

## Exercise 1. Using MAKER for Genome Annotation

The example here is from a workshop by Mark Yandell Lab (<http://www.yandell-lab.org/>)

Further readings:

1. Yandel Lab Workshop. [http://weatherby.genetics.utah.edu/MAKER/wiki/index.php/MAKER\\_Tutorial\\_for\\_WGS\\_Assembly\\_and\\_Annotation\\_Winter\\_School\\_2018](http://weatherby.genetics.utah.edu/MAKER/wiki/index.php/MAKER_Tutorial_for_WGS_Assembly_and_Annotation_Winter_School_2018) .
2. MAKER protocol from Yandell Lab. It is good reference. <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4286374/>
3. Tutorial for training Augustus <https://vcru.wisc.edu/simonlab/bioinformatics/programs/augustus/docs/tutorial2015/training.html>
4. Maker control file explained: [http://weatherby.genetics.utah.edu/MAKER/wiki/index.php/The\\_MAKER\\_control\\_files\\_explained](http://weatherby.genetics.utah.edu/MAKER/wiki/index.php/The_MAKER_control_files_explained)

### Part 1. Prepare working directory.

1. Copy the data file from /shared\_data/annotation2018/ into /workdir/\$USER, and decompress the file. You will also copy the maker software directory to /workdir/\$USER. The maker software directory including a large sequence repeats database. It would be good to put it under /workdir which is on local hard drive.

```
mkdir /workdir/$USER
mkdir /workdir/$USER/tmp
cd /workdir/$USER
cp /shared_data/annotation2019/maker_tutorial.tgz ./
cp -rH /programs/maker/ ./
cp -rH /programs/RepeatMasker ./
tar -zxf maker_tutorial.tgz
cd maker_tutorial
ls -l
```

### Part 2. Maker round 1 - Map known genes to the genome

Round 1 includes two steps:

- Repeat masking;
  - Align known transcriptome/protein sequences to the genome;
1. **[Optional] Build a custom repeat database.** This step is optional for this exercise, as it is a very small genome, it is ok without repeat masking. When you work on a real project, you can either download a database from RepBase (<https://www.girinst.org/replib/>, license required), or you can build a custom repeat database with your genome sequence. RepeatModeler is a software for building custom databases. The commands for building a repeat database are provided here.

```
cd example_02_abinitio
export PATH=/programs/RepeatModeler-2.0:$PATH
BuildDatabase -name pyu pyu_contig.fasta
RepeatModeler -pa 4 -database pyu -LTRstruct >& repeatmodeler.log
```

At the end of run, you would find a file "pyu-families.fa". This is the file you can supply to "rmlib=" in the control file.

## 2. Set environment to run Maker and create MAKER control files.

Every steps in Maker are specified by the Maker control files. The command "maker -CTL" will create three control files: maker\_bopts.ctl, maker\_exe.ctl, maker\_opts.ctl.

```
export
PATH=/workdir/$USER/maker/bin:/workdir/$USER/RepeatMasker:/programs/snap:$PATH
export ZOE=/programs/snap/Zoe
export LD_LIBRARY_PATH=/programs/boost_1_62_0/lib
cd /workdir/$USER/maker_tutorial/example_02_abinitio
maker -CTL
```

## 3. Modify the control file **maker\_opts.ctl**.

Open the **maker\_opts.ctl** file in a text editor (e.g. Notepad++ on Windows, BBEdit on Mac, or vi on Linux). Modify the following values. Put the modified file in the same directory "example\_02\_abinitio".

```
genome=pyu_contig.fasta

est=pyu_est.fasta
protein=sp_protein.fasta

model_org=simple
rmlib= #fasta file of your repeat sequence from RepeatModeler. Leave blank to
skip.
softmask=1

est2genome=1
protein2genome=1

TMP=/workdir/$USER/tmp #important for big genome, as the default /tmp is too
small
```

The modified **maker\_opts.ctl** file instructs MAKER to do two things.

a) Run RepeatMasker.

- The line "model\_org=simple" tells RepeatMasker to mask the low complexity sequence (e.g. "AAAAAAAAAAAAA").
- The line "rmlib=" sets "rmlib" to null, which tells RepeatMasker not to mask repeat sequences like transposon elements. If you have a repeat fasta file (e.g. output from RepeatModeler) that you need to mask, put the fasta file name next to "rmlib="
- The line "softmask=1" tells RepeatMasker to do soft-masking which converts repeats to lower case, instead of hard-masking which converts repeats to "N". "Soft-masking" is important so that short repeat sequences within genes can still be annotated as part of gene.
- If you run RepeatMasker separately, as described in <https://gist.github.com/darencard/bb1001ac1532dd4225b030cf0cd61ce2>, you should leave rmlib to null, but set rm\_gff to a repeat gff file.

b) Align the transcript sequences from the pyu\_est.fasta file and protein sequences from the sp\_protein.fasta file to the genome and infer evidence supported gene model.

