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

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

[1] **Abstract**

Introduction

Amplicon-based sequencing has provided incredible insights into Earth's microbial biodiversity. It has become common for studies to include sequencing millions of 16S rRNA gene sequences across hundreds of samples. This is three to four orders of magnitude greater sequencing depth than was previously achieved using Sanger sequencing. The increased sequencing depth has highlighted the limitations of existing bioinformatics tools for analyzing these sequences. Furthermore, these sequencing efforts have revealed novel taxonomic diversity that are not adequately represented in reference databases. Because of this, a critical component to the analysis of microbial biodiversity has been the assignment of 16S rRNA gene sequences and other marker gene sequences to operational taxonomic units (OTUs) without reliance on taxonomic classification.

A growing number of algorithms have been developed to assign sequences into OTUs and these can be clustered into three general categories. The first category of algorithms are classification-based and have been termed closed-reference or phylotyping. Sequences are compared to a reference collection and clustered based on the reference sequence that they are similar to. This approach is fast; however, struggles when a sequence is similar to multiple reference sequences that may have different taxonomies. The second category of algorithm have been called *de novo* because they assign sequences to OTUs without the use of a reference. These include hierarchical algorithms such as nearest, furthest, and average neighbor and methods that employ heuristics such as abundance or distance-based greedy clustering as implemented in USEARCH or VSEARCH, Sumaclust, OTUCLUST, and Swarm. These methods tend to be more computationally intense and it has proven difficult to know which method generates the best assignments. A third category of algorithm is a hybrid approach or open-reference clustering. Here sequences are assigned to OTUs using closed-reference clustering and sequences that are not within a threshold of a reference sequence are then clustered using a *de novo* approach. This category blends the strengths and weaknesses of the other method and adds the complication that closed-reference and *de novo* clustering use different OTU definitions.

Several metrics have emerged for assessing the quality of OTU assignment algorithms. These have included the time and memory required to run the algorithm, agreement between OTU assignments

3

and the sequences' taxonomy, sensitivity of an algorithm to stochastic processes, and the number of OTUs generated by the algorithm. Unfortunately, these methods fail to directly quantify the quality of the OTU assignments. An algorithm may complete with minimal time and memory requirements or generate an idealized number of OTUs, but the composition of the OTUs could be incorrect. These metrics also tend to be subjective. For instance, a method may appear to be recapitulate the taxonomy of a synthetic community with known taxonomic structure, but do a poor job when applied to real communities with poorly defined taxonomic structure or for sequences that are prone to misclassification. As an alternative, we developed an approach to objectively benchmark the clustering quality of OTU assignments. This approach counts 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 values can be synthesized into a single correlation coefficient, the Matthews' Correlation Coefficient (MCC), which measures the correlation between observed and predicted classifications and is robust to cases where there is an uneven distribution across the confusion matrix.

We previously used this approach to assess the OTU assignments of a variety of algorithms using synthetic and real datasets. Consistently, the average neighbor algorithm was identified as among the best or the best algorithm. The distance-based greedy clustering as implemented in VSEARCH has also performed well. The computational resources required to complete the average neighbor algorithm can be significant for large datasets and so there is a need for an algorithm that efficiently produces consistently high quality OTU assignments. Previously, we assessed the quality of the clusters after the completion of the algorithm or post hoc. In the current study we developed and benchmarked a new de novo clustering algorithm. Instead of assessing the methods post hoc, the new method uses a priori calculation of the MCC to direct the algorithm. The result is the OptiClust algorithm, which produces significantly better sequence assignments while making efficient use of computational resources.

4

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

5

84   and SumaClust were unable to cluster the human dataset in less than 50 hours and the average

85   neighbor algorithm required more than 45 GB of RAM. The USEARCH-based methods were unable

86   to cluster the human data using the 32-bit free version of the software that limits the amount of RAM

87   to approximately 3.5 GB. These data demonstrate that OptiClust generates significantly more robust

88   OTU assignments than existing methods across a diverse collection of datasets with performance

89   that is comparable to popular methods.

