# The Oyster River Protocol: A Multi Assembler and Kmer Approach For de novo Transcriptome Assembly

Matthew D. MacManes[1, *, •, ⋆]

[1] Department of Molecular, Cellular and Biomedical Sciences, University of New Hampshire, Durham NH, USA

[*] E-mail: macmanes@gmail.com

[•] Twitter: @MacManes

[⋆] Mailing Address: 46 College Road, 189 Rudman Hall. Durham NH 03824

# Abstract

Characterizing transcriptomes in non-model organisms has resulted in a massive increase in our understanding of biological phenomena. This boon, largely made possible via high-throughput sequencing, means that studies of functional, evolutionary and population genomics are now being done by hundreds or even thousands of labs around the world. For many, these studies begin with a de novo transcriptome assembly, which is a technically complicated process involving several discrete steps. The Oyster River Protocol (ORP), described here, implements a standardized and benchmarked set of bioinformatic processes, resulting in an assembly with enhanced qualities over other standard assembly methods. Specifically, ORP produced assemblies have higher `Detonate` and `TransRate` scores and mapping rates, which is largely a product of the fact that it leverages a multi-assembler and kmer assembly process, thereby bypassing the shortcomings of any one approach. These improvements are important, as previously unassembled transcripts are included in ORP assemblies, resulting in a significant enhancement of the power of downstream analysis. Further, as part of this study, I show that assembly quality is unrelated ~~to taxonomy, nor is it related to~~ with the number of reads generated, above 30 million reads. Code Availability: The version controlled open-source code is available at `https://github.com/macmanes-lab/Oyster_River_Protocol`. Instructions for software installation and use, and other details are available at `http://oyster-river-protocol.rtfd.org/`.


# Competing Interests

The author declares no competing interests.

# 1 Introduction

For all biology, modern sequencing technologies ~~has~~ have provided for an unprecedented opportunity to gain a deep understanding of genome level processes that underlie a very wide array of natural phenomena, from intracellular metabolic processes to global patterns of population variability. Transcriptome sequencing has been influential (1; 2), particularly in functional genomics (3; 4), and has resulted in discoveries not possible even just a few years ago. This in large part is due to the scale at which these studies may be conducted (5; 6). Unlike studies of adaptation based on one or a small number of candidate genes (e.g., (7; 8)), modern studies may assay the entire suite of expressed transcripts – the transcriptome – simultaneously. In addition to issues of scale, as a direct result of enhanced dynamic range, newer sequencing studies have increased

ability to simultaneously reconstruct and quantitate lowly- and highly-expressed transcripts ~~,~~(9; 10). Lastly, improved methods for the detection of differences in gene expression (e.g., (11; 12)) across experimental treatments ~~has~~ have resulted in increased resolution for studies aimed at understanding changes in gene expression.

As a direct result of their widespread popularity, a diverse toolset for the assembly of transcriptome ~~exist~~exists, with each potentially reconstructing transcripts others fail to reconstruct. Amongst the earliest of specialized de novo transcriptome assemblers were the packages `Trans-ABySS` (13), `Oases` (14), and `SOAPdenovoTrans` (15), which were fundamentally based on the popular de Bruijn graph-based genome assemblers `ABySS` (16), `Velvet` (17), and `SOAP` (18) ~~,~~respectively. These early efforts gave rise to a series of more specialized de novo transcriptome assemblers, namely `Trinity` (19), and `IDBA-Tran` (20). While the de Bruijn graph approach remains powerful, newly developed software explores novel parts of the algorithmic landscape, offering substantial benefits, assuming novel methods reconstruct different fractions of the transcriptome. `BinPacker` (21), for instance, abandons the de Bruijn graph approach to model the assembly problem after the classical bin packing problem, while `Shannon` (22) uses information theory, rather than a set of software engineer-decided heuristics. These newer assemblers, by implementing fundamentally different assembly algorithms, may reconstruct fractions of the transcriptome that other assemblers fail to accurately assemble.

In addition to the variety of tools available for the de novo assembly of transcripts, several tools are available for pre-processing of reads via read trimming ((e.g., `Skewer` (23), `Trimmomatic` (24), `Cutadapt` (25)), read normalization (`khmer` (26)), and read error correction (`SEECER` (27) and ~~RCorrector~~ `RCorrector` (28), `Reptile` (29))~~, and assembly verification (~~. Similarly, benchmarking tools that evaluate the quality of assembled transcriptomes including `TransRate` (30)~~)~~, `BUSCO` (Benchmarking Universal Single-Copy Orthologs - (31)), and ~~`RSEM-eval`~~`Detonate` (32) ~~).~~ have been developed. Despite the development of these evaluative tools, this manuscript describes the first systematic effort coupling them with the development of a de novo transcriptome assembly pipeline.

The ease with which these tools may be used to produce and characterize transcriptome assemblies belies the true complexity underlying the overall process (33; 34; 35; 36). Indeed, the subtle (and not so subtle) methodological challenges associated with transcriptome reconstruction may result in highly variable assembly quality. ~~Production of an accurate~~ In particular, while most tools run using default settings, these defaults may be sensible only for one specific (often unspecified) use case or data type. Because parameter optimization is both dataset-dependent and factorial in nature, an exhaustive optimization particularly of entire pipelines, is never possible. Given this, the production of a de novo transcriptome assembly requires a

large investment in time and resources~~. Each step in it's production requires~~ , with each step requiring careful consideration. Here, I propose an evidence-based protocol for assembly that results in the production of ~~the~~ high quality transcriptome assemblies, across a variety of commonplace experimental conditions or taxonomic groups.

This manuscript describes the development of ~~a multi-assembler and~~ The Oyster River Protocol[1] for transcriptome assembly. It explicitly considers and attempts to address many of the shortcomings described in (10), by leveraging a multi-kmer ~~protocol~~and multi-assembler strategy. This innovation is critical, as all assembly solutions treat the sequence read data in ways that bias transcript recovery. Specifically, with the development of assembly software comes the use of a set of heuristics ~~,~~that are necessary given the scope of the assembly problem itself. Given each software development team carries with it a unique set of ideas related to these heuristics while implementing various assembly algorithms, individual assemblers exhibit unique assembly behavior. By leveraging a multi-assembler approach, the strengths of one assembler may complement the weaknesses of another. In addition to biases related to assembly heuristics, it is well known that assembly kmer-length has important effects on transcript reconstruction, with shorter kmers more efficiently reconstructing lower-abundance transcripts relative to ~~longer assembly kmer-lengths~~more highly abundant transcripts. Given this, assembling with multiple different kmer lengths, then merging the resultant assemblies may effectively reduce this type of bias. Recognizing these issue, I hypothesize that an assembly that ~~resulted~~ results from the combination of multiple different assemblers and lengths of assembly-kmers ~~would~~ will be better than each individual assembly, across a variety of metrics.

In addition to developing an enhanced pipeline, the work suggests an exhaustive way of characterizing assemblies while making available a set of fully-benchmarked reference assemblies that may be used by other researchers in developing new assembly algorithms and pipelines. Although many other researchers have published comparisons of assembly methods, up until now these have been limited to single datasets assembled a few different ways (37; 38), thereby failing to provide more general insights.

[1] Named the Oyster River Protocol because the ideas, and some of the code, was developed while overlooking the Oyster River, located in Durham, New Hampshire. NB, the naming assembly of protocols after bodies of water was, to the best of my knowledge, first done by C. Titus Brown (The Eel Pond Protocol: `http://khmer-protocols.readthedocs.io/en/latest/mrnaseq/index.html`), and may have subconsciously influenced me in naming this protocol.

# 2 Methods

## 2.1 Datasets

In an effort at benchmarking the assembly and merging protocols, I downloaded a set of publicly available RNAseq datasets (Table 1) that had been produced on the Illumina sequencing platform. These datasets were chosen to represent a variety of taxonomic groups, so as to demonstrate the broad utility of the developed methods. Because datasets were selected randomly with respect to sequencing center and read number, they are likely to represent the typical quality of Illumina data circa 2014-2017.

