PyroTagger: A fast, accurate pipeline for analysis of rRNA amplicon pyrosequence data

Victor Kunin[1] and Philip Hugenholtz*[1]

[1]DOE Joint Genome Institute, 2800 Mitchell Drive, Walnut Creek, CA 94598, United States

Abstract

Pyrosequencing of small subunit ribosomal RNA amplicons (pyrotags) is rapidly gaining popularity as the method of choice for profiling microbial communities because it provides deep coverage with low cost. However, the large amount of data, and errors associated with the sequencing technology present significant analytical challenges. Here we describe PyroTagger, a computational pipeline for pyrotag analysis. The pipeline consists of read quality filtering and length trimming, dereplication, clustering at 97% sequence identity, classification and dataset partitioning based on barcodes. To speed up the rate-limiting clustering step we developed a novel purpose-built algorithm called pyroclust. PyroTagger is highly scalable, capable of processing hundreds of thousands of reads within minutes on a single CPU. Version 1.0 is available for public use at `http://pyrotagger.jgi-psf.org/`.

*corresponding author

This article was recommended by
Vanja Klepac-Ceraj as Technical interest
Morgan N Price as Technical interest
Dirk Gevers as Technical interest
Submitted on November 21, 2009
Accepted on February 21, 2010

1 Introduction

One emerging application of pyrosequencing [1] is microbial community profiling using small subunit (SSU) rRNA gene PCR amplicons [2]. In this application each read or pyrotag represents a member of the community and analysis typically involves clustering and classification to deduce community structure. Recently we and others demonstrated that pyrosequencing errors, in particular homopolymer length errors, can lead to inflated diversity estimates [3, 4]. To avoid interpreting sequencing errors as naturally occurring populations, we recommended accuracy trimming of sequences to 0.2% per-base error probability and clustering reads at a 97% sequence identity threshold [4]. The PCR also can introduce base errors and more importantly, chimeras of two or more DNA templates that need to be removed using chimera detection software [5]. Here we describe a fast, accurate pipeline for pyrotag classification called PyroTagger that implements these recommendations.

2 Methods

PyroTagger requires 3 files to operate; corresponding fasta and qual files from a 454 pyrosequencing run and an associated mapping file that contains the names and corresponding barcode-primer sequences of the multiplexed samples. Several sets of these files can be uploaded via web-interface if comparisons need to be made between runs. Compression of large files is recommended to reduce upload times

and network load. PyroTagger comprises the modules described below:

Input preprocessing. User-submitted files are uncompressed and carriage return characters are removed to ensure files are in UNIX format. Any ambiguous (wobble) characters in barcode-primer sequences in the mapping file are disambiguated. Only reads that exactly match disambiguated barcode-primer sequences are retained in the corresponding fasta and qual files.

Quality filtering. We have previously determined that stringent quality trimming reduces the number of spurious phylotypes by removing pyrosequencing errors [4]. Originally we used LUCY [6] to quality trim data to a 0.2% per-base error probability (Phred value of 27.0) [4], but this end-trimming program cannot remove low quality bases in otherwise high-quality regions (as is the case for homopolymeric errors) and demands additional dependencies for a standalone version of PyroTagger. Therefore, we replaced LUCY with a quality-filtering script that does not trim reads. Rather, it removes reads that, at a given read length, have $\geq$3% of bases with Phred values $<$27 (0.2% per-base error probability) to ensure that 97% clustering will absorb all erroneous reads. Length-dependent data loss using this filtering method is provided to the user as a graph (**Figure 1**). We have noted that lower quality sequencing runs may have unacceptably high data loss using this stringency threshold ($>$80% of reads, data not shown) and do not recommend using PyroTagger with these data. However, we have provided the user the ability to relax the stringency of the quality filtering to rescue low quality reads with no guarantee of the resulting richness estimates.

Length trimming. Reads are trimmed to a user-defined uniform length. Sequences for which the percentage of low quality bases exceed the user-specified threshold (see above) and sequences shorter than the length threshold are discarded. Uniform read lengths allow the use of faster computer algorithms to compare sequences and avoid the problem of unknown identity levels between non-overlapping sequence segments.

