Computational methods for the recovery of low-ranking taxonomic bins and haplotype-specific gene assemblies from shotgun metagenomes

Inaugural dissertation

for the attainment of the title of doctor
in the Faculty of Mathematics and Natural Sciences
at the Heinrich Heine University Düsseldorf

presented by

Ivan Gregor
from Prague

Düsseldorf, July 2017

from the institute for Computer Science
at the Heinrich Heine University Düsseldorf

Published by permission of the
Faculty of Mathematics and Natural Sciences at
Heinrich Heine University Düsseldorf

Supervisor:	Prof. Dr. Alice C. McHardy
Co-supervisor:	Prof. Dr. Martin J. Lercher

Date of the oral examination:

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Dissertation eigenständig und ohne fremde Hilfe angefertigt habe. Arbeiten Dritter wurden entsprechend zitiert. Diese Dissertation wurde bisher in dieser oder ähnlicher Form noch bei keiner anderen Institution eingereicht. Ich habe bisher keine erfolglosen Promotionsversuche unternommen.

Düsseldorf, den . . . .

(Ivan Gregor)

Statement of authorship

I hereby certify that this dissertation is the result of my own work. No other person's work has been used without due acknowledgement. This dissertation has not been submitted in the same or similar form to other institutions. I have not previously failed a doctoral examination procedure.

Summary

Metagenomics is the functional or sequence-based analysis of microbial DNA isolated directly from a microbial community of interest. As the cultivation conditions for most (~99%) microorganisms are unknown or too complex to reproduce in the laboratory, random shotgun and amplicon-sequencing based metagenome studies have led to substantial advances in our understanding of the structure and functions of microbial communities within the last decade. The key question of metagenome researchers is: "Who is there, what are they doing and who is doing what?" For instance, the human gut microbiome is a vast collection of symbiotic microorganisms. The gut microbiome performs many important biochemical functions for the host, where disorders of the microbiome are associated with many diverse diseases, e.g. the inflammatory bowel disease. Bioinformatics analyses are now able to describe the gut microbiome at a detailed genetic and functional level. The understanding of microbiome activity is essential to the development of personalized strategies in healthcare and to reveal new targets for drug development. Therefore, understanding microbial communities will improve our well-being and human health. Moreover, advances in sequencing technologies have been enormous in the last decade, while the throughput increased drastically, sequencing costs dropped. This enabled researchers to use next generation sequencing data as a common approach to study microorganisms originating from various environments, e.g. the human gut. Metagenome assembly and its subsequent taxonomic binning are two essential challenging tasks that are typically performed as a part of a metagenome sample analysis. We have developed *Snowball*, which is a strain aware gene assembler for metagenomes. To the best of our knowledge, this is the first gene assembler for metagenomic data that can distinguish gene variants of individual strains without using closely related reference genomes of the studied species. This is a very important property as metagenomes originating from novel environments oftentimes contain new unknown species for which there are no closely related reference genomes available. Moreover, for many purposes, including functional analysis of metagenomic data, it is sufficient to assemble only the coding sequences of the strains, as usually more than 85% of prokaryotic genomes are coding sequences. We have employed *Snowball* to assemble simulated reads generated from the recently published *Rhizobia* strains, which demonstrates the capability of our method to assemble gene sequences of closely related novel strains. We have also developed *PhyloPythiaS+* that is an automated composition based taxonomic binning method. This method is a successor to the *PhyloPythiaS* software. We have fully automated this

method by adding a new marker-gene based framework that automatically determines the most relevant taxa to be modeled and suitable training sequences directly from the input metagenome sample. To the best of our knowledge, this is the first method that combines taxonomic profiling and subsequent taxonomic composition based binning of the whole input metagenome sample. Moreover, we have developed a new *k*-mer counting algorithm that accelerated the whole method and showed state-of-the-art performance for the simultaneous enumeration of *4*–*6*-mers, which is commonly used for composition based binning. We have also extensively evaluated the whole automated taxonomic binning pipeline by comparing it to the other methods and devised several new evaluation measures. The results showed that our method performed especially well for samples originating from novel environments in comparison to the other methods. These results were also confirmed in the CAMI challenge, in which *PhyloPythiaS+* demonstrated its high recall and ability to correctly assign taxa that have longer taxonomic distances to the known reference genomes or draft genomes. *PhyloPythiaS+* has also already been employed in several research studies. We believe that our methods will be valuable for researchers studying species evolution, strain or gene diversity, genes under selection, virulent genes, metagenome samples originating from novel environments, for draft genome reconstruction and for the subsequent functional analysis of the studied metagenome microbial communities.

Zusammenfassung

Metagenomik ist die funktionale oder Sequenz-basierte Analyse mikrobieller DNA, die direkt aus Umweltproben von Interesse isoliert wird. Für die meisten Mikroorganismen (~99%) sind die Bedingungen für eine erfolgreiche Anreicherung in Kultur unbekannt oder zu komplex um sie im Labor zu reproduzieren. Daher sind Metagenomstudien auf der Basis von Shotgun- und Amplikon-Sequenzierung für unser Verständnis der Struktur und Funktionen der mikrobiellen Gemeinschaften entscheidend. Die Schlüsselfragen der Metagenom-Forscher lauten dabei: "Wer ist da, was macht die Gemeinschaft und wer macht was?" Das menschliche Darmmikrobiom ist beispielsweise eine riesige Sammlung von symbiotischen Mikroorganismen. Es übernimmt viele wichtige biochemische Funktionen für den Wirt, sodass Störungen des Mikrobioms mit vielen verschiedenen Krankheiten assoziiert werden, z.B. mit Reizdarm.

Bioinformatische Analysen sind heute in der Lage, das Darmmikrobiom auf einer detaillierten genetischen und funktionalen Ebene zu beschreiben. Sie bilden die Grundlage für personalisierte Strategien im Gesundheitswesen und es werden neue Ansatzstellen für die Arzneimittelentwicklung aufdecken. Darüber hinaus waren die Fortschritte in den Sequenzierungstechnologien im letzten Jahrzehnt enorm. Während der Datendurchsatz erheblich anstieg, sanken die Sequenzierungskosten. Dies ermöglicht es Forschern, die Next-Generation Sequenzdaten als Standardverfahren einzusetzen; also auch um Mikroorganismen zu untersuchen, die aus verschiedenen Umgebungen wie dem menschlichen Darm stammen. Die Assemblierung des Metagenoms und die anschließende taxonomische Zuordnung rekonstruierter DNA-Sequenzen sind dabei zwei wesentliche und anspruchsvolle Teilaufgaben einer Metagenom-Probenanalyse.

Hierfür haben wir *Snowball* entwickelt. *Snowball* ist nach unserem besten Wissen der erste Gen-Assembler für Metagenom-Daten, der Genvarianten einzelner Stämme unterscheiden kann, ohne eng verwandte Referenz-Genome der untersuchten Spezies zu verwenden. Das ist eine sehr wichtige Eigenschaft, weil Metagenome aus neuartigen Umgebungen oft neue unbekannte Spezies enthalten, für die es keine eng verwandten Referenz-Genome gibt. Gleichzeitig genügt es für viele Zwecke, einschließlich der Funktionsanalyse von Metagenom-Daten, nur die kodierenden Sequenzabschnitte der Genome zu assemblieren, weil in der Regel mehr als 85% der prokaryotischen Genome für Proteine kodieren. *Snowball* konzentriert sich auf diese Abschnitte.

Wir haben *Snowball* eingesetzt, um simulierte DNA-Abschnitte zu assemblieren, die aus kürzlich veröffentlichten *Rhizobia*-Stämmen generiert wurden. Dies zeigte die Fähigkeit

unserer Methode, die Gensequenzen von eng verwandten neuartigen Stämmen bei der Assemblierung aufzulösen. Um die rekonstruierten Metagenom-Sequenzen auch taxonomisch zuordnen zu können, haben wir zudem *PhyloPythiaS+* entwickelt. Diese Methode ist ein Nachfolger der Kompositions-basierten *PhyloPythiaS* Software. Wir haben diese Methode komplett automatisiert, indem wir ein neues Marker-Gen-basiertes Framework hinzugefügt haben, das automatisch die relevantesten Taxa modelliert und entsprechende Trainingssequenzen direkt aus der Metagenom-Probe bestimmt. Nach unserem besten Wissen ist dies die erste Methode, die das taxonomische Profiling und die anschließende taxonomische Kompositions-basierte Zuordnung der gesamten Metagenom-Probe kombiniert. Darüber hinaus haben wir einen neuen Zählalgorithmus für Nukleotidsequenzen der Länge *k* entwickelt, der die gesamte Methode beschleunigt. Der Zählalgorithmus zeigt eine State-of-the-Art Leistung für die gleichzeitige Aufzählung von Nukleotidsequenzen der Länge *4–6*, die üblicherweise für die taxonomische Kompositions-basierte Zuordnung der Metagenom-Proben verwendet werden. Wir haben die gesamte automatisierte Pipeline umfassend mit den Wettbewerbern verglichen und dafür mehrere neue Evaluierungskriterien entwickelt. Die Ergebnisse zeigen, dass unsere Methode im Vergleich zu den anderen Methoden besonders gut für die Metagenom-Proben aus neuartigen Umgebungen geeignet ist. Die hohe Sensitivität von *PhyloPythiaS+* und seine Fähigkeit zur korrekten Zuordnung von Taxa, die größere taxonomische Abstände zu den bekannten Referenz-Genomen haben, wurde auch in der CAMI Challenge bestätigt. *PhyloPythiaS+* wurde darüber hinaus bereits in mehreren Forschungsprojekten eingesetzt.

Wir glauben, dass unsere Methoden für Forscher in folgenden Bereiche wertvoll sind: Evolution von Arten, Diversität von Bakterienstämmen, Genvielfalt, Gene, die unter Selektion stehen, virulente Gene, Metagenom-Proben aus neuartigen Umgebungen, entwurfsweise Genom-Rekonstruktion und die anschließende Funktionsanalyse der untersuchten mikrobiellen Gemeinschaften.

Acknowledgements

This thesis would not be possible without great support from numerous people. Unfortunately, I cannot name all of you here. Therefore, thank you all very much!

First of all, I would like to thank my supervisor Prof. Dr. Alice C. McHardy for her continuous support, expertise and inspiration. I would like to also thank Dr. Alexander Schönhuth for the co-supervision of my gene assembly project, inspiring discussions and for hosting me at the CWI, Amsterdam. Second, I would like to thank all the members of the IRG1 group at the MPI, Saarbrücken, the AlgBio group at the HHU, Düsseldorf and the BIFO group at the HZI, Braunschweig. I would like to thank all of you for creating an inspiring environment, interesting discussions, your great ideas, proofreading my long manuscripts, giving me feedback and for being supportive. Here, I would like to name: Johannes Dröge, Lars Steinbrück, Sebastian G. A. Konietzny, Aaron Weimann, David Lähnemann, Rubén Garrido-Oter, Yao Pan, Andreas Bremges and Peter Hofmann. I would like to thank Gary Robertson (HZI), Klaus-Dieter Baer (HHU) and Joachim Büch (MPI) for their excellent and indispensable technical support. I would like to also thank Angela Rennwanz for being very helpful concerning administrative issues. Lastly, I would like to thank my family and friends for being very supportive, patient and encouraging all the time.

Table of Contents

1 Introduction

In this chapter, we will first introduce the field of metagenomics and sequencing. Sequencing platforms enable us to read DNA sequences consisting of letters A, C, G, and T, given a real metagenome sample containing a mixture of microorganisms. Then, we will introduce two core bioinformatics concepts: sequence assembly and taxonomic binning. Assembly methods are employed to assemble longer continuous sequences (contigs) from short DNA sequences (reads) that are output by the sequencing platforms. Taxonomic binning methods are then used to assign taxonomic identifiers to the assembled or unassembled DNA sequences. This can be used for the characterization of the composition and functional potential of a particular metagenome sample. We will also describe basic concepts of the methods for sequence analysis that were used as sub-routines in this work. We will conclude with the outline of this dissertation. Note that some formulations in this chapter originate from my publications (Gregor, Dröge, *et al.*, 2016; Gregor, Schönhuth, *et al.*, 2016).

1.1 Metagenomics

Metagenomics (Handelsman *et al.*, 1998) is the functional or sequence-based analysis of microbial DNA isolated directly from a microbial community of interest (Kunin *et al.*, 2008; Riesenfeld *et al.*, 2004). As the cultivation conditions for most (~99%) microorganisms are either unknown or too complex to reproduce in the laboratory (Hugenholtz, 2002), random shotgun and amplicon-sequencing based metagenome studies have led to substantial advances in our understanding of the structure and functions of microbial communities within the last decade (Pope, Smith, *et al.*, 2011; Kalyuzhnaya *et al.*, 2008; Turnbaugh *et al.*, 2010; Hess *et al.*, 2011; Schloissnig *et al.*, 2013; Blaser *et al.*, 2013; Zarowiecki, 2012).

Metagenomes that have been studied originate from various environments, e.g.:

- Lake Washington (Kalyuzhnaya *et al.*, 2008).
- Wastewater (Martín *et al.*, 2006).
- Acid mine drainage (Tyson *et al.*, 2004).
- Hot spring (Ward *et al.*, 1998).
- Agricultural soil (Tringe *et al.*, 2005).
- Leafs and roots of *Arabidopsis* (Bai *et al.*, 2015).
- Termite hindgut and gut (Warnecke *et al.*, 2007; Ikeda-Ohtsubo *et al.*, 2016).
- Tammar wallaby gut (Pope, Smith, *et al.*, 2011).

- Svalbard Reindeer rumen (Pope, Mackenzie, *et al.*, 2011).
- Human gut (Turnbaugh *et al.*, 2010; Giloteaux *et al.*, 2016).
- Human blood (Gyarmati *et al.*, 2016).
- Subway (MetaSUB International Consortium, 2016).

The central question of many metagenome researchers (Marx, 2016) is: "Who is there, what are they doing and who is doing what?"

Understanding microbial communities will improve our well-being and human health. It has also a potential to revolutionize chemical industry or even facilitate long human space missions. For instance, as described in (Kuczynski *et al.*, 2011), the human gut microbiome is a vast collection of symbiotic microorganisms. The gut microbiome performs numerous important biochemical functions for the host, where disorders of the microbiome are associated with many diverse diseases, e.g. Crohn's disease, ulcerative colitis or inflammatory bowel disease (W. Wang *et al.*, 2015). Bioinformatics analyses based on next generation technologies are now able to describe the gut microbiome at a detailed genetic and functional level. The understanding of microbiome activity is essential to the development of personalized strategies in healthcare and to reveal new targets for drug development. Many studies have found out that the variability in the microbiota both within a human subject and between different subjects is immense. Moreover, only a small fraction of the total taxa found within a single body site appears to be present across all time points. Therefore, several projects have been established to investigate this variety, e.g. the Human Microbiome Project (Aagaard *et al.*, 2013) or MetaHIT (Arumugam *et al.*, 2011). It is noteworthy that it is estimated that a human body contains ~10x more microorganisms than human cells, making up about 1-3% of the body's mass, while it is estimated that it encodes ~100x more unique genes than the human genome (Qin *et al.*, 2010). Although, other studies estimate that the number of microorganisms in the human body is of the same order as the number of the human cells (Sender *et al.*, 2016).

Another interesting example is the study of the human microbiome during long-duration space missions (Voorhies and Lorenzi, 2016). It is known that a balanced microbiome is essential for human health. However, a long stay on a spaceship reduces the microbiome diversity, as the air on a spaceship is heavily filtered and the astronaut's food contains a minimum amount of microbes. Moreover, the galactic cosmic radiation may have a negative impact on the crew microbiome, since most microbes are not

resistant to radiation. Another open question is how to enable the recovery of the astronaut's microbiome after an antibiotic treatment. The use of probiotics, fresh fruits and vegetables may help to solve those problems, although it brings a large set of new challenges that are yet to be solved. Finding strategies how to keep the crew microbiome healthy and how to rebalance a damaged microbiome is essential for survival and will also improve the human health on the Earth.

Although, many bioinformatics tools have been developed for the metagenome data analysis, there are still many challenges to be addressed. As described in (Sczyrba *et al.*, 2017), one of the main challenges is to enable detailed strain-level analysis of the metagenome samples, which would improve our understanding of the microbial communities. It would enable us to study microevolution and how organisms react to the changes in the environment. In the next sections, we will describe all the main challenges of metagenome assembly and taxonomic binning. Given the increasing amount of available metagenomic data and the need for fine-grained strain-level analysis, new bioinformatics methods and approaches are needed for the data analysis and interpretation.

1.2 Sequencing

After a metagenome sample, containing a mixture of living microorganisms, is taken from an environment of interest, the DNA from the sample is isolated for the sequencing. The result of the sequencing step is a large dataset containing a mixture of short DNA sequences, called reads, where it is not known what read comes from what member species' genome of a metagenome. The sequencing step requires substantial expertise and is typically performed in a specialized sequencing lab. Depending on the sequencing lab, employed methodology and sequencing platform, the quality of the resulting data varies. Errors and biases introduced by a particular sequencing platform in the sequencing step need to be considered and corrected in the subsequent steps (Laehnemann *et al.*, 2016). Typical sequencing errors are substitutions, insertions and deletions. Many sequencing technologies have been developed, e.g.:

- Illumina
- Sanger
- 454 pyrosequencing (Roche)
- SOLiD (Thermo Fischer)

- Ion Torrent (Thermo Fischer)
- GeneReader (Qiagen)
- Complete Genomics (Beijing Genomics Institute)
- Pacific Biosciences
- Oxford Nanopore Technologies

As described in (Goodwin *et al.*, 2016), sequencing platforms differ in throughput, cost, read length, error profile and read structure. Although, there are many sequencing platforms available, the Illumina sequencing platform is used in most of the metagenome studies. Illumina itself offers many types of sequencing machines. It produces paired-end or single-end reads of length 25–300 bp, where paired-end reads of length ≥100 bp are currently most popular. The output reads have typically very low substitution error of 0.1–1% and the cost varies between $7 and $1,000 per Gb. For instance, the Illumina ultra-high-throughput HiSeq X platform is capable of sequencing ~1,800 human genomes to 30× coverage per year, where the whole genome sequencing of the human genome currently cost less than $1,000.

Sequencing has became more affordable, it has been revolutionized and democratized in the last ten years due to the decreasing sequencing costs and increasing throughput of the sequencing platforms (Metzker, 2010; Goodwin *et al.*, 2016). Moreover, the revolution is likely to continue and the sequencing will become even more frequent in research studies and in clinical settings as a clinical tool in hospitals. As a consequence, bioinformatics analysis tools have to co-evolve, i.e. new scalable bioinformatics tools need to be developed to analyze the ever-growing amounts of sequencing data of all kinds.

1.3 Assembly

Oftentimes, sequenced reads do not carry enough information to be directly used in the subsequent analysis, as they are too short and erroneous. Therefore, short reads are typically assembled into longer continuous error-corrected sequences called contigs.

A contig is a set of reads that are related to one another by overlap of their sequences (Staden, 1980). The assembly problem defined according to the maximum parsimony approach is to find the shortest common superstring of the individual reads to which all the reads map with a sufficiently low error (Peltola *et al.*, 1984). However, the

disadvantage of this definition is over-compression, i.e. many variants of a sequence are compressed into one consensus sequence. Therefore, the assembly problem was re-defined according to the maximum likelihood approach (Myers, 1995). I.e., given the reads and a maximum error rate *e* with which a read can be mapped to a reconstructed consensus sequence. Find a reconstruction, such that the reads are mapped to a consensus sequence with at most *e* error rate and the starting points of the reads have the same distribution as the true underlying distribution.

There are several common problems that need to be typically addressed by the assembly algorithms:

- It is very difficult to distinguish between genuine strain variation and sequencing errors for similar but distinct strains within a metagenome sample (Laehnemann *et al.*, 2016). For instance, to distinguish single nucleotide polymorphisms (SNPs) from substitution errors introduced by a sequencing platform is a very difficult task. It has been shown in the CAMI challenge (Sczyrba *et al.*, 2017) that the current metagenome assemblers are not capable of reconstructing strain-level variants. Moreover, low-abundant strains are oftentimes considered to be sequencing errors and thus removed from the resulting assembly (Nagarajan and Pop, 2013; Zerbino and Birney, 2008).
- Repetitive sequences within a genome that are longer than a read length are usually assembled only into one consensus sequence or may lead to misassemblies. Although, resolving repetitive sequences has theoretical limits for short reads (Kingsford *et al.*, 2010), the use of the paired-end reads, the use of several libraries with different insert sizes and the use of a combination of two sequencing platforms were shown to overcome these limits in some cases (Treangen and Salzberg, 2011).
- In the case that not enough data was sequenced, the resulting assembly not only contains gaps, but it can also result in incorrect assembly. For instance, it can contain chimeras, i.e. artificial consensus sequences consisting of reads originating from different genomes.
- Uneven coverage and potentially also GC bias are further challenges for the assembly methods.
- Assembly is a very computationally demanding task; therefore heuristic approaches are needed to find an approximate solution that is good enough. Nevertheless, efficient use of the main memory and parallelization are required due to the increasing amounts of the sequencing data (D. Li *et al.*, 2015).

Even though many assembly methods have been developed so far, the current assembly tools still have issues to tackle all of the above-mentioned problems sufficiently well. Therefore, new assembly approaches still need to be developed.

1.4 Taxonomic binning

To answer the question: "Who is there and what are they doing?" – many taxonomic assignment methods have been developed to enable characterizing environmental microbial communities. The input of a taxonomic assignment method is either a set of raw reads output by a sequencing platform or longer assembled contigs of a metagenome sample. Given such input dataset, the goal of the taxonomic assignment methods is to assign a taxonomic identifier to each input sequence. A taxonomic identifier corresponds to a node in a tree-like hierarchical structure, called taxonomy. Here, we will refer to the NCBI taxonomy (Federhen, 2011) that is the most used taxonomy in the field of metagenomics, to our knowledge. Note that there are also other taxonomy databases, e.g. RDP (J. R. Cole *et al.*, 2007), Greengenes (DeSantis *et al.*, 2006) or Silva (Quast *et al.*, 2013). In the NCBI taxonomy (Fig 1.1), a sequence is assigned a taxonomic identifier at a particular taxonomic rank, where the main NCBI taxonomy ranks, as seen from the taxonomy "root", are: superkingdom, phylum, class, order, family, genus and species. Here, an assignment of a sequence to the root of the taxonomy is equivalent to a sequence not being assigned. Ideally, a sequence would be assigned to a taxonomic identifier as low in the taxonomy as possible (i.e. to genus or species), but not lower. For instance, if a sequence originates from a species that is a known species, the sequence should be assigned at the species rank. However, if a sequence originates from a species that is a novel species, but originates from a known genus, it should be assigned at the genus rank. Here, a species is considered to be a known species, if there is sufficient reference data available for this species that enables confident assignment of sequences to this species. Analogously, a genus is considered to be a known genus, if there is sufficient reference data available that enable confident assignment of the sequences to this genus. Therefore, if there is no reference data for a novel species in the reference sequence database available, the corresponding sequence should be assigned at a higher taxonomic rank, at which sufficient reference data is available, i.e. genus or higher. This means that a sequence cannot be assigned to taxa, for which there is no reference data available. Note that here the "sufficient reference data available" oftentimes depends on the taxonomic assignment method that is being used. For

instance, *PhyloPythiaS* needs only 100 kb of reference data for a species (Patil, Haider, *et al.*, 2011).

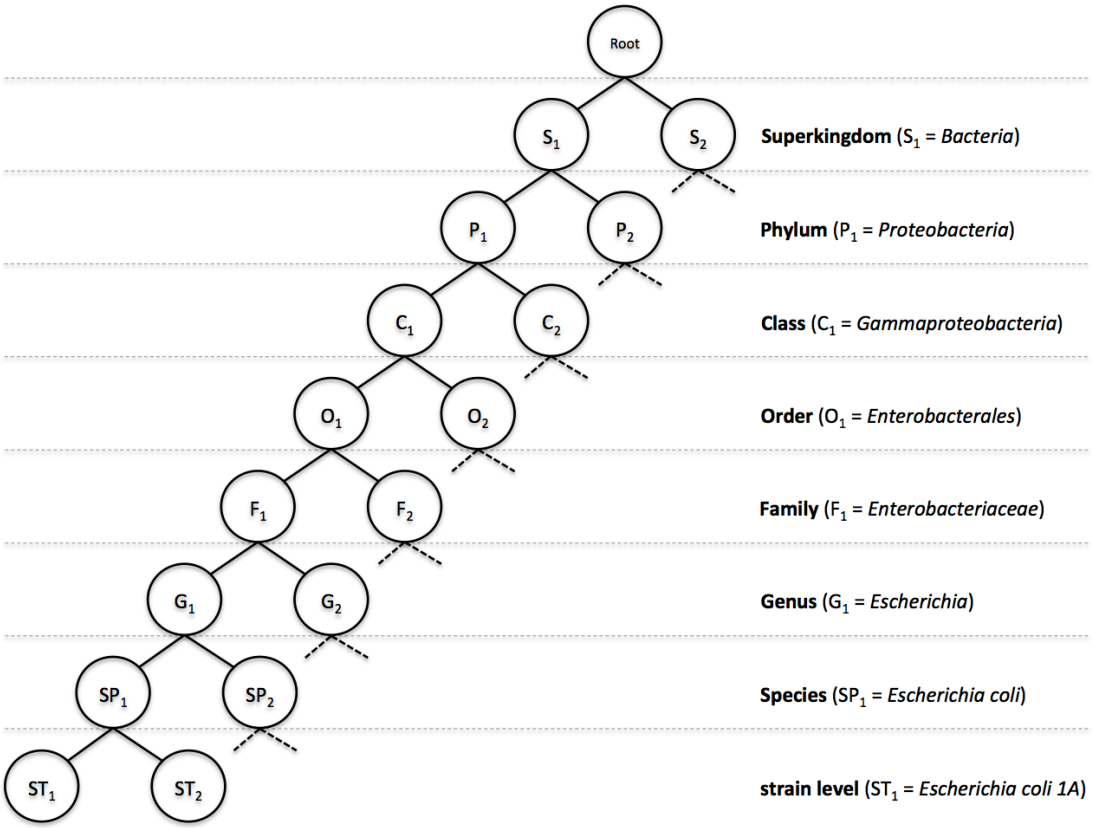


Figure 1.1. An example of the taxonomy representing the lineage of strain: *E. coli 1A*.

The result of a taxonomic assignment method can be interpreted as a taxonomic binning, where the resulting 'bins' of sequence fragments that were assigned the same taxonomic identifier represent draft genomes or pan-genomes of the different microbial community members. The subsequent analysis of these bins then allows characterizing the functional and metabolic potential for individual taxa.

Challenges for the taxonomic assignment methods:

- Correctly identify taxa that are part of the sampled microbial community.
- Correctly assign sequences originating from novel environments, i.e. metagenome samples containing novel taxa (e.g. species), for which there are no reference sequences (i.e. genomes or draft genomes) available.
- Correctly assign the taxonomic rank at which a sequence is assigned a taxonomic identifier, i.e. correctly assign a sequence to low-ranking taxa, but not lower.

- With increasing amounts of the sequencing data, high throughput taxonomic binning methods are required.
- Remove noise (e.g. sequencing errors) and contamination (e.g. human DNA).

As there is a plethora of taxonomic binning methods currently available, the CAMI challenge[1] was established to help researchers to decide what tool is best suited for a particular application (Sczyrba *et al.*, 2017). Individual tools have been evaluated using several datasets and ranked according to several metrics. Based on the results, a researcher can thus decide which tool s/he would use for a particular application. Even though many taxonomic binning tools have been developed, it is still considered to be a challenging task, in particular, to correctly assign metagenome sequences originating from novel environments and to correctly assign metagenome sequences to low-ranking taxa. Therefore, new taxonomic binning methods are still needed.

1.5 Methods for sequence analysis

In this section, we will introduce basic concepts of the methods for sequence analysis that we have used in our work as sub-routines. First, we will describe overlap and *de Bruijn* graphs that are used for sequence assembly. Then, we will describe how hidden Markov models can be used to find homologous sequences. We will conclude this section with two methods that can be used for the metagenome sequence classification: Bayesian classifier and support vector machines.

1.5.1 Overlap and *de Bruijn* graphs

The current assemblers are based on the overlap or *de Bruijn* graphs. The former class of assemblers is based on the overlap-layout-consensus approach, where each read represents a node and there is an edge connecting a pair of nodes for each pair of nodes that has sufficient overlap. By resolving the graph layout problem for a particular overlap graph, paths through the graph that correspond to the output contigs are found. Assemblers using this approach are, e.g. *TIGR* (Sutton *et al.*, 1995), *Celera* (Myers *et al.*, 2000), *ARACHNE* (Batzoglou *et al.*, 2002), *SGA* (Simpson and Durbin, 2012), *SAT* (Zhang *et al.*, 2014) and *SAVAGE* (Baaijens *et al.*, 2017). As the overlap graphs can become very large and difficult to traverse for the current sequencing projects that make use of short

[1] http://www.cami-challenge.org

Illumina reads, assemblers based on the *de Bruijn* graphs are oftentimes used instead of the overlap graphs. The first assembler that employed the *de Bruijn* graphs was the *EULER* assembler (Pevzner *et al.*, 2001). A nice description of a very popular tool using this approach is, e.g. *Velvet* (Zerbino and Birney, 2008). As described in (Pevzner *et al.*, 2001), this approach may look counterintuitive, as small pieces of a big puzzle are broken into even smaller pieces of fixed size k. To construct a *de Bruijn* graph from a set of reads for a given k, from each read of length n, $n - k + 1$ overlapping sub-sequences of length k, with an overlap length $k - 1$ (between each neighboring sub-sequences) are generated. Unique sub-sequences of length k, generated from the reads, correspond to the nodes of the corresponding *de Bruijn* graph. There is a directed edge connecting each node v_1 to each node v_2 if the last $k - 1$ nucleotides of node v_1 are the same as the first $k - 1$ nucleotides of node v_2. For a large dataset, consisting of short reads and containing many duplicate reads, the advantage of the *de Bruijn* graph is that its size corresponds to the number of unique sub-sequences – words of length k (i.e. k-mers), generated from the input reads. Even though such a graph is usually still bigger than the corresponding overlap graph in the main memory, the *de Bruijn* graph is oftentimes much manageable and faster to traverse.

Methods based on the overlap-layout-consensus approach consist of three main steps. In the first "overlap step", distances among all reads are computed, based on which the overlap graph is built. In the second "layout step", sub-graphs of the overlap graph representing longer continuous sequences (i.e. contigs) are identified. In the third "consensus" step, reads of the respective sub-graphs are put together to form consensus sequences of the contigs. As described in (Sutton *et al.*, 1995), the first "overlap step" can be done, e.g. using the Smith-Waterman algorithm (Smith and Waterman, 1981). This algorithm can be applied to find a pair of segments within a pair of long sequences (i.e. one segment within each of the long sequences), such that there is no other pair of segments within the long sequences with greater sequence similarity. The algorithm thus finds an optimal local alignment of the two long sequences, which can be used to derive a similarity score, i.e. the distance between the two long sequences. As described in (Pevzner *et al.*, 2001), the second "layout step" – the layout problem is equivalent to finding a Hamiltonian path in the overlap graph (Matousek and Nesetril, 2009), i.e. finding a path visiting every node of the graph exactly once. Unfortunately, the Hamiltonian path problem is a NP-complete problem; therefore there is no available polynomial algorithm that would solve it.

In the methods based on the *de Bruijn* graphs, *k*-mers are generated from the input reads, which are subsequently used to build the *de Bruijn* graphs. Note that there is no need to compute read overlaps, which makes the construction of the *de Bruijn* graphs faster in comparison to the construction of the corresponding overlap graphs. After a *de Bruijn* graph is build, the assembly problem reduces to finding an Eulerian path in the graph that is consistent with all the read paths, where a read path for a read is a path in the graph corresponding to a particular read (Pevzner *et al.*, 2001). An Eulerian path is a path visiting every edge of the graph exactly once (Matousek and Nesetril, 2009), where there is a linear algorithm for this problem.

The main challenges for the assemblers using either the overlap or the *de Bruijn* graphs are to eliminate errors and resolve repeats.

Sequencing errors in the overlap graphs are usually eliminated by inspecting the multiple sequence alignments of the overlapping reads, this strategy is used, e.g. in (Batzoglou *et al.*, 2002). In the *de Bruijn* graphs, sequencing errors create artifacts in the graphs that need to be removed. For instance, as described in (Zerbino and Birney, 2008; Zerbino *et al.*, 2009): "tips", "bubbles" and erroneous connections need to be removed. A "tip" is a chain of nodes that is connected to the graph only on one end. "Tips" shorter than $2 * k$ are removed, as it is likely that they represent two nearby sequencing errors. Two similar paths in the graph that both start in one node and both end in another node represent a "bubble". A "bubble" is created as a result of either a sequencing error or genuine polymorphism, e.g. a SNP. Similar paths can be found by a Dijkstra-like breadth-first search algorithm (Dijkstra, 1959; Zerbino and Birney, 2008). After the paths are found, the corresponding sequences are extracted and aligned. In the case that the sequences are similar enough, they are merged into a consensus sequence. As the last error correction step, paths with coverage lower than a certain threshold are removed, as they are likely to represent erroneous connections in the graph.

A repeat is either a sequence that is present multiple times in a genome or a set of very similar sequences that are present in a genome. While a repeat is represented as a set of nodes in the overlap graphs, it is represented by an edge in the *de Bruijn* graphs. Due to the representation, it is easier to resolve repeats using the paired-end information when using *de Bruijn* graphs in general. In the overlap graphs, repeats can be resolved based on the partial read overlaps and paired-end information, although this can be a challenging task (Pevzner *et al.*, 2001; Zerbino and Birney, 2008). Note, that repeats that

are longer than a fragment size are oftentimes more difficult to be resolved than shorter repeats.

Both overlap and *de Bruijn* graphs are employed in the current assemblers. While the overlap graphs tend to have better error correction for datasets containing longer reads, the use of the *de Bruijn* graphs is advantageous for assembling large datasets containing short reads and to resolve repeats.

1.5.2 Hidden Markov models

Hidden Markov models (HMMs) can be used to represent gene domains and to find remote homologs of the gene domains (Finn *et al.*, 2016; Eddy, 2011). We have used HMMs to group reads into gene domains in the *Snowball* gene assembler (Gregor, Schönhuth, *et al.*, 2016). In the *PhyloPythiaS+* taxonomic binning method, we have used HMMs to find marker genes within the input sequences (Gregor, Dröge, *et al.*, 2016). Given a multiple sequence alignment of homologous amino acid sequences of a particular gene domain, an HMM representing the gene domain can be built. Such an HMM can subsequently be used to detect remote homologs, i.e. given a query sequence and the HMM, the probability that the query sequence belongs to the gene domain can be computed.

The first order HMM is defined as a tuple $\langle S, W, t, e \rangle$, where S is a set of states, W is a set of words, t is a transition probability and e is an emission probability (Duda *et al.*, 2000). The probability that the current state is $s_i \in S$ given that the previous state was $s_{i-1} \in S$ is $t(s_i \mid s_{i-1})$. The probability that word $w_j \in W$ is emitted at state $s_i \in S$ is $e(w_j \mid s_i)$. The probability of a sequence of words (e.g. amino acids) $\langle w_1, \dots, w_m \rangle$ and the corresponding sequence of states $\langle s_1, \dots, s_m \rangle$ given an HMM is:

$$P(Start, s_1, w_1, s_2, w_2, \dots, s_m, w_m, Stop) = \prod_{i=1}^{m+1} e(w_i \mid s_i) * t(s_i \mid s_{i-1})$$

The probability that the query sequence $\langle w_1, \dots, w_n \rangle$ belongs to a particular gene domain can be computed using the forward algorithm as:

For each $s \in S$:

$$f_1(s) = e(w_1 \mid s) * t(s \mid Start)$$

For each $i \in \langle 2, \dots, n-1 \rangle$ and for each $s \in S$:

$$f_i(s) = e(w_i \mid s) * \sum_{s' \in S} t(s \mid s') * f_{i-1}(s')$$

For each $s \in S$:

$$f_n(s) = t(Stop \mid s) * \ e(w_n \mid s) * \sum_{s' \in S} t(s \mid s') * \ f_{n-1}(s')$$

The final probability is:

$$P(query\ sequence\ \langle w_1, \dots, w_n \rangle \mid HMM\ of\ a\ particular\ gene\ domain) = \sum_{s \in S} f_n(s)$$

The forward algorithm thus computes the final probability as the sum of probabilities of all possible paths through the HMM state diagram that could generate the query sequence $\langle w_1, \dots, w_n \rangle$.

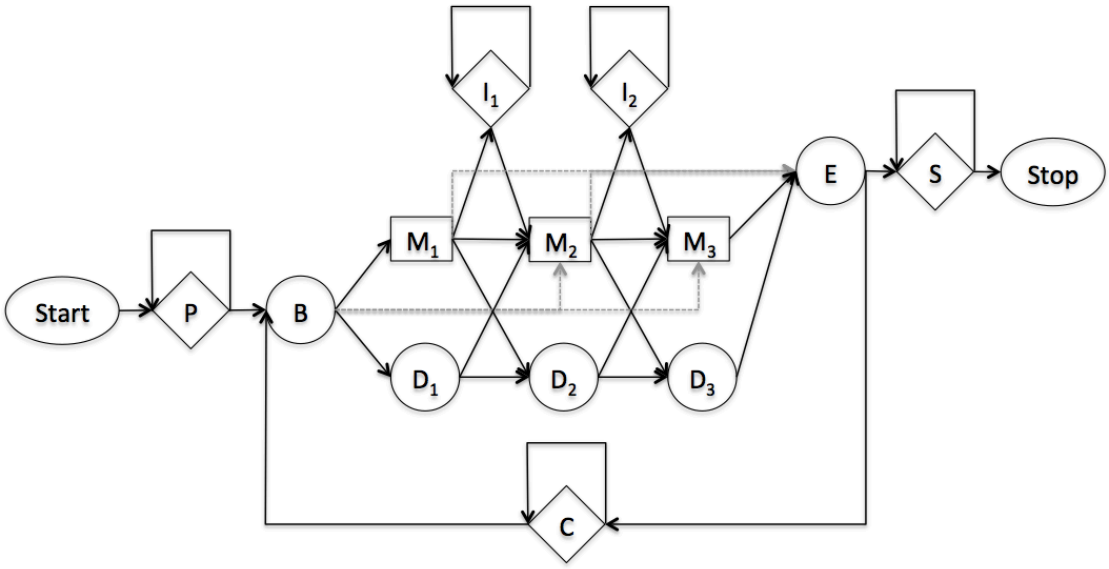


Figure 1.2. An example of a typical state diagram of an HMM representing a gene domain.

As described in (Johnson, 2006), an HMM for a gene domain can be represented as a state diagram (Fig. 1.2), where states are represented by nodes and directed edges represent transitions between the nodes. The rectangle match states $\langle M_1, M_2, M_3 \rangle$ correspond to the conserved columns of the corresponding multiple sequence alignment. The emit probabilities at the match states correspond to the normalized counts of amino acids in the respective conserved columns of the multiple sequence alignment. The diamond shape insert states $\langle I_1, I_2 \rangle$ represent insertions of amino acids between the match states. The round delete states $\langle D_1, D_2, D_3 \rangle$ represent silent states that do not emit any amino acid. The *P* (i.e. prefix) and *S* (i.e. suffix) states represent the prefix and suffix amino acid sequences of the gene domain within a query sequence. The *C* (i.e. copy) state allows multiple copies of the gene domain within a query sequence. The dotted lines allow a partial match of a query sequence to the gene domain. States *B* and *E* represent the beginning and the end of the gene domain, respectively.

1.5.3 Bayesian classifier

A naïve Bayesian classifier can be used for confident taxonomic classification of the bacterial 16S rRNA partial gene sequences (Q. Wang *et al.*, 2007; Rosen *et al.*, 2011). We have employed the naïve Bayesian classifier to taxonomically classify marker gene sequences found by HMMs within the input sequences of *PhyloPythiaS+* (Gregor, Dröge, *et al.*, 2016). These sequences carrying marker genes were subsequently used as

training data for the composition based taxonomic binning using structured support vector machines (SVMs). Let $C = \{c_1, \dots, c_n\}$ be a set of $n \in \mathbb{N}$ distinct classification classes, e.g. different genera $\{Escherichia, Salmonella, Haemophilus, \dots\}$. Let $P(c_i)$ for $i \in \{1, \dots, n\}$ be prior probabilities of the individual classes, e.g. $P(Escherichia)$ is the prior probability that a randomly drown sequence from a metagenome sample has genus *Escherichia*. Let f be a function that assigns a d-dimensional feature vector $\boldsymbol{v} \in \mathbb{R}^d$ to every query sequence s that can be drawn from a metagenome sample, i.e. $\boldsymbol{v} = f(s)$. Let $P(\boldsymbol{v} \mid c_i)$ for $i \in \{1, \dots, n\}$ be the class conditional probability (likelihood), i.e. the probability that feature vector $\boldsymbol{v}$ was produced by function f for a sequence originating from genus c_i. As described in (Duda *et al.*, 2000), the Bayes formula is defined as:

$$P(c_i \mid \boldsymbol{v}) = \frac{P(\boldsymbol{v} \mid c_i) * P(c_i)}{P(\boldsymbol{v})}$$

Where the evidence factor is:

$$P(\boldsymbol{v}) = \sum_{j=1}^{n} P(\boldsymbol{v} \mid c_j) * P(c_j)$$

Given a query sequence s drawn from a metagenome sample and the corresponding $\boldsymbol{v} = f(s)$, the posterior probability $P(c_i \mid \boldsymbol{v})$ is computed for each class, where $i \in \{1, \dots, n\}$. The sequence s is then assigned to the class (genus) c_j for which the posterior probability is the highest, i.e. $P(c_j \mid \boldsymbol{v}) = \max_{i \in \{1,\dots,n\}} \{P(c_i \mid \boldsymbol{v})\}$. Note that the evidence factor $P(\boldsymbol{v})$ is not necessary for the decision-making and is used only to guarantee that the posterior probabilities sum up to one.

As described in (Q. Wang *et al.*, 2007), for the classification of the bacterial 16S rRNA partial gene sequences, a feature space consisting of all 8-mers showed the best performance. An 8-mer is a word of length eight that is a sub-sequence of the query sequence s. As there are $(d = 4^8 = 65{,}536)$ distinct 8-mers, a query sequence s can be represented by a feature vector $\boldsymbol{v} \in \{0, 1\}^d$, where $v_e = 1$ if the 8-mer with index e is present in the query sequence s, else $v_e = 0$, for $e \in [1, \dots, d]$. Given a set of training sequences for each genus c_i, the probabilities $P(w_e \mid c_i)$ can be estimated, for each $i \in \{1, \dots, n\}$ and each $e \in [1, \dots, d]$. Where, $P(w_e \mid c_i)$ is the probability that an 8-mer with index e is contained within a sequence originating from genus c_i. Let $S_{\boldsymbol{v}}$ be the set of all

8-mers contained in the query sequence s, i.e. $\boldsymbol{v} = f(s)$ and $S_{\boldsymbol{v}} = \{w_e \mid \forall e \in [1, \ldots, d]\, s.t.\, v_e = 1\}$. The probability that the query sequence s originating from genus c_i contains a set of 8-mers $S_{\boldsymbol{v}}$ can be estimated as $P(S_{\boldsymbol{v}} \mid c_i) = \prod_{w_e \in S_{\boldsymbol{v}}} P(w_e \mid c_i)$. The class conditional probability from the Bayes formula is thus $P(\boldsymbol{v} \mid c_i) = P(S_{\boldsymbol{v}} \mid c_i)$. In the case that the prior probabilities are equal, i.e. all genera are equally likely, the classification depends only on the class conditional probabilities $P(\boldsymbol{v} \mid c_i)$. Therefore, a sequence s is assigned to class (genera) c_j, for which the class conditional probability $P(\boldsymbol{v} \mid c_j)$ is the highest. Note, that such a classifier is called "naïve" since we assume that the 8-mers represent independent features, although this condition is violated, as overlapping 8-mers are actually dependent features. Also note that the sequence classification at taxonomy ranks different from the genus rank is analogous.

1.5.4 Support vector machines

Support vector machines (SVMs) (Vapnik, 1995; Duda *et al.*, 2000) have been successfully employed to taxonomically classify DNA sequences of metagenome samples (McHardy *et al.*, 2007; Patil, Haider, *et al.*, 2011). We have also employed SVMs for the composition based taxonomic classification of metagenome sequences in *PhyloPythiaS+* (Gregor, Dröge, *et al.*, 2016). Let $X = \mathbb{R}^d$ be the input space of d-dimensional feature vectors representing DNA sequences of variable lengths. For instance, as described in (Patil, Haider, *et al.*, 2011), a feature vector for a DNA sequence corresponds to the frequencies of 4–6-mers that are further normalized by the sequence length and scaled, such that the individual features are from the normal distribution with zero mean and standard deviation one, i.e. $x_j \in \mathcal{N}(0,1)$ for $j \in [1, \ldots, d]$ and $(d = 4^4 + 4^5 + 4^6 = 5{,}376)$. Such a feature vector representation was chosen since the composition of genomic sequences carries a phylogenetic signal (McHardy *et al.*, 2007; Patil, Haider, *et al.*, 2011). Let $Y = \mathbb{Z}^r$ be the r-dimensional output space representing sequence labels, i.e. taxonomic assignments of the DNA sequences. Given a set of $N \in \mathbb{N}$ labeled training samples $S = \{(\boldsymbol{x_i}, \boldsymbol{y_i}) \in X \times Y \mid i \in [1, \ldots, N]\}$, the goal of a support vector machine framework is to learn the underlying function $F: X \to Y$ in the training phase. The challenge is to learn the function F, such that it has a good generalization property, i.e. it correctly assigns labels from Y to all the input data from X that were previously not seen in the training phase.

Let us consider a simple classifier that is based on linear SVMs, described in (Duda *et al.*, 2000). Let $X = \mathbb{R}^d$ be the d-dimensional input space as described in the previous paragraph. Let $Y = \{-1, 1\}$ be the one-dimensional output space representing two genera: $(-1) \sim Escherichia$ and $(1) \sim Salmonella$. Let S be the set of training samples $S = \{(\boldsymbol{x_i}, y_i) \in X \times Y \mid i \in [1, \dots, N]\}$. The goal of linear SVMs is to find a hyperplane, defined by the weight vector $\boldsymbol{w} \in \mathbb{R}^{d+1}$, which separates samples with label (-1) from samples with label (1) by maximizing the margin, i.e. the distance between the hyperplane and the closest point to the hyperplane. The classification function, i.e. the estimate of F, is defined as:

$$f(\boldsymbol{x}) = sign\left(\sum_{j\epsilon\,[1,\dots,d]} (w_j * x_j) + w_0\right)$$

In the formula, the $sign$ function assigns (1) for positive input values and (-1) for negative inputs values, else zero. The weight vector $\boldsymbol{w}$ can be found by:

$$\boldsymbol{w} = \arg\min_{\boldsymbol{w'}\in\mathbb{R}^{d+1}} \left(\frac{1}{2} * \|\boldsymbol{w'}\|^2\right)$$

Such that:

$$y_i * \left(\sum_{j\epsilon\,[1,\dots,d]} (w_j * x_{i,j}) + w_0\right) \geq 1, \forall i \in [1, \dots, N]$$

Here, we are searching for the simplest solution – the weight vector $\boldsymbol{w}$, such that all the training samples are classified correctly with the margin of at least one.

As the training samples are oftentimes not separable by a hyperplane, a certain error is allowed in the training phase. This is done by the introduction of the slack variables $\boldsymbol{\xi} \in \mathbb{R}^N$ and the trade-off parameter $C \in \mathbb{R}^+$. In this "relaxed" settings, the weight vector $\boldsymbol{w}$ can be found by:

$$\boldsymbol{w} = \arg\min_{\boldsymbol{w'}\in\mathbb{R}^{d+1}, \boldsymbol{\xi}\in\mathbb{R}^N} \left(\frac{1}{2} * \|\boldsymbol{w'}\|^2 + \frac{C}{N} * \sum_{i\in\,[1,\dots,N]} \xi_i\right), \xi_i \geq 0\ \forall i \in [1, \dots, N]$$

Such that:

$$y_i * \left(\sum_{j \epsilon\, [1,\ldots,d]} \left(w_j * x_{i,j}\right) + w_0 \right) \geq 1 - \xi_i, \forall i \in [1, \ldots, N]$$

In this formulation, the slack variable ξ_i represents a maximum error allowed per a training sample. The trade-off parameter C represents how many errors we allow in the training phase. Note that the detailed description of how the weight vector $\boldsymbol{w}$ can be found is described in (Vapnik, 1995; Duda *et al.*, 2000). After the weight vector $\boldsymbol{w}$ is found based on the training samples from S, the classification function f can be used to classify unlabeled DNA sequences, originating from either *Escherichia* or *Salmonella*, to either *Escherichia* or *Salmonella*.

In the taxonomic classification method *PhyloPythiaS* (Patil, Haider, *et al.*, 2011), structured SVMs were used. Let $X = \mathbb{R}^d$ be the d-dimensional input space as described in the first paragraph of this section. Let $Y = \{0, 1\}^r$ be the r-dimensional output space representing taxonomic assignments of the DNA sequences. Here, the output space is structured and represents a part, i.e. a sub-tree of the taxonomy tree, where each node of the sub-tree is assigned a unique index $q \in [1, \ldots, r]$. A label $\boldsymbol{y} \in Y$ represents a path in the taxonomy sub-tree from the root to one of its leafs, such that if the node of the taxonomy sub-tree with index $q \in [1, \ldots, r]$ is on the path then y_q is set to one, else zero. Let $\Delta(\boldsymbol{y}, \hat{\boldsymbol{y}})$ be the loss function defining the discrepancy between two outputs $\boldsymbol{y}, \hat{\boldsymbol{y}} \in Y$, which is defined as the length of the shortest path connecting two leaf nodes of the two paths $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$. Let $\Psi(\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{R}^m$ be the joint feature map that encodes correlations between different inputs and outputs. The classification function is defined as:

$$f(\boldsymbol{x}) = \arg\max_{\boldsymbol{y} \in Y} \left(\boldsymbol{w}^T \Psi(\boldsymbol{x}, \boldsymbol{y}) \right)$$

A DNA sequence corresponding to feature vector $\boldsymbol{x}$ is thus assigned to label $\boldsymbol{y}$ for which the scalar product $\boldsymbol{w}^T \Psi(\boldsymbol{x}, \boldsymbol{y})$ is the highest. The weight vector $\boldsymbol{w} \epsilon\, \mathbb{R}^{\boldsymbol{m}}$ can be found using the maximum margin structured support vector machine framework by:

$$\boldsymbol{w} = \arg \min_{\boldsymbol{w'} \in \mathbb{R}^m, \boldsymbol{\xi} \in \mathbb{R}^N} \left(\frac{1}{2} * \|\boldsymbol{w'}\|^2 + \frac{C}{N} * \sum_{i \in\, [1,\ldots,N]} \xi_i \right), \xi_i \geq 0 \; \forall i \in [1, \ldots, N]$$

Such that:

$$\boldsymbol{w}^T\left(\Psi(\boldsymbol{x_i}, \boldsymbol{y_i}) - \Psi(\boldsymbol{x_i}, \hat{\boldsymbol{y}})\right) \geq 1 - \frac{\xi_i}{\Delta(\boldsymbol{y_i}, \hat{\boldsymbol{y}})} \quad , \forall i \in [1, \dots, N] \; \forall \hat{\boldsymbol{y}} \in Y$$

For further details on the method, see to the supplemental material of (Patil, Haider, *et al.*, 2011).

1.6 Outline

This dissertation is a cumulative dissertation that consists of two main publications that I published as the first author in peer-reviewed international journals during my PhD studies. The goal of this dissertation was to enrich the field of bioinformatics and metagenomics by developing new methods that will help researchers to analyze and interpret their data. The articles are ordered in a logical order. As assembly is oftentimes performed before taxonomic binning, we first describe our gene assembler and then our taxonomic binning method. Chapter 1 puts our work into a larger scientific context. We introduce the field of metagenomics and describe methods for sequence analysis that we employed as sub-routines in our work. In Chapter 2, I list all publications that I co-authored during my PhD, where all the main listed publications have already been published in peer-reviewed international journals. The main publications of this dissertation are in Chapters 3 and 4, where both articles are identical to the published versions except for section numbering and formatting. Chapters 5 and 6 contain the synopsis and a list of all references used in this work, respectively. The original versions of the published articles can be found in Chapter 7.

***Snowball:* strain aware gene assembly of metagenomes**

We have developed a strain aware gene assembler for metagenomes, described in Chapter 3. To the best of our knowledge, this is the first gene assembler for metagenomic data that can distinguish gene variants of individual strains without using closely related reference genomes of the studied species. This is a very important property as metagenomes originating from novel environments oftentimes contain new unknown species for which there are no closely related reference genomes available. Moreover, for many purposes, including functional analysis of metagenomic data, it is sufficient to assemble only the coding sequences of the strains, as usually more than

85% of prokaryotic genomes are coding sequences (S. Cole and Saint-Girons, 1999). We believe that this method will be useful for researchers studying gene variation among strains, genes under selection, virulent genes and species evolution.

PhyloPythiaS+: a self-training method for the rapid reconstruction of low-ranking taxonomic bins from metagenomes

We have developed an automated composition based taxonomic binning method, described in Chapter 4. This method is a successor to the *PhyloPythiaS* (Patil, Haider, *et al.*, 2011) software. We have fully automated this method by adding a new marker-gene based framework that automatically determines the most relevant taxa to be modeled and suitable training sequences directly from the input metagenome sample. To the best of our knowledge, this is the first method that combines taxonomic profiling and subsequent taxonomic composition based binning of the whole input metagenome sample. Moreover, we developed a new *k*-mer counting algorithm that accelerated the whole method and showed state-of-the-art performance for the simultaneous enumeration of *4*–*6*-mers, which is commonly used for composition based binning. We also extensively evaluated the whole automated taxonomic binning pipeline by comparing it to the other methods and devised several new evaluation measures. The results showed that our method performed especially well for samples originating from novel environments in comparison to the other methods. These results were also confirmed in the CAMI challenge (Sczyrba *et al.*, 2017), in which *PhyloPythiaS+* demonstrated its high recall and ability to correctly assign taxa that have longer taxonomic distances to the known reference genomes or draft genomes. This is very important for researchers studying metagenome samples originating from novel environments, for draft genome reconstruction and for the subsequent functional analysis of the studied metagenome microbial communities.

2 Personal bibliography

Publications of the thesis

- I. Gregor, A. Schönhuth, and A. C. McHardy (2016). "*Snowball*: strain aware gene assembly of metagenomes." *Bioinformatics* 32 (17): i649-i657 doi:10.1093/bioinformatics/btw426 http://bioinformatics.oxfordjournals.org/content/32/17/i649.full

- I. Gregor, J. Dröge, M. Schirmer, C. Quince, and A. C. McHardy (2016). "*PhyloPythiaS+*: a self-training method for the rapid reconstruction of low-ranking taxonomic bins from metagenomes." *PeerJ* 4:e1603 doi:10.7717/peerj.1603 https://doi.org/10.7717/peerj.1603

Other publications

- I. Gregor, L. Steinbrück, and Alice C. McHardy (2013). "*PTree*: pattern-based, stochastic search for maximum parsimony phylogenies." *PeerJ* 1:e89 doi:10.7717/peerj.89 https://doi.org/10.7717/peerj.89
  Given an initial implementation of the tree reconstruction method in R, I re-implemented the method in Java (and one sub-routine in C). Based on the discussions with co-authors, I implemented several improvements that substantially decreased both the runtime and the parsimony costs of the reconstructed phylogenetic trees. I also evaluated the method and wrote the manuscript with co-authors.

