Northern Wild Rice Genome Annotation

Marissa Macchietto (mmacchie@umn.edu)
Tom Kono (konox006@umn.edu)
Minnesota Supercomputing Institute
University of Minnesota

2020-05-01

Contents

1 Overview

This project is annotation of the Northern Wild Rice (*Zizania palustris*) genome assembly. The principal investigator for the project is Dr. Jennifer Kimball at the University of Minnesota.

1.1 Formatting in this Report

This report will contain formatting cues to denote special pieces of information. Normal text will appear in this font.

`Paths and interactive commands appear in monospace.`

```
#!/bin/bash
echo "Scripts will appear as highlighted blocks"
```

1.2 Code Repository

All code used to generate the results for this project is kept in a UMN GitHub repository. It is private-access until publication, please contact Marissa Macchietto (mmachie@umn.edu) or Tom Kono (konox006@umn.edu) for access.

The link to the UMN GitHub repository is https://github.umn.edu/MSI-RIS/Kimball_WR_Annotation.

All directory and file paths that are not explicitly listed as MSI paths are given relative to the GitHub repository.

2 Input Data

2.1 RNAseq Data

Short-read RNAseq data were obtained from the University of Minnesota Genomics Center. The data were delivered as a single pair of FASTQ files for each of eight sampled tissues. The data are available on MSI systems at the following directory:

`/home/jkimball/data_release/umgc/novaseq/190828_A00223_0198_BHCMFVDRXX/Kimball_Project_003`

The tissues sampled for the study are as follows:

Tissue	Number of Read Pairs
Female	60,868,914
Flower	51,064,910
Leaf	49,202,727
Male	54,403,083
Root	56,674,486
Seed	51,243,914
Sheath	62,826,860
Stem	60,470,690

2.2 Publicly-Available Genome Data

Publicly-available genome data was used for the identification of orthologous gene groups (see below). The data were obtained from Ensembl Plants 46. The files required for orthologous gene group identification are the peptide sequence files. The species names and assembly/annotation versions are given below:

Species	Assembly/Annotation Version
Aegilops tauschii	`Aet 4.0`
Brachypodium distachyon	`Brachypodium_distachyon_v3.0`
Dioscorea rotundata	`TDr96_F1_Pseudo_Chromosome_v1.0`
Eragrostis tef	`ASM97063v1`
Hordeum vulgare	`IBSC_v2`

Species	Assembly/Annotation Version
Leersia perrieri	`Lperr_V1.4`
Musa acuminata	`ASM31385v1`
Oryza sativa Japonica	`IRGSP-1.0`
Panicum hallii	`PHallii_v3.1`
Saccharum spontaneum	`Sspon.HiC_chr_asm`
Setaria italica	`Setaria_italica_v2.0`
Sorghum bicolor	`Sorghum_bicolor_NCBIv3`
Triticum aestivum	`IWGSC`
Zea mays	`B73_RefGen_v4`

These were downloaded to the following directory on MSI:

`/home/jkimball/shared/WR_Annotation/Orthofinder/Ensembl_Plants_Grasses`

3 Workflow

The methods described in this document will provide scripts and commands that perform the analyses reported in the manuscript text.

3.1 RNAseq QA/QC

Quality control summaries of the RNAseq reads were generated with FastQC version 0.11.8. This is implemented in the `Scripts/Jobs/RNASeq_FastQC.sh` script, reproduced below:

```
#!/bin/bash -l
#PBS -l nodes=1:ppn=6,mem=16gb,walltime=12:00:00
#PBS -A jkimball
#PBS -W group_list=jkimball
#PBS -m abe
#PBS -M konox006@umn.edu
#PBS -q mesabi

module load fastqc/0.11.8

IN_DIR="/home/jkimball/data_release/umgc/novaseq/190828_A00223_0198_BHCMFVDRXX/Kimball_Project_003"
OUT_DIR="/panfs/roc/scratch/konox006/JKimball/RNASeq/FastQC"

mkdir -p "${OUT_DIR}"
fastqc -t 6 -f fastq -o "${OUT_DIR}" "${IN_DIR}"/*.fastq.gz
```

