**A Comparison of Metagenomic Classification Techniques**

by J. Jedediah Smith

**Introduction**

The rapid and accurate identification of microbial communities is a crucial tool in many different fields. In medicine, there is a need for metagenomic identification pipelines to detect novel viruses that could be transmitted via blood transfusion (Sauvage & Eloit 2016). New clinical diagnosis tools also utilize metagenomics to help detect diseases (Govender et al. 2021). In public health, COVID-19 tests have been performed using metagenomic identification workflows (Carbo et al. 2020). Food safety has begun to rely on these techniques too, utilizing them to ensure the quality of various products (Billington et al. 2022). Regardless of the context, it is clear that timely and precise metagenomic identification pipelines would be an advantage to these varied applications. While such workflows may consist of many parts, like quality control, assemblers, classifiers, and data visualization, it is the classifiers that perform taxonomic classification and determine which species are present in a particular sample.

           Initially, most metagenomic classification was accomplished with markers (Ye et al. 2019). Marker-based analysis, sometimes referred to as 16S or DNA barcoding, relies on sequencing and comparing highly conserved DNA barcode genes to identify the species present in a sample (Jovel et al. 2016, Badotti et al. 2017, Edgar 2018). For prokaryotes, the most commonly used gene is 16S, but other types of organisms like fungi or eukaryotes use ITS (Ye et al. 2019, Badotti et al. 2017). While marker-based techniques can be useful to quickly determine what species are present in a sample, their abundance estimates can be biased if the organisms present possess differing concentrations of the barcode genes (Edgar 2018, Wang et al. 2015). This makes it more challenging to accurately assess the exact composition of a large, diverse community. However, promising full-genome approaches have been developed that help eliminate this bias, making them the new go-to tool for metagenomic classification.

           Over the past few years, whole-metagenomic shotgun techniques have become more dominant (Ye et al. 2019). Shotgun analysis relies on the full sequencing of everything present in a sample, then comparing it to a vast genome or protein library to identify which species are present (Jovel et al. 2016, Wang et al. 2015). Most shotgun classification is strictly DNA-based, which means that identification is completed by matching sample sequence nucleotides against genome libraries, similar to NCBI’s BLASTn tool (Ye et al. 2019, Jovel et al. 2016). However, there are also protein-based tools, which instead rely on extensive protein libraries and translate sample sequences into amino acids for comparison, similar to NCBI’s BLASTx tool (Ye et al. 2019). Although shown to be more sensitive than the typical DNA-based approaches, protein-based classification has been found to be more computationally intensive, as it must consider all six potential reading frames of a given sequence (Ye et al. 2019). Such techniques are best used in specific circumstances where high precision is a necessity, such as to distinguish the difference between variants of the same sequence.

DNA-based shotgun classification can be performed on either short-read or long-read sequence data (Pearman et al. 2020). The more common method is short-read data, as long-read data has been historically associated with higher error rates (Portik et al. 2022). However, with recent technological developments like the advent of “third generation” sequencing, the length and quality of long-read data has substantially improved, making it a more promising alternative (Pearman et al. 2020, Portik et al. 2022). Although long-read data is still prone to more errors than short-read data, the advantage of longer sequences for classification is evidenced by their superior accuracy, matching that of protein-based shotgun techniques (Ye et al. 2019, Pearman et al. 2020, Portik et al. 2022). In general, long-read classification software takes more time to run and is more resource intensive than short-read, but this difference seems to be shrinking (Ye et al. 2019). Most classification software is tailored specifically for either short-read or long-read data, so it is important to note what a particular program prefers.

Raw sequence data is often fed directly into a classification program, but this is not a required procedure. To help aid with identification, short-read sequences can first be assembled into longer contigs by matching up their overlapping regions (Huang 1992). Recent research has shown that feeding these longer, assembled contigs into classifiers significantly improves identification accuracy, making it a viable solution to enhance short-read data when long-read techniques cannot be utilized (Tran & Phan 2020). This process of contig classification enables short-read data to potentially reach the accuracy of long-read data without relying on more expensive sequencing techniques (Tran & Phan 2020). It should therefore be possible to combine assemblers and classifiers into a metagenomic identification pipeline for short-read data that can compete with long-read techniques.

