Prose manual

Comparative Network Biology
klpoe@psb.vib-ugent.be
VIB center for Plant Systems Biology
Ghent University - Belgium

I. Introduction

The Curse suite enables quick and easy construction of expression atlases and co-expression networks from publicly available RNA-Sequencing (RNA-Seq) experiments. It consists of a web tool named Curse (Curator of Sequencing Experiments) and a portable pipeline named Prose (Processor of RNA-Sequencing Experiments). With Curse, users can browse studies hosted on the Sequence Read Archive (SRA) in order to select relevant RNA-Seq experiments for their compendium. A Prose package can then be downloaded to retrieve and process the raw sequencing data against user-provided transcript sequences. The end result consists of transcript- and gene-level expression atlases for each experiment in the compendium, with the option to construct co-expression networks as well. Curse and Prose are available at:
http://bioinformatics.psb.ugent.be/webtools/Curse

This manual describes the usage and inner workings of Prose. It implements a pipeline to download raw RNA-Seq data from SRA, apply quality control to it, and perform expression quantification. It requires transcript sequences provided by the user in FASTA, GFF, or GTF format.

Accessibility

Prose was created with the aim to make RNA-Seq data processing accessible and easy for anyone with a working computer. It features a lightweight pipeline, does not require the installation of any third-party software besides java, and running it can be as easy as executing two simple commands. Prose can be started from command line for people who know their way around a console, or can be run interactively and it will ask you everything it needs to get started. Pipeline options can be tweaked if need be, but the default settings work well enough to generate expression atlases that can rival the state-of-the-art.

Processing data for large compendia can be time consuming. To make it possible to process the data piece by piece, Prose can be safely stopped and restarted at any time. After restarting, it will check its previous progress, verify the integrity of the intermediary files, and then simply pick up where it left off.

Pipeline tools

Prose uses *prefetch* and *fastq-dump* from the SRA toolkit (v2.9.0) to download sequencing data, FastQC (v0.11.7) to detect adapter sequences, Trimmomatic (v0.38) to perform adapter clipping and quality trimming, and Kallisto (v0.44.0) for expression quantification. Prose manages and downloads all required executables and dependencies of these tools automatically, so no manual installation is required.

Java

Prose was written in the Java programming language to make it portable and compatible with most operating systems. Thanks to Java, Prose can be run on Windows, Linux, and MacOS. A requirement is that Java needs the correct runtime environment (jre 8) to be available on your system:
http://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html
Please make sure that the correct jre for your operating system is installed.

The prose package

Once a compendium has been constructed in Curse, a 'Prose package' can be downloaded from the Project page to process the RNA-Seq data (see Curse manual). This package is a zipped directory that needs to be extracted first. The extracted directory forms the base directory of Prose and it will be used to store all input-, intermediary-, and output files. To prevent loss of progress, do not modify, rename, or remove anything from this directory unless instructed to by this manual. If Prose encounters missing or erroneous files while running, it will erase the progress of the affected experiments and will restart their processing to avoid truncated output.

The most important files in the Prose package are:

- *Prepare.jar* Required to prepare transcript sequences
- *Prose.jar* Required to run Prose
- *metadata.tsv* Tab-separated file with experiment metadata
- *options.tsv* Tab-separated file with options for the pipeline

Interactive runs can be started with the following scripts:

- *run_prepare_windows.bat* To run the Prepare tool interactively in Windows
- *run_prepare_linux-macos.sh* To run the Prepare tool interactively in Linux or MacOS
- *run_prose_windows.bat* To run Prose interactively in Windows
- *run_prose_linux-macos.sh* To run Prose interactively in Linux or MacOS

During processing, several directories will be created as well:

- *output/* Stores individual experiment output as well as the final atlases and networks
- *reference/* Stores prepared transcriptomes and Kallisto indexes
- *tmp/* Stores intermediary processing data
- *tools/* Stores the executables of the tools in the pipeline

Both the Prepare tool and Prose will consider the directory their JAR files are located in as the Prose package. Missing files and directories are created by these tools automatically.

