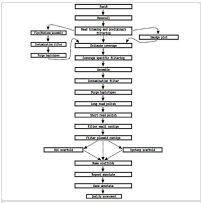

1

Jan 13, 2022

Long-read plant genome assembly and annotation: annotation V.1

Scott Ferguson[1], Ashley Jones[1], Justin Borevitz[1]

[1]Australian National University

1

protocol .

Scott Ferguson
Australian National University

With the introduction of long-read, third generation sequencing (e.g. Oxford Nanopore Technologies (ONT) and Pacific Biosciences (PacBio) and associated bioinformatics tools, we can now affordably assemble complex plant genomes with an accuracy and contiguity that was previously incredibly expensive to obtain. Long-read sequencing technology has dramatically improved our ability to generate de novo genome assemblies. In particular plant genomes, due to their large proportion of repetitive sequences, have highly benefited from long-read sequencing. Despite these improvements, challenges still remain in performing de novo assembly, namely in developing a reliable workflow and in tool choice. Here we present the annotation proportion of our long-read plant genome assembly, scaffolding, and annotation workflow. We have developed and tested this workflow on many complex plant genomes. By using this workflow with a sufficient coverage of long-reads, a highly contiguous and accurate plant genome can be assembled.

Scott Ferguson, Ashley Jones, Justin Borevitz 2022. Long-read plant genome assembly and annotation: annotation. **protocols.io** https://protocols.io/view/long-read-plant-genome-assembly-and-annotation-ann-bw44pgyw

protocol ,

Aug 04, 2021

Jan 13, 2022

52092

Repeat annotate

1

Now that your genome is complete we want to find and annotate it for repeat regions. To do

this we use EDTA and RepeatMasker. EDTA is first used to build a transposon (TE) sequence library. Next, RepeatMasker is used to find all TE and simple repeats within the genome.

Repeat mask

```
genome="/path to genome/genome.fasta"
cpus=XX
mkdir EDTA
cd  EDTA
EDTA.pl --genome $genome --anno 1 --threads $cpus
cd ../
RepeatMasker -pa $cpus -s -lib EDTA/$(basename
$genome).mod.EDTA.TElib.fa -dir repeatMask  -e ncbi $genome
```

Repeat annotate: soft mask

2 Output from ReapeatMasker will include a hard masked genome (all nucleotides within repeat regions will be changed to "N") among other outputs. We will use the repeat regions defined within the out file and BEDTools to create a soft masked version of your genome (all nucleotides within repeat regions will be changed to lowercase, i.e. A becomes a, C becomes c, etc).

First, we create a bed file from the out file. Next BEDTools is used to soft mask our genome using the regions within the bed file.

Soft mask

```
genome="/path to genome/genome.fasta"
out="/path to repeat results/genome.fasta.out"
label="speciesName"

sed -e '1,3d' $out | awk -v OFS='\t' '{print $5, $6-1, $7}' | sort -k1,1 -
k2,2 -V > ${label}.bed
bedtools maskfasta -soft -fi $genome -bed ${label}.bed -fo ${label}-
softMasked.fasta
```

Gene annotate

3 Gene annotation can be done using RNA and/or gene homology. The tool we use is BRAKER2, which is capable of using both RNA and protein sequences of known genes. BRAKER2 will use your RNA or gene sequences to train a hidden markov model that scans your genome for regions likely to be genes. It will annotate both introns and exons, producing a gtf or gff3 file from which gene nucleotide and protein sequences can be made.

For gene annotation use a soft masked version of your genome, steps 1-2 (above).

Installing BRAKER2 can be hard and is very specific to your compute environment.

Gene annotate: Gene sequences/homology

4 From the NCBI (or another gene repository) find your species genus and download all gene amino acid sequences available. We also like to download the Arabidopsis thaliana genes and concatenate them to your genus's genes. Use this gene dataset for gene finding with BRAKER2.

BRAKER2 homology

```
genome="/path to genome/softmasked.fasta"
geneSet="/path to gene sets/A_thaliana-genusGenes.faa"
cpus=XX
label=XXXXX

braker.pl --genome=${genome} \
  --prot_seq=${geneSet} \
  --epmode \
  --softmasking \
  --workingdir=/fast disk/species-annotate \
  --cores=${cpus} \
  --species=${label} \
  --gff3
```

Gene annotate: RNA

5 Gene annotation with RNA begins by aligning your RNA to your genome. To do this a split read aware aligner is needed. The two current best aligners for this task are STAR and HISAT2 (https://daehwankimlab.github.io/hisat2/), both aligners give good results. We prefer to use STAR.

RNA alignment with STAR may benefit from dataset specific parameters. A very useful manual for STAR can be found on the STAR's github; https://github.com/alexdobin/STAR

Set “genomeSAindexNbases” to $min(14, \frac{log_2(GenomeLength)}{2} - 1)$

STAR - RNA alignment

```
label=""
genome=""
RNAReads="/path to reads/RNA.fastq"
cpus=XX

# index genome
mkdir index
STAR --runThreadN $cpus \
  --runMode genomeGenerate \
  --genomeSAindexNbases XX \
  --genomeDir index \
  --genomeFastaFiles $genome \
  --outTmpDir /fast temp/disk space/

# align RNA reads
STAR --runThreadN $cpus \
   --runMode alignReads \
   --genomeDir index \
   --readFilesIn $RNAReads \
   --outFileNamePrefix /save dir/${label}-RNA \
   --outTmpDir /fast temp/disk space/ \
   --outSAMunmapped Within
```

6 Now that your RNA is aligned to your genome, we proceed to gene annotation with BRAKER2.

BRAKER2 RNA

```
label=""
genome="/path to genome/softmasked.fasta"
bam="/path to STAR output/XXX.bam"
cpus=XX

braker.pl --species=${label} \
 --genome=${genome} \
 --bam=${bam} \
--softmasking \
 --workingdir=/fast disk/species-annotate \
 --cores=${cpus}
```

Gene annotate: Create protein sequences

7 Whether you did your annotation with RNA or gene homology, BRAKER2 will produce a gtf file which describes the location of genes within your genome (for a description of the gtf file format see: http://asia.ensembl.org/info/website/upload/gff3.html).

To create both amino acid and nucleotide sequences of your genes run the following:

Create proetin sequences

```
genome="/path to genome/softmasked.fasta"
gtf="/path to braker results/braker.gtf"
label=""

/path to BRAKER2/bin/getAnnoFastaFromJoingenes.py -g $genome -f
$gtf -o $label
```