Orphan Prediction

Release 0.0.1

Urminder Singh

Dec 24, 2021

CONTENTS

CHAPTER

ONE

INTRODUCTION

MIND: *ab initio* gene predictions by **M**AKER combined with gene predictions **IN**ferred **D**irectly from alignment of RNA-Seq evidence to the genome.

BIND: *ab initio* gene predictions by **B**RAKER combined with gene predictions **IN**ferred **D**irectly from alignment of RNA-Seq evidence to the genome.

Overview

You will need an diverse, orphan-enriched RNA-Seq dataset from NCBI-SRA (as available) and the genome sequence of your species of interest. Run an *ab initio* prediction either by BRAKER or by MAKER (we provide Singularity containers to facilitate installation of these pipelines). Run the evidenced-based Direct Inference pipeine. Combine the *ab initio* predictions with the Direct Inference predictions using Mikado. Evaluate your predictions

CHAPTER

TWO

TUTORIAL

See the detailed scripts here **.**

2.1 Contents

2.1.1 Find an Orphan-Enriched RNA-Seq dataset from NCBI-SRA

See the detailed scripts here **.**

Notes:

This step is optional, and applies to species with >40 available SRA RNA-Seq runs. It is designed to provide a diverse dataset, rich in representation of all genes including orphan genes.

If the species you are annotating has fewer than ~40 SRA RNA-Seq runs, download it all and go on to one of the ab initio prediction methods.

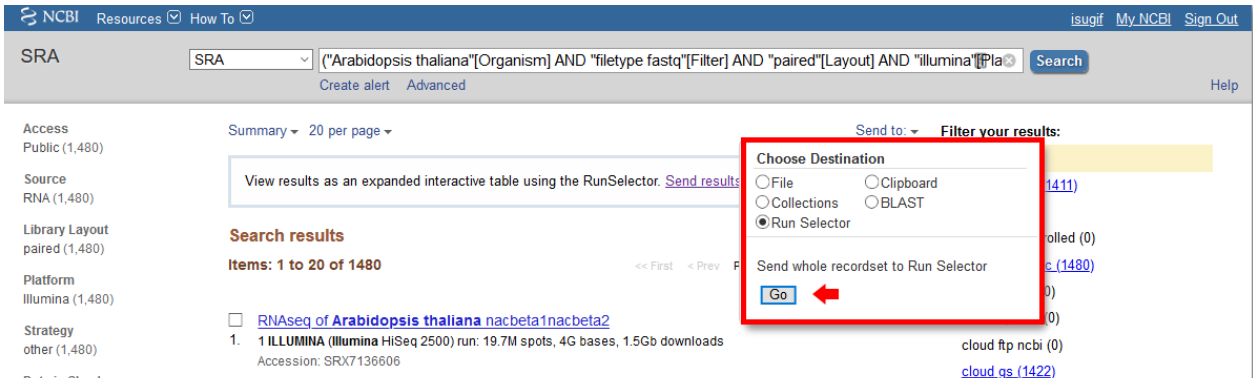
If you are relying solely on RNA-Seq data that you generate yourself, *maximize* representation of genes by including diverse conditions like reproductive tissues, biotic and abiotic stresses, and a variety of other tissue and organ types.

Select RNA-Seq SRR ID from NCBI-SRA website for your desired species:

Go to NCBI SRA page and search with "SRA Advanced Search Builder". This allows you to build a query and select the Runs that satisfy certain requirements. For example:

```
("Arabidopsis thaliana"[Organism] AND
  "filetype fastq"[Filter] AND
    "paired"[Layout] AND
    "illumina"[Platform] AND
    "transcriptomic"[Source])
```

Then export the results to "Run Selector" as follows:



SRA ("Arabidopsis thaliana"[Organism] AND "filetype fastq"[Filter] AND "paired"[Layout] AND "illumina"[Pla

Access
Public (1,480)

Source
RNA (1,480)

Library Layout
paired (1,480)

Platform
Illumina (1,480)

Strategy
other (1,480)

View results as an expanded interactive table using the RunSelector. Send results