The central purpose of this project is to explore DNA-based shotgun classification techniques to facilitate the creation of a commercially-scalable end-to-end metagenomic identification pipeline for CIAN Diagnostics. There are three experimental objectives I will address with this foray. First, determine whether a long-read, short-read, or contig approach is most accurate. There is little work on how the long-read and contig methods compare. While both have research that claim they are better than short reads, a three-way comparison would be insightful (Ye et al. 2019, Tran & Phan 2020, Portik et al. 2022). Second, compare performance between different short-read classifier programs: Kraken2, Centrifuge, and CCMetagen. While Kraken2 and Centrifuge have been compared in the past, CCMetagen is a newcomer that claims to be better than both of them (Ye et al. 2019, Marcelino et al. 2020). Third, establish how important quality control analysis is to a classification pipeline. The goal here is to see whether this extra step has an appreciable impact on classification results. Performance metrics such as precision, recall, F1 score, and F0.5 score will be analyzed to enable comparison. These are typical metrics to use when comparing classification tools (Portik et al. 2022). F1 is a metric that combines precision and recall into a single number. F0.5 is similar, but weights precision and recall equally. Better scores will indicate whether a certain method or tool is superior to another.

There is a hypothesis to accompany each of my experimental objectives. For the first, I hypothesize that long-read classification will prove the most accurate, as this is at the forefront of recent developments (Portik et al. 2022). Short-read contigs will probably be a close second, as this method has been shown to enhance the performance of standard short-read classification (Tran & Phan 2020). I suspect that short-read classification will have the worst performance. It has been show to be less reliable then the other techniques and serves as a baseline for this experiment (Ye et al. 2019). For the second, I anticipate that Kraken2 will be more superior for prokaryotes, but that CCMetagen will be better with eukaryotic species like fungi. While Kraken2 is a widely used classification program, it has been known to struggle with eukaryotic species, something that CCMetagen claims to address (Wood & Salzberg 2014, Marcelino et al. 2020). I expect Centrifuge to perform the worst. It is similar to Kraken2, but designed to run on smaller devices at the expense of computing power (Kim et al. 2016). For the third, I believe quality control will have a major impact on classification results.

**Materials & Methods**

Input data was sequenced from the ZymoBIOMICS™ Microbial Community DNA Standard, Catalog Nos. D6305 (200 ng) and D6306 (2000 ng) by CIAN Diagnostics at their local laboratories. This kit was accompanied by a list of the different bacteria and fungi present, as well as their percent abundance. Long-read data was collected by Oxford Nanopore Technologies (ONT) sequencing. Short-read data was collected by Illumina sequencing. All data was saved in a FASTQ format, which is the preferred input of most classification programs, including all of the ones used for these experiments. CIAN Diagnostics kindly provided access to the data, whereupon it was uploaded to my school server. I wrote SBATCH scripts for each classification program and ran them on our school server cluster.

Kraken2 was the first program planned to analyze short-read data. I followed the procedure explained in the Kraken2 GitHub Manual (Wood 2020). However, problems occurred right from the start. Kraken2 was installed through Conda, which accidently grabbed the older 2.0.7. version instead of the latest 2.1.2. version. This tripped errors on the first step, when trying to build a standard reference database from NCBI RefSeq data. However, once the program was updated, the build command still did not work. At first, this seemed to be a security issue, as the FTP function that Kraken2 uses is considered insecure by our campus firewall. But even after allowing Kraken2 a way though, the command did not work. This might have been because NCBI thought FTP was insecure too and blocked it. Only after our campus server administrator tweaked the “rsync\_from\_ncbi.pl” files to use SFTP was the reference database able to be downloaded. However, even after all that work, the final command to classify data refused to run, saying “cannot find file taxo.id” despite said file being clearly located in the indicated directory. At first I suspected this was a problem with downloading the reference database. However, reinstalling the database did not fix the problem. It seems such an error might be associated with a lack of memory, but I went back to rerun it after the RAM on our school server was increased and the same error persisted.