II. Quick start

If you have your transcript sequences ready and have extracted and navigated to the Prose package, you can get the processing started in two steps through command line:

```
### Prepare your transcript sequences
java –jar Prepare.jar –t <path_to_transcript_fasta>
### Run Prose
java –jar Prose.jar
```

This will prepare your transcript sequences and start processing all experiments in the metadata.tsv file. See Preparing transcriptomes on how to provide transcripts in GTF or GFF formats instead. Prose execution can be halted at any time by killing the process and resumed by repeating the second command. An 'output' directory will be created inside the Prose package, which will contain logs and output of each individual experiment. After completion, these individual files will be combined into gene- and transcript-level expression atlases, co-expression networks, and a single log file containing processing information (see Prose output).

The next chapters explain these two steps in more detail and describe how to tweak the options and run the two steps interactively.

III. Preparing transcriptomes

Prose uses Kallisto to perform expression quantification. It is a lightweight and fast tool that performs pseudo-alignment of RNA-Seq reads against a set of transcript sequences, referred to here as a 'transcriptome'. The first step in running Prose is preparing this transcriptome, which is done with the Prepare tool included in the Prose package. It is a simple conversion tool that takes transcriptomes in FASTA, GFF, or GTF format and generates a new transcript FASTA file and a text file that maps gene ids to transcript ids. These files are required by Prose to build indexes for Kallisto and are stored in the 'reference' directory.

Transcriptome content

In classical RNA-Seq processing, there is an alignment step where tools such as TopHat2 or STAR are used to perform 'spliced alignment' of reads against a genome sequence. 'Spliced' means that introns are still present in the reference sequence and that reads spanning these introns should be spliced as well in order to align them correctly. Kallisto is an 'alignment-free' tool and is not capable of performing spliced alignments. It requires transcript sequences without introns and should be provided with cDNA sequences.

Since many multi-exon genes are known to undergo alternative splicing, a single gene can have several transcript sequences. Even if you are only interested in gene-level analyses, it is recommended to provide all known transcript sequences to Prose. Prose will perform expression quantification at the transcript-level, but can aggregate these to gene-level expression atlases and co-expression networks (see Prose output). Alternatively, you can provide Prose with only the longest transcript sequence for each gene, but this may reduce quantification accuracy.

Input formats

The transcriptome can be provided in three different formats: FASTA, GFF3, or GTF. Input in GFF3 and GTF format requires a genome FASTA, which contains the chromosome sequences. Make sure your input files follow the correct format standards. Files downloaded from ensemblgenomes.org should all be compatible with Prose. For GFF3 and GTF formats, the Prepare tool will extract only the exons and will create correctly spliced sequences for all transcripts of a gene.

Species-specific transcriptomes

When running the Prepare tool, you can optionally specify the species name of the prepared transcriptome. If a species name is provided, experiments with a matching name in the 'species' column of the metadata.tsv file will be processed with this transcriptome. This name matching is case insensitive. If no species name is provided while preparing a transcriptome, it will become the default and will be used to process all experiments that do not have a matching species-specific transcriptome. This feature of Prose can be useful in case your compendium consists of multiple organisms but can be safely omitted in all other cases. See Prose output for more information on how output for different transcriptomes is generated. The log files will tell you with which transcriptome each experiment was processed.

Running the Prepare tool

Preparing transcriptomes through command line can be done with the Prepare.jar executable. When you are in the Prose base directory, you can run it as follows:

```
java -jar Prepare.jar
```

Running it without arguments or with '–h' or '–help' arguments will print a help text.

Providing a reference transcriptome can be done as follows:

```
### Transcript FASTA:
java -jar Prepare.jar -t <transcript_fasta>
### GTF file:
java -jar Prepare.jar -g <genome_fasta> --gtf <gtf_file>
### GFF file:
java -jar Prepare.jar -g <genome_fasta> --gff <gff_file>
```

A species name can be provided with the '–s' option (see Species-specific transcriptomes). Replace spaces with underscores or place the name between quotes. For example:

