Manual ClusterAlign v0.14

Warren Emmett

Any questions, comments and bugs please report to me at:
warren.emmett@cbm.fvg.it

1. What is ClusterAlign and what can it do for me?

ClusterAlign is an application that simplifies the use of read aligners on an SGE cluster environment. While allowing you to build a custom pipeline for the analysis of your next-generation sequencing data. If you have access to a cluster it will allow you to perform alignments using both modern and old read aligners in the shortest amount of time.

2. How can I use it?

ClusterAlign is composed of three components.

1. The ClusterAlign python script : cluster_align_v15.py
2. A general configuration file: Cluster_info.config
3. A configuration file for each aligner: Aligner.config (for Bowtie it will be Bowtie.config)

General Configuration file

This file contains all the information you will need to insert only once per cluster, largely concerning where is the best location for clusteralign to write both on the master server and on each cluster node.

The variables to change are:

# specify a temporary directory where data can be on the master server
master_tmp_dir= /mnt/lustre_0/data/bioinfo/SEQC/Analysis/warren/tmp

# direcotry on the cluster node where temporary information can be stored
tmp_node_dir = /scratch/tmp/read_data

NOTE:
1. Lines starting with # will be ignored by the aligner.
2. It is essential to keep the same line structure: variable = value. DO NOT REMOVE = sign!!!

Aligner Configuration file

The aligner configuration file provides ClusterAlign with all the information necessary to run this read aligner, what reads you wish to align, what reference to use and what output directory to use.

Input files are directories

This section includes arell the information regarding read, reference and output files. The variables to change are:

*#Read files are specified by **read_file1 and read_file2, specify "none" in read_file2 is it is a single-end analysis.***

read_file1 =/mnt/lustre_0/data/bioinfo/Zebrafish/s_8_1_sequence.txt
read_file2 =/mnt/lustre_0/data/bioinfo/Zebrafish/s_8_2_sequence.txt

#Input reference location here, this must be in the exact form expected by the aligner, in this case it is the prefix of the file used by bowtie.
ref_file1 = /mnt/lustre_0/data/bioinfo/Zebrafish/genome_Z8/Zv8_genome

 # The second reference file is only required when you wish to use two reads aligners.
ref_file2 = none

 # This directory specifies where you want the output stored
output_directory = /mnt/lustre_0/data/bioinfo/Zebrafish/BWA_sing

Advanced: Directories as input

In order to streamline the sequencing of large numbers of lanes to the same or different references a single directory can be analysed rather than specifieng individual files. This requires further parameter changes.

 # This variable specifies the directory containing all reads to be aligned. If this value is "none" or not present this analysis type is disabled.
read_directory = none

or
read_directory = /mnt/lustre_0/data/bioinfo/Zebrafish/s_12/

The remaining parameters concern identifying the read files within this directory. This is done by looking at the prefix (machine labeled lanes often start with "s_") and the the suffix (eg - .fastq/.fq/.txt) this must be provided to avoid any unwanted files being accepted.

Furthermore, if paired end mode is selected you need to tell the program how to distinguish between 2 read pairs and the other reads, Example below:

The read pair : **s_1_1_seq.fastq, s_1_2_seq.fastq** are similar upto the first 3 letters "s_1" which denotes the lane. This means the prefix is s_1 and is therefore the first 3 letters.

#for read files in the direcory, only select files with the following prefix eg. (s_) s_1_sequence.FASTQ
read_prefix = s_

#for read files in the direcory, only select files with the following suffix eg. (FASTQ) s_1_sequence.FASTQ
read_suffix= txt

 # prefix for paired reads - this means that the aligner will group reads that start with the same X letters
 # example
 # s_1_1_seq.fastq s_1_2_seq.fastq are a read pair and they are similar upto the first 3 letters "s_1"
#the prefix is s_1
 # it is the first 3 letters
paired_prefix_length = 3

Aligner Arguments

The most important part of the config file regards how you setup your aligner parameters. ClusterAlign functions by allowing you to enter and update the commands to run the aligner with your own custom parameters however, instead of including file names specific markers are used to identify the read , the reference and input/output.

