Evaluating Metagenome Assembly on a Complex Community

Sherine Awad [1], Luiz Irber [1], C. Titus Brown [1,*]

1 Population Health and Reproduction University of California, Davis, Davis, CA, USA

∗ E-mail: ctbrown@ucdavis.edu

Abstract

NEEDS ENHANCEMENTS

Motivation: With the emergence of de novo assembly, several work have been to done to assemble metagenomic data from de novo. Several assemblers exist that are based on different assembly techniques. However, we still lack a study that analyze different assemblers behavior on metagenomic data .

Problem statement: In this paper, we performed analytical study for metagnome assembly using different assemblers and different preprocessing treatments. The aim of the analysis is studying how well metagenome assembly works, and which assembly works best. In addition, the study analyzes the resource requirements of the assembly.

Approach: We used a mock community dataset for the analysis, and used its reference genome for benchmark evaluation. We quality filtered the reads, then we applied 2 other preprocessing steps: digital normalization and partitioning. We used 4 different assembler: Velvet, IDBA-UD, SPAdes, and MEGAHIT to assemble the reads using each treatment. We used QUAST to analyze assemblies accuracy.

Results: Results show that assembly works well. Velvet is the worst assembler in terms of accuracy and recourses utilizations. The results also showed that assembly counts to most of the reads.

Conclusions: Except for Velvet, assemblers works well. Further analysis is required to study which assembler is better used with each specific dataset. This step is left for our future work,

Author Summary

WHAT SHOULD BE WRITTEN HERE

Introduction

Metagenome is the sequencing of DNA in an environmental sample. While whole genome sequencing (WGS) usually targets one genome, metagenome targets several ones which introduces complexity to metagenome analyis due to genomic diversity and variable abundance within populations. Metagenomic assembly means the assembly of multiple genomes from mixed sequences of reads of multiple species in a microbial community. Most approaches for analyzing metagenomic data rely on mapping to reference genomes. However, not all microbial diversity of many environments are covered by reference databases. Hence, the need for de novo assembly of complex metagenomic data rises. Several assemblers exist that can be used for de novo assembly. In order to decide which assembly works best, we need to evaluate metagenome assembly generated by each assembler. In this paper, we provide, an evaluation for metegnome assembly generated by several assemblers and using different preprocessing treatments. We use a reference genome as a benchmark for the evaluation. The evaluation is based on assembly accuracy, and time and memory requirements. This evaluation shed light on doability of metagenome assembly and the minimum requirements needed for the assembly. In addition, knowing how each assembler works, helps deciding which assembler to use prior to assembly. However, the later point is left for our future work.

The comparative study in this paper is based on four different assemblers; Velvet [1], SPAdes [2], IDBA-UD [3], and MEGAHIT [4].

Velvet [1] is a group de Bruin graph-based sequence assembly methods for very short reads that can both remove errors. It also uses read pair information to resolve a large number of repeats. The

error correction algorithm merges the sequences that belongs together. Then the repeat solver algorithm separates parts that share overlaps.

SPAdes [2] is an assembler for both single-cell and standard (multicell) assembly. SPAdes generates single-cell assemblies and provides information about genomes of uncultivatable bacteria that vastly exceeds what may be obtained via traditional metagenomics studies.

IDBA-UD [3] is a de Bruijn graph approach for assembling reads from single cell sequencing or metagenomic sequencing technologies with uneven sequencing depths. IDBA-UD uses multiple depth-relative thresholds to remove erroneous k-mers in both low-depth and high-depth regions. It also uses paired-end information to solve the branch problem of low-depth short repeat regions. It applies and error correction step to correct reads of high-depth regions that can be aligned to high confident contigs.

MEGAHIT [4] is a new approach that constructs a succinct de Bruijn graph using multiple k-mers, and uses a novel "mercy k-mer" approach that preserves low-abundance regions of reads. It also uses GPUs to accelerate the graph construction.

Materials and Methods

Datasets

We used a diverse mock community data set containing 64 known species, sequenced with Illumina HiSeq, yielding 109,629,496 paired-end sequences with an untrimmed length of 11.07 Gbp and an estimated insert size of $\sim$ 380 [5].

We received the reference genomes from the original authors (posted on FigShare at 10.6084/m9.figshare.1506873) and the original reads are available through the NCBI Sequence Read Archive at Accession SRX200676. Figure 4 shows the coverage profile of the reference genome, and the percentage of read with that coverage.

Quality Filtering

We removed adapters with Trimmomatic v0.30 in paired-end mode with the Truseq adapters [6]. We next used the fastq_quality_filter from the FASTX-Toolkit v0.0.13.2 [7] to remove sequences using the parameters -Q33 -q 30 -p 50, which keeps all sequences with 50% or more bases with quality score greater than or equal to 30.

Mapping

We aligned all quality-filtered reads to the reference metagenome with bwa aln (v0.7.7.r441) [8]. We aligned paired-end and orphaned reads separately using bwa aln samse. We then used samtools (v0.1.19) [9] to convert SAM files to BAM files for both paired-end and orphaned reads. To count the unaligned reads, we find the records with the "4" flag in the SAM files [9].