Search results

Items: 1 to 20 of 1480

Choose Destination
File / Clipboard / Collections / BLAST / [✓ Run Selector]

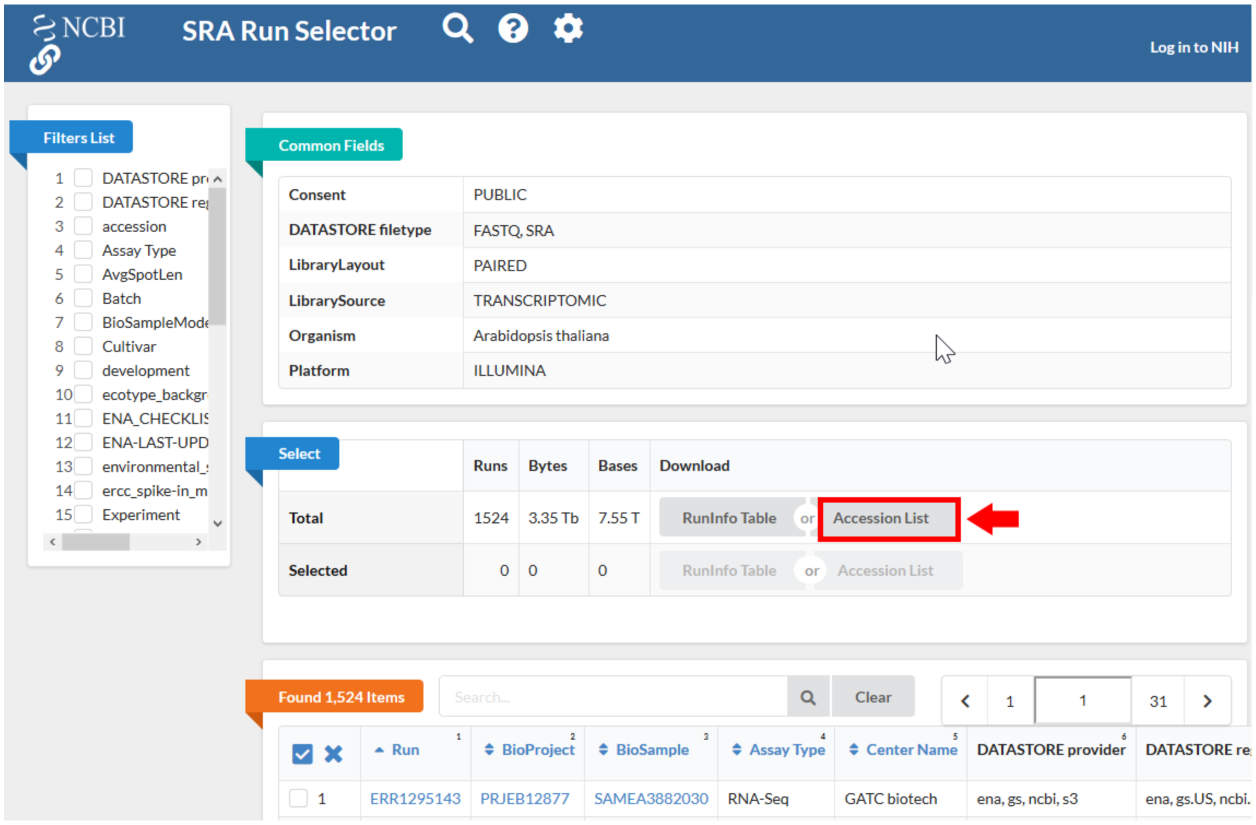
Send whole recordset to Run Selector

Go

RNAseq of **Arabidopsis thaliana** nacbeta1nacbeta2
1. 1 ILLUMINA (Illumina HiSeq 2500) run: 19.7M spots, 4G bases, 1.5Gb downloads
Accession: SRX7136606

cloud ftp ncbi (0)
cloud gs (1422)

Clicking the "Accession List" allows you to download all the SRR IDs in a text file format.



