Metagenomic Assembly Validation of an *in vitro* Mock Community

Ino de Bruijn[1,2,*], Johannes Alneberg[1], Linda d'Amore, Neil Hall, Umer Z. Ijaz[3], Christopher Quince[3], Anders F. Andersson[1]

1 KTH Royal Institute of Technology, Science for Life Laboratoy, School of Biotechnology, Division of Gene Technology, Stockholm, Sweden

2 BILS Bioinformatics Infrastructure for Life Sciences, Stockholm, Sweden

3 University of Glasgow, Glasgow, UK

∗ **E-mail: Corresponding author ino@ino.pm**

Abstract

Single genome assembly algorithms have been benchmarked with real sequencing data in the assembly challenges Assemblethon and GAGE. The *de novo* metagenomic assembly algorithms have so far only been evaluated using similated reads. In this paper we present a benchmark using an *in vitro* mock community of 52 species with known reference genomes. The mock community was configured in two different abundance configurations: an even distribution and a log-normal distribution similar to distributions of phyla in soil. The communities were sequenced with Illumina HiSeq paired end mode. The data is openly available for other researchers to experiment on. Here, the reads have been used to test various assembly recipes i.e. a combination of Velvet, Meta-Velvet, Ray, Minimus2, Newbler and Bambus2 resulting in a total of twenty-one different assembly recipes. The assemblies are assessed on coverage of the reference genomes and the purity per contig. Purity is a ratio based on the best alignment per contig as determined with MUMmer. We show that there are many impure contigs constructed, both for the even community and the log-normal community. There is a clear tradeoff between contig length and contig purity. Velvet performs best in terms of purity and coverage of the references, while Velvet or Ray followed by a kmer merging step with Minimus2 or Newbler gives the longest contigs covering the references with a minor decrease in purity. We furthermore show that a simple rule of thumb for obtaining pure contigs is selecting those with high coverage.

Author Summary

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 [1] as well as for the study of both free-living microbial communities [2] and microbial communities inside other organisms [3,4].

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 [5]. 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 [6], 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 [7]. 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 [8]. 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 [9]. 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 recipes for *de novo* assembly of metagenomic data have been evaluated. In an *in silico* performed comparison between Illumina, Sanger and 454 on cost of sequencing and resulting coverage of microbial communities, Illumina short read libraries were shown to be the best for communities of medium complexity [10]. Therefore we have chosen to assess the assembly recipes for Illumina paired short reads sepecifically. In previous studies mostly *in silico* metagenomic data sets have been used [7, 11]. In contrast the community of our study is an *in vitro* simulated metagenome consisting of 52 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 52 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 [12], Meta-Velvet [13], Newbler [14], Minimus2 [15], Ray Meta [16] and Bambus2 [17]. 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.

In a study by [7] three genome assemblers were evaluated: Phrap [18], Arachne [19] and Jazz [20]. 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 up to 20% increase in accurate gene prediction and a slightly better increase for inaccurate and missed genes. Sanger reads of 700 bp were used. This approach of using artificial communities has subsequently been used in adapted versions by several other assembly evaluation papers [10, 11]. In the benchmark by [11] the reads of the artificial communities were changed from Sanger to 454 and Illumina. For the Illumina reads, SSAKE [21] 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. [10] used a metagenome of 10, 100 and 400 species with simulated reads of Illumina, 454 and Sanger where the number of reads for each technology was based on sequencing cost. The sequencing cost was kept constant. 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 [22] and Illumina reads with SOAPdenovo [23]. Similar to the study of [11] a year earlier, the authors concluded that assembly contigs improves 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 [24], an oral community [25], the human gut [26] and the human microbiome [27]. 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.

Materials and Methods

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 recipes (see Figure **??** and Table 1). The recipes stem from current literature and our own ideas.

Mock community

The sequenced mock community consisted of 59 species. 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 [28]. The samples have been prepared with the Nextera 1ng sample preparation kit. The entire reference metagenome's size is about 200Mb. Mock community preparations and sequencing were performed by our collaborators, Christopher Quince at University of Glasgow and Linda D'Amore and Neil Hall at Liverpool's Centre for Genomics Research. Sequencing of the even and uneven community resulted in about 7,9Gb and 6.7Gb respectively.

Quality trimming

Before assembling the reads one often starts with pre-processing them by quality trimming and/or removing PCR duplicates. [10] 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 **??**). Reads were trimmed from the 3' end if the average quality score was below 20 in a window of 10 bases. If the resulting read is shorter than 20 it is discarded. Only pairs are used in the subsequent assembly, not the single reads.

