Joshua Burkhart
12/13/2015
BMI 650

Final Project Writeup

Aligner Choice

Considering our assignment involves RNA-Seq alignment, an evaluation paper[1] was reviewed before choosing STAR[2]. STAR was commended in the evaluation[1] for its execution speed, far lower than other RNA-Seq aligners evaluated; its ability to use annotation information to detect splice sites; and its high accuracy.

The package was built from source on church.ohsu.edu. As church.ohsu.edu runs g++ 4.6.3 (which doesn't support c++11), the `STAR/source/Makefile` had to be modified; `-std=c++11` was replaced with `-std=c++0x`. The reported version is STAR_2.5.0b_modified.

The chromosome name in the reference file provided for this assignment (GRCm38_chr1.fa) didn't match the provided GTF file (Mus_musculus.GRCm38.gtf) and had to be edited prior to building the STAR reference index.

Naïve Approach

Summary

This approach uses combinatorics and information theory to estimate the likelihood of a read aligning to a particular position in the reference.

Assuming a read length of 100 and a uniform random (UR) distribution of nucleotides A, T, C, and G, in our reference sequence, a read will align without mismatches to a reference with a probability of

$$.25^{100}$$

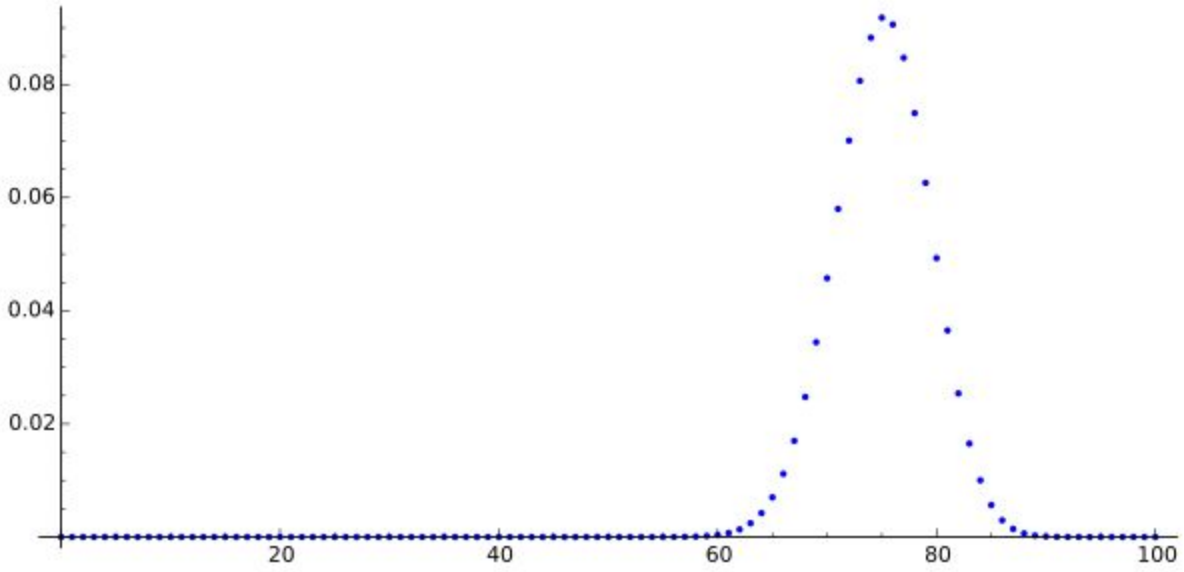
This is an acceptably low probability. Using a *good* limit for acceptable probability, such as .005, will enable us to allow for many more mismatches. First, lets see how to calculate the probability that a read will align with exactly one mismatch.

$$.25^{99} \times .75^{1} \times \binom{100}{1}$$

Here, we're multiplying the probability of 99 nucleotides matching by the probability of one nucleotide not matching by the number of ways one mismatched nucleotide can be chosen from a read length of 100. In general, we can use this form to derive a probability mass function (pmf) as below.

$$P(match)^{l-m} \times (1 - P(match))^{m} \times \binom{l}{m}$$

Where l = the read length, m is the number of mismatches, and $m \leq l$. The pmf for $P(match)$ = .25 and l = 100 is plotted[a] below. m is on the horizontal axis and probability is on the vertical axis.

