Updating transXpress pipeline

Nicolas Dias & Karl Hendrik Tamkivi
supervised by Richard Meitern

Introduction

Short-read RNA-seq data analysis, including de novo transcriptome assembly, ORF prediction, and protein annotation, is commonly used for studying non-model eukaryotes without a reference genome [1, 2]. The NCBI Sequence Read Archive (SRA) houses millions of RNA-seq datasets, including those from non-model organisms, providing a valuable resource for diverse biological research [3]. However, the number of eukaryotic transcriptome assemblies in the NCBI Transcriptome Shotgun Assembly (TSA) database remains limited, reflecting the challenges in generating and sharing high-quality assemblies [4]. De novo transcriptome assembly involves the use of multiple bioinformatic tools, which can present installation difficulties and lack consistent adherence to best practices [5].

In 2023, Fallon et al. introduced transXpress, a streamlined de novo transcriptome assembly and annotation pipeline implemented in Snakemake [6]. While transXpress facilitates the exploration and identification of novel genes and proteins in non-model organisms, it lacks convenient tools for orthologous gene identification, which is crucial in comparative genomics and evolutionary studies for understanding gene function in non-model organisms [7]. Therefore, our project aims to enhance the transXpress pipeline by integrating data from OrthoDB, the largest hierarchical catalogue of orthologs. This addition will provide annotations about the best matching orthologs from the database, complementing the existing transcriptomes. We tested the modified pipeline using sequence data from stone loach (*Barbatula barbatula*) collected in Lahemaa, Estonia, from the rivers of Purtse and Kunda.

Methods

Initial goal of this project was to test the original transXpress pipeline on our sequence data. Unfortunately even after following the tutorials on the transXpress GitHub repository [8], we ran into multiple errors and dependency conflicts. First change necessary in order to run the entire process is to modify the Python version in envs/default.yaml to avoid conflicts with numpy.

```
dependencies:
    - python==3.8
```

Secondly, all the rules within transXpress/Snakefile need to be modified. Problem arises from the fact that the SLURM cluster running transXpress.sh interprets the memory parameter in MB, although in the Snakefile the memory parameter values are originally given in a way where the .sh file would interpret them in GB. Thus the .sh file receives a minuscule amount of memory for the computations and in order to fix this issue, the params of the rules in the Snakefile need to be modified.

```
rule x:
    params:
    memory="y",memory_slurm="y*1000"
```

Where x is some process rule in the Snakefile and y is the original amount of memory intended for the process. In order to use the resources of the memory_slurm in the process, the SLURM section of transXpress.sh needs to be modified as follows.

```
"SLURM")
  echo "Submitting snakemake jobs to SLURM cluster"
  snakemake --conda-frontend conda --use-conda --latency-wait 60
  --restart-times 1 --jobs 10000
  --cluster "sbatch -o {log}.slurm.out -e {log}.slurm.err -n {threads}
  --mem {params.memory_slurm} --time=5-00:00:00" "$@"
```

After running the entire pipeline we were ready to modify the existing pipeline and set up the connection with OrthoDB, although it must be noted that even after the previous edits we did not manage to run the sequence assembly process with Trinity without errors and settled with slightly less efficient rnaSPAdes. We modified the initial pipeline so that each of the annotated proteins in the FASTA file would be complemented with the best matching ortholog name and bit score from OrthoDB (Fig. 1). Bit score is a statistical indicator used to evaluate the BLAST output, that measures sequence similarity independent of query sequence length and database size and is normalised based on the raw pairwise alignment score [9]. Higher bit score indicates better sequence similarity and thus we used this as our main metric to filter out the best match from the database.