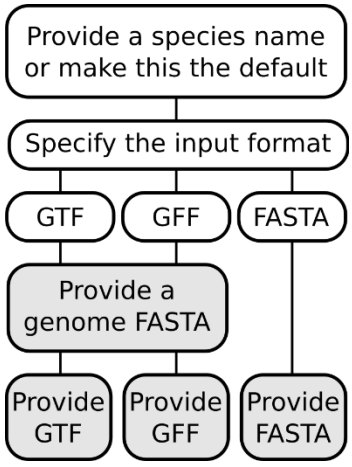
```
java –jar Prepare.jar –s Homo_sapiens –t <transcript_fasta>
### Or
java –jar Prepare.jar –s "Homo sapiens" -t <transcript_fasta>
```

Interactive mode

The Prepare tool can be started interactively through command line with the '-i' option:

```
java –jar Prepare.jar –i
```

Or by executing **run_prepare_windows.bat** or **run_prepare_linux-macos.sh**, depending on your operating system. It will open a command prompt to guide you through the following steps:



To provide files, a file chooser will be opened. If this fails, you will be asked to type out the path to the required file instead.

Details

Running the Prepare tool will create two new files in the reference directory in the Prose package. One will contain the transcript sequences in FASTA format. The other will be a two-column file that maps gene ids (column 1) to transcript ids (column 2). The names of these files indicate to which species they belong. If no species name is provided, the files will be named 'default_transcripts.fas' and 'default_transcript_map.txt'. If you e.g. provided "Homo sapiens" as species name, the files will be named 'homo_sapiens_transcript.fas' and 'homo_sapiens_transcript_map.txt'. Prose uses these file names to match the transcriptomes to the correct experiments. Removing or renaming these files will undo the transcriptome preparation step.

The Prepare tool uses information in the GFF3, GTF, or FASTA file to determine which transcript belongs to which gene. For GFF3 files, exon entries require a 'Parent' attribute while transcript entries require a 'Parent' and 'ID' attribute. For GTF files, exon entries require a 'transcript_id' and 'gene_id' attribute. For FASTA files, a 'gene=<gene_name>' or 'gene:<gene_name>' key-value pair is required in the sequence headers. Prepare will still finish successfully if these attributes are missing, but Prose will later be unable to generate gene-level expression atlases correctly.

It is recommended to filter out any unwanted exon features from the input files beforehand (e.g. snoRNA, transposable elements, …). Any feature with exons will otherwise end up in your atlas. If you encounter any difficulties with entry through the GFF or GTF format, you can try constructing a transcript FASTA file using alternative tools such as gffread or bedtools and use the resulting transcript FASTA as input for the Prepare tool.

IV. Processing experiments

Once your transcriptome is prepared, you can start processing experiments with Prose. To process an experiment, Prose only needs an experiment alias, the accession numbers of the runs, and optionally a species name. This information is listed for all your experiments in the metadata.tsv file, which can be opened and edited in Excel, OpenOffice Calc, or a text editor. Only the 'experimentAlias', 'runAccessions', and 'species' columns in this file are used by Prose. After editing, always save this file in the original TSV format.

While an experiment is being processed, intermediary files are stored in a folder named after the experiment alias in the 'tmp' directory. After each step in the pipeline, processing information will be written to an individual log file in the 'individual_logs' folder inside the 'output' directory. After processing, the Kallisto output will be stored per experiment in the 'individual_output' folder inside the 'output' directory. When Prose is started, it will first check the progress of the experiments. It will see which experiments were finished, which were interrupted during processing, and which were not yet started. For this, it will read the individual log file and verify this information against intermediate or output files. If the individual log file is missing, or if the information it contains does not match the output or intermediate files, it will erase all progress of the experiment and start processing it again. To avoid loss of progress, do not move or rename any of these files or folders while Prose is running or if you intend to restart Prose at a later time.

Running Prose

To run Prose, simply execute the following command:

```
java –jar Prose.jar
```

If you prepared your transcriptome correctly and the metadata.tsv file is intact, no further input will be required by the user. Prose will go over all experiments in the metadata.tsv file, process them individually, and compile their results in expression atlases and co-expression networks. Information about the processing will be written to a file named 'prose_log.txt' in the Prose package. Check it regularly for errors. Running Prose with the '-h' or'—help' options will print a help text.