Reference genome filtering

Some of the reference genomes were not similar enough to the genomes in the mock community for a fair comparison. We therefore selected only those references that had at least 90% of the genome covered by pairs stemming from the community with even abundances per genome. The quality trimmed pairs that did not align properly against this subset of 52 references were discarded. The references and their GID can be found in Supplementary Table S1. After filtering the pairs there were 3,8Gb and 3,1Gb left for the even and uneven community respectively.

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 recipies 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 for metagenomics is Ray [16]. 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.

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 i.e. $(k+1)$mers. 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 [29]. 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 connectivity within the graph and hence requires lower sequencing coverage of the genomes, but at the same time the risk increases that a kmer occurs multiple times within a genome, or in multiple genomes (hence ambiguous paths will exist). A larger k can overcome this problem if it is larger than the multiply occurring region. But a larger k also requires higher sequence coverage.

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. The contigs of each genome should have a distinct coverage peak due to the genome copy number of the corresponding genome 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.

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 [30], or with Minimus2, as done by for instance the Rnnotator pipeline [31]. Both Newbler and Minimus2 use an Overlap-Layout-Consensus method to merge contigs [15, 29].

For the scaffolding procedure Bambus2 was chosen since it was one of the better scaffolders for single genomes in the GAGE assessment paper [32] and is suitable for metagenomes as well [17]. For a flow diagram of previously mentioned approaches see Figure **??**. A total of twenty-one assembly recipies from the flow diagram have been tested. See Table 1 for an overview of the assembly recipies, Table **??** for versions of each program and Table **??** for the parameters of each recipe.

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 accuracy
- phylogenetic classification accuracy

This study focusses on the first three points, since those are expected to improve the functional annotation and the phylogenetic classification.

Aligning the assembly against 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 [33]. 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 (NUCleotide MUMmer) 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. We define the purity of an alignment by multiply the query coverage with the identity of the alignment. The *purity* of a contig is defined as its purest alignment. An impure contig can be the result of a rearrangement, an indel, copy number variation, inclusion of a kmer stemming from another genome or inclusion of a kmer that is a sequencing error.

Results

In Table ?? the length statistics of the various assemblies are shown. We chose to show only assemblies with a kmer of 31 to keep the information consise. The merged recipies are based on combining kmers from 19 up to 75 with a stepsize of 2.

Purity

Figure ?? shows the number of bases in contigs over different purity intervals and contig length intervals for the even community. In terms of delivering the least amount of impure contigs, velvetnoscaf does best. It however does not deliver very long contigs, raynoscaf does better at a cost of outputting more impure contigs. The metavelvetnoscaf recipe provides even more long contigs but also an even larger amount of impure contigs compared to the other two noscaf recipies. It becomes clear that one has to make a choice between length and purity when assembling by following one of these recipies. All the scaf recipies result in a large increase in the number of impure contigs. For the merging recipies with minimus2 and newbler there is very little difference between the two. In both cases there is an increase in contig lengths with a decrease in purity, but not as much as for the scaf recipies.

Metagenome coverage

The metagenome coverage of the different recipies for the even community can be seen in Figure ??. The light lines are computed using only completely pure contigs, the dark lines using the purest alignment of every contig. This gives an idea of the range of the metagenome coverage when using different cut off values for purity. The merge recipies do the best job of increasing contig lengths and coverage of the metagenome. Again there are only minor differences between minimus2 and newbler. Newbler results in slightly purer contigs. If we would only look at the light lines i.e. counting only completely pure contigs then it would seem the merging recipe is rather bad. Therefore in Figure ?? we plotted several different purity cutoffs for minimusvelvetnoscaf. The plot proves that most of the metagenome coverage is coming from only slightly impure contigs.

Kmer LCA analysis

The impurity of contig could come from rearrangements, including chimeric kmers and/or unknown kmers. An unknown kmer might come from an error in the sequencing or because the input DNA was slightly different from the reference. We refer to these kmers henceforth as erroneous kmers. In Figure ?? one can see that most of the chimeric kmers come from genomes whose LCA is either at the species or genus level. The sum of the chimeric kmers is larger than the number of kmers not stemming from any of the reference genomes. For velvetnoscaf31 contigs with an erroneous and chimeric kmer are occuring in an approximately equal ratio. There are however more than double as many chimeric kmers indicating that a chimeric contig often has more chimeric kmers than an erroneous contig has erroneous kmers.