We found chimeric alignments with the bwa mem aligner using the default parameters (v0.7.7.r441). Chimeric alignments cut reads in two (or more). For each chimeric alignment, in the SAM file, there will be a primary alignment and at least one secondary alignment tagged SA. To count the chimeric alignments, we count the records with the "SA" flag in the SAM file [9].

To extract the reads that contribute to unaligned contigs, we mapped the quality filtered reads to the unaligned contigs using bwa aln (v0.7.7.r441) [8]. Then we used samtools [9] to retrieve the reads that are mapped to the unaligned contigs.

Reference Coverage and Coverage Profile

To evaluate how much of the reference genome was contained in the read data, we used bwa aln to map reads to the reference genome. We then calculated how many reference bases are contained in at least

one mapped read (script `sam-calc-refcov-cmp.py`). To draw a coverage profile for the reference, we calculated how many times a reference base is contained in mapped reads (script `cov.py`).

Digital Normalization

We applied the `normalize-by-median.py` script from khmer v1.1 to execute abundance normalization ("digital normalization", [10]) on the data, retaining paired reads and using a k-mer size of 20 (-p -k 20). We executed digital normalization with 4 hash tables, each 1 GB in size (-N 4 -x 1e9). After read normalization, we used the `filter-abund.py` script to trim high-abundance reads (estimated k-mer coverage ≥ 20) at low-abundance k-mers (k-mer abundance ≤ 2) to remove erroneous k-mers [11] [12].

Partitioning

We next applied partitioning to the data [13, 14]. We first eliminated high-abundance k-mers that could join multiple species bins using the `filter-below-abund.py` script from khmer v1.1 using an abundance cutoff of 50 or higher. We then ran `do-partition.py` with a k-mer size of 32 and 4 Bloom filters each of size 1 gigabit for partitioning (-k 32 -N 4 -x 1e9). After partitioning, partitions were extracted to groups using the `extract-partitions.py` script with a maximum group size of 100,000 (`-X 100000`).

Metagenomes Assembly and evaluation

We assembled the reads using four different assemblers: Velvet [1], IDBA-UD [3], SPAdes [2], and MEGAHIT [4].

For Velvet v1.2.07 [1], we used k-mer values from 19 to 51 incremented by 2. We also used -fastq.gz for fastq format, -shortPaired for the pe files and -short for the se files. Also, we asked Velvet to automatically calculate expected coverage and coverage cutoff (`-exp_cov auto -cov_cutoff auto`). From among the many assemblies, one for each k-mer size, we then chose the assembly that had the most bases in contigs longer than 500 bp (script `calc-best-assembly.py`).

For IDBA-UD v1.1.1 [3], we used –pre_correction to perform pre-correction before assembly and -r for the pe files. For SPAdes v3.1.1 [2], we used –sc –pe1-12 where –sc is required for MDA (single-cell) data and –pe1-12 for file with interlaced reads for the first paired-end library.

For MEGAHIT [4], we used -l 101 -m 3e9 –cpu-only where -l is for maximum read length , -m is for max memory in byte to be used, and –cpu-only to use CPU not GPU.

We examined the assembly quality of each assembler and treatment using QUAST v2.3 [15] using quast.py and we use the default minimum contig length equal to 500.

Results

Initial Evaluation of the Reads

We removed primers and poor-quality sequences as described in Methods. We retained 11.00 Gbp in 109,153,498 paired-end sequences, and 14.9 Mbp in 235,966 orphaned reads. After quality filtering, 10.7 Gbp of sequence remained in 106,134,639 paired-end sequences, with 12.6 Mbp in 2,226 orphan sequences. In total, only 3.01% of the reads were removed (3302631 reads, 340345361 bp), indicating that the original reads are high quality (see also [12], where an independent analysis of error rates in this data set using k-mer abundances found a very low error rate).

We mapped quality filtered reads to the metagenome reference. We found 3,664,869 unaligned reads which represent 3.45% of the total quality filtered reads. These unaligned reads are either highly erroneous reads that cannot be mapped, or are from sequences not present in the reference genomes. For mapped

reads, the quality of mapping was high, with 92.0m reads (86.49% of the quality filtered reads) having a MAPQ ≥ 30.

We then evaluated the fraction of the reference genome covered by at least one read (see Methods for details). Quality filtered reads cover 203,030,147 (98.75%) bases of the reference genome (205,603,715 bp total).

Effect of Digital normalization and partitioning

After digital normalization, we retained 1687.59 Mbp in 6,853,716 paired-end sequences, and 5.86 Mbp in 64,638 orphaned reads. After partitioning, we got 29 partitions. For paired-end sequences, the largest partition has 1.38 Gbp, and the smallest partition has 7.14 Mbp, in total, 1651.53 Mbp. For orphaned sequences, the largest partition has 13.90 Mbp, the smallest partition has 2.52 Mbp, in total 24.6 Mbp. Digital normalized reads and partitioned reads covered 202,201,168 and 201,193,779 bases of the reference genome (205,603,715), or 98.34%, and 97.85% of the reference respectively.

For mapped reads, the quality of mapping was high, with 15.25m reads and 15.16m reads having MAPQ ≥ 30, using digital normalized reads, and partitioned reads respectively.

