X according to our PPM is the sequence where we choose the most likely (most often observed) nucleotide in each position. If two or more bases are tied, you can print either one of them. IUPAC code is the correct solution for ties, but we will skip it for simplicity. This sequence is called the consensus sequence. In our example, the consensus sequence is ACGCT.

(Step 5) To illustrate binding site prediction without going into more math we will now score any genomic sequence using the probabilities we read from our PPM. Because probabilities need to be multiplied they can get quite small (as small as $0.0625^5 \approx 9.5*10^{-7}$ here, but much smaller if we start with more than just 6 sequences). Because we fear underflow, we will score any possible sequence with the $\log_{10}$(probability) read from our PPM (while natural log and log base 10 behave similarly, we use $\log_{10}$ here for grading purposes). Logs are great because they can be summed. $\log_{10} 0.0625^5 = 5* \log_{10} 0.0625 \approx -6.02$, a much safer number to store. In summary then, for some genomic sequence stored in array seq[]:

$$Score[TFBS\ starting\ in\ position\ n] = \sum_{i=0}^{L-1} log_{10}(PPM[seq[n+i]][i])$$

Where seq[n+i] is the sequence's nucleotide in position n+i, and L is the length of the scored motif (5 in our example)

In our example if your toy genome seq=ACCGGTAC, then score[0] = $\log_{10}(0.5625)$ + $\log_{10}(0.6875)$ + $\log_{10}(0.3125)$ + $\log_{10}(0.0625)$ + $\log_{10}(0.1875) \approx -2.85$. (note: do not round!)

And if you scan the entire + strand of our toy genome (score[0], score[1], ..) you get:

Match seq	**Strand**	**Start in +**	**End in +**	**Score**
ACCGG	+	0	4	-2.849…
CCGGT	+	1	5	-2.594…
CGGTA	+	2	6	-4.112…
GGTAC	+	3	7	-5.543…

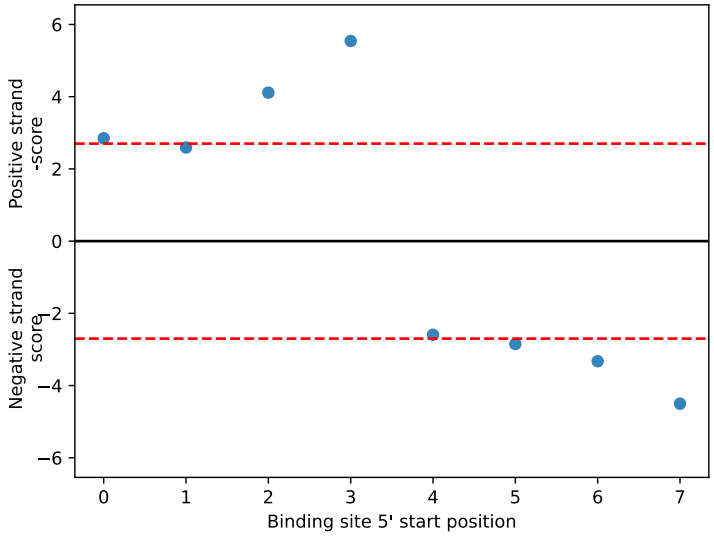
(Step 6) Alas, being the pro that you now are, you'll recall TF X can just as easily bind to the minus strand (sigh). Now, you can reverse complement seq and throw it at the same code, but we still want you to report the coordinates of each score on the + strand, like this for our example:

Match seq	Strand	Start in +	End in +	Score
GTACC	-	7	3	-4.502…
TACCG	-	6	2	-3.326…
ACCGG	-	5	1	-2.849…
CCGGT	-	4	0	-2.594…

(Step 7) Next, we'll give you a threshold score. Only predictions at or above this score should be pronounced a TFBS prediction. If in our toy example, the threshold we provide is -2.7, yielding only two binding site predictions to print out (note: the start and end positions are all 0 based and are inclusive). Round your score to the nearest hundredth when displaying values only (keep unrounded numbers for any other parts of the homework, including for plotting):

```
Match seq: CCGGT, strand=+, range=[1, 5], score=-2.59
Match seq: CCGGT, strand=-, range=[4, 0], score=-2.59
```