Table 1

| Type | Accession | Species | Num. Reads | Read Length |
|---|---|---|---|---|
| Animalia | ERR489297 | Anopheles gambiae | 206M | 100bp |
| Animalia | DRR030368 | Echinococcus multilocularis | 73M | 100bp |
| Animalia | ERR1016675 | Heterorhabditis indica | 51M | 100bp |
| Animalia | SRR2086412 | Mus musculus | 54M | 100bp |
| Animalia | DRR036858 | Mus musculus | 114M | 100bp |
| Animalia | DRR046632 | Oncorhynchus mykiss | 82M | 76bp |
| Animalia | SRR1789336 | Oryctolagus cuniculus | 31M | 100bp |
| Animalia | SRR2016923 | Phyllodoce medipapillata | 86M | 100bp |
| Animalia | ERR1674585 | Schistosoma mansoni | 39M | 100bp |
| Plant | DRR082659 | Aeginetia indica | 69M | 90bp |
| Plant | DRR053698 | Cephalotus follicularis | 126M | 90bp |
| Plant | DRR069093 | Hevea brasiliensis | 103M | 100bp |
| Plant | SRR3499127 | Nicotiana tabacum | 30M | 150bp |
| Plant | DRR031870 | Vigna angularis | 60M | 100bp |
| Protozoa | ERR058009 | Entamoeba histolytica | 68M | 100bp |

Table 1 lists the datasets used in this study. All datasets are publicly available for download by accession number at the European Nucleotide Archive or NCBI Short Read Archive.

## 2.2 Software

The Oyster River Protocol ~~is~~ can be installed on the Linux platform, and does not require superuser privileges, assuming `Linuxbrew` (39) is installed. The software is implemented as a stand-alone makefile

which coordinates all steps described below. All scripts are available at `https://github.com/macmanes-lab/Oyster_River_Protocol`, and run on the Linux platform. The software is version controlled and openly-licensed to promote sharing and reuse. A guide for users is available at `http://oyster-river-protocol.rtfd.io`.

## 2.3 Pre-assembly procedures

For all assemblies performed, Illumina sequencing adapters were removed from both ends of the sequencing reads, as were nucleotides with quality Phred $\leq$ ~~3~~2, using the program `Trimmomatic` version 0.36 (24), following the recommendations from (40). After trimming, reads were error corrected using the software `RCorrector` version 1.0.2 (28), following recommendations from (41). The code for running this step of the Oyster River protocols is available at `https://github.com/macmanes-lab/Oyster_River_Protocol/blob/master/oyster.mk#L134`. The trimmed and error corrected reads ~~where~~ were then subjected to de novo assembly.

## 2.4 Assembly

I assembled each trimmed and error corrected dataset using three different de novo transcriptome assemblers and three different kmer lengths, producing 4 unique assemblies. First, I assembled the reads using `Trinity` release 2.4.0 (19), and default settings (k=25), without read normalization. The decision to forgo normalization is based on previous work (42) showing slightly worse performance of normalized datasets. Next, the `SPAdes` RNAseq assembler (version 3.10) (43) was used, in two distinct runs, using kmer sizes 55 and 75. Lastly, reads were assembled using the assembler `Shannon` version 0.0.2 (22), using a kmer length of 75. These assemblers were chosen based on the fact that ~~(~~they [1~~) use an open-development~~ ] use an open-science development model, whereby end-users ~~man~~ may contribute code, ~~(~~[2~~) they~~ ] are all actively maintained and are undergoing continuous development, and ~~(~~[3~~)~~ ] occupy different parts of the algorithmic landscape.

This assembly process resulted in the production of four distinct assemblies. The code for running this step of the Oyster River protocols is available at `https://github.com/macmanes-lab/Oyster_River_Protocol/blob/master/oyster.mk#L142`.

## 2.5 Assembly Merging via OrthoFuse

To merge the four assemblies produced as part of the Oyster River Protocol, I developed new software that effectively merges transcriptome assemblies. Described in brief, `OrthoFuse` begins by concatenating all assemblies together, then forms groups of transcripts by running a version of `OrthoFinder` (44) packaged with the ORP, modified to accept nucleotide sequences from the merged assembly. These groupings represent groups of homologous transcripts. ~~Of note,~~ While isoform reconstruction using short-read data is notoriously poor, by increasing the inflation parameter ~~has been increase~~ by default to I=4, it attempts to prevent the collapsing of transcript isoforms into ~~a~~ single groups. After `Orthofinder` has completed, a modified version of `TransRate` version 1.0.3 (30) which is packaged with the ORP, is run on the merged assembly, after which the best (= highest contig score) transcript is selected from each group and placed in a new assembly file to represent the entire group. The resultant file, which contains the highest scoring contig for each orthogroup, may be used for all downstream analyses. `OrthoFuse` is run automatically as part of the Oyster River Protocol, and additionally is available as a stand ~~along~~ alone script, `https://github.com/macmanes-lab/Oyster_River_Protocol/blob/master/orthofuser.mk`.

## 2.6 Assembly Evaluation

All assemblies were evaluated using `ORP-TransRate`, `Detonate` version 1.11 (45), `shmlast` version 1.2 ~~(46)~~ (46), and `BUSCO` version 3.0.2 ~~(31)~~(31). `TransRate` evaluates transcriptome assembly contiguity by producing a score based on ~~contig~~ length-based and mapping metrics, while `Detonate` conducts an orthogonal analysis, producing a score that is maximized by an assembly that is representative of input sequence read data. `BUSCO` evaluates assembly content by searching the ~~assembly~~ assemblies for conserved single copy orthologs found in all Eukaryotes. We report default `BUSCO` metrics as described in (31). Specifically, "complete orthologs", are defined as query transcripts that are within 2 standard deviations of the length of the `BUSCO` group mean, while contigs falling short of this metric are listed as "fragmented". `Shmlast` implements the conditional reciprocal best hits (CRBH) test ~~(47)~~(47), conducted in this case against the Swiss-Prot protein database (downloaded October, 2017) using an e-value of 1E-10.

In addition to the generation of metrics to evaluation the quality of transcriptome assemblies, I generated a distance matrix of assemblies for each dataset using the `sourmash` package (48), in an attempt at characterizing the algorithmic landscape of assemblers. Specifically, each assembly was characterized using the `compute` function using 5000 independent sketches. The distance between assemblies was calculated using the `compare` function and a kmer length of 51. These distance matrices were visualized

using the `isoMDS` function of the MASS package (`https://CRAN.R-project.org/package=MASS`).

### 2.7 Statistics

All ~~statistics~~ statistical analyses were conducted in R version 3.4.0 (49). Violin plots were constructed using the beanplot (50) and the beeswarm R packages (`https://CRAN.R-project.org/package=beeswarm`). Expression distributions were plotted using the ggjoy package (`https://CRAN.R-project.org/package=ggjoy`). ~~Plots for visualizing the unique content of each assembly were constructed using the UpsetR package (51).~~

## 3 Results and Discussion

Fifteen RNAseq datasets, ranging in size from (30-206M paired end reads) were assembled using the Oyster River Protocol and with `Trinity`. Each assembly was evaluated using the software `BUSCO`, `shmlast`, `Detonate`, and `TransRate`. From these, ~~seven~~ several metrics were chosen to represent the quality of the produced assemblies. Of note, all the assemblies produced as part of this work are available at `https://www.dropbox.com/sh/ehxvd0ont9ge8id/AABZxRCwcpaxb7rXWclTBbJga`, and will be moved to dataDryad after acceptance. A file containing the evaluative metrics is available at `https://github.com/macmanes-lab/Oyster_River_Protocol/blob/master/manuscript/orp.csv`, while the distance matrices are available within the folder `https://github.com/macmanes-lab/Oyster_River_Protocol/blob/master/manuscript/`. R code used to conduct analyses and make figures is found at `https://github.com/macmanes-lab/Oyster_River_Protocol/blob/master/manuscript/R-analysis.Rmd`.

