Combining short- and long-read transcriptomes for targeted enzyme discovery

Mojca Juteršek[1], Marko Petek[1], Špela Baebler[1]

[1] National Institute of Biology, Department of Biotechnology and Systems Biology, Večna pot 121, SI-1000, Ljubljana, Slovenia

Abstract

The discovery of genes that code for a specific enzymatic activity is important in various fields of life science and provides valuable biotechnological tools. Many genes that contribute to the production of secondary metabolites and specialised metabolic pathways are still not identified. Due to the great diversity of metabolic functions found in nature and their rapid evolutionary adaptation, we need precise but high-throughput approaches for a targeted search based on minimal prior knowledge. In this chapter, we describe a transcriptomics pipeline that was used to search for candidate genes coding for a specific enzymatic activity in a non-model species. We generated and combined short- and long-read transcriptomic data to obtain reliable full-length transcript sequences along with information on allelic variation, isoform expression, and condition-specific expression. Based on protein domain annotations of coding sequences and transcriptomic data, we selected candidate genes for activity assays. We provide detailed instructions for analysis and quality control steps in our pipeline that can be applied to other biological questions.

Keywords

transcriptomics, Iso-Seq, *de novo* transcriptome assembly, enzyme identification

1 Introduction

All branches of life feature a diverse set of metabolic reactions and the resulting metabolites. Apart from relatively conserved and ubiquitous primary metabolism, which provides vital functions, the more diversified secondary metabolism enables niche adaptation to various environmental conditions and is a medium of intra- and inter-organismal communication. Discovery and characterisation of such an immense range of biological functionalities is an important endeavour, albeit laborious. It contributes knowledge, biological molecules, and

metabolites with applications in agriculture, pharmacy, chemical industry, and environmental protection, among other areas [1, 2].

Selection of the approach for the discovery of metabolic pathways mostly depends on the extent of prior knowledge and our aim. Searching for a specific metabolic functionality in an organism of interest is for example conceptually different to the discovery of novel enzyme families in uncharacterised communities [3, 4]. However, all approaches can employ high-throughput technologies and computational tools designed to mine large amounts of biological data for biological discovery [5–7]. One of the most developed areas of high-throughput biological research is transcriptomics, operating through different sampling approaches and a variety of RNA isolation methods, followed by parallel sequencing of short or long fragments and analysis of obtained sequences [8, 9]. Unlike genomics, it provides dynamic information on gene expression under different conditions and is therefore indispensable in functional genomics. Its methods still outperform the available toolbox for proteomics in terms of completeness and the amount of data obtained per cost unit.

Gene expression analysis can be particularly valuable for targeted discovery of secondary metabolism pathways. By its nature, secondary metabolism is responsive to various conditions or can be specific to different developmental stages, sex, and other biological determinants. In the case of a targeted search for a specific metabolic activity, transcriptomic data gathered in the desired biological context provides enriched information on gene activity, which usually indicates the potential functional involvement of detected genes [4]. This approach can benefit from short-read and long-read transcriptome sequencing. High-throughput long-read sequencing platforms, such as PacBio (PB) and Oxford Nanopore Technologies (ONT), produce reads that cover the full length of almost all eukaryotic transcripts [10]. Determining full-length transcript sequences enables target gene cloning and expression, as well as the design of reverse genetics approaches, for example, targeted mutagenesis and gene silencing. It can also provide information on the existence of gene isoforms and their expression, with different isoforms potentially having different functionalities. On the other hand, short-read sequencing still has some advantages over long-read sequencing [11]. It is more accurate at the nucleotide level and provides greater sequencing depth per dollar. The latter can be important for identifying novel sequences expressed at very low levels or only in limited cell types and tissues, as the sensitivity might be too low to detect them with long-read sequencing.

In this chapter, we are showcasing an approach of combining short- and long-read transcriptomics data from an economically important insect pest, citrus mealybug (*Planococcus citri*), to identify coding sequences with potential activity in the biosynthesis of a monoterpenoid sex pheromone [12]. Sex pheromones are applicable for sustainable pest control in agriculture and can be biotechnologically manufactured [13, 14]. Identified *P. citri* genes could therefore be expressed in heterologous systems for biotechnological pheromone production. We provide a comprehensive workflow for analysis, quality control, and assembly of short and long reads, followed by consolidation of the assemblies, functional annotation and search for candidate sequences (**Figure 1**). In our study [12], candidate sequences were cloned and expressed in a bacterial and plant system to test for target enzymatic activity.

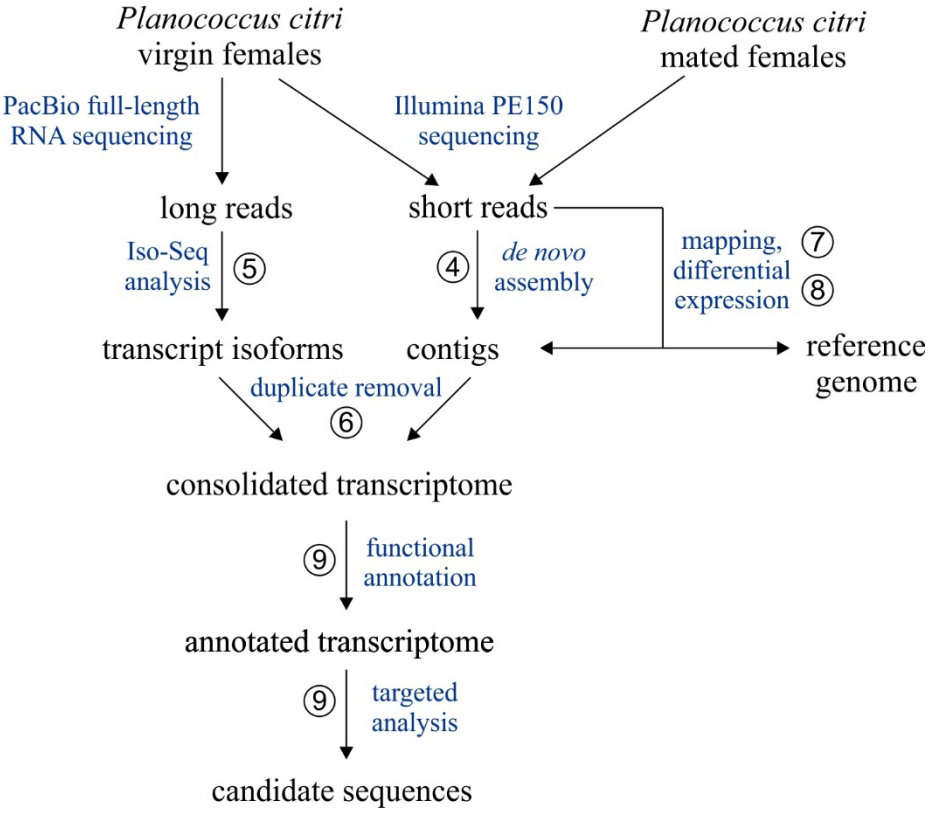

Figure 1: Short- and long-read transcriptomic analysis steps for targeted enzyme search presented in this chapter. The depicted approach was used to identify candidate sequences coding for specific enzymatic activity found in *P. citri* [12]. For each step, the number of the chapter subheading in which it is described in detail is given in a circle.

2 Experimental design

To obtain candidate sequences coding for target activity, it is important to carefully design the experiment. The first consideration should be the biological material used for RNA isolation. The selection of target tissue and sampling conditions can be guided by prior knowledge of tissue and condition-specific occurrence of biological phenomena under study. In our example, we intended to identify enzymes involved in *P. citri* sex pheromone biosynthesis. Therefore, we decided to sample and isolate RNA from *P. citri* females before mating (virgin females), when the pheromone is produced by females to attract males for mating, whereas its production stops after mating [15]. It is reasonable to assume that the expression of sex pheromone biosynthesis genes would be highest in adult virgin females before mating. Additionally, we decided to collect a contrasting sample from mated females, enabling us to perform differential gene expression analysis comparing read counts from samples of virgin and mated females and by that focus the search on genes with higher expression in virgin *P. citri* females. For other systems of interest, one could similarly consider different sampling parameters, such as the sex of specimens, time of day, exposure to different stress conditions, nutrient availability, developmental stage, and others. In our case, we also considered dissecting and sampling sex pheromone gland tissues, however, unlike for some other insect taxa [16, 17], in *P. citri* their anatomical position is unknown therefore we sampled whole organisms.

