Università
degli Studi
di Padova

A comparison of QCluster and isONclust

Clustering of short-read synthetic sequences

Alberto Schiabel

Algorithms for Bioinformatics • 11th February 2021

Outline

- Present QCluster and isONclust
- Describe the synthetic datasets used for the experiments
- Introduce the metrics chosen for cluster analysis
- Describe the experimental setup
- Explain the results
- On singletons and trivial clusters
- Suggest possible improvements
- Final considerations

QCluster and isONclust: Common Points

The similarities between QCluster and isONclust are:

- They perform strict partitioning clustering
- They leverage `FASTQ` quality values (*)
- They compute or exploit *a-priori* probabilities
- They're both based on k-mer statistics
- They let the user specify the size of k-mers

(*) Up to 70% of `FASTQ` files are metadata and quality metrics, rather than sequenced nucleotides

QCluster

- Developed in GNU `C++` at the University of Padova (2015)
- Based on `afcluster`, which uses `K-Means` (Lloyd)
  - Approximated and non-deterministic
  - It requires the number of clusters as an input
  - It allows executing multiple runs, returning just the best result
- It introduces a new D_2 distance based on `FASTQ` quality values
- It supports multiple distance metrics: L_2, χ^2, D_2^{*q}, ...
- It does not require alignments
- It uses quality value redistribution under the i.i.d. model
- It normalizes and centralizes the k-mer counts using AWP or AQP prior probability estimators

Paper: Clustering of reads with alignment-free measures and quality values
Authors: Matteo Comin, Andrea Leoni and Michele Schimd

isONclust

- Developed in `Python3` at Pennsylvania State University (2019)
- It doesn't support distance metrics, it's based on the concept of $minimizers$
- Uses a greedy clustering approach
  - Sorts s.t. longer sequences with higher quality values come first
  - One of the reads is fixed as representative of its cluster
  - $\forall\ k$-mer x, $H(x)$ returns all representatives that have x as minimizer
  - Clusters together candidates that share at least one minimizer with a given read, on the basis of *a-priori* probabilities over a sliding window w
  - Alignment-free in principle, but falls back to Smith-Waterman
- No possibility to specify the desired number of clusters
- It supports batch-parallelism

Paper: De novo clustering of long-read transcriptome data using a greedy, quality-value based algorithm
Authors: Kristoffer Sahlin and Paul Medvedev

Goals of the experiment

Choosing a subset of configuration parameters for QCluster and isONclust, we:

- Selected 100 distinct *originating* sequences the human cDNA assembly `GRCh38.p13` to generate 10000, 20000, and 50000 simulated sequences with fixed read lengths (100 and 700bp)
- Used QCluster and isONclust to cluster simulated reads according to their originating sequences, for every simulated dataset
  - The ground truth number of clusters is 100
- Repeated the process for every option in the range of the selected configuration parameters
- Showed that the clustering results are better than random partitions

Datasets (1/2)

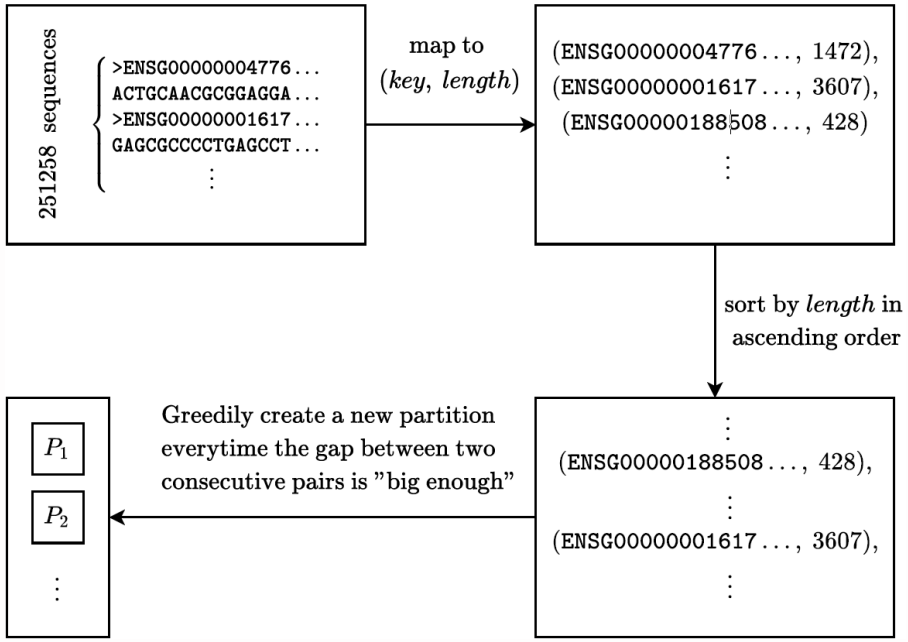
Genome assembly contains ~250000 sequences of very different and unbalanced sizes