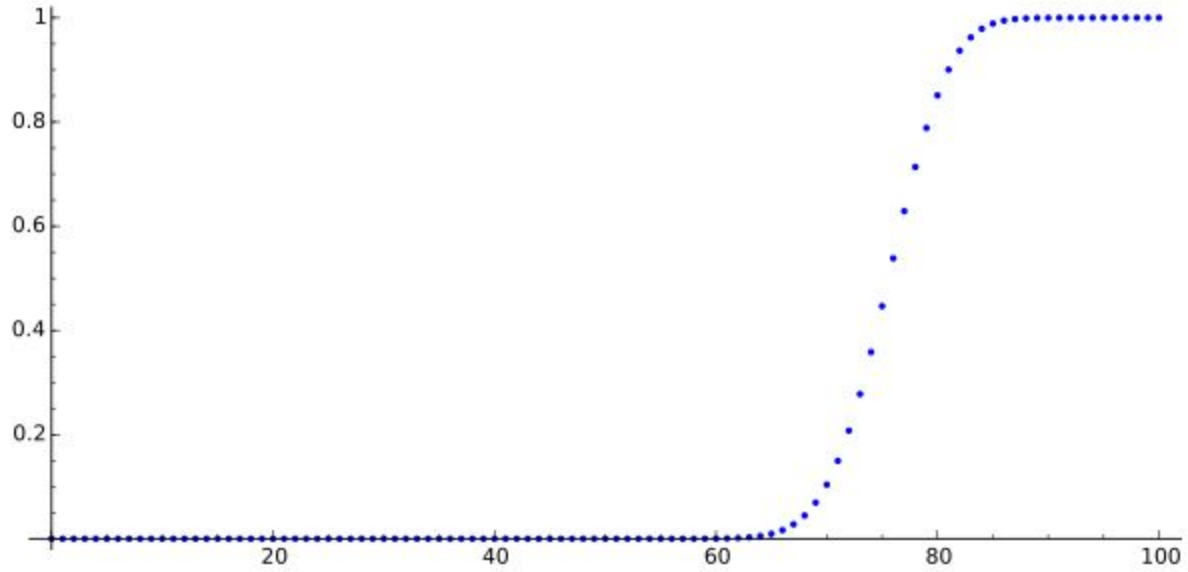


$$.25^{100-m} \times .75^{m} \times \binom{100}{m}$$

But we're trying to find a threshold, thus, we're interested in understanding the probability that a read will align with *m or fewer* mismatches. For this, we compute the cumulative mass function (cmf) as below.

$$\sum_{m=0}^{l} P(match)^{l-m} \times (1 - P(match))^{m} \times \binom{l}{m}$$

The cmf for *P(match)* = .25 and *l* = 100 is plotted[b] below. *m* is on the horizontal axis and probability is on the vertical axis.



$$\sum_{m=0}^{100} .25^{100-m} \times .75^{m} \times \binom{100}{m}$$

Judging from the cmf plot above, we can see that the highest value of *m* that will give us a probability of a read aligning lower than .005 is somewhere around 60; in fact, for *P(match)* = .25 and *l* = 100, it's 62.

This can be considered a starting point for the absolute upper bound of our mismatch threshold. There are still two glaring issues:

1. We're not aligning one read. We're aligning 43,918,181[c].
2. We're not aligning to a UR distribution.

First, we can adjust our cmf to compensate for the number of reads as below.

$$1 - \left(\sum_{m=0}^{100} 1 - \left(.25^{100-m} \times .75^{m} \times \binom{100}{m} \right) \right)^{43918181}$$

Neither Wolfram Alpha[3] nor Sage Math Cloud[4] produce plots for this cmf, though Wolfram Alpha[3] does produce a value, allowing us to calculate the highest value of *m* that will give us a probability of all our 43,918,181 reads aligning with a probability lower than .005: 42.

