Written by Hao Yuan
Shanghai Ocean University, China.
Created on May 2016

Last modified - Aug 2016 by Hao Yuan
New Traits:
(1) Modified I/O in Remove Replicates & Parse Reads to Genes(server)
(2) updated trinity version 2.2.0 and modified option "-max_memory 10G" as "-max_memory 1G"
(3) Generate consensus sequences of selected taxon
(4) Parallelization of Unzip Raw Data(local), retention of unzipped data

Last modified - Oct 2016 by Roa-Varón –
New traits:
Annotated version to clarify the process for the Gadiformes Fish Project (ARV)

____

Caution:
1. Back up original data before analysis!!!!
2. Make a sample list delimited by space
3. It's unnecessary to copy scripts to path you submits the job, for all scripts executed on cluster are located in $HOME/bin/

##
1) Unzip Raw Data(local)

Function:
Unzip all .gz files and gather in a folder

Input:
(1) a folder contains all .gz files(gz),
(2) gunzip_Files.pl,
(3) output dir

Output:
unziped files(fastq)

Usage example:
```
$ ./gunzip_Files.pl -dir=dir -outdir=outdir
```

##
2) Merge Raw Data in 2 Lanes(local)

Function:
Merge the data on lane1 and lane2 together

Input:
unziped fastq files(.fastq)

Output:
merged files(.fastq)

Usage example:

```
$   cd *_unzip

$   (for i in *_L001_R1_001.fastq; do cat ${i%_L001_R1_001.fastq}_L001_R1_001.fastq
${i%_L001_R1_001.fastq}_L002_R1_001.fastq > ${i%_L001_R1_001.fastq}_R1.fastq; done)

$   (for i in *_L001_R2_001.fastq; do cat ${i%_L001_R2_001.fastq}_L001_R2_001.fastq
${i%_L001_R2_001.fastq}_L002_R2_001.fastq > ${i%_L001_R2_001.fastq}_R2.fastq; done)

$  rm -f *_001.fastq
```

###
3) Merge Reads (local)

Function:
Merge the multiple raw data from single species

Input:
unzipped raw data

Output:
merged raw data

Usage example:

```
$./merge_reads.pl -dir=dir
```

###
4) Remove Adapter & Low-Quality Reads (local)

Function:
Trim the adapter and low-quality reads in fastq file

Input:
merged fastq files(.fastq), trim_galore, cutadapt

Output:
trimmed files(.fq) in *_renamed folder

Usage example:

```
$    (for i in *_R1.fastq; do trim_galore -a AGATCGGAAGAGCACACGTCTGAACTCCAGTCAC -a2
AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT --paired ${i%_R1.fastq}_R1.fastq
${i%_R1.fastq}_R2.fastq; done) >& trim.log.txt

$ rm -f *_trimming_report.txt

$ rm -f *.fastq
```

———————————————————————————————

Rename *_val_1.fq/*_val_2.fq as *_R1.fq/*_R2.fq

———————————————————————————————

###

5) Remove Replicates & Parse Reads to Genes(server)

Function:
Remove replicates in trimmed file, and parse them into corresponding genes

Input:
baits sequences, all *.fq files

Output:
gene files containing corresponding reads(.fq)

Caution:
(1) amount of fork should be around 8-10 or rmrep.pl would stop for filled RAM
(2) amount of split files always 48, because bands.pl always parallelized in 48 processed

Usage example:

*.sh format

```
#########################################

#!/bin/bash

#PBS -l nodes=4:ppn=12
#PBS -l walltime=240:00:00
#PBS -N preads1
#PBS -q small

cd /where/your/data/is

rmrep.pl -taxalist="sample1 sample2 sample3"
```

```
bandp.pl -query="query_species" -subject="sample1 sample2 sample3" > preads.1.log.txt

exit 0

####################################################################################
```

6) Preliminary Assembly(server)

Function: Assemble the reads to short contigs

Input:
(1) samplename_results (preads result) folder contains gene files
(2) Trinity