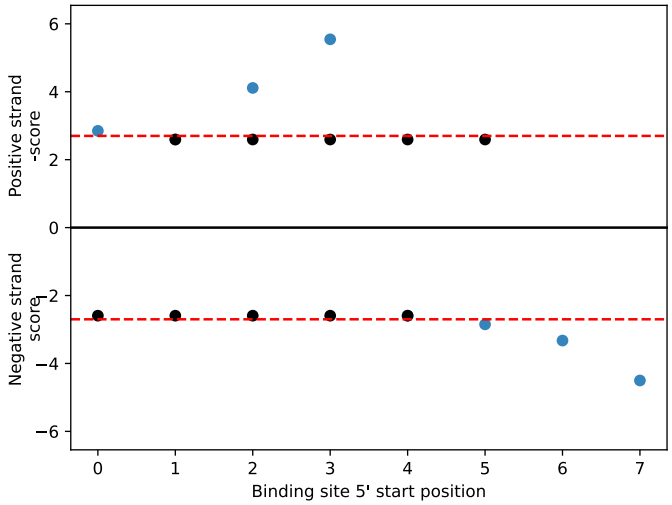
(Step 8) Now let's plot all of our predictions. On **one** graph plot: **a.** All + strand predictions as (start in + *I*, -score[*i*]), **note the minus**! **b.** All – strand predictions as (start in + *i*, score[*i*]) no minus here **c.** add dashed horizontal lines at y=threshold and y=–threshold. **d.** Add a black solid (divider) line y = 0 to separate + strand and – strand predictions. Have your graph look like the following, and label it like this (you can change the x and y ticks if your graph looks too busy):



Notice we played with spacing to get two y-axis labels for the price of one. Try to do the same; otherwise, make the best y-axis label you can – we won't grade you for the y-label in this plot.

(Step 9) To make the graph even prettier: for every position with a TFBS match above threshold don't just plot a single dot, extend the dots to all basepairs making up the TFBS. In our example (rounded here for visual but in real plotting plot unrounded values): Instead of plotting just (1, 2.59) for the only positive strand match above threshold, add dots at (2, 2.59), (3, 2.59), (4, 2.59) and (5, 2.59) – because your predicted match extends between positions 1 and 5. Likewise, for the only match above threshold on the negative strand plot (4, -2.59), (3, -2.59), (2, -2.59), (1, -2.59), (0, -2.59). Note that we are **not** simply adding more lines along the dashed red lines – the y-value in these extra dots is determined by the value of the corresponding score for the match.

This is what our toy graph looks like now (trust us, this will be pretty for longer seq):



(Step 10) Now we can see something we couldn't easily see before: Our + strand and – strand predictions actually overlap. Let's type what the overlap looks like:

```
motif position      0 1 2 3 4
seq + strand 5'-> A C C G G T ->3'
seq - strand 3'<- T G G C C A <-5'
motif position    4 3 2 1 0
```

Don't worry – we won't make you write code for this last bit, but we will ask you to type it in :)

Our walkthrough is now done! All you have to do is code it, plot it, and type it up:

Questions

Write code that takes as input a TF binding site file that has a TF binding site (of any length) in each line. Assume this file only contains uppercase "ATCG" characters, and all rows are of the same length. A TF binding site file from our toy example might look like:

ACCCG
ACCCT
ACGCA
ATGCT
CCGAT
TCGCT

And perform **(Step 1)**, **(Step 2)**, and **(Step 3)**. Run your code on the **MEF2A.sites.motif** file included on DropBox and display the PPM you get (below) from end of (Step 3) as an array of probabilities. For example, the PPM array from above would look like this:

```
[[0.56 0.06 0.06 0.19 0.19]
 [0.19 0.19 0.06 0.06 0.56]
 [0.19 0.69 0.31 0.69 0.06]
 [0.06 0.06 0.56 0.06 0.19]]
```

Note: probabilities should be printed rounded to the nearest hundredth (two decimals), but NOT rounded for calculation. Rows must be ordered (as above) as "ATCG" with no row or column headers.