### 3.1 ~~Trinity-assembled transcripts~~Assembled transcriptomes

The `Trinity` ~~assemblies~~ assembly of trimmed and error corrected reads generally completed on ~~standard~~ a standard Linux server using 24 cores, in less than 24 hours. RAM requirement is estimated to be close to 0.5Gb per million paired-end reads. The assemblies on average contained 176k transcripts (range 19k - 643k) and 97Mb (range 14MB - 198Mb). Other quality metrics will be discussed below, specifically in relation to the ORP produced assemblies.

## 3.2 ~~Oyster River Protocol- assembled transcripts~~

ORP assemblies generally completed on ~~standard~~ a standard Linux server using 24 cores in three days. Typically `Trinity` was the longest running assembler, with the individual `SPAdes` assemblies being the shortest. RAM requirement is estimated to be 1.5Gb - 2Gb per million paired-end reads, with `SPAdes` requiring the most. The assemblies on average contained 153k transcripts (range 23k - 625k) and 64Mb (range 8MB - 181Mb).

The distance between assemblies of a given dataset were calculated using `sourmash`, and a MDS plot was generated (Figure 1). Interestingly, each assembler tends to produce a specific signature which is relatively consistent between the fifteen datasets. `Shannon` differentiates itself from the other assemblers on the first (x) MDS axis, while the other assemblers (`SPAdes` and `Trinity`) are separated on the second (y) MDS axis.

Figure 1

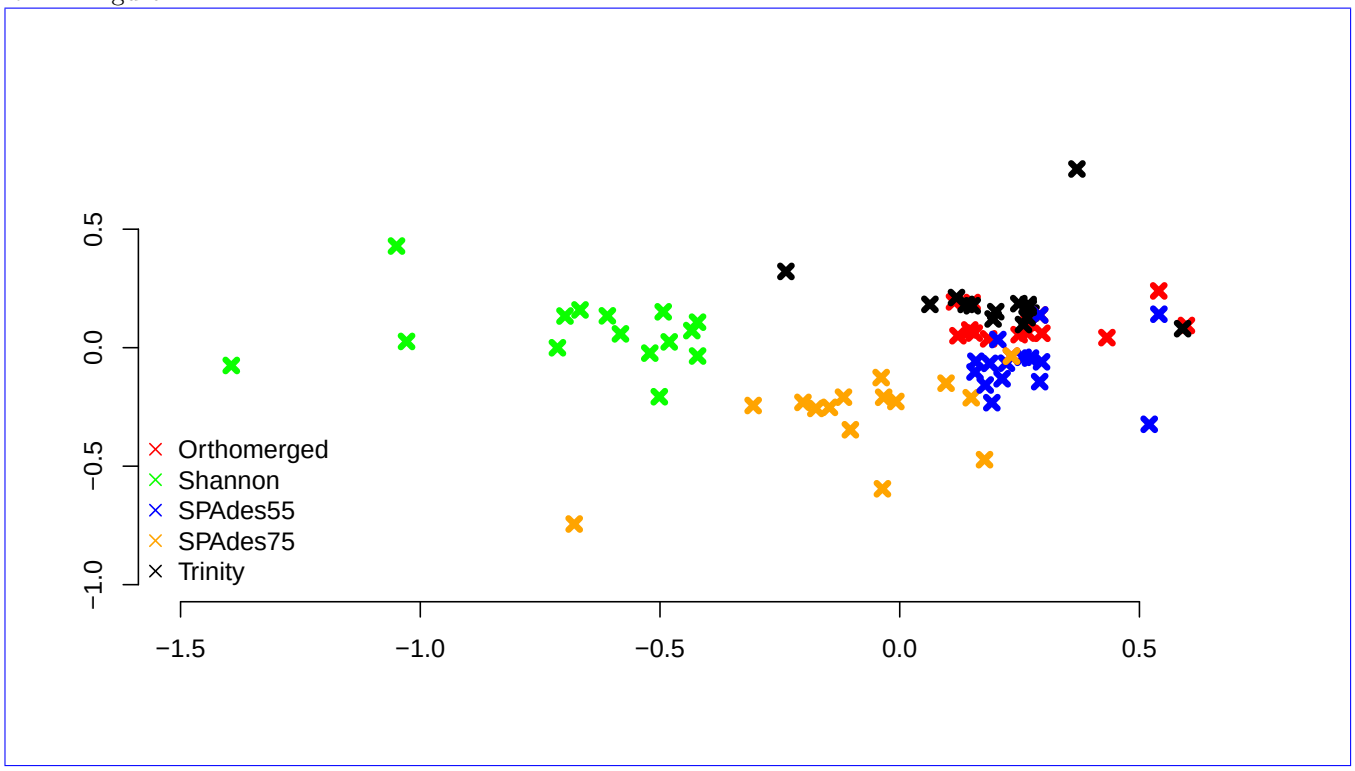


Figure 1. MDS plot describing the similarity within and between assemblers. Colored x's mark individual assemblies, with red marks corresponding to the ORP assemblies, green marks corresponding to the `Shannon` assemblies, blue marks corresponding to the `SPAdes55` assemblies, orange marks corresponding to the `SPAdes75` assemblies, and the black marks corresponding to the `Trinity` assemblies. In general assemblies produced by a given assembler tend to cluster together.

### 3.1.1 Assembly Structure

The structural integrity of each assembly was evaluated using the `TransRate` ~~software package. Using mapping metrics, I evaluated each of the `Trinity` and ORP produced assemblies (Figure 1).~~ and `Detonate` software packages. As many downstream ~~application~~ applications depend critically on accurate read mapping, ~~assemblies that maximize this metric are desirable~~assembly quality is correlated with increased mapping rates. The split violin plot presented in figure ~~1A visually represent~~ 2A visually represents the mapping rates of each assembly, with lines connecting the mapping rates of datasets assembled with `Trinity` and with the ORP, respectively. The average mapping rate of the `Trinity` assembled datasets was 87% (sd = 8%), while the average mapping rates of the ORP assembled datasets was 93% (sd=4%). This test is statistically significant (~~Two-sided~~ one-sided Wilcoxon rank sum test, p = 2E-2). ~~Figure 1B~~ Mapping rates of the other assemblies are less than that of the ORP assembly, but in most cases, greater than that of

the Trinity assembly. This aspect of assembly quality is critical. Specifically mapping rates measure how representative the assembly is of the reads. If we assume that the vast majority of generated reads come from the biological sample under study, when reads fail to map, that fraction of the biology is lost from all downstream analysis and inference. This study demonstrates that across a wide variety of taxa, assembling RNAseq reads with any single assembler alone may result in a decrease in mapping rate and in turn, the lost ability to draw conclusions from that fraction of the sample.

Figure 2B describes the distribution of `TransRate` assembly scores, which is a synthetic metric taking into account ~~multiple~~ the quality of read mapping and coverage-based statistics. The `Trinity` assemblies had an average optimal score of 0.35 (sd = .14), while the ORP assembled datasets had an average score of 0.46 (sd = .07). This test is statistically significant (~~One sided~~ one-sided Wilcoxon rank sum test, p-value = 1.8E-2). ~~Lastly, figure 1C~~ Optimal scores of the other assemblies are less than that of the ORP assembly, but in most cases, greater than that of the `Trinity` assembly. Figure 2C describes the distribution of ~~Detonate~~ `Detonate` scores. The `Trinity` assemblies had an average score of -6.9E9 (sd = 5.2E9), while the ORP assembled datasets had an average score of -5.3E9 (sd = 3.5E9). This test not is statistically significant, though in all cases, ~~scores~~ relative to all other assemblies, scores of the ORP assemblies are improved (become less negative), indicating that the ORP produced assemblies of higher quality.