- J. Dröge, I. Gregor, and A. C. McHardy (2014). "*Taxator-tk*: Precise Taxonomic Assignment of Metagenomes by Fast Approximation of Evolutionary Neighborhoods." *Bioinformatics* 31 (6): 817-824 doi:10.1093/bioinformatics/btu745 http://bioinformatics.oxfordjournals.org/content/31/6/817
  I contributed to the evaluation framework, including the preparation of the reference data, which was also used to evaluate the *PhyloPythiaS+* software. I also commented on the manuscript.

- P. B. Pope, A. K. Mackenzie, I. Gregor, W. Smith, M. A. Sundset, A. C. McHardy, M. Morrison, V. G. H. Eijsink (2012). "Metagenomics of the Svalbard Reindeer Rumen Microbiome Reveals Abundance of Polysaccharide Utilization Loci." *PLoS ONE* 7(6): e38571
  doi:10.1371/journal.pone.0038571
  http://dx.doi.org/10.1371/journal.pone.0038571
  I employed an initial version of the *PhyloPythiaS+* software to analyze the metagenome sample originating from the rumen of the Svalbard reindeer, which contained a dominant and novel *Bacteroidales* clade (*SRM-1),* for which we got the sample-derived data. The taxonomic binning that I computed was used for the further analysis. I described how the taxonomic binning was computed.

- W. Ikeda-Ohtsubo, J. F. H. Strassert, T. Köhler, A. Mikaelyan, I. Gregor, A. C. McHardy, S. Green Tringe, P. Hugenholtz, R. Radek, and A. Brune (2016). "'Candidatus Adiutrix intracellularis', an endosymbiont of termite gut flagellates, is the first representative of a deep-branching clade of Deltaproteobacteria and a putative homoacetogen." *Environmental Microbiology* 18(8):2548-2564
  doi:10.1111/1462-2920.13234
  http://onlinelibrary.wiley.com/doi/10.1111/1462-2920.13234/abstract
  I employed an initial version of the *PhyloPythiaS+* software to analyze the metagenome sample originating from the termite gut flagellates, which contained a novel deep-branching uncultured *Deltaproteobacterium*, for which we got the sample-derived data. The taxonomic binning that I computed was used for the further analysis.

- A. Sczyrba, P. Hofmann, P. Belmann, D. Koslicki, S. Janssen, J. Dröge, I. Gregor, S. Majda, J. Fiedler, E. Dahms, A. Bremges, A. Fritz, R. Garrido-Oter, T. S. Jørgensen, N. Shapiro, P. D. Blood, A. Gurevich, Y. Bai, D. Turaev, M. Z. DeMaere, R. Chikhi, N. Nagarajan, C. Quince, L. H. Hansen, S. J. Sørensen, B. K. H. Chia, B. Denis, J. L. Froula, Z. Wang, R. Egan, D. D. Kang, J. J. Cook, C. Deltel, M. Beckstette, C. Lemaitre, P. Peterlongo, G. Rizk, D. Lavenier, Y. W. Wu, S. W. Singer, C. Jain, M. Strous, H. Klingenberg, P. Meinicke, M. Barton, T. Lingner, H. Hung Lin, Y. C. Liao, G. G. Z. Silva, D. A. Cuevas, R. A. Edwards, S. Saha, V. C. Piro, B. Y. Renard, M. Pop, H. P. Klenk, M. Göker, N. Kyrpides, T. Woyke, J. A. Vorholt, P. Schulze-Lefert, E. M. Rubin, A. E. Darling, T. Rattei, A. C. McHardy (2017).

"Critical Assessment of Metagenome Interpretation – a benchmark of computational metagenomics software." *bioRxiv (preprint)*
doi: 10.1101/099127
http://biorxiv.org/content/early/2017/01/09/099127
Accepted also for a publication in *Nature Methods*.
I contributed to the initial design of the CAMI challenge. I provided the *PhyloPythiaS+* software and a set of evaluation measures in a Docker container. I computed the taxonomic binning for all the challenge datasets using *PhyloPythiaS+* and its marker gene component.

3 *Snowball*: strain aware gene assembly of metagenomes

Status	**published**
Journal	*Bioinformatics* (Impact factor 7.307)
Citation	I. Gregor, A. Schönhuth, and A. C. McHardy (2016). "*Snowball*: strain aware gene assembly of metagenomes." *Bioinformatics* 32 (17): i649-i657 doi:10.1093/bioinformatics/btw426
URL	http://bioinformatics.oxfordjournals.org/content/32/17/i649.full
Own contribution	75% Designed and performed the experiments (with co-authors) Analyzed the data (with co-authors) Wrote the manuscript (with co-authors)

Acknowledgement - Author self-archiving policy

This is a pre-copyedited, author-produced version of an article accepted for publication in *Bioinformatics* following peer review. The version of record "I. Gregor, A. Schönhuth, and A. C. McHardy (2016). *Snowball*: strain aware gene assembly of metagenomes. *Bioinformatics* 32 (17): i649-i657" is available online at:

http://bioinformatics.oxfordjournals.org/content/32/17/i649.full

and doi:10.1093/bioinformatics/btw426.

Abstract

Motivation: Gene assembly is an important step in functional analysis of shotgun metagenomic data. Nonetheless, strain aware assembly remains a challenging task, as current assembly tools often fail to distinguish among strain variants or require closely related reference genomes of the studied species to be available.

Results: We have developed *Snowball*, a novel strain aware gene assembler for shotgun metagenomic data that does not require closely related reference genomes to be available. It uses profile hidden Markov models (HMMs) of gene domains of interest to guide the assembly. Our assembler performs gene assembly of individual gene domains based on read overlaps and error correction using read quality scores at the same time, which results in very low per-base error rates.

Availability and Implementation: The software runs on a user-defined number of processor cores in parallel, runs on a standard laptop and is available under the GPL 3.0 license for installation under Linux or OS X at https://github.com/hzi-bifo/snowball.

3.1 Introduction

Metagenomics is the functional or sequence-based analysis of microbial DNA isolated directly from a microbial community of interest (Kunin *et al.*, 2008; Riesenfeld *et al.*, 2004). This enables the analysis of microorganisms that cannot be cultivated in a laboratory. After the DNA is isolated, it is sequenced using a high-throughput sequencing platform, which results in a large dataset of short sequenced genome fragments, called reads. For a read, it is unknown from which strain it originates. Given such sequenced shotgun metagenomic data, i.e. a dataset of short reads that originate from several genome sequences of distinct strains, gene assembly aims to reconstruct coding sequences of the individual strains contained in the dataset (Fig. 3.1).

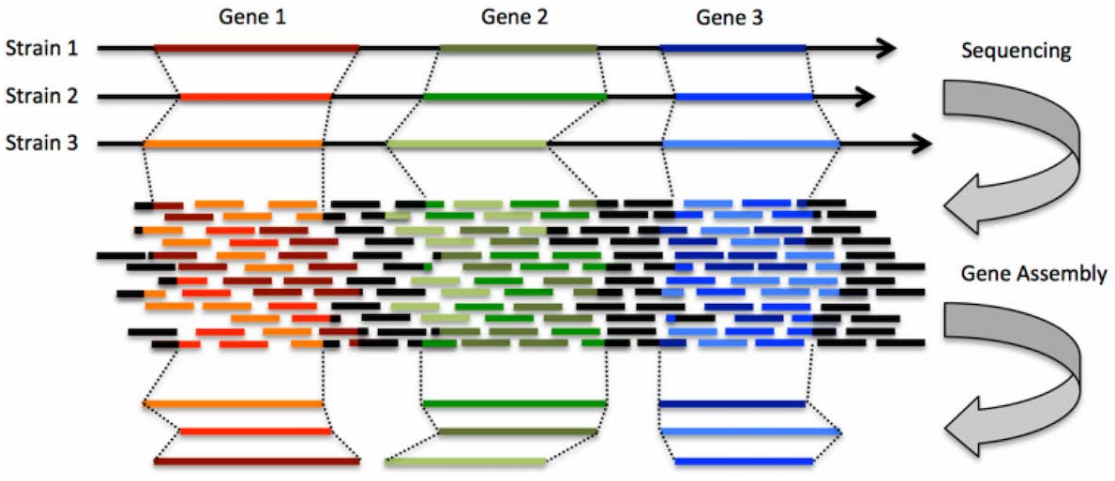


Figure 3.1. An example of the gene assembly problem.
In this example, the sequenced microbial community consists only of three distinct strains. Non-coding regions of the strain sequences are black, whereas coding regions are red, green and blue for genes 1, 2 and 3. Genes 1–3 are present in all three strains, although the location and gene sequences differ for distinct strains. The sequencing step results in a collection of short reads. Note that after the sequencing step, the origin of reads denoted by colours and positions within the respective strains in the figure is not known in the subsequent gene assembly step. Given a dataset containing all the short reads, the ultimate goal of the gene assembly is to determine the individual strain specific sequences of the genes.

Gene assembly is an important step in the analysis of shotgun metagenomic data. For many purposes, including functional analysis of metagenomic data, it is sufficient, and therefore convenient to assemble only the coding sequences of the strains. It has also been shown that genes assemble well (Kingsford *et al.*, 2010) even when only short reads are available. Moreover, metagenomic data consist mainly of prokaryotic species.

As usually more than 85% of prokaryotic genomes are coding sequences (S. Cole and Saint-Girons, 1999); gene assembly enables to recover large parts of the respective genomes.

Importantly, strain awareness is an essential goal in assembling metagenomes, since it enables us to study gene variation among strains of a species from the sequenced microbial community, which is where much phenotypic diversity also arises. However, the assembly of closely related strains remains a challenging task. Strain aware assembly, which is assembly that is sensitive to closely related haplotypic sequences has remained an open challenge in many genomics applications. In particular, low-abundance strains can interfere with sequencing errors in common error correction routines. To date, most assembly tools still aim to assemble consensus sequence, if closely related haplotypes are present (Marschall *et al.*, 2016).

There are few tools that enable strain variant reconstruction. They often rely on the availability of closely related reference genomes of the studied species (Ahn *et al.*, 2015; Töpfer *et al.*, 2014; Zagordi *et al.*, 2011), where reads are first mapped onto a reference genome, using a read mapping tool, e.g. BWA (H. Li and Durbin, 2009), strain variants are then identified through a reference guided strain aware assembly. As metagenome samples originating from novel environments typically consist of novel species without reference genomes available, there is a need for new reference-free approaches.

Tools that are often used for *de novo* metagenome assemblies are *Ray Meta* (Boisvert *et al.*, 2012), *MEGAHIT* (D. Li *et al.*, 2015), *IDBA-UD* (Peng *et al.*, 2012), *MetaVelvet* (Namiki *et al.*, 2012) or *SOAPdenovo2* (Luo *et al.*, 2012). All these tools are *k*-mer based, i.e. they transform reads into overlapping *k*-mers from which *de Bruijn* graphs are built, where paths in the graph correspond to the assembled contigs. This general approach, however, often fails to distinguish among strain variants. There has been recent debate on *k*-mer based approaches using *de Bruijn* graphs in strain aware assembly. In particular, *k*-mer based approaches can become misled, when low-abundance strains are involved, since the frequencies of the low-abundance strains are on the order of magnitude of the sequencing error rates. This leads to unpleasant interference in *k*-mer based error-correction steps, as low-abundance strains are often removed along with sequencing errors. For strain aware assembly, it is helpful to process reads at their full length, because this increases the power to distinguish low-frequent, co-occurring true mutations from sequencing errors. In this line, there has been recent evidence that

shorter genomes can be assembled through overlap graph based approaches, which make use of full-length reads, using short reads (Simpson and Durbin, 2012). It was also shown that one can perform strain aware assembly through iterative construction of overlap graphs (Töpfer *et al.*, 2014). For gene assembly from metagenomic data, the *SAT* assembler (Zhang *et al.*, 2014) can be employed. First, it assigns reads to gene domains of interest based on profile hidden Markov models (HMMs) (Eddy, 2011; Finn *et al.*, 2014) of the respective gene domains. Then, for each gene domain, separately, it builds overlap graphs based on the read overlaps, where the paths in the graphs correspond to the assembled contigs. However, the *SAT* assembler does not implement a sophisticated error-correction strategy, which is considered crucial for strain aware assembly. For the reconstruction of 16S genes, which are often used for phylotyping, *REAGO* (Yuan *et al.*, 2015) can be employed. Since it has been built for 16S genes, the use of *REAGO* in more generic settings remains unclear.

The current sequencing technologies still produce relatively short erroneous reads, making it difficult to distinguish sequencing errors from genuine strain variation (Laehnemann *et al.*, 2016). Therefore, reference-free strain reconstruction of the full-length sequences of individual strains is currently considered to be a tough computational challenge, as there may be no immediate sufficient information in the sequenced data if mutations are separated by too large stretches of sequence that agree for several strains. Therefore, new approaches are needed that push the limits imposed by the data.

Here, we present *Snowball*, a novel method for strain aware gene assembly from metagenomes that addresses the above-mentioned points. It does not require closely related reference genomes to be available. It uses profile HMMs of gene domains of interest as an input to guide the assembly. The HMM profile-based homology search is known to be capable of finding remote homology, including large number of substitutions, insertions and deletions, whereas simple read mapping onto a reference genome can find only very closely related homologs (Zhang *et al.*, 2014). Since our method does not make use of reference genomes, we allow for strain aware gene assembly also of novel species, where reference genomes are not yet available. We have developed a novel algorithm that performs gene assembly based on read overlaps. This allows correcting errors by making use of the error profiles that underlie the overlapping reads. The consequences are twofold: First, we obtain contigs affected by only very low per-base error rates. Second, since, this way, we determine which reads

stem from identical segments based on a statistically sound model, we can reliably distinguish between sequencing errors and strain-specific variants, even of very low-abundance strains. We consider these two features to represent the main improvements over the currently available assemblers. To the best of our knowledge, *Snowball* is the first tool that allows distinguishing among individual gene strain variants in metagenomes for a large set of gene domains without using reference genomes of related species.

In our experiments, we focused on distinguishing closely related strains from one species. Since two different species are substantially more divergent in terms of sequence than two different strains from the same species, good results on strains from one species also imply good or even better performance on datasets that contain several species – distinguishing species is the much easier task. We assessed the performance of *Snowball* using 21 simulated datasets, each containing 3–9 closely related *Escherichia coli* strains and on one simulated dataset containing ten recently published strains of a novel *Rhizobia* species (Bai *et al.*, 2015). The results for the latter demonstrate the capability of the *Snowball* assembler to assemble genes of novel strains. The results for all datasets confirm that the strength of *Snowball* is its very low per-base error, due to the incorporated error-correction. Moreover, it produced substantially longer contigs and recovered a larger part of the simulated reference data in comparison to the *SAT* assembler. *Snowball* is implemented in Python, runs on a user-defined number of processor cores in parallel, runs on a standard laptop, is freely available under the GPL 3.0 license and can be installed under Linux or OS X.

3.2 Methods

The input of *Snowball* are two FASTQ files containing Illumina self-overlapping paired-end reads, the corresponding insert size used for the library preparation and profile HMMs of gene domains of interest. The paired-end reads may originate from multiple closely related strains or from more evolutionary divergent taxa. We have thoroughly tested *Snowball* using simulated Illumina HiSeq 2500 paired-end reads generated by the *ART* read simulator (Huang *et al.*, 2012) with 150 bp read length and 225 bp mean insert size. In this setting, the average length of the self-overlaps of the read ends is 75 bp and the length of a consensus read that originates by joining of the self-overlapping read ends is 225 bp on average (Fig. 3.2, Section 3.3.4).

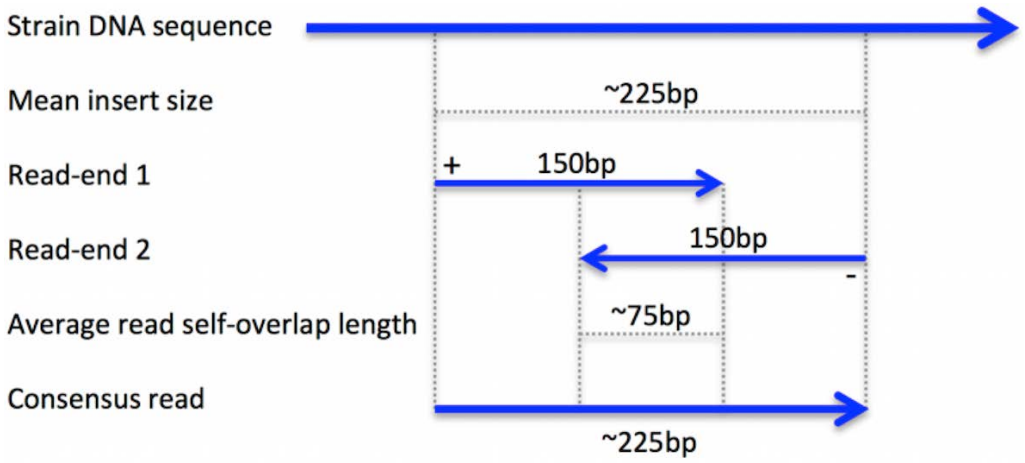


Figure 3.2. An example of a self-overlapping paired-end read.
Illumina HiSeq 2500 paired-end read consists of two 150 bp read ends, one on the positive strand (+) and one on the negative strand (–). In our example, the mean insert size (225 bp) is smaller than two times the read end length (2 × 150 bp), therefore the paired-end reads are self-overlapping with 75 bp overlap length on average. Such a self-overlapping read can be joined into a consensus read of 225 bp length on average.

The output is a FASTA or a FASTQ file containing annotated assembled contigs. For each contig, the annotation contains the name of a respective gene domain to which a contig belongs, coordinates of the coding sub-sequence within a contig sequence, coverage and quality score for each contig position. The coverage and quality score information can be used for subsequent quality filtering yielding less or shorter contigs of higher quality.

Our method consists of the following steps:

- [Consensus read reconstruction]

  Self-overlapping paired-end reads are joined into longer consensus reads (Section 3.2.1).

- [Assignment of consensus reads to gene domains]

  Profile HMMs of selected gene domains are employed to assign consensus reads to the respective gene domains, where one consensus read is assigned to at most one gene domain (Section 3.2.2).

- [Assembly of consensus reads into contigs]

  For each gene domain, in parallel, consensus reads are assembled into contigs (Sections 3.2.3–3.2.5). In the assembly step, consensus reads are iteratively joined into longer and error-corrected super-reads based on the consensus read overlaps. The super-reads are then output as annotated contigs, where a super-read represents a sequence that originates by joining of at least two consensus reads into a longer sequence.

3.2.1 Joining self-overlapping paired-end reads

Self-overlapping paired-end reads are joined into longer error-corrected consensus sequences. The use of a library containing self-overlapping paired-end reads is a powerful strategy for an initial error-correction (Schirmer *et al.*, 2015), which has been employed in e.g. *ALLPATHS* (Butler *et al.*, 2008). Given the mean insert size, we determine the self-overlap that results in the minimum Hamming distance between the overlapping ends of a paired-end read. A base with a higher quality score is chosen at a position within the overlap that contains mismatching bases for the respective position of the resulting consensus read sequence (Fig. 3.3).

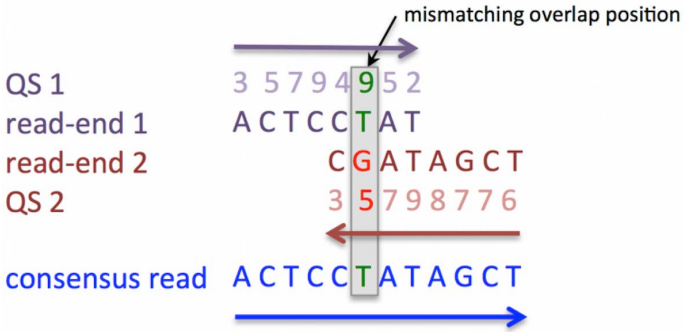


Figure 3.3. Joining of self-overlapping reads example.
The figure depicts a simplified example of a consensus read reconstruction. At the mismatching overlap position, read-end 1 has *T* with quality score (QS) 9, while read-end 2 has *G* with quality score 5. The resulting consensus read will have *T* at the respective position, since *T* is supported by a higher quality score than *G*. The computation of the quality scores for the consensus read is explained in the Section 3.2.3.

As the substitution error rate of the Illumina reads increases towards the ends of the paired-end reads (Minoche *et al.*, 2011), this step results in longer consensus reads with overall lower substitution error, where the overlapping regions are almost error-free. It is also an efficient read quality filtering step, as the paired-end reads that cannot be joined, due to high substitution error rate, an insertion or a deletion within the overlapping region, are filtered out. For instance, by joining of the 150 bp paired-end Illumina HiSeq 2500 self-overlapping reads with 225 bp mean insert size results in consensus reads of length 225 bp on average. While the default error profile of the *ART* read simulator (Huang *et al.*, 2012) yields 150 bp paired-end reads with ~2.37% substitution error, the joined consensus reads had only ~1.08% substitution error in our experiments. These longer, error-corrected consensus reads with low substitution error rate are convenient building blocks to start with in the subsequent steps of our method.

3.2.2 Assigning reads to gene domains

Consensus reads are annotated using profile HMMs of gene domains of interest and assigned to respective gene domains (Fig. 3.4). By default, we use the *Pfam-A* (Finn *et al.*, 2014) (version 27) profile HMMs of 14,831 gene domains and *AMPHORA 2* (Wu and Scott, 2012) profile HMMs of 31 bacterial ubiquitous single-copy genes that are often used for phylotyping. A profile HMM of a gene domain is a probabilistic model representing a multiple sequence alignment of representative gene sequences belonging to a particular gene domain. The model can be used to annotate a query sequence (e.g. a consensus read). The annotation mainly consists of a score, start/stop positions within a query sequence and HMM start/stop coordinates. The score roughly corresponds to a probability that a query sequence belongs to the particular gene domain, i.e. if the score is high for a query sequence then it is very probable that it belongs to the respective gene domain. The start/stop positions within a query sequence define a sub-sequence of a query sequence that was identified to belong to the gene domain. The HMM start/stop coordinates correspond to the estimated coordinates of the query sub-sequence within the multiple sequence alignment of the respective profile HMM.

Each consensus read is translated into six protein sequences using all six reading frames (i.e. also considering the reverse complementary sequences). The *hmmsearch* command of the *HMMER 3* (Eddy, 2011) software is used to annotate the protein sequences. For each consensus read, only the reading frame with the highest score is considered. A consensus read is assigned to at most one gene domain to which it was queried with the highest score. Consensus reads with low scores (i.e. lower than default value: 40) are filtered out and not considered in the subsequent steps. If a protein sequence corresponding to a reverse complementary consensus read sequence was annotated, the corresponding reverse complementary DNA sequence of a respective consensus read is considered in the next steps. The coding DNA sub-sequence of a consensus read sequence is denoted as a (partial) coding region. The start and end HMM coordinates within a respective profile HMM are stored as part of the consensus read annotation.

As a result of this step, consensus reads are annotated and assigned to 'bins' representing individual gene domains, where one consensus read is assigned to at most one gene domain. Gene domains are building blocks of individual genes. Therefore, a 'bin' does not only contain consensus reads belonging to gene variants of individual

strains. It can also contain different genes of one strain, several copies of one gene of one strain or even 'broken' gene copies.

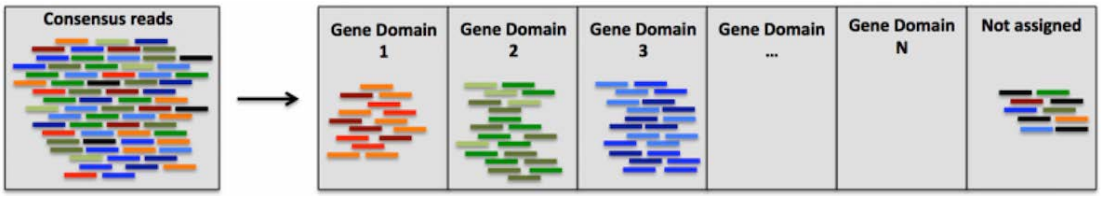


Figure 3.4. Assignment of consensus reads to gene domains.
Consensus reads are assigned to individual gene domains using profile HMMs. Consensus reads that cannot be assigned to any of the gene domains with sufficient confidence remain unassigned. A consensus read is assigned to at most one gene domain.

3.2.3 Consensus sequence representation

We represent consensus sequences, i.e. consensus reads and super-reads using probability matrices. A super-read is a longer error-corrected sequence that originates by joining overlapping consensus reads (or consensus reads with super-reads) in the *Snowball* algorithm (Section 3.2.5).

For construction of such super-reads, we make use of the error profiles that come along with Illumina paired-end reads. These reads are stored in FASTQ files together with the corresponding quality scores (Fig. 3.5a). A quality score for a read position represents a probability that a base was sequenced correctly, i.e. it represents a probability that a particular base is present at a respective position in the FASTQ file (Fig. 3.5b). The complement probability represents a probability that a different base is at the respective position. The probability that different base X is present at a particular position corresponds to one third of the complement probability in our model, which reflects that apart from the correct nucleotide, there are 3 different choices for X. Note that these probabilities are only estimates, as provided by the Illumina sequencing platform.

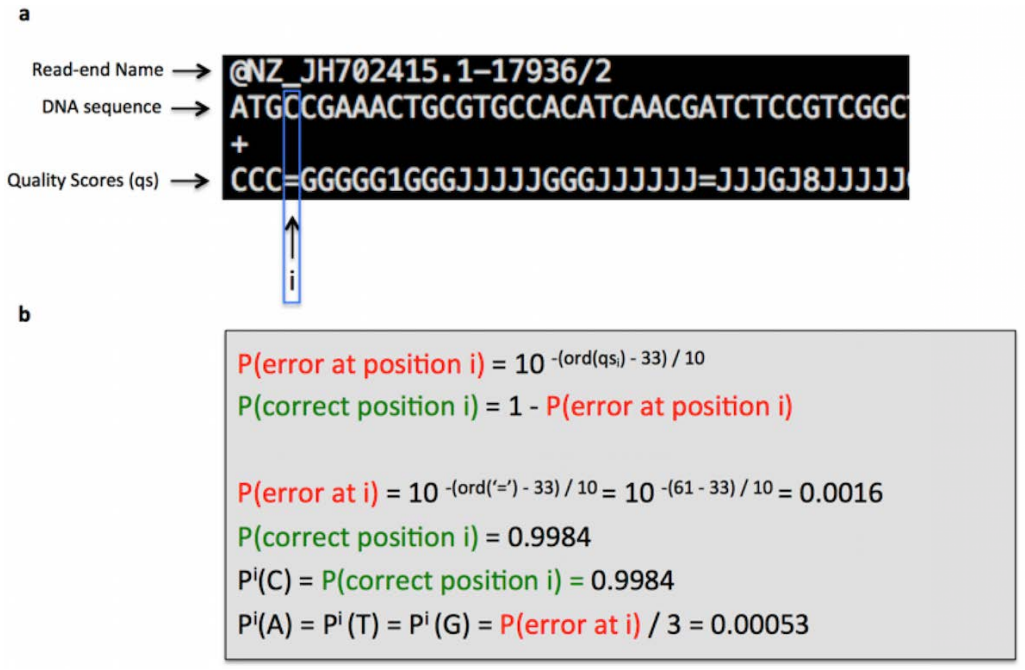


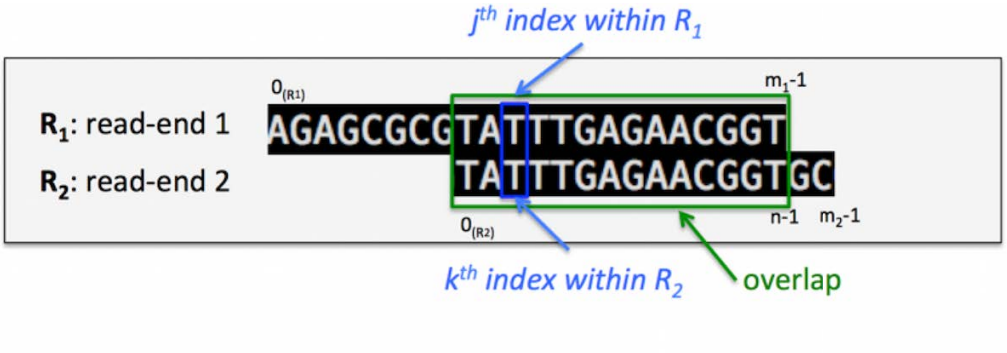
Figure 3.5. FASTQ file data representation.

(Panel a) depicts an example of a read end representation in a FASTQ file. The entry consists of the read end name, the DNA sequence of the respective end of a paired-end read and the quality score for each position of the DNA sequence, which are ASCII coded. (Panel b) explains the meaning of the quality scores. From quality score qs_i at position i, we compute the probability that position i was correctly sequenced, where the ord function assigns an ASCII number to an input ASCII character. Before translating the resulting number $ord(qs_i)$ into the corresponding probability, one has to subtract 33, by convention. The probability that base C is at position i is equal to the probability that position i was sequenced correctly. In our model, the probability of A, T or G being at position i is equal to the probability that position i was sequenced incorrectly divided by three.

In our model, a probability matrix represents a consensus sequence, where each sequence position is represented by a probability distribution over DNA bases $\{A, C, T, G\}$. An example of a probability matrix corresponding to a consensus sequence of two overlapping sequences is depicted in (Fig. 3.6). At a particular position within a consensus sequence, we compute the expected probability of a base as the average probability of the respective base probabilities of the individual reads covering the position. The individual base probabilities are derived from the quality scores (Fig. 3.5). Let R be the set of all read ends that were joined into consensus sequence c and cover position p_c within c. The probability of a base $X \in \{A, C, T, G\}$ being at position p_c within the consensus sequence c is:

$$P^{p_c}(X) = \frac{1}{|R|} \sum_{r \in R} P_r^{p_r}(X)$$

where p_r for a read $r \in R$ is the position within r that corresponds to position p_c within the consensus sequence c. The base with the highest probability in the probability matrix at a particular position is the base of the consensus DNA sequence at the respective position.



	R_1		$R_1 \cap R_2$			R_2	
P(A)	$P_1^0(A)$	...	...	$[\ P_1^j(A) + P_2^k(A)\ ] / \lvert R \rvert$	...	...	$P_2^{m_2-1}(A)$
P(C)	$P_1^0(C)$	...	...	$[\ P_1^j(C) + P_2^k(C)\ ] / \lvert R \rvert$	...	...	$P_2^{m_2-1}(C)$
P(T)	$P_1^0(T)$	...	...	$[\ P_1^j(T) + P_2^k(T)\ ] / \lvert R \rvert$	...	...	$P_2^{m_2-1}(T)$
P(G)	$P_1^0(G)$	...	...	$[\ P_1^j(G) + P_2^k(G)\ ] / \lvert R \rvert$	...	...	$P_2^{m_2-1}(G)$
$\lvert R \rvert$	1	1	2	2	2	1	1

Figure 3.6. Probability matrix example.
In this example of a probability matrix construction, two overlapping read ends are joined into a consensus sequence and represented as a probability matrix. The subscripts of individual probabilities correspond to either read end R_1 or R_2. The superscripts of individual probabilities correspond to the positions within respective read end sequences. The probability arguments are DNA bases $\{A, C, T, G\}$. The $|R|$ values correspond to the coverage, i.e. the number of read ends covering a particular position within the consensus sequence.

3.2.4 Overlap probabilities and error correction

The computation of overlap probabilities of two overlapping sequences is an essential part of the *Snowball* algorithm. Given two overlapping sequences S_1 and S_2, represented by probability matrices (Fig. 3.6), where n is the length of the overlapping region, the overlap probability at position $i \in [0, \dots, n-1]$ is computed as:

$$P^i_{overlap} = \sum_{X \in \{A,C,T,G\}} P^i_1(X) * P^i_2(X)$$

where, $P^i_1(X)$ is the probability that sequence S_1 has base X at overlap position i; probability $P^i_2(X)$ is defined analogously for sequence S_2. The overall overlap probability of S_1 and S_2 is the product of individual position overlap probabilities normalized by overlap length n (Töpfer *et al.*, 2014):

$$P_{overlap} = \sqrt[n]{\prod_{i \in [0,\ldots,n-1]} P^i_{overlap}}$$

As a score that represents the 'expected length' of an overlap, taking into account the individual overlap position probabilities, we compute the expected number of correct positions within the overlap as:

$$Length\ Expected = \sum_{i \in [0,\ldots,n-1]} P^i_{overlap}$$

A single overlap score that enables us to rank different sequence overlaps is computed as a product of the overall overlap probability and the expected overlap length:

$$Score\ Overlap = P_{overlap} * Length\ Expected$$

The overlap score penalizes both overlaps with low overlap probability and short overlaps, since long overlaps with high overlap probability are required. The minimum required expected length of an overlap represents the support for the overlap probability, as the overlap probability is based only on the bases within the overlap, therefore the number of the bases outside of the overlap should remain as small as possible, since we cannot make any statement about the bases outside of the overlap.

In the *Snowball* algorithm, consensus reads are iteratively joined into longer super-reads based on the overlap probabilities, expected overlap lengths and the overlap scores (Section 3.2.5). By default, two sequences S_1 and S_2 can be joined into a consensus sequence if the overall overlap probability is at least 0.8 and the expected length of the overlap is at least $0.5 * min\big(length(S_1), length(S_2)\big)$. The high overall overlap probability ensures that the overlap consists of mostly matching positions, that

there are no mismatching positions with high quality scores and that mismatches are allowed only at positions with low quality scores. For datasets with overall high quality scores, the minimum overlap probability parameter can be increased to 0.9 or 0.95. In the *Snowball* algorithm, when a consensus sequence could be joined with multiple consensus sequences with sufficient overlap probability and expected overlap length, it is joined with the sequence with which it has the highest overlap score.

3.2.5 The *Snowball* algorithm

For each gene domain, the *Snowball* algorithm iteratively joins consensus reads into longer error-corrected super-reads. The input of the algorithm consists of annotated consensus reads of a particular gene domain represented via probability matrices (Sections 3.2.1–3.2.3). The resulting super-reads are output as annotated contigs. Note that the method can be highly parallelized, since the *Snowball* algorithm runs for each gene domain separately.

Consensus reads are first sorted in an increasing order according to the HMM start coordinates, that denote an estimated start position of a consensus read within the multiple sequence alignment of the profile HMM. This layout suggests which pairs of consensus reads are likely to have an overlap (Fig. 3.7), where consensus reads that are next to each other are likely to have longer overlaps than other pairs of consensus reads. As a starting point of the algorithm, we choose a consensus read with the largest sum of overlap lengths with other consensus reads and put it into the *working set*. The reason for this choice is that such a consensus read is within the highest coverage of the alignment corresponding to the respective profile HMM, where highly covered regions are likely to be covered by reads originating from similar but distinct genomes. Therefore, the chosen consensus read is very likely to overlap with consensus reads originating from distinct gene variants, which will help to resolve these gene variants early in the algorithm.

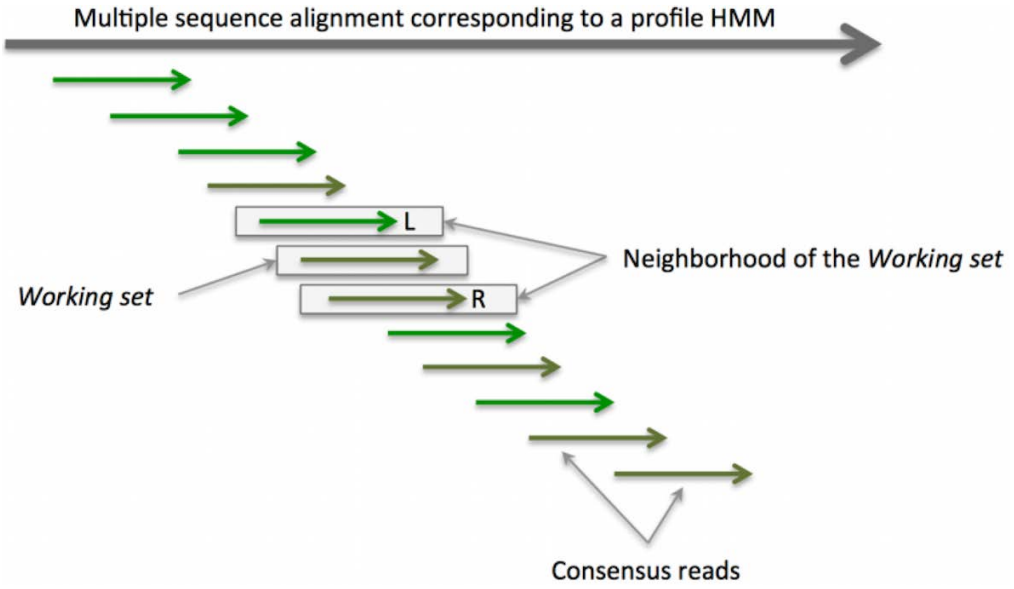


Figure 3.7. Initial layout of consensus reads.

Consensus reads sorted according to the HMM start coordinates. In the neighbourhood of the consensus read, that is in the *working set*, there are two closest consensus reads, one on the left (*L*) and one on the right (*R*).

The main idea of the algorithm is that it iteratively tries to extend consensus sequences from the *working set* into longer consensus sequences by joining them with consensus reads that are in their neighbourhood, considering the consensus read layout (Fig. 3.7). In one iteration, first a consensus read from the neighbourhood (i.e. *L* or *R*) is joined with one of the consensus sequences from the *working set.* Second, two consensus reads (i.e. *L* and *R*) that are in the neighbourhood of the *working set* are added to the *working set* or both consensus reads from the neighbourhood of the *working set* (i.e. *L* and *R*) are joined into a consensus sequence and added to the *working set*. A consensus read and a consensus sequence (or two consensus reads) are joined only if they have a sufficient overlap as defined in the Section 3.2.4. If there is more than one overlap of a consensus read from the neighbourhood (i.e. *L* or *R*) and a consensus sequence from the *working set*, given that also the overlap between *L* and *R*, is sufficient, the pair that has the highest overlap score is chosen. If there is no sufficient overlap between a consensus sequence from the *working set* and a consensus read *L* or *R* in the neighbourhood and the overlap between *L* and *R* is also not sufficient, both consensus reads are added to the *working set* as they are likely to originate from distinct gene variants than the gene variants already represented in the *working set*.

Pseudo code of the algorithm:

(1) Input: a list of consensus reads of a particular gene domain.

(2) Sort the input list according to the HMM start coordinates in the increasing order.

(3) Find a consensus read representing the starting point – as told above, a consensus read with the largest sum of overlap lengths with other consensus reads – and add it into the *working set*.

(4) The neighbourhood of the *working set* consist of at most two consensus reads, one that is the closest on the left (*L*) and one that is the closest on the right (*R*) of the *working set*.

(5) For each consensus sequence *S* from the *working set* and for each pair (*L*, *S*) and (*S*, *R*), and for (*L*, *R*), compute:

   a. overlap probability

   b. expected overlap length

   c. overlap score

(6) If there is a sufficient overlap between at least one pair (*L*, *S*), (*S*, *R*) or (*L*, *R*), the pair with the highest overlap score is chosen, as defined in the Section 3.2.4. Let (*L*, *S*) be the pair with the highest overlap. Remove *S* from the *working set*. Join (*L*, *S*) into a consensus sequence (i.e. a super-read), as defined in the Section 3.2.3 and add it into the *working set*. Redefine *L*, as the first consensus read on the left of *L*. If (*S*, *R*) is the pair with the highest score, proceed analogously. If (*L*, *R*) is the pair with the highest score, join (*L*, *R*) into a consensus sequence (i.e. a super-read) and add it into the *working set*. Redefine *L* and *R* analogously.

(7) If there is no sufficient overlap found in step (6), add *L* and *R* into the *working set* and redefine *L* and *R* in the same way as in (6).

(8) If the neighbourhood is not empty, i.e. *L* or *R* was redefined at step (6) or (7), go to step (5). If *L* or *R* cannot be redefined, it is not considered in the next steps of the algorithm.

(9) Output super-reads as annotated contigs.

In the algorithm, a consensus sequence is represented via a probability matrix as described in the Section 3.2.3. Mismatching bases within a sufficient overlap most likely represent a substitution error, where one of the bases has a relatively low quality score, thus, the base with a higher quality score corrects such a substitution error.

Substitutions representing genuine strain variation are represented by overlap positions with different bases with relatively high quality scores. Therefore, such overlaps of consensus reads representing different strains almost never pass the minimum required overlap probability threshold. Consensus reads containing insertion or deletion errors have very low overlap probabilities with other consensus reads or super-reads and are therefore unlikely to be joined into longer consensus sequences. Thus, super-read positions with coverage of at least two are mostly error-corrected in terms of insertion and deletion sequencing errors.

3.3 Results

We evaluated *Snowball* using 21 simulated datasets, each containing 3–9 closely related *E. coli* strains and one simulated dataset containing ten novel recently published *Rhizobia* strains (Bai *et al.*, 2015) (Section 3.3.4). We recall that good performance on different strains implies good performance on different species, which is why we put the emphasis on distinguishing between closely related strains in our experiments. Thereby, our aim was to answer the following questions: Were the contigs assembled correctly? How long are the resulting contigs? Did the assembly recover the reference strain sequences from which the input paired-end reads were generated? As a reference method, we used the *SAT* assembler (Zhang *et al.*, 2014), because this is to the best of our knowledge the only currently available gene assembler of gene domains of interest for metagenomic data that does not require closely related reference genomes to be available. In our experiments, we observed that *Snowball* was faster than *SAT*. The runtime of *Snowball* was limited by the runtime of the *HMMER 3* software, i.e. our method spent most of the runtime in this step (Section 3.2.2).

3.3.1 Per-base error

We computed the per-base error for all assembled contigs of all simulated datasets (Fig. 3.8). For each contig, we determined the reference strain sequence and coordinates of a particular contig sequence within a respective reference sequence from which it originates. The per-base error is defined as the percentage of bases that differ between a contig sequence and the respective sub-sequence of the reference sequence, i.e. it corresponds to the Hamming distance between the two sequences, normalized by the length of the overlap. Note, that closely related strains share large sequence regions; therefore, a contig can be well mapped onto several reference sequences of distinct

strains. In this case, a reference sequence, onto which a contig maps with the lowest Hamming distance, is considered to be the reference strain sequence from which it originates. If a contig maps onto several sequences of different strains, with exactly the same error, we consider it to originate from all these strains. The coverage of a contig position is equal to the number of read ends covering a respective position. In the *Snowball* algorithm, we keep track of all consensus reads that a contig consists of. For the *SAT* assembler, we have used *BWA* (H. Li and Durbin, 2009) to map consensus reads onto the contigs. We computed the per-base error for each coverage $[3, \dots, 30]$ separately. Low-coverage positions typically have a higher per-base error, as there is not enough information available to correct sequencing errors. This is most pronounced at positions with coverage one, where the per-base error corresponds to the substitution error of a respective sequencing platform (~2.37% for our simulated datasets). At positions with higher coverage, the error-correction mechanism built into the *Snowball* algorithm yields very low (~0.02%) per-base error (Fig. 3.8). For the *SAT* assembler, contig positions with high coverage correspond to consensus sequences containing reads of several strains, which yields a relatively high per-base error (Fig. 3.8). This shows that the error-correction incorporated in the *Snowball* algorithm is indispensable for the assembly of closely related strains.

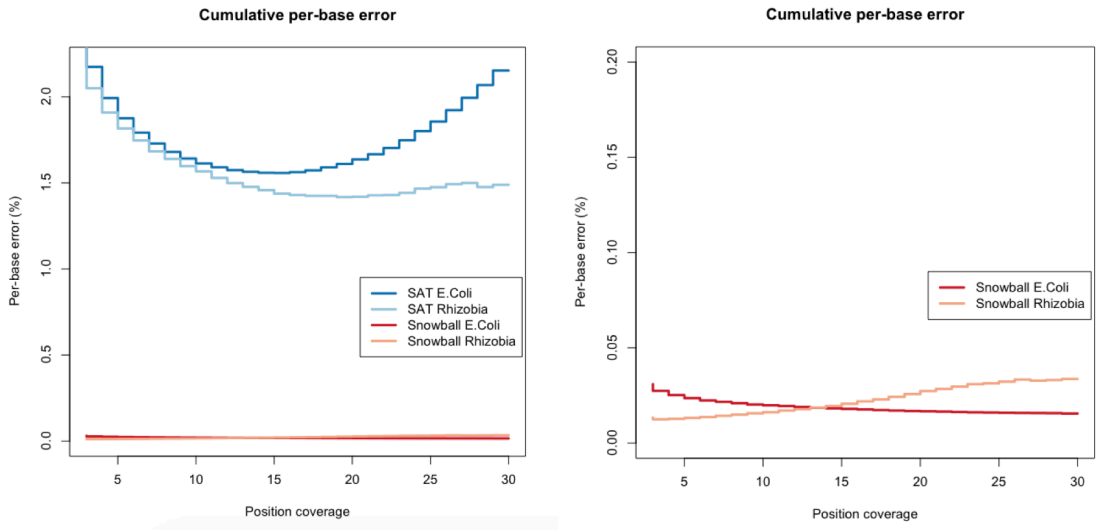


Figure 3.8. Cumulative per-base error.

Cumulative per-base error for the *Snowball* and *SAT* assemblers. We computed the per-base error in a cumulative way, i.e. for $X \in [3, \dots, 30]$ (on the horizontal x-axes), Y (on the vertical y-axes) is equal to the per-base error at contig positions with coverage greater or equal to X.

3.3.2 Relative contig length

We computed the average number of assembled contigs and the average cumulative length of all contigs (in Kb) per strain (Fig. 3.9). As the assembled contigs should cover the full length of the respective gene sequences sufficiently well, we aligned each contig to the respective profile HMM and computed the fraction of the model (i.e. the corresponding multiple sequence alignment) it covers. For each contig, this gave us an estimate of its relative length with respect to the particular profile HMM. We used this information to compute the results, e.g. using only longer contigs covering at least 50% (60%, 70%, etc.) of respective profile HMMs. This analysis showed that *Snowball* produced substantially more, longer contigs than the *SAT* assembler.

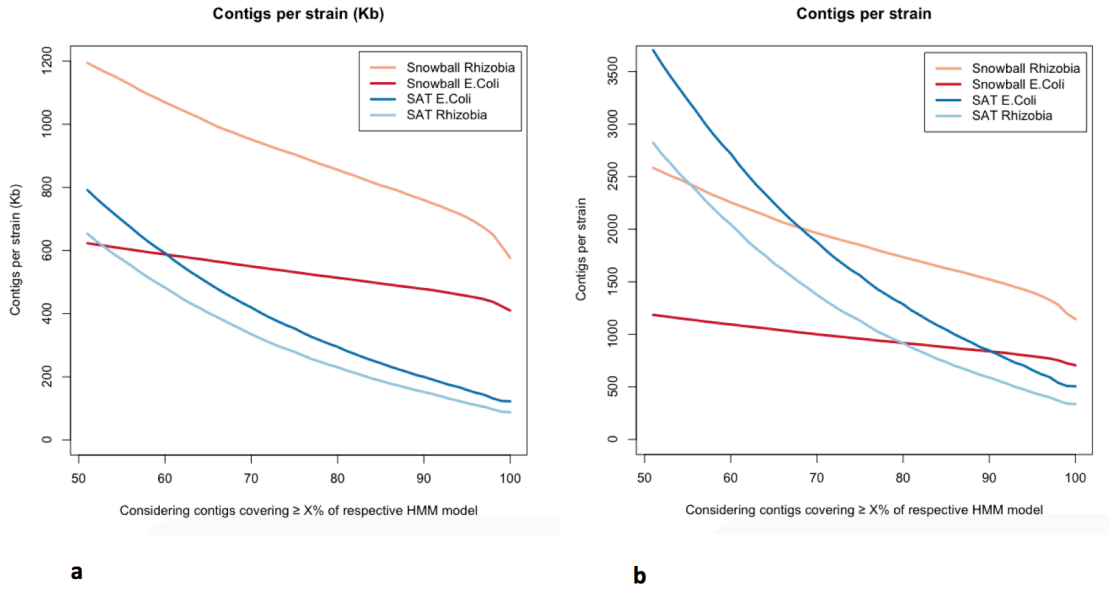


Figure 3.9. Contigs per strain.

Cumulative average contig length per strain, considering only contigs covering X% of respective profile HMMs (panel *a*). Average number of contigs per strain, considering only contigs covering $\geq X$% of respective profile HMMs (panel *b*). Here, the variable X corresponds to the values on the (horizontal) *x*-axes of the graphs.

3.3.3 Reference coverage

We computed which parts of the reference strain sequences, from which the input reads were generated, were recovered by the assembled contigs, per strain on average (Fig. 3.10). As explained in the Section 3.3.1, assembled contigs may map onto one or more reference strain sequences with the same minimum Hamming distance. We considered a

contig to cover all the reference strain sequences, onto which it can be mapped with exactly the same minimum per-base error. Positions of reference sequences that are covered by at least one contig are denoted as covered positions. For each strain, we computed the number and percentage of the covered positions. Moreover, as explained in the Section 3.3.2, we computed these measures for contigs covering $\geq X\%$ of respective profile HMMs (where the variable X corresponds to the values on the x-axes of the graphs). The overall relatively low coverage of the reference sequences can be explained by low sequencing coverage of some of the reference strain sequences (Supplementary Table S1–S8). Also, as we only assemble coding sequences of the reference strain sequences, for which we have used profile HMMs as the input, regions of the reference strain sequences that are not covered by the profile HMMs remain unassembled. Nevertheless, this analysis showed that *Snowball* recovered substantially more reference strain sequences than the *SAT* assembler.

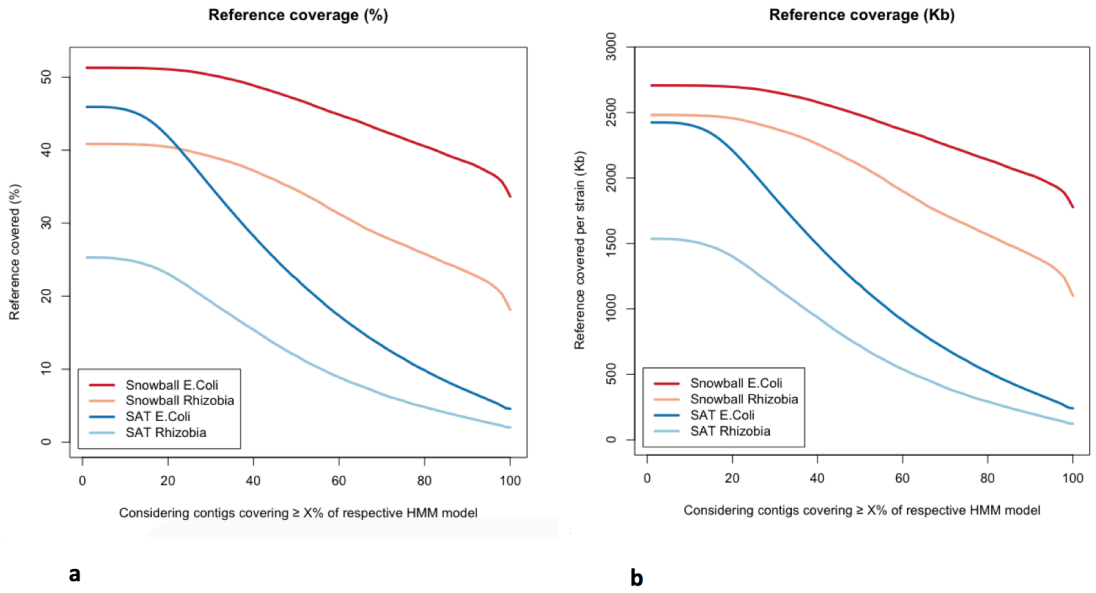


Figure 3.10. Coverage of the reference strain sequences.

Percentage of the recovered reference strains, per strain on average, considering only contigs covering $\geq X\%$ of respective profile HMMs (panel *a*). Corresponding absolute values (Kb) are depicted in (panel *b*). The variable X corresponds to the values on the x-axes.

3.3.4 Simulated datasets details

We have based our evaluation on 22 simulated datasets (Table 3.1, Supplementary Table S1–S8). The strain abundances correspond to randomly drawn numbers from the log-normal distribution (mean = 1, standard deviation = 2), where the numbers were

limited to interval $[1, \ldots, 50]$, to avoid both data explosion and extremely low strain abundances. The *ART* (Huang *et al.*, 2012) read simulator (version 2.3.6) was employed to generate Illumina HiSeq 2500 paired-end reads (read length = 150 bp, mean insert size = 225, standard deviation = 23), where the strain coverage used for the read simulation also corresponds to the strain abundance. The abundance of a particular strain thus informs us with which coverage the strain genome within a simulated dataset was sequenced. We used the default *ART* Illumina HiSeq 2500 empirical error profile, which yields reads with ~2.37% substitution error. For each dataset, we provide per-dataset results (Table 3.1, Sections 3.3.1–3.3.3) that show that *Snowball* performed substantially better than the *SAT* assembler for all simulated datasets.

Table 3.1. Overview of simulated datasets.

Dataset	Strains per dataset	Per-base error (%) at position coverage ≥ 5[a]		Contig length (Kb) 75% HMM model[b]		Reference coverage 75% HMM model (%)[c]	
		Snowball	SAT	Snowball	SAT	Snowball	SAT
1	3	0.019	1.613	913	229	41.3	7.5
2		0.035	1.823	1080	628	44.4	15.1
3		0.006	1.603	865	186	43.0	6.7
4	4	0.036	1.666	740	306	43.1	10.7
5		0.011	1.813	691	253	42.6	9.7
6		0.007	1.648	700	303	45.5	11.2
7	5	0.012	1.809	614	408	44.9	13.5
8		0.012	1.791	622	393	44.8	13.5
9		0.022	2.064	665	411	40.9	12.6
10	6	0.022	1.853	518	378	42.1	11.8
11		0.045	1.822	557	308	39.0	10.7
12		0.033	2.009	571	407	40.2	12.4
13	7	0.028	1.861	447	316	42.6	11.7
14		0.041	1.866	496	293	38.9	10.9
15		0.018	2.034	488	367	41.7	12.0
16	8	0.017	2.152	408	443	44.6	12.7
17		0.030	1.869	428	294	38.3	10.5
18		0.038	2.227	453	440	39.3	11.6
19	9	0.019	1.884	349	265	40.9	9.7
20		0.014	2.035	360	314	40.4	10.7
21		0.044	2.270	424	430	42.2	13.8
22	10	0.013	1.909	905	279	27.0	5.7

[a]Per-base error (%) at contig positions with coverage ≥ 5 (Fig. 3.8).

[b]Cumulative contig length (Kb) at $X = 75$ of (Fig. 3.9a).

[c]Percentage of recovered data at $X = 75$ of (Fig. 3.10a).

Datasets 1-21 consist of *E. coli* strains (Supplementary Table S1–S7).

Dataset 22 consists of *Rhizobia* strains (Supplementary Table S8).

3.4 Conclusions

We describe *Snowball*, a novel strain aware gene assembler for reconstruction of gene domains of interest from shotgun metagenomic data of microbial communities. *Snowball* performs gene assembly of individual gene domains based on read overlaps and error-correction using read quality scores at the same time, which result in very low per-base error rates. Our method uses profile HMMs to guide the assembly. Nonetheless, it does not require closely related reference genomes of the studied species to be available. We have assessed the performance of *Snowball* using 21 simulated datasets, each containing 3–9 closely related *E. coli* strains and one simulated dataset containing ten recently published *Rhizobia* strains (Bai *et al.*, 2015), which demonstrates the capability of the *Snowball* assembler to assemble novel strains. We have compared our *Snowball* assembler to the *SAT* assembler, which, to our knowledge, establishes the current state of the art in gene assembly. The results showed that *Snowball* had substantially lower per-base error, assembled more, longer contigs and recovered more data from the input paired-end reads. We have shown that the incorporation of the error-correction mechanism is indispensable for the assemblies of closely related strains. To our knowledge, *Snowball* is the first strain aware gene assembler that does not require closely related reference genomes of the studied species to be available. The assembly of closely related strains is still a challenging task for most of the current assemblers, including the *SAT* assembler. We believe that our tool will be valuable for studying species evolution (e.g. genes under selection) and strain or gene diversity (e.g. virulence genes). *Snowball* is implemented in Python, runs on a user-defined number of processor cores in parallel, runs on a standard laptop and can be easily installed under Linux or OS X.

