Assignment 1: Mapping

Abdullah Faqih Al Mubarok - vpx267

September 22, 2023

1 Exact alignment (Rabin-Karp)

Consider the text ANCGTAACGTAACGA and pattern TAACG

1. What is the length of the strings and what is the size of the alphabet?

   Answer:

   The length of the text string and the pattern are 15 and 5 respectively. The size of alphabet is 5 A, C, G, T, N where N should represents any other bases.

2. How many shifts and how many comparisons do we need to do in order to perform an exhaustive naive search?

   Answer:

   The shifts will be $(n - m + 1)$. Since $n = 15$ and $m = 5$, we have 11 shifts. In addition, we have $(n - m + 1)m$ comparisons which is 55.

3. Consider the Rabin-Karp algorithm and define a rolling hash function.

   Answer:

   Based on the slide of Advanced Bioinformatics course (page 5), the Rabin-Karp algorithm is a hash function that tries to utilize the hash value from the preceding window so that it does not need to recalculate the hash function from scratch. The chosen hash function also needs to also aware of the position for avoiding overlap (i.e. the hash value of AAC is different compared to ACA). Then we can define the rolling hash function as the third hash function of Robin-Karp algorithm on the Advance Bioinformatics slide (page 5):

$$\begin{aligned} h(S, i, j) &= (\sum_{n=0}^{j} S_{i+n} * 10^n) \mod q \\ &= ((h(S, i-1, j) - S_{i-1} * 10^j) * 10 + S_{i+j}) \mod q \end{aligned}$$

   where i is shift, j is window length, and S is the string, and q is an arbitrary large prime number.

4. Explain why your hash function is a rolling hash function.

   Answer:

   It is considered as a rolling hash function since it does not need to recalculate the hash function from scratch (which sometimes expensive). It uses the hash value from the previous window and subtract it with the removed base and add it with the additional base.

5. Compute the hash of the pattern and the hash of each shift. Is your choice of hash function good? How would you measure if a hash function is "good"?

   Answer:

   First of all, we need to define the integer representation of each base as:

$$S = \begin{cases} 0, & \text{if A} \\ 1, & \text{if C} \\ 2, & \text{if G} \\ 3, & \text{if T} \\ 4, & \text{if N} \end{cases}$$

   I chose $q = 3,137$ to avoid collision since there might be 5^5 combination of substring.

   Next, we can find the hash of the pattern TAACG:

$$\begin{aligned} S(TAACG) &= 30012 \\ h(TAACG) &= ((3 * (10^4)) + (0 * (10^3)) + (0 * (10^2)) + (1 * (10^1)) + (2)) mod 3137 \\ &= 1779 \end{aligned}$$

   Then we can start to calculate the hash value of each shift of the text ANCG-TAACGTAACGA:

   shift 0

$$\begin{aligned} S(ANCGT) &= 04123 \\ h(text, 0, 5) &= ((0 * (10^4)) + (4 * (10^3)) + (1 * (10^2)) + (2 * (10^1)) + (3)) mod 3137 \\ &= 986 \end{aligned}$$

   shift 1

$$\begin{aligned} S(NCGTA) &= 41230 \\ h(text, 1, 5) &= ((h(text, 0, 5) - (0 * (10^4))) * 10 + 0) mod 3137 \\ &= ((986 - (0 * (10^4))) * 10 + 0) mod 3137 \\ &= 449 \end{aligned}$$

shift 2

$$\begin{aligned} S(CGTAA) &= 12300 \\ h(text, 2, 5) &= ((h(text, 1, 5) - (4 * (10^4))) * 10 + 0) mod 3137 \\ &= (449 - (4 * (10^4))) * 10 + 0) mod 3137 \\ &= 2889 \end{aligned}$$

shift 3

$$\begin{aligned} S(GTAAC) &= 23001 \\ h(text, 3, 5) &= ((h(text, 2, 5) - (1 * (10^4))) * 10 + 1) mod 3137 \\ &= (2889 - (1 * (10^4))) * 10 + 1) mod 3137 \\ &= 1042 \end{aligned}$$

shift 4

$$\begin{aligned} S(TAACG) &= 30012 \\ h(text, 4, 5) &= ((h(text, 3, 5) - (2 * (10^4))) * 10 + 2) mod 3137 \\ &= (1042 - (2 * (10^4))) * 10 + 2) mod 3137 \\ &= 1779 \end{aligned}$$

