MSc Thesis - Benchmark of *De Novo* Short Read Assembly Strategies for Metagenomics

Ino de Bruijn
Supervisor: Assistant Prof. Anders Andersson

April 19, 2013

Contents

Chapter 1

Introduction

Metagenomics, the sequencing of environmental DNA, has demonstrated to be a promising approach for the discovery and investigation of microbes that cannot be cultured in the laboratory [7] as well as for the study of both free-living microbial communities [1] and microbial communities inside other organisms [24, 9].

In a typical shotgun metagenomics experiment the DNA of a community is isolated and high throughput sequencing is performed on a random sample of the isolated DNA [20]. The reads can either be analyzed as such, by e.g. blast searches against reference databases to obtain a functional profile of the microbial community [28], or they can be assembled to form longer stretches of DNA stemming from the same or closely related organisms that can subsequently be analyzed with regards to phylogenetic affiliation and functional properties. The output of the assembly process often includes scaffolds, contigs and unassembled reads [17]. One of the problems with assembling is that chimeric contigs or scaffolds may be formed. Closely related sequences are more likely to form chimeras and since closely related strains often occur in the same environment this is a challenge. Also, it is difficult to determine whether the formation of a chimera is natural due to homologous recombination or an error in the assembly process [29]. Another problem with assembly is variations in gene content among closely related strains, since a gene inserted in a subpopulaton will cause conflicting assembly results [8]. After assembling the reads, a process called binning is performed, where the resulting scaffolds and contigs are assigned to phylogenetically related groups. Finally, gene calling and functional annotations are performed on the scaffolds.

In our studies several strategies for *de novo* assembly of metagenomics have been evaluated. Illumina short read libraries have *in silico* been shown to work well on communities of medium complexity [18], therefore we have chosen to assess the assembly strategies for Illumina paired short reads sepecifically. In previous

studies mostly *in silico* metagenomic data sets have been used [23, 17]. In contrast the community of our study is an *in vitro* simulated metagenome consisting of 59 species with completed or nearly completed genomes so the quality of our assesment is not dependent on the realisticness of read simulators. An even and uneven distribution of the 59 species were created in vitro. The community has been sequenced with different type of library preparations to be able to test the difference in library preparation as well. The following assembly programs have been tested: Velvet [33], Meta-Velvet [22], Newbler [25], Minimus2 [27], Ray Meta [**?**] and Bambus2 [11]. The quality of the assemblies have been evaluated by mapping the constructed contigs or scaffolds to the collection of reference genomes, hereafter referred to as the reference metagenome. In addition two pipelines have been constructed, one to perform the assemblies and another to perform the validation given there is a reference metagenome available.

Chapter 2

Related Work

In a study by Mavromatis et al. [17] three genome assemblers were evaluated: Phrap [4], Arachne [3] and Jazz [2]. For the evaluation three artificial communities were constructed of low, medium and high complexity by selecting Sanger reads from 113 isolate genomes. The low complexity community had one dominating population with several low-abundance ones, the medium more than one dominating population and the complex community had no dominating population at all. Resulting contigs were evaluated on chimericity and length distribution. Compared to using the original reads for gene annotation, assembly was demonstrated to give not more than 20% increase in accurate gene prediction and a slightly better increase for inaccurate and missed genes, however 700 bp long Sanger reads were used in this study. This approach of using artificial communities has subsequently been used in adapted versions by several other assembly evaluation papers [23, 18]. In the benchmark by Pignatelli and Moya [23] the reads of the artificial communities were changed from Sanger to 454 and Illumina. For the Illumina reads, SSAKE [30] and Velvet were used to perform the assembly. No difference in chimericity between using the simulated 454 reads or the Illumina reads was spotted. The main cause of chimericity was sequence similarity of the organisms, no relation with genome coverage was found. At the functional level metagenomic assembly turned out to be counterproductive compared to using the original reads for annotation. Mende et al. [18] used a metagenome of 10, 100 and 400 species with simulated reads of Illumina, 454 and Sanger with sequencing depth based on sequencing cost for each technology. All of the technologies provided similar coverage for 10 species. Illumina was superior for 100 species due to the higher coverage one can get for a similar price. Sanger performed best for 400 species because of longer read length. Sanger reads were assembled with Arachne; 454 reads with Celera [21] and Illumina reads with SOAPdenovo [14]. In contrast to the study of Pignatelli and Moya [23] a year earlier, contigs now did turn out to improve functional annotation of the metagenome. Furthermore using Illumina paired end

data to determine contig links and construct scaffolds, although introducing more chimerism, resulted in an even better functional annotation. Beyond using simulated reads or real reads of *in silico* communities there has not been a comparison of assembly algorithms using an *in vitro* community yet. In vitro communities have been used previously with success to assess DNA extraction techniques for sequencing a low complexity community of nine bacterial genera [31], an oral community [5], the human gut [32] and the human microbiome [10]. The advantage of using an *in vitro* community for assembly evaluation is that one does not have to rely on the correctness of sequencing simulators, the assessment can thus be as good as the similarity of the *in vitro* community to a real community.

Chapter 3

Methods and Materials

To determine the quality of metagenomic assembly a mock community of species with known genomes was constructed *in vitro* and sequenced with Illumina. The resulting reads have been assembled using a combination of Velvet, Meta-Velvet, Ray, Minimus2, Newbler and Bambus2 resulting in nineteen different assembly strategies (see Figure 3.1 and Table 3.1). The strategies stem from current literature and our own ideas.

3.1   Mock community

The mock community consists of 49 bacterial and 10 archaeal species with finished or nearly completed genomes (Table 3). The species have been chosen such that there are a number of closely related organisms and more distant ones. The number of species is about equal to the number of species one would find in the human gut. The abundances of DNA from each species have been fixed in two types of configurations before sequencing. In the first configuration, the even configuration, all species have approximately equal genome copy numbers. In the second configuration, the uneven configuration, the phyla are mixed in proportions similar to log-normal distributions of phyla in soil [6]. The samples have been prepared with the Nextera 50ng sample preparation kit. The entire reference metagenome's size is about 195Mb.

3.2   Quality trimming

Before assembling the reads one often starts with pre-processing them by quality trimming and/or removing PCR duplicates. Mende et al. [18] demonstrated that quality trimming could drastically improve the assembly. Before each assembly the same quality trimming procedure has been performed. For quality trimming

the program sickle was used (see Table 1). Reads were trimmed from the 3' end if the average quality score was below 20 in a window of 10 bases.

3.3   Assembly

In the assembly procedure reads are combined into contiguous sequences called contigs. Contigs can afterwards be joined using paired read information into longer scaffolds. In the scaffolding process contigs might be extended and repeats might be solved so scaffolding is not restricted to just the ordering of contigs.

There are a plethora of different assemblers available and by pre-processing reads and combining different assemblers an even larger amount of assembly strategies is possible. Velvet is one of the most used assembly programs and was therefore included in this assessment. Velvet's metagenomic counterpart, Meta-Velvet, is performed after executing Velvet so it is possible to determine how the metagenomic specific parameters improve the assembly. Another popular assembler that has recently received an update for metagenomics is Ray (TODO: cite). Ray is based on MPI and is runnable over multiple nodes distributing both memory and processor load, which makes it an ideal candidate for large metagenomic projects.