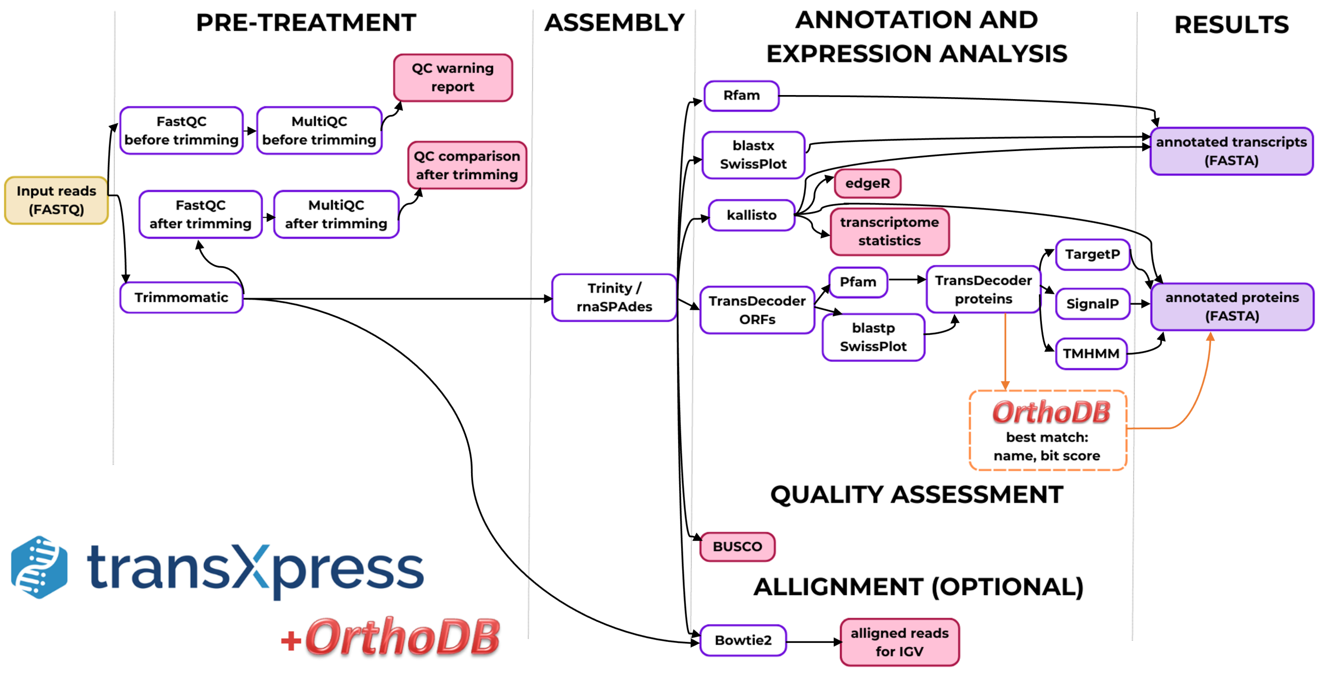


Figure 1: Original transXpress pipeline schema with an added OrthoDB best match annotation option shown in orange.

In order to connect the existing pipeline to OrthoDB and pull relevant annotations from it, several Snakefile rule changes and additions were required (Fig. 2). Firstly it is necessary to download two .tab files from OrthoDB and unzip them with rule download_orthodb (App. 1a). Further processes can be made faster with filtering out all of the unnamed genes and hypothetical proteins using the rule filter_orthodb (App. 1b). Next the filtered gene data can be inserted into a local BLAST database with the help of a rule makedb_orthodb (App. 1c), in order to make the future BLASTs faster and more efficient. The previous three steps need to be performed only once and the created database can then be used without reloading it in the future.

The linking of OrthoDB data with the existing transXpress pipeline happens after the fasta_split_pep stage of the initial pipeline that outputs multiple .pep files containing protein data. These files are essential for signalp, targetp, tmhmm and the newly added orthodb_blastp (App. 1d), which all receive identical input and thus produce easily rejoinable output as a result. These four outputs are then edited using the original annotation_merge_pep rule from the transXpress pipeline, which creates four .out files that can be merged with the help of annotated_fasta that required some modifications (App. 1e) to also merge the best match annotations from OrthoDB. As a result of this process, each protein in the trancriptome_annotated.pep has additional annotations about the best matching orthologous gene name and its bit score.