We have 28,969 and 26,960 chimeric reads using digital normalized reads and partitioned reads respectively. Digital normalization and partitioning decreased chimeric alignments which is less than the chimeric reads found using quality filtered reads (310,131).

Compute Cost of Assembly

We estimated time and memory requirements for each of them. We also estimated the running time and memory utilization for each assembler under both treatments and compared to assemblers time and memory requirements using quality filtered reads.

Digital normalization utilized 74.93 GB of memory and took around 3 hours and 53 minutes to run. Partitioning utilized 21.78 GB and around 2 hours and a half to run.

For Velvet assemblies, table 2 row 3, it took ~ 60 hours using quality filtered reads, while it took only ~ 6 hours using digital normalizations and ~ 4 hours using partitioning. For IDBA-UD assemblies, table 2 row 6, it took ~ 33 hours using quality filtered reads, while it took ~ 6 hours using digital normalization and ~ 8 hours using partitioning. SPAdes assemblies utilized ~ 67 hours using quality filtered reads while it took ~ 15 hours and ~ 7 hours using digital normalization and partitioning respectively, table 2 row 9. Finally, for MEGAHIT, it took ~ 2 hours, $\sim$ half an hour, and $\sim$ hour and a half using quality-filtered reads, digital normalization, and partitioning respectively, table 2 row 12.

For Velvet assemblies, table 2 row 4, it used 98.40 GB of memory using quality filtered reads, while it used one 52.67 GB and 35.23 GB of memory when applying digital normalization and partitioning respectively. For IDBA-UD assemblies, table 2 row 7, it used used 123.84 GB of memory using quality filtered reads, while it used one 99.88 GB and 76.53 GB of memory when applying digital normalization and partitioning respectively. For SPAdes assemblies, table 2 row 10, it used 381.79 GB of memory using quality filtered reads, while it used one 121.52 GB and 94.70 GB of memory when applying digital normalization and partitioning respectively. For MEGAHIT, table 2 row 13 it utilizes 33.41 GB, 18.89 GB, 13.17 GB for quality-filtered reads, digital normalization, and partitioning respectively. See Table 2 for more details. Clearly, MEGAHIT is the best assembler in terms of memory and time utilization. We also conclude that Digital normalization and partitioning treatments reduced time and memory requirements of assembly while they didn't affect assemblies quality (see above).

Assembly Comparisons

All of the assembler/treatment comparisons (with the exception of Velvet/QC, which will not be considered further) recovered approximately 90% of the reference genome. IDBA/QC yielded the best assembled

length (as measured by N50), but also had the largest number of bases in misassembled contigs; however, MEGAHIT (when compared treatment-to-treatment) had the fewest number of bases in misassembled contigs, while still retaining a high N50.

IDBA/QC again performed best in both assembly length and genome recovery when looking at small contigs: Figure 1 and Figure 2 show that IDBA/QC has more data in long contigs than do any of the other assembler/treatment pairs.

Digital normalization increased the number of bases in misassembled contigs, in general, while the partitioning treatment always had the fewest number of bases in misassembled contigs. Digital normalization always decreased the N50, and partitioning after digital normalization also always decreased the N50.

All assemblies have approximately the same number of unaligned bases, suggesting that data missing from the reference is being assembled. We examine this further below.

Evaluation against Reference Genome

We aligned each assembly to the reference genome using Quast [15]. To evaluate the quality of the seembly, we used different quality metrics, including the unaligned length and genome fraction percentage. The genome fraction % is the percentage of aligned bases in the reference. A base in the reference is aligned if there is at least one contig with at least one alignment to this base. The unaligned length is 8,977,149 bp, 10,709,716 bp, 10,597,529 bp, and 10,686,421 bp using quality filtered reads for Velvet, IDBA-UD, SPAdes, and MEGAHIT respectively. The genome fraction percentage is 72.949 %, 90.969 %, 90.424%, and 90.358% using quality filtered reads for Velvet, IDBA-UD, SPAdes, and Megahit respectively. Using digital normalization, the genome fraction is 89.043%, 91.003%, 90.173%, and 89.92% for Velvet, IDBA-UD, SPAdes, and Megahit respectively. Using partitioning, the genome fraction is 88.879%, 90.082%, 89.272%, and 88.769% for Velvet, IDBA-UD, SPAdes, and MEGAHIT respectively. Except for Velvet assemblies, digital normalization and partitioning did not effect genome fraction percenatge. However, for Velvet assemblies, digital normalization and partitioning substantially increased the genome fraction percentage. See Table 1 for more details.

Metagenome assemblies account for the majority of reads

To further evaluate assemblies, we mapped the quality filtered reads to each assembly. Then we extracted the unaligned sequences. Table 4 shows the number and percentages of unaligned sequences from mapping quality filtered reads to each assembly. For all assemblies, the full set of trimmed reads were used for mapping.

For quality-filtered assembly, the number of unaligned reads is 6,801,329, 1,490,609 and 2,100,555, and 1,559,300 for Velvet, IDBA-UD, SPAdes, and MEGAHIT respectively. This represents 6.40%, 1.40%, 1.97%, and 1.47% of the total number of reads respectively. These percentages reflect errors or low coverage reads. See Table 4 for more details.

