1 DATASETS

This section provides metrics about each dataset. The *E. coli* original dataset had large coverage (> 200x) so we subsampled it dataset with seqtk[1] down to target 50x.

	C. elegans	*D. melanogaster*	*H. sapiens*	*E. coli* Nanopore	*E. coli* Pacbio
|sequences|	740774	1327569	1075867	25469	37404
Shortest sequence (b)	35	5	30	152	35
Mean Length (Kb)	10958	6827	6744	10110	6894
Median Length (Kb)	9822	4568	5089	5515	6672
N50 (Kb)	16572	11853	10568	20073	9064
N90 (Kb)	6502	3533	3537	4701	4218
L10 (sequences)	27013	29049	21703	400	1354
L50 (sequences)	191637	243356	220369	3807	10502
L90 (sequences)	480553	779045	652055	13969	25787
Coverage	81x	63x	29x	49x	49x
Accession	*	SRR6702603	PRJEB23027	SRR8494911	SRR8494940
Publication			Jain *et al.* (2018)	Maio *et al.* (2019)	Maio *et al.* (2019)

Table 1. Information and metrics about the dataset using in our evaluation of `yacrd` and `fpa`. * The *C. elegans* dataset come from Pacbio DevNet `https://github.com/PacificBiosciences/DevNet/wiki/C.-elegans-data-set`.

[1] `https://github.com/lh3/seqtk`

2 REPEATABILITY INFORMATION

2.1 `DASCRUBBER`

`DASCRUBBER` commit number `0e90524` was used, and a custom pipeline was built using `DASCRUBBBER-wrapper`[2] as an inspiration, as well as recommendations from the authors: `https://github.com/thegenemyers/DASCRUBBER/issues/7` and `https://github.com/thegenemyers/DASCRUBBER/issues/20`. See below (section 2.9) for the URL of the custom pipeline.

2.2 `MiniScrub`

We use version of commit `3d11d3e`. We did not run `MiniScrub` in GPU mode so we followed te authors instructions for installation and run `https://bitbucket.org/berkeleylab/jgi-miniscrub/`.

2.3 `yacrd`

We use version 0.5.1.

2.4 `fpa`

We use version 0.5.

2.5 `bwa-mem`

We use version 0.7.17-r1188.

2.6 `Minimap2`

We use version 2.16-r922.

2.7 `Wtdbg2`

We use version of commit `8908a31`.

2.8 QUAST

We use version v5.0.2.

2.9 Script and reproduction of analysis

All information to repeat our analysis can be found at this address
`https://gitlab.inria.fr/pmarijon/yacrd-and-fpa-upstream-tools-for-lr-genome-assembly`

[2] `https://github.com/rrwick/DASCRUBBER-wrapper`

3 YACRD PARAMETER OPTIMISATION

yacrd is very dependent on the mechanism used to find common regions between reads. We rely on Minimap2 for this task. Minimap2 is based on short sequence seeds to find common regions between reads. In all-against-all alignment, it takes as parameter a distance between two seeds (-g, default: 10,000 bases). In yacrd we assume that regions with low seed coverage have low quality, and therefore need to be scrubbed. Yet with the default seed distance, it may happen that Minimap2 finds two consecutive seeds that correspond to two "good" read regions separated by one "bad" read region. Therefore this parameter needs to be tuned.

Another important parameter is the read coverage threshold to consider that a read region is of sufficient quality (yacrd parameter -c).

We have changed these two parameters as follows: i) the maximum distance between the two seeds from 50 to 2450 with a step of 100, ii) the minimum coverage before eliminating the region from 1 to 15 with a step of 1.

We evaluated the influence of these parameters on several metrics:

- Number of chimeric reads
- Number of reads
- Number of bases
- And in Miniasm and Wtdbg2 assemblies,
  - NGA50
  - Total length
  - Number of contigs
  - Number of indels per 100 kpb
  - Number of mismatches per 100 kbp

We ran this evaluation on *H. sapiens*, *C. elegans* and *E. coli* Pacbio dataset. The raw data is available in:

- *H. sapiens* (ONT ultra-long R9.4): https://gitlab.inria.fr/pmarijon/yacrd-and-fpa-upstream-tools-for-lr-genome-as raw/master/data/yacrd_parameter_test_h_sapiens_ont.csv
- *C. elegans* (Pacbio P6-C4): https://gitlab.inria.fr/pmarijon/yacrd-and-fpa-upstream-tools-for-lr-genome-assemb blob/master/data/yacrd_parameter_test_c_elegans_pb.csv
- *E. coli* (Pacbio Sequel): https://gitlab.inria.fr/pmarijon/yacrd-and-fpa-upstream-tools-for-lr-genome-assembly/ blob/master/data/yacrd_parameter_test_e_coli_pb.csv

For *H. sapiens* Nanopore dataset we find that a value of 500 for the -g parameter and 4 for the -c parameter optimizes the number of contigs in Miniasm assembly and NGA50, and remains reasonable across the other metrics. We therefore recommend to use this value for Nanopore data and we used it in all of our results.

For *C. elegans* PacBio dataset P6-C4, using a similar reasoning, optimal values are different and are 800 for the -g parameter and 4 for the -c parameter.

For *E. coli* PacBio Sequel dataset, using similar reasoning, optimal values are different and are 5000 for the -g parameter and 3 -c parameter.

We therefore used the above values for all datasets obtained with the same sequencing technology.

4 MAPPING OF SCRUBBED READS

To compute quality metrics, for each dataset we mapped both scrubbed and raw reads against their respective reference genomes with `bwa-mem` (we used `ont2d` preset for Nanopore reads, and `pacbio` preset for Pacbio reads). The mapping results were analyzed using a custom Python script[3] which reports the number of mapped reads, the sum of edit distances between each read and the matching reference sequence, the sum of positions of the genome mapped by a read, and the error rate.

To count the number of chimeric reads for each dataset, we remapped reads against each reference genome with `Minimap2` (we used `map-ont` preset for Nanopore reads, and `map-pb` preset for Pacbio reads). We analyzed the PAF (Pairwise Alignment Format) file outputted by `Minimap2` with a custom Python script[4]. This script parses a PAF file and associates to each read a list of pairs of starting/ending mapping positions. For each read, if two pairs of positions overlap in the corresponding list, they are merged. If, after merging, there remains more than one pair of positions, the read is marked as chimeric. To manage circular genomes we ignore reads with mapping positions near to the beginning/ending of the genome (within a distance of $\texttt{reference length} - 0.1 \times \texttt{reference length}$ from the beginning/ending).

Table 2 shows that scrubbing reduces the number of reads and the number of bases mapped against the reference, but the error rate is reduced too (at least 1% for `yacrd` and at least 2% for `DASCRUBBER`) and the number of chimeric reads was reduced by two or more.

		`bwa-mem`				`Minimap2`
Dataset	Scrubber	# reads mapped	Edit distance	Mapping length	Error rate	# chimeric reads
C. elegans	raw	643138	903621479	6542507928	13.8115	71704
	yacrd	575517	758579062	5932958881	12.7858	15157
	dascrubber	576467	700895648	6128772910	11.4361	9285
D. melanogaster	raw	954622	1238009380	7353191408	16.8364	59864
	yacrd	843483	968115342	6468730379	14.9661	28076
	dascrubber	792138	857944894	6543861920	13.1107	24826
H. sapiens	raw	808709	1274720337	6053626797	21.0571	25888
	yacrd	698139	929843201	4889850725	19.0158	5216
	dascrubber	615789	813646386	4823555914	16.8682	1640
E. coli Nanopore	raw	19873	36411589	232858822	15.6368	351
	yacrd	18790	29819875	207863123	14.3459	64
	dascrubber	18275	29216052	223383847	13.0789	50
	miniscrub	24242	15740209	136642860	11.5192	58
E. coli Pacbio	raw	31945	29162389	175640234	16.6035	7374
	yacrd	24728	22150527	146552898	15.1143	15157
	dascrubber	26883	20315636	158261992	12.8367	63
	miniscrub	10304	3050308	32249990	9.4583	37

Table 2. Statistics of reads mapped to their respective reference, before and after scrubbing.

[3] `https://gitlab.inria.fr/pmarijon/optimizing-early-steps-of-lr-assembly/blob/master/script/get_mapping_info.py`

[4] `https://gitlab.inria.fr/pmarijon/optimizing-early-steps-of-lr-assembly/blob/master/script/found_chimera.py`

5 QUALITY OF ASSEMBLY

To assess the quality of assemblies with and without scrubbing, we ran both `Miniasm` and `Wtdbg2` from scrubbed reads and raw reads with recommended parameters for each sequencing technology. After assembly we ran QUAST with parameter `--min-identity 80.0`.

