Jun 12, 2024

Structural and Functional Annotation of bee genome

DOI
dx.doi.org/10.17504/protocols.io.3byl49wwogo5/v1

Rafael Rodrigues Ferrari[1], Thiago Mafra Batista[1]

[1]Universidade Federal do Sul da Bahia

bioinfo

Thiago Mafra Batista
Universidade Federal do Sul da Bahia

OPEN ACCESS

DOI: **dx.doi.org/10.17504/protocols.io.3byl49wwogo5/v1**

Protocol Citation: Rafael Rodrigues Ferrari, Thiago Mafra Batista 2024. Structural and Functional Annotation of bee genome. **protocols.io https://dx.doi.org/10.17504/protocols.io.3byl49wwogo5/v1**

Manuscript citation:
Rafael R Ferrari, Thiago M Batista, Qing-Song Zhou, Heron O Hilário, Michael C Orr, Arong Luo, Chao-Dong Zhu, The Whole Genome of *Colletes collaris* (Hymenoptera: Colletidae): An Important Step in Comparative Genomics of Cellophane Bees, *Genome Biology and Evolution*, Volume 15, Issue 5, May 2023, evad062, https://doi.org/10.1093/gbe/evad062

Protocol status: Working
We use this protocol and it's working

Created: June 11, 2024

Last Modified: June 12, 2024

Protocol Integer ID: 101589

Keywords: gene prediction, functional annotation, maker2, interproscan

Abstract

This protocol provides detailed, step-by-step instructions for students and researchers to annotate nuclear genomes, using the genome of the Asian bee *Colletes collaris* as an example. We begin with the identification of repetitive regions, followed by gene prediction, functional annotation of protein-coding genes, and identification of non-coding RNAs (ncRNAs and tRNAs).

IDENTIFICATION OF REPETITIVE REGIONS

1 **Identification of repetitive regions**

****RepearModeler (on Kiko)****

Building a database

```
$ /home/thiagomafra/instaladores/RepeatModeler-2.0.3/BuildDatabase
-name collaris13 ./genome.nextpolish.fasta
```

Run RepeatModeler

```
$ /home/thiagomafra/instaladores/RepeatModeler-2.0.3/RepeatModeler
-database collaris13 -pa 6 -trf_dir /home/thiagomafra/bin/ > log
```

****RepeatMasker (on kiko)****

Masking genome

#Use the -families.fa

```
$ /home/thiagomafra/instaladores/RepeatMasker/RepeatMasker -pa 24
-lib /home/thiagomafra/collaris/repeatmodeler_run/mafra/collaris-
families.fa -dir . \
-small -gff
/home/thiagomafra/collaris/repeatmasker_run/fafinha/run3_final/gen
ome.nextpolish.fasta > log
```

Generating a summary

```
$/usr/local/bin/perl
/home/thiagomafra/instaladores/RepeatMasker/util/buildSummary.pl
./genome.nextpolish.fasta.out > ./genome.nextpolish.masked.summary
```

Generating a .gff3 for use in MAKER

Converting the .out into a .gff3

```
$/home/thiagomafra/instaladores/RepeatMasker/util/rmOutToGFF3.pl
./genome.nextpolish.fasta.out > ./genome.nextpolish.masked.gff3
```

Isolating complex repeats

```
$grep -v -e "Satellite" -e ")n" -e "-rich"
./genome.nextpolish.masked.gff3 >
./genome.nextpolish.masked.complex.gff3
```

Reformatting the .complex.gff3

```
$cat ./genome.nextpolish.masked.complex.gff3 | perl -ane '$id;
if(!/^\#/){@F = split(/\t/, $_); chomp $F[-1];$id++; $F[-1] .=
"\;ID=$id"; $_ = join("\t", @F)."\n"} print $_' >
./genome.nextpolish.masked.complex.formatted.gff3
```

GENE PREDICTION (STRUCTURAL ANNOTATION)

2

****MAKER2 (https://github.com/sujaikumar/assemblage/blob/master/README-annotation.md) (on kiko)****
Generate control files

```
$maker -CTL
```

MAKER 1st pass

Edit file maker_opts.ctl

Run Maker

```
$mpiexec.openmpi -np 12 maker -base pass1 &> log
```

2.1 **Train Augustus (convert the combined .gff file into Augustus HMMs)**

Filter gff file

```
$awk '{if ($2=="maker") print }'
../../pass1.maker.output/pass1.all.gff > maker_pass1.gff
```