3.1.1 Low-Quality Base and Adapter Removal

Low-quality bases and adapter contamination were removed with Trimmomatic version 0.33. We removed contamination from the standard Illumina adapters and performed window-based quality trimming. The script that performs trimming with Trimmomatic is available in the `Scripts/Jobs/RNASeq_Trim.sh` script, reproduced below:

```
#!/bin/bash
#PBS -l mem=16gb,nodes=1:ppn=4,walltime=4:00:00
#PBS -A jkimball
#PBS -W group_list=jkimball
#PBS -m abe
#PBS -M konox006@umn.edu
#PBS -q mangi

# Load modules
```

```
module load trimmomatic/0.33

READS_DIR="/home/jkimball/data_release/umgc/novaseq/190828_A00223_0198_BHCMFVDRXX/Kimball_Project_003"
OUT_DIR="/panfs/roc/scratch/konox006/JKimball/RNASeq/Trimmed_Reads"

# Find all R1 files in the reads directory
R1=($(find "${READS_DIR}" -mindepth 1 -maxdepth 1 -type f -name '*R1_001.fastq.gz' | sort -V))

# Get the current one with the PBS array ID
CURR_R1="${R1[${PBS_ARRAYID}]}"
# Build the R2 path by replacing R1 with R2
R2="${R1/R1/R2}"

# Build the samplename from the read name
SAMPLENM=$(basename "${CURR_R1}" | cut -f 1 -d '_')

cd "${OUT_DIR}"
# Trim it
java -jar "${TRIMMOMATIC}"/trimmomatic.jar \
    PE \
    -threads "${PBS_NUM_PPN}" \
    "${CURR_R1}" "${R2}" \
    "${SAMPLENM}_1P.fq.gz" "${SAMPLENM}_1U.fq.gz" "${SAMPLENM}_2P.fq.gz" "${SAMPLENM}_2U.fq.gz" \
    ILLUMINACLIP:$TRIMMOMATIC/adapters/all_illumina_adapters.fa:4:15:7:2:true LEADING:3 TRAILING:3 SLIDINGWINDOW:4:15
 ↪   MINLEN:18
```

3.1.2 Ribosomal RNA Quantification and Removal

Ribosomal RNA sequences were removed from the libraries for the purposes of transcriptome assembly. While ribosomal RNA is technically part of the transcriptome, they are highly repetitive and abundant, which causes the required resources for assembly to be unfeasible. We screened for ribosomal RNA sequences and removed them.

3.1.2.1 Ribosomal Sequence Database

A database of ribosomal sequences was downloaded from the SILVA database, current as of 2019-02-20 (release 132). The sequences were curated by the SILVA team, and the curation process includes collapsing of identical gene sequences and removal of unalignable portions of cloned sequences. We further processed the sequence collection by de-duplication with the `dedupe2.sh` tool from the BBTools suite version 38.39. We followed a post by Brian Bushnell on SeqAnswers (http://seqanswers.com/forums/showthread.php?t=63258):

```
# On MSI interactive compute session
dedupe2.sh \
    in=SILVA_132_LSU_SSU_Ref.fasta.gz \
    out=SILVA_132_LSU_SSU_Ref_Dedup.fasta.gz
```

We then used the `kcompress.sh` tool from BBTools to collapse the de-duplicated sequences by shared k-mers of 31bp, keeping only sequences that share k-mers that show up at least 5 times:

```
# On MSI interactive compute session
kcompress.sh \
    in=SILVA_132_LSU_SSU_Ref_Dedup.fasta.gz \
    out=SILVA_132_LSU_SSU_Ref_Dedup_Kmers.fasta \
    k=31 \
    min=5
```

Finally, we used `bbduk.sh` to trim the collapsed sequences to remove entries that were shorter than 100bp:

```
# On MSI interactive compute session
bbduk.sh \
    in=SILVA_132_LSU_SSU_Ref_Dedup_Kmers.fasta \
    out=SILVA_132_LSU_SSU_Ref_Dedup_Kmers_min100.fasta.gz \
    minlen=100
```