Decreasing Coverage did not effect assembly quality

We extracted random samples of size 25%, 50%, and 75% of the raw reads . We quality filtered each sample, then applied digital normalization and partitioning. For quality filtered reads, Velvet assembly genome fraction decreased from 72.95% using all reads to 71.31%, 70.09%, and 63.53% using 75%, 50%, and 25% of reads respectively. IDBA assembly genome fraction decreased from 90.97% using all reads to 86.90%, 84.66%, and 76.98% using 75%, 50%, and 25% of reads respectively. SPAdes assembly genome fraction decreased from 90.42% using all reads to 87.54%, 86.00%, and 80.05% using 75%, 50%, and 25% of reads respectively. Megahit assembly genome fraction decreased from 90.36% using all reads to 86.00%, 83.87%, and 76.19% using 75%, 50%, and 25% of reads respectively. Figure 3 shows genome fraction

percentages for different assemblies using different coverage. We conclude that decreasing coverage did not highly affect assemblies qualities.

Assembly Errors

Misassembled contigs length is the total number of bases in misassembled contigs. As shown in Table 1, using quality filtered reads, mis-assemblies contigs length are 16,566,891, 21,777,032, 28,238,787 and 11,927,502 for Velvet, SPAdes, IDBA-UD, and MEGAHIT respectively which represents 8.057%,10.59%, 13.73% and 5.801% of the reference genome respectively.

Using digital normalization, mis-assemblies contigs length are 25,594,315, 27,668,818, 23,103,154, and 17,319,534 for Velvet, SPAdes, IDBA-UD, and MEGAHIT respectively which represents 12.45%, 13.46%, 11.24% and 8.42% of the reference genome respectively.

Using partitioning, mis-assemblies contigs length are 16,922,852, 18,440,791, 14,338,099, and 11,814,070 for Velvet, SPAdes, IDBA-UD, and MEGAHIT respectively which represents 8.23%, 8.97%, 6.97% and 5.75% of the reference genome respectively.

Although IDBA-UD has the highest NG50 (see above), IDBA-UD shows the highest mis-assemblies contigs length using digital normalization and partitioning. It also shows a high mis-assemblies contigs length using quality filtering but not the highest.

Using quality filtering, mismatches percentages are 0.05%, 0.08%, 0.09%, and 0.07% for Velvet, IDBA-UD, SPAdes, and MEGAHIT respectively. Indels percentages are 0.02%, 0.014%, 0.013%, and 0.007% using Velvet, IDBA-UD, SPAdes, and MEGAHIT respectively. Percentages are computed with respect to the reference genome. See Table 3 for more details about mis-assemblies contigs, the types of misassembly events, mismatches and indels lengths.

More about misassemblies

We looked for the genomes that has 10 regions or more that are misassemled by all the assemblers. Using quality filtering, Enterococcus_faecalis_V583, Desulfovibrio_vulgaris_DP4, Sulfurihydrogenibium_yellowstonense_SS-5,Thermus_thermophilus_HB27, and Pyrococcus_furiosus_DSM_3638 has 70, 26, 22, 56, and 11 regions that are misassembled by all the assemblers. The misassembled regions are scattered with average gap length equal to 548,867, 875,212, 229,042, 305,389, and 284,824 for Enterococcus_faecalis_V583, Desulfovibrio_vulgaris_DP4, Sulfurihydrogenibium_yellowstonense_SS-5,Thermus_thermophilus_HB27, and Pyrococcus_furiosus_DSM_3638 respectively. This shows that these genomes contain problematic regions, however, the regions are not localized.

More about uncovered regions

To find the uncovered regions in the reference genome, we find the regions that don't contain any alignment positions when mapping the assembly to the reference. @CTB So far uncovered regions are not common among all assemblers??

To more investigate the uncovered regions in the reference, we aligned the reference genome to each assembly. Using quality filtering, we got zero unaligned contigs, in addition to 29,776,904, 2,062,384, 2,063,510, and 2,063,473 partially unaligned length for Velvet, IDBA-UD, SPAdes, and MEGAHIT respectively which represent 14.48%, 1.003%, 1.003%, and 1.003% of the reference genome. Using digital normalization, we got also zero unaligned contigs, with partial unaligned length 2,046,452, 2,067,700, 2,063,803, and 2,063,457 for Velvet, IDBA-UD, SPAdes, and MEGAHIT respectively which represent 1.0%, 1.005%, 1.003%, and 1.003% of the reference genome. Using partitioning, we got also zero unaligned contigs, with partial unaligned length 2,053,035, 2,071,046, 2,069,811 and 2,068,118 for Velvet, IDBA-UD, SPAdes, and MEGAHIT respectively which represent 1.0%, 1.007%, 1.006%, and 1.005% of the reference genome.

In conclusion, $\sim$ 1.01% of the reference genome is not covered by the assemblies, except for Velvet/QC, which has 14.48% of the reference uncovered.

More about unalignments

We extracted the reads that contributed to unaligned contigs of each assembler (see Methods:Mapping). We mapped those reads to the reference genome. We find that most of the reads contributing to the unaligned contigs don't exist in the reference genome. Figure 5 shows a histogram for the percentage of reads contributed to unaligned contigs (orange columns) and the portion of those reads that mapped to the reference genome (green columns).

