F1000Research Evaluating Metagenome Assembly on a Complex Community

Sherine Awad[1], **Luiz Irber**[2], **and C. Titus Brown**[3]

[1,2,3]**Department of Population Health and Reproduction, University of California, Davis, California**

Abstract

Metagenome assembly is a challenging problem due to the biodiversity of the microorganisms. Most assemblers are designed for whole genome assembly and not capable of dealing with metagenomic samples. However, in order to decide which assembler works best for metagenome, we need to evaluate metagenome assembly generated by each assembler.

In this paper, we used three assemblers ; IDBA-UD, SPAdes, and Megahit to assemble metagenome mock community data and evaluate the assembly process in terms of resources utilization, assembly quality, genome fraction covered, duplication ratio, misassemblies and partial alignments.

The results show only small differences in content recovery between assemblers. However, Megahit is much faster and produces shorter contig lengths than IDBA-UD and SPAdes.

Introduction

Metagenomics refers to sequencing of DNA from a mixture of organisms, often from an environmental or uncultured sample. Unlike whole genome sequencing, metagenomics targets a mixture of genomes, which introduces metagenome-specific challenges in analysis. Most approaches to analyzing metagenomic data rely on mapping or comparing sequencing reads to reference sequence collections. However, reference databases contain only a small subset of microbial diversity (cite: geba), and the much of the remaining diversity is evolutionarily distant and search techniques may not access it.

As sequencing capacity increases and sequence data is generated from many more environmental samples, metagenomics is increasingly using de novo assembly techniques to generate new reference genomes and metagenomes. There are a number of metagenome assemblers that are widely used. However, evaluating the results of these assemblers is challenging due to the general lack of good quality reference metagenomes. Below, we evaluate three commonly assemblers - SPAdes, IDBA, and MEGAHIT - on a mock community containing 64 species of microbes with known genomes.

Moya et al. in [1] evaluated metagenome assembly using simulated two 454 viral metagenome and six assemblers. The assemblies were evaluated based on several metrics including N50, percentages of reads assembled, accuracy when compared to the reference genome. In addition to, chimeras per contigs and the effect of assembly on taxonomic and functional annotations.

Mavromatis et al. in [2] provided a benchmark study to evaluate the fidelity of metagenome process methods. The study used simulated metagenomic data sets constructed at different complexity levels. The datasets were assembled using Phrap v3.57, Arachne v.2 [3] and JAZZ. [4] This study evaluates assembly, gene prediction, and binning methods. However, the study did not evaluate the assembly quality against a reference genome.

Rangwala et al. in [5] presented an evaluation study of metagenome assembly. The study used a de Bruijn graph based assembler ABYSS [6] to assemble simulated metagnome reads of 36 bp. The data set is classified at different complexity levels. The study compares the quality of the assembly of the data sets in terms of quality measures of contigs length, assembly accuracy. The study also took into consideration the effect of kmer size and the degree of chimericity. However, the study evaluated the assembly based on one assembler, and did not evaluate assembly against several assemblers. Also, both previous studies used simulated data, which may lack confounders of assembly such as sequencing artifacts and GC bias.

Shakya et al. (2013) constructed a complex synthetic community of organisms by mixing DNA isolated from individual cultures of 64 bacteria and archaea. In addition to performing 16s amplicon analysis and doing 454 sequencing, the authors shotgun sequenced the mixture with Illumina (@cite). While the authors concluded that this metagenomic sequencing generally outperformed amplicon sequencing, they conducted a mapping based analysis rather than an assembly based analysis.

In this paper, we evaluate metagenome assembly on the Illumina data set from Shakya et al. (2013) using three assemblers; IDBA-UD [7], SPAdes [8], and MEGAHIT [9]. These three assemblers were chosen because they are actively used and highly cited.

SPAdes [8] is an assembler for both single-cell and standard (multicell) assembly. (More description here @CTB.)