CCMetagen was the second program planned to analyze short-read data. I followed the procedure explained in the CCMetagen GitHub Manual (Marcelino 2022). Because the then current 1.4.0. version did not have built-in commands to install reference databases, I had to retrieve the standard database using other methods. Initially, I tried to use wget. However, I was stymied by problems accessing our campus cluster. Once this was resolved, the file host website appeared to block any attempts to download the files using wget. Thankfully, our campus server administrator was able to download the files manually through his browser and get them from his computer onto the server with WinSCP. This workaround enabled me to build the reference database with no further problems. However, upon running the command to classify my sequence data, it tripped errors saying “cannot allocate memory.” Our server administrator increased the swap space to 31 GB, but this was ineffective. I tried down-sampling the data but this did not fix the problem. My program director oversaw changes to our SLURM cluster so that it utilized 95% of node memory. She also got the server memory increased to 96 GB per node. This helped overcome the previous memory error, but resulted in a “no space left on device” error instead. The server should have plenty of space, so I’m not sure why this occurred.

Centrifuge was the third program planned to analyze short-read data. I followed the procedure explained in the Centrifuge Manual on the John Hopkins University website (Kim 2021). The 4.8.2. version was used. Like Kraken2, it had errors when trying to build the reference database. At first this was thought to be related to troubles with accessing our campus SLURM cluster, an issue that initially stalled progress with all three short-read programs. However, the errors persisted even after that was resolved. I did not find any useful help threads for the error I was getting, so I redirected my efforts towards debugging Kraken2 and CCMetagen, as they appeared to be more promising classifiers in the first place. It looked like the errors might have been related to FTP as happened to Kraken2. This would make sense since some of the same developers worked on both programs. But this is just a theory.

BugSeq was used to analyze long-read data. It is a cloud-based tool so no complicated installation procedure was necessary, just data (BugSeq 2023). The version used was listed as “latest” on their website, but should be whatever build follows their 7 November 2022 4.0 release. All I had to do was upload the data and select some options. Platform was specified as Nanopore. Device & Chemistry was set to MinION/GridION/Flongle – R10.4/10.4.1. Within 6 hours I received an email with a link to online interactive results, which become unavailable after 30 days. However, I was able to download all of the results I needed and analyze them. Although used as a classifier here, BugSeq is actually a full pipeline. It did a quality control check, assembled reads into longer contigs, and classified both contigs and raw reads using their own custom reference database.

After many failures with the short-read classifiers, it was suggested I explore whether BugSeq could also process short-reads. I was initially skeptical, as my preliminary research seemed to imply that BugSeq was only a long-read classifier. However, I emailed BugSeq and they informed me of recent updates that expanded support to short-read data. Though, contigs assembled by another source were still not permitted. And so I uploaded the short-read data to their website and had it classified too. The same version of BugSeq was used. Platform was set to Illumina. I did not have to specify Device & Chemistry. Results were emailed to me and downloaded in a similar manner as before. Notably, however, I only got classification results for the assembled contigs, not the raw short-reads. In a follow-up email, the BugSeq team sent over the missing files, but explained that their raw short-read classification procedure is still experimental and not yet optimized for general use. They stated specifically that these results should not be used for abundance estimations.

I had planned to implement FastQC as a quality control measure after I got results from non-quality controlled data. However, the quality control screening from BugSeq found both the long and short reads were already high quality. With nothing to trim, I did not further pursue this aspect of the project, and instead focused to trying to debug the short-read classifiers.