Filters List: 1 DATASTORE pr, 2 DATASTORE re, 3 accession, 4 Assay Type, 5 AvgSpotLen, 6 Batch, 7 BioSampleMode, 8 Cultivar, 9 development, 10 ecotype_backgr, 11 ENA_CHECKLIS, 12 ENA-LAST-UPD, 13 environmental_, 14 ercc_spike-in_m, 15 Experiment

Consent	PUBLIC
DATASTORE filetype	FASTQ, SRA
LibraryLayout	PAIRED
LibrarySource	TRANSCRIPTOMIC
Organism	Arabidopsis thaliana
Platform	ILLUMINA

	Runs	Bytes	Bases	Download
Total	1524	3.35 Tb	7.55 T	RunInfo Table or Accession List
Selected	0	0	0	RunInfo Table or Accession List

	Run	BioProject	BioSample	Assay Type	Center Name	DATASTORE provider	DATASTORE re
1	ERR1295143	PRJEB12877	SAMEA3882030	RNA-Seq	GATC biotech	ena, gs, ncbi, s3	ena, gs.US, ncbi.

Download RNA-Seq raw reads:

```
while read line; do
   ./01_runSRAdownload.sh ${line};
done < SRR_Acc_List.txt
```

Note: depending on how much data you find, this can take a lot of time and resources (disk usage). You may need to narrow down and select only a subset of total SRA runs. Another way to choose datasets with maximal orphan representation is to select SRRs most likely to be diverse (eg: stress response, flowering tissue, or SRRs with very deep coverage).

Download the CDS sequences for the organism you are annotating, and build transcriptome for kallisto index:

```
wget https://www.arabidopsis.org/download_files/Genes/Araport11_genome_release/Araport11_
↪blastsets/Araport11_genes.201606.cds.fasta.gz
gunzip Araport11_genes.201606.cds.fasta.gz
kallisto index -i ARAPORT11cds Araport11_genes.201606.cds.fasta
```

For each SRR ID, run the Kallisto qualitification:

```
while read line; do
   02_runKallisto.sh ARAPORT11cds ${line};
done < SRR_Acc_List.txt
```

Merge the tsv files containing counts and TPM:

```
03_joinr.sh *.tsv >> kallisto_out_tair10.txt
```

Note: For every SRR id, the file contains 3 columns, `effective length`, `estimated counts` *and* `transcript per million`.

Run phylostratr to infer phylostrata of genes, and identify orphan genes:

1. Build a phylogenic tree for your species, and download proteins sequences for target species:

   ```
   ./04_runPhylostratRa.R
   ```

2. Run Blast to compare query proteins and target proteins:

   ```
   while read line; do
   # 3702 is taxid for our focal species A.thaliana.
   # You can replace your own protein sequences for your focal species if protein␣
   ↪downloaded from uniprot is not your desired version.
      05_runBLASTp.sh ${line} 3702.faa;
   done < uniprot_list.txt
   ```

3. Process Blast output and stratify phylostrata level for each query gene:

   ```
   ./06_runPhylostratRb.R
   ```

   Note: Phylostratr will run protein blast automatically if it doesn't detect blast database and output files in working directory, so you can skip step2 to obtain blast output. However, it may takes a long time depend on the number of species and the size of your query genes. You can also use `strata_diamond` *instead of* `strata` *in* `06_runPhylostratRb.R`, *it will use Diamond Blast instead of Blast-plus. Diamond blast is much faster than Blast-plus, but may not sensitive as Blast-plus.*

Select Orphan-rich RNA-Seq data:

Once the orphan (species-specific) genes are identified, count the total number of orphan genes expressed (>1TPM) in each SRR, rank them based on % orphan expressed. Depending on how much computational resources you have, you can select the top X number of SRRs to use them as evidence for direct inference and as training data.

Note: for Arabidopsis thaliana, we used all of the SRRs that expressed over 60% of the orphan genes (=38 SSRs).

