OptiClust: Improved method for assigning amplicon-based sequence data to operational taxonomic units

```
##
## Attaching package: 'dplyr'


## The following objects are masked from 'package:stats':
##
##     filter, lag


## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

Sarah L. Westcott and Patrick D. Schloss†

† To whom correspondence should be addressed: pschloss@umich.edu

Department of Microbiology and Immunology, University of Michigan, Ann Arbor, MI

Abstract

Introduction

Amplicon-based sequencing has provided incredible insights into Earth's microbial biodiversity.

Numerous methods have been proposed

Needed an objective criteria to evaluate the various methods methods

Average neighbor is consistently the best method, but others including USEARCH and VSEARCH resulted in comparable results

Difficulty with amount of memory and time required to complete average neighbor algorithm is a significant hurdle to completion of the method.

Instead of retrospectively evaluating methods for their ability to correctly assign sequences we sought to develop a method that would prospectively assign sequences to OTUs to optimize classification metrics.

Clustering quality within the algorithm is assessed by counting the number of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) based on the pairwise distances. Sequence pairs that are within the user-specified threshold and are clustered together represent TPs and those in different OTUs are FNs. Those sequence pairs that have a distance larger than the threshold and are not clustered in the same OTU are TNs and those in the same OTU are FPs. These counts are used to calculate the optimization metric.

Results

OptiClust algorithm.

The OptiClust algorithm uses the metric that should be used to assess clustering quality, a list of all sequence names in the dataset, and the pairs of sequences that are within a desired threshold of each other (e.g. 0.03). A detailed description of the algorithm is provided for a toy dataset in the Supplementary Material. Briefly, the algorithm starts by placing each sequence either within its own OTU or into a single OTU. The algorithm proceeds by interrogating each sequence and re-calculating the metric for the cases where the sequence stays in its current OTU, is moved to each of the other OTUs, or is moved into a new OTU. The location that results in the best clustering quality indicates whether the sequence should remain in its current OTU or be moved to a different or new OTU. Each iteration consists of interrogating every sequence in the dataset. Although numerous options are available within the mothur-based implementation of the algorithm (e.g. sensitivity, specificity, accuracy, F1 score, etc.), the default metric is MCC because it includes all four parameters from the confusion matrix. The algorithm continues until the optimization metric stabilizes or until it reaches a defined stopping criteria.

OptiClust-generated OTUs are more robust than those from other methods.

To evaluate the OptiClust algorithm and compare its performance to other algorithms, we utilized six datasets including two synthetic communities and four previously published large datasets generated from soil, marine, human, and murine samples (Table 1). When we seeded the OptiClust algorithm with each sequence in a separate OTU and ran the algorithm until complete convergence, the MCC values were 15.3 and 16.6% higher than the OTUs using average neighbor and distance-based greedy clustering (DGC) with VSEARCH, respectively (Figure 1). The number of OTUs formed by the various methods was negatively correlated with their MCC value (ρ=-0.47; p=0). The OptiClust algorithm was considerably faster than the hierarchical algorithms and somewhat slower than the heuristic-based algorithms. Across the six datasets, the OptiClust algorithm was 39.5-times faster than average neighbor and 2.5-times slower than DGC with VSEARCH. The human dataset was a challenge for a number of the algorithms. OTUCLUST and SumaClust were unable to cluster the human dataset in less than 50 hours and the average neighbor algorithm required more than 48 GB of RAM. The USEARCH-based methods were unable to cluster the human data using the 32-bit free version of the software that limits the amount of RAM to approximately 3.5 GB. These data demonstrate that OptiClust generates significantly more robust OTU assignments than existing methods across a diverse collection of datasets with performance that is comparable to popular methods.

OptiClust stopping criteria.

