Evaluating Metagenome Assembly on a Complex Community

Sherine Awad [1], 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 XX known species, sequenced with Illumina HiSeq, yielding XX paired-end sequences with an untrimmed length of XX and an estimated insert size of XX [5]. The two unfiltered read files each contained 5.5 Gbp of sequence data. (@CTB is this data set size, or number of bases?) (@SAM that's the number of bases using readstats) In total, the data set contained 11.1 Gbp of DNA sequence in 109,629,496 reads. We received the reference genomes from the original authors (posted on FigShare at doi@@) and the original reads are available through the NCBI Short Read Archive at Accession XXX.

Quality Filtering

We removed adapters with Trimmomatic v0.30 in paired-end mode with the TruSeq3 adapters [6]. The command line options used was XXX. @CTB - please find out what adapters we should use for trimming. It's probably not TruSeq3. @SAM what we used TruSeq3-PE.fa:2:30:10?? We next used the fastq_quality_filter (@CTB what program exactly?) from the FASTX-Toolkit v0.0.13.2 [7] to remove sequences with (@CTB describe what those parameters do) using the parameters -Q33 -q 30 -p 50 where -q is the minimum quality score to keep and -p is the minimum percent of bases that must have [-q] quality (@SAM can't explain q33) (cite fastx, give version).

Mapping

We aligned all quality-filtered reads to the reference metagenome with bwa aln (v0.7.7.r441) (@SAM I use bwa aln by loading bwa) and bwa samse (XXX give version and any command line options that aren't default). We aligned paired-end and orphaned reads separately, and we aligned each as single reads using bwa samse. We then used samtools (v0.1.19) to convert SAM files to BAM files for both paired-end and orphaned reads. To find the unaligned reads, we find the reads with "4" flag in the bam files.

We found chimeric reads with the bwa mem aligner with the default parameters (v0.7.7.r441). To find the Chimeric reads, we find the reads with 'SA' flag in the sam file. @SAM Should we mention again converting to samtools? (XXX give bwa version).

Reference Coverage

We used bwa aln (v0.7.7.r441) [8] to map quality filtered reads, digital normalized reads and partitioned reads to the reference genome generating a sam file from each. Then we used "sam-calc-refcov-cmp.py" script, the reference genome and the generated sam files to find out the reference coverage.

Digital Normalization

We applied the `normalize-by-median.py` script from khmer v1.1 to execute abundance normalization ("digital normalization", @cite Brown 2012) [9] on the data, retaining paired reads with -p 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 – @CTB check these parameters) to remove erroneous k-mers [10] (@cite Zhang et al., 2014 PLoS One; Zhang et al., 2015 (peerJ)). @SAM what is zhang et al. peerJ 2015?

@CTB was a second round of digital normalization run here? @SAM yes, this is the second round

Partitioning

We next applied partitioning to the data [11, 12] (cite Pell et al. 2012, Howe et al., 2014). We first eliminated high-abundance k-mers that join multiple species bins using `filter-below-abund.py` script from khmer v1.1. We next ran `do-partition.py` with a k-mer size of 32 and 4 Bloom filters each of size 1 GB 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]. (@CTB please check and use appropriate capitalization - look at the original papers to see what they use. I think it's IDBA-UD, MEGAHIT, SPADes.)

For Velvet [1], we used kmers 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 used exp_cov auto cov_cutoff auto for expected coverage and coverage cutoff respectively.

For IDBA-UD [3], we also used –pre_correction and -r for the pe files. For SPAdes [2], we used –sc –pe1-12 where –sc is required for MDA (single-cell) data and –pe1-12 @SAM? 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.

Spades? YYYY

Megahit? YYYY

We examined the assembly quality of each assembler and treatment using QUAST v2.3 [13] using quast.py with these parameters -R for the reference metagenome and -o for the output and we use the default min contig length equal to 500.