2.1.2 *Ab initio* predictions

Pick one of the 2 *ab initio* **prediction methods below:**

Run BRAKER

- Align RNA-Seq with splice aware aligner (STAR or HiSat2 preferred; HiSat2 used here)
- Generate BAM file for each SRA-SRR id, merge them to generate a single sorted BAM file
- Run BRAKER

Run MAKER

- Align RNA-Seq with splice aware aligner (STAR or HiSat2 preferred; HiSat2 used here)
- Generate BAM file for each SRA-SRR id, merge them to generate a single sorted BAM file
- Run Trinity to generate transcriptome assembly using the BAM file
- Run TransDecoder on Trinity transcripts to predict ORFs and translate them to protein
- Run MAKER with transcripts (Trinity), proteins (TransDecoder and SwissProt), in homology-only mode
- Use the MAKER predictions to train SNAP and AUGUSTUS. Self-train GeneMark
- Run second round of MAKER with the above (SNAP, AUGUSTUS, and GeneMark) ab initio predictions plus the results from previous MAKER rounds.

BRAKER prediction

See the scripts used for BRAKER here **.**

- Input files for Braker

```
RNA-Seq raw reads fastq files: *_1.fastq.gz, *_2.fastq.gz
reference genome fasta file: TAIR10_chr_all.fas
```

- To simplify handling of files, combine all the forward reads to one file and all the reverse reads to another.

```
cat *_1.fastq.gz >> forward_reads.fq.gz
cat *_2.fastq.gz >> reverse_reads.fq.gz
```

- Do RNA-Seq alignment by Hisat2:

```
./01_runHisat2.sh forward_reads.fq.gz reverse_reads.fq.gz TAIR10_chr_all.fas
```

- Run BRAKER by your installed tools:

```
./02_runBraker.sh TAIR10_chr_all.fas TAIR10_rnaseq.bam
```

- Run BRAKER by singularity comtainer:

  If you have difficulty to install BRAKER, we suggest you to download the singularity container **for this step.**

  To use the container, follow the instruction of the container to install and copy directory.

```
./02_runBraker_singularity.sh TAIR10_chr_all.fas TAIR10_rnaseq.bam
```

Note: We only used RNA-Seq as evidence in our cases. BRAKER also accept protein evidence in different prediction mode. However, It is not always best to use all evidence. You can test and determine which one is best for the species you are working on. See the difference among BRAKER modes on their main page.

MAKER predictions

See the scripts used for MAKER here **.**

Input files for Maker

```
RNA-Seq raw reads fastq files: *_1.fastq.gz, *_2.fastq.gz
reference genome fasta file: TAIR10_chr_all.fas
```

Merge RNA-Seq raw reads

To simplify handling of files, combine all the forward reads to one file and all the reverse reads to another.

```
cat *_1.fastq.gz >> forward_reads.fq.gz
cat *_2.fastq.gz >> reverse_reads.fq.gz
```

Run trinity to predict transcripts and their inferred proteins

1. Run trinity for *de novo* transcriptome assembly:

```
./01_runTrinity.sh forward_reads.fq.gz reverse_reads.fq.gz
```

Note: You will get the transcripts fasta file in trinity_run folder.

2. Predict CDSs from transcriptome:

```
./02_runTransDecoder.sh trinity.fasta
```

Note: You will get the protein sequence (`trinity.fasta.transdecoder.pep` *) in working directory.*

MAKER requires five (non-automated) steps

1. Generate the CTL files:

```
module load GIF/maker
module rm perl/5.22.1
maker -CTL
```

   This will generate 3 CTL files (`maker_opts.ctl`, `maker_bopts.ctl` and `maker_exe.ctl`), you will need to edit them to make changes to the MAKER run. For the first round, change these lines in `maker_opts.ctl` file:

```
genome=TAIR10_chr_all.fas
est=trinity.fasta
protein=trinity.fasta.transdecoder.pep
est2genome=1
protein2genome=1
TMP=/dev/shm
```

2. Execute MAKER `03_maker_start.sh` in a slurm file. It is essential to request more than 1 node with multiple processors to run this efficiently.

```
# Define a base name for maker output folder as the first argument.
./03_maker_start.sh maker_case
```