Furthermore, we aligned the unaligned contigs of each assembly to the unaligned contigs of IDBA-UD quality filtered assembly. Using quality filtered reads, the reference fraction (the reference here is the unaligned contigs of IDBA-UD quality filtered assembly) equals to 80.613%, 91.922%, and 92.715% for Velvet, SPAdes, and MEGAHIT respectively. The unaligned length is 2,475,529, 2,174,574, and 916,247 for Velvet, SPAdes, and MEGAHIT respectively using quality filtered reads, representing 37.06%, 32.56% and 13.72% of the reference length. See Table 5 for more details. Velvet unalignments show the highest percentages of IDBA-UD QC unalignments. IDBA-UD diginorm unalignments shows the less percentage of IDBA-UD QC unalignments.

**To further explore the unalignment, we downloaded fusa seeds from Fungenes. We used blast to map the unaligned reads to fusa seeds, using IDBA-UD/digital normalized assemblies, and no hits found.

Discussion

Dataset has a high quality

Having almost all reads aligned to the reference, shows that the data has a high quality. The high quality of the data is one reason the assemblers work the same (see below). However, understanding the behaviors of assemblers on less quality data is essential to decide which assembler is best to use. Analyzing the behaviors of assemblers on different datasets, using different quality and signatures is left for our future work.

Assembly works pretty well

Except for Velvet assembly using quality filtered reads, the genome fraction percentage is 88% or higher. Unaligned length is less than 1% for all assemblies. Misassembled length is less than 1.3% for all assemblies. Velvet shows the least performance in terms of accuracy and time, and memory utilizations. Some genomes contain problematic regions that causes misassemblies. We conclude that assembly works well although there are some rooms for improvements including enhancing accuracy, and decreasing time and memory requirements.

Digital normalization and partitioning significantly reduce running time and memory utilizations

The difference between genome fraction percentage using quality filtered reads versus digital normalizations and partitioning doesn't exceed 1%. However, the time and memory resource are reduced a lot using digital normalization and partitioning. This also means that digital normalization throws unnecessary reads. We conclude that digital normalization and partitioning are beneficial steps for assembly to reduce time and memory utilities while not affecting quality.

Digitial normalization and partitioning do not affect mis-assemblies and unalignments

Except for Velvet assemblies, misassemblies are not affected by digital normalization and partitioning. Mapping the unaligned contigs of different assemblies to the unaligned contigs of IDBA-UD assembly using quality filtered , shows genome fraction percentage is 91% or higher. This means the unaligned contigs are common among assemblies. These common unaligned contigs are likely to be unknowns, new assemblies, or contamination. This indicates that digital normalization did not throw necessary reads. In addition, digital normalization and partitioning enhance assembly time and memory utilizations without affecting assembly accuracy.

Assembly recovers content not in the reference

Assemblies account for the majority of reads

Acknowledgments

References

1. Zerbino DR, Birney E (2008) Velvet: algorithms for de novo short read assembly using de bruijn graphs. Genome Research 18: 821-829.

2. Bankevich A, Nurk S, Antipov D, Gurevich AA, Dvorkin M, et al. (2012) Spades: A new genome assembly algorithm and its applications to single-cell sequencing. Journal of Computational Biology 19: 455-477.

3. Peng Y, Leung HC, Yiu S, Chin FY (2012) Idba-ud: a de novo assembler for single-cell and metagenomic sequencing data with highly uneven depth. Bioinformatics 28: 1420-1428.

4. Li D, Liu CM, Luo R, Sadakane K, Lam TW (2014) Megahit: An ultra-fast single-node solution for large and complex metagenomics assembly via succinct de bruijn graph. Bioinformatics .

5. Migun S, Quince C, Campbell J, Yang Z, Schadt C, et al. (2013) Comparative metagenomic and rrna microbial diversity characterization using archaeal and bacterial synthetic communities. Enivromental Microbiology 15: 1882-1899.

6. Anthony B, Marc L, Bjoern U (2014) Trimmomatic: a flexible trimmer for illumina sequence data. Bioinformatics : btu170.

7. Lab H. Fastx toolkit. URL `http://hannonlab.cshl.edu/fastx_toolkit/index.html`.

8. Li H, Durbin R (2009) Fast and accurate short read alignment with burrows–wheeler transform. Bioinformatics 25: 1754–1760.

9. Li H, Handsaker B, Wysoker A, Fennell T, Ruan J, et al. (2009) The sequence alignment/map format and samtools. Bioinformatics 25: 2078–2079.

10. Brown CT, Howe A, Zhang Q, Pyrkosz AB, Brom TH (2012) A reference-free algorithm for computational normalization of shotgun sequencing data. arXiv preprint arXiv:12034802 .

11. Zhang Q, Pell J, Canino-Koning R, Howe AC, Brown CT (2014) These are not the k-mers you are looking for: Efficient online k-mer counting using a probabilistic data structure. PLoS One 9.

12. Zhang Qingpeng AS, Titus B (2015) Crossing the streams: a framework for streaming analysis of short dna sequencing reads. PeerJ PrePrints 3:e1100 https://dxdoiorg/107287/peerjpreprints890v1 .