Results

Initial Evaluation of the Reads

(@CTB how? put in Methods; then remove this whole sentence). We mapped quality filtered reads to the metagenome reference. We found 3,664,869 unaligned reads which represents 3.45% of the quality filtered reads. These unaligned reads represent 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 having a MAPQ $\geq$ 30. (@CTB put this "Having almost all reads aligned to the reference, shows that the data has a high quality" in discussion.)

We then evaluated the fraction of the reference genome covered by at least one read. (sam-calc-refcov-cmp.py script goes in Methods.) Quality filtered reads cover 203,030,147 bases of the reference genome (205,603,715), or 98.75% of the reference.

We also used bwa mem [8] to look for chimeric reads where a read spans two different genes (@CTB what does this mean? Do you have a reference for this?). (@CTB: The detail of how you did this belongs in methods.) @SAM

Quality filtering did not change the number of reads

We performed quality filtering to remove primers and poor-quality sequences. After primer trimming, we retained 11.00 Gbp in 109,153,498 paired-end sequences, and 14.9 Mbp in 235,966 orphaned reads. After quality filtering, 107 Gbp of sequence remained in 106,134,639 paired-end sequences, with 12.6 Mbp in 192,226 orpan 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 Zhang et al., 2015, PeerJ preprint, where an independent analysis of error rates in this data set using k-mer abundances found a very low error rate).

Effect of Digital normalization and partitioning

@CTB please format and shorten this subsection as above, and spell check :). 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, in the largest partition we retained 1379.27 Mpb, in the smallest partition retained 7.14 Mbp, and in total, we retained1651.53 Mbp. For orphaned sequences, in the largest partition we retained 13.90 Mbp, in the smallest partition we retained 2.52 Mbp, and in total we retained 24.6 Mbp. Knowing that quality filtered reads retained 10547.79 Mbp in paired-end sequences and 184.44 Mbp in orphaned sequences (see above for more details), clearly, digital normalization and partitioning decrease the total number of reads.

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 $\geq$ 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 reads. (See above for quality filtered reads no. of chimeric reads).

Metagenome assemblers recover the great majority of the known content

We used four assemblies and run each one using each treatment; quality filtering, digital normalization, and partitioning. The unaligned length for assembly 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 % 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 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. Clearly, digital normalization and partitioning did not effect assembly results. See table 1 for more details.

Figure 1 shows total length of different assemblies using different min contig length. Increasing the min contig length slightly decreased the total assembly length. Figure 2 shows genome fraction percentages using different min contig length.

Compute Cost of Assembly

please spell check. To evaluate whether digital normalization and partitioning are doable treatments, 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, it took $\sim$ 60 hours using quality filtered reads, while it took only $\sim$ 6 hours using digital normalizations and $\sim$ 4 hours using partitioning. For IDBA-UD assemblies, it took $\sim$ 33 hours using quality filtered reads, while it took $\sim$ 6 hours using digital normalization and $\sim$ 8 hours using partitioning. SPAdes assemblies utilized $\sim$ 67 hours using quality filtered reads while it took $\sim$ 15 hours and $\sim$ 7 hours using digital normalization and partitioning respectively. Finally, for MEGAHIT, it took $\sim$ 2 hours, $\sim$ half an hour, and $\sim$ hour and a half using quality-filtered reads, digital normalization, and partitioning respectively.

For Velvet assemblies, it used 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, 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, 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, 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). This means that digital normalization throws unnecessary reads.

Assembly Output Statistics

@CTB I think we should order things like so: 1. Effect of digital normalization on reads. 2. Compute Cost of assembly. 3. Assembly output statistics. 4. Evaluation against reference genome. while leaving the other sections alone.