Dereplication. Dereplication of the length-

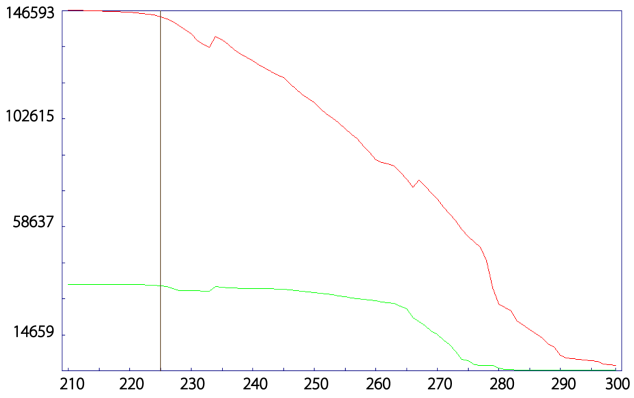

Figure 1: A graph of length-dependent data loss for quality-based filtering (reads are removed with $\geq$3% low quality bases ($<$Q27) over a given length). The green line shows the results of the small dataset [3], and the red line the results of the larger dataset [7]. Note the non-uniform loss of data with length. In some instances the fraction of low quality bases is reduced as sequence length increases and therefore more reads pass the 3% low quality threshod. For the larger dataset, length trimming at 225 bases (brown line) balances the desire for using the longest comparable region with the lowest data loss.

trimmed reads is achieved using hashing, with sequences as hash keys and occurrences as hash values. This step is implemented to occur together with length trimming but is separated here for clarity. The non-redundant sequences are sorted and written to an output file in the order of abundance in the dataset, most abundant sequences first. This is important since the following clustering stage will assume that correct sequences are more abundant than sequence error variants and will therefore appear earlier in the non-redundant sequence file.

Clustering. Dereplicated reads are clustered at a 97% sequence identity threshold and the most abundant unique sequence is used as a cluster representative. We previously established that a 97% threshold combined with read quality trimming produces reasonable estimates of microbial diversity [4]. Since clustering is the computational bottleneck in the pipeline, we developed a new two-stage algorithm called Pyroclust to scale with dataset size. Most algorithms for sequence comparison either attempt to find the best matching hits to a query sequence in a database or to find the best alignment between two sequences. In contrast, Pyroclust only attempts to group sequences within a user-specified distance, which does not require finding the best match or computing the complete pairwise alignment. It achieves this by taking advantage of particular properties of the pyrotag data and analysis pipeline, specifically identical sequence length, 5' positional homology of the reads and a high clustering threshold. The algorithm proceeds in two stages and is summarized in **Figure 2**. In future releases, we may modify or replace this algorithm to improve performance as datasets become larger.

Classification and sample partitioning. Representative sequences from each cluster are classified by comparison to the greengenes [8] and silva [9] databases for bacteria/archaea and eukaryotes respectively. The full taxonomy string for each reference sequence is exported from the greengenes and silva databases to allow taxonomic identification of pyrotag sequences. The current version of PyroTagger uses BlastN [10] to identify the top hit and its associated taxonomy for each 97% cluster representative. To improve performance, BlastN is performed in two stages; first a fast search using a word length of 90, then a second slower search using a word length of 30 for clusters lacking matches in the first round. Classified reads are separated into their respective samples using the barcodes provided in the original mapping file. Putatively chimeric clusters are identified as sequences having a best Blast alignment <90% of the trimmed read length to the reference database, >90% sequence identity to the best Blast match and cluster size ≤2. This provides a conservative estimate of chimeras in the dataset which are particularly prevalent in lower abundance clusters [3]. Putative chimeric clusters are flagged in the cluster classification output files (see below). Higher-level taxonomic information is collated for each sample by summing the number of reads belonging to a given phylum, excluding putatively chimeric reads.

Output files. Six compressed text files are emailed to the user: i) cluster_classification.xls; number of reads for, and classification of, each 97% cluster separated by sample, ii) cluster_classification_percent.xls; same as output (i) but counts are replaced by percentages, iii) phylum_classification.xls; number of reads for each phylum separated by sample, iv) phylum_classification_percent.xls; same as output (iii) but counts are replaced by percentages, v) cluster_representatives.fasta; the most abundant unique representative sequence for each 97% cluster and vi) clusters2reads.txt; a full mapping of read assignments to clusters. In addition, by clicking on the provided link, users can access a log file that contains metrics on the progress of their run, such as the current stage of the pipeline execution, number of input reads and the number of reads that pass quality filtering.