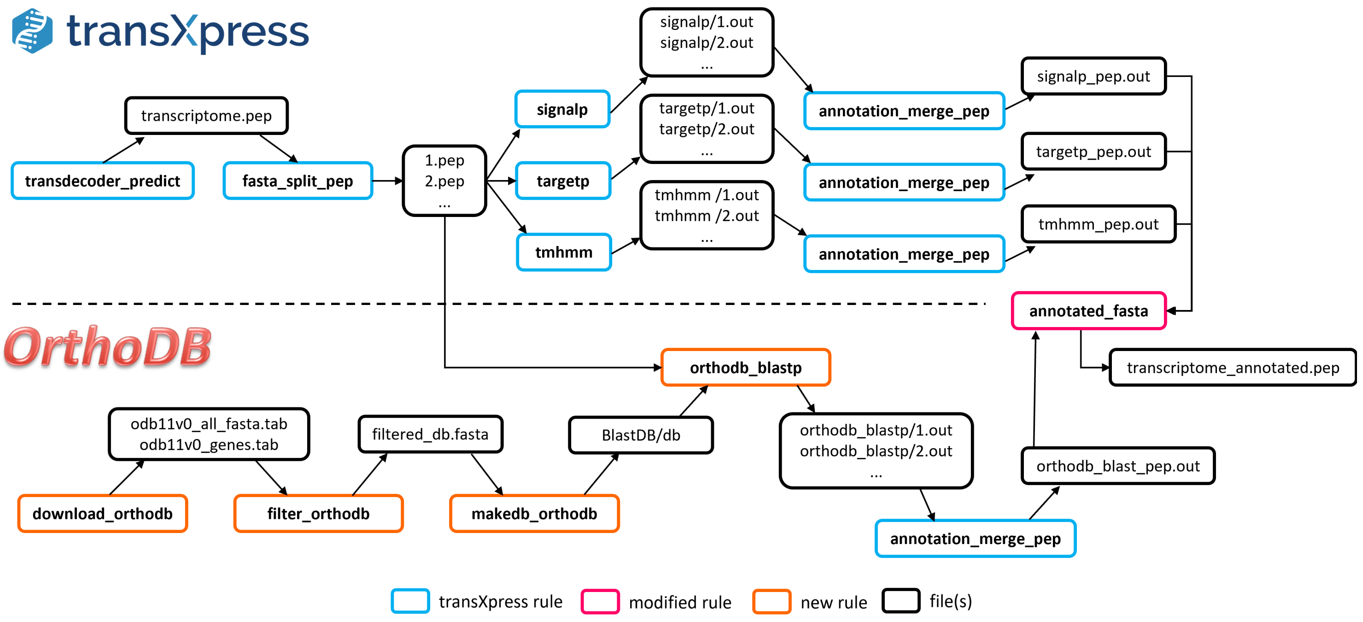
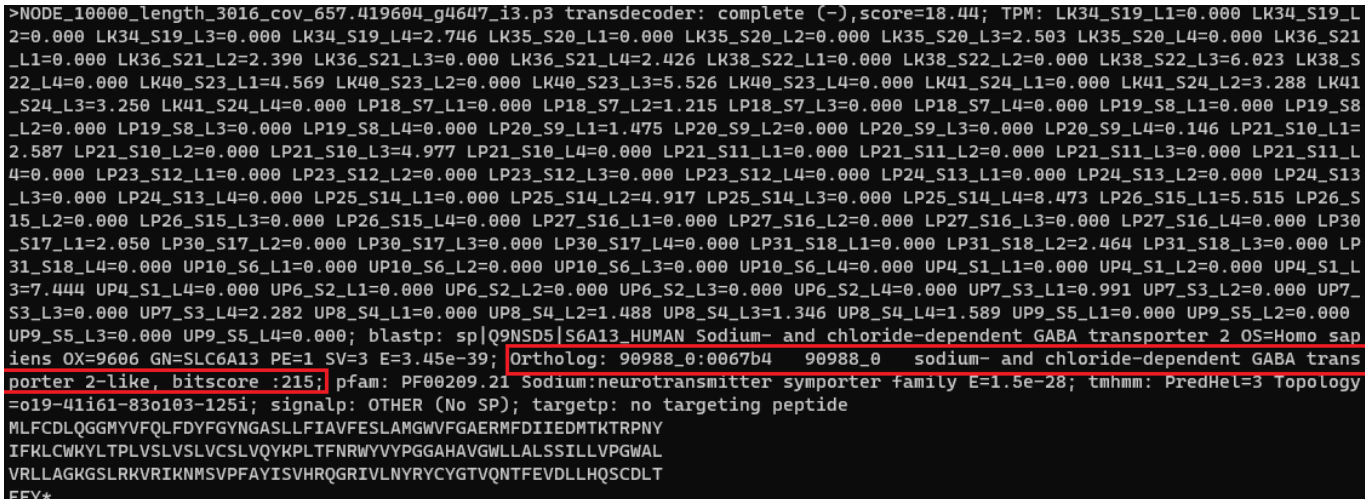


Figure 2: Relevant Snakefile rule schema edits to compliment transXpress annotations with best match annotations from OrthoDB.