Second, we can adjust our cmf to account for the actual distribution of nucleotides in our reference. The software developed for this assignment[5] calculates the following ratios for the reference transcripts.

```
A:0.28908129592053294
T:0.28797606910667517
C:0.20011402374243129
G:0.19987491778448566
```

We can use these ratios to better calculate the probability of nucleotides matching as below.

$$\begin{aligned} P(match) &= max(P(matching\ A's), P(matching\ T's), P(matching\ C's), P(matching\ G's)) \\ &\approx max(.289..., .287..., .200..., .199...) \\ &\approx 0.28908129592053294 \end{aligned}$$

And adjust our cmf as below.

$$1 - \left(\sum_{m=0}^{100} 1 - \left(0.28908129592053294^{100-m} \times (1 - 0.28908129592053294)^{100} \times \binom{100}{m} \right) \right)^{43918181}$$

With this, the highest value of *m* that will give us a probability of all our 43,918,181[c] reads aligning to a reference sequence matching the distribution of our own with a probability lower than .005 is calculated to be 39. This value should be considered a hard upper bound on the number of mismatches we can allow during an alignment.

Assumptions

1. The aligner will handle all splice sites equally.
2. The aligner will treat all mismatches equally.

Limitations

1. Gaps are ignored.
2. Read and reference nucleotide match and mismatch probabilities are not rigorous and could be more accurately approximated using a higher-order Markov Model; this calculation can be

time consuming. Obviously, at an *l*-order Markov Model, we'd be better off using a simulation of artificial reads.

Alignment Percentages[e]

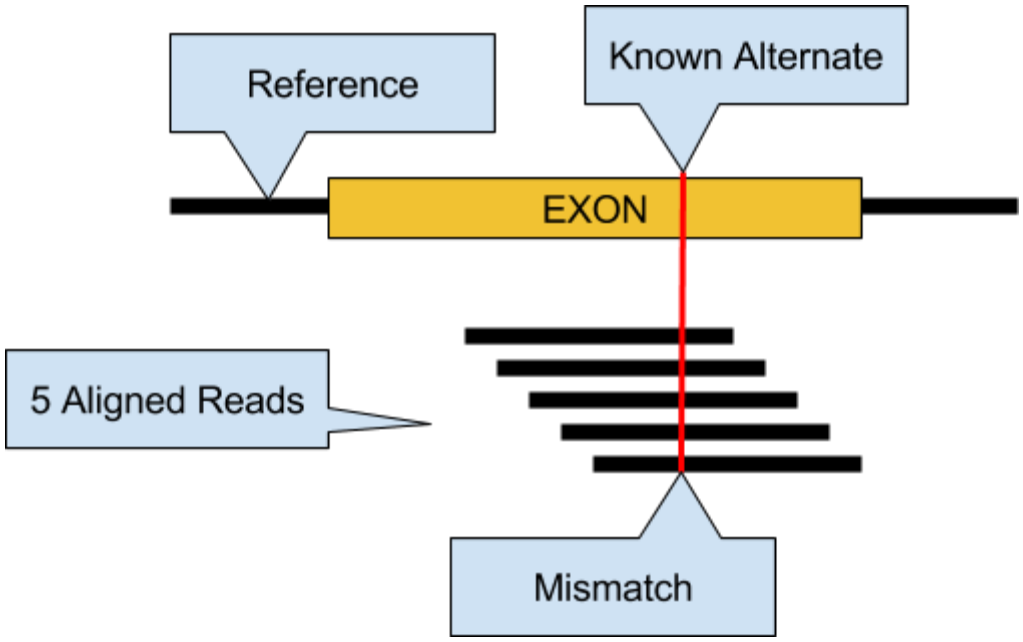
```
                          Uniquely mapped reads % |    12.78%
        % of reads mapped to multiple loci |    0.71%
```