Output:
folders containing all assembled file for each taxon named by "*.Trinity.fasta"

Usage example:

1. run trinity
*.sh format

```
###########################################
#!/bin/bash

#PBS -l nodes=4:ppn=12
#PBS -l walltime=240:00:00
#PBS -N runtrinity15
#PBS -q small

cd /where/your/data/is

runtrinity.pl -species="sample1 sample2 sample3" -outdir=dir > runtrinity15.log

exit 0

#############################################
```

2. Compress the outdir and download

3. Delete the remaining *_results folders

```
$ rm -rf *_results
```

6.1 Fetch Trinity Output

usage example:

```
$ ./mv_trinity.pl -species="sample 1 sample 2 sample 3" -outdir=geneious > mv_trinity.log
```

##
7) Assign Files with 1 contig and 2 contigs to Different Folders (local) - ARV: Pre-Geneious

Function: Identify files with one or more contigs and put file with one contig in a folder, two or more contigs in another folder

Input:
Output file in step 1, idcontig.pl

Output:
folders named by *_geneious1, *_geneious2

Usage example:
```
$ ./idcontig.pl -species="sample1 sample2 sample3"
```

##
8) Geneious(local)

Function: Assemble short contigs to long contigs

Usage:
1. Import the gene files in *_geneious2
2. De novo assemble on batch
3. Batch export to folders named by sample name which are placed under the same folder containing all geneious output

##
9) Unwrap & Cat & Merge (local)

Function 1: cat the file with same gene in geneious output first
Function 2: unwrap the sequence in the file with only one contig
Function 3: merge the processed files in (1) and (2) together according to the sample ID

Input:
geneious2best.pl, species ID, folder1: speciesID_geneious1 e.g index123_geneious1, folder2: speciesID_geneious2 e.g index123_geneious2
Output:
a folder named "merge" including all the assembled contigs for each sample

Usage example:
move all *_geneious1 and *_geneious2 to the same folder
```
$  ./geneious2getbest.pl -species="sample1 sample2 sample3"
```

ARV: In order to run the scripts, it needs Python and BioPhyton dependencies

##
10) Predict Frames of Query (local)

Function: predict the first codon of query

Input:
(1) a bait sequence fasta(*.fas)
(2) a fasta file containing amino acid sequences with ENSMBL geneIDs for headers.(*.pep.all.fa)
(3) the first column of a onehitCDSmarker file(*.onehitCDSmarkers.column1.txt)
(4) predictFrames, ensmbl2frames.py

Caution:
(1) MAKE SURE ALL THE FORMAT OF INPUT FILES IS THE SAME AS SAMPLE FILES!!!!!
(2) if you got errors, try trim_redundant.pl on *.pep.all.fa and check whether a header in first line of *.onehitCDSmarkers.column1.txt, delete if you got one

Output files:
(1) Updated bait fasta file(*.frames.fas)
(2) Comma separated value file for the stop results containing the sequences translated(*.frameResults.alns.csv)
(3) Comma separated value file for the alignment results containing the scores of the alignments and the hypothetical maximum score(*.frameResults.stops.csv)
(4) List of query where each frame contains inappropriate stop codons(*.strangeBaits.txt)

Usage example: (cds=coding DNA sequence)
(1) 3 descriptions in query id(e.g. 1.100321124.100320916)
```
$./predictFrames -b=baitSeq.fas -r=Gallus_gallus.WASHUC2.70.pep.all.fa  -
cds=Gallus_gallus.onehitCDSmarkers.column1.txt -fas=Python_molorus.frames.fas  -v=True -
aln=Python_molorus.frameResults.alns.csv -stop=Python_molorus.frameResults.stops.csv >
Python_molorus.strangeBaits.txt
```

or

(2) 4 descriptions in query id(e.g. Danio_rerio.1.10018393.10018273)
```
$./predictFrames  -ex=1 -b=Oreochromis_niloticus.fas -r=fourReference.pep.all.fa -
cds=Oreochromis_niloticus.onehitCDSmarkers.column1.txt -v=True -
aln=Oreochromis_niloticus.frameResults.alns.csv -fas=Oreochromis_niloticus.frames.fas -
stop=Oreochromis_niloticusframeResults.stops.csv >Oreochromis_niloticus.strangeBaits.txt
```