shift 5

$$\begin{aligned} S(AACGT) &= 00123 \\ h(text, 5, 5) &= ((h(text, 4, 5) - (3 * (10^4))) * 10 + 3) mod 3137 \\ &= (1779 - (3 * (10^4))) * 10 + 3) mod 3137 \\ &= 123 \end{aligned}$$

shift 6

$$\begin{aligned} S(ACGTA) &= 01230 \\ h(text, 6, 5) &= ((h(text, 5, 5) - (0 * (10^4))) * 10 + 0) mod 3137 \\ &= (123 - (0 * (10^4))) * 10 + 0) mod 3137 \\ &= 1230 \end{aligned}$$

shift 7

$$\begin{aligned} S(CGTAA) &= 12300 \\ h(text, 7, 5) &= ((h(text, 6, 5) - (0 * (10^4))) * 10 + 0) mod 3137 \\ &= (1230 - (0 * (10^4))) * 10 + 0) mod 3137 \\ &= 2889 \end{aligned}$$

shift 8

$$\begin{aligned} S(GTAAC) &= 23001 \\ h(text,8,5) &= ((h(text,7,5) - (1*(10^4)))*10+1) mod 3137 \\ &= (2889 - (1*(10^4)))*10+1) mod 3137 \\ &= 1042 \end{aligned}$$

shift 9

$$\begin{aligned} S(TAACG) &= 30012 \\ h(text,9,5) &= ((h(text,8,5) - (2*(10^4)))*10+2) mod 3137 \\ &= (1042 - (2*(10^4)))*10+2) mod 3137 \\ &= 1779 \end{aligned}$$

shift 10

$$\begin{aligned} S(AACGA) &= 00120 \\ h(text,10,5) &= ((h(text,9,5) - (3*(10^4)))*10+0) mod 3137 \\ &= (1779 - (3*(10^4)))*10+0) mod 3137 \\ &= 120 \end{aligned}$$

To determine whether a hash function is "good", there are two main factors: the hash function needs to be deterministic and it has minimal collision.

The above chosen hash function is considerably good since the q for the modulo operation takes account the possible hash values to avoid collision (which is also reflected on the above hash value calculation) and it is also deterministic (the same substrings have the same hash value).

6. How many operations did the exhaustive search with the Rabin-Karp algorithm require?

Answer:

To calculate the total operations, we need to consider two steps:

(a) Preprocessing
   i. We need to calculate the hash function of the pattern which equals to O(m)
   ii. rolling hash value generation which equals O(1) x O(n-m+1)

(b) Matching
   We need to calculate:
   i. hash comparison O(1)
   ii. character comparison. Since it is hard to have the perfect hash function, we might still compare each character after finding a hash function. The worst case is O(m)*O(n-m+1) = O(nm)

Hence we have total complexity of O(m) + (O(1) x O(n-m+1)) +O(1) + O(nm) = O(nm)

2 Data Structures

Consider the sequence T = ”ORCINUSORCA” And the query sequence S = ”ORC”

1. List all of the suffixes

   Answer:

   Since T has 11 characters, it has 11 suffixes:

   ORCINUSORCA$

   RCINUSORCA$

   CINUSORCA$

   INUSORCA$

   NUSORCA$

   USORCA$

   SORCA$

   ORCA$

   RCA$

   CA$

   A$

2. Draw the suffix trie and suffix tree for T.

   Answer:

   I made the two drawings manually using app.diagram.net. Figure 1 and figure 2 show the suffix tree and trie respectively.

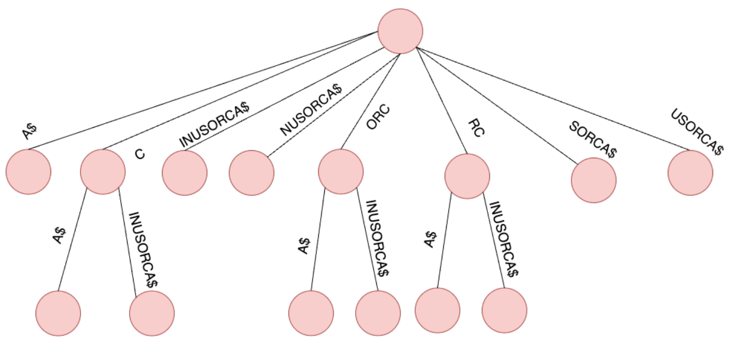

Figure 1: Suffix Tree of T

3. Write the Burrows-Wheeler matrix and highlight the suffix array and BWT.

   Answer:

   Table 1 shows the BWT Matrix. We can see from the table that the BWT Transform is ACRRCIS$OOUN

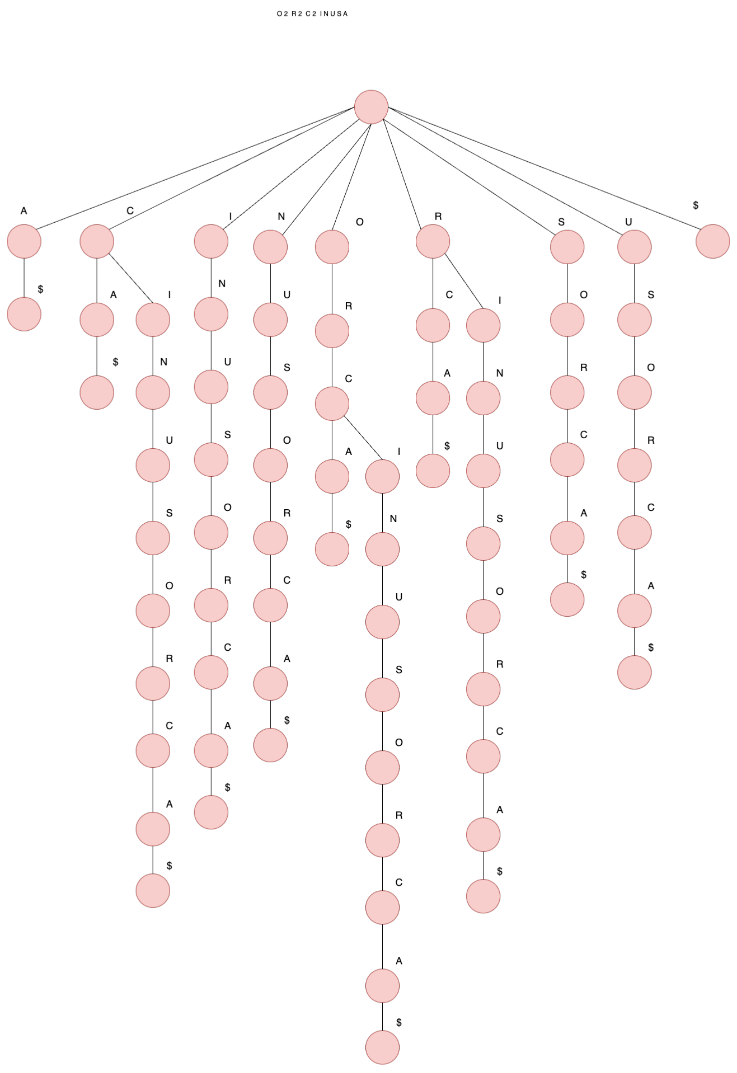

Figure 2: Suffix Trie of T

4. Make the character table and occurrence function.

   Answer:

   Character table and occurrence function are represented by table 2 and table 3 respectively.

5. Identify the position for which the query sequence S occurs within the sequence T. Use the FM index approach using the suffix array, character table and occurrence function (show your backward search calculations).

   Answer:

   Step 1:

$$\begin{aligned} i &= C[C] + 1 = 2 + 1 = 3 \\ j &= C[C + 1] = C[I] = 4 \end{aligned} \tag{1}$$

Suffix Array Pos	**Suffix**	**BWT**
12	\$	A
11	A\$	C
10	CA\$	R
3	CINUSORCA\$	R
4	INUSORCA\$	C
5	NUSORCA\$	I
8	ORCA\$	S
1	ORCINUSORCA	\$
9	RCA\$	O
2	RCINUSORCA\$	O
7	SORCA\$	U
6	USORCA\$	N

