Project 1: Low dimensional representation with big genomics data

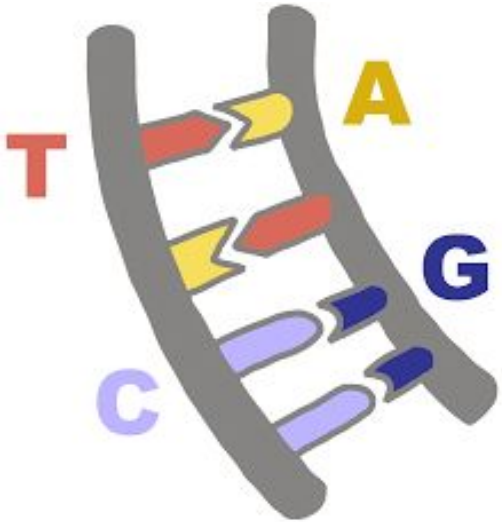
Anna Cameron and Henry Savich

Background: Genomes

Genomes consist of sequences of adenine (A), guanine (G), thymine (T), and cytosine (C). We call each element of the sequence a base pair (BP)

There are approx. 3 billion BP in a human genome

The location of a sequence of DNA in the whole genome tells us what genes it relates to



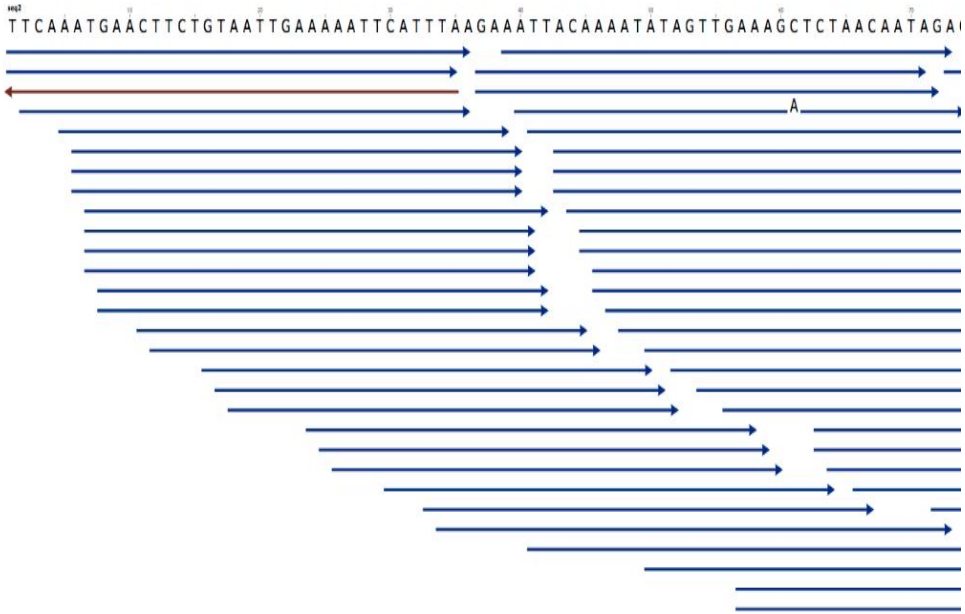
Background: Problem Statement

We have methods to read **short** sequences of DNA

We want to know where in the human genome they fall

They may not match exactly because of errors in reading and different genomes

This problem is called **Genome Alignment**



Background: Minhash

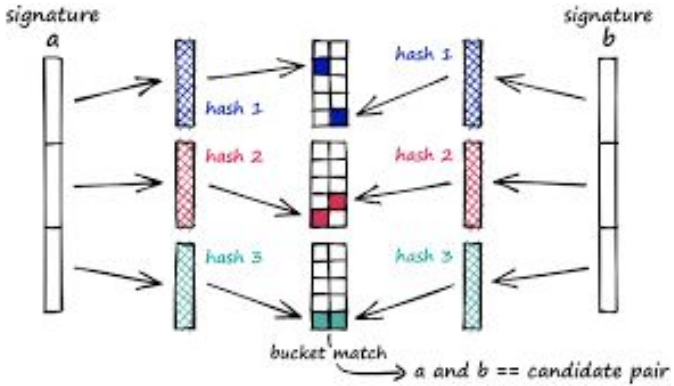
Computationally efficient way to find nearest neighbors is sparse boolean space

Robust to inexact matches

Idea: Compare the first element of shuffled **K-mers**

Kmers are very short sequences of characters we can use to analyze sequences as bags-of-words

