Proteome Generation Pipeline

Hardware Requirements

Given the high memory requirements of some of the pipeline steps – especially when using large input files – we recommend the use of a Linux server to run each of the steps described below. In our own implementation using the Biowulf high-performance computing resource at NIH (https://hpc.nih.gov), runs required 100 GB to 200 GB of memory and 16 to 32 CPUs on average for the Trinity step to finish in a timely manner (1 to 3 days). However, depending on the size of the input data set, it was sometimes necessary to increase the memory allowance to 800 GB. For some of these memory-intensive runs, the run time for the full pipeline ranged from 3 to 8 days. The worked example from this manual required ~170 GB of disk storage; users should allocate more disk storage for larger data sets.

Software Components

fasterq-dump	Component of the SRAtoolkit, at https://github.com/ncbi/sra-tools (v 2.9.6)
FastQC	https://github.com/s-andrews/FastQC (v 0.11.8)
Trim Galore!	https://github.com/FelixKrueger/TrimGalore (v 0.6.5)
Rcorrector	https://github.com/mourisl/Rcorrector (v 1.0.3.1)
Trinity	https://github.com/trinityrnaseq/trinityrnaseq/releases (v 2.9.0)
TransDecoder	https://github.com/TransDecoder/TransDecoder/releases (v 5.5.0)
BUSCO	https://gitlab.com/ezlab/busco/-/releases (v 4.0.2)
InterProScan	https://github.com/ebi-pf-team/interproscan (v 5.33-72.0)

The version numbers of each software component used for the current AniProtDB release are shown above. Please note that you may need to work with your system administrator to ensure that the above tools are available in your path. The parameters suggested below for individual tools have worked well for our datasets but can be adjusted as needed for specific use cases.

Instructions

1. Download files of interest from SRA or any public repository. Alternatively, the user may use their own RNA-seq data sets.

   To download files from an SRA project using a data set accession number, users should use the `fasterq-dump` tool with the `--split-3` flag to ensure that if a data set contains unpaired reads, they will be written to a separate file. Users should also confirm that the accession numbers correspond only to RNA-seq data sets, as some SRA BioProjects also include genomic data within the same BioProject. For example, the *Hoilungia hongkongensis* SRA project PRJNA377631 contains three experiment sets, but only one of these data sets (SRR5311041) contains RNA-seq data.

   Keep in mind that a BioProject may contain more than one RNA-seq data set; if this is the case, and if the additional data sets are of biological interest, each of the data sets will need to be processed separately through Steps 1-5 of this pipeline. All output files resulting from Step 5 will then be combined at the point of transcriptome assembly (Step 6).

   Example usage:

   ```
   $ fasterq-dump --split-3 SRR5311041
   ```

   In order to prepare these files for a later step in the pipeline that uses Trinity, the headers of the paired-end RNA-seq files need to have `/1` and `/2` appended to their sequence headers. We suggest using `awk` to edit the sequence headers and then overwriting the original dumped files to save space:

```
$ awk '{ if (NR%4==1 || NR%4==3) { print $1"/1" } else { print } }'
  SRR5311041_1.fastq > temp_SRR5311041_1.fastq
$ awk '{ if (NR%4==1 || NR%4==3) { print $1"/2" } else { print } }'
  SRR5311041_2.fastq > temp_SRR5311041_2.fastq
$ mv temp_SRR5311041_1.fastq SRR5311041_1.fastq
$ mv temp_SRR5311041_2.fastq SRR5311041_2.fastq
```

2. Assess the quality of the sequencing reads

FastQC provides a modular set of analyses for assessing the overall quality of raw sequencing data, flagging significant errors that may have arisen during the sequencing process. Additional documentation can be found on the FastQC web site, at *https://www.bioinformatics.babraham.ac.uk/projects/fastqc/* .