Benchmarking. Performance benchmarking was done on a single CPU of an AMD Opteron 2Gz machine with 12Gb of RAM using a small artificial community dataset of 46,341 454-FLX reads [3] and a larger termite hindgut dataset of 229,222 454-FLX reads [7] trimmed to 225 bp. Both datasets are available as supplementary information and at `http://pyrotagger.jgi-psf.org/sequences/`. Input file uploading times were measured for a remote DSL

```
Parameters:
#1: input_file=Input file with sequences of identical length ordered with more frequent sequences first
#2: float iden_thr=Identity threshold between any two sequences
#3: int (short_k_mer_length,long_k_mer_length)=Length of short_k_mers and long_k_mers

#### Data structures:
container sequence # contains name and sequence
hash k_mers_hash # Keys are long_k_mers, values are pointers to sequences
hash clusters_hash # Keys are sequences, values are arrays of sequences
int seq_length= length_of_each_sequence(I_f) # all sequences are of identical length
int min_common_k_mers=(seq_length-short_k_mer_length+1)-short_k_mer_length*(seq_length(1 - iden_thr))
String long_k_mer
hash deleted_clusters # keys are sequences
int (i,j)

#### Algorithm flow
# Stage 1, coarse & quick clustering
foreach sequence S_x in input_file
  array long_k_mers_array = split_sequence_to_k_mers(S_x, long_k_mer_length)
  foreach long_k_mer in long_k_mers_array
    array seqs_containing_long_k_mer = k_mers_hash{long_k_mer}
    bool matching_sequence_found = false
    foreach sequence S_y in seqs_containing_long_k_mer
      int num_common_shortwords_S_x_S_y=find_common_k_mers(S_x,S_y,short_k_mer_length)
      if (num_common_shortwords_S_x_S_y > min_common_k_mers) &&
         (optimized_Needleman-Wunsch_identity(S_x,S_y) >= iden_thr)
        push clusters_hash{S_y},S_x    # pushing into anonymous array inside a hash
        matching_sequence_found = true # S_x is now part of cluster represented by S_y
        break foreach S_y
      end if
    end foreach S_y
  end foreach long_k_mer
  if matching_sequence_found == false # S_x is a centroid of a new cluster
    foreach long_k_mer in long_k_mers_array
      push k_mers_hash{long_k_mer},S_x
    end foreach long_k_mer
    push clusters_hash{S_x},S_x
  end if
end foreach S_x

# Stage 2, refined & slow clustering
array clusters_array = sort_by_descending_number_of_ members(keys clusters_hash)
for i=1:clusters_array_size
  sequence S_x = clusters_array[i]
  next i if exists deleted_clusters{C_x}
  for j=i+1:clusters_array_size
    next j if exists deleted_clusters{C_y}
    sequence S_y = clusters_array[j]
    int num_common_shortwords_S_x_S_y=find_common_k_mers(S_x,S_y,short_k_mer_length)
    if (num_common_shortwords_S_x_S_y > min_common_k_mers) &&
       (optimized_Needleman-Wunsch_identity(S_x,S_y)  >= iden_thr)
      foreach sequence S_yy in C_y
        push clusters_hash{C_x}, S_yy
      end
      deleted_clusters{C_y}=true
      delete clusters_hash{C_y}
      break for j
    end if
  end for j
end for i

# Output clusters
array clusters_array = sort_by_descending_number_of_ members(keys clusters_hash)
foreach sequence S in clusters_array
  print_representative_sequence_and_names_of_member_sequences(S)
end foreach
```

Figure 2: Pseudocode summarizing the Pyroclust algorithm. The current implementation uses words of 6 and 80 nucleotides for short and long words respectively. The source code of the perl implementation of this algorithm is available for download as part of the PyroTagger pipeline at `http://pyrotagger.jgi-psf.org/release/`.

connection service (~350 kbps).

For accuracy benchmarking, 90 reference Sanger clone sequences, from which the small pyrotag dataset was derived, were clustered at 97% sequence identity to determine the actual richness and species abundance of this artificial community dataset. This was done using PyroTagger and a manual tree-based method. For the PyroTagger analysis, Sanger sequences were 5'-trimmed to *E.coli* position 824 and clustered with and without the pyrosequencing reads. For the tree-based analysis, the 90 Sanger reference sequences and 39 pyrotag cluster representatives were loaded into an ARB database [11] and manually aligned. A 50% consensus filter was generated from the sequences comprising 163 unambiguously saligned positions (to mimic the comparable length used by PyroTagger) and a neighbor-joining tree generated using the filter. Sequences were then manually clustered at a 97% identity threshold based on tree topology and checking uncorrected similarities between sequences in a cluster to determine richness and abundance of the dataset. Note that the 90 clones were mixed at different concentrations to mimic a natural community and therefore weighting was applied to the clone sequences to determine actual abundances. The ARB database is available at `http://pyrotagger.jgi-psf.org/sequences/quince`.