This output file was then used as a reference database to remove ribosomal RNA contaminants from the *Z. palustris* RNAseq data with `bbduk.sh`. This is implemented in the `Scripts/Jobs/BBDuk_RNA.sh` script, reproduced below:

```
#!/bin/bash -l
#PBS -l nodes=1:ppn=12,mem=31gb,walltime=24:00:00
#PBS -A jkimball
#PBS -W group_list=jkimball
#PBS -m abe
#PBS -M konox006@umn.edu
#PBS -q mesabi

module load java/openjdk-11.0.2

# Use the SILVA database to screen for rRNA contamination. This reference file
# was generated for CHURP!
SILVA="/home/msistaff/public/CHURP_Deps/v0/db/SILVA_132_LSU_SSU_Ref_Dedup_Kmers_min100.fasta.gz"
BBDUK="/home/jkimball/shared/Software/bbmap/bbduk.sh"

# READS_DIR="/home/jkimball/data_release/umgc/novaseq/190828_A00223_0198_BHCMFVDRXX/Kimball_Project_003"
READS_DIR="/panfs/roc/scratch/konox006/JKimball/RNASeq/Trimmed_Reads"
OUT_DIR="/panfs/roc/scratch/konox006/JKimball/RNASeq/rRNA_Screen"

# Make an array of the R1 sequences
R1=($(find "${READS_DIR}" -maxdepth 1 -mindepth 1 -type f -name '*R1_001.fastq.gz' | sort -V))

# Grab the current R1 using the array ID
CURR_R1="${R1[${PBS_ARRAYID}]}"
# Get the basename because we will need it to define output names
R1BNAME=$(basename ${CURR_R1})
R2BNAME="${R1BNAME/_R1_001.fastq.gz/_R2_001.fastq.gz}"
```

```
# run BBDUK
_JAVA_OPTIONS="-Xmx30g" "${BBDUK}" \
    in="${CURR_R1}" \
    in2="${READS_DIR}/${R2BNAME}" \
    ref="${SILVA}" \
    k=25 \
    editdistance=1 \
    out="${OUT_DIR}/${R1BNAME/_R1_001.fastq.gz/_R1_rRNADep.fastq.gz}" \
    out2="${OUT_DIR}/${R2BNAME/_R2_001.fastq.gz/_R2_rRNADep.fastq.gz}" \
    outm="${OUT_DIR}/${R1BNAME/_R1_001.fastq.gz/_R1_rRNAMatch.fastq.gz}" \
    outm2="${OUT_DIR}/${R2BNAME/_R2_001.fastq.gz/_R2_rRNAMatch.fastq.gz}" \
    outs="${OUT_DIR}/${R1BNAME/_R1_001.fastq.gz/_Single_rRNAMatch.fastq.gz}" \
    stats="${OUT_DIR}/${R1BNAME/_R1_001.fastq.gz/_BBDuk_Stats.txt}" \
    ordered=true \
    prealloc=t
```

The proportion of ribosomal RNA in each sample is given in the following table:

Tissue	Ribosomal RNA Content
Female	6.77%
Flower	69.19%
Leaf	30.73%
Male	22.07%
Root	86.38%
Seed	8.38%
Sheath	63.74%
Stem	45.07%

The `_R1_rRNADep.fastq.gz` and `_R2_rRNADep.fastq.gz` reads were concatenated to produce single R1 and R2 FASTQ files for transcriptome assembly, called `WR_8Tissues.R1.trimmed.fastq` and `WR_8Tissues.R2.trimmed.fastq`, respectively.

3.2 Transcriptome Assembly

The concatenated reads that passed the quality control procedure described above were used to assemble the protein-coding transcriptome. The assembly was carried out with Trinity version 2.8.6. The assembly script is available in `Scripts/Jobs/Trinity_k25.sh`, reproduced below:

```
#!/bin/bash
#PBS -l mem=998gb,nodes=1:ppn=32,walltime=96:00:00
#PBS -A jkimball
#PBS -W group_list=jkimball
#PBS -m abe
#PBS -M konox006@umn.edu
#PBS -q ram1t

# Load modules
```

```
module load java/jdk1.8.0_171
module load bowtie2/2.3.4.1.CentOS7
module load samtools/1.7
module load R/3.5.0
module load ncbi_blast+/2.7.1.CentOS7
module load jellyfish/2.1.3
module load python3/3.6.3_anaconda5.0.1
module load salmon/0.14.1

# Define paths
TRINITY="/home/jkimball/shared/Software/trinityrnaseq-v2.8.6/Trinity"
READS_DIR="/panfs/roc/scratch/konox006/JKimball/RNASeq/Trimmed_reads"
# Global scratch is very slow
#OUTPUT_DIR="/panfs/roc/scratch/konox006/JKimball/RNASeq/trinity_k25_Transcriptome_Asm"
OUTPUT_DIR="/home/jkimball/shared/WR_Annotation/Transcriptome_Asm_trinity"

mkdir -p "${OUTPUT_DIR}"
cd "${OUTPUT_DIR}"

# Set some more parameters for the assembly
MIN_CONTIG=250

# Run Trinity
${TRINITY} \
    --seqType fq \
    --max_memory 996G \
    --left "${READS_DIR}/WR_8Tissues.R1.trimmed.fastq" \
    --right "${READS_DIR}/WR_8Tissues.R2.trimmed.fastq" \
    --SS_lib_type RF \
    --CPU 32 \
    --KMER_SIZE 25\
    --monitoring \
    --monitor_sec 30 \
    --min_contig_length ${MIN_CONTIG} \
    --output "${OUTPUT_DIR}"
```

3.3 Genome Annotation

Annotation was carried out with the Funannotate workflow version , which includes repeat masking, *ab initio* and evidence-based gene model prediction, and gene model refining. The transcripts assembled by Trinity were used as evidence for gene model prediction.

