【workflow】maker-基因组注释 v1.1 3/2-23

Last Edited	@March 8, 2023 2:24 PM
Entry	Bioinfo-learning Extracurricular HEMU-DB
Type	workflow
Materials	

https://github.com/EdwardZhu02/maker-annot-workflow

主要参考资料

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/c24cec06-93c9-43c4-8c98-c5e824efe279/using-maker-for-genome-annotation-cornell.pdf

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/fb3ceca2-4c35-4222-bce6-86d3ab198e65/Campbell-2014-Genome-annotation-and-curation-usin.pdf

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/e434674d-f799-4b7e-8ea2-33d801afd36b/Running-MAKER-with-little-to-no-evidence.md

目录

0 Genome Pre-processing

将NCBI的by-chromosome .fna files merge为一个fasta file。

另外，为防止基因组中fasta headers过长影响下游流程，使用一个**in-house perl script进行fasta header的转换**。转换后的headers以 `Seq_1` `Seq_2` 等顺序命名。

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/56c1f152-0be0-499d-9b1c-40d6dd648c0a/translate_fasta_headers.pl

当然，也会生成一个translation tab，格式是 `[原header] \t [转换后的header] \n`

```
#!/bin/bash

# Obtain folder name (Accession Number)
project_path=$(cd `dirname $0`; pwd)
project_name="${project_path##*/}"

# Concatenate all '.fna' files into a single '.fa' file
for i in $(ls *.fna);
do
  cat ${i}
  # Output a blank line after each '.fna file
  echo
done > ./${project_name}.original.fa
wait

# [Optional] Delete original '.fna' files and sequence reports ('.jsonl')
# rm ./*.fna
# rm ./*.jsonl

perl translate_fasta_headers.pl \
  --out=./${project_name}.fa \
  ./${project_name}.original.fa # Input for long-seq name FA
wait

mkdir ./${project_name}_FA_sequence_id_translation
mv ./${project_name}.fa.translation.tab ./${project_name}_FA_sequence_id_translation/
rm ./${project_name}.original.fa
wait
```

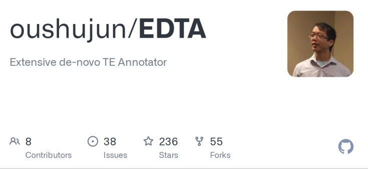
1 Repeat Annotation & Masking - EDTA

maker workflow中可使用的Repeat identification方法很多，这里使用Oushujun et. al. 2018年发表的**EDTA (Extensive *de-novo* TE Annotator) Pipeline**.

GitHub - oushujun/EDTA: Extensive de-novo TE Annotator

This package is developed for automated whole-genome de-novo TE annotation and benchmarking the annotation performance of TE libraries. The EDTA package was designed to filter out false discoveries in raw TE candidates and generate a high-quality non-redundant TE library for whole-genome TE

https://github.com/oushujun/EDTA


oushujun/**EDTA**

Extensive de-novo TE Annotator

8 Contributors 38 Issues 236 Stars 55 Forks

在笔者所写的workflow in-house script中，默认一个物种的基因组存放在**以其Accession Number（GCA/GCF开头）命名的文件夹**中。文件夹内有不同染色体的fna文件，或已经合并过的fna文件。

```
#!/bin/bash

# Obtain folder name (Accession Number)
project_path=$(cd `dirname $0`; pwd)
project_name="${project_path##*/}"

# Do not pick up previous results (--overwrite 1)
# Do not use RepeatModeler to identify remaining TEs (--sensitive 0)
perl /data5/Andropogoneae_LTR_Project/AUX_SOFTWARES/EDTA/EDTA.pl \
  --genome ./${project_name}.fa \
  --step all \
  --overwrite 1 \
  --sensitive 0 \
  --anno 1 \
  --threads 8 \
```

如果出现TIR注释失败的问题（no TIRs found in the genome），可能是由于内存不足导致的——TIR-learner需要耗费非常大的内存，**About 10-20GB/thread。**

此时可以选择将TIR-learner的中间文件用rsync删除（不能用rm，太多小文件）后，再不覆盖地运行一遍EDTA。

```
#!/bin/bash

project_path=$(cd `dirname $0`; pwd)
project_name="${project_path##*/}"

mkdir ./${project_name}.fa.mod.EDTA.raw/rsync_empty
rsync --delete-before -d ./${project_name}.fa.mod.EDTA.raw/rsync_empty/ ./${project_name}.fa.mod.EDTA.raw/TIR/
rm -r ./${project_name}.fa.mod.EDTA.raw/TIR
rm -r ./${project_name}.fa.mod.EDTA.raw/rsync_empty

perl /data5/Andropogoneae_LTR_Project/AUX_SOFTWARES/EDTA/EDTA.pl \
  --genome ./${project_name}.fa \
  --step all \
  --overwrite 0 \
  --sensitive 0 \
  --anno 1 \
  --threads 8 \
```

