

paraGSEA tutorial

Shunyun Yang

September , 2016

1 Build instructions

paraGSEA runs on Mac and Linux as a command line application. You can download the source code of paraGSEA from Github with the following command on a standard terminal.

git clone <https://github.com/ysycloud/paraGSEA.git>

Note that this requires that you already have a Github account and that the computer you are working on has an SSH key registered on Github. If this is not the case, follow the instructions from <https://help.github.com/articles/generating-ssh-keys/>.

This should download a directory named paraGSEA. To build paraGSEA, execute the following.

cd paraGSEA

make all

make install

This should succeed on most Linux systems because make is available by default. If this is not the case, you can obtain it by typing **sudo apt-get install make** on Ubuntu. Other Linux systems can also easily obtain the *make* tool by some simple commands. On Mac, you need to install *XCode*, which may take some time. First, you will need an Apple ID, then you will need to download it from the developer website of Apple <https://developer.apple.com/xcode/downloads/>. Then, you may need to follow the instructions shown on the following link to install the command line version of *make*. <http://stackoverflow.com/q/10265742/1248687>.

Calling make should create some executable files. Note that you need root authority to run **make install** command, then you can running the commands of paraGSEA in any path of this system. If you cannot, the application can be only used in paraGSEA/bin directory. To check that the building is successful, execute the following command.

./quick_search_serial

If you obtain the output shown below, then everything went fine and you are done with the build. If not, then something went wrong. In this case, you can explain how to reproduce the problem on <https://github.com/ysycloud/paraGSEA/issues>. Then, we will solve it for you as quick as possible.

Usage: quick_search_serial [options]

general options:

-n --topn: The first and last N GSEA records ordered by ES. [default 10]

input/output options:

-i --input: input file/a parsed profiles's file from pretreatment stage.

-s --sample: input file/a parsed sample sequence number file from pretreatment stage.

-r --reference: input a directory includes referenced files about genesymbols and cids.

2 paraGSEA basics

paraGSEA implements a MPI and OpenMP-Based parallel GSEA algorithm for multi-core or cluster architecture. But some pretreatment tasks for original data must use *Iktools*, that is an open-source tool with a variety of implementations. In our work, we use the *Matlab* version.

Therefore, make sure you have installed the common tools we listed below before you could use paraGSEA.

1. Matlab R2009a and above

2. MPI

3. gcc compiler supports OpenMP

There are mainly three parts of work in paraGSEA.

First, we implement GSEA approach in efficient parallel strategy with MPI and OpenMP to perform a quick search task, which needs users input a gene set and it will output the top N results after searching the profile data set by carrying out GSEA calculations. In this part, on the one hand, we reduced the computational overhead of standard procedure to calculate the Enrichment Score by pre-sorting, indexing and removing the prefix sum. On the other hand, we will take a global permutation method to wipe off the redundant overhead of estimation of significance level step.

Second, we expanded GSEA's application to quickly compare two gene profile sets to get an Enrichment Score matrix of every gene profile pairs. In this part, in addition to using the previous optimization strategies, our implementation also allows to generate a second level of parallelization by creating several threads per MPI process. The assignment of tasks to threads or processes is performed through a strict load balancing strategy, which leads to a better performance.

Third, we clustered the gene profile based on the Enrichment Score matrix which we can get by the second part. In this part, Enrichment Score is served as the metric to measure the similarity between two gene profiles. We implemented a general clustering algorithm like K-Medoids which is an improved version of K-Means. The algorithm can quickly converge and then output the corresponding results.

3 Input formats and Pretreatment

The original input data stored in the HDF5 file format with a 'gctx' suffix. In order to use and analysis the data, we must use *Iktools* (<https://github.com/cmap/11ktools>), which is an Open-Source project published in github, to parse it and extract the information we care about.

There is an example file '**modzs_n272x978.gctx**' in 'paraGSEA/data' directory. It is our profiles data set. '**n272x978**' means there are 272 profiles with 978 genes for each. In this file, every gene has a '*rid*', which is corresponding to a gene name(symbol). Every profile has a '*cid*', which identifies a set of experimental conditions to get this profile. There is example of '*cid*' shown below and others must keep in the same format.

CPC006_A549_6H:BRD-U88459701-000-01-8:10

Every part of '*cid*' means different experimental condition. Using the '*cid*' above as an example. '**A549**' means the cell line, '**6H**' means duration, '**BRD-U88459701-000-01-8**' means perturbation, '**10**' means concentration whose unit is 'um'. By splitting '*cid*' to get every part condition of this profile, we provide user-friendly parsed method to allow user set their own conditions of profile they need.

In order to achieve this goal, we must generate some reference data to facilitate our main work. There is a *Matlab* script in 'paraGSEA/matlab_for_parse' directory named '**genReferenceforNewDataSet.m**' to help us finish this work.

To use this script, we should first set the MATLAB path:

Enter the “pathtool” command, click “Add with Subfolders...”, and select the directory ‘paraGSEA/matlab_for_parse’. Then executing the following command in *Matlab* environment.

```
datasource='../data/modzs_n272x978.gctx';  
genReferenceforNewDataSet
```

When we get a new profile file keeps in correct format with a ‘gctx’ suffix, we can set its path in ‘**datasource**’ variable. Then, three files will generate in ‘../data/Reference’ directory.

1. Gene_List.txt: all gene names of every profile in original order recorded in new data source file.

2. Samples_Condition.txt: treatment conditions of all profiles in original order recorded in new data source file.

3. Samples_RowByteOffset.txt: Bytes offset of every line in file2 in order to locate Specific line conditions directly without loading all file2 into memory.

Note that the file ‘**annot.mat**’ in ‘paraGSEA/matlab_for_parse’ provides all relationships of ‘rid’ corresponding to gene name in human from now. Only with its help, can we get ‘**Gene_List.txt**’ file.

After generating the reference data, we also provide two *Matlab* scripts to support user-friendly parsed method to allow user set their own conditions of profile they need and extract corresponding profiles to analysis.

You can easily use the script ‘example/runPreGSEAbymatlab.sh’ to parse the original data. Or you can easily use the script ‘example/runparaPreGSEAbymatlab.sh’ to parse the original data in a more efficient way but you must make sure that you have a multicores environment first.

The input and output file path you must set in both two script, and an extra core number is also supposed to be set in the second script. However, you must notice that the number of cores must be smaller than the actual core number in your system.

4 Quick search

5 Compare profiles

6 Clustering profiles