Processing individual experiments

Prose can also be used to process a single experiment. This can be an experiment in you metadata.tsv file, a new experiment hosted on SRA, or an experiment for which you have FASTQ files locally on your system. If you want to process a single experiment from the metadata.tsv file, simply provide Prose with the experiment alias:

```
java –jar Prose.jar –a <experiment_alias>
```

Prose will exit on an error if this alias was not found in your metadata.tsv file.

If you want to process an experiment from SRA that is not in your metadata.tsv file, you need to provide an alias and the run accessions:

```
java -jar Prose.jar -a <experiment_alias> -r <run_accessions> (-s <species_name>)
```

The run accessions should be provided in a comma-separated list without spaces. For example: '-r SRR847505,SRR847506'. With the –s option, a species name can be provided that should match one of your prepared transcriptomes. If no species name is provided, the experiment will be processed with the default transcriptome (see Species-specific transcriptomes). The alias should not match any experiment in your metadata.tsv file, otherwise you risk confusing this new experiment with the one described in your metadata.tsv file.

To provide local FASTQ files from a single experiment for processing, use the following command:

```
### Single-end reads:
java -jar Prose.jar -a <alias> -1 <forward_fastq_file>
### Paired-end reads:
java -jar Prose.jar -a <alias> -1 <forward_fastq_file> -2 <reverse_fastq_file>
```

If you have multiple forward (and reverse) FASTQ files, you can provide them in a comma-separated list without spaces. Again, an optional species name can be provided with the –s option. The alias should also not match any experiment in your metadata.tsv file.

Command-line options

The following options can be passed to Prose through command line:

-h|--help

Prints a detailed help text that explains how to run Prose through the command line.

--disable-clipping

With this option, Prose will skip adapter detection and clipping with FastQC and Trimmomatic respectively.

--disable-trimming

With this option, Prose will skip quality trimming with Trimmomatic. If adapter clipping is disabled as well, FastQC and Trimmomatic will be entirely skipped in the pipeline.

-t

Use this option to specify the number of threads to run Prose on. This can speed up processing by allowing it to process experiments in parallel. (see Computational cost: Multithreading).

--compile

With the –compile option, Prose will not process any new experiments. Instead, it will compile the expression atlases and co-expression networks for all experiments that were already completed (see Prose output).

The options for each tool in the pipeline can be viewed and edited in the options.tsv file. Run Prose with the –h|--help option or see 'Pipeline options' in this manual for more information.

Interactive mode

Like the Prepare tool, Prose can be run interactively by executing the **run_prose_windows.bat** or **run_prose_linux-macos.sh** script, depending on your operating system. Alternatively, an interactive session can be started through command-line with the '–i' option:

```
java –jar Prose.jar –i
```

Prose will guide you through a series of questions before starting processing and will give you a file chooser if you want to process FASTQ files. All the options available for command-line execution will be prompted through the interactive mode as well.

The pipeline

For each experiment, Prose will go through four steps to produce transcript-level expression values as raw pseudocounts and normalized TPM values:

Step 1: Downloading

Using *prefetch* from the SRA toolkit, Prose will download all runs of an experiment. The downloaded files will be in the SRA format, which is a compressed binary format. Prose then uses *fastq-dump* from the SRA toolkit to extract the sequencing data and store them in gzipped FASTQ files. Data from all runs is concatenated into a single FASTQ file in case of single-end reads, or in one forward and one reverse FASTQ file in case of paired-end reads.

Step 2: Searching for adapter sequences

Prose uses FastQC to search for overrepresented sequences in the FASTQ files. If it finds sequences that are recognized as adapters, it will store them for the next step. For paired-end reads, any adapter found in the forward or reverse reads is reported. Only the adapters known by FastQC can be detected. This list of known adapters can be added to and is located in:
<prose_package>/tools/<operating_system>/FastQC/Configuration/adapter_list.txt
This file will only exist if Prose has already been run once and it has downloaded the tool data.