Table 3 shows a summary of outputted metrics for `Miniasm`. Scrubbing increases both the NGA50 and the length of the largest alignment. The size of the largest contig is often decreased but the contigs quality seems better as the number of misassemblies decreases. Finally the number of indels and mismatches per 100kb are quite stable. We thus observe that scrubbing improves assembly metrics, `yacrd` and `DASCRUBBER` having similar results, better than `MiniScrub`.

Table 4 shows a summary of outputted metrics for `Wtdbg2`. Contrarily to `Miniasm`, NGA50 is not always improved by scrubbing. The size of the largest contig increases while the number of misassemblies decreases. This could be interpreted as a better assembly. Regarding these two metrics, `yacrd` has better results than `DASCRUBBER`.

Dataset	Scrubber	#contigs	NGA50	Largest contig	Largest alignment	Asm/Ref length	Indels per 100kb	Mismatches per 100kb	# mis-assemblies
C. elegans (Pacbio P6-C4)	yacrd	0.68	1.02	1.48	1.1	1.1	0.96	0.94	0.7 2
	dascrubber	0.58	1.26	0.97	1.55	1.08	0.94	0.9	0.54
D. melanogaster (ONT Minion)	yacrd	0.8	1.57	0.64	1.27	0.93	0.96	0.96	0.65
	dascrubber	0.84	2.41	1.45	2.48	0.96	0.96	0.94	0.83
H. sapiens (ONT ultra-long R9.4)	yacrd	2.14	4.72	0.38	5.19	1.41	0.97	0.95	0.25
	dascrubber	2.78	4.26	0.32	4.48	1.36	0.97	0.95	0.12
E. coli (ONT Minion)	yacrd	1	2.6	1	2.43	0.99	0.96	0.95	0.6
	dascrubber	1	2.65	1	2.48	0.99	0.97	0.96	0.6
	miniscrub	9	0.46	0.62	1.6	0.98	0.83	0.77	0.8
E. coli (Pacbio Sequel)	yacrd	1	1.96	0.96	1	1.02	0.99	0.99	0.63
	dascrubber	0.75	2.73	2.55	2.36	1.02	0.99	1.01	0.36

(a) Ratio of assembly metrics after scrubbing on assembly without scrubbing. Column Asm/Ref length report the total length of assembly against reference length, not against raw assembly length.

Dataset	Scrubber	#contigs	NGA50	Largest contig	Largest alignment	Total length	Indels per 100kb	Mismatches per 100kb	# mis-assemblies
C. elegans (Pacbio P6-C4)	raw	226	432112	5422030	1231264	114194187	7842.91	1944.78	1396
	yacrd	154	440776	8039734	1362861	110987109	7587.54	1827.39	1015
	dascrubber	131	544677	5262439	1907915	108636024	7405.35	1744.85	754
D. melanogaster (ONT Minion)	raw	423	423007	8745435	2396453	138733599	5789.82	4233.35	2126
	yacrd	339	664130	5559421	3053469	134302689	5587.09	4044.89	1375
	dascrubber	357	1018097	12708694	5953687	137569022	5537.66	3988.95	1765
H. sapiens (ONT ultra-long R9.4)	raw	184	96225	15987693	857015	202082384	6554.02	4089.56	1745
	yacrd	394	453748	6008000	4444926	203039148	6366.5	3891.98	432
	dascrubber	512	410370	5041373	3837755	195781855	6377.04	3887.84	209
E. coli (ONT Minion)	raw	1	1450762	5147604	1553466	5147604	5279.79	4341.81	5
	yacrd	1	3775907	5161073	3775907	5161073	5083.69	4104.31	3
	dascrubber	1	3850663	5168753	3850663	5168753	5113.64	4160.78	3
	miniscrub	9	670066	3172759	2478579	5136537	4382.29	3337.49	4
E. coli (Pacbio Sequel)	raw	4	499610	1974889	1083557	5417095	8011.42	1856.96	11
	yacrd	4	983113	1910204	1089886	5345453	7974.73	1856.19	7
	dascrubber	3	1362738	5042223	2552164	5331569	7963.32	1870.92	4

(b) Exact value of assembly metrics without scrubbing and with scrubbing

Table 3. `Miniasm` assembly statistics.

