CLAN: the CrossLinked reads ANalysis tool
User Manual v0.0

By Cuncong Zhong, April 10, 2018
cczhong@ku.edu

License

Overview

CLAN is a software for aligning crosslinked reads against the reference genome. Examples of the existing next-generation sequencing technologies that generate crosslinked reads include, but not limited to, CLASH (1), LIGR-Seq (2), hiCLIP (3), and PARIS (4). CLAN demonstrated higher mapping sensitivity compared to other mapping tools such as BLAST (5), Bowtie2 (6), and STAR (7). CLAN is especially useful when analyzing ultra-short duplex reads, such as those generated for probing miRNA-mRNA interactions. This manual serves as a guide for the installation and execution of CLAN.

Download

CLAN can be downloaded from https://sourceforge.net/projects/clan-mapping.

Installation

CLAN was implemented using GNU C++. Current CLAN release only supports Linux (tested under RetHat Linux with gcc version 4.8.5).

After download, first unzip the CLAN package:

```
[cczhong@watson Codes]$ tar -zcf CLAN_v0.03-x86_84.tar.gz CLAN_release/
```

Then, go into the unzipped CLAN folded and compile:

```
[cczhong@watson Codes]$ cd CLAN_release/
[cczhong@watson Codes]$ make
```

You should find executables "`clan_index`", "`clan_search`", and "`clan_output`" under directory "`CLAN_release/bin`".

Quick start

We will use CLASH data analysis as an example for a quick start of using CLAN. By following these steps, you should also be able to reproduce the results reported in our manuscript describing CLAN.

We will work on the CLASH data set SRR959751, we recommend adaptor trimming and quality trimming, if necessary, prior to CLAN mapping. CLAN accepts both FASTQ and FASTA formats. We will use FASTQ format in this tutorial. The input file looks like:

```
[cczhong@watson CLAN_release]$ head ~/Works/CLAN/Data/CLASH_trimmed_cut.fastq
@SRR959751.39 HWI-EAS293_70410:5:1:1002:15658 length=55
ATGGCACTGGTAGAATTCACTG
+SRR959751.39 HWI-EAS293_70410:5:1:1002:15658 length=55
500-:5625AAAA7AAAAAAAA
@SRR959751.93 HWI-EAS293_70410:5:1:1014:6662 length=55
TCACCTGGAGCATGTTTCT
+SRR959751.93 HWI-EAS293_70410:5:1:1014:6662 length=55
55.2.3003AAAAAAA7AA
```