For example:

As you would normally for a command for an aligner:
/bin/BLAST **reads.fastq** **genome.bit** **alignments.out**

instead of using the names you use markers:
single_node_analysis = /bin/BLAST **ALIGNER_Read1** **ALIGNER_Ref** **ALIGNER_output1**

IMPORTANT: FULL PATHS MUST BE USED FOR ALL ALIGNERS/TOOLS

We will now look at the three customizable steps in the pipeline. Read preprocessing, Node alignment and post analysis processing.

Read Preprocessing

The first step in the pipeline is **read processing** this includes any steps necessary to convert your FASTQ reads into a format readable by your aligner. Read preprocessing is identified by "**read_pp**".

For example: To run BLAT: FASTQ reads must be converted to FASTA.
read_pp = /bin/fastq_to_fasta -i pp_Read -o pp_Out

NOTE: If not necessary leave this as "**none**".

The first marker (pp_Read) identifies the read file to be converted. Please note that this marker is not numbered as this conversion is done regardless of how many reads are specified. pp_Out is the second marker specifying with output that has to be saved. As previously shown this marker corresponds to a file that can be utilized in any following lines as pp_In.

Node analysis alignment

ClusterAlign recognizes two modes for alignment namely, single-end and paired-end analysis identified by single_node_analysis and paired_node_analysis respectively.

Further more, in several aligners multiple lines are necessary. In this case start each line with the variable and = sign followed by the next step in the alignment pipeline (See EXAMPLE1).

OUTPUT FILES

An important aspect to keep in mind is how to tell ClusterAlign what to do with output files for each step. In EXAMPLE1 a single output is given for the first line "**ALIGNER_output1**" this file is then used in the second line with the marker "**ALIGNER_in1**". Also notice that the second line ends again with "**ALIGNER_output1".** The output marker can be re-used, this marker is then associated to the new output. If you wish to keep this file (you want it returned to the final output directory) then refrain from using it in the remaing lines and use another output marker ie. "**ALIGNER_output2**".

EXAMPLE 1

single_node_analysis = /bin/bwa/bwa-0.5.0/bwa aln -t 4 ALIGNER_Ref1 ALIGNER_Read1 > **ALIGNER_output1**

single_node_analysis =/bin/bwa/bwa-0.5.0/bwa samse ALIGNER_Ref1 **ALIGNER_in1** ALIGNER_Read1 > **ALIGNER_output1**
RESULT: A single output is specified (ALIGNER_output1) to be included in the final directory

EXAMPLE 2

single_node_analysis = /bin/bwa/bwa-0.5.0/bwa aln -t 4 ALIGNER_Ref1 ALIGNER_Read1 > **ALIGNER_output1**

single_node_analysis =/bin/bwa/bwa-0.5.0/bwa samse ALIGNER_Ref1 **ALIGNER_in1** ALIGNER_Read1 > **ALIGNER_output2**
RESULT: ALIGNER_output1 and ALIGNER_output2 are included in the final output directory

Each file output must be specified and in cases where there are multiple output files for a single run different numbers must be used.

The next thing to note is the ALIGNER_Ref1 marker that indicates the reference genome specified in the aligner configuration file (***ref_file1***). Upto two references can be specified, this provides the option of including a second alignment program with its own reference.

The last marker used is the ALIGNER_Read1 / ALIGNER_Read2 marker that identifies the read number given earlier as either ***read_file1*** or ***read_file2***.

Post analysis alignment

Post alignment analysis, denoted as **post_analysis.** This includes any analysis a user wishes to do with the completed data. This step functions very similarly to the previous examples. The only difference is the inclusion of a new marker set identified as "**Results_1/ Results_2/ Results_3**" this denotes the output results from your alignment analysis ie. if you have used EXAMPLE2 as your alignment step you would be able to use **Results_1/ Results_2** (to correspond to **ALIGNER_output1/ ALIGNER_output2**) in post analysis.

EXAMPLE3

post_analysis = /bin/cufflinks/cufflinks -p 4 **Results_1**

3. How can I include my own custom aligners?

A template for the aligner script is provided and following the above rules this can be edited according to the user's needs to produce a unique pipeline script for their aligner. IMPORTANT: the name of your script must follow the convention "name.config" eg. "bowtie.config". The first argument in the ClusterAlign script will then be this name ie.

Python ClusterAlign.py bowtie ……