13. Pell J, Hintze A, Canino-Koning R, Howe A, Tiedje JM, et al. (2012) Scaling metagenome sequence assembly with probabilistic de bruijn graphs. Proceedings of the National Academy of Sciences 109: 13272–13277.

14. Howe AC, Jansson JK, Malfatti SA, Tringe SG, Tiedje JM, et al. (2014) Tackling soil diversity with the assembly of large, complex metagenomes. Proceedings of the National Academy of Sciences 111: 4904–4909.

15. Gurevich A, Saveliev V, Vyahhi N, Tesler G (2013) Quast: quality assessment tool for genome assemblies. Bioinformatics 28: 1072-1075.

Figure Legends

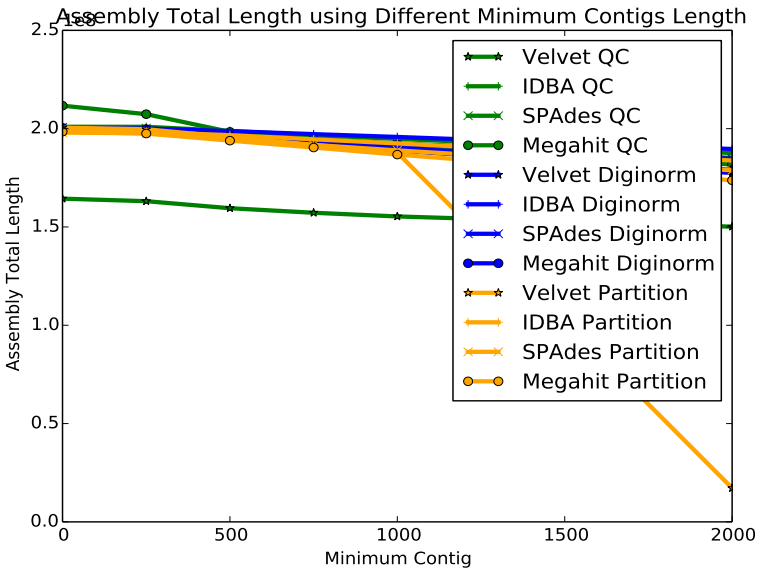

Figure 1. *Total Length of assemblies in basepairs based on different min contigs length.*

Tables

Supporting Information Legends

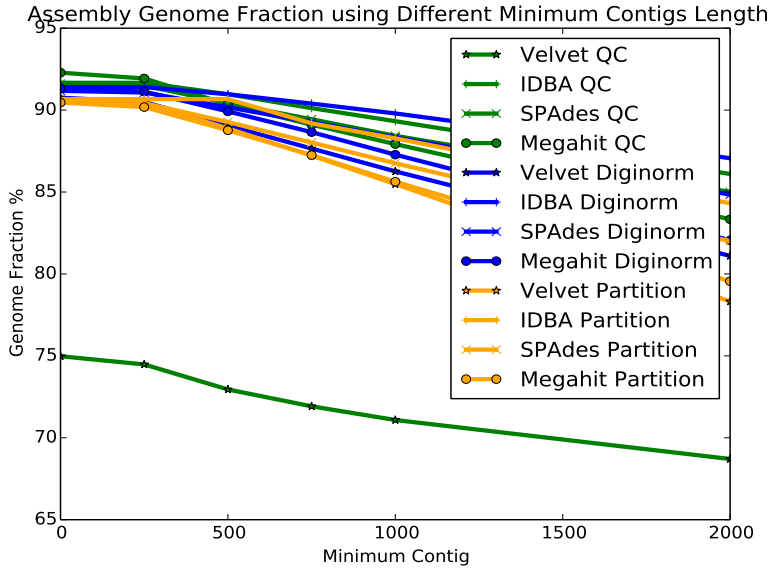

Figure 2. *Genome Fraction of assemblies in basepairs based on different min contigs length.*

