Exercise 1. Using MAKER for Genome Annotation

If you are following this guide for your own research project, please make the following modifications:

1. In this exercise, SNAP was used for gene prediction. When you are working on your own genome, we recommend that you use Augustus. The instructions for using Augustus is in appendix.
2. In the exercise, you will be using 2 CPU cores. When you are working on your own genome, you should use all CPU cores on your machine. When you run the command: "/usr/local/mpich/bin/mpiexec -n 2", replace 2 with number of cores available on your machine.
3. The steps for Repeatmodeler and Repeatmasker are optional in the exercise, but required when you work on your own genome.

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 .
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

Part 1. Prepare working directory.

1. Copy the data file from /shared_data/annotation2018/ into /workdir/$USER, and de-compress 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 -1
```

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

Run everything in "screen".

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/repbase/, 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.by.

```
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. “AAAAAAAAAAAA”.
- 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
alt_splice=0 # 0: keep one isoform per gene; 1: identify splicing variants of
the same gene
keep_preds=1 # keep genes even without evidence support, set to 0 if no
```

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/mpiexec -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/mpiexec -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 -n -d
pyu_rnd3.maker.output/pyu_rnd3_master_datastore_index.log>pyu_rnd3.noseq.gff

fasta_merge -d pyu_rnd3.maker.output/pyu_rnd3_master_datastore_index.log
```

After this, you will get a new gff3 file: pyu_rnd3.noseq.gff, and protein and transcript fasta files.

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

Run Part 1 & 2.

In the same screen session, set up 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
```

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

```
mkdir augustus1
cd augustus1
gff3_merge -d ../pyu_rnd1.maker.output/pyu_rnd1_master_datastore_index.log
```

After this step, you will see a new gff file pyu_rnd1.all.gff from round 1.

```


## 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 formated file named pyu.gb
##We keep 2000 bp up- and down-stream of each gene for training the models
gff2gbSmallDNA.pl maker_rnd1.gff pyu_contig.fasta 2000 pyu.gb

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

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

## initial training
etraining --species=pyu pyu.gb

## the initial model should be in the directory
ls -ort $AUGUSTUS_CONFIG_PATH/species/pyu

##create a smaller test set for evaluation before and after optimization. Name
the evaluation set pyu.gb.evaluation.
randomSplit.pl pyu.gb 200
mv pyu.gb.test pyu.gb.evaluation

# use the first model to predict the genes in the test set, and check the
results
augustus --species=pyu pyu.gb.evaluation >& first_evaluate.out
grep -A 22 Evaluation first_evaluate.out

# optimize the model. this step is very time consuming. It could take days. To
speed things up, you can create a smaller test set
# the following step will create a test and training sets. the test set has 1000
genes. This test set will be splitted into 24 kfolds for optimization (the kfold
can be set up to 48, with processed with one cpu core per kfold. Kfold must be
same number as as cpus). The training, prediction and evaluation will be
performed on each bucket in parallel (training on hh.gb.train+each bucket, then
comparing each bucket with the union of the rest). By default, 5 rounds of
optimization. As optimization for large genome could take days, I changed it to
3 here.

randomSplit.pl pyu.gb 1000
optimize_augustus.pl --species=hh --kfold=24 --cpus=24 --rounds=3 --
onlytrain=pyu.gb.train pyu.gb.test >& log &

#train again after optimization
etraining --species=pyu pyu.gb

# use the optionized model to evaluate again, and check the results
```

```
augustus --species=pyu pyu.gb.evaluation >& second_evaluate.out
grep -A 22 Evaluation second_evaluate.out
```

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

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
augustus_species=pyu # augustus species model you just built
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
alt_splice=0 # 0: keep one isoform per gene; 1: identify splicing variants of
the same gene
keep_preds=1 # keep genes even without evidence support, set to 0 if no
```

Run maker with the new augustus model

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

Create gff and fasta output files:

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

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

fasta_merge -d pyu_rnd3.maker.output/pyu_rnd3_master_datastore_index.log
```

After this, you will get a new gff3 file: pyu_rnd3.noseq.gff, and protein and transcript fasta files.

To make the gene names shorter, use the following commands:

```
maker_map_ids --prefix pyu_ --justify 8 --iterate 1 pyu_rnd3.all.gff > id_map
map_gff_ids id_map pyu_rnd3.all.gff
map_fasta_ids id_map pyu_rnd3.all.maker.proteins.fasta
map_fasta_ids id_map pyu_rnd3.all.maker.transcripts.fasta
```