整个workflow在8 threads的quota下大约需要运行一周 (~2Gb Genome)，如果一切顺利，整个目录下的结果文件分布大概长这样：

```
$ ls
#EDTA_opted0507.sh                          GCA_018135685.1.fa.mod.EDTA.raw
#GCA_018135685.1_ASM1813568v1_genomic.fna  GCA_018135685.1.fa.mod.EDTA.TEanno.gff3
#GCA_018135685.1.fa                         GCA_018135685.1.fa.mod.EDTA.TEanno.sum
#GCA_018135685.1.fa.mod                     GCA_018135685.1.fa.mod.EDTA.TElib.fa
#GCA_018135685.1.fa.mod.EDTA.anno           GCA_018135685.1.fa.mod.MAKER.masked
#GCA_018135685.1.fa.mod.EDTA.combine        GCA_018135685.1_FA_sequence_id_translation
#GCA_018135685.1.fa.mod.EDTA.final          nohup.out
#GCA_018135685.1.fa.mod.EDTA.intact.gff3   sequence_report.jsonl
```

2 Transcriptome Assembly & RNA-seq Evidence Modeling - Trinity

Haas, B. J. et. al. (2013). De novo transcript sequence reconstruction from RNA-seq using the Trinity platform for reference generation and analysis. *Nature Protocols.*

6.4-6.5 Trinity 大流程实战: 以Dev文为例

2.1 fastq质量控制 - FASTP

生成 `json` 格式的质量报告文件，便于统计和数据可视化。为篇幅简洁考虑，此处只显示双端数据的处理：

```
#!/bin/bash
fastp \
  -i ./SRR8449829_1.fastq \
  -I ./SRR8449829_2.fastq \
  -o ./SRR8449829_1_Filtered.fastq \
  -O ./SRR8449829_2_Filtered.fastq \
  -j ./SRR8449829_Report.json -h ./SRR8449829_Report.html
```

或者可以直接用笔者先前写的**RNA-seq Automaton (Version 4.0.1)**中FASTP模块进行。

> https://s3-us-west-2.amazonaws.com/secure.notion-static.com/b194f93c-0903-42e9-bf82-d0e07f014723/Automaton_Ver4.0.1.py

使用方法如下，这个Automaton可以自动识别单端/双端测序数据，并根据fastq文件是否压缩 **(.fastq.gz / .fastq)** 采取不同的处理措施，最终生成一步到位的命令集。

```
python3 Automaton_Ver4.0.1.py [path-to-folder-containing-fastps] fastp [parallel-task-number]
```

2.2 Read Merge 和 Trinity 部署

在此Workflow中，笔者使用双端测序的fastq reads。单端测序fastq的部署流程可参照上面提及的文章。

```
#!/bin/bash

cat ./*_1_Filtered.fastq > left_all.fq
cat ./*_2_Filtered.fastq > right_all.fq

Trinity --seqType fq --no_version_check \
--left left_all.fq \
--right right_all.fq \
--CPU 30 \
--max_memory 100G \
--min_contig_length 300 \
--output ./trinity_result
```

结果会输出至 `./trinity_result` 文件夹。

如果没有已安装trinity的conda环境，则可手动配置，在**运行时指明 `$TRINITY_HOME/Trinity.pl` 的位置**即可。

2.3 Trinity组装结果去冗余 - cd-hit-est

Trinity的组装结果会出现在 ./trinity_result/**Trinity.fasta**

```
#!/bin/bash
cd-hit-est \
  -i ./trinity_result/Trinity.fasta \
  -o ./cdhit_rmredun_Trinity.fasta \
  -c 0.98a -d 0 \
  -T 8 -M 40960
```

核苷酸序列使用cd-hit-est命令，氨基酸序列使用cd-hit命令。

- `-c 0.9` 相似性（clustering threshold）,表示相似性大于等于90%的为一类;
- `-d 0` 使用 fasta 标题中第一个空格前的字段作为序列名字;
- `-M 40960` 使用40GB RAM; `-T 20` 使用20线程;

这样既可获得去冗余后的fasta格式转录本。

3 Initial MAKER Run - Evidence Mapping

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/ce22d1a2-436a-4970-bdae-9ee429bca115/maker_env.yaml

生成MAKER的控制文件：在项目根目录运行 `maker -CTL`

Three control files are created: **maker_bopts.ctl, maker_exe.ctl, maker_opts.ctl.**
In most cases, you do not need to change content of the files maker_bopts.ctl and maker_exe.ctl. → **主要的运行配置需要在 `maker_opts.ctl` 中设置**。

