CGMethods homework
due May 14, 2018

Download and unpack the homework.tar.gz file from JHBox.

Please provide all scripts in plain text documents that you create in nano or a similar text-only editor. You can put multiple unix commands in one file but please keep each python program in its own file, with a .py suffix. Please name your files and comment your commands using the question name and letter. If you do this correctly, your unix commands should run as a shell script, though we didn't really practice that so that's not a component of your grade.

example:
###question 1a
unix command that does something

###question 1b
another command

Each question is worth 5 points.

1. Parsing and overlapping tabular data
The file, scer_gene.md, lists annotated genes and other features in the *Saccharomyces cerevisiae* genome.

a. Write a single unix command (with pipes if needed) to print the number of genes that are on the plus strand and the number that are on the minus strand. Note that the file contains annotated cds features as well as gene features. Submit your unix command.

You're going to overlap the gene features with the nucleosome free regions that were annotated by Lee et al. in a hallmark paper (Lee W, et al A high-resolution atlas of nucleosome occupancy in yeast. Nat Genet 2007). Notice that yeast annotations traditionally use Roman numerals for chromosome names, so you'll need to translate the chromosome names.

b. Fix the numbering problem, using a python program that takes the scer_gene.md file and outputs a nearly identical file that has numbers instead of Roman numerals. (you'll need to use a dictionary) Name your new file scer_gene_fixed.md. Your program can either write to standard output and redirect it into a file, or your program can create a file directly. Note: since each line ends with a newline character and the print command, by default, prints a newline, you'll end up with a double-spaced file. Figure out how to strip the newline character off of a line as you read it. Submit the python program, not any output.

c. Write a Python program that reads your fixed file, and for every gene (not cds) on chromsome 11, use a function to find the gene's start position (this depends on which strand the gene is annotated), and finally print out just the chromosome, start position, and the name of each gene. Submit the python program, not any output.

d. Using a piped unix command (you may need to use awk), print the chromsome, gene start position, and gene name for every gene on the plus strand of chromosome 11, in reverse order by chromosome coordinate. Submit the unix command.

e. Now do the overlaps. Using GenomicRanges in R, overlap the nucleosome free regions with all genes. The nucleosome free regions were determined at steady state. How many genes overlap nucleosome free regions?

f. I might expect that tRNAs would be nucleosome free, as they are needed in most biological states (though tRNA levels are regulated more tightly than you'd expect). The tRNAs are easy to pick out, as they are the only lines in scer_gene_fixed.md that contain parentheses. Use grep to create a file with only tRNAs, and a file with no tRNAs, starting from scer_gene_fixed.md (use man grep to figure out how to grep for "not" matches). In R, overlap each set of genes with the nucleosome free regions. How many tRNAs do you have, and how many of them overlap?

g. How many genes do you have and how many overlap nucleosome free regions?

h. You now have two gene sets for a competitive gene set enrichment test. Is there a significant difference in the fraction overlapping each gene set?

2. Whole genome sequencing alignment

Now look at WJH_1.fastq.gz and WJH_2.fastq.gz. These are subsetted whole genome sequencing files.

a. How big are the gzipped files, in MB? If your OS has unzipped the files when you downloaded them, then gzip the files again and record the sizes. (I suggest you turn the automatic uncompression feature off in your browser preferences, as it can inflate files to spectacular sizes)

b. Unzip the files. How big are the unzipped fastq files, in MB?

c. Write a single unix command, using head, that displays the first 4 lines of both fastq files. Submit this command (not the output)

Uncompress chr19.fa.gz.

d. Open an editor and write a shell script to make bowtie2 indices of the chr19.fa file. Make sure that the script is annotated well enough that someone else could run it and understand what it does. Run the script to create the indices. (don't submit the indices). Save the script and submit it.

e. Write a shell script to align the WJH files (as paired end sequences) to the chr19 sequence. Do the alignment once in default mode, and then again, using the bowtie2 options to trim 30bp off the 3' end of each read. The bowtie2 help will tell you how to do this. It's a common option, as the sequence at the ends of reads is often lower quality than the sequence at the beginning of each read. Make sure the script to do both alignments is well annotated, and submit it.

f. bowtie2 gives you a summary of the alignment. What percentage of reads aligned with the default parameters? with the trimming? (don't submit the output files).

g. Write a shell script that takes the sam files from either of the bowtie2 alignments and creates viewable bam and bai files (using samtools). Annotate your script and submit it. (don't submit the bam or bai files)

Load the bam file into IGV, and use igvtools to create a tdf that you can add to the view (this gives a much better overview of the coverage). Load the repetitive element annotations from the remote server, using File->load from server. The SINE, LINE, and low complexity tracks should be enough. You may have to resize the window to make sure that the tracks appear. You'll notice that a lot of the alignment density occurs at these repeats.

h. Use IGV's File->Save Image tool to make a .png from any interesting portion of chromosome 19. Submit your image.

i. Use cut and grep to make a summary file from the bowtie2 output (sam file). The file should have, for each aligned read, the chromosome and coordinate for the alignment. Save your command. (hint: you aligned to a sequence named 'chr19'). Don't submit the output file.

j. Open R. Read in the parsed file from the previous step, with read.table. Make a histogram of the start site coordinates and save the histogram to a pdf. Make sure that the histogram has an informative title and axes (use ?hist to find out how to do this), and save and submit your R commands and the histogram.