Dataset	Scrubber	#contigs	NGA50	Largest contig	Largest alignment	Asm/Ref length	Indels per 100kb	Mismatches per 100kb	# mis-assemblies
C. elegans (Pacbio P6-C4)	yacrd	0.87	1.05	1.09	1	1.06	0.93	0.93	0.93
	dascrubber	0.72	1.02	1.04	1.11	0.98	0.9	0.7	0.35
D. melanogaster (ONT Minion)	yacrd	0.51	1.02	1.09	0.98	0.93	0.84	0.57	0.43
	dascrubber	0.61	0.87	0.83	0.98	0.93	0.85	0.62	0.41
H. sapiens (ONT ultra-long R9.4)	yacrd	0.6	0.98	11.68	1	0.97	0.94	0.86	0.44
	dascrubber	0.61	0.36	2.97	0.49	0.92	0.9	0.82	0.13
E. coli (ONT Minion)	yacrd	0.56	1.57	1.7	1.7	1.02	1.01	1	0
	dascrubber	1.11	0.81	0.64	0.57	0.87	1.06	1.18	1.5
	miniscrub	1	1.65	1.13	1.13	0.95	1.2	1.77	2
E. coli (Pacbio RS II)	yacrd	0.27	5.49	1.65	2.42	0.98	1.55	1.99	1.5
	dascrubber	0.64	1.14	0.43	0.67	1.01	1.24	1.46	0.75

(a) Ratio of assembly metrics after scrubbing on assembly without scrubbing. Column Asm/Ref length report the total length of assembly against reference length, not against raw assembly length.

Dataset	Scrubber	#contigs	NGA50	Largest contig	Largest alignment	Total length	Indels per 100kb	Mismatches per 100kb	# mis-assemblies
C. elegans (Pacbio P6-C4)	raw	139	565278	6301328	1880328	106873707	212.25	114.82	1396
	yacrd	122	593039	6919398	1880831	106276350	106,89	198,35	577
	dascrubber	100	578041	6577520	2084274	105265557	191.21	79.93	485
D. melanogaster (ONT Minion)	raw	945	1274655	22883959	5747639	144439108	1589.69	523.13	3938
	yacrd	484	1305125	24923636	5624012	135024912	1331.34	298.73	1675
	dascrubber	578	1114519	18994352	5625082	134142906	1348.97	324.48	1633
H. sapiens (ONT ultra-long R9.4)	raw	810	1513450	2435917	9247318	217462699	3588.91	368.93	1316
	yacrd	485	1482513	28462688	9268500	210552669	3370.08	318.69	582
	dascrubber	496	545902	7234785	4524362	200220997	3224.69	302.44	177
E. coli (ONT Minion)	raw	9	678871	1434432	1432545	5045762	767.87	156.21	2
	yacrd	5	1068201	2435917	2434921	5133519	778.16	155.46	0
	dascrubber	10	546569	917645	821696	4395460	817.43	184.84	3
	miniscrub	9	1117217	1621361	1619652	4773046	924.23	275.84	4
E. coli (Pacbio RS II)	raw	11	583235	2474045	1323293	5021940	170.49	46.03	4
	yacrd	3	3207692	4100960	3207692	5134707	264,26	91,77	3
	dascrubber	7	664896	1075736	892884	5093533	211.5	67.38	3

(b) Exact value of assembly metrics without scrubbing and with scrubbing

Table 4. `Wtdbg2` assembly statistics.

6 FPA

To evaluate `fpa`, we ran two different pipelines. The first one uses directly `Miniasm` without `fpa` and with recommended parameters. The second one runs `fpa` to filter out reads (`Minimap2` output is piped to `fpa` directly) before running `Miniasm` on filtered reads with recommended parameters. Using `fpa` we removed `internal match` and overlap shorter than 2000 (options `drop -i -l 2000`). This sort of overlap is ignored by `Miniasm` during the assembly step but is used during the read filtering step.

Table 5 shows the impact of using `fpa` on time, memory and assembly metrics. Using `fpa` decreases both disk usage and total computation time of downstream analysis while having no impact or a positive one on assembly metrics. Usage of `fpa` does not radically affect mapping wall-clock time and memory usage, but it reduces by 13% to 67% the memory usage and CPU time of the assembly step (the computation time of `fpa` is included in the mapping time). Moreover the size of the PAF file produced by `Minimap2` is reduced by 40% to 79 %.