Example usage:

```
$ mkdir fastqc
$ fastqc -o fastqc/ SRR5311041_1.fastq SRR5311041_2.fastq
```

The FastQC output report (in HTML format) will be written to the output directory chosen by the user. In the example above, the output files will have the suffix `_fastqc.html` that can be opened using any web browser. It is important to check the ‘Per base sequence quality’ plot, which provides the distribution of quality scores across all bases at each position in the reads. Ideally, the plot will show all bases within the green area of the plot, above a pre-set quality score of 28. (As a reference, a quality score of 20 indicates a base call accuracy of 99%, while a quality score of 30 indicates a base call accuracy of 99.9%.) If the distribution of quality scores is located entirely below the green area of the plot, the data set should be discarded. Otherwise, the data can be passed to the next step in the pipeline, for trimming low quality bases from the ends of each read.

Users should also examine the ‘Overrepresented sequences’ table, which displays sequences that occur in more than 0.1% of the total number of reads. If there are overrepresented sequences, users should do a quick BLAST search of these sequences

against the GenBank nucleotide (`nt`) database to assess possible sources of contamination; these may include ribosomal sequences, microbial sequences, and host RNA. Our own rule of thumb is to discard the data set if these overrepresented sequences represent more than 5% of total reads.

3. Perform quality and adapter trimming

   Using the results of the FastQC analysis as a guide, users should run Trim Galore to remove low-quality bases, overrepresented sequences, and adapters from the ends of reads.

   Example usage:

```
$ mkdir trimgalore
$ cd trimgalore
$ trim_galore --paired -q 30 --stringency 5 --length 70
  ../SRR5311041_1.fastq ../SRR5311041_2.fastq
```

   Using the parameters above, Trim Galore will trim bases from each end of the read when the per-base quality score is below 30 and discard reads (and their mate pairs) if the reads are shorter than 70 bp after trimming and adapter removal. The `-a` and `-a2` flags let the user specify adapter sequences or specific overrepresented sequences that should be removed from the files. The output files will have the suffix `_val_1.fq` and `_val_2.fq`.

4. Assess the quality of trimmed sequencing reads

   Following the Trim Galore step, a second round of FastQC should be carried out to ensure that the overall read quality has improved after trimming.

Example usage:

```
$ mkdir fastqc_2ndRound
$ fastqc -o fastqc_2ndRound/ SRR5311041_1_val_1.fq
  SRR5311041_2_val_2.fq
```

As in Step 2, data sets with low quality scores or a high percentage of overrepresented sequences should be discarded at this point. See above for details.

5. Perform error correction to correct sequencing errors

   For data sets meeting the above quality control criteria, Rcorrector was then used to further optimize the individual reads by correcting random sequencing errors prior to performing *de novo* transcriptome assembly.

   Example usage:

```
$ mkdir rcorrector
$ cd rcorrector
$ run_rcorrector.pl -1 ../SRR5311041_1_val_1.fq -2
  ../SRR5311041_2_val_2.fq -t 16 -k 31
```

   The `-t` flag sets the number of threads to use and the `-k` flag specifies the kmer length. The output files will have the suffix `.cor.fq`.

6. Transcriptome assembly

   Trinity is used to perform *de novo* assembly of the trimmed and error-corrected transcript sequences created in Steps 1-5.

Example usage:

```
$ mkdir trinity
$ cd trinity
$ Trinity --seqType fq --max_memory 240G --left
  ../SRR5311041_1_val_1.cor.fq --right ../SRR5311041_2_val_2.cor.fq
  --CPU 12 --min_kmer_cov=2 --SS_lib_type FR --output
  trinity_Hhongkongensis
```

The flag `--min_kmer_cov` sets the minimum number of k-mers necessary for a transcript to be considered real and `flag --SS_lib_type` specifies the orientation of the paired-end reads. Of note, and depending on the size of the input files, the `--max_memory` flag may need to be increased to at least 800 GB. In our experience, run times may be as long as eight days, depending on the data set. Once the run is complete, the output transcript file will have the filename `Trinity.fasta` and be located within the `trinity_out_dir` directory by default. Alternatively, the output directory can be specified using the `--output` flag.