Another consideration of the experimental setup should be the choice of sequencing approaches and sequencing depth. Since we were searching for a unique enzymatic activity, presumably recently evolved and not well understood, we wanted to perform a broad search and obtain information on sequence variation, such as expression of isoforms, allelic variation and presence of single nucleotide polymorphisms (SNPs). We also aimed to clone and express candidate sequences, therefore we required reliable full-length sequences that would support successful cDNA amplification or synthesis. Additionally, we hypothesized that the sex pheromone biosynthesis genes are expressed in a specific tissue and therefore RNA samples collected from whole organisms might contain only limited copies of target transcripts. To cover all these requirements, we decided on both long- and short-read sequencing, with the first providing full-length isoform sequences and the second contributing high accuracy and sufficient depth.

3 Materials

The presented analysis was done mostly on the Linux operating system with publicly available command-line tools. In the following protocol, we provide and explain commands and parameters important to our use case (see **Note 1**). However, we do not provide detailed instructions for the installation of used tools. We recommend using conda (https://conda.io/projects/conda/en/latest/index.html) for computational environment and package management (i.e. reproducible installation of tools and their dependencies), as most presented tools are available for installation through conda. Below is a list of all software used in the presented pipeline with links to their current GitHub repositories or user guides, which all include installation instructions:

- BBTools v39.05 (https://jgi.doe.gov/data-and-tools/software-tools/bbtools/)
    - BBDuk (https://jgi.doe.gov/data-and-tools/software-tools/bbtools/bb-tools-user-guide/bbduk-guide/)
    - BBMerge (https://jgi.doe.gov/data-and-tools/software-tools/bbtools/bb-tools-user-guide/bbmerge-guide/)
    - Clumpify (https://jgi.doe.gov/data-and-tools/software-tools/bbtools/bb-tools-user-guide/clumpify-guide/)
    - Tadpole (https://jgi.doe.gov/data-and-tools/software-tools/bbtools/bb-tools-user-guide/tadpole-guide/)
- rnaSPAdes [18] (https://gensoft.pasteur.fr/docs/SPAdes/3.14.0/rnaspades_manual.html, https://github.com/ablab/spades)
- rnaQUAST [19] (https://github.com/ablab/rnaquast)
- minimap2 [20] (https://github.com/lh3/minimap2)
- cDNA_cupcake (https://github.com/Magdoll/cDNA_Cupcake)
- samtools [21] (https://www.htslib.org/)
- CD-HIT [22] (https://sites.google.com/view/cd-hit)
- BUSCO [23] (https://busco.ezlab.org/)
- Mmseqs2 [24] (https://github.com/soedinglab/MMseqs2)
- STAR [25] (https://github.com/alexdobin/STAR)
- IGV [26] (https://igv.org/)
- matchAnnot (https://github.com/TomSkelly/MatchAnnot)

- InterProScan [27] (https://interproscan-docs.readthedocs.io/en/latest/index.html)
- R (https://www.r-project.org/)
    - edgeR [28] (https://bioconductor.org/packages/release/bioc/html/edgeR.html)
    - limma [29] (https://bioconductor.org/packages/release/bioc/html/limma.html)
- transDecoder (https://github.com/TransDecoder/TransDecoder)

4 *De novo* transcriptome assembly

Short reads obtained with Illumina sequencing can be used for mapping to reference genome or transcriptome, followed by read counting, differential expression analysis, and SNP analysis or reference-guided transcriptome assembly. If good quality reference is not available, short RNA-Seq reads can be also *de novo* assembled into contigs (see **Note 2**).

We generated short Illumina reads for eight samples – four from virgin and four from mated *P. citri* females (see **Note 3**). If you did not generate sequencing data for your project or want to complement your data, you can use publicly available data sets. Most short-read data is deposited at NCBI's SRA database (https://www.ncbi.nlm.nih.gov/sra) (see **Note 4**).

After you obtained your FASTQ files, create a folder for your analysis and a folder for generated or downloaded short-read files:

```
mkdir /MyAnalysis
mkdir /MyAnalysis/short
mkdir /MyAnalysis/short/raw
```

4.1 Pre-processing short Illumina reads

Before using FASTQ files for *de novo* assembly, it is important to do quality control and pre-processing, including, but not limited to adapter trimming, removing low-quality reads, and removing contaminating reads from other organisms.

You can first concatenate all FASTQ files from the same study into a single file, simplifying downstream processing. Only reads of the same length should be concatenated into one file and if sequencing was done in paired-end mode, FASTQ files with forward and reverse reads should be concatenated into two separate files. Names of files with forward and reverse reads usually include _1 and _2 suffixes, respectively:

```
cd /MyAnalysis/short/raw

cat ./*_1.fastq > ./MyStudyShort_1.fastq
```

```
cat ./*_2.fastq > ./MyStudyShort_2.fastq
```

4.1.1 Quality and adapter trimming

With concatenated FASTQ files, you can proceed to read quality filtering and adapter trimming using BBTools' BBDuk script:

```
cd ..
mkdir ./bbduk_filter
cd ./bbduk_filter

/PathToInstalledTool/bbduk.sh \
      -Xmx230g \
      in=../raw/MyStudyShort_1.fastq \
      in2=../raw/MyStudyShort_2.fastq \
      out=./MyStudy_trimmed.fastq \
      ktrim=r \
      k=23 \
      mink=11 \
      hdist=1 \
      ref=./adapters.fa \
      tbo \
      tpe \
      maxns=0 \
      trimq=20 \
      qtrim=r \
      maq=12
```

We specified forward and reverse read FASTQ files as inputs (`in`, `in2`), memory usage (230 GB with `-Xmx230g`), k-mer-based adapter and contaminant trimming parameters, and quality trimming and filtering parameters. Adapter and contaminant trimming is performed to the right side of reads (`ktrim=r`), using k-mer length of 23 (`k=23`) or 11 at read tips (`mink=11`) and maximum hamming distance of 1 (`hdist=1`). A reference FASTA file with adapter sequences ("adapters.fasta") is provided (`ref`) (see **Note 5**). Trim by overlap (`tbo`) and even pair trimming (`tpe`) are enabled. For quality trimming, regions with average quality below 20 (`trimq=20`) are discarded to the right (`qtrim=r`). The script will also discard reads with average quality below 12 after trimming (`maq=12`) and reads with Ns (`maxns=0`). The output will be a quality- and adapter-trimmed FASTQ file with both forward and reverse reads (path to and output name file specified with `out`). Other known contaminating sequences and sequencing artefacts can be further trimmed in a second round with the trimmed FASTQ file as an input and a list of FASTA files with contaminating sequences (see **Note 5**).

```
/PathToInstalledTool/bbduk.sh \
      -Xmx230g \
      in=./MyStudy_trimmed.fastq \
      out=./MyStudy_filtered.fastq \
      k=31 \
      ref=./sequencing_artefacts.fa,./phix174_ill.ref.fa
```

4.1.2 Error correction

After quality and contaminant trimming, sequencing error correction can be performed. One option for Illumina reads is the Bayes Hammer method [30]. We used it within the rnaSPAdes tool [18], later used for *de novo* assembly (see **Subheading 4.2**), by running it in `--only-error-correction` mode:

```
cd ..
mkdir ./error-corr
cd ./error-corr

/PathToInstalledTool/spades.py \
      --only-error-correction \
      -m 230 \
      -t 32 \
      -o . \
      --pe-12 ../bbduk_filter/MyStudy_filtered.fastq
```

Input is the filtered FASTQ file, for which the type is specified (file with interlaced forward and reverse paired-end reads, `--pe-12`). The command also specifies thread (`-t`) and memory usage (`-m`) and current directory as the output directory (`-o`). Spades error correction will generate two FASTQ files with corrected forward and reverse paired reads, respectively, and a FASTQ file with corrected unpaired reads.

Additional read correction can be performed using overlap-based error correction integrated into the BBTools' BBMerge tool. Several rounds of such correction can be performed. As input, we take the output of Bayes Hammer correction, processing paired and unpaired reads separately.

For paired files:

```
/PathToInstalledTool/bbmerge.sh \
      -Xmx230g \
      in1=./MyStudy_trimmed2_1.00.0_0.cor.fastq \
      in2=./MyStudy_trimmed2_2.00.0_0.cor.fastq \
      out=./MyStudy_ecco_PE.fastq \
      ecco \
      mix \
```

```
      vstrict \
      adapters=default
```

For unpaired files:

```
/PathToInstalledTool/bbmerge.sh \
-Xmx230g \
      in=./MyStudy_trimmed2_unpaired.00.0_0.cor.fastq \
      out=./MyStudy_ecco_SE.fastq \
      ecco \
      mix \
      vstrict \
      adapters=default
```

The parameters used above call for error correction of overlapping parts of paired reads (`ecco`) under very strict conditions (`vstrict`), while outputting both mergeable and unmerged reads (`mix`) in the same output file (`out`). We also specified to consider a list of common adapter sequences (default).