Extracting pure contigs

There are a plethora of ways one can postprocess a metagenomic assembly. Now that we have demonstrated there is quite some impurity in metagenomic assemblies, especially for assemblers outputting longer contigs, it would be ideal to get a confidence score per contig that reflects its purity without a reference genome. Depending on the postprocessing desired a confidence threshold can be chosen to only include certain contigs. We ran FRCbam and REAPR on the raynoscaf31 assembly. Unfortunately for both reference less validation tools we could not find a set of error indicators that would be an indication of impurity i.e. chimericity, indels, erroneousness or rearrangements. A very simple rule of thumb is to simply use contigs coverage as an indication of contig purity. In Figure ?? one can see that the pure bases are mostly in contigs with a high coverage mean. Figure ?? shows the relation between coverage mean and purity.

Acknowledgments

References

References

1. Eisen J (2007) Environmental shotgun sequencing: its potential and challenges for studying the hidden world of microbes. PLoS Biol 5: e82.

2. Andersson A, Banfield J (2008) Virus population dynamics and acquired virus resistance in natural microbial communities. Science 320: 1047-1050.

3. Qin J, Li R, Raes J, Arumugam M, Burgdorf K, et al. (2010) A human gut microbial gene catalogue established by metagenomic sequencing. Nature 464: 59-65.

4. Hess M, Sczyrba A, Egan R, Kim T, Chokhawala H, et al. (2011) Metagenomic discovery of biomass-degrading genes and genomes from cow rumen. Science 331: 463-467.

5. Morgan J, Darling A, Eisen J (2010) Metagenomic sequencing of an in vitro-simulated microbial community. PLoS One 5: e10209.

6. Tringe S, von Mering C, Kobayashi A, Salamov A, Chen K, et al. (2005) Comparative metagenomics of microbial communities. Science 308: 554-557.

7. Mavromatis K, Ivanova N, Barry K, Shapiro H, Goltsman E, et al. (2007) Use of simulated data sets to evaluate the fidelity of metagenomic processing methods. Nat Methods 4: 495-500.

8. Tyson G, Chapman J, Hugenholtz P, Allen E, Ram R, et al. (2004) Community structure and metabolism through reconstruction of microbial genomes from the environment. Nature 428: 37-43.

9. Hallam S, Konstantinidis K, Putnam N, Schleper C, Watanabe Y, et al. (2006) Genomic analysis of the uncultivated marine crenarchaeote cenarchaeum symbiosum. Proc Natl Acad Sci U S A 103: 18296-18301.

10. Mende D, Waller A, Sunagawa S, Jarvelin A, Chan M, et al. (2012) Assessment of metagenomic assembly using simulated next generation sequencing data. PLoS One 7: e31386.

11. Pignatelli M, Moya A (2011) Evaluating the fidelity of de novo short read metagenomic assembly using simulated data. PLoS One 6: e19984.

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

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

14. Quinn N, Levenkova N, Chow W, Bouffard P, Boroevich K, et al. (2008) Assessing the feasibility of GS FLX pyrosequencing for sequencing the atlantic salmon genome. BMC Genomics 9: 404.

15. Sommer D, Delcher A, Salzberg S, Pop M (2007) Minimus: a fast, lightweight genome assembler. BMC Bioinformatics 8: 64.

16. Boisvert S, Raymond F, Godzaridis E, Laviolette F, Corbeil J (2012) Ray meta: scalable de novo metagenome assembly and profiling. Genome Biol 13: R122.

17. Koren S, Treangen T, Pop M (2011) Bambus 2: scaffolding metagenomes. Bioinformatics 27: 2964-2971.

18. de la Bastide M, McCombie W (2007) Assembling genomic DNA sequences with PHRAP. Curr Protoc Bioinformatics Chapter 11: Unit11.4.

19. Batzoglou S, Jaffe D, Stanley K, Butler J, Gnerre S, et al. (2002) ARACHNE: a whole-genome shotgun assembler. Genome Res 12: 177-189.

20. Aparicio S, Chapman J, Stupka E, Putnam N, Chia J, et al. (2002) Whole-genome shotgun assembly and analysis of the genome of fugu rubripes. Science 297: 1301-1310.

21. Warren R, Sutton G, Jones S, Holt R (2007) Assembling millions of short DNA sequences using SSAKE. Bioinformatics 23: 500-501.

22. Myers E, Sutton G, Delcher A, Dew I, Fasulo D, et al. (2000) A whole-genome assembly of drosophila. Science 287: 2196-2204.

23. Li R, Zhu H, Ruan J, Qian W, Fang X, et al. (2010) De novo assembly of human genomes with massively parallel short read sequencing. Genome Res 20: 265-272.

24. Willner D, Daly J, Whiley D, Grimwood K, Wainwright C, et al. (2012) Comparison of DNA extraction methods for microbial community profiling with an application to pediatric bronchoalveolar lavage samples. PLoS One 7: e34605.