3.3.1   Contiging

Velvet, Ray and Meta-Velvet all use a de Bruijn graph to determine overlaps between reads. This involves cutting up the reads in sizes of a specified kmer size and let edges represent overlaps between kmers. This way the graph, or the computational requirements, grow with the number of unique kmers in the library instead of the number of reads. For a more elaborate description of de Bruijn Graphs for sequence assembly see [19]. The resulting contigs are constructed by following paths in the graph. The paths that can be unambiguously followed are called unitigs. Ambiguous paths can be solved by using coverage information or paired-end information. Contigs thus consist of one or multiple unitigs. Choosing the right kmer size is important. A shorter k gives more coverage but at the same time the risk increases that the kmer occurs multiple times within the genome, or in multiple genomes. A larger k can overcome this problem if it is larger than the multiply occurring region, thereby creating unique kmers instead of one kmer with more coverage.

How the assemblers differ

Velvet, Ray and Meta-Velvet differ in the way the graph is traversed. Velvet, meant for single genomes, looks for one coverage peak in the coverage distribution and tries to follow that, where the main idea is that the genome is approximately uniformly covered. Nodes in the graph below a certain coverage threshold are considered errors and ones with high coverage repeats. Meta-Velvet looks for multiple peaks in the coverage distribution. Each genome should have a distinct coverage peak due to its genome copy number and genome length being different from the other genomes in the metagenome. Meta-Velvet makes use of that property. Ray looks for 'seeds' in the graph and extends those seeds iteratively weighting choises by the number of reads supporting a certain path. The seeds are unitigs in the graph with a specific coverage. The metagenomic update to Ray changes the seed selection by looking at the coverage peak in the graph locally instead of globally.

3.3.2   Merging

A way to get the advantage from both short and long kmers is by merging contigs generated in multiple assemblies with different kmer lengths. This is possible with Newbler, as done by Luo et al. [15], or with Minimus2, as done by for instance the Rnnotator pipeline [16]. Both Newbler and Minimus2 use an Overlap-Layout-Consensus method to merge contigs [27, 19].

3.3.3   Scaffolding

For the scaffolding procedure Bambus2 was chosen since it was one of the better scaffolders for single genomes in the GAGE assessment paper [26] and is suitable for metagenomes as well [11]. For a flow diagram of previously mentioned approaches see Figure 3.1. A total of twelve assembly strategies from the flow diagram have been tested. See Table 3.1 for an overview of the assembly strategies, Table 1 for versions of each program and Table 2 for the parameters of each strategy.

Assembly strategy name	Contiging	Merging	Scaffolding
velvetnoscaf	Velvet	-	-
velvetscaf	Velvet	-	Velvet
minimus2velvetnoscaf	Velvet	Minimus2	-
newblervelvetnoscaf	Velvet	Newbler	-
bambus2velvetnoscaf	Velvet	-	Bambus2
bambus2minimus2velvetnoscaf	Velvet	Minimus2	Bambus2
bambus2newblervelvetnoscaf	Velvet	Newbler	Bambus2
metavelvetnoscaf	Meta-Velvet	-	-
metavelvetscaf	Meta-Velvet	-	Meta-Velvet
minimus2metavelvetnoscaf	Meta-Velvet	Minimus2	-
newblermetavelvetnoscaf	Meta-Velvet	Newbler	-
bambus2metavelvetnoscaf	Meta-Velvet	-	Bambus2
bambus2minimus2metavelvetnoscaf	Meta-Velvet	Minimus2	Bambus2
bambus2newblermetavelvetnoscaf	Meta-Velvet	Newbler	Bambus2
raynoscaf	Ray	-	-
rayscaf	Ray	-	Ray
minimus2raynoscaf	Ray	Minimus2	-
newblerraynoscaf	Ray	Newbler	-

Table 3.1: Assembly strategies

Input reads

illumina_pe

Process reads

quality_trim

processed_reads

De Bruijn Graph

velvet

meta_velvet

Output

contigs_single_kmer

scaffolds_single_kmer

Merging contigs

cd_hit

newbler

minimus2

merged_contigs

Scaffolding

bwa

contigs_with_linkage_info
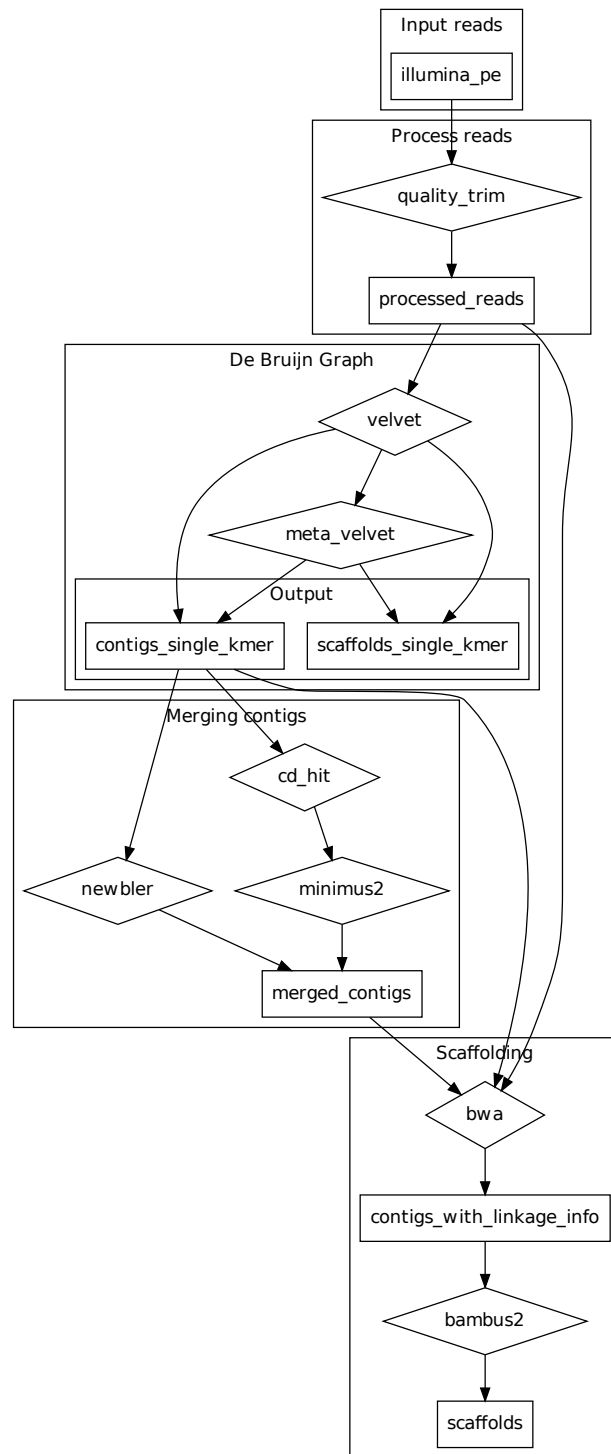
bambus2

scaffolds

Figure 3.1: Assembly strategies using a combination of Velvet, Meta-Velvet, Ray, Minimus2, Newbler and Bambus2.

3.4   Validation

The validation of a metagenomic assembly in case a reference metagenome is available often focuses on one or more of the following points:

- contig or scaffold length distribution

- contig/scaffold coverage of the reference metagenome

- chimericity of the contigs/scaffolds

- functional annotation

- phylogenetic classification

This study focusses on the first three points, since those are expected to improve the functional annotation and the phylogenetic classification. Mende et al. [18] showed that this was indeed the case for functional annotation.

3.4.1   Length distribution

A variety of metrics are used in this validation. For the contig/scaffold length distribution of an assembly a common metric called the L50 value is used. L50 is a median value weighted by contig length. Half of all the bases in the assembly is in contigs equal to or larger than L50.