3.5 Supplementary material

The supplementary material is available in Chapter 7 and at *Bioinformatics* online: http://bioinformatics.oxfordjournals.org/content/32/17/i649.full

4 *PhyloPythiaS+*: a self-training method for the rapid reconstruction of low-ranking taxonomic bins from metagenomes

Status	**published**
Journal	*PeerJ* (Impact factor 2.2)
Citation	I. Gregor, J. Dröge, M. Schirmer, C. Quince, and A. C. McHardy (2016). "*PhyloPythiaS+*: a self-training method for the rapid reconstruction of low-ranking taxonomic bins from metagenomes." *PeerJ* 4:e1603 doi:10.7717/peerj.1603
URL	https://doi.org/10.7717/peerj.1603
Own contribution	75% Designed and performed the experiments (with co-authors) Analyzed the data (with co-authors) Wrote the manuscript (with co-authors)

Abstract

Background. Metagenomics is an approach for characterizing environmental microbial communities *in situ,* it allows their functional and taxonomic characterization and to recover sequences from uncultured taxa. This is often achieved by a combination of sequence assembly and binning, where sequences are grouped into 'bins' representing taxa of the underlying microbial community. Assignment to low-ranking taxonomic bins is an important challenge for binning methods as is scalability to Gb-sized datasets generated with deep sequencing techniques. One of the best available methods for species bins recovery from deep-branching phyla is the expert-trained *PhyloPythiaS* package, where a human expert decides on the taxa to incorporate in the model and identifies 'training' sequences based on marker genes directly from the sample. Due to the manual effort involved, this approach does not scale to multiple metagenome samples and requires substantial expertise, which researchers who are new to the area do not have.

Results. We have developed *PhyloPythiaS+*, a successor to our *PhyloPythia(S)* software. The new (+) component performs the work previously done by the human expert. *PhyloPythiaS+* also includes a new *k*-mer counting algorithm, which accelerated the simultaneous counting of 4-6-mers used for taxonomic binning 100-fold and reduced the overall execution time of the software by a factor of three. Our software allows to analyze Gb-sized metagenomes with inexpensive hardware, and to recover species or genera-level bins with low error rates in a fully automated fashion. *PhyloPythiaS+* was compared to *MEGAN*, *taxator-tk*, *Kraken* and the generic *PhyloPythiaS* model. The results showed that *PhyloPythiaS+* performs especially well for samples originating from novel environments in comparison to the other methods.

Availability. *PhyloPythiaS+* in a virtual machine is available for installation under Windows, Unix systems or OS X on: https://github.com/algbioi/ppsp/wiki.

4.1 Introduction

Metagenomics is the functional or sequence-based analysis of microbial DNA isolated directly from a microbial community of interest (Riesenfeld *et al.*, 2004; Kunin *et al.*, 2008). As the cultivation conditions for most microorganisms are unknown or too complex to reproduce in the laboratory (Hugenholtz, 2002), random shotgun and amplicon-sequencing based metagenome studies have led to substantial advances in our understanding of the structure and functions of microbial communities within the last decade (Kalyuzhnaya *et al.*, 2008; Turnbaugh *et al.*, 2010; Hess *et al.*, 2011; Pope, Smith, *et al.*, 2011; Zarowiecki, 2012; Schloissnig *et al.*, 2013; Blaser *et al.*, 2013). The taxonomic classification or 'binning' of metagenome samples is often performed after sequence assembly (Peng *et al.*, 2011; Laserson *et al.*, 2011; Boisvert *et al.*, 2012; Namiki *et al.*, 2012; Pell *et al.*, 2012). This is a computationally demanding task, which for metagenome samples results in a mixture of sequence fragments of varying lengths, originating from the different microbial community members. A taxonomic binning defines 'bins' of sequence fragments that were assigned the same taxonomic identifier, representing draft genomes or pan-genomes of the different microbial community members. Taxonomic binning methods use sequence homology, sequence composition and similarities of contigs in read coverage or gene counts, see (Dröge and McHardy, 2012) for a recent review. The subsequent analysis of these bins allows characterizing the functional and metabolic potential for individual taxa. For instance, in a collaboration with Mark Morrison's group, a functional and metabolic analysis of a draft genome recovered by taxonomic binning from the gut of the Australian Tammar Wallaby metagenome led to the isolation and subsequent characterization of a new and previously uncultivated bacterium (Pope, Smith, *et al.*, 2011). Different from binning methods, taxonomic profiling tools (Wu and Eisen, 2008; Stark *et al.*, 2009; Liu *et al.*, 2011; Meinicke *et al.*, 2011; Wu and Scott, 2012; Segata *et al.*, 2012; Sunagawa *et al.*, 2013; Silva *et al.*, 2013) return a taxonomic profile for a metagenome sample to represent the taxonomic composition of the underlying sampled community.

Composition-based binning methods assign metagenome sequences based on their *k*-mer signature, which is derived from the counts of short oligomers (*k*-mers) for a sequence (Karlin and Burge, 1995; Deschavanne *et al.*, 1999). Our previously developed *PhyloPythia(S)* (*PPS*) (McHardy *et al.*, 2007; Patil, Haider, *et al.*, 2011) software uses this information in combination with a structured output support vector machine framework for taxonomic classification. Composition-based signatures are global

genomic properties, which can be estimated from any sufficiently sized sequence sample for a taxon; e.g., for *PP(S)*, 100 kb of reference sequences for a taxon are sufficient for accurate assignment, also for low ranking taxa. Thus, no complete genome sequences of related organisms are required for assignment, which is often a limiting factor for the homology-based methods. Composition-based methods are very fast, with classification runtimes increasing linearly with the size of the sequence sample, whereas the runtime of alignment-based methods is proportional to the product of the reference collection size and the sequence sample size. As the current sequencing technologies produce Gb-sized metagenome samples (Metzker, 2010; Loman *et al.*, 2012), scalability and computational efficiency are becoming increasingly important for computational metagenomic methods. Therefore, we have developed a fully automated taxonomic binning software, that can rapidly process large metagenome samples. *PhyloPythiaS+* (*PPS+*) is the successor to our previously described *PPS* software and improves on it in several important ways. We provide an automated marker-gene based framework for design and creation of sample-derived structured output support vector machine models, which allows the generation of accurate sample-derived models without user intervention or expert knowledge. *PPS+* is the first tool that combines taxonomic profiling and subsequent taxonomic composition based binning of the whole metagenome sample, which is particularly valuable for the draft genome reconstruction of taxa from deep-branching phyla. By implementation of a faster *k*-mer counting algorithm, we substantially increased its throughput to 0.5 Gb/h. *PPS+* is distributed in a virtual machine which facilitates installation under all common operating systems and runs on inexpensive hardware available to most users.

4.2 Methods

The classification of a shotgun metagenome sequence sample with *PPS+* proceeds in two phases (Fig. 4.1): In the first phase, the newly developed (+) component identifies sample-derived training sequences and the taxa to be modeled by searching for copies of 34 ubiquitous taxonomic marker genes in the metagenome sample. The marker gene analysis results in taxonomic assignments for a small fraction of the sample. Based on the taxa abundance profile derived from these assignments and the sequences available in the reference sequence collections, our method determines which taxa will be modeled and which are the sample-derived data that will be used for training *PPS*.

The second phase is the composition-based taxonomic assignment of the entire metagenome sample using *PPS* models trained using the data generated in the first phase. *PPS* models can be reused to classify further metagenome samples, e.g., additional samples from the same community.

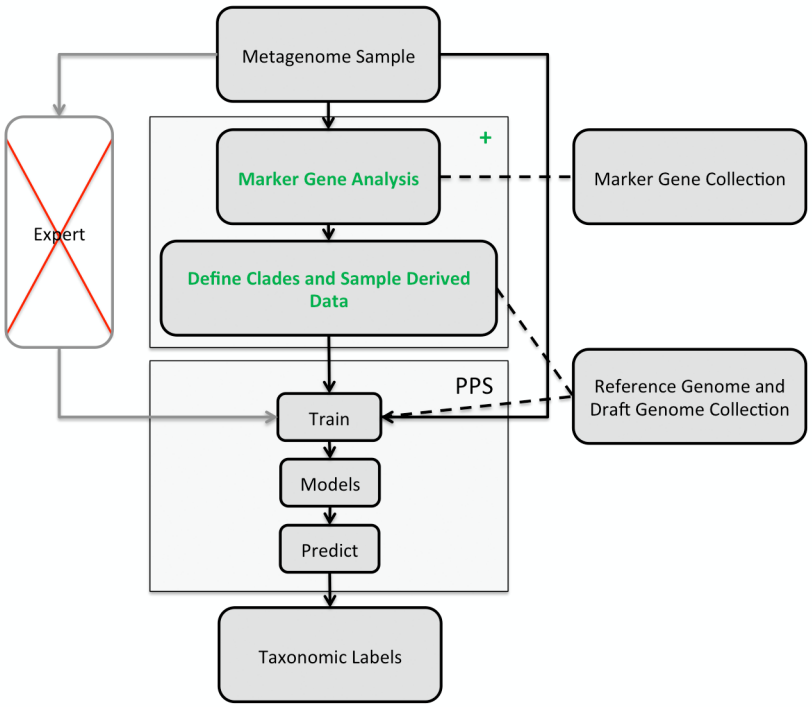


Figure 4.1. Illustration of the *PhyloPythiaS+* workflow.
The recommended use of *PPS* is that a human expert specifies the taxa to incorporate in a composition-based taxonomic metagenome classifier and identifies the relevant 'training' sequences based on marker genes directly from the sample. The inclusion of contigs originating directly from members of the microbial community, as 'training' sequences, is very important for achieving good classification accuracy, as many members of microbial communities are underrepresented in public sequence collections. In *PPS+*, the step of deciding which taxa to include in the model and defining suitable 'training' sequences was automated in the + component, based on marker genes, genome and draft genome sequence collections. The data generated by the + component are then used to build the *PPS* models, that are subsequently used to generate the taxonomic binning of the entire metagenome sequence sample.

4.2.1 PhyloPythiaS

Assignment with *PPS* proceeds in two steps: In the training step, an ensemble of structured output Support Vector Machines (SVMs) (Joachims *et al.*, 2009) for the specified part of the NCBI taxonomy, defined by the taxa being modeled, are trained using the sample-derived training sequences and additional data for these taxa from a

customized reference collection of sequenced genomes and draft genomes (Supplemental Text S1, Section 3.3). The list of modeled taxa and sample-derived data are generated with the + component of *PPS+*. The list of taxa restricts the taxonomic output space that is modeled, i.e., a sequence from a metagenome sample will be assigned to a leaf node taxon or a corresponding higher-ranking taxon of the learned taxonomy.

In the prediction step, the *PPS* model ensemble identifies the taxon which best matches a query sequence in terms of its *k*-mer profile and assigns to it the respective taxonomic identifier. By default, sequences of 1 kb or more are classified (*PPS+* configuration parameter: *minSeqLen*).

4.2.2 The + component of *PhyloPythiaS+*

The input for the + component of *PhyloPythiaS+* is the metagenome sample. This step returns a list of clades and sample-derived data for the subsequent *PPS* training. The + component performs the following steps:

(1) *Marker gene identification*: DNA sequences from the sample are translated in all six reading frames (i.e., also considering reverse complement sequences) to protein sequences. In both the translated and untranslated sequences, regions with similarity to the DNA or protein Hidden Markov model (HMM) profiles of 34 taxonomically informative marker genes (Wu and Eisen, 2008; Stark *et al.*, 2009; Liu *et al.*, 2011; Wu and Scott, 2012; Segata *et al.*, 2012; Sunagawa *et al.*, 2013) are identified (Supplemental Text S1, Sections 3.3 and 6.1). The corresponding DNA marker gene sequences from these regions are used for further analysis.

(2) *Taxonomic marker gene assignment*: The marker gene sequences are assigned a taxonomic identifier using the composition-based Naïve Bayes classifier (Schloss *et al.*, 2009) (Supplemental Text S1, Section 6.2).

(3) *Taxonomic sequence assignment*: If a sequence contains multiple marker genes, multiple taxonomic identifiers are identified in Step 2. Then the highest bootstrap confidence score (*hcs*) returned by the Naïve Bayes classifier (NBC) for one of the markers on the fragment is identified. We use all marker gene assignments with confidence scores larger than *hcs * (1 – candidatePlTopPercentThreshold)*. The default setting for the configuration

parameter *candidatePlTopPercentThreshold* is 0.1. From the set of taxonomic identifiers, the lowest taxon *t* is identified for which all other assignments are either to the same taxon *t* or defined at higher-ranking parental taxa of *t.* Taxon *t* is consequently used for the overall fragment assignment. The confidence score for the fragment is set to the smallest confidence score for the set of retained marker gene assignments.

(4) *(Optional: Taxonomic scaffold assignment):* Scaffolding information (i.e., the mapping of contigs to scaffolds) can be used to obtain more training data for the relevant taxa. Assembled contigs can be grouped into scaffolds based on the paired-end information after the assembly. As all contigs of a particular scaffold originate from the same strain, all contigs of the respective scaffold should have the same taxonomic label. Here, we make use of this scaffolding information, such that unassigned contigs of a particular scaffold can be assigned based on the assigned contigs of the respective scaffold. In the first step, the taxonomic identifiers of all assigned contigs for a scaffold are corrected as follows: Let us consider that *n* taxonomically assigned contigs of a scaffold are placed along a common path from the root *r* down to a low-ranking clade *lc* in the reference taxonomy. The unassigned contigs of a scaffold are not among these *n* contigs. To obtain a consistent assignment for all the contigs of a scaffold and to correct for 'outlier' contig assignments to low ranking taxa, contigs are reassigned according to the following: All *n* assigned contigs of the respective scaffold are reassigned to the lowest taxon *c*, which lies on the path from *r* to *lc*, where *c* is chosen such that at least *(agThreshold * n)* of the contigs are assigned on the path from *c* to *lc*. In the second step, unassigned contigs are assigned to the same taxon *c*, if a sufficient number of contigs have already been assigned. Let us denote the sum of all contig lengths for a scaffold as *l* and the sum of all assigned contig lengths of the respective scaffold as *al*. If $al/l \geq assignedPartThreshold$, then the unassigned contigs of a scaffold are also assigned to clade *c* (see the configuration parameters: *placeContigsFromTheSameScaffold = True*, *agThreshold = 0.3*, *assignedPartThreshold = 0.5*).

(5) *Assignment path truncation*: Contigs assigned to a lower-ranking taxon than the specified lowest rank are reassigned to the parental taxon of this lowest rank (configuration parameter: *rankIdCut*).

(6) *Taxa selection for model specification*: Any taxon for which at least 100 kb of sample-derived data have been identified can be modeled. Furthermore, species can be modeled if at least 300 kb of reference sequences are available in the

reference sequence database, and higher-ranking taxa can be modeled if data for at least three distinct species with this requirement (>300 kb per species) are available. Contigs assigned to taxa for which there are fewer data are subsequently assigned to higher taxonomic ranks for which sufficient data are available to allow their use as sample-derived training data (configuration parameters: *minGenomesWgs = 3 or 1*, *minBpPerSpecies = 300,000*, *minBpToModel = 100,000*).

(7) *Abundant taxa selection*: To reduce the number of taxa to the most relevant ones, the least abundant taxon is removed iteratively. This is defined as the taxon to which the minimum number of bp is assigned. Sequences assigned to this taxon are reassigned to the closest defined taxon at a parental rank. The algorithm ends when the number of leaf taxa is less than or equal to the maximum number of taxa to be modeled (configuration parameter: *maxLeafClades = 50*; this can be set realistically up to 800).

Balancing training data: The part of the taxonomy that will be modeled with *PPS* is defined by the taxa identified in the previous step. It has leaf nodes at different ranks above the specified rank cut-off, and internal nodes. Only leaf node taxa and sample-derived training data assigned to leaf node taxa in the preceding steps are specified as input for *PPS* training. To balance the training data across clades, a maximum of 400 kb of sample-derived training data are selected for each leaf node taxon (configuration parameter: *maxSSDfileSize*). For this selection, contigs are used in order of decreasing confidence values and then in order of decreasing length. The balancing of training data can be switched off by setting the configuration parameter (*maxSSDfileSize*) to a large number.

4.2.3 Simultaneous counting of multiple short *k*-mers

We provide *PPS+* with a new custom *k*-mer counting algorithm that is based on the Rabin Karp string matching algorithm (Karp and Rabin, 1987). The algorithm is highly optimized to count occurrences of short DNA sequences. It is very fast, as it is memory efficient, because it does not need any large helper data structure similar to suffix trees. It explores the locality of reference, uses very fast bit shift operations and is efficiently implemented in C. Its complexity is *O(n),* where *n* is the length of the DNA sequence that is being considered. It enumerates *k*-mers up to hundred times faster than when using suffix trees that were employed in *PPS*. This made *PPS+* overall up to 3x faster than *PPS*.

Because the algorithm allows to simultaneously enumerate k-mers of consecutive lengths in one run, it is at least 2–7x faster than the state-of-the-art software *Jellyfish* (Marcais and Kingsford, 2011) and 11x faster than *KAnalyze* (Audano and Vannberg, 2014) in the scenario used in *PPS+*, i.e., when calculating k-mers of length 4, 5, and 6 for every sequence (Table S1, Supplemental Text S1, Section 2). We also found that the state-of-the-art k-mer counting methods *KMC 2* (Deorowicz *et al.*, 2015) and *Turtle* (Roy *et al.*, 2014) are not applicable to our problem setting, as *KMC 2* can count only k-mers ≥ 10 and *Turtle* is prohibitively slow for sequences ≥ 16 kb.

4.2.3.1 Algorithm description

Let us assume that we are given an array a, which represents a DNA sequence of length n where all letters are encoded as numbers 0, 1, 2, 3 (where $A \sim 0$, $T \sim 1$, $G \sim 2$, $C \sim 3$) and let $a_0, \dots, a_{n-1}$ denote the respective entries. We would like to count the occurrences of all k-mers of length k and store the counts in an array c of length 4^k, which is initialized by zeros. Each k-mer maps to a unique index in the array c. The index of the first k-mer in our sequence is calculated according to:

$$\text{index}_0 = a_0 * 4^{k-1} + a_1 * 4^{k-2} + \cdots + a_{k-2} * 4^1 + a_{k-1} * 4^0$$

The index of the $(i+1)$th k-mer of the sequence is computed from the ith index as:

$$\text{index}_{i+1} = (\text{index}_i - a_i * 4^{k-1}) * 4 + a_{i+k} * 4^0$$

When an index is identified, the corresponding k-mer count at this index position in array c is incremented by one. For instance, the DNA sequence $ATGCATG$ is encoded in array a as $[0, 1, 2, 3, 0, 1, 2]$. For $k = 2$, we would add two counts for the k-mer AT in array c at the index position $0 * 4 + 1 = 1$, two counts for TG at the index position $1 * 4 + 2 = 6$, one count for GC at the index position $2 * 4 + 3 = 11$ and one count for CA at index position $3 * 4 + 0 = 12$. The multiplication operation $X * 4^m$ can be computed using the bit shift operation $X \ll (2 * m)$, which is usually much faster than multiplication.

4.2.3.2 Counting k-mers of different lengths at once

If $index_i$ is the index of the ith k-mer of length k, the index of the ith $(k-j)$-mer (of length $k-j$) can be simultaneously computed using the bit shift operation as $index_i \gg (2 * j)$ (for j $\in$ [1, … , k-1]) and the corresponding counter at the computed index of a respective counter array of length 4^{k-j} is incremented. The end of a DNA sequence can be handled by adding several non-DNA characters to its end.

4.3 Results

We evaluated *PPS+* by comparing it to homology-based methods (*MEGAN4*, *taxator-tk*) (Huson *et al.*, 2011; Dröge *et al.*, 2014), the fast taxonomic binning program *Kraken* (Wood and Salzberg, 2014), the composition-based method *PhyloPythia* trained under expert guidance (a recommended but time-consuming procedure) and to a generic *PPS* model using default settings (Supplemental Text S1, Sections 3.5–3.8). For a performance comparison of *PPS* to methods with prohibitive runtimes for large datasets, such as *PhymmBL* (Brady and Salzberg, 2011) and *CARMA3* (Gerlach and Stoye, 2011), and the web-based tool *NBC* (Rosen *et al.*, 2011) see (Patil, Haider, *et al.*, 2011; Patil, Roune, *et al.*, 2011; Dröge *et al.*, 2014), as *PPS* has already been compared to these methods with favorable outcomes. For a comparison with 'taxonomy-free' binning software *CLARK* (Ounit *et al.*, 2015) see (Supplemental Text S1, Section 7). We did not compare *PPS+* to profiling tools such as (Liu *et al.*, 2011), as *PPS+* is a binning method that assigns a taxonomic label to each input sequence. As benchmark datasets, we created two simulated datasets, one with a uniform (137 Mb) and one with a log-normal (66 Mb) distribution of 47 community members (Supplemental Text S1, Section 3.1, Datasets S1 and S2). We also used two real datasets, a metagenome sample from the guts of two obese human twins (255 Mb) (Turnbaugh *et al.*, 2010) and a cow rumen metagenome sample (319 Mb) from (Hess *et al.*, 2011) (Supplemental Text S1, Section 3.2, Datasets S3–S6) for evaluation.

4.3.1 Benchmarks with simulated datasets

We constructed the simulated datasets by assembling simulated reads with an empirical error profile. The details on how the simulated reads were generated and assembled can be found in (Supplemental Text S1, Section 3.1). For the evaluation, precision and recall

were calculated (Supplemental Text S1, Section 3.9). Furthermore, these measures were also calculated with a 'correction', to account for the case where the sequences of one taxon were consistently assigned to a different taxon, as for draft genome reconstruction, it is more important that the sequences are assigned consistently than that the taxonomic identifier is correct. To assess the performance of the different methods in assigning the simulated sequence fragments without related reference genomes being available, 'new strain', 'new species' and 'new genus' scenarios were simulated by removing all sequence data from the taxa of the simulated test dataset at each rank from the reference data. Furthermore, for *PPS+*, we distinguished whether the reference data were excluded (masked) from the reference sequence (RS) collection or also from the marker gene (MG) collection, since the MG collection included sequences for 15 times more distinct species than the RS collection. There were therefore two different situations to consider (Table 4.1).

Table 4.1. Test scenarios.

Test scenario	Rank masked from RS	Rank masked from MG
1.	None	None
2.	Strain	None
3.	Species	None
4.	Genus	None
5.	Strain	Strain
6.	Species	Strain
7.	Genus	Strain
8.	Species	Species
9.	Genus	Genus

Test scenarios where data was removed (masked) up to the specified rank for the corresponding taxa represented in the simulated metagenome datasets from the reference collections. RS denotes the reference collection of complete or draft genomes; MG indicates the reference collection of marker genes (Supplemental Text S1, Section 3.3).

PPS+ showed a substantially improved precision and recall over the *PPS* generic model, which demonstrated the impact of the improved selection of training data and modeled taxa (Figs. 4.2A and 4.2C, S1A–S1D and S3A–S3D). *PPS+* almost always had higher precision and recall than *MEGAN4* and *Kraken,* except when almost all test data were included in the reference sequences (Figs. 4.2A and 4.2C, S1A–S1C, S1E, S3A–S3C, S3E, S14A and S14C). This was even more pronounced when comparing bin quality using the corrected measures (Figs. 4.2B and 4.2D, S2A–S2C, S2E, S4A–S4C, S4E, S14B and S14D).

When comparing *PPS+* to *taxator-tk*, *PPS+* had substantially improved recall, particularly for lower ranks (Figs. 4.2A and 4.2C, S1A–S1C, S1F, S3A–S3C and S3F); while *taxator-tk* outperformed all other methods in terms of precision (Figs. 4.2A and 4.2C, S1A–S1F and S3A–S3F). Both methods were similarly precise when analyzing bin recovery, independent of assigning the taxonomic identifiers to the corrected measures (Figs. 4.2B and 4.2D, S2A–S2C, S2F, S4A–S4C and S4F). As a strong point of *PPS+*, we also observed that it more rarely predicted wrong taxa that were not a part of the sample than the other methods (Fig. S5). For example, for the genus rank in Scenarios 3 and 8, *PPS+* assigned sequences to only 2–5 false positive taxa, while *taxator-tk* identified 20, *MEGAN4* 37 and *PPS* 59 false ones. If *PPS+* identified wrong taxa, these were usually very closely related to the true taxa.

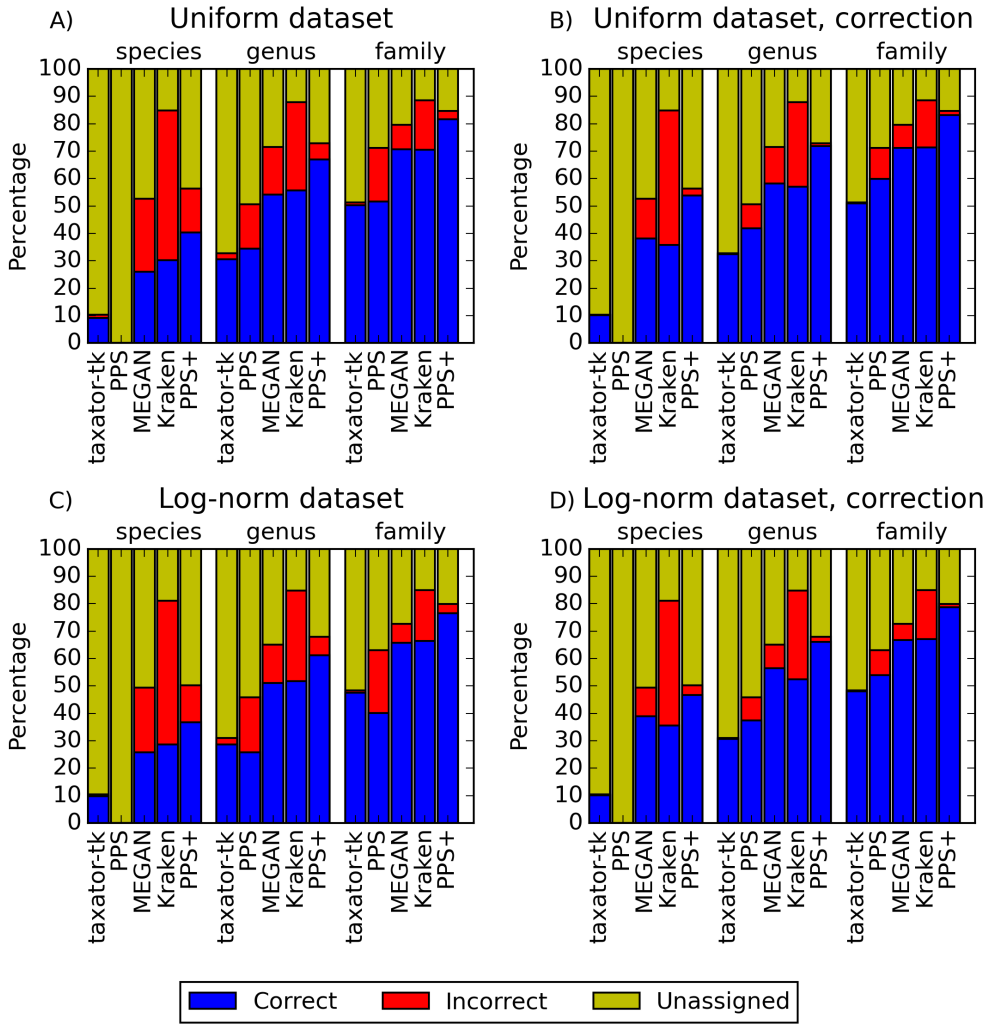


Figure 4.2. Performance comparisons with simulated datasets.

(A) and (C) show the fraction of correct, incorrect and unassigned bp for simulated datasets with uniform and log-normally distributed species abundance for *PhyloPythiaS+*, the generic *PhyloPythiaS* model, *MEGAN4, Kraken* and *taxator-tk* for assignments at the species, genus and family ranks. Results were averaged over all test 'scenarios' (Table 4.1), where sequences of the same strain, species or genus from the simulated metagenomes were removed from the genome, draft genome and marker gene reference sequence collections (Figs. S1, S3, S14A and S14C). (B) and (D) show the portion of consistently (correct), inconsistently (incorrect) and unbinned (unassigned) bp without consideration of the taxonomic identifiers (Figs. S2, S4, S14B and S14D, Supplemental Text S1, Section 3.9.2). The exact values and the corresponding precision, recall and f_1-score are contained in (Table S2–S5) for (A–D), respectively.

4.3.2 Benchmarks with real datasets

4.3.2.1 Comparison of scaffold and contig assignments

For each taxonomic rank, the percentage and the total number of kb (% agreement and kb agreement) that were assigned the same taxonomic identifier were calculated for the real datasets, based on the assignments of scaffold and contig sequences (Supplemental Text S1, Section 3.10.1). For the chunked cow rumen dataset (Supplemental Text S1, Section 3.2.2), *taxator-tk* had the highest assignment consistency (Table 4.2); however, it assigned much fewer data than the other methods at lower taxonomic ranks. A detailed comparison is given in heat maps (Figs. S6–S13). *PPS+* performed substantially better by both measures than the generic *PPS* model in almost all cases. *PPS+* was also more consistent than *MEGAN4* for all lower ranks and assigned many more sequences than *MEGAN4* overall. For instance, at the genus rank, the scores were 84.3 and 56 '% agreement', as well as 33,724 and 13,726 'kb agreement' for *PPS+* and *MEGAN4,* respectively. The overall low numbers for *Kraken* suggests that it is rather not applicable to samples containing novel taxa. Also, the low number of consistently assigned bp by *MEGAN4* and *taxator-tk* to lower taxonomic ranks reflects the availability of few related reference genome sequences for the cow rumen metagenome sample, which is not an issue for a composition-based method *PPS+*.

For the human gut microbiome, extensive sequencing of isolate cultures has resulted in a large collection of several hundred reference genome sequences. Accordingly, for the human gut dataset, *taxator-tk, MEGAN4* and *Kraken* assigned many more sequences than they did for the cow rumen dataset (Tables 4.2 and 4.3). For *Kraken* and *MEGAN4*, this was most pronounced for the genus and species ranks, even though this was also caused by counting scaffolds containing only one contig being consistent to itself. The most consistent method was again *taxator-tk*, but it also assigned fewer sequences than the other methods. *PPS+* performed better than the generic *PPS* model in all cases in terms of both measures (Table 4.3). *PPS+* and *MEGAN4* showed comparable consistency, with *PPS+* being more consistent for the class, order and species ranks, and *MEGAN4* being more consistent for the superkingdom, family and genus ranks. However, *PPS+* consistently assigned (kb agreement) more sequences than *MEGAN4*, except for the genus and species ranks. Thus, in the case of larger collections of related isolate genome sequences being available, composition- and homology-based methods perform similarly well.

The taxonomic scaffold-contig consistency of the assignments was additionally evaluated (Tables S6 and S7) using a set of measures (Supplemental Text S1, Section 3.10.2) that provide more detailed insights into assignment consistency (Supplemental Text S1, Section 5.1) and support the conclusions in this section.

Table 4.2. Comparison of contig and scaffold assignments of the chunked cow rumen dataset.

Method	Rank	% agreement	kb agreement
PPS+	Phylum	73.9	**153,774**
PPS	Phylum	67.8	75,538
MEGAN4	Phylum	74.2	43,380
taxator-tk	Phylum	**98.2**	59,702
Kraken	Phylum	*67.0*	*33,558*
PPS+	Class	86.0	**99,596**
PPS	Class	*58.5*	43,931
MEGAN4	Class	68.5	33,780
taxator-tk	Class	**97.7**	*23,190*
Kraken	Class	*58.5*	27,536
PPS+	Order	88.4	**98,616**
PPS	Order	63.8	41,349
MEGAN4	Order	68.9	32,650
taxator-tk	Order	**98.0**	*22,368*
Kraken	Order	*57.0*	26,410
PPS+	Family	80.0	**46,343**
PPS	Family	55.8	19,158
MEGAN4	Family	55.0	15,790
taxator-tk	Family	**98.9**	*7,276*
Kraken	Family	*45.2*	18,370
PPS+	Genus	84.3	**33,724**
PPS	Genus	63.2	12,938
MEGAN4	Genus	56.0	13,726
taxator-tk	Genus	**99.1**	*6,042*
Kraken	Genus	*43.7*	16,912
PPS+	Species	91.6	9,821
PPS	Species	N/A	N/A
MEGAN4	Species	54.6	8,502
taxator-tk	Species	**100.0**	*292*
Kraken	Species	*38.1*	**14,186**

Contigs of the cow rumen dataset of at least 10 kb were divided into chunks of 2 kb for evaluation of assignment consistency (Supplemental Text S1, Section 3.2.2). The contigs and scaffolds of the chunked cow rumen dataset were assigned using *PPS+*, the generic *PPS* model, *MEGAN4*, *taxator-tk* and *Kraken*. For each method, up to two taxonomic identifiers were assigned to each contig at each rank, i.e., one identifier came from the contig assignment and the second identifier came from the corresponding scaffold assignment. Contigs with less than two taxonomic assignments at each rank were not considered in this comparison. The measure '% agreement' was the percentage of contigs with the same two taxonomic identifiers at a particular rank, whereas 'kb agreement' was the total number of kb of contigs with the same taxonomic identifiers (Supplemental Text S1, Section 3.10.1). Bold numbers correspond to the best values, whereas italic numbers indicate the worst values.

Table 4.3. Comparison of contig and scaffold assignments of the human gut metagenome dataset.

Method	Rank	% agreement	kb agreement
PPS+	Phylum	99.0	**140,283**
PPS	Phylum	*97.0*	124,884
MEGAN4	Phylum	99.0	127,658
taxator-tk	Phylum	**100.0**	*104,475*
Kraken	Phylum	97.6	123,428
PPS+	Class	99.5	**134,707**
PPS	Class	96.9	118,068
MEGAN4	Class	98.5	122,131
taxator-tk	Class	**100.0**	*84,228*
Kraken	Class	*96.3*	121,071
PPS+	Order	99.5	**134,127**
PPS	Order	97.3	117,185
MEGAN4	Order	98.6	121,811
taxator-tk	Order	**100.0**	*83,337*
Kraken	Order	*96.3*	121,003
PPS+	Family	94.0	**110,664**
PPS	Family	92.6	97,066
MEGAN4	Family	96.2	98,582
taxator-tk	Family	**99.8**	*43,751*
Kraken	Family	*89.4*	109,151
PPS+	Genus	95.3	82,992
PPS	Genus	91.9	58,883
MEGAN4	Genus	96.1	86,495
taxator-tk	Genus	**99.9**	*34,667*
Kraken	Genus	*88.3*	**97,097**
PPS+	Species	94.7	43,329
PPS	Species	N/A	N/A
MEGAN4	Species	93.5	64,554
taxator-tk	Species	**99.7**	*10,314*
Kraken	Species	*81.3*	**94,591**

Contig and scaffold sequences of the human gut metagenome dataset were assigned using *PPS+*, the generic *PPS* model, *MEGAN4*, *taxator-tk* and *Kraken*. The measures '% agreement' and 'kb agreement' were used to compare individual methods (Supplemental Text S1, Section 3.10.1). Bold numbers correspond to the best values, whereas italic numbers indicate the worst values.

4.3.2.2 Comparison to an expert binning based on marker genes

A taxonomic binning generated by *PhyloPythia* (*PP*) with expert guidance for sample-derived model construction (Turnbaugh *et al.*, 2010) was compared to the *PPS+* assignments. Scaffolds that were unassigned by either method were not considered. The *PP* expert binning and the *PPS+* binning agreed well, down to the order rank (Table 4.4).

For the family and genus ranks, the overlap of both methods dropped to 69.5–74.1%, which may partly be due to changes in the NCBI taxonomy since the generation of the expert binning in 2009. Both *PPS+* and *PP* assignments were highly consistent with the MG assignments made by the + component of *PPS+* alone, though only a small number of scaffolds with marker genes could be compared (7–23% for different ranks). While *PPS+* had a larger overlap ('% agreement') with the MG assignments at the genus rank, *PP* had a larger overlap ('% agreement') with the MG assignments at the family rank. Moreover, we compared the number of taxonomic assignments for individual methods (Fig. 4.3): *PPS+* assigned sequences to low-ranking taxa down to the species level, in agreement with the MG assignments, while *PP* often assigned the respective sequences only to the parental taxa. This demonstrates that *PPS+* can generate a high quality taxonomic binning in a fully automated manner.

Table 4.4. Comparison to an expert binning based on marker genes.

Comparison	Rank	% agreement	kb agreement
PP vs PPS+	Superkingdom	99.6	**160,617**
MG *vs PP*	Superkingdom	**99.7**	38,314
MG *vs PPS+*	Superkingdom	*99.5*	*38,220*
PP vs PPS+	Phylum	*95.4*	**149,213**
MG *vs PP*	Phylum	96.9	*17,771*
MG *vs PPS+*	Phylum	**98.7**	18,065
PP vs PPS+	Class	*97.0*	**145,887**
MG *vs PP*	Class	98.1	*17,599*
MG *vs PPS+*	Class	**100.0**	17,869
PP vs PPS+	Order	*98.0*	**145,373**
MG *vs PP*	Order	98.3	*17,494*
MG *vs PPS+*	Order	**100.0**	17,764
PP vs PPS+	Family	*69.5*	**95,779**
MG *vs PP*	Family	**90.7**	13,047
MG *vs PPS+*	Family	83.7	*12,013*
PP vs PPS+	Genus	*74.1*	**78,686**
MG *vs PP*	Genus	91.6	12,235
MG *vs PPS+*	Genus	**94.9**	*11,479*

Comparison of the taxonomic assignments of *PPS+* versus *PhyloPythia* (*PP*), with expert guidance for sample-derived model construction (Turnbaugh *et al.*, 2010) for the human gut scaffolds (161,343 kb) based on marker genes (MG), using the + component of *PPS+*. The measure '% agreement' represents the percentage of bp assigned by both methods to the same taxonomic identifiers at a given rank, whereas 'kb agreement' is the corresponding number of kb assigned by both methods to the same taxonomic identifier. Scaffolds assigned by only one method are not considered in this comparison. Bold numbers correspond to the best values, whereas italic numbers indicate the worst values.

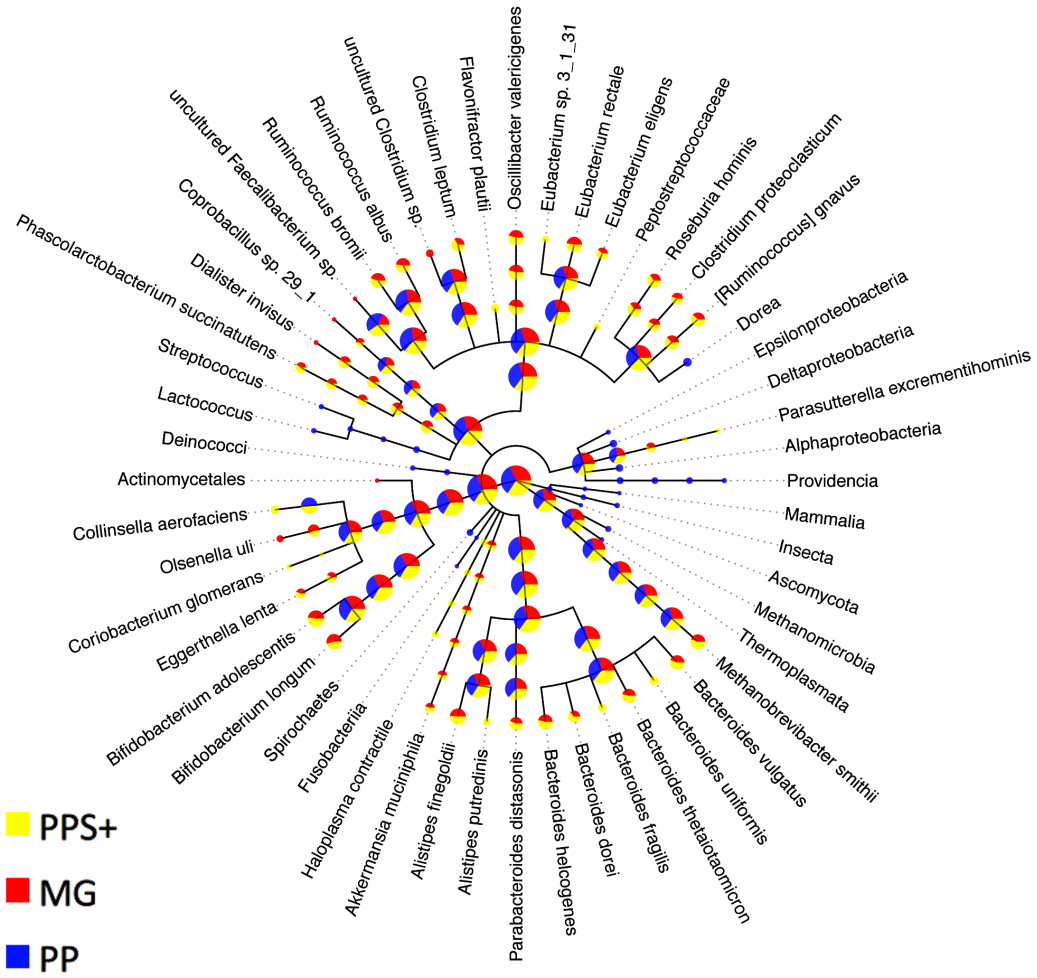


Figure 4.3. Comparison to expert binning based on marker genes.

The amount of assigned bp by *PhyloPythia* (*PP*), *PhyloPythiaS+* (*PPS+*) and taxonomically informative marker genes directly (MG) to each taxon are indicated by the pie chart sizes on a log-scale for the human gut metagenome sample (Turnbaugh *et al.*, 2010; Patil, Haider, *et al.*, 2011). *PhyloPythiaS+* automatically determined the taxa to model from the sample. For the expert-trained *PhyloPythia*, the taxa to model were specified by an expert, and were included in the model if they were covered by sufficient reference sequence data retrieved separately from the sample and from sequenced human gut isolates. *PhyloPythiaS+* assigned sequences to low-ranking taxa down to the species level, in agreement with the marker gene assignments, while *PhyloPythia* often assigned these sequences to the parental taxa. For the MG assignments, a negligible amount – only two contigs (3.6 kb) of two scaffolds (231 kb) – were used as sample-derived training data for *PPS+*; as mainly sample contigs (2.5 Mb) that were not part of scaffolds were used as sample-derived data to train *PPS*.

4.3.3 Throughput comparison

The throughput of the individual methods for contig assignments of the human gut sample was calculated (Supplemental Text S1, Sections 3.3, 3.4 and 5.3). The throughput of *Kraken* substantially varied between 38.4 Mb/h and 4.2 Gb/h in our experiments, depending on whether its large (~200 GB) reference database was already loaded in the main memory or not, therefore *Kraken* is the fastest method in high performance environments. When only the prediction step of *PPS+* was considered, *PPS+* assigned up to 0.5 Gb/h and was more than 7 times faster than the homology-based methods (Fig. 4.4). This is relevant when *PPS* models are reused for the classification of another sample. Moreover, unlike the homology-based tools and *Kraken*, *PPS+* can be run on a standard laptop, as it requires much less main memory (see Supplemental Text S1, Section 3.4 for the hardware configurations used).

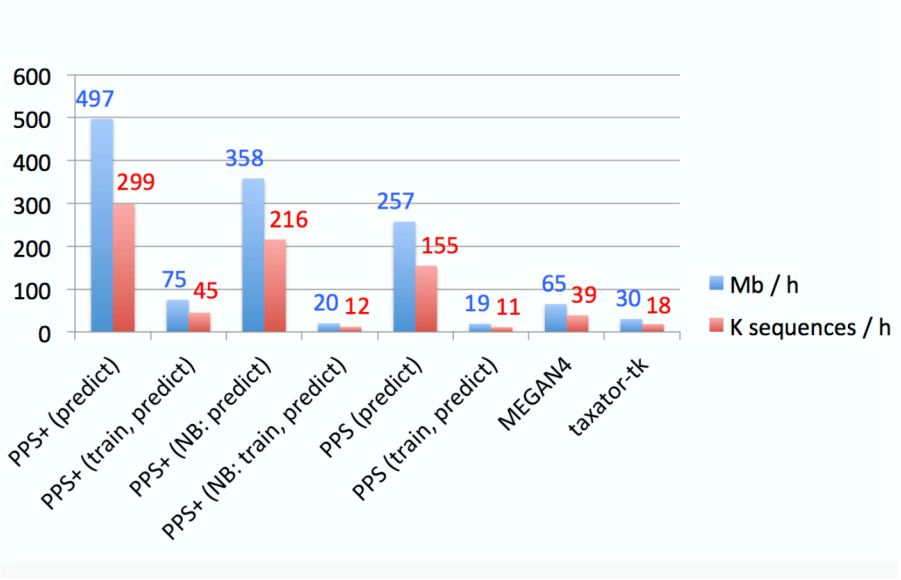


Figure 4.4. Empirical comparison of execution times.

The throughput was measured in Mb and the number of sequences classified within 1 h with one execution thread, using all assembled contigs of the human gut metagenome dataset on a server computer with an AMD Opteron 6386 SE 2.8 GHz processor and 512 GB of RAM. Default settings were used for all methods (Supplemental Text S1, Sections 3.5–3.7). Both *MEGAN4* and *taxator-tk* were run using *BLAST*. For *MEGAN4,* only the runtime of *BLAST* was considered, as the runtime of the subsequent algorithm was negligible. For *PhyloPythiaS* and *PhyloPythiaS+*, the throughput was calculated for the prediction step and both steps (training and prediction). The former is relevant when using previously generated models for the classification of multiple samples. The execution time shown for *PhyloPythiaS* is approximately three times better than that for the original release, as we incorporated the new *k*-mer counting algorithm. *PhyloPythiaS+* was the only method that could also be executed on a standard laptop (NB) with an Intel i5 M520 2.4 GHz processor, 4 GB of RAM and 150 GB disk space.

4.4 Conclusions

We describe a taxonomic assignment program that produces accurate assignments with a precision of 80% or more also for low-ranking taxa from metagenome samples. *PPS+* is a fully automated successor of the *PhyloPythiaS* software. It automatically determines the most relevant taxa to be modeled and suitable training sequences directly from the input sample, which are then used to generate a sample-specific structured output SVM taxonomic classifier for the taxonomic binning of a sample. This enables its use for researchers without experience in the field or time to search for suitable training sequences for the manual construction of well-matching taxonomic classifier to a particular metagenome sequence sample.

PPS+ is best suited for the analysis of large NGS metagenome samples with assembled contigs (>1 kb) carrying marker genes or datasets including the high quality longer PacBio (Chin *et al.*, 2013) consensus reads. Contrary to some recent methods for the taxonomic profiling or binning of multiple similar samples (Sunagawa *et al.*, 2013), *PPS+* can be also applied to individual samples. *PPS+* requires only 100 kb of sample-derived data to model a bin, while homology-based methods require large related reference genome or draft genome sequence collections for substantial assignments to low-ranking taxa. Our experiments on both real and simulated metagenome samples showed that *PPS+* automatically reconstructed many low-ranking bins from metagenome samples, such as for genera and species, representing draft genomes or pan-genomes of different community members.

The novel implementation of the *k*-mer counting algorithm accelerated *k*-mer counting 100-fold in comparison to the original *PPS* software and made *PPS+* overall up to three times faster. The method performed favorably in comparison to all state-of-the-art *k*-mer counting software for the simultaneous enumeration of 4-6-mers, commonly used for composition-based binning.

PPS models can be reused when classifying multiple samples from the same or similar environments. When comparing assignment with *PPS+* to *MEGAN4* and *taxator-tk*, *PPS+* showed a competitive processing time, allowing to process up to 0.5 Gb of sequences per hour with a given *PPS* model on a single core with much lower main memory requirements, while *MEGAN4* processed 0.065 Gb and *taxator-tk* 0.03 Gb (Fig. 4.4). The fastest method in the comparison was *Kraken* with up to 4.2 Gb/h; however, we have

found that *Kraken* should be used only for well-studied environments, for which many closely related (draft) genomes have been sequenced, as an alternative to alignment-based methods, as its use for samples originating from novel environments is very limited (Fig. 4.2).

In terms of assignment quality, we found that *PPS+* often outperformed *MEGAN4* and *Kraken* in terms of precision, recall and consistency. *Taxator-tk* performed best in terms of precision and consistency, but assigned substantially fewer sequences to low taxonomic ranks. *PPS+* also excelled in determining the taxa that were part of the simulated metagenome community. We found that the fully automated *PPS+* binning can be as good as an expert-guided binning with the original *PhyloPythia* implementation. *PPS+* also showed a substantially improved assignment performance compared to the generic *PPS* model.

To conclude, the newly introduced self-training (+) component and the faster *k*-mer counting algorithm implemented in *PPS+* allow users to generate high quality taxonomic binnings of metagenome samples in a high-throughput fashion, without requiring expensive hardware, manual intervention and expert knowledge. It should be helpful to a wide range of users. An initial version of the software has been already employed for the taxonomic binning of a metagenome sample from reindeer guts by (Pope, Mackenzie, *et al.*, 2011) and it is currently used in several other projects: for instance, a *PPS+* binning of shotgun metagenome samples indicated the likely metabolite flow and participating microbial phylotypes for a biogas-producing microbial community tolerant of high ammonia levels (Supplemental Text S2).

PPS+ is distributed with a large reference sequence collection (containing Bacterial and Archaeal data) in a virtual machine, which makes it easy to install. This allows metagenome sample analysis on a standard laptop under Windows, Unix or OS X systems.

4.5 Supplementary material

The supplementary material is available in Chapter 7 and at *PeerJ* online:
https://doi.org/10.7717/peerj.1603

5 Synopsis

The main objective of my PhD project was to develop methods for the haplotype-specific gene assembly from shotgun metagenomes and for the taxonomic classification of metagenome sequences to low-ranking taxonomic bins. We have developed *Snowball* (Gregor, Schönhuth, *et al.*, 2016), which is to the best of our knowledge the first gene assembler that is able to distinguish among gene sequences of individual strains of a species, given self-overlapping paired-end reads of a sequenced metagenome sample. We have employed *Snowball* to assemble simulated reads generated from the recently published *Rhizobia* strains (Bai *et al.*, 2015), which demonstrates the capability of our method to assemble gene sequences of closely related novel strains. We have developed *PhyloPythiaS+* (Gregor, Dröge, *et al.*, 2016), which is to the best of our knowledge the first method that performs profiling based on marker genes and consequent composition-based taxonomic binning, given assembled contigs ($>$ 1 kb) or high quality longer PacBio (Chin *et al.*, 2013) consensus reads generated from a metagenome sample. We have extensively evaluated our method with real and simulated datasets and compared it to the closest competitors. Our experiments showed that *PhyloPythiaS+* outperformed the competing tools in the scenarios, where the input metagenome sample contained novel taxa (e.g. species). *PhyloPythiaS+* also correctly assigned more sequences to the low-ranking taxonomic bins than the other tools in the comparison. Moreover, *PhyloPythiaS+* performed well in the CAMI challenge (Sczyrba *et al.*, 2017), especially in terms of recall; and has already been successfully employed in several studies (Pope, Mackenzie, *et al.*, 2011; Daims *et al.*, 2015; Ikeda-Ohtsubo *et al.*, 2016; Frank, Pan, *et al.*, 2016; Frank, Arntzen, *et al.*, 2016; Otten *et al.*, 2016; Driscoll *et al.*, 2017; Zhu *et al.*, 2017). We believe that our methods will be valuable for researchers studying species evolution, strain or gene diversity, genes under selection, virulent genes, metagenome samples originating from novel environments, for draft genome reconstruction and for the subsequent functional analysis of the studied metagenome microbial communities.

6 References

Aagaard,K. *et al.* (2013) The Human Microbiome Project strategy for comprehensive sampling of the human microbiome and why it matters. *FASEB Journal*, **27**, 1012–1022.

Ahn,T.-H. *et al.* (2015) Sigma: Strain-level inference of genomes from metagenomic analysis for biosurveillance. *Bioinformatics*, **31**, 170–177.

Arumugam,M. *et al.* (2011) Enterotypes of the human gut microbiome. *Nature*, **473**, 174–180.

Audano,P. and Vannberg,F. (2014) KAnalyze: a fast versatile pipelined K-mer toolkit. *Bioinformatics*, **30**, 2070–2072.

Baaijens,J.A. *et al.* (2017) De novo assembly of viral quasispecies using overlap graphs. *Genome Research*, **27**, 835–848.

Bai,Y. *et al.* (2015) Functional overlap of the Arabidopsis leaf and root microbiota. *Nature*, **528**, 364–369.

Batzoglou,S. *et al.* (2002) ARACHNE: A Whole-Genome Shotgun Assembler. *Genome Research*, **12**, 177–189.

Blaser,M. *et al.* (2013) The microbiome explored: recent insights and future challenges. *Nature Reviews Microbiology*, **11**, 213–217.

Boisvert,S. *et al.* (2012) Ray Meta: scalable de novo metagenome assembly and profiling. *Genome Biology*, **13**, R122.

Brady,A. and Salzberg,S. (2011) PhymmBL expanded: confidence scores, custom databases, parallelization and more. *Nature Methods*, **8**, 367–367.

Butler,J. *et al.* (2008) ALLPATHS: de novo assembly of whole-genome shotgun microreads. *Genome Research*, **18**, 810–820.

Chin,C.S. *et al.* (2013) Nonhybrid, finished microbial genome assemblies from long-read SMRT sequencing data. *Nature Methods*, **10**, 563–569.

Cole,J.R. *et al.* (2007) The ribosomal database project (RDP-II) - introducing myRDP space and quality controlled public data. *Nucleic Acids Research*, **35**, D169–D172.

Cole,S. and Saint-Girons,I. (1999) Bacterial Genomes – All Shapes and Sizes. In, Charlebois,R. (ed), *Organization of the Prokaryotic Genome*. Washington, DC, pp. 35–62.

Daims,H. *et al.* (2015) Complete nitrification by Nitrospira bacteria. *Nature*, **528**, 504–509.

Deorowicz,S. *et al.* (2015) KMC 2: fast and resource-frugal k-mer counting. *Bioinformatics*, **31**, 1569–1576.

DeSantis,T.Z. *et al.* (2006) Greengenes, a chimera-checked 16S rRNA gene database and workbench compatible with ARB. *Applied and Environmental Microbiology*, **72**, 5069–5072.

Deschavanne,P.J. *et al.* (1999) Genomic signature: characterization and classification of species assessed by chaos game representation of sequences. *Molecular Biology and Evolution*, **16**, 1391–1399.

Dijkstra,E.W. (1959) A note on two problems in connexion with graphs. *Numerische Mathematik*, **1**, 269–271.

Driscoll,C.B. *et al.* (2017) Towards long-read metagenomics: complete assembly of three novel genomes from bacteria dependent on a diazotrophic cyanobacterium in a freshwater lake co-culture. *Standards in Genomic Sciences*, **12**, 1–16.