The resulting paired-end FASTQ files for paired-end and unpaired single-end reads can be sorted so that similar reads are positioned near each other in the file with the BBTools' clumpify tool. By using the `ecc` parameter, we also perform error-correction with 6 passes on reads with identity to consensus greater than 0.98 (`minid`):

```
/PathToInstalledTool/clumpify.sh \
      -Xmx200g \
      in=./MyStudy_ecco_PE.fastq \
      out=./MyStudy_eccc_PE.fastq \
      ecc \
      passes=6 \
      minid=0.98

/PathToInstalledTool/clumpify.sh \
      -Xmx200g \
      in=./MyStudy_ecco_SE.fastq \
      out=./MyStudy_eccc_SE.fastq \
      ecc \
      passes=6 \
      minid=0.98
```

An additional round of correction can be performed with BBTools's tadpole tool, using `ecc` parameter to specify k-mer count-based error correction:

```
/PathToInstalledTool/tadpole.sh \
      -Xmx200g \
```

```
      in=./MyStudy_eccc_PE.fastq \
      out=./MyStudy_ecct_PE.fastq \
      ecc


/PathToInstalledTool/tadpole.sh \
      -Xmx200g \
      in=./MyStudy_eccc_SE.fastq \
      out=./MyStudy_ecct_SE.fastq \
      ecc
```

Finally, corrected paired reads can be merged with BBTools' BBMerge tool:

```
/PathToInstalledTool/bbmerge.sh \
      -Xmx200g \
      in=./MyStudy_ecct_PE.fastq \
      out=./MyStudy_merged.fastq \
      outu=./MyStudy_unmerged.fastq \
      rem \
      k=62 \
      extend2=50 \
      adapters=default
```

The output includes both a FASTQ file with merged (`out`) and a FASTQ file with unmerged reads (`outu`). Merging is performed with enabled read extension of up to 50 nucleotides (`extend2`) in cases of failed merge (no overlap). It also restricts merging extended reads for which the predicted insert size after extension does not match the insert size before extension (`rem`).

4.2 *De novo* transcriptome assembly with rnaSPAdes

Polished short reads are ready for *de novo* transcriptome assembly with rnaSPAdes in `--only-assembler` mode. Before the assembly, you can concatenate the merged paired-end reads ("MyStudy_merged.fastq") and single-end corrected reads ("MyStudy_ecct_SE.fastq") into one FASTQ file with single reads:

```
cat ./MyStudy_merged.fastq ./MyStudy_ecct_SE.fastq >
./MyStudy_SE.fastq

cd ..
mkdir ./spades_assembly
cd ./spades_assembly

/PathToInstalledTool/rnaspades.py \
      --only-assembler \
```

```
-k 29,49 \
-m 245 \
-t 32 \
-o . \
--ss-rf \
--pe1-fr \
--pe1-12 ../error-corr/MyStudy_unmerged.fastq
--pe1-s ../error-corr/MyStudy_SE.fastq
```

Using the above command, the contigs will be assembled using k-mers of lengths 29 and 49 (`-k`). We determine the type of sequencing libraries – strand-specific (`--ss-rf`) and type of Illumina reads – paired-end, with a forward-reverse orientation (`--pe1-fr`). We provide two input FASTQ files originating from the same library - one with interlaced reads from a paired-end library (unmerged reads, `--pe1-12`), and one with unpaired reads from a paired-end library (unpaired and merged paired reads, which we concatenated in the previous step, `--pe1-s`). Additionally, we also specified memory and thread usage (`-m, -t`) and output directory (`-o`). Output includes a FASTA file with assembled transcripts ("transcripts.fasta") and two FASTA files with hard-filtered (long and more reliable transcripts with high expression) and soft-filtered transcripts (including also short transcripts with low expression), respectively.

4.3 *De novo* transcriptome quality assessment with rnaQUAST

De novo assembled transcriptome should undergo quality control to assess the completeness and correctness of obtained contigs. If an annotated genome for the species of interest is available, several quality metrics can be calculated at once using the rnaQUAST tool [19]. Its outputs include general metrics (transcript number, average length, N50, …) and reference-related metrics, such as number of transcripts aligning, misaligning or not aligning to the reference, NA50, assembly completeness, and assembly specificity (based on different metrics related to coverage of reference genome coding sequences with mapped sequences).

Prepare a folder for rnaQUAST analysis and a folder for reference genome assembly and annotation files:

```
cd ..
mkdir ./rnaQaust_QC
cd ./rnaQaust_QC
mkdir /MyAnalysis/genome
```

Upload your reference genome data to the "/MyAnalysis/genome" folder and run rnaQUAST:

```
/PathToInstalledTool/rnaQUAST.py \
     --transcripts ../spades_assembly/transcripts.fasta \
     --reference /MyAnalysis/genome/MyGenome.fa \
     --gtf /MyAnalysis/genome/MyGenome.gtf \
     --output_dir . \
     --threads 20
```

rnaQUAST output includes text reports and plots with calculated quality metrics and summary text and PDF files, which include the most important metrics.

5 Processing Iso-Seq long reads

Iso-Seq is a full-length sequencing and analysis method using SMRT sequencing technology and pipelines for isoform discovery. Outputs of the Iso-Seq clustering pipeline are FASTA files with high- and low-quality isoforms. Depending on the purpose of the study, low-quality transcripts can be discarded or included in downstream analyses. In our case, we decided to include low-quality isoforms and we therefore concatenated both isoform FASTA files into one using `cat` command (see **Note 6**).

First, create a folder for the Iso-Seq analysis and put the data provided by the sequencing facility in the "input" subfolder:

```
mkdir /MyAnalysis/long
mkdir /MyAnalysis/long/input
mkdir /MyAnalysis/long/output
cd /MyAnalysis/long

cat ./input/hq_isoforms.fasta ./input/lq_isoforms.fasta >
./input/hq_lq_isoforms.fasta
```

Isoform sequences can be further refined and filtered using different tools and pipelines. Among the recommended polishing steps before downstream transcriptome analyses is collapsing the redundant sequences representing the same isoform. If a reference genome is available, sequences can be collapsed to unique isoforms by mapping the isoform FASTA file to the reference fasta. Mapping can be done using minimap2 [20]:

```
/PathToInstalledTool/minimap2 \
     -t 30 \
     -ax splice \
     -uf \
```

```
    --secondary=no \
    -C5 \
    -O6,24 \
    -B4 \
    /MyAnalysis/genome/MyGenome.fa ./input/hq_lq_isoforms.fasta >
    ./output/hq_lq_isoforms.fasta.sam 2>
    ./output/hq_lq_isoforms.fasta.sam.log
```

The above command defines the input reference genome assembly and the isoform FASTA to be mapped to the reference. It also calls for two output files – SAM file and a "log" file. We specified the following mapping parameters: mapping spliced long reads (`-ax splice`), considering forward transcript strand only (`-uf`), not outputting secondary alignments (`--secondary no`), cost for a non-canonical GT-AG splicing of 5 (`-C5`), gap open penalty of 6 and 24 (`-O6,24`), mismatching penalty of 4 (`-B4`), and the number of used threads (`-t`).

After mapping, isoform collapsing can be performed using a Python script provided by the cDNA_cupcake GitHub repository (https://github.com/Magdoll/cDNA_Cupcake). The script requires a sorted SAM file, which can be generated with samtools [21]:

```
/PathToInstalledTool/samtools sort \
    -k 3,3 \
    -k 4,4n \
    ./output/hq_lq_isoforms.fasta.sam >
    ./output/hq_lq_isoforms.fasta.sorted.sam
```

Sorted SAM is used as an input for the cDNA_cupcake Python script:

```
/PathToInstalledTool/collapse_isoforms_by_sam.py \
--input ./input/hq_lq_isoforms.fasta \
-s ./output/hq_lq_isoforms.fasta.sorted.sam \
--dun-merge-5-shorter \
-o ./output/isoforms
```

We specified the isoform FASTA file (`--input`) and sorted SAM file (`-s`). The command also includes the option to skip collapsing shorter 5' transcripts (`--dun-merge-5-shorter`). It will generate "./output/isoforms.collapsed.*" files, including a GFF format file with unique isoforms, a FASTA file containing sequences of representative isoforms ("isoforms.collapsed.rep.fa"), and a text file with information on groups of isoforms collapsed to the same gene ("isoforms.collapsed.group.txt").