In addition to reporting synthetic metrics related to assembly structure, `TransRate` reports individual metrics related to specific elements of assembly quality. One such metric estimates the rate of chimerism, a phenomenon which is known to be problematic in de novo assembly (33; 52). Rates of chimerism are relatively constant between all assemblers, ranging from 10% for the `Shannon` assembly, to 12% for the `SPAdes75` assembly. The chimerism rate for the ORP assemblies averaged 10.5% ($\pm$ 4.7%). While the new method would ideally improve this metric by exclusively selecting non-chimeric transcripts, this does not seem to be the case, and may be related to the inherent shortcomings of short-read transcriptome assembly.

Of note, consistent with all short-read assemblers (33), the ORP assemblies may not accurately reflect the true isoform complexity. Specifically, because of the way that single representative transcripts are chosen from a cluster of related sequences, some transcriptional complexity may be lost. Consider the cluster containing contigs {AB, A, B} where AB is a false-chimera, selecting a single representative transcript with the best score could yield either A or B, thereby excluding an important transcript in the final output. We believe this type of transcript loss is not common, based on how contigs are scored (Table 1, Figure 3, (30)), though strict demonstration of this is not possible, given the lack of high-quality reference genomes for the majority of the datasets. More generally, mapping rates, `Detonate` and `TransRate` score improvements suggest that this type of loss is not widespread.

Figure 2

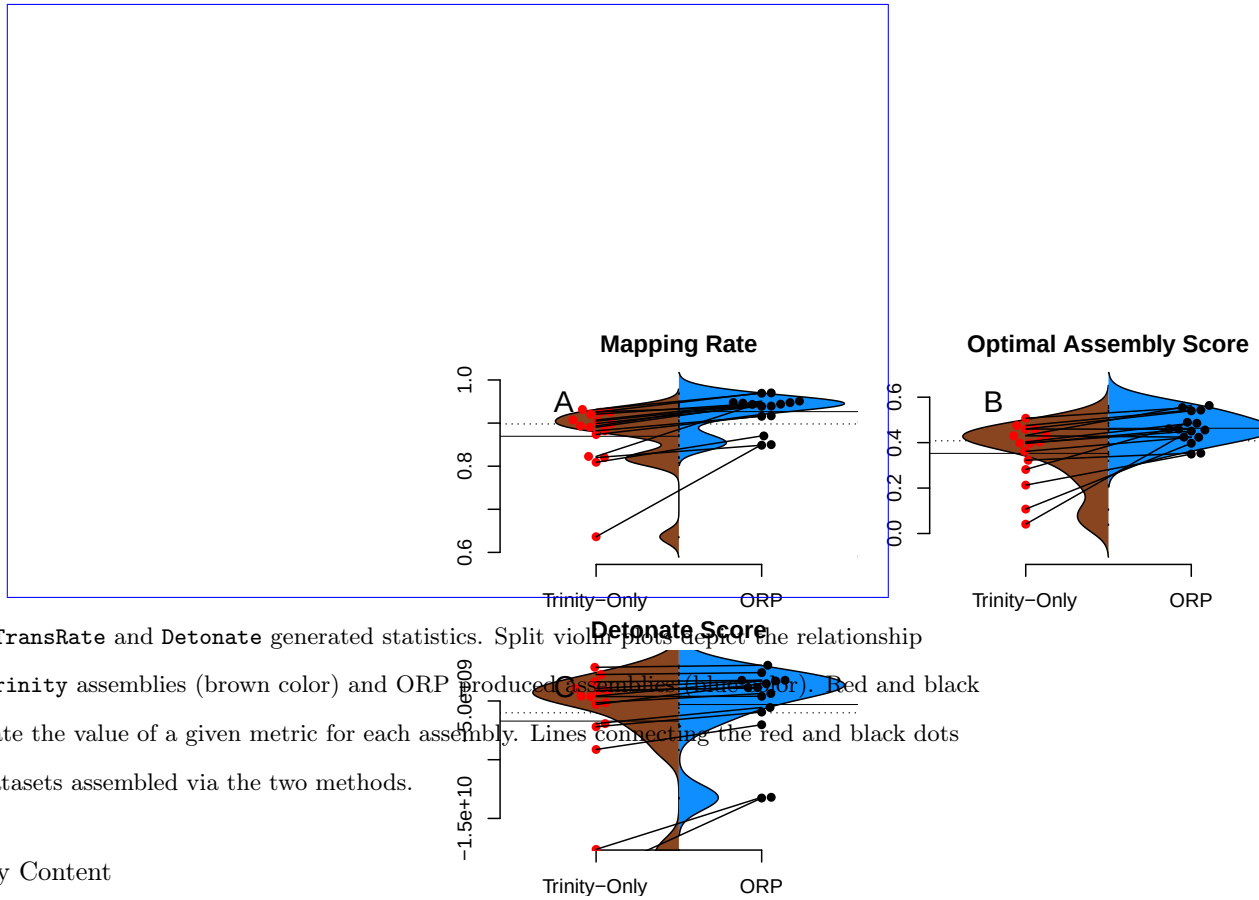


Figure 2. `TransRate` and `Detonate` generated statistics. Split violin plots depict the relationship between `Trinity` assemblies (brown color) and ORP produced assemblies (blue color). Red and black dots indicate the value of a given metric for each assembly. Lines connecting the red and black dots connect datasets assembled via the two methods.

### 3.1.2 Assembly Content

The genic content of assemblies was measured using the software package `Shmlast`~~using the~~ , which implements the conditional reciprocal blast test against the Swiss-prot database~~, and `BUSCO` using the Eukaryota database. Depicted in Figure 2A, `Trinity`~~ . Presented in Table 2 and in Figure 3A, ORP assemblies recovered on average ~~12722~~ 13364 (sd=~~3195) reciprocal~~ 3391) blast hits, while ~~the ORP assemblies recovered 13363 (sd~~all other assemblies recovered fewer (minimum `Shannon`, mean=~~3391). While this result is not significantly different, OPR assemblies, in every case ,~~ 10299). In every case across all assemblers, the ORP assembler retained more reciprocal blast hits~~. regarding~~ , though only the comparison between the ORP assembly and `Shannon` was significant (one-sided Wilcoxon rank sum test, p = 4E-3). Notably, in all cases, each assembler was both missing transcripts contained in other assemblies, and contributed unique transcripts to the final merged assembly (Table 2), highlighting the utility of using multiple assemblers.

Table 2

| Assembly | Genes | Delta | Unique |
|---|---|---|---|
| Concatenated | 14674 ± 3590 | | |
| SPAdes55 | | −1739 ± 758 | 570 ± 266 |
| SPAdes75 | | −2711 ± 2047 | 301 ± 195 |
| Shannon | | −4375 ± 3508 | 302 ± 241 |
| Trinity | | −1952 ± 803 | 520 ± 301 |

Table 2 describes the number of genes contained in the assemblies, with the row labelled concatenated representing the combined average (± standard deviation) number of genes contained in all assemblies of a given dataset. The other rows contain information about each assembly. The column labelled delta contains the average number (± standard deviation) of genes missing, relative to the concatenated number. The unique column contains the average number of genes (± standard deviation) unique to that assembly.

Regarding `BUSCO` scores, `Trinity` assemblies contained on average 86% (sd = 21%) of the full-length orthologs as defined by the `BUSCO` developers, while the ORP assembled datasets contained on average 86% (sd = 13%) of the full length transcripts. Other assemblers contained fewer full-length orthologs. The `Trinity` and ORP assemblies were missing, on average 4.5% (sd = 8.7%) of orthologs. The `Trinity` assembled datasets contained 9.5% (sd = 17%) of fragmented transcripts while the ORP assemblies each contained on average 9.4% (sd = 9%) of fragmented orthologs. The other assemblers in all cases contained more fragmentation. The rate of transcript duplication, depicted in figure ~~2B~~ 3B is 47% (sd = 20%) for `Trinity` assemblies, and 34% (sd = 15%) for ORP assemblies. This result is statistically significant (~~Two~~ One sided Wilcoxon rank sum test, p-value = ~~0.05).~~ 0.02). Of note, all other assemblers produce less transcript duplication than does the ORP assembly, but none of these differences arise to the level of statistical significance.