90   ***OptiClust stopping criteria.***

91   By default, the mothur-based implementation of the algorithm stops when the optimization metric

92   changes by less than 0.0001; however, this can be altered by the user. This implementation

93   also allows the user to stop the algorithm if a maximum number of iterations is exceeded. By

94   default mothur uses a maximum value of 100 iterations. The justification for allowing incomplete

95   convergence was based on the observation that numerous iterations are performed that extend

96   the time required to complete the clustering with minimal improvement in clustering. We evaluated

97   the results of clustering to partial convergence (i.e. a change in the MCC value that was less

98   than 0.0001) or until complete convergence of the MCC value (i.e. until it did not change between

99   iterations) when seeding the algorithm with each sequence in a separate OTU (Figure 1). The

100 small difference in MCC values between the output from partial and complete convergence resulted

101 in a difference in the median number of OTUs that ranged between 2.0 and 19.0 OTUs. This

102 represented a difference of less than 0.13%. Among the four natural datasets, between 3 and 5

103 were needed to achieve partial convergence and between 9.50 and 14 iterations were needed to

104 reach full convergence. The additional steps required between 2.0 and 3.2 times longer to complete

105 the algorithm. These results suggest that achieving full convergence of the optimization metric

106 adds computational effort; however, considering full convergence took between 4 and 28 minutes

107 the extra effort was relatively small. Although the mothur's default setting is partial convergence,

108 the remainder of our analysis used complete convergence to be more conservative.

109 ***Effect of seeding OTUs on OptiClust performance.***

110 As implemented within mothur, the OptiClust algorithm either starts with each sequence in a

111 separate OTU or with all of the sequences in a single OTU. We repeated the complete convergence

6

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

7

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 N^3, 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 N^2, 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

8

169 (Figure 3). For each method, the MCC values decreased as the taxonomic resolution increased;

170 however, the decrease in MCC as not as large as the difference between clustering methods. As

171 the resolution of the taxonomic levels increased, the clustering quality remained high, relative to

172 clusters formed from the entire dataset (i.e. kingdom-level). The MCC values when splitting the

173 datasets at the class and genus levels were within 97.4 and 93.0%, respectively, of the MCC values

174 obtained from the entire dataset. These decreases in MCC value resulted in the formation of as

175 many as 4.1 and 21.4% more OTUs, respectively, than were observed from the entire dataset. For

176 the datasets included in the current analysis, the use of the cluster splitting heuristic is not worth

177 the loss in clustering quality. However, as datasets become larger, it may be necessary to use the

178 heuristic to clustering the data into OTUs. For example, we were unable to cluster the full human

179 data using less than 45 GB of RAM; however, when we split the dataset at the family level, we were

180 able to cluster the data with the limited resources.

181 **Discussion**

182 Myriad methods have been proposed for assigning 16S rRNA gene sequences to OTUs that

183 each claim improved performance based on speed, memory usage, representation of taxonomic

184 information, and number of OTUs. Each of these metrics is subjective and do not actually indicate

185 the quality of the clustering. This led us to propose using the MCC as a metric for assessing the

186 quality of clustering, post hoc. Here, we described a new clustering method that seeks to optimize

187 clustering based on an objective criterion that measures clustering quality, a priori. In the OptiClust

188 algorithm clustering is driven by optimizing a metric that assesses whether any two sequences

189 should be grouped into the same OTU. The result are clusters that are significantly more robust

190 and is efficient in the time and memory required to cluster the sequences into OTUs. This makes

191 it more tractable to analyze large datasets without sacrificing clustering quality as was previously

192 necessary using heuristic methods.

193 The cluster optimization procedure is dependent on the metric that is chosen for optimization. We

194 employed the MCC because it includes the four values from a confusion matrix. Other algorithms

195 such as the furthest neighbor and nearest neighbor algorithms minimize the number of FP and

9

FN, respectively; however, these suffer because the number of FN and FP are not controlled. Alternatively, one could optimize based on the sensitivity, specificity, or accuracy, which are each based on two values from the confusion matrix or they could optimize based on the F1 score, which is based on three values from the confusion matrix. Because these metrics do not balance all four parameters equally, it is likely that one parameter will dominate in the optimization procedure. For example, optimizing for sensitivity could lead to a large number of FPs. Since we would like to minimize both FPs and FNs and not just the total number of false assignments, we decided to optimize utilizing the MCC. It is possible that other metrics could be developed and employed for optimization of the clustering.