For the Funannotate workflow, the sequence names in the genome FASTA file must be shorter than 16 characters. The `Scripts/Data_Handling/reformat.fasta.header.py` script will perform this operation:

```
# On MSI interactive compute session
module load python3/3.6.3_anaconda5.0.1
python reformat.fasta.header.py
```

Next, the genome was repeat-masked with RepeatModeler version 1.0.11 and RepeatMasker version 4.0.5. This is implemented in the `Scripts/Jobs/repeatmasker.sh`, reproduced below:

```
#!/bin/bash
#PBS -l mem=32GB,nodes=1:ppn=24,walltime=200:00:00
#PBS -A jkimball
#PBS -m abe
#PBS -j oe
#PBS -M mmacchie@umn.edu
#PBS -q max
#PBS -W group_list=jkimball
#PBS -N repeatmasker

module load repeatmodeler/1.0.11 repeatmasker/4.0.5 perl/modules.centos7.5.26.1


echo -n "Ran: "
date

GENOME='/home/jkimball/shared/WR_Annotation/genome/zizania_palustris_13Nov2018_okGsv.fasta'

#cd /panfs/roc/scratch/konox006/JKimball/RNASeq/genome/

# 1. Run RepeatModeler
## Build repeat database
#BuildDatabase -name oryza_sativa -engine ncbi ${GENOME}

#echo -n "Done: "
#date


## Given a genomic database from above, build, refine, and classify consensus models of interspersed repeats
#RepeatModeler -engine ncbi -pa 24 -database oryza_sativa

## Combine repeat modeler results with repeatmasker (RM) database
### Convert RM.embl to fasta
#buildRMLibFromEMBL.pl /panfs/roc/msisoft/repeatmasker/4.0.5/Libraries/RepeatMaskerLib.embl > RepeatMaskerLib.fasta
### Combine RM fasta with RM fasta (consensi.fa.classified)
#cat RepeatMaskerLib.fasta consensi.fa.classified > combined_repeat_libs.fasta


cd /home/jkimball/shared/WR_Annotation/repeatmasker/
# 2. Run RepeatMasker
RepeatMasker -xsmall -x -gff -pa 6 -s -lib combined_repeat_libs.fasta -e ncbi ${GENOME}


echo -n "Done: "
date
```

Note that the `BuildDatabase`, `RepeatModeler`, `RepeatMaskerLib`, and `RepeatMasker` commands are separate parts of a single workflow to mask repeat regions in the genome. They are commented here because this script was run iteratively.

The RNAseq reads were also mapped to the genome with STAR version 2.7.1a. This is implemented in the `Scripts/Jobs/star.sh` file, reproduced below:

```
#!/bin/bash
#PBS -l mem=250GB,nodes=1:ppn=24,walltime=35:00:00
#PBS -A jkimball
#PBS -m abe
#PBS -j oe
#PBS -M mmacchie@umn.edu
#PBS -q ram256g
#PBS -W group_list=jkimball
```

```
#PBS -N star

module load star/2.7.1a



#GENOME='/panfs/roc/scratch/konox006/JKimball/RNASeq/genome/'
#R1='/panfs/roc/scratch/konox006/JKimball/RNASeq/Trimmed_reads/WR_8Tissues.R1.trimmed.fastq'
#R2='/panfs/roc/scratch/konox006/JKimball/RNASeq/Trimmed_reads/WR_8Tissues.R2.trimmed.fastq'
#INDEX='/panfs/roc/scratch/konox006/JKimball/RNASeq/genome/'
#OUT='/panfs/roc/scratch/konox006/JKimball/RNASeq/Aligned_reads/'

echo -n "Ran: "
date


# Index rice genome
#STAR --runMode genomeGenerate --runThreadN 16 --genomeFastaFiles ${GENOME} --genomeDir ${INDEX}

# did not work
#STAR --runMode alignReads --runThreadN 24 --genomeDir ${INDEX} --readFilesIn ${R1} ${R2} --outFileNamePrefix
 ↪   ${OUT}/rice_rnas --outSAMtype BAM Unsorted

# Align all reads - works
#STAR --runMode alignReads --runThreadN 24 --genomeDir ${INDEX} --readFilesIn ${R1} ${R2} --outFileNamePrefix
 ↪   ${OUT}/rice_rnas2 --limitBAMsortRAM 24846526061 --outSAMtype BAM SortedByCoordinate


GENOME='/home/jkimball/shared/WR_Annotation/genome/reformatted_genome/zizania_palustris_13Nov2018_okGsv.fasta.masked.headerreformatted'
R1='/home/jkimball/shared/WR_Annotation/Trimmed_reads/WR_8Tissues.R1.trimmed.fastq'
R2='/home/jkimball/shared/WR_Annotation/Trimmed_reads/WR_8Tissues.R2.trimmed.fastq'
OUT='/home/jkimball/shared/WR_Annotation/Aligned_reads/'
INDEX='/home/jkimball/shared/WR_Annotation/genome/reformatted_genome/'
STAR --runMode genomeGenerate --runThreadN 24 --genomeFastaFiles ${GENOME} --genomeDir ${INDEX}
STAR --runMode alignReads --runThreadN 24 --genomeDir ${INDEX} --readFilesIn ${R1} ${R2} --outFileNamePrefix
 ↪   ${OUT}/rice_rnas_chromreformatted --limitBAMsortRAM 24846526061 --outSAMtype BAM SortedByCoordinate


echo -n "Done: "
date
```