In cases where there are multiple RNA-seq data sets from a single BioProject being analyzed, transcripts from each set of left and right reads can be combined at this step by specifying the files containing the results of Step 5 as a comma-separated list, as follows:

```
$ Trinity --seqType fq --max_memory 240G --left
../SRR5821541_1_val_1.cor.fq,../SRR5821542_1_val_1.cor.fq,../SRR5821543_
1_val_1.cor.fq,../SRR5821544_1_val_1.cor.fq,../SRR5821545_1_val_1.cor.fq
,../SRR5821546_1_val_1.cor.fq,../SRR5821547_1_val_1.cor.fq,../SRR5821548
_1_val_1.cor.fq,../SRR5821549_1_val_1.cor.fq,../SRR5821550_1_val_1.cor.f
q,../SRR5821551_1_val_1.cor.fq,../SRR5821552_1_val_1.cor.fq,../SRR582155
3_1_val_1.cor.fq,../SRR5821554_1_val_1.cor.fq --right
../SRR5821541_2_val_2.cor.fq,../SRR5821542_2_val_2.cor.fq,../SRR5821543_
2_val_2.cor.fq,../SRR5821544_2_val_2.cor.fq,../SRR5821545_2_val_2.cor.fq
,../SRR5821546_2_val_2.cor.fq,../SRR5821547_2_val_2.cor.fq,../SRR5821548
_2_val_2.cor.fq,../SRR5821549_2_val_2.cor.fq,../SRR5821550_2_val_2.cor.f
q,../SRR5821551_2_val_2.cor.fq,../SRR5821552_2_val_2.cor.fq,../SRR582155
```

```
3_2_val_2.cor.fq,../SRR5821554_2_val_2.cor.fq --CPU 32 --min_kmer_cov=2
--SS_lib_type FR --output trinity_Uunicinctus
```

7. Identifying open reading frames and translation to proteins

   TransDecoder is used to identify open reading frames within the transcripts from the *de novo* assemblies generated in Step 6. `TransDecoder.LongOrfs` detects the open reading frames, and `TransDecoder.Predict` then generates protein translations for each isoform.

   Example usage:

```
$ mkdir transdecoder
$ cd transdecoder
$ TransDecoder.LongOrfs -t ../trinity_Hhongkongensis/Trinity.fasta
$ TransDecoder.Predict -t ../trinity_Hhongkongensis/Trinity.fasta
```

   The main output file of this step is `Trinity.fasta.transdecoder.pep`, which is located in the working directory.

8. Filter out incomplete and redundant proteins to create the final proteome file

   To ensure that accurate BUSCO completeness scores are generated in the next step, it is important that each proteome file contains one and only one copy of each full-length protein, as having more than one copy of a given protein sequence can artificially inflate the BUSCO score.

Example usage:

```
$ mkdir proteome
$ cd proteome
$ awk '/^>/ {printf("\n%s\n",$0);next; } { printf("%s",$0);} END
  {printf("\n");}' < ../Trinity.fasta.transdecoder.pep | sed "1d" >
  Trinity.fasta.transdecoder_singleLine.fasta
$ perl filter_redundancy.pl Hhongkongensis
```

The `awk` command converts a multi-line FASTA file to a single-line FASTA file, the required format for the filtering script `filter_redundancy.pl`. This script reads in the file `Trinity.fasta.transdecoder_singleLine.fasta`, removes any proteins that do not have a start and stop codon, and creates a new proteome file, `Hhongkongensis_proteins.fasta`, using the single parameter (the species name) indicated on the command line as the basis for naming the file. The output of this step is the final proteome file for this species.

9. Assessing proteome completeness

In the third and final quality control checkpoint, BUSCO is used to search for the presence of a core set of conserved genes in the final proteome file. Here, a BUSCO score threshold of 70% is applied, allowing the user to retain proteomes from less-represented phyla as a way to increase taxonomic breadth.