The OptiClust algorithm is relatively simple. For each sequence it effectively ask whether the MCC value will increase if the sequence is moved to a different OTU including creating a new OTU. If the value does not change, it remains in the current OTU. The algorithm repeats until the MCC value stabilizes. Assuming that the algorithm is seeded with each sequence in a separate OTU, it does not appear that the algorithm converges to a local optimum. Furthermore, execution of the algorithm with different random number generator seeds produces OTU assignments of consistently high quality. Future improvements to the implementation of the algorithm could provide optimization to further improve its speed and susceptibility to find a local optimum. Users are encourage to repeat the OTU assignment several times to confirm that they have found the best OTU assignments.

Our previous MCC-based analysis of numerous clustering algorithms indicated that the average neighbor algorithm consistently produced the best OTU assignments with the DGC-based method using USEARCH also producing robust OTU assignments. The challenge in using the average neighbor algorithm is that it requires a large amount of RAM and is computationally demanding. This led to the development of a splitting approach that divides the clustering across distinct taxonomic groups. The improved performance provided by the OptiClust algorithm likely makes such splitting unnecessary for most current datasets. We have demonstrated that although the OTU assignments made at the genus level are still better than that of other methods, the quality is not as good as that found without splitting. The loss of quality is likely due to misclassification because of limitations in the clustering algorithms and reference databases. The practical significance of such small differences in clustering quality remain to be determined; however, based on the current analysis, it

225 does appear that the number of OTUs is artificially inflated. Regardless, the best clustering quality

226 should be pursued given the available computer resources.

227 The time required to execute the OptiClust algorithm scaled proportionally to the number of unique

228 sequences raised to the second or third power and the amount of RAM required scaled to the

229 second power. The power for the time requirement is affected by the similarity of the sequences

230 in the dataset with datasets containing more similar sequences having a higher power. Also, the

231 number of unique sequences is the basis for both the amount of time and memory required to

232 complete the algorithm. Both the similarity of sequences and number of unique sequences can be

233 driven by the sequencing error since any errors will increase the number of unique sequences and

234 these sequences will be closely related to the perfect sequence. This underscores the importance

235 of reducing the noise in the sequence data. If sequencing errors are not remediated and are

236 relatively randomly distributed, then it is likely that the algorithm will require an unnecessary amount

237 of time and RAM to complete.

238 The rapid expansion in sequencing capacity has demanded that the algorithms used to assign

239 16S rRNA gene sequences to OTUs be efficient while maintaining robust assignments. Although

240 database-based approaches have been proposed to facilitate this analysis, they are limited by their

241 limited coverage of bacterial taxonomy and by the inconsistent process used to name taxa. Thus,

242 the ability to assign sequences to OTUs with an a priori focus on clustering quality will significantly

243 enhance downstream analysis. The development of the OptiClust algorithm represents a significant

244 advance that is likely to have numerous other applications.

Materials and Methods

246 ***Sequence data and processing steps.*** To evaluate the OptiClust and the other algorithms we

247 created two synthetic sequence collections and four sequence collections generated from previously

248 published studies. The V4 region of the 16S rRNA gene was used from all datasets because it

249 is a popular region that can be fully sequenced with two-fold coverage using the commonly used

250 MiSeq sequencer from Illumina (1). The method for generating the simulated datasets followed the

11

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 requiring an 80% confidence score (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 (AGC) and (distance-based greedy clustering) DGC as implemented in USEARCH (v.6.1; **???**) and 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.

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

12

279 3% were false negatives (FNs). To synthesize the matrix into a single metric we used the Matthews

280 Correlation Coefficient using the `sens.spec` command in mothur using the following equations.

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

281 To assess the reproducibility of the algorithms we randomized the starting order of each sequence

282 collection ten times and ran each algorithm on each randomized collection. We then measured the

283 MCC for each randomization and quantified their percent coefficient of variation (% CV; 100 times

284 the ratio of the standard deviation to the mean).

