Project Proposal: Comparative analysis of Gene Finding tools when applied to *Trichoderma* genomes

Connor Burbridge[1]

[1]USask NSID: cbe453 , USask ID no. 11162928 , Supervisors: Dave Schneider & Tony Kusalik,

July 10, 2023

Contents

1 Pre-work

1.1 Existing selection of Genome Assemblies

Comparison of several different *Trichoderma* genomes is important in the context of gene finding tools as different gene finding tools will find different genes and understanding how these tools behave in the context of different *Trichoderma* genomes could prove useful for those looking to find genes in similar fungal genomes in the future. To accent the processing for genomes of interest, those being DC1 and Tsht20, we should include other previously assembled *Trichoderma* assemblies. Currently selected genomes include *Trichoderma reesei*, *Trichoderma harzianum*, and *Trichoderma virens?*, with *Trichoderma reesei* being the 'reference' in this case, as it is well studied and there are several patents involving it's use a organsim for production of compounds such as antibiotics in industrial applications.

1.2 Assembly

The foundation of this project is base ond the sequencing of two novel *Trichoderma* strains identified in prairie regions of Canada (Alberta and Saskatchewan). To assemble these genomes, a hybrid assembly process was used, following default assembly parameters with MASuRcA, which utilizes the Flye assembler if both Nanopore and Illumina data are used as inputs, which are the inputs in this case. The next paragraph describes the process of working with MASuRcA.

MASuRcA 4.0.3 was run using the Compute Canada software stack available on Copernicus. Prior to loading the MASuRcA environment, the GCC/9.0 and StdEnv/2020 modules must be loaded first. This version of the software is not ideal, but the Anaconda installation of versino 4.0.9 consistently failed, even in a fresh environment. Building the software from scratch is a potential option. In addition to this difficulty, the assemblies were performed in the p2irc_rsmi scratch space on Copernicus as I was encountering permissions issues when trying to run the assembly in the Roots datastore. I don't know exactly why (microsoft permissions problems from datastore?, but there were permission issues associated with scripts being copied to datastore as part of the assembly process. All assembly materials were copied back to datastore after assembly.

Initially, a configuration file must be generated to run MASuRcA with the

optimal combination of assembly tools for the data supplied (Flye + polishing). Running MASuRcA from the commandline tool utilizes the CABOG celera-based assembler, which is noted as being slower and results in an assembly with similar or worse quality than one using Flye.

To generate the config file, run the following: masurca -g config.txt

The config file was then altered to provide input file, options and allowable number of threads for assembly. All other assembly parameters were left untouched. Insert lengths for the Illumina read data used the recommended values (stated to work for most Illumina reads), although these could be modified with input from Brendan. The config files for both assemblies are available in the assebly directories.

To generate the assembly.sh script, run: masurca config.txt

Once the assembly.sh script is generated, execute the assembly using: ./assemble.sh

Final assemblies are placed in directories with the prefix flye.mr.* Quast analysis of the genomes was also performed, with the output being placed in directories named quast within the assembly directory.

To run Quast: quast -o ./ -t 16 assembly-file.fasta

In an attempt to produce higher quality assemblies of DC1 and Tsth20, It has been suggested that I try a set of tools call NextDenovo and NextPolish as they have produced excellent assemblies based on previous experience from supervisors.

Installation of NextDenovo was straightforward. Simply download the compressed tar file from their website and unpack it. NextDenovo requires Python versions 2 and 3 along with a package called parallel to aid in parallel processing of datasets. I installed parallel using pip in the bioinformatics conda environment in the scracth space of Copernicus.

Initial attempts to run the example dataset resulted in some permissions errors, which I have encountered with other tools in the past. Thank you datastore. To remedy this, I copied the installation to RSMI's scratch space on Copernicus. Once the approriate permissions were given to run nextDenovo, I was able to run the example dataset assembly without issue. Future assemblies of DC1 and Tsth20 will be performed in this scratch space to avoid permissiions issues and results will be copied to datastore.

1.3 Repeat Masking

In order to evaluate the performance of gene finding tools in repetitive or low complexity regions in the context of *Trichoderma* genomes, we must first identify said regions in the genomes considered. To do this, RepeatMasker has been selected as a tool to identify repeat regions based on a fungal subset of the Dfam database by specifying the fungi species tag to RepeatMasker when running the program. The program was configured with options to produce several output formats for each genome considered, which will allow for more informative downstream analysis of results. All commands for repeat masking are located withing the processing directory for each strain/genome.

The Installation procedure was somewhat indepth, requiring RepeatMasker configuration, which itself requires downloading an appropriate repeat database (Dfam in this case, included with RepeatMasker), installation of Tandem Repeat Finder (TRFM) and installation of a sequence search tool, for which I chose HMMER from the list of potential tools as I am generally familiar with its use. The path to the installation of TRFM is required during configuration along with the search tool of choice, a simple selection of 4 tools that will have an autocompleted path in my case, since HMMER is installed via anaconda.

General command for running RepeatMasker: /datastore/Roots/Connor/masters/software/repe
-pa 10 -a -small -species fungi -html -gff -dir ./ path-to-genome/genome.fasta

2 Software Installation

Currently, GeneMark-ES and Braker2 have been difficult to install and have not been successfully installed yet.

GenomeThreader installed successfully via Anaconda in the bioinformatics environments on cnic-gifs-aio-18001 (rsmi01).

3 Gene Finding

Now that we have covered information about assembling and installation of these tools, we can cover the gene finding portion of this work.