3 Results and Discussion

Performance. The performance of different modules of the PyroTagger pipeline for two samples of differing size and species richness is overviewed in **Figure 3**. The input file uploading, preprocessing, quality filtering and length trimming scale linearly with dataset size (O(n) complexity). The complexity of the dereplication step also is O(n) because it uses hashing and is mostly limited by disk operations (reading and writing) typically only taking seconds, even for datasets comprising millions of reads. For example, it takes 4 seconds to dereplicate the 229,222 read dataset on a single AMD 64-bit processor (**Figure 3**).

The computational bottleneck in the pipeline is

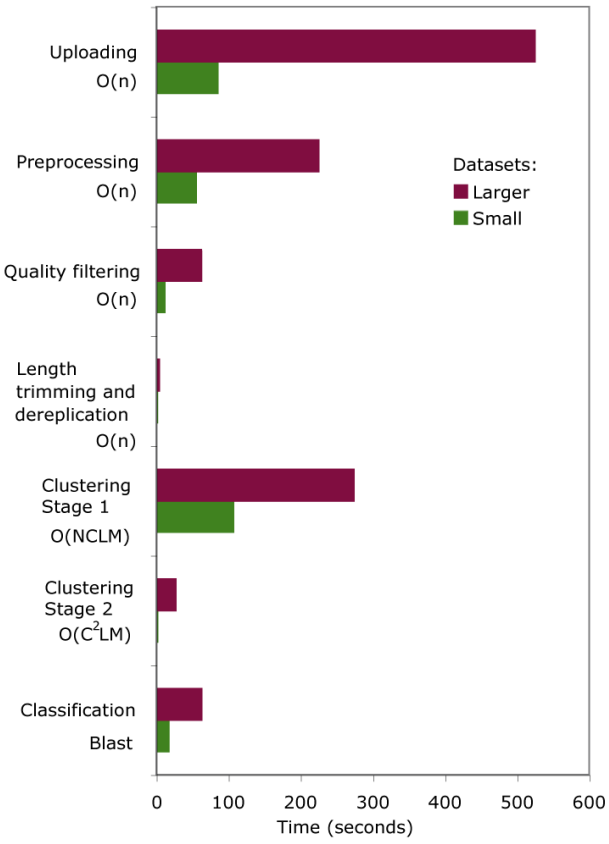

Figure 3: Performance benchmarking of the major Pyrotagger modules on two 454-FLX pyrotag datasets trimmed to 225 bp. The larger dataset [7] has 5X more reads and 10X more 97% clusters than the smaller dataset [3].

the sequence comparisons required for clustering. To address this bottleneck, we developed a 2-stage clustering algorithm, Pyroclust, which avoids most pairwise sequence comparisons (**Figure 2**). The complexity of the Pyroclust algorithm depends on the number of initial sequences N, number of clusters C, sequence length L and a permissible number of mismatches M. In the first coarse clustering stage of the algorithm, the worst case scenario is that every sequence is aligned to every cluster, creating O(NCLM) complexity. However, in practice most sequences are not aligned or aligned only to a small number of clusters, nearing O(NLM) complexity. The reason for this is twofold; i) most sequences do not share long words and thus will not be directly compared to each other, and ii) once a match is found, the query sequence is recruited to a cluster and only the cluster centroid will be compared from this point on. This results in approximately linear scaling by dataset size. Although the larger dataset is 5X as large as the smaller dataset tested, the processing time for stage 1 was only ∼2X longer for the larger dataset (**Figure 3**).

In the second fine clustering stage, every coarse cluster must be aligned to all others, adding $O(C^2LM)$ complexity. Therefore, the running time of stage 2 is highly dependent on the cluster (species) richness of the sample. For species-rich samples where C is high, stage 2 may become more computationally intensive than stage 1. For the two datasets tested, stage 2 times were substantially shorter than stage 1 (**Figure 3**) because cluster richness was moderate in both cases.

The final classification and sample partitioning module uses Blast and like the initial steps, does not impact performance, typically taking two minutes against the greengenes and silva databases.

The performance outlook for PyroTagger looks sound at least in the near term as it can analyze millions of reads (N) with a comparable length of up to 450 bp (L) in a few hours on a single CPU (data not shown). However, as the sequencing technologies continue to improve, both N and L will increase and the performance of PyroTagger will need to be reevaluated.