While the majority of the `BUSCO` metrics were unchanged, the number of orthologs recovered in duplicate (>1 copy), was decreased when using the ORP. This difference is important, given that the relative frequency of transcript duplication may have important implications for downstream abundance estimation, with less duplication potentially resulting in more accurate estimation. Although gene expression quantitation software (53; 54) probabilistically assigns reads to transcripts in an attempt at mitigating this issue, a primary solution related to decreasing artificial transcript duplication could offer significant advantages.

Figure 3

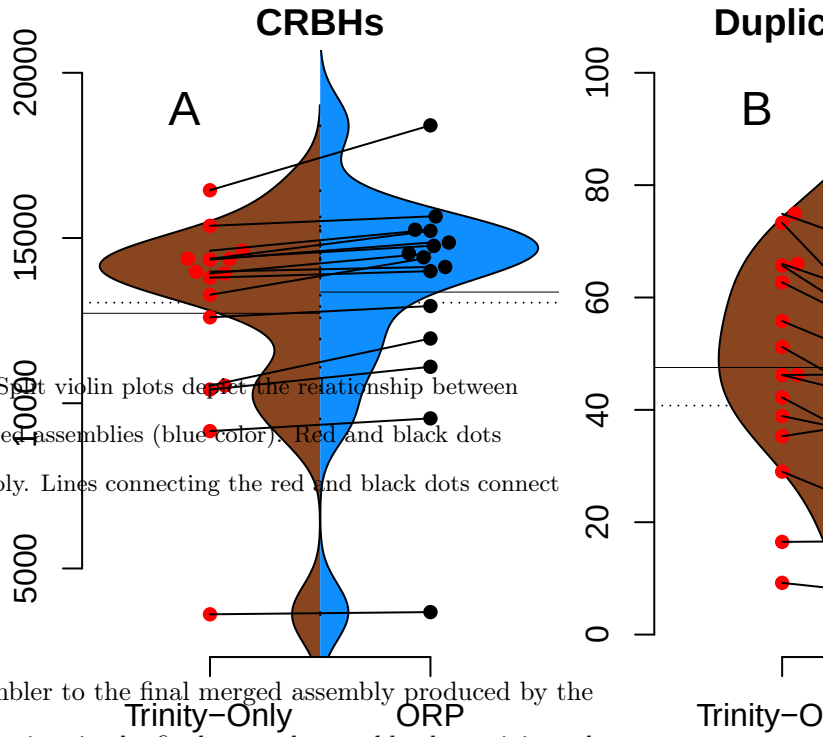


Figure 3. `Shmlast` and `BUSCO` generated statistics. Split violin plots depict the relationship between `Trinity` assemblies (brown color) and ORP produced assemblies (blue color). Red and black dots indicate the value of a given metric for each assembly. Lines connecting the red and black dots connect datasets assembled via the two methods.

### 3.1.3 Assembler Contributions

To understand the relative contribution of each assembler to the final merged assembly produced by the Oyster River Protocol, I counted the number of transcripts in the final merged assembly that originated from a given assembler (Figure 4). On average, 36% of transcripts in the merged assembly were produced by the `Trinity` assembler. 16% were produced by `Shannon`. `SPAdes` run with a kmer value of length=55 produced 28% of transcripts, while `SPAdes` run with a kmer value of length=75 produced 20% of transcripts

Figure 4

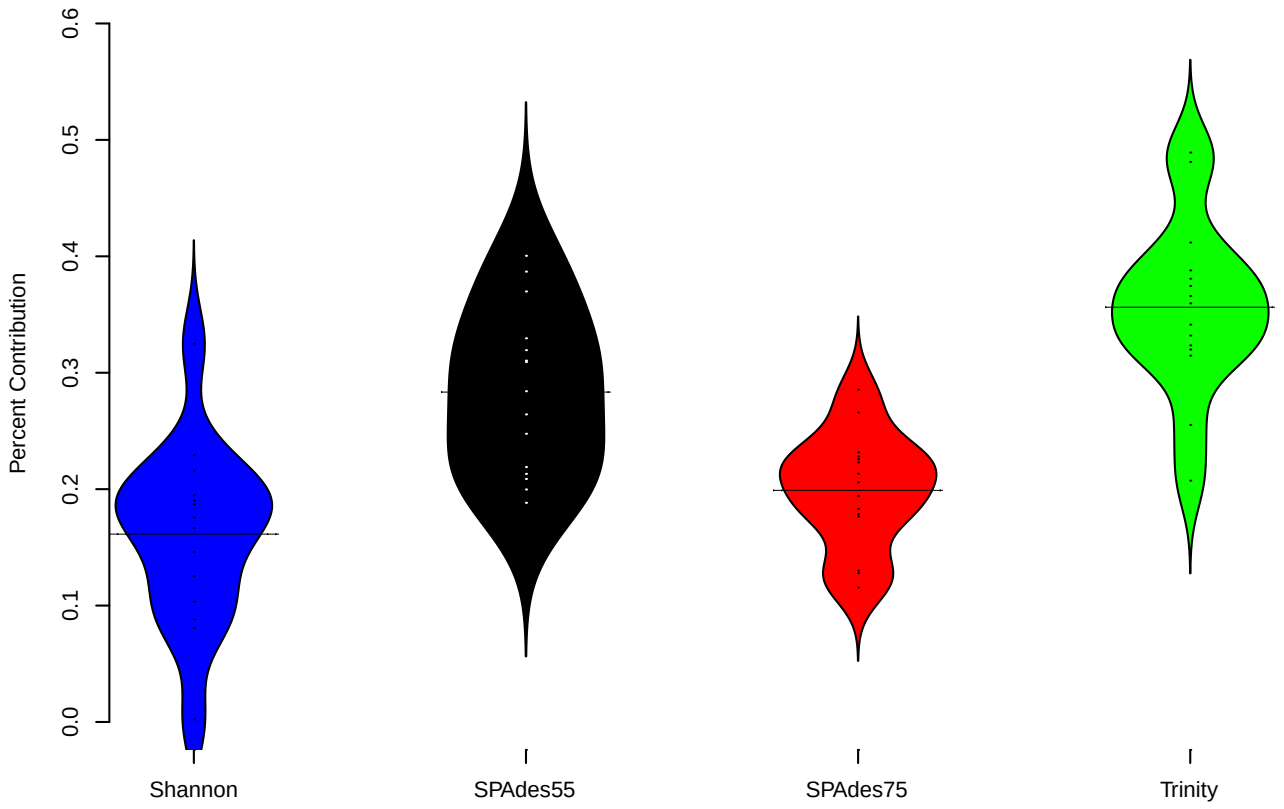


Figure 4 describes the percent contribution of each assembler to the final ORP assembly.

To further understand the potential biases intrinsic to each assembler, I plotted the distribution of gene expression estimates for each merged assembly, broken down by the assembler of origin (Figure 45, depicting four randomly selected representative assemblies). As is evident, most transcripts are lowly expressed, with `SPAdes` and `Trinity` both doing a sufficient job in reconstructing these transcripts. Of note, the `SPAdes` assemblies using kmer-length=75 is biased, as expected, towards more highly expressed transcripts relative to kmer-length 55 assemblies. `Shannon` demonstrates a unique profile, consisting of, almost exclusively high-expression transcripts, ~~given~~ showing a previously undescribed bias against low-abundance transcripts.

