Intructions for extracting sequences from HTS target enrichment reads

Miao Sun ( *cactusresponsible[AT]gmail.com*)

July 16, 2019

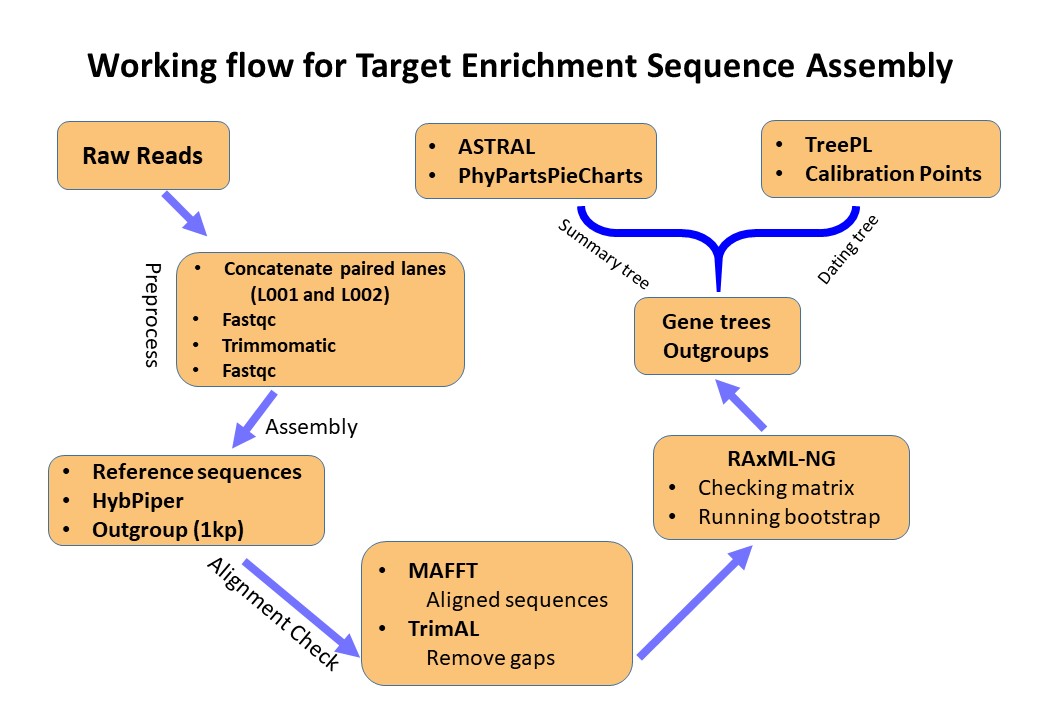
***This page is used to document steps of processing target enrichment reads from raw reads all the way down to phylogenetic tree reconstruction.***

*Most scripts used here are wrotten in bash/shell and R*

*I also made some assumptions that:*

* You are working on [HiPerGator](https://www.rc.ufl.edu/services/hipergator/) at Uiversity of Florida
* I applied the same rule to name our files and your file tree is the same as mine (see below)
* Using [Universal Probe Set for Targeted Sequencing of 353 Nuclear Genes](https://academic.oup.com/sysbio/article/68/4/594/5237557)

### Geeral workingflow



workingflow diagram

## Data

The data used in this instruction was generated by [RAPiD Genomics](www.rapid-genomics.com).

Within the data directory there is a SampleSheet **csv** file with the barcodes, filenames, and sample codes.