Accuracy. Classification accuracy has been extensively addressed elsewhere [12]. To test the accuracy of cluster richness and abundance estimates, we used the reference dataset of Quince et al [3]. This is the 46,341 454-FLX read dataset also used for performance testing (above), which was obtained by sequencing a non-equimolar mix of 90 rRNA clones. Accuracy was determined by co-analyzing the Sanger reference and pyrotag datasets with PyroTagger to ensure comparability of clusters. The cluster richness and abundance of the Sanger clones was independently verified using a tree-based assessment of clusters (ARB database is available at `http://pyrotagger.jgi-psf.org/sequences/quince`) with weighting applied to account for the variable clone concentrations in the mixture. The PyroTagger and tree-based analysis produced 34 and 35 clusters respectively from the 90 reference clones using a 97% identity threshold. PyroTagger analysis of the 46,341 pyrotags produced a richness estimate of 39 clusters of which 34 had matches to Sanger sequences according to the more accurate tree-based method (**Figure 4**). Of the 5 clusters with no Sanger match, three were manually verified chimeras that eluded the chimera detection script, and the other two were the result of cluster splitting of borderline cases. No spurious clusters due to low quality sequences were produced indicating that the quality filtering step is efficient. In future versions of PyroTagger we will explore the possibility of refining the chimera detection component of the pipeline. The abundance estimates of the weighted Sanger and pyrotag datasets were consistent with abundant and rare 'populations' being correlated (R^2=0.966; **Figure 4**) and therefore PyroTagger provides a sound prediction of abundance distribution in this artificial community.

In summary, PyroTagger will quickly and accurately analyze amplicon pyrosequencing datasets up to at least hundreds of thousands of reads on a single processor. PyroTagger is not region-specific and can be used on any amplified segment of the SSU rRNA gene. It also can be used on other highly conserved genes, such as large subunit rRNAs, provided an appropriate reference database is supplied. The

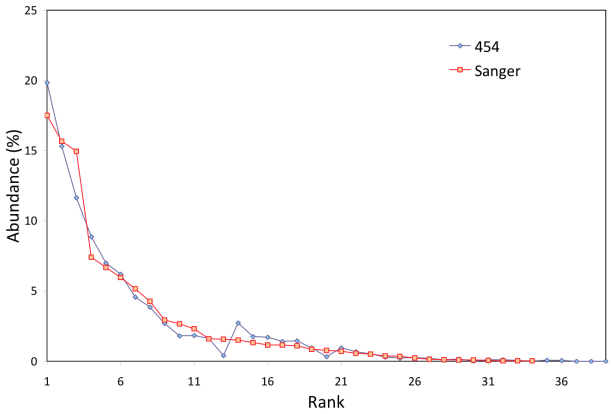

Figure 4: Rank abundance plots of 97% clusters produced from 454-FLX data (blue) and 90 associated reference Sanger sequences (red).

program is available as a web-based application at `http://pyrotagger.jgi-psf.org/` and the source code can be downloaded from `http://pyrotagger.jgi-psf.org/release/`.

4 Acknowledgments

We thank Amrita Pati and Ed Kirton for assistance with writing the pseudocode (Fig. 2) and Chris Quince for supplying the artificial community data and clone frequencies, and members of the Microbial Ecology Program at JGI for beta-testing PyroTagger. VK was supported in part by NSF grant OPP0632359. This work was performed under the auspices of the US Department of Energy's Office of Science, Biological and Environmental Research Program, and by the University of California, Lawrence Berkeley National Laboratory under contract No. DE-AC02-05CH11231, Lawrence Livermore National Laboratory under Contract No. DE-AC52-07NA27344, and Los Alamos National Laboratory under contract No. DE-AC02-06NA25396.

5 Author contributions

VK performed experiments; VK and PH conceived experiments; VK and PH analyzed data; VK and PH wrote the paper.

6 Conflicts of interest

VK, PH declared no conflict of interest.