3.4.2   Coverage of the reference metagenome

For determining how well the assemblies matched the reference metagenome the assemblies were mapped against the reference metagenome using MUMmer 3.1 [12]. MUMmer finds maximal exact matches longer than l and clusters them if they are no more than g nucleotides apart. The alignments are afterwards extended for each cluster if the combined length of its matches is at least c. The alignments are extended in between the matches of the cluster and on the ends using a Smith-Waterman dynamic programming algorithm. The MUMmer package contains multiple scripts that make use of this approach. NUCmer (<u>NUC</u>leotide MUM<u>mer</u>) is a script included in the MUMmer package for DNA sequence alignment of a set of query contigs against a set of reference contigs. The command for NUCmer used was: `nucmer --maxmatch -c65 -g90 -l20`. The maxmatch parameter makes sure all exact matches are used, whether they are unique or not, so contigs that consist only of a shared region or a repetitive element will be included in the alignments as well. Afterwards the script `show-coords` was used on the

resulting alignment file to extract information about each alignment such as its location in both the query and the reference, percent identity, percent similarity and percent of the reference and query covered.

To indicate how well the metagenome is covered by the assembled contigs we use a metric called genome contig coverage and metagenome contig coverage. For each genome in the reference metagenome, genome contig coverage is determined by dividing the number of bases that are covered by contigs mapping to it by the total number of bases in the genome. Metagenome contig coverage is the number of bases covered by all contigs mapping to it divided by the total number of bases in the metagenome. In both cases only the best alignment for each contig is used. The best alignment is the one with the highest purity score (for an explanation of the purity score see Section 3.4.3. In case a contig aligns equally well to multiple locations, only the first alignment is considered. This could for instance be a contig that represents a repeat or a shared region between genomes. By only counting one such alignment not resolving a repeat or shared region is penalized, do note that in general there's little difference between counting all such alignments or only one in bacterial and archaeal genomes [**?**].

3.4.3 Chimericity

In previous research by Mavromatis et al. [17] chimericity of each contig was determined by the reads mapping to it. This could be done because reads were selected from finished genome projects, thus the origin of each read is known. At each taxonomic level from strain to domain the contigs were assigned to the phylogenetic group where the majority of reads stem from. The chimericity was then defined as the number of reads that belong to another phylogenetic group divided by the total number of reads mapping to the contig. Mende et al. [18] noted that certain reads might actually belong to multiple species since there are shared regions between species. Therefore they proposed a metric called contig score, which is determined by mapping each contig with blastn to the original genomes, taking the Highest Scoring Segment Pair (HSP) and multiplying the percent identity times the percent contig coverage. We use a similar metric, determined with MUMmer and referred to as purity instead to indicate its relation to chimericity. For each alignment the purity p is determined by multiplying percent contig coverage by percent identity divided by 10,000 to get the purity as a ratio between 0 and 1. The purity of a contig is the maximum purity of all its alignments. Purity of an entire assembly is computed as well. If n is the number of contigs, $l_1, ..., l_n$ the length of each contig in nucleotides and $p_1, ..., p_n$ the purity of each contig then purity P of an entire assembly is calculated as:

$$P = \frac{\sum_{i=1}^{n} l_i * p_i}{\sum_{i=1}^{n} l_i} \qquad (3.1)$$

For both scaffolds and contigs, only nucleotides are counted when determining l.

3.5 Pipelines

The vast amount of assemblies and validations that had to be performed on a variety of libraries gave rise to a need for automation. This resulted in the development of two pipelines: MetAssemble and MASMVALI (Metagenomic ASseMbly VALIdation). The former for performing the assemblies, the latter for validating them given there is a reference metagenome available.

3.5.1 MetAssemble

MetAssemble implements the strategies in Table 3.1 and Figure 3.1. The MetAssemble pipeline was written as a set of rules in GNU make [?]. GNU make was chosen for a number of reasons:

- It integrates skipping computation of previously computed files. Recomputation is unfortunately only based on modification date, not on the actual file contents, but it is good enough for our purposes.

- GNU make is widely available on Unix and Unix-like systems.

- After the validation one would want to share only the commands to do a specific assembly i.e. not the entire pipeline. Doing a dry-run in GNU make gives you the exact commands that generated a certain file.

One of the biggest problems in metagenomic assembly are the computational demands [?]. The size of the reads and the number of reads as well as the complexity and the structure of the sequenced community affect the computational demands. Therefore it is diffcult to determine the resource usage in advance. In our benchmark several strategies consist of parts with different computational requirements. Ray for instance runs over multiple nodes whereas merging with Newbler or Minimus2 requires one node with a lot of memory. Claiming all these computational resources for the entire run of the pipeline would be a waste. To circumvent this issue an approach where each step in the pipeline is scheduled as a separate job using a job scheduler has been chosen. A library was developed for GNU make to schedule rules as jobs using either sbatch or qsub. The jobs are scheduled with dependencies on other jobs and the resource usage set for the rule [?].

3.5.2 MASMVALI

After the assembly has been completed, the assembly can be validated by alignment against the reference metagenome. MASMVALI (Metagenomic ASseMbly VALIdation) is a pipeline written in Python[**?**] that calculates the statistics described in Section 3.4. As minimum input it takes the alignments found by MUMmer, a reference metagenome and an assembly. MASMVALI as a pipeline can generate a HTML report that shows a collection of the plots that are used in this thesis [**?**]. It is also possible to use MASMVALI as a package to do more analyses on generated assemblies using the available statistics. See for an example the available ipython notebook [**?**] [**?**] [**?**].

Chapter 4

Results

4.1 Mock community libraries

All assemblies in Table 3.1 have been performed on the following libraries:

50ng even

> Community prepared with 50 ng Nextera library prepartion kit and equal genome copy numbers. The library has been sequenced with Illumina and contains ~10M paired reads, each single read has a length of 100.

50ng uneven

> Community prepared with 50 ng Nextera library prepartion kit and phyla mixed similar to log-normal distributions of phyla in soil. The library has been sequenced with Illumina and contains ~10M paired reads, each single read has a length of 100.

The original reads of each library were mapped against the reference genome using BWA [13] and default values. Afterwards duplicates were removed with MarkDuplicates from Picard 1.77 (see also Table 1). In Figure 4.1 for the even library, the even spread of the coverage over all the genomes in the metagenome can be observed. The median coverage over all bases is 10. The uneven library covers a couple of genomes very well 4.2. The median coverage over all bases is 3.
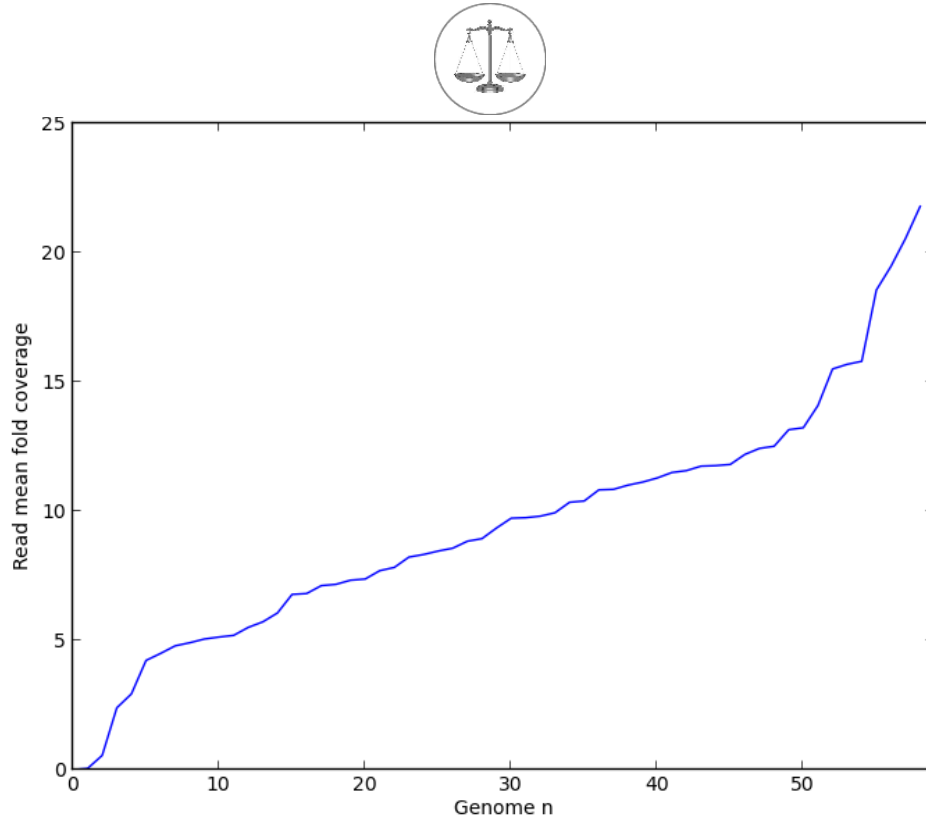
Figure 4.1: Read mean coverage of 50ng even per genome. Median coverage over all bases is 10.