1. We should select sufficiently large sequences of length $l \geq 1000$
2. The size differences between sequences should be negligible
3. We should have a sufficiently large pool of sequences that satisfy (1) and (2)

Datasets (1/2)

Genome assembly contains ~250000 sequences of very different and unbalanced sizes

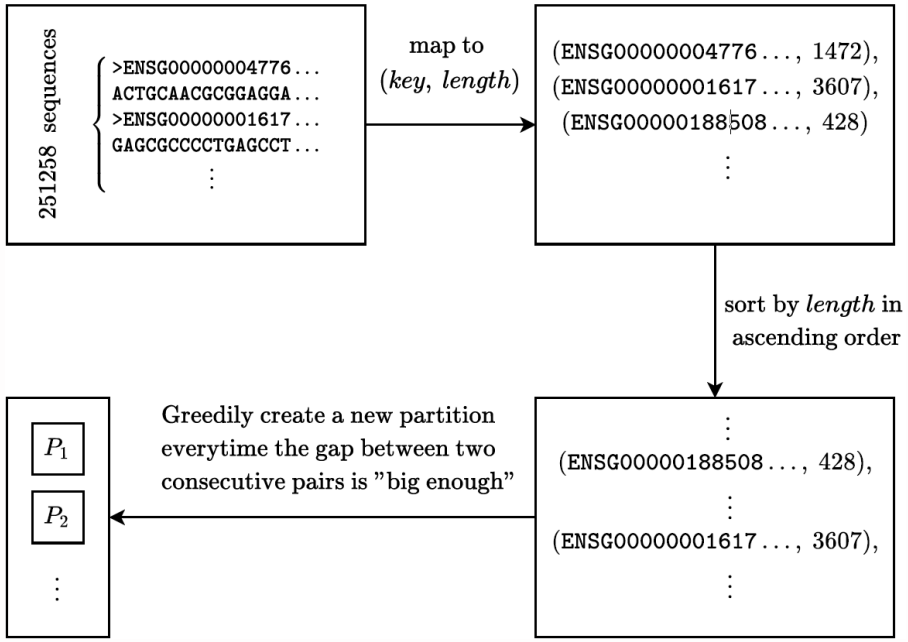
1. We should select sufficiently large sequences of length $l \geq 1000$
2. The size differences between sequences should be negligible
3. We should have a sufficiently large pool of sequences that satisfy (1) and (2)



Datasets (1/2)

Genome assembly contains ~250000 sequences of very different and unbalanced sizes

1. We should select sufficiently large sequences of length $l \geq 1000$
2. The size differences between sequences should be negligible
3. We should have a sufficiently large pool of sequences that satisfy (1) and (2)



$$\frac{length(s_j) - length(s_i)}{\sigma} > \alpha = 0.0005$$

	$\lvert P_i \rvert$	$\text{length}_{\text{min}}$	$\text{length}_{\text{max}}$	Gap
P_1	237799	41	4748	4707
P_2	1254	4750	4951	201
$\boldsymbol{P_3}$	**1107**	**5481**	**5572**	**91**
P_4	927	5165	5204	39

Datasets (2/2)

Starting from the 100 originating sequences, we created 6 synthetic datasets using SimLoRD with:

- Number n of sequences: 10000, 20000, 50000
- Fixed sequence length fl: 100, 700

E.g., we called the synthetic dataset with 10000 sequences of length 700 as `n-10000-fl-700`.

Datasets (2/2)

Starting from the 100 originating sequences, we created 6 synthetic datasets using SimLoRD with:

- Number n of sequences: 10000, 20000, 50000
- Fixed sequence length fl: 100, 700

E.g., we called the synthetic dataset with 10000 sequences of length 700 as `n-10000-fl-700`.

SimLoRD generated two files:

- A `simulated.fastq` file that contains the actual simulated long reads with the read error qualities
- A `simulated.sam` file that contains the alignment reference for the corresponding `fastq` file.