The next step is to prepare the reference sequence file. The reference sequence file is expected to be in the FASTA format. For example, the file could be the human genome reference hg38 directed downloaded from UCSC Genome Browser. In this tutorial, since our primary goal is to analyze the miRNA-mRNA interaction, we compile the reference with annotated microRNA sequences and the 3'UTR sequences of protein-coding genes. The reference file looks like:

```
[cczhong@watson CLAN_release]$ head ~/Works/CLAN/Results/hg38_MIR3UTR.fa
>chr1:17336-17536-||NR_106918.1-MIR6859-1
ATCAGACACAGGTACAGCACATAGGCCAGGAGCCAGGGGGTGACGGGTGGCTCGGCTCGGGAGGCCTGGGACCCCAC
AGTGCACGCTGTGCCCCTGATGATGTGGGAGAGGAACATGGGCTCAGGACAGCGGGTGTCAGCTTGCCTGACCCCCA
TGTCGCCTCTGTAGGTAGAAGAAGTATGTCTTCCTGGACCCCCTGGC
>chr1:30403-30603+||NR_036051.1-MIR1302-2
ATAATTTCGTAGCATAAATATGTCCCAAGCTTAGTTTGGGACATACTTATGCTAAAAAACATTATTGGTTGTTTATC
TGAGATTCAGAATTAAGCATTTTATATTTTATTTGCTGCCTCTGGCCACCCTACTCTCTTCCTAACACTCTCTCCCT
CTCCCAGTTTTGTCCGCCTTCCCTGCCTCCTCTTCTGGGGGAGTTAG
>chr1:187858-188058-||NR_107062.1-MIR6859-2
CAGACACAGGTACAGCACATAGGCCAGGAGCCAGGGGGTGACTGGGGTGGCTCGGCTCGGGAGGCCTGGGACCCCAC
AGTGCACGCTGTGCCCCTGATGATGTGGGAGAGGAACATGGGCTCAGGACAGCGGGTGTCAGCTTGCCTGACCCCCA
TGTCGCCTCTGTAGGTAGAAGAAGTATGTCTTCCTGGACCCCCTGGC
>chr1:632313-632513-||NR_106781.1-MIR6723
CGCTTCGAAGCGAAGGCTTCTCAAATTATGAAAATTATTAATATTACTGCTGTTAGAGAAATGAATGAGCCTACAGA
TGATAGGATATTTCATGTGGTGTATGCATCGGGATAGTCCGAGTAACGTCGGGGCATTCCGGATAGGCCGAGAAAGT
GTTGTGGGAAGAAAGTTAGATTTACGCCGATGAATATGATAGCGAAA
>chr1:1167098-1167298+||NR_029639.1-MIR200B
CCGGACCCAGCTCGGGCAGCCGTGGCCATCTTACTGGGCAGCATTGGATGGAGTCAGGTCTCTAATACTGCCTGGTA
ATGATGACGGCGGAGCCCTGCACGCAGCGACCGGCCGACCCCGTCCCGGCCCCCAGGGCCTCCCGCCGAGCCCCACA
CTCCCCTCCCACACAGCCCGCTCACCGGACCCCACCCCGTCCGGGCC
```

Now, we can build the CLAN index using executable "`clan_index`":

```
[cczhong@watson          CLAN_release]$          ./bin/clan_index         -f
~/Works/CLAN/Results/hg38_MIR3UTR.fa -d index/hg38_MIR3UTR
```

Here, "`–f`" is used to specify the location of the reference sequence file; and "`–d`" is used to specify the prefix of the output index files.

"`clan_index`" will output two index files with prefix "`index/hg38_MIR3UTR`":

```
[cczhong@watson CLAN_release]$ ls ./index
hg38_MIR3UTR.bwt  hg38_MIR3UTR.fmi
```

Then, we can start running "`clan_search`" to align the reads against the constructed index:

```
[cczhong@watson          CLAN_release]$          ./bin/clan_search          -r
~/Works/CLAN/Data/CLASH_trimmed_cut.fastq                                   -o
~/Works/CLAN/Results/CLASH_MIR3UTR.clan                                     -f
~/Works/CLAN/Results/hg38_MIR3UTR.fa                                        -d
~/Codes/CLAN_release/index/hg38_MIR3UTR -s -t 16
```

Here, "`–r`" is used to specify the location of the input read file; "`–o`" is used to specify the output file; "`–f`" is used to specify the reference file, and "`–d`" is used to specify the prefix of the pre-built index files. Since CLASH is a strand-specific technology, we use "`–s`" to reinforce strand-specific mapping; we also use "`–t`" to allow the program to run with 16 threads concurrently.

The output file "`~/Works/CLAN/Results/CLASH_MIR3UTR.clan`" is a binary file that cannot be viewed directly. We will use to "`clan_output`" to convert it into a viewable format:

```
[cczhong@watson          CLAN_release]$          ./bin/clan_output          -i
~/Works/CLAN/Results/CLASH_MIR3UTR.clan                                     -o
~/Works/CLAN/Results/CLASH_MIR3UTR.map -f ~/Works/CLAN/Results/hg38_MIR3UTR.fa
-r ~/Works/CLAN/Data/CLASH_trimmed_cut.fastq
```

Here, "`–i`" is used to specify the location of the binary output file of "`clan_search`"; "`–o`" is used to specify the location of the viewable output file; "`–f`" is used to specify the location of the reference file, and "`–r`" is used to specify the location of the read file.

