User manual of GIST 3.0

Yingcheng Sun
ys3345@cumc.columbia.edu
02/22/2020

New features

1. **Support batch processing mode for clinical trials**. Previous versions of GIST only allow users to generate query and "gist" scores for clinical trials one at a time. In GIST 3.0, both queries and the "gist" scores can be obtained in batch mode that supports multiple clinical trials to be proceeded in one single request. It will be very convenient for users, especially there are a large number of requests.

2. **Processing time is dramatically decreased**. The processing time of Gist score's calculation is dramatically decreased by about 90% in GIST 3.0. It makes the GIST tool for frequent use available.

3. **"One-step" operation for the gist calculation**. In the last version, users need to generate the search query first and copy it from the Terminal window to the API development platform such as Postman, and then obtain the results. These manual operations are very risky and not convenient. In GIST 3.0, we put the whole process into "one-step". The only thing user need to do is tell GIST the target clinical ID list, and all other operations will be automatically done on the backend.

4. **All results are automatically saved**. In GIST 2.0, users have to copy and save the results to files by themselves. In GIST 3.0, all generated queries and eligibility criteria traits with GIST scores will be automatically saved as text files named by their clinical trial ID. They are very easy to organized and proceeded for further analysis.

5. **Output are beautifully and clearly formatted**. In GIST 2.0, the output such as generated queries and eligibility criteria with GIST scores are shown in raw format and very hard to read. Users have to format them by themselves such as copying the results to "Postman" tool and "beautify" them. In GIST 3.0, all results are presented in a well-organized way as a beautiful and clear format under JSON standard. Also, the log info in the Terminal window are also shown more friendly to the users.

How to use?

INSTALLATION:

1. Download GitHub repository
2. go into each folder and type “npm i” this will download all the packages required for the scripts/servers to run properly.

SETTING AND RUNNING:

C2Q Folder - C2Q reads the input table and create an input JSON object for the server: For this script to run as desired, you will need to have a proper sql table in an sql database (One row per eligibility criterion, required columns: nct_id, condition_concept_id, condition_concept_name, criteria_concept_id, criteria_concept_name, include, criteria_domain, min_value, max_value, male_allowed, female_allowed [last two for gender only]).

When you have the data ready you can format the script to your database. You must change the following in the “*index.js*” file:

- If you are using a mysql database, the index.js file need to change as the mysql package needs to be downloaded instead Postgres. But since the main part of the C2Q program is isolated from the query, it can be easily done.

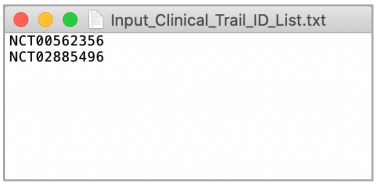
- For Postgres, you need to change the following parts of “*client*” with the proper config of your server where the sql input table will be. Change the *user property* to the user of your database. Change the *host* to an ip if the server is not on your local machine. Change the *password* to the password for the user of the account. Change the *database* where the input table is in. Change the *port* to where your server is hosted.

You can change the following in the “*crit2query.js*” file as you like:

- To obtain a small sample set, we limit the patient number to 1000, it can be set as any number or completely removed in “*crit2query.js*”: *joins.reduce((prev, val) => prev + val) + **" limit 1000;"**,*

- The query generator has a slight preference for Postgres. Line 9 of crit2query.js, “on” is only part of the POSTgres sql language. The “on” keyword is meant to grab the latest date of a measurement value. TO combat this, you will need to find a query template that solely grabs the latest/relevant measurement value based on measurement dates.

Next, input the clinical trial ID in the *“Input Clinical Trial ID List.text”* file, one ID per line, like this:



After setting everything well, run the criteria2query query generation script by navigating to the '*criteria2query*' folder in terminal and running *'node index.js'*:

```
Hao-iMac:c2q ys3345$ node index.js
------------QUERY GENERATION------------
You input 2 clincial trials: NCT00562356,NCT02885496
Start to generate query for No. 1 clinical trail: NCT00562356
Query Generated!
Start to generate query for No. 2 clinical trail: NCT02885496
Query Generated!
```

Each generated query is saved as a text file named by the clinical trial ID and all files are located in the "*query*" folder under "C2Q" folder.

GIST Folder is irrelevant, it is just for reference for data formatting.

Server Folder - the server receives beautiful inputs and runs the gist calculations.

- If you do not have Python running environment, you might need to install Python 2 or 3 first, and also need install the *sklearn* package.

- Next, you must change the following in the "*server.js*" file: change the following parts of "*client*" with the proper config of your server where the sql input table will be. Change the *user property* to the user of your database. Change the *host* to an ip if the server is not on your local machine. Change the *password* to the password for the user of the account. Change the *database* where the input table is in. Change the *port* to where your server is hosted.