Dröge,J. and McHardy,A.C. (2012) Taxonomic binning of metagenome samples generated by next-generation sequencing technologies. *Briefings in Bioinformatics*, **13**, 646–655.
Dröge,J. *et al.* (2014) Taxator-tk: precise taxonomic assignment of metagenomes by fast approximation of evolutionary neighborhoods. *Bioinformatics*, **31**, 817–824.
Duda,R.O. *et al.* (2000) Pattern classification Second edition. A Wiley-Interscience Publication, New York.
Eddy,S.R. (2011) Accelerated Profile HMM Searches. *PLoS Computational Biology*, **7**, e1002195.
Federhen,S. (2011) The NCBI Taxonomy database. *Nucleic Acids Research*, **40**, D136–D143.
Finn,R.D. *et al.* (2014) Pfam: the protein families database. *Nucleic Acids Research*, **42**, D222–D230.
Finn,R.D. *et al.* (2016) The Pfam protein families database - towards a more sustainable future. *Nucleic Acids Research*, **44**, D279–D285.
Frank,J.A., Arntzen,M.Ø., *et al.* (2016) Novel Syntrophic Populations Dominate an Ammonia-Tolerant Methanogenic Microbiome. *mSystems*, **1**, e00092–16.
Frank,J.A., Pan,Y., *et al.* (2016) Improved metagenome assemblies and taxonomic binning using long-read circular consensus sequence data. *Scientific Reports*, **6**, 1–10.
Gerlach,W. and Stoye,J. (2011) Taxonomic classification of metagenomic shotgun sequences with CARMA3. *Nucleic Acids Research*, **39**, e91.
Giloteaux,L. *et al.* (2016) Reduced diversity and altered composition of the gut microbiome in individuals with myalgic encephalomyelitis/chronic fatigue syndrome. *Microbiome*, **4**, 1–12.
Goodwin,S. *et al.* (2016) Coming of age: ten years of next-generation sequencing technologies. *Nature Reviews Genetics*, **17**, 333–351.
Gregor,I., Dröge,J., *et al.* (2016) PhyloPythiaS+: a self-training method for the rapid reconstruction of low-ranking taxonomic bins from metagenomes. *PeerJ*, **4**, e1603.
Gregor,I., Schönhuth,A., *et al.* (2016) Snowball: strain aware gene assembly of metagenomes. *Bioinformatics*, **32**, i649–i657.
Gyarmati,P. *et al.* (2016) Metagenomic analysis of bloodstream infections in patients with acute leukemia and therapy-induced neutropenia. *Scientific Reports*, **6**, 1–8.
Handelsman,J. *et al.* (1998) Molecular biological access to the chemistry of unknown soil microbes: a new frontier for natural products. *Chemistry & biology*, **5**, R245–R249.
Hess,M. *et al.* (2011) Metagenomic discovery of biomass-degrading genes and genomes from cow rumen. *Science*, **331**, 463–467.
Huang,W. *et al.* (2012) ART: a next-generation sequencing read simulator. *Bioinformatics*, **28**, 593–594.
Hugenholtz,P. (2002) Exploring prokaryotic diversity in the genomic era. *Genome Biology*, **3**, reviews0003.1–0003.8.
Huson,D.H. *et al.* (2011) Integrative analysis of environmental sequences using MEGAN4. *Genome Research*, **21**, 1552–1560.
Ikeda-Ohtsubo,W. *et al.* (2016) 'Candidatus Adiutrix intracellularis', an endosymbiont of termite gut flagellates, is the first representative of a deep-

branching clade of Deltaproteobacteria and a putative homoacetogen. *Environmental Microbiology*, **18**, 2548–2564.
Joachims,T. *et al.* (2009) Cutting-plane training of structural SVMs. *Machine Learning*, **77**, 27–59.
Johnson,S. (2006) Remote protein homology detection using hidden markov models. 1–106.
Kalyuzhnaya,M.G. *et al.* (2008) High-resolution metagenomics targets specific functional types in complex microbial communities. *Nature Biotechnology*, **26**, 1029–1034.
Karlin,S. and Burge,C. (1995) Dinucleotide relative abundance extremes: a genomic signature. *Trends in genetics*, **11**, 283–290.
Karp,R.M. and Rabin,M.O. (1987) Efficient randomized pattern-matching algorithms. *IBM Journal of Research and Development*, **31**, 249–260.
Kingsford,C. *et al.* (2010) Assembly complexity of prokaryotic genomes using short reads. *BMC Bioinformatics*, **11**, 1–11.
Kuczynski,J. *et al.* (2011) Experimental and analytical tools for studying the human microbiome. *Nature Reviews Genetics*, **13**, 47–58.
Kunin,V. *et al.* (2008) A bioinformatician's guide to metagenomics. *Microbiology and Molecular Biology Reviews*, **72**, 557–578.
Laehnemann,D. *et al.* (2016) Denoising DNA deep sequencing data-high-throughput sequencing errors and their correction. *Briefings in Bioinformatics*, **17**, 154–179.
Laserson,J. *et al.* (2011) Genovo: de novo assembly for metagenomes. *Journal of Computational Biology*, **18**, 429–443.
Li,D. *et al.* (2015) MEGAHIT: an ultra-fast single-node solution for large and complex metagenomics assembly via succinct de Bruijn graph. *Bioinformatics*, **31**, 1674–1676.
Li,H. and Durbin,R. (2009) Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics*, **25**, 1754–1760.
Liu,B. *et al.* (2011) Accurate and fast estimation of taxonomic profiles from metagenomic shotgun sequences. *BMC Genomics*, **12**, S4.
Loman,N.J. *et al.* (2012) High-throughput bacterial genome sequencing: an embarrassment of choice, a world of opportunity. *Nature Reviews Microbiology*, **10**, 599–606.
Luo,R. *et al.* (2012) SOAPdenovo2: an empirically improved memory-efficient short-read de novo assembler. *GigaScience*, **1**, 1–6.
Marcais,G. and Kingsford,C. (2011) A fast, lock-free approach for efficient parallel counting of occurrences of k-mers. *Bioinformatics*, **27**, 764–770.
Marschall,T. *et al.* (2016) Computational pan-genomics: status, promises and challenges. *Briefings in Bioinformatics*, 1–18.
Martín,H.G. *et al.* (2006) Metagenomic analysis of two enhanced biological phosphorus removal (EBPR) sludge communities. *Nature Biotechnology*, **24**, 1263–1269.
Marx,V. (2016) Microbiology: the road to strain-level identification. *Nature Methods*, **13**, 401–404.
Matousek,J. and Nesetril,J. (2009) Invitation to Discrete Mathematics. Second edition. Oxford University Press, Oxford.
McHardy,A.C. *et al.* (2007) Accurate phylogenetic classification of variable-length DNA fragments. *Nature Methods*, **4**, 63–72.

Meinicke,P. *et al.* (2011) Mixture models for analysis of the taxonomic composition of metagenomes. *Bioinformatics*, **27**, 1618–1624.
MetaSUB International Consortium (2016) The Metagenomics and Metadesign of the Subways and Urban Biomes (MetaSUB) International Consortium inaugural meeting report. *Microbiome*, **4**, 24.
Metzker,M.L. (2010) Sequencing technologies — the next generation. *Nature Reviews Genetics*, **11**, 31–46.
Minoche,A.E. *et al.* (2011) Evaluation of genomic high-throughput sequencing data generated on Illumina HiSeq and genome analyzer systems. *Genome Biology*, **12**, R112.
Myers,E.W. (1995) Toward Simplifying and Accurately Formulating Fragment Assembly. *Journal of Computational Biology*, **2**, 275–290.
Myers,E.W. *et al.* (2000) A whole-genome assembly of Drosophila. *Science*, **287**, 2196–2204.
Nagarajan,N. and Pop,M. (2013) Sequence assembly demystified. *Nature Reviews Genetics*, **14**, 157–167.
Namiki,T. *et al.* (2012) MetaVelvet: an extension of Velvet assembler to de novo metagenome assembly from short sequence reads. *Nucleic Acids Research*, **40**, e155.
Otten,T.G. *et al.* (2016) Elucidation of Taste- and Odor-Producing Bacteria and Toxigenic Cyanobacteria in a Midwestern Drinking Water Supply Reservoir by Shotgun Metagenomic Analysis. *Applied and Environmental Microbiology*, **82**, 5410–5420.
Ounit,R. *et al.* (2015) CLARK: fast and accurate classification of metagenomic and genomic sequences using discriminative k-mers. *BMC Genomics*, **16**, 1–13.
Patil,K.R., Haider,P., *et al.* (2011) Taxonomic metagenome sequence assignment with structured output models. *Nature Methods*, **8**, 191–192.
Patil,K.R.K., Roune,L.L., *et al.* (2011) The PhyloPythiaS web server for taxonomic assignment of metagenome sequences. *PLoS ONE*, **7**, e38581.
Pell,J. *et al.* (2012) Scaling metagenome sequence assembly with probabilistic de Bruijn graphs. *Proceedings of the National Academy of Sciences of the United States of America*, **109**, 13272–13277.
Peltola,H. *et al.* (1984) SEQAID: a DNA sequence assembling program based on a mathematical model. *Nucleic Acids Research*, **12**, 307–321.
Peng,Y. *et al.* (2011) Meta-IDBA: a de Novo assembler for metagenomic data. *Bioinformatics*, **27**, i94–i101.
Peng,Y.Y. *et al.* (2012) IDBA-UD: a de novo assembler for single-cell and metagenomic sequencing data with highly uneven depth. *Bioinformatics*, **28**, 1420–1428.
Pevzner,P.A. *et al.* (2001) An Eulerian path approach to DNA fragment assembly. *Proceedings of the National Academy of Sciences of the United States of America*, **98**, 9748–9753.
Pope,P.B., Mackenzie,A.K., *et al.* (2011) Metagenomics of the svalbard reindeer rumen microbiome reveals abundance of polysaccharide utilization Loci. *PLoS ONE*, **7**, e38571.
Pope,P.B., Smith,W., *et al.* (2011) Isolation of Succinivibrionaceae implicated in low methane emissions from Tammar wallabies. *Science*, **333**, 646–648.
Qin,J. *et al.* (2010) A human gut microbial gene catalog established by metagenomic sequencing. *Nature*, **464**, 59–65.

Quast,C. *et al.* (2013) The SILVA ribosomal RNA gene database project: improved data processing and web-based tools. *Nucleic Acids Research*, **41**, D590–D596.
Riesenfeld,C.S. *et al.* (2004) Metagenomics: genomic analysis of microbial communities. *Annual Review of Genetics*, **38**, 525–552.
Rosen,G.L. *et al.* (2011) NBC: the Naive Bayes Classification tool webserver for taxonomic classification of metagenomic reads. *Bioinformatics*, **27**, 127–129.
Roy,R.S. *et al.* (2014) Turtle: Identifying frequent k-mers with cache-efficient algorithms. *Bioinformatics*, **30**, 1950–1957.
Schirmer,M. *et al.* (2015) Insight into biases and sequencing errors for amplicon sequencing with the Illumina MiSeq platform. *Nucleic Acids Research*, **43**, e37.
Schloissnig,S. *et al.* (2013) Genomic variation landscape of the human gut microbiome. *Nature*, **493**, 45–50.
Schloss,P.D. *et al.* (2009) Introducing mothur: open-source, platform-independent, community-supported software for describing and comparing microbial communities. *Applied and Environmental Microbiology*, **75**, 7537–7541.
Sczyrba,A. *et al.* (2017) Critical Assessment of Metagenome Interpretation – a benchmark of computational metagenomics software. *bioRxiv*, 099127.
Segata,N. *et al.* (2012) Metagenomic microbial community profiling using unique clade-specific marker genes. *Nature Methods*, **9**, 811–814.
Sender,R. *et al.* (2016) Revised Estimates for the Number of Human and Bacteria Cells in the Body. *PLoS Biology*, **14**, e1002533.
Silva,G.G.Z. *et al.* (2013) FOCUS: an alignment-free model to identify organisms in metagenomes using non-negative least squares. *PeerJ*, **2**, e425.
Simpson,J.T. and Durbin,R. (2012) Efficient de novo assembly of large genomes using compressed data structures. *Genome Research*, **22**, 549–556.
Smith,T.F. and Waterman,M.S. (1981) Identification of common molecular subsequences. *Journal of Molecular Biology*, **147**, 195–197.
Staden,R. (1980) A New Computer Method for the Storage and Manipulation of Dna Gel Reading Data. *Nucleic Acids Research*, **8**, 3673–3694.
Stark,M. *et al.* (2009) MLTreeMap - accurate Maximum Likelihood placement of environmental DNA sequences into taxonomic and functional reference phylogenies. *BMC Genomics*, **11**, 1–11.
Sunagawa,S. *et al.* (2013) Metagenomic species profiling using universal phylogenetic marker genes. *Nature Methods*, **10**, 1196–1199.
Sutton,G.G. *et al.* (1995) TIGR Assembler: A New Tool for Assembling Large Shotgun Sequencing Projects. *Genome Science and Technology*, **1**, 9–19.
Töpfer,A. *et al.* (2014) Viral Quasispecies Assembly via Maximal Clique Enumeration. *PLoS Computational Biology*, **10**, e1003515.
Treangen,T.J. and Salzberg,S.L. (2011) Repetitive DNA and next-generation sequencing: computational challenges and solutions. *Nature Reviews Genetics*, **13**, 36–46.
Tringe,S.G. *et al.* (2005) Comparative metagenomics of microbial communities. *Science*, **308**, 554–557.
Turnbaugh,P.J. *et al.* (2010) Organismal, genetic, and transcriptional variation in the deeply sequenced gut microbiomes of identical twins. *Proceedings of the National Academy of Sciences of the United States of America*, **107**, 7503–7508.

Tyson,G.W. *et al.* (2004) Community structure and metabolism through reconstruction of microbial genomes from the environment. *Nature*, **428**, 37–43.
Vapnik,V.N. (1995) The Nature of Statistical Learning Theory Springer Science & Business Media, New York.
Voorhies,A.A. and Lorenzi,H.A. (2016) The Challenge of Maintaining a Healthy Microbiome during Long-Duration Space Missions. *Frontiers in Astronomy and Space Sciences*, **3**, 1–7.
Wang,Q. *et al.* (2007) Naive Bayesian classifier for rapid assignment of rRNA sequences into the new bacterial taxonomy. *Applied and Environmental Microbiology*, **73**, 5261–5267.
Wang,W. *et al.* (2015) Metagenomic analysis of microbiome in colon tissue from subjects with inflammatory bowel diseases reveals interplay of viruses and bacteria. *Inflammatory Bowel Diseases*, **21**, 1419–1427.
Ward,D.M. *et al.* (1998) A natural view of microbial biodiversity within hot spring cyanobacterial mat communities. *Microbiology and Molecular Biology Reviews*, **62**, 1353–1370.
Warnecke,F. *et al.* (2007) Metagenomic and functional analysis of hindgut microbiota of a wood-feeding higher termite. *Nature*, **450**, 560–565.
Wood,D.E. and Salzberg,S.L. (2014) Kraken: ultrafast metagenomic sequence classification using exact alignments. *Genome Biology*, **15**, R46.
Wu,M. and Eisen,J.A. (2008) A simple, fast, and accurate method of phylogenomic inference. *Genome Biology*, **9**, R151.
Wu,M. and Scott,A.J. (2012) Phylogenomic analysis of bacterial and archaeal sequences with AMPHORA2. *Bioinformatics*, **28**, 1033–1034.
Yuan,C. *et al.* (2015) Reconstructing 16S rRNA genes in metagenomic data. *Bioinformatics*, **31**, i35–i43.
Zagordi,O. *et al.* (2011) ShoRAH: estimating the genetic diversity of a mixed sample from next-generation sequencing data. *BMC Bioinformatics*, **12**, 1–5.
Zarowiecki,M. (2012) Metagenomics with guts. *Nature Reviews Microbiology*, **10**, 674–674.
Zerbino,D.R. and Birney,E.E. (2008) Velvet: algorithms for de novo short read assembly using de Bruijn graphs. *Genome Research*, **18**, 821–829.
Zerbino,D.R. *et al.* (2009) Pebble and Rock Band: Heuristic Resolution of Repeats and Scaffolding in the Velvet Short-Read de Novo Assembler. *PLoS ONE*, **4**, e8407.
Zhang,Y. *et al.* (2014) A Scalable and Accurate Targeted Gene Assembly Tool (SAT-Assembler) for Next-Generation Sequencing Data. *PLoS Computational Biology*, **10**, e1003737.
Zhu,T. *et al.* (2017) Draft Genome Sequences of Nine Cyanobacterial Strains from Diverse Habitats. *Genome Announcements*, **5**, e01676–16.

7 Journal versions of the published articles

Bioinformatics, 32, 2016, i649–i657
doi: 10.1093/bioinformatics/btw426
ECCB 2016

OXFORD

Snowball: strain aware gene assembly of metagenomes

I. Gregor[1,2], A. Schönhuth*,†, [3] and A. C. McHardy[1,2,*,†]

[1]Department of Algorithmic Bioinformatics, Heinrich-Heine-University Düsseldorf, Düsseldorf 40225, Germany, [2]Computational Biology of Infection Research, Helmholtz Center for Infection Research, Braunschweig 38124, Germany and [3]Centrum Wiskunde & Informatica, Amsterdam, XG 1098, The Netherlands

*To whom correspondence should be addressed.

†The authors wish it to be known that, in their opinion, the last two authors should be regarded as joint Last Authors.

Abstract

Motivation: Gene assembly is an important step in functional analysis of shotgun metagenomic data. Nonetheless, strain aware assembly remains a challenging task, as current assembly tools often fail to distinguish among strain variants or require closely related reference genomes of the studied species to be available.

Results: We have developed *Snowball*, a novel strain aware gene assembler for shotgun metagenomic data that does not require closely related reference genomes to be available. It uses profile hidden Markov models (HMMs) of gene domains of interest to guide the assembly. Our assembler performs gene assembly of individual gene domains based on read overlaps and error correction using read quality scores at the same time, which results in very low per-base error rates.

Availability and Implementation: The software runs on a user-defined number of processor cores in parallel, runs on a standard laptop and is available under the GPL 3.0 license for installation under Linux or OS X at https://github.com/hzi-bifo/snowball.

Contact: AMC14@helmholtz-hzi.de or a.schoenhuth@cwi.nl

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

Metagenomics is the functional or sequence-based analysis of microbial DNA isolated directly from a microbial community of interest (Kunin et al., 2008; Riesenfeld et al., 2004). This enables the analysis of microorganisms that cannot be cultivated in a laboratory. After the DNA is isolated, it is sequenced using a high-throughput sequencing platform, which results in a large dataset of short sequenced genome fragments, called reads. For a read, it is unknown from which strain it originates. Given such sequenced shotgun metagenomic data, i.e. a dataset of short reads that originate from several genome sequences of distinct strains, gene assembly aims to reconstruct coding sequences of the individual strains contained in the dataset (Fig. 1).

Gene assembly is an important step in the analysis of shotgun metagenomic data. For many purposes, including functional analysis of metagenomic data, it is sufficient, and therefore convenient to assemble only the coding sequences of the strains. It has also been shown that genes assemble well (Kingsford et al., 2010) even when only short reads are available. Moreover, metagenomic data consist mainly of prokaryotic species. As usually more than 85% of prokaryotic genomes are coding sequences (Cole and Saint-Girons, 1999); gene assembly enables to recover large parts of the respective genomes.

Importantly, strain awareness is an essential goal in assembling metagenomes, since it enables us to study gene variation among strains of a species from the sequenced microbial community, which is where much phenotypic diversity also arises. However, the assembly of closely related strains remains a challenging task. Strain aware assembly, which is assembly that is sensitive to closely related haplotypic sequences has remained an open challenge in many genomics applications. In particular, low-abundance strains can interfere with sequencing errors in common error correction routines. To date, most assembly tools still aim to assemble consensus sequence, if closely related haplotypes are present (Marschall *et al.*, 2016).

There are few tools that enable strain variant reconstruction. They often rely on the availability of closely related reference genomes of the studied species (Ahn et al., 2015; Töpfer et al., 2014; Zagordi et al., 2011), where reads are first mapped onto a reference genome, using a read mapping tool, e.g. *BWA* (Li and Durbin, 2009), strain variants are then identified through a reference guided strain aware assembly. As metagenome samples originating from

© The Author 2016. Published by Oxford University Press. All rights reserved. For Permissions, please e-mail: journals.permissions@oup.com

Downloaded from http://bioinformatics.oxfordjournals.org/ by guest on September 1, 2016

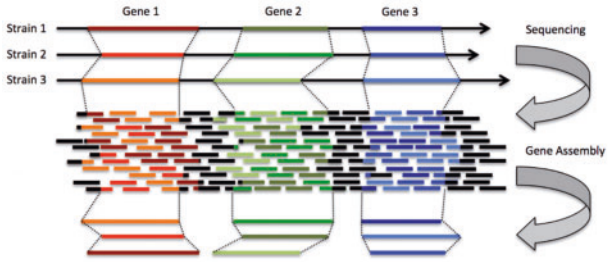


Fig. 1. An example of the gene assembly problem. In this example, the sequenced microbial community consists only of three distinct strains. Non-coding regions of the strain sequences are black, whereas coding regions are red, green and blue for genes 1, 2 and 3. Genes 1–3 are present in all three strains, although the location and gene sequences differ for distinct strains. The sequencing step results in a collection of short reads. Note that after the sequencing step, the origin of reads denoted by colours and positions within the respective strains in the figure is not known in the subsequent gene assembly step. Given a dataset containing all the short reads, the ultimate goal of the gene assembly is to determine the individual strain specific sequences of the genes

novel environments typically consist of novel species without reference genomes available, there is a need for new reference-free approaches.

Tools that are often used for *de novo* metagenome assemblies are Ray Meta (Boisvert et al., 2012), MEGAHIT (Li et al., 2015), *IDBA-UD* (Peng et al., 2012), *MetaVelvet* (Namiki et al., 2012) or *SOAPdenovo2* (Luo et al., 2012). All these tools are *k*-mer based, i.e. they transform reads into overlapping *k*-mers from which De Bruijn graphs are built, where paths in the graph correspond to the assembled contigs. This general approach, however, often fails to distinguish among strain variants. There has been recent debate on *k*-mer based approaches using De Bruijn graphs in strain aware assembly. In particular, *k*-mer based approaches can become misled, when low-abundance strains are involved, since the frequencies of the low-abundance strains are on the order of magnitude of the sequencing error rates. This leads to unpleasant interference in *k*-mer based error-correction steps, as low-abundance strains are often removed along with sequencing errors. For strain aware assembly, it is helpful to process reads at their full length, because this increases the power to distinguish low-frequent, co-occurring true mutations from sequencing errors. In this line, there has been recent evidence that shorter genomes can be assembled through overlap graph based approaches, which make use of full-length reads, using short reads (Simpson and Durbin, 2012). It was also shown that one can perform strain aware assembly through iterative construction of overlap graphs (Töpfer et al., 2014). For gene assembly from metagenomic data, the *SAT* assembler (Zhang et al., 2014) can be employed. First, it assigns reads to gene domains of interest based on profile hidden Markov models (HMMs) (Eddy, 2011; Finn et al., 2014) of the respective gene domains. Then, for each gene domain, separately, it builds overlap graphs based on the read overlaps, where the paths in the graphs correspond to the assembled contigs. However, the *SAT* assembler does not implement a sophisticated error-correction strategy, which is considered crucial for strain aware assembly. For the reconstruction of 16S genes, which are often used for phylotyping, *REAGO* (Yuan et al., 2015) can be employed. Since it has been built for 16S genes, the use of *REAGO* in more generic settings remains unclear.

The current sequencing technologies still produce relatively short erroneous reads, making it difficult to distinguish sequencing errors from genuine strain variation (Laehnemann et al., 2015). Therefore, reference-free strain reconstruction of the full-length sequences of individual strains is currently considered to be a tough computational challenge, as there may be no immediate sufficient information in the sequenced data if mutations are separated by too large stretches of sequence that agree for several strains. Therefore, new approaches are needed that push the limits imposed by the data.

Here, we present *Snowball*, a novel method for strain aware gene assembly from metagenomes that addresses the above-mentioned points. It does not require closely related reference genomes to be available. It uses profile HMMs of gene domains of interest as an input to guide the assembly. The HMM profile-based homology search is known to be capable of finding remote homology, including large number of substitutions, insertions and deletions, whereas simple read mapping onto a reference genome can find only very closely related homologs (Zhang et al., 2014). Since our method does not make use of reference genomes, we allow for strain aware gene assembly also of novel species, where reference genomes are not yet available. We have developed a novel algorithm that performs gene assembly based on read overlaps. This allows correcting errors by making use of the error profiles that underlie the overlapping reads. The consequences are twofold: First, we obtain contigs affected by only very low per-base error rates. Second, since, this way, we determine which reads stem from identical segments based on a statistically sound model, we can reliably distinguish between sequencing errors and strain-specific variants, even of very low-abundance strains. We consider these two features to represent the main improvements over the currently available assemblers. To the best of our knowledge, *Snowball* is the first tool that allows distinguishing among individual gene strain variants in metagenomes for a large set of gene domains without using reference genomes of related species.

In our experiments, we focused on distinguishing closely related strains from one species. Since two different species are substantially more divergent in terms of sequence than two different strains from the same species, good results on strains from one species also imply good or even better performance on datasets that contain several species—distinguishing species is the much easier task. We assessed the performance of *Snowball* using 21 simulated datasets, each containing 3–9 closely related *Escherichia coli* strains and on one simulated dataset containing ten recently published strains of a novel *Rhizobia* species (Bai et al., 2015). The results for the latter demonstrate the capability of the *Snowball* assembler to assemble genes of novel strains. The results for all datasets confirm that the strength of *Snowball* is its very low per-base error, due to the incorporated error-correction. Moreover, it produced substantially longer contigs and recovered a larger part of the simulated reference data in comparison to the *SAT* assembler. *Snowball* is implemented in Python, runs on a user-defined number of processor cores in parallel, runs on a standard laptop, is freely available under the GPL 3.0 license and can be installed under Linux or OS X.

2 Methods

The input of *Snowball* are two FASTQ files containing Illumina self-overlapping paired-end reads, the corresponding insert size used for the library preparation and profile HMMs of gene domains of interest. The paired-end reads may originate from multiple closely related strains or from more evolutionary divergent taxa. We have thoroughly tested *Snowball* using simulated Illumina HiSeq 2500 paired-end reads generated by the *ART* read simulator (Huang

Downloaded from http://bioinformatics.oxfordjournals.org/ by guest on September 1, 2016

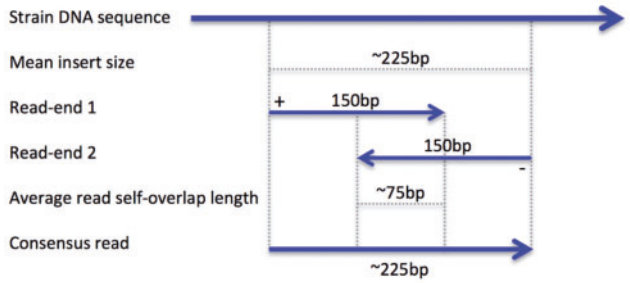


Fig. 2. An example of a self-overlapping paired-end read. Illumina HiSeq 2500 paired-end read consists of two 150bp read ends, one on the positive strand (+) and one on the negative strand (−). In our example, the mean insert size (225 bp) is smaller than two times the read end length (2 × 150 bp), therefore the paired-end reads are self-overlapping with 75 bp overlap length on average. Such a self-overlapping read can be joined into a consensus read of 225 bp length on average

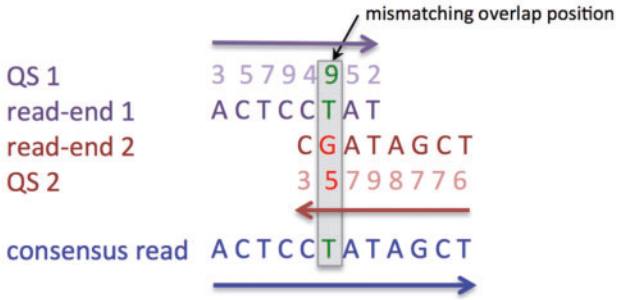


Fig. 3. Joining of self-overlapping reads example. The figure depicts a simplified example of a consensus read reconstruction. At the mismatching overlap position, read-end 1 has *T* with quality score (QS) 9, while read-end 2 has *G* with quality score 5. The resulting consensus read will have *T* at the respective position, since *T* is supported by a higher quality score than *G*. The computation of the quality scores for the consensus read is explained in the Section 2.3

et al., 2012) with 150 bp read length and 225 bp mean insert size. In this setting, the average length of the self-overlaps of the read ends is 75 bp and the length of a consensus read that originates by joining of the self-overlapping read ends is 225 bp on average (Fig. 2, Section 3.4). The output is a FASTA or a FASTQ file containing annotated assembled contigs. For each contig, the annotation contains the name of a respective gene domain to which a contig belongs, coordinates of the coding sub-sequence within a contig sequence, coverage and quality score for each contig position. The coverage and quality score information can be used for subsequent quality filtering yielding less or shorter contigs of higher quality.

Our method consists of the following steps:

- [Consensus read reconstruction]
  Self-overlapping paired-end reads are joined into longer consensus reads (Section 2.1).
- [Assignment of consensus reads to gene domains]
  Profile HMMs of selected gene domains are employed to assign consensus reads to the respective gene domains, where one consensus read is assigned to at most one gene domain (Section 2.2).
- [Assembly of consensus reads into contigs]
  For each gene domain, in parallel, consensus reads are assembled into contigs (Sections 2.3–2.5). In the assembly step, consensus reads are iteratively joined into longer and error-corrected super-reads based on the consensus read overlaps. The super-reads are then output as annotated contigs, where a super-read represents a sequence that originates by joining of at least two consensus reads into a longer sequence.

2.1 Joining self-overlapping paired-end reads

Self-overlapping paired-end reads are joined into longer error-corrected consensus sequences. The use of a library containing self-overlapping paired-end reads is a powerful strategy for an initial error-correction (Schirmer et al., 2015), which has been employed in e.g. *ALLPATHS* (Butler et al., 2008). Given the mean insert size, we determine the self-overlap that results in the minimum hamming distance between the overlapping ends of a paired-end read. A base with a higher quality score is chosen at a position within the overlap that contains mismatching bases for the respective position of the resulting consensus read sequence (Fig. 3). As the substitution error rate of the Illumina reads increases towards the ends of the paired-end reads (Minoche et al., 2011), this step results in longer consensus reads with overall lower substitution error, where the overlapping regions are almost error-free. It is also an efficient read quality filtering step, as the paired-end reads that cannot be joined, due to high substitution error rate, an insertion or a deletion within the overlapping region, are filtered out. For instance, by joining of the 150 bp paired-end Illumina HiSeq 2500 self-overlapping reads with 225 bp mean insert size results in consensus reads of length 225 bp on average. While the default error profile of the *ART* read simulator (Huang et al., 2012) yields 150 bp paired-end reads with ∼2.37% substitution error, the joined consensus reads had only ∼1.08% substitution error in our experiments. These longer, error-corrected consensus reads with low substitution error rate are convenient building blocks to start with in the subsequent steps of our method.

2.2 Assigning reads to gene domains

Consensus reads are annotated using profile HMMs of gene domains of interest and assigned to respective gene domains (Fig. 4). By default, we use the *Pfam-A* (Finn et al., 2014) (version 27) profile HMMs of 14 831 gene domains and *AMPHORA 2* (Wu and Scott, 2012) profile HMMs of 31 bacterial ubiquitous single-copy genes that are often used for phylotyping. A profile HMM of a gene domain is a probabilistic model representing a multiple sequence alignment of representative gene sequences belonging to a particular gene domain. The model can be used to annotate a query sequence (e.g. a consensus read). The annotation mainly consists of a score, start/stop positions within a query sequence and HMM start/stop coordinates. The score roughly corresponds to a probability that a query sequence belongs to the particular gene domain, i.e. if the score is high for a query sequence then it is very probable that it belongs to the respective gene domain. The start/stop positions within a query sequence define a sub-sequence of a query sequence that was identified to belong to the gene domain. The HMM start/stop coordinates correspond to the estimated coordinates of the query sub-sequence

Fig. 4. Assignment of consensus reads to gene domains. Consensus reads are assigned to individual gene domains using profile HMMs. Consensus reads that cannot be assigned to any of the gene domains with sufficient confidence remain unassigned. A consensus read is assigned to at most one gene domain

Downloaded from http://bioinformatics.oxfordjournals.org/ by guest on September 1, 2016

within the multiple sequence alignment of the respective profile HMM.

Each consensus read is translated into six protein sequences using all six reading frames (i.e. also considering the reverse complementary sequences). The *hmmsearch* command of the *HMMER 3* (Eddy, 2011) software is used to annotate the protein sequences. For each consensus read, only the reading frame with the highest score is considered. A consensus read is assigned to at most one gene domain to which it was queried with the highest score. Consensus reads with low scores (i.e. lower than default value: 40) are filtered out and not considered in the subsequent steps. If a protein sequence corresponding to a reverse complementary consensus read sequence was annotated, the corresponding reverse complementary DNA sequence of a respective consensus read is considered in the next steps. The coding DNA sub-sequence of a consensus read sequence is denoted as a (partial) coding region. The start and end HMM coordinates within a respective profile HMM are stored as part of the consensus read annotation.

As a result of this step, consensus reads are annotated and assigned to 'bins' representing individual gene domains, where one consensus read is assigned to at most one gene domain. Gene domains are building blocks of individual genes. Therefore, a 'bin' does not only contain consensus reads belonging to gene variants of individual strains. It can also contain different genes of one strain, several copies of one gene of one strain or even 'broken' gene copies.

2.3 Consensus sequence representation

We represent consensus sequences, i.e. consensus reads and super-reads using probability matrices. A super-read is a longer error-corrected sequence that originates by joining overlapping consensus reads (or consensus reads with super-reads) in the *Snowball* algorithm (Section 2.5).

For construction of such super-reads, we make use of the error profiles that come along with Illumina paired-end reads. These reads are stored in FASTQ files together with the corresponding quality scores (Fig. 5a). A quality score for a read position represents a probability that a base was sequenced correctly, i.e. it represents a probability that a particular base is present at a respective position in the FASTQ file (Fig. 5b). The complement probability represents a probability that a different base is at the respective position. The probability that different base X is present at a particular position corresponds to one third of the complement probability in our model, which reflects that apart from the correct nucleotide, there are 3 different choices for X. Note that these probabilities are only estimates, as provided by the Illumina sequencing platform.

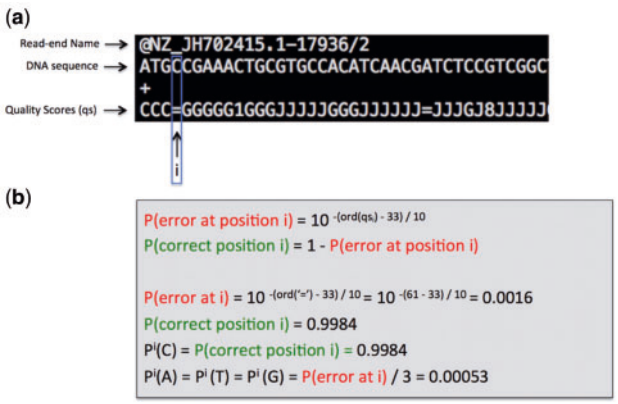


Fig. 5. FASTQ file data representation. (Panel a) depicts an example of a read end representation in a FASTQ file. The entry consists of the read end name, the DNA sequence of the respective end of a paired-end read and the quality score for each position of the DNA sequence, which are ASCII coded. (Panel b) explains the meaning of the quality scores. From quality score qs_i at position i, we compute the probability that position i was correctly sequenced, where the *ord* function assigns an ASCII number to an input ASCII character. Before translating the resulting number $ord(qs_i)$ into the corresponding probability, one has to subtract 33, by convention. The probability that base C is at position i is equal to the probability that position i was sequenced correctly. In our model, the probability of A, T or G being at position i is equal to the probability that position i was sequenced incorrectly divided by three

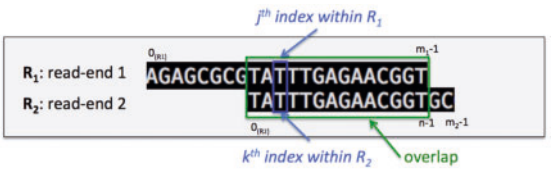
In our model, a probability matrix represents a consensus sequence, where each sequence position is represented by a probability distribution over DNA bases $\{A, C, T, G\}$. An example of a probability matrix corresponding to a consensus sequence of two overlapping sequences is depicted in (Fig. 6). At a particular position within a consensus sequence, we compute the expected probability of a base as the average probability of the respective base probabilities of the individual reads covering the position. The individual base probabilities are derived from the quality scores (Fig. 5). Let R be the set of all read ends that were joined into consensus sequence c and cover position p_c within c. The probability of a base $X \in \{A, C, T, G\}$ being at position p_c within the consensus sequence c is:

$$P^{p_c}(X) = \frac{1}{|R|} \sum_{r \in R} P_r^{p_r}(X)$$

where p_r for a read $r \in R$ is the position within r that corresponds to position p_c within the consensus sequence c. The base with the highest probability in the probability matrix at a particular position is the base of the consensus DNA sequence at the respective position.

2.4 Overlap probabilities and error correction

The computation of overlap probabilities of two overlapping sequences is an essential part of the *Snowball* algorithm. Given two overlapping sequences S_1 and S_2, represented by probability matrices (Fig. 6), where n is the length of the overlapping region, the overlap probability at position $i \in [0, \ldots, n-1]$ is computed as:



	R_1			$R_1 \cap R_2$			R_2
P(A)	$P_1^0(A)$	...	...	$[P_1^j(A) + P_2^k(A)] / \|R\|$	...	...	$P_2^{m_2-1}(A)$
P(C)	$P_1^0(C)$	...	...	$[P_1^j(C) + P_2^k(C)] / \|R\|$	...	...	$P_2^{m_2-1}(C)$
P(T)	$P_1^0(T)$	...	...	$[P_1^j(T) + P_2^k(T)] / \|R\|$	...	...	$P_2^{m_2-1}(T)$
P(G)	$P_1^0(G)$	...	...	$[P_1^j(G) + P_2^k(G)] / \|R\|$	...	...	$P_2^{m_2-1}(G)$
$\|R\|$	1	1	2	2	2	1	1

Fig. 6. Probability matrix example. In this example of a probability matrix construction, two overlapping read ends are joined into a consensus sequence and represented as a probability matrix. The subscripts of individual probabilities correspond to either read end R_1 or R_2. The superscripts of individual probabilities correspond to the positions within respective read end sequences. The probability arguments are DNA bases $\{A, C, T, G\}$. The $|R|$ values correspond to the coverage, i.e. the number of read ends covering a particular position within the consensus sequence

Downloaded from http://bioinformatics.oxfordjournals.org/ by guest on September 1, 2016

$$P^i_{\text{overlap}} = \sum_{X \in \{A, C, T, G\}} P^i_1(X) * P^i_2(X)$$

where, $P^i_1(X)$ is the probability that sequence S_1 has base X at overlap position i; probability $P^i_2(X)$ is defined analogously for sequence S_2. The overall overlap probability of S_1 and S_2 is the product of individual position overlap probabilities normalized by overlap length n (Töpfer et al., 2014):

$$P_{\text{overlap}} = \sqrt[n]{\prod_{i \in [0,.,n-1]} P^i_{\text{overlap}}}$$

As a score that represents the 'expected length' of an overlap, taking into account the individual overlap position probabilities, we compute the expected number of correct positions within the overlap as:

$$\text{Length Expected} = \sum_{i \in [0,.,n-1]} P^i_{\text{overlap}}$$

A single overlap score that enables us to rank different sequence overlaps is computed as a product of the overall overlap probability and the expected overlap length:

$$\text{Score Overlap} = P_{\text{overlap}} * \text{Length Expected}.$$

The overlap score penalizes both overlaps with low overlap probability and short overlaps, since long overlaps with high overlap probability are required. The minimum required expected length of an overlap represents the support for the overlap probability, as the overlap probability is based only on the bases within the overlap, therefore the number of the bases outside of the overlap should remain as small as possible, since we cannot make any statement about the bases outside of the overlap.

In the *Snowball* algorithm, consensus reads are iteratively joined into longer super-reads based on the overlap probabilities, expected overlap lengths and the overlap scores (Section 2.5). By default, two sequences S_1 and S_2 can be joined into a consensus sequence if the overall overlap probability is at least 0.8 and the expected length of the overlap is at least 0.5 * min[length(S_1), length(S_2)]. The high overall overlap probability ensures that the overlap consists of mostly matching positions, that there are no mismatching positions with high quality scores and that mismatches are allowed only at positions with low quality scores. For datasets with overall high quality scores, the minimum overlap probability parameter can be increased to 0.9 or 0.95. In the *Snowball* algorithm, when a consensus sequence could be joined with multiple consensus sequences with sufficient overlap probability and expected overlap length, it is joined with the sequence with which it has the highest overlap score.

2.5 The *Snowball* algorithm

For each gene domain, the *Snowball* algorithm iteratively joins consensus reads into longer error-corrected super-reads. The input of the algorithm consists of annotated consensus reads of a particular gene domain represented via probability matrices (Sections 2.1–2.3). The resulting super-reads are output as annotated contigs. Note that the method can be highly parallelized, since the *Snowball* algorithm runs for each gene domain separately.

Consensus reads are first sorted in an increasing order according to the HMM start coordinates, that denote an estimated start position of a consensus read within the multiple sequence alignment of the profile HMM. This layout suggests which pairs of consensus reads are likely to have an overlap (Fig. 7), where consensus reads that are next to each other are likely to have longer overlaps than other pairs of consensus reads.

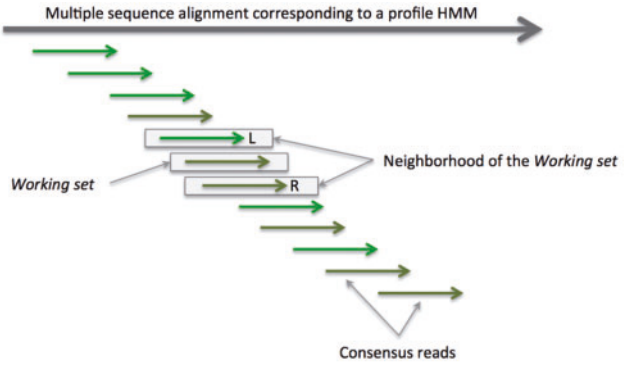


Fig. 7. Initial layout of consensus reads. Consensus reads sorted according to the HMM start coordinates. In the neighbourhood of the consensus read, that is in the *working set,* there are two closest consensus reads, one on the left (*L*) and one on the right (*R*)

As a starting point of the algorithm, we choose a consensus read with the largest sum of overlap lengths with other consensus reads and put it into the *working set*. The reason for this choice is that such a consensus read is within the highest coverage of the alignment corresponding to the respective profile HMM, where highly covered regions are likely to be covered by reads originating from similar but distinct genomes. Therefore, the chosen consensus read is very likely to overlap with consensus reads originating from distinct gene variants, which will help to resolve these gene variants early in the algorithm.

The main idea of the algorithm is that it iteratively tries to extend consensus sequences from the *working set* into longer consensus sequences by joining them with consensus reads that are in their neighbourhood, considering the consensus read layout (Fig. 7). In one iteration, first a consensus read from the neighbourhood (i.e. *L* or *R*) is joined with one of the consensus sequences from the *working set*. Second, two consensus reads (i.e. *L* and *R*) that are in the neighbourhood of the *working set* are added to the *working set* or both consensus reads from the neighbourhood of the *working set* (i.e. *L* and *R*) are joined into a consensus sequence and added to the *working set*. A consensus read and a consensus sequence (or two consensus reads) are joined only if they have a sufficient overlap as defined in the Section 2.4. If there is more than one overlap of a consensus read from the neighbourhood (i.e. *L* or *R*) and a consensus sequence from the *working set*, given that also the overlap between *L* and *R*, is sufficient, the pair that has the highest overlap score is chosen. If there is no sufficient overlap between a consensus sequence from the *working set* and a consensus read *L* or *R* in the neighbourhood and the overlap between *L* and *R* is also not sufficient, both consensus reads are added to the *working set* as they are likely to originate from distinct gene variants than the gene variants already represented in the *working set*.

Pseudo code of the algorithm:

1. Input: a list of consensus reads of a particular gene domain.
2. Sort the input list according to the HMM start coordinates in the increasing order.
3. Find a consensus read representing the starting point—as told above, a consensus read with the largest sum of overlap lengths with other consensus reads—and add it into the *working set*.
4. The neighbourhood of the *working set* consists of at most two consensus reads, one that is the closest on the left (*L*) and one that is the closest on the right (*R*) of the *working set*.
5. For each consensus sequence *S* from the *working set* and for each pair (*L*, *S*) and (*S*, *R*), and for (*L*, *R*), compute:

Downloaded from http://bioinformatics.oxfordjournals.org/ by guest on September 1, 2016

   a. overlap probability
   b. expected overlap length
   c. overlap score
6. If there is a sufficient overlap between at least one pair (L, S), (S, R) or (L, R), the pair with the highest overlap score is chosen, as defined in the Section 2.4. Let (L, S) be the pair with the highest overlap. Remove S from the *working set*. Join (L, S) into a consensus sequence (i.e. a super-read), as defined in the Section 2.3 and add it into the *working set*. Redefine L, as the first consensus read on the left of L. If (S, R) is the pair with the highest score, proceed analogously. If (L, R) is the pair with the highest score, join (L, R) into a consensus sequence (i.e. a super-read) and add it into the *working set*. Redefine L and R analogously.
7. If there is no sufficient overlap found in step (6), add L and R into the *working set* and redefine L and R in the same way as in (6).
8. If the neighbourhood is not empty, i.e. L or R was redefined at step (6) or (7), go to step (5). If L or R cannot be redefined, it is not considered in the next steps of the algorithm.
9. Output super-reads as annotated contigs.

In the algorithm, a consensus sequence is represented via a probability matrix as described in the Section 2.3. Mismatching bases within a sufficient overlap most likely represent a substitution error, where one of the bases has a relatively low quality score, thus, the base with a higher quality score corrects such a substitution error. Substitutions representing genuine strain variation are represented by overlap positions with different bases with relatively high quality scores. Therefore, such overlaps of consensus reads representing different strains almost never pass the minimum required overlap probability threshold. Consensus reads containing insertion or deletion errors have very low overlap probabilities with other consensus reads or super-reads and are therefore unlikely to be joined into longer consensus sequences. Thus, super-read positions with coverage of at least two are mostly error-corrected in terms of insertion and deletion sequencing errors.

3 Results

We evaluated *Snowball* using 21 simulated datasets, each containing 3–9 closely related *E. coli* strains and one simulated dataset containing ten novel recently published *Rhizobia* strains (Bai et al., 2015) (Section 3.4). We recall that good performance on different strains implies good performance on different species, which is why we put the emphasis on distinguishing between closely related strains in our experiments. Thereby, our aim was to answer the following questions: Were the contigs assembled correctly? How long are the resulting contigs? Did the assembly recover the reference strain sequences from which the input paired-end reads were generated? As a reference method, we used the *SAT* assembler (Zhang et al., 2014), because this is to the best of our knowledge the only currently available gene assembler of gene domains of interest for metagenomic data that does not require closely related reference genomes to be available.

In our experiments, we observed that *Snowball* was faster than *SAT*. The runtime of *Snowball* was limited by the runtime of the *HMMER 3* software, i.e. our method spent most of the runtime in this step (Section 2.2).

3.1 Per-base error

We computed the per-base error for all assembled contigs of all simulated datasets (Fig. 8). For each contig, we determined the reference strain sequence and coordinates of a particular contig sequence within a respective reference sequence from which it originates. The per-base error is defined as the percentage of bases that differ between a contig sequence and the respective sub-sequence of the reference sequence, i.e. it corresponds to the Hamming distance between the two sequences, normalized by the length of the overlap. Note, that closely related strains share large sequence regions; therefore, a contig can be well mapped onto several reference sequences of distinct strains. In this case, a reference sequence, onto which a contig maps with the lowest hamming distance, is considered to be the reference strain sequence from which it originates. If a contig maps onto several sequences of different strains, with exactly the same error, we consider it to originate from all these strains. The coverage of a contig position is equal to the number of read ends covering a respective position. In the *Snowball* algorithm, we keep track of all consensus reads that a contig consists of. For the *SAT* assembler, we have used *BWA* (Li and Durbin, 2009) to map consensus reads onto the contigs. We computed the per-base error for each coverage [3,...,30] separately. Low-coverage positions typically have a higher per-base error, as there is not enough information

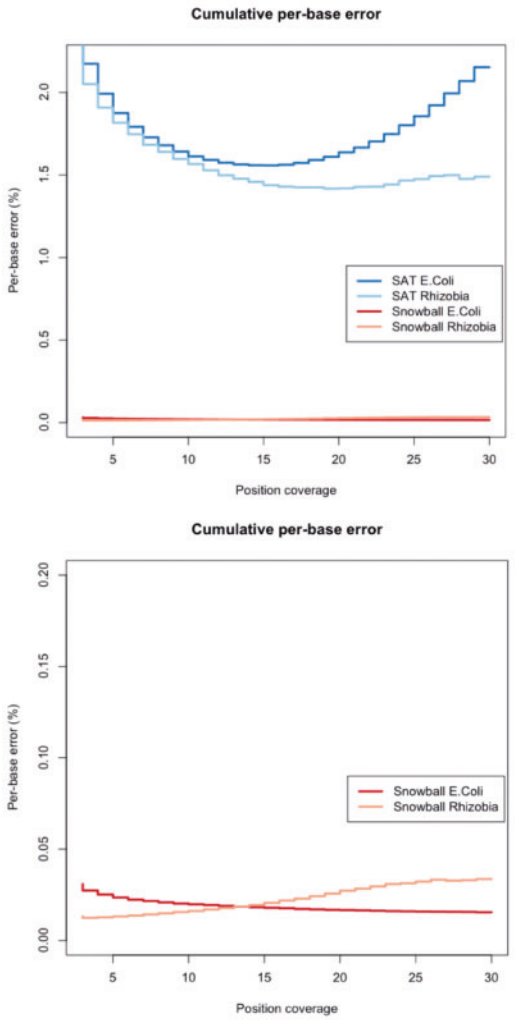


Fig. 8. Cumulative per-base error. Cumulative per-base error for the *Snowball* and *SAT* assemblers. We computed the per-base error in a cumulative way, i.e. for $X \in [3,\ldots,30]$ (on the horizontal x-axes), Y (on the vertical y-axes) is equal to the per-base error at contig positions with coverage greater or equal to X

Downloaded from http://bioinformatics.oxfordjournals.org/ by guest on September 1, 2016

available to correct sequencing errors. This is most pronounced at positions with coverage one, where the per-base error corresponds to the substitution error of a respective sequencing platform (~2.37% for our simulated datasets). At positions with higher coverage, the error-correction mechanism built into the *Snowball* algorithm yields very low (~0.02%) per-base error (Fig. 8). For the *SAT* assembler, contig positions with high coverage correspond to consensus sequences containing reads of several strains, which yields a relatively high per-base error (Fig. 8). This shows that the error-correction incorporated in the *Snowball* algorithm is indispensable for the assembly of closely related strains.

3.2 Relative contig length

We computed the average number of assembled contigs and the average cumulative length of all contigs (in Kb) per strain (Fig. 9). As the assembled contigs should cover the full length of the respective gene sequences sufficiently well, we aligned each contig to the respective profile HMM and computed the fraction of the model (i.e. the corresponding multiple sequence alignment) it covers. For each contig, this gave us an estimate of its relative length with respect to the particular profile HMM. We used this information to compute the results, e.g. using only longer contigs covering at least 50% (60%, 70%, etc.) of respective profile HMMs. This analysis showed that *Snowball* produced substantially more, longer contigs than the *SAT* assembler.

3.3 Reference coverage

We computed which parts of the reference strain sequences, from which the input reads were generated, were recovered by the assembled contigs, per strain on average (Fig. 10). As explained in the Section 3.1, assembled contigs may map onto one or more reference strain sequences with the same minimum hamming distance. We considered a contig to cover all the reference strain sequences, onto which it can be mapped with exactly the same minimum per-base error. Positions of reference sequences that are covered by at least one contig are denoted as covered positions. For each strain, we computed the number and percentage of the covered positions. Moreover, as explained in the Section 3.2, we computed these measures for contigs covering $\geq X\%$ of respective profile HMMs (where the variable X corresponds to the values on the x-axes of the graphs). The overall relatively low coverage of the reference sequences can be explained by low sequencing coverage of some of the

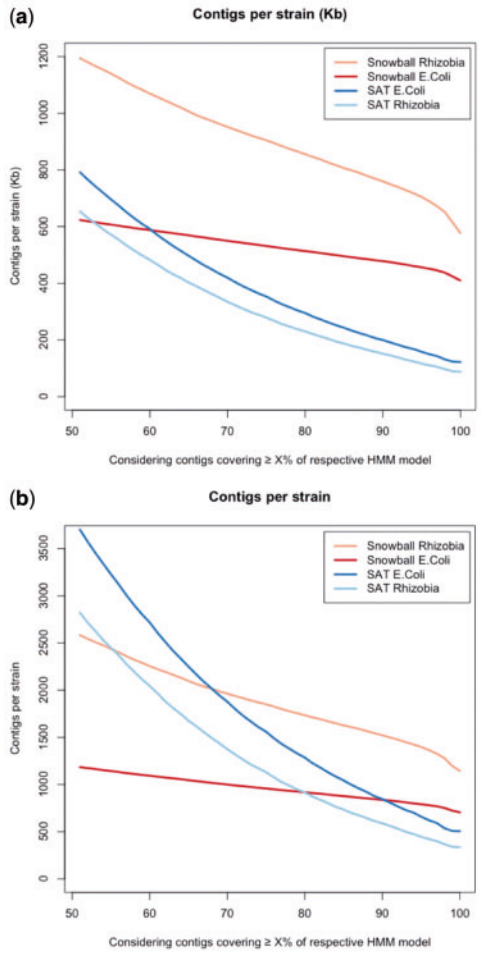


Fig. 9. Contigs per strain. Cumulative average contig length per strain, considering only contigs covering $X\%$ of respective profile HMMs (panel *a*). Average number of contigs per strain, considering only contigs covering $\geq X\%$ of respective profile HMMs (panel *b*). Here, the variable X corresponds to the values on the (horizontal) x-axes of the graphs

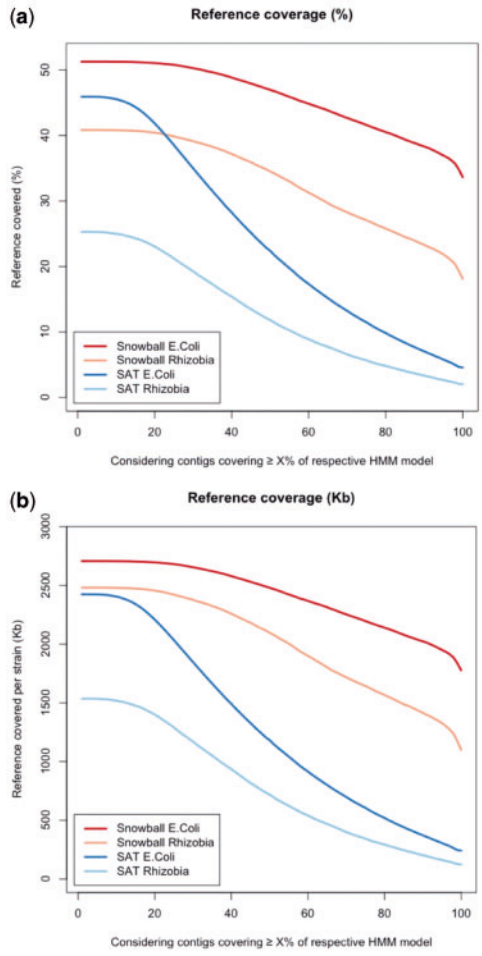


Fig. 10. Coverage of the reference strain sequences. Percentage of the recovered reference strains, per strain on average, considering only contigs covering $\geq X\%$ of respective profile HMMs (panel *a*). Corresponding absolute values (Kb) are depicted in (panel *b*). The variable X corresponds to the values on the x-axes

Downloaded from http://bioinformatics.oxfordjournals.org/ by guest on September 1, 2016

reference strain sequences (Supplementary Tables S1–S8). Also, as we only assemble coding sequences of the reference strain sequences, for which we have used profile HMMs as the input, regions of the reference strain sequences that are not covered by the profile HMMs remain unassembled. Nevertheless, this analysis showed that *Snowball* recovered substantially more reference strain sequences than the *SAT* assembler.