***Note that Plates1-4 were sequenced on two lanes (L001 and L002), so there are two sets of fastq files per sample.***

**Raw Data:**

* This data has been demultiplexed using Illuminas BCLtofastq. No quality trimming or processing has been done beyond demutiplexing.
* The adapters used are below, *“BCBCBCBC”* stands for the barcodes.
  + i7: GATCGGAAGAGCACACGTCTGAACTCCAGTCAC-BCBCBCBC-ATCTCGTATGCCGTCTTCTGCTTG
  + i5: AATGATACGGCGACCACCGAGATCTACAC-BCBCBCBC-ACACTCTTTCCCTACACGACGCTCTTCCGATCT

## Assembly methods

Currently, Three ways you can analyze high-throughput sequencing reads using target enrichment:

1. HybPiper

[Publication](https://bsapubs.onlinelibrary.wiley.com/doi/full/10.3732/apps.1600016)

[Code in github](https://github.com/mossmatters/HybPiper)

1. aTRAM

[Publication](https://journals.sagepub.com/doi/10.1177/1176934318774546)

[Code in github](https://github.com/moskalenko/aTRAM)

1. SECAPR

[Publication](https://peerj.com/articles/5175/)

[Code in github](https://github.com/AntonelliLab/seqcap_processor)

*In this tutorial, I am only focus on* [*HybPiper*](https://github.com/mossmatters/HybPiper)

### HybPiper

**Preprocess:**

1. Concatenate all lanes (L001 and L002; only if you have them on separate plates!)  
   Example,

`cat RAPiD-Genomics\_F076\_UFL\_###\_P003\_WD02\_i5-503\_i7-72\_S22\_L001\_R1\_001.fastq.gz RAPiD-Genomics\_F076\_UFL\_###\_P003\_WD02\_i5-503\_i7-72\_S60\_L002\_R1\_001.fastq.gz > P003\_WD02\_72\_R1.fastq.gz`

or run in a batch manner:

`bash fastq\_lane\_cat.sh sample\_ID\_file Seq\_ID\_table`

Example,

`bash fastq\_lane\_cat.sh Evgeny\_13.txt UFL\_394803\_SampleSheet.csv`

This bash script will take two input files: one is sample ID file, and the other is sequence ID table. The formact and content of each file is as shown below:

Example,  
[cactus]$ head -6 XXX\_88.txt CPG00213 CPG00216

[cactus]$ head -6 UFL\_XXX\_SampleSheet\_XXX86.csv RG\_Sample\_Code,Customer\_Code,i5\_Barcode\_Seq,i7\_Barcode\_Seq,Sequence\_Name,Sequencing\_Cycle

UFL\_394803\_P002\_WG08,D\_4566,TAAGATTA,TTCACGCA,RAPiD-Genomics\_F076\_UFL\_394803\_P002\_WG08\_i5-506\_i7-68\_S171\_L001\_R1\_001.fastq.gz,2x150 …

UFL\_394803\_P002\_WG12,D\_4571,TAAGATTA,CGCATACA,RAPiD-Genomics\_F076\_UFL\_394803\_P002\_WG12\_i5-506\_i7-42\_S175\_L001\_R1\_001.fastq.gz,2x150

1. **fastqc** to quick check the quality; and later on can be used for comparison after trim and clean.

* scripts needed:  
  fastqc.sh check\_result.sh mean.R

Example,

module load ufrc fastqc  
srundev -t time  
fastqc \*.gz -o FastQC\_result

For slurm job scripts see:

**fastqc.sbatch** [./Scripts/fastqc/fastqc.sbatch]

* after runing *fastqc.sh*, it will put fastqc results into a folder called *FastQC\_result*;
* Copy scripts *check\_result.sh*, and *mean.R*, into *FastQC\_result*, then excute the bash script, it will generate a summary table *Illumina\_FastQC\_report.csv* for reads quality. Other details see folder *unzip\_file*.
* **note:**
  + *check\_result.sh* and *mean.R* have to work together, you have to put them under the same directory
  + the R script is automatically invoked, you don’t need to modify anything.
  + here is the example cmd (assuming you are in *FastQC\_result* folder) cp /path/to/scripts/check\_result.sh /path/to/scripts/mean.R .
  + bash check\_result.sh

