BIOINFORMATICS ORIGINAL PAPER

Vol. 29 no. 17 2013, pages 2103–2111
doi:10.1093/bioinformatics/btt357

Sequence analysis

Advance Access publication June 19, 2013

A Sensitive and Accurate protein domain cLassification Tool (SALT) for short reads

Yuan Zhang[1], Yanni Sun[1,*] and James R. Cole[2]

[1]Department of Computer Science and Engineering and [2]Center for Microbial Ecology, Michigan State University, East Lansing, MI 48824, USA

Associate Editor: Janet Kelso

ABSTRACT

Motivation: Protein domain classification is an important step in functional annotation for next-generation sequencing data. For RNA-Seq data of non-model organisms that lack quality or complete reference genomes, existing protein domain analysis pipelines are applied to short reads directly or to contigs that are generated using *de novo* sequence assembly tools. However, these strategies do not provide satisfactory performance in classifying short reads into their native domain families.

Results: We introduce SALT, a protein domain classification tool based on profile hidden Markov models and graph algorithms. SALT carefully incorporates the characteristics of reads that are sequenced from the domain regions and assembles them into contigs based on a supervised graph construction algorithm. We applied SALT to two RNA-Seq datasets of different read lengths and quantified its performance using the available protein domain annotations and the reference genomes. Compared with existing strategies, SALT showed better sensitivity and accuracy. In the third experiment, we applied SALT to a non-model organism. The experimental results demonstrated that it identified more transcribed protein domain families than other tested classifiers.

Availability: The source code and supplementary data are available at https://sourceforge.net/projects/salt1/

Contact: yannisun@msu.edu

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

Received and revised on May 2, 2013; accepted on June 11, 2013

1 INTRODUCTION

Inferring functions from sequences remains important in analyzing different types of sequencing data including those generated by the next-generation sequencing (NGS) technologies. One basic step during the functional analysis is to assign sequences into different functional categories, such as families of protein domains (or domains for short), which are independent folding and functional units in a majority of annotated protein sequences.

Protein domain analysis has been widely used for functional annotations of RNA-Seq data (Li *et al.*, 2011; Mutasa-Göttgens *et al.*, 2012; Orshinsky *et al.*, 2012; Schmid *et al.*, 2012). In particular, quantifying the expression levels of protein domains helps us understand how transcriptional changes of domains are associated with sequencing conditions, sampling tissues or experimental treatments in RNA-Seq data. For example, computational domain analysis was applied to identify domains that play a role in vernalization and efflux transporters in the gibberellin response in sugar beets (Mutasa-Göttgens *et al.*, 2012). Domain analysis is also frequently used to evaluate and compare gene annotation quality of different gene-finding tools (Li *et al.*, 2011) or to compare domain composition of data sampled using different techniques (Schmid *et al.*, 2012).

The state-of-the-art method for protein domain analysis is still based on comparative sequence analysis, where query sequences are annotated via comparison with characterized sequence databases. Depending on the alignment algorithms and the target databases, domain analysis methods can be divided into two groups. The first one is to compare the sequences against publicly available reference protein sequence databases using pairwise alignment tools such as BLAST (Altschul *et al.*, 1990). The second method is a profile-based similarity search, which classifies queries into characterized protein domain or family databases such as Pfam (Finn *et al.*, 2010), TIGRFAM (Haft *et al.*, 2003), FIGfams (Meyer *et al.*, 2009) and so forth. There also exist comprehensive protein domain search tools such as InterProScan (Quevillon *et al.*, 2005), which combines different sequence and profile-based domain recognition methods from the InterPro (Hunter *et al.*, 2009) consortium member databases into one resource.

Compared with the first method, the profile-based method has two advantages. First, its running time mainly depends on the growth of the data to be analyzed because the number of families/domains grows slowly and is much smaller than the sequence databases such as NCBI-nr. Second, previous work (Durbin *et al.*, 1998) has shown that using position-specific conservation information can improve the sensitivity of a remote protein homology search, which is especially important for identifying new homologs in NGS data. One of the most widely used profile-based domain classification tool is HMMER (http://hmmer.janelia.org/), which relies on profile hidden Markov models (profile HMMs) to deliver sensitive domain classification for remote homologs.

Existing domain classification tools are mainly designed for complete or near-complete domain member sequences rather than fragmentary and short reads in NGS data. BLAST-based and profile-based domain classification tools rely on alignment scores to distinguish true homologous sequences from false ones. Short reads incur marginal alignment scores and thus can be easily missed by these tools. In particular, BLAST-based tools

*To whom correspondence should be addressed.

have low sensitivity for short reads of remote homologs (Wommack *et al.*, 2008). Although HMMER can achieve high sensitivity in recognizing remote homologs of a domain family, it shows low sensitivity in classifying partial sequences of remote homologs (Zhang and Sun, 2012).

When the reference genomes are available, genome-wide gene and domain annotations and read mapping positions can be combined to determine the membership of reads. However, many NGS datasets lack complete or quality reference genomes, such as complicated metagenomic datasets and transcriptomic data of some non-model organisms. For these data, protein domain analysis is applied to the short reads directly or to the contigs that are generated by *de novo* sequence assembly tools. Contigs produced by *de novo* sequence assembly tools enable NGS data analysis to take advantage of conventional bioinformatics tools designed for longer sequences. However, using sequence assembly tools as the first step is not ideal for protein domain classification. First, the quality of read assembly deteriorates significantly in complicated NGS datasets (Jeffrey and Zhong, 2011). Different sequence assembly tools generate different sets of contigs. There is no consensus on the best assembly tool. Second, successful *de novo* assembly requires high sequencing depth, which is difficult to achieve for all domain regions. It is often observed that some domain regions are highly transcribed, whereas others are transcribed much less. Third, different protein-coding genes can share the same domain. As a result, similar domain regions can contribute to the nodes forming crossing points between different paths in a De Bruijn graph. Those repeat-like sequences add difficulty to *de novo* sequence assembly. Thus, there is a need for an alternative and better domain classification tool for NGS data lacking reference genomes.

In this work, we designed a Sensitive and Accurate protein domain cLassification Tool (SALT), which uses profile HMMs and family-specific contig generation algorithms to classify short reads into domains. SALT is mainly designed for domain classification in transcriptomic data of non-model organisms that lack complete or high-quality reference genomes and for metagenomic datasets. In this work, we focus on RNA-Seq of non-model organisms. To evaluate the performance of SALT, we applied SALT to RNA-Seq data of species with quality reference genomes. Read mapping and genome-wide domain annotations are combined as the 'ground truth' for evaluating the read classification sensitivity and specificity. SALT was benchmarked with several popular domain classification strategies. The comparison shows that SALT can correctly classify significantly more reads into their native domains while keeping the same or better specificity. Finally, we demonstrated the utility of SALT on an RNA-Seq dataset of a non-model organism.