1. Using the code you wrote **(in Step 4)**, what is the consensus sequence of MEF2A?

Write code that takes the PPM you computed in question 1, and scans it across a DNA sequence, provided in fasta format, and a binding site threshold; use these to perform **(Step 5-9)**. Assume all inputs are correct and the Fasta file always only has 1 sequence (i.e., 1 header) and only contain "ATCG" bases. Run your code on **DNA_sequence.fa** as your DNA sequence and **-7.5** as the binding site prediction threshold. Include the following:

2. Binding sites that have a match score higher than the threshold in the same format as **(Step 7).** Here are two real examples you should have found from running your code, include these two **and** the rest you found:

```
Match seq: TTCCATAAATAGTCC, strand=+, range=[67, 81], score=-4.52
Match seq: TTCCAAAAATAGCAT, strand=-, range=[250, 236], score=-3.27
```

3. Plot from **(Step 9)**

4. If all went well, the last plot should show you a single instance where an above threshold prediction on the plus strand overlaps an above threshold prediction on the minus strand. Describe this overlap in the same format we did with our toy example **(Step 10).**

   (Optional: stare at the **step 10** overlap between the two predictions. If we tried to build a 3-D model for it, most likely both predictions would overlap over much physical space, suggesting that – if our model is right – TF X can sometimes bind to +, sometimes to -, and sometimes to neither, but never to both simultaneously).

   (Optional 2: Stare at your predictions from **step 9**. How easy is it to prove that your predictions are correct? Well, you can try eg, ChIP-seq in some cell type, and see if TF X binds at these locations. But what if it doesn't? Maybe it will in some other context. This is no different from the challenge of proving gene predictions are real – if you find evidence in some context, great. But if you haven't, there's just too many contexts to ever *ever* sample them all – so how to tell when you're wrong? hmmm)

Ia) [Extra credit; optional] Transcription factor binding site prediction

This class assumes no background in Machine Learning (ML). If you have such background and are curious how ML features in Genomics research, you are welcome to give this question a go, purely for extra credit. You do not need to answer both questions for extra credit; you may attempt one or both questions (or none); they will be evaluated independently.

In the last section of this homework, we built a simple statistical model for detecting transcription factor binding sites. Specifically, we assumed that each base operated independently, and we used the known binding DNA sequences to infer what was the most likely observation at each position.

In more recent literature, the usage of these PFM matrices has become less popular, with machine learning approaches taking their place. In particular, a class of deep learning neural networks called Convolutional Neural Networks, or CNNs, has become widely adopted for predicting transcription factor binding sites. You may have heard of CNNs as being a popular and powerful class of deep learning architecture for image recognition and classification – turns out they are very powerful for analyzing genomic sequences as well!

1. One of the seminal works proposed for predicting transcription factor binding with a CNN was the Basset model (Kelley et al.) (link). Read through the paper and describe how the model works at a high level. In your summary, discuss how convolutional "kernels" learn motifs similar to the PFMs that we explicitly constructed above. Also discuss ways in which their approach is stronger, as well as ways in which their approach may be weaker than explicitly building PFMs.

   You do not need to discuss or fully understand the mathematics of deep learning to answer this – it is sufficient to describe their ideas in a general, conceptual sense.

2. These CNNs, like Basset, are typically trained using extremely large datasets and take researchers months to fully tune and train. To get a taste of what it is like to build a machine learning model for transcription factor binding prediction, we will use a smaller toy dataset. In the DropBox folder, you will see the file: tfbs_extra_credit_train.txt. This file contains two columns in a tab-separated format; the first column corresponds to a DNA sequence, and the second column corresponds to a "label" for that sequence, which in our case is a binary 0/1 label indicating whether that DNA sequence is a non-binder or binder for our given TF, respectively.

