Below are four possible projects. The rules are: don't cheat, anything else is fine. Implement in whatever language you choose. Bonus points for some weird language (Smalltalk? FORTRAN77?).

Your code should work, and please provide everything necessary to run it.

For your report, write up what you did, any potential problems you had, and anything you found you found interesting/cool about implementation.

Tips: start early, ask questions, and talk to each other.

1 SNP caller implementation

Given some DNA-seq data in BAM format, find the posterior probability of the putative genotypes at several positions in the genome.

- **Given data:** Some DNA-seq data is given in BAM format which means it has already been aligned to the genome. You can read about the BAM format here: https://samtools.github.io/hts-specs/SAMv1.pdf. In short, you need to find the observations that are relevant for each putative SNP and the corresponding error.
- **Given data:** There are some putative SNPs in `putative_snps.tsv` with reference to the genome `chr21.fasta`. Note, some of the putative SNPs have `NA` for the prior probabilities. Deal with this however you choose, just explain how you did it.
- (Extra credit: 5 points) Align the data using an existing tool and talk about the parameters you chose and why. To align the data you can make a FASTQ file from the BAM file. Tools you might look at are `bwa` or `bowtie2`.
- Under the model of your choosing, estimate the posterior probability of each genotype. *You must write up the model and implement it.*

Note: there are number of libraries that might help getting reads that overlap the putative SNPs. Google is your friend: "how do I find reads that overlap a position python". You might also find the BEDTOOLs package helpful.

Deliverables

- Math for a model.
- A working implementation that may be executed something like this: `snpcaller reads.bam metadata.tsv`
- metadata.tsv is a file that contains the positions you are interested in testing relative to the reference genome. You can use whichever format you like, but here's one possibility:

chromosome	position	reference_base
1	2301	A

- output of your choosing, which minimally states:

chromosome	position	putative_genotype	posterior_probability	n_reads
1	2301	AA	0.97	7

- A write up of your approach, along with the results.

2 Pseudoalignment implementation

[10 points extra credit simply for attempting this one]

Given some RNA-seq data in FASTA format, find the vector of equivalence class counts. Your implementation should in the least do the following:

- Given a gene annotation, an index should be produced which can be read in by your pseudoalignment procedure.
- Your pseudoalignment procedure should take in (1) the previously mentioned index, (2) RNA-seq data and output equivalence class counts, and (3) a k-mer length.

Please provide equivalence class counts in the following format:

counts	number of items in equivalence class	isoforms in equivalence class
30	0	NA
5	1	ENST000003679
10379	2	ENST000003679,ENST000009216

You may implement either the naive version presented in lecture using the hash table or the skipping (colored de Bruijn graph) version[1]. If you are feeling extra weird, you can also implement it using a suffix array.

Note: there are some annoyances we didn't talk about in class. The most immediate is dealing with the reverse complement. Basically, you have to map both the forward and the reverse strand. The other is that you have to keep track of *new* equivalence classes that arise when taking the intersection in novel ways.

Finally, provide some basic statistics about the size of the equivalence classes and how much data is mapping to them. This sort of summary can be provided in a figure/plot.

An additional tip: you can take the aligned data and compute equivalence classes directly from it to check your answer.

3 RNA-seq Expectation Maximization implementation

[10 points extra credit simply for attempting this one]

Warning: the most difficult of all.

Given some RNA-seq alignments in BAM format and a isoform annotation table, find the relevant equivalence classes, then estimate relative abundance of the isoforms.

You can learn about the BAM format here. You can use whatever library you choose to parse the reads. In python, PySam is pretty good and in R Rsamtools is decent. If you are

using C/C++ there are official libraries as well (the actual BAM format is in C). If you are a masochist, there is probably a Java implementation floating around.

You may implement the RSEM version[2] which has a likelihood in read-space or the IsoEM/kallisto[1] version which is in equivalence class space. The kallisto version is a bit more work because it will require computing equivalence classes from the reads, whereas the RSEM version can be done directly from the reads. The kallisto version will be drastically faster than the RSEM version because of the reduced complexity of the likelihood function.

Please provide the transcript abundance table outputting the following format:

transcript	length	effective_length	expected_counts	tpm
ENST000003679	1219	1019	517	2.93

The reads provided are single-end reads, so you will have to assume a fragment length distribution. Treat it as $N(200, 10)$ (with appropriate discretization/normalizations).

In your report you might plot some metric against the various iterations of the EM. Take your pick.

You can run *kallisto* or *RSEM* or other quantification tools to check your answer against.

Extra extra credit (10 points): output the final posterior probability of each read mapping to a particular transcript.

4 Your own project

Please propose your own project in an email with at least a paragraph long description. I reserve the right to reject your proposal or amended it, but if you submit it earlier, you can have a chance to revise it.

References

[1] Bray, N. L., Pimentel, H., Melsted, P., and Pachter, L. Near-optimal probabilistic rna-seq quantification. *Nature biotechnology 34*, 5 (2016), 525–527.

[2] Li, B., Ruotti, V., Stewart, R. M., Thomson, J. A., and Dewey, C. N. Rna-seq gene expression estimation with read mapping uncertainty. *Bioinformatics 26*, 4 (2010), 493–500.