Results

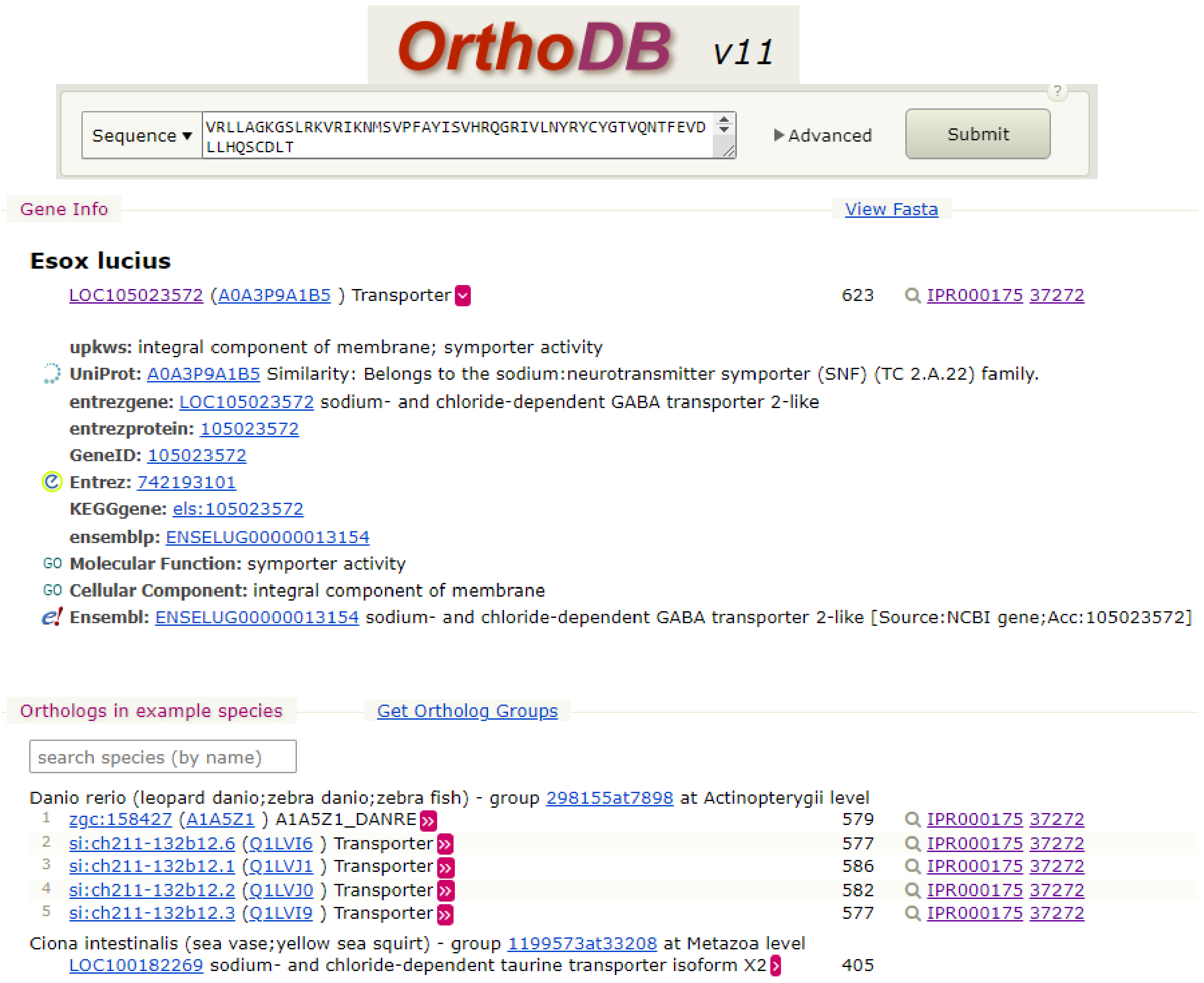
As a result of this project we were able to modify the existing transXpress pipeline so that it compliments the transcriptome annotations with the name and bit score of the best matching ortholog from OrthoDB database (Fig. 3). The updated annotations can later be exported as .csv files and further analysed in preferred environments.



```
>NODE_10000_length_3016_cov_657.419604_g4647_i3.p3 transdecoder: complete (-),score=18.44; TPM: LK34_S19_L1=0.000 LK34_S19_L
2=0.000 LK34_S19_L3=0.000 LK34_S19_L4=2.746 LK35_S20_L1=0.000 LK35_S20_L2=0.000 LK35_S20_L3=2.503 LK35_S20_L4=0.000 LK36_S21
_L1=0.000 LK36_S21_L2=2.390 LK36_S21_L3=0.000 LK36_S21_L4=2.426 LK38_S22_L1=0.000 LK38_S22_L2=0.000 LK38_S22_L3=6.023 LK38_S
22_L4=0.000 LK40_S23_L1=4.569 LK40_S23_L2=0.000 LK40_S23_L3=5.526 LK40_S23_L4=0.000 LK41_S24_L1=0.000 LK41_S24_L2=3.288 LK41
_S24_L3=3.250 LK41_S24_L4=0.000 LP18_S7_L1=0.000 LP18_S7_L2=1.215 LP18_S7_L3=0.000 LP18_S7_L4=0.000 LP19_S8_L1=0.000 LP19_S8
_L2=0.000 LP19_S8_L3=0.000 LP19_S8_L4=0.000 LP20_S9_L1=1.475 LP20_S9_L2=0.000 LP20_S9_L3=0.000 LP20_S9_L4=0.146 LP21_S10_L1=
2.587 LP21_S10_L2=0.000 LP21_S10_L3=4.977 LP21_S10_L4=0.000 LP21_S11_L1=0.000 LP21_S11_L2=0.000 LP21_S11_L3=0.000 LP21_S11_L
4=0.000 LP23_S12_L1=0.000 LP23_S12_L2=0.000 LP23_S12_L3=0.000 LP23_S12_L4=0.000 LP24_S13_L1=0.000 LP24_S13_L2=0.000 LP24_S13
