Storey Lab Notebook

Peter Edge

July 3, 2014

June 9, 2014

Today I did not yet have Emily's code available to study or access to the cetus data, so instead I focused on preparing my workstation and reading literature related to my project. I focused on Skelly et al (http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3202289/) because it is perhaps the most relevant to what I am doing.

June 10, 2014

Today I briefly introduced myself to Emily's snp_pipepline code before spending most of my day at lab safety training.

June 11, 2014

Today I focused on reading Emily's code in the snp_pipepline.py file, and familiarizing myself with the libraries, tools, and paradigms it uses. I read through the documentation for ruffus (http://www.ruffus.org.uk) as well as reading up a bit on python iterators.

June 12, 2014

Today I finished studying the details of Emily's snp_pipepline and began trying to use pysam myself for the manipulation of sam files. I created directories RM_bowtie_test and S288C_bowtie_test in SeqSorter/sample_data and used them to align the test reads to the RM and S288C genomes using bowtie:

```
cd git/SeqSorter/sample_data
mkdir RM_bowtie_test S288C_bowtie_test
bowtie2-build S288C_reference_genome_R64-1-1_20110203/S288C_reference_sequence_R64
    -1-1_20110203.fsa S288C_reference_genome_R64-1-1_20110203
bowtie2-build RM11_1A/assembly/genome.fa RM_bowtie_test/RM_index
bowtie2-build S288C_reference_genome_R64-1-1_20110203/
    S288C_reference_sequence_R64-1-1_20110203.fsa S288C_bowtie_test/BY_index
bowtie2 -x RM_bowtie_test/RM_index -U E2a0_sample.fastq
    -S RM_bowtie_test/RM_bowtie_out.sam
bowtie2 -x S288C_bowtie_test/BY_index -U E2a0_sample.fastq
    -S S288C_bowtie_test/BY_bowtie_out.sam
```

I then started experimenting with pysam to understand how the library is used to access data from the alignment files.

June 13, 2014

Today I narrowed my focus to the method compare_mappings(infiles, outfile), since this is where the initial focus of my project will be. I familiarized myself with using pysam for navigating SAM files, using the alignment SAMs from the test dataset (created yesterday).

June 16 - 20, 2014

This week was spent coding, testing, and analyzing the initial prototype of SeqSort.py. All notes relevant to this process are in the messages appended to the github commits page and the issues tracker, while notes that pertain to code only are in the comments of SeqSort.py and testSeqSort.py:

`https://github.com/dgrtwo/SeqSorter`

Equation for the calculation of Pr(read | BY):

$$Pr(read|BY) = \prod_{i=1}^{L} \begin{cases} 1 - Pr(\text{miscall}), & \text{if } R_L \text{ matches } BY_L \\ \frac{Pr(\text{miscall})}{3}, & \text{otherwise} \end{cases}$$

Pr(read | RM) is calculated in the same fashion. Pr(BY | read) is calculated as follows, via Baiyes Rule with the law of total probability, (priors of 0.5 are assumed for BY and RM and divide out):

$$Pr(BY|read) = \frac{Pr(read|BY)}{Pr(read|BY) + Pr(read|RM)}$$

SortSeq computes the probability of BY and probability of RM for reads that bowtie maps to both.

June 23-27, 2014

This week David outlined for me a way to add a layer of sophistication for SortSeq, at least for our current experiment: handling the problem of read quality scores being an inaccurate representation of the probability of a mismatch. I will start by accumulating information about the proportion of bases for a given quality score in the RNAseq reads that are likely to be mismatches, and then "correcting" the values associated with various quality scores.

The code for this analysis is in the file SeqSorter/estimateErrorFreq.py, and it walks through pairs of aligned_reads (RNA read aligned to BY, and same RNA read aligned to RM) in the two mappings, and builds a nested dictionary of this structure:

```
result[(BY chromosome, RM chromosome, BY position, RM position, consensus base)]
[RNA base][quality score] = count
```

That is, for reads that map to both chromosomes, if both genomes have the same base in that position, then we keep track of the specific spots the base mapped to in both BY and RM, what the base was in the RNA read, and what its quality score was. From this information, we build a set of 'likely consensus mappings' situations where both genomes agree on a base, and most aligned RNA bases agree on the base. Then, any base that disagrees with this likely consensus can be defined as random error. Then, we can tally the total amount of random errors that were associated with a base of a given quality, and therefore compute the likelihood of error from a given quality score empirically.