By default, the mothur-based implementation of the algorithm stops when the optimization metric changes by less than 0.0001; however, this can be altered by the user. This implementation also allows the user to stop the algorithm if a maximum number of iterations is exceeded. By default mothur uses a maximum value of 100 iterations. The justification for allowing incomplete convergence was based on the observation that numerous iterations are performed that extend the time required to complete the clustering with minimal improvement in clustering. We evaluated the results of clustering to partial convergence (i.e. a change in the MCC value that was less than 0.0001) or until complete convergence of the MCC value (i.e. until it did not change between iterations) when seeding the algorithm with each sequence in a separate OTU (Figure 1). The small difference in MCC values between the output from partial and complete convergence resulted in a difference in the median number of OTUs that ranged between 2.0 and 19.0 OTUs. This represented a difference of less than 0.13%. Among the four natural datasets, between 3 and 5 were needed to achieve partial convergence and between 9.50 and 14 iterations were needed to reach full convergence. The additional steps required between 2.0 and 3.2 times longer to complete the algorithm. These results suggest that achieving full convergence of the optimization metric adds computational effort; however, considering full convergence took between 4 and 28 minutes the extra effort was relatively small. Although the mothur's default setting is partial convergence, the remainder of our analysis used complete convergence to be more conservative.

Effect of seeding OTUs on OptiClust performance.

As implemented within mothur, the OptiClust algorithm either starts with each sequence in a separate OTU or with all of the sequences in a single OTU. We repeated the complete convergence analysis, but seeded the algorithm with all sequences in a single OTU. We found that the MCC values for clusters generated seeding OptiClust with the sequences as a single OTU were between 0.2 and 11.5% lower than when seeding the algorithm with sequences in separate OTUs (Figure 1). Interestingly, with the exception of the 2 dataset (0.3% more OTUs), the number of OTUs was as much as 7.0% lower (4) than when the algorithm was seeded with sequence in separate OTUs. Finally, the amount of time required to cluster the data when the algorithm was seeded with a single OTU was between 1.3 and 3.4-times longer than if sequences were seeded as separate OTUs.

This analysis demonstrates that seeding the algorithm with sequences as separate OTUs results in the best OTU assignments in the shortest amount of time.

OptiClust-generated OTUs are as stable as those from other algorithms.

One concern that many have with *de novo* clustering algorithms is that their output is sensitive to the initial order of the sequences. An additional concern with the OptiClust algorithm is that it may stabilize at a local optimum. To evaluate these concerns we compared the results obtained using ten randomizations of the order that sequences were given to the algorithm. The median the coefficient of variation across the six datasets for MCC values obtained from the replicate clusterings using OptiClust was 0.1% (Figure 1). We also measured the coefficient of variation for the number of OTUs across the six datasets for each method. The median coefficient of variation for the number of OTUs generated using OptiClust was 0.1%. Confirming our previous results, all of the methods we tested were stable to stochastic processes. Of the method that involved randomization, the coefficient of variation for MCC values considerably smaller than the other methods and the coefficient of variation for the number of OTUs was comparable to the other methods. The variation observed in clustering quality suggests that the algorithm does not appear to converge to a locally optimum MCC value. More importantly, the random variation does yield output of a similarly high quality.

Time and memory required to complete Optimization-based clustering scales efficiently.

Although not as important as the quality of clustering, the amount of time and memory required to assign sequences to OTUs is a legitimate concern. To evaluate how the speed and memory usage scaled with the number of sequences in the dataset, we measured the time required and maximum RAM usage to cluster 20, 40, 60, 80, and 100% of the unique sequences from each of the natural datasets using the OptiClust algorithm (Figure 2). Within each iteration of the algorithm, each sequence is compared to every other sequence and each comparison requires a recalculation of the confusion matrix. This would result in a worst case algorithmic complexity on the order of Nˆ3, where N is the number of unique sequences. Because the algorithm only needs to keep track of the sequence pairs that are within the threshold of each other, it is likely that the implementation of the algorithm is more efficient. To empirically determine the algorithmic complexity, we fit a power law function to the data in Figure 2A. We observed power coefficients between 2.1 and 3.0 for the human and marine datasets, respectively. The algorithm requires storing a matrix that contains the pairs of sequences that are close to each other as well as a matrix that indicates which sequences are clustered together. The memory required to store these matrices is on the order of Nˆ2, where N is the number of unique sequences. In fact, when we fit a power law function to the data in Figure 2B, the power coefficients were 2.0. This analysis suggests that doubling the number of sequences in a dataset would increase the time required to cluster the data by 4 to 8-fold and increase the RAM required by 4-fold. It is possible that future improvements to the implementation of the algorithm could improve this performance.

Cluster splitting heuristic generates OTUs that are as good as non-split approach (Figure 6).