_L3=0.000 LP24_S13_L4=0.000 LP25_S14_L1=0.000 LP25_S14_L2=4.917 LP25_S14_L3=0.000 LP25_S14_L4=8.473 LP26_S15_L1=5.515 LP26_S
15_L2=0.000 LP26_S15_L3=0.000 LP26_S15_L4=0.000 LP27_S16_L1=0.000 LP27_S16_L2=0.000 LP27_S16_L3=0.000 LP27_S16_L4=0.000 LP30
_S17_L1=2.050 LP30_S17_L2=0.000 LP30_S17_L3=0.000 LP30_S17_L4=0.000 LP31_S18_L1=0.000 LP31_S18_L2=2.464 LP31_S18_L3=0.000 LP
31_S18_L4=0.000 UP10_S6_L1=0.000 UP10_S6_L2=0.000 UP10_S6_L3=0.000 UP10_S6_L4=0.000 UP4_S1_L1=0.000 UP4_S1_L2=0.000 UP4_S1_L
3=7.444 UP4_S1_L4=0.000 UP6_S2_L1=0.000 UP6_S2_L2=0.000 UP6_S2_L3=0.000 UP6_S2_L4=0.000 UP7_S3_L1=0.991 UP7_S3_L2=0.000 UP7_
S3_L3=0.000 UP7_S3_L4=2.282 UP8_S4_L1=0.000 UP8_S4_L2=1.488 UP8_S4_L3=1.346 UP8_S4_L4=1.589 UP9_S5_L1=0.000 UP9_S5_L2=0.000
UP9_S5_L3=0.000 UP9_S5_L4=0.000; blastp: sp|Q9NSD5|S6A13_HUMAN Sodium- and chloride-dependent GABA transporter 2 OS=Homo sap
iens OX=9606 GN=SLC6A13 PE=1 SV=3 E=3.45e-39; Ortholog: 90988_0:0067b4   90988_0   sodium- and chloride-dependent GABA trans
porter 2-like, bitscore :215; pfam: PF00209.21 Sodium:neurotransmitter symporter family E=1.5e-28; tmhmm: PredHel=3 Topology
=o19-41i61-83o103-125i; signalp: OTHER (No SP); targetp: no targeting peptide
MLFCDLQGGMYVFQLFDYFGYNGASLLFIAVFESLAMGWVFGAERMFDIIEDMTKTRPNY
IFKLCWKYLTPLVSLVSLVCSLVQYKPLTFNRWYVYPGGAHAVGWLLALSSILLVPGWAL
VRLLAGKGSLRKVRIKNMSVPFAYISVHRQGRIVLNYRYCYGTVQNTFEVDLLHQSCDLT
FFY*
```