Although we will use protein domains when describing the methods and results, SALT can be applied to both domain families and protein families. In this work, we used the families in Pfam (Finn *et al.*, 2010), which include both domain families and protein families.

2 RELATED WORK

HMMER is the representative implementation of a profile HMM-based domain classification tool. In conjunction with the Pfam database (Finn *et al.*, 2010), which contains >10 000 annotated protein domain families, HMMER can accurately classify query protein sequences into existing domain families. However, HMMER can fail to recognize short reads sequenced from remote homologs of a domain family. Our previous work (Zhang and Sun, 2012) evaluated how read lengths affected the performance of HMMER on a large number of domains. Profile HMM-based tools have a sensitivity of 0.9 in classifying reads of 80 bp into domains with an average sequence identity >40%. However, for poorly conserved domains, which are not rare, a significant number of reads cannot be correctly classified. Even for a domain with good average conservation along each position, short reads that are sequenced from less well-conserved regions tend to be missed by existing methods.

HHblits (Remmert *et al.*, 2012) uses HMM–HMM alignment to conduct fast iterative protein sequence searches and can generate accurate alignment. However, it is not especially designed for NGS data and has unsatisfactory classification sensitivity for short and fragmentary reads, which we show in our experiments.

To take advantage of conventional functional analysis pipelines, *de novo* sequence assembly tools can be applied to produce contigs, which are used as input to gene finding or domain analysis tools. The performance of short read assemblers depends on read length, complexity of the data, sequencing depth and so forth. Recent reviews (Jeffrey and Zhong, 2011; Miller *et al.*, 2010) summarize several challenges of *de novo* assembly tools. Recently, several popular *de novo* assembly tools were applied to the transcriptomes of non-model species (e.g. *Radix balthica* and sugar beets) (Feldmeyer *et al.*, 2012; Mutasa-Göttgens *et al.*, 2012). The assemblies generated by the tested tools differed in total number of contigs, contig length and number and quality of gene hits, showing the need for better assembly methodology.

Some other work (Zhang and Sun, 2011) aimed to develop domain classification tools for reads containing frameshift errors. The insertion or deletion errors in homopolymer regions of pyrosequencing reads can cause frameshifts, compounding domain classification. This work focused on short reads that are usually produced by the Illumina platform, which tends to have substitution rather than insertion or deletion errors. SALT is able to deal with substitution errors during domain classification.

3 METHODS

3.1 Overview of SALT

As a protein domain classification tool, SALT takes query reads and a list of protein domain families of interest as input. The output is a list of transcribed families and the classified reads for each family. Figure 1 shows a piece of a genomic sequence with annotated domains and the sequenced reads. In this example, if a read is sequenced from domain X in a gene, we call it a positive read with regard to domain X. Otherwise, it is a negative read with regard to domain X. An ideal domain classification program should classify only the positive reads into the corresponding domain family.

Positive reads have the following features: (i) many positive reads tend to share higher sequence similarity with the underlying family than negative reads, yielding higher alignment scores; (ii) positive reads can be assembled into contigs that have statistically significant alignment scores against their native families. Existing domain classification tools,

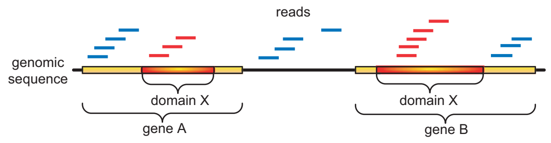

Fig. 1. Two genes, their domain organizations and the sequenced reads. Domain X occurs in two different genes. Both genes are transcribed and sequenced. Red lines: positive reads. Blue lines: negative reads

such as HMMER, mainly use the first feature. However, directly applying HMMER to short reads tends to yield unsatisfactory sensitivity because short positive reads have marginal alignment scores. *De novo* assembly tools can partially solve this problem, but their performance depends a lot on sequencing depth, similarities of domain copies in different genes and so forth.

SALT takes advantage of both features by incorporating profile-based domain alignment methodology into a supervised contig generation algorithm. The pipeline of SALT (Supplementary Fig. 1) can be divided into three main stages: profile HMM-based filtration, contig generation and contig selection. In the first stage, we align query reads against profile HMMs, which represent the underlying domain families, using a modified Viterbi algorithm. A loose position-specific score cutoff (Zhang and Sun, 2012) is used to keep most positive reads. The first stage trades specificity for sensitivity, as some negative reads pass the filtration stage. The rationale behind the second and third stages for removing negative reads is that the positive reads tend to be sequenced from continuous regions of a domain and their assembled contigs can yield statistically significant alignment scores and *E*-values. The second stage builds a family-specific *hit graph* using a supervised graph construction algorithm. Contigs can be efficiently generated from each hit graph. The third stage aligns these contigs to input families using HMMER. *E*-values are computed to choose sequences that are part of domain regions. Reads from the high-scoring contigs are classified into the corresponding family.

3.2 Stage 1: profile HMM-based filtration

The first stage of SALT aligns reads to profile HMMs, which are built on homologous domain sequences from Pfam. The software suite HMMER uses Viterbi and forward algorithms to determine the membership of query sequences via *E*-value cutoffs. However, the sensitivity of HMMER drops significantly for short sequences or poorly conserved protein domain families. To address this problem, SALT uses position-specific score thresholds (PSSTs) and a modified Viterbi algorithm (Zhang and Sun, 2012) to align reads to profile HMMs. For the standard Viterbi algorithm on a profile HMM, we refer users to the detailed introduction in Durbin *et al.* (1998). The modified Viterbi algorithm used in SALT differs from the standard one in the following aspects: (i) SALT directly aligns a DNA sequence against a profile HMM by implicitly translating the DNA sequence into peptides under different reading frames; (ii) a local alignment can start and end with any state without any transition penalty; and (iii) SALT uses PSSTs as alignment score cutoffs.