285 To assess how the the memory and time requirements scaled with the number of sequences

286 included in each sequence collection, we randomly subsampled 20, 40, 60, or 80% of the unique

287 sequences in each collection. We obtained 10 subsamples at each depth for each dataset and ran

288 each collection (N= 50 = 5 sequencing depths x 10 replicates) through each of the algorithms. We

289 used the timeout script to quantify the maximum RAM used and the amount of time required to

290 process each sequence collection (https://github.com/pshved/timeout). We limited each algorithm

291 to 45 GB of RAM and 50 hours using a single processor.

292 **_Data and code availability._** The workflow utilized commands in GNU make (v.3.81), GNU bash

293 (v.4.1.2), mothur (v.1.39.0; **???**), and R (v.3.3.0; **???**). Within R we utilized the wesanderson

294 (v.X.X.X; **???**), dplyr (v.X.X.X; **???**), tidyr (v.X.X.X; **???**), cowplot (v.X.X.X; **???**), and ggplot2

295 (v.X.X.X; **???**) packages. A reproducible version of this manuscript and analysis is available at

296 https://github.com/SchlossLab/Westcott_OptiClust_mSystems_2015.

Figures

Figure 1. Comparison of de novo clustering algorithms. Plot of MCC (A), number of OTUs (B), and execution times (C) for the comparison of *de novo* clustering algorithms when applied to four natural and two synthetic datasets. The first three columns of each figure contain the results of clustering the datasets (i) seeding the algorithm with one sequence per OTU and allowing the algorithm to proceed until the MCC value no longer changed; (ii) seeding the algorithm with one sequence per OTU and allowing the algorithm to proceed until the MCC changed by less than 0.0001; (iii) seeding the algorithm with all of the sequences in one OTU and allowing the algorithm to proceed until the MCC value no longer changed. The human dataset could not be clustered by the average neighbor, Sumaclust, USEARCH, or OTUCLUST with less than 45 GB of RAM or 50 hours of execution time. The median of 10 re-orderings of the data is presented for each method and dataset. The confidence interval is indicated by the error bars, which are typically smaller than the plotting symbol.

Figure 2. OptiClust performance The average execution time (A) and memory usage (B) required to cluster the four natural datasets. The confidence intervals indicate the range between the minimum and maximum values. The y-axis is scaled by the square root to demonstrate the relationship between the time and memory requirements relative to the number of unique sequences squared.

Figure 3. Effects of taxonomically splitting the datasets on clustering quality. The datasets were split at each taxonomic level based on their classification using a naive Bayesian classifier and clustered using average neighbor, VSEARCH-based DGC, and OptiClust.

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

14

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

343 and reduced community diversity across the alaska tundra ecosystem. Front Microbiol **7**.

344 doi:10.3389/fmicb.2016.00579.

345 8. **Henson MW**, **Pitre DM**, **Weckhorst JL**, **Lanclos VC**, **Webber AT**, **Thrash JC**. 2016. Artificial

346 seawater media facilitate cultivating members of the microbial majority from the gulf of mexico.

347 mSphere **1**:e00028–16. doi:10.1128/msphere.00028-16.

348 9. **Schloss PD**. 2010. The effects of alignment quality, distance calculation method, sequence

349 filtering, and region on the analysis of 16S rRNA gene-based studies. PLoS Comput Biol

350 **6**:e1000844. doi:10.1371/journal.pcbi.1000844.

351 10. **Edgar RC**, **Haas BJ**, **Clemente JC**, **Quince C**, **Knight R**. 2011. UCHIME improves sensitivity

352 and speed of chimera detection. Bioinformatics **27**:2194–2200. doi:10.1093/bioinformatics/btr381.

353 11. **Wang Q**, **Garrity GM**, **Tiedje JM**, **Cole JR**. 2007. Naive bayesian classifier for rapid assignment

354 of rRNA sequences into the new bacterial taxonomy. Applied and Environmental Microbiology

355 **73**:5261–5267. doi:10.1128/aem.00062-07.