3.4 Simulated datasets details

We have based our evaluation on 22 simulated datasets (Table 1, Supplementary Tables S1–S8). The strain abundances correspond to randomly drawn numbers from the log-normal distribution (mean = 1, standard deviation = 2), where the numbers were limited to interval [1,. . .,50], to avoid both data explosion and extremely low strain abundances. The *ART* (Huang et al., 2012) read simulator (version 2.3.6) was employed to generate Illumina HiSeq 2500 paired-end reads (read length = 150 bp, mean insert size = 225, standard deviation = 23), where the strain coverage used for the read simulation also corresponds to the strain abundance. The abundance of a particular strain thus informs us with which coverage the strain genome within a simulated dataset was sequenced. We used the default *ART* Illumina HiSeq 2500 empirical error profile, which yields reads with ∼2.37% substitution error. For each dataset, we provide per-dataset results (Table 1, Sections 3.1–3.3) that show that *Snowball* performed substantially better than the *SAT* assembler for all simulated datasets.

4 Conclusions

We describe *Snowball*, a novel strain aware gene assembler for reconstruction of gene domains of interest from shotgun metagenomic data of microbial communities. *Snowball* performs gene assembly of individual gene domains based on read overlaps and error-correction using read quality scores at the same time, which result in very low per-base error rates. Our method uses profile HMMs to guide the assembly. Nonetheless, it does not require closely related reference genomes of the studied species to be available. We have assessed the performance of *Snowball* using 21 simulated datasets, each containing 3–9 closely related *E. coli* strains and one simulated dataset containing ten recently published *Rhizobia* strains (Bai et al., 2015), which demonstrates the capability of the *Snowball* assembler to assemble novel strains. We have compared our *Snowball* assembler to the *SAT* assembler, which, to our knowledge, establishes the current state of the art in gene assembly. The results showed that *Snowball* had substantially lower per-base error, assembled more, longer contigs and recovered more data from the input paired-end reads. We have shown that the incorporation of the error-correction mechanism is indispensable for the assemblies of closely related strains. To our knowledge, *Snowball* is the first strain aware gene assembler that does not require closely related reference genomes of the studied species to be available. The assembly of closely related strains is still a challenging task for most of the current assemblers, including the *SAT* assembler. We believe that our tool will be valuable for studying species evolution (e.g. genes under selection) and strain or gene diversity (e.g. virulence genes). *Snowball* is implemented in Python, runs on a user-defined number of processor cores

Table 1. Overview of simulated datasets

Dataset	Strains per dataset	Per-base error (%) at position coverage ≥5[a]		Contig length (Kb) 75% HMM model[b]		Ref. cov. 75% HMM model (%)[c]	
		Snowball	SAT	Snowball	SAT	Snowball	SAT
1	3	0.019	1.613	913	229	41.3	7.5
2		0.035	1.823	1080	628	44.4	15.1
3		0.006	1.603	865	186	43.0	6.7
4	4	0.036	1.666	740	306	43.1	10.7
5		0.011	1.813	691	253	42.6	9.7
6		0.007	1.648	700	303	45.5	11.2
7	5	0.012	1.809	614	408	44.9	13.5
8		0.012	1.791	622	393	44.8	13.5
9		0.022	2.064	665	411	40.9	12.6
10	6	0.022	1.853	518	378	42.1	11.8
11		0.045	1.822	557	308	39.0	10.7
12		0.033	2.009	571	407	40.2	12.4
13	7	0.028	1.861	447	316	42.6	11.7
14		0.041	1.866	496	293	38.9	10.9
15		0.018	2.034	488	367	41.7	12.0
16	8	0.017	2.152	408	443	44.6	12.7
17		0.030	1.869	428	294	38.3	10.5
18		0.038	2.227	453	440	39.3	11.6
19	9	0.019	1.884	349	265	40.9	9.7
20		0.014	2.035	360	314	40.4	10.7
21		0.044	2.270	424	430	42.2	13.8
22	10	0.013	1.909	905	279	27.0	5.7

[a]Per-base error (%) at contig positions with coverage ≥5 (Fig. 8).

[b]Cumulative contig length (Kb) at $X = 75$ of (Fig. 9a).

[c]Percentage of recovered data at $X = 75$ of (Fig. 10a). Datasets 1–21 consist of *E. coli* strains (Supplementary Table S1–S7). Dataset 22 consists of *Rhizobia* strains (Supplementary Table S8).

Downloaded from http://bioinformatics.oxfordjournals.org/ by guest on September 1, 2016

in parallel, runs on a standard laptop and can be easily installed under Linux or OS X.

Acknowledgements

The authors would like to thank David Laehnemann and Andreas Bremges for their valuable feedback on the manuscript; and Rubén Garrido Oter for providing the novel *Rhizobia* strains.

Funding

IG and ACM were funded by the Heinrich Heine University Düsseldorf and the Helmholtz Center for Infection Research. AS was funded by the Netherlands Organization for Scientific Research (NWO), through Vidi grant No. 639.072.039.

Conflict of Interest: The authors have declared that no competing interests exist.

References

Ahn,T.H. *et al.* (2015) Sigma: strain-level inference of genomes from metagenomic analysis for biosurveillance. *Bioinformatics*, **31**, 170–177.

Bai,Y. *et al.* (2015) Functional overlap of the Arabidopsis leaf and root microbiota. *Nature*, **528**, 364–369.

Boisvert,S. *et al.* (2012) Ray Meta: scalable de novo metagenome assembly and profiling. *Genome Biol.*, **13**, R122.

Butler,J. *et al.* (2008) ALLPATHS: de novo assembly of whole-genome shotgun microreads. *Genome Res.*, **18**, 810–820.

Cole,S. and Saint-Girons,I. (1999) Bacterial genomes—all shapes and sizes. In: Charlebois,R. (ed), *Organization of the Prokaryotic Genome*. ASM Press, Washington, DC, pp. 35–62.

Eddy,S.R. (2011) Accelerated profile HMM searches. *PLoS Comput. Biol.*, **7**, e1002195.

Finn,R.D. *et al.* (2014) Pfam: the protein families database. *Nucleic Acids Res.*, **42**, D222–D230.

Huang,W. *et al.* (2012) ART: a next-generation sequencing read simulator. *Bioinformatics*, **28**, 593–594.

Kingsford,C. *et al.* (2010) Assembly complexity of prokaryotic genomes using short reads. *BMC Bioinformatics*, **11**, 21.

Kunin,V. *et al.* (2008) A bioinformatician's guide to metagenomics. *Microbiol. Mol. Biol. Rev.*, **72**, 557–578.

Laehnemann,D. *et al.* (2015) Denoising DNA deep sequencing data-high-throughput sequencing errors and their correction. *Brief. Bioinform.*, **17**, 154–179.

Li,D. *et al.* (2015) MEGAHIT: an ultra-fast single-node solution for large and complex metagenomics assembly via succinct de Bruijn graph. *Bioinformatics*, **31**, 1674–1676.

Li,H. and Durbin,R. (2009) Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics*, **25**, 1754–1760.

Luo,R. *et al.* (2012) SOAPdenovo2: an empirically improved memory-efficient short-read de novo assembler. *GigaScience*, **1**, 18.

Marschall,T. *et al.* (2016) Computational pan-genomics: status, promises and challenges. *BioRxiv*, **043430**, 1–33.

Minoche,A.E. *et al.* (2011) Evaluation of genomic high-throughput sequencing data generated on Illumina HiSeq and genome analyzer systems. *Genome Biol.*, **12**, R112.

Namiki,T. *et al.* (2012) MetaVelvet: an extension of Velvet assembler to de novo metagenome assembly from short sequence reads. *Nucleic Acids Res.*, **40**, e155.

Peng,Y.Y. *et al.* (2012) IDBA-UD: a de novo assembler for single-cell and metagenomic sequencing data with highly uneven depth. *Bioinformatics*, **28**, 1420–1428.

Riesenfeld,C.S. *et al.* (2004) Metagenomics: genomic analysis of microbial communities. *Annu. Rev. Genet.*, **38**, 525–552.

Schirmer,M. *et al.* (2015) Insight into biases and sequencing errors for amplicon sequencing with the Illumina MiSeq platform. *Nucleic Acids Res.*, **43**, e37.

Simpson,J.T. and Durbin,R. (2012) Efficient de novo assembly of large genomes using compressed data structures. *Genome Res.*, **22**, 549–556.

Töpfer,A. *et al.* (2014) Viral quasispecies assembly via maximal clique enumeration. *PLoS Comput. Biol.*, **10**, e1003515.

Wu,M. and Scott,A.J. (2012) Phylogenomic analysis of bacterial and archaeal sequences with AMPHORA2. *Bioinformatics*, **28**, 1033–1034.

Yuan,C. *et al.* (2015) Reconstructing 16S rRNA genes in metagenomic data. *Bioinformatics*, **31**, i35–i43.

Zagordi,O. *et al.* (2011) ShoRAH: estimating the genetic diversity of a mixed sample from next-generation sequencing data. *BMC Bioinformatics*, **12**, 119.

Zhang,Y. *et al.* (2014) A scalable and accurate targeted gene assembly tool (SAT-Assembler) for next-generation sequencing data. *PLoS Comput. Biol.*, **10**, e1003737.

Downloaded from http://bioinformatics.oxfordjournals.org/ by guest on September 1, 2016

Snowball: Strain aware gene assembly of metagenomes

I. Gregor, A. Schönhuth, A. C. McHardy

Supporting Tables S1-S8

Dataset	Reads (Mbp)	[a]Dataset Strains (Accessions)	[b]Strain coverage	[c]Ref. Strain (Mbp)
1	159	NZ_AKLX00000000	15.6	5.489
		NZ_ANLR00000000	10.8	5.192
		NZ_AKLB00000000	3.5	5.384
2	323	NZ_AEZS00000000	19.7	5.450
		NZ_AIGZ00000000	15.1	5.125
		NZ_AIGV00000000	26.6	5.216
3	138	NZ_AIHQ00000000	9.6	5.201
		NZ_AIHO00000000	5.5	5.246
		NZ_AIHS00000000	11.3	5.230

Supporting Table 1. Parameters of three simulated datasets, each containing 3 *E. coli* strains. (column a) Accession numbers of individual strains of simulated datasets. (column b) Strain coverage of respective strains in the datasets. (column c) Size of individual reference strains in Mbp.

Dataset	Reads (Mbp)	[a]Dataset Strains (Accessions)	[b]Strain coverage	Ref. Strain (Mbp)
4	223	NZ_AKLR00000000	16.4	5.479
		NZ_AMTI00000000	8.6	5.426
		NZ_ANMA00000000	5.7	5.127
		NZ_AMTZ00000000	10.9	5.410
5	192	NZ_AIEZ00000000	11.5	5.154
		NZ_AIFB00000000	13.8	5.298
		NZ_AFJB00000000	3.5	5.134

		NZ_AIEW00000000	8.0	5.198
6	216	NZ_AIHQ00000000	12.9	5.201
		NZ_AIHR00000000	3.4	5.210
		NZ_AIHP00000000	6.2	5.265
		NZ_AIHS00000000	18.9	5.230

Supporting Table 2. Parameters of three simulated datasets, each containing 4 *E. coli* strains.

Dataset	Reads (Mbp)	[a]Dataset Strains (Accessions)	[b]Strain coverage	Ref. Strain (Mbp)
7	351	NZ_AMUR00000000	4.9	5.376
		NZ_AMTU00000000	20.1	5.298
		NZ_AMTJ00000000	19.6	5.434
		NZ_AKMA00000000	3.3	5.552
		NZ_AKMK00000000	17.7	5.417
8	336	NZ_ANLX00000000	23.6	5.373
		NZ_AOEI00000000	3.7	5.164
		NZ_ANLU00000000	8.0	5.341
		NZ_ABHS00000000	11.6	5.933
		NZ_AOEG00000000	15.7	5.177
9	343	NZ_AIFW00000000	2.3	5.324
		NZ_AFAH00000000	16.7	5.012
		NZ_AIFZ00000000	14.1	5.077
		NZ_AHAX00000000	10.8	4.956
		NZ_AMTG00000000	26.0	4.725

Supporting Table 3. Parameters of three simulated datasets, each containing 5 *E. coli* strains.

Dataset	Reads (Mbp)	[a]Dataset Strains (Accessions)	[b]Strain coverage	Ref. Strain (Mbp)
10	380	NZ_AMTY00000000	37.3	5.477
		NZ_ABHP00000000	2.1	5.656
		NZ_AMVF00000000	15.4	5.411
		NZ_AMTM00000000	3.5	5.131
		NZ_ABHQ00000000	7.4	5.706

		NZ_AKLV00000000	4.3	5.447
11	323	NZ_AEZS00000000	1.3	5.450
		NZ_AEZT00000000	5.0	5.279
		NZ_AIGW00000000	14.7	5.231
		NZ_AIGX00000000	24.7	5.551
		NZ_AAJX00000000	4.9	5.427
		NZ_AIGY00000000	9.3	5.305
12	409	NZ_AIFW00000000	12.4	5.324
		NZ_AIFZ00000000	2.4	5.077
		NZ_AFAD00000000	25.1	4.744
		NZ_AIFY00000000	13.4	4.982
		NZ_AFAH00000000	3.7	5.012
		NZ_AHAW00000000	25.1	5.107

Supporting Table 4. Parameters of three simulated datasets, each containing 6 *E. coli* strains.

Dataset	Reads (Mbp)	[a]Dataset Strains (Accessions)	[b]Strain coverage	Ref. Strain (Mbp)
13	368	NZ_AMTP00000000	3.2	5.741
		NZ_ANMB00000000	7.8	5.344
		NZ_AKMF00000000	4.3	5.319
		NZ_ANLV00000000	1.8	5.346
		NZ_AMTM00000000	16.0	5.131
		NZ_AMUW00000000	7.3	5.375
		NZ_AKLR00000000	28.8	5.479
14	356	NZ_AIGW00000000	14.5	5.231
		NZ_AIGY00000000	5.3	5.305
		NZ_AAJV00000000	16.1	5.528
		NC_013353	8.8	5.449
		NZ_AFAA00000000	1.3	5.871
		NZ_AIHB00000000	16.2	5.489
		NZ_AKNI00000000	3.5	5.242
15	437	NC_012759	9.7	4.578
		NZ_AIFW00000000	10.2	5.324
		NZ_AMTG00000000	39.9	4.725
		NZ_AFAD00000000	16.0	4.744
		NZ_AMTH00000000	2.2	4.729

		NZ_AIFV00000000	2.7	5.390
		NZ_AHAW00000000	9.7	5.107

Supporting Table 5. Parameters of three simulated datasets, each containing 7 *E. coli* strains.

Dataset	Reads (Mbp)	[a]Dataset Strains (Accessions)	[b]Strain coverage	Ref. Strain (Mbp)
16	618	NZ_AOER00000000	16.3	5.268
		NZ_AKKX00000000	3.0	5.528
		NZ_AMTE00000000	5.7	5.322
		NC_013008	4.5	5.528
		NZ_AKML00000000	5.8	5.409
		NZ_AIFF00000000	47.4	5.503
		NZ_AKKZ00000000	2.7	5.544
		NZ_AMTF00000000	29.3	5.345
17	393	NZ_AIGN00000000	6.0	5.486
		NZ_AFAI00000000	16.6	5.560
		NZ_AIGM00000000	2.0	5.195
		NZ_AMVC00000000	6.8	5.609
		NZ_AIGI00000000	1.5	5.460
		NZ_AEZV00000000	4.5	5.409
		NZ_AIGK00000000	28.0	5.408
		NZ_AIGL00000000	6.9	5.362
18	576	NZ_AIFX00000000	45.2	5.118
		NC_012759	37.3	4.578
		NZ_AFAB00000000	9.9	5.639
		NZ_AIFY00000000	4.1	4.982
		NZ_AFAD00000000	3.3	4.744
		NZ_AHAX00000000	1.8	4.956
		NZ_AFAH00000000	7.4	5.012
		NZ_AIFZ00000000	7.1	5.077

Supporting Table 6. Parameters of three simulated datasets, each containing 8 *E. coli* strains.

Dataset	Reads (Mbp)	[a]Dataset Strains (Accessions)	[b]Strain coverage	Ref. Strain (Mbp)

19	396	NZ_AMVA00000000	16.3	5.328
		NZ_AMTZ00000000	1.4	5.410
		NZ_AKMD00000000	5.2	5.336
		NZ_ANLX00000000	3.1	5.373
		NZ_AKKY00000000	3.5	5.438
		NZ_AKML00000000	36.8	5.409
		NZ_AKLQ00000000	4.3	5.525
		NZ_AKMJ00000000	2.2	5.403
		NZ_AKMF00000000	1.5	5.319
20	495	NZ_AMTY00000000	1.2	5.477
		NZ_AMTV00000000	1.3	5.430
		NZ_AMUR00000000	14.2	5.376
		NZ_ABHM00000000	3.9	5.618
		NZ_ABHP00000000	40.8	5.656
		NZ_AMTN00000000	6.9	5.400
		NZ_AOEN00000000	1.8	5.202
		NZ_AKLS00000000	14.6	5.396
		NZ_AMTJ00000000	5.4	5.434
21	675	NZ_AIFW00000000	15.5	5.324
		NZ_AFAD00000000	27.3	4.744
		NZ_AIFX00000000	14.1	5.118
		NZ_AIFZ00000000	21.1	5.077
		NZ_AHAW00000000	25.6	5.107
		NZ_AMTH00000000	4.3	4.729
		NC_012759	2.0	4.578
		NZ_AHAX00000000	10.6	4.956
		NZ_AIFV00000000	13.3	5.390

Supporting Table 7. Parameters of three simulated datasets, each containing 9 *E. coli* strains.

Dataset	Reads (Mbp)	[a]Dataset Strains (Accessions)	[b]Strain coverage	Ref. Strain (Mbp)
22	832	GCA_001424085.1	15.3	6.585
		GCA_001424505.1	5.1	6.561
		GCA_001425605.1	1.9	6.368
		GCA_001424965.1	26.2	6.584
		GCA_001424985.1	3.0	5.351

		GCA_001426665.1	8.7	6.311
		GCA_001428925.1	16.5	5.351
		GCA_001426565.1	7.4	5.266
		GCA_001426685.1	17.5	6.043
		GCA_001429075.1	33.3	6.319

Supporting Table 8. Parameters of a simulated dataset containing 10 *Rhizobia* strains.

PhyloPythiaS+: a self-training method for the rapid reconstruction of low-ranking taxonomic bins from metagenomes

Ivan Gregor[1,2,3], Johannes Dröge[1,2,3], Melanie Schirmer[4], Christopher Quince[5] and Alice C. McHardy[1,2,3]

[1] Max-Planck Research Group for Computational Genomics and Epidemiology, Max-Planck Institute for Informatics, Saarbrücken, Germany
[2] Department of Algorithmic Bioinformatics, Heinrich-Heine-University Düsseldorf, Düsseldorf, Germany
[3] Computational Biology of Infection Research, Helmholtz Center for Infection Research, Braunschweig, Germany
[4] The Broad Institute of MIT and Harvard, Cambridge, MA, United States
[5] School of Engineering, University of Glasgow, Glasgow, United Kingdom

ABSTRACT

Background. Metagenomics is an approach for characterizing environmental microbial communities *in situ*, it allows their functional and taxonomic characterization and to recover sequences from uncultured taxa. This is often achieved by a combination of sequence assembly and binning, where sequences are grouped into 'bins' representing taxa of the underlying microbial community. Assignment to low-ranking taxonomic bins is an important challenge for binning methods as is scalability to Gb-sized datasets generated with deep sequencing techniques. One of the best available methods for species bins recovery from deep-branching phyla is the expert-trained *PhyloPythiaS* package, where a human expert decides on the taxa to incorporate in the model and identifies 'training' sequences based on marker genes directly from the sample. Due to the manual effort involved, this approach does not scale to multiple metagenome samples and requires substantial expertise, which researchers who are new to the area do not have.

Results. We have developed *PhyloPythiaS+*, a successor to our *PhyloPythia(S)* software. The new (+) component performs the work previously done by the human expert. *PhyloPythiaS+* also includes a new *k*-mer counting algorithm, which accelerated the simultaneous counting of 4–6-mers used for taxonomic binning 100-fold and reduced the overall execution time of the software by a factor of three. Our software allows to analyze Gb-sized metagenomes with inexpensive hardware, and to recover species or genera-level bins with low error rates in a fully automated fashion. *PhyloPythiaS+* was compared to *MEGAN*, *taxator-tk*, *Kraken* and the generic *PhyloPythiaS* model. The results showed that *PhyloPythiaS+* performs especially well for samples originating from novel environments in comparison to the other methods.

Availability. *PhyloPythiaS+* in a virtual machine is available for installation under Windows, Unix systems or OS X on: https://github.com/algbioi/ppsp/wiki.

Subjects Bioinformatics, Computational Biology, Genomics, Taxonomy
Keywords Metagenomics, Taxonomic classification, Machine learning, Bioinformatics

Submitted 14 October 2015
Accepted 24 December 2015
Published 8 February 2016

Corresponding author
Alice C. McHardy,
Alice.McHardy@helmholtz-hzi.de

Academic editor
Jonathan Eisen

Additional Information and Declarations can be found on page 17

DOI **10.7717/peerj.1603**

Copyright
2016 Gregor et al.

Distributed under
Creative Commons CC-BY 4.0

OPEN ACCESS

INTRODUCTION

Metagenomics is the functional or sequence-based analysis of microbial DNA isolated directly from a microbial community of interest (*Riesenfeld, Schloss & Handelsman, 2004*; *Kunin et al., 2008*). As the cultivation conditions for most microorganisms are unknown or too complex to reproduce in the laboratory (*Hugenholtz, 2002*), random shotgun and amplicon-sequencing based metagenome studies have led to substantial advances in our understanding of the structure and functions of microbial communities within the last decade (*Kalyuzhnaya et al., 2008*; *Turnbaugh et al., 2010*; *Hess et al., 2011*; *Pope et al., 2011b*; *Zarowiecki, 2012*; *Schloissnig et al., 2013*; *Blaser et al., 2013*). The taxonomic classification or 'binning' of metagenome samples is often performed after sequence assembly (*Peng et al., 2011*; *Laserson, Jojic & Koller, 2011*; *Boisvert et al., 2012*; *Namiki et al., 2012*; *Pell et al., 2012*). This is a computationally demanding task, which for metagenome samples results in a mixture of sequence fragments of varying lengths, originating from the different microbial community members. A taxonomic binning defines 'bins' of sequence fragments that were assigned the same taxonomic identifier, representing draft genomes or pan-genomes of the different microbial community members. Taxonomic binning methods use sequence homology, sequence composition and similarities of contigs in read coverage or gene counts, see *Dröge & McHardy (2012)* for a recent review. The subsequent analysis of these bins allows characterizing the functional and metabolic potential for individual taxa. For instance, in a collaboration with Mark Morrison's group, a functional and metabolic analysis of a draft genome recovered by taxonomic binning from the gut of the Australian Tammar Wallaby metagenome led to the isolation and subsequent characterization of a new and previously uncultivated bacterium (*Pope et al., 2011b*). Different from binning methods, taxonomic profiling tools (*Wu & Eisen, 2008*; *Stark et al., 2009*; *Liu et al., 2011*; *Meinicke, Asshauer & Lingner, 2011*; *Wu & Scott, 2012*; *Segata et al., 2012*; *Sunagawa et al., 2013*; *Silva et al., 2013*) return a taxonomic profile for a metagenome sample to represent the taxonomic composition of the underlying sampled community.

Composition-based binning methods assign metagenome sequences based on their *k*-mer signature, which is derived from the counts of short oligomers (*k*-mers) for a sequence (*Karlin & Burge, 1995*; *Deschavanne et al., 1999*). Our previously developed *PhyloPythia(S)* (*PPS*) (*McHardy et al., 2007*; *Patil, Roune & McHardy, 2011*) software uses this information in combination with a structured output support vector machine framework for taxonomic classification. Composition-based signatures are global genomic properties, which can be estimated from any sufficiently sized sequence sample for a taxon; e.g., for *PP(S)*, 100 kb of reference sequences for a taxon are sufficient for accurate assignment, also for low ranking taxa. Thus, no complete genome sequences of related organisms are required for assignment, which is often a limiting factor for the homology-based methods. Composition-based methods are very fast, with classification runtimes increasing linearly with the size of the sequence sample, whereas the runtime of alignment-based methods is proportional to the product of the reference collection size and the sequence sample size. As the current sequencing technologies produce Gb-sized metagenome samples (*Metzker, 2010*; *Loman et al., 2012*), scalability and computational efficiency are becoming

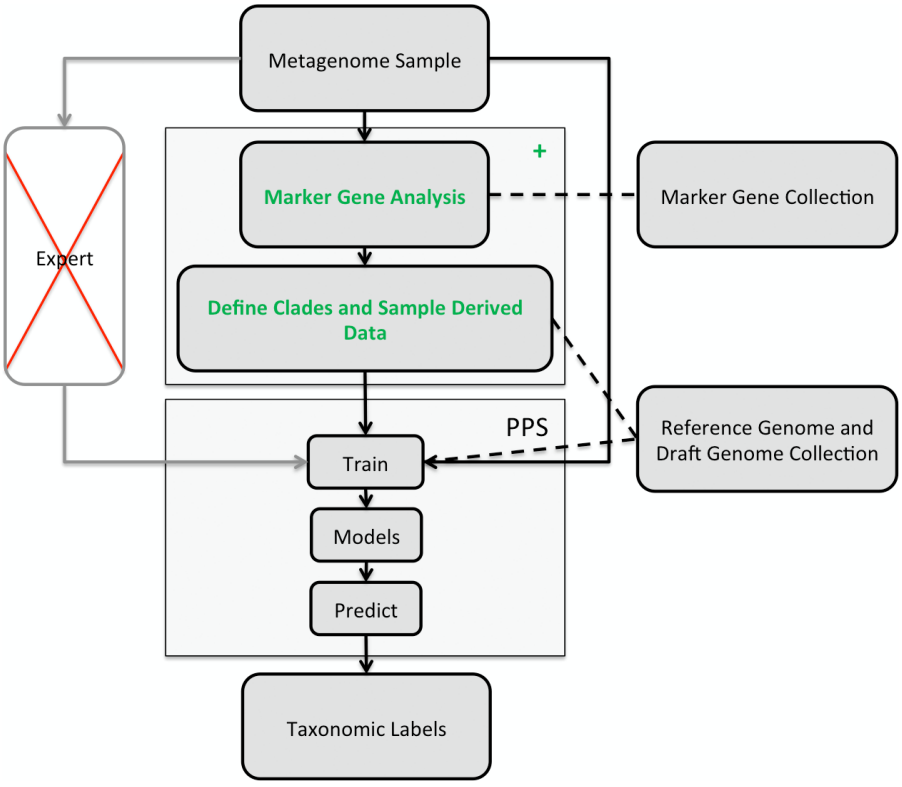


Figure 1 Illustration of the *PhyloPythiaS+* workflow. The recommended use of *PPS* is that a human expert specifies the taxa to incorporate in a composition-based taxonomic metagenome classifier and identifies the relevant 'training' sequences based on marker genes directly from the sample. The inclusion of contigs originating directly from members of the microbial community, as 'training' sequences, is very important for achieving good classification accuracy, as many members of microbial communities are underrepresented in public sequence collections. In *PPS+*, the step of deciding which taxa to include in the model and defining suitable 'training' sequences was automated in the + component, based on marker genes, genome and draft genome sequence collections. The data generated by the + component are then used to build the *PPS* models, that are subsequently used to generate the taxonomic binning of the entire metagenome sequence sample.

increasingly important for computational metagenomic methods. Therefore, we have developed a fully automated taxonomic binning software, that can rapidly process large metagenome samples. *PhyloPythiaS+* (*PPS+*) is the successor to our previously described *PPS* software and improves on it in several important ways. We provide an automated marker-gene based framework for design and creation of sample-derived structured output support vector machine models, which allows the generation of accurate sample-derived models without user intervention or expert knowledge. *PPS+* is the first tool that combines taxonomic profiling and subsequent taxonomic composition based binning of the whole metagenome sample, which is particularly valuable for the draft genome reconstruction of taxa from deep-branching phyla. By implementation of a faster *k*-mer counting algorithm, we substantially increased its throughput to 0.5 Gb/h. *PPS+* is distributed in a virtual machine which facilitates installation under all common operating systems and runs on inexpensive hardware available to most users.

METHODS

The classification of a shotgun metagenome sequence sample with *PPS+* proceeds in two phases (Fig. 1): In the first phase, the newly developed (+) component identifies sample-derived training sequences and the taxa to be modeled by searching for copies of 34 ubiquitous taxonomic marker genes in the metagenome sample. The marker gene analysis results in taxonomic assignments for a small fraction of the sample. Based on the taxa abundance profile derived from these assignments and the sequences available in the reference sequence collections, our method determines which taxa will be modeled and which are the sample-derived data that will be used for training *PPS*.

The second phase is the composition-based taxonomic assignment of the entire metagenome sample using *PPS* models trained using the data generated in the first phase. *PPS* models can be reused to classify further metagenome samples, e.g., additional samples from the same community.

PhyloPythiaS

Assignment with *PPS* proceeds in two steps: In the training step, an ensemble of structured output Support Vector Machines (SVMs) (*Joachims, Finley & Yu, 2009*) for the specified part of the NCBI taxonomy, defined by the taxa being modeled, are trained using the sample-derived training sequences and additional data for these taxa from a customized reference collection of sequenced genomes and draft genomes (Suplemental Information 1, Section 3.3). The list of modeled taxa and sample-derived data are generated with the + component of *PPS+*. The list of taxa restricts the taxonomic output space that is modeled, i.e., a sequence from a metagenome sample will be assigned to a leaf node taxon or a corresponding higher-ranking taxon of the learned taxonomy.

In the prediction step, the *PPS* model ensemble identifies the taxon which best matches a query sequence in terms of its *k*-mer profile and assigns to it the respective taxonomic identifier. By default, sequences of 1 kb or more are classified (*PPS+* configuration parameter: *minSeqLen*).

The + component of *PhyloPythiaS+*

The input for the + component of *PhyloPythiaS+* is the metagenome sample. This step returns a list of clades and sample-derived data for the subsequent *PPS* training. The + component performs the following steps:

(1) *Marker gene identification*: DNA sequences from the sample are translated in all six reading frames (i.e., also considering reverse complement sequences) to protein sequences. In both the translated and untranslated sequences, regions with similarity to the DNA or protein Hidden Markov model (HMM) profiles of 34 taxonomically informative marker genes (*Wu & Eisen, 2008*; *Stark et al., 2009*; *Liu et al., 2011*; *Wu & Scott, 2012*; *Segata et al., 2012*; *Sunagawa et al., 2013*) are identified (Supplemental Information 1, Section 3.3 and 6.1). The corresponding DNA marker gene sequences from these regions are used for further analysis.

(2) *Taxonomic marker gene assignment*: The marker gene sequences are assigned a taxonomic identifier using the composition-based Naïve Bayes classifier (*Schloss et al., 2009*) (Supplemental Information 1, Section 6.2).

(3) *Taxonomic sequence assignment*: If a sequence contains multiple marker genes, multiple taxonomic identifiers are identified in Step 2. Then the highest bootstrap confidence score (*hcs*) returned by the Naïve Bayes classifier (NBC) for one of the markers on the fragment is identified. We use all marker gene assignments with confidence scores larger than $hcs * (1 - candidatePlTopPercentThreshold)$. The default setting for the configuration parameter *candidatePlTopPercentThreshold* is 0.1. From the set of taxonomic identifiers, the lowest taxon t is identified for which all other assignments are either to the same taxon t or defined at higher-ranking parental taxa of t. Taxon t is consequently used for the overall fragment assignment. The confidence score for the fragment is set to the smallest confidence score for the set of retained marker gene assignments.

(4) *(Optional: Taxonomic scaffold assignment):* Scaffolding information (i.e., the mapping of contigs to scaffolds) can be used to obtain more training data for the relevant taxa. Assembled contigs can be grouped into scaffolds based on the paired-end information after the assembly. As all contigs of a particular scaffold originate from the same strain, all contigs of the respective scaffold should have the same taxonomic label. Here, we make use of this scaffolding information, such that unassigned contigs of a particular scaffold can be assigned based on the assigned contigs of the respective scaffold. In the first step, the taxonomic identifiers of all assigned contigs for a scaffold are corrected as follows: Let us consider that n taxonomically assigned contigs of a scaffold are placed along a common path from the root r down to a low-ranking clade lc in the reference taxonomy. The unassigned contigs of a scaffold are not among these n contigs. To obtain a consistent assignment for all the contigs of a scaffold and to correct for 'outlier' contig assignments to low ranking taxa, contigs are reassigned according to the following: All n assigned contigs of the respective scaffold are reassigned to the lowest taxon c, which lies on the path from r to lc, where c is chosen such that at least $(agThreshold * n)$ of the contigs are assigned on the path from c to lc. In the second step, unassigned contigs are assigned to the same taxon c, if a sufficient number of contigs have already been assigned. Let us denote the sum of all contig lengths for a scaffold as l and the sum of all assigned contig lengths of the respective scaffold as al. If $al/l \geq assignedPartThreshold$, then the unassigned contigs of a scaffold are also assigned to clade c (see the configuration parameters: $placeContigsFromTheSameScaffold = True$, $agThreshold = 0.3$, $assignedPartThreshold = 0.5$).

(5) *Assignment path truncation*: Contigs assigned to a lower-ranking taxon than the specified lowest rank are reassigned to the parental taxon of this lowest rank (configuration parameter: *rankIdCut*).

(6) *Taxa selection for model specification*: Any taxon for which at least 100 kb of sample-derived data have been identified can be modeled. Furthermore, species can be modeled if at least 300 kb of reference sequences are available in the reference sequence database, and higher-ranking taxa can be modeled if data for at least three distinct species with

this requirement (>300 kb per species) are available. Contigs assigned to taxa for which there are fewer data are subsequently assigned to higher taxonomic ranks for which sufficient data are available to allow their use as sample-derived training data (configuration parameters: *minGenomesWgs = 3 or 1*, *minBpPerSpecies = 300,000*, *minBpToModel = 100,000*).

(7) *Abundant taxa selection*: To reduce the number of taxa to the most relevant ones, the least abundant taxon is removed iteratively. This is defined as the taxon to which the minimum number of bp is assigned. Sequences assigned to this taxon are reassigned to the closest defined taxon at a parental rank. The algorithm ends when the number of leaf taxa is less than or equal to the maximum number of taxa to be modeled (configuration parameter: *maxLeafClades = 50*; this can be set realistically up to 800).

Balancing training data: The part of the taxonomy that will be modeled with *PPS* is defined by the taxa identified in the previous step. It has leaf nodes at different ranks above the specified rank cut-off, and internal nodes. Only leaf node taxa and sample-derived training data assigned to leaf node taxa in the preceding steps are specified as input for *PPS* training. To balance the training data across clades, a maximum of 400 kb of sample-derived training data are selected for each leaf node taxon (configuration parameter: *maxSSDfileSize*). For this selection, contigs are used in order of decreasing confidence values and then in order of decreasing length. The balancing of training data can be switched off by setting the configuration parameter (*maxSSDfileSize*) to a large number.

Simultaneous counting of multiple short *k*-mers

We provide *PPS+* with a new custom *k*-mer counting algorithm that is based on the Rabin Karp string matching algorithm (*Karp & Rabin, 1987*). The algorithm is highly optimized to count occurrences of short DNA sequences. It is very fast, as it is memory efficient, because it does not need any large helper data structure similar to suffix trees. It explores the locality of reference, uses very fast bit shift operations and is efficiently implemented in C. Its complexity is $O(n)$, where n is the length of the DNA sequence that is being considered. It enumerates *k*-mers up to hundred times faster than when using suffix trees that were employed in *PPS*. This made *PPS+* overall up to 3x faster than *PPS*. Because the algorithm allows to simultaneously enumerate *k*-mers of consecutive lengths in one run, it is at least 2–7x faster than the state-of-the-art software *Jellyfish* (*Marcais & Kingsford, 2011*) and 11x faster than *KAnalyze* (*Audano & Vannberg, 2014*) in the scenario used in *PPS+*, i.e., when calculating *k*-mers of length 4, 5, and 6 for every sequence (Table S1, Supplemental Information 1, Section 2). We also found that the state-of-the-art *k*-mer counting methods *KMC 2* (*Deorowicz et al., 2015*) and *Turtle* (*Roy, Bhattacharya & Schliep, 2014*) are not applicable to our problem setting, as *KMC 2* can count only *k*-mers ≥ 10 and *Turtle* is prohibitively slow for sequences ≥ 16 kb.

Algorithm description

Let us assume that we are given an array a, which represents a DNA sequence of length n where all letters are encoded as numbers *0, 1, 2, 3* (where *A ~0, T ~1, G ~2, C~3*) and let $a_0, \ldots, a_{n-1}$ denote the respective entries. We would like to count the occurrences of all

k-mers of length k and store the counts in an array c of length 4^k, which is initialized by zeros. Each k-mer maps to a unique index in the array c. The index of the first k-mer in our sequence is calculated according to:

$$\text{index}_0 = a_0 * 4^{k-1} + a_1 * 4^{k-2} + \cdots + a_{k-2} * 4^1 + a_{k-1} * 4^0.$$

The index of the $(i+1)$th k-mer of the sequence is computed from the (i)th index as:

$$\text{index}_{i+1} = (\text{index}_i - a_i * 4^{k-1}) * a_{i+k} * 4^0.$$

When an index is identified, the corresponding k-mer count at this index position in array c is incremented by one. For instance, the DNA sequence *ATGCATG* is encoded in array a as [0, 1, 2, 3, 0, 1, 2]. For $k = 2$, we would add two counts for the k-mer AT in array c at the index position $0 * 4 + 1 = 1$, two counts for TG at the index position $1 * 4 + 2 = 6$, one count for GC at the index position $2 * 4 + 3 = 11$ and one count for CA at index position $3 * 4 + 0 = 12$. The multiplication operation $X * 4^m$ can be computed using the bit shift operation $X \ll 2 * m$, which is usually much faster than multiplication.

Counting k-mers of different lengths at once

If $index_i$ is the index of the ith k-mer of length k, the index of the ith $(k-j)$-mer (of length $k-j$) can be simultaneously computed using the bit shift operation as $index_i \gg (2 * j)$ (for $j \in [1..k-1]$) and the corresponding counter at the computed index of a respective counter array of length $4^{(k-j)}$ is incremented. The end of a DNA sequence can be handled by adding several non-DNA characters to its end.

RESULTS

We evaluated *PPS+* by comparing it to homology-based methods (*MEGAN4*, *taxator-tk)* (*Huson et al., 2011*; *Dröge, Gregor & McHardy, 2014*), the fast taxonomic binning program *Kraken* (*Wood & Salzberg, 2014*), the composition-based method *PhyloPythia* trained under expert guidance (a recommended but time-consuming procedure) and to a generic *PPS* model using default settings (Supplemental Information 1, Section 3.5–3.8). For a performance comparison of *PPS* to methods with prohibitive runtimes for large datasets, such as *PhymmBL* (*Brady & Salzberg, 2011*) and *CARMA3* (*Gerlach & Stoye, 2011*), and the web-based tool *NBC* (*Rosen, Reichenberger & Rosenfeld, 2011*) see *Patil et al. (2011)*; *Patil, Roune & McHardy (2011)*; *Dröge, Gregor & McHardy (2014)*, as *PPS* has already been compared to these methods with favorable outcomes. For a comparison with 'taxonomy-free' binning software *CLARK* (*Ounit et al., 2015*) see (Supplemental Information 1, Section 7). We did not compare *PPS+* to profiling tools such as (*Liu et al., 2011*), as *PPS+* is a binning method that assigns a taxonomic label to each input sequence. As benchmark datasets, we created two simulated datasets, one with a uniform (137 Mb) and one with a log-normal (66 Mb) distribution of 47 community members (Supplemental Information 1, Section 3.1, Datasets S1 and S2). We also used two real datasets, a metagenome sample from the guts of two obese human twins (255 Mb) (*Turnbaugh et al., 2010*) and a cow rumen metagenome sample (319 Mb) from *Hess et al. (2011)* (Supplemental Information 1, Section 3.2, Datasets S3–S6) for evaluation.

Table 1 Test scenarios. Test scenarios where data was removed (masked) up to the specified rank for the corresponding taxa represented in the simulated metagenome datasets from the reference collections. RS denotes the reference collection of complete or draft genomes; *MG* indicates the reference collection of marker genes (Supplemental Information 1, Section 3.3).

Test scenario	Rank masked from RS	Rank masked from *MG*
1.	None	None
2.	Strain	None
3.	Species	None
4.	Genus	None
5.	Strain	Strain
6.	Species	Strain
7.	Genus	Strain
8.	Species	Species
9.	Genus	Genus

Benchmarks with simulated datasets

We constructed the simulated datasets by assembling simulated reads with an empirical error profile. The details on how the simulated reads were generated and assembled can be found in (Supplemental Information 1, Section 3.1). For the evaluation, precision and recall were calculated (Supplemental Information 1, Section 3.9). Furthermore, these measures were also calculated with a 'correction,' to account for the case where the sequences of one taxon were consistently assigned to a different taxon, as for draft genome reconstruction, it is more important that the sequences are assigned consistently than that the taxonomic identifier is correct. To assess the performance of the different methods in assigning the simulated sequence fragments without related reference genomes being available, 'new strain,' 'new species' and 'new genus' scenarios were simulated by removing all sequence data from the taxa of the simulated test dataset at each rank from the reference data. Furthermore, for *PPS+*, we distinguished whether the reference data were excluded (masked) from the reference sequence (RS) collection or also from the marker gene (*MG*) collection, since the *MG* collection included sequences for 15 times more distinct species than the RS collection. There were therefore two different situations to consider (Table 1).

PPS+ showed a substantially improved precision and recall over the *PPS* generic model, which demonstrated the impact of the improved selection of training data and modeled taxa (Figs. 2A and 2C, S1A–S1D and S3A–S3D). *PPS+* almost always had higher precision and recall than *MEGAN4* and *Kraken,* except when almost all test data were included in the reference sequences (Figs. 2A and 2C, S1A–S1C, S1E, S3A–S3C, S3E, S14A). This was even more pronounced when comparing bin quality using the corrected measures (Figs. 2B and 2D, S2A– S2C, S2E, S4A– S4C, S4E, S14A and S14D). When comparing *PPS+* to *taxator-tk*, *PPS+* had substantially improved recall, particularly for lower ranks (Figs. 2A and 2C, S1A–S1C, S1F, S3A–S3C, S3F); while *taxator-tk* outperformed all other methods in terms of precision (Figs. 2A and 2C, S1A–S1F and S3A–S3F). Both methods were similarly precise when analyzing bin recovery, independent of assigning the taxonomic

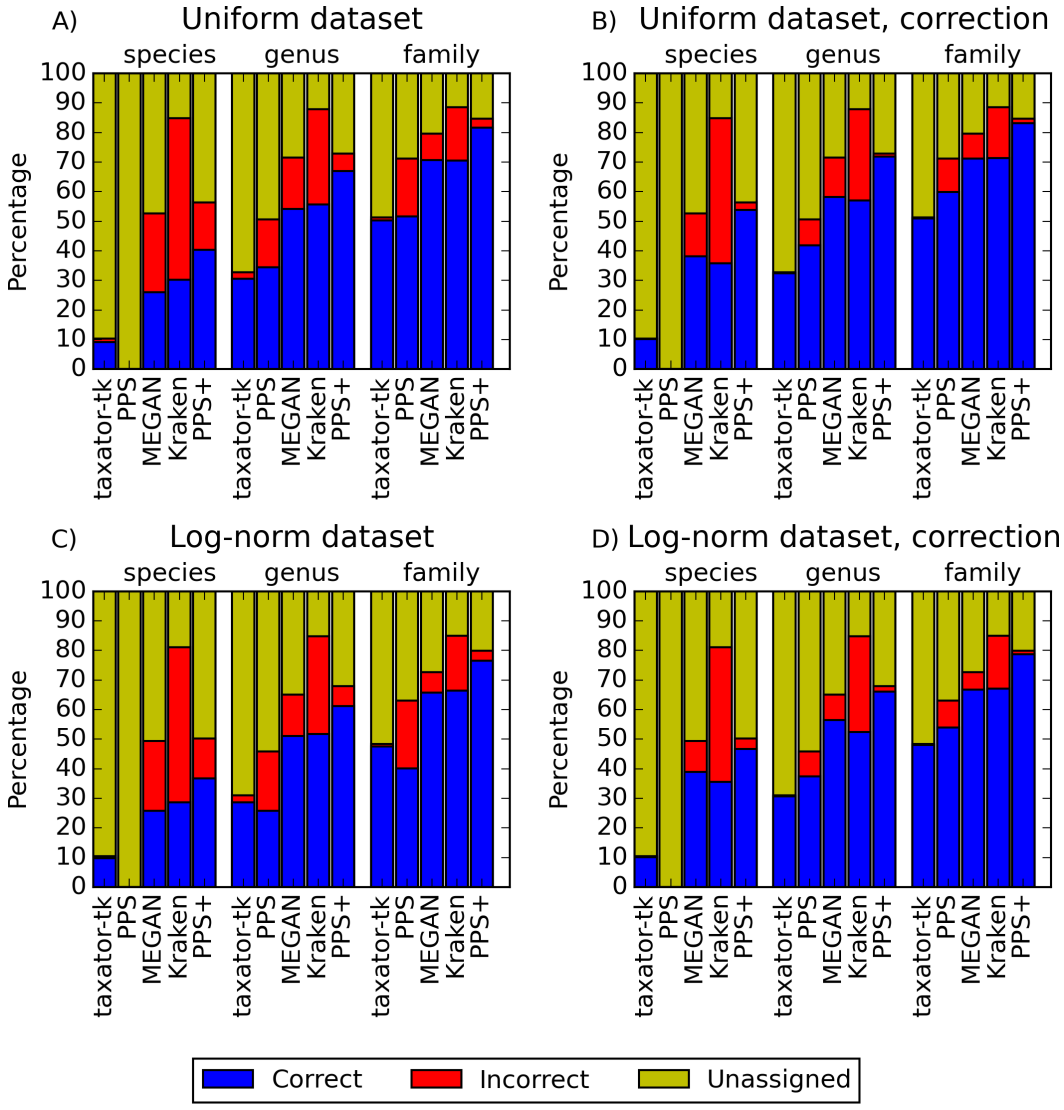


Figure 2 **Performance comparisons with simulated datasets.** (A) and (C) show the fraction of correct, incorrect and unassigned bp for simulated datasets with uniform and log-normally distributed species abundance for *PhyloPythiaS+*, the generic *PhyloPythiaS* model, *MEGAN4, Kraken* and *taxator-tk* for assignments at the species, genus and family ranks. Results were averaged over all test 'scenarios' (Table 1), where sequences of the same strain, species or genus from the simulated metagenomes were removed from the genome, draft genome and marker gene reference sequence collections (Figs. S1, S3, S14A and S14C). (B) and (D) show the portion of consistently (correct), inconsistently (incorrect) and unbinned (unassigned) bp without consideration of the taxonomic identifiers (Figs. S2, S4, S14B and S14D, Supplemental Information 1, Section 3.9.2). The exact values and the corresponding precision, recall and f_1-score are contained in (Tables S2–S5) for (A–D), respectively.

identifiers to the corrected measures (Figs. 2B and 2D, S2A– S2C, S2F, S4A– S4C and S4F). As a strong point of *PPS+* , we also observed that it more rarely predicted wrong taxa that were not a part of the sample than the other methods (Fig. S5). For example, for the genus rank in Scenarios 3 and 8, *PPS+* assigned sequences to only 2–5 false positive taxa, while

taxator-tk identified 20, *MEGAN4* 37 and *PPS* 59 false ones. If *PPS+* identified wrong taxa, these were usually very closely related to the true taxa.

Benchmarks with real datasets

Comparison of scaffold and contig assignments

For each taxonomic rank, the percentage and the total number of kb (% agreement and kb agreement) that were assigned the same taxonomic identifier were calculated for the real datasets, based on the assignments of scaffold and contig sequences (Supplemental Information 1, Section 3.10.1). For the chunked cow rumen dataset (Supplemental Information 1, Section 3.2.2), *taxator-tk* had the highest assignment consistency (Table 2); however, it assigned much fewer data than the other methods at lower taxonomic ranks. A detailed comparison is given in heat maps (Figs. S6–S13). *PPS+* performed substantially better by both measures than the generic *PPS* model in almost all cases. *PPS+* was also more consistent than *MEGAN4* for all lower ranks and assigned many more sequences than *MEGAN4* overall. For instance, at the genus rank, the scores were 84.3 and 56 '% agreement', as well as 33,724 and 13,726 'kb agreement' for *PPS+* and *MEGAN4*, respectively. The overall low numbers for *Kraken* suggests that it is rather not applicable to samples containing novel taxa. Also, the low number of consistently assigned bp by *MEGAN4* and *taxator-tk* to lower taxonomic ranks reflects the availability of few related reference genome sequences for the cow rumen metagenome sample, which is not an issue for a composition-based method *PPS+*.

For the human gut microbiome, extensive sequencing of isolate cultures has resulted in a large collection of several hundred reference genome sequences. Accordingly, for the human gut dataset, *taxator-tk*, *MEGAN4 and Kraken* assigned many more sequences than they did for the cow rumen dataset (Tables 2 and 3). For *Kraken* and *MEGAN4*, this was most pronounced for the genus and species ranks, even though this was also caused by counting scaffolds containing only one contig being consistent to itself. The most consistent method was again *taxator-tk*, but it also assigned fewer sequences than the other methods. *PPS+* performed better than the generic *PPS* model in all cases in terms of both measures (Table 3). *PPS+* and *MEGAN4* showed comparable consistency, with *PPS+* being more consistent for the class, order and species ranks, and *MEGAN4* being more consistent for the superkingdom, family and genus ranks. However, *PPS+* consistently assigned (kb agreement) more sequences than *MEGAN4*, except for the genus and species ranks. Thus, in the case of larger collections of related isolate genome sequences being available, composition- and homology-based methods perform similarly well.

The taxonomic scaffold-contig consistency of the assignments was additionally evaluated (Table S6 and Table S7) using a set of measures (Supplemental Information 1, Section 3.10.2) that provide more detailed insights into assignment consistency (Supplemental Information 1, Section 5.1) and support the conclusions in this section.

Comparison to an expert binning based on marker genes

A taxonomic binning generated by *PhyloPythia* (*PP*) with expert guidance for sample-derived model construction (*Turnbaugh et al., 2010*) was compared to the *PPS+* assignments. Scaffolds that were unassigned by either method were not considered.

Table 2 Comparison of contig and scaffold assignments of the chunked cow rumen dataset. Contigs of the cow rumen dataset of at least 10 kb were divided into chunks of 2 kb for evaluation of assignment consistency (Supplemental Information 1, Section 3.2.2). The contigs and scaffolds of the chunked cow rumen dataset were assigned using *PPS+* , the generic *PPS* model, *MEGAN4*, *taxator-tk* and *Kraken*. For each method, up to two taxonomic identifiers were assigned to each contig at each rank, i.e., one identifier came from the contig assignment and the second identifier came from the corresponding scaffold assignment. Contigs with less than two taxonomic assignments at each rank were not considered in this comparison. The measure '% agreement' was the percentage of contigs with the same two taxonomic identifiers at a particular rank, whereas 'kb agreement' was the total number of kb of contigs with the same taxonomic identifiers (Supplemental Information 1, Section 3.10.1). Bold numbers correspond to the best values, whereas italic numbers indicate the worst values.

Method	Rank	% agreement	kb agreement
PPS+	Phylum	73.9	**153,774**
PPS	Phylum	67.8	75,538
MEGAN4	Phylum	74.2	43,380
taxator-tk	Phylum	**98.2**	59,702
Kraken	Phylum	*67.0*	*33,558*
PPS+	Class	86.0	**99,596**
PPS	Class	*58.5*	43,931
MEGAN4	Class	68.5	33,780
taxator-tk	Class	**97.7**	*23,190*
Kraken	Class	*58.5*	27,536
PPS+	Order	88.4	**98,616**
PPS	Order	63.8	41,349
MEGAN4	Order	68.9	32,650
taxator-tk	Order	**98.0**	*22,368*
Kraken	Order	*57.0*	26,410
PPS+	Family	80.0	**46,343**
PPS	Family	55.8	19,158
MEGAN4	Family	55.0	15,790
taxator-tk	Family	**98.9**	*7,276*
Kraken	Family	*45.2*	18,370
PPS+	Genus	84.3	**33,724**
PPS	Genus	63.2	12,938
MEGAN4	Genus	56.0	13,726
taxator-tk	Genus	**99.1**	*6,042*
Kraken	Genus	*43.7*	16,912
PPS+	Species	91.6	9,821
PPS	Species	N/A	N/A
MEGAN4	Species	54.6	8,502
taxator-tk	Species	**100.0**	*292*
Kraken	Species	*38.1*	**14,186**

Table 3 Comparison of contig and scaffold assignments of the human gut metagenome dataset. Contig and scaffold sequences of the human gut metagenome dataset were assigned using *PPS+*, the generic *PPS* model, *MEGAN4*, *taxator-tk* and *Kraken*. The measures '% agreement' and 'kb agreement' were used to compare individual methods (Supplemental Information 1, Section 3.10.1). Bold numbers correspond to the best values, whereas italic numbers indicate the worst values.

Method	Rank	% agreement	kb agreement
PPS+	Phylum	99.0	**140,283**
PPS	Phylum	*97.0*	124,884
MEGAN4	Phylum	99.0	127,658
taxator-tk	Phylum	**100.0**	*104,475*
Kraken	Phylum	97.6	123,428
PPS+	Class	99.5	**134,707**
PPS	Class	96.9	118,068
MEGAN4	Class	98.5	122,131
taxator-tk	Class	**100.0**	*84,228*
Kraken	Class	*96.3*	121,071
PPS+	Order	99.5	**134,127**
PPS	Order	97.3	117,185
MEGAN4	Order	98.6	121,811
taxator-tk	Order	**100.0**	*83,337*
Kraken	Order	*96.3*	121,003
PPS+	Family	94.0	**110,664**
PPS	Family	92.6	97,066
MEGAN4	Family	96.2	98,582
taxator-tk	Family	**99.8**	*43,751*
Kraken	Family	*89.4*	109,151
PPS+	Genus	95.3	82,992
PPS	Genus	91.9	58,883
MEGAN4	Genus	96.1	86,495
taxator-tk	Genus	**99.9**	*34,667*
Kraken	Genus	*88.3*	**97,097**
PPS+	Species	94.7	43,329
PPS	Species	N/A	N/A
MEGAN4	Species	93.5	64,554
taxator-tk	Species	**99.7**	*10,314*
Kraken	Species	*81.3*	**94,591**

The *PP* expert binning and the *PPS+* binning agreed well, down to the order rank (Table 4). For the family and genus ranks, the overlap of both methods dropped to 69.5–74.1%, which may partly be due to changes in the NCBI taxonomy since the generation of the expert binning in 2009. Both *PPS+* and *PP* assignments were highly consistent with the *MG* assignments made by the + component of *PPS+* alone, though only a small number of scaffolds with marker genes could be compared (7–23% for different ranks). While *PPS+* had a larger overlap ('% agreement') with the *MG* assignments at the genus rank, *PP* had a larger overlap ('% agreement') with the *MG* assignments at the family rank. Moreover, we compared the number of taxonomic assignments for individual methods

Table 4 Comparison to an expert binning based on marker genes. Comparison of the taxonomic assignments of *PPS+* versus *PhyloPythia* (*PP*), with expert guidance for sample-derived model construction (*Turnbaugh et al., 2010*) for the human gut scaffolds (161, 343 kb) based on marker genes (*MG*), using the + component of *PPS+*. The measure '% agreement' represents the percentage of bp assigned by both methods to the same taxonomic identifiers at a given rank, whereas 'kb agreement' is the corresponding number of kb assigned by both methods to the same taxonomic identifier. Scaffolds assigned by only one method are not considered in this comparison. Bold numbers correspond to the best values, whereas italic numbers indicate the worst values.