3.2.1 Position-specific score threshold PSST was designed for MetaDomain (Zhang and Sun, 2012) to increase the short read classification sensitivity by profile HMMs. Instead of using a family-specific score cutoff, PSST defines different cutoffs depending on the alignment positions. For an alignment falling between two match states M_i and M_j in a profile HMM M, the score cutoff is $\gamma U_{i,j}$, where γ is a user-specified parameter in the range of [0, 1]. $U_{i,j}$ is the maximum alignment score between M_i and M_j. Gamma controls the strength of the filtration. Large γ decreases the sensitivity of filtration, whereas small γ introduces false-positive hits. Users can adjust γ to accommodate their specific needs. The default value of γ is 0.3, which was used in all our experiments. When γ is small, a read may be classified into multiple families in the first stage. We require that each read be assigned to at most three input families that generate the best alignment scores.

3.3 Stage 2: contig generation

The filtration stage uses loose PSSTs to trade specificity for sensitivity. Some negative reads can pass the filtration stage. Given medium to high sequencing coverage, positive reads are usually sequenced from continuous regions in a domain. Accordingly, their alignments to the underlying profile HMM overlap. This stage assembles reads with overlapping alignments. Specifically, we construct a family-specific *hit graph* using reads classified to the underlying family by the first stage. A path-searching algorithm is then applied to each hit graph to generate contigs for each domain family.

3.3.1 Constructing a hit graph for a family A standard overlap graph is defined as $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, in which each read is a node and overlaps larger than a given cutoff are indicated by directed edges. A hit graph originates from an overlap graph. However, its construction and the graph traversal algorithms are different from a standard overlap graph. We describe the hit graph construction procedure using Figure 2 as an example, focusing on the differences between a hit graph and a standard overlap graph.

3.3.2 Step 1 A node is created for each read passing the filtration stage for a domain family. In Figure 2, a read h_i creates a node v_i in the graph. Thus, different from a standard overlap graph, only reads that are likely to be classified into a family are used to build the hit graph. In addition, a hit graph is family specific. N hit graphs will be built for N input families.

3.3.3 Step 2 Edges are created using alignment positions of reads. For any two reads h_i and h_j that are classified into a family by the filtration stage, their overlap is computed if and only if the alignments of h_i and h_j overlap. If the overlap between h_i and h_j is at least k^*, which is a user-specified overlap threshold, a directed edge is created from h_i to h_j. For example, in Figure 2A, which shows the alignment layout for 13 reads, the alignment of h_1 and h_2 overlap and h_1 has a smaller starting position. SALT only tests whether a suffix of h_1 is a prefix of h_2. In addition, as the alignments of reads h_1, h_2 and h_3 share no overlap with alignments of reads from h_6 to h_{13}, SALT does not test the sequence overlaps between any two reads from the two sets. If two reads have the same starting position in their alignments, the bi-directional overlaps will be computed. Figure 2B shows the added edges. Different from the standard overlap graph construction, this step takes advantage of the alignment positions and presents an efficient family-specific graph construction procedure.

The graph construction stage allows substitution sequencing errors, which are the major error type with the Illumina sequencing technology. During the sequence overlap computation, at most e errors are permitted; $e = 2$ in the current implementation. The parameter e can be adjusted to fit the error rate of the given data. Given e, the overlap $o_{i,j}$ from read h_i to h_j is the size of the longest suffix of h_i that has Hamming distance $\leq e$ to a prefix of h_j.

Users can specify k^*, which can be estimated based on the same rationale as the choice of k-mer size in a *de novo* sequence assembly tool. The default value of k^* is set to half of the average read length for relatively short reads. When the reads are long (~100 bp), using the default value is too stringent for hit graph construction. In addition, the filtration stage is more specific with longer reads. Thus, we recommend using a smaller value than the default one.

It is worth mentioning that using alignment information to supervise the graph construction was also used for RNA virus population

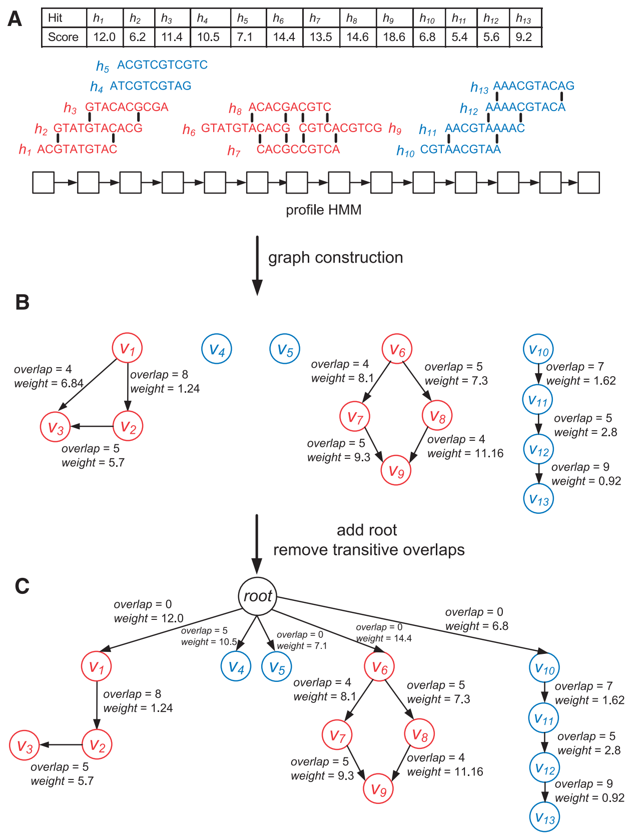

Fig. 2. (**A**) Thirteen reads and their alignment layout with regard to the profile HMM represented by its match states. The alignment scores are shown in the table. Blue reads: negative reads. Red reads: positive reads. (**B**) The constructed hit graph when $k^* = 4$. For simplicity of explanation, mismatches are not allowed in this simple example. Red nodes are created by positive reads. Blue nodes are created by negative reads. (**C**) The hit graph after removing transitive overlaps and adding the root node

estimation (Eriksson *et al.*, 2008). But the application, input data and alignment stage are all different from SALT.

3.3.4 Step 3 The graph G built so far contains many near-identical paths because of edges created by transitive overlaps (Schatz *et al.*, 2010). For example, the overlap between reads h_1 and h_3 in Figure 2A is a transitive overlap because longer overlaps exist between h_1 and h_2, h_2 and h_3. Transitive overlaps add unnecessary edges without contributing to read classification because those reads can be identified using an alternative (usually longer) path. Thus, all edges corresponding to transitive overlaps [such as (v_1, v_3)] are removed.