Table 1: BWT Matrix

a	\$	A	C	I	N	O	R	S	U
C[a]	0	1	2	4	5	6	8	10	11

Table 2: Character Table

POS	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
BWT	**A**	**C**	**R**	**R**	**C**	**I**	**S**	**\$**	**O**	**O**	**U**	**N**
\$	0	0	0	0	0	0	0	1	1	1	1	1
A	1	1	1	1	1	1	1	1	1	1	1	1
C	0	1	1	1	2	2	2	2	2	2	2	2
I	0	0	0	0	0	1	1	1	1	1	1	1
N	0	0	0	0	0	0	0	0	0	0	0	1
O	0	0	0	0	0	0	0	0	1	2	2	2
R	0	0	1	2	2	2	2	2	2	2	2	2
S	0	0	0	0	0	0	1	1	1	1	1	1
U	0	0	0	0	0	0	0	0	0	0	1	1

Table 3: Occurrence Function

Step2:

$$\begin{aligned} i &= C[R] + OCC(R, 3-1) + 1 \\ &= 8 + 0 + 1 = 9 \end{aligned} \tag{2}$$

$$\begin{aligned} j &= C[R] + OCC(R, 4) \\ &= 8 + 2 = 10 \end{aligned} \tag{3}$$

Step 3:

$$\begin{aligned} i &= C[O] + OCC(O, 9-1) + 1 \\ &= 6 + 0 + 1 = 7 \end{aligned} \tag{4}$$

$$\begin{aligned} j &= C[0] + OCC(O, 10) \\ &= 6 + 2 = 8 \end{aligned} \tag{5}$$

Then we look up the BWT range of 7 and 8 in table 1 and return its suffix position which are 1 and 8.

In conclusion, the query "ORC" can be found on the position 1 and 8 of the text "ORCINUSORCA"

3 Mapping Statistics

There are two fastq.gz files in the directory /TEACHING/BIOINF23/assignment1_mapping. One is called L10.fastq.gz and contains single end reads of length 10, and one is called L30.fastq.gz and has reads of length 30. You will align both of these files and investigate the effect of read length on the resulting mapped reads. Both of these files are from the organism Mycobacterium Leprae. You will also find the reference sequence to map against mleprae_reference_genome.fasta.gz.

Due to the simulation software used, the read ID contains the information of the positions for which the sequence originates. Once aligned, there can potentially be differences between the true chromosomal position and the position it is aligned to, we call this mis-alignment.

e.g. @T0_RID0_S0_NZ_CP029543.1:2084294-2084323_length:30_R1 is a read (ID) originating from chromosome NZ_CP029543.1 from position 2084294-2084323 We can use this fact to investigate whether the reads generated are mapping to the correct position by comparing the position in the read ID to the position in the bam file.

1. What is the filesize of the reference file? How large (in bases) is the bacterial reference genome?

   Answer:

   The filesize of the reference genome is 980K and its total bases is 3,187,112 bases. Below are the commands:

   ```
   # finding the file size
   du -sh mleprae_reference_genome.fasta.gz
   #finding the base size
   zcat mleprae_reference_genome.fasta.gz  | tail -n +2
      | wc -lc | awk '{print $2 - $1}'
   ```

2. Use bwa aln (and bwa samse) to align the two fastq files to the bacterial reference.

   Answer:

   Align the files first:

   ```
   bwa aln -t 2 mleprae_reference_genome.fasta.gz L10.
      fastq.gz > L10_SA.sai
   bwa aln -t 2 mleprae_reference_genome.fasta.gz L30.
      fastq.gz > L30_SA.sai
   ```

   Produce the SAM files:

   ```
   bwa samse mleprae_reference_genome.fasta.gz L10_SA.
      sai L10.fastq.gz > output_L10_aln.sam
   bwa samse mleprae_reference_genome.fasta.gz L30_SA.
      sai L30.fastq.gz > output_L30_aln.sam
   ```

3. Sort and index the resulting files using samtools, and make sure they are saved in bam format.

   Answer:

   Sort the two files and output into .bam format:

   ```
   samtools sort output_L10_aln.sam -o
      output_sorted_L10_aln.bam
   samtools sort output_L30_aln.sam -o
      output_sorted_L30_aln.bam
   ```

   index the two bam files:

   ```
   samtools index output_sorted_L10_aln.bam
   samtools index output_sorted_L30_aln.bam
   ```

4. Filter out the unaligned reads and create new bam files that contain only the mapped reads.

   Answer:

   The unmapped with can be filtered out with '-F 4' command:

   ```
   samtools view -b -F 4 output_sorted_L10_aln.bam >
      output_sorted_filtered_L10_aln.bam
   samtools view -b -F 4 output_sorted_L30_aln.bam >
      output_sorted_filtered_L30_aln.bam
   ```

5. The script /TEACHING/BIOINF23/assignment1_mapping/get_stats.sh takes a bam file and out- puts two numbers - firstly the number of reads that map correctly (i.e. those where the start position in the bam file matches the read ID), and then the number that do not map correctly. Use this script to find out how many reads map correctly and incorrectly in the two bam files (e.g. bash /TEACHING/BIOINF23/assignment1_mapping/get_stats.sh L30.bam)

   Answer:

```
./get_stats.sh output_sorted_filtered_L10_aln.bam
./get_stats.sh output_sorted_filtered_L30_aln.bam
```

from the script, the sorted and filtered mapped L10 has 15231 correctly mapped and 84768 incorrectly mapped. Meanwhile the sorted and filtered mapped L30 has 99685 correctly mapped and 313 incorrectly mapped.

6. Now create two new bam files where you filter the reads so we only retain reads with a mapping quality of greater than or equal to 1.

   Answer:

   First, we filter based on the mapping quality of the previous processed bam files:

   ```
   samtools view -b -q 1 output_sorted_filtered_L10_aln.
       bam > output_sorted_filtered_mapqual_L10_aln.bam
   samtools view -b -q 1 output_sorted_filtered_L30_aln.
       bam > output_sorted_filtered_mapqual_L30_aln.bam
   ```

7. Repeat the exercise in question 5 with the two new filtered bam files. How do the numbers differ?

   Answer:

   ```
   ./get_stats.sh output_sorted_filtered_mapqual_L10_aln
       .bam
   ./get_stats.sh output_sorted_filtered_mapqual_L30_aln
       .bam
   ```

   After additional filtering with the quality greater than equal to 1, the L10 reads have 1681 correctly mapped and 53 incorrectly mapped while all of the L30 reads are correctly mapped (99452)

8. Is the proportion of incorrectly mapped reads greater or smaller in the original bams or the quality filtered bams? Why do you think that is?

   Answer:

   To check that, we need to check the original bams file:

   ```
   ./get_stats.sh output_sorted_L10_aln.bam
   ./get_stats.sh output_sorted_L30_aln.bam
   ```

   From the above command, the original L10 reads and L30 reads have 84768 and 313 incorrectly mapped respectively.

   The original bams have higher incorrectly mapped reads since they still contain very low map quality reads (MAPQ < 1) which have higher chance to being incorrectly mapped.

9. Given the results there, do you think it is relevant to include mapping filters in downstream analysis?

   It is important to include the mapping quality filtering since we only want to include the reads that we are confident mapped into the correct location thus reducing noise.

4 Ancient Data Analysis

In this part you are given two .fq.gz files in the directory /TEACHING/BIOINF23/assignment1_mapp
namely inspector.fastq.gz and barnaby.fastq.gz. One of these files is sequencing data from an ancient bacteria, and one is from a modern bacteria. There were some troubles in the lab, and we have no record of which one is which. Your goal is to ascertain which one is which.

1. Briefly describe what characterizes ancient DNA from modern DNA Ther are several things that distinguish the aDNA Briggs et al. (2007):

   Answer:

   - aDNAs tend to be degraded into smaller sizes
   - They are chemically damaged which caused by contaminations of exogenous factors which leads deamination of cytosinse (C to T) Jónsson et al. (2013).

2. Perform adapter trimming on the provided sample sequence files using fastp. Make sure to discard reads shorter than 30 bp. (-l parameter). Remember this is single end (SE) data.

   Answer:

   Below script does the length filter and adapting trimming for the two yeast data:

   ```
   fastp -l 30 --detect_adapter_for_pe -i inspector.
     fastq.gz -o inspector_trimmed.fastq.gz
   fastp -l 30 --detect_adapter_for_pe -i barnaby.fastq.
     gz -o barnaby_trimmed.fastq.gz
   ```

3. How many reads contained adapters in both datasets? (Hint: Look at the output from fastp)

   Answer:

   Based on the log outputs, it can be observed that the inspector.fastq.gz file contains 798731 reads with adapters, while the barnaby.fastq.gz file contains 705877 reads with adapters.