Step 3: Adapter clipping & Quality trimming

Prose uses Trimmomatic to perform adapter clipping and quality trimming. It will also remove reads that become too short after these procedures. Refer to the Trimmomatic manual for a more detailed explanation of each step. Adapter clipping and quality trimming can be disabled with command-line arguments or through prompts in the interactive mode of Prose. Three features of Trimmomatic are used and their options can be viewed and changed in the options.tsv file:

ILLUMINACLIP:<adapter_file>:<seed_mismatches>:<palindrome_threshold>:<simple_threshold>

The adapter file is the file created in Step 2 and will contain the adapters detected by FastQC and their reverse complements. For paired-end sequencing, adapters found in the forward FASTQ file will be clipped from the reverse FASTQ file as well and vice-versa. Seed mismatches indicate how many mismatches are allowed in the alignment. The simple threshold determines how accurate the match must be. All adapters are given to Trimmomatic as 'simple' adapters and so the palindrome threshold is not used by Prose.

SLIDINGWINDOW:<window_size>:<required_quality>

The sliding window approach of Trimmomatic is used to trim low quality nucleotides from the end of the reads. If the average quality of nucleotides inside the window fall below a required quality threshold, the read is trimmed.

MINLEN:<length>

All reads that are shorter than the specified length will be removed. In the case of paired-end reads, both reads in a pair will be removed if at least one of the reads fails this threshold.

Step 4: Expression quantification

Prose uses Kallisto to quantify expression at the transcript level. Kallisto is an alignment-free tool, which means that it does not perform a classical alignment of the reads to the transcript sequences but rather aligns reads to a graph of k-mers. To do this, Kallisto needs to build an index of each transcriptome. This will be done by Prose before it starts processing any experiments. The only option required for the indexing step is the k-mer size, which can be edited in the options.tsv file. By default, this is set at 31. Important to note is that the k-mer size should be smaller than the read length. Be sure to keep the MINLEN option of Trimmomatic at least one nucleotide larger than the k-mer size if you choose to edit these options.

To quantify expression, Kallisto needs an estimation of the fragment length. This is the length of the cDNA fragments that were sequenced. For paired-end reads, Kallisto can estimate this length by itself but for single-end reads, it requires input from the user. For public data on SRA, this fragment length is rarely included in the metadata and it does not have a dedicated metadata field, making it impossible to find automatically. However, most sequencing protocols produce fragments of more or less the same length. By default, Prose uses a fragment length of 200 nucleotides with a standard deviation of 20. This option can be changed in the options.tsv file.

Kallisto produces two expression values for each transcript: pseudocounts and TPM values. Pseudocounts represent an estimate of how many reads were aligned to each transcript. TPM or "Transcript Per Million" values are measures that normalize these pseudocounts for the transcript lengths and sequencing depth of the experiment. The sum of the TPM values over all the transcripts should be the same in all experiments. Please read this article from the RNA-Seq blog for a more detailed explanation.

Even though Kallisto can process stranded RNA-Seq data, Prose will process it as if it was unstranded. The first reason for this is that stranded and unstranded runs cannot be distinguished automatically based on the metadata from SRA. The second reason is that co-expression signals can get disrupted if an atlas consists of mixed stranded and unstranded experiments.

Pipeline options

Options used for each step of the pipeline can be viewed and edited in the options.tsv file. These options are read in by Prose when it starts. Options for co-expression network creation are included in the options.tsv file, but are described later on (see Prose output: Co-expression networks).

---

trimmomatic_minlen

The minimum length of reads after clipping/trimming. If a read becomes shorter than this, it will be discarded. This value should be larger than the k-mer length. See MINLEN in the Trimmomatic manual. (default = 32)

trimmomatic_illuminaclip_seedMismatches

The minimum number of mismatches between a read and the seed of the adapter to consider it for clipping. See the 'seed mismatches' option of ILLUMINACLIP in the Trimmomatic manual. (default = 2)

trimmomatic_illuminaclip_simpleClipThreshold

The alignment score threshold between a read and the adapter sequence to consider it for clipping. See the 'simpleClipThreshold' option of ILLUMINACLIP in the Trimmomatic manual. Palindrome clipping is not used by Prose. Each detected adapter is clipped from forward and reverse reads with the simple clip threshold. (default = 10)

trimmomatic_slidingwindow_windowSize

The size of the sliding window to evaluate read quality for trimming. See the 'windowSize' option of SLIDINGWINDOW in the Trimmomatic manual. (default = 4)