25. Diaz P, Dupuy A, Abusleme L, Reese B, Obergfell C, et al. (2012) Using high throughput sequencing to explore the biodiversity in oral bacterial communities. Mol Oral Microbiol 27: 182-201.

26. Wu G, Lewis J, Hoffmann C, Chen Y, Knight R, et al. (2010) Sampling and pyrosequencing methods for characterizing bacterial communities in the human gut using 16s sequence tags. BMC Microbiol 10: 206.

27. Human Microbiome Project Consortium (2012) A framework for human microbiome research. Nature 486: 215-221.

28. Doroghazi J, Buckley D (2008) Evidence from GC-TRFLP that bacterial communities in soil are lognormally distributed. PLoS One 3: e2910.

29. Miller J, Koren S, Sutton G (2010) Assembly algorithms for next-generation sequencing data. Genomics 95: 315-327.

30. Luo C, Tsementzi D, Kyrpides N, Read T, Konstantinidis K (2012) Direct comparisons of illumina vs. roche 454 sequencing technologies on the same microbial community DNA sample. PLoS One 7: e30087.

31. Martin J, Bruno V, Fang Z, Meng X, Blow M, et al. (2010) Rnnotator: an automated de novo transcriptome assembly pipeline from stranded RNA-seq reads. BMC Genomics 11: 663.

32. Salzberg S, Phillippy A, Zimin A, Puiu D, Magoc T, et al. (2012) GAGE: A critical evaluation of genome assemblies and assembly algorithms. Genome Res 22: 557-567.

33. Kurtz S, Phillippy A, Delcher A, Smoot M, Shumway M, et al. (2004) Versatile and open software for comparing large genomes. Genome Biol 5: R12.

Figure Legends

Tables

Assembly recipe name	Contiging	Merging	Scaffolding
velvetnoscaf	Velvet	-	-
velvetscaf	Velvet	-	Velvet
velvetnoscafminimus2	Velvet	Minimus2	-
velvetnoscafnewbler	Velvet	Newbler	-
velvetnoscafbambus2	Velvet	-	Bambus2
velvetnoscafminimus2bambus2	Velvet	Minimus2	Bambus2
velvetnoscafnewblerbambus2	Velvet	Newbler	Bambus2
metavelvetnoscaf	Meta-Velvet	-	-
metavelvetscaf	Meta-Velvet	-	Meta-Velvet
metavelvetnoscafminimus2	Meta-Velvet	Minimus2	-
metavelvetnoscafnewbler	Meta-Velvet	Newbler	-
metavelvetnoscafbambus2	Meta-Velvet	-	Bambus2
metavelvetnoscafminimus2bambus2	Meta-Velvet	Minimus2	Bambus2
metavelvetnoscafnewblerbambus2	Meta-Velvet	Newbler	Bambus2
raynoscaf	Ray	-	-
rayscaf	Ray	-	Ray
raynoscafminimus2	Ray	Minimus2	-
raynoscafnewbler	Ray	Newbler	-
raynoscafbambus2	Ray	-	Bambus2
raynoscafminimus2bambus2	Ray	Minimus2	Bambus2
raynoscafnewblerbambus2	Ray	Newbler	Bambus2