3.3.5 Step 4 We add edge weights, which are defined based on both the overlaps and the alignment scores of reads. Contigs connecting positive reads can be aligned to the underlying domain family with much better scores than the short reads, rendering significantly better discriminative power. Thus, we assign the edge weights so that the path weight is proportional to the alignment score of the corresponding contig. Negative reads that are sequenced from a continuous region may happen to pass the filtration stage and thus can form a path in G as well. However, their summed alignment score (i.e. the path weight) will be much smaller. Thus, by outputting the long paths, we have a better chance to pick positive reads.

The *weight* of an edge (v_i, v_j) between reads h_i and h_j is proportional to the increased alignment score contributed by the ending read h_j: $(v_i, v_j).weight = \frac{h_j.score*(|h_j|-o_{i,j})}{|h_j|}$, where $h_j.score$ is the alignment score of the read h_j,$|h_j|$ is the read length and $o_{i,j}$ is the overlap between h_i and h_j defined in this section. The edge weights computed using the above equation are shown in Figure 2B. To incorporate alignment scores for nodes that have zero in-degree (i.e. no incoming edges, such as v_1 and v_6 in Fig. 2B), we add a *root* node to **G** such that there is an edge from the root to every node with zero in-degree. The *overlap* of these edges is set to 0. Naturally, the edge weights are equal to the alignment scores of these nodes (reads). For examples, refer to the edges $(root, v_1)$ and $(root, v_6)$ in Figure 2C.

The aforementioned steps are applied to each family that has at least one classified read after the filtration stage. The pseudocode for the construction of **G** can be found in Supplementary Procedure S1.

3.3.6 Find the K longest paths Every path in a hit graph defines a contig. The weight of a path is the sum of weights of all the edges in the path. Contigs assembled from positive reads generally have larger scores and thus larger path weights. Therefore, to classify positive reads, we need to find the *K* longest paths (KLPs) in G, where *K* is a parameter that controls the number of contigs that will be generated.

To use a small set of paths to cover a majority of positive reads, we should avoid outputting a path that is a subpath of any other generated path. Considering that the weights of all edges are positive, the KLPs will always begin with the root and end with a node without outgoing edges (i.e. sinks).

Using the sorted alignment positions to construct **G** ensures that **G** is a directly acyclic graph (DAG). To find the KLPs in **G**, we apply a dynamic programming algorithm extended from the algorithm that finds the longest path in a DAG. Let $V' = \{v'_1, v'_2, ..., v'_N\}$ be the list of all nodes after topological sorting of **G**. Let *S* be a list of all sinks in **G**. Let *P*[0..*N*] be an array of $N+1$ lists such that *P*[*j*] keeps at most *K* longest paths that end with v'_j. *P*[0] is reserved for the root and is initialized to 0. Then we can fill *P*[0..*N*] using the following recurrence relation:

$$P[j] = KLargest_{(v'_i, v'_j) \in E, 1 \le k \le K}(P[i][k] + (v'_i, v'_j).weight),$$

where *KLargest* returns the *K* largest values in a list. This equation shows that the KLPs ending with a node are chosen from all the KLPs ending with its predecessors. The KLPs of **G** are the KLPs among all the paths that end with a sink node.

There are several algorithms available to find the *K* shortest simple (loopless) paths between two nodes in a graph (Brander and Sinclair, 1995; Eppstain, 1994; Yen, 1971). These algorithms can also be applied to the KLP problem by simply negating the weights of all edges. A hit graph is usually sparse, and therefore |**E**| is close to |**V**| (data will be shown in the 'Results' section). Moreover, *K* is generally smaller than |**V**|. Therefore, our DP algorithm is more efficient to solve our KLP problem than other general *K* shortest simple (loopless) path algorithms.

The choice of *K* aims at including contigs assembled from positive reads. Large *K* slows down the algorithm; small *K* can make SALT miss positive reads. As the size of the graph is not large because of the filtration step, we trade efficiency for sensitivity in the current parameter choice. By default, $K = |S|$, where $|S|$ is the number of sinks in **G**. When the hit graph is large, a smaller *K* is recommended for efficiency purpose.

3.4 Stage 3: *E*-value computation and contig selection

Although most of the top *K* longest paths contain only positive reads, some of them may be assembled from negative reads. In principle, some negative reads that are sequenced from continuous regions outside of domains can pass the filtration stage because of the loose PSSTs. These reads can form paths in **G**. When *K* is large, the paths containing negative reads will be included in KLPs. This happens more often for domains with low sequence conservations. Thus, we need a reliable and well-tested scoring system and cutoffs to distinguish between positive and negative contigs, which consist of positive and negative reads, respectively. To that end, we choose HMMER because it can provide efficient and reliable *E*-value computation. We first translate candidate contigs into peptides using multi-frame translation. Then these peptides are aligned against the input family using HMMER. As the contigs are much longer than the reads, HMMER is able to select positive contigs from the input using *E*-value cutoffs. Users can specify the *E*-value cutoff. The default *E*-value cutoff is 10^{-6} for this stage to ensure high specificity. All reads of the selected contigs will be classified into the input family. This is the final classification result for each family.

3.5 Running time analysis

Let the number of input reads be N and the average read length be $|h|$. The time complexity of the first stage is $O(N|h|M)$ for one family, where M is the number of match states in a family. M is mainly determined by the size of a domain family and $M << N$. Suppose there are N_1 reads passing the filtration stage. Usually, $N_1 << N$. In the second stage, the time complexity of graph construction is $O(\tau|h|^2N_1)$, where τ is the average number of overlapping alignments. During graph construction, we use alignment positions to guide the overlap computation, avoiding the all-against-all comparison needed in a standard overlap graph construction. The time complexity of searching for the KLPs is $O(K\log K|\mathbf{E}| + |\mathbf{V}|)$, where $|\mathbf{V}|$ is the number of nodes and $|\mathbf{E}|$ is the number of edges in the graph. According to the graph construction procedure, $|\mathbf{E}| \approx |\mathbf{V}| = N_1$. Suppose there are N_2 contigs produced from the hit graph. The time complexity of the last stage is determined by HMMER. Because of various optimization techniques and heuristics, the latest version of HMMER is as fast as BLAST (Eddy, 2009). In addition, under the default setup, N_2 is at most the number of sink nodes in the hit graph, and thus $N_2 < N_1$. As a result, the last stage only adds a small overhead to the overall time complexity. The overall time complexity of SALT is determined by the first stage, which is the bottleneck because of large input size N.