IDBA-UD [7] 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 also applies an error correction step to correct reads of high-depth regions that can be aligned to high confident contigs.

MEGAHIT [9] is a newer approach that constructs a succinct de Bruijn graph using multiple k-mer sizes, and uses a novel “mercy k-mer” approach that preserves low-abundance regions of reads. It also can use GPUs to accelerate the graph construction.

Below, we evaluate the performance of these three assemblers using the synthetic (“mock”) community data from the Shakya et al. study. The performance of each assembler is compared in terms of resource utilization, covered genome fraction, duplication ratio, gene recovery, contig misassembly, and contig length.

In contrast to some other evaluations (assemblathon 2), we work with a synthetic community with a known answer, do not tune the default parameters, and investigate memory and runtime as a key part of the process. This mimics default user experience. (Expand.)

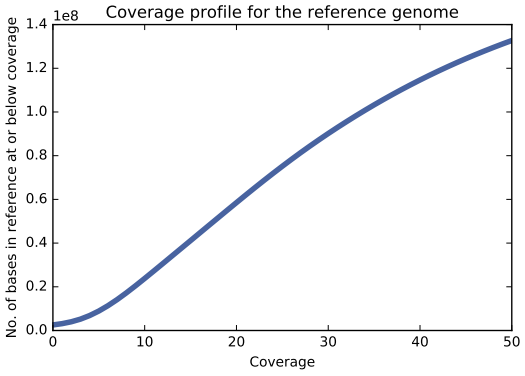

Figure 1. Cumulative coverage profile for the reference metagenome, based on read mapping.

Datasets

We used a diverse mock community data set constructed by pooling DNA from 64 species of bacteria and archaea

and sequencing them with Illumina HiSeq. The raw data set consisted of 109,629,496 reads from Illumina HiSeq 101 bp paired-end sequencing (2x101) with an untrimmed total length of 11.07 Gbp and an estimated fragment size of 380 bp [10].

The original reads are available through the NCBI Sequence Read Archive at Accession SRX200676. We received the 64 reference genomes from the original authors. They consist of 205.6 Mbp of assembled genomes in 64 contigs, and are available for download at https://dx.doi.org/10.6084/m9.figshare.1506873.v2

Methods

The analysis code and run scripts for this paper are available at: https://github.com/dib-lab/2015-metagenome-assembly/. The scripts and overall pipeline were examined by the first and senior authors for correctness. In addition, the bespoke reference-based analysis scripts were tested by running them on a single-colony *E. coli* MG1655 data set with a high quality reference genome [11].

Quality Filtering

We removed adapters with Trimmomatic v0.30 in paired-end mode with the Truseq adapters [12], using light quality score trimming as recommended in MacManes, 2014 [13].

Reference Coverage Profile

To evaluate how much of the reference metagenome was contained in the read data, we used `bwa aln` to map reads to the reference genome. We then calculated how many reference bases were covered by how many mapped reads (custom script `coverage-profile.py`).

Assemblers

We assembled the quality-filtered reads using three different assemblers: IDBA-UD [7], SPAdes [8], and MEGAHIT [9]. For IDBA-UD v1.1.1 [7], we used `-pre_correction` to perform pre-correction before assembly and -r for the pe files.

For SPAdes v3.9.0 [8], we used `-meta -pe1-12 -pe1-s` where `-meta` is recommended when working with metagenomic data sets, `-pe1-12` specifies the interlaced reads for the first paired-end library, and `-pe1-s` provides the orphan reads remaining from quality trimming.

For MEGAHIT [9], we used -l 101 `-m 3e9 -cpu-only` where `-l` is for maximum read length, `-m` is for max memory in bytes to be used in constructing the graph, and `-cpu-only` to use only the CPU and no GPUs. We also used `-presets meta-large` for large and complex metagenomes, and `-12` and `-r` to specify the interleaved-paired-end and single-end files respectively.

