GENE-SWitCH

The regulatory GENomE of SWine and CHicken: functional annotation during development

Protocol WP2
Chicken gene annotation with Isoseq sequencing using isoseq nextflow pipeline and downstream analysis.

Authors: Sébastien Guizard (UEDIN)

Workpackage: WP2

Version: 1.0

Protocol associated with Deliverable(s):	D2.1 D2.2
Submission date to FAANG:	23/03/2022

Research and Innovation Action, SFS-30-2018-2019-2020 Agri-Aqua Labs
Duration of the project: 01 July 2019 – 30 June 2023, 48 months

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 817998

Table of contents

1 Summary 4
2 Protocol description 5
2.1 Genome annotation 5
2.2 Read count computing 7
2.3 Long Non Coding RNA detection 10
2.4 Add Ensembl IDs 10
2.5 Add information to GTF file 10

Table of figures

Figure 1: Isoseq nexflow pipeline5
Figure 2: Read counting process8

1 Summary

GENE-SWitCH aims at identifying functional elements located in the genomes of the pig and chicken working on seven different tissues at three different developmental stages.

The seven tissues analysed in GENE-SWitCH are:

- Cerebellum
- Lung
- Kidney
- Dorsal skin
- Small intestine
- Liver
- Skeletal muscle

The three developmental stages are:

- Early organogenesis (E8 chick embryo and D30 pig foetuses)
- Late organogenesis (E15 chick embryo and D70 pig foetuses)
- Newborn piglets and hatched chicks

For each tissue at each time point, an Isoseq long read sequencing has been done. The raw subreads needs to be processed to generates definitive consensus sequences. The reads are mapped on the reference genome with uLTRA. The gene models are cleaned with TAMA. The resulting annotation files have to be processed to convert to the GFF format and add information (read count, annotation confidence).

2 Protocol description

The following protocol has been applied on either chicken and pig isoseq datasets.

2.1 Genome annotation

The isoseq raw subreads processing, the genome mapping and the gene model annotation creation are made using the isoseq nextflow pipeline.

First the pipeline generates Circular Consensus Sequences (CCS) using `ccs` Pacbio's program. The raw data are divided into batches of sequences that are processed in parallel. On each ccs batch, the program `LIMA` (from Pacbio) select CCS with appropriate primers pairs and removes them from the sequence. The resulting sequences are then processed by Pacbio's `isoseq3 refine`. It selects non-chimeric sequences with poly(A) tail. The sequences in `BAM` format are converted into `FASTA` format using `bamtools convert`. The program `polyacleanup` from `TAMA` is then apply to remove remaining polyA tails. At the end of the cleaning process, sequences are called Full Length Non Chimeric (FLNC).

Each batch of FLNC is aligned on the reference genome with the program `uLTRA`.

Each alignment is processed by `TAMA collapse`, for false positive and redundancy removing. A bed annotation file is generated for each batch of sequences. Those batches are regrouped based on their sample of origin and merge into one annotation by `TAMA merge`.

