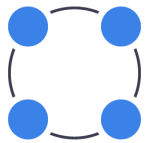
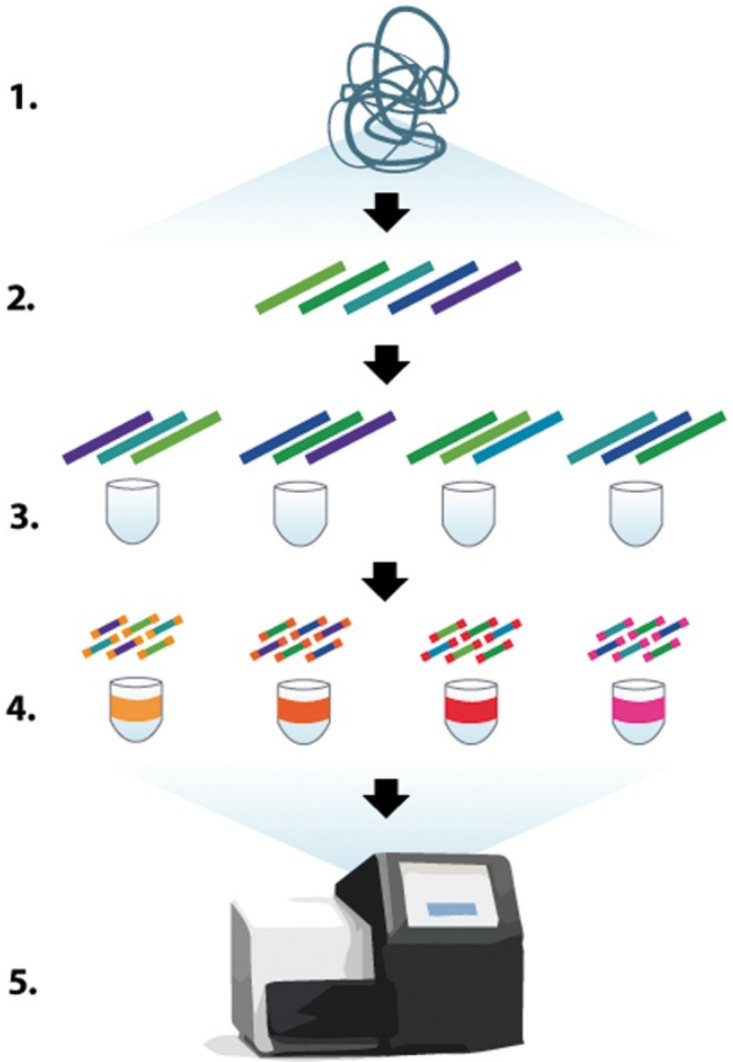
Speed-efficient data structures for cloudSPAdes

Evgeny Polevikov

Scientific advisors: Anton Bankevich, Ivan Tolstoganov
(Center for Algorithmic Biotechnology, St. Petersburg State University)