The cDNA_cupcake repository also includes a Python script for filtering away isoforms with 5' degradation, an artefact potentially introduced during cDNA synthesis:

```
/PathToInstalledTool/filter_away_subset.py
./output/isoforms.collapsed
```

The above command will generate "./output/isoforms.collapsed.filtered.*" files, including a GFF format file with filtered isoforms, a FASTA file containing filtered sequences of representative isoforms ("isoforms.collapsed.filtered.rep.fa"), and a text file with information on full-length reads associated with each isoform ("isoforms.collapsed.filtered.abundance.txt").

6 Sequence consolidation and transcriptome completeness analysis

6.1 Collapsing identical sequences

Merging *de novo* and Iso-Seq transcriptome assemblies results in a redundant set of transcript sequences. It is therefore useful to collapse identical sequences, albeit of different lengths, retaining only the longest unique sequences. This can be done with CD-HIT tool [22] at a 100% sequence identity threshold (see **Note 7**). If you plan to use different transcriptomic resources, you should first concatenate FASTA files of transcriptomes of interest. In our case, we combined Iso-Seq isoforms and *de novo* assembled contigs:

```
mkdir /MyAnalysis/combine
cd /MyAnalysis/combine

cat \
../long/output/isoforms.collapsed.filtered.rep.fa \
../short/spades_assembly/transcripts.fasta >
./full_transcriptome.fasta

/PathToInstalledTool/cdhit-est \
     -i ./full_transcriptome.fasta \
     -o ./collapsed_transcriptome.fasta \
     -c 1 \
     -M 100000 \
     -t 30
```

If you wish to be more stringent and collapse very similar but not identical sequences, you can use lower identity thresholds (`-c`). This depends on the downstream analyses, as more stringent filtering can remove unwanted technical variability (sequencing and assembly errors) but also true biological variability, potentially important for your study. When searching for sequences coding for target activity, it might be beneficial to retain as much variability as possible, as you

can remove technical errors later in the analysis but keep information on possible functionally important SNPs.

6.2 Assessment of transcriptome completeness

After sequence consolidation, you should assess the quality of your transcript set and its usefulness for your research goal. When searching for coding sequences with specific metabolic functionality, it is important to have a complete and accurate sequence resource, encompassing the whole pool of expressed genome in the target organism. Apart from reference genome-based assessment of completeness (for example with rnaQUAST tool, as described in **Subsection 3.3**), a more general approach is to determine the coverage of universal single-copy genes with your transcriptomic data. Complete transcriptome should include orthologs of most single-copy genes known to be universally conserved and expressed in the taxonomic clade of your species of interest. This approach is implemented in the BUSCO tool [23], which determines the completeness of a sequence dataset by searching for orthologs of BUSCOs – Benchmarking sets of Universal Single-Copy Orthologs:

```
/PathToInstalledTool/busco \
     -i ./collapsed_transcriptome.fasta \
     -l insecta_odb10 \
     -o transcriptomeBUSCO \
     -m tran \
     -c 30
```

To run BUSCO, you must define the input files (`-i`) and the type of input data (`-m`; genome, protein, or transcriptome FASTA file). It is also recommended to select the most appropriate BUSCO lineage dataset to be used (`-l`), which is a set of marker genes present as single-copy genes in at least 90 % of species from a specific evolutionary lineage. In our case, we used the BUSCO dataset specific for insects. Additionally, you can specify the output path with prefixes for output file names (`-o`) and the number of threads to use (`-c`). The results are numbers and percentages of complete BUSCOs, complete but duplicated BUSCOs, fragmented BUSCOs, and missing BUSCOs, given in a text file and as a stacked bar plot image (**Figure 2**). Complete, nonredundant transcriptomes without assembly or sequencing errors will have a high percentage of complete BUSCOs and minimal numbers of duplicated, fragmented, and missing BUSCOs.

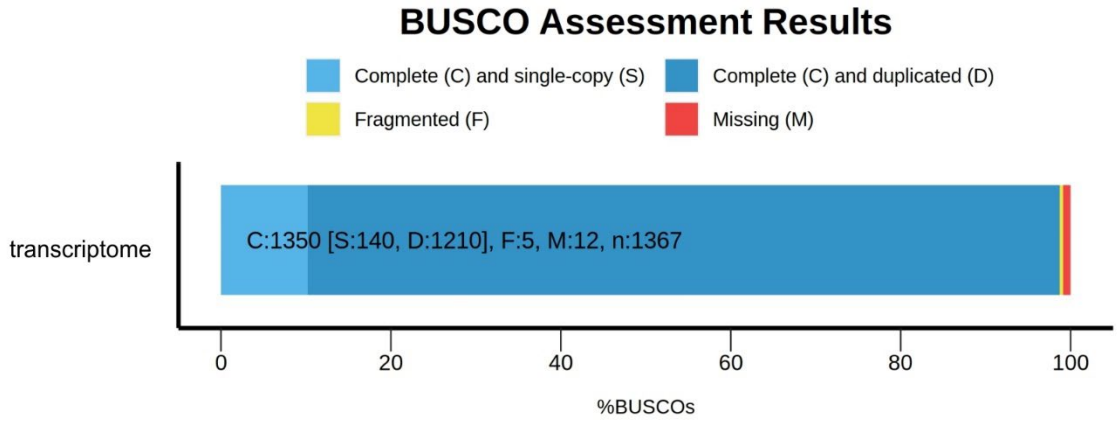

Figure 2: Results of BUSCO completeness assessment. The analysed transcriptome has good completeness (1350 out of 1367 complete BUSCOs) but high redundancy (1210 out of 1350 complete BUSCOs are duplicated). There are only a few fragmented and missing BUSCOs.

6.3 Assessment of transcriptome contamination

RNA samples from target organisms are prone to contamination by nucleic acids from other organisms – symbionts of your species of interest, their food, airborne organisms, and human nucleic acids from the operator. Additionally, foreign nucleic acids can be introduced with reagents during RNA isolation and library preparation, and during sequencing as a result of index hopping causing misassignments of reads between samples sequenced on the same lane. To obtain a transcriptome of your target species, contaminating sequences need to be identified and removed. Alternatively, you might be interested in sequences from symbiotic organisms and need to identify and separate them from other sequences.

To identify sequences from non-target organisms, you can run tools for taxonomic classification, for example, mmseqs2 taxonomy workflow [24, 31]. You must first download and create a reference sequence database with taxonomic information against which your query sequences will be aligned. In our case, we selected to use the non-redundant NCBI database (NR). You also need to create a database of your query sequences:

```
mkdir /MyAnalysis/combine/taxonomy
cd /MyAnalysis/combine/taxonomy

/PathToInstalledTool/mmseqs databases NR NRdatabase ./Database

/PathToInstalledTool/mmseqs createdb
../collapsed_transcriptome.fasta ./QueryDatabase
```

```
/PathToInstalledTool/mmseqs createindex QueryDatabase
./QueryDatabase_index
```

You can then run the taxonomy workflow and create different types of outputs:

```
/PathToInstalledTool/mmseqs taxonomy \
./QueryDatabase \
./Database \
./taxonomyResult \
.
```

The first specified parameter is the path to the created query database, followed by the path to the reference database. With the next two parameters, you specify the name of the output files and the path to the output folder. The result is a report text file with a taxon tree and the numbers of sequences assigned to each taxon. You can also generate a text file in which each sequence is annotated with a taxonomic unit (or unassigned, if no high-confidence matches were found):

```
/PathToInstalledTool/mmseqs createtsv \
./QueryDatabase \
./taxonomyResult \
./taxonomyResult.tsv
```

Additionally, mmseqs can also generate Kraken [32] and Krona-style [33] taxonomy reports . For Kraken-style report, which can be visualised with Pavian [34], use:

```
/PathToInstalledTool/mmseqs taxonomyreport \
./QueryDatabase \
./taxonomyResult \
./taxonomyResult_report
```

To generate Krona HTML report, use :

```
/PathToInstalledTool/mmseqs taxonomyreport \
./QueryDatabase \
./taxonomyResult \
./Kronareport.html \
--report-mode 1
```

Inspect the output files and identify any major contaminants. Transcript sequences with a high probability of originating from non-target species can be discarded from the transcriptome FASTA file if needed.