Strain Specific / General Approach

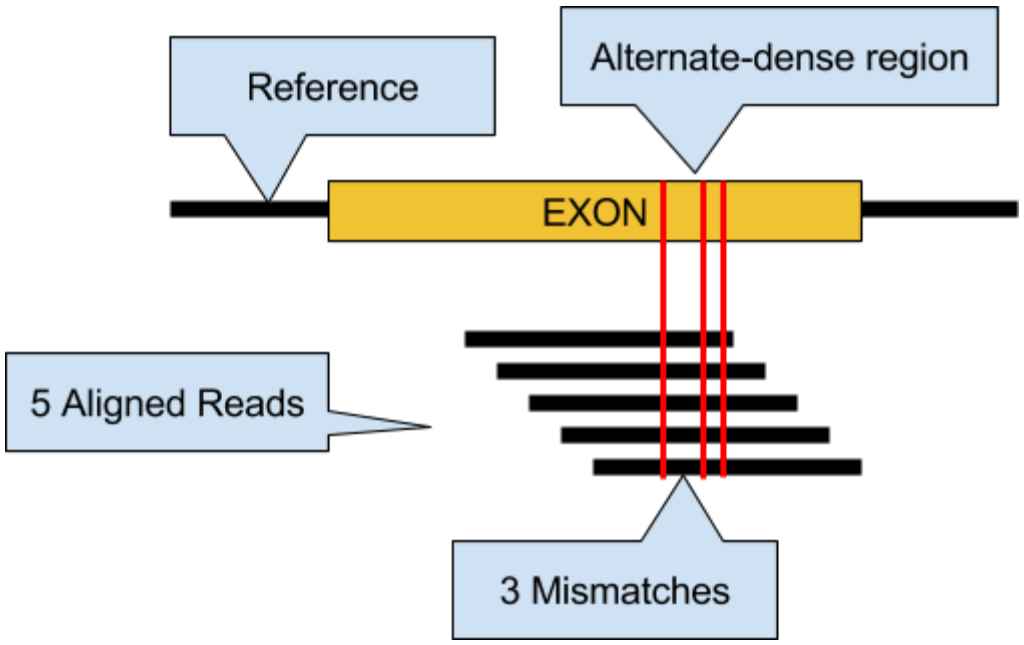
Summary

Remembering that the above calculated value can be considered a strong upper bound for this problem, we'll aim to find a lower biologically-motivated threshold by calculating the fewest possible mismatches we can allow without significantly dropping read coverage.

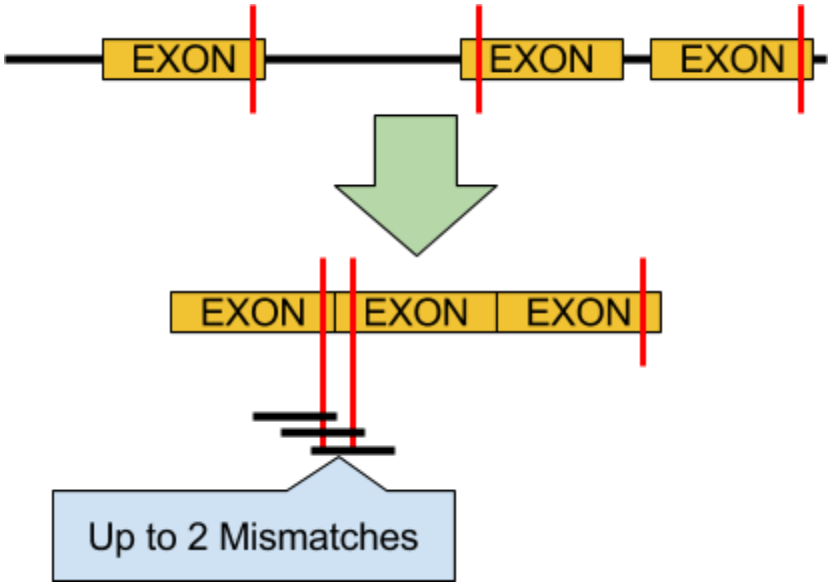
The VCF file shows alternate calls that, if within exons, can be expected to count as mismatches when aligned to the reference.



If some number of alternates, *a*, falls within the length of a read, *l*, at least *a* mismatches must be allowed to occur for an alignment to take place. We'll refer to any region of length *l* with positive *a* as alternate-dense.



Remembering that our reads were generated using RNA-Seq, we must consider the stitching process that occurs following transcription whereby distal exons can become adjacent in a transcript prior to assessing alternate-density.



It's important to identify these regions because they're likely regions of interest in our study. If they weren't, we'd not be using a VCF file! The software developed for this assignment[5] identifies the following transcript to contain the region of length 100 with highest alternate-density.

```
Transcript ID: ENSMUST00000194797
Alternate Count: 20
Alternates:
171573768, 171573770, 171573771, 171573786, 171573789, 171573804,
171573808, 171573810, 171573820, 171573822, 171573828, 171573833,
171573834, 171573837, 171573844, 171573847, 171573852, 171573856,
171573857, 171573861
```

Considering the maximum alternate-density for our dataset is 20 and that the region is likely highly interesting to us, we'll now consider 20 a hard lower bound on the number of mismatches we can allow during an alignment.