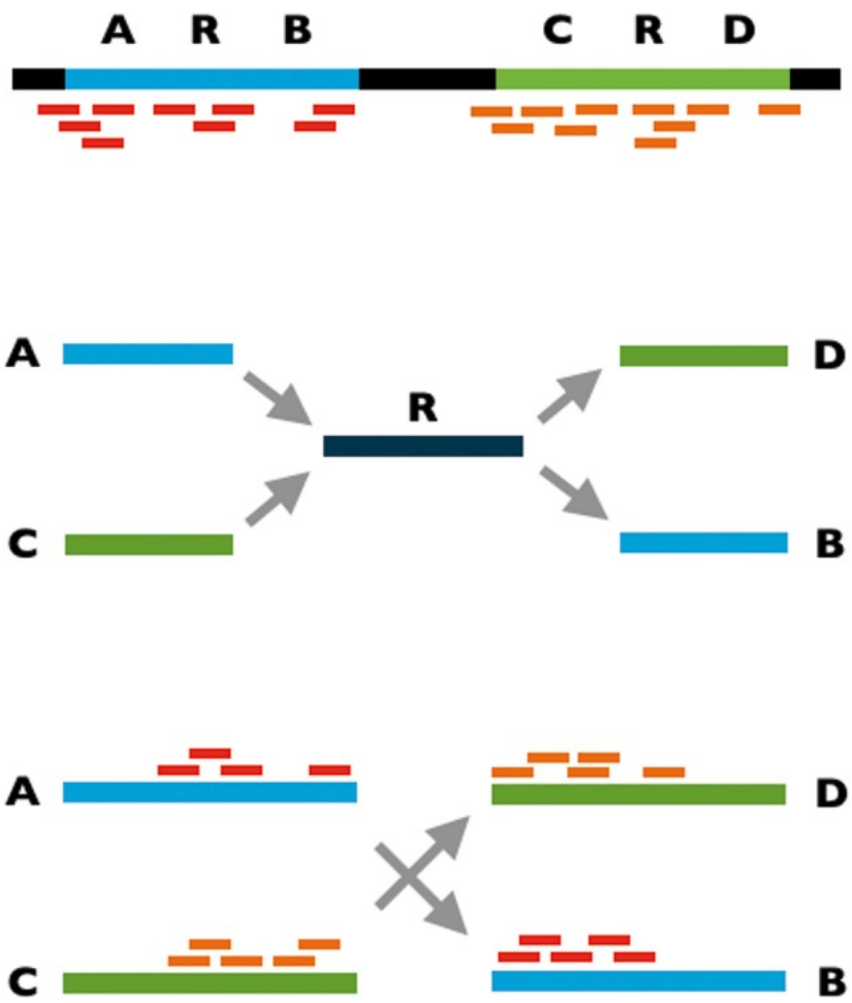


Genome assembly from synthetic long read (SLR) clouds



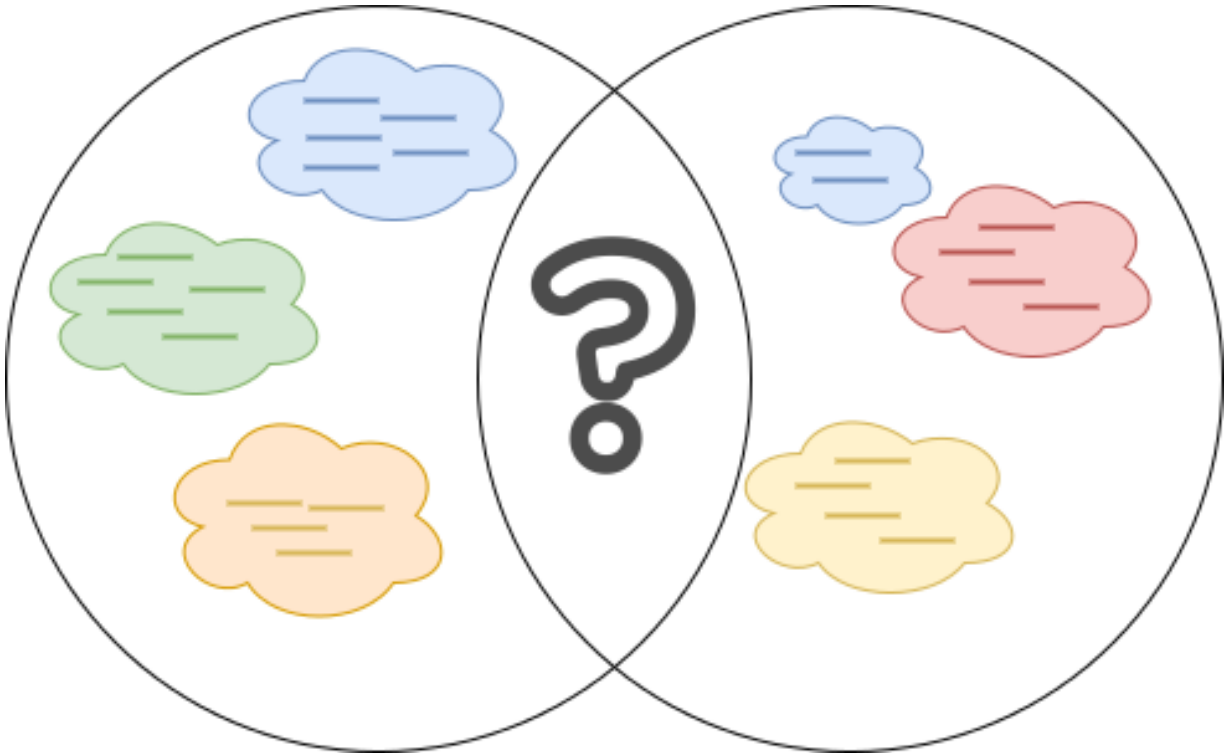
Kuleshov et al., Genome assembly from synthetic long read clouds, Bioinformatics (2016)