7 Mapping short and long reads

7.1 Mapping long sequences to the reference genome

If a reference genome of the species you are analysing is available, *de novo* assembled contigs and Iso-Seq isoforms can be mapped to it. Mapping to the genome can be informative in several ways. It aligns the transcripts to existing gene models and provides novel structural information or corrections to gene model predictions. It can also provide evidence for missing gene models in the genome assembly or even evidence for gaps in the assembly, in the case of unmapped transcripts. By comparing gene models to assembled and sequenced transcripts, we can predict the correct gene structure with higher confidence. Additionally, we can also detect single nucleotide polymorphisms and allelic variations.

First, generate the folders for mapping results:

```
mkdir /MyAnalysis/mapping
mkdir /MyAnalysis/mapping/contigs
cd /MyAnalysis/mapping/contigs
```

Besides the already mentioned minimap2, which we used for mapping Iso-Seq isoforms, STARlong is another option for mapping full-length transcripts. We applied STARlong for mapping *de novo* assembled contigs. First, a genome index needs to be generated:

```
mkdir /MyAnalysis/mapping/index_genome
```

```
/PathToInstalledTool/STARlong \
     --runThreadN 32 \
     --runMode genomeGenerate \
     --genomeDir ../index_genome \
     --genomeFastaFiles /MyAnalysis/genome/MyGenome.fa \
     --sjdbGTFfile /MyAnalysis/genome/MyGenome.gtf \
```

To generate a genome index, the user must define the `genomeGenerate` mode and provide reference FASTA and annotation (GFF or GTF) files (`--genomeFastaFiles`, `--sjdbGTFfile`). We also specified the directory in which we want the index files to be saved (`--genomeDir`) and the number of used threads (`-t`). With the generated index, you can proceed to mapping in `alignReads` mode:

```
/PathToInstalledTool/STARlong \
     --runMode alignReads \
     --runThreadN 32 \
```

```
--genomeDir ../index_genome \
--readFilesIn
/MyAnalysis/short/spades_assembly/transcripts.fasta \
--outFileNamePrefix ./TRtoGenome_STARlong \
--outSAMtype BAM SortedByCoordinate \
--outReadsUnmapped Fastx \
--seedPerReadNmax 100000 \
--seedPerWindowNmax 1000 \
--alignTranscriptsPerReadNmax 100000 \
--alignTranscriptsPerWindowNmax 10000
```

The command specifies the input FASTA file with transcripts to be mapped (`--readFilesIn`), path to genome index folder (`--genomeDir`), output directory and output file prefix (`--outFileNamePrefix`), and sorted by coordinate BAM as the output alignment file (`--outSAMtype`). Additionally, we also wanted all unmapped transcripts to be put out in a separate FASTA file (`--outReadsUnmapped`). We also specified four additional mapping parameters (last four rows), defining the maximum numbers of seeding and aligning per window and read.

The sorted BAM file is useful for the visualisation of transcript mapping to the genome, for example with IGV (Integrative Genomics Viewer) [26]. You can view your genes/transcripts of interest and check for discrepancies between gene models and assembled or sequenced transcript sequences (**Figure 3**). It is also useful to generate a list connecting mapped transcripts and genome features at the site of mapping. This can be done with the MatchAnnot Python script by first converting the mapping BAM file to SAM format using samtools:

```
/PathToInstalledTool/samtools view \
-@ 32 \
-O SAM \
-o ./TRtoGenome_STARlong.Aligned.out.sam \
./TRtoGenome_STARlong.Aligned.sortedByCoord.out.bam
```

The above command defines the output file name and location (`-o`), output file type (`-O`), input file and number of used threads (`-@`). The generated SAM file needs to be sorted by coordinates, which can be done using the bash `sort` command:

```
sort -k 3,3 -k 4,4n ./TRtoGenome_STARlong.Aligned.out.sam >
./TRtoGenome_STARlong.Aligned.sortedByCoord.out.sam
```

Sorted SAM is used as an input for the MatchAnnot script together with the genome annotation file:

```
python /PathToInstalledTool/matchAnnot.py \
--gtf=/MyAnalysis/genome/MyGenome.gtf \
--format=alt \
./TRtoGenome_STARlong.Aligned.sortedByCoord.out.sam >
./TRtoGenome_STARlong.Aligned.sortedByCoord.out.sam.matchAnnot.txt
```

You need to specify the input SAM alignment file, reference GTF annotation file and its format (`--format`, standard GENCODE GTF, an alternative version of GTF or a pickle file), and name and location of the output text file. The output text file includes a "result" line for each mapped transcripts sequence, which lists the feature at the site of mapping and matching score. The score goes from 0 to 5, with a score of 0 assigned to transcripts with overlap to gene but little to no exon congruence and a score of 5 to transcripts with a one-for-one match to gene exon structure.

Figure 3: Visualisation of long and short reads mapping to the reference genome in IGV. Visualisation of mapping files (indexed BAM files) of short reads (top track) and *de novo* assembled contigs (middle track) to the reference genome (bottom track). Assembled contig and short read coverage imply a different gene model than the models predicted in the reference genome.

7.2 Mapping short reads to genome and transcriptome reference

For differential expression analysis and to analyse SNPs and allelic variations, short reads can be mapped either to reference genome (if available) and/or transcriptome (Iso-Seq isoforms or *de novo* assembled contigs). For *de novo* transcriptome assembly, short read mapping statistics are also important for quality control assessment, as a high fraction of reads should map back to the assembled contigs.

Short-read mapping can be done using STAR [25]. First, reference indices need to be generated for each type of reference (genome – as given above for long read mapping, *de novo* contigs, and Iso-Seq isoforms). We already generated an index file for the reference genome in the

"/MyAnalysis/mapping/index_genome" folder. For a transcriptome reference, we do not need to provide an annotation GTF/GFF file:

```
cd /MyAnalysis/mapping/
mkdir /MyAnalysis/mapping/short
mkdir /MyAnalysis/mapping/index_denovo
mkdir /MyAnalysis/mapping/index_IsoSeq

/PathToInstalledTool/STAR \
     --runMode genomeGenerate \
     --runThreadN 32 \
     --genomeDir ./index_denovo \
     --genomeFastaFiles ../short/spades_assembly/transcripts.fasta

/PathToInstalledTool/STAR \
     --runMode genomeGenerate \
     --runThreadN 32 \
     --genomeDir ./index_IsoSeq \
     --genomeFastaFiles
     ../long/output/isoforms.collapsed.filtered.rep.fq
```

With generated reference indices, we can proceed to short read mapping in `alignReads` mode:

```
/PathToInstalledTool/STAR \
     --runMode alignReads \
     --runThreadN 32 \
     --genomeDir ./index_genome \
     --readFilesIn \
     /MyAnalysis/short/raw/Sample1_1.fastq.gz \
     /MyAnalysis/short/raw/Sample1_2.fastq.gz \
     --outFileNamePrefix ./short/mapToGenome_Sample1_ \
     --outSAMtype BAM SortedByCoordinate \
     --readFilesCommand pigz -c -d
```

We specified the input short read FASTA files (`--readFilesIn`), in our case a forward (_1) and reverse (_2) read files, path to reference index directory (`--genomeDir`), output directory and output file prefix (`--outFileNamePrefix`), and sorted by coordinate BAM as the output alignment file (`--outSAMtype`). If you are working with compressed input files, you can also specify `–readFilesCommand` (see **Note 8**). The above example applies to mapping to reference genome. For mapping to transcriptome data, change the `--genomeDir` parameter to specify the path to index folder of transcriptomic sequences. For mapping multiple samples, you can also create a "for" loop using bash syntax.

Output BAM files can be used for mapping visualisation with IGV. You can, for example, inspect gene structure, allelic variation, and SNPs (**Figure 4**).