Although the only classifier that I got to work was BugSeq, I was still able to quantify comparison metrics for long, short, and contig classification. This was accomplished using the abundance estimates present in the kreport results files to create a confusion matrix. Where the estimates for a particular species overlapped with the original kit, the percentage was marked as true positive. Where the estimate of a species exceeded those in the kit, the excess was marked as false positive. Where the estimate of a species was less than those in the kit, the difference was marked as false negative, and the false positive amount was subtracted to prevent it from being counted twice. From there, precision, accuracy, F1, and F0.5 were calculated. It should be noted that the raw short-read classification data was not intended to be used for abundance estimation. In order to produce any meaningful results from it, I had to match at the genome level, as so few species were identified.

**Results**

Both the short and long read data summaries obtained from BugSeq covered a plethora of detailed information, such as plasmid detection, genotypic resistance markers, and sequence length distribution. The analysis of these details is outside the scope of my project. I instead focused on the quality control and abundance estimation portions, as these directly relate to my experimental objectives.

The quality of both long and short sequence data was quite good, as indicated by the BugSeq mean quality scores histograms illustrated in Figure 1 and Figure 2. The long-read data has a few reads at the very beginning with a low quality Phred score, but has a good quality over all. The short-read data does not have any low quality scores. If there were any major problems, BugSeq would have trimmed the samples. No such trimming appears to have occurred.

The classification results, presented as abundance estimations, were quite accurate. As shown in Figure 3, assembled long-read contigs, raw long-reads, and assembled short-read contigs all did quite well at identifying the bacterial species present. Only raw long-reads did alright with the more sparse fungal species. Raw short-read classification performed poorly across the board. It was so bad I had to use genus level classification rather than a species like the others. However, BugSeq did inform me that their raw short-read classification was experimental and should not be used for abundance estimations, as done here. It should also be noted that while assembled short-read contigs clearly performed better than their raw counterparts, assembled long-read contigs seems to have performed worse. This is exhibited in the large increase of unidentified reads from one to the other, as well as far worse accuracy with fungal species.

While the abundance estimation data is helpful to compare, there are additional metrics I planned to inspect: precision, recall, F1, and F0.5. This data is presented in Figure 4. As shown, the discrepancy between assembled long-read and raw long-read appears to be less prominent, especially when looking at F1 and F0.5. However, the raw long-read retains a higher recall, which may be more desirable under certain circumstances. The assembled short-read contigs still appear to perform much better than the raw short-reads. While not quite as good as the long-read data, it is clearly closer in caliber to the long-read results than the raw short-read ones.

**Conclusions**

Despite technical complications, I was able to gather enough data to assess my first hypothesis. As shown clearly by the BugSeq data, raw long-read data proved superior to assembled short-read contigs. Additionally, the short-read contigs were superior to the raw short-read data. Though, it must be noted that the raw short-read data gathered with BugSeq is not supposed to be used for abundance estimation as done here. These results are not particularly surprising and reflect the trends shown in several different papers (Ye et al. 2019, Tran & Phan 2020, Portik et al. 2022). Though the results support my first hypothesis, further work to confirm the discrepancy between assembled short-read contigs and raw short-read data may be valuable.

I had not initially planned to gather data on assembled long-read contigs. The use of contigs was considered to augment the classification accuracy of short-read data, as has been done in past research (Tran & Phan 2020). But BugSeq provided that information anyways. Oddly enough, performance seemed to decline with the assembled long-reads, which is the opposite of what it did for short-reads. It seems that when data is already long, the assembler introduces an extra level of uncertainty that has an impact on performance. While it has been shown that k-mer based classifiers benefit more from longer contigs, I have no reason to believe that BugSeq used a different classifier between the long-read contigs and short-read contigs (Tran & Phan 2020). Further research into long-read contig classification may be insightful, as there is not much literature surrounding it.

My second hypothesis cannot be assessed. I only got results from BugSeq. Due to technical difficulties, no results were gathered from other programs like Kraken2, Centrifuge, and CCMetagen. While it would have been interesting to compare their performance, there is no way around a complete lack of results from the other programs. Further work might shed light on which short-read classifiers are more accurate, provided one is able to get them to run properly.