To begin, I ran GeneMark-ES as it requires no prior information or alignments in order to run. In this case GeneMark-ES has an option specifically for fungal genomes, which I chose to use in this case.

GenomeThreader is currently undergoing a test run with only two cDNA files for the SRA accession SRR5229930. The command itself was straightforward to run, although I am waiting for a successful run to finish to confirm that. The only other thing to mention was that GenomeThreader seems to only accept FASTA files as input. FastQ files were not accepted and failed with an illegal character error for the @ headers on the FastQ files. To rectify this, I used seqret to convert the FastQ files to FastA files.

GenomeThreader update: GenomeTHreader was running for over 1000 hours, so I eventually ended the task. Unsure on whether ot not this is my fault or an issue with the program or installation. The program was using 250+ GB of memory while running so I assume something was happening, but it could be a memory leak or just a really slow program. Either way, I don't think it is an appropriate option for this project.

Seqret basic syntax: **seqret -fastq seqfile.fastq -fasta seqfile.fasta**

Braker2 has been successfully run on the RSMI box with the help of Brook from Research Computing. Issues with Anaconda and glibc incompatibilities have been frustrating and difficult to deal with. Brook has set up several modules including an initialization script to get things up and running AND create a reloadable environment for reuse. Once the environment has been loaded, one must load the Hisat2 module from compute canada as well as an htslib module (more detail to come). Once all modules are loaded, there are a few environemnt variables that need to be set. Alternatively, these variables can be set within the braker2.pl command, which have higher priority over environment variables and probably makes things easier to track.

The variables that need to be set are AUGUSTUS_CONFIG_PATH and TSEBRA_PATH. Augustus, by defuault, tries to write species information to the location where the software is installed. In this case, we don'thave write permissions to the compute canada software stack hosted byt Research Computing, so the AUGUSTUS_CONFIG_PATH variable must be set in order to create a writeable directory. As long as that path has a directory within it called braker, and a species directory within the braker directory, things should go smoothly. TSEBRA is a set of scripts also made by the creators of Braker and is required to merge results from the various gene prediction tools involved in the Braker2 pipeline. The TSEBRA_PATH simply points to the directory where TSEBRA is located Both Braker2 and TSEBRA can be cloned directly from GitHub (links to come)

4 Downstream Analysis

After completion of the processing portion of this work, the results must be processed in a useful way, which includes both the biological implications of the gene calls as well as the computational, or gene finding features, of the the selected programs. To better understand how gene finders perform in these two classes, we must define an appropriate plan for analysis of the results produced so far. Currently, downstream analysis plan has been broken down into several sections.

4.1 Basic Analysis

Basic analysis of gene finding results is an important part of this research. Total gene, transcript and protein counts will be identified for each genome and gene finding tool combination. Comparing the general outputs of these programs will provide an idea of their performance in different *Trichoderma* genomes. Analysis for these results can performed through simple shell scripting with grep and other unix tools, although processing through Python might provide results that are easier to reproduce with proper programming. Having one script with several modules that can be rerun at will would be easier to handle than multiple shell scripts. This thinking for processing will be applied to subsequent sections of this as well.

4.2 Distribution of Gene Lengths

One important aspect of gene finding tools to consider is the distribution of gene lengths predicted by each individual tool. Certain tools, such as GeneMark are based on pre-defined models, which may limit the length of predicted genes, while tools such as Braker2, which incorporate RNAseq data, may predict a wider distribution of gene lengths depending on the input dataset used. Regardless, the ability of a gene finding tool to predict a wider range of gene lengths can be usefull if users are looking for short or larger genes. To help determine whether or not these tools find shorter genes, or small RNAs, the genomes of interest have been annotated using Infernal along with the Rfam database. These annotation results will also be included with results from other annotation processes further down the line. Again, these results can be produced with a Python script. The resulting data could then be used as input to violin plots for each genome and set

of tools considered in this analysis process. Violin plots should provide a good visualization of gene lengths as well as the number of genes found with specific lengths. Means could also be compared staatistically for genomes and the mutliple tools considered as well.

4.3 Intersection of Gene Calls, smallRNAs and Repetitive Regions

Annotation of all three features in the title are important in assessing the ability of gene finding tools. Even more important, is the potential for for overlap between gene calls and small RNAS as well as repetitive regions the genomes. As discussed in the previous subsection, the distribution of gene lengths predicted by a gene finding tool can be an important metric for users. Overlapping predicted genes from tools alongside the output from Infernal and the Rfam database may provide insight into whether or not these gene finding tools are able to predict RNAs of very short length. In addition to small RNAs, repetitive regions in *Trichoderma* genomes hold potential for recombination and gene content, although the inherent nature of these repetitive regions (low nucleotide diveristy) suggests that gene content should be low, based on the nucleotides required for start and stop codons. Analysis of these intersections can be perfomed via bedtools or through biopython (I believe). Again, having all processing steps included in one script as separate functions that can be called at whim will make further processing easier if changes need to be made.

4.4 Shared Gene Content with Yeast

While considering novel gene calls can be useful, comparing those calls to a well-studied close relative can provide a rudimentary validation of the calls as a ground truth. In this case, a comparison to Yeast will be made. The agreed upon number for successful gene-finding as compared to Yeast is roughly 80-85% of gene content. This will confirm that at least most of a closely related fungal genome's content is predicted and shared by the gene calls for *Trichoderma*. Results for this processing can be produced with a simple BLAST search and apropriate cutoff values (i.e. query coverage, percent nucelotide identity, E-score, etc.). Other tools for evaluation will certainly be considered, although I will need to look into this process further.