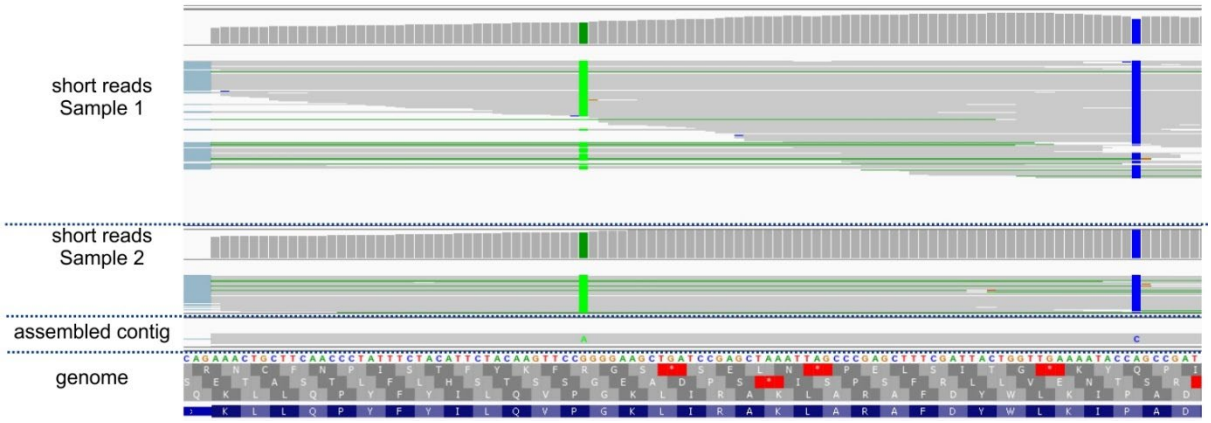

Figure 4: Assessment of SNPs in IGV. Visualisation of mapping files (indexed BAM files) of short reads from two biological replicate samples (upper two tracks) to the reference genome (bottom track). Short reads include nucleotide substitutions at two different positions compared to the reference genome sequence. Contig assembled from short reads includes the SNPs as well (middle track).

Additionally, BAM files can be used to generate count data, by summarising the number of reads mapping to each feature (gene model in reference genome or contigs and isoforms in reference transcriptome). Counting mapped short reads to transcriptome reference can be done with the samtools idxstats tool. You first need to index BAM files generated by STAR mapping:

```
/PathToInstalledTool/samtools index \
-@ 6 \
./short/mapToGenome_Sample1_Aligned.sortedByCoord.out.bam

/PathToInstalledTool/samtools idxstats \
-@ 6 \
./short/mapToGenome_Sample1_Aligned.sortedByCoord.out.bam >
./short/mapToGenome_Sample1_Aligned.counts
```

For counting reads mapped to annotated reference genome you have other options (see **Note 9**).

8 Differential expression analysis

Gene- and transcript-level read counts are used for differential expression analysis. Here, we used statistical analysis in R with edgeR and limma packages [35]. Below we list crucial steps of our analysis, while the whole script is available on GitHub (https://github.com/NIB-SI).

After first importing your count data for all samples and organising it in a data frame (`x`) with samples as columns and all genes or transcripts as rows, you can proceed to specifying your experimental groups. In our case, we had two groups with four replicates in each group:

```
group <- factor(c(1,1,1,1,2,2,2,2))
```

With a count data table and specified experimental groups, you can create a DGEList object:

```
y <- edgeR::DGEList(counts=x, group=group)
```

Before statistical calculations, you need to filter out low-expressed genes:

```
keep.exprs <- edgeR::filterByExpr(y, group = group, min.count = 50,
min.total.count = 100)
y_filter <- y[keep.exprs, keep.lib.sizes=TRUE]
```

We specified that we want to keep only genes that have at least 50 counts in some samples and a minimal total count across all samples of 100. After filtering, we can calculate normalisation factors using TMM normalisation:

```
y_filter <- calcNormFactors(y_filter, method=”TMM”)
```

After filtering and normalisation, you can proceed to linear model fitting, by defining the model matrix and contrasts:

```
design <- model.matrix(~0+group)
colnames(design) <- c("virgin ","mated")
contrastMatrix = limma::makeContrasts("virgin-mated", levels=design)
```

In the final steps, you fit the linear model, compute the statistics and output a results table:

```
fit <- edgeR::voomLmFit(y_filter, design)
fit2 <- limma::contrasts.fit(fit, contrastMatrix)
fit2 <- limma::eBayes(fit2)
results <- limma::topTable(fit2, coef=1, number=1000000,
sort.by="none")
```

The result table will include log2 fold-changes (logFC), p-values and FDR-adjusted p-values for each gene in the specified contrast set by the `topTable` parameter "coef". In the above example the "coef=1" outputs the first contrast, namely "virgin-mated".

9 Candidate selection

9.1 Identification of open reading frames

With sequence and expression information, we can proceed to candidate selection. First, we should identify open reading frames (ORFs) in transcriptome sequences and translate them to protein sequences. This can be done with transDecoder [36]:

```
cd /MyAnalysis/combine

/PathToInstalledTool/TransDecoder.LongOrfs \
-t ./collapsed_transcriptome.fasta
```

TransDecoder.LongOrfs will by default extract ORFs that are at least 100 amino acids long (300 nucleotides). From the extracted ORFs, you can predict putative coding sequences with TransDecoder.Predict:

```
/PathToInstalledTool/TransDecoder.Predict \
-t ./collapsed_transcriptome.fasta \
```

Among other output files will be "collapsed_transcriptome.fasta.transdecoder.pep", a FASTA format file with putative coding sequences. It can be used for functional annotations, for example with InterProScan protein domain search.

9.2 Functional annotation with InterProScan

Looking for a specific enzymatic activity, you might have some preexisting knowledge on possible protein families that perform similar reactions. In this case, it is sensible to extract protein sequences with homology to target protein families and domains from your transcriptomic data. InterPro [37] is a protein classification resource that combines protein function signatures from different protein databases and offers classification of target sequences based on the presence of specific signatures (domains, motifs, and other conserved sites). For the characterisation of novel sequences, you can use the InterProScan tool [27]:

```
/PathToInstalledTool/interproscan.sh \
     --applications Pfam \
```

```
--seqtype p \
--input ./collapsed_transcriptome.fasta.transdecoder.pep \
--output-dir . \
--disable-precalc \
--iprlookup \
--formats TSV \
--cpu 100
```

We define the input file (`--input`), input file type (`--seqtype`, protein sequences in our case), output directory (`--output-dir`), and tab-separated output file format (`--formats`). With the specified parameters, InterProScan will perform Pfam-based search (`--applications Pfam`) and map the results to InterPro database entries (`--iprlookup`). Output tab-separated file will include highest scoring Pfam protein domain matches to query proteins with confidence values (E-value) and matching InterPro entries. From this file, you can extract IDs of query sequences that likely contain your target protein domain or a conserved site.

9.3 Selection of candidate sequences

When you extract target transcript sequences from your species of interest, you can perform manual evaluations and curation of selected candidates. First, you can cross-check candidates with the results of the taxonomic assessment (see **Subheading 6.3**) and determine their most probable taxa of origin. If you are interested only in sequences encoded in the genome of your target species, you can discard candidates originating from contaminating sequences. You can also check the confidence values of InterPro annotation (see **Subheading 9.2**) and inspect the protein sequences of candidates to possibly eliminate candidates with low similarity to target protein domains or families.

The next important step is to address sequence redundancy, which can be estimated from BUSCO results (see **Subheading 6.2**) – the redundant dataset will have a high percentage of complete but duplicated BUSCOs. Transcriptome obtained with the presented pipeline might be highly redundant, as we did not perform any assembly thinning and collapsed only sequences with 100% identity (see **Subheading 6.1**). Therefore, the same locus can be represented with many different sequences in our transcriptome dataset. These differences can arise from biological variation – mRNA processing, alternative splicing, allelic variation, SNPs, or from technical variation (sequencing errors, assembly errors). Additionally, we combined transcripts from two separate sequencing approaches – *de novo* assembled short

reads and Iso-Seq – which could each include sequences corresponding to the same mRNA transcript. By comparing and clustering all candidate sequences you should obtain consensus sequences for each locus with additional information on biological variability while discarding transcript variations most probably resulting from technical errors.

The most reliable and thorough approach for the selection of consensus sequences is manual assessment, which is of course more feasible for a lower number of candidate sequences. First, you can cluster your candidate sequences based on sequence identity with CD-HIT (see **Subheading 6.1**) using identity thresholds below 100%. You can either perform clustering on the nucleotide level with CD-HIT-EST or use CD-HIT for protein sequences. One of the output files will have information on formed clusters of sequences with identity above the threshold. We recommend that you also perform multiple sequence alignments for each cluster. This can be done with several tools, for example, online tools (Clustal Omega at EMBL-EBI servers, https://www.ebi.ac.uk/jdispatcher/msa/clustalo), or applications such as MEGA-X [38] and Geneious (https://www.geneious.com/). You can inspect the alignments and determine types of sequence variations. To evaluate whether they could be of biological origin or technical errors, you can use short-read mapping data (see **Subheading 7.2**). By visualising short-read mappings to candidate transcripts (e.g. in IGV, **Figure 4**) you can determine which variations have the highest read coverage and support. You can also determine allelic variations by inspecting the frequency of SNPs in short-read data. Ambiguities can be also resolved by giving priority to Iso-Seq transcripts compared to *de novo* assembled contigs, especially in cases of discrepancies at 5' or 3'-ends, as full-length sequencing should be more reliable compared to assembly algorithms.