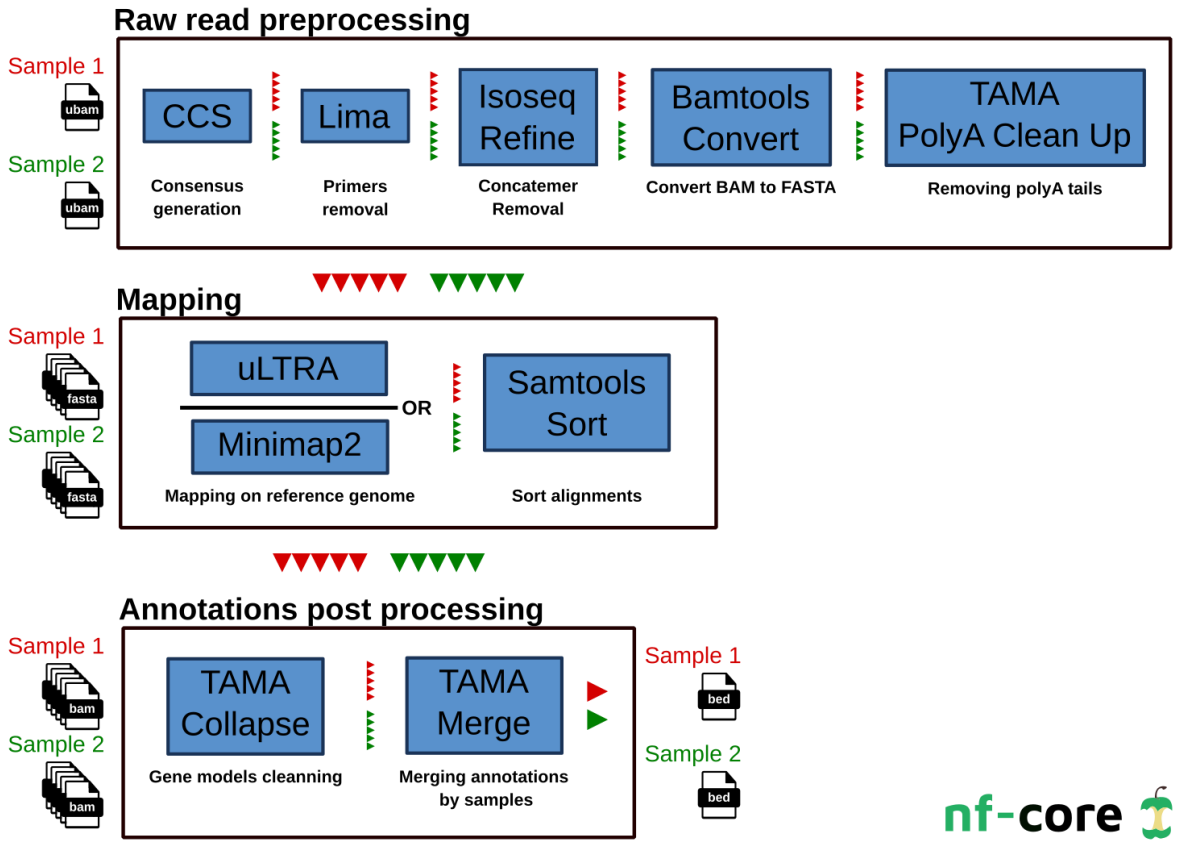

Figure 1: Isoseq nexflow pipeline

To run the pipeline, a data directory is created and the following input files are stored in it:

- Samples subreads (.bam)
- Pacbio index files (.bam.pbi)
- Reference genome (.fasta)
- Primer sequences (.fasta)
- Ensembl gene annotation (.gtf)

The genome used is Gallus Gallus 6 (GRCg6a) and its gene annotation from Ensembl release-105. Contrary to PacBio's instructions, the primer sequence remains unmodified and the polyA sequence is conserved:

```
>5p
TGGATTGATATGTAATACGACTCACTATAG
>3p
AAAAAAAAAAAAAAAAAACGCCTGAGA
```

The following command line has been used to run the pipeline on each samples:

```
nextflow run sguizard/isoseq \
-r dev-rkv-eddie \
--input ../data/ \
--primers ../data/primers_complete.fasta \
--fasta ../data/Gallus_gallus.GRCg6a.dna.toplevel.fa \
--chunk 300 \
--rq 0.90 \
--min_passes 2 \
--five_prime 2000 \
--splice_junction 10 \
--three_prime 2000 \
--capped \
--ultra \
--gtf ../data/Gallus_gallus.GRCg6a.105.gtf \
-profile singularity,eddie
```

The files 30 bed files obtained for the 21 pairs of tissues/development stages have been merged together using `TAMA merge`.

```
tama_merge.py -f inputChicken.tsv -d merge_dup -a 2000 -m 10 -z 2000 -p chicken 1>
merge.log 2> merge.err
```

