How to use TraRECo

Seokhyun Yoon (syoon@dku.edu), June 19, 2017

Requirements

1. Hardware: around **20GB of free memory**
2. **MATLAB™ 2009 or later** installed on your computer (No toolboxes are required though) - TraRECo is a MATLAB™ script and you need MATLAB™ to run it.
3. **Your patience** - TraRECo was developed as a MATLAB script to reduce debugging time, while it is much slower than other assemblers developed using C/C++. Currently, we are converting TraRECo into C/C++ language, which will be TraRECo version 1.0. Typically, it takes 5 to 10 days to assemble 50~100M reads of length 75~100 bases. ☹

Preparation and Running TraRECo

I'm assuming you know how to use MATLAB and how to run MATLAB script from MATLAB command window.

1. Unzip the TraRECo_v0.6.3_distr.zip downloaded
2. Check the directory(folder) to see the following items
    - **trareco_v063.m** - a MATLAB script containing all the functions required to run TraRECo version 0.6.3
    - **main_script_example_paired.m and main_script_example_single.m** - The MATLAB script examples showing how to use TraRECo. You can edit the script to perform your own assembly. Just open it using MATLAB editor. You can see more description in this script.
    - **config_paired.par and config_single.par** - text file configuring some assembly parameters, such as nominal read length, normalized distance threshold and so on. Before running the main script, you may want to edit this configuration file. The file name (and its path if it is not in the same folder) must be provided to the main script, as already did in main_script_example_paired.m and main_script_example_single.m.
    - **compile_cmex.m** and Five **c source files** (sub_tcn_v04b.c, ...) - These c source contains interface with and can be called by MATLAB. To make them callable by MATLAB, you must compile them first by running compile_cmex.m using MATLAB command window. Before compiling, you may need to run "mex -setup" in MATLAB command window to choose c compiler. (Check https://kr.mathworks.com/help/matlab/matlab_external/introducing-mex-files.html for more details on MEX function)
    - **sample_data/ folder** - The folder contains sample paired-end read data and reference transcriptome generated using the Flux Simulator (available at http://sammeth.net/confluence/display/SIM/Home). The reference transcriptome is used to evaluate the assembled transcripts by running run_blast.m.
    - **run_blast.m** - Align assembled transcripts with reference transcriptome to count the number of candidates that match to one reference transcript for given target coverage. See the main scripts to see how to use run_blast.m. Note that, to run this function, you need BLAST SW installed on your computer. (available at https://www.ncbi.nlm.nih.gov/guide/howto/run-blast-local/)
3. **Compile c source files by running compile_cmex.m** using MATLAB command window. This will creates mex files with extension ".mex….".

4. **Run main_script_example_paired.m (or main_script_example_single.m)** from MATLAB command window and see how TraRECo works. The resulting transcriptome file(s) will be created at the directory (folder) you specified in your configuration file with its name given by **xxx_MinCvgDepth_y.y.fasta**, where **xxx** is the file name prefix you specified in your configuration and **y.y** (e.g., 3.0) is the coverage depth threshold of the transcripts contained in the file, i.e., the file contains only those transcript candidates whose estimated coverage depth is equal to or greater than **y.y**.
5. **Checking out the final assembly results in xxx_MinCvgDepth_y.y.fasta** – The file is basically a FASTA file with header line and base sequences pair, as shown below.

```
>T_3_21_1_1_33.000000   3   21  33.000000   68  214710  214711
CTGATTTTCCCAAAGAATTAAAGCGATGAAGGGAGAAGACAAGATTCTTTCACCCCCTTTAAAAAAAA
>T_4_22_1_1_66.000000   4   22  66.000000   66  214710  214711
CAAAGAATTAAAGCGATGAAGGGAGAAGACAAGATTCTTTCACCCCCTTTTCTTCATCAGAAAAAA
:
```

As shown above, the header line contains seven items: the first item is a string representing (candidate) **Transcript ID** with a prefix “T_” and the last 6 items are all numbers representing respectively **the serial number**, **group index**, **estimated abundance** (average read coverage depth), **length of the sequence**, **the number of reads used** in the assembly and **the number of reads provided** by the input file(s), where only the estimated abundance is in floating point number while all others are integer number.

Some (candidate) Transcripts may share the same group index, which means they are from the same sub-graph and, hopefully, isoforms from the same gene. Note, however, that some of genes can be merged into one sub-graph due to sequence repeats among different genes and/or similar sequence contained in many different genes. Note also that the transcript ID contains some of the information mentioned below delimited by ‘_’, which is useful when you run BLAST as the transcript ID will be shown as query sequence id.