Run the criteria2query query generation script by navigating to the 'criteria2query' folder in terminal and running 'node server.js':

```
Hao-iMac:server ys3345$ node server.js
------------GIST CALCULATION------------
You input 2 clincial trials: NCT00562356,NCT02885496
Start to process No. 1 clinical trail: NCT00562356
starting hypersurface process
training
Total number of eligible patients: 3843
Current sGIST array: 1,0.8097497074833393,0.9273541232131047,0.9436333112885995,0.9174467110952841,0.36
77061606552373,0.5668464160350003,0.3716106221702193
No 0s, moving on!
Start to process No. 2 clinical trail: NCT02885496
starting hypersurface process
training
Total number of eligible patients: 2870
Current sGIST array: 1,0.6207712265350765,0.9861881263671974,0.9273541232131047,0.08823828661545506,1,0
.9734318563361652,0.6932390497023961
No 0s, moving on!
```

The calculation result for each query is saved as a text file named by the clinical trial ID and the output is well organized in JSON format. All files are located in the "*results*" folder under "*server*" folder. An example is shown as follows:


```
{
"output": [
    {
        "criteria_concept_name": "Non-insulin dependent diabetes mellitus",
        "criteria_concept_id": 201826,
        "criteria_domain": "Condition",
        "criteria_elig_binary": 1,
        "criteria_query": "with  c201826 as ( select distinct person_id as c201826 from condition_occurrence where condition_concept_id = 201826)",
        "column_name": "c201826",
        "sq_score": 1
    },
    {
        "criteria_concept_name": "Hemoglobin; glycosylated (A1c)",
        "criteria_concept_id": 2212392,
        "criteria_domain": "Measurement",
        "criteria_elig_binary": 1,
        "criteria_min": 6.5,
        "criteria_max": 11,
        "criteria_query": "with  m2212392 as ( select distinct on (person_id) person_id,  value_as_number as m2212392, measurement_date from measurement where measurement_concept_id = 2212392 order by 1, 3 desc)",
        "column_name": "m2212392",
        "mean": 8.758072126916472,
        "std": 1.876706497305556,
        "tally": 0.3086684687287209,
        "n_min": -1.2032100054458816,
        "n_max": 1.1946076151027003,
        "sq_score": 0.8097497074833393
    },
    {
        "criteria_concept_name": "Liver Disease",
        "criteria_concept_id": 194984,
        "criteria_domain": "Condition",
        "criteria_elig_binary": 0,
        "criteria_query": "with  c194984 as ( select distinct person_id as c194984 from condition_occurrence where condition_concept_id = 194984)",
        "column_name": "c194984",
        "sq_score": 0.9273541232131047
    },
    {
        "criteria_concept_name": "Creatinine; blood",
        "criteria_concept_id": 2212294,
        "criteria_domain": "Measurement",
        "criteria_elig_binary": 1,
        "criteria_min": -99999,
        "criteria_max": 1.8,
        "criteria_query": "with  m2212294 as ( select distinct on (person_id) person_id,  value_as_number as m2212294, measurement_date from measurement where measurement_concept_id = 2212294 order by 1, 3 desc)",
        "column_name": "m2212294",
        "mean": 1.1023460613846088,
        "std": 0.6362336730383266,
        "tally": 0.18320105820105820582,
        "n_min": -157175.11754527554,
        "n_max": 1.0965372758152723,
        "sq_score": 0.9436333112885995
    },
    {
        "criteria_concept_name": "Myocardial Infarction",
        "criteria_concept_id": 312327,
        "criteria_domain": "Condition",
        "criteria_elig_binary": 0,
        "criteria_query": "with  c312327 as ( select distinct person_id as c312327 from condition_occurrence where condition_concept_id = 312327)",
        "column_name": "c312327",
        "sq_score": 0.9174467110952841
    },
    {
        "criteria_concept_name": "Age",
        "criteria_concept_id": 4156190,
        "criteria_domain": "Person",
        "criteria_elig_binary": 1,
        "criteria_query": "with age as ( select person_id, extract(year from age((year_of_birth || '-' || month_of_birth || '-' || day_of_birth)::date)) as age from person)",
        "criteria_min": 18,
        "criteria_max": 80,
        "column_name": "age",
        "mean": 83.30501856844489,
        "std": 12.635495349760234,
        "tally": 0.6322938393447627,
        "n_min": -5.168378188646803,
        "n_max": -0.2615662051200561 3,
        "sq_score": 0.3677061606552373
    },
    {
        "criteria_concept_name": "Gender",
        "criteria_concept_id": 4135376,
        "criteria_domain": "Person",
        "criteria_elig_binary": 1,
        "criteria_query":  "gender as ( select distinct person_id, gender_concept_id as gender from person)",
        "male_allowed": 0,
        "female_allowed": 1,
        "column_name": "gender",
        "sq_score": 0.5668464160350003
    },
    {
        "criteria_concept_name": "Metformin",
        "criteria_concept_id": 1503297,
        "criteria_domain": "Drug",
        "criteria_elig_binary": 1,
        "criteria_query": "with  d1503297 as ( select distinct person_id as d1503297 from drug_era where drug_concept_id = 1503297)",
        "column_name": "d1503297",
        "sq_score": 0.3716106221702193
    }
],
"mGIST": 0.04887571857353615
}
```