We previously described a heuristic to accelerate OTU assignments where sequences were classified to taxonomic groups and within each taxon sequences were assigned to OTUs using the average neighbor clustering algorithm. This can accelerate the clustering and reduce the memory requirements because the number of unique sequences is effectively reduced by splitting sequences across taxonomic groups. Furthermore, because sequences in different taxonomic groups are assumed to belong to different OTUs they are independent, which permits parallelization and additional reduction in computation time. Reduction in clustering quality are encountered in this approach if there are errors in classification or if two sequences within the desired threshold belong to different taxonomic groups. It is expected that these errors would increase as the taxonomic level goes from kingdom to genus. To characterize the clustering quality, we calculated the MCC values using OptiClust, average neighbor, and DGC with VSEARCH when splitting at each taxonomic level (Figure 3). For each method, the MCC values decreased as the taxonomic resolution increased; however, the decrease in MCC as not as large as the difference between clustering methods. As the resolution of the taxonomic levels increased, the clustering quality remained high, relative to clusters formed from the entire dataset (i.e. kingdom-level). The MCC values when splitting the datasets at the class and genus levels were within 97.4 and 93.0%, respectively, of the MCC values obtained from the entire dataset. These decreases in MCC value resulted in the formation of as many as 4.1 and 21.4% more OTUs, respectively, than were observed from the entire dataset. For the datasets included in the current analysis, the use of the cluster splitting heuristic is not worth the loss in clustering quality. However, as datasets become larger, it may be necessary to use the heuristic to clustering the data into OTUs. For example, we were unable to cluster the full human data using less than 48 GB of RAM; however, when we split the dataset at the family level, we were able to cluster the data with the limited resources.

Discussion

Restate most important contributions * Optimized clustering based on an objective criteria. Results are meaningfully better than existing methods * Added benefits include smaller CPU and RAM footprint * Result is efficient analysis of large datasets without sacrificing clustering quality as has been experienced in heuristic methods.

Choice of objective criteria * Value of using a metric that is based on all four parameters * Preference is for Matthew's Correlation Coefficient because… other options include F1 Score and accuracy

Preference for OptiClust over cluster splitting approach * Risks are mis-classification and artificially splitting similar sequences between OTUs because they classify to different taxa * Still potential to merge the methods for more efficient processing.

Data quality - analysis uses unique sequences and we should not expect an infinite number of unique sequences unless there is a large amount of random sequencing error.

Based on our model, we propose the following…

Optimization of clustering using composite metrics significantly improves clustering quality (Figure 2, 3, 4). There are multiple metrics available to assess clustering quality. The mothur-based implementation of OptiClust allows the user to use the MCC, F1 score, accuracy, sensitivity, specificity, and the sum of true positives and negatives. The MCC, F1 score, and accuracy are preferred because each of them incorporates all four values from the confusion matrix while others only utilize two values. It is relatively straightforward to implement other metrics. **Comparison of MCC, accuracy, and F1 score to each other for each dataset using**

the full set of sequences. Each metric outperformed the observed values for the other clustering algorithms indicating that regardless of the metric one uses to evaluate clustering quality, the OptiClust algorithm generates better OTU assignments than any of the other methods. The number of OTUs generated by an algorithm is often used as a metric for clustering quality. We did not observe a significant correlation between clustering quality as measured by MCC, F1 score, or accuracy and the number of OTUs generated by each algorithm. **Interestingly, the values of the MCC, F1 score, and accuracy generated when using each metric to optimize clustering were similar across implementations.** Based on these results and its previous use in the OTU assignment literature, the remainder of our analysis uses the MCC metric for optimization.

Materials and Methods

Sequence data and processing steps. To evaluate the OptiClust and the other algorithms we created two synthetic sequence collections and four sequence collections generated from previously published studies. The V4 region of the 16S rRNA gene was used from all datasets because it is a popular region that can be fully sequenced with two-fold coverage using the commonly used MiSeq sequencer from Illumina (1). The method for generating the simulated datasets followed the approach used by Kopylova et al. (2) and Schloss (3). Briefly, we randomly selected 10,000 uniques V4 fragments from 16S rRNA gene sequences that were unique from the SILVA non-redundant database (4). A community with an even relative abundance profile was generated by specifying that each sequence had a frequency of 100 reads. A community with a staggered relative abundance profile was generated by specifying that the abundance of each sequence was a randomly drawn integer sampled from a uniform distribution between 1 and 200. Sequence collections collected from human feces (5), murine feces (6), soil (7), and seawater (8) were used to characterize the algorithms' performance with natural communities. These sequence collections were all generated using paired 150 or 250 nt reads of the V4 region. We re-processed all of the reads using a common analysis pipeline that included quality score-based error correction (1), alignment against a SILVA reference database (4, 9), screening for chimeras using UCHIME (10), and classification using a naive Bayesian classifier with the RDP training set (11).