4 RESULTS

To show the utility of SALT, we applied it to three RNA-Seq datasets and compared its performance with HMMER, HHblits and a pipeline that uses *de novo* sequence assembly tools and HMMER. For brevity of description, we use assembly+HMMER to refer to the complete pipeline, where 'assembly' can be replaced by the name of a specific sequence assembly tool. This pipeline was used to classify query reads into input families based on the following steps: (i) use *de novo* sequence assembly tools to generate contigs from query reads, (ii) use HMMER to align translated contigs against input families and (iii) classify reads in the aligned regions of the contigs into the corresponding families. The inputs to HMMER and HHblits are always the translated peptides from the reads or contigs. For simplicity, when we say we classify reads into protein domain families using HMMER or HHblits, the translated peptides are actually used.

For the first two experiments, annotated reference genomes were available. We first determined the true membership of query reads based on protein domain annotations and read mapping positions in the genome. We then classified query reads into input families using four different types of classifiers: HMMER, HHblits, assembly+HMMER and SALT. Performance of the classifiers was determined by comparing the true membership of reads and predicted membership by the classifiers. We used four metrics to evaluate the performance of classifiers: sensitivity, false positive rate (FP rate), positive predicted value (PPV) and F-score. Let D represent an input family. Let A and B be the set of positive and negative reads of D, respectively. Let C be the set of reads classified to D by a classifier. The sensitivity of a classifier is defined by $\frac{|A\cap C|}{|A|}$. The FP rate is defined by $\frac{|C-A|}{|B|}$. The PPV is defined by $\frac{|A\cap C|}{|C|}$. The F-score considers both sensitivity and PPV and can be used as a single metric to evaluate the performance of a classifier. It is defined by

$$\mathrm{F-score} = \frac{2 \times \mathrm{sensitivity} \times \mathrm{PPV}}{\mathrm{sensitivity} + \mathrm{PPV}}.$$

To compare the performance of the classifiers on all input families, we first calculate a metric for each input family and report the average of the values of the metric over all input families.

In the last experiment, we conducted protein domain classification on an RNA-Seq dataset sequenced from a non-model organism. Although we did not have the annotations of the reference genome, we were able to show that SALT identified more transcribed protein domain families that were related to this species.

4.1 Protein domain classification of very short reads

In this experiment, we conducted protein domain classification on an Illumina RNA-Seq dataset sequenced from the transcriptome of one strain of *Burkholderia cenocepacia* named AU1054 in the growth medium cystic fibrosis (Yoder-Himes *et al.*, 2009). We downloaded 3 361 008 reads of 41 bp from the website provided by the authors. For assembly+HMMER, we chose Velvet (Zerbino and Birney, 2008) and SSAKE (Warren *et al.*, 2007) because Velvet is widely used and SSAKE is designed for very short reads. The experimental results show that SALT has much better performance than HMMER, HHblits and assembly+HMMER in classifying very short reads.

4.1.1 Determining the true membership of reads First, we downloaded the genome and its protein domain annotations from the IMG website (http://img.jgi.doe.gov/). There are 2181 annotated protein domains. Second, we mapped the reads back to the genome using Bowtie (Langmead *et al.*, 2009), with two mismatches allowed. Third, we compared the mapping positions of the reads and protein domains in the genome. Let l be the length of the overlap between a read and a protein domain and L be the length of the read. If $l \geq 0.8L$, this read has a significant overlap with the protein domain and is thus defined to be a positive read of the protein domain family. If $l < 0.5L$, the overlap is insignificant, and this read is a negative read of the protein domain family. Otherwise, the significance of the overlap is ambiguous, and this type of read was not evaluated in our experiments.

4.1.2 Performance evaluation Of the 2181 annotated protein domain families, we are interested in those that are transcribed in the input dataset. There is no commonly accepted criterion to define transcribed protein domains. In this work, we define protein domains that have at least 10 mapped reads as transcribed protein domains. There were 378 protein domain families with at least 10 mapped reads, which were used as the input families. When we adjusted this threshold, the performance comparison between different classifiers was generally unchanged.

We tried several different k-mer sizes for SSAKE and found that when the k-mer size was 16, SSAKE had the best overall performance. It generated 25 944 contigs with an average length of 60.36 bp. For Velvet, we used VelvetOptimizer (bioinformatics.net.au/software.velvetoptimiser.shtml) to search for the best assembly result by trying k-mer sizes from 15 to 39 bp. We tried both 'Lbp' and 'tbp' as the optimization function of VelvetOptimiser. The former optimizes the total number of base pairs in large contigs and the latter optimizes the total number of base pairs in all contigs. We found that 'Lbp' produced much higher sensitivity with similar PPV compared with 'tbp' and the optimal k-mer size for 'Lbp' was 25. Therefore, we report the performance of Velvet+HMMER using 'Lbp' as the optimization function. For SALT, we set γ to the default value (0.3) for the filtration stage. On average, 0.23% of the input reads passed the filtration for one input family, showing that only a small portion of reads could pass the filtration stage and be used as the input to the graph construction. The overlap threshold k^* was set to 20 by default ($k^* = \frac{\text{read length}}{2}$). The average number of nodes and edges in a hit graph were 1575 and 1879, respectively.

E-value cutoffs determine the trade-off between sensitivity and FP rate of all classifiers. We plotted the ROC curves of these classifiers by changing the E-value cutoff of HMMER and HHblits from 10 to 10^{-9} with ratio 0.1 (Fig. 3).

HMMER was highly specific with FP rate $\leq 1.2 \times 10^{-7}$. However, its sensitivity was only 9.81% even with relaxed E-value cutoffs. When the E-value cutoff was changed from 10^{-5} to 10^{-6}, its sensitivity dropped significantly. When the E-value cutoff was $<10^{-6}$, its sensitivity was almost 0. Both HHblits and SSAKE+HMMER had low sensitivity and high FP rate. The highest sensitivity of HHblits was 6.72% and its lowest FP rate was 6.75×10^{-7}. The highest sensitivity of SSAKE+HMMER was 8.25% and its lowest FP rate was 1.65×10^{-6}. Although Velvet generated longer contigs than SSAKE, both of them missed most positive reads. The sensitivity of SALT was much higher than that of HMMER and Velvet+HMMER with a comparable FP rate. The experimental results also show that when we increased the E-value cutoff from 0.01 to 10, the performance of these classifiers remained almost unchanged. To boost the sensitivity of HMMER, HHblits and assembly+HMMER, we used 0.01 as their default E-value cutoff in our experiments hereafter unless otherwise specified. Based on this experiment, the default E-value cutoff for SALT was set to 10^{-6}. Although this is a stringent cutoff, as it yields good trade-off between sensitivity and PPV for very short reads, it is expected to be appropriate for datasets containing longer reads. Table 1 compares the average performance of these classifiers under their default E-value cutoffs.