   Your task is to build a machine learning model to predict, given an input DNA sequence, whether the TF in this problem binds or not. You may use any language you want, as well as any external packages you want (e.g., PyTorch, numpy, scikit-learn, etc.). Once you have built your model, submit your predictions on the sequences in tfbs_extra_credit_test.txt as a **newline-delimited series of values between [0, 1]**. The autograder will compare your predictions to the known binding status and tell you your score (we will obviously not reveal the correct labels for the test set). You will

receive credit based on how well your model does compared to a random classifier.

Here are some useful tidbits that you might consider:
- To give a machine learning model a DNA sequence, you must first "featurize" it – represent a string in a numeric format that is consumable by a mathematical model. To do this, you might consider something like k-mer featurization, or a one-hot encoding.
- To avoid model overfit, we recommend that you create a validation split based on the training data we have given you and use it to evaluate how well you are generalizing.

II) Histone Modification Analysis

Description

Another exciting day in the lab! Your colleagues from the wet lab call. They just raised great antibodies for three histone modifications. They then grabbed some human lung tissue (don't ask) and did a histone ChIP-seq assay on each. They now have thousands and thousands of ChIP-seq peaks from each modification, and they wonder what do they mean? What are they marking?

To start answering their questions you will take two additional sets of peaks, also derived in lung: One for DNase-seq which marks lung active gene regulatory regions. And RNA-seq data which will allow you to highlight lung transcribing genes.

You will then overlap the active cis-regulatory regions set, and separately the actively transcribing genes with each of the three histone mark. But that's just a number. Is it surprisingly high? Low? You'll then randomly shuffle each histone mark set across the genome 100 times and count the intersection of the active cis and gene sets with these shuffles. (There are much more robust scale free ways of doing this, but this will suffice for us). Finally, you will ask does the real histone set overlap with the cis set *a lot more than its random shuffles would suggest*. If so, then this mark may be associated with active cis elements, and you can postulate additional experiments to prove it (but not here).

Questions

1. As promised, download three histone ChIP-seq sets in human lung tissue: H3K9me3, H3K27ac, and H3K36me3. (To enjoy this exercise more, try **not** to look at your class lecture notes or elsewhere to see what they do before finishing this homework. But the UNIX tutorial slides will be useful to refresh over!). Go to the project description pages below. You'll see the histone modification in "Target" and the tissue type in "Biosample summary". Go to the "File details" tab in the "Files" section and download **hg19** peaks in bed narrowPeak formats (not bigBed narrowPeak). Don't mix them with files/data from any other human assembly (like hg38) or pseudoreplicated peaks by mistake. Check the Accession ID we have given you to make sure you are downloading the correct file.

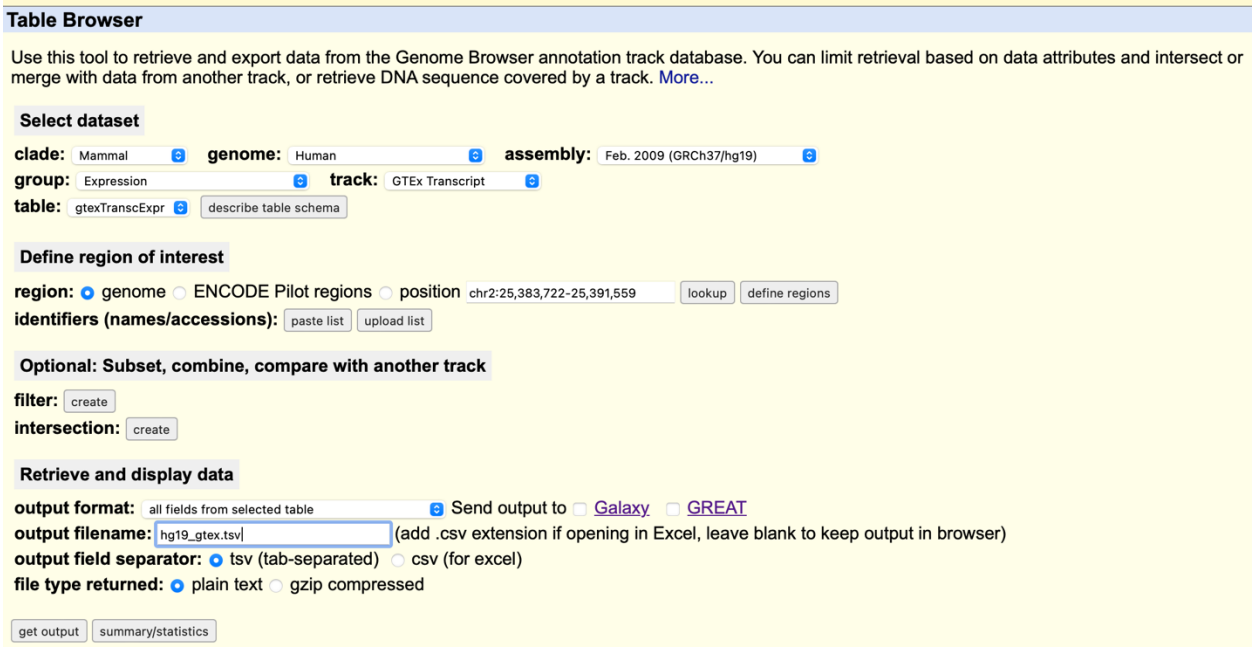
Histone modification	Project description page	Accession ID
H3K9me3	https://www.encodeproject.org/experiments/ENCSR728FLA/	ENCFF278IHQ
H3K27ac	https://www.encodeproject.org/experiments/ENCSR540ADS/	ENCFF305GIA
H3K36me3	https://www.encodeproject.org/experiments/ENCSR671NXL/	ENCFF441LCH