Figure 3: An example of transcriptome_annotated.pep output. Annotations about the best matching ortholog are shown with red.

The given ortholog annotations can be conveniently checked using the OrthoDB web interface [10] with the “Sequence” search condition. After inserting the entire protein sequence “MLFCD…” that encodes sodium- and chloride-dependent GABA transporter 2-like protein to the search, we can quickly see that the database contains information about multiple orthologous genes with transporting functionality of the species *Danio rerio* e.g the zebra fish who belongs to the class of *Actinopterygii* e.g. bony fishes (Fig. 4). This confirms that the ortholog annotations generated by the modified pipeline are valid and can be trusted.



VRLLAGKGSLRKVRIKNMSVPFAYISVHRQGRIVLNYRYCYGTVQNTFEVD
LLHQSCDLT

Gene Info | View Fasta

Esox lucius

LOC105023572 (A0A3P9A1B5) Transporter 623 IPR000175 37272

upkws: integral component of membrane; symporter activity
UniProt: A0A3P9A1B5 Similarity: Belongs to the sodium:neurotransmitter symporter (SNF) (TC 2.A.22) family.
entrezgene: LOC105023572 sodium- and chloride-dependent GABA transporter 2-like
entrezprotein: 105023572
GeneID: 105023572
Entrez: 742193101
KEGGgene: els:105023572
ensemblp: ENSELUG00000013154
Molecular Function: symporter activity
Cellular Component: integral component of membrane
Ensembl: ENSELUG00000013154 sodium- and chloride-dependent GABA transporter 2-like [Source:NCBI gene;Acc:105023572]

Orthologs in example species | Get Ortholog Groups

search species (by name)

Danio rerio (leopard danio;zebra danio;zebra fish) - group 298155at7898 at Actinopterygii level

1	zgc:158427 (A1A5Z1) A1A5Z1_DANRE	579	IPR000175 37272
2	si:ch211-132b12.6 (Q1LVI6) Transporter	577	IPR000175 37272
3	si:ch211-132b12.1 (Q1LVJ1) Transporter	586	IPR000175 37272
4	si:ch211-132b12.2 (Q1LVJ0) Transporter	582	IPR000175 37272
5	si:ch211-132b12.3 (Q1LVI9) Transporter	577	IPR000175 37272

Ciona intestinalis (sea vase;yellow sea squirt) - group 1199573at33208 at Metazoa level
LOC100182269 sodium- and chloride-dependent taurine transporter isoform X2 405

Figure 4: Query result of the previously shown ortholog from the OrthoDB web interface.

Discussion

The initial setup tutorial provided on the transXpress GitHub page appears comprehensive and includes informative examples [8]. However, the setup process and running the pipeline without modifications turned out to be more challenging than expected, which can be attributed to the numerous dependencies and libraries incorporated within the workflow. This is still surprising, considering that the pipeline is based on a scientific article published in BMC Bioinformatics, a reputable peer-reviewed journal where one could expect peer-testing.

After resolving the conflicts and major errors encountered earlier, establishing the connection between OrthoDB and the transXpress pipeline became feasible, thanks to the clear documentation provided for OrthoDB data structures. By adding a few extra rules to the Snakefile and utilising some rules from the original pipeline, it became convenient to include additional annotations regarding orthologs. However, further improvements could be made to the modified pipeline. For instance, incorporating an option to search for orthologs from a specific species or a species group and annotating the three or even five best matches from the database would enhance its functionality. However, implementing these features would require a more intricate set of rules, as simple line-by-line comparisons would no longer yield satisfactory results.