ARV:
Input files:
(1) a bait sequence fasta (*.fas) = Gadus_morhua.fas

(2) a fasta file containing amino acid sequences with ENSMBL geneIDs for headers.(*.pep.all.fa) = fourReference.pep.all.fa
(3) the first column of a onehitCDSmarker file(*.onehitCDSmarkers.column1.txt) = Gadus_morhua.onehitCDSmarkers.column1.txt
(4) predictFrames, ensmbl2frames.py

Output files:
(1) Updated bait fasta file(*.frames.fas) = Gadus_morhua.fas
(2) Comma separated value file for the stop results containing the sequences translated(*.frameResults.alns.csv) = Gadus_morhua.framneResultrs.alns.csv
(3) Comma separated value file for the alignment results containing the scores of the alignments and the hypothetical maximum score(*.frameResults.stops.csv) = Gadus_morhuaframeResults.stops.csv
(4) list of query where each frame contains inappropriate stop codons(*.strangeBaits.txt) = Gadus_morhua.strangeBaits.txt

Usage example:

```
./predictFrames  -ex=1 -b=Gadus_morhua.fas -r=fourReference.pep.all.fa -
cds=Gadus_morhua.onehitCDSmarkers.column1.txt -v=True -
aln=Gadus_morhua.frameResults.alns.csv -fas=Gadus_morhua.frames.fas -
stop=Gadus_morhuaframeResults.stops.csv >Gadus_morhua.strangeBaits.txt
```

##
11) Trim stop codon in query(local)

Function1: Trim stop codon in query
Function2: make every query start from first codon

Input:
(1) *.frames.fas
(2) outfile name
(3) trim_stop_codon.pl
(4) translate.pm

Output:
(1) updated *.frames.fas

Usage example:
```
./trim_stop_codon.pl -query_frames=*.frames.fas -outfile=out
```

ARV:
Input files: Gadus_morhua.frames.fas; translate.pm; trim_stop_codon.pl; trim_stop.job
Outputfiles: "out" smaller file after trimming

##
12) Retrieve Best Sequence from Genes of each Sample(server)

Function1: Retrieve best non-intron inserted sequence from genes of each sample
Function2: Detect intron, correct reading frames

Input:
(1) Output from Unwrap&Cat&Merge,
(2) *.frames.fas, BLOSUM80.bla

Output:
(1) folders named after *.resultnf(non-flankings)
(2) *.resultf(flankings)

Usage example:

(1) make dirs for *.frames.fas(query), Output from Unwrap&Cat&Merge(subject) and outfile(result)

```
   $ mkdir query
   $ mkdir subject
   $ mkdir result
```

(2) run getbest.pl

*.sh format
##

```
#!/bin/bash

#PBS -l nodes=4:ppn=12
#PBS -l walltime=240:00:00
#PBS -N getbest1
#PBS -q small

cd /where/your/folders/are

getbest.pl -query=query_species -subject="sub1 sub2 sub3" -matrix=BLOSUM80.bla

exit 0
```

###
12) Retrieve Best Sequence from Genes of each Sample(server) - ARV version

Function1: Retrieve best non-intron inserted sequence from genes of each sample
Function2: Detect intron, correct reading frames

(1) make dirs for *.frames.fas (query), Output from Unwrap&Cat&Merge (subject), outfile (result), all the scripts (script)

```
   $ mkdir query
   $ mkdir subject
   $ mkdir result
   $ mkdir script
```