- The lines "est2genome=1" and "protein2genome=1" tell MAKER to align the transcript sequences from the pyu\_est.fasta file and protein sequences from the sp\_protein.fasta file to the genome. These two files are used to define evidence supported gene model.
- The lines "est=pyu\_est.fasta" and "protein=sp\_protein.fasta" specify the fasta file names of the EST and protein sequences. In general, the EST sequence file contains the assembled transcriptome from RNA-seq data. The protein sequence file include proteins from closely related species or swiss-prot. If you have multiple protein or EST files, separate file names with ",".

4. **[Do it at home]** Execute repeat masking and alignments. This step takes an hour. Run it in "screen". In the command: "mpiexec -n 2 " means that you will parallelize Maker using MPI, and use two threads at a time. When you work on a real project, it will take much longer, and you should increase this "-n" setting to the number of cores.

Set Maker environment if it is new session:

```
export
PATH=/workdir/$USER/maker/bin:/workdir/$USER/RepeatMasker:/programs/snap:$PATH
export ZOE=/programs/snap/Zoe
export LD_LIBRARY_PATH=/programs/boost_1_62_0/lib
```

Execute the commands:

```
cd /workdir/qisun/maker_tutorial/example_02_abinitio
/usr/local/mpich/bin/mpiexec -n 2 maker -base pyu_rnd1 >& log1 &
```

After it is done, you can check the log1 file. You should see a sentence: Maker is now finished!!!

## Part 3. Maker round 2 - Gene prediction using SNAP

### 1. Train a SNAP gene model.

SNAP is software to do *ab initio* gene prediction from a genome. In order to do gene prediction with SNAP, you will first train a SNAP model with alignment results produced in the previous step.

If you skipped the step "4. **[Do it at home]** Execute Maker round 1", you can copy the result files from this directory: /shared\_data/annotation2019/

```
cd /workdir/qisun/maker_tutorial/example_02_abinitio
cp /shared_data/annotation2019/pyu_rnd1.maker.output.tgz ./
tar xvfz pyu_rnd1.maker.output.tgz
```

Set Maker environment if it is new session:

```
export
PATH=/workdir/$USER/maker/bin:/workdir/$USER/RepeatMasker:/programs/snap:$PATH
export ZOE=/programs/snap/Zoe
export LD_LIBRARY_PATH=/programs/boost_1_62_0/lib
```

The following commands will convert the MAKER round 1 results to input files for building a SNAP mode.

```
mkdir snap1
cd snap1
gff3_merge -d ../pyu_rnd1.maker.output/pyu_rnd1_master_datastore_index.log
maker2zff -l 50 -x 0.5 pyu_rnd1.all.gff
```

The “-l 50 -x 0.5” parameter in maker2zff commands specify that only gene models with AED score>0.5 and protein length>50 are used for building models. You will find two new files: genome.ann and genome.dna.

Now you will run the following commands to train SNAP. The basic steps for training SNAP are first to filter the input gene models, then capture genomic sequence immediately surrounding each model locus, and finally uses those captured segments to produce the HMM. You can explore the internal SNAP documentation for more details if you wish.

```
fathom -categorize 1000 genome.ann genome.dna
fathom -export 1000 -plus uni.ann uni.dna
forge export.ann export.dna
hmm-assembler.pl pyu . > ../pyu1.hmm
mv pyu_rnd1.all.gff ../
cd ..
```

After this, you will find two new files in the directory example\_02\_abinitio:

pyu\_rnd1.all.gff: A gff file from round 1, which is evidence based genes.

pyu1.hmm: A hidden markov model trained from evidence based genes.

## 2. Use SNAP to predict genes.

Modify directly on the maker\_opts.ctl file that you have modified previously.

Before doing that, you might want to save a backup copy of maker\_opts.ctl for round 1.

```
cp maker_opts.ctl maker_opts.ctl_backup_rnd1
```

Now modify the following values in the file: maker\_opts.ctl

```
maker_gff= pyu_rnd1.all.gff
est_pass=1 # use est alignment from round 1
protein_pass=1 #use protein alignment from round 1
rm_pass=1 # use repeats in the gff file
snaphmm=pyu1.hmm
est= # remove est file, do not run EST blast again
protein= # remove protein file, do not run blast again
model_org= #remove repeat mask model, so not running RM again
rmlib= # not running repeat masking again
repeat_protein= #not running repeat masking again
est2genome=0 # do not do EST evidence based gene model
protein2genome=0 # do not do protein based gene model.
pred_stats=1 #report AED stats
```

Run maker with the new control file. This step takes a few minutes. (A real project could take hours to finish). You will use the option “-base pyu\_rnd2” so that the results will be written into a new directory “pyu\_rnd2”.

```
/usr/local/mpich/bin/mpirun -n 2 maker -base pyu_rnd2 >& log2 &
```

Again, make sure the log2 file ends with "Maker is now finished!!!!".

## Part 4. Maker round 3 - Retrain SNAP model and do another round of SNAP gene prediction

You might need to run two or three rounds of SNAP. So you will repeat Part 2 again. Make sure you will replace snap1 to snap2, so that you would not over-write previous round.