4. What is the mean length of the reads before and after trimming? (Hint: There is a command for this near the bottom of the day 2 exercises)

   Answer: First, we calculate the average length for the unprocessed files:

   ```
   zcat inspector.fastq.gz | awk 'NR%4==2{sum+=length($0
     )}END{print sum/(NR/4)}'
   zcat barnaby.fastq.gz | awk 'NR%4==2{sum+=length($0)}
     END{print sum/(NR/4)}'
   ```

   from the above command, The initial mean length of the inspector.fastq.gz file prior to trimming and length filtering is 125, whereas the barnaby.fastq.gz file is 87.7787.

```
zcat inspector_trimmed.fastq.gz | awk 'NR%4==2{sum+=
   length($0)}END{print sum/(NR/4)}'
zcat barnaby_trimmed.fastq.gz | awk 'NR%4==2{sum+=
   length($0)}END{print sum/(NR/4)}'
```

Meanwhile, after the trimming, the mean length of the inspector is 89.9716 while barnaby is 59.1209

5. Perform bwa alignment using aln and samse. For each sample, sort the sam file, save it as a bam, and index it. Remember, if you are struggling you can find the output for this question in /TEACHING/BIOINF23/assignment1_mapping/output/part4/. Note that the reference is not the same as the last part. It is /TEACHING/BIOINF23/assignment1

   Answer:

   First, we do it for the inspector first:

```
bwa aln paeruginosa.fasta.gz inspector_trimmed.fastq.
   gz > inspector_trimmed_SA.sai
bwa samse paeruginosa.fasta.gz inspector_trimmed_SA.
   sai inspector_trimmed.fastq.gz >
   output_inspector_aln.sam
samtools sort output_inspector_aln.sam -o
   output_sorted_inspector_aln.bam
samtools index output_sorted_inspector_aln.bam
```

   then for the barnaby:

```
bwa aln paeruginosa.fasta.gz barnaby_trimmed.fastq.gz
    > barnaby_trimmed_SA.sai
bwa samse paeruginosa.fasta.gz barnaby_trimmed_SA.sai
    barnaby_trimmed.fastq.gz > output_barnaby_aln.sam
samtools sort output_barnaby_aln.sam -o
   output_sorted_barnaby_aln.bam
samtools index output_sorted_barnaby_aln.bam
```

6. For each sample, create new bam files with just the aligned reads. What proportion of reads remain?

   Answer:

   To just output the mapped reads, we can use the ”-F 4” filter and applies it into the two files:

```
samtools view -F 4 output_sorted_inspector_aln.bam -o
    output_sorted_filtered_inspector_aln.bam
samtools view -F 4 output_sorted_barnaby_aln.bam -o
   output_sorted_filtered_barnaby_aln.bam
```

   To know the remaining reads of the two mapped files we execute the bloew command:

```
awk -v before="`samtools view -c
   output_sorted_barnaby_aln.bam`" -v after="`
   samtools view -c
   output_sorted_filtered_barnaby_aln.bam`" 'BEGIN {
   print(after/before)}'
awk -v before="`samtools view -c
   output_sorted_inspector_aln.bam`" -v after="`
   samtools view -c
   output_sorted_filtered_inspector_aln.bam`" 'BEGIN
   {print(after/before)}'
```

Based on the printed results, the inspector had 89.7% reads remaining after the filtering while barnaby has 36.8%.

7. What is the average depth of each sample (aligned reads only)? Use samtools depth barnaby.mapped..bam—datamash mean 3

   Answer:

   ```
   samtools depth output_sorted_filtered_inspector_aln.
      bam|datamash mean 3
   samtools depth output_sorted_filtered_barnaby_aln.bam
      |datamash mean 3
   ```

   Based on the command above, the average depth of inspector is 10.775 while barnaby is 2.3889

8. Use mapDamage to identify the nucleotide misincorporation and fragmentation patterns, and de- scribe (and include) the output plots generated by MapDamage found in the files Fragmisincorpo- ration plot.pdf and Length plot.pdf. Make sure to use the command line options –no-stats

   Answer:

   Below is the command used to produce the plots using mapDamage:

   ```
   mapDamage -i output_sorted_filtered_inspector_aln.bam
       -r paeruginosa.fasta.gz --no-stats
   mapDamage -i output_sorted_filtered_barnaby_aln.bam -
      r paeruginosa.fasta.gz --no-stats
   ```

   Figure 3, 4, 5, 6 show the mapDamage results for barnaby and inspector

9. Based on your results obtained from the previous questions, which of the files looks ancient and which one looks modern? Why?

