GENE-SWitCH

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

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

Authors: Sébastien Guizard (UEDIN)

Workpackage: WP2

Version: 1.1

Protocol associated with Deliverable(s):	D2.1 D2.2
Submission date to FAANG:	16/12/2021

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

Table of figures

1 Summary

GENE-SWitCH aims at identifying functional elements located in the genomes of 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

2.1 Annotation – Isoseq Pipeline

An annotation has generated each of the 31 subreads set. The isoseq raw subreads processing, the genome mapping and the gene model annotation creation has been made using the isoseq nextflow pipeline.

First the pipeline generates 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, and trim it. The produces sequences are called Full Length Non Chimeric (FLNC). Before the clustering step, the sequence batches are merged using with Pacbio's `pbmerge` program. Next, the program `isoseq3 cluster` regroups similar sequences and create a consensus, called HIFI, for each cluster.

Before proceeding to the mapping, the not clustered FLNC and HIFI are merged using `pbmerge` and a fastq file is created using `samtools fastq`. The set of reads is aligned on the reference with the program `uLTRA`.

To accelerate the computations, the resulting alignment is split based on the chromosome with `bamtools split`. Each alignment is processed by `TAMA collapse`, for false positive and redundancy removing. To finish, the complete set of annotation bed files is merged in one unique bed file with `TAMA merge`.

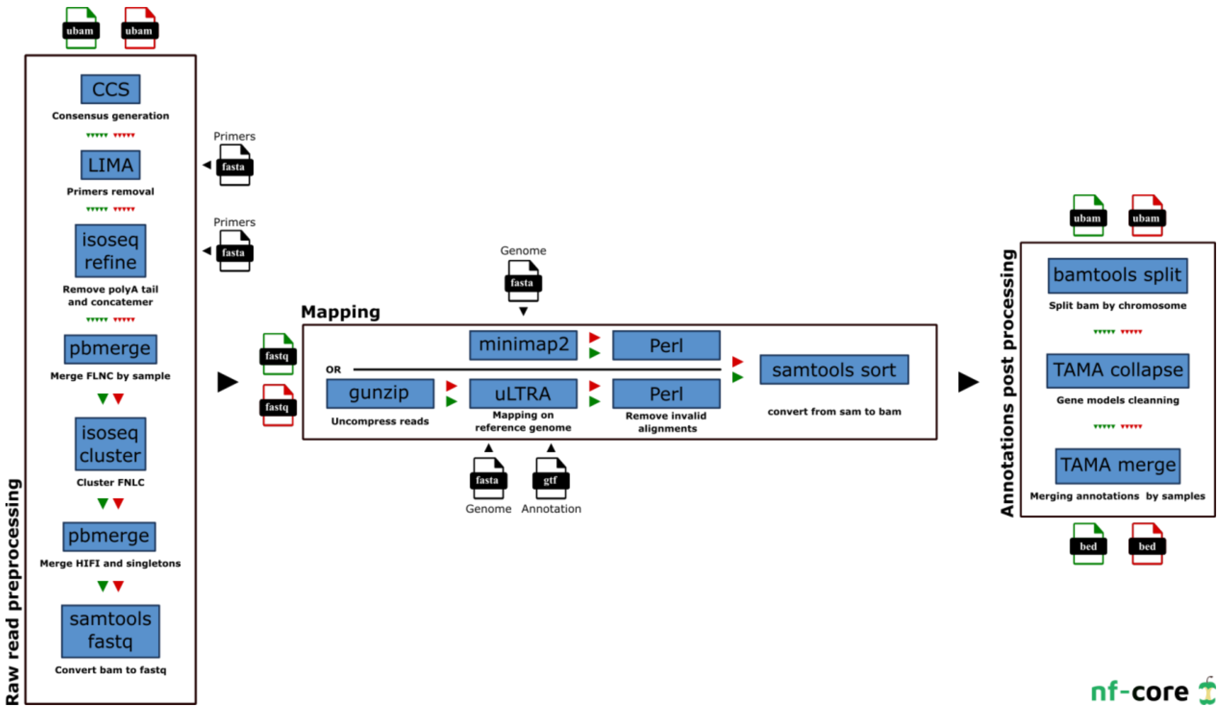

Figure 1: Isoseq nexflow pipeline

To run the pipeline, a data directory has been created and the following input files have been stored in it:

- Samples subreads (.bam)
- Pacbio index files (.bam.pbi)

- Genome (.fasta)
- Primer sequences (.fasta)
- Ensembl gene annotation (.gtf)