We kept the default SimLoRD error probabilities:

$$prob_{ins} = 11\% \qquad prob_{del} = 4\% \qquad prob_{sub} = 1\%$$

Clustering Metrics and Types

The greater the similarity within a cluster and the greater the difference between clusters, the better the clustering quality.

Clustering Metrics and Types

The greater the similarity within a cluster and the greater the difference between clusters, the better the clustering quality.

- **Homogeneity**: best when each cluster contains only members of a single class
- **Completeness**: best when all members of a given class are assigned to the same cluster
- **V-Measure**: harmonic mean between Homogeneity and Completeness
- **Purity**: extent to which a cluster contains only elements from a single class

Clustering Metrics and Types

The greater the similarity within a cluster and the greater the difference between clusters, the better the clustering quality.

- **Homogeneity**: best when each cluster contains only members of a single class
- **Completeness**: best when all members of a given class are assigned to the same cluster
- **V-Measure**: harmonic mean between Homogeneity and Completeness
- **Purity**: extent to which a cluster contains only elements from a single class

- **Adjusted Mutual Information**: information-theoretical interpretation *adjusted for chance*
- **Adjusted Rand Index**: measure of similarity between two clusterings *adjusted for chance*

Clustering Metrics and Types

The greater the similarity within a cluster and the greater the difference between clusters, the better the clustering quality.

- **Homogeneity**: best when each cluster contains only members of a single class
- **Completeness**: best when all members of a given class are assigned to the same cluster
- **V-Measure**: harmonic mean between Homogeneity and Completeness
- **Purity**: extent to which a cluster contains only elements from a single class

- **Adjusted Mutual Information**: information-theoretical interpretation *adjusted for chance*
- **Adjusted Rand Index**: measure of similarity between two clusterings *adjusted for chance*

We distinguished clustering types:

- **Trivial**: a cluster contains just 5 or less sequences
- **Singleton**: a cluster contains a single sequence
- The relevant clusters are non-trivial

Experimental Setup (1/3): Steps

Step 1: Save the clustering results in `CSV`/`TSV` files for each simulated dataset:

- Compute a baseline random clustering. Time: $\mathcal{O}(n)$, $n \in \{10000, 20000, 50000\}$
- Run QCluster fixing the desired number of clusters (`-c 100`), varying the k-mer length (`-k`) in the $[4,\ 9]$ range and using either the L_2, χ^2, or D_2^{*q} distances. We didn't use random seeds, we kept a single K-Means run and used the default AWP estimator.
- Run isONclust varying the k-mer length (`--k`) in the $[4,\ 11]$ range and using either 20 or 50 as window size (`--w`). We leveraged batch parallelism on two CPUs.

Step 2: Collect the relevant quality metrics and statistics for each clustering results

Step 3: Aggregate, analyse and visualise the relationships between the varying clustering software parameters and the collected metrics

Experimental Setup (2/3): Data collection

- Scripts and Python modules to run QCluster, isONclust and redirect their result to `TSV` files. They have two columns: read ID and assigned cluster ID
- Files are stored in a hierarchy on folders based on dataset name and parameter combination
- For each experiment, clustering metrics and statistics are stored along with software parameters
  - 3 single-row CSV files are produced for overall, trivial, and singleton clusters
  - A single-row CSV file is produced for overall, trivial, and singletong number of clusters

Experimental Setup (3/3): Data visualization

- Distinguish between clustering type (trivial / non-trivial)
- "Orthogonal" approach:
  - Use the range of two variables in the x and y axes
  - Discriminate by color according to a third variable, hue
  - Fix the other parameters, or compute on the average of the other parameters

Examples:

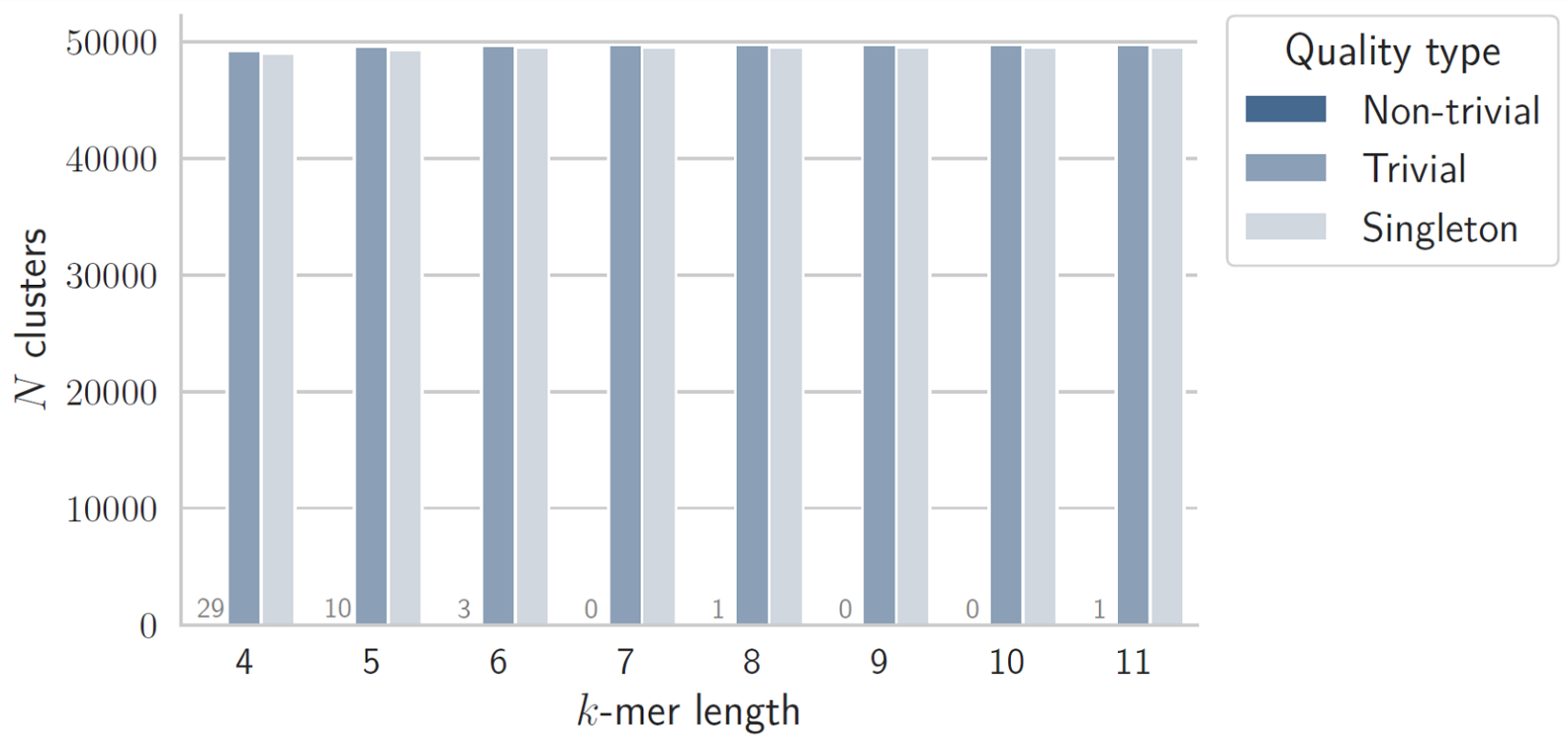
- Observe *clustering metric* as the k-mer length varies, discriminating by dataset
- Observe *clustering metric* as the number of sequences in the dataset varies, discriminating by read length, fixing a specific k
- Observe *clustering metric* as the read length varies, discriminating by QCluster's distance measure, fixing a specific k and dataset size

clusters: QCluster (non-trivial)