4.2 Length distribution

In Figure 4.3 all assembly strategies are compared over length distributions. Each line represents a single assembly and the dots are L50 values with the cutoff set at 100 bases. For all strategies that use a single kmer, the assembly that had the highest L50 value was used for the plot.

It can be seen that a high L50 value alone is not enough to determine the best length distribution. In the contiging strategies (first row) one can see that Meta-Velvet scores very well on L50, but the sum of bases is a lot worse compared to Ray and Velvet especially for the uneven sample. Depending on what size of contigs one plans to use for post-processing the best choice differs. Longer than 100, velvet and to a lesser extent ray give more total bases, but longer than 1000 Meta-Velvet would seem a better candidate. In the second row in Figure 4.3, the
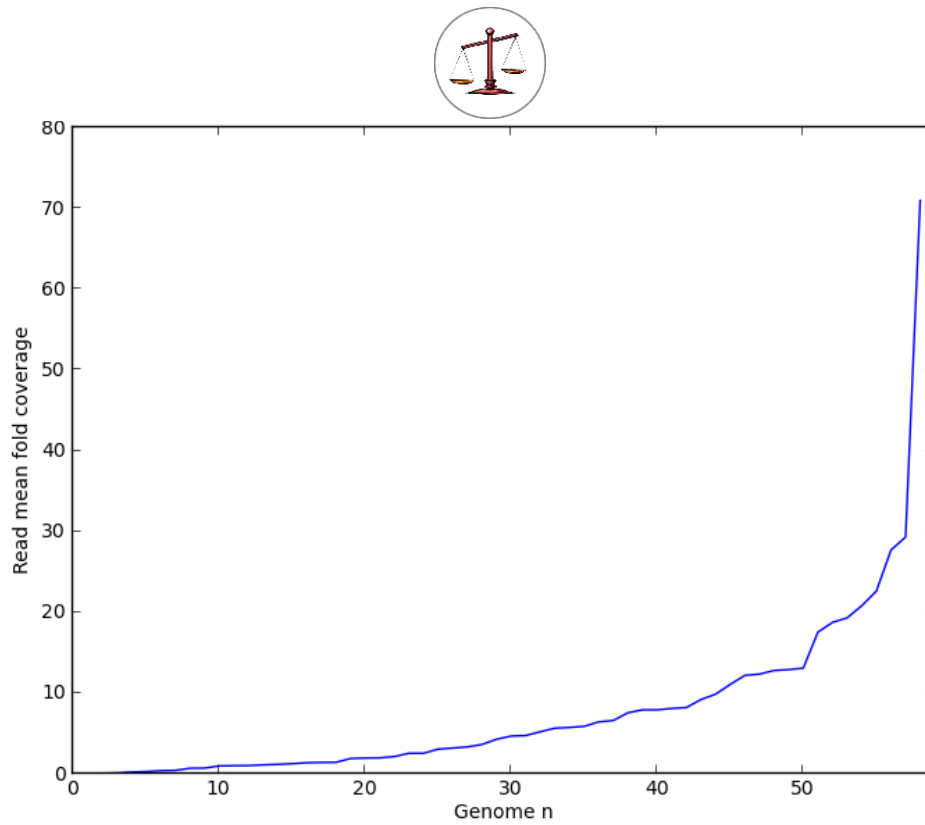
Figure 4.2: Read mean coverage of 50ng uneven per genome. Median coverage over all bases is 3.

merging strategies show that the distributions become more similar, increasing L50 and bringing up the sum.
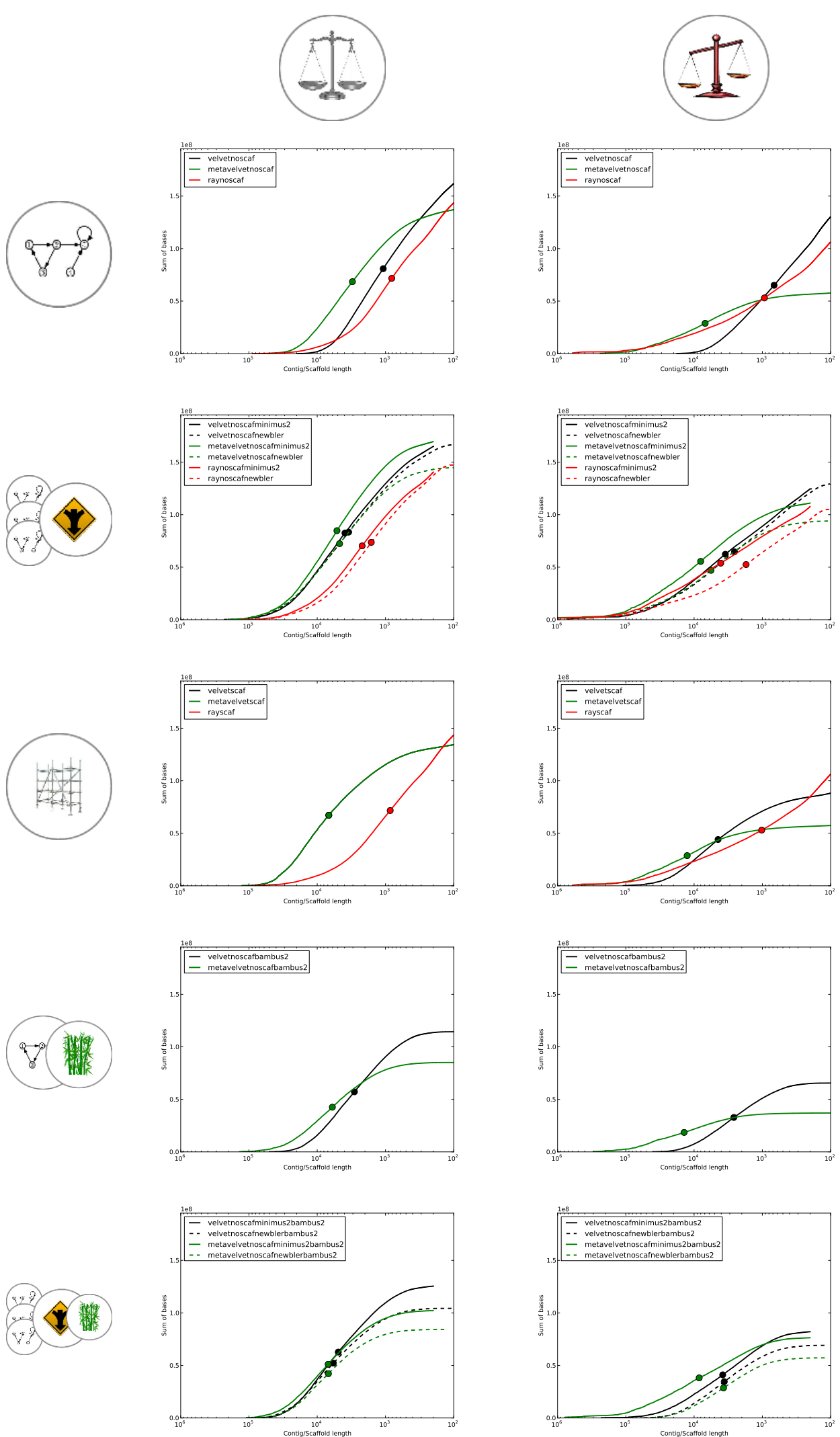
4.2.1 Merging