1. First train a new SNAP model.

```
mkdir snap2
cd snap2
gff3_merge -d ../pyu_rnd2.maker.output/pyu_rnd2_master_datastore_index.log
maker2zff -l 50 -x 0.5 pyu_rnd2.all.gff
fathom -categorize 1000 genome.ann genome.dna
fathom -export 1000 -plus uni.ann uni.dna
forge export.ann export.dna
hmm-assembler.pl pyu . > ../pyu2.hmm
mv pyu_rnd2.all.gff ..
cd ..
```

2. Use SNAP to predict genes.

Modify directly on the maker\_opts.ctl file that you have modified previously.

Before doing that, you might want to save a backup copy of maker\_opts.ctl for round 2.

```
cp maker_opts.ctl maker_opts.ctl_backup_rnd2
```

Now modify the following values in the file: maker\_opts.ctl

```
maker_gff=pyu_rnd2.all.gff
snaphmm=pyu2.hmm
```

Run Maker:

```
/usr/local/mpich/bin/mpirun -n 2 maker -base pyu_rnd3 >& log3 &
```

Use the following command to create the final merged gff file. The “-n” option would produce a gff file without genome sequences:

```
gff3_merge -s -n -d
pyu_rnd3.maker.output/pyu_rnd3_master_datastore_index.log>pyu_rnd3.noseq.gff
```

After this, you will get a new gff3 file: pyu\_rnd3.noseq.gff.

3. Generate AED plots.

```
/programs/maker/AED_cdf_generator.pl -b 0.025 pyu_rnd2.all.gff > AED_rnd2
/programs/maker/AED_cdf_generator.pl -b 0.025 pyu_rnd3.noseq.gff > AED_rnd3
```

You can use Excel or R to plot the second column of the AED\_rnd2 and AED\_rnd3 files, and use the first column as the X-axis value. The X-axis label is "AED", and Y-axis label is "Cumulative Fraction of Annotations "

## Part 5. Visualize the gff file in IGV

You can load the gff file into IGV or JBrowse, together with RNA-seq read alignment bam files. For instructions of running IGV and loading the annotation gff file, you can read under "part 4" of this document:

<http://biohpc.cornell.edu/doc/RNA-Seq-2019-exercise1.pdf>

## Appendix: Training Augustus model

Modified from <https://vcru.wisc.edu/simonlab/bioinformatics/programs/augustus/docs/tutorial2015/training.html>.

Starting files:

- GFF file produced from each round of Maker run, E.g. pyu\_rnd1.all.gff
- Genome fasta file, e.g. pyu\_contig.fasta

```
##set augustus environment
cp -r /programs/Augustus-3.3.3/config/ /workdir/$USER/augustus_config
export LD_LIBRARY_PATH=/programs/boost_1_62_0/lib
export AUGUSTUS_CONFIG_PATH=/workdir/$USER/augustus_config/
export LD_LIBRARY_PATH=/programs/boost_1_62_0/lib
export LC_ALL=en_US.utf-8
export LANG=en_US.utf-8
export PATH=/programs/augustus/bin:/programs/augustus/scripts:$PATH

## filter gff file, only keep maker annotation in the filtered gff file
awk '{if ($2=="maker") print }' pyu_rnd1.all.gff > maker_rnd1.gff

##convert the maker gff and fasta file into a Genbank formatted file named pyu.gb
##we keep 1000 bp up- and down-stream of each gene for training the models
gff2gbSmallDNA.pl maker_rnd1.gff pyu_contig.fasta 1000 pyu.gb

##randomly split the pyu.gb into a training set and a smaller test set
randomSplit.pl pyu.gb 100

## check number of genes in training and test set
grep -c LOCUS pyu.gb*

## train model
## first create a new Augustus species named
new_species.pl --species=pyu

## initial training
etrainig --species=pyu pyu.gb.train

## the initial model should be in the directory
```

```
ls -ort $AUGUSTUS_CONFIG_PATH/species/pyu

# use the first model to predict the genes in the test set
augustus --species=pyu pyu.gb.test | tee firsttest.out

#evaluate the results
grep -A 22 Evaluation firsttest.out

# optimize the model. this step could take a very long time
optimize_augustus.pl --species=pyu pyu.gb.train

#train again
etraining --species=pyu pyu.gb.train
```

After these steps, the species model is in the directory  
/workdir/\$USER/augustus\_config/species/pyu.

When running next round of Maker, you should specify the species name ("pyu" in this case) in the line: augustus\_species=pyu.

Also, make sure to set Augustus environments when running next round of Maker.

```
export LD_LIBRARY_PATH=/programs/boost_1_62_0/lib
export AUGUSTUS_CONFIG_PATH=/workdir/$USER/augustus_config/
export LD_LIBRARY_PATH=/programs/boost_1_62_0/lib
export LC_ALL=en_US.utf-8
export LANG=en_US.utf-8
export PATH=/programs/augustus/bin:/programs/augustus/scripts:$PATH
```