We can now view the output file "`~/Works/CLAN/Results/CLASH_MIR3UTR.map`" as a regular text file:

```
[cczhong@watson Results]$ less CLASH_MIR3UTR_new.map
```

……

```
SRR959751.252 HWI-EAS293_70410:5:1:1036:16971 length=55 0        1        23
23        chr7:100093747-100093947-||NR_029510.1-MIR93:150-172;chr7:100093667-
100093947-||NR_029510.1-MIR93:150-172
SRR959751.254 HWI-EAS293_70410:5:1:1036:16306 length=55 0        19       33
33      chr19:31128141-31140229-||XR_935899.2-LOC105372356:6174-6188
SRR959751.254 HWI-EAS293_70410:5:1:1036:16306 length=55 0        3        18
33      chr3:77846570-78029966-||XR_940977.2-LOC105377171:154487-154502
```

The output of CLAN is a tab-delimited file. The first column indicates the unique read ID, the second column corresponds to the solution ID; the third and the fourth columns indicate the start and end position of the mapped region of the read (1-based); the fifth column indicates the total length of read, and the sixth column correspond to the location(s) (semicolon-delimited) in the reference where the read got mapped.

Let's look at the first line, which describes the mapping of the read "`SRR959751.252 HWI-EAS293_70410:5:1:1036:16971 length=55`". The read is mapped as a non-chimeric read, because no other record of the read shares the same solution ID with it. The read is also mapped entirely (all 23 bases of the read are mapped, which can be determined from the second to the fourth columns). The read is mapped to two locations in the reference, which are separated using semicolon ";" in the sixth column. Each record for the reference location, for example "`chr7:100093747-100093947-||NR_029510.1-MIR93:150-172`", is further divided into two fields with a colon ":". The first field "`chr7:100093747-100093947-||NR_029510.1-MIR93`" is the ID of the reference sequence; and the second field "`150-172`" is the start and end location (1-based) where the reference is mapped.

Now we look at the second and the third columns, both of which describe the mapping of the read "`SRR959751.254 HWI-EAS293_70410:5:1:1036:16306 length=55`". This read is mapped as a chimeric read, which can be determined by the existence of two records having the same solution ID of "`0`". Further looking at the mapped locations of the read, we found that the first half of the read "`3        18`" is uniquely mapped to "`chr3:77846570-78029966-||XR_940977.2-LOC105377171:154487-154502`"; and the second half of the read "`19        33`" is uniquely mapped to "`chr3:77846570-78029966-||XR_940977.2-LOC105377171:154487-154502`". This read thus suggest a potential interaction between the two corresponding locations.

The user can perform downstream analysis using the information provided by CLAN's mapping thereafter.

Parameters

`clan_index`: build index from reference sequences

"`-f`": Specifies the location of the reference sequence file. The file is expected to be in FASTA format. This tag is mandatory.
"`-d`": Specifies the prefix of index files that are going to be constructed. The two extensions of the index files are going to be "`.bwt`" and "`.fmi`". This tag is mandatory.
"`-h`": Prints the help information.

`clan_search`: mapping the reads to the reference

"`-r`": Specifies the location of the read file. The file is expected to be in either FASTQ or FASTA format. CLAN will attempt to detect the format automatically. This tag is mandatory.
"`-o`": Specifies the location of the output file. The file is a binary file and needs to be converted before viewing. This tag is mandatory.
"`-f`": Specifies the location of the reference sequence file. The file is expected to be in FASTA format. This tag is mandatory.