3.1 Evidence Preparation & Configuration

最初的MAKER run需要几个evidence，即

1. Trinity 使用物种不同组织 RNA-seq 组装完成的转录本(transcriptome)： `cdhit_rmredun_Trinity.fasta`
2. 通用蛋白数据库 (本workflow使用uniprot): `uniprot_sprot.fasta`

> 本地地址在**/data21/zhuyuzhi/databases**，也可以从https://www.uniprot.org/downloads下载到

3. 近源物种中已知的蛋白数据 (本workflow中，由于注释的都是高粱族基因组，故使用高粱族中注释较好的**玉米 *Zea mays* B73v4** 蛋白数据输入)

> Index of /Zm-B73-REFERENCE-GRAMENE-4.0 (maizegdb.org)

4. EDTA注释出的Transposable Elements，使用**TE家族代表序列（TElib.fa）**和**全基因组TE注释（TEanno.gff3）**

综上所述，需要**配置和更改如下几行**。以下存成一个新的 `.ctl` 文件，用于第一步的maker run：

```
#-----Genome (these are always required)
genome=/data21/zhuyuzhi/maker_test_Microstegium/microstegium_genome.fa

#-----EST Evidence (for best results provide a file for at least one)
est=/data21/zhuyuzhi/maker_test_Microstegium/cdhit_rmredun_Trinity.fasta

#-----Protein Homology Evidence (for best results provide a file for at least one)
protein=/data21/zhuyuzhi/databases/ZmB73v4_proteins/Zm-B73-REFERENCE-GRAMENE-4.0_Zm00001d.2.protein.fa,/data21/zhuyuzhi/databases/
uniprot_sprot.fasta

#-----Repeat Masking (leave values blank to skip repeat masking)
model_org=simple
rmlib=/data21/zhuyuzhi/maker_test_Microstegium/microstegium_TElib.fa
rm_gff=/data21/zhuyuzhi/maker_test_Microstegium/microstegium_TEanno.gff3

#-----Gene Prediction
run_evm=1 #run EvidenceModeler, 1 = yes, 0 = no
est2genome=1 #infer gene predictions directly from ESTs, 1 = yes, 0 = no
protein2genome=1 #infer predictions from protein homology, 1 = yes, 0 = no
```

额外放一个全文件示例在这里。以备不时之需。

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/702b7852-ee3e-4522-96ed-3d7c7c9ca208/maker_opts-firstrun-microstegium.ctl

3.2 Task Deployment - 并行与非并行

首先是非并行时的maker run，也是tutorial中所建议使用的。不过非常慢。在这里放进来仅作为参考。

```
#!/bin/bash
# Environment req: /home/test3/miniconda3/envs/maker
maker \
  -base Microstegium_rnd1 \
  -cpus 16 \
  --ignore_nfs_tmp \
  maker_opts-firstrun-microstegium-parallel.ctl maker_bopts.ctl maker_exe.ctl
```

maker的并行化依赖于MPI framework。MPI是基于slurm的一个并行计算框架。故而mpiexec depends on node是否为可用状态。

Nevertheless，在lab的本地服务器上，当maker试图部署任务时，却出现 `srun: error: Unable to allocate resources: Zero Bytes were transmitted or received` 。调查lab 三台节点的可用情况，发现either down or drained. 重启需要root权限，所以**这一步直接放在并行上跑了**。

```
$ sinfo
#PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
#all*         up   infinite      1  down* node1
#all*         up   infinite      1  drain master
#all*         up   infinite      1   down node2
```

并行上使用的script如下：

```
#!/bin/bash
#SBATCH -p v6_384
#SBATCH -N 1
#SBATCH -n 40
#SBATCH -o ./logs/maker_round1.out
#SBATCH -e ./logs/maker_round1.err
source activate maker

export LIBDIR='/public1/home/sc80041/miniconda3/envs/maker/share/RepeatMasker/Matrices'
export LIBDIR='/public1/home/sc80041/miniconda3/envs/maker/share/RepeatMasker/Libraries'

mpiexec -n 40 maker \
  -base Microstegium_rnd1 \
  maker_opts-firstrun-microstegium-parallel.ctl maker_bopts.ctl maker_exe.ctl
```

运行结束后，检查maker evidence masking情况，发现在第一轮的maker run中只有极少量的基因/TE被map到原序列上。这意味着需要返回第一轮重新寻找问题。

这是**第一轮maker run中存在较多的contig mask失败造成的**。检查failed contig的log，发现是Perl Module `GFFDB::` 存在问题。

```
doing repeat masking
ERROR: Unknown strand in GFFDB::_sort_exons!
--> rank=NA, hostname=node2
```