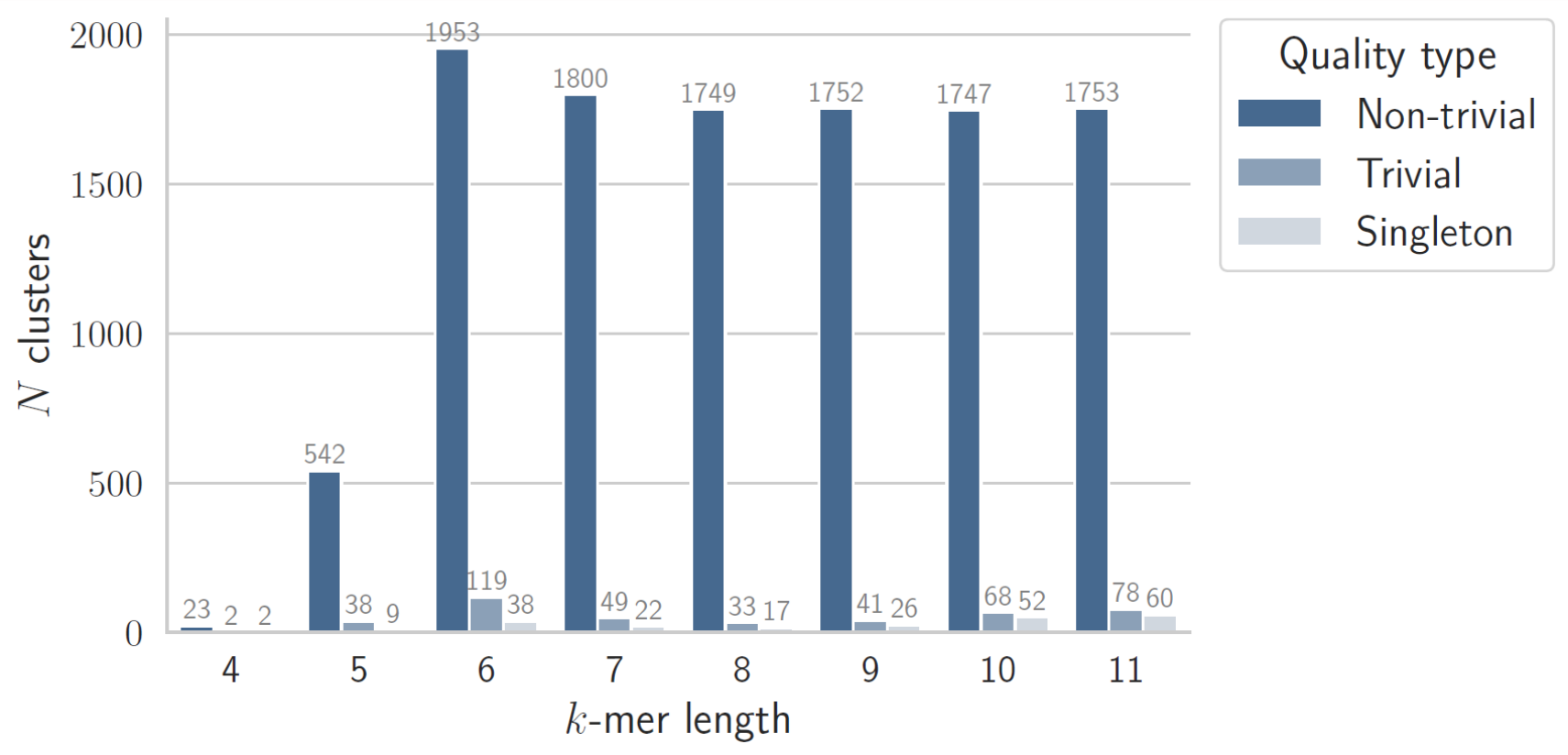
		Read length=100			Read length=700		
k	Distance	N=10000	N=20000	N=50000	N=10000	N=20000	N=50000
		# clusters			# clusters		
4	D_2^{*q}	98	98	**100**	**100**	99	**100**
	L_2	**100**	99	**100**	99	**100**	**100**
	X^2	98	**100**	99	99	**100**	**100**
5	D_2^{*q}	44	62	78	99	**100**	**100**
	L_2	46	67	96	98	**100**	**100**
	X^2	34	65	93	99	**100**	**100**
6	D_2^{*q}	3	21	84	97	**100**	**100**
	L_2	16	27	85	96	99	**100**
	X^2	6	34	88	98	**100**	**100**
7	D_2^{*q}	7	15	86	96	99	**100**
	L_2	6	23	0	96	**100**	**100**
	X^2	8	25	0	93	**100**	**100**
8	D_2^{*q}	4	0	0	92	0	0
	L_2	5	0	0	92	0	0
	X^2	7	0	0	95	0	0

Table 5 QCluster: Number of non-trivial clusters. Highest results are in bold.

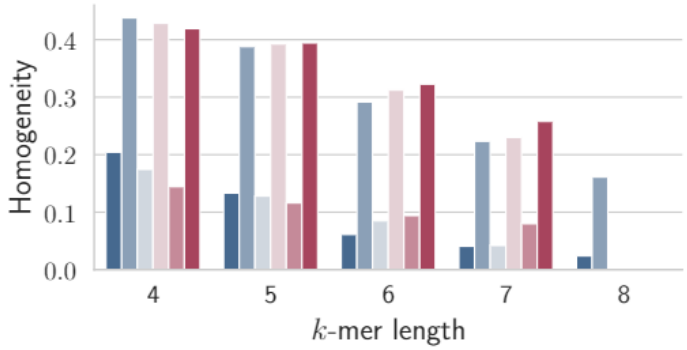
clusters: isONclust (n-50000-fl-100, w=50)