Example usage:

```
$ mkdir BUSCO
$ cd BUSCO
$ busco -i ../Hhongkongensis_proteins.fasta -c 8 -o Hhongkongensis
  -l metazoa_odb10 -m proteins
```

For this analysis, the set of core metazoan genes available within BUSCO (`metazoa_odb10`) was used to assess the completeness of the final proteome file,

`Hhongkongensis_proteins.fasta`. The `-m` flag specifies the input data are proteins and the `-o` flag specifies the output directory. Using the commands above, the following results were generated:

```
***** Results: *****

C:89.5%[S:52.1%,D:37.4%],F:2.2%,M:8.3%,n:954
854     Complete BUSCOs (C)
497     Complete and single-copy BUSCOs (S)
357     Complete and duplicated BUSCOs (D)
21      Fragmented BUSCOs (F)
79      Missing BUSCOs (M)
954     Total BUSCO groups searched
```

With a completeness (`C`) score of 89.5%, the final proteome, created in Step 8, passes our stated quality threshold of 70%.

10. Identification of protein domains

InterProScan is a tool that integrates predictive information about the function of a protein from a number of partner resources, allowing the user to determine whether the protein belongs to specific protein families and which protein domains are found within its sequence. The command line version of InterProScan enables the analysis of an entire proteome per run.

Example usage:

```
$ cd ../
$ ./interproscan.sh -appl CDD -t p -f tsv -pa -goterms
  -iprlookup -i Hhongkongensis_proteins.fasta
  -o interproscan_Hhongkongensis_CDD_results.txt
```

In the example above (and for the AniProtDB proteomes), the Conserved Domain Database (CDD) was chosen as the reference domain database and is specified using the `--appl` flag. CDD content includes NCBI-curated domains, which use three-dimensional structural information to explicitly define domain boundaries; CDD also uses

models from external source databases (Pfam, SMART, COG, PRK, and TIGRFAMs) to reliably detect protein domains.

The `-t` flag specifies the type of molecule (`p` for protein) and the `-f` flag specifies the format of the output file (here, `tsv` for tab-separated values). The `-pa` and `-goterms` flags can be used to include pathway annotations and Genome Ontology (GO) term annotations, respectively. Both of these flags require the use of `-iprlookup` in the command in order to include InterPro annotations in the output file. The `-o` flag specifies the name of the output file. The resulting file, `interproscan_Hhongkongensis_CDD_results.txt`, can be opened using Excel and can also be used as input to the Perl script described in the next step. This output file contains the list of protein domains found in each protein.

11. Retrieving proteins containing a domain of interest

The script `parse_interproscan.pl` can be used to retrieve protein sequences containing a domain of interest. The script requires the accession number of the domain, the species name (used to generate the output file and output directory), the final proteome file, and the InterProScan results file as input.

Note: For this step, the input proteome file needs to be converted to single-line format if it is not already in that format. To convert a multi-line FASTA file into a single-line FASTA file, issue the following command prior to proceeding with this step:

```
$ awk '/^>/ {printf("\n%s\n",$0);next; } { printf("%s",$0);} END
  {printf("\n");}' < infile.fasta | sed "1d" > outfile.fasta
```

Example usage:

```
$ perl parse_interproscan.pl -d cd00934
  -s "Hoilungia hongkongensis" -f Hhongkongensis_proteins.fasta
  -i interproscan_Hhongkongensis_CDD_results.txt
```

The `-d` flag specifies the protein domain of interest, while the `-s` flag specifies the name of the species and also names the output directory. The proteome file and the InterProScan files are specified with the `-f` and `-i` flags, respectively.

In this example, the domain of interest is `cd00934` (PTB, a phosphotyrosine-binding domain). This script will create a directory called `Hoilungia_hongkongensis_cd00934/` and a file named `output_Hoilungia_hongkongensis_cd00934_proteins.fasta` that contains all *H. hongkongensis* proteins that contain this domain (six, in this case).

Note that sequence titles generated by `parse_interproscan.pl` contain information about the domains present within each protein sequence:

```
>Hhongkongensis_21824 LengthOf_303aa_1X_cd00934
```
or
```
>Hhongkongensis_5066 LengthOf_759aa_1X_cd11862_AND_2X_cd00992
```

In these two examples, the length of the input sequence is shown in the definition line (`303aa` and `759aa`, respectively). In the first case, one occurrence of the `cd00934` was found, as indicated by the `1X` preceding the CDD accession number. The second protein has two domains (`cd11862` and `cd00992`), with one occurrence of the first domain (`1X`) and two occurrences of the second (`2X`).