Resolving repeats using SLR clouds



Kuleshov et al., Genome assembly from synthetic long read clouds, Bioinformatics (2016)

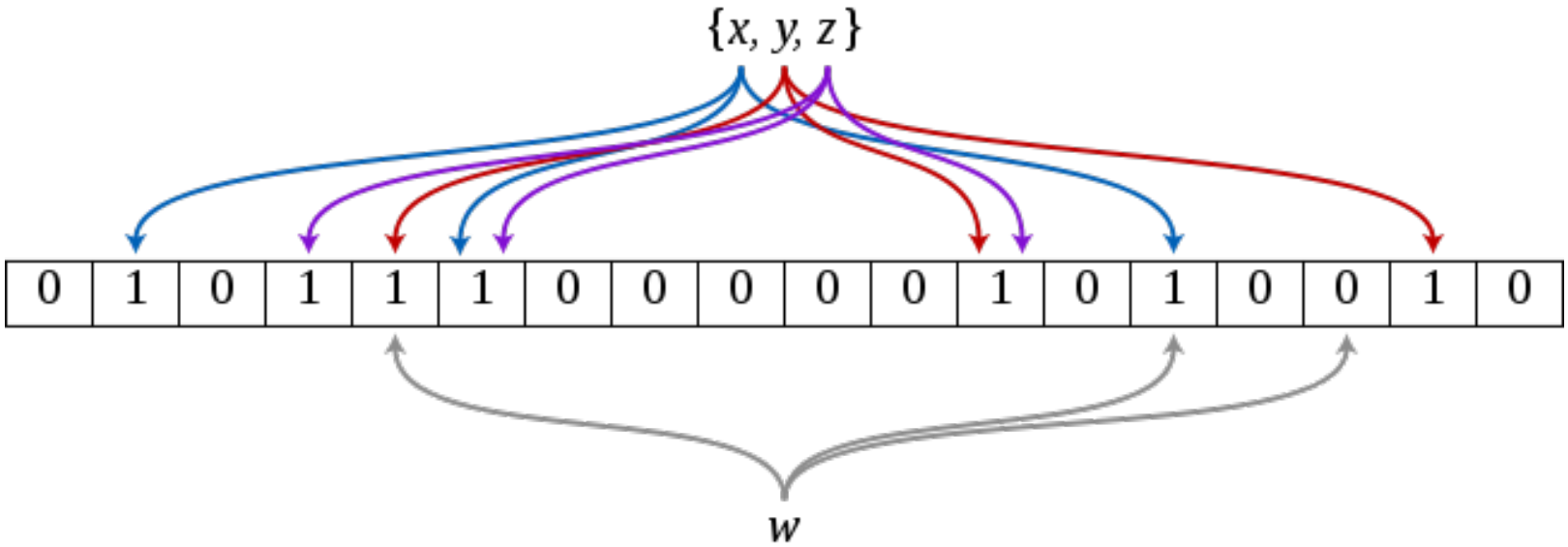
Resolving repeats using SLR clouds



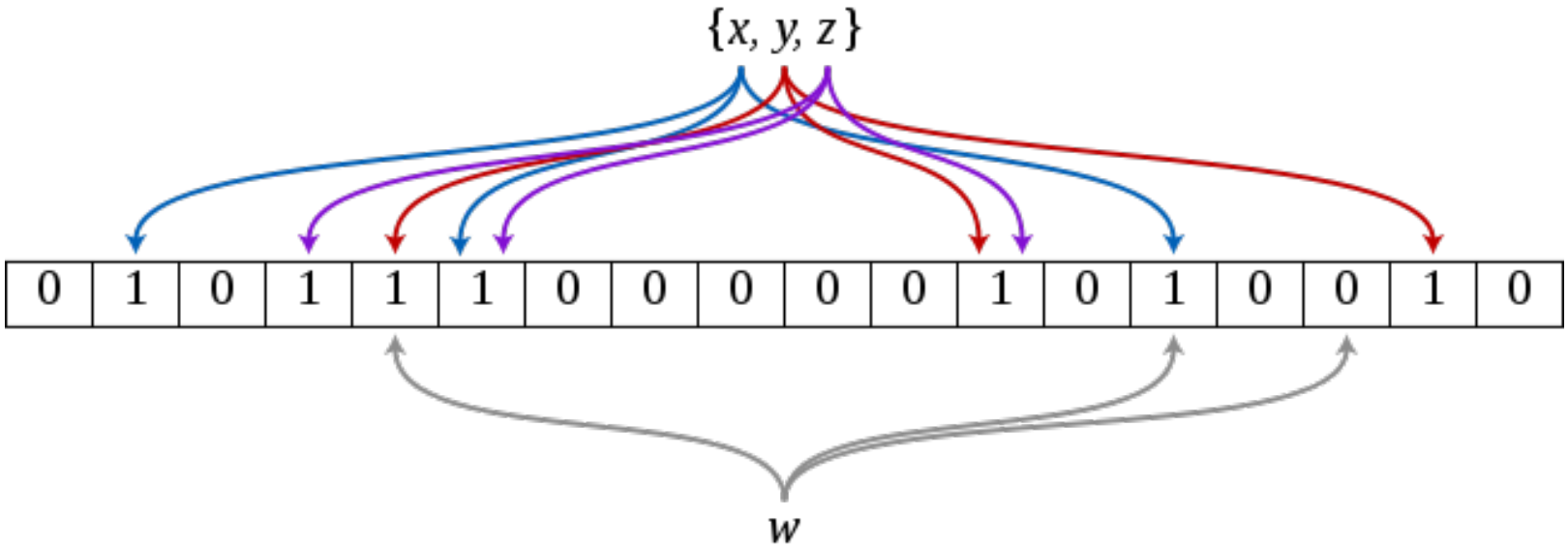
- We need to find the number of common elements in two sets of barcoded (colored) reads.
- An algorithm should be space and time efficient.

Bloom filter

The main idea: we store a bit array and use a set of hash functions, each of which map items to one of the array positions.



Bloom filter