这个错误与EDTA的TE annotation GFF3 strand表示方式有关。**Perl Module `GFFDB::` 无法识别标识为 `?` 的strand。**检查后发现

```
$ cat microstegium_TEanno.gff3 | cut -f7 | sort | uniq -c
# 528428 +
# 528946 -
#  14676 .
#   9852 ?
```

简单写一个python script 使用 `.` 替换掉 `GFFDB.pm` 识别不出的 `?` strand，问题解决。

```
#!/usr/bin/python
import sys

with open(sys.argv[1], mode='r') as in_gff_fh:
  with open(sys.argv[2], mode='w') as out_gff_fh:
    for entry in in_gff_fh:
      if not entry.startswith("#"):
        entry_split_list = entry.split("\t")

        if entry_split_list[6] == "?":
          entry_split_list[6] = "."

        out_gff_fh.write("\t".join(entry_split_list))
      else:
        out_gff_fh.write(entry)
```

3.3 Possible Errors & Solutions

科学网-基因组注释工具MAKER的报错与解决措施-2022.12.02-姚刚的博文 (sciencenet.cn)

1. **ERROR: Could not determine if DFam is installed**：由于RepeatMasker的DFam位置未设置造成，添加环境变量即可。需要强调的是，**并行上因为node之间存在交换数据的hard link，需要设置绝对路径而不是包含 `~` 等的相对路径。**

   ```
   # Lab server 本地加上这个
   export LIBDIR=/home/test3/miniconda3/envs/maker/share/RepeatMasker/Matrices
   export LIBDIR=/home/test3/miniconda3/envs/maker/share/RepeatMasker/Libraries

   # Parallel compute 并行加上这个
   export LIBDIR='/public1/home/sc80041/miniconda3/envs/maker/share/RepeatMasker/Matrices'
   export LIBDIR='/public1/home/sc80041/miniconda3/envs/maker/share/RepeatMasker/Libraries'
   ```

2. **EDTA GFF3注释的strand格式问题**：详见上文。

4 Gene Model Training

4.1 SNAP

> 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 of transcriptome sequences to the genome**, which you produced in the previous step.

首先，根据先前的transcriptome & protein evidence导出draft gene models，**阈值 AED>0.25 (Default: 0.5) 和 Aa_count>=50**

注释编辑距离（AED）：*用于评估基因组注释的准确性，AED是一个介于 0 和 1 之间的数字，衡量注释与支持它的evidence的拟合优度，0 表示与可用证据完全一致，1 表示缺乏对注释基因模型的支持*

```
#!/bin/bash
# Environment req: /home/test3/miniconda3/envs/maker
mkdir -p ./snap/round1 && cd ./snap/round1
gff3_merge -d ../../Microstegium_rnd1.maker.output/Microstegium_rnd1_master_datastore_index.log
# Export 'confident' gene models from MAKER
maker2zff -x 0.25 -l 50 Microstegium_rnd1.all.gff
```

验证导出draft gene model的基本信息： `gene-stats.log` & `validate.log` ，同时生成进一步模型训练所需的参数。

```
#!/bin/bash
# Environment req: /home/test3/miniconda3/envs/maker
# Stat gathering and validation
fathom genome.ann genome.dna -gene-stats > gene-stats.log 2>&1
fathom genome.ann genome.dna -validate > validate.log 2>&1

# Collect training sequences and annotations, plus 1000 surrounding bp for training
fathom genome.ann genome.dna -categorize 1000 > categorize.log 2>&1
fathom uni.ann uni.dna -export 1000 -plus > uni-plus.log 2>&1

# Create training params
mkdir params
cd params
forge ../export.ann ../export.dna > ../forge.log 2>&1
cd ..
```

在此基础上组装Hidden Markov Model，用于新一轮的maker run。

```
#!/bin/bash
# Environment req: /home/test3/miniconda3/envs/maker
hmm-assembler.pl Microstegium_rnd1.zff.length50_aed0.25 ./params > Microstegium_rnd1.zff.length50_aed0.25.hmm
```

整个SNAP流程生成**evidence在基因组上位置注释** **`Microstegium_rnd1.all.gff`** **和在前述阈值下被训练出的hmm模型** **`Themeda_rnd1.zff.length50_aed0.25.hmm`** 。这两文件用于下一步分析。