Contig or scaffold N50 is a weighted median statistic such that 50% of the entire assembly is contained in contigs or scaffolds equal to or larger than this value. Using quality filtering, N50 is 38,028, 49,773, 42,773, and 35,136 for Velvet, IDBA-UD, SPAdes, and MEGAHIT respectively. The NG50 statistic is the same as N50 except that it is 50% of the known reference genome size that must be achieved. Using quality filtered reads, NG50 is 22,223, 45,748, 38,841, and 32,251 for Velvet, IDBA-UD, SPAdes, and Megahit respectively. N50 and NG50 decreases with digital normalization and partitioning. For diginorm assemblies, N50 is 18,944 47,828, 35,580, and 35,427 for Velvet, IDBA-UD, SPAdes, and MEGAHIT respectively. For partitioning, N50 is 8,504, 26,575, 22,319, and 17,492, for Velvet, IDBA-UD, SPAdes, and MEGAHIT respectively. See Table 1 for more details. IDBA-UD has the highest NG50 although it has a high misassemblies contigs length, mismatches percentages and indels percentages (see below). For all assemblers and treatments, N50 is higher than NG50. This shows that using N50 may give false vision about assembly quality.

Assembly Errors

Misassembled contigs length is the total number of bases in misassembled contigs. As shown in Table 1, using quality filtered reads, misassemblies contigs length are 16,566,891, 21,777,032, 28,238,787 and 11,927,502 for Velvet, SPAdes, IDBA-UD, and MEGAHIT respectively. Using digital normalization, misassemblies contigs length are 25,594,315, 27,668,818, 23,103,154, and 17,319,534 for Velvet, SPAdes, IDBA-UD, and MEGAHIT respectively. Using partitioning, misassemblies contigs length are 16,922,852, 18,440,791, 14,338,099, and 11,814,070 for Velvet, SPAdes, IDBA-UD, and MEGAHIT respectively. Although IDBA-UD has the highest NG50 (see above), IDBA-UD shows the highest misassemblies contigs length using digital normalization and partitioning. It also shows a high misassemblies contigs length using quality filtering but not the highest. Using quality filtering, mismatches percentages are 0.066%, 0.082%, 0.093%, and 0.078% for Velvet, IDBA-UD, SPAdes, and MEGAHIT respectively. Indels percentages are 0.031%, 0.015%, 0.014%, and 0.008% using Velvet, IDBA-UD, SPAdes, and MEGAHIT respectively. Percentages are computed with respect to their corresponding assembly total length. See Table 3 for more details about misassemblies contigs, the types of missassembly events, mismatches and indels lengths.

To find out if there is a portion of the reference genome that is misassembled by most of the assemblers, we used QUAST [13] to align the reference genome to each assembly. Then we looked into the misassemblied contigs of the reference to each assembly. They all are portions of the reference genome because of the direction of mapping (align reference to assemblies). Using quality filtered reads, Velvet has 56 misassembled contigs (the lowest number), while SPAdes and MEGAHIT has the highest misassembled contigs(63). IDBA-UD has 62 misassembled contigs. We also found that 55 contigs are common among assemblers. Note that we compared only contigs names because we look for which contigs are commonly misassembled not which bases are commonly misassembled (CTB?). Comparing the misassemblies length among assemblers, it varies even with assemblers which shares the same misassembled contigs. This means that assemblers differ on how much portions of a given contig is misassembled. We conclude that most of the misassembled contigs are common among assemblers using quality filtered reads. Using diginorm and partitioning, each assembler has 63 misassembled sequences and we have 63 common misassembled sequences. This means all assemblers fail to assemble the same sequences in the reference genome.

Evaluation against Reference Genome

We aligned the reference genome to each assembly to find out what exists in the reference genome and not found in the assemblies. Aligning the reference genome to Velvet, IDBA-UD, SPAdes and MEGAHIT quality filtered assemblies, 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 18.11%, 1.03%, 1.03%, and 0.97% of their corresponding assemblies length respectively.

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 1.02%, 1.03%, 1.03%, and 1.03% of their corresponding assemblies length respectively.