   Each folder contains:
   query = Gadus_morhua.frames.fas (trimmed file generated in step 11: trim stop codon)
   subject = results from step (9) Unwrap & Cat & Merge
   scripts = BLOSUM80.bla ; dna2aa.pm ; getbest.pl ; score_matrix.pm ; smithwaterman.pm ; getbest.job (to submit job via Hydra)
   result = outfile Gadus_morhua.resultf & Gadus_morhua.resultnf)

qsub job example generated on https://hydra-3.si.edu/tools/QSubGen/

```
# /bin/sh
# ----------------Parameters---------------------- #
#$  -S /bin/sh
#$ -q mThC.q
#$ -cwd
#$ -j y
#$ -N getbest_test
#$ -o getbest_test.log
#$ -m bea
#$ -M aroavaron@vims.edu
#
# ----------------Modules------------------------- #
module load bioinformatics/bioperl/1.6.924
#
# ----------------Your Commands------------------- #
#
echo + `date` job $JOB_NAME started in $QUEUE with jobID=$JOB_ID on $HOSTNAME
#
./getbest.pl -query=Gadus_morhua -subject="27 29 61" -matrix=BLOSUM80.bla
exit 0
#
echo = `date` job $JOB_NAME done
```

##

13) Remove Paralogs in Non-Flanking Files(server)

Function1: remove paralogs

Input
*.resultnf folder from last step, database(under ~/thirdstore/lichenhong/fishbaits/)

Output:
reblasted *.resultnf, *.log

Usage example:

(1) make dirs for *.resultnf folder from last step(query), outfile and *.log(genebin) and blastn output(blastout)

```
   mkdir query
   mkdir genebin
   mkdir blastout
```

(2) run reblast.pl
*.sh format

```
##########################################

#!/bin/bash

#PBS -l nodes=4:ppn=12
#PBS -l walltime=240:00:00
#PBS -N reblast1
#PBS -q small

cd /where/your/folders/are

reblast.pl -query=wildcard part of *.resultnf -subject=species_name >reblast.log

exit 0

###############################################################################
```

13) Remove Paralogs in Non-Flanking Files (server) - ARV version

Function1: remove paralogs

Input:
*.resultnf folder from last step, database(from ~/thirdstore/lichenhong/fishbaits/)

Output:

(1) reblasted *.resultnf, *.log

Usage example:

(1) make dirs for *.resultnf folder from last step(query), outfile and *.log(genebin) and blastn output(blastout)

```
    mkdir blastout (output files)
    mkdir Gadus_morhua.db (Gadus_morhua.db.nhr; Gadus_morhua.db.nin;
Gadus_morhua.db.nsq)
    mkdir genebin (empty)
    mkdir query (*.resultnf - Original file has more than 14.000 files. So, it was divided in 10
folders e.g. Gadus_morua1.resultnf, Gadus_morua1.resultnf, etc)
    mkdir subject Gadus_morhua.genome.fas
    scripts : reblast.pl; reblast1.job....depending of the amount of subdivisions done in the query
```

(2) run reblast.pl

*.job format

```
##############################################

# /bin/sh
# ----------------Parameters---------------------- #
#$  -S /bin/sh
#$ -q mThC.q
#$ -l mres=6G,h_data=6G,h_vmem=6G
#$ -pe mthread 24-64
#$ -cwd
#$ -j y
#$ -N reblast1
#$ -o reblast1.log
#$ -m bea
#$ -M aroavaron@vims.edu
#
# ----------------Modules------------------------- #
module load bioinformatics/bioperl/1.6.924
module load bioinformatics/blast
#
# ----------------Your Commands------------------- #
#
echo + `date` job $JOB_NAME started in $QUEUE with jobID=$JOB_ID on $HOSTNAME
#
./reblast.pl -fork=$NSLOTS -query=Gadus_morhua1 -subject=Gadus_morhua >reblast1.log
exit 0
#
```

echo = `date` job $JOB_NAME done

##
14) Merge intron inserted sequences and select best sequences(local)

Function 1: merge intron sequences after reblast
Function 2: substitute intron sequences with pre-selected best non-intron inserted sequences if they got higher score

Input:
(1) folder of flank and nonflank sequences (after removing paralogs in no flanking regions)
(2) Scripts: merge_intron.pl translate.pm

