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 *1ktools*, 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-Mediods 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 *1ktools* (https://github.com/cmap/l1ktools), 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.

1. **PreGSEA.m**
2. **paraPreGSEA.m**

For example, you can execute following script.

```
file_input='../data/modzs_n272x978.gctx';
file_name='../data/data_for_test.txt';
file_name_cidnum='../data/data_for_test_cidnum.txt';
cell_id_set={'A549','MCF7','A375','A673','AGS'};
pert_set={'BRD-A51714012-001-04-9','BRD-A51714012-001-03-1'};
duration = '6H';
concentration='10';
PreGSEA;
```

Here, we can see '**file_input**' represents the original profile file, '**file_name**' represents needed profiles the script extracts, '**file_name_cidnum**' represents the sequence number of profiles the script extracts, '**cell_id_set**' represents the cell lines set where the profiles should be get from, '**pert_set**' represents the perturbations that should be used in experiments to get the profiles, '**duration**' represents the time to carry out the experiments and '**concentration**' represents the concentration is supposed to be kept during the experiments.

If you not set these parameters, there will be some default values in '**PreGSEA**'. However, those may not fit your need or definitely correct. '**paraPreGSEA**' script can parse the original data in a more efficient way by a multi-thread method but you must make sure that you have a multi-core environment first. There is another parameter '**cores**' can be set to determine the parallel level. However, you must notice that the number of cores must be smaller than the actual core number in your system.

Moreover, because the results be splitted into several parts, we need finish some remedial work by *unix* command line to combine them into whole correct file. Fortunately, we provide two shell scripts to handle all the processes.

1. **example/runPreGSEAbyMatlab.sh**
2. **example/runparaPreGSEAbyMatlab.sh**

The only thing that users need to do is just set some parameters in these two scripts.

4 Quick search

Once we get the standard txt file which is parsed from the original input data, paraGSEA can read it quickly and then keep on subsequent calculations. As we mentioned above, Quick Search 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.

It is worth mentioning that there are several implementations in three versions. The MPI version can run on multiple nodes to handle larger amounts of data. Moreover, it

supports parallel IO. The OpenMP implemented a more lightweight version of parallel computing, and there is no extra overhead of communication between nodes. Actually, there is no advantage of Serialized version as compared to the previous two, it is just for comparative analysis.

The Usage of three version is shown below.

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.

Usage: quick_search_omp [options]
general options:
-t --thread: the number of threads. [default 1]"
-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.

Usage: quick_search_mpi [options]
general options before command by MPI:
-n process_num : Total number of processes. [default 1]
-ppn pernum: the number of processes in each node. [default 1]
-hostfile hostfile: list the IP or Hostname of nodes. [default localhost]
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.

The Usages have been detailed enough. Only note that '*-i –input'* corresponding to '**file_name**', '*-s –sample'* corresponding to '**file_name_cidnum**' and '*-r –reference'*

corresponding to '../data/Reference' directory we set in pretreatment stage.

Here is an example.

./quick_search_serial -i data/data_for_test.txt -s data/data_for_test_cidnum.txt -n 5 -r data/Reference

In principle, you can conduct following interactive process.

Profile Set is Loading...!
profilenum:272 genelen:978
loading IO and prework time: 0.0237 s
which way do you want to input the GeneSet(0 -> standard input , others -> file input):1
input the path of file that has GeneSet until 'exit'(each line has a Gene Symbol/name):
data/GeneSet.txt

printf the high level of TopN GSEA result:
NO.1 -> SampleConditions: cid: CPC006_SKLU1_6H:BRD-K56343971-001-02-3:10; cell_line: SKLU1; perturbation: BRD-K56343971-001-02-3; duration: 6H; concentration: 10um
ES:0.315086 NES:2.516304 pv:0.0000000000
NO.2 -> SampleConditions: cid: LJP001_MCF10A_24H:BRD-K56343971-001-04-9:0.08; cell_line: MCF10A; perturbation: BRD-K56343971-001-04-9; duration: 24H; concentration: 0.08um
ES:0.225345 NES:1.819965 pv:0.0014360776
NO.3 -> SampleConditions: cid: NMH001_NEU.KCL_6H.4H:BRD-K69726342-001-02-6:10; cell_line: NEU.KCL; perturbation: BRD-K69726342-001-02-6; duration: 6H.4H;concentration: 10um
ES:0.223448 NES:1.814151 pv:0.0019234482
NO.4 -> SampleConditions: cid: LJP001_BT20_24H:BRD-K56343971-001-04-9:0.4; cell_line: BT20; perturbation: BRD-K56343971-001-04-9; duration: 24H; concentration: 0.4um
ES:0.221078 NES:1.823220 pv:0.0021228018
NO.5 -> SampleConditions: cid: LJP001_MCF7_24H:BRD-K56343971-001-04-9:2; cell_line: MCF7; perturbation: BRD-K56343971-001-04-9; duration: 24H; concentration: 2um
ES:0.216035 NES:1.730512 pv:0.0025818102

printf the low level of TopN GSEA result:
NO.1 -> SampleConditions: cid: LJP001_HS578T_6H:BRD-K56343971-001-04-9:0.4; cell_line: HS578T; perturbation: BRD-K56343971-001-04-9; duration: 6H; concentration: 0.4um
ES:-0.229569 NES:-1.896862 pv:-0.0004835776
NO.2 -> SampleConditions: cid: CPC006_HEC108_6H:BRD-U88459701-000-01-8:10; cell_line: HEC108; perturbation: BRD-U88459701-000-01-8; duration: 6H; concentration: 10um
ES:-0.219655 NES:-1.800255 pv:-0.0016587932
NO.3 -> SampleConditions: cid: CPC006_SNUC5_6H:BRD-K56343971-001-02-3:10; cell_line: SNUC5; perturbation: BRD-K56343971-001-02-3; duration: 6H; concentration: 10um
ES:-0.202629 NES:-1.662714 pv:-0.0034929736
NO.4 -> SampleConditions: cid: CPC004_A375_6H:BRD-A51714012-001-03-1:10; cell_line: A375; perturbation: BRD-A51714012-001-03-1; duration: 6H; concentration: 10um
ES:-0.194224 NES:-1.584745 pv:-0.0050603876

NO.5 -> SampleConditions: cid: CPC006_SNGM_6H:BRD-U88459701-000-01-8:10; cell_line: SNGM;
perturbation: BRD-U88459701-000-01-8; duration: 6H; concentration: 10um
ES:-0.192112 NES:-1.586885 pv:-0.0059169393
finish GSEA time: 0.1536 s
input the path of file that has GeneSet until 'exit'(each line has a Gene Symbol/name):
exit

Note that you can choose input a gene set directly or input a file path where there is a gene set. Second way may be more convenient such as the example shows.

The other two versions are totally same interactive processes, where we no longer give an example.

5 Compare profiles

6 Clustering profiles