All three assemblies were executed on the same high-memory buy-in node on the Michigan State University High Performance Compute Cluster, and we recorded RAM and CPU time of each assembly job using the `qstat` utility at the end of each run.

Unless otherwise mentioned, we eliminated all contigs less than 500 bp from each assembly prior to further analysis.

Mapping

We aligned all quality-filtered reads to the reference metagenome with bwa aln (v0.7.7.r441) [14]. We aligned paired-end and orphaned reads separately. We then used samtools (v0.1.19) [15] to convert SAM files to BAM files for both paired-end and orphaned reads. To count the unaligned reads, we included only those records with the "4" flag in the SAM files [15].

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) [14]. Then we used samtools to retrieve the reads that mapped to the unaligned contigs.

k-mer Presence

In order to examine k-mer presence for a k-mer size of 20, we built a k-mer counting table from the given quality filtered reads using `load-into-counting.py` from khmer [**?**]. Then we calculate abundance distribution of the k-mers in the quality filtered reads using the premade k-mer counting table using `abundance-dist.py`. We followed the same approach to examine k-mer presence in assemblies.

Assembly analysis using Nucmer

We used the NUCmer tool from MUMmer3.23 [16] to align assemblies to the reference genome with options `--coords -p`. Then we parsed the generated ".coords" file using a custom script `analyze_assembly.py`, and calculated several analysis metrics across all three assemblies at two alignment identities, 95% and 99%.

Reference-based analysis of the assemblies

We analyzed metrics for three different sets of contigs, based on the NUCmer alignments. We used the unfiltered NUCmer alignments for the analyses termed "ambiguous." We also subjected the alignments to two different filtering criteria, "best-hit" and "no-misassemblies." In the best-hit approach, among all alignments of a contig, we took into consideration the longest alignment with an identity above a specified identity threshold (either 95% or 99%). In the no-misassemblies approach, we only counted contigs that have precisely one alignment within the reference.

In all approaches, we flag a base in the reference genome as "covered" if it is contained in a kept alignment. We define the duplication ratio as the percentages of bases in the reference covered by two or more kept alignments. We define misassemblies as those contigs that are divided into different parts when mapped to the reference. The

Table 1. Jaccard containment of the reference in the reads

k-mer size	% reference in reads
21	96.8%
31	95.9%
41	94.9%
51	94.1%

number of misassembled contigs is equal to the number of aligned contigs (both totally and partially) in the ambiguous approach, minus the number of aligned contigs in the no-misassemblies approach.

All approaches have a non-zero duplication ratio within the reference because we do not explicitly discard contigs that map to the same location in the reference.

Gene annotations using Prokka

We used prokka [17] to annotate the reference genome using `-metagenome`. Then we parsed the testasm.tbl output file to get the coordinates of CDS genes and searched the alignments for how many genes were contained in those alignments.

Results

The raw data is high quality

The reads contains 11,072,579,096 bp (11.07 Gbp) in 109,629,496 reads with 101.0 average length (2x101bp Illumina HiSeq).

Trimming removed 686,735 reads (0.63%). After trimming, we retained 108,422,358 paired reads containing 10.94 Gbp with an average length of 100.9 bases. A total of 46.56 Mbp remained in 520,403 orphan reads with an average length of 89.5 bases. In total, the quality trimmed data contained 10.98 Gbp in 108,942,761 reads. This quality trimmed ("QC") data set was used as the basis for all further analyses.

The reference metagenome is mostly present in the reads

We next evaluated the fraction of the reference genome covered by at least one read (see Methods for details). Quality filtered reads cover 203,058,414 (98.76%) bases of the reference metagenome (205,603,715 bp total size). Figure 1 shows the cumulative coverage profile of the reference metagenome, and the percentage of bases with that coverage. Most of the reference metagenome was covered at least minimally; only 3.33% of the reference metagenome had mapping coverage <5, and 1.24% of the bases in the reference were not covered by any reads in the QC data set.