Similarly, the BAM file from STAR must have the sequence names adjusted to match the genome that was prepared for Funannoate. The `Scripts/Data_Handling/reformatted.sam.py` script will perform this:

```
# On MSI interactive compute session
module load python3/3.6.3_anaconda5.0.1
module load samtools/1.9
samtools view ../Aligned_reads/rice_rnas2Aligned.sortedByCoord.out.bam >
 ↪   ../Aligned_reads/rice_rnas2Aligned.sortedByCoord.out.sam
# change scaffold IDs in sam file
python reformatted.sam.py ../Aligned_reads/rice_rnas2Aligned.sortedByCoord.out.sam
 ↪   ../Aligned_reads/rice_rnas2Aligned.sortedByCoord.out.chromreformatted.sam
# convert sam file back into bam file
samtools view -S -b ../Aligned_reads/rice_rnas2Aligned.sortedByCoord.out.chromreformatted.sam >
 ↪   ../Aligned_reads/rice_rnas2Aligned.sortedByCoord.out.chromreformatted.bam
```

Now, we run Funannotate with the reformatted and masked genome FASTA, the mapping of RNAseq against the genome, and the Trinity transcriptome assembly. Funannotate requires access to both "GeneMark" and "Augustus," which can be obtained

by contacting the MSI help desk. The script that runs the Funannotate workflow is available at `Scripts/Jobs/funannotate.sh`, and is reproduced below:

```
#!/bin/bash
#PBS -l mem=62GB,nodes=1:ppn=24,walltime=250:00:00
#PBS -A jkimball
#PBS -m abe
#PBS -j oe
#PBS -M mmacchie@umn.edu
#PBS -q max
#PBS -W group_list=jkimball
#PBS -N funannotate_update

module load funannotate/1.5.1
#module unload gmap
#module load gmap/2012-10-31

export EVM_HOME=/panfs/roc/msisoft/evidencemodeler/1.1.1/
export AUGUSTUS_CONFIG_PATH=/scratch.global/mmacchie/config/
export AUGUSTUS_BIN_PATH=/scratch.global/mmacchie/bin/
export GENEMARK_PATH=/panfs/roc/msisoft/genemark/4.32/
export BAMTOOLS_PATH=/panfs/roc/msisoft/bamtools/2.5.1/bin/
export PATH=/panfs/roc/msisoft/augustus/3.2.3.CentOS7/scripts/:$PATH

echo -n "Ran: "
date

cd /home/jkimball/shared/WR_Annotation/Annotation/
GENOME='/home/jkimball/shared/WR_Annotation/genome/reformatted_genome/zizania_palustris_13Nov2018_okGsv.fasta.masked.headerreformatted'
TRINITY='/home/jkimball/shared/WR_Annotation/Transcriptome_Asm_trinity/Trinity.fasta'
RNABAM='/home/jkimball/shared/WR_Annotation/Aligned_reads/rice_rnas_chromreformattedAligned.sortedByCoord.out.bam'
FASTQ_R1='/home/jkimball/shared/WR_Annotation/Trimmed_reads/WR_8Tissues.R1.trimmed.fastq'
FASTQ_R2='/home/jkimball/shared/WR_Annotation/Trimmed_reads/WR_8Tissues.R2.trimmed.fastq'

#--- check species options
# funannotate species

#---Predict gene models ----#
# run on amdsmall; mem=62GB,nodes=1:ppn=32,walltime=96:00:00
#funannotate predict -i ${GENOME} --species "rice" --transcript_evidence ${TRINITY} --rna_bam ${RNABAM} -o wild_rice3
 ↪   --cpus 32
# "--species or --augustus_species?"

# run on max; mem=62GB,nodes=1:ppn=24,walltime=120:00:00
funannotate update -i wild_rice3 --cpus 24 --left ${FASTQ_R1} --right ${FASTQ_R2} --trinity ${TRINITY} --jaccard_clip

# run on max; mem=2GB,nodes=1:ppn=1,walltime=300:00:00
# funnannotate update threw an error at the PASA step - required 1 core for the sqlite database. So I am going to run
 ↪   the step that crashed with one core, and try to resume the job.
#cd /home/jkimball/shared/WR_Annotation/Annotation/wild_rice3/update_misc/pasa/
#/panfs/roc/msisoft/pasa/2.3.3/scripts/cDNA_annotation_comparer.dbi -G
 ↪   /panfs/roc/groups/1/jkimball/shared/WR_Annotation/Annotation/wild_rice3/update_misc/genome.fa --CPU 1 -M
 ↪   '/panfs/roc/groups/1/jkimball/shared/WR_Annotation/Annotation/rice'  >
 ↪   pasa_run.log.dir/rice.annotation_compare.32323.out

# run on 2020-01-24
#mkdir /home/jkimball/shared/WR_Annotation/Annotation/wild_rice3/update_misc/pasa2/
#cd /home/jkimball/shared/WR_Annotation/Annotation/wild_rice3/update_misc/pasa2/
#/panfs/roc/msisoft/pasa/2.3.3/scripts/cDNA_annotation_comparer.dbi -G
 ↪   /panfs/roc/groups/1/jkimball/shared/WR_Annotation/Annotation/wild_rice3/update_misc/genome.fa --CPU 24 -M
 ↪   '/panfs/roc/groups/1/jkimball/shared/WR_Annotation/Annotation/rice'  >
 ↪   pasa_run.log.dir/rice.annotation_compare.32323.out


echo -n "Done: "
```

```
date
```