3. Upon completion, train SNAP and AUGUSTUS:

```
#Use the same base name as previous step for first argument.
./04_maker_process.sh maker_case
```

4. Train GeneMark with genome sequence:

```
./05_runGeneMark.sh TAIR10_chr_all.fas
```

5. Once complete, modify the following lines in `maker_opts.ctl` file:

```
snaphmm=maker.snap.hmm
gmhmm=gmhmm.mod
# Define a species as you want, but the name should not be existing in the augustus/
↪config/species folder.
augustus_species=maker_20171103
```

   Then, `03_maker_start.sh` again:

```
# Use the same base name as previous step for first argument.
./03_maker_start.sh maker_case
```

6. Finalize predictions:

```
./06_maker_finalize.sh maker_case
```

   You will get the predicted gene models (`maker_case.gff`), protein sequences (`maker_case.maker.proteins.fasta`) and transcript sequence (`maker_case.maker.transcripts.fasta`) in the working directory.

2.1.3 Direct Inference evidence-based predictions

Automated pipeline

We provide an automated pipeline using pyrpipe and snakemake. This pipeline can be easily configured and executed in an automated manner (**HIGHLY RECOMMENDED**).

The automated pipeline can be easily scaled on an HPC by executing multiple samples in parallel. We recommend using this implementation of the pipeline for simplicity of use and reproducibility of results.

The pipeline can easily be modified and shared. For additional information see: pyrpipe (https://doi.org/10.1093/nargab/lqab049) and orfipy (https://doi.org/10.1093/bioinformatics/btab090) pipelines.

Direct Inference prediction by steps

If you prefer to run the Direct Inference pipeline step by step, the details are explained below.

Direct Inference predictions by steps

See the scripts used for Direct Inference prediction here .

Singularity container

For installation, we suggest that you to download the singularity container . This container includes all tools required to run the pipeline.

To use the container, add `singularity run --cleanenv evidence.sif` before the tools in each script. For example:

```
singularity run --cleanenv evidence.sif hisat2
```

Input files for DirectInf

```
RNA-Seq raw reads fastq files: *_1.fastq.gz, *_2.fastq.gz
reference genome fasta file: TAIR10_chr_all.fas
list of input prediction: list.txt
```

Do RNA-Seq alignment by Hisat2

```
while read line; do
      ./01_runHisat2.sh ${line} TAIR10_chr_all.fas;
done < SRR_Acc_List.txt
```

Note: Cufflinks and Class2 may takes a long time to process BAM file with deep depth region. Better to generate single bam file for each RNA-Seq data for next step.

Run Transcriptome assemblies using Class2, Cufflinks and Stringtie

```
while read line; do
        ./02_runClass2.sh ${line}_sorted.bam;
        ./03_runCufflinks.sh ${line}_sorted.bam;
        ./04_runStringtie.sh ${line}_sorted.bam;
 done < SRR_Acc_List.txt
```

Note: Class2, Cufflinks and StringTie generates a gtf format transcripts for each SRR sample. You can use more transcriptome assembler as you need.

Generate high confidence splice sites

```
./05_runPortCullis.sh TAIR10_rnaseq.bam
```

Note: The merged bam file (`TAIR10_rnaseq.bam` *) obtained from the process of braker. You can also provide multiple single bam files got from 01_runHisat2.sh , but it takes some time to merge bam files. It's faster to provide merged bam if you already run braker.*

Here generate bed and tab file in `portcullis_out/3-filt`, we only use bed file.

Consolidate all the transcripts, and predict potential protein coding sequence

1. Make a configure file and prepare transcripts:

   You should prepare a `` `list.txt `` as below to include gtf path (1st column), gtf abbrev (2nd column), stranded-specific or not (3rd column):

```
SRRID1_class.gtf    cs_SRRID1    False
SRRID1_cufflinks.gtf cl_SRRID1     False
SRRID1_stringtie.gtf st_SRRID1    False
SRRID2_class.gtf    cs_SRRID2    False
SRRID2_cufflinks.gtf cl_SRRID2     False
SRRID2_stringtie.gtf st_SRRID2    False
...
```

   Then run the script as below:

```
./06_runMikado_round1.sh TAIR10_chr_all.fas junctions.bed list.txt DI
```

   This will generate `DI_prepared.fasta` file that will be used for predicting ORFs in the next step.