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 1.02%, 1.04%, 1.04%, and 1.04 % of their corresponding assemblies length respectively.

Metagenome assemblies account for the majority of reads

To further evaluate assemblies, we mapped the quality filtered reads to each assembly using bwa [8]. 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 treatments assemblies, the full set of trimmed reads were used for mapping. Default parameters were used, and both paired ends and singletons were mapped. Samtools [14] was used for format conversion from SAM to BAM format, and also to calculate the percentage of mapped reads. We conclude that assemblies account for the majority of reads.

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 4.26 %, 0.75%, 1.06%, and 0.79% of each corresponding assembly length respectively. These percentages reflect errors or low coverage reads. See Table 4 for more details.

The de novo assemblies recover unexpected genomic sequence from the mock community

We aligned the unaligned contigs of each assembly with different treatments to the unaligned contigs of IDBA-UD assembly using quality filtered treatment. Using quality filtered reads, the genome fraction equals 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 which is IDBA-UD unaligned contigs using quality filtered reads. 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, we found few hits. Mapping the unaligned reads to the quality filtered reads, we found some hits [how to summarize] with e-value $\leq 1e-6$. Mapping the unaligned reads to the reference genomes, we found some hits [how to summarize?] . This means that the reference genome contains these reads and they are missed by the assemblers.

Discussion

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 assemblers and using different treatments. Misassembled length is less than 1.3% for all assemblers and using different treatments. We conclude that assembly works well although there are some rooms for improvements including enhancing accuracy, and decreasing time and memory requirements. Velvet shows the least performance in terms of accuracy and time, and memory utilizations. However, the difference between other assemblers are not significant. Hence, more investigations are needed to decide what assembler to use prior assembly. Such analysis is left for our future work.

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. We conclude that digital normalization and partitioning are beneficial steps for assembly to reduce time and memory utilities.

Digitial normalization and partitioning do not affect misassemblies and unalignments

Except for Velvet assemblies, misassemblies are not affected by digital normalization and partitioning. Mapping the unaligned contigs of different assemblies and different treatments 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 assemblers with various treatments and they are likely to be unknowns, new assemblies, or contamination. This indicates that digital normalization and partitioning enhance assembly time and memory requirements without affecting assembly accuracy.

Assembly recovers content not in the reference

We have parts of the reference misassembled by every assembler if the experiment is theoretically correct. We have parts that are not aligned to the reference although they exist in the reference as seen by blast [CHECK RESULTS WITH TITUS]. So what are the content recovered by assembly and not in the reference?

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 (2013) Aligning sequence reads, clone sequences and assembly contigs with bwa-mem. arXiv preprint arXiv:13033997 .

9. 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 .

10. 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.

11. 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.

12. 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.

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

14. Heng L, Handsaker B, Wysoker A, Fennell T, Ruan J, et al. (2009) The sequence alignment/map format and samtools. Bioinformatics 25: 2078-2079.

Figure Legends

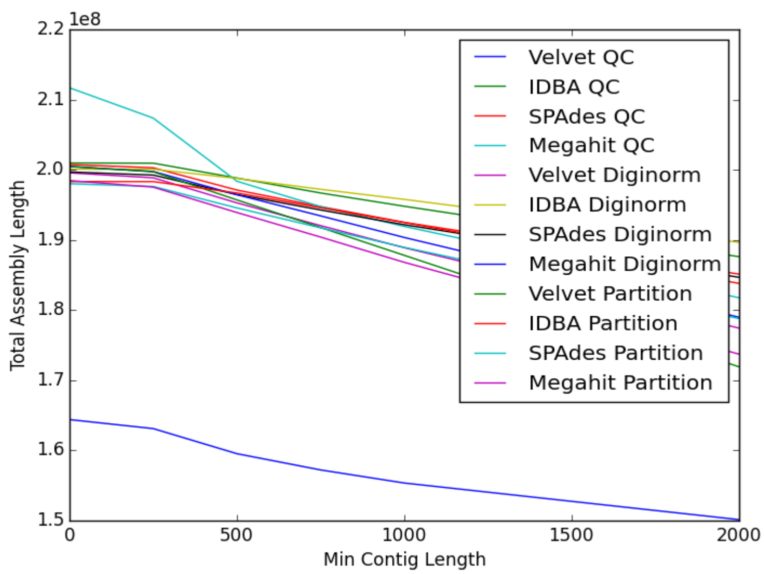

Figure 1. *Total Length of assemblies in basepairs based on different min contigs length.*