   Depending on which browser you use, you might be prompted for user/password. If so, leave everything empty and hit "Ok". If you see a broken website (happened to us on Firefox) try to search for the Accession ID then search for Download (or switch web browser).

   Unzip the files, only keep the first three columns (chrom, start, and end) and save the files as **hg19_H3K9me3_lung.bed** (line count: 248,593), **hg19_H3K27ac_lung.bed** (line count: 154,221), and **hg19_H3K36me3_lung.bed** (line count: 264,747), respectively. Report the last 5 lines/intervals from each file below, without doing any sort operations.

2. Download lung DNase-seq data from here: https://www.encodeproject.org/files/ENCFF547ZRE/ (project description page). Unzip the file, only keep the first three columns (chrom, start, and end) and save the file as **hg19_lung_dnase.bed**. This file will have 171,059 lines with no header. Report the last 5 lines/intervals from this file, without doing any sort or uniq operations.

3. Now we will download RNA-seq data to get all the transcripts expressed in human lung. From the table browser using the configuration below save the output to **hg19_gtex.tsv**.



Then filter the "chrom" field to only contain chr1-22, X, Y as you did in Homework 2 Parts II and III. The "expScores" field contain which tissue the transcript is found to be expressed in delimited by commas (you can click "describe table schema" for details). Since our histone modifications are measured in lung, we will take only the transcripts expressed in lung. Save the lines that contain an expression score greater than 0 (> 0) for lung which is the 36th value (or 35 if the index is zero based) then only output the first three columns of the file (chrome, chromStart, chromEnd); sort the file and keep only unique lines in the file hg19_lung_gtex.bed. For example, if you had the next 3 lines (lung score bolded and red):