Note that this script runs two major components of the Funannotate workflow, `predict` and `update`. Funannotate takes a **long time** and uses many resources, but the job can be resubmitted and Funannotate should pick up where it left off if it dies. This script has one of the routines commented-out because the workflow was run iteratively.

Estimates of annotation completeness were generated by searching the annotated proteins for single-copy orthologues with BUSCO 4.0. The `poales_odb10` database of orthologues was used for searching. The script is available in `Scripts/Jobs/busco_annotations.sh`:

```
#/bin/bash -l
#PBS -l nodes=1:ppn=12,mem=31gb,walltime=12:00:00
#PBS -m abe
#PBS -M mmacchie@umn.edu
#PBS -A jkimball
#PBS -W group_list=jkimball


module load python3/3.6.3_anaconda5.0.1
. /panfs/roc/msisoft/anaconda/anaconda3-5.0.1/etc/profile.d/conda.sh
conda activate /home/msistaff/konox006/.conda/envs/busco4_msi/

PASA="/home/jkimball/shared/WR_Annotation/Annotation/wild_rice3/update_misc/pasa/augustus.pasa.proteins.longestiso.fa"
OUT="/home/jkimball/shared/WR_Annotation/funannotate_BUSCO"

cd "${OUT}"

busco \
    -m proteins \
    -i "${PASA}" \
    -o busco \
    -l poales_odb10 \
    -c 12
```

The predicted peptide sequences from Funannotate were searched against the NCBI nonredundant peptide database for functional annotation. The BLAST searches were perfomed with the script in `Scripts/Jobs/funannotate_blast.sh`. The peptide sequences from Funannotate were split into smaller batches to take advantage of the highly parallel compute environment:

```
#!/bin/bash -l
#PBS -l nodes=1:ppn=24,pmem=1950mb,walltime=36:00:00
#PBS -A jkimball
#PBS -W group_list=jkimball
#PBS -M mmacchie@umn.edu
#PBS -m abe
#PBS -q mangi

module load ncbi_blast+/2.8.1

# Split the longest protein isoforms into multiple files for blast
# cd /home/jkimball/shared/WR_Annotation/Annotation/wild_rice3/update_misc/pasa/
# split -l 4000 -a 2 -d augustus.pasa.proteins.longestiso.fa WR_Prot_
```

```
cd /home/jkimball/shared/WR_Annotation

BLAST_DB="/panfs/roc/risdb_new/blast/current/nr"
#PROT="/home/jkimball/shared/WR_Annotation/Annotation/wild_rice3/update_misc/genome.proteins.fa"
# Find all protein FASTA files - uses transcripts IDs
#PROTS=($(find /home/jkimball/shared/WR_Annotation/Annotation/wild_rice3/update_misc/pasa/PASA_proteins_BLAST/
 ↪   -mindepth 1 -maxdepth 1 -name 'WR_Prot_*' | sort -V))

# Find all protein FASTA files - uses gene IDs
PROTS=($(find /home/jkimball/shared/WR_Annotation/Annotation/wild_rice3/update_misc/pasa/PASA_proteins_BLAST2/
 ↪   -mindepth 1 -maxdepth 1 -name 'WR_Prot_*' | sort -V))
CURR_PROT=${PROTS[${PBS_ARRAYID}]}

# This blast database is from Jan 2019
blastp \
    -query "${CURR_PROT}" \
    -db "${BLAST_DB}" \
    -out "WR_Protein_NR_BLASTP_${PBS_ARRAYID}.xml" \
    -evalue "1e-10" \
    -outfmt 5 \
    -max_target_seqs 3 \
    -qcov_hsp_perc 50 \
    -num_threads "${PBS_NUM_PPN}"
```

The BLAST search results were then used as input for the BLAST2GO software to assign gene ontology terms to each predicted gene model.