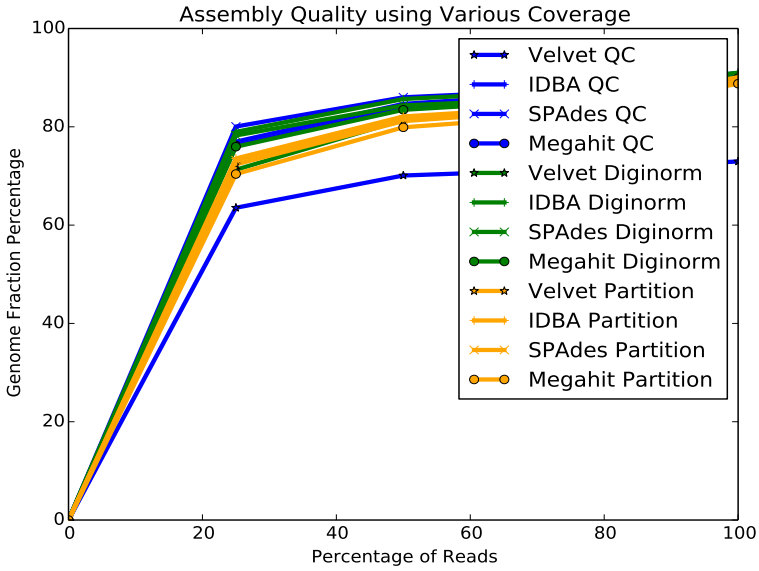

Figure 3. *Genome Fraction of assemblies using different read coverage.*

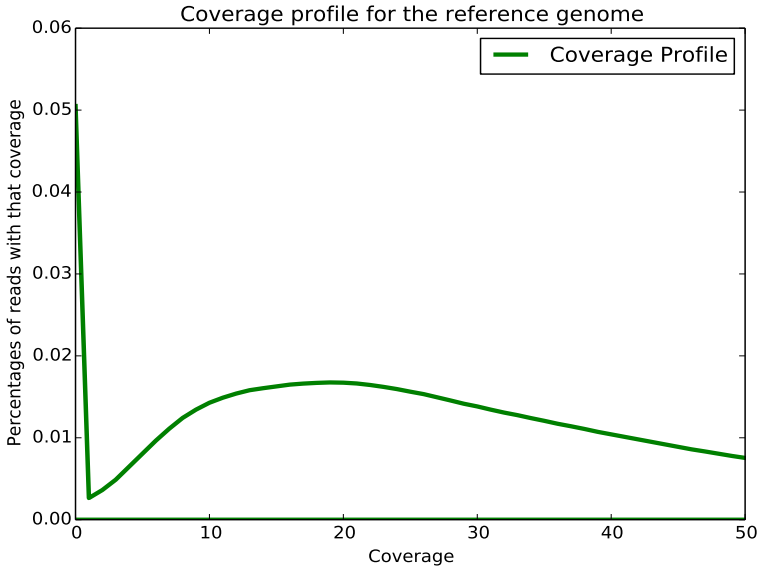

Figure 4. *Reference genome coverage.*

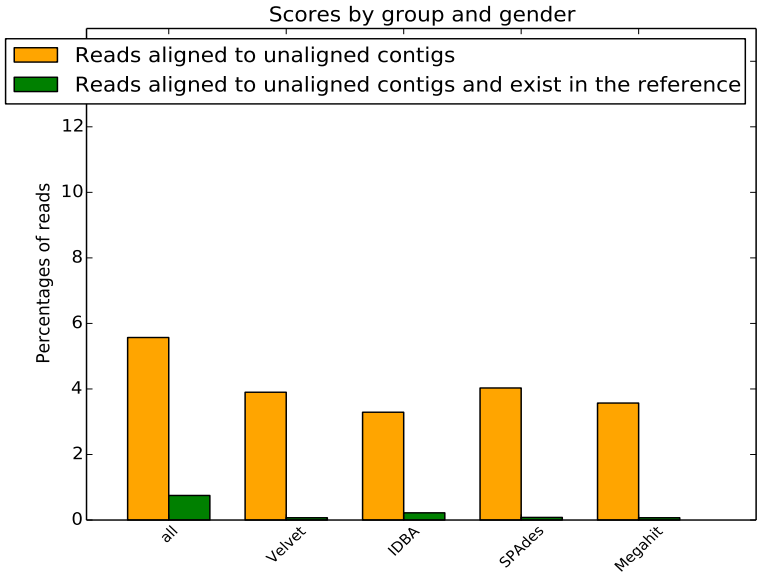

Figure 5. *Histogram for unaligned reads.*

Table 1. Assembly Quality Metrics

Treatment/Quality Metric	Quality Filtering	Digital Normalization	Partition
(1) Velvet			
Genome Fraction	72.949	89.043	88.879
Unaligned Length	8,977,149	10,909,693	11,317,834
Misassembled contigs length	16,566,891	25,594,315	16,922,852
N50	38,028	18,944	8,504
NG50	22223	17212	7905
(2) IDBA-UD			
Genome Fraction	90.969	91.003	90.082
Unaligned Length	10,709,716	10,637,811	10,644,357
Misassembled contigs length	21,777,032	27,668,818	18,440,791
N50	49773	47828	26575
NG50	45748	44351	24326
(3) SPAdes			
Genome Fraction	90.424	90.173	89.272
Unaligned Length	10,597,529	10,621,398	10,500,235
Misassembled contigs length	28,238,787	23,103,154	14,338,099
N50	42773	35580	22319
NG50	38841	32598	19909
(4) MEGAHIT			
Genome Fraction	90.358	89.92	88.769
Unaligned Length	10,686,421	10,581,435	10,564,244
Misassembled contigs length	11,927,502	17,319,534	11,814,070
N50	35,136	27,302	17,492
NG50	32251	25248	15393

Table 2. Running Time and Memory Utilization

Treatment/Quality Metric	Quality Filtering	Digital Normalization	Partition
(1) Velvet			
Running Time	60:42:52	6:48:46	4:30:36
Memory Utilization in GB	98.40	52.67	35.23
(2) IDBA-UD			
Running Time	33:53:46	6:34:24	8:30:29
‘ Memory Utilization in GB	123.84	99.88	89.25
(3) SPADes			
Running Time	67:02:16	15:53:10	7:54:26
Memory Utilization in GB	381.79	121.52	123.7
(4) MEGAHIT			
Running Time	1:52:55	0:30:23	1:23:28
Memory Utilization in GB	33.41	18.89	189.55