4.2 Augustus

Augustus训练Gene Model的时同时运行BUSCO评估基因组组装/注释质量的高低。

在第一轮Augustus的训练中，笔者使用**Putative gene sequence和其上下游各1000bp的序列**。这可以从第一轮evidence-based maker run生成的GFF3注释中提取。

```
#!/bin/bash
# Environment req: /home/test3/miniconda3/envs/maker
mkdir -p ./augustus/round1 && cd ./augustus/round1

awk -v OFS="\t" '{ if ($3 == "mRNA") print $1, $4, $5 }' \
  ../../snap/round1/Microstegium_rnd1.all.gff | \
awk -v OFS="\t" '{ if ($2 < 1000) print $1, "0", $3+1000; else print $1, $2-1000, $3+1000 }' | \
bedtools getfasta \
  -fi ../../microstegium_genome.fa \
  -bed - \
  -fo Microstegium_rnd1.all.maker.transcripts1000.fasta
```

针对可能出现的**序列结尾基因+1000bp index超出contig长度**的问题，workflow是这样解释的：

> You will likely get warnings from BEDtools that certain coordinates could not be used to extract FASTA sequences. This is because the end coordinate of a transcript plus 1000 bp is beyond the total length of a given scaffold. This script does account for transcripts being within the beginning 1000bp of the scaffold, but there was no easy way to do the same with transcrpts within the last 1000bp of the scaffold. This is okay, however, as we still end up with sequences from thousands of gene models and **BUSCO will only be searching for a small subset of genes itself**.

接下来配置busco-dependent databases。

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/71809c6a-0b92-4f7b-8fc1-d8e5d0beccaf/busco_env.yaml

在busco v5数据集 Index of /v5/data/lineages/ (ezlab.org) 中选取植物相关的数据库。

类群	数据库	BUSCO groups数量
真核生物	**eukaryota_odb10.2020-09-10.tar.gz**	255
绿色植物	**viridiplantae_odb10.2020-09-10.tar.gz**	425
有胚植物	**embryophyta_odb10.2020-09-10.tar.gz**	1614
真双子叶植物	**eudicots_odb10.2020-09-10.tar.gz**	2326
豆目	**fabales_odb10.2020-08-05.tar.gz**	5366

一次busco run只能使用一个lineage-specific dataset。因本文中注释单子叶禾本科高粱族物种的Genome，遂选择**有胚植物Embryophyta**的busco数据集。

User guide BUSCO v5.4.4 (ezlab.org) busco各参数的设置指引，其中可用的**近缘起始hmm (-sp)** 位于 `$AUGUSTUS_CONFIG_PATH/species`

```
#!/bin/bash
# Environment req: /home/test3/miniconda3/envs/busco
mkdir -p ./augustus/round1 && cd ./augustus/round1

busco \
  -i Microstegium_rnd1.all.maker.transcripts1000.fasta \
  -o Microstegium_rnd1_maker \
  --lineage_dataset path_to_busco_db/embryophyta_odb10/ \
  --mode genome \
  -sp maize \
  --evalue 1e-03 \
  --long --augustus \
  --cpu 40 --offline
```

busco输出的short summary可以反映gene prediction的大致质量。

```
# BUSCO version is: 5.2.2
# The lineage dataset is:  (Creation date: 2020-09-10, number of genomes: 50, number of BUSCOs: 1614)
# Summarized benchmarking in BUSCO notation for file path_to_project/Themeda/augustus/round1/Themeda_rnd1.all.maker.transcripts100
0.fasta
# BUSCO was run in mode: genome
# Gene predictor used: augustus

  ***** Results: *****

  C:91.1%[S:78.7%,D:12.4%],F:3.5%,M:5.4%,n:1614
  1471  Complete BUSCOs (C)
  1271  Complete and single-copy BUSCOs (S)
  200 Complete and duplicated BUSCOs (D)
  57  Fragmented BUSCOs (F)
  86  Missing BUSCOs (M)
  1614  Total BUSCO groups searched
```

接着进行一些augustus生成HMM数据的post-processing，以便下一轮的maker run。

```
echo $AUGUSTUS_CONFIG_PATH
#/public1/home/sc80041/miniconda3/envs/maker/config/
# augustus存放物种hmm文件的位置，maker可直接调用

cd Themeda/augustus/round1/Themeda_rnd1_maker/run_/augustus_output/retraining_parameters/BUSCO_Themeda_rnd1_maker
rename 's/BUSCO_Themeda_rnd1_maker/Themeda_triandra/' *

# 重命名config中的project nomenclature
sed -i 's/BUSCO_Themeda_rnd1_maker/Themeda_triandra/g' Themeda_triandra_parameters.cfg
sed -i 's/BUSCO_Themeda_rnd1_maker/Themeda_triandra/g' Themeda_triandra_parameters.cfg.orig1

mkdir $AUGUSTUS_CONFIG_PATH/species/Themeda_triandra
cp ./Themeda_triandra*  $AUGUSTUS_CONFIG_PATH/species/Themeda_triandra/
```