The "inputChicken.tsv" file is a four-column tabulation separated file. It contains the list of annotation to merge (column 1), the indication if the sequenced RNAs were capped (column 2), the priority annotation merging (column 3), and an id (column 4).

```
E15_Skin_P2_cleaned.bed      capped    1,1,1    E15_Skin_P2
HC_Lung_cleaned.bed          capped    1,1,1    HC_Lung
E8_Brain_cleaned.bed         capped    1,1,1    E8_Brain
HC_Skin_P1_cleaned.bed       capped    1,1,1    HC_Skin_P1
E15_Skin_P1_cleaned.bed      capped    1,1,1    E15_Skin_P1
E15_Muscle_P1_cleaned.bed    capped    1,1,1    E15_Muscle_P1
HC_Liver_P2_cleaned.bed      capped    1,1,1    HC_Liver_P2
HC_Brain_P1_cleaned.bed      capped    1,1,1    HC_Brain_P1
E15_Liver_cleaned.bed        capped    1,1,1    E15_Liver
HC_Muscle_cleaned.bed        capped    1,1,1    HC_Muscle
E8_Liver_cleaned.bed         capped    1,1,1    E8_Liver
E8_Lung_cleaned.bed          capped    1,1,1    E8_Lung
HC_Skin_P2_cleaned.bed       capped    1,1,1    HC_Skin_P2
E8_Muscle_cleaned.bed        capped    1,1,1    E8_Muscle
HC_Liver_P1_cleaned.bed      capped    1,1,1    HC_Liver_P1
E15_Brain_P1_cleaned.bed     capped    1,1,1    E15_Brain_P1
E8_Ileum_cleaned.bed         capped    1,1,1    E8_Ileum
E15_Kidney_P1_cleaned.bed    capped    1,1,1    E15_Kidney_P1
E8_Skin_cleaned.bed          capped    1,1,1    E8_Skin
E15_Lung_P1_cleaned.bed      capped    1,1,1    E15_Lung_P1
E15_Lung_P2_cleaned.bed      capped    1,1,1    E15_Lung_P2
HC_Ileum_P1_cleaned.bed      capped    1,1,1    HC_Ileum_P1
HC_Brain_P2_cleaned.bed      capped    1,1,1    HC_Brain_P2
E15_Kidney_P2_cleaned.bed    capped    1,1,1    E15_Kidney_P2
E15_Muscle_P2_cleaned.bed    capped    1,1,1    E15_Muscle_P2
E15_Ileum_cleaned.bed        capped    1,1,1    E15_Ileum
HC_Kidney_cleaned.bed        capped    1,1,1    HC_Kidney
HC_Ileum_P2_cleaned.bed      capped    1,1,1    HC_Ileum_P2
E8_Kidney_cleaned.bed        capped    1,1,1    E8_Kidney
E15_Brain_P2_cleaned.bed     capped    1,1,1    E15_Brain_P2
```

The combined bed file has been converted to GTF format is using TAMA script tama_convert_bed_gtf_ensembl_no_cds.py.

```
tama_convert_bed_gtf_ensembl_no_cds chicken.bed chicken.gtf
```

The GTF file has been modified to add:

- The number of reads supporting each transcript annotation or 'read count'
- The results of lncRNA detection with feelnc
- Ensembl ids for already annotated genes and transcripts
- Annotations sample of origin

2.2 Read count computing

TAMA include a python script dedicated to read count computing. By running it after each TAMA collapse or TAMA merge, it's possible to keep track of read count through the cleaning/merging process. In this case, tama_read_support_levels.py has been run after pipeline's TAMA collapse (one bed file per read batch), pipeline's TAMA merge (one bed file per sample), and the last TAMA merge (one bed file for all samples).