   Answer:

   From the figure 3, it seems that the barnaby reads are chemically damaged (substitution of C to T at the beginning of the reads and the substitution of G to A at the end of the reads). Meanhwile, the inspector does not have any substitution as depicted on figure 5. In addition, the length distribution of the inspector reads are

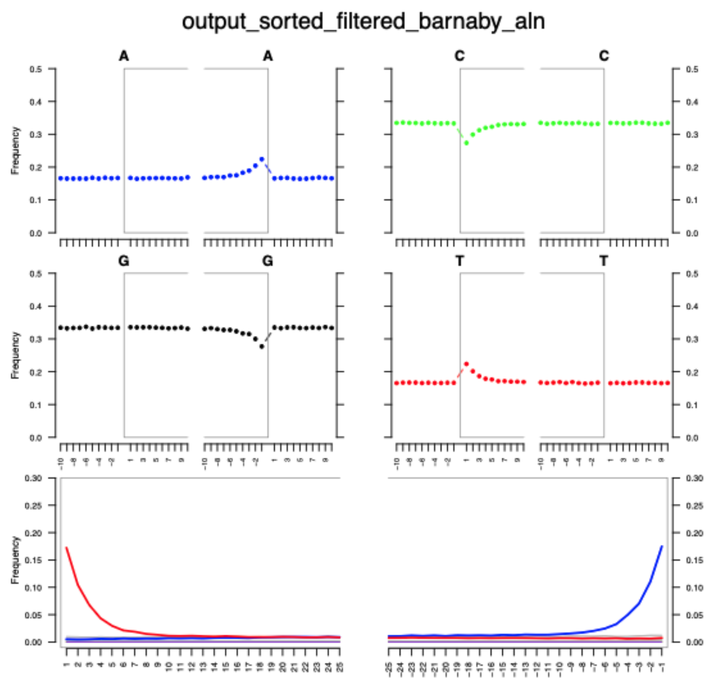

Figure 3: Misincorportaion Plot of Barnaby

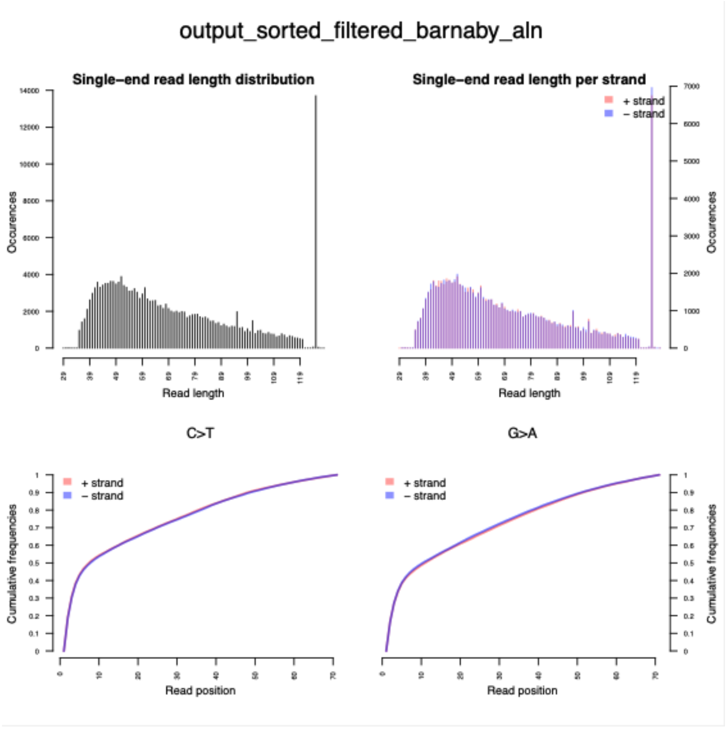

Figure 4: Length Distribution of Barnaby

normally distributed while the barnaby is not. Therefore, we can conclude that the barnaby is the ancient and the inspector is the modern one.

