Command Line Scripts:

Trinity

Due to memory and processing requirements, assembly processing was conducted using the UTS eResearch ARCLab (https://clusterportal.feit.uts.edu.au) which contains virtual computers running Linux-based operating systems with hardware specifications of: 144 GB RAM, 1TB HD and 28 cores with multi-threading enabled. The following command was used during assembly, a description of each command and constraint can be found by referring to Table S1.

./Trinity --seqType fq --max_memory 100G --left leftread.fastq --right rightread.fastq --CPU 28 --SS_lib_type RF --normalize_reads --output trinity_output/

Table 2.2 Trinity assembly command information with corresponding constraint and meaning.

Command	Constraint and meaning
--seqType fq	Input files are fq files containing sequences and quality scores
--max_memory 100G	Maximum memory that can be used by this process cannot exceed 100GB at any time.
--left/--right	Input names of forward and reverse raw read files
--CPU 28	Maximum processing of up to 28 cores can be utilised at any time.
--SS_lib_type RF	Indicates that the strand specific library type is double stranded (R= Reverse, F=Forward)
--normalize_reads	Data normalisation will be performed during assembly using in silico normalisation built into the software to record coverage levels of unique coverage levels.

Before proceeding, TPM estimations were calculated for each ant species using a perl script, 'align_and_estimate', included in the Trinity software package. This process compares the assembled .fasta file to the original raw, forward and reverse, FastQ files. The command used for this step was:

./align_and_estimate_abundance.pl --transcripts inputname.fasta --est_method RSEM --aln_method bowtie2 --trinity_mode --prep_reference

A description of each command and constraint can be found by referring to Table 2.3. The output file, from TPM estimation, resulted in a .txt file with information of each Trinity ID number, estimated abundance, and actual abundance.

Table S2: Trinity abundance estimation command information with corresponding constraint and meaning

Command	Constraint and meaning
--transcripts	Name of assembled fasta file
--est_method	Alignment estimation method. RSEM uses an inbuilt software called eXpress and is an alignment based method for predicting transcript count
--aln_method	Alignment method. Bowtie2 is an external software that is recommended to be used with RSEM.
--trinity_mode	Creates a gene_trans_map file in order for alignment to differentiate between isoforms.
--prep_reference	Builds a target index from the input files

Bowtie

bowtie2 --local -p 5 --no-unal -x bowtie2_indicies/Trinity.fasta -q -1 ../trimmomatic/pp_adapt5_R1_paired_trimmed.fq.gz -2 ../trimmomatic/pp_adapt5_R2_paired_trimmed.fq.gz | samtools view -Sb - | samtools sort -no - - > bowtie2.namesorted.bam

Blastx

The BLASTx command employed is detailed below, and a description of each option can be found in Table 2.5.

blastx -query filename.fasta -db databasename -outfmt "6 qseqid sseqid salltitles pident length slen mismatch gapopen qstart qend sstart send evalue bitscore" -num_alignments 1 -max_hsps 1 -num_threads 28 –out filename.txt

Table S3 BLASTx command information with corresponding constraint and meaning.**[54, 57]**

Command	Constraint and meaning
-query	Fasta file containing sequences for blasting.
-db	Database name that search will be blasting against.
qseqid	Return Sequence ID for match
ssegid	Return Gene ID for match
saltitles	Return all subject titles
pident	Return percentage identity match
length	Return length alignment
slen	Return subject sequence length

mismatch	Return number of mismatches
gapopen	Return number of gap openings
qstart	Return position of start of alignment in query
qend	Return position of end of alignment in query
sstart	Return position of start of alignment in database
send	Return position of end of alignment in database
evalue	Return e-value score
bitscore	Return bitscore
-num_alignments	Return the number of alignments of the query sequence to the best match
-max_hsps	Setting is 1 instructing BLASTx to return only the single best match.
-num_threads	Maximum number of CPU's/cores to be used at any time
-out	Output filename

The first step in annotating the transcriptomes was that the BLASTx package and the entirety of the NCBI NR database (100 GB) was downloaded to a local server. Subsequently, a command was run to unzip the data and format the database into a nucleotide library using an inbuilt 'makeblastdb' perl script. In order to process all data in a reasonable timeframe, the .fasta Trinity output file was split into 10–20 parts and allowed to run in parallel on different servers to reduce run time. The following command was used to split the file into distinct parts directly after the final nucleotide of a particular contig:

awk 'BEGIN {n_seq=0;} /^>/ {if(n_seq%1000==0){file=sprintf("outputfilename%d.fasta",n_seq);} print >> file; n_seq++; next;} { print >> file; }' < inputfilename.fasta

A typical file would contain 120,000 lines of information and would run on a single server using 28 cores and 240 GB of RAM and require 3–4 days of continual processing till completion. After completion, each group of fasta files were combined into a master file using a 'cat' command.