Table 1. Assembly recipies

Supporting Information Legends

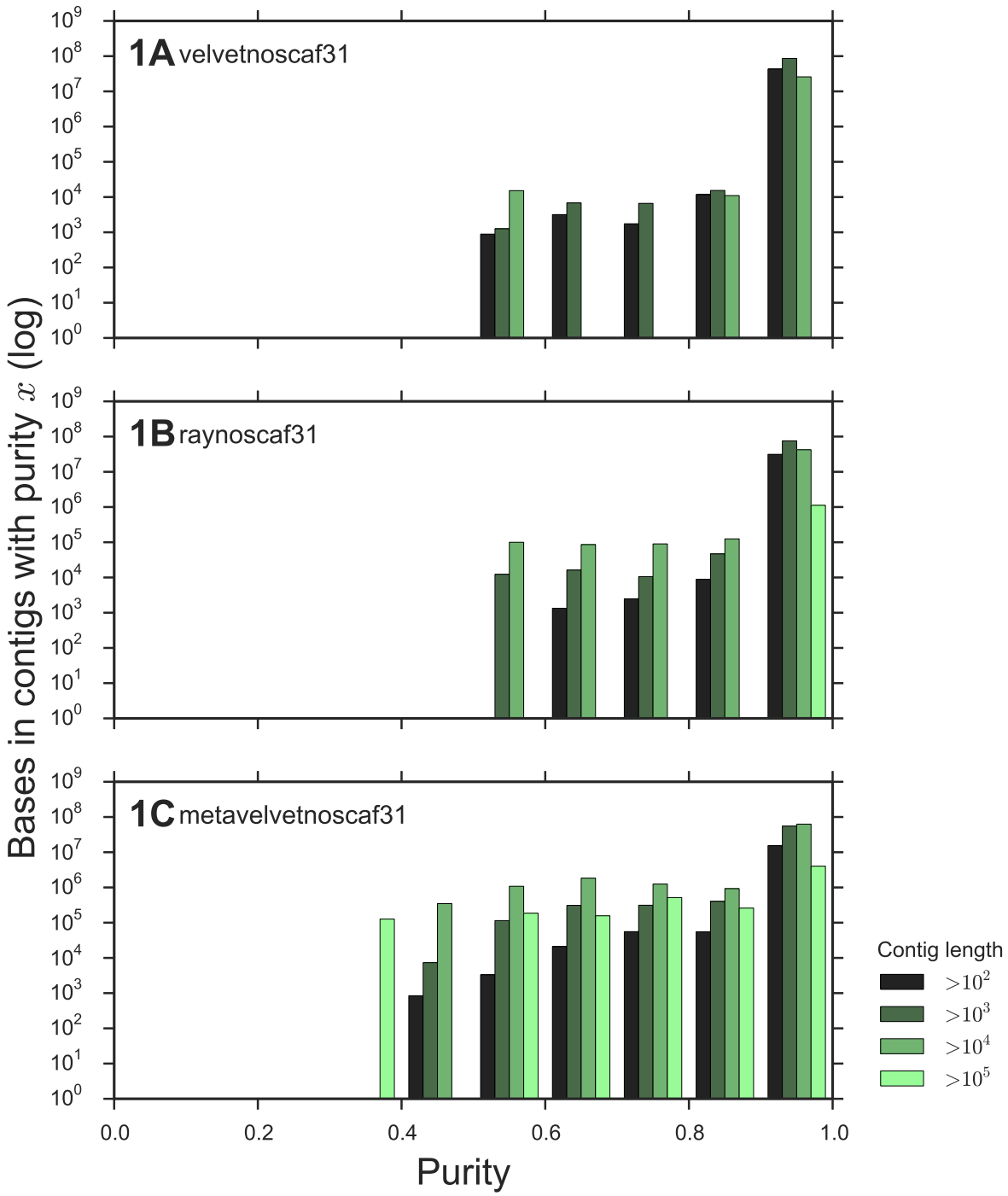

Figure 1. Figure 1. Distribution of bases in contigs over purity and length intervals for the mock community with even abundances per genome. Three different assembly recipes are shown: velvetnoscaf31 (A), raynoscaf31 (B) and metavelvetnoscaf31 (C). The velvet recipe gives the purest contigs, but they are not very long. From the top panel to the bottom panel a trend can be noticed: longer conitgs are produced at a cost of purity.