trimmomatic_slidingwindow_requiredQuality

The minimum average nucleotide quality in the sliding window to evaluate it for trimming. See the 'requiredQuality' option of SLIDINGWINDOW in the Trimmomatic manual. (default = 15)

kallisto_k-mer_size

The k-mer size to use when indexing the reference transcriptome. A rule of thumb is that this value should be shorter than the read length. For transcriptomes with many similar transcripts, longer k-mer sizes might improve quantifications. (default = 31)

kallisto_fragmentLength_mean

The average length of the sequenced fragment. For paired-end reads, kallisto estimates these based on the distance between paired reads. For single-end reads, it requires estimates from the user. For most purposes, an average fragment length of 200 will work fine. (default = 200).

kallisto_fragmentLength_sd

The standard deviation on the length of the sequenced fragment. (default = 20)

retry

By default, Prose will skip experiments for which one of the tools in the pipeline fails. If you would like Prose to retry processing of failed experiments, set this option to a value larger than 1. (default = 1)

max_download_threads

The maximum number of threads that can be downloading at the same time. Unlike CPU, bandwidth is shared among all threads. Running multiple downloads will slow down the download speed for each individual thread. (default = 1)

Prose output

After Prose finishes processing of all experiments, or if it is run with the –compile option, it will generate transcript- and gene-level atlases for all successfully processed experiments and, if enabled, will generate co-expression networks. Experiments that are not in the metadata.tsv file but were processed individually by the user will be included in the final results.

Expression atlases

The expression atlases are tab-separated files with experiment aliases as columns, genes or transcripts as rows, and pseudocounts or TPM as values. Gene-level atlases are created by summing the pseudocounts or TPM values of all their transcripts. Experiments processed with different transcriptomes will be placed in separate atlas files. These files will be named after the species name provided when preparing the transcriptome, followed by the date of completion and a suffix describing the content:

output/<species_name>_<date>_transcript_counts.tsv	transcript-level pseudocounts
output/<species_name>_<date>_transcript_tpm.tsv	transcript-level TPM values
output/<species_name>_<date>_gene _counts.tsv	gene-level pseudocounts
output/<species_name>_<date>_gene _tpm.tsv	gene-level TPM values

For the default transcriptome, file name will start with ‘default’.

Experiments for which processing failed or was not yet completed will not be included in the atlases. Check the prose_log.txt file for any errors and check the atlas files to make sure your final output is complete. Should processing of any experiments have failed, you can simply restart Prose and it will try to process missing experiments again. Increase the 'retry' option in the options.tsv file if you want Prose to make multiple attempts each time a pipeline step fails. Besides atlas files, Prose will also generate a final log file containing processing information. This will be a tab-separated file named after the date of completion ('31-07-18_log.tsv'). It will contain the following columns of information for each processed experiment:

alias	The experiment alias
species index	The transcriptome index used to process this experiment
layout	Single-end or paired-end
raw_forward_reads	The number of downloaded forward reads
raw_reverse_reads	The number of downloaded reverse reads (0 for single-end reads)
qc_clip_enabled	'true' if adapter clipping was enabled
qc_trim_enabled	'true' if quality trimming was enabled
adapters_found	Number of adapters found by FastQC and clipped by Trimmomatic
clean_forward_reads	Number of forward reads left after Trimmomatic was run
clean_reverse_reads	Number of reverse reads left after Trimmomatic was run
reads_mapped	Number of reads successfully mapped by Kallisto
reads_unique	Number of reads mapped uniquely by Kallisto

Always check this log file to spot any bad samples in your atlas. Low values for 'clean_forward_reads' or 'reads_mapped' indicate problems with the read quality or alignment respectively. Also check if the species index used is the correct one for each experiment.

Co-expression networks

If the expression of two genes increases and decreases simultaneously across multiple samples in the atlas, these genes are said to be co-expressed. Co-expression between genes can indicate a functional relationship. Co-expression networks, where genes are the nodes and an edge indicates co-expression between two genes, are often used in functional analyses. Prose offers two basic methods to infer co-expression networks from your atlases: k-Nearest Neighbor (knn) and Highest Reciprocal Rank (hrr). Both methods use the Pearson correlation coefficient (pcc) between the TPM expression profiles of two genes to quantify co-expression. Larger pcc values indicate that the expression profiles of two genes are more similar.