In order to evaluate reconstructability with De Bruijn graph assemblers, we next examined k-mer containment of the reference in the reads for k of 21, 31, 41, and 51 (Table 1). The k-mer overlap decreases from 96.8% to 94.1% as the k-mer size increases. This could be caused by low coverage of some portions of the reference and/or variation between the reads and the reference.

Some individual reference genomes are poorly represented in the reads.

Table 2. Top uncovered genomes

Genome	Read coverage	21-mer presence
B. xenovorans	XX.XX%	91.6%
D. vulgaris DP4	91.98%	79.1%
T. thermophilus HB27	91.06%	79.7%
E. faecalis V583	78.14%	68.0%
F. nucleatum	47.30%	17.8%

To see if specific reference genomes exhibited low coverage, we analyzed read mapping coverage and 21-mer containment for individual genomes. Of the 64 reference genomes used in the metagenome, 59 had a per-base mapping coverage above 95% and a 21-mer containment in the QC reads above 95%. The remaining five varied significantly in both metrics (Table 3), with *F. nucleatum* the lowest – only 47.30% of the bases in the reference genome are covered by one or more mapped reads, and only 17.8% of the 21-mers in the reference genome are present in the reads at any abundance.

We next did a 51-mer containment analysis of each reference genome in the reads. 99% or more of the constituent 51-mers for 46 of the 64 reference genomes were present in the reads, suggesting that each entire genome was present at some minimal coverage.

We excluded the remaining 18 genomes from any comparative analysis of assembly quality, because interpreting coverage and misassembly analysis for these genomes would be unproductive. See “Strain variation analysis”, below, for further analysis of these genomes.

MEGAHIT is the fastest and lowest-memory assembler evaluated

Table 3. Running Time and Memory Utilization

Assembler	Runtime	RAM
MEGAHIT	0hr 56m	34.4 GB
IDBA-UD	17h 12m	149.1 GB
SPAdes	41h 14m	391.5 GB

We ran three commonly used metagenome assemblers on the QC data set: IDBA-UD, SPAdes, and MEGAHIT. We recorded the time and memory usage of each (Table 3). In computational requirements, MEGAHIT outperformed both SPAdes and IDBA-UD considerably, producing an assembly in one hour – approximately 17 times faster than IDBA and 42 times faster than SPAdes. MEGAHIT used only 34.4 GB of RAM – 1/5 to 1/11th the memory used by IDBA and SPAdes, respectively.

The assemblies contain most of the raw data

Table 4. Read and k-mer exclusion from assemblies

Assembly	Unmapped Reads	51-mers omitted
IDBA	3,328,674 (3.05%)	3.4%
SPAdes	3,879,573 (3.56%)	4.2%
MEGAHIT	5,848,494 (5.37%)	3.7%

We assessed read inclusion in assemblies by mapping the QC reads to the length-filtered assemblies and counting the remaining unmapped reads. Depending on the assembly, between 3.3 million and 5.9 million reads (3.0-5.4%) did not map to the assemblies (Table 4). Here, the MEGAHIT assembly was distinguished by representing 2 million fewer reads than the IDBA and SPAdes assemblies.

k-mer inclusion, however, was more closely matched by the assemblies, with all three assemblies containing between 95.8% and 96.6% of the 51-mers in the k-mer trimmed reads.

Much of the reference is covered by the assemblies

Table 5. Contig coverage of reference with "loose" alignment conditions.

Assembly	% covered	% Duplication
MEGAHIT	96.2%	0.72%
SPAdes	95.8%	0.99%
IDBA	95.6%	0.88%

(Update this section to include k-mer overlap.)

We next evaluated the extent to which the assembled contigs recovered the "known/true" metagenome sequence by aligning each assembly to the adjusted reference (Table 5). Each of the three assemblers generates contigs that cover more than 95.6% of the reference metagenome at high identity (99%) with little duplication (0.72-0.99%).