You might encounter similar sequences with relatively high variability for which you cannot easily decide whether they originate from the same loci or recent duplication events and therefore represent paralogs. In this case, we recommend BLAST searches against different non-redundant and experimental databases, for example, TSA (Transcriptome Shotgun Assembly), and determine whether you can find convincing evidence for the existence of orthologs of each sequence in related species. The latter suggests that your sequences of interest are paralogs, which can be further substantiated with phylogenetic analyses and evaluation of the evolutionary time of duplication events.

The above selection process should result in a list of candidate isoforms, which can be tested and validated experimentally. For this, the list should be concise, especially if there is no option

for high-throughput experimental testing. To further shorten the list of candidates, any prior knowledge and experimental data can be used. In our case, we also considered differential gene expression data, expecting higher expression of genes coding for target functionality in samples of virgin females, compared to mated females (see **Subheading 8**). We, therefore, gave higher priority to candidates with strong and significant differential expression.

10 Making your data FAIR

To facilitate fast, efficient, and equitable scientific development, research should be conducted and disseminated by standards of open science [39, 40]. One important pillar of open science is applying FAIR principles [41] to your experimental data. FAIR guiding principles for scientific data management and stewardship propose that data should be findable, accessible, interoperable, and reusable. This encompasses, among other approaches, the use of standard and machine-readable (meta)data formats, diligent and thorough metadata reporting, and the use of public repositories.

The data management plan should be made before the start of your project. The plan specifies (among other things) the types of data generated, requirements and plan for their local storage, minimum information metadata standards used for each data type, and plan of data sharing, curation, and preservation.

10.1 Organise your pipeline and results with pISA-tree

Experimental data is usually uploaded and processed locally on personal computers. If not organised in a standard manner, it is usually difficult and time-consuming for an external collaborator or reader of your work to find, reuse or reanalyse your data. Therefore, besides data, computational analysis should also follow FAIR principles. This implies that the tools, pipelines and analysis results are well-organised, documented and reproducible. To cope with the local organisation and storage of research data and metadata, we developed the pISA-tree system [42], which establishes a standardised data storage hierarchy and guides users to provide sufficient metadata. It encompasses a set of consecutive batch files that generate a standardised directory structure and template metadata files. It also interactively guides users to input notes and descriptions of the aims and approaches of conducted research. Folders generated by pISA-tree follow the hierarchical order of Project-Investigation-Study-Assay, expanding the ISA framework [43]. In this way, you can organise different experiments,

analyses, and pipelines of the same project, with the subfolders on the assay level containing the actual research data. The standard directory tree for bioinformatic assays will include subfolders such as “input”, “output”, “scripts”, and “reports” encouraging you to meaningfully organise your data for every step of your analysis.

The pipeline presented in this chapter was conveniently organised with pISA-tree, with the bioinformatic search for candidate genes organised as an investigation. Each subheading was further organised as a study and each analysis or processing step as an assay (see **Note 10**).

10.2 Share data and results through public databases

After finishing and publishing your project, your locally organised data should be made publicly accessible, unless there are justifiable restrictions. When planning your project, you should define whether your data can be shared or not and find appropriate repositories for each type of generated data. Below, we provide suggestions for data repositories applicable to the types of data generated and used in this chapter.

Raw high-throughput sequencing data can be deposited to specialised public repositories. RNA-Seq and Iso-Seq data can be uploaded to NCBI’s SRA (Sequencing Read Archive, https://www.ncbi.nlm.nih.gov/sra/). Since RNA-Seq data was used for gene expression analysis, we uploaded our raw data and expression analysis results to GEO (Gene Expression Omnibus, https://www.ncbi.nlm.nih.gov/geo/), which automatically uploads raw sequencing data to SRA as well. Transcriptome assemblies generated from your sequencing data can be deposited to NCBI’s TSA database (Transcriptome Shotgun Assembly).

If possible, you should also share your analysis steps and results. There are several cloud repositories specialised for FAIR research data storage and sharing, for example, Zenodo (https://zenodo.org/), Dataverse Project (https://dataverse.org/), and FAIRDOMHub (https://fair-dom.org/fairdomhub). You can upload folders with raw data, metadata, notes, methodology descriptions, results, scripts, and any other data important to your research project. If you organised the data with pISA-tree tool, your investigation folders can be either uploaded to FAIRDOMHub using pISA-tree’s auxiliary R libraries or compressed and uploaded to one of the general research data repositories.

11 Notes

Note 1: For other research questions, you should check user guides and consider adapting the parameters. Additionally, tools might have been updated and their use and options changed. We included links to support pages for all the used tools.

Note 2: For our study, we decided on *de novo* assembly, as only a fragmented reference genome was available.

Note 3: Raw FASTQ read files from our study are available at NCBI (GEO accession GSE179660, https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE179660).

Note 4: You can download deposited data through a web browser from search results or use the command line tool SRA Toolkit (https://github.com/ncbi/sra-tools).

Note 5: If you know the sequences of adapters used in sequencing library preparation, you can prepare a FASTA file with adapter sequences and use it for adapter trimming. If you are aware of any other contaminating sequences (for example determined by FastQC quality control [44], https://github.com/s-andrews/FastQC), you can prepare additional FASTA files for filtering. We used artefact and contaminant files available on BBMap's Github page: https://github.com/BioInfoTools/BBMap/tree/master/resources.

Note 6: We hypothesized that target transcripts might be present in RNA samples at very low copy numbers, resulting in lower sequencing coverage and low isoform quality.

Note 7: For nucleotide sequences, you need to use CD-HIT-EST.

Note 8: The `readFilesCommand` parameter can be used to decompress input FASTQ files, for example using the gzip or pigz program. If working with larger files, decompression can be even faster using more parallelised programs such as rapidgzip.

Note 9: If you are mapping short reads to the annotated reference genome and want more control over which mapped reads should be counted (based on mapping score, coverage of reference region, mismatches, multimapping etc.), you can use "featureCounts" (https://subread.sourceforge.net/featureCounts.html). To count just uniquely mapped reads you can also use and specify `--quantMode` parameter in STAR command. Option `--quantMode GeneCounts` will give you an output text file with counts for each gene model specified in the annotation file.

Note 10: During the project, each project partner organised the data and results with pISA-tree locally. Investigation folders from all partners were uploaded on FAIRDOMHub at the end of the project and are publicly available: https://fairdomhub.org/investigations/550.

12 Acknowledgements

The presented pipeline was used in the scope of "ERACoBioTech" EU project SUSPHIRE (Sustainable Production of Pheromones for Insect Pest Control in Agriculture) funded by the Horizon 2020 research and innovation program under grant agreement No. 722361. This work was also funded by the Slovenian Ministry of Education, Science and Sport and Slovenian Research and Innovation Agency (grant agreement P4-0165).