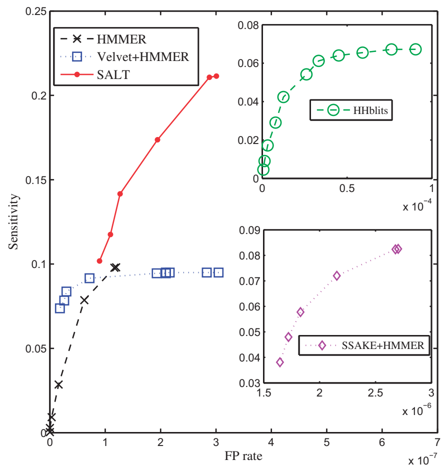

Fig. 3. ROC curves of different classifiers. HHblits and SSAKE+HMMER are listed in separate embedded windows because their FP rates are orders of magnitude larger than others

HMMER had the lowest FP rate and Velvet had the highest PPV. However, SALT had much higher sensitivity with good FP rate and PPV. Therefore, its F-score was significantly larger than the other classifiers, showing that it had the best overall classification performance. The running time of HMMER was much smaller than that of the other classifiers. The filtration stage of SALT is the bottleneck and costs most of the running time. The average running time of the three stages of SALT was 6.20 min, 2.47 min and 0.01 min, respectively. Because different input families can be analyzed independently, parallelization techniques can be applied to speed up this stage. This will be part of our future work.

As we described in the 'Methods' section, the filtration stage trades specificity for sensitivity. It determines the upper bound of the sensitivity of SALT using loose score cutoffs. The remaining stages are used to reduce the high FP rate introduced by the filtration stage. Our experimental results are consistent with the expectation. After filtration, the sensitivity was 39.32% and the PPV was 13.22%. The final sensitivity and PPV were 21.10% and 85.44%, respectively.

4.2 Protein domain classification of an RNA-Seq data of *Arabidopsis*

In this experiment, we applied SALT to an RNA-Seq dataset sequenced from a normalized cDNA library of *Arabidopsis* using paired-end Illumina sequencing (Marquez *et al.*, 2012). There were 9 559 784 reads of 76 bp from each end. The length of query reads was much longer than in the first experiment. We used Velvet and Oases (Schulz *et al.*, 2012) as the *de novo* sequence assembly tools in the assembly+HMMER pipeline. Oases takes into account a dynamic range of expression levels and is specially designed for the assembly of RNA-Seq data. The experimental results show that although HMMER, HHblits and assembly+HMMER have higher sensitivity with long query reads, SALT still had the best performance.

We used the starting ends of the paired-end reads as our benchmark dataset. We downloaded all the coding sequences (CDS) of *Arabidopsis thaliana* reference genome version TAIR10 (www.arabidopsis.org) and mapped the reads to the CDS using Bowtie with two mismatches allowed. Totally, 30.26% of the reads were mapped to the CDS. We annotated the protein domains in the CDS using HMMER with gathering thresholds. The set of reads mapped to each protein domain was

Table 1. Performance comparison of SALT against the other classifiers on the RNA-Seq dataset of *Burkholderia cenocepacia*

Classifiers	Sens (%)	FP rate	PPV (%)	F-score	Time (m)
HMMER	9.81	**1.20e-07**	89.37	0.1767	**0.03**
HHblits	6.40	4.51e-05	27.86	0.1041	0.65
SSAKE	8.06	7.23e-06	50.53	0.1415	1474
Velvet	9.50	2.16e-07	**94.03**	0.1725	2.50
SALT	**21.10**	3.00e-07	85.44	**0.3385**	8.68

Note: All these classifiers were run on a 2.2GHz dual-core AMD Opteron machine. The running time is the average running time on all input families. 'SSAKE' and 'Velvet' represent the pipelines of SSAKE+HMMER and Velvet+HMMER, respectively. The optimal value for each metric is highlighted in bold.

Table 2. Performance comparison of SALT against the other classifiers on the RNA-Seq data set of *Arabidopsis*

Classifiers	Sens (%)	FP rate	PPV (%)	F-score	Time (m)
HMMER	49.72	2.45e-06	86.26	0.6308	**0.31**
HHblits	50.09	2.32e-04	14.91	0.2298	11.98
Velvet	35.93	9.19e-06	64.45	0.4614	57.65
Oases	42.30	1.02e-05	64.59	0.5112	400.21
SALT	**58.46**	**2.84e-06**	**87.89**	**0.7021**	171.65

Note: All these classifiers were run on a 2.2GHz dual-core AMD Opteron machine. The running time is the average running time on all input families. 'Velvet' and 'Oases' represent the pipelines of Velvet+HMMER and Oases+HMMER, respectively. The optimal value for each metric is highlighted in bold.

determined using the same methodology as in the first experiment. There were 3188 protein domain families with at least 10 mapped reads.

Velvet achieved the best assembly result when the *k*-mer size was 45. It generated 44 161 contigs with an average length of 443.99 bp. The range of *k*-mer sizes used for Oases was from 31 to 61, and it generated 57 824 contigs with an average length of 356.46 bp. All parameters of SALT were set to their default values. Specifically, $k^* = 38$ because the read length is 76. Table 2 shows the comparison of the average performance of these classifiers on all input families.

SALT had the best performance on all metrics (Table 2). Oases+HMMER had better sensivitiy and similar PPV compared with Velvet+HMMER. However, both its sensitivity and PPV were much lower than for SALT. For all input families, the total number of reads correctly classified by SALT, HMMER, HHblits, Velvet+HMMER and Oases+HMMER were 968 068; 681 306; 741 160; 681 831; and 776,805, respectively. Significantly, more reads ($> 1.91 \times 10^5$) were correctly identified by SALT. Meanwhile, the sensitivities of all classifiers were significantly improved compared with those in the first experiment. This shows that profile HMM-based methods have better discriminative power with long sequences. Although Velvet+HMMER and Oases+HMMER correctly classified more reads than HMMER alone, their average sensitivities over all families were much lower than that of HMMER. A closer look at the results showed that Velvet/Oases+HMMER tended to perform well on highly transcribed families but missed a majority of reads for lowly transcribed families. On the other hand, HMMER and SALT are not as sensitive as Velvet and Oases to the transcriptional levels of domains. Thus, the numbers of families that are identified as transcribed by Velvet+HMMER and Oases+HMMER are smaller than for the other classifiers.