At 99% identity with the loose mapping approach, approximately 1.8% of the reference is missed by all three assemblers, while 0.9% is uniquely covered by MEGAHIT, 0.6% is uniquely covered by SPAdes, and 0.4% is uniquely covered by IDBA.

The generated contigs are broadly accurate

When counting only the best (longest) alignment per contig at a 99% identity threshold, each of the three assemblies recovers more than 87.3% of the reference, with MEGAHIT recovering the most – 93.8% of the reference (Table 6).

(CTB: add discussion of accuracy/mismatches/indels here.)

Table 6. Contig accuracy measured by reference coverage with strict alignment.

Assembly	% covered
MEGAHIT	93.8%
IDBA	89.5%
SPAdes	87.3%

Some statistics.

We computed the NGA50 for each individual genome and assembly. (see Figure 2). The NGA50 statistics for individual genomes vary widely, but MEGAHIT yields the lowest NGA50 for 42 of the 46 genomes, while SPAdes yields the highest NGA50 for 38 of the 46 genomes.

Longer contigs are less likely to be chimeric.

Table 7. Chimeric contigs by contig length.

Assembly	> 50kb	> 5kb	> 500 bp
MEGAHIT	0%	3.9%	15.5%
IDBA	0%	6.3%	22.2%
SPAdes	0%	8.8%	18.4%

We evaluated the rate of chimerism in contigs at three different contig length cutoffs: 500bp, 5kb, and 50kb (Table 7). We found that the percentage of contigs that match to the genomes of two or more different species drop as the minimum contig size increases, to the point where contigs longer than 50kb had no chimeric contigs in any assembly.

Upon further analysis, 99% of the chimeric contigs greater than 5kb were formed between genomes from pairs of closely related species. Similarly, 80% of the chimeric contigs at the lowest cutoff (500 bp) were between pairs or groups of closely related species.

There are many reads not represented in the reference

7.6 million reads (7.00%) from the QC data set did not map anywhere in the reference.

Plan:

- analyze k-mer content of 7.6m reads with sbt gather against RefSeq
- extract genomes that match there, along with mircea reference
- do an analysis showing that there is higher match to some strain vars, maybe also with assembled contigs; make point that we may be missing true strains.
- muse on possible explanations;
- discuss shewanella situation
- shall we bring in the 454 data?

Reference table 8.

Table 8. RefSeq strains detected in unalignable contigs

Match	Fraction
NZ_LN831027.1 F. nucleatum NZ_AXNV01000001.1 F. nucleatum CTI-6 NZ_KI965381.1 F. nucleatum 13_3C NZ_JANA01000001.1 F. sp. OBRC1	0.13
NC_013968.1 H. volcanii DS2 plasmid	0.08
NC_006461.1 T. thermophilus HB8	0.03
NC_008751.1 D. vulgaris DP4 NC_002937.3 D. vulgaris str. Hildenborough	0.06
NC_003911.12 Ruegeria pomeroyi DSS-3	0.05
NC_003272.1 Nostoc sp. PCC 7120	0.07
NC_009665.1 Shewanella baltica OS185** **NC_011663.1 Shewanella baltica OS223****	0.07
NZ_CH959317.1 Sulfitobacter sp. NAS-14.1 NZ_CH959311.1 Sulfitobacter sp. EE-36	0.04
NZ_JNKC01000001.1 P. ruminis DSM 24773*	0.02
NC_010483.1 Thermotoga sp. RQ2	0.01
NZ_AFHH01000001.1 E. faecalis OG1X	0.01
NZ_JMLF01000001.1 D. radiodurans	0.01
(Total)	0.60

Discussion

Assembly recovers basic content sensitively and accurately.