Produce a Genbank-formated file named collaris.gb

```
$gff2gbSmallDNA.pl maker_pass1.gff
/home/thiagomafra/collaris/genome.nextpolish.fasta 2000
collecolla2.gb
```

#To check the number of genes in training set

```
$grep -c LOCUS collecolla2.gb
```

Create a new Augustus species name

```
$new_species.pl --species=collecolla2
```

Initiate training

```
$etraining --species=collecolla2 collecolla2.gb
```

#The initial model should be in the directory below:

```
$AUGUSTUS_CONFIG_PATH/species/collecolla
```

Run a test set for evaluation before optimization

```
$randomSplit.pl collecolla2.gb 200

$mv collecolla2.gb.test collecolla2.gb.evaluation
```

Predict the genes and check the results

```
$augustus --species=collecolla2 collecolla2.gb.evaluation >&
first_evaluate.out

$grep -A 22 Evaluation first_evaluate.out
```

Optimize model

```
$randomSplit.pl collecolla2.gb 1000

$optimize_augustus.pl --species=collecolla2 --kfold=4 --cpus=12 --
rounds=5 --onlytrain=collecolla2.gb.train collecolla2.gb.test >&
log
```

Train again after optimization

```
$etraining --species=collecolla2 collecolla2.gb
```

Use the optionized model to evaluate again and check the results

```
$augustus --species=collecolla2 collecolla2.gb.evaluation >&
second_evaluate.out

$grep -A 22 Evaluation second_evaluate.out
```

2.2 ***MAKER 2nd pass***

Generate new control files

```
$maker -CTL
```

Produce separate gffs

```
$awk '{if ($2=="est2genome") print }' pass1.all.gff >
pass1.all.est2genome.gff

$awk '{if ($2=="protein2genome") print }' pass1.all.gff >
pass1.all.protein2genome.gff

$awk '{if ($2~"repeat") print }' pass1.all.gff >
pass1.all.repeats.gff
```

Edit file maker_opts.ctl

Run Maker

```
$mpiexec.openmpi -np 12 maker -base pass2
```

Combine all the fasta and GFFs

```
$gff3_merge -n -d pass2.5_master_datastore_index.log

$fasta_merge -d pass2.5_master_datastore_index.log
```

Check AED

```
$AED_cdf_generator.pl -b 0.05 pass2.5.all.gff > AED_tab

$sed 's/AED:/AED:\t/g'
pass2.5.all.maker.augustus_masked.transcripts.fasta | grep ">" |
awk '{print $5*10}' | sed 's/\./\t/g' | cut -f 1 | sort | uniq -c
| sort -k2,2n
```

Rename genes

```
$maker_map_ids --prefix CCOLL_ --justify 5 --iterate 1 --abrv_gene
G --abrv_tran T pass2.3.all.gff > pass2.3.all.maker.id.map

$map_gff_ids pass2.3.all.maker.id.map pass2.3.all.gff

$map_fasta_ids pass2.3.all.maker.id.map
pass2.3.all.maker.transcripts.fasta

$map_fasta_ids pass2.3.all.maker.id.map
pass2.3.all.maker.proteins.fasta
```

FUNCTIONAL ANNOTATION

3 ****Diamond (https://github.com/bbuchfink/diamond/wiki)****

Run the .maker.transcripts.fasta against Swiss-Prot

Prepare a .pbs file to run the analysis remotely on Sagarana

```
diamond blastx -q
/home/fafinha/collaris/Diamond_run/final_run/pass2.5.all.maker.tra
nscripts.renamed.fasta -f 100 -k 5 --sensitive -p 64 \
-d /home/fafinha/collaris/Diamond_run/uniprot_sprot.fasta -o
/home/fafinha/collaris/Diamond_run/final_run/blastx_vs_sprot.daa -
-query-cover 0.5 \
--subject-cover 0.5
```

Convert de .daa output file into a .fmt6 file

```
$diamond view -a blastx_vs_sprot.daa -o blastx_vs_sprot.fmt6 -f 6
```

Keep only best hits

```
$cat blastx_vs_sprot.fmt6 | sort -k1,1 -k12,12nr -k11,11n | sort -
k1,1 -u > blastx_vs_sprot_besthits.fmt6
```

Produce a list of identifiers that matched Swiss-Prot

```
$cat blastx_vs_sprot.fmt6 | awk '{print $1}' | uniq >
uniprot_hits.list
```

Produce a .fasta file with the sequences that did not match Swiss-Prot

```
$seqkit grep -v -f uniprot_hits.list
pass2.all.maker.transcripts.fasta > uniprot_nohits.fasta
```