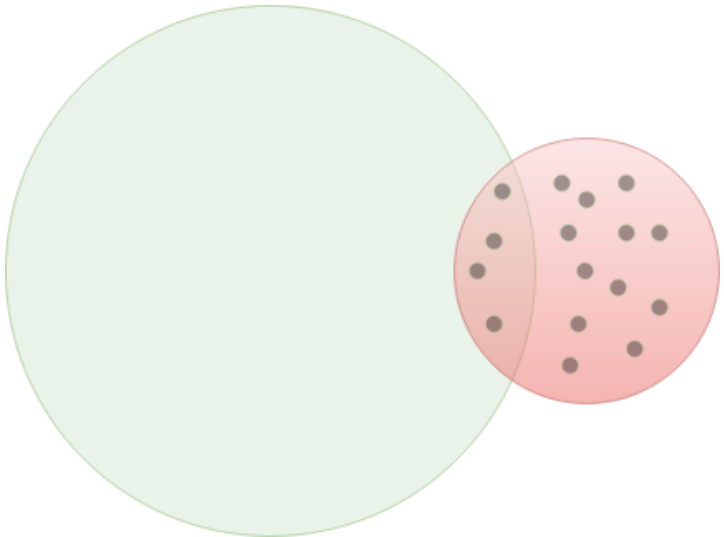


- Advantages: a bloom filter is a space efficient, it does not store the data items at all.
- Disadvantages: bloom filters do not allow to find an intersection of sets of different sizes efficiently and have false positive error.

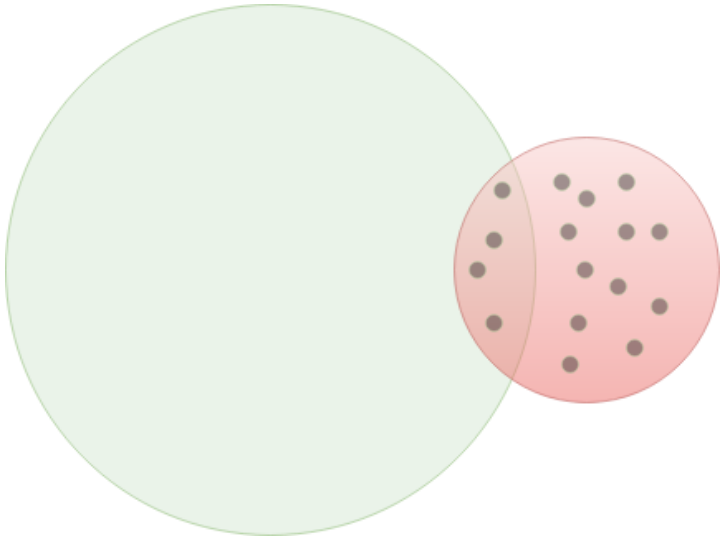
Containment Min Hash

The main idea:

1. Create a sample of elements from the set of a smaller size.
2. Create a bloom filter for elements from the set of a larger size.
3. Use the sample and the bloom filter in order to estimate an intersection size.