或者不进行pre-processing，直接**使用** `BUSCO_Themeda_rnd1_maker` **作为物种名**。

```
cd Themeda/augustus/round1/Themeda_rnd1_maker/run_/augustus_output/retraining_parameters/BUSCO_Themeda_rnd1_maker

mkdir /public1/home/sc80041/miniconda3/envs/maker/config/species/BUSCO_Themeda_rnd1_maker
cp ./BUSCO_Themeda_rnd1_maker*  /public1/home/sc80041/miniconda3/envs/maker/config/species/BUSCO_Themeda_rnd1_maker/
```

5 MAKER Annotation /w Models & Evidences

在两种Gene Predictor运行完后，运行第二轮的maker对这些gene model进行下一步的优化措施。

首先，从第一轮maker evidence-based annotation中提取**est、protein和repeat数据**。第一轮maker生成的注释文件有以下几个entry：

```
cat Themeda_rnd1.all.gff | cut -f2 | sort | uniq -c
#495566 blastn
#3860077 blastx
#457610 est2genome
#75054 evm
#305027 maker
#2637988 protein2genome
#383876 repeat_gff:edta
#1772962 repeatmasker
```

其中blastn/blastx entries为mapping上的破碎片段，弃之。

```
# Run in project root folder
mkdir ./maker-round1-annot-gff
cd ./snap/round1

# transcript alignments
awk '{ if ($2 == "est2genome") print $0 }' Themeda_rnd1.all.gff > ../../maker-round1-annot-gff/Themeda_rnd1.all.maker.est2genome.g
ff &
# protein alignments
awk '{ if ($2 == "protein2genome") print $0 }' Themeda_rnd1.all.gff > ../../maker-round1-annot-gff/Themeda_rnd1.all.maker.protein2
genome.gff &
# repeat alignments
awk '{ if ($2 ~ "repeat") print $0 }' Themeda_rnd1.all.gff > ../../maker-round1-annot-gff/Themeda_rnd1.all.maker.repeats.gff &
```

下面更改配置文件：

1. **Removing the FASTA sequences files to map and replacing them with the GFFs** (`est_gff`, `protein_gff`, and `rm_gff`, respectively).
2. Removing「**model_org**」entry.
3. **Specify the path to the SNAP HMM and the species name for Augustus,** so that these gene prediciton programs are run.
4. **Switch `est2genome` and `protein2genome` to 0** so that gene predictions are based on the Augustus and SNAP gene models.

变更如下几行：

```
est=
est_gff=/public1/home/sc80041/zhuyuzhi/andro_maker/Themeda/maker-round1-annot-gff/Themeda_rnd1.all.maker.est2genome.gff

protein=
protein_gff=/public1/home/sc80041/zhuyuzhi/andro_maker/Themeda/maker-round1-annot-gff/Themeda_rnd1.all.maker.protein2genome.gff

model_org= #select a model organism for DFam masking in RepeatMasker
rmlib=
rm_gff=/public1/home/sc80041/zhuyuzhi/andro_maker/Themeda/maker-round1-annot-gff/Themeda_rnd1.all.maker.repeats.gff

snaphmm=/public1/home/sc80041/zhuyuzhi/andro_maker/Themeda/snap/round1/Themeda_rnd1.zff.length50_aed0.25.hmm
augustus_species=BUSCO_Themeda_rnd1_maker #Augustus gene prediction species model

run_evm=0 #run EvidenceModeler, 1 = yes, 0 = no
est2genome=0 #infer gene predictions directly from ESTs, 1 = yes, 0 = no
protein2genome=0 #infer predictions from protein homology, 1 = yes, 0 = no
```

并行上使用script如下：

```
#!/bin/bash
#SBATCH -p v6_384
#SBATCH -N 1
#SBATCH -n 40
#SBATCH -o ./logs/maker_round2.out
#SBATCH -e ./logs/maker_round2.log

source activate maker

export LIBDIR='/public1/home/sc80041/miniconda3/envs/maker/share/RepeatMasker/Matrices'
export LIBDIR='/public1/home/sc80041/miniconda3/envs/maker/share/RepeatMasker/Libraries'

mpiexec -n 40 maker \
  -base Themeda_rnd2 \
  maker_opts-secondrun-themeda-parallel.ctl maker_bopts.ctl maker_exe.ctl
```

6 Quality Assessment & Iterative Training

maker可以利用上一轮的 *ab initio* gene prediction 进行新的预测，Iterative training直接通过重复step4 & 5来实现。