	Dataset	*C. elegans*		*D. melanogaster*		*H. sapiens* chr 1		*E. coli* Nanopore		*E. coli* Pacbio	
	Pipeline	w/o `fpa`	`fpa`	w/o `fpa`	`fpa`	w/o `fpa`	`fpa`	w/o `fpa`	`fpa`	w/o `fpa`	`fpa`
Time (s)	Mapping	3296	3247	3510	3659	1570	1558	26	29	23	24
	Assembly	297	139	782	186	103	50	4	2	2	1
	Total	3593	3386	4292	3845	1673	1608	30	31	25	25
Memory	Mapping (GB)	51	51	53	54	41	40	3	3	4	4
	Assembly (Mbp)	4788	2594	13836	5335	1797	587	52	45	33	22
	PAF size	32G	9.5G	54G	11G	8.9G	3.2G	141M	82M	85M	38M
Assembly	# contigs	168	150	423	381	184	216	5	5	8	9
	NGA50	407821	438055	423007	455307	96225	106259	1450762	1246808	562741	292111
	# misassemblies	1212	1149	2126	1840	1745	1502	5	5	8	9
	length	112248122	111641079	138733599	136623341	202082384	198386315	5147604	5283927	5394119	5395896
per 100kb	# mismatches	1893.44	1854.95	4233.35	4190.43	4089.56	4065.95	4341.81	4425.24	1862.72	1841.66
	# indels	7700.42	7628.39	5789.82	5742.05	6554.02	6534.92	5279.79	5376.03	7968.63	7945.11

Table 5. Impact of `fpa` on assembly using `Miniasm`.

7 COMBINATION OF `yacrd` AND `fpa`

To evaluate the effect of running both `yacrd` and `fpa`, we ran two different pipelines. The first one uses a standard `Miniasm` pipeline (called 'basic'): `Minimap2` plus `Miniasm` with recommended parameters. The second one (called 'extended') runs `yacrd` with best parameters for each dataset, then `Minimap2` with recommend parameter on scrubbed reads and pipes the results in `fpa` to filter out `internal matches` and overlaps shorter than 2000 (option `drop -i -l 2000`), and finally runs `Miniasm` on scrubbed reads with filtered overlap.

Table 6 shows how the integration of both `yacrd` and `fpa` in `Miniasm` pipeline ('extended' row) compares against standard `Miniasm` ('simple' row). Each pipeline is based on `Minimap2` so their memory usages are equivalent. The extended pipeline takes twice more time because `Minimap2` is run twice (once for `yacrd` and once for `Miniasm`). `Minimap2` is a time bottleneck in both pipelines.

The extended pipeline improves the quality of assemblies, in terms of NGA50, number of indels and mismatches per 100 kbp, and misassemblies. It also decreases the number of contigs while keeping the total length of assemblies similar.

Dataset	*C. elegans*		*D. melanogaster*		*H. sapiens*		*E. coli* Nanopore		*E. coli* Pacbio	
Pipeline	simple	extended	simple	extended	simple	extended	simple	extended	simple	extended
# contigs	226	171	423	345	184	367	1	1	4	3
NGA50	432112	451479	423007	715276	96225	488573	1450762	3775889	499610	1271550
Largest contig	5422030	4224860	8745435	5559611	15987693	6875897	5147604	5186180	1974889	4960107
Largest alignment	1231264	1527213	2396453	3053469	857015	4444801	1553466	3775889	1083557	1465922
Total length	114194187	110177189	138733599	134443509	202082384	202405973	5147604	5186180	5417095	5355278
# indels per 100 kbp	7842.91	7380.12	5789.82	5593.09	6554.02	6359.25	5279.79	5097.12	8011,42	7969,99
# mismatches per 100 kbp	1944.78	1720.16	4233.35	4052.42	4089.56	3884.23	4341.81	4113.01	1856,96	1844,42
# misassemblies	1396	907	2126	1412	1745	363	5	3	11	8

Table 6. Impact of `yacrd` and `fpa` on assembly using `Miniasm`. Simple match to basic `Miniasm` pipeline and extend match to version with `yacrd` and `fpa`.

REFERENCES

Jain, M. *et al.* (2018). Nanopore sequencing and assembly of a human genome with ultra-long reads. *Nature Biotechnology*, **36**(4), 338–345.
Maio, N. D. *et al.* (2019). Comparison of long-read sequencing technologies in the hybrid assembly of complex bacterial genomes. *bioRxiv*.