~~Lastly, though the same read data were assembled, each assembler reconstructed unique transcripts. Using the dataset DRR069093 as an example, across the four different assemblies, a sum of 276852~~

SwissProt entries were matched. Of these 86% were recovered in all four assemblies. The `SPAdes` assembly using a kmer value of 55 recovered 96% of all transcripts, while the `SPAdes` assembly using a kmer value of 75 recovered 93%. The `Trinity` assembly recovered 96% of the transcripts, while `Shannon` recovered 90%. Depicted in Figure 4, the `SPAdes` assembly using a kmer value of 55 recovered 3749 unique transcripts, `Trinity` recovered 3055, `Shannon` recovered 2526, and `SPAdes` assembly using a kmer value of 75 recovered 775.

# 4 Discussion

For non-model organisms lacking reference genomic resources, the error correction, adapter and quality trimmed reads should be assembled de novo into transcripts. While the assembly package `Trinity` (19) is thought to currently be the most accurate stand-alone assembler (32), this study demonstrates that a merged assembly with multiple assemblers (and kmer lengths) results in the highest quality assembly. Specifically, the Oyster River Protocol, which contains a recipe for read error correction, quality trimming, assembly with multiple software packages and merging, resulted in a final assembly, the structure of which was greatly improved.

`TransRate` scores were significantly improved by using the Oyster River Protocol for transcriptome assembly. One metric in particular, the read mapping metric, was vastly improved (Figure 1A). The aspect of quality that this metric assays is critical - specifically measuring how representative of the reads the assembly is. If we assume that the vast majority of generated reads come from the biological sample under study, when reads fail to map, that fraction of the biology is lost. Troublesome, this biology is lost from all downstream analysis and inference. This study conclusively demonstrates that across a wide variety of taxa, assembling with `Trinity` alone may result in a substantial decrease in mapping rate and in turn, the lost ability to draw conclusions from that fraction of the sample.

In contrast to `TransRate` scores, the `BUSCO` metrics were essentially unchanged by assembly with the Oyster River Protocol. The recovery of complete orthologs, the proportion of orthologs reconstructed in fragmented form, and missing orthologs were stable across both assembly methods. The number of orthologs recovered in duplicate (>1 copy), was decreased when using the ORP. Here, I hypothesize that the relative frequency of transcript duplication may have important implications for downstream abundance estimation, with less duplication potentially resulting in more accurate estimation. While gene expression quantitation software (53; 54) probabilistically assigns reads to transcripts in an attempt at mitigating this

issue, while not evaluated as part of this work, a primary solution related to decreasing artificial transcript duplication could offer significant advantages.

## 3.1 Each Assembler Recovers Different Transcripts

The main benefit of the Oyster River Protocol is related to the fact that assemblies are constructed four different ways, using three different assemblers (`Trinity`, `Shannon`, `SPAdes`) and three different values for kmer length (k=25,55,75). As described above, each assembler carries with it a set of heuristics, and these heuristics These differences may reflect a set of assembler-specific heuristics which translate into differential recovery of distinct fractions of the transcript community. Figure 3 depicts this process. Looking at the distribution of gene expression, within the `SPAdes` assemblies, kmer length influences the recovery of transcripts, with longer kmers shifting the distribution to more highly expressed transcripts. Interestingly, `Shannon` seems to have a very different set of expression-based biases, demonstrating an apparent bias against low-abundance transcripts. Trinity exhibits a typical distribution, similar to the `SPAdes` assembler using a shorter value for kmer length.

5 and Table 2 describe the outcomes of these processes in terms of transcript recovery. Taken together, these expression profiles suggest a mechanism by which the ORP outperforms , `Trinity`, and presumably other single-assembler assemblies. While there is substantial overlap in transcript recovery, each assembler recovers unique transcripts (Table 2 and Figure 5) , based on expression (and potentially other properties), which when merged together into a final assembly, increases the completeness

## 3.1 Does Taxonomy Influence Assembly Quality?

Because I was interested in designing a study with broad applicability, I chose read datasets that represented a variety of Eukaryotic groups. Although not originally designed for this purpose, this decision may allow me to understand the influence that intrinsic properties of transcription and transcriptome complexity in different taxonomic groupings may have on assembly. Figure 5 depicts several previously described assembly metrics, broken down by assembly method and by taxonomic group. Given the small sample (n=4 vertebrate, n=5 plant, n=6 invertebrate), it is impossible to draw strong conclusions, but generally, both `Trinity` and the Oyster River Protocol perform equally well across groups. Invertebrate assembly seems to be the most variable in resultant quality, though this may be driven by low sample size coupled with the specific (potentially low quality) datasets chosen at random.

Figure 5

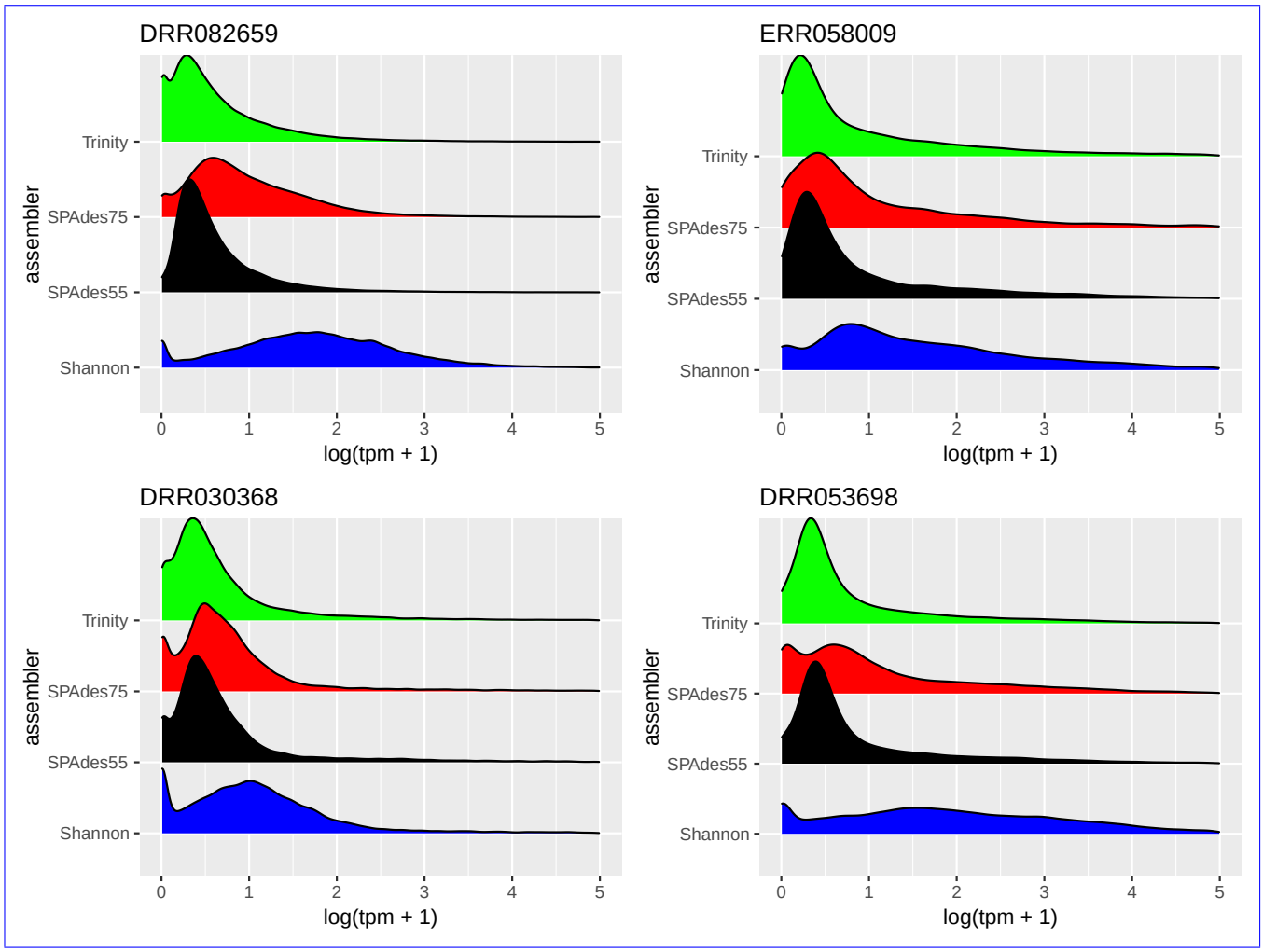


Figure 5 depicts the distribution of gene expression (log(TPM+1)), broken down by individual assembly, for four representative datasets. As predicted, the use of a higher kmer value with the `SPAdes` assembler resulted in biasing reconstruction towards more highly expressed transcripts. Interestingly, `Shannon` uniquely exhibits a bias towards the reconstruction of high-expression transcripts (or away from low-abundance transcripts).