Implementation of clustering algorithms. In addition to the OptiClust algorithm we evaluated ten different *de novo* clustering algorithms. These included three hierarchical algorithms, average neighbor (AN), nearest neighbor (NN), and furthest neighbor (FN), which are implemented in mothur (v.1.39.0; **???**). Seven heuristic methods were also used including abundance-based greedy clustering (UAGC) and distance-based greedy clustering (UDGC) as implemented in USEARCH (v.6.1; **???**), abundance-based greedy clustering (VAGC) and distance-based greedy clustering (VDGC) as implemented in VSEARCH (v.X.X.X; **???**), OTUClust (v.X.X.X; **???**), SumaClust (v.X.X.X; **???**), and Swarm (v.2.1.1; **???**). With the exception of Swarm each of these methods uses distance-based thresholds to report OTU assignments. To judge the quality of the Swarm-generated OTU assignments we calculated the MCC value using thresholds incremented by 1% between 0 and 5% and selected the threshold that provided the optimal MCC value (**???**). We also assessed the ability of a previously describe heuristic to cluster sequences using the UDGC and OptiClust algorithms. In this heuristic sequences are split into bacterial families based on sequence classification and clustered within the taxonomic family (**???**). Finally, for our benchmarking analysis, we evaluate the memory and time requirements when using 8 processors with the UDGC and VDGC algorithms and with the taxonomic splitting heuristic.

Benchmarking. We evaluated the quality of the sequence clustering, reproducibility of the clustering, the speed of clustering, and the amount of memory required to complete the clustering. To assess the quality of the clusters generated by each method, we counted the cells within a confusion matrix that indicated how well the clusterings represented the distances between the pair of sequences (**???**). Pairs of sequences that were in the same OTU and had a distance less than 3% were true positives (TPs), those that were in different OTUs and had a distance greater than 3% were true negatives (TNs), those that were in the same OTU and had a distance greater than 3% were false positives (FPs), and those that were in different OTUs and had a distance less than 3% were false negatives (FNs). To synthesize the matrix into a single metric we used the Matthew's Correlation Coefficient, F1 score, and accuracy using the `sens.spec` command in mothur using the following equations.

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

$$F1_score = \frac{2 \times TP}{2 \times TP + FP + FN}$$

$$Accuracy = \frac{TP + TN}{(TP + FP + TN + FN)}$$

To assess the reproducibility of the algorithms we randomized the starting order of each sequence collection ten times and ran each algorithm on each randomized collection. We then measured the MCC, F1 score, and accuracy for each randomization and quantified their coefficient of variation (CV; the ratio of the standard deviation to the mean).

To assess how the the memory and time requirements scaled with the number of sequences included in each sequence collection, we randomly subsampled 20, 40, 60, or 80% of the unique sequences in each collection. We obtained 10 subsamples at each depth for each dataset and ran each collection (N= 50 = 5 sequencing depths x 10 replicates) through each of the algorithms. We used the timeout script to quantify the maximum RAM used and the amount of time required to process each sequence collection (https://github.com/pshved/timeout). We limited each algorithm to 48 GB of RAM, 50 hours, and unless otherwise specified, a single processor.

Data and code availability. The workflow utilized commands in GNU make (v.3.81), GNU bash (v.4.1.2), mothur (v.1.39.0; **???**), and R (v.3.3.0; **???**). A reproducible version of this manuscript and analysis is available at https://github.com/SchlossLab/Westcott_OptiClust_mSystems_2015.