Author contribution

Nicolas was mainly responsible for debugging the original transXpress pipeline and tried running the process with Trinity assembly. Mostly designed the connection between transXpress pipeline and OrthoDB and made important edits to the original Snakefile. Karl Hendrik in parallel ran the entire pipeline using rnaSPAdes assembly using helpful edits made by Nicolas, was responsible for the final report - reading relevant papers, writing the text, creating the visualisations and designing the final document in LaTeX. Also published the project results to a GitHub fork of transXpress [11].

References

[1] Rory Stark, Marta Grzelak, and James Hadfield. “RNA sequencing: the teenage years”. en. In: *Nature Reviews Genetics* 20.11 (Nov. 2019). Number: 11 Publisher: Nature Publishing Group, pp. 631–656. ISSN: 1471-0064. DOI: `10.1038/s41576-019-0150-2`. URL: `https://www.nature.com/articles/s41576-019-0150-2` (visited on 05/29/2023).

[2] M. P. Torrens-Spence, T. R. Fallon, and J. K. Weng. “Chapter Four - A Workflow for Studying Specialized Metabolism in Nonmodel Eukaryotic Organisms”. en. In: *Methods in Enzymology*. Ed. by Sarah E. O’Connor. Vol. 576. Synthetic Biology and Metabolic Engineering in Plants and Microbes Part B: Metabolism in Plants. Academic Press, Jan. 2016, pp. 69–97. DOI: `10.1016/bs.mie.2016.03.015`. URL: `https://www.sciencedirect.com/science/article/pii/S0076687916001270` (visited on 05/29/2023).

[3] *NCBI TRANSCRIPTOMIC SRA*. 2023. URL: `https://www.ncbi.nlm.nih.gov/sra/?term=TRANSCRIPTOMIC%5BSource%5D` (visited on 05/29/2023).

[4] *NCBI Sequence Set Browser*. 2023. URL: `https://www.ncbi.nlm.nih.gov/Traces/wgs/?view=TSA` (visited on 05/29/2023).

[5] Ana Conesa et al. “A survey of best practices for RNA-seq data analysis”. In: *Genome Biology* 17.1 (Jan. 2016), p. 13. ISSN: 1474-760X. DOI: `10.1186/s13059-016-0881-8`. URL: `https://doi.org/10.1186/s13059-016-0881-8` (visited on 05/29/2023).

[6] Timothy R. Fallon et al. “transXpress: a Snakemake pipeline for streamlined de novo transcriptome assembly and annotation”. In: *BMC Bioinformatics* 24.1 (Apr. 2023), p. 133. ISSN: 1471-2105. DOI: `10.1186/s12859-023-05254-8`. URL: `https://doi.org/10.1186/s12859-023-05254-8` (visited on 05/29/2023).

[7] Anish M.S. Shrestha, Joyce Emlyn B. Guiao, and Kyle Christian R. Santiago. “Assembly-free rapid differential gene expression analysis in non-model organisms using DNA-protein alignment”. In: *BMC Genomics* 23.1 (Feb. 2022), p. 97. ISSN: 1471-2164. DOI: `10.1186/s12864-021-08278-7`. URL: `https://doi.org/10.1186/s12864-021-08278-7` (visited on 06/02/2023).

[8] *transXpress: a Snakemake pipeline for streamlined de novo transcriptome assembly and annotation*. en. 2023. URL: `https://github.com/transXpress/transXpress` (visited on 06/02/2023).

[9] Ricardo Avila. *E-values and Bit-scores in BLAST*. 2021. URL: `https://ravilabio.info/notes/bioinformatics/e-value-bitscore.html` (visited on 06/02/2023).

[10] *OrthoDB*. 2023. URL: `https://www.orthodb.org/` (visited on 05/29/2023).

[11] *transXpress_OrthoDB*. en. 2023. URL: `https://github.com/khtamkivi/transXpress_OrthoDB` (visited on 06/08/2023).