首先从第二轮maker run中导出gff-format annotation和全部的fasta-format transcript & protein sequences：Use the following command to create the final merged gff file. The **“-n” option would produce a gff file without genome sequences.**

```
#!/bin/bash
# Environment req: /home/test3/miniconda3/envs/maker
mkdir ./maker-round2-annot-gff

gff3_merge -s -d \
  ./Themeda_rnd2.maker.output/Themeda_rnd2_master_datastore_index.log \
  > ./maker-round2-annot-gff/Themeda_rnd2.all.gff

gff3_merge -s -n -d \
  ./Themeda_rnd2.maker.output/Themeda_rnd2_master_datastore_index.log \
  > ./maker-round2-annot-gff/Themeda_rnd2.noseq.gff

cd ./Themeda_rnd2.maker.output
fasta_merge -d Themeda_rnd2_master_datastore_index.log
```

6.1 Evaluation of Gene Models

第二轮maker run的annotation质量评估可以从以下几个方面入手。

1 - Count the **number of gene models and the gene lengths** after each round.

```
cat ./maker-round2-annot-gff/Themeda_rnd2.noseq.gff | awk '{ if ($3 == "gene") print $0 }' | awk '{ sum += ($5 - $4) } END { print
NR, sum / NR }' > ./maker-round2-annot-gff/genemodel_num_and_avg_genelen.txt &

# Themeda: 40566 3295.98
```

2 - Visualize the AED distribution. AED ranges from 0 to 1 and quantifies the confidence in a gene model based on empirical evidence. Basically, **the lower the AED, the better a gene model is likely to be**. Ideally, **95% or more of the gene models will have an AED of 0.5 or better (lower)** in the case of good assemblies. You can use this `AED_cdf_generator.pl` script to help with this.

```
perl stepex2-visualize-AED-distribution.pl -b 0.025 ./maker-round2-annot-gff/Themeda_rnd2.noseq.gff > ./maker-round2-annot-gff/aed
_distribution.txt &
```

以0.025作为fraction map annotation的AED累积情况。

```
# AED 累积gene%
......
0.350 0.891
0.375 0.905
0.400 0.923
0.425 0.933
0.450 0.945
0.475 0.952
0.500 0.960
0.525 0.964
......
```

可见，AED=0.5 threshold上覆盖了96%的genes，说明满足前述的标准，至此annotation质量已经达标，无需iterative maker run。

3 - Run BUSCO one last time using the species Augustus HMM and take a look at the results (this will be quick since we are not training Augustus). Also, **only include the transcript sequences (not the 1000 bp on each side)** and be sure to take the best (i.e., longest) transcript for each gene so you aren't artificially seeding duplicates.

7 Downstream Processing

Rename the IDs that MAKER sets by default for genes and transcripts.

MAKER comes with some scripts to do just this and to swap them out in the GFF and FASTA output (instructions for generated are above). The commands below first create custom IDs and store them as a table, and then use that table to rename the original GFF and FASTA files (they are overwritten, but it is possible to regenerate the raw ones again).

> 本workflow中，gene nomenclature prefix取**物种latin name属名第一个letter+种名前三个letter**

Justify numeric gene ids to this length : `6` - 最终生成的**Gene ID为** `Ttria_000001` **等**

```
#!/bin/bash
# Run in project root folder
mkdir ./maker-final-curation

# Copy final annotation to new folder, leaving original files as backup
cp ./Themeda_rnd2.maker.output/Themeda_rnd2.all.maker.proteins.fasta ./maker-final-curation/Themeda_rnd2.all.maker.proteins.fasta
cp ./Themeda_rnd2.maker.output/Themeda_rnd2.all.maker.transcripts.fasta ./maker-final-curation/Themeda_rnd2.all.maker.transcripts.
fasta

cp ./maker-round2-annot-gff/Themeda_rnd2.all.gff ./maker-final-curation/Themeda_rnd2.all.gff
cp ./maker-round2-annot-gff/Themeda_rnd2.noseq.gff ./maker-final-curation/Themeda_rnd2.noseq.gff

# create naming table
maker_map_ids --prefix Ttria_ --justify 6 ./maker-round2-annot-gff/Themeda_rnd2.all.gff > ./maker-final-curation/Themeda_rnd2.all.
maker.name.map

# replace names in GFF files
map_gff_ids ./maker-final-curation/Themeda_rnd2.all.maker.name.map ./maker-final-curation/Themeda_rnd2.all.gff &
map_gff_ids ./maker-final-curation/Themeda_rnd2.all.maker.name.map ./maker-final-curation/Themeda_rnd2.noseq.gff &

# replace names in FASTA headers
map_fasta_ids ./maker-final-curation/Themeda_rnd2.all.maker.name.map ./maker-final-curation/Themeda_rnd2.all.maker.transcripts.fas
ta
map_fasta_ids ./maker-final-curation/Themeda_rnd2.all.maker.name.map ./maker-final-curation/Themeda_rnd2.all.maker.proteins.fasta
```