Assuming a 0.1% error rate[6] (and that said error rate is UR), we can estimate the ratio of reads whose alignments to our alternate-dense region would be rejected despite that region being its biological origin. We'll simply multiply the probability of a miscalled base in an alignment with $l = 100$ by the number of bases that aren't already mismatched (20 in this region).

$$.001 \times (100 - 20) = .08$$

Disregarding 8% of the reads in this region seems it could cause a steep drop in coverage; allowing for an additional mismatch will allow us to reject a much lower proportion. Here we'll multiply the probability of a miscalled base by the number of bases that aren't already mismatched (20 in this region) by the probability of a second miscalled base by the number of bases that still aren't mismatched (now 21 in this region).

$$\prod_{m=20}^{21} .001 \times (100 - m) = 0.00632$$

Disregarding only .6% of our reads seems far more appropriate.

Considering the region is known to be alternate-dense, it's possible some biological mechanism is in place that increases the likelihood of point mutations. The rate of novel point mutations, in this, or any other region, is likely a complicated calculation. We'll acknowledge its presence but assume it's rather low. We'll allow for a single additional mismatch to account for both the known sequencing error rate[6] and the likelihood of a point mutation, increasing our threshold to 21. Considering that this is well below our previously calculated theoretical mismatch threshold hard upper bound (of 39), 21 seems an appropriate value for this parameter.

Assumptions

3. Exons are not reordered during stitching.
4. Novel point mutation events are exceedingly rare.
5. The aligner will handle all splice sites equally.
6. The aligner will treat all mismatches equally.

Limitations

1. Gaps are ignored.

Alignment Percentages[e]

Uniquely mapped reads % |	**12.78%**
% of reads mapped to multiple loci |	**0.71%**

References

[1] Engström, Pär G., et al. "Systematic evaluation of spliced alignment programs for RNA-seq data." *Nature methods* 10.12 (2013): 1185-1191.

[2] Dobin, Alexander, et al. "STAR: ultrafast universal RNA-seq aligner." *Bioinformatics* 29.1 (2013): 15-21.

[3] http://www.wolframalpha.com/

[4] https://cloud.sagemath.com/

[5] https://github.com/joshuaburkhart/BioinformaticsAlgorithms/tree/master/final_proj

[6] Christina Zheng, zheng@ohsu.edu

Notes

[a] plotted using the Sage Math Cloud[4] with the python code below.

```
pmf=lambda m: ([.25^(100-m) * .75^(m) * binomial(100,m)])
pts = [point([i,pmf(i)[0]]) for i in range(0,101)]
p = add(pts)
show(p)
```

[b] plotted using the Sage Math Cloud[4] with the python code below.

```
cmf=lambda x: add([.25^(100-x) * .75^(x) * binomial(100,x) for x in
range(0, x)])
pts2 = [point([j,cmf(j)]) for j in range(0,101)]
p2 = add(pts2)
show(p2)
```

