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

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

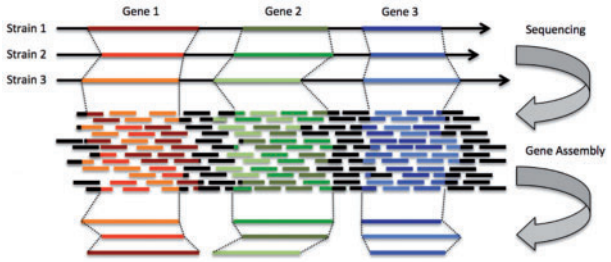


Fig. 1. An example of the gene assembly problem. In this example, the sequenced microbial community consists only of three distinct strains. Non-coding regions of the strain sequences are black, whereas coding regions are red, green and blue for genes 1, 2 and 3. Genes 1–3 are present in all three strains, 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

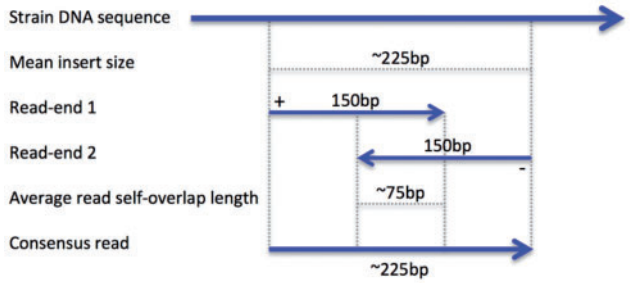


Fig. 2. An example of a self-overlapping paired-end read. Illumina HiSeq 2500 paired-end read consists of two 150bp read ends, one on the positive strand (+) and one on the negative strand (−). In our example, the mean insert size (225 bp) is smaller than two times the read end length (2 × 150 bp), therefore 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

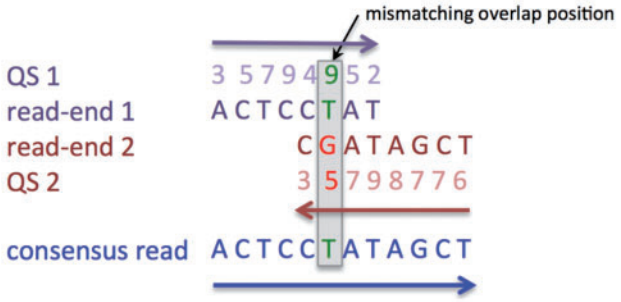


Fig. 3. Joining of self-overlapping reads example. The figure depicts a simplified example of a consensus read reconstruction. At the mismatching overlap position, read-end 1 has *T* with quality score (QS) 9, while read-end 2 has *G* with quality score 5. The resulting consensus read will have *T* at 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

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.

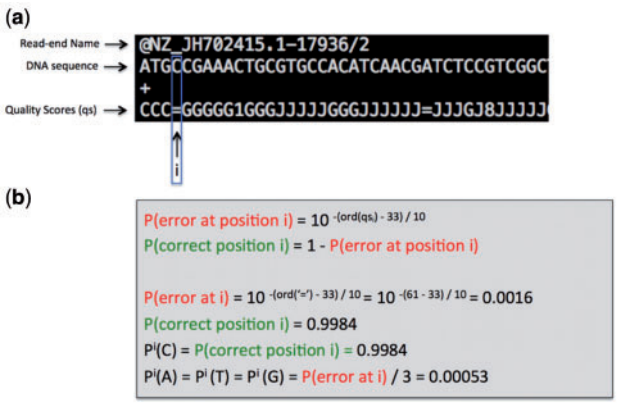


Fig. 5. FASTQ file data representation. (Panel a) depicts an example of a read end representation in a FASTQ file. The entry consists of the read end name, the DNA sequence of the respective end of a paired-end read and the quality score for each position of the DNA sequence, which are ASCII coded. (Panel 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

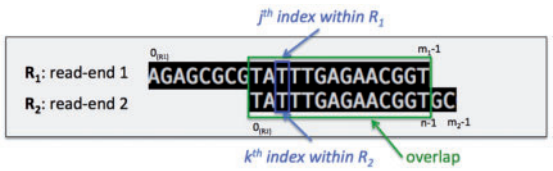
In our model, a probability matrix represents a consensus sequence, where each sequence position is represented by a probability distribution over DNA bases $\{A, C, T, G\}$. An example of a probability matrix corresponding to a consensus sequence of two overlapping sequences is depicted in (Fig. 6). 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

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

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:

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

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:

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

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 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.

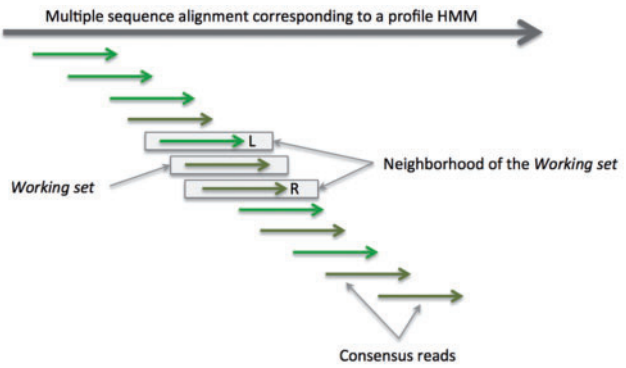


Fig. 7. Initial layout of consensus reads. Consensus reads sorted according to the HMM start coordinates. In the neighbourhood of the consensus read, that is in the *working set,* there are two closest consensus reads, one on the left (*L*) and one on the right (*R*)

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:

   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

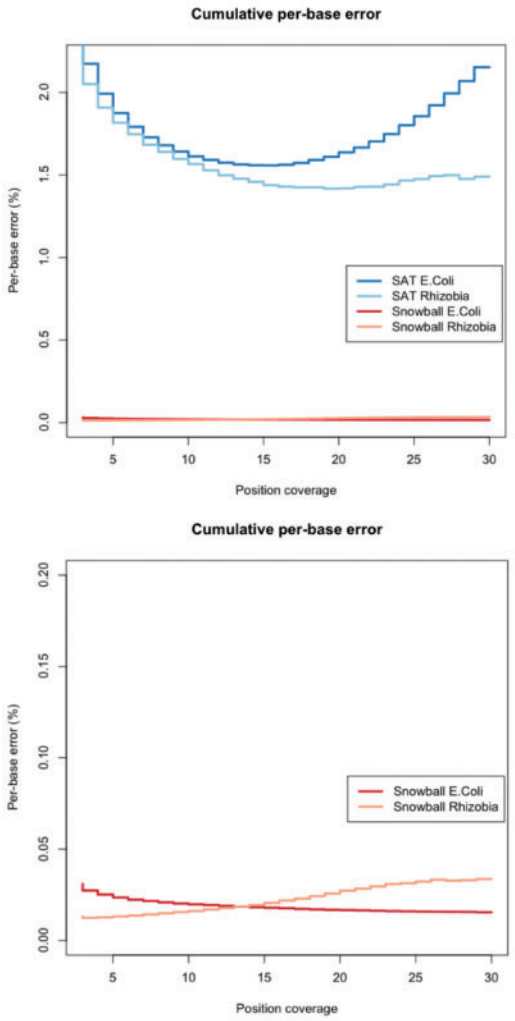


Fig. 8. Cumulative per-base error. Cumulative per-base error for the *Snowball* and *SAT* assemblers. We computed the per-base error in a cumulative way, i.e. for $X \in [3,\ldots,30]$ (on the horizontal *x*-axes), Y (on the vertical *y*-axes) is equal to the per-base error at contig positions with coverage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

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

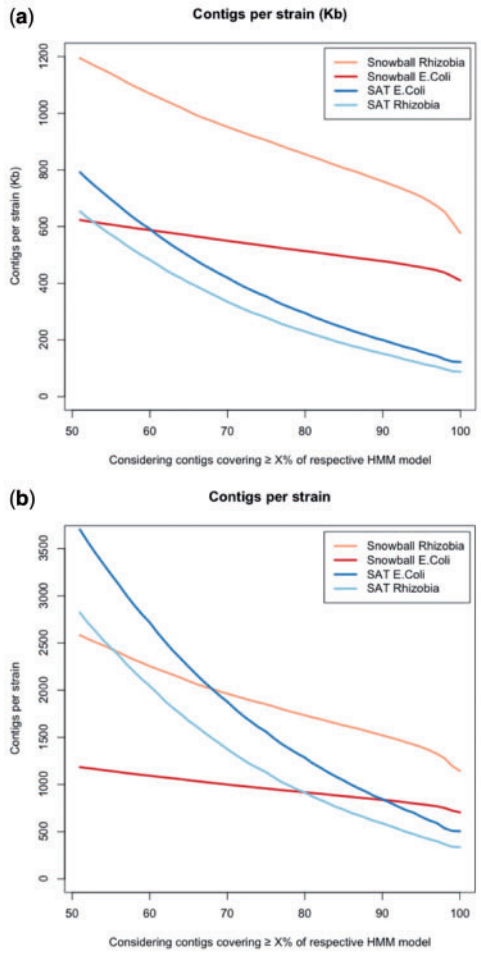


Fig. 9. Contigs per strain. Cumulative average contig length per strain, considering only contigs covering $X\%$ of respective profile HMMs (panel *a*). Average number of contigs per strain, considering only contigs covering $\geq X\%$ of respective profile HMMs (panel *b*). Here, the variable X corresponds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

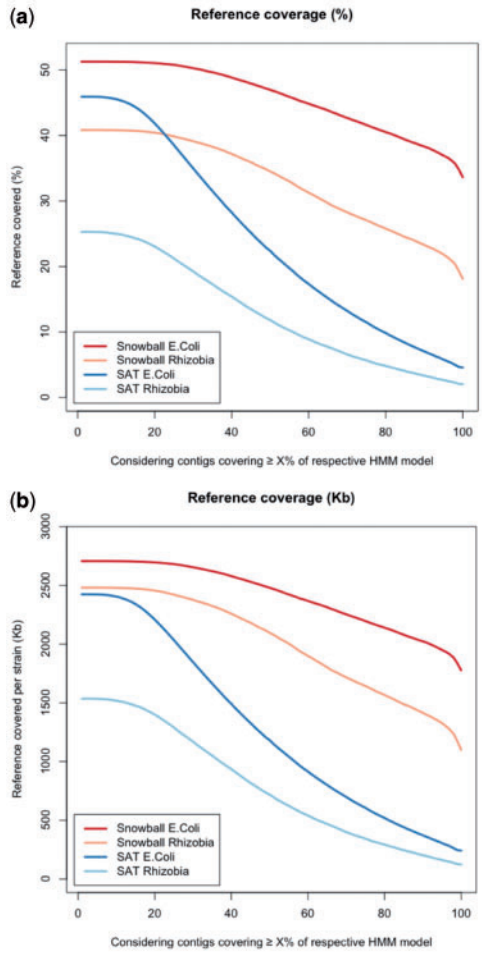


Fig. 10. Coverage of the reference strain sequences. Percentage of the recovered reference strains, per strain on average, considering only contigs covering $\geq X\%$ of respective profile HMMs (panel *a*). Corresponding absolute values (Kb) are depicted in (panel *b*). The variable X corresponds to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

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).

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.