It is not possible to assess my third hypothesis either. I assumed that the sequence data would have a number of low quality reads to be trimmed. This, in turn, would enable a comparison between classification with low quality reads versus higher quality. However, all of the sequence data was of a high quality. The only conclusion that can be drawn here is that Illumina and ONT sequence data is not as low quality as I had initially expected.

While it is clear that further work will be necessary to establish more definitive answers to my experimental objectives, valuable progress has been made towards this end. The results I’ve gathered point to long-read classification as being the best, followed by contigs, and then short-reads. Furthermore, BugSeq has been identified as a simple, user-friendly hybrid platform that can analyze long-read, short-read, and 16S data. It is free to use for academic research, though additional settings are available to labs that pay. Further research towards comparing classifiers would be valuable, especially now that BugSeq has recently begun to offer support for short-read classification. Although the efficacy of its long-read classification is well supported by published literature, such work has not yet been done to compare its short-read work (Fan et al. 2021, Portik et al. 2022). I believe this would be a good avenue for future research.

**Citations**

Badotti, F., de Oliveira, F. S., Garcia, C. F., Vaz, A. B. M., Fonseca, P. L. C., Nahum, L. A., Oliveira, G., & Góes-Neto, A. (2017). Effectiveness of ITS and sub-regions as DNA barcode markers for the identification of Basidiomycota (Fungi). *BMC Microbiology*, *17*(1), 42.<https://doi.org/10.1186/s12866-017-0958-x>

Billington, C., Kingsbury, J. M., & Rivas, L. (2022). Metagenomics Approaches for Improving Food Safety: A Review. *Journal of Food Protection*, *85*(3), 448–464.<https://doi.org/10.4315/JFP-21-301>

BugSeq, B. Inc. (2023, January 11). *BugSeq Manual*.<https://docs.bugseq.com/input/>

Carbo, E. C., Sidorov, I. A., Zevenhoven-Dobbe, J. C., Snijder, E. J., Claas, E. C., Laros, J. F. J., Kroes, A. C. M., & de Vries, J. J. C. (2020). Coronavirus discovery by metagenomic sequencing: A tool for pandemic preparedness. *Journal of Clinical Virology*, *131*, 104594.<https://doi.org/10.1016/j.jcv.2020.104594>

Edgar, R. C. (2018). Updating the 97% identity threshold for 16S ribosomal RNA OTUs. *Bioinformatics*, *34*(14), 2371–2375.<https://doi.org/10.1093/bioinformatics/bty113>

Fan, J., Huang, S., & Chorlton, S. D. (2021). BugSeq: A highly accurate cloud platform for long-read metagenomic analyses. *BMC Bioinformatics*, *22*(1), 160.<https://doi.org/10.1186/s12859-021-04089-5>

Govender, K. N., Street, T. L., Sanderson, N. D., & Eyre, D. W. (2021). Metagenomic Sequencing as a Pathogen-Agnostic Clinical Diagnostic Tool for Infectious Diseases: A Systematic Review and Meta-analysis of Diagnostic Test Accuracy Studies. *Journal of Clinical Microbiology*, *59*(9), e02916-20.<https://doi.org/10.1128/JCM.02916-20>

Huang, X. (1992). A contig assembly program based on sensitive detection of fragment overlaps. *Genomics, 14*(1), 18–25. <https://doi.org/10.1016/S0888-7543(05)80277-0>

Jovel, J., Patterson, J., Wang, W., Hotte, N., O’Keefe, S., Mitchel, T., Perry, T., Kao, D., Mason, A. L., Madsen, K. L., & Wong, G. K.-S. (2016). Characterization of the Gut Microbiome Using 16S or Shotgun Metagenomics. *Frontiers in Microbiology, 7*. <https://doi.org/10.3389/fmicb.2016.00459>

Kim, D. (2021). *Centrifuge Manual*.<https://ccb.jhu.edu/software/centrifuge/manual.shtml>