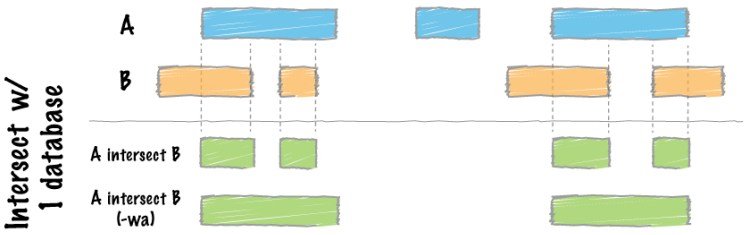
```
chr17 32582236    32583228    ENST00000624362   666   +     CCL2  53
      0.54,2.39,3.06,1.03,1.12,0.36,0.29,0.10,0.06,0.07,0.0,0.04,0.09,0.07,
0.06,0.09,0.06,0.05,0.15,0.08,0.48,0.0,2.30,0.16,0.12,1.77,1.31,0.95,0.29,0.9
3,3.57,1.01,0.49,0.75,0.15,2.35,0.19,0.03,0.75,0.11,0.78,0.59,0.77,0.33,0.29,
1.27,0.33,0.76,0.05,0.42,0.15,0.18,0.0    8656892223  60989
chr1  79862751    79864053    ENST00000572924   444   -     PCYT2 53
      0.25,0.27,0.14,0.19,0.22,0.21,0.21,0.22,0.29,0.30,2.07,2.39,0.56,0.32,
0.24,0.30,0.34,0.19,0.39,0.26,0.23,0.20,0.25,0.40,0.30,0.10,0.28,0.15,0.36,
0.11,0.20,0.09,0.08,0.22,0.76,0.34,0.18,0.05,0.42,0.45,0.11,0.36,0.47,0.16,
0.11,0.44,0.20,0.25,0.75,0.28,0.27,0.29,0.03    7102185149  61617
chr13 21714024    21714096    ENST00000516319   444   -     SNORD27     53
      0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0
,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,
0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,12.02,0.0,0.0,0.0,0.0 5191140594
      20084
```

Your output should include:

```
chr17  32582236    32583228
chr1   79862751    79864053
```

How many **unique** lines are in your final file?

4. Finally using the UCSC Genome Browser, download hg19 chromosome sizes for chr1-22, X, and Y similarly to Homework 2 (but for hg19). Save this as **hg19_chrom_sizes.tsv.** Now we want to count how many modified histone regions have any overlap with lung DNase-seq data or lung RNA-seq data. To do this we can use bedtools intersect with flag -wa where -a are the histone files (one at a time) and -b are the data set files (ditto), respectively. Note from the figure below that order matters with -wa:



Fill out the table below with these counts:

	Lung DNase-seq	Lung RNA-seq
H3K9me3		
H3K27ac		
H3K36me3		

5. To create a randomly shuffled regions use bedtools shuffle with histone modification files as the input file ("-i") and the chromosome sizes you made in question 4 as the genome file ("-g"). Repeat this 100 times for each histone modifications. To help us check your results, use the –seed flag, set it to 1 for the first shuffle, 2 for the second, etc, up to 100. This way your hundred iterations will be exactly like our 100 iterations. Now intersect each shuffle with each functional set using bedtools intersect as you did in question 5 and count the intersecting regions. At the end of this you should have 600 numbers (100 each for histone and functional set combination). Hint: to avoid doing this iterative process manually, check out the UNIX primer for how to write loops in BASH and how to use screen or tmux to run programs that take a while in the background. Fill the table below (for visibility round to the nearest hundredth but don't round for calculating question 7):

	Lung DNase-seq		Lung RNA-seq	
	Mean	Standard deviation	Mean	Standard deviation
H3K9me3				
H3K27ac				
H3K36me3				

6. Now calculate the fold and the z-score using the tables you made in questions 5 and 6. Fold is calculated by the (empirical count)/(mean of the null) and the z-score is calculated by (empirical count – mean of the null)/(standard deviation of the null) and fill the table below (round to the nearest hundredth).

	Lung DNase-seq		Lung RNA-seq	
	Fold	Z-score	Fold	Z-score
H3K9me3				
H3K27ac				
H3K36me3				

7. You're *finally* ready for the fun! For simplicity assume any result with a z-score under 100 to be statistically non-significant (i.e., ignore it). Sort the remaining (significant) overlaps by their fold (magnitude of the phenomena, from high to low).
    a. Which mark correlates most strongly and with what? Read the literature on this mark and try to comment on our match.
    b. What's the second strongest correlation (by fold)?
    c. What's the third strongest? What's surprising there?
    d. Is there any mark for which you couldn't say anything definitive?

    (Think: If so, can you imagine other lung functional sets to correlate with? Maybe this mark's true correlation lies elsewhere. This question is just for you to think about, doesn't require an answer.)