1. Trim and clean reads using Trimmomatic, and preapre for next step — Hybpiper.

* scripts needed:
* Trimmomatic.sbatch
* *If you have a few sample you can just run bash Trimmomatic.sh on dev node, which is not necessary to schedule a slurm job.*
* For large number of samples, submission to SLURM in HPC is required.
* run: sbatch Trimmomatic.sbatch  
  *modify the recources requested to suit for your samples*

**Sequence Assembly using Hybpiper:**

1. run hybpiper  
   nohup bash HybPiper\_summary.sh Lucas27 &
2. if want introns run intron script on accession folders out putted from previous step
3. to retrieve the supercontig sequences from the above run put them all in one place (so mv P\*W\* seq\_dir):  
   module load python  
   python HybPiper/retrieve\_sequences.py baits1.fasta seq\_dir dna  
   *just exons use DNA, if you run intronerate use supercontig*

**Alignment**

1. run mafft script on individual gene
2. Phyx — rename sequence names
3. TrimAL — remove gaps in the alignment

**Outgroup**  
*skip this step if you already have outgroup data from Target Enrichment or don’t neeed 1kp data*

Beside the data generated from Target Enrichment of 353 universial probe sets, I also included some species with [1kP transcriptome data](http://www.onekp.com/public_data.html) as Outgroups.

Given I have no pre-knowledge, of how 1kP transcriptome data will be compatible with alignments of 353 nuclear genes, so I used reference sequences of 353 nuclear genes to assemble 1kP data of those outgroup species in two ways. Then I aligned them, comapred and select one of best, or I choose the consensus sequence using Geneious.

* [raw reads](http://www.onekp.com/public_read_data.html)
* [SOAPdenovo Assembly](http://www.onekp.com/public_data.html)

**RAxML-NG**  
10. run raxml

Three scripts used (./Scripts/raxml-ng/):

* raxmlng\_laucher.sh
* raxml\_NG\_check.sbatch
* raxml\_NG\_model.sbatch

These three scripts will run sequentially. By providing a list with all the genera, raxmlng\_laucher.sh will go through each genus folder, creacte a “raxml” folder (where the raxml tree recontruction will happen), and looking for how many gene alignments were assembled for each genus; these numbers will be insert as a array job parameter for the first raxml script raxml\_NG\_check.sbatch.

For each alignment, raxml\_NG\_check.sbatch will run raxml-ng “–parse” checking, for purpose that:

* MSA sanity check (see [Tutorial](https://github.com/amkozlov/raxml-ng/wiki/Tutorial))
* Compress alignment patterns as RAxML Binary Alignment (.rba file)  
  *It will laoding faster for raxml, comapared to FASTA or PHYLIP (see* [*Tutorial*](https://github.com/amkozlov/raxml-ng/wiki/Tutorial) *)*
* Getting estimated computation recources (e.g., Model, memory, and optimal number of CPUs/threads)

*If the script detected that one alignment required larger mem (default is* ***1g*** *) or more threads (default is* ***1*** *), then it will lauch the third Script raxml\_NG\_model.sbatch, otherwise it will complete the job using current script with configurarion of default computation recource requirest*

If the third script raxml\_NG\_model.sbatch is launched, it will submit a new independent slurm job, with updated computation recources request based on the “–parse” results from raxml\_NG\_check.sbatch script.

**Acknowledgements**

[Rebecca L. Stubbs](http://www.rebeccalstubbs.com/) & [Johanna R Jantzen](https://www.johannajantzen.com/about-me.html) *sharing her modified scripts for runing* [*PhyPartsPieCharts*](https://github.com/mossmatters/MJPythonNotebooks/blob/master/BakingPieCharts.ipynb)

[Andre A Naranjo](https://scholar.google.com/citations?user=iabk-6MAAAAJ&hl=en)  
*sharing slurm job scripts for running Hybpiper*

[Matt Gitzendanner](https://magitz.biology.ufl.edu/home/)  
*helping with the slurm job schedule and raxml-ng MPI issues and other miscellaneous trouble shooting*