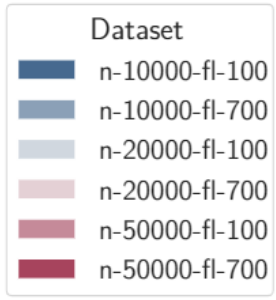


clusters: isONclust (n-50000-fl-700, w=50)

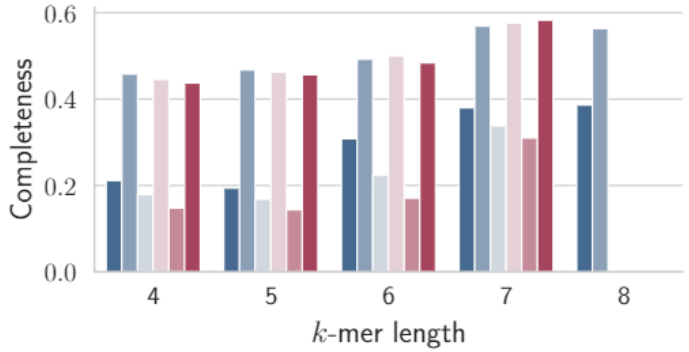


Metrics: QCluster (all clusters)

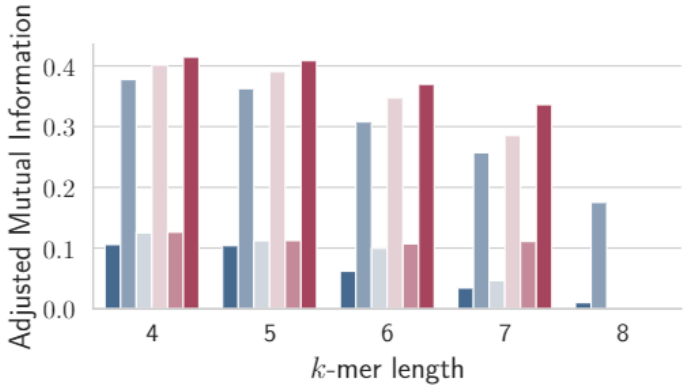


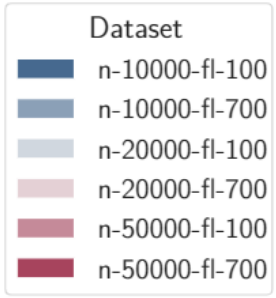


(a) Homogeneity vs k-mer length.

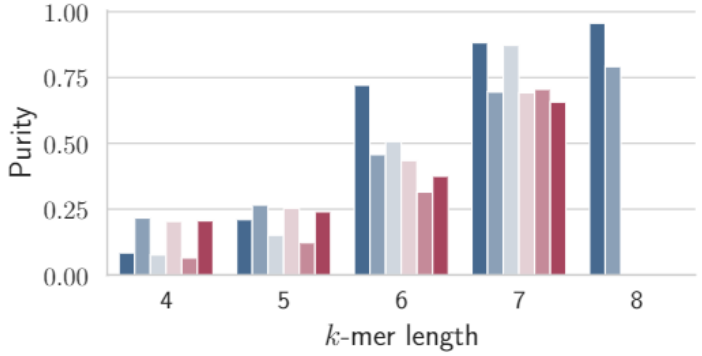


(b) Completeness vs k-mer length.