## 3.1 ~~Does Read Depth Influence~~ Quality ~~?~~is independent of read depth

This study included read datasets of a variety of sizes. Because of this, I was interested in understanding if the number of reads used in assembly was strongly related to the quality of the resultant assembly. Conclusively, this study demonstrates that between 30 million paired-end reads and 200 million paired-end reads, no strong patterns in quality are evident (Figure 6). This finding is in line with previous work, (42) suggesting that assembly metrics plateau at between 20M and 40M read pairs, with sequencing beyond this

level resulting in minimal gain in performance.

Figure 6

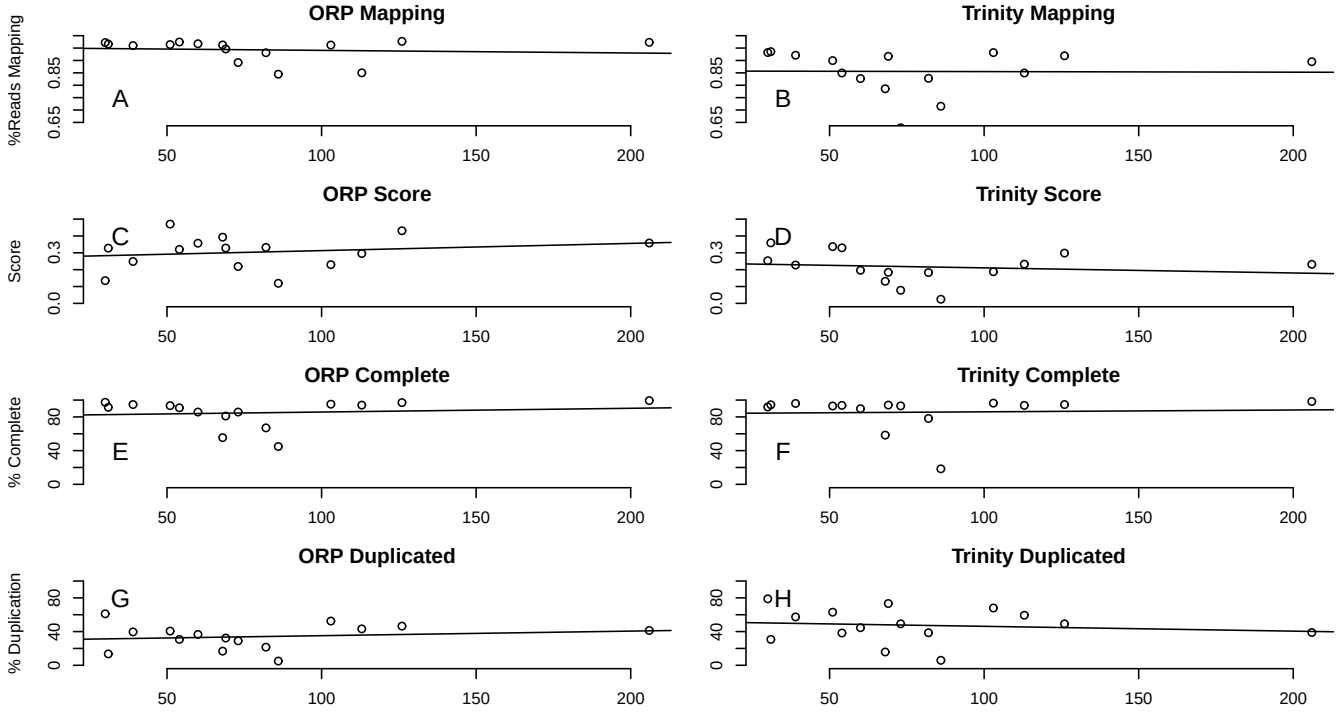


Figure 6 depicts the relationship between a subset of assembly metrics and the number of read pairs. There is no significant relationship. In all cases the x-axis is millions of paired-end reads.

# 4 Conclusions

For non-model organisms lacking reference genomic resources, the error ~~correction, adapter and quality trimmed reads should~~ corrected, adapter- and quality-trimmed reads must be assembled de novo into transcripts. While the assembly package `Trinity` (19) is thought to currently be the most accurate stand-alone assembler (32), a merged assembly with multiple assemblers results in ~~the highest quality assembly~~higher quality assemblies. Specifically, use of the Oyster River Protocol, which contains a recipe for read error correction, quality trimming, assembly with multiple software packages, and merging resulted in a final assembly, the structure of which was greatly improved.

Specifically, the improvements in assembly metrics described here are attributed to the multi-way approach, where three different assemblers and three different kmer lengths were used. This approach allows the strengths of one ~~approach~~ assembler to effectively complement the weaknesses of another, thereby resulting in a more complete assembly than otherwise possible. These enhancements are important, as

unassembled transcripts are invisible to all downstream analysis.

## Acknowledgments

This work was significantly improved by discussions with Richard Smith-Unna, Rob Patro, C. Titus Brown, Brian Haas and many others. More generally, the work and ~~it's~~ its presentation has been influenced by supporters of the Open Access and Open Science movements.

## References

1. Mortazavi A, Williams BA, Mccue K, Schaeffer L, Wold B (2008) Mapping and quantifying mammalian transcriptomes by RNA-Seq. Nature Methods 5: 621–628.

2. Wang Z, Wang Z, Gerstein M, Snyder M (2009) RNA-Seq: a revolutionary tool for transcriptomics. Nature Reviews Genetics 10: 57–63.

3. Lappalainen T, Sammeth M, Friedländer MR, t Hoen PAC, Monlong J, et al. (2013) Transcriptome and genome sequencing uncovers functional variation in humans. Nature 501: 506–511.

4. Cahoy JD, Emery B, Kaushal A, Foo LC, Zamanian JL, et al. (2008) A transcriptome database for astrocytes, neurons, and oligodendrocytes: a new resource for understanding brain development and function. The Journal of neuroscience : the official journal of the Society for Neuroscience 28: 264–278.

5. Li X, Kim Y, Tsang EK, Davis JR, Damani FN, et al. (2017) The impact of rare variation on gene expression across tissues. Nature 550: 239–243.

6. Tan MH, Li Q, Shanmugam R, Piskol R, Kohler J, et al. (2017) Dynamic landscape and regulation of RNA editing in mammals. Nature 550: 249–254.

7. Fitzpatrick M, Ben-Shahar Y, Vet L, Smid H, Robinson GE, et al. (2005) Candidate genes for behavioural ecology. Trends In Ecology & Evolution 20: 96–104.

8. Panhuis TM (2006) Molecular evolution and population genetic analysis of candidate female reproductive genes in Drosophila. Genetics 173: 2039–2047.

9. Wolf JBW (2013) Principles of transcriptome analysis and gene expression quantification: an RNA-seq tutorial. Molecular Ecology Resources 13: 559–572.

10. Vijay N, Poelstra JW, Künstner A, Wolf JBW (2013) Challenges and strategies in transcriptome assembly and differential gene expression quantification. A comprehensive in silico assessment of RNA-seq experiments. Molecular Ecology 22: 620–634.