Comparison	Rank	% agreement	kb agreement
PP vs PPS+	Superkingdom	99.6	**160,617**
MG vs PP	Superkingdom	**99.7**	38,314
MG vs PPS+	Superkingdom	*99.5*	*38,220*
PP vs PPS+	Phylum	*95.4*	**149,213**
MG vs PP	Phylum	96.9	*17,771*
MG vs PPS+	Phylum	**98.7**	18,065
PP vs PPS+	Class	*97.0*	**145,887**
MG vs PP	Class	98.1	*17,599*
MG vs PPS+	Class	**100.0**	17,869
PP vs PPS+	Order	*98.0*	**145,373**
MG vs PP	Order	98.3	*17,494*
MG vs PPS+	Order	**100.0**	17,764
PP vs PPS+	Family	*69.5*	**95,779**
MG vs PP	Family	**90.7**	13,047
MG vs PPS+	Family	83.7	*12,013*
PP vs PPS+	Genus	*74.1*	**78,686**
MG vs PP	Genus	91.6	12,235
MG vs PPS+	Genus	**94.9**	*11,479*

(Fig. 3): *PPS+* assigned sequences to low-ranking taxa down to the species level, in agreement with the *MG* assignments, while *PP* often assigned the respective sequences only to the parental taxa. This demonstrates that *PPS+* can generate a high quality taxonomic binning in a fully automated manner.

Throughput comparison

The throughput of the individual methods for contig assignments of the human gut sample was calculated (Supplemental Information 1, Section 3.3, 3.4 and 5.3). The throughput of *Kraken* substantially varied between 38.4 Mb/h and 4.2 Gb/h in our experiments, depending on whether its large (~200 GB) reference database was already loaded in the main memory or not, therefore *Kraken* is the fastest method in high performance environments. When only the prediction step of *PPS+* was considered, *PPS+* assigned up to 0.5 Gb/h and was more than 7 times faster than the homology-based methods (Fig. 4). This is relevant when *PPS* models are reused for the classification of another sample. Moreover, unlike the homology-based tools and *Kraken*, *PPS+* can be run on a standard laptop, as it requires much less main memory (see Supplemental Information 1, Section 3.4 for the hardware configurations used).

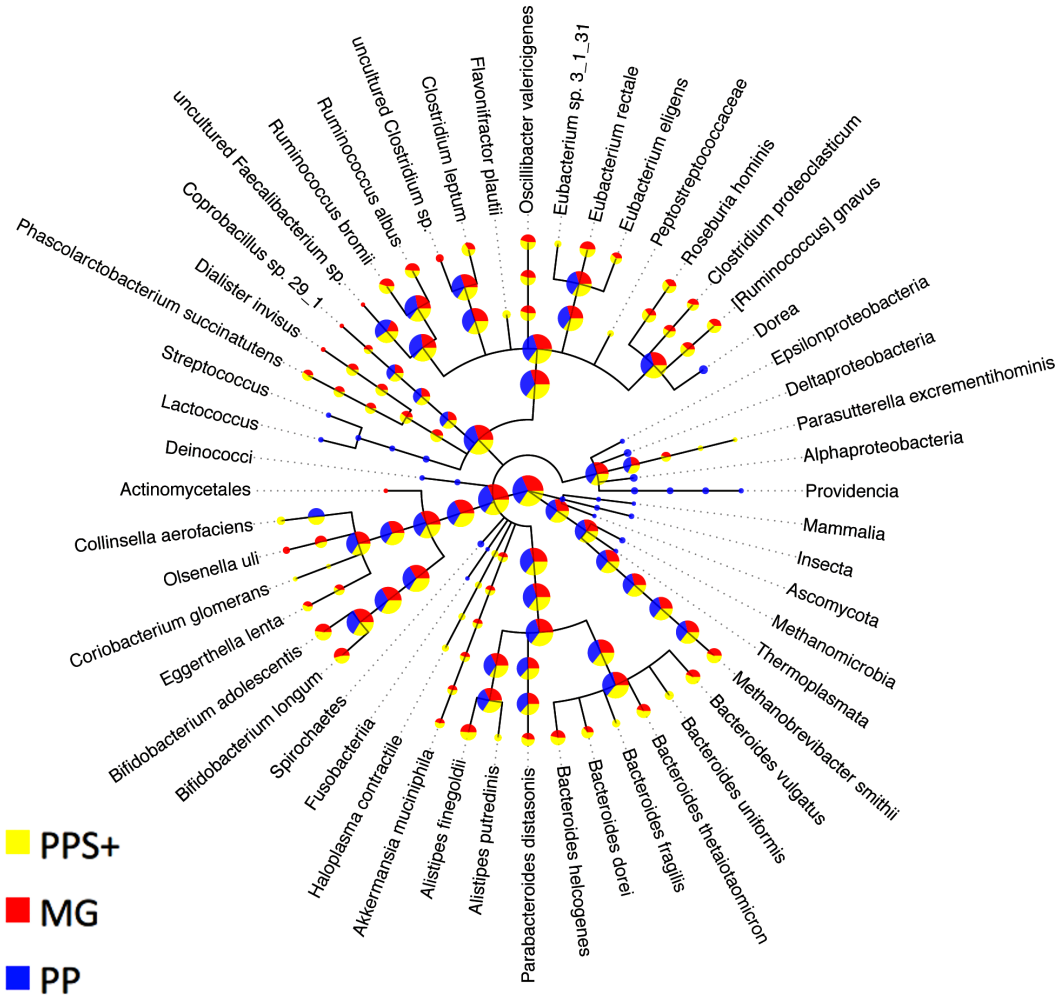


Figure 3 Comparison to expert binning based on marker genes. The amount of assigned bp by *PhyloPythia* (*PP*), *PhyloPythiaS+* (*PPS+*) and taxonomically informative marker genes directly (*MG*) to each taxon are indicated by the pie chart sizes on a log-scale for the human gut metagenome sample (*Turnbaugh et al., 2010*; *Patil, Roune & McHardy, 2011*). *PhyloPythiaS+* automatically determined the taxa to model from the sample. For the expert-trained *PhyloPythia*, the taxa to model were specified by an expert, and were included in the model if they were covered by sufficient reference sequence data retrieved separately from the sample and from sequenced human gut isolates. *PhyloPythiaS+* assigned sequences to low-ranking taxa down to the species level, in agreement with the marker gene assignments, while *PhyloPythia* often assigned these sequences to the parental taxa. For the *MG* assignments, a negligible amount—only two contigs (3.6 kb) of two scaffolds (231 kb)—were used as sample-derived training data for *PPS+*; as mainly sample contigs (2.5 Mb) that were not part of scaffolds were used as sample-derived data to train *PPS*.

CONCLUSIONS

We describe a taxonomic assignment program that produces accurate assignments with a precision of 80% or more also for low-ranking taxa from metagenome samples. *PPS+* is a fully automated successor of the *PhyloPythiaS* software. It automatically determines the most relevant taxa to be modeled and suitable training sequences directly from

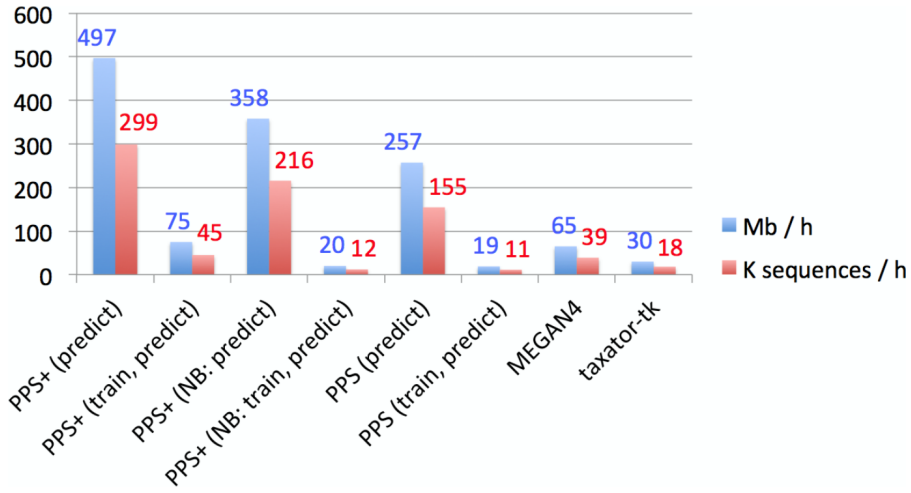


Figure 4 Empirical comparison of execution times. The throughput was measured in Mb and the number of sequences classified within 1 h with one execution thread, using all assembled contigs of the human gut metagenome dataset on a server computer with an AMD Opteron 6386 SE 2.8 GHz processor and 512 GB of RAM. Default settings were used for all methods (Supplemental Information 1, Section 3.5–3.7). Both *MEGAN4* and *taxator-tk* were run using *BLAST*. For *MEGAN4*, only the runtime of *BLAST* was considered, as the runtime of the subsequent algorithm was negligible. For *PhyloPythiaS* and *PhyloPythiaS+*, the throughput was calculated for the prediction step and both steps (training and prediction). The former is relevant when using previously generated models for the classification of multiple samples. The execution time shown for *PhyloPythiaS* is approximately three times better than that for the original release, as we incorporated the new *k*-mer counting algorithm. *PhyloPythiaS+* was the only method that could also be executed on a standard laptop (NB) with an Intel i5 M520 2.4 GHz processor, 4 GB of RAM and 150 GB disk space.

the input sample, which are then used to generate a sample-specific structured output SVM taxonomic classifier for the taxonomic binning of a sample. This enables its use for researchers without experience in the field or time to search for suitable training sequences for the manual construction of well-matching taxonomic classifier to a particular metagenome sequence sample.

PPS+ is best suited for the analysis of large NGS metagenome samples with assembled contigs (> 1kb) carrying marker genes or datasets including the high quality longer PacBio (*Chin et al., 2013*) consensus reads. Contrary to some recent methods for the taxonomic profiling or binning of multiple similar samples (*Sunagawa et al., 2013*), *PPS+* can be also applied to individual samples. *PPS+* requires only 100 kb of sample-derived data to model a bin, while homology-based methods require large related reference genome or draft genome sequence collections for substantial assignments to low-ranking taxa. Our experiments on both real and simulated metagenome samples showed that *PPS+* automatically reconstructed many low-ranking bins from metagenome samples, such as for genera and species, representing draft genomes or pan-genomes of different community members.

The novel implementation of the *k*-mer counting algorithm accelerated *k*-mer counting 100-fold in comparison to the original *PPS* software and made *PPS+* overall up to three

times faster. The method performed favorably in comparison to all state-of-the-art *k*-mer counting software for the simultaneous enumeration of 4–6-mers, commonly used for composition-based binning.

PPS models can be reused when classifying multiple samples from the same or similar environments. When comparing assignment with *PPS+* to *MEGAN4* and *taxator-tk*, *PPS+* showed a competitive processing time, allowing to process up to 0.5 Gb of sequences per hour with a given *PPS* model on a single core with much lower main memory requirements, while *MEGAN4* processed 0.065 Gb and *taxator-tk* 0.03 Gb (Fig. 4). The fastest method in the comparison was *Kraken* with up to 4.2 Gb/h; however, we have found that *Kraken* should be used only for well-studied environments, for which many closely related (draft) genomes have been sequenced, as an alternative to alignment-based methods, as its use for samples originating from novel environments is very limited (Fig. 2).

In terms of assignment quality, we found that *PPS+* often outperformed *MEGAN4* and *Kraken* in terms of precision, recall and consistency. *Taxator-tk* performed best in terms of precision and consistency, but assigned substantially fewer sequences to low taxonomic ranks. *PPS+* also excelled in determining the taxa that were part of the simulated metagenome community. We found that the fully automated *PPS+* binning can be as good as an expert-guided binning with the original *PhyloPythia* implementation. *PPS+* also showed a substantially improved assignment performance compared to the generic *PPS* model.

To conclude, the newly introduced self-training (+) component and the faster *k*-mer counting algorithm implemented in *PPS+* allow users to generate high quality taxonomic binnings of metagenome samples in a high-throughput fashion, without requiring expensive hardware, manual intervention and expert knowledge. It should be helpful to a wide range of users. An initial version of the software has been already employed for the taxonomic binning of a metagenome sample from reindeer guts by *Pope et al. (2011a)* and it is currently used in several other projects: for instance, a *PPS+* binning of shotgun metagenome samples indicated the likely metabolite flow and participating microbial phylotypes for a biogas-producing microbial community tolerant of high ammonia levels (Supplemental Information 2).

PPS+ is distributed with a large reference sequence collection (containing Bacterial and Archaeal data) in a virtual machine, which makes it easy to install. This allows metagenome sample analysis on a standard laptop under Windows, Unix or OS X systems.

ACKNOWLEDGEMENTS

The authors thank Phillip B. Pope and Jeremy Frank for their summary of the *PPS+* results for shotgun metagenome data from a biogas-producing microbial community (Supplemental Information 2); and Rubén Garrido Oter for generating the pie tree figures.

ADDITIONAL INFORMATION AND DECLARATIONS

Funding

ACM, IG and JD were funded by the Max-Planck society, Heinrich Heine University Düsseldorf and Helmholtz Center for Infection Research. MS was supported by Unilever R & D Port Sunlight, Bebington, UK. CQ was supported by an Engineering and Physical Sciences Research Council Career Acceleration Fellowship [EP/ H003851/1]. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Grant Disclosures

The following grant information was disclosed by the authors:
Max-Planck society, Heinrich Heine University Düsseldorf.
Helmholtz Center for Infection Research.
Unilever R & D Port Sunlight, Bebington, UK.
Engineering and Physical Sciences Research Council Career Acceleration Fellowship: EP/H003851/1.

Competing Interests

Alice C. McHardy is an Academic Editor for PeerJ. The authors declare there are no competing interests.

Author Contributions

- Ivan Gregor conceived and designed the experiments, performed the experiments, analyzed the data, contributed reagents/materials/analysis tools, wrote the paper, prepared figures and/or tables.
- Johannes Dröge conceived and designed the experiments, performed the experiments, reviewed drafts of the paper.
- Melanie Schirmer performed the experiments, reviewed drafts of the paper.
- Christopher Quince reviewed drafts of the paper.
- Alice C. McHardy conceived and designed the experiments, wrote the paper, analyzed the data.

Data Availability

The following information was supplied regarding data availability:

All the external resources can be found at: https://github.com/algbioi/ppsp/wiki.

Supplemental Information

Supplemental information for this article can be found online at http://dx.doi.org/10.7717/peerj.1603#supplemental-information.

REFERENCES

Audano P, Vannberg F. 2014. KAnalyze: a fast versatile pipelined K-mer toolkit. *Bioinformatics* **30**:2070–2072 DOI 10.1093/bioinformatics/btu152.

Blaser M, Bork P, Fraser C, Knight R, Wang J. 2013. The microbiome explored: recent insights and future challenges. *Nature Reviews Microbiology* **11**:213–217 DOI 10.1038/nrmicro2973.

Boisvert S, Raymond F, Godzaridis É, Laviolette F, Corbeil J. 2012. Ray Meta: scalable de novo metagenome assembly and profiling. *Genome Biology* **13**:R122 DOI 10.1186/gb-2012-13-12-r122.

Brady A, Salzberg S. 2011. PhymmBL expanded: confidence scores, custom databases, parallelization and more. *Nature Methods* **8**:367–367 DOI 10.1038/nmeth0511-367.

Chin CS, Alexander DH, Marks P, Klammer AA, Drake J, Heiner C, Clum A, Copeland A, Huddleston J, Eichler EE, Turner SW, Korlach J. 2013. Nonhybrid, finished microbial genome assemblies from long-read SMRT sequencing data. *Nature Methods* **10**:563–569 DOI 10.1038/nmeth.2474.

Deorowicz S, Kokot M, Grabowski S, Debudaj-Grabysz A. 2015. KMC 2: fast and resource-frugal k-mer counting. *Bioinformatics* **31(10)**:1569–1576 DOI 10.1093/bioinformatics/btv022.

Deschavanne PJ, Giron A, Vilain J, Fagot G, Fertil B. 1999. Genomic signature: characterization and classification of species assessed by chaos game representation of sequences. *Molecular Biology and Evolution* **16**:1391–1399 DOI 10.1093/oxfordjournals.molbev.a026048.

Dröge J, Gregor I, McHardy AC. 2014. Taxator-tk: precise taxonomic assignment of metagenomes by fast approximation of evolutionary neighborhoods. *Bioinformatics* **31(6)**:817–824 DOI 10.1093/bioinformatics/btu745.

Dröge J, McHardy AC. 2012. Taxonomic binning of metagenome samples generated by next-generation sequencing technologies. *Briefings in Bioinformatics* **13(6)**:646–655 DOI 10.1093/bib/bbs031.

Gerlach W, Stoye J. 2011. Taxonomic classification of metagenomic shotgun sequences with CARMA3. *Nucleic Acids Research* **39**:e91–e91 DOI 10.1093/nar/gkr225.

Hess M, Sczyrba A, Egan R, Kim T-W, Chokhawala H, Schroth G, Luo S, Clark DS, Chen F, Zhang T, Mackie RI, Pennacchio LA, Tringe SG, Visel A, Woyke T, Wang Z, Rubin EM. 2011. Metagenomic discovery of biomass-degrading genes and genomes from cow rumen. *Science* **331**:463–467 DOI 10.1126/science.1200387.

Hugenholtz P. 2002. Exploring prokaryotic diversity in the genomic era. *Genome Biology* **3**: REVIEWS0003.

Huson DH, Mitra S, Ruscheweyh HJ, Weber N, Schuster SC. 2011. Integrative analysis of environmental sequences using MEGAN4. *Genome Research* **21**:1552–1560 DOI 10.1101/gr.120618.111.

Joachims T, Finley T, Yu C-NJ. 2009. Cutting-plane training of structural SVMs. *Machine Learning* **77**:27–59 DOI 10.1007/s10994-009-5108-8.

Kalyuzhnaya MG, Lapidus A, Ivanova N, Copeland AC, McHardy AC, Szeto E, Salamov A, Grigoriev IV, Suciu D, Levine SR, Markowitz VM, Rigoutsos I, Tringe SG, Bruce DC, Richardson PM, Lidstrom ME, Chistoserdova L. 2008. High-resolution metagenomics targets specific functional types in complex microbial

communities. *Nature Biotechnology* **26**:1029–1034 DOI 10.1038/nbt.1488.

Karlin S, Burge C. 1995. Dinucleotide relative abundance extremes: a genomic signature. *Trends in Genetics* **11**:283–290 DOI 10.1016/S0168-9525(00)89076-9.

Karp RM, Rabin MO. 1987. Efficient randomized pattern-matching algorithms. *IBM Journal of Research and Development* **31**:249–260 DOI 10.1147/rd.312.0249.

Kunin V, Copeland A, Lapidus A, Mavromatis K, Hugenholtz P. 2008. A bioinformatician's guide to metagenomics. *Microbiology and Molecular Biology Reviews* **72**:557–578 DOI 10.1128/MMBR.00009-08.

Laserson J, Jojic V, Koller D. 2011. Genovo: de novo assembly for metagenomes. *Journal of Computational Biology* **18**:429–443 DOI 10.1089/cmb.2010.0244.

Liu B, Gibbons T, Ghodsi M, Treangen T, Pop M. 2011. Accurate and fast estimation of taxonomic profiles from metagenomic shotgun sequences. *BMC Genomics* **12**:S4 DOI 10.1186/1471-2164-12-S2-S4.

Loman NJ, Constantinidou C, Chan JZM, Halachev M, Sergeant M, Penn CW, Robinson ER, Pallen MJ. 2012. High-throughput bacterial genome sequencing: an embarrassment of choice, a world of opportunity. *Nature Reviews Microbiology* **10**:599–606 DOI 10.1038/nrmicro2850.

Marcais G, Kingsford C. 2011. A fast, lock-free approach for efficient parallel counting of occurrences of k-mers. *Bioinformatics* **27**:764–770 DOI 10.1093/bioinformatics/btr011.

McHardy AC, Martín HG, Tsirigos A, Hugenholtz P, Rigoutsos I. 2007. Accurate phylogenetic classification of variable-length DNA fragments. *Nature Methods* **4**:63–72 DOI 10.1038/nmeth976.

Meinicke P, Asshauer KP, Lingner T. 2011. Mixture models for analysis of the taxonomic composition of metagenomes. *Bioinformatics* **27**:1618–1624 DOI 10.1093/bioinformatics/btr266.

Metzker ML. 2010. Sequencing technologies—the next generation. *Nature Reviews Genetics* **11**:31–46 DOI 10.1038/nrg2626.

Namiki T, Hachiya T, Tanaka H, Sakakibara Y. 2012. MetaVelvet: an extension of Velvet assembler to de novo metagenome assembly from short sequence reads. *Nucleic Acids Research* **40(20)**:e155 DOI 10.1093/nar/gks678.

Ounit R, Wanamaker S, Close TJ, Lonardi S. 2015. CLARK: fast and accurate classification of metagenomic and genomic sequences using discriminative k-mers. *BMC Genomics* **16**:236 DOI 10.1186/s12864-015-1419-2.

Patil KRK, Roune LL, McHardy ACA. 2011. The *PhyloPythiaS* web server for taxonomic assignment of metagenome sequences. *PLoS ONE* **7**:e38581–e38581 DOI 10.1371/journal.pone.0038581.

Patil KR, Haider P, Pope PB, Turnbaugh PJ, Morrison M, Scheffer T, McHardy AC. 2011. Taxonomic metagenome sequence assignment with structured output models. *Nature Methods* **8**:191–192 DOI 10.1038/nmeth0311-191.

Pell J, Hintze A, Canino-Koning R, Howe A, Tiedje JM, Brown CT. 2012. Scaling metagenome sequence assembly with probabilistic de Bruijn graphs. *Proceedings of*

the National Academy of Sciences of the United States of America **109**:13272–13277 DOI 10.1073/pnas.1121464109.

Peng Y, Leung HCM, Yiu SM, Chin FYL. 2011. Meta-IDBA: a de Novo assembler for metagenomic data. *Bioinformatics* **27**:i94–i101 DOI 10.1093/bioinformatics/btr216.

Pope PB, Mackenzie AK, Gregor I, Smith W, Sundset MA, McHardy AC, Morrison M, Eijsink VGH. 2011a. Metagenomics of the svalbard reindeer rumen microbiome reveals abundance of polysaccharide utilization Loci. *PLoS One* **7**:e38571–e38571 DOI 10.1371/journal.pone.0038571.

Pope PB, Smith W, Denman SE, Tringe SG, Barry K, Hugenholtz P, McSweeney CS, McHardy AC, Morrison M. 2011b. Isolation of Succinivibrionaceae implicated in low methane emissions from Tammar wallabies. *Science* **333**:646–648 DOI 10.1126/science.1205760.

Riesenfeld CS, Schloss PD, Handelsman J. 2004. Metagenomics: genomic analysis of microbial communities. *Annual Review of Genetics* **38**:525–552 DOI 10.1146/annurev.genet.38.072902.091216.

Rosen GL, Reichenberger ER, Rosenfeld AM. 2011. NBC: the Naive Bayes Classification tool webserver for taxonomic classification of metagenomic reads. *Bioinformatics* **27**:127–129 DOI 10.1093/bioinformatics/btq619.

Roy RS, Bhattacharya D, Schliep A. 2014. Turtle: Identifying frequent k-mers with cache-efficient algorithms. *Bioinformatics* **30**:1950–1957 DOI 10.1093/bioinformatics/btu132.

Schloissnig S, Arumugam M, Sunagawa S, Mitreva M, Tap J, Zhu A, Waller A, Mende DR, Kultima JR, Martin J, Kota K, Sunyaev SR, Weinstock GM, Bork P. 2013. Genomic variation landscape of the human gut microbiome. *Nature* **493**:45–50 DOI 10.1038/nature11711.

Schloss PD, Westcott SL, Ryabin T, Hall JR, Hartmann M, Hollister EB, Lesniewski RA, Oakley BB, Parks DH, Robinson CJ, Sahl JW, Stres B, Thallinger GG, Van Horn DJ, Weber CF. 2009. Introducing mothur: open-source, platform-independent, community-supported software for describing and comparing microbial communities. *Applied and Environmental Microbiology* **75**:7537–7541 DOI 10.1128/AEM.01541-09.

Segata N, Waldron L, Ballarini A, Narasimhan V, Jousson O, Huttenhower C. 2012. Metagenomic microbial community profiling using unique clade-specific marker genes. *Nature Methods* **9**:811–814 DOI 10.1038/nmeth.2066.

Silva GGZ, Cuevas DA, Dutilh BE, Edwards RA. 2013. FOCUS: an alignment-free model to identify organisms in metagenomes using non-negative least squares. *PeerJ* **2**:e425–e425 DOI 10.7717/peerj.425.

Stark M, Berger SA, Stamatakis A, Mering von C. 2009. MLTreeMap—accurate Maximum Likelihood placement of environmental DNA sequences into taxonomic and functional reference phylogenies. *BMC Genomics* **11**:461–461 DOI 10.1186/1471-2164-11-461.

Sunagawa S, Mende DR, Zeller G, Izquierdo-Carrasco F, Berger SA, Kultima JR, Coelho LP, Arumugam M, Tap J, Nielsen HB, Rasmussen S, Brunak S, Pedersen

O, Guarner F, De Vos WM, Wang J, Li J, Doré J, Ehrlich SD, Stamatakis A, Bork P. 2013. Metagenomic species profiling using universal phylogenetic marker genes. *Nature Methods* **10**:1196–1199 DOI 10.1038/nmeth.2693.

Turnbaugh PJ, Quince C, Faith JJ, McHardy AC, Yatsunenko T, Niazi F, Affourtit J, Egholm M, Henrissat B, Knight R, Gordon JI. 2010. Organismal, genetic, and transcriptional variation in the deeply sequenced gut microbiomes of identical twins. *Proceedings of the National Academy of Sciences of the United States of America* **107**:7503–7508 DOI 10.1073/pnas.1002355107.

Wood DE, Salzberg SL. 2014. Kraken: ultrafast metagenomic sequence classification using exact alignments. *Genome Biology* **15**:R46 DOI 10.1186/gb-2014-15-3-r46.

Wu M, Eisen JA. 2008. A simple, fast, and accurate method of phylogenomic inference. *Genome Biology* **9**:R151–R151 DOI 10.1186/gb-2008-9-10-r151.

Wu M, Scott AJ. 2012. Phylogenomic analysis of bacterial and archaeal sequences with AMPHORA2. *Bioinformatics* **28**:1033–1034 DOI 10.1093/bioinformatics/bts079.

Zarowiecki M. 2012. Metagenomics with guts. *Nature Reviews Microbiology* **10**:674–674 DOI 10.1038/nrmicro2879.

Minor Corrections: Under the section "Algorithm description", the second equation should read "$index_{i+1} = (index_i - a_i * 4^{k-1}) * 4 + a_{i+k} * 4^0$", instead of "$index_{i+1} = (index_i - a_i * 4^{k-1}) * a_{i+k} * 4^0$".

The reference "Patil KR, Roune L, McHardy AC. 2011. The PhyloPythiaS web server for taxonomic assignment of metagenome sequences. PLoS ONE 7:e38581" should be "Patil KR, Haider P, Pope PB, Turnbaugh PJ, Morrison M, Scheffer T, McHardy AC. 2011. Taxonomic metagenome sequence assignment with structured output models. *Nature Methods*, 8, 191-192." The citation is used in the second paragraph of the Introduction section, and in the Figure 3 legend, and should be cited in the manuscript as "Patil, Haider & McHardy 2011", instead of "Patil, Roune & McHardy, 2011".

In the section "Benchmarks with simulated datasets", there are some incorrect figure references: "PPS+ almost always had higher precision and recall than MEGAN4 and Kraken, except when almost all test data were included in the reference sequences (Figs. 2A and 2C, S1A–S1C, S1E, S3A–S3C, S3E, S14A). This was even more pronounced when comparing bin quality using the corrected measures (Figs. 2B and 2D, S2A–S2C, S2E, S4A–S4C, S4E, S14A and S14D)" should be "PPS+ almost always had higher precision and recall than MEGAN4 and Kraken, except when almost all test data were included in the reference sequences (Figs. 2A and 2C, S1A–S1C, S1E, S3A–S3C, S3E, S14A and S14C). This was even more pronounced when comparing bin quality using the corrected measures (Figs. 2B and 2D, S2A–S2C, S2E, S4A–S4C, S4E, S14B and S14D)."

Supplemental Text S1

PhyloPythiaS+: A Self-Training Method for the Rapid Reconstruction of Low-Ranking Taxonomic Bins from Metagenomes

Table of Contents

1 Extended abstract

Metagenomics is an approach for characterizing environmental microbial communities *in situ,* it allows their functional and taxonomic characterization and to recover sequences from uncultured taxa. A major aim is to reconstruct (partial) genomes for individual community members from metagenomes. For communities of up to medium diversity (e.g. excluding environments such as soil), this is often achieved by a combination of sequence assembly and binning, where sequences are grouped into 'bins' representing taxa of the underlying microbial community from which they originate. If sequences can only be binned to higher-ranking taxa than strain or species, these bins offer less detailed insights into the underlying microbial community. Therefore, assignment to low-ranking taxonomic bins is an important challenge for binning methods as is scalability to Gb-sized datasets generated with deep sequencing techniques. Due to the importance of a match of the training data to the test dataset in machine learning for achieving high classification accuracy, one of the best available methods for the recovery of species bins from an individual metagenome sample (Patil et al., 2011; Pope et al., 2011) is the expert-trained *PhyloPythiaS* package, where a human expert identifies the 'training' sequences directly from the sample using marker genes and contig coverage information and based on data availability decides on the taxa to incorporate into the composition-based taxonomic model. The sequences of a metagenome sample are consequently assigned to these or higher ranking taxa by *PhyloPythiaS*. Because of the manual effort involved, this approach does not scale to multiple metagenome samples and requires substantial expertise, which researchers who are new to the area may not have. Other methods for draft genome reconstruction use multiple related metagenome samples as input (Albertsen et al., 2013; Imelfort et al., 2014) or are not distributed as a software package (Iverson et al., 2012).

With these challenges in mind, we have developed *PhyloPythiaS*+, a successor to our previously described method *PhyloPythia(S)* (McHardy et al., 2007; Patil et al., 2011). The newly developed + component performs the work of the human expert. It screens the metagenome sample for sequences carrying copies of one of 34 taxonomically informative marker genes (Wu & Scott, 2012) (Section 3.3). Identified marker genes are taxonomically classified using an extensive reference gene collection. The + component then decides which taxa to incorporate into the composition-based taxonomic model based on the amount of

available sequence data identified from the metagenome sample, genome and draft genome reference sequence collections (Figure 1).

We evaluated *PhyloPythiaS*+ on metagenome datasets of assembled simulated reads with Illumina GAII error profiles generated from a log-normal or uniform abundance distribution over 47 strains, and two real metagenome datasets from human gut and cow rumen samples (Tables 2–3, S6–S7, Sections 3). *PhyloPythiaS*+ had substantially higher overall precision and recall than the generic *PhyloPythiaS* model, because of the better match of the composition-based taxonomic model to the sequenced microbial community (Figs 2 and S1–S4, Section 3.9). It performed similarly well to an expert-trained *PhyloPythia* model without requiring manual effort (Figure 3, Table 4). Comparisons to sequence-similarity-based methods such as the popular MEtaGenome ANalyser (*MEGAN*, version 4) (Huson et al., 2011) and our own *taxator-tk* (Dröge, Gregor & McHardy, 2014) software showed a substantial increase in correct assignments to low taxonomic ranks for *PhyloPythiaS+,* while maintaining acceptably low error rates (Figs 2 and S1–S5). The largest improvement in comparison to the other methods was observed for taxa from deep-branching lineages, such as from genera or families without sequenced genomes but with marker gene data for the strain or species available (Fig. S1–S4, Table 1: Test Scenarios 2–4). This is currently the case for 39,201 species represented in our 16S reference gene collection.

PhyloPythiaS+ includes a new *k*-mer counting algorithm based on the Rabin Karp string matching algorithm. The algorithm accelerated *k*-mer counting 100-fold and reduced the overall execution time of the software by a factor of three in comparison to the original *PhyloPythiaS* release (Figure 4). We found that 500 and 360 Mb/hour could be assigned by *PhyloPythiaS*+ on a single CPU core of a standard compute server and a laptop, respectively. Our software thus allows to analyze Gb-sized metagenomes with inexpensive hardware, and to recover species or genera-level bins with low error rates in a fully automated fashion. *PhyloPythiaS*+ is distributed in a virtual machine and is easy to install for all common operating systems.

2 The evaluation of the *k*-mer counting algorithms

The main advantage of our method is that we do not use additional helper data structures such as suffix trees, since we work directly with arrays that represent DNA sequences. The only larger data structure that is necessary is a one-dimensional array that contains the counts of individual *k*-mers. The algorithm also processes one sequence at a time and thus there is no need to store all the sequences in the main memory, which makes the algorithm memory-efficient (e.g. less than one MB of the main memory in the scenario used in *PPS+*). To compute the next index from a previous index, we need to perform only two bit shift operations, one addition, one subtraction and one read operation (of a_{i+k}). This ensures complexity *O(n),* where *n* is the length of the DNA sequence that is being considered.

Our *k*-mer counting algorithm was compared to *Jellyfish* (version 1.1.1), *Jellyfish* (version 2.2) (Marcais & Kingsford, 2011) and *KAnalyze* (version 0.9.7) (Audano & Vannberg, 2014) (Table S1). All programs were run for *k*-mers $$k \in [4, \dots, 9]$$.

Jellyfish (version 1.1.1) was run with default parameters as:
jellyfish count -m $k -c 3 -s 10000000 -t 1 --both-strands -o OUTPUT.txt INPUT.fasta

Jellyfish (version 2.2.) was run with the following parameters, as this yielded better runtimes as the default parameters:
jellyfish count -m $k -c 16 -s 1000000 --both-strands -o OUTPUT.txt INPUT.fasta

KAnalyze (version 0.9.7) was run as:
count -k $k -d 1 -f fasta -r -o OUTPUT.txt INPUT.fasta

Our *k*-mer counting algorithm was run as:
fasta2kmers -i INPUT.fasta -f OUTPUT.txt -j $k -k $k
However, for the simultaneous counting of *k*-mers 4, 5, and 6, the program was run as:
fasta2kmers -i INPUT.fasta -f OUTPUT.txt -j 4 -k 6

3 Benchmark settings

3.1 Simulated datasets details and generation

Our simulated mock community comprised 47 strains from 45 different species (37 different genera) defined at all major taxonomic ranks, i.e. at superkingdom, phylum, class, order, family, genus and species rank. Two simulated datasets were generated with different abundance profiles, one with a uniform distribution and one with a log-normal distribution (μ=1, σ=2).

A custom read simulator was used which utilizes position- and nucleotide-specific substitution patterns derived from experimental datasets. This allowed us to generate reads with more realistic error profiles than we would with read simulators such as *pIRS* (Hu et al., 2012), *ART* (Huang et al., 2012) or *MetaSim* (Richter et al., 2007). Furthermore, we could thus specify and test different species abundance distributions for the microbial community and generate very large datasets due to the parallelization of the simulation program. We did not use the simulated datasets from Mavromatis *et al.* (Mavromatis et al., 2007), as these are substantially smaller than the current metagenome datasets.

Both simulated datasets were generated based on Illumina GAII error profiles where the standard library preparation method was used. The insert size distribution was also based on the experimental dataset. For each dataset, 15 million paired-end reads of 90 bp were generated with an average insert size of 291 bp. The first 10 bp of the 100 bp reads in the experimental dataset were trimmed because of fluctuations in the nucleotide distributions at the starting positions, which indicated partial remains of the barcode sequence. The read simulator produces output in FASTA format, which was converted into a pseudo-FASTQ format for the downstream analysis with uniformly high quality scores. The reads were then assembled with *Metassembler* (Debruijn, 2014) using *Velvet* (Zerbino & Birney, 2008), run with different *k*-mer sizes ranging between 19 and 75, and were subsequently merged with *Minimus2* (Treangen et al., 2011). This assembly procedure resulted in a larger assembled dataset than assembly with *SOAPdenovo2* (Luo et al., 2012), *Metavelvet* (Namiki et al., 2012) or *Newbler* (Roche, 2014). Contig sequences longer than 1000 bp were considered further. The contigs were subsequently mapped with *BLAST* (Camacho et al., 2009) onto the reference genomes to recover their taxonomic identifiers.

Properties of the simulated datasets:

Distribution	Contigs	Mb
Uniform	14,393	137
Log-normal	13,284	66

List of strains used to generate simulated datasets:

Strain name	Accession number
Acidobacterium capsulatum ATCC 51196	CP001472.1
Akkermansia muciniphila ATCC BAA-835	CP001071.1
Archaeoglobus fulgidus DSM 4304	AE000782.1
Bacteroides thetaiotaomicron VPI-5482	AE015928.1
Bacteroides vulgatus ATCC 8482	CP000139.1
Bordetella bronchiseptica RB50	BX470250.1
Caldicellulosiruptor bescii DSM 6725	CP001393.1
Caldicellulosiruptor saccharolyticus DSM 8903	CP000679.1
Chlorobium limicola DSM 245	CP001097.1
Chlorobium phaeobacteroides DSM 266	CP000492.1
Chlorobium phaeovibrioides DSM 265	CP000607.1
Chlorobium tepidum TLS	AE006470.1
Chloroflexus aurantiacus J-10-fl	CP000909.1
Clostridium thermocellum ATCC 27405	CP000568.1
	AE001825.1
Deinococcus radiodurans R1	AE000513.1
Dickeya dadantii 3937	CP002038.1
Dictyoglomus turgidum DSM 6724	CP001251.1
Enterococcus faecalis V583	AE016830.1
Fusobacterium nucleatum subsp. *nucleatum* ATCC 25586	AE009951.2
Gemmatimonas aurantiaca T-27	AP009153.1
Herpetosiphon aurantiacus DSM 785	CP000875.1
Hydrogenobaculum sp. Y04AAS1	CP001130.1
Ignicoccus hospitalis KIN4/I	CP000816.1
Methanocaldococcus jannaschii DSM 2661	L77117.1
Methanococcus maripaludis C5	CP000609.1
Methanococcus maripaludis S2	BX950229.1
Nitrosomonas europaea ATCC 19718	AL954747.1
Pelodictyon phaeoclathratiforme BU-1	CP001110.1
Persephonella marina EX-H1	CP001230.1

Porphyromonas gingivalis ATCC 33277	AP009380.1
Pyrobaculum aerophilum str. IM2	AE009441.1
Pyrobaculum calidifontis JCM 11548	CP000561.1
Rhodopirellula baltica SH 1	BX119912.1
Ruegeria pomeroyi DSS-3	CP000031.1
Salinispora arenicola CNS-205	CP000850.1
Salinispora tropica CNB-440	CP000667.1
Shewanella baltica OS185	CP000753.1
Shewanella baltica OS223	CP001252.1
Sulfolobus tokodaii str. 7	BA000023.2
Sulfurihydrogenibium sp. YO3AOP1	CP001080.1
Thermoanaerobacter pseudethanolicus ATCC 33223	CP000924.1
Thermotoga neapolitana DSM 4359	CP000916.1
Thermotoga petrophila RKU-1	CP000702.1
Thermotoga sp. RQ2	CP000969.1
Thermus thermophilus HB8	AP008226.1
Treponema denticola ATCC 35405	AE017226.1
Zymomonas mobilis subsp. *mobilis* ZM4	AE008692.2

3.2 Real datasets

For the evaluation using real metagenome samples from actual microbial communities, we used two metagenome samples from the guts of obese human twins (Turnbaugh et al., 2010) and the dataset of a lignocellulose-degrading community from within a cow rumen (Hess et al., 2011).

3.2.1 Human gut dataset

The contigs from both samples, TS28 and TS29, were pooled. In the same way, scaffolds from TS28 and TS29 were pooled. All scaffolds were longer than 1000 bp. The dataset was generated with a 454 GS FLX Titanium sequencer.

Properties of the real human gut dataset:

FASTA file	Sequences	Mb
Contigs	153,564	255.2
Contigs ≥ 1000 bp	63,399	187.1
Scaffolds	18,172	164.4

3.2.2 Cow rumen dataset

The same dataset as in Dröge *et al.* (Dröge, Gregor & McHardy, 2014) was used. As the scaffolds of the assembled contigs were of lower quality than the contigs, scaffolds were split into contigs at all gaps consisting of at least 200 "N" characters. We subsequently split the resulting contigs of at least 10 kb into 'chunks' of 2000 bp, resulting in at least five chunks for each contig. The dataset was generated with Illumina GAIIx and Illumina HiSeq 2000 sequencers.

Properties of the real chunked cow rumen dataset:

FASTA file	Sequences	Mb
Contigs	159,263	318.5
Scaffolds	12,192	369.4

3.3 Reference data

The NCBI taxonomy (Federhen, 2011), downloaded on 11/22/2012, was used as the reference taxonomy. The following reference databases from the NCBI were pooled to generate our reference sequence (RS) collection: NCBI genomes (downloaded on 11/22/2012), NCBI draft bacterial genomes (downloaded on 11/22/2012), the NCBI human microbiome project (downloaded on 10/16/2012) and NCBI RefSeq (Sayers et al., 2008) microbial version 56. Subsequently, duplicate sequences were removed to make the RS collection non-redundant. This RS collection contained sequences for 841 different genera, 2543 different species and 4516 different strains. The total size of the RS collection was 16 Gb.

In the marker gene (MG) analysis, the following MG sequence collections and HMM profiles were used: For the 16S and 23S MG analysis, bacterial and archaeal reference sequences from the SILVA database (Pruesse et al., 2007) were retrieved (version 111, released on 7/27/2012). The corresponding taxonomic identifiers were mapped onto the NCBI taxonomy. The resulting collection contained 126,742 sequences for 39,201 different species (199 Mb in total).

For the 5S MG analysis, MG sequences were retrieved from NCBI on 2/8/2013 via Maglott *et al.* (Maglott et al., 2004); the collection contained 12,424 sequences for 1278 species (5.8 Mb in total).

In addition, reference sequences for the following 31 bacterial marker gene families were retrieved from NCBI on 2/8/2013 via Maglott *et al.* (Maglott et al., 2004): *dnaG*, *infC*, *pgk*, *rpoB*, *tsf*, *frr*, *nusA*, *pyrG*, *rpmA*, *smpB*, *rpsC*, *rpsI*, *rpsK*, *rpsS*, *rpsB*, *rpsE*, *rpsJ*, *rpsM*, *rplA*, *rplB*, *rplC*, *rplD*, *rplE*, *rplF*, *rplK*, *rplL*, *rplM*, *rplN*, *rplP*, *rplS* and *rplT*. This MG collection contained 63,530 sequences for 1380 different species (52 Mb in total).

HMM profiles for the 16S, 23S, and 5S marker genes were retrieved from Huang *et al*. (Huang, Gilna & Li, 2009) HMM profiles trained on the protein families for the 31 bacterial MG were retrieved from Wu & Scott. (Wu & Scott, 2012)

3.4 Test environments

The benchmarks were run on different hardware configurations. When measuring runtime, Hardware Configurations 1 or 2 were used if not stated otherwise.

1. Server: AMD Opteron Processor 6386 SE, 2.8 GHz; 512 GB RAM; local SSD storage; Debian GNU/Linux 7.1.
2. Laptop: Intel i5 M520 2.4 GHz; 4 GB RAM; 7200 rpm laptop storage; Windows 7 64-bit, Ubuntu 12.04 64-bit; Oracle VirtualBox 4.2.12: 2 GB RAM, 8 GB swap, 140 GB HDD, Ubuntu 12.04 64-bit.
3. Server: Intel Xeon CPU X5660, 2.8 GHz; 73 GB RAM; network storage; Debian GNU/Linux 6.0.7.
4. Server: AMD Opteron Processor 6174, 2.2 GHz; 100 GB RAM; local storage; Debian GNU/Linux 6.0.7.
5. Laptop: Intel i5 2557M 1.7 GHz; 4GB RAM, SSD storage, OS X 10.7.

3.5 *MEGAN4* configuration

NCBI BLAST (version 2.2.27+) was used to generate alignments (Section 3.4, HW Configuration 1), using 15 threads; the tabbed output format (7) was used. *MEGAN4* (4.70.4)

(Huson et al., 2011) was used for taxonomic assignment on a laptop (Section 3.4, HW Configuration 2) using the following settings: *minsupport=5, minscore=2, toppercent=20, mincomplexity=0.44*. The runtime of *MEGAN4* was just a few seconds, as the *LCA* algorithm it uses is simple and fast. Construction of the *BLAST* database from the reference sequence collection required 6 h 55 m, with the size of the database being 4 GB. To simulate the new strain, species and genus scenarios (Table 1: Test Scenarios 5, 8 and 9), the corresponding alignments of sequences present in both the test and reference data were removed from the *BLAST* output.

Runtimes of *BLAST* for the different metagenome datasets:

Dataset	Runtime
Simulated uniform	52 m 11 s
Simulated log-normal	19 m 18 s
Chunked cow rumen (contigs)	43 m 29 s
Chunked cow rumen (scaffolds)	42 m 56 s
Human gut (contigs)	44 m 05 s
Human gut (scaffolds)	25 m 37 s

3.6 *Taxator-tk* configuration

LAST (version 287) (Frith, Hamada & Horton, 2010) was used to produce alignments using one thread, output format 1 (maf). Constructing the *LAST* database for the reference sequence database required 81 h 29 min. The size of the resulting database was 91 Gb (Section 3.4, HW Configurations 1 and 4).

Taxator-tk (Dröge, Gregor & McHardy, 2014) was then employed to process metagenome sequence fragments using 15 threads and to produce taxonomic assignments using one thread for the input sequences (Section 3.4, HW Configuration 4). For the simulated datasets, the corresponding alignments of sequences present in both the test and reference data were removed to simulate the new strain, species and genus scenarios (Table 1: Test Scenarios 5, 8 and 9).

Commands

LAST command:

```
lastal -f 1 lastDb query.fna | lastmaf2alignments.py | sort | gzip > alignments.gz
```

BLAST command:

```
blastn -db blastDb -query query.fna -num_threads 15 -outfmt '6 qseqid qstart qend qlen sseqid sstart send bitscore evalue nident length' -out alignments.blast
```

Produce fragments:

```
cat alignments.blast | alignments-filter -b 50 | taxator -a rpa -q query.fna -f ref.fna -g ref_all.tax -p 15 | sort > fragments.gff3
```

Produce assignments:

```
cat fragments.gff3 | binner > assignments.tax
```

Runtimes of *LAST* for the different metagenome datasets:

Dataset	Runtime (HC 1)	Runtime (HC 4)
Simulated uniform	9 h 56 m 27 s	12 h 10 m 57 s
Simulated log-normal	5 h 02 m 03 s	6 h 16 m 02 s
Chunked cow rumen (contigs)	12 h 23 m 29 s	15 h 39 m 24 s
Chunked cow rumen (scaffolds)	15 h 15 m 20 s	19 h 15 m 12 s
Human gut (contigs)	10 h 29 m 12 s	13 h 48 m 57 s
Human gut (scaffolds)	7 h 41 m 05 s	10 h 16 m 20 s

Runtimes of *taxator-tk* for different metagenome datasets:

Dataset	Process fragments	Bin
Simulated uniform	36 h 54 m 02 s	17.4 s
Simulated uniform (new strain)	8 h 53 m 20 s	18.2 s
Simulated uniform (new species)	4 h 44 m 27 s	18.1 s
Simulated uniform (new genus)	54 m 39 s	17.5 s
Simulated log-normal	25 h 25 m 49 s	16.8 s
Simulated log-normal (new strain)	3 h 09 m 16 s	17.9 s
Simulated log-normal (new species)	2 h 06 m 29 s	17.4 s
Simulated log-normal (new genus)	36 m 34 s	16.9 s
Chunked cow rumen (contigs)	3 h 03 m 07 s	24.9 s
Chunked cow rumen (scaffolds)	46 m 59 s	19.2 s
Human gut (contigs)	6 h 38 m 56 s	22.5 s
Human gut (scaffolds)	2 h 47 m 50 s	18.6 s

3.7 *PPS*+ and *PPS* generic model configurations

PPS+ benchmarks were run using one thread (Section 3.4, HW Configuration 3). The *PPS*+ configuration file contained in the VM distribution specifies the default values of the parameters used (configuration file name: config_ppsp_vm_refNCBI20121122_example.cfg).

PPS was run using one thread (Section 3.4, HW Configuration 3). *PPS* was trained to include the 200 most abundant genera in the reference sequences (Section 3.3). The *PPS* models were built down to the genus rank, as this is the default setting of *PPS*.

3.8 *Kraken* configuration

Kraken (version 0.10.5) and its dependency *Jellyfish* (1.1.11) were installed on a high-performance server (Section 3.4, HW Configuration 1). Four *Kraken* databases were built using our custom reference data collection (Section 3.3). For the real datasets (Section 3.2) and the simulated datasets (Sections 3.1) – for the first scenario (Table 1: Test Scenarios 1), *kraken_db_all* database was built from all the reference sequence data (Section 3.3). To simulate the new strain, new species and new genus scenarios (Table 1: Test Scenarios 5, 8 and 9), we generated corresponding *Kraken* databases *kraken_db_new_strain*, *kraken_db_new_species* and *kraken_db_new_genus*. For instance, *kraken_db_new_strain* database does not contain the strains from which the simulated datasets were generated. When we use the *kraken_db_new_strain* database, we simulate the scenario in which all strains of a metagenome sample are unknown, i.e. (Table 1: Test Scenarios 5). This approach ensures that all the methods in comparison use the same reference data for the classification in respective test scenarios (Table 1). For instance, to create the *Kraken kraken_db_all* database, we performed the following steps:

1. Create directory *for_kraken_all* containing all the reference sequences that are used to build a custom reference database. Note that the sequence names in the FASTA files have to be in the format specified in the *Kraken* documentation.
2. Create empty directory *kraken_db_all* for the generated database.
3. Inside directory *kraken_db_all*, create directory *taxonomy* and place there the following NCBI taxonomy files: *gi_taxid_nucl.dmp, names.dmp, nodes.dmp.*
4. Switch to directory *for_kraken_all* and run the following command to add all the reference sequences to the *Kraken* database *kraken_db_all*:

*for file in *.fna; do kraken-build --add-to-library $file --db kraken_db_all --threads 40; done*

5. Set the *PATH* variable to contain also the installation *bin* directory of *Jellyfish*.
6. Build the *Kraken kraken_db_all* database:
   kraken-build --build --db kraken_db_all --threads 40
7. Perform taxonomic assignment of contigs contained in FASTA file *contigs.fna* and store the results in *contigs_lab.csv*:
   kraken --preload --db kraken_db_all --threads 40 contigs.fna > contigs_lab.csv

3.9 Assignment quality measures

3.9.1 Micro-averaged precision and recall

To assess the quality of the taxonomic assignments for the simulated datasets, we evaluated the micro-averaged precision (sometimes also known as the micro-averaged specificity) and the micro-averaged recall (sometimes also known as the micro-averaged sensitivity) of taxonomic assignments for the different methods, as detailed below. Both measures were calculated based on the number of assigned bp for each taxonomic rank, instead of per assigned fragment, as the correct assignment of larger sequence fragments is more beneficial for the retrieval of "draft genome" bins than for short fragments.

The micro-averaged precision was defined as:

$$p^l = \frac{\sum_{i=1}^{N_p^l} TP_i^l}{\sum_{i=1}^{N_p^l} TP_i^l + FP_i^l};$$

and micro-averaged recall was defined as:

$$r^l = \frac{\sum_{i=1}^{N_r^l} TP_i^l}{\sum_{i=1}^{N_r^l} TP_i^l + FN_i^l},$$

where l denotes the taxonomic rank evaluated, such as species, genus, family, order, class, phylum or superkingdom; $(TP_i^l + FN_i^l)$ is the number of bp from taxon i; $(TP_i^l + FP_i^l)$ is the number of bp assigned to taxon i and TP_i^l is the number of bp correctly assigned to taxon i. The precision is micro-averaged over all bins N_p^l to which a sequence fragment was assigned and the recall is micro-averaged over all N_r^l taxa present in the simulated dataset at rank l.

The micro-averaged precision is the fraction of correctly assigned bp from all predictions for a particular taxonomic rank and represents a measure of confidence for the predictions of a method. The micro-averaged recall is the fraction of correct assignments of the test sample for a particular taxonomic rank. To avoid an uninformative increase of the micro-averaged recall by having unassigned sequences, which belong to no taxon at a given rank, our test datasets were generated from sequenced isolates with taxa defined at all major taxonomic ranks. Note that for simplification, we denoted the micro-averaged precision as 'precision' and the micro-averaged recall as 'recall' in this document.

3.9.2 Taxonomic assignment correction for assessment of bin quality

Often, a species within a metagenome sample is not directly represented among the reference sequences; however, this respective species is closely related to a species for which there is enough data in the RS or MG collections. In this case, the species from the sample may be consistently assigned to the closely related species. This error does not impact draft genome reconstruction in terms of reconstructing a bin as a set of sequences originating from the same sample population, but the assigned identifier itself is incorrect. To quantify the binning performance independently from taxonomic label assignment, we applied a correction procedure and re-computed the corrected precision and recall values: If most of the sequences (i.e. at least *(correctLabelThreshold * 100)% bp*) from one taxon were consistently assigned to a false identifier, their identifiers were changed to the correct one, and precision and recall were re-computed. The default setting for the configuration parameter *correctLabelThreshold* was 0.9. The precision and recall were always calculated with and without this correction.

3.10 Scaffold-contig consistency definitions

3.10.1 Comparison of scaffold and contig assignments

To assess the consistency of scaffold and contig assignments for a metagenome sample, we define the following measures at all major taxonomic ranks (i.e. superkingdom, phylum, class, order, family, genus and species). The idea of these measures is that each contig is assigned up to two taxonomic identifiers: one from the contig assignment and the other from the scaffold assignment. These two taxonomic labels are then compared. If we considered contigs with two identical taxonomic labels to be correctly assigned and contigs with two

distinct taxonomic labels to be as incorrectly assigned, then "% agreement" resembles a measure of precision (i.e. correctly assigned bp ÷ correctly and incorrectly assigned bp), while "kb agreement" indicates recall (i.e. the total number of correctly assigned bp).