2. Predict potential CDS from transcripts:

```
./07_runTransDecoder.sh DI_prepared.fasta
```

   We will use `DI_prepared.fasta.transdecoder.bed` in the next step.

   Note: Here we only kept complete CDS for next step. You can revise `07_runTransDecoder.sh` *to use both incomplete and complete CDS if you need.*

3. Pick best transcripts for each locus and annotate them as gene:

```
./08_runMikado_round2.sh DI_prepared.fasta.transdecoder.bed DI
```

This will generate:

```
mikado.metrics.tsv
mikado.scores.tsv
DI.loci.gff3
```

Optional: Filter out transcripts with redundant CDS

```
./09_rm_redundance.sh DI.loci.gff3 TAIR10_chr_all.fas
```

Optional: Filter out transcripts whose predicted proteins mapped to transposon elements

```
./10_TEsorter.sh filter.pep.fa DI.loci.gff3
```

Note: `filter.pep.fa` *is an output from previous step for removing redundant CDSs. You can also use all protein sequence if you don't want to remove redundant CDSs.*

2.1.4 Combine *ab initio* and Direct Inference evidence-based predictions

Pick one of the 2 combined predictions below according to your choice in ab initio

1. **Merge BRAKER with Direct Inference (BIND):**

   Use Mikado to combine BRAKER-generated predictions with Direct Inference evidence-based predictions.

2. **Merge MAKER with Direct Inference (MIND):**

   Use Mikado to combine MAKER-generated predictions with Direct Inference evidence-based predictions.

BIND prediction

Merge gene predictions of **B** RAKER (`braker-final.gff3`) with gene predictions **IN** ferred **D** irectly (`DI-final.gff3`).

See the scripts used for BIND here **.**

Note: See the details to generate these two predictions in `braker` *and* `DirectInf` *.*

Input files for BIND

```
BRAKER prediction: braker-final.gff3
Direct Inference prediction: DI-final.gff3
reference genome fasta file: TAIR10_chr_all.fas
splice junction file: junctions.bed # from DirectInf step
list of input prediction: list_BIND.txt
```

Consolidate all the transcripts from BRAKER and DirInf, and predict potential protein coding sequence

1. Make a configure file and prepare transcripts:

   You should prepare a `list_BIND.txt` as below to include gtf path (1st column), gtf abbrev (2nd column), stranded-specific or not (3rd column):

   ```
   braker-final.gff3	br	False
   DI-final.gff3	DI	False
   ```

   Then run the script as below:

   ```
   ./01_runMikado_round1.sh TAIR10_chr_all.fas junctions.bed list_BIND.txt BIND
   ```

   This will generate *BIND_prepared.fasta* file that will be used for predicting ORFs in the next step.

   Note: `junctions.bed` *is the same file generate from DirectInf step.*

2. Predict potential CDS from transcripts:

   ```
   ./02_runTransDecoder.sh BIND_prepared.fasta
   ```

   We will use *BIND_prepared.fasta.transdecoder.bed* in the next step.

   Note: Here we only kept complete CDS for next step. You can revise `02_runTransDecoder.sh` *to use both incomplete and complete CDS if you need.*

3. Pick best transcripts for each locus and annotate them as gene:

   ```
   ./03_runMikado_round2.sh BIND_prepared.fasta.transdecoder.bed BIND
   ```

   This will generate:

   ```
   mikado.metrics.tsv
   mikado.scores.tsv
   BIND.loci.gff3
   ```

Optional: Filter out transcripts with redundant CDS

```
./04_rm_redundance.sh BIND.loci.gff3 TAIR10_chr_all.fas
```

Optional: Filter out transcripts whose predicted proteins mapped to transposon elements

Note: `filter.pep.fa` *is an output from previous step for removing redundant CDSs. You can also use all protein sequence if you don't want to remove redundant CDSs.*

MIND prediction

Merge gene predictions of **M** AKER (`Maker-final.gff3`) with gene predictions **IN** ferred **D** irectly (`DI-final.gff3`).