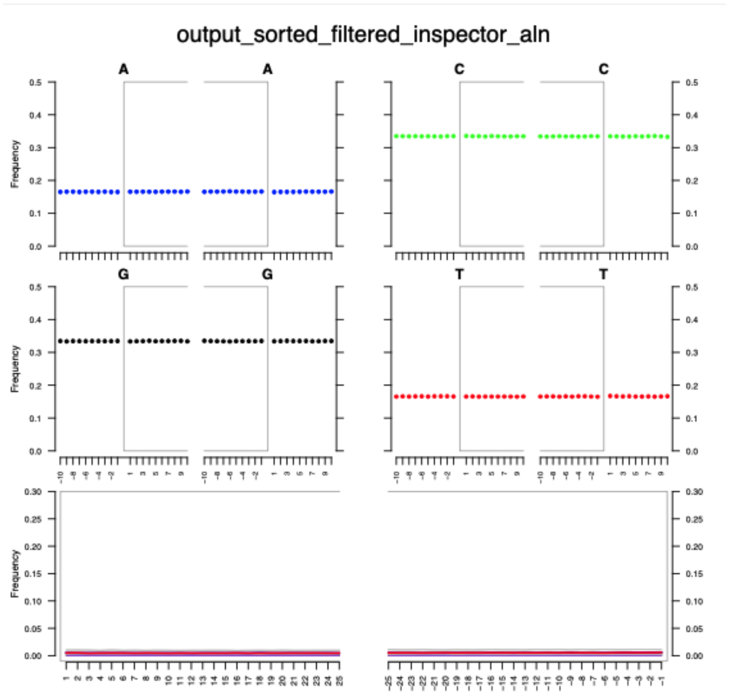

Figure 5: Misincorportaion Plot of Inspector

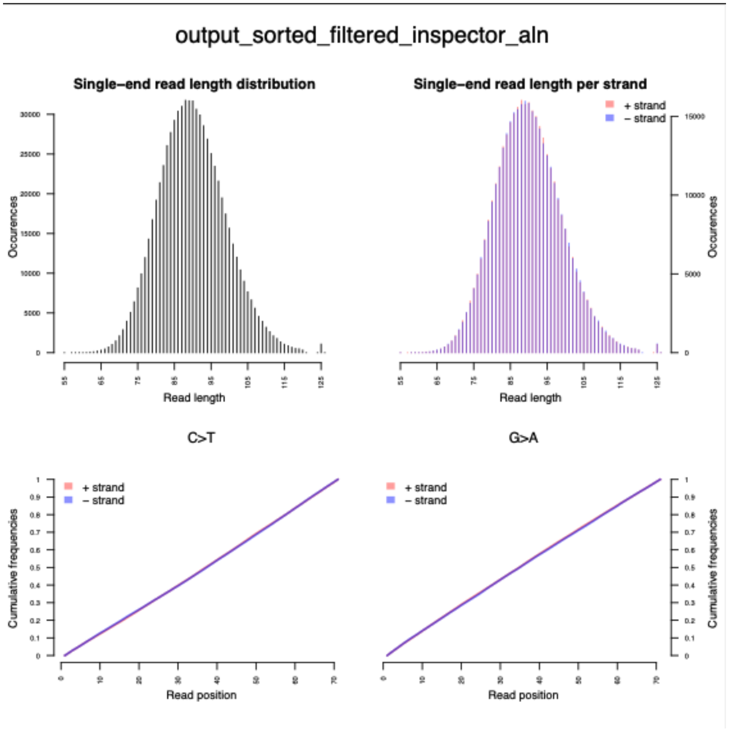

Figure 6: Length Distribution of Inspector

References

Briggs, Adrian W. et al. (2007). “Patterns of damage in genomic DNA sequences from a Neandertal”. In: *Proceedings of the National Academy of Sciences* 104(37), pp. 14616–14621. DOI: 10.1073/pnas.0704665104.

Jónsson, Hákon et al. (2013). “mapDamage2.0: fast approximate Bayesian estimates of ancient DNA damage parameters”. In: *Bioinformatics* 29(13), pp. 1682–1684. DOI: 10.1093/bioinformatics/btt193.