Let us assume that a metagenome sample consists of m scaffolds $s_0, \ldots, s_{m-1}$ and n contigs $c_0, \ldots, c_{n-1}$, where scaffold s_k consists of n_k contigs $c_{k(0)}, \ldots, c_{k(n_k-1)}$. Let function l denotes the taxonomic identifier of a contig or a scaffold at the taxonomic rank being considered, i.e. $l(c_i)$ is a label of the i^{th} contig and $l(s_k)$ is the label of the k^{th} scaffold. The lengths of contig c_i and scaffold s_k are denoted by $len(c_i)$ and $len(s_k)$, respectively. Now, we can define the consistency measures 'kb agreement' (Def. 0a) and '% agreement' (Def. 0b) as:

0a) 'kb agreement':

$$a_{kb} = \sum_{k=0}^{m-1} \sum_{j \in \{k(0),\ldots,k(n_k-1)\},\ l(s_k)\ and\ l(c_j)\ defined,\ l(s_k)=l(c_j)} len(c_j);$$

0b) '% agreement':

$$a_{\%} = \frac{a_{kb}}{\sum_{j=0}^{n-1} len(c_j)}.$$

In other words, in 'kb agreement' (Def. 0a), the index *k* goes over all scaffolds, the index *j* goes over all contigs within a corresponding scaffold. If both labels of scaffold *k* and contig *j* are defined and assigned to the same taxa, then the length of contig *j* is added to the overall sum of lengths of consistently assigned contigs.

3.10.2 Taxonomic scaffold-contig assignment consistency

To provide more detailed insights into the evaluation of the binning results of real metagenome datasets, we introduced new detailed measures of the scaffold-contig consistency (described below).

We assume that all contigs $c_0, \ldots, c_{n-1}$ of a particular scaffold originated from the same organism and thus should be assigned the same taxonomic identifier. Let us denote an identifier of contig c_i as l_i. Each path p_i from the root of the taxonomy to identifier l_i represents a hypothesis about the identifier of the whole scaffold. We base our definition on the assumption that the most representative identifier of a scaffold corresponds to the path to which the identifiers of all taxonomically assigned contigs that do not lie on the path have the

shortest collective weighted distance. Note that we do not have to consider the path p_i from the root to l_i as a potential taxonomic identifier if there is a path p_j from the root to the taxonomic identifier l_j of another contig c_j for which l_i lies on p_j and $i \neq j$, as the shortest collective weighted distance of all contigs of a scaffold to path p_j is always lower than the collective weighted distance to path p_i. Let us denote the length of contig c_i as $|c_i|$ (counted in bp). Let us define the weight of contig c_i as $w_i = \frac{|c_i|}{\sum_{j=0}^{n-1} |c_j|}$. Let *tax_dist($l_i$, p_j)* be the taxonomic distance (i.e. the number of edges in the reference taxonomy) between identifier l_i and the closest identifier l_k that lies on path p_j (i.e. this is simply the distance between identifier l_k and path p_j). The weighted distance from path p_j to all other identifiers l_i is defined as: $dist(p_j) = \sum_{i=0}^{n-1} w_i * tax\_dist(l_i, p_j)$. Let p_k be the path with the minimum weighted distance (*dist*) from all other identifiers. All contigs c_i that lie on path p_k are considered to be consistently assigned within the scaffold; all contigs c_j that do not lie on the path are considered to be inconsistent. The consistency of the scaffold is then defined as:

1) Proportion of consistently assigned contigs:

$$\frac{|\{c_i \mid l_i \, on \, p_k\}|}{|\{c_i \mid i=0 \ldots n-1\}|};$$

2) Proportion of consistent contigs in bp:

$$\frac{\sum_{\{i \mid l_i \, on \, p_k\}} |c_i|}{\sum_{i=0}^{n-1} |c_i|};$$

3) Average distance to the path:

$$\frac{\sum_{i=0}^{n-1} tax\_dist(l_i, \; p_k)}{n};$$

4) Average weighted distance to the path:

$$dist(p_k);$$

5) Average distance to the scaffold identifier:

$$\frac{\sum_{i=0}^{n-1} tax\_dist(l_i, \; l_k)}{n};$$

6) Average weighted distance to the scaffold identifier:

$$\sum_{i=0}^{n-1} w_i * tax\_dist(l_i, l_k).$$

The first definition is the coarsest measure and the last is the finest for taxonomic assignment consistency.

We can also group the scaffolds using l_k and compute the measures for individual taxa. However, these groups do not correspond to the assigned bins, as a scaffold's taxonomic identifier does not always correspond to the taxonomic identifier of the lowest assigned contig of that scaffold.

The consistency of the entire sample can also be defined as the (weighted) average of these measures. Let $s_0, \ldots, s_{m-1}$ be all scaffolds in the sample, where if a contig is not assigned to a scaffold, an artificial scaffold that contains this one contig is created. We can also consider only scaffolds that contain only a certain number of contigs or those that are at least x bp long, for example.

Thus if we compute these measures for two different binning methods, we can assess the consistency of the respective taxonomic assignments at six different levels. However, be aware that it is recommended to also look at the number of bp assigned at different taxonomic ranks by each method, since the consistency of a method that assigns everything to the root of the taxonomy seems to be perfect according to these scaffold-contig consistency definitions.

4 Detailed results for the simulated datasets

This section provides a detailed description of the results of the benchmarks with simulated datasets in nine different test scenarios (Table 1). *PPS+*, *PPS* generic model, *MEGAN4* and *taxator-tk* were compared to each other in terms of precision and recall (Section 3.9). The nine different scenarios evaluate assignment performances for different evolutionary distances between the sample sequences and the available reference sequences. For instance, in (Table 1: Test Scenario 6), all sequences from the species included in the simulated communities were excluded from the reference sequence collection and all sequences of the same strains were excluded from the marker gene sequence collection.

4.1 Uniform dataset

For *PPS+*, a drop in both precision and recall was only observed for low-level taxonomic assignments when removing reference data from the same strain, species or genera from the reference sequence (RS) collection and also from the MG collection (Table 1: Test Scenarios 2, 3 and 4 versus Test Scenarios 5, 8 and 9), which demonstrated that for microbial community members that have been profiled by 16S sequencing but which have no sequenced genomes available, *PPS+* can perform highly accurate low-level taxonomic assignments, unlike from all other tested methods (Figs S1a and S1c–S1f).

In more detail, *PPS+* showed substantially higher precision and recall than the *PPS* generic model for all test scenarios (Fig. S1a–S1d, Table 1: Test Scenarios 1–9). *PPS+* also showed substantially higher precision and recall than *MEGAN4* for the assignment of sequences from new strains, species and genera (Figs S1a and S1e, Table 1: Test Scenarios 2–4), when these were represented in the reference collection as marker genes. An exception was the unrealistic case, when all of the simulated metagenome data were available in the reference sequence collection (Table 1: Test Scenario 1).

Simulating the situation where the microbial community members have not been observed in profiling before, we removed these strains from the MG collection and the reference sequences (RS) for the strains, species or genera of the simulated metagenome datasets (Table 1: Test Scenarios 5, 6 and 7). We removed more data from the reference sequence (RS) collection than from the MG collection to simulate the situation where a closer relative can be found among the marker genes and a more distant one among the sequenced genomes, as many taxa have been profiled but have not had their genomes sequenced. *PPS+* assignment quality (both precision and recall) dropped in comparison to the situation where strains have been profiled (Fig. S1a,b). However, it was still better than *MEGAN4* (Figure S1e) for all ranks, except for the lowest-level assignment (species), when the strains were removed from the RS collection only (Table 1: Test Scenario 5). As the removal of strain-level data in many cases also removed all data for the respective species from the RS collection, both methods made false assignments to related species in these scenarios.

When we removed even more reference data from the MG collection to simulate the binning of microbial community members for which no members of the same species or genera have

been profiled or sequenced before (Figure S1c, Table 1: Test Scenarios 8 and 9), the precision for ranks above remained high (Table 1: Test Scenario 8, genus rank: 88.5%; Test Scenario 9, family rank: 73.2%), while the recall dropped moderately. However, *PPS+*'s assignments were still substantially better than those of *MEGAN4* for these ranks (Figure S1e, Test Scenario 8, genus rank: 81.6%; Test Scenario 9, family rank: 58.9%). For lower ranks for which all reference data were removed, both methods had low precision and recall due to false positive assignments.

Taxator-tk showed a lower recall than *PPS+* across all tested scenarios (Figs S1a–S1c and S1f), but showed outstanding precision for the order rank and above (close to 100%), and never dropped below 89% at lower ranks. Thus this method could also be used for taxonomic profiling to determine the presence of particular taxa reliably in a given sample.

4.2 Log-normal dataset

Even though the log-normal dataset was more challenging for all the tools, this benchmark yielded similar conclusions as the benchmark with the uniform dataset.

PPS+ performed substantially better than the generic *PPS* model in terms of the precision and recall in all test scenarios (Fig. S3a–S3d, Table 1: Test Scenarios 1–9).

At low taxonomic ranks (i.e. family, genus and species), *PPS+* outperformed *MEGAN4* in terms of precision and recall in almost all test scenarios (Figs S3a–S3c and S3e, Table 1: Test Scenarios 2–9), except at the family rank in the 'new strain' scenario, where *MEGAN4* had slightly better precision (96.7%) than *PPS+* (94.8%) (Figs S3b, S3c and S3e, Table 1: Test Scenario 5). In the unrealistic case, where all reference data remained in the reference (RS and MG) collections, *MEGAN4* had better precision and recall (Figs S3a–S3c and S3e, Table 1: Test Scenario 1).

Overall, *PPS+* showed substantially better recall than *taxator-tk*, whereas *taxator-tk* showed mostly better precision (Figs S3a–S3c and S3f, Table 1: Test Scenarios 1–9). Moreover, in the case where microbial community members have been profiled by 16S but have no sequenced genomes, *PPS+* showed a very high precision at low taxonomic ranks (i.e. family, genus and species) 99.5–89.6% (Figs S3a and S3f, Table 1: Test Scenarios 2–4). In several cases, *PPS+*

showed better precision than *taxator-tk*; for example, at the family rank, the precision was 98.1% for *PPS+* vs 91.9% for *taxator-tk* (Figs S3a and S3f, Table 1: Test Scenario 4) and at the genus rank, it was (scenario 2) 96.1%, (scenario 3) 96.3% for *PPS+* vs (scenario 2) 91%, (scenario 3) 86.9% for *taxator-tk* (Figs S3a and S3f, Table 1: Test Scenarios 2, 3).

4.3 Benchmarks with corrections

In the test scenarios when the reference data were excluded from the MG database (Table 1: Test Scenarios 5–9), the precision of *PPS+* for low taxonomic ranks (i.e. genus and species) was lower than the precision of *taxator-tk* because of the way *PPS+* chooses the taxa that are modeled. If the sequences from the same strains as those of the simulated metagenome samples were removed from the MG reference database at the strain, species or genus ranks, the marker gene analysis assigned sequences of the metagenome sample that would otherwise have a very good match to the respective MG database sequences to corresponding closely related taxa.

In the subsequent *PPS* training phase, the sample-derived data were used to train closely related clades; moreover, reference sequences from closely related clades were used as training data as well. However, for the draft genome reconstruction, it is necessary to infer consistent bins from a metagenome sample. The actual identifiers of the bins are of lower importance. Therefore, we recomputed the precision and recall measures with a correction to account for the phenomenon described above (Section 3.9, Figs S2a–S2f and S4a–S4f, Table 1: Test Scenarios 1–9).

The corrected precision of *PPS+* was substantially better than it was without the correction for all scenarios. The difference for the other methods is not that pronounced, since they choose clades to which metagenome sequences are assigned in a different way. When comparing *PPS+* to *MEGAN4* using these corrections, *PPS+* showed higher precision and recall. When comparing *PPS+* to *taxator-tk*, *PPS+* had higher recall; however, neither method was consistently more precise.

5 Detailed results for the real datasets

5.1 Taxonomic scaffold-contig assignment consistency

To assess the quality of taxonomic assignments for these samples, we evaluated the consistency of taxonomic assignments for contigs originating from the same scaffold using a set of measures (Section 3.10.2). These measures assessed the degree to which the taxonomic identifiers of scaffolds and their constituent contigs were consistent relative to each other. This method looked beyond identical identifiers (Section 3.10.1) by taking the relative distances between two taxa in the reference taxonomy into account (Table S6 and S7).

The basic idea of these measures is that a scaffold is assigned to a taxonomic identifier of one of its constituent contigs, such that the collective distance of all contig assignments for the respective scaffold to path *p* in the taxonomy defined by the scaffold identifier is the shortest. The consistency of individual contig assignments is then assessed relative to path *p*: If a contig lies on *p*, it is considered to be assigned consistently; if it does not lie on *p*, it is assigned inconsistently. These measures were computed for the assignments of the chunked cow rumen and the human gut datasets.

Overall, *PPS+* performed better in terms of the consistent assignment of sequences to low taxonomic ranks for the chunked cow rumen dataset and the human gut dataset than the generic *PPS* model and *MEGAN4* (Table S6 and S7, Def. 6). For both datasets, *taxator-tk* showed the highest consistency according to almost all measures; however, it assigned fewer data to lower taxonomic ranks (family, genus and species) than the other methods.

For the chunked cow rumen dataset, the generic *PPS* model assigned more contigs consistently than *PPS+* (Table S6, Def. 2); however this came at the cost of many contigs being assigned to higher taxonomic ranks by *PPS* (Table S6, Defs 0a, 6). *MEGAN4* showed a higher overall consistency than *PPS+* (Table S6, Def. 2) but this was mostly due to many contigs being assigned at higher taxonomic ranks (Table S6, Def. 6). For lower taxonomic ranks or when also taking sequence length into account (instead of the number of assigned sequences), *MEGAN4* was less consistent than *PPS+* (Table S6, Defs 0b, 3–6).

For the human gut dataset, *PPS*+ performed better than the generic *PPS* model according to all measures (Table S7, Def. 0–6). *PPS*+ was again more consistent than *MEGAN4* when taking sequence lengths into account (Table S7, Defs 2, 4, 6). These measures are more informative for taxonomic binning than the sequence-count based measures (Table S7, Defs 1, 3, 5), as obtaining large bins is desirable. These results also imply that *MEGAN4* assigned substantially more (predominantly short) sequences to lower taxonomic ranks than *PPS*+ (Table S7, Def. 0a).

5.2 Evaluation summary

Our evaluation showed that *PPS*+ performed substantially better than the generic *PPS* model (Tables 2–3, S6–S7). Moreover, the results of *PPS*+ were comparable to a sample-derived model generated according to expert specifications (Table 4). *Taxator-tk* had the highest consistency of all the methods; however, it assigned substantially fewer sequences to low taxonomic ranks than the other methods (Tables 2–3, S6–S7). Our benchmark experiments also confirmed that if the metagenome sequences were closely related to the reference sequences, such as for the human gut dataset, the homology-based methods assigned more sequences correctly to low taxonomic ranks than they did across larger taxonomic distances, as was the case for the cow rumen dataset (Tables 2–3, S6–S7). *PPS*+ was not that sensitive to this distance. For *PPS*+, only few taxonomically informative marker genes have to be identified from the sample, for which a substantially larger marker gene reference collection exists than that for genome and draft genome sequences, in terms of the number of species represented in the reference collection. *PPS*+ often made more consistent assignments than *MEGAN4* and often assigned the most sequences of all the tested methods to lower taxonomic ranks (Tables 2–3, S6–S7).

5.3 Throughput comparison

The throughput of the individual methods for contig assignments of the human gut sample was calculated as either Mb or the number of sequences assigned per hour with one thread using the same reference sequences (Sections 3.3 and 3.4). *PPS* and *PPS*+ directly use sequences in FASTA format as references, while for *MEGAN4* and *taxator-tk BLAST* or *LAST* databases were initially constructed. Database construction took 6 h 55 m and 81 h 29 min on

our servers, respectively for *BLAST* and *LAST*, and was not considered in runtime comparisons. As most time in *PPS+* is spent with model construction, assignment can be further accelerated when reusing models to classify multiple metagenome samples. In this setting, where we consider only the prediction phase of *PPS+*, *PPS+* was more than 7 times faster (up to 0.5 Gb/h) than the homology-based methods (Figure 4). As only a relatively small reference sequence database of 16 Gb was used, runtimes of *BLAST* and *LAST* searches in the homology-based tools would proportionally increase when using larger reference collections.

Unlike the homology-based tools, for which similarity searches require the use of more hardware with more CPUs and main memory, *PPS+* can run on a standard laptop computer. *PPS+* on a laptop with an Intel i5 M520 2.4 GHz processor and 4 GB of RAM was ~1.5–4 times slower than it was on the server with an AMD Opteron 6386 SE 2.8 GHz processor and 512 GB of RAM, mainly due to insufficient RAM on the laptop installed, which caused extensive use of the swap space.

6 External tools

6.1 HMMER 3

The search command (*hmmsearch*) of the *HMMER 3* (Eddy, 2011) package with e-value cut-off set to 0.01 is used.

6.2 MOTHUR

The *MOTHUR* (Schloss et al., 2009) Naïve Bayes classifier with the following default parameters is used. The number of bootstrap replicas is set to 300. The corresponding confidence score cut-off is set to 80. For the 16S analysis (i.e. 3 (5S, 16S, 23S) out of 34 marker genes), a small part of the code from (Huang, Gilna & Li, 2009) was used.

7 Evaluation of the *CLARK* software

CLARK (Ounit et al., 2015) is a straightforward, fast, taxonomy-free, *k*-mer based binning tool for metagenome reads and contigs. It is a taxonomy-free tool, since a user has to first

decide, on which taxonomic rank s/he would like to assign sequences of a metagenome sample, and then a taxonomic identifier is assigned to all input sequences at a particular taxonomic rank. However, this is different from taxonomic binning tools such as *PPS+*, *taxator-tk* (Dröge, Gregor & McHardy, 2014), *MEGAN* (Huson et al., 2011), or *Kraken* (Wood & Salzberg, 2014), since a taxonomic binning tool first has to automatically decide on a taxonomic rank on which a sequence will be assigned and then it assigns a taxonomic identifier to the sequence at a particular rank. In *CLARK*, the first step has to be done manually, which makes the tool unsuitable for the analysis of metagenome samples originating from novel environments. For instance, if a metagenome sample contained species, that were all novel species and a user decided to assign all the sequences of the sample at the species rank, then all the assignments would have been incorrect. Therefore, it is an indispensable feature of a taxonomic metagenome binning tool to also automatically and correctly determine a taxonomic rank of an assignment. This makes the application of *CLARK* limited only to the environments that has been well studied, for which there have been many reference (draft) genomes sequenced, and that does not contain novel taxa. For such, well studied, environments, *CLARK* offers a substantial speed-up in comparison to, e.g. *BLAST* (Camacho et al., 2009). Nevertheless, it is unsuitable for the analysis of metagenome samples originating from novel environments.

We have evaluated *CLARK* in the "new strain", "new species", and "new genus" scenarios with a simulated dataset with uniform distribution (Section 3.1). For the "new strain" scenario, we have excluded all the strains of the simulated dataset from the reference sequence collection and built the *CLARK* reference database at the species rank. In this "new strain" scenario, the precision of *CLARK* at the species rank was 36.8% and recall 24.7%. The corrected measures were 57.3% and 37.6%, respectively (Section 3.9).

For the "new species" scenario, we have excluded all the species of the simulated dataset from the reference sequence collection and built the *CLARK* reference database at the genus rank. In this "new species" scenario, the precision at the genus rank was 83.2% and recall 57.9%. The corrected measures were 85.1% and 59.6%, respectively.

For the "new genus" scenario, we have built the *CLARK* reference database at the family rank. In this "new genus" scenario, the precision at the family rank was 57.3% and recall 33.3%. The corrected measures were 57.6% and 33.8%, respectively.

Note that these precision and recall values cannot be directly compared to the results of other taxonomic binning methods, as we have manually determined, on which taxonomic rank the

assignments were made by building the *CLARK* reference database at a particular rank. However, if the *CLARK* was extended from taxonomy-free binning software to a taxonomy binning software, its performance would be similar to *Kraken*, as both methods are based on the occurrences of long *k*-mers (k≈31).

8 References

Albertsen M, Hugenholtz P, Skarshewski A, Nielsen KL, Tyson GW, Nielsen PH 2013. Genome sequences of rare, uncultured bacteria obtained by differential coverage binning of multiple metagenomes. *Nature Biotechnology* 31:533–538.

Audano P, Vannberg F 2014. KAnalyze: a fast versatile pipelined K-mer toolkit. *Bioinformatics* 30:2070–2072.

Camacho C, Coulouris G, Avagyan V, Ma N, Papadopoulos J, Bealer K, Madden TL 2009. BLAST+: architecture and applications. *BMC bioinformatics* 10:421–421.

Debruijn I 2014. MetAssemble. *Available at https://github.com/inodb/metassemble/* (accessed January 23, 2014).

Dröge J, Gregor I, McHardy AC 2014. Taxator-tk: Fast and Reliable Taxonomic Assignment of Metagenomes by Approximating Evolutionary Neighborhoods. *Bioinformatics*.

Eddy SR 2011. Accelerated Profile HMM Searches. *PLoS Computational Biology* 7:e1002195–e1002195.

Federhen S 2011. The NCBI Taxonomy database. *Nucleic Acids Research* 40:D136–D143.

Frith MC, Hamada M, Horton P 2010. Parameters for accurate genome alignment. *BMC Bioinformatics* 11:80.

Hess M, Sczyrba A, Egan R, Kim T-W, Chokhawala H, Schroth G, Luo S, Clark DS, Chen F, Zhang T, Mackie RI, Pennacchio LA, Tringe SG, Visel A, Woyke T, Wang Z, Rubin EM 2011. Metagenomic discovery of biomass-degrading genes and genomes from cow rumen. *Science* 331:463–467.

Hu X, Yuan J, Shi Y, Lu J, Liu B, Li Z, Chen Y, Mu D, Zhang H, Li N, Yue Z, Bai F, Li H, Fan W 2012. pIRS: Profile-based Illumina pair-end reads simulator. *Bioinformatics* 28:1533–1535.

Huang W, Li L, Myers JR, Marth GT 2012. ART: a next-generation sequencing read

simulator. *Bioinformatics* 28:593–594.

Huang Y, Gilna P, Li W 2009. Identification of ribosomal RNA genes in metagenomic fragments. *Bioinformatics* 25:1338–1340.

Huson DH, Mitra S, Ruscheweyh HJ, Weber N, Schuster SC 2011. Integrative analysis of environmental sequences using MEGAN4. *Genome Research* 21:1552–1560.

Imelfort M, Parks D, Ben J Woodcroft, Dennis P, Hugenholtz P, Tyson GW 2014. GroopM: An automated tool for the recovery of population genomes from related metagenomes. *PeerJ*.

Iverson V, Morris RM, Frazar CD, Berthiaume CT, Morales RL, Armbrust EV 2012. Untangling Genomes from Metagenomes: Revealing an Uncultured Class of Marine Euryarchaeota. *Science* 335:587–590.

Luo R, Liu B, Xie Y, Li Z, Huang W, Yuan J, He G, Chen Y, Pan Q, Liu Y, Tang J, Wu G, Zhang H, Shi Y, Liu Y, Yu C, Wang B, Lu Y, Han C, Cheung DW, Yiu S-M, Peng S, Xiaoqian Z, Liu G, Liao X, Li Y, Yang H, Wang J, Lam TW, Wang J 2012. SOAPdenovo2: an empirically improved memory-efficient short-read de novo assembler. *GigaScience* 1:1–6.

Maglott DD, Ostell JJ, Pruitt KDK, Tatusova TT 2004. Entrez Gene: gene-centered information at NCBI. *Nucleic Acids Research* 33:D54–D58.

Marcais G, Kingsford C 2011. A fast, lock-free approach for efficient parallel counting of occurrences of k-mers. *Bioinformatics* 27:764–770.

Mavromatis K, Ivanova N, Barry K, Shapiro H, Goltsman E, McHardy AC, Rigoutsos I, Salamov A, Korzeniewski F, Land M, Lapidus A, Grigoriev I, Richardson P, Hugenholtz P, Kyrpides NC 2007. Use of simulated data sets to evaluate the fidelity of metagenomic processing methods. *Nature Methods* 4:495–500.

McHardy AC, Martín HG, Tsirigos A, Hugenholtz P, Rigoutsos I 2007. Accurate phylogenetic classification of variable-length DNA fragments. *Nature Methods* 4:63–72.

Namiki T, Hachiya T, Tanaka H, Sakakibara Y 2012. MetaVelvet: an extension of Velvet assembler to de novo metagenome assembly from short sequence reads. *Nucleic Acids Research* 1–12.

Ounit R, Wanamaker S, Close TJ, Lonardi S 2015. CLARK: fast and accurate classification of metagenomic and genomic sequences using discriminative k-mers. *BMC Genomics* 16:236.

Patil KR, Haider P, Pope PB, Turnbaugh PJ, Morrison M, Scheffer T, McHardy AC 2011. Taxonomic metagenome sequence assignment with structured output models. *Nature*

Methods 8:191–192.

Pope PB, Smith W, Denman SE, Tringe SG, Barry K, Hugenholtz P, McSweeney CS, McHardy AC, Morrison M 2011. Isolation of Succinivibrionaceae implicated in low methane emissions from Tammar wallabies. *Science* 333:646–648.

Pruesse E, Quast C, Knittel K, Fuchs BM, Ludwig W, Peplies J, Glöckner FO 2007. SILVA: a comprehensive online resource for quality checked and aligned ribosomal RNA sequence data compatible with ARB. *Nucleic Acids Research* 35:7188–7196.

Richter DC, Ott F, Auch AF, Schmid R, Huson DH 2007. MetaSim: a sequencing simulator for genomics and metagenomics. *CORD Conference Proceedings* 3:e3373–e3373.

Roche 2014.Newbler. *Available at http://www.454.com/products/analysis-software/* (accessed January 23, 2014).

Sayers EW, Barrett T, Benson DA, Bryant SH, Canese K, Chetvernin V, Church DM, DiCuccio M, Edgar R, Federhen S, Feolo M, Geer LY, Helmberg W, Kapustin Y, Landsman D, Lipman DJ, Madden TL, Maglott DR, Miller V, Mizrachi I, Ostell J, Pruitt KD, Schuler GD, Sequeira E, Sherry ST, Shumway M, Sirotkin K, Souvorov A, Starchenko G, Tatusova TA, Wagner L, Yaschenko E, Ye J 2008. Database resources of the National Center for Biotechnology Information. *Nucleic Acids Research* 37:D5–15.

Schloss PD, Westcott SL, Ryabin T, Hall JR, Hartmann M, Hollister EB, Lesniewski RA, Oakley BB, Parks DH, Robinson CJ, Sahl JW, Stres B, Thallinger GG, Van Horn DJ, Weber CF 2009. Introducing mothur: open-source, platform-independent, community-supported software for describing and comparing microbial communities. *Applied and Environmental Microbiology* 75:7537–7541.

Treangen TJ, Sommer DD, Angly FE, Koren S, Pop M 2011. Next generation sequence assembly with AMOS. *Current Protocols in Bioinformatics* 11:11.8.1–11.8.18.

Turnbaugh PJ, Quince C, Faith JJ, McHardy AC, Yatsunenko T, Niazi F, Affourtit J, Egholm M, Henrissat B, Knight R, Gordon JI 2010. Organismal, genetic, and transcriptional variation in the deeply sequenced gut microbiomes of identical twins. *Proceedings of the National Academy of Sciences of the United States of America* 107:7503–7508.

Wood DE, Salzberg SL 2014. Kraken: ultrafast metagenomic sequence classification using exact alignments. *Genome Biology* 15:R46.

Wu M, Scott AJ 2012. Phylogenomic analysis of bacterial and archaeal sequences with AMPHORA2. *Bioinformatics* 28:1033–1034.

Zerbino DR, Birney EE 2008. Velvet: algorithms for de novo short read assembly using de Bruijn graphs. *Genome Research* 18:821–829.

Supplemental Text S2

PhyloPythiaS+: A Self-Training Method for the Rapid Reconstruction of Low-Ranking Taxonomic Bins from Metagenomes

Personal communication from Dr. P. B. Pope.

Norges miljø- og biovitenskapelige universitet

DEPARTMENT OF CHEMISTRY, BIOTECHNOLOGY AND FOOD SCIENCE

P.O. BOX 5003
N-1432 ÅS

TELEPHONE: +47 6494 6232
E-MAIL: phil.pope@nmbu.no
E-MAIL: phillip.b.pope@gmail.com

October 7, 2014

2810
2005

To Whom It May Concern:

A PPS+ binning of shotgun metagenome samples indicated the likely metabolite flow and participating microbial phylotypes for a biogas-producing microbial community tolerant of high ammonia levels

Methane is the energy-rich component of biogas and is formed as the end product during anaerobic degradation of organic material in bioreactors by a consortium of mainly uncultured microorganisms. One of the key problems in biogas reactors are high ammonia levels, which are associated with unstable process performance and increased risk of process failure. Therefore, characterizing the microbiome structure and function within a stable biogas reactor operating at high ammonia levels (run on slaughterhouse and industrial lignocellulosic waste: SwRI-ha) was of considerable interest to us. From two replicate reactor samples we generated approximately 48 Gb of shotgun sequence using paired-end Illumina HiSeq sequencing and assembled these with SOAPdenovo. **PPS+ was then applied for taxon-bin recovery, which reconstructed and taxonomically assigned eight draft genomes bins (Table 1)**, including uncultured phylotypes of species representing syntrophic acetate-oxidizing bacteria, methanogens (non-acetoclastic) and different fermentative bacteria (carbohydrate and amino-acid). These bins thus likely represent organisms known to produce acetate from the reactor substrate, organisms known to convert the acetate to carbon dioxide and hydrogen gas, as well as for organisms producing methane from carbon dioxide and hydrogen gas as opposed to acetoclastic methanogens. A functional analysis of these bins revealed some of the essential genes for each of these pathways, in support of their putative roles. Thus, the taxonomic bins reconstructed with PPS+ from the shotgun metagenome samples allowed us to determine the likely metabolite flow from the substrates to the end product for a unique biogas-producing microbial community tolerant of high ammonia levels (Frank and Pope, personal communication).

Sincerely,

Phillip B. Pope

Department of Chemistry, Biotechnology and Food Science
Norwegian University of Life Sciences
Post Office Box 5003
1432, Ås
Norway
Phone: +47 6496 6232
Email: phil.pope@nmbu.no

Norges miljø- og
biovitenskapelige
universitet

TABLE 1

2810
2005

Genome ID	Base Pairs	Contigs	TaxonId	Scientific Name
pTaa-1	1358596	377	499229	Tepidanaerobacter acetatoxydans
pBah-1	910935	143	86665	Bacillus halodurans
pMcb-1	534996	177	83986	Methanoculleus bourgensis
pSmw-1	487779	158	863	Syntrophomonas wolfei
pMsb-1	292500	131	2208	Methanosarcina barkeri
pMml-1	292331	96	1080712	Methanomassiliicoccus luminyensis
pMcm-1	99651	53	2198	Methanoculleus marisnigri
pAbc-1	90549	53	81468	Aminobacterium colombiense

NORWEGIAN UNIVERSITY OF LIFE SCIENCES
P.O.BOX 5003, NO-1432 AAS, NORWAY
www.umb.no
postmottak@umb.no
TEL +47 64 96 50 00
FAX +47 64 96 50 01

Supplemental Datasets S1–S6

PhyloPythiaS+: A Self-Training Method for the Rapid Reconstruction of Low-Ranking Taxonomic Bins from Metagenomes

This document describes how to configure the software to reproduce the results.

Download datasets

The datasets for the benchmarks can be downloaded from:
https://github.com/algbioi/datasets

Supplemental Dataset S1: Simulated dataset with uniform distribution.

Supplemental Dataset S2: Simulated dataset with log-normal distribution.

Supplemental Dataset S3: Contigs of the real chunked cow rumen dataset.

Supplemental Dataset S4: Scaffolds of the real chunked cow rumen dataset.

Supplemental Dataset S5: Contigs of the real human gut dataset.

Supplemental Dataset S6: Scaffolds of the real human gut dataset.

Each file is a 7z archive and can be extracted, e.g. using command: `7za x archive.7z`
Each extracted directory contains a `readme.txt` file describing all the files contained in the directory.

Software installation

Follow the installation instructions and go through the tutorial. Both can be found here: https://github.com/algbioi/ppsp/wiki

Real datasets

Follow the tutorial:

- Create the pipeline directory in directory: `/apps/pps/tests`
- Use configuration file:
  `/apps/pps/tools/config_ppsp_vm_refNCBI20121122_example.cfg`
  as a template (i.e. copy this file and modify it appropriately).
- Make sure, you set the following parameters in the configuration file: `pipelineDir`

inputFastaFile
inputFastaScaffoldsFile
scaffoldsToContigsMapFile

- Run the pipeline using command:

```
ppsp -c CONFIGURATION_FILE -n -g -o s16 mg -t -p c s v -r -s
```

- Analyze the results as described in the tutorial.

Simulated datasets

Follow the tutorial:

- Create the pipeline directory in directory: /apps/pps/tests
- Use configuration file:
  /apps/pps/tools/config_ppsp_vm_refNCBI20121122_example.cfg
  as a template (i.e. copy this file and modify it appropriately).
- Make sure, you set the following parameters in the configuration file:
  pipelineDir
  inputFastaFile
  referencePlacementFileOut
  excludeRefSeqRank (e.g. excludeRefSeqRank=species)
  excludeRefMgRank (e.g. excludeRefSeqRank=strain)
- Run the pipeline using command:

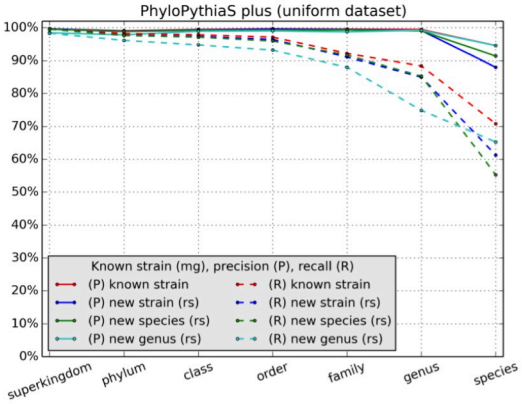
```
ppsp -c CONFIGURATION_FILE -n -g -o s16 mg -t -p c -r -s
```

- Analyze the results as described in the tutorial

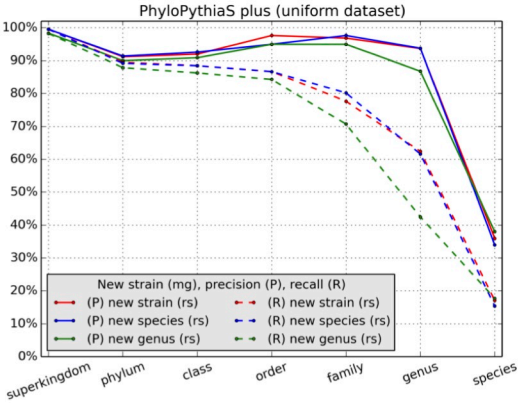
Supplemental Figures S1 – S14

PhyloPythiaS+: A Self-Training Method for the Rapid Reconstruction of Low-Ranking Taxonomic Bins from Metagenomes

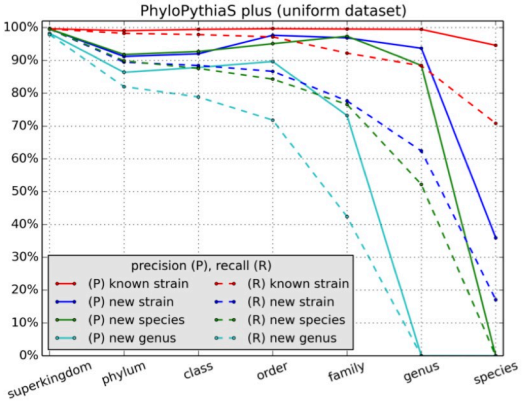
a


PhyloPythiaS plus (uniform dataset)
Known strain (mg), precision (P), recall (R)
(P) known strain
(R) known strain
(P) new strain (rs)
(R) new strain (rs)
(P) new species (rs)
(R) new species (rs)
(P) new genus (rs)
(R) new genus (rs)
superkingdom
phylum
class
order
family
genus
species

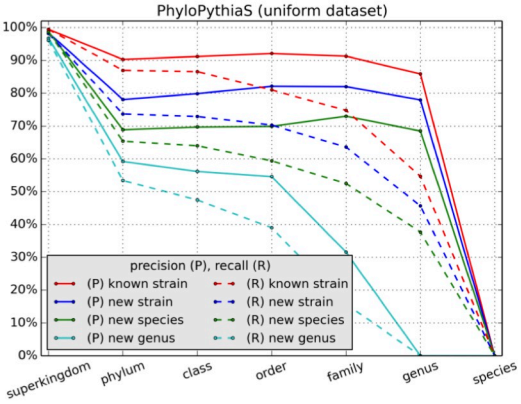
b


PhyloPythiaS plus (uniform dataset)
New strain (mg), precision (P), recall (R)
(P) new strain (rs)
(R) new strain (rs)
(P) new species (rs)
(R) new species (rs)
(P) new genus (rs)
(R) new genus (rs)
superkingdom
phylum
class
order
family
genus
species

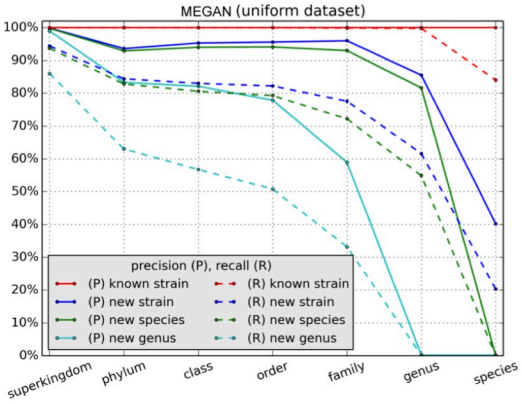
c


PhyloPythiaS plus (uniform dataset)
precision (P), recall (R)
(P) known strain
(R) known strain
(P) new strain
(R) new strain
(P) new species
(R) new species
(P) new genus
(R) new genus
superkingdom
phylum
class
order
family
genus
species

d


PhyloPythiaS (uniform dataset)
precision (P), recall (R)
(P) known strain
(R) known strain
(P) new strain
(R) new strain
(P) new species
(R) new species
(P) new genus
(R) new genus
superkingdom
phylum
class
order
family
genus
species

e


MEGAN (uniform dataset)
precision (P), recall (R)
(P) known strain
(R) known strain
(P) new strain
(R) new strain
(P) new species
(R) new species
(P) new genus
(R) new genus
superkingdom
phylum
class
order
family
genus
species

f

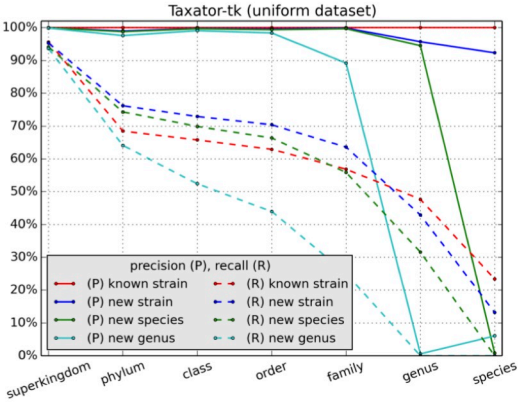
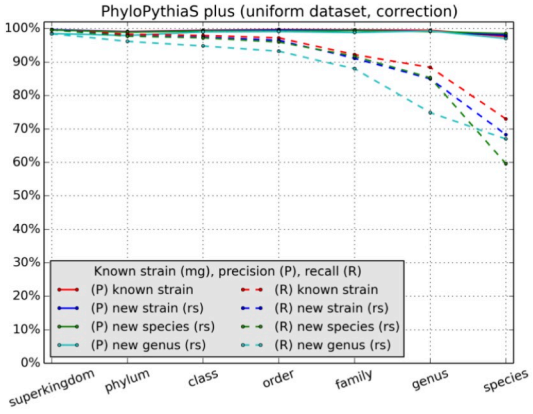

Taxator-tk (uniform dataset)
precision (P), recall (R)
(P) known strain
(R) known strain
(P) new strain
(R) new strain
(P) new species
(R) new species
(P) new genus
(R) new genus
superkingdom
phylum
class
order
family
genus
species

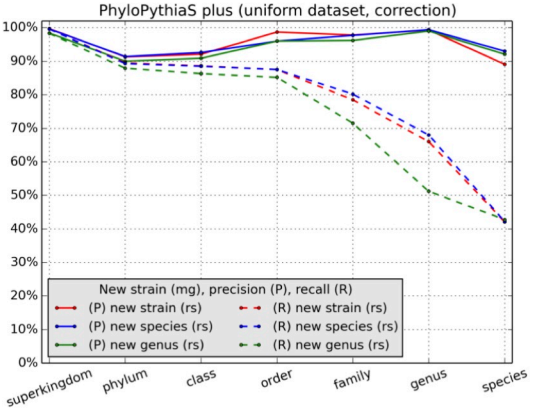
Figure S1. Benchmark results for the simulated dataset with uniform distribution.

Precision (P) and recall (R) (Supplemental Text S1, Section 3.9.1) at different taxonomic ranks were calculated for (panels a–c) *PPS+*, (panel d) the generic *PPS* model, (panel e) *MEGAN4* and (panel f) *taxator-tk* in all test scenarios (Table 1: Test Scenarios 1–9). In parentheses, (mg) and (rs) denote whether the sequences at a given taxonomic rank were masked from the marker gene or from the reference sequence collections, respectively (Supplemental Text S1, Sections 3.1 and 3.3). If not stated, sequences were masked from both reference collections.

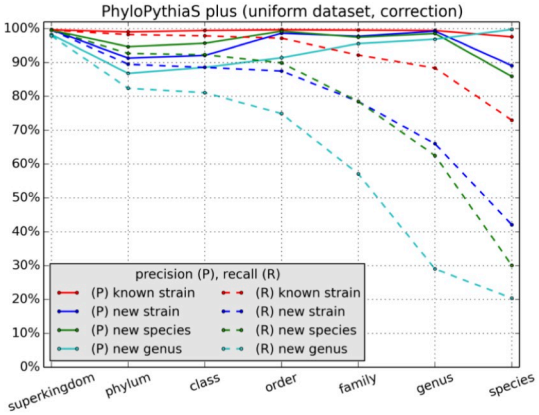
a


PhyloPythiaS plus (uniform dataset, correction)
Known strain (mg), precision (P), recall (R)
(P) known strain
(R) known strain
(P) new strain (rs)
(R) new strain (rs)
(P) new species (rs)
(R) new species (rs)
(P) new genus (rs)
(R) new genus (rs)
superkingdom
phylum
class
order
family
genus
species

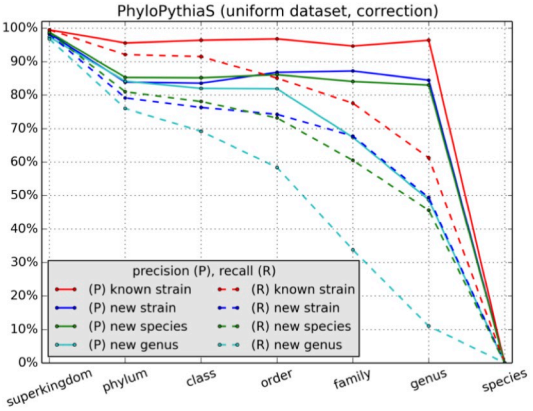
b


PhyloPythiaS plus (uniform dataset, correction)
New strain (mg), precision (P), recall (R)
(P) new strain (rs)
(R) new strain (rs)
(P) new species (rs)
(R) new species (rs)
(P) new genus (rs)
(R) new genus (rs)
superkingdom
phylum
class
order
family
genus
species

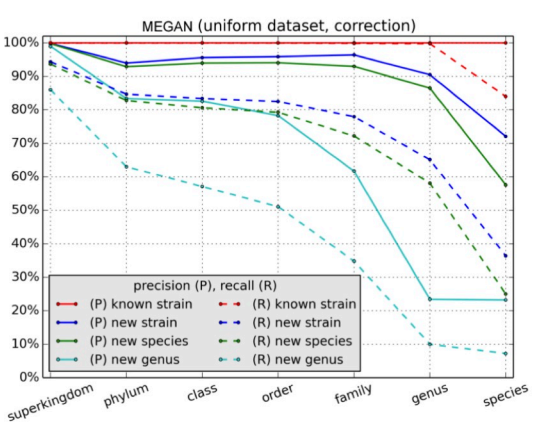
c


PhyloPythiaS plus (uniform dataset, correction)
precision (P), recall (R)
(P) known strain
(R) known strain
(P) new strain
(R) new strain
(P) new species
(R) new species
(P) new genus
(R) new genus
superkingdom
phylum
class
order
family
genus
species

d


PhyloPythiaS (uniform dataset, correction)
precision (P), recall (R)
(P) known strain
(R) known strain
(P) new strain
(R) new strain
(P) new species
(R) new species
(P) new genus
(R) new genus
superkingdom
phylum
class
order
family
genus
species

e


MEGAN (uniform dataset, correction)
precision (P), recall (R)
(P) known strain
(R) known strain
(P) new strain
(R) new strain
(P) new species
(R) new species
(P) new genus
(R) new genus
superkingdom
phylum
class
order
family
genus
species

f

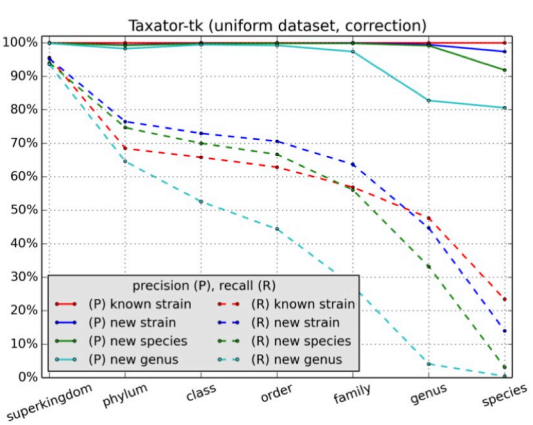
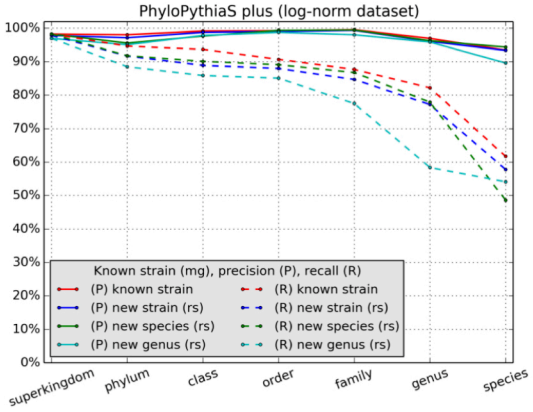

Taxator-tk (uniform dataset, correction)
precision (P), recall (R)
(P) known strain
(R) known strain
(P) new strain
(R) new strain
(P) new species
(R) new species
(P) new genus
(R) new genus
superkingdom
phylum
class
order
family
genus
species

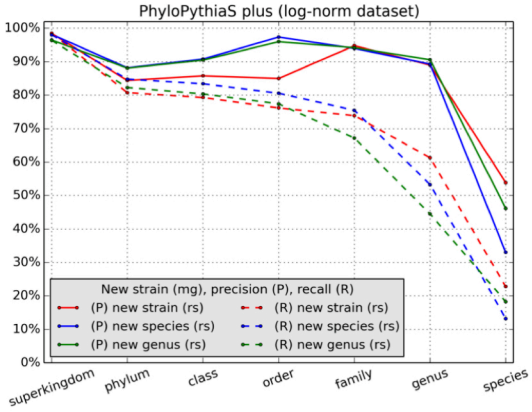
Figure S2. Benchmark results for the simulated dataset with uniform distribution using 'correction'.

Precision (P) and recall (R) were calculated with a 'correction' (Supplemental Text S1, Section 3.9) at different taxonomic ranks for (panels a–c) *PPS+*, (panel d) the generic *PPS* model, (panel e) *MEGAN4* and (panel f) *taxator-tk* in all test scenarios (Table 1: Test Scenarios 1–9, Supplemental Text S1, Section 3.1).

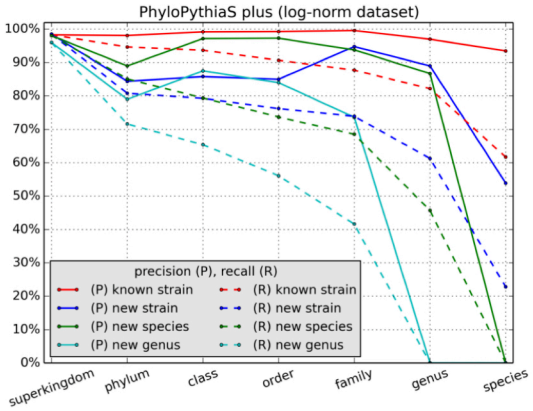
a


PhyloPythiaS plus (log-norm dataset)
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
Known strain (mg), precision (P), recall (R)
(P) known strain
(R) known strain
(P) new strain (rs)
(R) new strain (rs)
(P) new species (rs)
(R) new species (rs)
(P) new genus (rs)
(R) new genus (rs)
superkingdom
phylum
class
order
family
genus
species

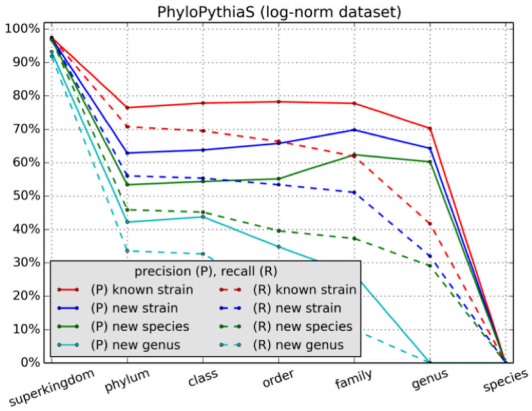
b


PhyloPythiaS plus (log-norm dataset)
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
New strain (mg), precision (P), recall (R)
(P) new strain (rs)
(R) new strain (rs)
(P) new species (rs)
(R) new species (rs)
(P) new genus (rs)
(R) new genus (rs)
superkingdom
phylum
class
order
family
genus
species

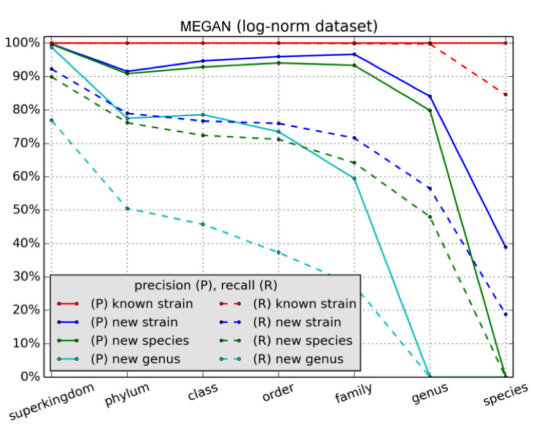
c


PhyloPythiaS plus (log-norm dataset)
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
precision (P), recall (R)
(P) known strain
(R) known strain
(P) new strain
(R) new strain
(P) new species
(R) new species
(P) new genus
(R) new genus
superkingdom
phylum
class
order
family
genus
species

d


PhyloPythiaS (log-norm dataset)
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
precision (P), recall (R)
(P) known strain
(R) known strain
(P) new strain
(R) new strain
(P) new species
(R) new species
(P) new genus
(R) new genus
superkingdom
phylum
class
order
family
genus
species

e


MEGAN (log-norm dataset)
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
precision (P), recall (R)
(P) known strain
(R) known strain
(P) new strain
(R) new strain
(P) new species
(R) new species
(P) new genus
(R) new genus
superkingdom
phylum
class
order
family
genus
species

f

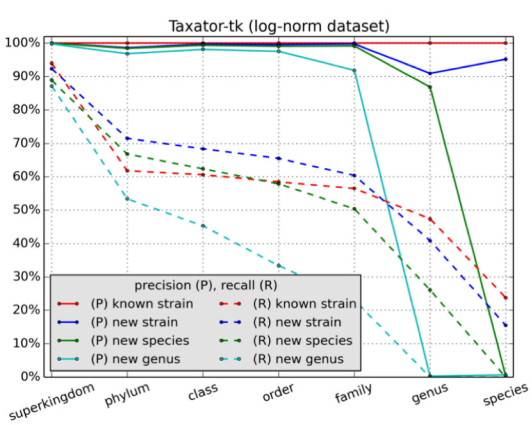
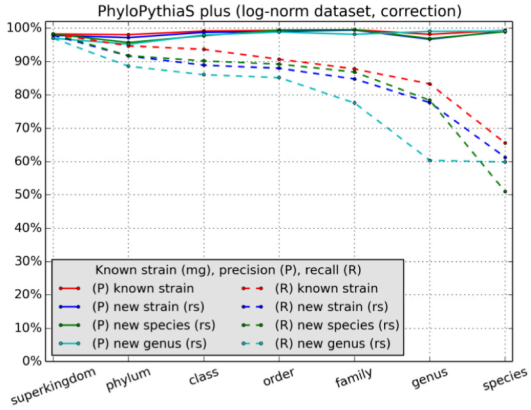

Taxator-tk (log-norm dataset)
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
precision (P), recall (R)
(P) known strain
(R) known strain
(P) new strain
(R) new strain
(P) new species
(R) new species
(P) new genus
(R) new genus
superkingdom
phylum
class
order
family
genus
species

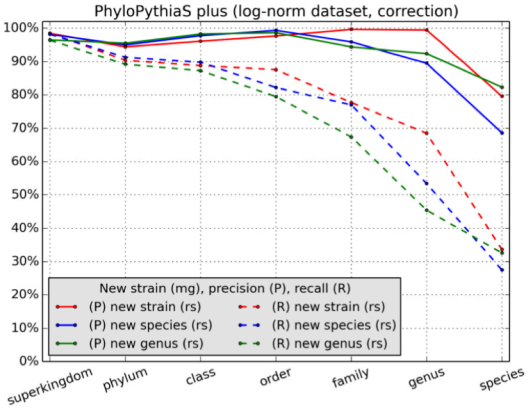
Figure S3. Benchmark results for the simulated dataset with the log-normal distribution.

Precision (P) and recall (R) (Section 3.9.1) at different taxonomic ranks were calculated for (panels a–c) *PPS+*, (panel d) the generic *PPS* model, (panel e) *MEGAN4* and (panel f) *taxator-tk* in all test scenarios (Table 1: Test Scenarios 1–9, Supplemental Text S1, Section 3.1).