To show the performance of different classifiers on input families of different transcription levels, we investigated the performance of HMMER, Velvet+HMMER, Oases+HMMER and SALT on transcribed families when we changed the threshold for the number of mapped reads from 10 to 100 with a step size of 10. For input families that have at least 100 mapped reads, the reads per kilo base per million was 193.81. The performance of HMMER remained almost unchanged when we changed the threshold. This is because HMMER aligns each read independently. The sensitivity of Velvet+HMMER increased from 35.93% to 38.18%, and its PPV increased from 64.45% to 72.58%. The sensitivity of Oases+HMMER increased from 42.30% to 44.32%, and its PPV increased from 64.59% to 75.52%. The sensitivity of SALT increased from 58.46% to 59.39%, and its PPV increased from 87.89% to 91.05%. These results show that while Velvet/Oases+HMMER require sufficient coverage to assemble short reads, SALT can assemble short reads of variant coverage and achieves better classification performance.

4.3 Protein domain classification of a non-model organism

In this experiment, we show the utility of SALT in classifying a paired-end RNA-Seq dataset from the non-model organism *Radix balthica* (Feldmeyer *et al.*, 2012) into protein domain families. When the reference genome is not available, read mapping and genome annotation cannot be used to provide 'ground truth' about read membership. Thus, we focus on comparing the transcribed domains as well as the total number of reads classified into these families. In this experiment, the pipelines of assembler+HMMER will be represented by the assemblers' names for simplicity.

We first downloaded the dataset from NCBI SRA (SRP005151). There are 8 283 222 reads of 101 bp in each end. We also downloaded the contigs generated by several *de novo* sequence assembly tools from Feldmeyer *et al.* (2012). These tools include Velvet, SeqMan NGen (http://www.dnastar.com/t-sub-products-genomics-seqman-ngen.aspx) (hereafter called NGen) and Oases. The authors used 31 as the *k*-mer sizes for Velvet and NGen. For Oases, they used both 21 (Oases21) and 31 (Oases31) as *k*–mer sizes.

We searched for the keywords 'animal'and 'snail' in the Pfam website and obtained a list of 84 protein domain families that had these keywords in their description. These protein domain families were used as input families. If >10 reads could be classified to a family, we report that this family is transcribed.

We aligned the contigs generated by the assembly tools against the input families using HMMER, with 0.01 as the *E*-value cutoff. The parameters we used for SALT were as follows: $\gamma = 0.3$ (default) and *E*-value $= 10^{-6}$ (default); $k^* = 41$. As the reads are much longer than for the previous two experiments, we used a smaller k^* than the default value (50). Our choice of k^* is

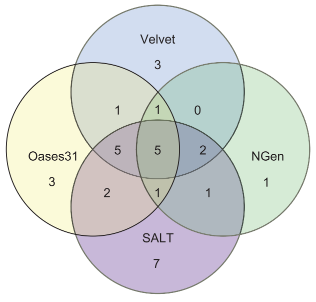

Fig. 4. Numbers of unique and shared transcribed families identified by different classifiers

already larger than k-mer sizes used by the tested assembly tools and is not likely to introduce random overlaps.

Even with a more stringent E-value cutoff, SALT identified 23 transcribed families, which are more than the other classifiers. Moreover, it classified 4457 reads into transcribed families, which are more than the other classifiers except NGen (8338 reads). Although NGen classified the largest number of reads, it only identified 13 transcribed families, which are thus likely to be highly transcribed families. Consistent with our second experiment, Velvet identified a smaller number of transcribed domains than SALT. A complete list of the number of identified transcribed families and classified reads by these classifiers can be found in Supplementary Table S1. The relationship between the sets of transcribed domains identified by different strategies is shown in Figure 4.

SALT identified seven transcribed families that were not identified by any other tool. To verify these uniquely identified families, we investigated whether these domains existed in the contigs generated by the sequence assembly tools. The output of running HMMER between these families and the contigs included statistically significant alignments. However, the number of reads consisting of the aligned regions was <10, and thus these domains were not reported as transcribed by assembly+HMMER. This indicates that these domains are likely to be transcribed, but the *de novo* sequence assembly tools failed to assemble many positive reads in the dataset.

SALT is able to identify more functional protein domains that reflect the species' reactions to the environment (Supplementary Table S2). For example, PF07829 is an alpha-A conotoxin PIVA-like protein family. It is the major paralytic toxin found in the venom produced by the piscivorous snail *Conus purpurascens* (Finn *et al*., 2010). The detailed annotations of these families are listed in Pfam's website.

5 DISCUSSION

Most existing domain analyses in RNA-Seq data still rely on traditional domain classification tools. In this work, we analyzed why a commonly used tool HMMER can incur low sensitivity for analyzing NGS data. We conducted a correlation study between the sensitivity of HMMER and other features based on the dataset in the second experiment. These features included the following: (i) the alignment score between the domain region and the input family (F_1); (ii) the length of the domain region in the transcript (F_2); and (iii) the normalized alignment score of the domain region in the transcript (F_3), defined as $F_3 = \frac{F_1}{F_2}$. The correlation coefficients between the sensitivity and these features were −0.0214, −0.0788 and 0.8415, respectively. The normalized alignment score of the domain region in the transcript had a strong positive linear relationship with the sensitivity of HMMER. This implies that reads sequenced from close homologs of a domain family can be better classified using profile HMM-based methods. On the other hand, reads sequenced from remote homologs of a domain family are harder to classify. We further found that the correlation coefficient between the sensitivity and average pairwise sequence identity in a domain was 0.5728. This indicates that the performance of profile HMM-based methods is related to the sequence conservation level of a domain.

The choice of parameters is important for SALT to provide good overall performance. Here, we give some analysis of the parameter K, the number of generated candidate contigs. The default value of K is the number of sinks in the hit graph. When the graph size is small, this K value works well. However, when the graph size is large, such as the hit graphs constructed from metagenomic datasets, the efficiency of SALT will decrease significantly. Theoretically, the number of sequenced contigs of a domain is the product of the average number of contigs in a domain and the number of copies of the domain in the genome. The former one can be estimated by the sequencing depth, and the latter can be inferred from existing domain organizations in Pfam. This approximation provides a better trade-off between classification performance and efficiency. However, if the users do not have good estimates for these values, the default K value is recommended.