The genome used is Gallus Gallus 6 (GRCg6a) and its gene annotation from Ensembl release-104. The primers used are:

```
>5p
TGGATTGATATGTAATACGACTCACTATAG
>3p
CGCCTGAGA
```

The complete command line is:

```
nextflow run sguizard/isoseq \
-input data/ \
--primers data/primers.fasta
--fasta data/Gallus_gallus.GRCg6a.dna.toplevel.fasta \
--chunk 100 \
--require_polya \
--min_polya_length 20 \
--rq 0.99 \
--five_prime 2000 \
--splice_junction 10 \
--three_prime 2000 \
--capped \
--run_cluster \
--ultra \
--gtf data/Gallus_gallus.GRCg6a.104.gtf \
-profile singularity
```

2.2 Merging Annotations

The files 31 bed files obtained for the 21 pairs of tissues/development stages were merged together using `TAMA merge`.

```
tama_merge.py -f 31samples_filelist.tsv -d merge_dup -a 2000 -m 10 -z 2000 -p
31SamplesFiles 1> merge.log 2> merge.err
```

The “`31samples_filelist.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).

```
merged_m64036_210413_213334.subreads.bed    capped  1,1,1   E8Brain
merged_m64036_210319_151149.subreads.bed    capped  1,1,1   E8Ileum
merged_m64036_210320_212651.subreads.bed    capped  1,1,1   E8Kidney
merged_m64036_210315_182820.subreads.bed    capped  1,1,1   E8Liver
merged_m64036_210322_035446.subreads.bed    capped  1,1,1   E8Lung
merged_m64036_210412_151836.subreads.bed    capped  1,1,1   E8Muscle
merged_m64036_210415_040208.subreads.bed    capped  1,1,1   E8Skin
merged_m64036e_210713_144608.subreads.bed   capped  1,1,1   E15BrainP2
merged_m64036e_210719_150424.subreads.bed   capped  1,1,1   E15BrainP1
merged_m64036_210622_160059.subreads.bed    capped  1,1,1   E15Ileum
merged_m64036_210623_220011.subreads.bed    capped  1,1,1   E15KidneyP2
merged_m64036_210630_203419.subreads.bed    capped  1,1,1   E15KidneyP1
merged_m64036_210418_013328.subreads.bed    capped  1,1,1   E15Liver
merged_m64036_210702_024547.subreads.bed    capped  1,1,1   E15LungP1
merged_m64036_210625_041134.subreads.bed    capped  1,1,1   E15LungP2
merged_m64036_210626_102305.subreads.bed    capped  1,1,1   E15MuscleP2
merged_m64036_210703_085717.subreads.bed    capped  1,1,1   E15MuscleP1
merged_m64036e_210714_204516.subreads.bed   capped  1,1,1   E15SkinP2
merged_m64036e_210803_155018.subreads.bed   capped  1,1,1   E15SkinP1
merged_m64036e_210828_025255.subreads.bed   capped  1,1,1   HCBrainP1
merged_m64036e_210909_045748.subreads.bed   capped  1,1,1   HCBrainP2
merged_m64036e_210717_090846.subreads.bed   capped  1,1,1   HCIleumP2
merged_m64036e_210806_040105.subreads.bed   capped  1,1,1   HCIleumP1
merged_m64036e_210812_170130.subreads.bed   capped  1,1,1   HCKidney
merged_m64036e_210716_025706.subreads.bed   capped  1,1,1   HCLiverP2
merged_m64036e_210804_214905.subreads.bed   capped  1,1,1   HCLiverP1
merged_m64036e_210815_202444.subreads.bed   capped  1,1,1   HCLung
merged_m64036e_210907_224558.subreads.bed   capped  1,1,1   HCMuscleP2
merged_m64036e_210826_204115.subreads.bed   capped  1,1,1   HCMuscleP1
merged_m64036e_210829_090412.subreads.bed   capped  1,1,1   HCSkinP1
merged_m64036e_210910_110947.subreads.bed   capped  1,1,1   HCSkinP2
```

2.3 Convert Bed File To GTF File

The combined bed file has been converted is using TAMA script `tama_convert_bed_gtf_ensembl_no_cds.py`.

```
tama_convert_bed_gtf_ensembl_no_cds.py 31SamplesFiles.bed 31SamplesFiles.gtf
```

2.4 Add Annotations Confidence

The GTF file has been updated to include a confidence attribute. The two possible values for this attribute are "low" and "high". If an annotation is supported by only one FLNC, the confidence is set to "low". If an annotation is supported by at least 2 FLNC or one HIFI, the confidence is set to "high". This modification has been made with two homemade R scripts: get_low_high_confidence_annotations.R and add_confidence_annotation.R.