All three assemblers performed well in assembling contigs from the content that was fully present in reads and k-mers. After length filtering, all three assemblies contained more than 95% of the reference (Table 5); even with removal of secondary alignments, more than 87% was recovered by each assembler (Table 6). About half the constituent genomes had an NG50 of 50kb or higher and an NG90 of 10kb (Table **??**), which, while low for current Illumina single-genome sequencing @cite, is sufficient to recover operon-level relationships for many genes.

(Mention genes / prokka analysis.)

Chimeric contigs form between closely related species.

A primary interest of this study is to examine chimerism, in which a single contig is formed from multiple genomes. Naively we expected chimerism to behave like misassembly, in which longer contigs would be more likely to contain large-scale errors. Surprisingly, we found less chimerism the longer the contigs were; the shorter contigs contained a high rate of chimerae, while contigs longer than 50kb had none (Table 7).

Analysis of the genomic pattern of chimeric contigs provides a possible explanation: the majority of chimeric contigs at all sizes are formed between closely related species or strain variants, suggesting that chimeric contigs are generated primarily where multiple species overlap in the assembly graph. However, the same complexity in the local assembly graph that leads to chimera may also limit the extension of contigs, resulting in shorter chimeric contigs.

MEGAHIT performs best by several metrics.

MEGAHIT is clearly the most efficient computationally, outperforming both SPAdes and IDBA by 5-10x in memory and 17-42x in time (Table 3). While the MEGAHIT assembly had 2m fewer reads mapping to it than the other assemblies (Table 1), MEGAHIT covered more of the reference genome with both loose and strict alignments (Table 5 and Table 6), with little duplication.

While MEGAHIT yields the lowest NGA50 of the three assemblers on this data set, MEGAHIT also had the lowest rate of chimerism. As discussed above, this may be an unavoidable tradeoff; either way, for complex populations, we believe conservative assembly is a feature rather than a bug.

Between the assemblers, the assembly content differs by only a small amount when loose alignments are allowed: all three assemblers miss more content (approximately 1.8% of the reference) than they generate uniquely (0.9% or less). This suggests that combining assemblies may have little value in terms of recovering additional metagenome content.

Conclusions

Assembly works well. There is no big difference between assemblers' performance in terms of assembly quality. In terms of cost, Megahit is much faster and utilizes less memory.

- If you have short-read sequencing only, use MEGAHIT.
- MEGAHIT can be useful for exploratory analysis of content as well.
- Metagenome assembly is good at recovering content of well covered regions. It is less good at long-range contiguity. It is better than reference based analysis at recovering content.
- Resequence mock community genomes, as in evolutionary studies.

Author contributions

In order to give appropriate credit to each author of an article, the individual contributions of each author to the manuscript should be detailed in this section. We recommend using author initials and then stating briefly how they contributed.

Competing interests

No competing interest to our knowledge.

Grant information

This work is funded by Moore and NIH.

Acknowledgments

Michael R. Crusoe

References

[1] Jorge F Vázquez-Castellanos, Rodrigo García-López, Vicente Pérez-Brocal, Miguel Pignatelli, and Andrés Moya. Comparison of different assembly and annotation tools on analysis of simulated viral metagenomic communities in the gut. *BMC genomics*, 15(1):1, 2014.

[2] Konstantinos Mavromatis, Natalia Ivanova, Kerrie Barry, Harris Shapiro, Eugene Goltsman, Alice C McHardy, Isidore Rigoutsos, Asaf Salamov, Frank Korzeniewski, Miriam Land, et al. Use of simulated data sets to evaluate the fidelity of metagenomic processing methods. *Nature methods*, 4(6):495–500, 2007.

[3] David B Jaffe, Jonathan Butler, Sante Gnerre, Evan Mauceli, Kerstin Lindblad-Toh, Jill P Mesirov, Michael C Zody, and Eric S Lander. Whole-genome sequence assembly for mammalian genomes: Arachne 2. *Genome research*, 13(1):91–96, 2003.