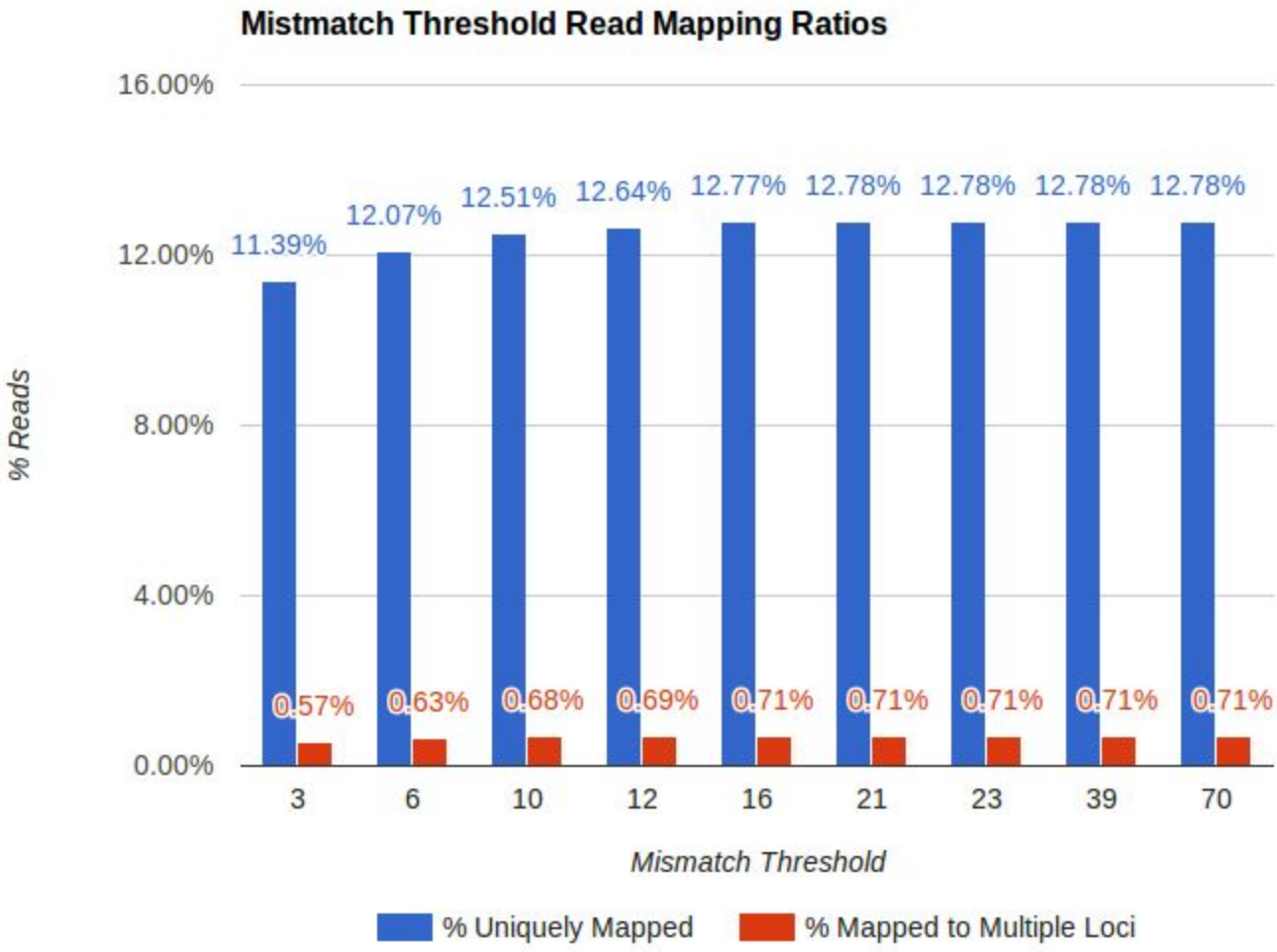
[c] calculated from the data given for this assignment with the below python function.

```
def parse_fastq(fq):
     d = {}
     with open(fq) as FileObj:
     next(FileObj)
     read_cnt = 0
     for line in FileObj:
          Ns = line.count('N')
          read_cnt += 1
          if Ns in d:
               d[Ns] += 1
          else:
               d[Ns] = 1
               print('Read with {0} Ns detected:
{1}'.format(Ns,line.rstrip()))
          try:
               next(FileObj)
               next(FileObj)
               next(FileObj)
          except StopIteration:
               break
     print(d)
     print('read_cnt: {0}'.format(read_cnt))
```

[d] viewable in the data given for this assignment with the below command.

```
$ tail -n20 ~/BMI650/input/PWK_R1.fastq
```

[e] STAR execution results for the below chart are available on church.ohsu.edu.


Mistmatch Threshold Read Mapping Ratios
16.00%
12.00%
8.00%
4.00%
0.00%
% Reads
11.39%
12.07%
12.51%
12.64%
12.77%
12.78%
12.78%
12.78%
12.78%
0.57%
0.63%
0.68%
0.69%
0.71%
0.71%
0.71%
0.71%
0.71%
3
6
10
12
16
21
23
39
70
Mismatch Threshold
% Uniquely Mapped
% Mapped to Multiple Loci