13 Bibliography

1. Owen C, Patron NJ, Huang A, Osbourn A (2017) Harnessing plant metabolic diversity. Curr Opin Chem Biol 40:24–30. https://doi.org/10.1016/j.cbpa.2017.04.015

2. Rosenberg J, Commichau FM (2019) Harnessing Underground Metabolism for Pathway Development. Trends Biotechnol 37:29–37. https://doi.org/10.1016/j.tibtech.2018.08.001

3. Robinson SL, Piel J, Sunagawa S (2021) A roadmap for metagenomic enzyme discovery. Nat Prod Rep 38:1994–2023. https://doi.org/10.1039/D1NP00006C

4. Delli-Ponti R, Shivhare D, Mutwil M (2021) Using Gene Expression to Study Specialized Metabolism—A Practical Guide. Front Plant Sci 11:625035. https://doi.org/10.3389/fpls.2020.625035

5. Yang D, Du X, Yang Z, et al (2014) Transcriptomics, proteomics, and metabolomics to reveal mechanisms underlying plant secondary metabolism. Eng Life Sci 14:456–466. https://doi.org/10.1002/ELSC.201300075

6. Zhu F, Wen W, Cheng Y, et al (2023) Integrating multiomics data accelerates elucidation of plant primary and secondary metabolic pathways. aBIOTECH 4:47–56. https://doi.org/10.1007/S42994-022-00091-4/FIGURES/3

7. Mutwil M (2020) Computational approaches to unravel the pathways and evolution of specialized metabolism. Curr Opin Plant Biol 55:38–46.

https://doi.org/10.1016/J.PBI.2020.01.007

8. Stark R, Grzelak M, Hadfield J (2019) RNA sequencing: the teenage years. Nat Rev Genet 20:631–656. https://doi.org/10.1038/s41576-019-0150-2

9. Lowe R, Shirley N, Bleackley M, et al (2017) Transcriptomics technologies. PLOS Comput Biol 13:e1005457. https://doi.org/10.1371/JOURNAL.PCBI.1005457

10. Byrne A, Cole C, Volden R, Vollmers C (2019) Realizing the potential of full-length transcriptome sequencing. Philos Trans R Soc B Biol Sci 374:20190097. https://doi.org/10.1098/RSTB.2019.0097

11. Amarasinghe SL, Su S, Dong X, et al (2020) Opportunities and challenges in long-read sequencing data analysis. Genome Biol 21:30. https://doi.org/10.1186/s13059-020-1935-5

12. Juteršek M, Gerasymenko IM, Petek M, et al (2024) Transcriptome-informed identification and characterization of Planococcus citri cis- and trans-isoprenyl diphosphate synthase genes. iScience 27:. https://doi.org/10.1016/j.isci.2024.109441

13. Mateos Fernández R, Petek M, Gerasymenko I, et al (2022) Insect pest management in the age of synthetic biology. Plant Biotechnol J 20:25–36. https://doi.org/10.1111/pbi.13685

14. Mateos-Fernández R, Moreno-Giménez E, Gianoglio S, et al (2021) Production of Volatile Moth Sex Pheromones in Transgenic Nicotiana benthamiana Plants. BioDesign Res 2021:9891082. https://doi.org/10.34133/2021/9891082

15. Levi-Zada A, Fefer D, David M, et al (2014) Diel periodicity of pheromone release by females of Planococcus citri and Planococcus ficus and the temporal flight activity of their conspecific males. Naturwissenschaften 101:671–678. https://doi.org/10.1007/s00114-014-1206-y

16. Nuo S-M, Yang A-J, Li G-C, et al (2021) Transcriptome analysis identifies candidate genes in the biosynthetic pathway of sex pheromones from a zygaenid moth , Achelura yunnanensis (Lepidoptera : Zygaenidae). PeerJ 9:e12641. https://doi.org/10.7717/peerj.12641

17. Yao S, Zhou S, Li X, et al (2021) Transcriptome Analysis of Ostrinia furnacalis Female

Pheromone Gland: Esters Biosynthesis and Requirement for Mating Success. Front Endocrinol (Lausanne) 12:736906. https://doi.org/10.3389/fendo.2021.736906

18. Bushmanova E, Antipov D, Lapidus A, Prjibelski AD (2019) RnaSPAdes: A de novo transcriptome assembler and its application to RNA-Seq data. Gigascience 8:1–13. https://doi.org/10.1093/gigascience/giz100

19. Bushmanova E, Antipov D, Lapidus A, et al (2016) RnaQUAST: A quality assessment tool for de novo transcriptome assemblies. Bioinformatics 32:2210–2212. https://doi.org/10.1093/bioinformatics/btw218

20. Li H (2018) Minimap2: Pairwise alignment for nucleotide sequences. Bioinformatics 34:3094–3100. https://doi.org/10.1093/bioinformatics/bty191

21. Danecek P, Bonfield JK, Liddle J, et al (2021) Twelve years of SAMtools and BCFtools. Gigascience 10:giab008. https://doi.org/10.1093/GIGASCIENCE/GIAB008

22. Fu L, Niu B, Zhu Z, et al (2012) CD-HIT: Accelerated for clustering the next-generation sequencing data. Bioinformatics 28:3150–3152. https://doi.org/10.1093/bioinformatics/bts565

23. Manni M, Berkeley MR, Seppey M, Zdobnov EM (2021) BUSCO: Assessing Genomic Data Quality and Beyond. Curr Protoc 1:e323. https://doi.org/10.1002/cpz1.323

24. Steinegger M, Söding J (2017) MMseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. Nat Biotechnol 35:1026–1028. https://doi.org/10.1038/nbt.3988

25. Dobin A, Davis CA, Schlesinger F, et al (2013) STAR: Ultrafast universal RNA-seq aligner. Bioinformatics 29:15–21. https://doi.org/10.1093/bioinformatics/bts635

26. Thorvaldsdóttir H, Robinson JT, Mesirov JP (2012) Integrative Genomics Viewer (IGV): high-performance genomics data visualization and exploration. Brief Bioinform 14:178–192. https://doi.org/10.1093/BIB/BBS017

27. Jones P, Binns D, Chang H-Y, et al (2014) InterProScan 5: Genome-scale protein function classification. Bioinformatics 30:1236–1240. https://doi.org/10.1093/bioinformatics/btu031

28. Robinson MD, McCarthy DJ, Smyth GK (2009) edgeR: A Bioconductor package for

differential expression analysis of digital gene expression data. Bioinformatics 26:139–140. https://doi.org/10.1093/bioinformatics/btp616

29. Ritchie ME, Phipson B, Wu D, et al (2015) limma powers differential expression analyses for RNA-sequencing and microarray studies. Nucleic Acids Res 43:e47–e47. https://doi.org/10.1093/NAR/GKV007

30. Nikolenko SI, Korobeynikov AI, Alekseyev MA (2013) BayesHammer : Bayesian clustering for error correction in single-cell sequencing. BMC Genomics 14:S7. https://doi.org/10.1186/1471-2164-14-S1-S7

31. Mirdita M, Steinegger M, Breitwieser F, et al (2021) Fast and sensitive taxonomic assignment to metagenomic contigs. Bioinformatics 37:3029–3031. https://doi.org/10.1093/bioinformatics/btab184

32. Wood DE, Salzberg SL (2014) Kraken: Ultrafast metagenomic sequence classification using exact alignments. Genome Biol 15:R46. https://doi.org/10.1186/GB-2014-15-3-R46/FIGURES/5

33. Ondov BD, Bergman NH, Phillippy AM (2011) Interactive metagenomic visualization in a Web browser. BMC Bioinformatics 12:385. https://doi.org/10.1186/1471-2105-12-385/FIGURES/4

34. Breitwieser FP, Salzberg SL (2020) Pavian: interactive analysis of metagenomics data for microbiome studies and pathogen identification. 36:1303–1304. https://doi.org/10.1093/bib/bbx120.Kim

35. Smyth GK, Ritchie ME, Law CW, et al (2018) RNA-seq analysis is easy as 1-2-3 with limma, Glimma and edgeR. F1000Research 5:1408. https://doi.org/10.12688/f1000research.9005.3

36. Haas B (2023) TransDecoder. https://github.com/TransDecoder/TransDecoder

37. Paysan-Lafosse T, Blum M, Chuguransky S, et al (2023) InterPro in 2022. Nucleic Acids Res 51:D418–D427. https://doi.org/10.1093/NAR/GKAC993

38. Kumar S, Stecher G, Li M, et al (2018) MEGA X: Molecular evolutionary genetics analysis across computing platforms. Mol Biol Evol 35:1547–1549. https://doi.org/10.1093/molbev/msy096

39. McKiernan EC, Bourne PE, Brown CT, et al (2016) How open science helps researchers succeed. Elife 5:e16800. https://doi.org/10.7554/eLife.16800

40. Woelfle M, Olliaro P, Todd MH (2011) Open science is a research accelerator. Nat Chem 3:745–748. https://doi.org/10.1038/nchem.1149

41. Wilkinson MD, Dumontier M, Aalbersberg IjJ, et al (2016) The FAIR Guiding Principles for scientific data management and stewardship. Sci Data 3:160018. https://doi.org/10.1038/sdata.2016.18

42. Petek M, Zagorščak M, Blejec A, et al (2022) pISA-tree - a data management framework for life science research projects using a standardised directory tree. Sci Data 9:685. https://doi.org/10.1038/s41597-022-01805-5

43. Sansone SA, Rocca-Serra P, Field D, et al (2012) Toward interoperable bioscience data. Nat Genet 44:121–126. https://doi.org/10.1038/ng.1054

44. Andrews S (2010) FastQC: A Quality Control Tool for High Throughput Sequence Data