[4] Samuel Aparicio, Jarrod Chapman, Elia Stupka, Nik Putnam, Jer-ming Chia, Paramvir Dehal, Alan Christoffels, Sam Rash, Shawn Hoon, Arian Smit, et al. Whole-genome shotgun assembly and analysis of the genome of fugu rubripes. *Science*, 297(5585):1301–1310, 2002.

[5] Anveshi Charuvaka and Huzefa Rangwala. Evaluation of short read metagenomic assembly. *BMC genomics*, 12(2): 1, 2011.

[6] Jared T Simpson, Kim Wong, Shaun D Jackman, Jacqueline E Schein, Steven JM Jones, and Inanç Birol. Abyss: a parallel assembler for short read sequence data. *Genome research*, 19(6):1117–1123, 2009.

[7] Yu Peng, Henry C.M. Leung, S.M. Yiu, and Francis Y.L. Chin. Idba-ud: a de novo assembler for single-cell and metagenomic sequencing data with highly uneven depth. *Bioinformatics*, 28:1420–1428, 2012.

[8] Anton Bankevich, Sergey Nurk, Dmitry Antipov, Alexey A. Gurevich, Mikhail Dvorkin, Alexander S. Kulikov, Valery M. Lesin, Sergey I. Nikolenko, Son Pham, Andrey D. Prjibelski, Alexey V. Pyshkin, Alexander V. Sirotkin, Nikolay Vyahhi, Glenn Tesler, Max A. Alekseyev, and Pavel A. Pevzner. Spades: A new genome assembly algorithm and its applications to single-cell sequencing. *Journal of Computational Biology*, 19(5):455–477, 2012.

[9] Dinghua Li, Ruibang Luo, Chi-Man Liu, Chi-Ming Leung, Hing-Fung Ting, Kunihiko Sadakane, Hiroshi Yamashita, and Tak-Wah Lam. Megahit v1. 0: A fast and scalable metagenome assembler driven by advanced methodologies and community practices. *Methods*, 102:3–11, 2016.

[10] Shakya Migun, Christopher Quince, James Campbell, Zamin Yang, Christopher Schadt, and Mircea Podar. Comparative metagenomic and rrna microbial diversity characterization using archaeal and bacterial synthetic communities. *Enivromental Microbiology*, 15(6):1882–1899, 2013.

[11] H Chitsaz, JL Yee-Greenbaum, G Tesler, MJ Lombardo, CL Dupont, JH Badger, M Novotny, DB Rusch, LJ Fraser, NA Gormley, O Schulz-Trieglaff, GP Smith, DJ Evers, PA Pevzner, and RS Lasken. Efficient de novo assembly of single-cell bacterial genomes from short-read data sets. *Nat Biotechnol*, 29(10):915–21, 2011.

[12] Anthony M. Bolger, Marc Lohse, and Bjoern Usadel. Trimmomatic: A flexible trimmer for illumina sequence data. *Bioinformatics*, 30(15):2114–2120, 2014.

[13] Matthew D MacManes. On the optimal trimming of high-throughput mrna sequence data. *Frontiers in genetics*, 5: 13, 2014.

[14] Heng Li and Richard Durbin. Fast and accurate short read alignment with burrows–wheeler transform. *Bioinformatics*, 25(14):1754–1760, 2009.

[15] Heng Li, Bob Handsaker, Alec Wysoker, Tim Fennell, Jue Ruan, Nils Homer, Gabor Marth, Goncalo Abecasis, Richard Durbin, et al. The sequence alignment/map format and samtools. *Bioinformatics*, 25(16):2078–2079, 2009.

[16] Stefan Kurtz, Adam Phillippy, Arthur L Delcher, Michael Smoot, Martin Shumway, Corina Antonescu, and Steven L Salzberg. Versatile and open software for comparing large genomes. *Genome biology*, 5(2):1, 2004.

[17] Torsten Seemann. Prokka: rapid prokaryotic genome annotation. *Bioinformatics*, page btu153, 2014.