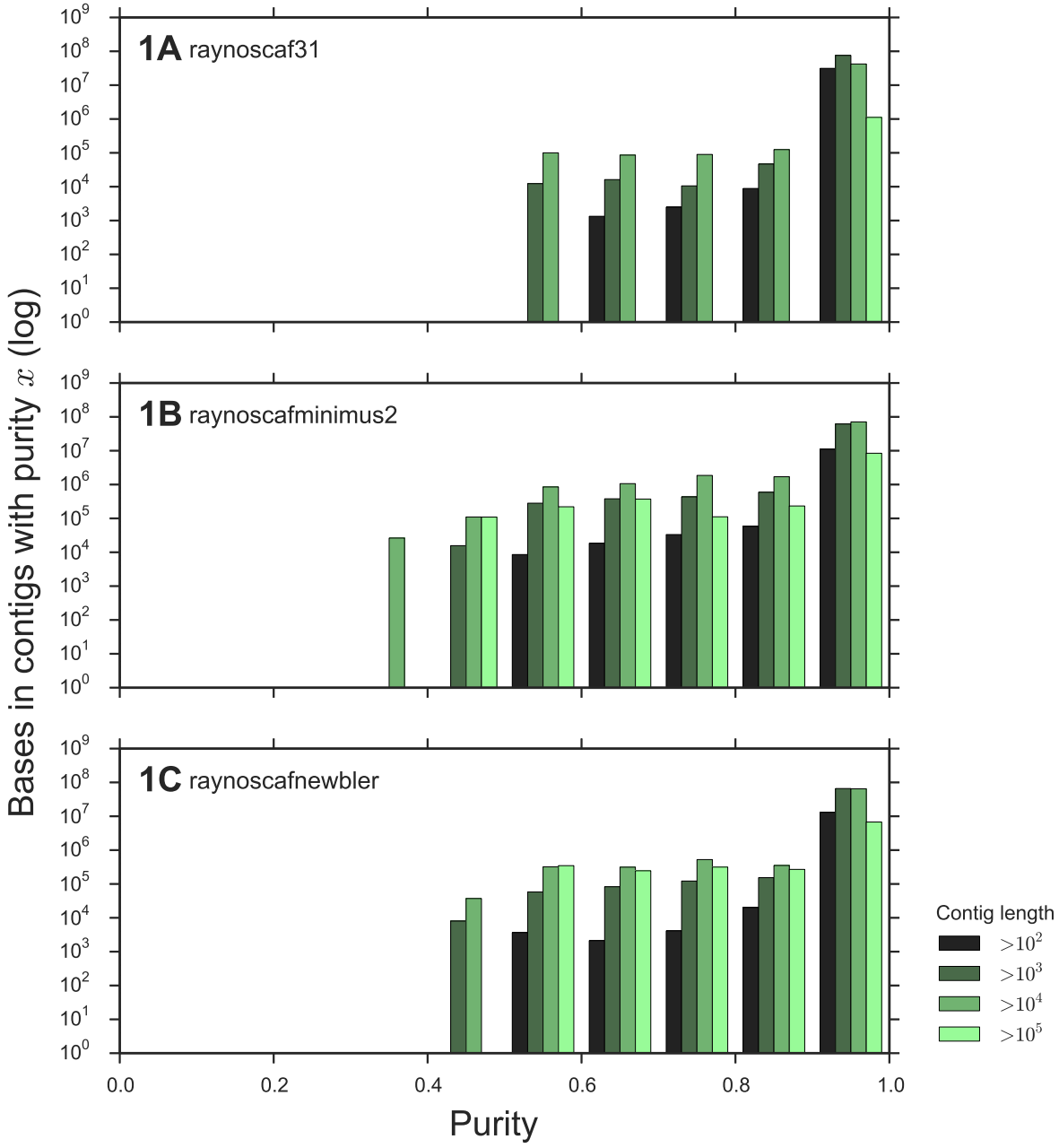

Figure 2. Figure 2. Distribution of bases in contigs over purity and length intervals for the mock community with even abundances per genome. Three different assembly recipes are shown: raynoscaf31 (A), raynoscafminimus2 (B) and raynoscafnewbler (C). Merging Ray assemblies over kmers 19 to 75 with a stepsize of 2 using Minimus2 and Newbler results in longer but impurer contigs. The Newbler recipe is more stringent than Minimus2.

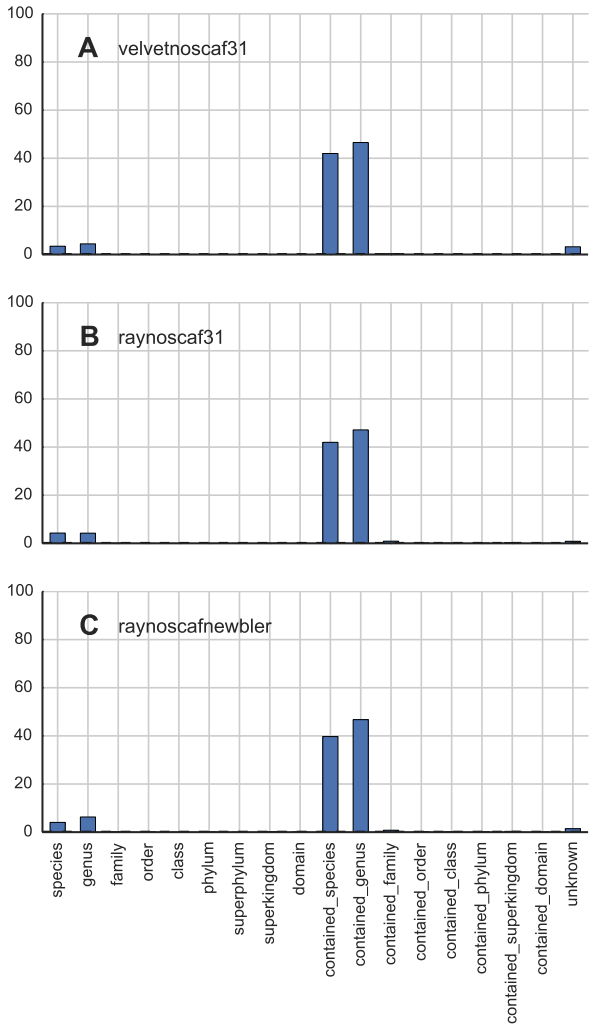

Figure 3. Figure 3. LCA for each kmer that did not belong to the reference genome. Three different assembly recipes are shown: velvetnoscaf31 (A), raynoscaf31 (B) and raynoscafnewbler (C).