Kim, D., Song, L., Breitwieser, F. P., & Salzberg, S. L. (2016). Centrifuge: Rapid and sensitive classification of metagenomic sequences. *Genome Research*, *26*(12), 1721–1729.<https://doi.org/10.1101/gr.210641.116>

Marcelino, V. R. (2023). *CCMetagen Manual* [HTML].<https://github.com/vrmarcelino/CCMetagen> (Original work published 2018)

Marcelino, V. R., Clausen, P. T. L. C., Buchmann, J. P., Wille, M., Iredell, J. R., Meyer, W., Lund, O., Sorrell, T. C., & Holmes, E. C. (2020). CCMetagen: Comprehensive and accurate identification of eukaryotes and prokaryotes in metagenomic data. *Genome Biology*, *21*(1), 103.<https://doi.org/10.1186/s13059-020-02014-2>

Pearman, W. S., Freed, N. E., & Silander, O. K. (2020). Testing the advantages and disadvantages of short- and long- read eukaryotic metagenomics using simulated reads. *BMC Bioinformatics, 21*(1), 220. <https://doi.org/10.1186/s12859-020-3528-4>

Portik, D. M., Brown, C. T., & Pierce-Ward, N. T. (2022). Evaluation of taxonomic classification and profiling methods for long-read shotgun metagenomic sequencing datasets. *BMC Bioinformatics*, *23*(1), 541.<https://doi.org/10.1186/s12859-022-05103-0>

Sauvage, V., & Eloit, M. (2016). Viral metagenomics and blood safety. *Transfusion Clinique et Biologique*, *23*(1), 28–38.<https://doi.org/10.1016/j.tracli.2015.12.002>

Tran, Q., & Phan, V. (2020). Assembling Reads Improves Taxonomic Classification of Species. *Genes*, *11*(8), 946.<https://doi.org/10.3390/genes11080946>

Wang, W.-L., Xu, S.-Y., Ren, Z.-G., Tao, L., Jiang, J.-W., & Zheng, S.-S. (2015). Application of metagenomics in the human gut microbiome. *World Journal of Gastroenterology, 21*(3), 803–814. <https://doi.org/10.3748/wjg.v21.i3.803>

Wood, D. E. (2020). *Kraken 2 Manual*. GitHub.<https://github.com/DerrickWood/kraken2/wiki/Manual>

Wood, D. E., & Salzberg, S. L. (2014). Kraken: Ultrafast metagenomic sequence classification using exact alignments. *Genome Biology*, *15*(3), R46.<https://doi.org/10.1186/gb-2014-15-3-r46>

Ye, S. H., Siddle, K. J., Park, D. J., & Sabeti, P. C. (2019). Benchmarking Metagenomics Tools for Taxonomic Classification. *Cell*, *178*(4), 779–794.<https://doi.org/10.1016/j.cell.2019.07.010>

**Figures & Tables**

**Chart

Description automatically generated**

**Figure 1:** Long-read mean quality scores histogram, generated by BugSeq. There are a small number of low quality reads at the very beginning, but not enough to warrant BugSeq trimming it. All of the rest of the reads are high quality.

**Chart

Description automatically generated**

**Figure 2:** Short-read mean quality scores histogram, generated by BugSeq. All of the scores are very high. There are two lines because it is paired end data.

Table

Description automatically generated

**Figure 3:** BuqSeq species abundance estimations for assembled long-reads, raw long-reads, assembled short-reads, and raw short-reads. Actual is the true abundance of each species in the sample. Unidentified means BugSeq was unable to classify. Other represents misidentifications or anything not classified on the species level. Raw short-reads are only classified on the genus level due to poor performance. Note that raw short-read classification is still experimental within BugSeq and they specifically stated its results should not be used to abundance estimations.

Table

Description automatically generated

**Figure 4:** Comparison metrics for the BugSeq abundance estimations shown in Figure 3. This includes measurements of precision, recall, F1, and F0.5 for the same categories.