Figure 4.3: Sum of bases as a function of contig length, x-axis is log-scaled and inverted.

4.3   Metagenome coverage

4.4   Chimericity

The assemblies have been mapped against the reference metagenome using MUMmer as described in Section 3.4. In Figure **??** a comparison of the assemblies of 50ng even on purity and L50 can be seen. The upper right corner contains the assembly that has the least amount of chimericity and the best contig lengths. The assemblies seem to perform a lot better for smaller k. One would expect to find a sweet spot for k where lowering k would result in more kmer coverage but more connectivity in the de Bruijn graph and thus possibly impurer or even shorter contigs. Upping k would reduce coverage too much to increase contig length. Such a spot is indeed visible for *velvetnoscaf*,*metavelvetnoscaf* and *velvetscaf* in the top left corner although the purity does not decrease substantially for smaller k, whereas the L50 doubles or even triples. The contig extension seems to be bound by kmer coverage. Another peculiarity on first sight is that *metavelvetnoscaf* seems to perform identically to *velvetscaf*. The results are similar because for both these assemblies the parameter `-exp_cov auto` was used for `velvetg` and the scaffolding parameter of Velvet and Meta-Velvet's default parameters apparently don't improve the assembly. Meta-Velvet does not improve the Velvet assembly, because it is based on the idea that genomes have different coverage and tries to follow paths in the graph with identical coverage. Since all genomes have similar coverage `velvetg -exp_cov auto` performs equally well as `meta-velvetg -exp_cov auto`. In a library with uneven genome abundances, following paths of identical coverage in the graph does improve the assembly compared to using one expected coverage as can be seen in Figure **??**, where assemblies of an unevenly balanced community are compared. Why only setting the expected coverage and not the scaffolding of Velvet increases L50 is unclear. For merging the assemblies of different kmers into one, Newbler outperforms Minimus2 both on purity and on L50. An even greater L50 length can be acquired by scaffolding afterwards with Bambus2, which performs exceptionally well. The best assemblies are acquired by merging Meta-Velvet assemblies with Minimus or Newbler and scaffolding them with Bambus2. Scaffolding of Meta-Velvet is significantly worse performing on purity than Bambus2. For the uneven community, the best assemblies are done by a combination of Meta-Velvet and Bambus2. Merging contigs before scaffolding is disadvantageous in this case. The distribution of the points from both Meta-Velvet and Velvet follow a more distorted path compared to the even community. There seem to be some specific clusters visible but no explanation for the cause of this behaviour has been found.
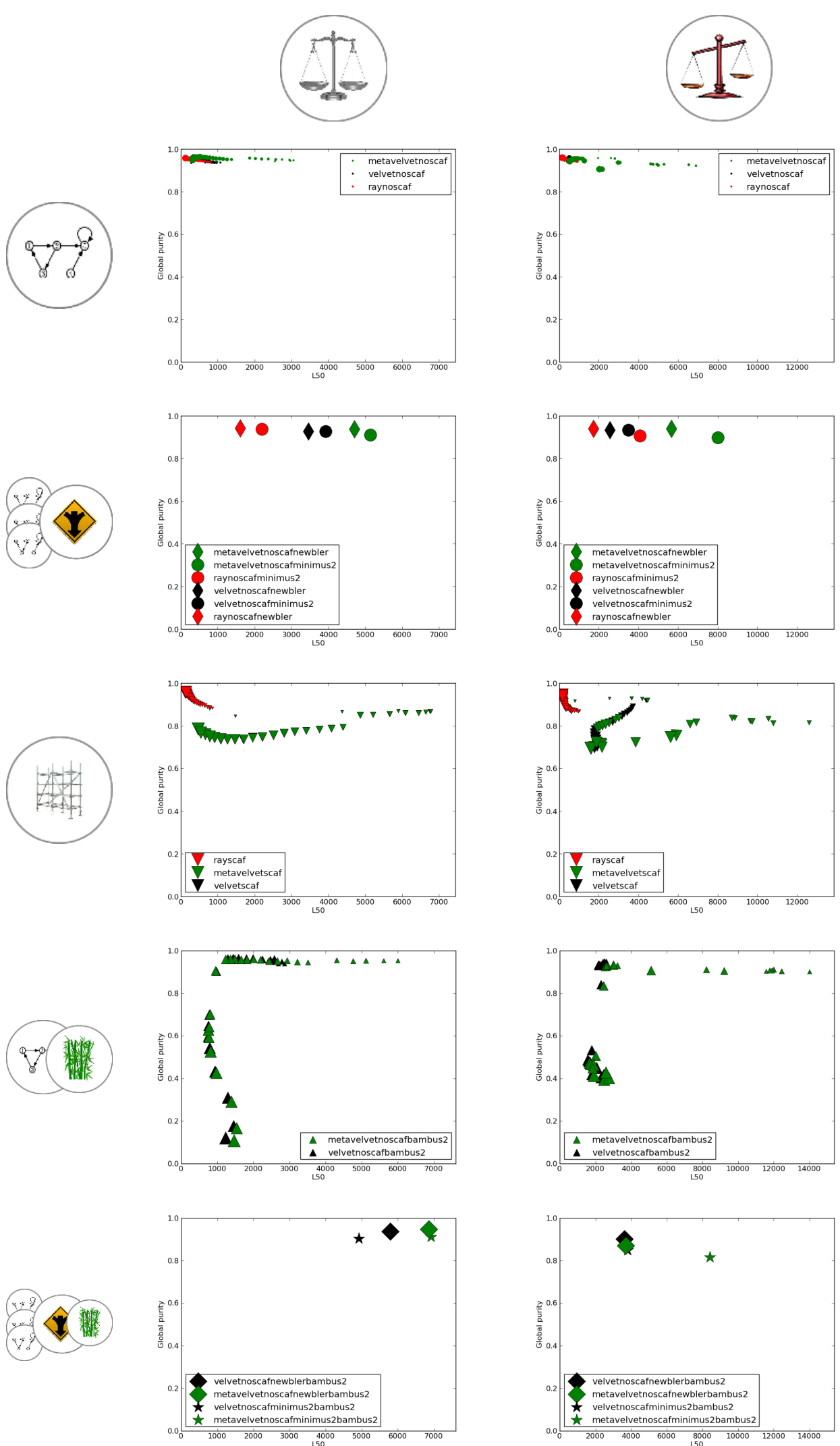
19

Figure 4.4: Global purity as a function of L50. Compared over all assembly strategies.