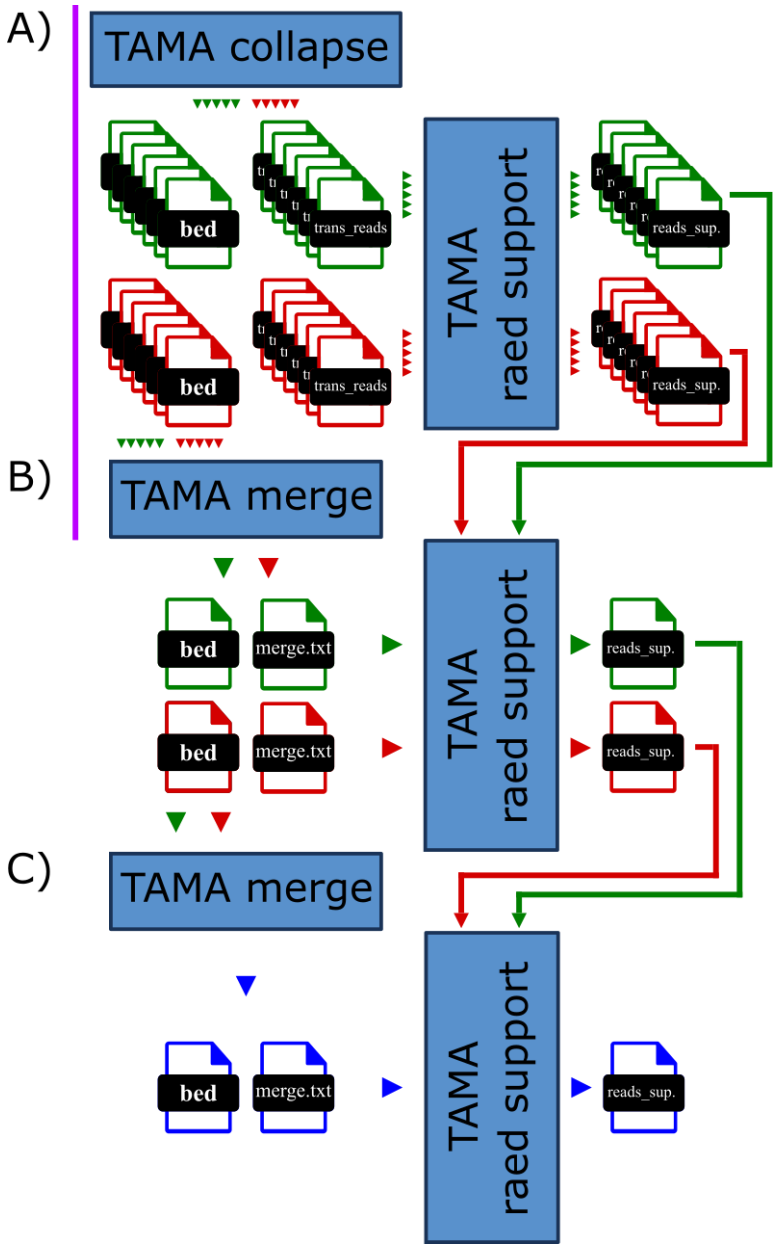

Figure 2: Read counting process – Purple bar indicates programs launch by the pipeline; others are launched manually. A) TAMA read support is launched on each trans_reads.bed files produced by TAMA collapse. B) TAMA read support is run on each sample merge.txt their associated chromosome read_support file. C) TAMA read support is run on the merge.txt of all samples with their associated sample read_support files

The first counting needs to be run on each `trans_read.bed` files produced by `TAMA collapse` (Fig 2. A). For each of them, a `read_support` file is created. If all `trans_read.bed` files are gathered in the same directory, the following `fish shell` loop will run the script on each of them:

```
for FILE in (find 09_GSTAMA_COLLAPSE/ -name '*_trans_read.bed')
    set FILE_SHORT (string replace -r '^.+/' '' $FILE)
    set ID (string replace "_trans_read.bed" "" $FILE_SHORT)
    set OUTFILE $ID"_"filelist.txt
    echo -e "$ID\t$FILE\ttrans_read" > $OUTFILE
    set CMD "tama_read_support_levels.py -f $OUTFILE -m no_merge -o $ID"
    echo $CMD
    eval $CMD
end
```

At each loop iteration, it will create a `filelist.tsv` input file composed of three tabulations separated columns (an ID, the `trans_read.bed` file and the file type) and run `tama_read_support_levels.py`.

Next, the generated `read_support` files are used to compute the read count at sample level (first `TAMA merge`). For each sample, a `filelist.tsv` file listing `read_support` files must be created. The first column is the ID, the second is the file path and the third one contains `read_support` as file type. The script can be started with following command:

```
for ID in (ls /*_read_support.txt|perl -pe 's/.+\///; s/\..+txt//g'|sort|uniq)
    set INPUT $ID"_input.tsv"

    for RS in (ls ../01-read_support_collapse/$ID*_read_support.txt)
        set ID2 (echo $RS|perl -pe 's/\.\.\/01-read_support_collapse\///g;
s/_read_support.txt/_collapsed.bed/g')
        echo -e "$ID2\t$RS\tread_support" >> $INPUT
    end

    set MERGE $ID"_merge.txt"
    set CMD "ln -sf ../../05-merging_files/02-merge.txt_by_samples/"$MERGE
    eval $CMD
    set CMD "tama_read_support_levels.py -f $INPUT -m $MERGE -o $ID"
    echo $CMD
    eval $CMD
end
```

Finally, the global read count can be computed using the `read_support` files generated for each sample. A fish shell loop can be used to generate the file list file. The `tama_read_support_levels.py is run using the resulting filelist file and the merge.txt generated during TAMA merge process.`

```
for RS in (ls ../02-read_support_merge1/*_read_support.txt)
    set ID (string replace -r '_read_support.txt' '' $RS|string replace -r '\.\.\/02-
read_support_merge1\/' '')
    echo -e "$ID\t$RS\tread_support" >> chicken_input.tsv
end

ln -s ../../04-merge_annotations/chicken_merge.txt

tama_read_support_levels.py -f chicken_input.tsv -m chicken_merge.txt -o chicken
```

2.3 Long Non Coding RNA detection

The program feelnc perform Long Non-Coding RNA (lncRNA) detection with an alignment-free method. By analysing nucleotide compositional bias of coding and non-coding regions of the genome, the program can infer the transcript coding status.