Run Diamond agains Trembl

Prepare a .pbs file to run the analysis remotely on Sagarana

```
diamond blastx -q
/home/fafinha/collaris/Diamond_run/final_run/uniprot_nohits.fasta
-f 100 -k 5 --sensitive -p 64 --query-cover 0.5 --subject-cover
0.5 \
-d /home/fafinha/collaris/Diamond_run/uniprot_trembl.dmnd -o
/home/fafinha/collaris/Diamond_run/final_run/blastx_vs_trembl.daa
```

Convert de .daa output file into a .fmt6 file

```
$diamond view -a blastx_vs_trembl.daa -o blastx_vs_trembl.fmt6 -f
6
```

Keep only best hits

```
$cat blastx_vs_trembl.fmt6 | sort -k1,1 -k12,12nr -k11,11n | sort
-k1,1 -u > blastx_vs_trembl_besthits.fmt6
```

Merge output files

```
$cat blastx_vs_sprot_besthits.fmt6 blastx_vs_trembl_besthits.fmt6
> blastx_combined_output.fmt6
```

3.1 ****InterProScan (https://github.com/ebi-pf-team/interproscan)****

Identy protein domains

```
$/uvstorage/my_interproscan/interproscan-5.53-87.0/interproscan.sh
-i ./pass2.all.maker.proteins.fasta -iprlookup -goterms -pa --cpu
64 -f tsv
```

ANNOTATION PROCESSING

3.2 *****Add functional annotation to Maker outputs*****

```
$maker_functional_gff ../uniprot_sprot_trembl.fasta
blastx_combined_output.fmt6 pass2.3.all.renamed.gff >
pass2.3.all.blastx.gff

$ipr_update_gff pass2.all.blastx.gff
pass2.3.all.maker.proteins.tsv > collaris_genome_annotation.gff

$maker_functional_fasta ../uniprot_sprot_trembl.fasta
blastx_combined_output.fmt6 pass2.all.maker.transcripts.fasta >
collaris_transcripts_annotation.fasta

$maker_functional_fasta ../uniprot_sprot_trembl.fasta
blastx_combined_output.fmt6 pass2.all.maker.proteins.fasta >
collaris_proteins_annotation.fast
```

3.3 *******ANNOTATION STATISTICS*******

****Count complete sequences (on kiko)****

Genes

```
$awk '{if ($2=="maker" && $3=="gene")print}'
collaris_genome_annotation.gff > complete.gene.gff

$bedtools getfasta -fi ../../genome.nextpolish.fasta -bed
complete.gene.gff -name > complete.gene.fasta
```

CDS

```
$awk '{if ($2=="maker" && $3=="CDS")print}'
collaris_genome_annotation.gff > cds.gff

$bedtools getfasta -fi ../../genome.nextpolish.fasta -bed cds.gff
-name > cds.fasta
```

Exons

```
$awk '{if ($2=="maker" && $3=="exon")print}'
collaris_genome_annotation.gff > exon.gff

$bedtools getfasta -fi ../../genome.nextpolish.fasta -bed exon.gff
-name > exon.fasta
```

Introns

```
$python /home/thiagomafra/instaladores/Extract-intron-from-
gff3/scripts/extract_intron_gff3_from_gff3.py
collaris_genome_annotation.gff introns.gff

$awk '/intron\t/{print}' introns.gff | sort -k1,1 -k4,2n >
introns_processed.gff

$bedtools getfasta -fi ../../genome.nextpolish.fasta -bed
introns_processed.gff -name > intron.fasta
```

****Count nucleotides in fasta files (on kiko)****

```
$for file in *.fasta; do echo $file; cat $file | grep -v ">" | wc
-m; done
```