We can use parallelism to speed up minhash



Data set

Reads.fq file

- Includes the name of a read, the read sequence, mark, and the quality of a read sequence
- We are only interested in the read sequence

Reference_chr21_... file

- Consists reference genomes split into 500 bins

Read_position_benchmark.csv file

- File to compare our results to

Methods: Technology

ACCRE

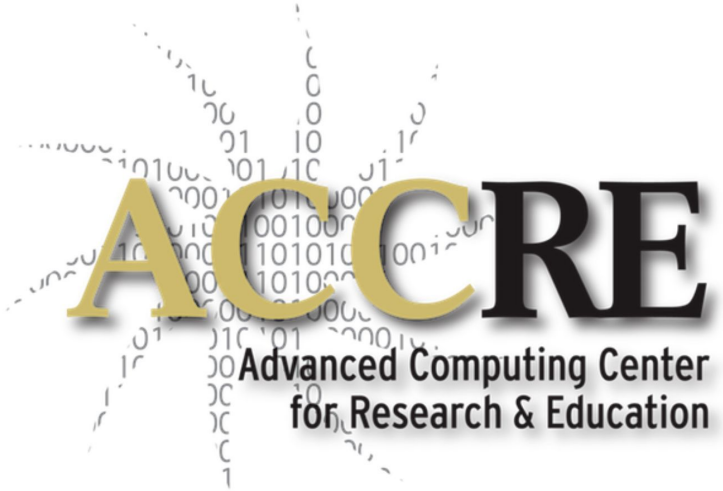
- 12 CPU cores (more CPU cores -> more likely for server to crash)

Jupyter Notebook

Python Multiprocessing

- Used 20 pools

-Numpy, Pandas



Methods: Split into K-mers

72,530 distinct 15-mers

Created a dataframe with 72,530 rows

```
distinct_kmers = set()
k = 15
for r in reads + bins:
    for i in range(len(r) - k+1):
        distinct_kmers.add(r[i:i+k])
distinct_kmers =
np.array(list(distinct_kmers))
```

	read0	read1
CCTTGGAAAAAGTTG	False	False
ACAGAGAAAACTGAG	False	False
GAAGTTGTCAAGTTT	False	False
CACACTCATACACAA	False	False
GGTGACATTTGAGTA	False	False
...	...	...
TGTATAGGCACATCA	False	False
CTGCAAGCTCCGCCT	False	False
TATTCAAGTTTATAG	False	False
CATTCAAGTTCTCCC	False	False

Methods: Get Minhash Signatures

Used 1000 hashes to generate signatures

Found a very simple Numpy solution:

```
rng = np.random.default_rng(seed = 5460)
hash_ = rng.permutation(len(distinct_kmers))
kmer_vector = reads_df['read0']
minhash = np.min(hash_[kvec])
```

Takes around 330 seconds with k = 15 and a signature length of 1000

Methods: Calculate Jaccard Similarities

We find the most likely bin by calculating the Jaccard similarities between read signatures and bin signatures:

```
def get_similarity(read_sigs, bin_sigs):
    sims = [np.mean(r == bin_sigs,axis=1) for r in read_sigs]
    return(sims)
```

	readname	bin	bin_start	sim
1	read1	bin20000000_20000100	20000000	0.33
501	read501	bin20011800_20011900	20011800	0.70
1001	read1001	bin20022700_20022800	20022700	0.22
1501	read1501	bin20033100_20033200	20033100	0.46

Experiments

Kmer length	Signature Length	Time (s)	MSE
3	10	0.262	3.30e+08
3	100	0.404	1.62e+08
3	1000	1.73	1.14e+08
15	10	4.65	2.69e+06
15	100	40.8	1,120
15	1000	332	896
40	10	3.33	1.70e+08
40	100	40.1	6.66e+07

Results

We compared our results (bin_start) to the reference file:

Pearson Correlation Coefficient: 0.999997

MSE: 896

RMSE: 29.93

Accuracy (off by <50): 95%

Note: Because the resolution of our minhash algorithm was only 100, whereas the reference alignment gave the exact start, the minimum MSE we could expect is 841

Conclusions

K-mers need to be the right size, not too small or too large

More hashes -> Higher accuracy but takes longer

Minhash is an efficient way to compare the similarity between two sets of information

Parallel processing is a great way to speed up the processing of large amounts of information

QUESTIONS?