A nextflow pipeline (sguizard/nf-feelnc) has been developed, based on FAANG/analysis-TAGADA pipeline, to run all feelnc analysing steps. The program has been applied on the final annotation, with same genome and reference annotation used for isoseq annotations.

```
nextflow run sguizard/nf-feelnc \
-r dev \
--ref_annotation Gallus_gallus.GRCg6a.105.gtf \
--new_annotation chicken.gtf \
--genome Gallus_gallus.GRCg6a.dna.toplevel.fa \
-profile singularity,eddie
```

2.4 Add Ensembl IDs

The TAMA merge program offer the possibility to keep the ids from one source when two annotations are merged. To detect genes already annotated, ensembl annotation has been merge to isoseq annotation.

```
tama_format_gtf_to_bed12_ensembl.py \
Gallus_gallus.GRCg6a.105.gtf \
Gallus_gallus.GRCg6a.105.bed > format.log > format.err

echo -e "chicken_clean.bed\tcapped\t1,1,1\tisoseq" >> inputEnsembl.tsv
echo -e "Gallus_gallus.GRCg6a.105.bed\tcapped\t1,1,1\tensembl" >> inputEnsembl.tsv

tama_merge.py \
-f inputEnsembl.tsv \
-d merge_dup \
-a 2000 \
-m 10 \
-z 2000 \
-p chicken_ensembl \
-cds ensembl \
-s ensembl 1> merge.log 2> merge.err
```

2.5 Add information to GTF file

The first step has been to link read count, Ensembl IDs, feelnc biotype and samples origin to each isoseq annotation. The R script link_readCount_EnsIDs_sourceLine.R generates a modified bed files gathering the required information using:

- isoseq and Ensembl merged annotations (`chicken_ensembl.bed`)
- `trans_report.txt` file generated at isoseq and Ensembl annotations (`chicken_ensembl_trans_report.txt`)
- the read count file (`chicken_read_support.txt`)
- feelnc annotation (`novel.gtf`)

The compiled information has been incorporated to the isoseq GTF file with a simple R script.

```
library(tidyverse)

read_gff  <- function(file) {require(readr);read_tsv(file,col_names = c("sequence",
"source",  "feature","start",  "end",  "score",  "strand",  "phase",  "attributes"),
col_types = cols(sequence = "c", source = "c", feature = "c", start = "i", end =
"i", score = "c", strand = "c", phase = "c", attributes = "c"), comment = "#")}

gtf <-
    read_gff('chicken.gtf') %>%
    mutate(transcript_id = str_extract(attributes, 'G\\d+\\.\\d+'))

d <- read_tsv(
    'chicken_ensembl_all_info.bed',
    col_types = cols(
        'chrom'          = "c", 'chromStart'         = "i", 'chromEnd'    = "i",
        'name'           = "c", 'score'              = "c", 'strand'      = "c",
        'thickStart'     = "c", 'thickEnd'           = "c", 'itemRgb'     = "c",
        'blockCount'     = "i", 'blockSizes'         = "c", 'blockStarts' = "c",
        'tama_gene_id'   = "c", 'tama_transcript_id' = "c", 'ens_gene'    = "c",
        'ens_transcript' = "c", 'sources'            = "c", 'old_id'      = "c",
        'feelnc_biotype' = "c", 'trans_read_count'   = "i",
        'source_line'    = "c")) %>%
    select(ens_gene:source_line) %>%
    filter(sources %in% c('isoseq', 'ensembl,isoseq')) %>%
    rename(transcript_id = old_id)

left_join(gtf, d) %>%
    select(-transcript_id, -sources) %>%
    mutate(
        attributes       = str_replace(attributes, ';$', ''),
        ens_gene         = paste0('ens_gene "', ens_gene, '\"'),
        ens_transcript   = paste0('ens_transcript "', ens_transcript, '\"'),
        feelnc_biotype   = paste0('feelnc_biotype "', feelnc_biotype, '\"'),
        trans_read_count = paste0('trans_read_count "', trans_read_count, '\"'),
        source_line      = paste0('source_line "', source_line, '\"')) %>%
    unite(attributes, attributes:source_line, sep = '; ') %>%
    write_tsv(
        'Chicken_isoseq-annotation.gtf',
        col_names = FALSE,
        quote = 'none',
        escape = 'none')
```