****Count evidence from interproscan (on sagarana)****

```
$cat pass2.5.all.maker.proteins.renamed.tsv | awk '{print $1}' |
uniq | wc -l
$grep "Pfam" pass2.3.all.maker.proteins.renamed.fasta.tsv | awk
'{print $1}' | uniq | wc -l
$grep "Reactome" pass2.3.all.maker.proteins.renamed.fasta.tsv |
awk '{print $1}' | uniq | wc -l
$grep "Gene3D" pass2.3.all.maker.proteins.renamed.fasta.tsv | awk
'{print $1}' | uniq | wc -l
$grep "GO" pass2.3.all.maker.proteins.renamed.fasta.tsv | awk
'{print $1}' | uniq | wc -l
$grep "PANTHER" pass2.3.all.maker.proteins.renamed.fasta.tsv | awk
'{print $1}' | uniq | wc -l
$grep "MetaCyc" pass2.3.all.maker.proteins.renamed.fasta.tsv | awk
'{print $1}' | uniq | wc -l
$grep "MobiDBLite" pass2.3.all.maker.proteins.renamed.fasta.tsv |
awk '{print $1}' | uniq | wc -l
$grep "MetaCyc" pass2.3.all.maker.proteins.renamed.fasta.tsv | awk
'{print $1}' | uniq | wc -l
$grep "ProSiteProfiles"
pass2.3.all.maker.proteins.renamed.fasta.tsv | awk '{print $1}' |
uniq | wc -l
$grep "SMART" pass2.3.all.maker.proteins.renamed.fasta.tsv | awk
'{print $1}' | uniq | wc -l
$grep "CDD" pass2.3.all.maker.proteins.renamed.fasta.tsv | awk
'{print $1}' | uniq | wc -l
$grep "Coils" pass2.3.all.maker.proteins.renamed.fasta.tsv | awk
'{print $1}' | uniq | wc -l
$grep "ProSitePatterns"
pass2.3.all.maker.proteins.renamed.fasta.tsv | awk '{print $1}' |
uniq | wc -l
$grep "PRINTS" pass2.3.all.maker.proteins.renamed.fasta.tsv | awk
'{print $1}' | uniq | wc -l
$grep "TIGRFAM" pass2.3.all.maker.proteins.renamed.fasta.tsv | awk
'{print $1}' | uniq | wc -l
$grep "PIRSF" pass2.3.all.maker.proteins.renamed.fasta.tsv | awk
'{print $1}' | uniq | wc -l
$grep "Hamap" pass2.3.all.maker.proteins.renamed.fasta.tsv | awk
'{print $1}' | uniq | wc -l
$grep "SFLD" pass2.3.all.maker.proteins.renamed.fasta.tsv | awk
'{print $1}' | uniq | wc -l
```

****Count evidence from UniProt****

```
$cat blastx_combined_output.fmt6 | wc -l
$cat blastx_vs_sprot_besthits.fmt6 | wc -l
$cat blastx_vs_trembl_besthits.fmt6 | wc -l
```

SEARCHING FOR NON-CODING RNAs (ncRNAs)

4 ****Infernal (https://github.com/EddyRivasLab/infernal)****

Prepare a subject database

Download the Rfam database

```
$wget http://ftp.ebi.ac.uk/pub/databases/Rfam/CURRENT/Rfam.tar.gz
```

Decompress the file

```
$tar -xvzf Rfam.tar.gz
```

Merge all .cm files

```
$cat *.cm > rfam.cm
```

Run Infernal

```
$cmsearch -g --noali -E 1e-6 --tblout infernal.tblout -o
infernal.out rfam.cm genome.nextpolish.fasta
```

Prepare a list of identifiers for annotation

Go to https://rfam.org/search#tabview=tab5

Check all the boxes and then hit 'submit'

Go to the bottom of the page and click on 'Show the unformatted list'

Copy and paste the list onto a text editor and save the file as 'rfam-types.txt'

Produce a list of identifiers

```
$cat rfam-types.txt | awk '{ print $1 }' > rfam-ids.txt
```

Filter Infernal results

```
$grep -f rfam-ids.txt infernal.out > infernal.hits.out

$cat infernal.hits.out | awk '{ print $2 }' >
infernal.hits.filtered.out

$grep -f infernal.hits.filtered.out rfam-types.txt >
ncRNAs_annotation.txt
```

SEARCHING FOR TRANSPORTING RNAs (tRNAs) ONLY

5 ****tRNAscan-SE (https://github.com/UCSC-LoweLab/tRNAscan-SE)****

Prepare a .pbs file to run the analysis remotely on Sagarana

```
/home/fafinha/bin/tRNAscan-SE-2.0/tRNAscan-SE
/home/fafinha/collaris/NextPolish_run/NexDenovo/run1_RF_final/long
_short_reads/01_rundir/genome.nextpolish.fasta\
-o /home/fafinha/collaris/tRNAscan-SE_run/genome/trnascan.output -
j /home/fafinha/collaris/tRNAscan-SE_run/genome/trnascan.gff \
-a /home/fafinha/collaris/tRNAscan-SE_run/genome/trnascan.fasta -G
-I --thread 128
```

Count the number of identified tRNAs

```
$grep -c ">" trnascan.fasta #(including pseudo tRNAs)
```