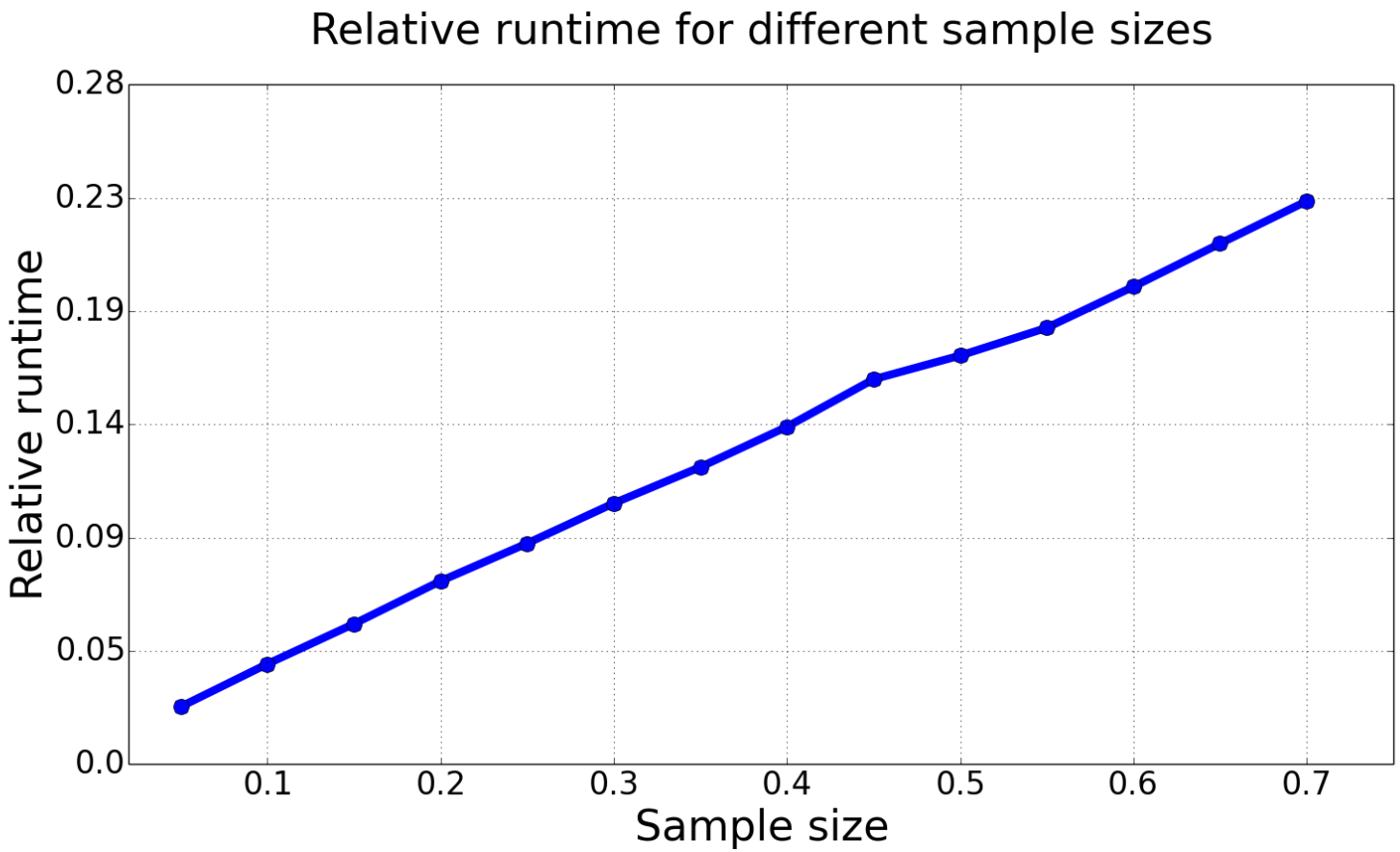
Containment Min Hash



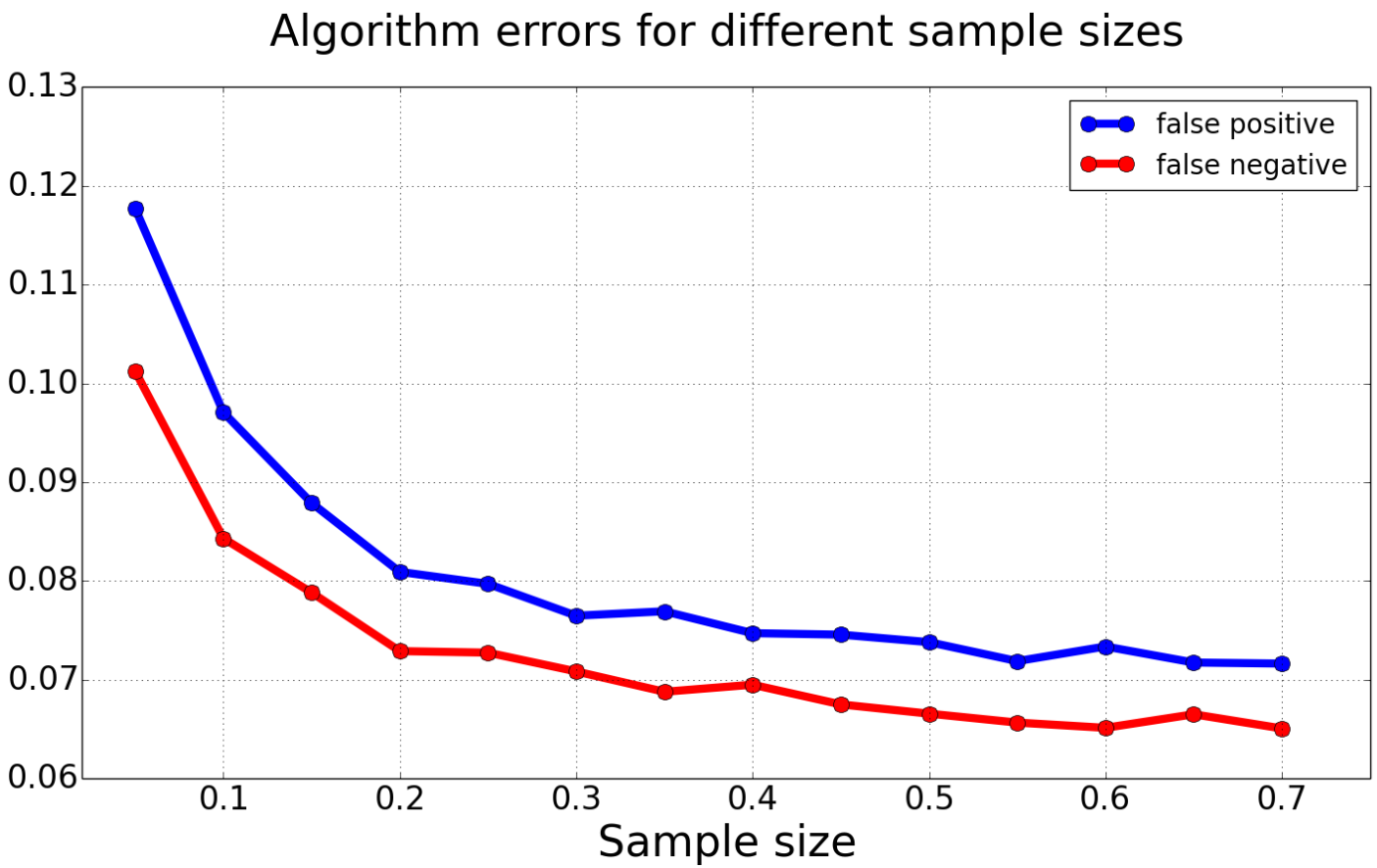
- Advantages: time and space efficient, works for sets of different sizes.
- Disadvantages: has false positive and false negative errors.