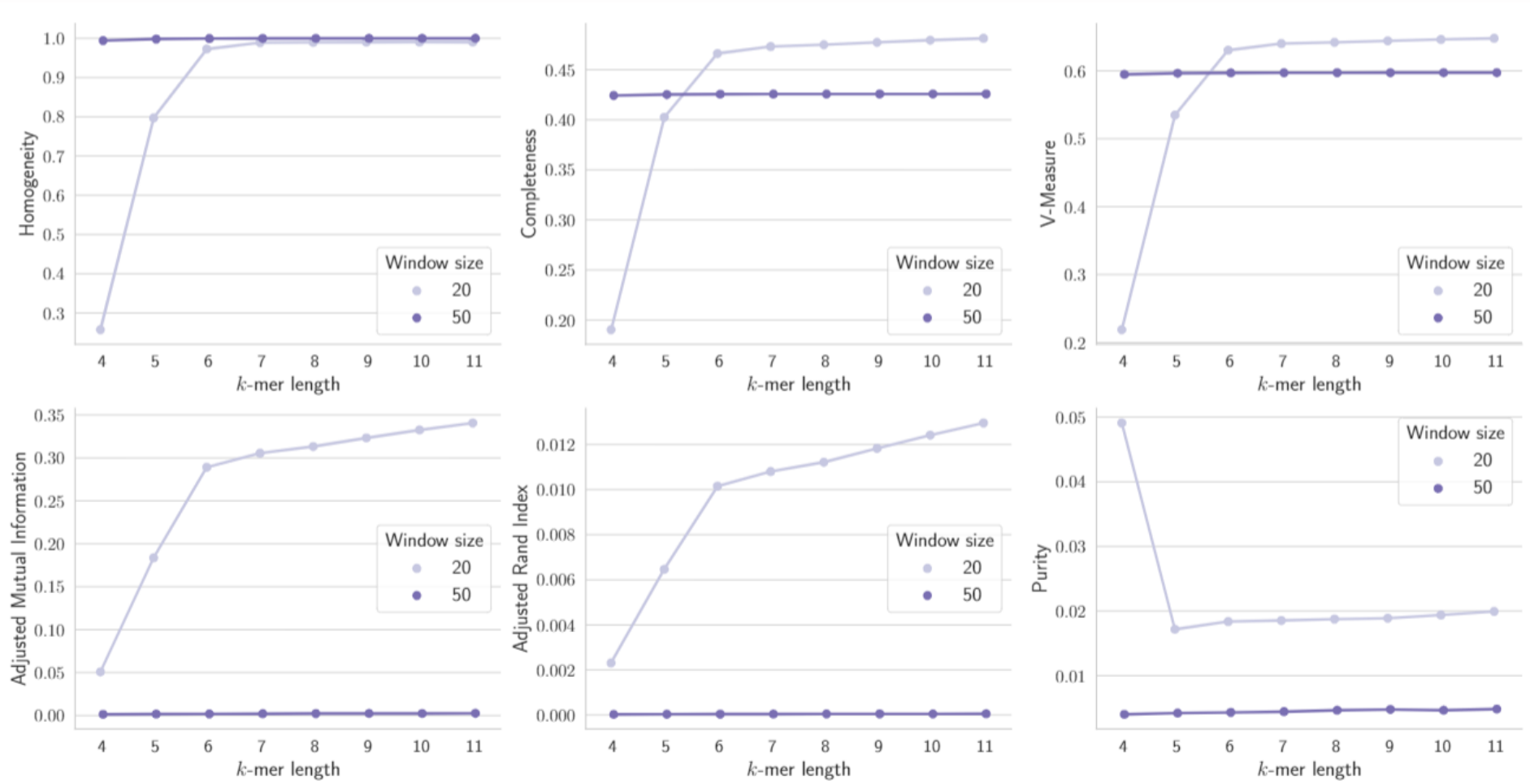


(d) Adjusted Mutual Information (AMI) vs k-mer length.