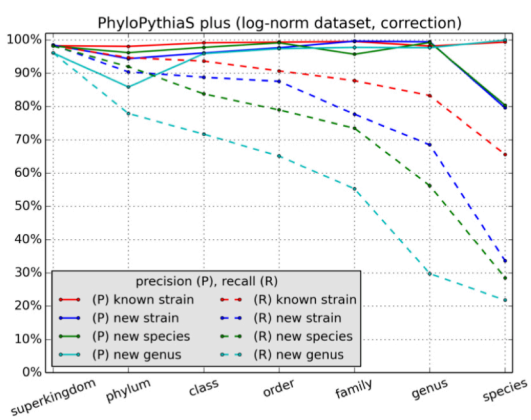
a


PhyloPythiaS plus (log-norm dataset, correction)
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
Known strain (mg), precision (P), recall (R)
(P) known strain
(R) known strain
(P) new strain (rs)
(R) new strain (rs)
(P) new species (rs)
(R) new species (rs)
(P) new genus (rs)
(R) new genus (rs)
superkingdom
phylum
class
order
family
genus
species

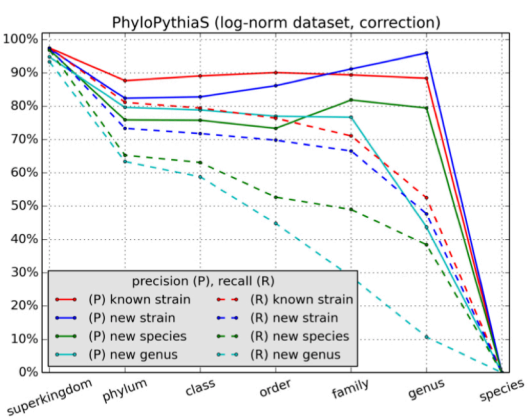
b


PhyloPythiaS plus (log-norm dataset, correction)
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
New strain (mg), precision (P), recall (R)
(P) new strain (rs)
(R) new strain (rs)
(P) new species (rs)
(R) new species (rs)
(P) new genus (rs)
(R) new genus (rs)
superkingdom
phylum
class
order
family
genus
species

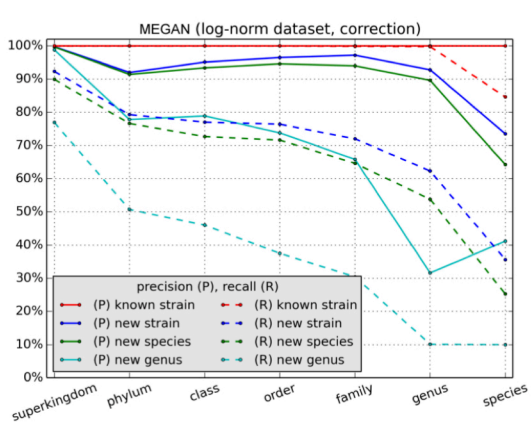
c


PhyloPythiaS plus (log-norm dataset, correction)
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
precision (P), recall (R)
(P) known strain
(R) known strain
(P) new strain
(R) new strain
(P) new species
(R) new species
(P) new genus
(R) new genus
superkingdom
phylum
class
order
family
genus
species

d


PhyloPythiaS (log-norm dataset, correction)
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
precision (P), recall (R)
(P) known strain
(R) known strain
(P) new strain
(R) new strain
(P) new species
(R) new species
(P) new genus
(R) new genus
superkingdom
phylum
class
order
family
genus
species

e


MEGAN (log-norm dataset, correction)
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
precision (P), recall (R)
(P) known strain
(R) known strain
(P) new strain
(R) new strain
(P) new species
(R) new species
(P) new genus
(R) new genus
superkingdom
phylum
class
order
family
genus
species

f

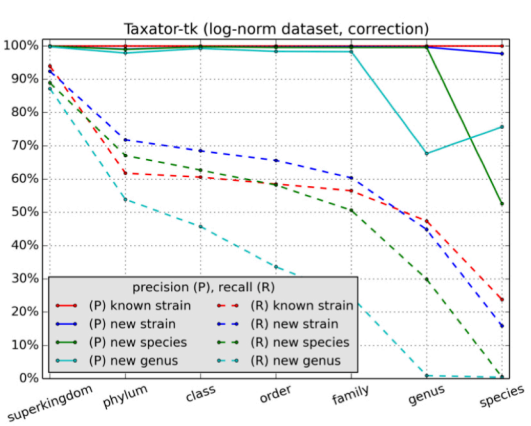

Taxator-tk (log-norm dataset, correction)
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
precision (P), recall (R)
(P) known strain
(R) known strain
(P) new strain
(R) new strain
(P) new species
(R) new species
(P) new genus
(R) new genus
superkingdom
phylum
class
order
family
genus
species

Figure S4. Benchmark results for the simulated dataset with the log-normal distribution using 'correction'.

Precision (P) and recall (R) were calculated with a 'correction' (Supplemental Text S1, Section 3.9) at different taxonomic ranks for (panels a–c) *PPS*+, (panel d) the generic *PPS* model, (panel e) *MEGAN4* and (panel f) *taxator-tk* in all test scenarios (Table 1: Test Scenarios 1–9, Supplemental Text S1, Section 3.1).

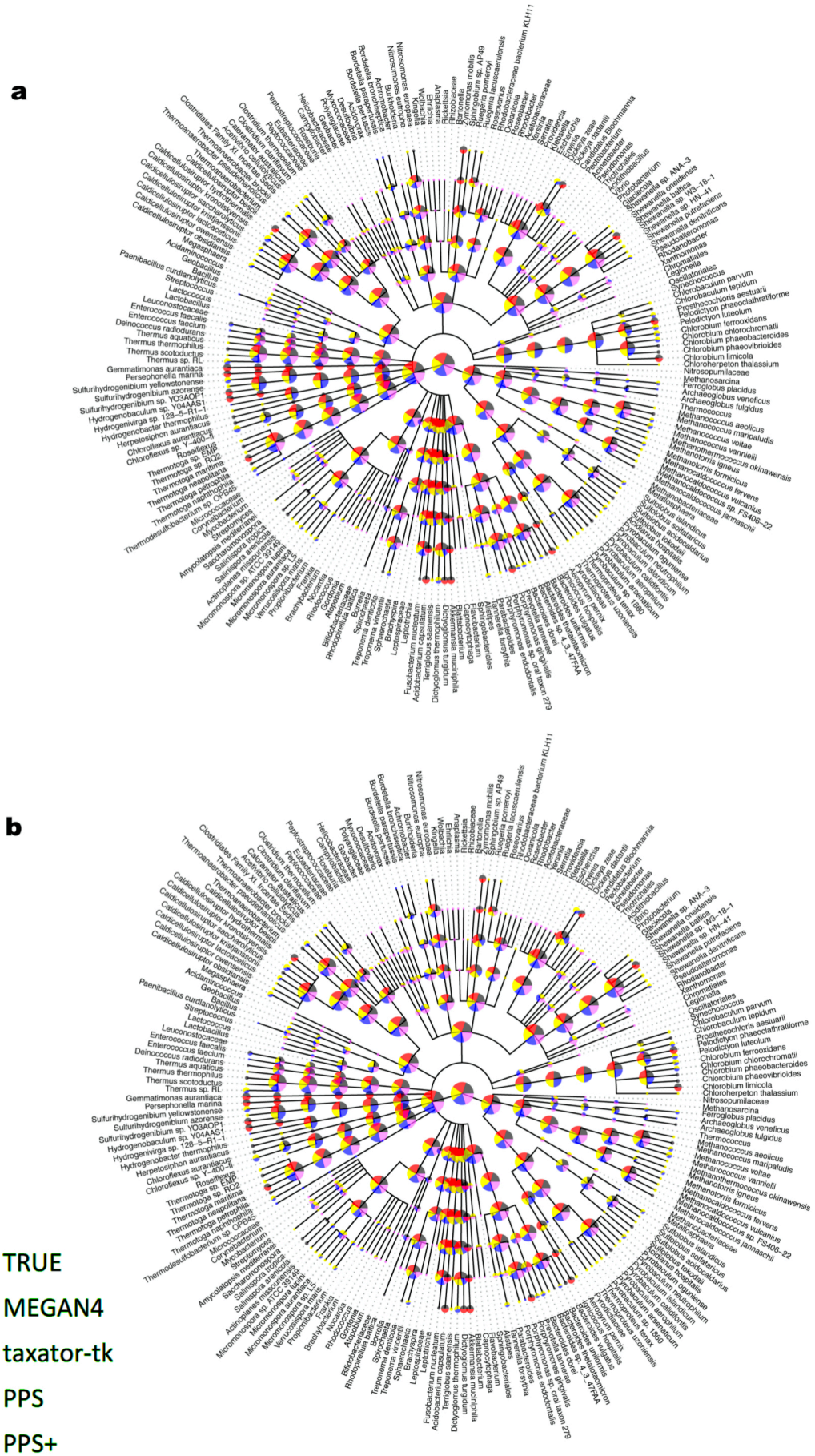
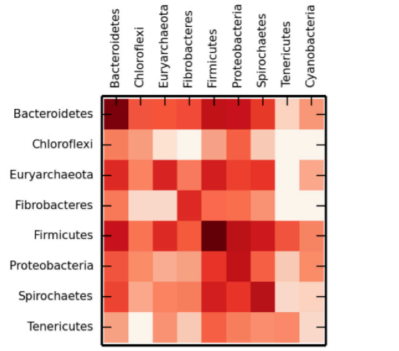

a
b
TRUE
MEGAN4
taxator-tk
PPS
PPS+

Figure S5. Base pairs assigned to individual taxa for a simulated metagenome of a microbial community with log-normally distributed species abundance.

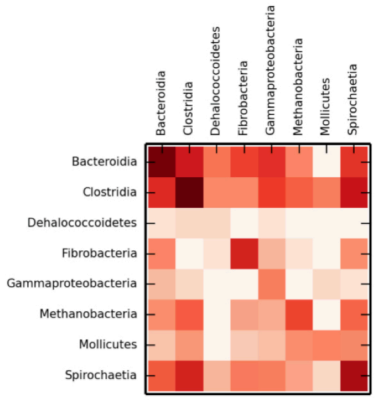
The number of taxonomic assignments to each taxon in bp is indicated on a log-scale by the pie chart sizes for *PPS*+, the generic *PPS* model, *taxator-tk*, *MEGAN4* and the underlying standard of truth (TRUE). There were 47 strains present in the simulated metagenome sample. Assignments to taxa not shown in black in the chart are to false taxa that are not present in the simulated metagenome. Panel (a) shows the scenario where sequences from the same species as those of the simulated dataset were excluded from the reference sequences but not the marker gene databases (Table 1: Test Scenario 3). Panel (b) shows the scenario where sequences from the same species as those of the simulated dataset were excluded from the reference sequence and marker gene databases (Table 1: Test Scenario 8).

a



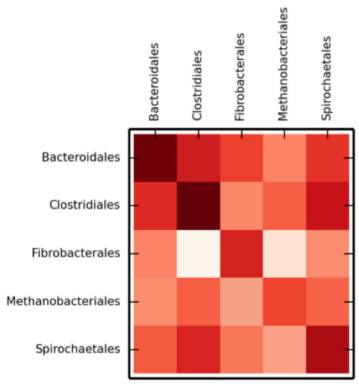
Consistency, PhyloPythiaS plus, Cow Rumen, phylum, agree: 153,774kb 73.9%

b



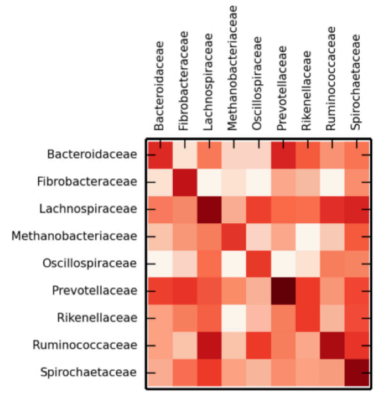
Consistency, PhyloPythiaS plus, Cow Rumen, class, agree: 99,596kb 86.0%

c



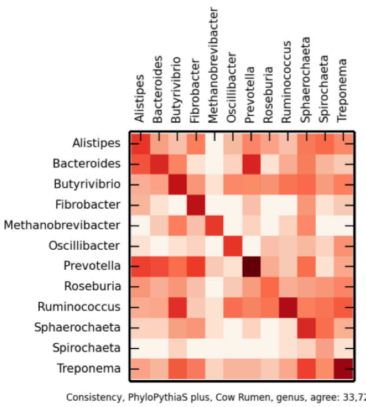
Consistency, PhyloPythiaS plus, Cow Rumen, order, agree: 98,616kb 88.4%

d



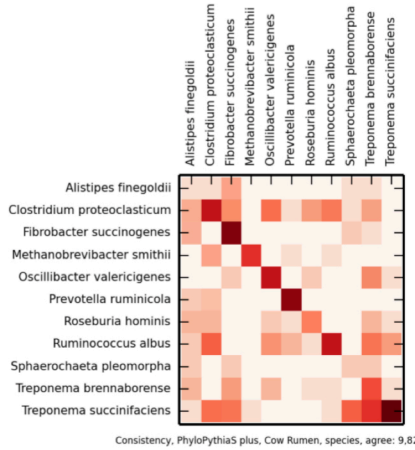
Consistency, PhyloPythiaS plus, Cow Rumen, family, agree: 46,343kb 80.0%

e



Consistency, PhyloPythiaS plus, Cow Rumen, genus, agree: 33,724kb 84.3%

f

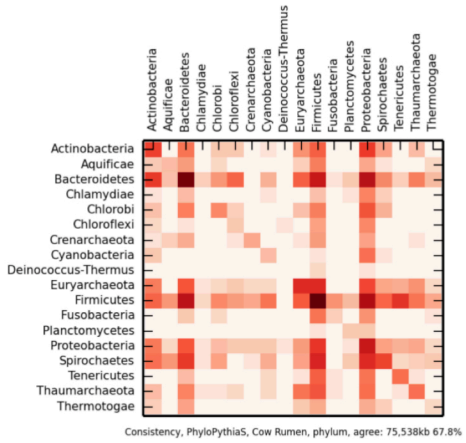


Consistency, PhyloPythiaS plus, Cow Rumen, species, agree: 9,821kb 91.6%

Figure S6. Comparison of scaffold and contig assignments using *PPS+* for the chunked cow rumen dataset.

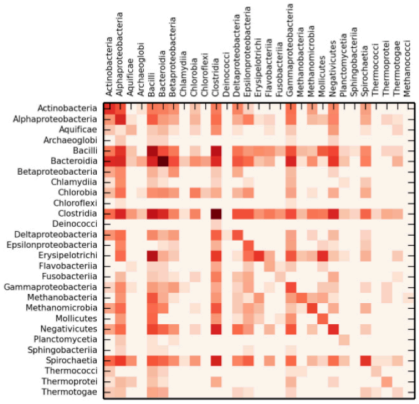
The comparisons were performed at different taxonomic ranks using heat maps (Supplemental Text S1, Sections 3.2.2 and 3.10.1). The rows correspond to scaffolds and the columns correspond to contig assignments. (panel a) Phylum; (panel b) class; (panel c) order; (panel d) family; (panel e) genus; (panel f) species.

a



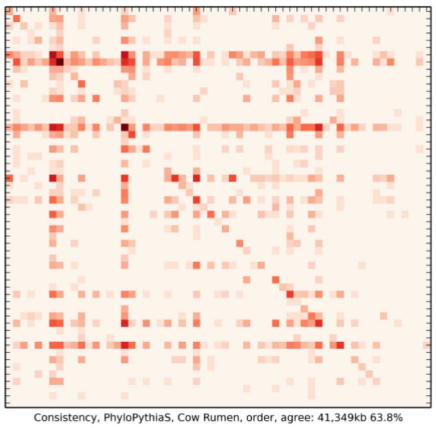
Consistency, PhyloPythiaS, Cow Rumen, phylum, agree: 75,538kb 67.8%

b



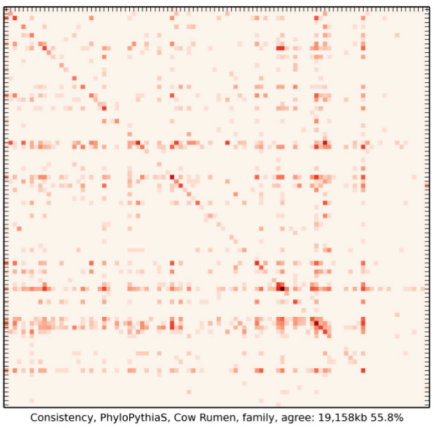
Consistency, PhyloPythiaS, Cow Rumen, class, agree: 43,931kb 58.5%

c



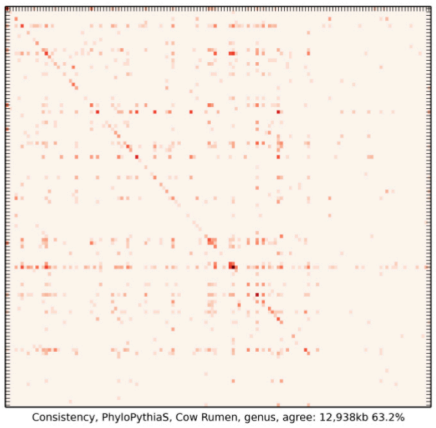
Consistency, PhyloPythiaS, Cow Rumen, order, agree: 41,349kb 63.8%

d



Consistency, PhyloPythiaS, Cow Rumen, family, agree: 19,158kb 55.8%

e

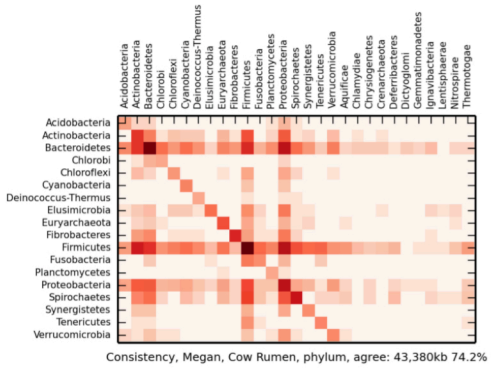


Consistency, PhyloPythiaS, Cow Rumen, genus, agree: 12,938kb 63.2%

Figure S7. Comparison of scaffold and contig assignments using the generic *PPS* model for the chunked cow rumen dataset.

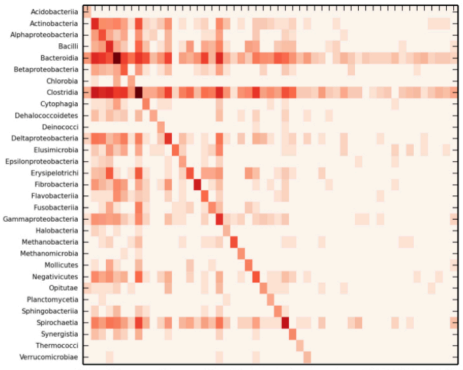
The comparisons were performed at different taxonomic ranks using heat maps (Supplemental Text S1, Sections 3.2.2 and 3.10.1). The rows correspond to scaffolds and the columns correspond to contig assignments. (panel a) Phylum; (panel b) class; (panel c) order; (panel d) family; (panel e) genus.

a



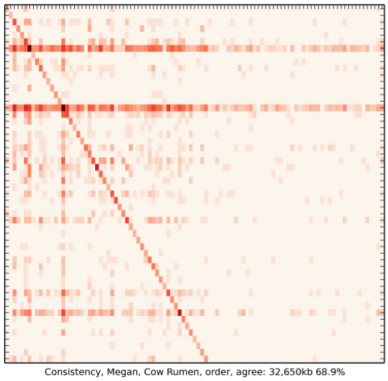
Consistency, Megan, Cow Rumen, phylum, agree: 43,380kb 74.2%

b



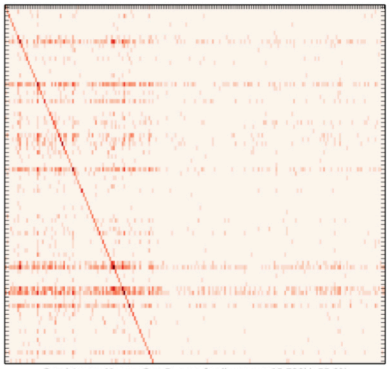
Consistency, Megan, Cow Rumen, class, agree: 33,780kb 68.5%

c



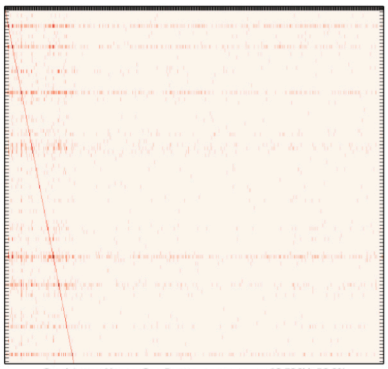
Consistency, Megan, Cow Rumen, order, agree: 32,650kb 68.9%

d



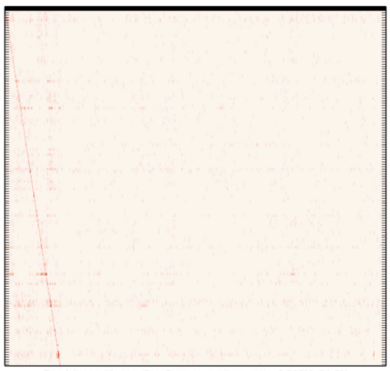
Consistency, Megan, Cow Rumen, family, agree: 15,790kb 55.0%

e



Consistency, Megan, Cow Rumen, genus, agree: 13,726kb 56.0%

f

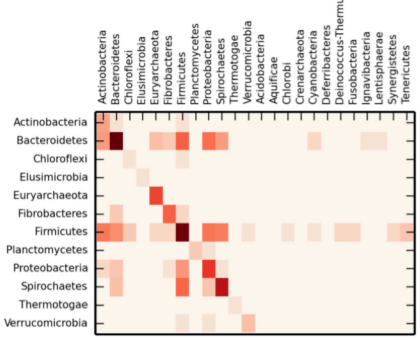


Consistency, Megan, Cow Rumen, species, agree: 8,502kb 54.6%

Figure S8. Comparison of scaffold and contig assignments using *MEGAN4* for the chunked cow rumen dataset.

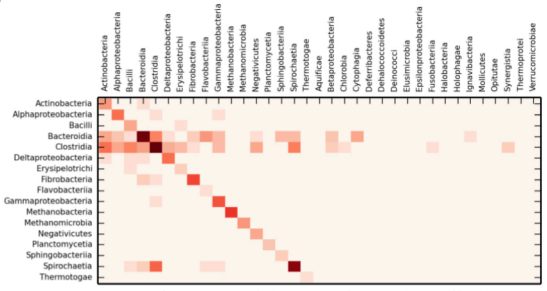
The comparisons were performed at different taxonomic ranks using heat maps (Supplemental Text S1, Section 3.2.2 and 3.10.1). The rows correspond to scaffolds and the columns correspond to contig assignments. (panel a) Phylum; (panel b) class; (panel c) order; (panel d) family; (panel e) genus; (panel f) species.

a



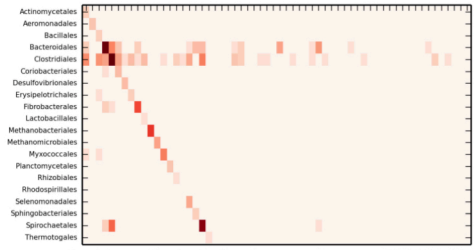
Consistency, Taxator-tk, Cow Rumen, phylum, agree: 59,702kb 98.2%

b



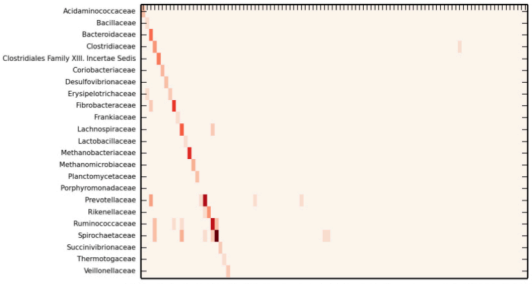
Consistency, Taxator-tk, Cow Rumen, class, agree: 23,190kb 97.7%

c



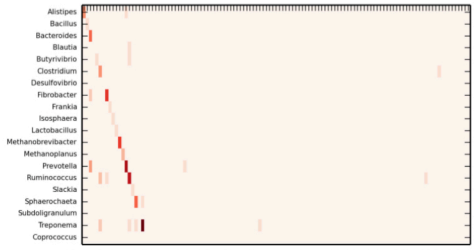
Consistency, Taxator-tk, Cow Rumen, order, agree: 22,368kb 98.0%

d



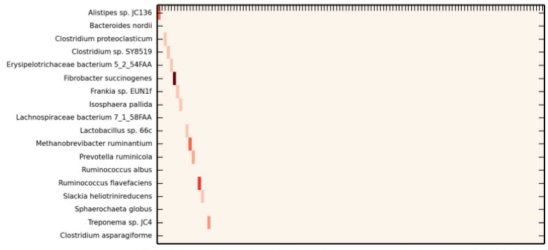
Consistency, Taxator-tk, Cow Rumen, family, agree: 7,276kb 98.9%

e



Consistency, Taxator-tk, Cow Rumen, genus, agree: 6,042kb 99.1%

f

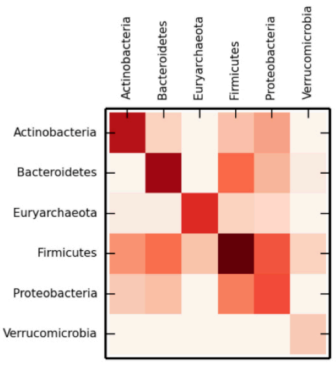


Consistency, Taxator-tk, Cow Rumen, species, agree: 292kb 100.0%

Figure S9. Comparison of scaffold and contig assignments using *taxator-tk* for the chunked cow rumen dataset.

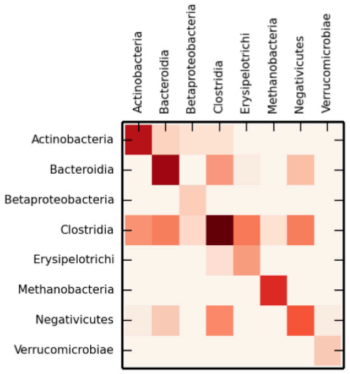
The comparisons were performed at different taxonomic ranks using heat maps (Supplemental Text S1, Section 3.2.2 and 3.10.1). The rows correspond to scaffolds and the columns correspond to contig assignments. (panel a) Phylum; (panel b) class; (panel c) order; (panel d) family; (panel e) genus; (panel f) species.

a



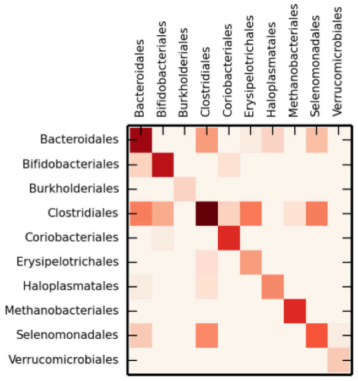
Consistency, PhyloPythiaS plus, Human Gut, phylum, agree: 140,283kb 99.0%

b



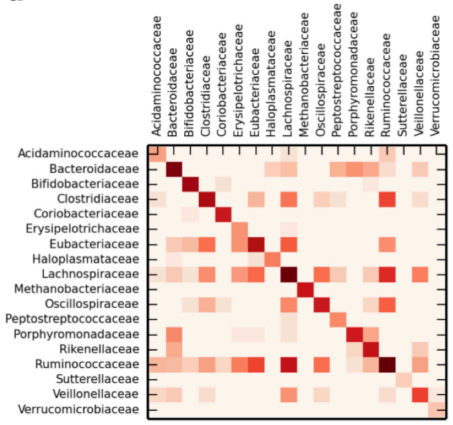
Consistency, PhyloPythiaS plus, Human Gut, class, agree: 134,707kb 99.5%

c



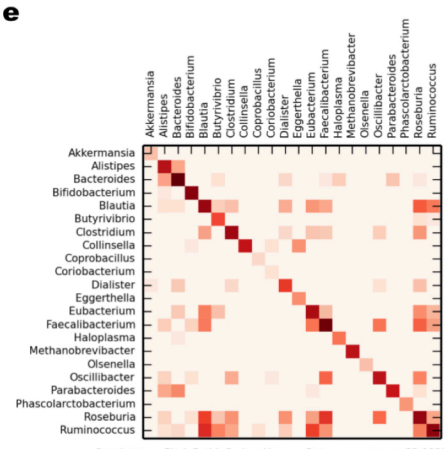
Consistency, PhyloPythiaS plus, Human Gut, order, agree: 134,127kb 99.5%

d



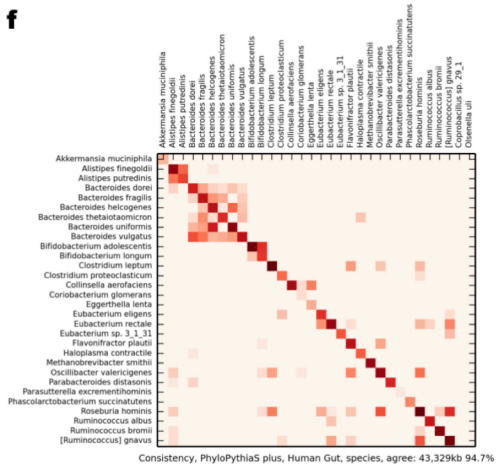
Consistency, PhyloPythiaS plus, Human Gut, family, agree: 110,664kb 94.0%

e



Consistency, PhyloPythiaS plus, Human Gut, genus, agree: 82,992kb 95.3%

f

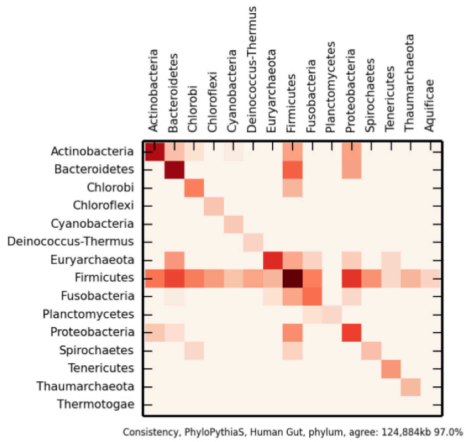


Consistency, PhyloPythiaS plus, Human Gut, species, agree: 43,329kb 94.7%

Figure S10. Comparison of scaffold and contig assignments using *PPS+* for the human gut dataset.

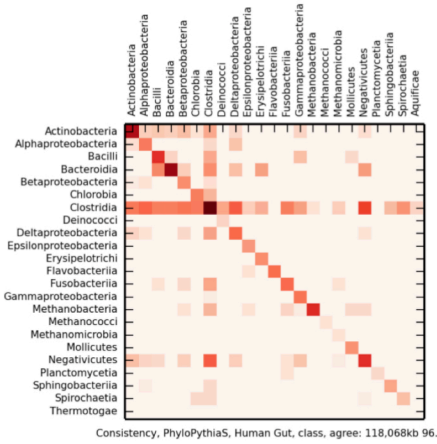
The comparisons were performed at different taxonomic ranks using heat maps (Supplemental Text S1, Sections 3.2.1 and 3.10.1). The rows correspond to scaffolds and the columns correspond to contig assignments. (panel a) Phylum; (panel b) class; (panel c) order; (panel d) family; (panel e) genus; (panel f) species.

a



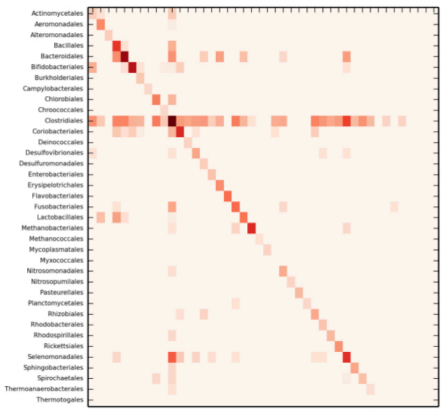
Consistency, PhyloPythiaS, Human Gut, phylum, agree: 124,884kb 97.0%

b



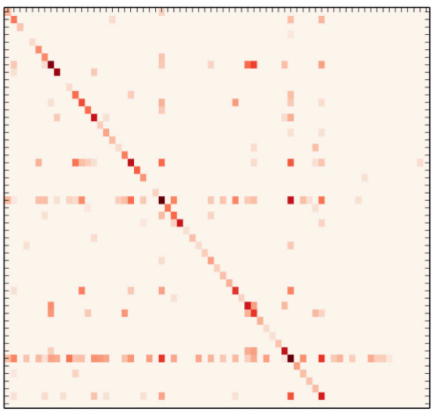
Consistency, PhyloPythiaS, Human Gut, class, agree: 118,068kb 96.9%

c



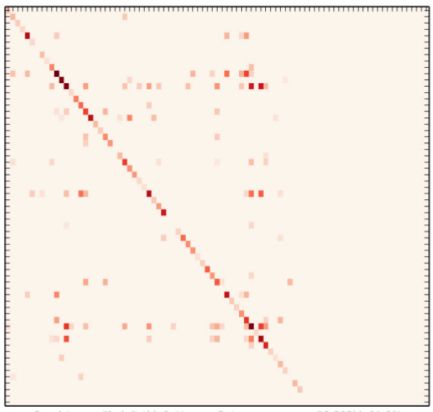
Consistency, PhyloPythiaS, Human Gut, order, agree: 117,185kb 97.3%

d



Consistency, PhyloPythiaS, Human Gut, family, agree: 97,066kb 92.6%

e

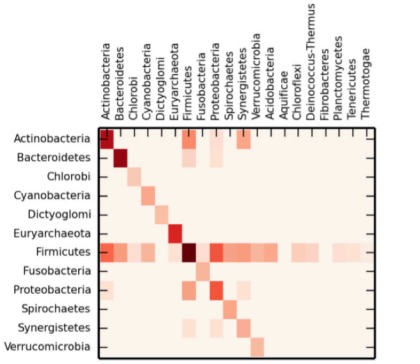


Consistency, PhyloPythiaS, Human Gut, genus, agree: 58,883kb 91.9%

Figure S11. Comparison of scaffold and contig assignments using the generic *PPS* model for the human gut dataset.

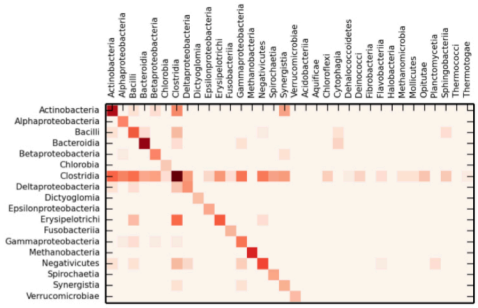
The comparisons were performed at different taxonomic ranks using heat maps (Supplemental Text S1, Sections 3.2.1 and 3.10.1). The rows correspond to scaffolds and the columns correspond to contig assignments. (panel a) Phylum; (panel b) class; (panel c) order; (panel d) family; (panel e) genus.

a



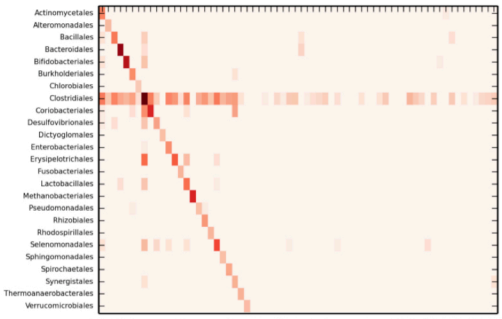
Consistency, Megan, Human Gut, phylum, agree: 127,658kb 99.0%

b



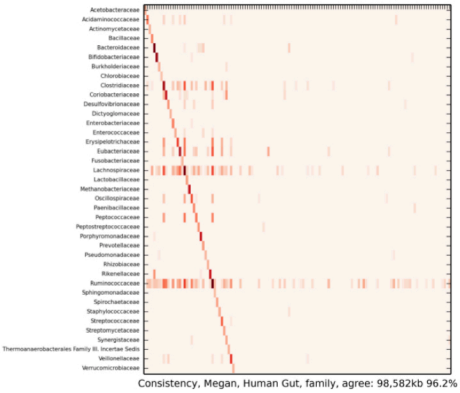
Consistency, Megan, Human Gut, class, agree: 122,131kb 98.5%

c



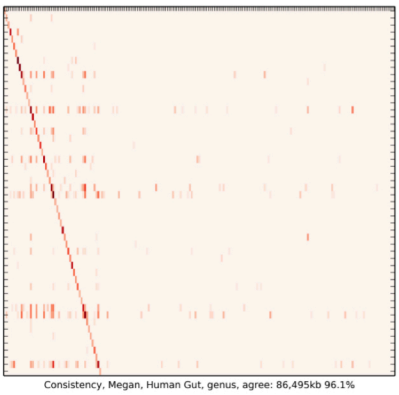
Consistency, Megan, Human Gut, order, agree: 121,811kb 98.6%

d



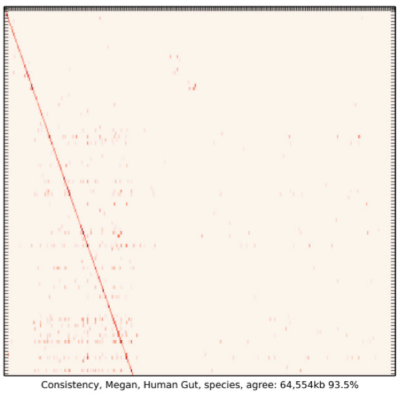
Consistency, Megan, Human Gut, family, agree: 98,582kb 96.2%

e



Consistency, Megan, Human Gut, genus, agree: 86,495kb 96.1%

f

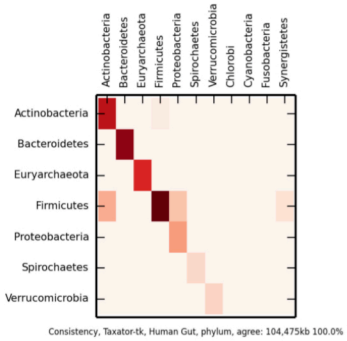


Consistency, Megan, Human Gut, species, agree: 64,554kb 93.5%

Figure S12. Comparison of scaffold and contig assignments using *MEGAN4* for the human gut dataset.

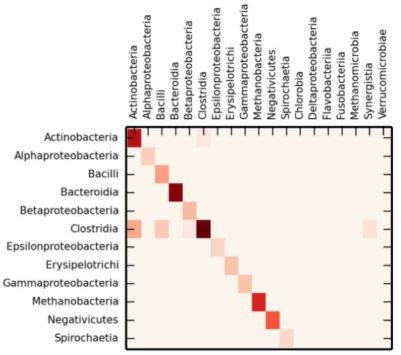
The comparisons were performed at different taxonomic ranks using heat maps (Supplemental Text S1, Sections 3.2.1 and 3.10.1). The rows correspond to scaffolds and the columns correspond to contig assignments. (panel a) Phylum; (panel b) class; (panel c) order; (panel d) family; (panel e) genus; (panel f) species.

a



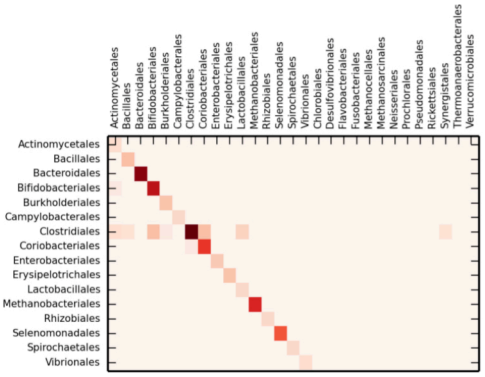
Consistency, Taxator-tk, Human Gut, phylum, agree: 104,475kb 100.0%

b



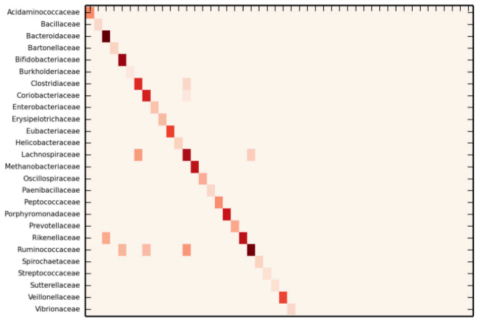
Consistency, Taxator-tk, Human Gut, class, agree: 84,228kb 100.0%

c



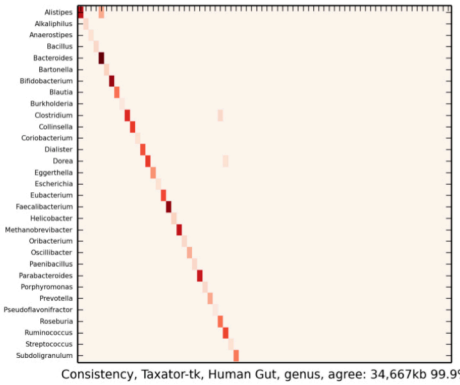
Consistency, Taxator-tk, Human Gut, order, agree: 83,337kb 100.0%

d



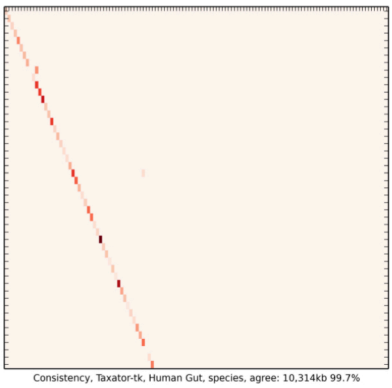
Consistency, Taxator-tk, Human Gut, family, agree: 43,751kb 99.8%

e



Consistency, Taxator-tk, Human Gut, genus, agree: 34,667kb 99.9%

f



Consistency, Taxator-tk, Human Gut, species, agree: 10,314kb 99.7%

Figure S13. Comparison of scaffold and contig assignments using *taxator-tk* for the human gut dataset.

The comparisons were performed at different taxonomic ranks using heat maps (Supplemental Text S1, Sections 3.2.1 and 3.10.1). The rows correspond to scaffolds and the columns correspond to contig assignments. (panel a) Phylum; (panel b) class; (panel c) order; (panel d) family; (panel e) genus; (panel f) species.

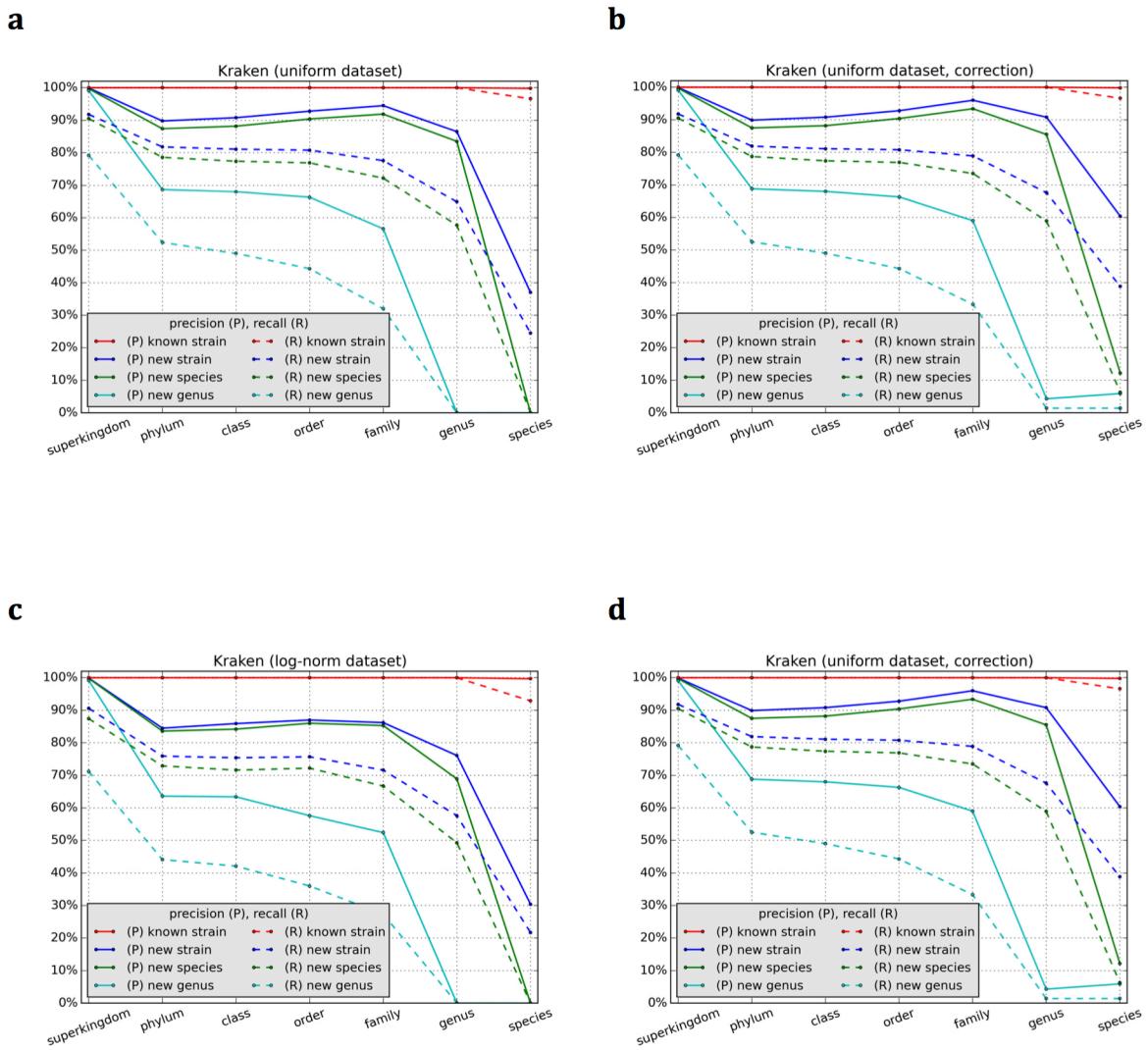


Figure S14. Benchmark results for the simulated datasets with the *Kraken* software.

Precision (P) and recall (R) (Supplemental Text S1, Section 3.9) at different taxonomic ranks were calculated for the *Kraken* software in four test scenarios (Table 1: Test Scenarios 1, 5, 8, 9, Supplemental Text S1, Section 3.1) using the simulated datasets with the uniform (panel *a* and *b*) and log-norm (panel *c* and *d*) distribution.

Supplemental Tables S1 – S7

PhyloPythiaS+: A Self-Training Method for the Rapid Reconstruction of Low-Ranking Taxonomic Bins from Metagenomes

Table S1. Runtime comparison of the *k*-mer counting algorithms *Jellyfish*, *KAnalyze*, and the new *k*-mer counting algorithm implemented in *PPS+*.

k-mer lengths	*PPS+*	*Jellyfish 1.1.1*	*Jellyfish 2.2*	*KAnalyze 0.9.7*
4	7.5 s	6.2 s	21.7 s	29.5 s
5	7.5 s	6.2 s	22.0 s	34.7 s
6	7.5 s	6.2 s	21.9 s	39.1 s
4, 5, 6	**9.0 s**	**18.6 s**	**1m 5 s**	**1m 43 s**
7	7.6 s	6.2 s	22.4 s	43.9 s
8	8.0 s	6.2 s	23.0 s	48.9 s
9	8.4 s	6.3 s	24.6 s	54.4 s
7, 8, 9	11.1 s	18.7 s	1 m 10 s	2 m 27 s

The relevant combination of *k*-mers that is typically counted for taxonomic binning is marked in bold. The benchmark was run in one thread on a server with an Intel Xeon (CPU X5660, 2.8 GHz) processor, nevertheless we observed that *Jellyfish 1.1.1* took approximately 30% more CPU resources than specified. Parallel runs of the methods can be done by splitting the input FASTA file, running multiple instances of a tool for each file separately in parallel and merging of the result files, thus the runtimes scale approximately linearly with the number of CPUs used. As a benchmark dataset, concatenated contigs from (Turnbaugh et al., 2010) (255 Mb) were used.

Table S2. Exact values corresponding to (Fig. 2A).

Method	Rank	F_1-score (%)	Precision (%)	Recall = Correct (%)	Incorrect (%)	Unassigned (%)
taxator-tk	Family	66.6	98.2	50.4	0.9	48.7
PPS	Family	60.4	72.6	51.7	19.5	28.8
MEGAN	Family	78.8	88.9	70.7	8.8	20.4
Kraken	Family	74.7	79.6	70.4	18.0	11.5
PPS+	Family	88.4	96.4	81.6	3.0	15.4
taxator-tk	Genus	46.1	93.2	30.6	2.2	67.2
PPS	Genus	45.8	68.2	34.5	16.1	49.4
MEGAN	Genus	63.1	75.7	54.1	17.4	28.5
Kraken	Genus	59.3	63.4	55.7	32.1	12.2
PPS+	Genus	77.4	91.8	66.9	6.0	27.1
taxator-tk	Species	16.7	87.8	9.2	1.3	89.6
PPS	Species	N/A	N/A	N/A	N/A	100.0
MEGAN	Species	34.2	49.6	26.1	26.5	47.4
Kraken	Species	32.8	35.7	30.3	54.6	15.2
PPS+	Species	51.5	71.4	40.3	16.1	43.6

Table S3. Exact values corresponding to (Fig. 2B).

Method	Rank	F_1-score (%)	Precision (%)	Recall = Correct (%)	Incorrect (%)	Unassigned (%)
taxator-tk	Family	67.4	99.4	51.0	0.3	48.7
PPS	Family	70.0	84.2	59.9	11.3	28.8
MEGAN	Family	79.3	89.5	71.2	8.3	20.4
Kraken	Family	75.8	80.7	71.4	17.1	11.5
PPS+	Family	90.1	98.3	83.2	1.5	15.4
taxator-tk	Genus	48.8	98.9	32.4	0.4	67.2
PPS	Genus	55.5	82.7	41.8	8.8	49.4
MEGAN	Genus	68.0	81.5	58.3	13.2	28.5
Kraken	Genus	60.7	64.9	57.0	30.8	12.2
PPS+	Genus	83.2	98.6	71.9	1.0	27.1
taxator-tk	Species	18.5	98.2	10.2	0.2	89.6
PPS	Species	N/A	N/A	N/A	N/A	100.0
MEGAN	Species	50.0	72.5	38.1	14.4	47.4
Kraken	Species	38.7	42.2	35.8	49.1	15.2
PPS+	Species	68.9	95.5	53.9	2.5	43.6

Table S4. Exact values corresponding to (Fig. 2C).

Method	Rank	F_1-score (%)	Precision (%)	Recall = Correct (%)	Incorrect (%)	Unassigned (%)
taxator-tk	Family	64.2	98.5	47.6	0.7	51.7
PPS	Family	49.2	63.5	40.1	23.0	36.9
MEGAN	Family	76.3	90.7	65.8	6.8	27.4
Kraken	Family	71.8	78.1	66.4	18.6	15.0
PPS+	Family	85.0	95.7	76.5	3.4	20.0
taxator-tk	Genus	43.7	92.3	28.6	2.4	69.0
PPS	Genus	35.2	56.0	25.7	20.2	54.1
MEGAN	Genus	61.9	78.6	51.1	13.9	35.0
Kraken	Genus	56.0	61.1	51.7	33.0	15.3
PPS+	Genus	72.9	90.1	61.2	6.7	32.1
taxator-tk	Species	17.8	94.1	9.8	0.6	89.6
PPS	Species	N/A	N/A	N/A	N/A	100.0
MEGAN	Species	34.6	52.3	25.9	23.6	50.5
Kraken	Species	31.6	35.4	28.6	52.4	19.0
PPS+	Species	48.9	73.1	36.7	13.5	49.8

Table S5. Exact values corresponding to (Fig. 2D).

Method	Rank	F_1-score (%)	Precision (%)	Recall = Correct (%)	Incorrect (%)	Unassigned (%)
taxator-tk	Family	64.8	99.4	48.1	0.3	51.7
PPS	Family	66.1	85.4	53.9	9.2	36.9
MEGAN	Family	77.3	92.0	66.7	5.8	27.4
Kraken	Family	72.5	78.9	67.1	17.9	15.0
PPS+	Family	87.5	98.5	78.7	1.2	20.0
taxator-tk	Genus	47.0	99.3	30.8	0.2	69.0
PPS	Genus	51.2	81.4	37.3	8.5	54.1
MEGAN	Genus	68.5	86.9	56.5	8.5	35.0
Kraken	Genus	56.8	61.9	52.4	32.3	15.3
PPS+	Genus	78.7	97.3	66.1	1.9	32.1
taxator-tk	Species	18.5	97.3	10.2	0.3	89.6
PPS	Species	N/A	N/A	N/A	N/A	100.0
MEGAN	Species	52.0	78.6	38.8	10.6	50.5
Kraken	Species	39.3	43.9	35.6	45.5	19.0
PPS+	Species	62.2	93.1	46.7	3.5	49.8

Table S6. Scaffold-contig consistency of the chunked cow rumen dataset.

Measure	*PPS+*	*PPS*	*MEGAN4*	*Kraken*	*taxator-tk*	Def.
Scaffolds considered	12,192	12,192	9456	7859	11,447	
Consistent contigs	128,685	137,747	116,726	104,633	151,585	
/	/	/	/	/	/	1
total contigs	159,263	159,263	135,362	119,939	153,185	
Consistent count %	*80.80*	86.49	86.23	87.24	**98.96**	1
Consistent kbp	257,370	275,494	233,452	209,266	303,170	
/	/	/	/	/	/	2
total kbp	318,526	318,526	270,724	239,878	306,370	
Consistent bp %	*80.80*	86.49	86.23	87.24	**98.96**	2
Avg. distance to path	0.38	0.30	0.50	*0.60*	**0.02**	3
Avg. weighted distance to path	0.38	0.30	0.50	*0.60*	**0.02**	4
Avg. distance to scaffold label	3.16	3.43	5.89	*7.23*	**2.65**	5
Avg. weighted distance to scaffold label	3.16	3.43	5.89	*7.23*	**2.65**	6
Family: contigs (kb assigned)	**71,660**	43,118	55,904	45,752	*13,626*	
Family: consistency '% agreement'	80.0	55.8	55.0	*45.2*	**98.9**	0b
Genus: contigs (kb assigned)	**53,705**	28,077	53,008	44,600	*10,596*	
Genus: consistency '% agreement'	84.3	63.2	56.0	*43.7*	**99.1**	0b
Species: contigs (kb assigned)	26,121	N/A	41,204	**42,626**	*1426*	
Species: consistency '% agreement'	91.6	N/A	54.6	*38.1*	**100.0**	0b

Contigs of the cow rumen dataset of at least 10 kb were divided into chunks of 2 kb for evaluation of assignment consistency (Supplemental Text S1, Section 3.2.2). Scaffold-contig consistency of the assignments made by *PPS+*, the generic *PPS* model, *MEGAN4*, *Kraken* and *taxator-tk* for the chunked cow rumen dataset, computed via different definitions (Supplemental Text S1, Section 3.10.2). The table also contains the number of kb of contigs assigned at low taxonomic ranks (family, genus and species) and the corresponding consistency (% agreement) (Supplemental Text S1, Section 3.10.1). Bold numbers correspond to the best values, whereas italic numbers indicate the worst values.

Table S7. Scaffold-contig consistency of the human gut metagenome dataset.

Measure	*PPS+*	*PPS*	*MEGAN4*	*Kraken*	*taxator-tk*	Def.
Scaffolds considered	47,983	47,983	83,973	75,926	99,202	
Consistent contigs	64,197	63,954	99,647	88,214	117,576	
/	/	/	/	/	/	
total contigs	66,480	66,480	101,613	92,900	117,630	1
Consistent count %	96.57	*96.20*	98.07	94.96	**99.95**	1
Consistent kbp	181,207	179,798	191,429	166,075	217,517	
/	/	/	/	/	/	
total kbp	189,517	189,517	200,478	190,001	217,720	2
Consistent bp %	95.62	94.87	95.49	*87.41*	**99.91**	2
Avg. distance to path	0.06	0.07	0.05	*0.14*	**0**	3
Avg. weighted distance to path	0.07	0.10	0.12	*0.35*	**0**	4
Avg. distance to scaffold label	0.63	*0.72*	0.38	0.61	**0.29**	5
Avg. weighted distance to scaffold label	**0.53**	0.58	0.73	*1.15*	0.62	6
Family: contigs (kb assigned)	146,046	118,679	161,452	**173,238**	*74,793*	
Family: consistency '% agreement'	94.0	92.6	96.2	*53.4*	**99.8**	0b
Genus: contigs (kb assigned)	110,762	71,934	149,448	**159,556**	*61,242*	
Genus: consistency '% agreement'	95.3	91.9	96.1	*88.3*	**99.9**	0b
Species: contigs (kb assigned)	61,969	N/A	114,716	**162,726**	*20,687*	
Species: consistency '% agreement'	94.7	N/A	*93.5*	81.3	**99.7**	0b

Scaffold-contig consistency of the assignments made by *PPS+*, the generic *PPS* model, *MEGAN4*, *Kraken* and *taxator-tk* of the human gut dataset (Supplemental Text S1, Section 3.2.1) computed using different definitions (Supplemental Text S1, Section 3.10). Bold numbers correspond to the best values, whereas italic numbers indicate the worst values.