Tables

Supporting Information Legends

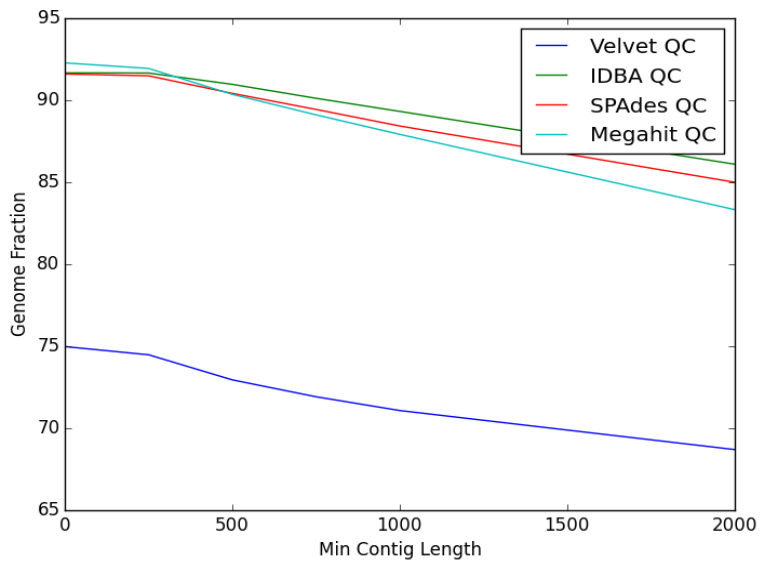

Figure 2. *Genome Fraction of assemblies in basepairs based on different min contigs length.*

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. Misassemblies

Assembly	Quality Filtering	Digital Normalization	Partition
(1) Velvet			
Misassemblies	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	0.066%	0.089%	0.0911%
Indels Length	50,190	181,453	346,988
Indels Percentage	0.031%	0.093%	0.178%
(3) IDBA-UD			
Misassemblies	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	0.082%	0.116%	0.117%
Indels Length	30,433	43,358	42,523
Indels Percentage	0.015%	0.022%	0.022%
(2) SPAdes			
Misassemblies	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	0.093%	0.124%	0.120%
Indels Length	27,328	32,783	21,516
Indels Percentage	0.014%	0.017%	0.011%
(4) MEGAHIT			
Misassemblies	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	0.078%	0.11%	0.10%
Indels Length	15,298	18,195	16,517
Indels Percentage	0.008%	0.009%	0.008%

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	4.26%	1.73%	1.99 %
(2) IDBA-UD			
No. of Unaligned Sequences	1,490,609	1,738,371	2,297,377
Percentage	0.75%	0.87%	1.17%
(3) SPAdes			
No. of Unaligned Sequences	2,100,555	2,439,158	2,804,006
Percentage	1.06%	1.24%	1.44%
(4) MEGAHIT			
No. of Unaligned Sequences	1,559,300	2,082,881	2,747,427
Percentage	0.79%	1.06%	1.42%

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

Treatment/Quality Metric	Quality Filtering	Digital Normalization	Partition
(1) Velvet			
Genome 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			
Genome Fraction	-	91.53%	94.72%
Unaligned Length	-	498,299	1,320,036
Percentage of unaligned	-	7.46%	19.76%
(3) SPAdes			
Genome 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			
Genome 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