Algorithm runtime

We used a sample of edges (≈ 1500) and found an intersection for every ordered pair:

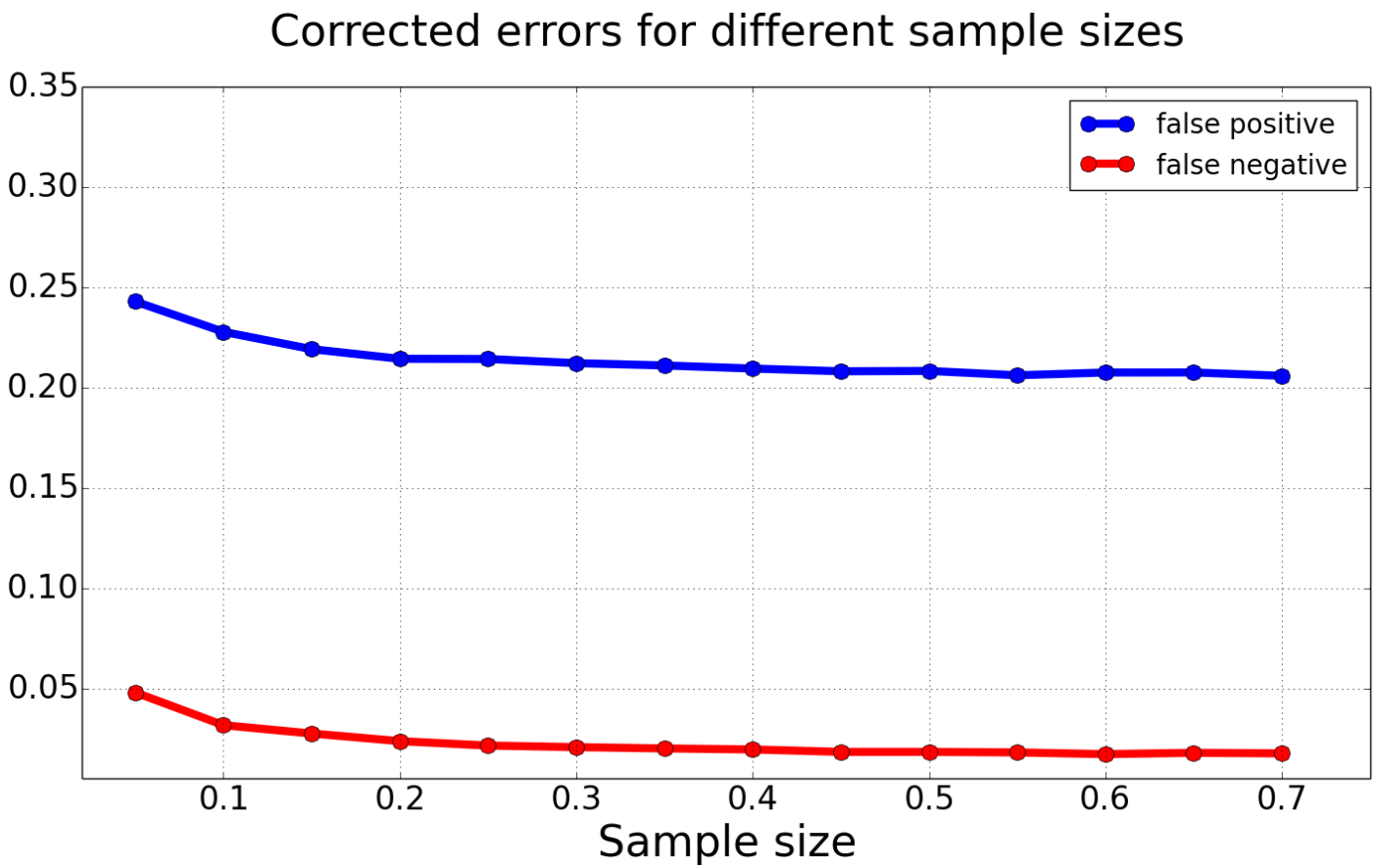


Algorithm errors



Corrected errors

We use lower threshold in order to decrease false positive error:



Memory consumption

- n – number of elements in the initial set
- p – desired false positive probability
- m – required number of bits for bloom filter

$$m = -\frac{nlnp}{(ln2)^2}$$

In our case $p = 10^{-4}$ and $m \approx 19.2n$, so we store about 60% of the initial data.

Results

- A bloom filter and containment min hash data structures were used in order to speed up the process of repeat resolving in cloudSPAdes.
- We managed to speed up the algorithm run time and lower memory consumption but our algorithm has false positive and false negative error.