Chapter 5

Discussion

Discuss.

Bibliography

[1] A.F. Andersson and J.F. Banfield. Virus population dynamics and acquired virus resistance in natural microbial communities. *Science*, 320(5879):1047–1050, May 2008. doi: 10.1126/science.1157358.

[2] S. Aparicio, J. Chapman, E. Stupka, N. Putnam, J.M. Chia, P. Dehal, A. Christoffels, S. Rash, S. Hoon, A. Smit, M.D. Gelpke, J. Roach, T. Oh, I.Y. Ho, M. Wong, C. Detter, F. Verhoef, P. Predki, A. Tay, S. Lucas, P. Richardson, S.F. Smith, M.S. Clark, Y.J. Edwards, N. Doggett, A. Zharkikh, S.V. Tavtigian, D. Pruss, M. Barnstead, C. Evans, H. Baden, J. Powell, G. Glusman, L. Rowen, L. Hood, Y.H. Tan, G. Elgar, T. Hawkins, B. Venkatesh, D. Rokhsar, and S. Brenner. Whole-genome shotgun assembly and analysis of the genome of fugu rubripes. *Science*, 297(5585):1301–1310, Aug 2002. doi: 10.1126/science.1072104.

[3] S. Batzoglou, D.B. Jaffe, K. Stanley, J. Butler, S. Gnerre, E. Mauceli, B. Berger, J.P. Mesirov, and E.S. Lander. ARACHNE: a whole-genome shotgun assembler. *Genome Res*, 12(1):177–189, Jan 2002. doi: 10.1101/gr.208902.

[4] M. de la Bastide and W.R. McCombie. Assembling genomic DNA sequences with PHRAP. *Curr Protoc Bioinformatics*, Chapter 11:Unit11.4, Mar 2007. doi: 10.1002/0471250953.bi1104s17.

[5] P.I. Diaz, A.K. Dupuy, L. Abusleme, B. Reese, C. Obergfell, L. Choquette, A. Dongari-Bagtzoglou, D.E. Peterson, E. Terzi, and L.D. Strausbaugh. Using high throughput sequencing to explore the biodiversity in oral bacterial communities. *Mol Oral Microbiol*, 27(3):182–201, Jun 2012. doi: 10.1111/j.2041-1014.2012.00642.x.

[6] J.R. Doroghazi and D.H. Buckley. Evidence from GC-TRFLP that bacterial communities in soil are lognormally distributed. *PLoS One*, 3(8):e2910, 2008. doi: 10.1371/journal.pone.0002910.

[7] J.A. Eisen. Environmental shotgun sequencing: its potential and challenges for studying the hidden world of microbes. *PLoS Biol*, 5(3):e82, Mar 2007. doi: 10.1371/journal.pbio.0050082.

[8] S.J. Hallam, K.T. Konstantinidis, N. Putnam, C. Schleper, Y. Watanabe, J. Sugahara, C. Preston, J. de la Torre, P.M. Richardson, and E.F. DeLong. Genomic analysis of the uncultivated marine crenarchaeote cenarchaeum symbiosum. *Proc Natl Acad Sci U S A*, 103(48):18296–18301, Nov 2006. doi: 10.1073/pnas.0608549103.

[9] M. Hess, A. Sczyrba, R. Egan, T.W. Kim, H. Chokhawala, G. Schroth, S. Luo, D.S. Clark, F. Chen, T. Zhang, R.I. Mackie, L.A. Pennacchio, S.G. Tringe, A. Visel, T. Woyke, Z. Wang, and E.M. Rubin. Metagenomic discovery of biomass-degrading genes and genomes from cow rumen. *Science*, 331(6016): 463–467, Jan 2011. doi: 10.1126/science.1200387.

[10] Human Microbiome Project Consortium. A framework for human microbiome research. *Nature*, 486(7402):215–221, Jun 2012. doi: 10.1038/nature11209.

[11] S. Koren, T.J. Treangen, and M. Pop. Bambus 2: scaffolding metagenomes. *Bioinformatics*, 27(21):2964–2971, Nov 2011. doi: 10.1093/bioinformatics/btr520.

[12] S. Kurtz, A. Phillippy, A.L. Delcher, M. Smoot, M. Shumway, C. Antonescu, and S.L. Salzberg. Versatile and open software for comparing large genomes. *Genome Biol*, 5(2):R12, 2004. doi: 10.1186/gb-2004-5-2-r12.

[13] H. Li and R. Durbin. Fast and accurate long-read alignment with burrows-wheeler transform. *Bioinformatics*, 26(5):589–595, Mar 2010. doi: 10.1093/bioinformatics/btp698.

[14] R. Li, H. Zhu, J. Ruan, W. Qian, X. Fang, Z. Shi, Y. Li, S. Li, G. Shan, K. Kristiansen, S. Li, H. Yang, J. Wang, and J. Wang. De novo assembly of human genomes with massively parallel short read sequencing. *Genome Res*, 20(2):265–272, Feb 2010. doi: 10.1101/gr.097261.109.

[15] C. Luo, D. Tsementzi, N. Kyrpides, T. Read, and K.T. Konstantinidis. Direct comparisons of illumina vs. roche 454 sequencing technologies on the same microbial community DNA sample. *PLoS One*, 7(2):e30087, 2012. doi: 10.1371/journal.pone.0030087.

[16] J. Martin, V.M. Bruno, Z. Fang, X. Meng, M. Blow, T. Zhang, G. Sherlock, M. Snyder, and Z. Wang. Rnnotator: an automated de novo transcriptome

assembly pipeline from stranded RNA-seq reads. *BMC Genomics*, 11:663, 2010. doi: 10.1186/1471-2164-11-663.

[17] K. Mavromatis, N. Ivanova, K. Barry, H. Shapiro, E. Goltsman, A.C. McHardy, I. Rigoutsos, A. Salamov, F. Korzeniewski, M. Land, A. Lapidus, I. Grigoriev, P. Richardson, P. Hugenholtz, and N.C. Kyrpides. Use of simulated data sets to evaluate the fidelity of metagenomic processing methods. *Nat Methods*, 4(6):495–500, Jun 2007. doi: 10.1038/nmeth1043.

[18] D.R. Mende, A.S. Waller, S. Sunagawa, A.I. Jarvelin, M.M. Chan, M. Arumugam, J. Raes, and P. Bork. Assessment of metagenomic assembly using simulated next generation sequencing data. *PLoS One*, 7(2):e31386, 2012. doi: 10.1371/journal.pone.0031386.

[19] J.R. Miller, S. Koren, and G. Sutton. Assembly algorithms for next-generation sequencing data. *Genomics*, 95(6):315–327, Jun 2010. doi: 10.1016/j.ygeno.2010.03.001.

[20] J.L. Morgan, A.E. Darling, and J.A. Eisen. Metagenomic sequencing of an in vitro-simulated microbial community. *PLoS One*, 5(4):e10209, 2010. doi: 10.1371/journal.pone.0010209.

[21] E.W. Myers, G.G. Sutton, A.L. Delcher, I.M. Dew, D.P. Fasulo, M.J. Flanigan, S.A. Kravitz, C.M. Mobarry, K.H. Reinert, K.A. Remington, E.L. Anson, R.A. Bolanos, H.H. Chou, C.M. Jordan, A.L. Halpern, S. Lonardi, E.M. Beasley, R.C. Brandon, L. Chen, P.J. Dunn, Z. Lai, Y. Liang, D.R. Nusskern, M. Zhan, Q. Zhang, X. Zheng, G.M. Rubin, M.D. Adams, and J.C. Venter. A whole-genome assembly of drosophila. *Science*, 287(5461):2196–2204, Mar 2000.