3.4 Expression Profiling

3.4.1 Preparing the Genome for Read Mapping

Annotations in GFF3 format from the Funannotate workflow were used for expression profiling and tissue specificity calculations. Gene annotations in GFF3 format were converted to GTF format with the `gffread` utility from the Cufflinks package:

```
# On MSI interactive compute session
module load cufflinks/2.2.1
gffread genome.gff3 -T -o genome.gtf
```

The putative splice sites were then extracted from the GTF using the `extract_splice_sites.py` script from the HISAT2 package:

```
# On MSI interactive compute session
module load python2/2.7.12_anaconda4.2
module load hisat2/2.1.0
extract_splice_sites.py genome.gtf > splice_sites.txt
```

The splice sites file and the genome assembly were used to build a splice-aware index for read mapping with HISAT2. This is available in the `Scripts/Jobs/Make_HISAT_Idx.sh` script, reproduced below:

```
#!/bin/bash
#PBS -l mem=251gb,nodes=1:ppn=8,walltime=24:00:00
#PBS -A jkimball
#PBS -W group_list=jkimball
#PBS -m abe
#PBS -M konox006@umn.edu
#PBS -q ram256g

# Load modules
module load hisat2/2.1.0
module load samtools/1.9

cd /home/jkimball/shared/WR_Annotation/HISAT2_Idx
hisat2-build -p 8 --ss ./splice_sites.txt genome.fa WR_genome
```

3.4.2 Gene Expression Quantification

Raw reads were trimmed with Trimmomatic using the same script as shown in the "RNAseq QC/QC" section. Reads were **not** depleted of ribosomal RNA for expression profiling, to more accurately estimate the proportion of ribosomal vs. protein coding mRNA present in each library. The steps of read mapping, BAM filtering, and expression counts generation were performed with the CHURP workflow, developed by the RIS group at MSI. Read mapping was carried out with HISAT2 version 2.1.0, alignment filtering was performed with SAMTools version 1.9, and expression counts were generated with featureCounts version 1.6.2. See the manual page for the workflow to see more details about the workflow.

The resulting counts matrix from the gene expression quantification was used to calculate the "tau" index of tissue specificity, as described by Yanai et al. (2005). Genes with tau values greater than 0.8 were considered to be "tissue-specific," and the tissue in which the gene had highest expression was identified. The script that performs the tau calculation and assignment of tissue specificity is available in `Scripts/Analysis/Tissue_Specificity.R`.

Note that for the R script mentioned above, three iterations of tissue specificity analyses were performed. Because the root, flower, and leaf sheath samples had high levels of ribosomal RNA content (see table above), they were iteratively excluded from tissue specificity estimates.

3.5 Orthologuous Gene Group Identification

Orthologous gene groups were identified between *Zizania palustris* and other grass species using Orthofinder version 2.3.11. The peptide sequences from the species given in the

"Publicly-Available Genome Data" section were downloaded from Ensembl Plants release 46. Orthofinder was run with BLAST output (rather than Diamond), and using the default option for calculating distance matrices and inferring trees ("dendroblast"). BLAST search results were computed ahead of time to take advantage of the highly parallelized compute environment on MSI.

The BLAST commands were generated with the `-op` option to Orthofinder:

```
# On MSI interactive compute session
module load python3/3.6.3_anaconda5.0.1
module load ncbi_blast+/2.8.1
/home/jkimball/shared/Software/OrthoFinder_Source_2.3.11/OrthoFinder_source/orthofinder.py \
    -S blast \
    -f /home/jkimball/shared/WR_Annotation/Orthofinder/Ensembl_Plants_Grasses \
    -op \
    > Orthofinder_BLAST_Cmds.txt
```

The `Orthofinder_BLAST_Cmds.txt` file was edited to change the `-num_threads` argument to `8`. Then, these searches were run with the `Scripts/Jobs/Orthofinder_BLAST.sh` script, reproduced below:

```
#!/bin/bash
#PBS -l pmem=1950mb,nodes=1:ppn=96,walltime=96:00:00
#PBS -m abe
#PBS -M konox006@umn.edu
#PBS -q mangi
#PBS -A jkimball
#PBS -W group_list=jkimball

module load parallel
module load ncbi_blast+/2.8.1

# Run the BLAST searches separately
IN_CMDS="/home/jkimball/konox006/Projects/Kimball_WR_Annotation/Scripts/Jobs/Orthofinder_BLAST_Cmds.txt"
# Run the BLAST searches in batches of 12
parallel -j 12 < "${IN_CMDS}"
```