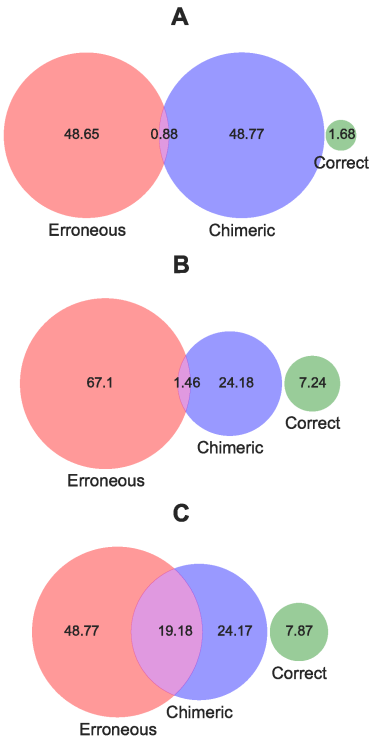

Figure 4. Figure 4. Type of impure contigs based on Kraken analysis Three different assembly recipes are shown: velvetnoscaf31 (A), raynoscaf31 (B) and raynoscafnewbler (C).

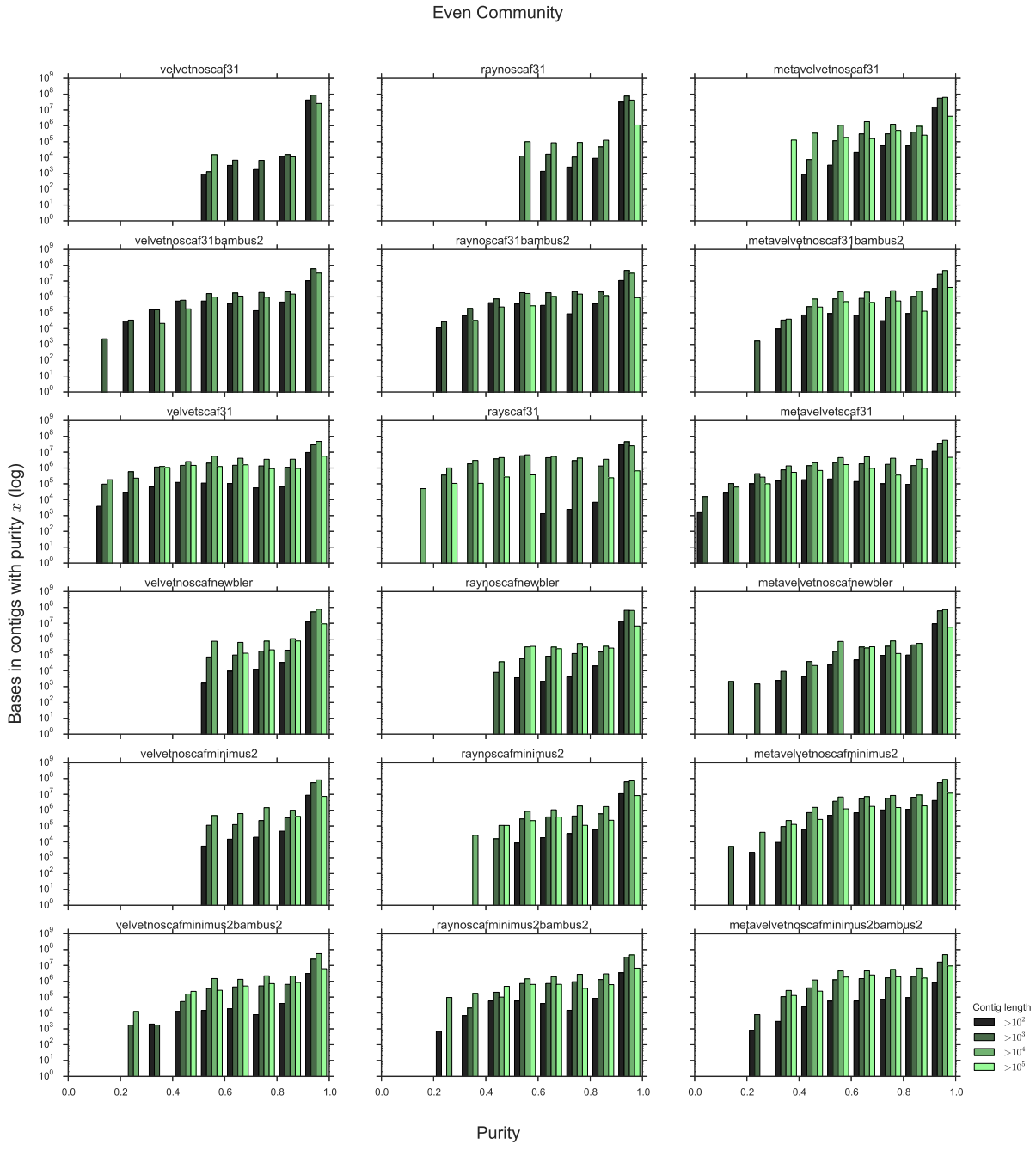