[22] T. Namiki, T. Hachiya, H. Tanaka, and Y. Sakakibara. Metavelvet: An extension of velvet assembler to de novo metagenome assembly from short sequence reads. In *ACM Conference on Bioinformatics, Computational Biology and Biomedicine*, 2011.

[23] M. Pignatelli and A. Moya. Evaluating the fidelity of de novo short read metagenomic assembly using simulated data. *PLoS One*, 6(5):e19984, 2011. doi: 10.1371/journal.pone.0019984.

[24] J. Qin, R. Li, J. Raes, M. Arumugam, K.S. Burgdorf, C. Manichanh, T. Nielsen, N. Pons, F. Levenez, T. Yamada, D.R. Mende, J. Li, J. Xu, S. Li, D. Li, J. Cao, B. Wang, H. Liang, H. Zheng, Y. Xie, J. Tap, P. Lepage, M. Bertalan, J.M. Batto, T. Hansen, D. Le Paslier, A. Linneberg, H.B.

Nielsen, E. Pelletier, P. Renault, T. Sicheritz-Ponten, K. Turner, H. Zhu, C. Yu, S. Li, M. Jian, Y. Zhou, Y. Li, X. Zhang, S. Li, N. Qin, H. Yang, J. Wang, S. Brunak, J. Dore, F. Guarner, K. Kristiansen, O. Pedersen, J. Parkhill, J. Weissenbach, MetaHIT Consortium, P. Bork, S.D. Ehrlich, and J. Wang. A human gut microbial gene catalogue established by metagenomic sequencing. *Nature*, 464(7285):59–65, Mar 2010. doi: 10.1038/nature08821.

[25] N.L. Quinn, N. Levenkova, W. Chow, P. Bouffard, K.A. Boroevich, J.R. Knight, T.P. Jarvie, K.P. Lubieniecki, B.A. Desany, B.F. Koop, T.T. Harkins, and W.S. Davidson. Assessing the feasibility of GS FLX pyrosequencing for sequencing the atlantic salmon genome. *BMC Genomics*, 9:404, 2008. doi: 10.1186/1471-2164-9-404.

[26] S.L. Salzberg, A.M. Phillippy, A. Zimin, D. Puiu, T. Magoc, S. Koren, T.J. Treangen, M.C. Schatz, A.L. Delcher, M. Roberts, G. Marcais, M. Pop, and J.A. Yorke. GAGE: A critical evaluation of genome assemblies and assembly algorithms. *Genome Res*, 22(3):557–567, Mar 2012. doi: 10.1101/gr.131383. 111.

[27] D.D. Sommer, A.L. Delcher, S.L. Salzberg, and M. Pop. Minimus: a fast, lightweight genome assembler. *BMC Bioinformatics*, 8:64, 2007. doi: 10. 1186/1471-2105-8-64.

[28] S.G. Tringe, C. von Mering, A. Kobayashi, A.A. Salamov, K. Chen, H.W. Chang, M. Podar, J.M. Short, E.J. Mathur, J.C. Detter, P. Bork, P. Hugen-holtz, and E.M. Rubin. Comparative metagenomics of microbial communities. *Science*, 308(5721):554–557, Apr 2005. doi: 10.1126/science.1107851.

[29] G.W. Tyson, J. Chapman, P. Hugenholtz, E.E. Allen, R.J. Ram, P.M. Richardson, V.V. Solovyev, E.M. Rubin, D.S. Rokhsar, and J.F. Banfield. Community structure and metabolism through reconstruction of microbial genomes from the environment. *Nature*, 428(6978):37–43, Mar 2004. doi: 10.1038/nature02340.

[30] R.L. Warren, G.G. Sutton, S.J. Jones, and R.A. Holt. Assembling millions of short DNA sequences using SSAKE. *Bioinformatics*, 23(4):500–501, Feb 2007. doi: 10.1093/bioinformatics/btl629.

[31] D. Willner, J. Daly, D. Whiley, K. Grimwood, C.E. Wainwright, and P. Hugenholtz. Comparison of DNA extraction methods for microbial commu-nity profiling with an application to pediatric bronchoalveolar lavage samples. *PLoS One*, 7(4):e34605, 2012. doi: 10.1371/journal.pone.0034605.

[32] G.D. Wu, J.D. Lewis, C. Hoffmann, Y.Y. Chen, R. Knight, K. Bittinger, J. Hwang, J. Chen, R. Berkowsky, L. Nessel, H. Li, and F.D. Bushman. Sampling and pyrosequencing methods for characterizing bacterial communities in the human gut using 16s sequence tags. *BMC Microbiol*, 10:206, 2010. doi: 10.1186/1471-2180-10-206.

[33] D.R. Zerbino and E. Birney. Velvet: algorithms for de novo short read assembly using de bruijn graphs. *Genome Res*, 18(5):821–829, May 2008. doi: 10.1101/gr.074492.107.

Program	Version	URL
sickle	commit 2013-4-3 e43598aa6bbb74b3254d2884a9dea1b0639d464	`https://github.com/najoshi/sickle`
Velvet	1.2.01	`https://github.com/dzerbino/velvet/commit/5d5b636c52e94fde745cb8bf753ffa134126e04d`
Meta-Velvet	1.2.01	
Minimus2	Amos commit 2012-3-6	`http://amos.git.sourceforge.net/git/gitweb.cgi?p=amos/amos;a=commit;h=7c70d6c36dba244ae0b266fc52a5206ab202a36c`
Newbler	RunAssembly 2.6 (20110517_1502)	
MUMmer	3.23	`http://sourceforge.net/projects/mummer/files/mummer/3.23/MUMmer3.23.tar.gz/download`
CD-HIT	4.5.7	
Picard	1.77	
bwa	0.6.1-r104	
Bambus2	Amos commit 2012-3-6	`http://amos.git.sourceforge.net/git/gitweb.cgi?p=amos/amos;a=commit;h=7c70d6c36dba244ae0b266fc52a5206ab202a36c`
samtoafg	Amos commit 2012-3-6	`http://amos.git.sourceforge.net/git/gitweb.cgi?p=amos/amos;a=commit;h=7c70d6c36dba244ae0b266fc52a5206ab202a36c`
Ray	2.1.0	`http://denovoassembler.sourceforge.net/`

Table 1: Program versions used for the analysis

Assembly strategy name	Program Parameters
velvetnoscaf	Run velveth $dir 31,84,2 -fastq -shortPaired $pairs.fastq and run velvetg -scaffolding no on the resulting directories.
velvetscaf	Run velveth $dir 31,84,2 -fastq -shortPaired $pairs.fastq and run velvetg -scaffolding yes -exp_cov auto on the resulting directories.
metavelvetnoscaf	Run velveth $dir 31,84,2 -fastq -shortPaired $pairs.fastq and run velvetg -scaffolding no -exp_cov auto -read_trkg yes && -scaffolding no on the resulting directories.
metavelvetscaf	Run velveth $dir 31,84,2 -fastq -shortPaired $pairs.fastq and run velvetg -scaffolding no -exp_cov auto -read_trkg yes && -scaffolding yes on the resulting directories.
raynoscaf	Run Ray -k $kmersize -i $pairs.fastq -o output_dir.
minimus2*	Concatenate contigs larger than 200 and run cd-hit-est -c 0.99 -i $concat.fasta -o $derep.fasta to remove similar sequences. Run toAmos -s $derep.fasta -o $derep.afg followed by minimus2 $derep. From http://ged.msu.edu/angus/metag-assembly-2011/velvet-multik.html
newbler*	Cut all contigs up in chunks of 1999 bases with an overlap of 1900 bases and run runAssembly -o $dir $cut-up-contigs.fasta. From [15].
bambus2*	Map paired reads with bwa using default parameters to contigs and remove duplicate reads with MarkDuplicates. Run samtoafg to convert to afg, import to Amos bnk with bank-transact and finally run goBambus2.