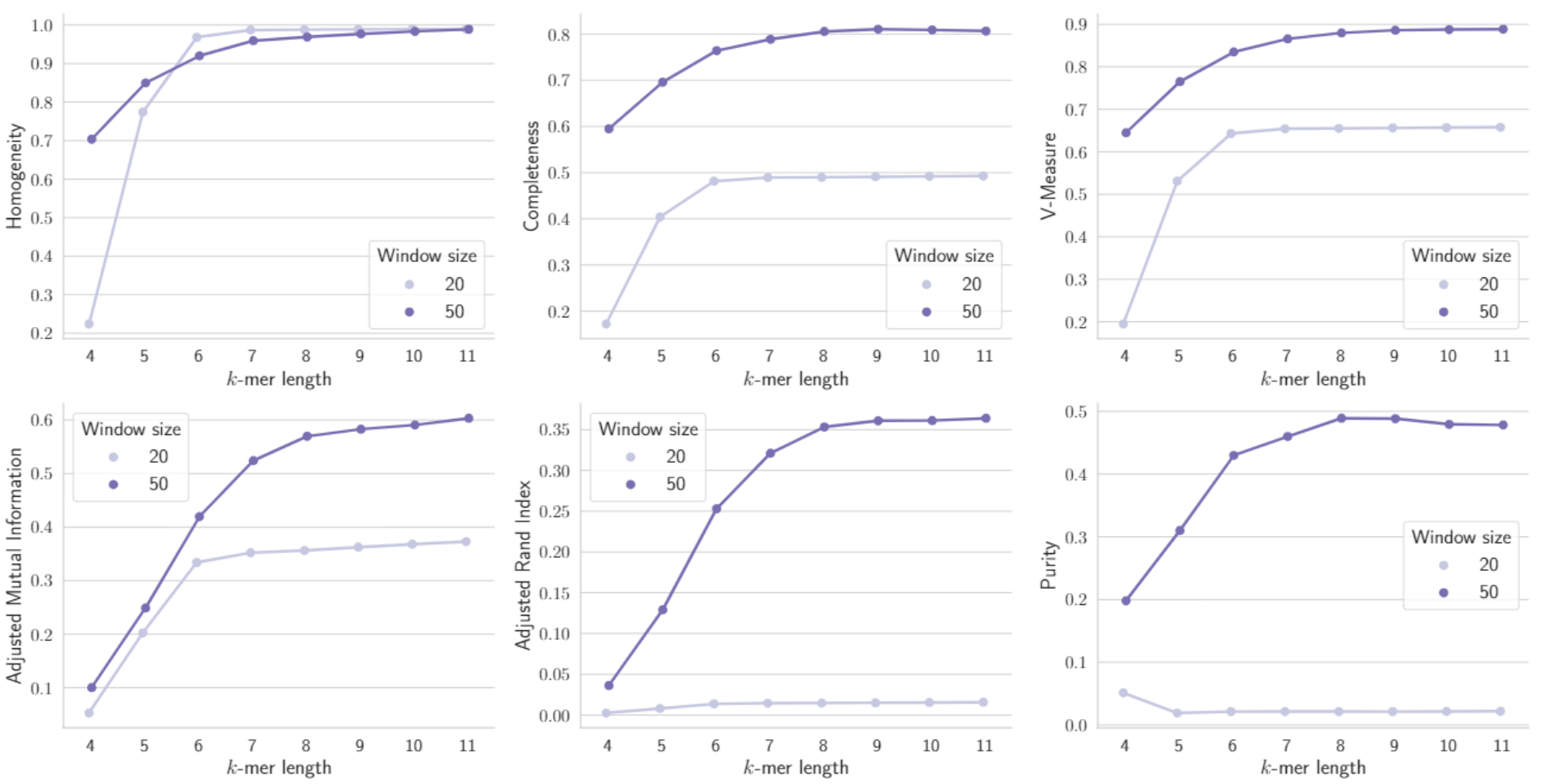


(f) Purity vs k-mer length.

Metrics: isONclust **(n-50000-fl-100, all clusters)**



Metrics: isONclust (n-50000-fl-100, no singleton)



Key Takeways

- QCluster is better when the number of clusters is known and the k-mer length is small
  - Shorter reads create denser non-trivial clusters with D_2^{*q} distance
  - D_2^{*q} consumes less memory than L_2 and χ^2, but the difference between distances isn't significant for short-reads when using AWP
  - We expect better results in practice via multiple runs of K-Means
  - Too slow and memory-intensive!

- isONclust is not well suited for very short reads, but scales extremely well to k-mer lengths
  - For larger reads, no clear winner between the two chosen window sizes, k is not that influent after $k \geq 6$
  - Too many clusters and singletons, but the non-trivial ones contain sibling sequences with high probability
  - Fast and reasonable memory consumption

Personal Suggestions

For QCluster:

- Make the software portable (it does not compile on Windows)
- Provide parallel and possibly distributed support with `k-means||`
- Add documentation to encourage open-source contributions to tackle the memory issues

For ISONCLUST:

- Give the possibility to provide the number of desired clusters
- Lock the third-party dependencies' versions to ensure reproducible builds
- Give more instructions on how to pre-compute probabilities for k-mer lengths outside of the $[4, 31]$ range

For both:

- Provide some structured documentation of the design from the software engineering perspective
- Be explicit about the well-known "gotchas"

Third-party tools used

- **Docker**
- **Pandas**
- **Seaborn**
- SimLoRD
- BioPython
- PySAM
- Scikit-Learn

Deliverables

Open-source Repository

The Python 3 project for our experiments is available on Github at the following link:

https://github.com/jkomyno/bioalgo-QCluster-vs-isONclust

Report

The written report for this project is available in the same repository, at this link. The report also contains the references used to complete this project.

Thank you for your attention

Questions?