Programming for Bioinformatics | BIOL7200

Genomic Intervals

Assume that the user gives you correct inputs all the time. **Your script will be graded on the output produced and not on how the errors are handled.**

There is no public CI.

For this assignment, the module list is "`sys`", "`re`", "`os`", and "`argparse`". The grading CI will not install missing modules. Do not use `input()` for any input.

Instructions for submission

- **This assignment is due Monday, November 26, 2018 at 11:59pm. Late submissions will not be graded**
- Your code must be available on GitLab at the above time to be graded
- Name your script as **overlapBed.py**
- Your code should run as `./overlapBed.py -i1 <Input file 1> -i2 <Input file 2> -m <INT: minimal overlap> [-j Optional: join the two entries] –o <Output file>`
- **DO NOT HARDCODE** any file name!
- Use `#!/usr/bin/env python` as your shebang
- **Your script should finish within 30min, partial credit will be awarded up to 3 hours of run time.** Bonus: If your code finishes the overlap on the provided files and produces the correct output within 3 min, we will award you an additional 20% bonus points.

This is a very common task in genome analysis. Often we want to find functional elements that overlap with other elements, *e.g.* transcription start sites contained in transposable elements, predicted transcription factor binding sites within DNase hypersensitivity sites, or even some set of genes or horizontally transferred regions. Maybe you found some peaks in your ChIP-seq data and you want to see if those overlap with known enhancers, or perhaps you have a set of SNPs and you would like to know what genes they fall in. If you do not know what those words mean, go look them up!

Comparing ALL coordinates of the first set versus ALL coordinates of the second set will work for small sets only. That method scales with the multiplication of the sizes of the two sets. If you have 10^6 members in each set (not at all unrealistic) do you want to make 10^{12} comparisons? You need to find a better way.

What your script should do:

1. Take in two files that contain sets of coordinates in BED format. You can assume that the contents of both files are sorted by chromosome, start and stop. Having the data sorted allows you to greatly speed up the process of the overlap.
   Definitions for BED file format can be found: https://genome.ucsc.edu/FAQ/FAQformat#format1
2. Take in an option for the minimum percent overlap, *i.e.* percent of bases of any given member of the first set that must be in a member of the second set to be counted as overlapping.
3. Print to a file the members of the first set which overlap with the second set and meet the minimum overlap and other conditions specified. Each overlap should only be printed once in this manner.

4. Allow an option to print both the member of the first set and the member of the second set that it overlaps with on the same line, in effect 'joining' the rows.

Input files: Two BED files (TE.bed, Intron.bed) can be found at http://jordan.biology.gatech.edu/biol7200/

Examples:

Overlap of the following two sets with a minimum of 100% overlap would yield one row.

Set 1:

```
chr1  1500  1750  1     0     +
chr1  4500  5000  2     0     +
chr1  8500  9500  3     0     +
```

Set 2:

```
chr1  1000  2000  a     0     -
chr1  3000  4000  b     0     -
chr1  9000  11000 c     0     -
```

Output:

```
chr1  1500  1750  1     0     +
```

If the joining option (from #4) was given:

```
chr1  1500  1750  1     0     +     chr1  1000  2000  a     0     -
```

If the minimum overlap was 50% instead of 100% you would get two rows:

```
chr1  1500  1750  1     0     +
chr1  8500  9500  3     0     +
```

Notes:

The overlap of two coordinates can be calculated as

```
MIN(stop_one, stop_two) – MAX(start_one, start_two)
```

Anything greater than 0 is the number of overlapping bases, otherwise it indicates no overlap.

Deliverables:

- Code: `overlapBed.py`
- A README file: this file should contain the number of lines in your output when you run your script with: `./overlapBed.py -i1 TE.bed -i2 Intron.bed -m 80 -o output`

Syntax: `./overlapBed.py -i1 <Input file 1> -i2 <Input file 2> -m <INT: minimal overlap> [-j Optional: join the two entries] –o <Output file>`

Example command: `./overlapBed.py -i1 TE.bed -i2 Intron.bed -m 50 -j -o testOutput`

This command finds overlaps between TE.bed and Intron.bed, prints out the pairs of overlapping entries from both bed files that have minimum percent overlap of 50%.

Additional note:

- This is a great problem that demonstrates how a simple solution can often get you a pretty fast code. We advise you to sketch out a solution (*i.e.*, pseudocode) before you start implementing the actual code. Some of the best solutions we have seen in the past used a nested loop and a handful of `if` statements.