See the scripts used for MIND here **.**

Note: See the details to generate these two predictions in `maker` *and* `DirectInf` *.*

Input files for MIND

```
BRAKER prediction: maker-final.gff3
Direct Inference prediction: DI-final.gff3
reference genome fasta file: TAIR10_chr_all.fas
splice junction file: junctions.bed # from DirectInf step
list of input prediction: list_MIND.txt
```

Consolidate all the transcripts from MAKER and DirInf, and predict potential protein coding sequence

1. Make a configure file and prepare transcripts:

   You should prepare a `list_MIND.txt` as below to include gtf path (1st column), gtf abbrev (2nd column), stranded-specific or not (3rd column):

   ```
   maker-final.gff3    mk    False
   DI-final.gff3 DI     False
   ```

   Then run the script as below:

   ```
   ./01_runMikado_round1.sh TAIR10_chr_all.fas junctions.bed list_MIND.txt MIND
   ```

   This will generate *MIND_prepared.fasta* file that will be used for predicting ORFs in the next step.

   Note: `junctions.bed` *is the same file generate from DirectInf step.*

2. Predict potential CDS from transcripts:

```
./02_runTransDecoder.sh MIND_prepared.fasta
```

We will use *MIND_prepared.fasta.transdecoder.bed* in the next step.

Note: Here we only kept complete CDS for next step. You can revise `02_runTransDecoder.sh` *to use both incomplete and complete CDS if you need.*

3. Pick best transcripts for each locus and annotate them as gene:

```
./03_runMikado_round2.sh MIND_prepared.fasta.transdecoder.bed MIND
```

This will generate:

```
mikado.metrics.tsv
mikado.scores.tsv
MIND.loci.gff3
```

Optional: Filter out transcripts with redundant CDS

```
./04_rm_redundance.sh MIND.loci.gff3 TAIR10_chr_all.fas
```

Optional: Filter out transcripts whose predicted proteins mapped to transposon elements

Note: `filter.pep.fa` *is an output from previous step for removing redundant CDSs. You can also use all protein sequence if you don't want to remove redundant CDSs.*

2.1.5 Downstream analysis to evaluate and annnotate gene predictions

See the scripts used for downstream evaluation here **.**

1. Run `Mikado Compare` to compare prediction with known annotation

2. Run `Salmon` to quantify predictions

3. Run `Ribotricer` to verify translation signal for predicted protein coding genes

4. Run `BUSCO` to see how well the conserved genes are represented in your final predictions

5. Run `OrthoFinder` to find and annotate orthologs present in your predictions

6. Run `phylostratR` to find orphan genes in your predictions [For theory and details, see: https://doi.org/10.1093/bioinformatics/btz171]

7. You can also add functional annotation to your genes using homology and `InterProScan` (We didn't have this step in our paper).

2.1.6 Tools used for prediction

Tool	Purpose
SRA Tools (v. 2.9.6)	SRA access
Hisat2 (v. 2.2.0)	Alignment
STAR (v. 2.7.7a)	Alignment
Kallisto (v. 0.46.2)	Quantification
Samtools (v. 1.10)	Tools
CLASS2 (v. 2.1.7)	Transcript Assembly
Stringtie (v. 1.3.3)	Transcript Assembly
Cufflinks (v. 2.2.1)	Transcript Assembly
Trinity (v. 2.6.6)	Transcript Assembly
Porticullis (v. 1.2.2)	Tools
Transdecoder (v. 3.0.1)	CDS prediction
Mikado (v. 2.0)	Direct Inference prediction
Phylostratr (v. 0.2.0)	Phylostratigraphy
BLAST (v. 3.11.0)	Tools
Braker (v. 2.1.2)	*Ab initio* prediction
Maker (v. 2.31.10)	*Ab initio* prediction
GMAP-GSNAP (v. 2019-05-12)	Alignment
GeneMark (v. 4.83)	*Ab initio* Prediction