Appendix

1a: Download OrthoDB data

```
rule download_orthodb:
   """
   Downloads the 2 useful files from OrthoDB and unzips them
   """
   output:
     "OrthoDB/odb11v0_all_fasta.tab",
     "OrthoDB/odb11v0_genes.tab"
   log:
     "logs/download_orthodb.log"
   conda:
     "envs/blast.yaml"
   params:
     memory="2",memory_slurm="2000"
   threads:
     1
   shell:
     """
     wget --directory-prefix OrthoDB
     "https://data.orthodb.org/download/odb11v0_all_fasta.tab.gz" &> {log}
     wget --directory-prefix OrthoDB
     "https://data.orthodb.org/download/odb11v0_genes.tab.gz" &> {log}
     gunzip OrthoDB/odb11v0_all_fasta.tab.gz &>> {log}
     gunzip OrthoDB/odb11v0_genes.tab.gz &>> {log}
     """
```

1b: Filtering the OrthoDB data

```
rule filter_orthodb:
  """
  Removes all the unnamed genes
  """
  input:
    fasta="OrthoDB/odb11v0_all_fasta.tab",
    genes="OrthoDB/odb11v0_genes.tab"
  output:
    "OrthoDB/filtered_db.fasta"
  log:
    "logs/filter_orthodb.log"
  params:
    memory="2",memory_slurm="2000"
  threads:
    1
  run:
    genes=open('OrthoDB/odb11v0_genes.tab', 'r')
    fasta=open('OrthoDB/odb11v0_all_fasta.tab', 'r')
    db=open("OrthoDB/filtered_db.fasta",'w')
    while True:
      l1=genes.readline()
      l2=fasta.readline()
      l3=fasta.readline()
      if l1=="" or l2=="" or l3=="":
        genes.close()
        fasta.close()
        db.close()
        break
      t=l1.split("\t")
      if t[-1]!='hypothetical protein\n' and (not t[-1].isupper())
      and ("uncharacterized" not in t[-1]):
        db.write(l2[:-1]+"\t"+t[-1])
        db.write(l3)
```

1c: Creating the local BLAST database

```
rule makedb_orthodb:
  """
  Create the BlastDB for faster queries
  """
  input:
    filtered_db="OrthoDB/filtered_db.fasta"
  output:
    "OrthoDB/BlastDB/db.pot"
  log:
    "logs/makedb_orthodb.log"
  conda:
    "envs/blast.yaml"
  params:
    memory=2, memory_slurm="2000"
  threads:
    1
  shell:
    """
    makeblastdb -in OrthoDB/filtered_db.fasta -dbtype prot -out OrthoDB/BlastDB/db
    """
```

1d: Running parallel comparisons with the OrthoDB data

```
rule orthodb_blastp_parallel:
  """
  Runs blast search on OrthoDB database in parallel.
  """
  input:
    fasta="annotations/chunks_pep/{index}.pep",
    db="OrthoDB/BlastDB/db.pot"
  output:
    output="annotations/orthodb_blastp/{index}.out",
  log:
    "logs/orthodb_blastp{index}.log"
  conda:
    "envs/blast.yaml"
  params:
    memory="4",memory_slurm="4000"
  threads:
    2
  shell:
    """
    blastp -query {input[fasta]} -db OrthoDB/BlastDB/db
    -num_threads {threads} -evalue {config[e_value_threshold]}
    -task blastp-fast -max_hsps 1 -max_target_seqs 1 -outfmt
    '6 qseqid stitle bitscore' -out {output.output} &> {log}
    """
```

1e: Modifications to the annotated_fasta rule

```
rule annotated_fasta:
  """
  Puts annotations in the headers of transcripts/proteins in the
  .fasta/.pep transcriptome files.
  """
  input:
  ...
    orthodb_results="annotations/orthodb_blastp_pep.out"
  ...
  run:
  ...
    orthodb_annotations = {}
  ...
```

```
## Load orthodb results
      print ("Loading orthodb results from", input["orthodb_results"],
      file=log_handle)
      with open(input["orthodb_results"]) as input_handle:
        csv_reader = csv.reader(input_handle, delimiter="\t")
        for row in csv_reader:
          if (len(row) < 3): continue
          orthodb_annotations[row[0]]= row[1] + ", bitscore :" + str(row[2])

  ...
    if record.id in orthodb_annotations:
        record.description += "; Ortholog: " + orthodb_annotations.get(record.id)
  ...
```