Figure 5. Figure S1. Distribution of bases in contigs over purity and length intervals for the mock community with even abundances per genome.

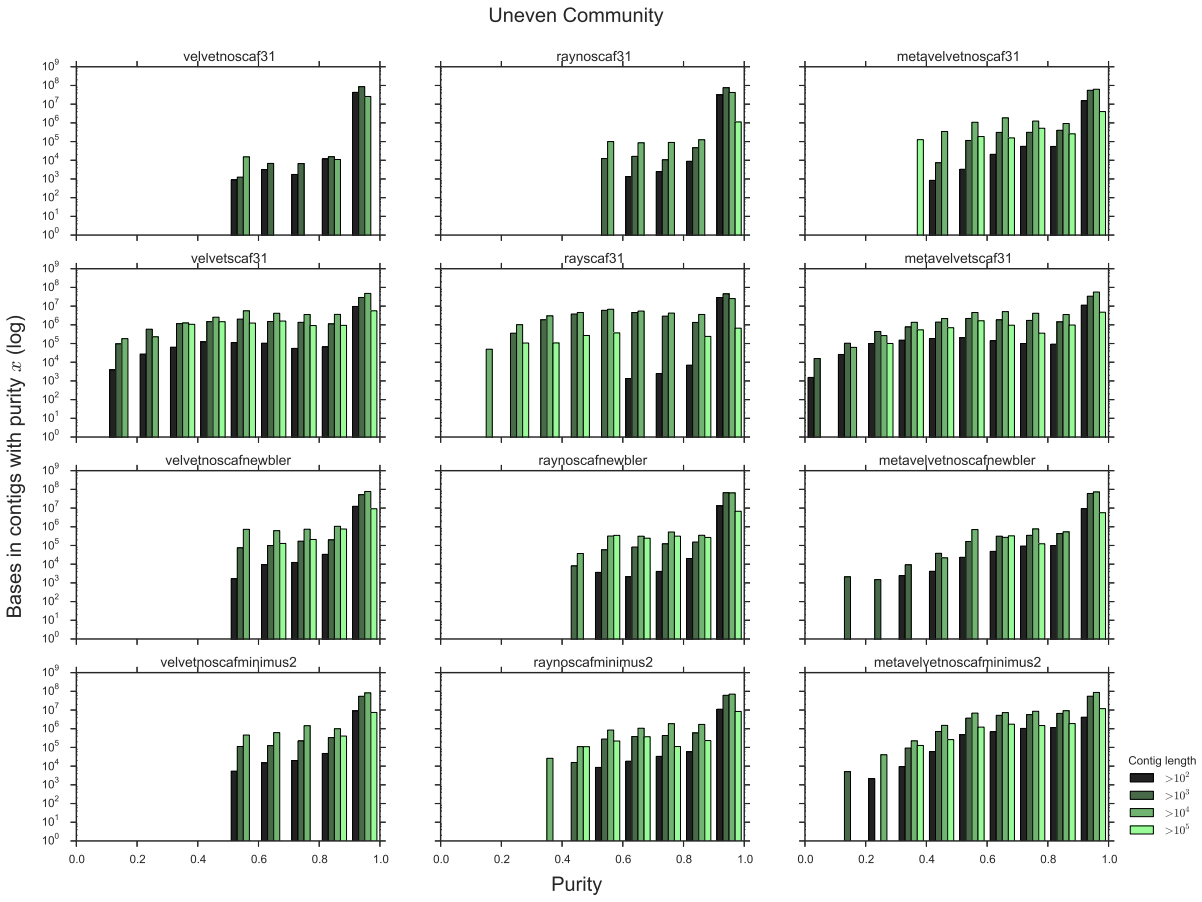

Figure 6. Figure S2. Distribution of bases in contigs over purity and length intervals for the mock community with even abundances per genome.