Table 2: Program parameters per assembly strategy.

Table 3: Mock community

No	Genome Name	Genome Size (bp)	Domain	Phylum	Class
1	Acidobacterium capsulatum ATCC 51196	4127356	Bacteria	Acidobacteria	Acidobacteriae
2	Akkermansia muciniphila ATCC BAA-835	2664102	Bacteria	Verrucomicrobia	Verrucomicrobiae
3	Anaerocellum thermophilum Z-1320, DSM 6725	2919718	Bacteria	Firmicutes	Clostridia
4	Bacteroides thetaiotaomicron VPI-5482	6293399	Bacteria	Bacteroidetes	Bacteroidia
5	Bacteroides vulgatus ATCC 8482	5163189	Bacteria	Bacteroidetes	Bacteroidia
6	Bordetella bronchiseptica RB50	5339179	Bacteria	Proteobacteria	Betaproteobacteria
7	Burkholderia xenovorans LB400	973113	Bacteria	Proteobacteria	Betaproteobacteria
8	Caldicellulosiruptor saccharolyticus DSM 8903	2970275	Bacteria	Firmicutes	Clostridia
9	Chlorobaculum tepidum TLS	2154946	Bacteria	Chlorobi	Chlorobia
10	Chlorobium limicola DSM 245	2763181	Bacteria	Chlorobi	Chlorobia
11	Chlorobium phaeobacteroides DSM 266	3133902	Bacteria	Chlorobi	Chlorobia
12	Chlorobium phaeovibrioides DSM 265	1966858	Bacteria	Chlorobi	Chlorobia
13	Chloroflexus aurantiacus J-10-fl	5258541	Bacteria	Chloroflexi	Chloroflexi
14	Clostridium thermocellum ATCC 27405	3843301	Bacteria	Firmicutes	Clostridia
15	Deinococcus radiodurans R1	3284156	Bacteria	Thermi	Deinococci
16	Desulfovibrio desulfuricans desulfuricans ATCC 27774	2873437	Bacteria	Proteobacteria	Deltaproteobacteria
17	Desulfovibrio piger ATCC 29098	2826240	Bacteria	Proteobacteria	Deltaproteobacteria
18	Dictyoglomus turgidum DSM 6724	1855560	Bacteria	Dictyoglomi	Dictyoglomia
19	Enterococcus faecalis V583	3359974	Bacteria	Firmicutes	Bacilli
20	Fusobacterium nucleatum nucleatum ATCC 25586	2174500	Bacteria	Fusobacteria	Fusobacteria
21	Gemmatimonas aurantiaca T-27T	4636964	Bacteria	Gemmatimonadetes	Gemmatimonadetes
22	Herpetosiphon aurantiacus ATCC 23779	6785430	Bacteria	Chloroflexi	Chloroflexi
23	Hydrogenobaculum sp. Y04AAS1	1559514	Bacteria	Aquificae	Aquificae
24	Leptothrix cholodnii SP-6	4909403	Bacteria	Proteobacteria	Betaproteobacteria
25	Nitrosomonas europaea ATCC 19718	2812094	Bacteria	Proteobacteria	Betaproteobacteria
26	Nostoc sp. PCC 7120	7211789	Bacteria	Cyanobacteria	unclassified
27	Pelodictyon phaeoclathratiforme BU-1	3018238	Bacteria	Chlorobi	Chlorobia
28	Persephonella marina EX-H1	2467104	Bacteria	Aquificae	Aquificae
29	Porphyromonas gingivalis ATCC 33277	2354886	Bacteria	Bacteroidetes	Bacteroidia
30	Rhodopirellula baltica SH 1	7145576	Bacteria	Planctomycetes	Planctomycetacia

Continued on next page

Table 3 Mock Community – continued from previous page

No	Genome Name	Genome Size (bp)	Domain	Phylum	Class
31	Rhodospirillum rubrum ATCC 11170	4406557	Bacteria	Proteobacteria	Alphaproteobacteria
32	Ruegeria pomeroyi DSS-3	4601053	Bacteria	Proteobacteria	Alphaproteobacteria
33	Salinispora arenicola CNS-205	5786361	Bacteria	Actinobacteria	Actinobacteria
34	Salinispora tropica CNB-440	5183331	Bacteria	Actinobacteria	Actinobacteria
35	Shewanella baltica OS185	5312910	Bacteria	Proteobacteria	Gammaproteobacteria
36	Shewanella baltica OS223	5358884	Bacteria	Proteobacteria	Gammaproteobacteria
37	Sulfitobacter sp. EE-36	3547243	Bacteria	Proteobacteria	Alphaproteobacteria
38	Sulfitobacter sp. NAS-14.1	4002069	Bacteria	Proteobacteria	Alphaproteobacteria
39	Sulfurihydrogenibium sp. YO3AOP1	1838442	Bacteria	Aquificae	Aquificae
40	Sulfurihydrogenibium yellowstonense SS-5	1534471	Bacteria	Aquificae	Aquificae
41	Thermoanaerobacter pseudethanolicus ATCC 33223	2362816	Bacteria	Firmicutes	Clostridia
42	Thermotoga neapolitana DSM 4359	1884562	Bacteria	Thermotogae	Thermotogae
43	Thermotoga petrophila RKU-1	1824357	Bacteria	Thermotogae	Thermotogae
44	Thermotoga sp. RQ2	1877693	Bacteria	Thermotogae	Thermotogae
45	Thermus thermophilus HB8	2116056	Bacteria	Thermi	Thermi
46	Treponema denticola ATCC 35405	2843201	Bacteria	Spirochaetes	Spirochaetes
47	Wolinella succinogenes DSM 1740	2110355	Bacteria	Proteobacteria	Epsilonproteobacteria
48	Zymomonas mobilis mobilis ZM4	2223497	Bacteria	Proteobacteria	Alphaproteobacteria
49	Archaeoglobus fulgidus DSM 4304	2178400	Archaea	Euryarchaeota	Archaeoglobi
50	Ignicoccus hospitalis KIN4/I	1297538	Archaea	Crenarchaeota	Thermoprotei
51	Methanocaldococcus jannaschii DSM 2661	1 664 970	Archaea	Euryarchaeota	Methanococci
52	Methanococcus maripaludis C5	1 780 761	Archaea	Euryarchaeota	Methanococci
53	Methanococcus maripaludis S2	1 661 137	Archaea	Euryarchaeota	Methanococci
54	Nanoarchaeum equitans Kin4-M	490 885	Archaea	Nanoarchaeota	Nanoarchaea
55	Pyrobaculum aerophilum IM2	2 222 430	Archaea	Crenarchaeota	Thermoprotei
56	Pyrobaculum calidifontis JCM 11548	2 009 313	Archaea	Crenarchaeota	Thermoprotei
57	Pyrococcus horikoshii OT3	1 738 505	Archaea	Euryarchaeota	Thermococci
58	Sulfolobus tokodaii 7(S311)	2 694 756	Archaea	Crenarchaeota	Thermoprotei
59	Treponema vincentii I	2512734	Bacteria	Spirochaetes	Spirochaetia