The filtration stage of SALT dominates the time complexity of SALT. As alternatives to PPST used in this stage, we could use HMMER or HHblits to align query reads against the input families. However, as shown in our experiments, both HMMER and HHblits have low sensitivity for short reads. Therefore, PSST is an important strategy that increases the sensitivity of SALT. For long query reads, we could use HMMER or HHblits in the filtration stage to obtain a better trade-off between sensitivity and running time.

6 CONCLUSION AND FUTURE WORK

In this work, we introduced SALT, designed for transcriptomic data of organisms without high-quality reference genomes. Experiments on two RNA-Seq datasets from annotated genomes showed that SALT had higher sensitivity than existing tools with comparable specificity. When we applied SALT to an RNA-Seq dataset of a non-model organism, SALT identified more transcribed species-related families than alternative pipelines.

We plan to adapt SALT to domain classification in metagenomic data. In addition, where protein families are available, we will test SALT's performance on gene assembly. We also plan to improve the efficiency of SALT in several aspects. First, the hit graph construction is currently implemented using the Python library of NetworkX (http://networkx.lanl.gov/). The overhead of object constructions can be greatly decreased if we implement it using C++. Second, the independence of different input families allows us to use parallelization techniques such as MPI. Finally, we will provide users with a systematical way to better estimate K based on the sequencing data and domain information.

Funding: This work was supported by MSU start-up funding.

Conflict of Interest: none declared.

REFERENCES

Altschul,S.F. *et al.* (1990) Basisc local alignment search tool. *J. Mol. Biol.*, **215**, 403–410.

Brander,A.W. and Sinclair,M.C. (1995) A comparative study of k-shortest path algorithms. In: *Proceedings of 11th UK Performance Engineering Workshop for Computer and Telecommunications Systems*.

Durbin,R. *et al.* (1998) *Biological Sequence Analysis Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, UK.

Eddy,S.R. (2009) A new generation of homology search tools based on probabilistic inference. *Genome Inform.*, **23**, 205–211.

Eppstain,D. (1994) Finding the k shortest paths. In: *Proceedings of 25th IEEE Annual Symposium on Foundation of Computer Science*. pp. 154–165.

Eriksson,N. *et al.* (2008) Viral population estimation using pyrosequencing. *PLoS Comput. Biol.*, **4**, e1000074.

Feldmeyer,B. *et al.* (2012) Short read Illumina data for the *de novo* assembly of a non-model snail species transcriptome (*Radix balthica*, Basommatophora, Pulmonata), and a comparison of assembler performance. *BMC Genomics*, **12**, 317.

Finn,R.D. *et al.* (2010) The Pfam protein families database. *Nucleic Acids Res.*, **38(Suppl. 1)**, D211–D222.

Haft,D.H. *et al.* (2003) The TIGRFAMs database of protein families. *Nucleic Acids Res.*, **31**, 371–373.

Hunter,S. *et al.* (2009) InterPro: the integrative protein signature database. *Nucleic Acids Res.*, **37(Suppl. 1)**, D211–D215.

Jeffrey,A.M. and Zhong,W. (2011) Next-generation transcriptome assembly. *Nature Rev. Genet.*, **12**, 671–682.

Langmead,B. *et al.* (2009) Ultrafast and memory-efficient alignment of short dna sequences to the human genome. *Genome Biol.*, **10**, R25.

Li,Z. *et al.* (2011) RNA-seq improves annotation of protein-coding genes in the cucumber genome. *BMC Genomics*, **12**, 540.

Marquez,Y. *et al.* (2012) Transcriptome survey reveals increased complexity of the alternative splicing landscape in *Arabidopsis*. *Genome Res.*, **22**, 1184–1195.

Meyer,F. *et al.* (2009) FIGfams: yet another set of protein families. *Nucleic Acids Res*, **37**, 6643–6654.

Miller,J.R. *et al.* (2010) Assembly algorithms for next-generation sequencing data. *Genomics*, **95**, 315–327.

Mutasa-Göttgens,E. *et al.* (2012) A new RNAseq-based reference transcriptome for sugar beet and its application in transcriptome-scale analysis of vernalization and gibberellin responses. *BMC Genomics*, **13**, 99.

Orshinsky,A.M. *et al.* (2012) RNA-seq analysis of the *Sclerotinia homoeocarpa* creeping bentgrass pathosystem. *PLoS One*, **7**, e41150.

Quevillon,E. *et al.* (2005) InterProScan: protein domains identifier. *Nucleic Acids Res.*, **33(Suppl. 2)**, W116–W120.

Remmert,M. *et al.* (2012) HHblits: lightning-fast iterative protein sequence searching by HMM-HMM alignment. *Nat. Methods*, **9**, 173–175.

Schatz,M.C. *et al.* (2010) Assembly of large genomes using second-generation sequencing. *Genome Res.*, **20**, 1165–1173.

Schmid,M.W. *et al.* (2012) A powerful method for transcriptional profiling of specific cell types in eukaryotes: laser-assisted microdissection and RNA sequencing. *PLoS One*, **7**, e29685.

Schulz,M.H. *et al.* (2012) Oases: robust *de novo* RNA-seq assembly across the dynamic range of expression levels. *Bioinformatics*, **28**, 1086–1092.

Warren,R.L. *et al.* (2007) Assembling millions of short DNA sequences using SSAKE. *Bioinformatics*, **23**, 500–501.

Wommack,K.E. *et al.* (2008) Metagenomics: read length matters. *Appl. Environ. Microbiol.*, **74**, 1453–1463.

Yen,J.Y. (1971) Finding the K shortest loopless paths in a network. *Manag. Sci.*, **17**, 712–716.

Yoder-Himes,D.R. *et al.* (2009) Mapping the burkholderia cenocepacia niche response via high-throughput sequencing. *Proc. Natl Acad. Sci. USA*, **106**, 3976–3981.

Zerbino,D.R. and Birney,E. (2008) Velvet: algorithms for *de novo* short read assembly using de Bruijn graphs. *Genome Res.*, **18**, 821–829.

Zhang,Y. and Sun,Y. (2011) HMM-FRAME: accurate protein domain classification for metagenomic sequences containing frameshift errors. *BMC Bioinformatics*, **12**, 198.

Zhang,Y. and Sun,Y. (2012) MetaDomain: a profile HMM-based protein domain classification tool for short sequences. In: *Proceedings of Pacific Symposium on Biocomputing (PSB)*.