The knn method simply returns the top 'k' co-expressing genes for each query gene, where 'k' is a positive integer. As a network, it can be seen as a directed graph where 'k' edges depart from each gene. The hrr method is an extension of knn, where only reciprocal edges in the knn network are kept as undirected edges. In other words, only edges between genes that are in each other's top 'k' co-expressing genes are kept.

When hrr or knn network building is enabled, Prose will generate one of four possible output files containing a knn or hrr network for genes or transcripts:

output/<species_name>_<date>_gene -level_knn<k>.tsv
output/<species_name>_<date>_gene -level_hrr<k>.tsv

output/<species_name>_<date>_transcript -level_knn<k>.tsv

output/<species_name>_<date>_transcript-level_hrr<k>.tsv

This file will contain four columns: (1) the query gene, (2) the connected gene, (3) the co-expression rank, and (4) the pcc value. Which file Prose will create is dictated by the options.tsv file.

The following options in the options.tsv file relate to network building:

hrr_threshold

By default, creating co-expression networks is disabled in Prose. To enable it, set this to a value larger than 0. This value will be the 'k' value in knn or hrr. A recommended value is 100. Hint: Leave this option disabled until all experiments are processed, then enabled it and use the –compile option to create networks. (default = 0)

hrr_minimum_expression

Prose sets a TPM expression value threshold on all genes. If a gene does not have an expression value larger than this in any of the experiments, it will be excluded from the hrr/knn network. (default = 2.0)

hrr_for_transcripts_instead_of_genes

By default, knn/hrr networks are built at the gene level. Set this option to 1 to create transcript-level networks instead. (default = 0)

knn_instead_of_hrr

By default, Prose will generate hrr networks. Set this to 1 to generate knn networks instead. (default = 0)

Computational cost

Prose was built to be as efficient as possible with computational resources. It is lightweight enough to run on an average laptop, but can be deployed in parallel on a cluster system just as easily.

Memory

The amount of RAM memory a Prose thread requires depends on the size of the transcriptome you are processing and the size of the sequencing data. For most experiments, 2G of memory per thread should suffice. Large experiments on large transcriptomes may however require up to 4G of memory. When running prose on e.g. four threads, the memory usage may spike up to four times that amount.

Network

The most time-consuming and vulnerable step of the Prose pipeline is downloading the sequencing data from SRA. Compressed SRA files range from a few hundred megabytes to tens of gigabytes per run and so, depending on your internet connection, these files may take some time to download. When multithreading is enabled multiple SRA files will be downloaded in parallel, but because these threads will compete for the same bandwidth they will slow each other down.

You may experience closed connections to the SRA server while downloading. This occurs if the SRA server is under heavy load and will be more problematic for larger files. If you experience a lot of failed downloads (check the 'prose_log.txt' file), try disabling multi-threading or run Prose again at a later time. If Prose fails to download a specific experiment, it will skip it and try the next one. Increase the 'retry' option in the options.tsv file if you want Prose to make multiple attempts.

Multithreading

The pipeline roughly consists of a download step and a processing step. Downloading only requires network bandwidth while processing is CPU intensive. To improve the pipeline's speed, these two steps will occur in parallel as much as possible: while one thread is downloading, another will be processing data. The maximum number of threads that can perform each of these tasks can be changed. Increasing the number of processing threads with the –t option often improves the overall speed of the pipeline, since they can each use a separate CPU. The maximum number of downloading threads can be increased with the max_download_threads option in the options.tsv file, but since these threads will have to share the same bandwith, this is unlikely to improve the pipeline's speed. Setting either of these options to zero will disable the parallelized downloading and processing.

On the cluster

Prose can be run in parallel on a cluster system. You can either run one instance of Prose with multithreading enabled, or you can run multiple instances of Prose in parallel using the –a option to specify which experiment from the metadata.tsv file to process. On most cluster systems, you will need to add additional Java options to your command:

```
java –jar  -XX:ParallelGCThreads=1 -Xms256m -Xmx1024m Prose.jar
```