11. Robinson MD, McCarthy DJ, Smyth GK (2010) edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. Bioinformatics 26: 139–140.

12. Love MI, Huber W, anders S (2014) Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. Genome Biology 15: 550.

13. Robertson G, Schein J, Chiu R, Corbett R, Field M, et al. (2010) De novo assembly and analysis of RNA-seq data. Nature Methods 7: 909–912.

14. Schulz MH, Zerbino DR, Vingron M, Birney E (2012) Oases: robust de novo RNA-seq assembly across the dynamic range of expression levels. Bioinformatics 28: 1086–1092.

15. Xie Y, Wu G, Tang J, Luo R, Patterson J, et al. (2014) SOAPdenovo-Trans: de novo transcriptome assembly with short RNA-Seq reads. Bioinformatics 30: 1660–1666.

16. Simpson JT, Wong K, Jackman SD, Schein JE, Jones SJM, et al. (2009) ABySS: A parallel assembler for short read sequence data. Genome Research 19: 1117–1123.

17. Zerbino DR, Birney E (2008) Velvet: Algorithms for de novo short read assembly using de Bruijn graphs. Genome Research 18: 821–829.

18. Li R, Li Y, Kristiansen K, Wang J (2008) SOAP: short oligonucleotide alignment program. Bioinformatics 24: 713–714.

19. Haas BJ, Papanicolaou A, Yassour M, Grabherr M, Blood PD, et al. (2013) De novo transcript sequence reconstruction from RNA-seq using the Trinity platform for reference generation and analysis. Nature Protocols 8: 1494–1512.

20. Peng Y, Peng Y, Leung HCM, Leung HCM, Yiu SM, et al. (2013) IDBA-tran: a more robust de novo de Bruijn graph assembler for transcriptomes with uneven expression levels. Bioinformatics 29: i326–i334.

21. Liu J, Li G, Chang Z, Yu T, Liu B, et al. (2016) BinPacker: Packing-Based De Novo Transcriptome Assembly from RNA-seq Data. PLOS Computational Biology 12: e1004772.

22. Kannan S, Hui J, Mazooji K, Pachter L, Tse D (2016) Shannon: An Information-Optimal de Novo RNA-Seq Assembler. bioRxiv .

23. Jiang H, Lei R, Ding SW, Zhu S (2014) Skewer: a fast and accurate adapter trimmer for next-generation sequencing paired-end reads. BMC Bioinformatics 15: 182.

24. Bolger AM, Lohse M, Usadel B (2014) Trimmomatic: a flexible trimmer for Illumina sequence data. Bioinformatics 30: btu170–2120.

25. Martin M (2011) Cutadapt removes adapter sequences from high-throughput sequencing reads. EMBnetjournal 17: pp. 10–12.

26. Pell J, Hintze A, Canino-Koning R, Howe A, Tiedje JM, et al. (2012) Scaling metagenome sequence assembly with probabilistic de Bruijn graphs. Proceedings of the National Academy of Sciences 109: 13272–13277.

27. Le HS, Schulz MH, McCauley BM, Hinman VF, Bar-Joseph Z (2013) Probabilistic error correction for RNA sequencing. Nucleic Acids Research 41: 1–11.

28. Song L, Florea L (2015) Rcorrector: efficient and accurate error correction for Illumina RNA-seq reads. GigaScience 4: 48.

29. Yang X, Yang X, Dorman KS, Dorman KS, Aluru S, et al. (2010) Reptile: representative tiling for short read error correction. Bioinformatics 26: 2526–2533.

30. Smith-Unna R, Boursnell C, Patro R, Hibberd JM, Kelly S (2016) TransRate: reference-free quality assessment of de novo transcriptome assemblies. Genome Research 26: 1134–1144.

31. Simão FA, Waterhouse RM, Ioannidis P, Kriventseva EV, Zdobnov EM (2015) BUSCO: assessing genome assembly and annotation completeness with single-copy orthologs. Bioinformatics 31: 3210–3212.

32. Li B, Fillmore N, Bai Y, Collins M, Thomson JA, et al. (2014) Evaluation of de novo transcriptome assemblies from RNA-Seq data. Genome Biology 15: 663–21.

33. Ungaro A, Pech N, Martin JF, McCairns RJS, Mévy JP, et al. (2017) Challenges and advances for transcriptome assembly in non-model species. PloS one 12: e0185020–21.

34. Wang S, Gribskov M (2017) Comprehensive evaluation of de novo transcriptome assembly programs and their effects on differential gene expression analysis. Bioinformatics 33: 327–333.

35. Moreton J, Izquierdo A, Emes RD (2015) Assembly, Assessment, and Availability of De novo Generated Eukaryotic Transcriptomes. Frontiers in Genetics 6: 361.

36. Yang Y, Smith SA (2013) Optimizing de novo assembly of short-read RNA-seq data for phylogenomics. BMC Genomics 14: 328.

37. Marchant A, Mougel F, Mendonça V, Quartier M, Jacquin-Joly E, et al. (2016) Comparing de novo and reference-based transcriptome assembly strategies by applying them to the blood-sucking bug Rhodnius prolixus. Insect Biochemistry and Molecular Biology 69: 25–33.

38. Finseth FR, Harrison RG (2014) A comparison of next generation sequencing technologies for transcriptome assembly and utility for RNA-Seq in a non-model bird. PloS one 9: e108550.

39. Jackman SD, Birol I (2016) Linuxbrew and Homebrew for cross-platform package management [version 1; not peer reviewed]. In: F1000.

40. MacManes MD (2014) On the optimal trimming of high-throughput mRNA sequence data. Frontiers in Genetics 5: 13.

41. MacManes MD, Eisen MB (2013) Improving transcriptome assembly through error correction of high-throughput sequence reads. PeerJ 1: e113.

42. MacManes MD (2015) An opinionated guide to the proper care and feeding of your transcriptome. biorxivorg : 1–23.

43. Chikhi R, Medvedev P (2014) Informed and automated k-mer size selection for genome assembly. Bioinformatics 30: 31–37.

44. Emms DM, Kelly S (2015) OrthoFinder: solving fundamental biases in whole genome comparisons dramatically improves orthogroup inference accuracy. Genome Biology 16: 157.

45. Li B, Fillmore N, Bai Y, Collins M, Thomson JA, et al. (2014) Evaluation of de novo transcriptome assemblies from RNA-Seq data. Genome Biology 15: 553.

46. Scott C (2017) shmlast: An improved implementation of Conditional Reciprocal Best Hits with LAST and Python. The Journal of Open Source Software 2: 1–4.

47. Aubry S, Kelly S, Kümpers BMC, Smith-Unna RD, Hibberd JM (2014) Deep Evolutionary Comparison of Gene Expression Identifies Parallel Recruitment of Trans-Factors in Two Independent Origins of C4 Photosynthesis. PLOS Genetics 10: e1004365.

48. Titus Brown C, Irber L (2016) sourmash: a library for MinHash sketching of DNA. The Journal of Open Source Software 1: 27–1.

49. R Core Development Team F (2011) R: A Language and Environment for Statistical Computing .

50. Kampstra P (2008) Beanplot: A boxplot alternative for visual comparison of distributions .

51. Conway JR, Lex A, Gehlenborg N (2017) UpSetR: An R Package for the Visualization of Intersecting Sets and their Properties. Bioinformatics .

52. Singhal S (2013) De novotranscriptomic analyses for non-model organisms: an evaluation of methods across a multi-species data set. Molecular Ecology Resources 13: n/a–n/a.

53. Patro R, Duggal G, Love MI, Irizarry RA, Kingsford C (2017) Salmon provides fast and bias-aware quantification of transcript expression. Nature Methods 14: 417–419.

54. Bray NL, Pimentel H, Melsted P, Pachter L (2016) Near-optimal probabilistic RNA-seq quantification. Nature Biotechnology 34: 525–527.