Then, the Orthofinder algorithm was run with with the pre-computed BLAST searches. The script is available in `Scripts/Jobs/Orthofinder.sh`, reproduced below:

```
#!/bin/bash
#PBS -l mem=128gb,nodes=1:ppn=1,walltime=24:00:00
#PBS -m abe
#PBS -M konox006@umn.edu
#PBS -q ram256g
#PBS -A jkimball
#PBS -W group_list=jkimball

module load python3/3.6.3_anaconda5.0.1
module load ncbi_blast+/2.8.1

# Set paths
#ORTHOFINDER="/home/jkimball/shared/Software/OrthoFinder_2.3.11/OrthoFinder/orthofinder"
ORTHOFINDER="/home/jkimball/shared/Software/OrthoFinder_Source_2.3.11/OrthoFinder_source/orthofinder.py"
BLAST_DIR="/home/jkimball/shared/WR_Annotation/Orthofinder/Orthogroups/Results_Mar04/WorkingDirectory"

python "${ORTHOFINDER}" \
    -a "${PBS_NUM_PPN}" \
    -b "${BLAST_DIR}"
```

3.6 Preparation of NCBI Submission Files

The genome FASTA and the GFF3 annotations file were used to produce annotation files for submission to NCBI. The format for the NCBI Sequin feature table is described here. Because of its differences from the GFF3 file that we have from Funannotate, we generated lookup tables for the genes, transcripts, exons, and coding sequences that hold the relevant information for the feature table. Then, we parsed the GFF3 and emitted data in the feature table format.

First, the scaffolds in the genome were renamed to the chromosome naming scheme, using `Scripts/Data_Handling/scaffold_renamer.py`:

```
# On MSI interactive compute session
module load python3/3.6.3_anaconda5.0.1
python scaffold_renamer.py
```

Then, the GFF3 file was sorted by position so that the genes appear in linear order on the chromosomes, using `Scripts/Data_Handling/reorder.gff.file.py`:

```
# On MSI interactive compute session
module load python3/3.6.3_anaconda5.0.1
python reorder.gff.file.py
```

The resulting re-ordered GFF was translated to the same chromosomal naming scheme as the genome assembly, using `Scripts/Data_Handling/update.annotations.py`:

```
# On MSI interactive compute session
module load python3/3.6.3_anaconda5.0.1
python update.annotations.py
```

Next, the re-ordered and updated GFF3 was screened for partial gene features, by marking those without annotated untranslated regions (UTRs) with `Scripts/Data_Handling/screen.gene.UTRs.py`:

```
# On MSI interactive compute session
module load python3/3.6.3_anaconda5.0.1
python screen.gene.UTRs.py
```

The GFF was also used to produce lookup tables for exon boundaries, coding sequence junction points, and coding sequence metadata. The script that produces a lookup table for transcripts is `Scripts/Data_Handling/Make_mRNA_Exon_Table.py`, and the script that

generates the CDS lookup table is `Scripts/Data_Handling/Classify_CDS_For_NCBI_Submission.py`. The CDS classification script requires the Biopython library to be installed:

```
# On MSI interactive session
module load python3/3.6.3_anaconda5.0.1
python Make_mRNA_Exon_Table.py
 ↪  rice.gene_structures_post_PASA_updates.21917.gff3 | gzip -c >
 ↪  Zp_mRNA_Exon_Positions.txt.gz
python Classify_CDS_For_NCBI_Submission.py
 ↪  rice.gene_structures_post_PASA_updates.21917.gff3 genome.fasta | gzip -c >
 ↪  Zp_CDS_Classifications_Phase.txt.gz
```

These files were then used as input for the `Scripts/Data_Handling/create.feature.table.py` script:

```
# On MSI interactive compute session
module load python3/3.6.3_anaconda5.0.1
python create.feature.table.py
```

The resulting feature table file is then used as input for the `tbl2asn` program from NCBI that converts a Sequin feature table to the ASN.1 format for annotation submission. This is handled by the `Scripts/Jobs/tbl2asn.sh` script, reproduced below:

```
#!/bin/bash
#PBS -l mem=60GB,nodes=1:ppn=4,walltime=96:00:00
#PBS -A jkimball
#PBS -m abe
#PBS -j oe
#PBS -M mmacchie@umn.edu
#PBS -q amd2tb
#PBS -W group_list=jkimball
#PBS -N tbl2asn

cd /home/jkimball/shared/WR_Annotation/NCBI_submission

#./linux64.tbl2asn -i
 ↪  ../genome/reformatted_genome/zizania_palustris_13Nov2018_okGsv.fasta.masked.headerreformatted.fa -f
 ↪  feature.table.test.tbl -t template.sbt -o zpal.asn -V vbt -s T -m B -l paired-ends -a r10k -W T

./linux64.tbl2asn -i ../genome/zizania_palustris_13Nov2018_okGsv_renamedNCBI2.fasta -f feature.table.NCBI.tbl -t
 ↪  ncbi_template.sbt -o zpal.asn -V vbt -s T -m B -l paired-ends -a r10k -W T -j "[organism=Zizania palustris]"
```