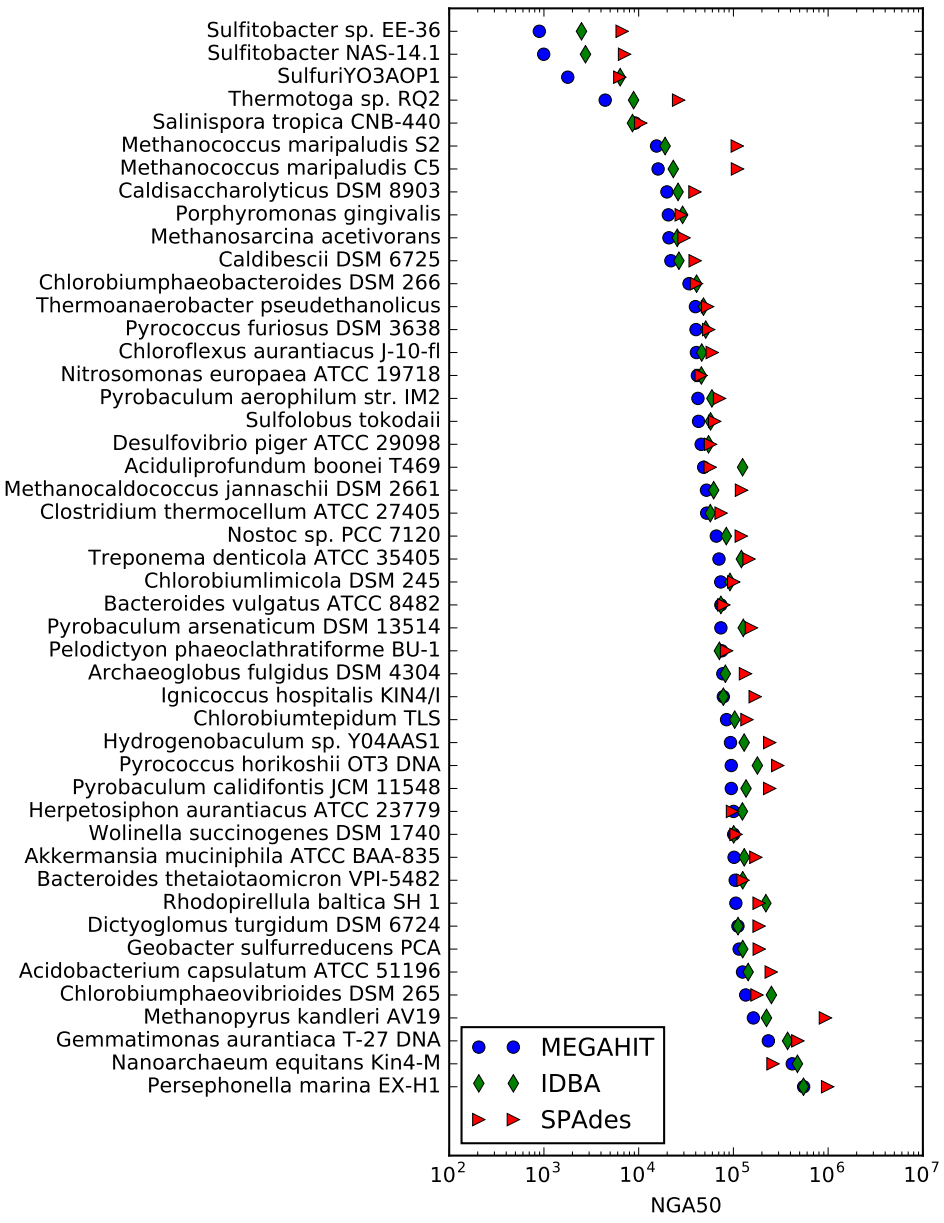

Figure 2. NGA50 by genome and assembler.