Output:
1) sequences without flanking seqs, intron inserted seqs merged
2) sequences with flanking seqs, intron inserted seqs merged
3) sequences with flanking seqs, intron inserted seqs unmerged
4) sequence info
5) aa sequences

Usage example:
(1) verbose output is needed including 3) and 4)
```
./merge_intron.pl -flank=*.resultf -non_flank=*.resultnf -query=query_species -v
```

(2) verbose output is unnecessary
```
./merge_intron.pl -flank=*.resultf -non_flank=*.resultnf -query=query_species
```

ARV:
```
.merge_intron.pl -flank=Gadus_morhua.resultf -non_flank=Gadus_morhua.resultnf -
query=Gadus_morhua
```

##
15) Align AA sequences in mafft (local)

Function:
align AA sequences in batch

Input:
(1) dir for unaligned AA seq,
(2) mafft_AA.pl,
(3) mafft

Output:
(1) dir for aligned AA seq

Usage example:

```
$ ./mafft_AA.pl -dir=dir
```

ARV: (note nt = nucleotides)
(1) Input files: dir of unaligned nt seq ("aa_non_flank_out" from previous step);
(2) Script: mafft_AA.pl & *.job file to submit job in Hydra

Output:
(1) dir for aligned AA seq   "aa_non_flank_out_aligned"

Usage example:

```
$ ./mafft_AA.pl -dir=aa_non_flank_out
```

*.job example

```
# /bin/sh
# ----------------Parameters---------------------- #
#$  -S /bin/sh
#$ -q mThC.q
#$ -l mres=6G,h_data=6G,h_vmem=6G
#$ -pe mthread 24-64
#$ -cwd
#$ -j y
#$ -N align_aa
#$ -o align_aa.log
#$ -m bea
#$ -M aroavaron@vims.edu
#
# ----------------Modules------------------------- #
module load bioinformatics/bioperl/1.6.924
module load bioinformatics/mafft
#
# ----------------Your Commands------------------- #
#
echo + `date` job $JOB_NAME started in $QUEUE with jobID=$JOB_ID on $HOSTNAME
echo + NSLOTS = $NSLOTS
#
./mafft_AA.pl -dir=aa_non_flank_out
#
echo = `date` job $JOB_NAME done
```

##
16) Aligned AA 2 Aligned DNA (local)

Function: translate aligned aa seq back to aligned dna seq

Input:
(1) dir of unaligned nt seq,
(2) aligned aa seq
(3) dir name of output folder,
(4) aa2dna_aln.pl

Output:
aligned dna seq

Usage example:
```
$ ./aa2dna_aln.pl -dir_nt=dir_nt -dir_aa=dir_aa -outdir=outdir
```

ARV: (note nt = non-translated)
Function 1: translate aligned aa seq back to aligned dna seq

Input:
(1) dir unaligned nt seq (non_flank_out - step 14)
(2) dir aligned aa seq (aa_non_flank_out_aligned - step 15)
(3) dir name of output folder - aligned_DNA
(4) script: aa2dna_aln.pl

Output: aligned dna seq

Usage example:
```
$ ./aa2dna_aln.pl -dir_nt=dir_nt -dir_aa=dir_aa -outdir=aligned_DNA
```

##
17) Generate Consensus Seqs

Function: generate consensus cds sequence

Input:
(1) dir of aligned nt cds seqs
(2) taxon for consensus
(3) name of output file(.fa or .fasta)
(4) consensus_select.pl

Output:
consensus seqs(.fa or .fasta)

Usage example:
```
$ ./consensus_select.pl -dir=dir -taxon="taxon1 taxon2 taxon3" -outfile=consensus.fa
```

ARV
Function: generate consensus cds sequence (cds=coding DNA sequence)

Input:
(1) dir of aligned nt cds seqs results from step 16
(2) taxon for consensus
(3) name of output file (.fa or .fasta),
(4) script: consensus_select.pl

Output:
(1) consensus seqs (.fa or .fasta)

Usage example:

```
$ ./consensus_select.pl -dir=dir -taxon="taxon1 taxon2 taxon3" -outfile=consensus.fa
```