"`–d`": Specifies the prefix of the previously constructed index file. The two extensions of the index files are going to be "`.bwt`" and "`.fmi`". This tag is mandatory.
"`–s`": Specifies strand-specific mapping. If the tag is not set, CLAN will map the reads to both locations. This tag is optional. Default FALSE.
"`–e`": Specifies whether to disable penalty of the insert sequence between the two arms. When set to false, CLAN mapping will be more stringent. We do not recommend modifying the default setting without thorough understanding of the CLAN algorithm. This tag is optional. Default TRUE.
"`–t`": Specifies the number of threads to use. This tag is optional. Default 1.
"`–m`": Specifies the maximum number of hits for multi-mapped reads reporting. If a read got mapped to too many reference locations, the read could be too short or come from a repeat region. This tag is optional. Default 20.
"`–k`": Species the length of flanking bases when recording non-maximal fragments. Increasing this parameter will lead to higher mapping sensitivity and longer running time. We do not recommend modifying the default setting without thorough understanding of the CLAN algorithm. This tag is optional. Default 4.
"`–l`": Specifies the minimum length for each fragment, or the seed length. Increasing this parameter will lead to lower mapping sensitivity and faster running time. We do not recommend modifying the default setting without thorough understanding of the CLAN algorithm. This tag is optional. Default 10.
"`–p`": Specifies the penalty for introducing one extra arm. Increasing this parameter will suppress chimeric mapping and encourage non-chimeric mapping. We do not recommend modifying the default setting without thorough understanding of the CLAN algorithm. This tag is optional. Default 10.
"`–v`": Specifies the maximum number of overlapping bases allowed between the two duplex arms. Increasing the parameter will lead to higher mapping sensitivity and slower running time. We do not recommend modifying the default setting without thorough understanding of the CLAN algorithm. This tag is optional. Default 5.
"`–h`": Prints the help information.

`clan_output`: convert binary file to viewable text file

"`–i`": Specifies the location of the binary file output by "`clan_search`". This tag is mandatory.
"`–o`": Specifies the location of the output viewable file. This tag is mandatory.
"`–f`": Specifies the location of the reference sequence file. The file is expected to be in FASTA format. This tag is mandatory.
"`–r`": Specifies the location of the read file. The file is expected to be in either FASTQ or FASTA format. CLAN will attempt to detect the format automatically. This tag is mandatory.
"`–m`": Specifies the output format. 1: CLAN format; 2: BLAST tab format, 3: SAM format. This tag is optional. Default 1.
"`–h`": Prints the help information.

References

1. Helwak, A., Kudla, G., Dudnakova, T. and Tollervey, D. (2013) Mapping the human miRNA interactome by CLASH reveals frequent noncanonical binding. *Cell*, **153**, 654-665.
2. Sharma, E., Sterne-Weiler, T., O'Hanlon, D. and Blencowe, B.J. (2016) Global Mapping of Human RNA-RNA Interactions. *Mol Cell*, **62**, 618-626.
3. Sugimoto, Y., Vigilante, A., Darbo, E., Zirra, A., Militti, C., D'Ambrogio, A., Luscombe, N.M. and Ule, J. (2015) hiCLIP reveals the in vivo atlas of mRNA secondary structures recognized by Staufen 1. *Nature*, **519**, 491-494.
4. Lai, D. and Meyer, I.M. (2016) A comprehensive comparison of general RNA-RNA interaction prediction methods. *Nucleic Acids Res*, **44**, e61.
5. Altschul, S.F., Madden, T.L., Schaffer, A.A., Zhang, J., Zhang, Z., Miller, W. and Lipman, D.J. (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res*, **25**, 3389-3402.
6. Langmead, B., Trapnell, C., Pop, M. and Salzberg, S.L. (2009) Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biol*, **10**, R25.
7. Dobin, A., Davis, C.A., Schlesinger, F., Drenkow, J., Zaleski, C., Jha, S., Batut, P., Chaisson, M. and Gingeras, T.R. (2013) STAR: ultrafast universal RNA-seq aligner. *Bioinformatics*, **29**, 15-21.