2.5 Add Annotations Read Count

The GTF file has been also modified to add the read count for each transcript. The read count of an annotation is the sum of FLNC supporting this annotation. If the annotation is supported by an HIFI, then the read count is incremented by the number of FLNC used to create this HIFI.

`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 chromosome per sample), pipeline's `TAMA merge` (one bed file per sample), and the last `TAMA merge` (one bed file for all samples).

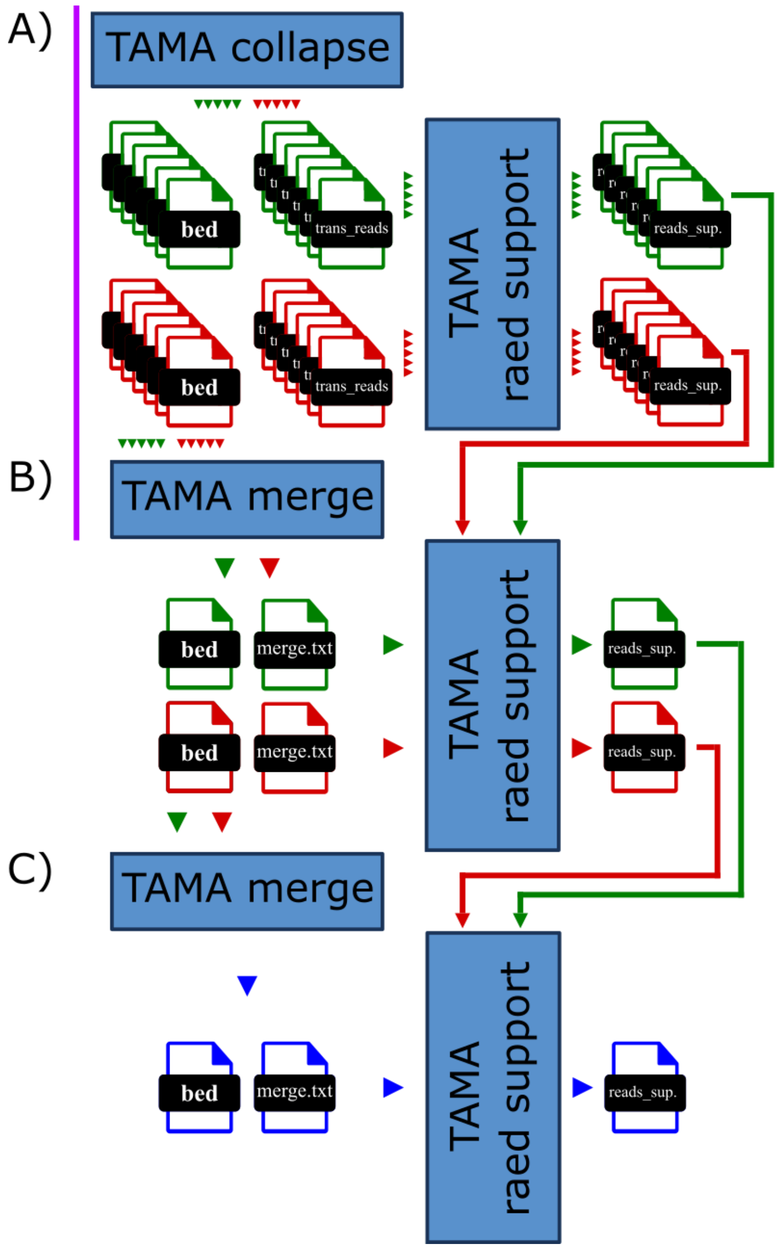

Figure 2: Read counting process – Purple bar indicates programs launch by the pipeline, others are launched manually. A) TAMA read support is launched on each trans_reads.bed files produced by TAMA collapse. B) TAMA read support is run on each sample merge.txt their associated chromosome read_support 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` 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 i in (ls *.bed)
    set FILE $i
    set ID (string replace "_tc_trans_read.bed" "" $FILE)
    set FILELIST $ID"_"filelist.txt
    echo -e "$ID\t$FILE\ttrans_read" > $FILELIST
    set CMD "tama_read_support_levels.py -f $FILELIST -m no_merge -o $ID"
    echo $CMD
    eval $CMD
end
```

At each loop round, 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:

```
tama_read_support_levels.py \
-f <SAMPLE>_filelist.tsv \
-m <SAMPLE>_merge.txt \
-o <SAMPLE>
```

Finally, the global read count can be computed using the `read_support` files generated for each sample. The `filelist.tsv list` sample's `read_support` files:

```
E15BrainP1     E15BrainP1_read_support.txt     read_support
E15BrainP2     E15BrainP2_read_support.txt     read_support
E15Ileum       E15Ileum_read_support.txt       read_support
E15KidneyP1    E15KidneyP1_read_support.txt    read_support
E15KidneyP2    E15KidneyP2_read_support.txt    read_support
E15Liver       E15Liver_read_support.txt       read_support
E15LungP1      E15LungP1_read_support.txt      read_support
E15LungP2      E15LungP2_read_support.txt      read_support
E15MuscleP1    E15MuscleP1_read_support.txt    read_support
E15MuscleP2    E15MuscleP2_read_support.txt    read_support
E15SkinP1      E15SkinP1_read_support.txt      read_support
E15SkinP2      E15SkinP2_read_support.txt      read_support
E8Brain        E8Brain_read_support.txt        read_support
E8Ileum        E8Ileum_read_support.txt        read_support
E8Kidney       E8Kidney_read_support.txt       read_support
E8Liver        E8Liver_read_support.txt        read_support
E8Lung         E8Lung_read_support.txt         read_support
E8Muscle       E8Muscle_read_support.txt       read_support
E8Skin         E8Skin_read_support.txt         read_support
HCBrainP1      HCBrainP1_read_support.txt      read_support
HCBrainP2      HCBrainP2_read_support.txt      read_support
HCIleumP1      HCIleumP1_read_support.txt      read_support
HCIleumP2      HCIleumP2_read_support.txt      read_support
HCKidney       HCKidney_read_support.txt       read_support
HCLiverP1      HCLiverP1_read_support.txt      read_support
HCLiverP2      HCLiverP2_read_support.txt      read_support
HCLung         HCLung_read_support.txt         read_support
HCMuscleP1     HCMuscleP1_read_support.txt     read_support
HCMuscleP2     HCMuscleP2_read_support.txt     read_support
HCSkinP1       HCSkinP1_read_support.txt       read_support
HCSkinP2       HCSkinP2_read_support.txt       read_support
```

The last `tama_read_support_levels.py` is launched with this following command:

```
tama_read_support_levels.py  -f  input_filelist.tsv  -m  31SamplesFiles_merge.txt  -o
31SamplesFiles
```

The global read count must be corrected to include the number of FLNC for each HIFI. This information can be obtained by running `tama_read_support_levels.py` on the cluster report generated by isoseq3 cluster. When all `cluster_report.csv` files are gathered in a directory, the script can be run using the following `fish shell` loop:

```
for i in (ls *.csv)
    set FILE $i
    set ID (string replace ".cluster_report.csv" "" $FILE)
    set FILELIST $ID"_"filelist.txt
    echo -e "$ID\t$FILE\tcluster" > $FILELIST
    set CMD "tama_read_support_levels.py -f $FILELIST -m no_merge -o $ID"
    echo $CMD
    eval $CMD
end
```

The complete set of `read_support` is merged into one file before read_count correction.

```
head -n1 E8Brain_read_support.txt| \
perl -pe 's/\n$/\tsample/g' > 31SamplesFiles_cluster_read_support.txt

for i in (ls *read_support.txt)
    set FILE $i
    set SAMP (string replace "_read_support.txt" "" $FILE)
    set  CMD  "perl  -pe  '\$_  =  \"\"  if  /^merge/;  s/\n\$/\t$SAMP\n/g'  $FILE  >>
31SamplesFiles_cluster_read_support.txt"
    echo $CMD
    eval $CMD
end
```

The read count correction is made using a homemade R script: count_correction.R

2.6 Add Ensembl 105 Gene Ids

The Ensembl 105 gene annotation were merged to 21 timepoints annotations using TAMA merge. Ensembl gtf file has been converted into bed12 file using `tama_format_gtf_to_bed12_ensembl.py` script.

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

The annotations have been using TAMA merge:

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

The inputEnsembl.tsv file list the two files to merge:

```
31SamplesFiles.bedcapped1,1,1isoseq
Gallus_gallus.GRCg6a.105.bedno_cap1,1,1ensembl
```

The resulting bed file has been used to extract two lists:

- TAMA gene ids and their corresponding Ensembl gene ids
- TAMA transcripts ids and their corresponding Ensembl transcripts ids

Those lists have been gene rated using two R scripts:

- tama_gene_id_to_ensembl_gene_id.R
- tama_transcript_id_to_ensembl_transcript_id.R

Finally, the two generated lists were used to update the GFT file using another script: add_ensembl_ids_to_gtf.R