Table 3. mis-assemblies

Assembly	Quality Filtering	Digital Normalization	Partition
(1) Velvet			
mis-assemblies	917	5271	5202
Relocations	592	998	1036
Translocations	309	4262	4153
Inversions	16	11	13
Misassembled Contigs Length	16,566,891	25,594,315	16,922,852
Mismatches	104,740	174,446	178,348
Percentage of Mismatches	0.05%	0.08%	0.09%
Indels Length	50,190	181,453	346,988
Indels Percentage	0.02%	0.09%	0.17%
(3) IDBA-UD			
mis-assemblies	1223	1094	960
Relocations	613	668	578
Translocations	580	398	350
Inversions	30	28	32
Misassembled Contigs Length	21,777,032	27,668,818	18,440,791
Mismatches	162,733	231,432	230,840
Percentage of Mismatches	0.08%	0.11%	0.11%
Indels Length	30,433	43,358	42,523
Indels Percentage	0.01%	0.02%	0.02%
(2) SPAdes			
mis-assemblies	894	997	753
Relocations	608	613	496
Translocations	267	368	239
Inversions	19	16	18
Misassembled Contigs Length	28,238,787	23,103,154	14,338,099
Mismatches	184,630	244,849	235,396
Percentage of Mismatches	0.09%	0.12%	0.11%
Indels Length	27,328	32,783	21,516
Indels Percentage	0.01%	0.02%	0.01%
(4) MEGAHIT			
mis-assemblies	738	880	748
Relocations	448	593	513
Translocations	172	274	222
Inversions	118	13	13
Misassembled Contigs Length	11,927,502	17,319,534	11,814,070
Mismatches	152,964	207,349	203,515
Percentage of Mismatches	0.07%	0.10%	0.10%
Indels Length	15,298	18,195	16,517
Indels Percentage	0.01%	0.01%	0.01%

Table 4. Mapping quality-filtered reads to assemblies

Treatment	Quality Filtering	Digital Normalization	Partition
(1) Velvet			
No. of Unaligned Sequences	6,801,329	3,375,222	3,890,205
Percentage	6.40%	3.17%	3.66%
(2) IDBA-UD			
No. of Unaligned Sequences	1,490,609	1,738,371	2,297,377
Percentage	1.40%	1.63%	2.16%
(3) SPAdes			
No. of Unaligned Sequences	2,100,555	2,439,158	2,804,006
Percentage	1.98%	2.29%	2.64%
(4) MEGAHIT			
No. of Unaligned Sequences	1,559,300	2,082,881	2,747,427
Percentage	1.47%	1.96%	2.58%

Table 5. Mapping unaligned contigs to Idba quality-filtered unaligned contigs

Treatment/Quality Metric	Quality Filtering	Digital Normalization	Partition
(1) Velvet			
Unaligned IDBA Fraction	80.613 %	92.03%	92.982%
Unaligned Length	2,475,529	3192491	3,868,558
Percentage of unaligned	37.06%	47.8%	57.92%
(2) IDBA-UD			
Unaligned IDBA Fraction	-	91.53%	94.72%
Unaligned Length	-	498,299	1,320,036
Percentage of unaligned	-	7.46%	19.76%
(3) SPAdes			
Unaligned IDBA Fraction	91.92%	93.959%	94.826%
Unaligned Length	2,174,574	1,951,911	2,398,664
Percentage of unaligned	32.56%	29.22%	35.91%
(4) MEGAHIT			
Unaligned IDBA Fraction	92.715%	91.838%	92.219%
Unaligned Length	916,247	1,569,436	3,832,050
Percentage of unaligned	13.72%	23.5%	57.37%

Table 6. Supplementary Table: More Assembly Quality Metrics

Treatment/Quality Metric	Quality Filtering	Digital Normalization	Partition
(1) Velvet			
N75	13301	6084	3771
NG75	1186	4805	3214
L50	1013	2455	6037
LG50	1806	2740	6641
L75	2777	7026	14734
LG75	11087	8460	16867
(2) IDBA-UD			
N75	11693	12154	7834
NG75	9617	10221	6461
L50	828	896	1536
LG50	899	970	1712
L75	2986	3025	5062
LG75	3467	3484	6002
(2) SPAdes			
N75	11263	10554	6900
NG75	9005	8379	5401
L50	974	1192	1846
LG50	1078	1325	2108
L75	3276	3768	5840
LG75	3908	4495	7198
(4) MEGAHIT			
N75	8166	7230	5271
NG75	6601	5632	4030
L50	1199	1582	2490
LG50	1306	1757	2848
L75	4164	5063	7670
LG75	4907	6147	9581

Table 7. Comparision between N50 and NG50

Treatment	Quality Filtering	Digital Normalization	Partition
(1) Velvet			
N50	38,028	18,944	8,504
NG50	22,223	17,212	7,905
(2) IDBA-UD			
N50	49,773	47,828	26,575
NG50	45,748	44,351	24,326
(3) SPAdes			
N50	42,773	35,580	22,319
NG50	38,841	32,598	19,909
(4) MEGAHIT			
N50	35,136	27,302	17,492
NG50	32,251	25,248	15,393