接着对gene model按AED进行过滤，筛除掉AED>0.5的models

```
perl stepex3-gff-aed-quality-filter.pl -a 0.5 ./maker-final-curation/Themeda_rnd2.noseq.gff > ./maker-final-curation/themeda_aed0.
5_noseq_final.gff
```

此外因使用的Genome是经过**fasta header conversion**的（见Step0 Pre-processing），所以需要在最终注释中将其还原，并生成还原后的Genome fasta & GFF-style annotation。这一步不难，所以放到最后来。

```
cat ./themeda_translation.tab | cut -d" " -f 1 > ./maker-final-curation/themeda_seq_translation_brief.tab
#
#==Before conversion==
#Seq_1  JAENPR010000004.1 Themeda triandra isolate sf006 tig00000016, whole genome shotgun sequence
#Seq_2  JAENPR010000002.1 Themeda triandra isolate sf006 tig00000023, whole genome shotgun sequence
#
#==After conversion==
#Seq_1  JAENPR010000004.1
#Seq_2  JAENPR010000002.1

perl stepex4-translate_fasta_headers.pl -t ./maker-final-curation/themeda_seq_translation_brief.tab themeda_genome.fa > ./maker-fi
nal-curation/themeda_genome_converted_final.fa
```

接下来处理GFF，仅保留基因注释并像前面一样转换contig headers。这里又写了一个in-house script.

```
#!/usr/bin/python

# USAGE:
# python3 step8-extract-genic-gff-convert.py \
#     ./maker-final-curation/themeda_seq_translation_brief.tab \
#     ./maker-final-curation/themeda_aed0.5_noseq_final.gff \
#     ./maker-final-curation/themeda_aed0.5_genic_final.gff
import sys

seq_tr_dict = {}  # Seq_1 (Nomenclature IDs) -> NC000000.1 (Meaningful sequence names)

with open(sys.argv[1], mode='r') as in_seqtr_tab_fh:
  for entry in in_seqtr_tab_fh:
    #print(entry.split("\t"))
    if entry.split("\t")[1]:
      seq_tr_dict[str(entry.split("\t")[0])] = str(entry.split("\t")[1]).rstrip("\n")
    else:
      print("Invalid translation tab. Check again.")


with open(sys.argv[2], mode='r') as in_gff_fh:
  with open(sys.argv[3], mode='w') as out_gff_fh:

    for entry in in_gff_fh:
      if not entry.startswith("#"):
        entry_split_list = entry.split("\t")

        if str(entry_split_list[2]) == "contig":
          if seq_tr_dict[str(entry_split_list[0])]:
            entry_split_list[-1] = "ID=%s;Name=%s\n" % (seq_tr_dict[str(entry_split_list[0])], seq_tr_dict[str(entry_split_list[0])])
            entry_split_list[0] = seq_tr_dict[str(entry_split_list[0])]
          else:
            print("Warning: no sequence match in translation tab:\n%s" % entry)
          out_gff_fh.write("\t".join(entry_split_list))

        if str(entry_split_list[2]) in ["gene", "mRNA", "exon", "CDS", "five_prime_UTR", "three_prime_UTR"]:
          if seq_tr_dict[str(entry_split_list[0])]:
            entry_split_list[0] = seq_tr_dict[str(entry_split_list[0])]
          else:
            print("Warning: no sequence match in translation tab:\n%s" % entry)
          out_gff_fh.write("\t".join(entry_split_list))
      else:
        out_gff_fh.write(entry)
```

最终结果：

- 基因注释：themeda_aed0.5_genic_final.gff
- 转录本：Themeda_rnd2.all.maker.transcripts.fasta
- 蛋白：Themeda_rnd2.all.maker.proteins.fasta
- soft-mask且转换过headers的基因组：themeda_genome_converted_final.fa

8 Protein Annotation & Functional Enrichment

对蛋白序列使用**本地eggNOG-mapper**跑一遍注释。

```
#!/bin/bash
# Environment req: /data21/zhuyuzhi/condaenvs/eggnog_mapper

python /data21/zhuyuzhi/databases/eggNOG_mapper/emapper.py \
  -i Themeda_rnd2.all.maker.proteins.fasta \
  -m diamond \
  --output Ttria \
  --cpu 20 \
  --excel
```

生成**蛋白注释文件** `Ttria.emapper.annotations` ，用于下一步的分析。

```
ls
#Themeda_rnd2.all.maker.proteins.fasta  Ttria.emapper.annotations  Ttria.emapper.hits  Ttria.emapper.seed_orthologs
```

9 Appendix

Scripts used

```
https://github.com/EdwardZhu02/maker-annot-workflow
```