Figures

Figure 1. OptiClust performance. Plot of MCC (A) and execution times (B) for the different starting conditions (C) A Number of steps required to converge with different thresholds

Figure 2. MCC values (A) and their coefficient of variation (B) and the number of observed OTUs (C) for comparison of *de novo* clustering algorithms when applied to four natural and two synthetic datasets. For the purposes of this analysis, clustering was limited to 48 GB of RAM and 50 hours of execution time. The median of 10 re-orderings of the data is presented for each method and dataset.

Figure 3. Demonstration of how execution time (A) and memory usage (B) scale with the number of unique sequences for each clustering algorithm. For the purposes of this analysis, clustering was limited to 48 GB of RAM and 50 hours of execution time.

Figure 4. Comparison of cluster.split and cluster for average neighbor, VSEARCH-based abundance-based greedy clustering, and OptiClust.

Supplemental text. Worked example of how OptiClust algorithm clusters sequences into OTUs.

Table 1. Description of datasets used to evaluate the OptiClust algorithm and compare its performance to other algorithms.

References

1. **Kozich JJ**, **Westcott SL**, **Baxter NT**, **Highlander SK**, **Schloss PD**. 2013. Development of a dual-index sequencing strategy and curation pipeline for analyzing amplicon sequence data on the MiSeq Illumina sequencing platform. Applied and Environmental Microbiology **79**:5112–5120. doi:10.1128/aem.01043-13.

2. **Kopylova E**, **Navas-Molina JA**, **Mercier C**, **Xu ZZ**, **Mahé F**, **He Y**, **Zhou H-W**, **Rognes T**, **Caporaso JG**, **Knight R**. 2016. Open-source sequence clustering methods improve the state of the art. mSystems **1**:e00003–15. doi:10.1128/msystems.00003-15.

3. **Schloss PD**. 2016. Application of a database-independent approach to assess the quality of operational taxonomic unit picking methods. mSystems **1**:e00027–16. doi:10.1128/msystems.00027-16.

4. **Pruesse E**, **Quast C**, **Knittel K**, **Fuchs BM**, **Ludwig W**, **Peplies J**, **Glockner FO**. 2007. SILVA: A comprehensive online resource for quality checked and aligned ribosomal RNA sequence data compatible with ARB. Nucleic Acids Research **35**:7188–7196. doi:10.1093/nar/gkm864.

5. **Baxter NT**, **Ruffin MT**, **Rogers MAM**, **Schloss PD**. 2016. Microbiota-based model improves the sensitivity of fecal immunochemical test for detecting colonic lesions. Genome Medicine **8**. doi:10.1186/s13073-016-0290-3.

6. **Schloss PD**, **Schubert AM**, **Zackular JP**, **Iverson KD**, **Young VB**, **Petrosino JF**. 2012. Stabilization of the murine gut microbiome following weaning. Gut Microbes **3**:383–393. doi:10.4161/gmic.21008.

7. **Johnston ER**, **Rodriguez-R LM**, **Luo C**, **Yuan MM**, **Wu L**, **He Z**, **Schuur EAG**, **Luo Y**, **Tiedje JM**, **Zhou J**, **Konstantinidis KT**. 2016. Metagenomics reveals pervasive bacterial populations

and reduced community diversity across the alaska tundra ecosystem. Front Microbiol **7**. doi:10.3389/fmicb.2016.00579.

8. **Henson MW**, **Pitre DM**, **Weckhorst JL**, **Lanclos VC**, **Webber AT**, **Thrash JC**. 2016. Artificial seawater media facilitate cultivating members of the microbial majority from the gulf of mexico. mSphere **1**:e00028–16. doi:10.1128/msphere.00028-16.

9. **Schloss PD**. 2010. The effects of alignment quality, distance calculation method, sequence filtering, and region on the analysis of 16S rRNA gene-based studies. PLoS Comput Biol **6**:e1000844. doi:10.1371/journal.pcbi.1000844.

10. **Edgar RC**, **Haas BJ**, **Clemente JC**, **Quince C**, **Knight R**. 2011. UCHIME improves sensitivity and speed of chimera detection. Bioinformatics **27**:2194–2200. doi:10.1093/bioinformatics/btr381.

11. **Wang Q**, **Garrity GM**, **Tiedje JM**, **Cole JR**. 2007. Naive bayesian classifier for rapid assignment of rRNA sequences into the new bacterial taxonomy. Applied and Environmental Microbiology **73**:5261–5267. doi:10.1128/aem.00062-07.