References

[1] Margulies, M. and Egholm, M. and Altman, W.E. and Attiya, S. and Bader, J.S. and Bemben, L.A. and Berka, J. and Braverman, M.S. and Chen, Y.J. and Chen, Z. and Dewell, S.B. and Du, L. and Fierro, J.M. and Gomes, X.V. and Godwin, B.C. and He, W. and Helgesen, S. and Ho, C.H. and Ho, C.H. and Irzyk, G.P. and Jando, S.C. and Alenquer, M.L. and Jarvie, T.P. and Jirage, K.B. and Kim, J.B. and Knight, J.R. and Lanza, J.R. and Leamon, J.H. and Lefkowitz, S.M. and Lei, M. and Li, J. and Lohman, K.L. and Lu, H. and Makhijani, V.B. and McDade, K.E. and McKenna, M.P. and Myers, E.W. and Nickerson, E. and Nobile, J.R. and Plant, R. and Puc, B.P. and Ronan, M.T. and Roth, G.T. and Sarkis, G.J. and Simons, J.F. and Simpson, J.W. and Srinivasan, M. and Tartaro, K.R. and Tomasz, A. and Vogt, K.A. and Volkmer, G.A. and Wang, S.H. and Wang, Y. and Weiner, M.P. and Yu, P. and Begley, R.F. and Rothberg, J.M. Genome sequencing in microfabricated high-density picolitre reactors. *Nature*, 437(7057):376–80, 2005.

[2] Sogin, M.L. and Morrison, H.G. and Huber, J.A. and Mark Welch, D. and Huse, S.M. and Neal, P.R. and Arrieta, J.M. and Herndl, G.J. Microbial diversity in the deep sea and the underexplored "rare biosphere". *Proc. Natl. Acad. Sci. U.S.A.*, 103(32):12115–20, 2006.

[3] Quince, C. and Lanzén, A. and Curtis, T.P. and Davenport, R.J. and Hall, N. and Head,

I.M. and Read, L.F. and Sloan, W.T. Accurate determination of microbial diversity from 454 pyrosequencing data. *Nat. Methods*, 6(9):639–41, 2009.

[4] Kunin, V. and Engelbrektson, A. and Ochman, H. and Hugenholtz, P. Wrinkles in the rare biosphere: pyrosequencing errors lead to artificial inflation of diversity estimates. *Environ. Microbiol.*, 10(3), 2009.

[5] Huber, T. and Faulkner, G. and Hugenholtz, P. Bellerophon: a program to detect chimeric sequences in multiple sequence alignments. *Bioinformatics*, 20(14):2317–9, 2004.

[6] Chou, H.H. and Holmes, M.H. Dna sequence quality trimming and vector removal. *Bioinformatics*, 17(12):1093–104, 2001.

[7] Engelbrektson, A. and Kunin, V. and Wrighton, K.C. and Zvenigorodsky, N. and Chen, F. and Ochman, H. and Hugenholtz, P. Experimental factors affecting pcr-based estimates of microbial species richness and evenness. *ISME J*, 2010.

[8] DeSantis, T.Z. and Hugenholtz, P. and Larsen, N. and Rojas, M. and Brodie, E.L. and Keller, K. and Huber, T. and Dalevi, D. and Hu, P. and Andersen, G.L. Greengenes, a chimera-checked 16s rrna gene database and workbench compatible with arb. *Appl. Environ. Microbiol.*, 72(7):5069–72, 2006.

[9] Pruesse, E. and Quast, C. and Knittel, K. and Fuchs, B.M. and Ludwig, W. and Peplies, J. and Glöckner, F.O. Silva: a comprehensive online resource for quality checked and aligned ribosomal rna sequence data compatible with arb. *Nucleic Acids Res.*, 35(21):7188–96, 2007.

[10] Altschul, S.F. and Madden, T.L. and Schäffer, A.A. and Zhang, J. and Zhang, Z. and Miller, W. and Lipman, D.J. Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic Acids Res.*, 25(17):3389–402, 1997.

[11] Ludwig, W. and Strunk, O. and Westram, R. and Richter, L. and Meier, H. and Yadhukumar, . and Buchner, A. and Lai, T. and Steppi, S. and Jobb, G. and Förster, W. and Brettske, I. and Gerber, S. and Ginhart, A.W. and Gross, O. and Grumann, S. and Hermann, S. and Jost, R. and König, A. and Liss, T. and Lüssmann, R. and May, M. and Nonhoff, B. and Reichel, B. and Strehlow, R. and Stamatakis, A. and Stuckmann, N. and Vilbig, A. and Lenke, M. and Ludwig, T. and Bode, A. and Schleifer, K.H. Arb: a software environment for sequence data. *Nucleic Acids Res.*, 32(4):1363–71, 2004.

[12] Liu, Z. and DeSantis, T.Z. and Andersen, G.L. and Knight, R. Accurate taxonomy assignments from 16s rrna sequences produced by highly parallel pyrosequencers. *Nucleic Acids Res.*, 36(18):e120, 2008.