OpenTargets Code Test

Thank you for your interest in Open Targets!

The first step in our interview process is always a code test. We would like you to complete as much as you can of the problems described below. You must use python. We suggest you should not dedicate more than 3-5 hours to the task.

You are welcome to use any external library, as long as you provide clear instruction to install it. Reinventing the wheel is not valued. Working code, proper testing, useful documentation and code reusability are highly valued.

If you have any doubts, please state your assumptions, document your process and attempt to continue with the test.

If you really get stuck you can email to data@opentargets.org

Problem A - query a REST API

The goal of this test is to assess your ability to query a remote documented REST API, fetch and analyse the data, and test your code to specifications.
You will use our targetvalidation.org REST API, which is documented at http://api.opentargets.io/v3/platform/docs

We would like you to build a program in python that can query our REST API to get a data value labelled as `association_score.overall` for a given target ID or disease ID. As explained in the API documentation, target IDs can be specified in the Ensembl Gene ID format (eg. `ENSG00000157764`) while disease IDs are specified in the Experimental Factor Ontology format (eg. `EFO_0002422`).

Your code should:

- Query the `https://api.opentargets.io/v3/platform/public/association/filter` REST API endpoint to get target to disease association information. The `target` parameter can be used to query for target-related information (eg. use the string `ENSG00000157764` as a target id) and the `disease` parameter can be used to query for disease-related information (eg. use the string `EFO_0002422` as a disease id).
- From the returned JSON object parse for each entry the value returned at `association_score.overall`
- Print out to stdout the maximum, minimum, average and standard deviation values of `association_score.overall`
- Parse the arguments passed from the command line so that:
  - `python my_code_test.py -t ENSG00000157764` will run an analysis for a target
  - `python my_code_test.py -d EFO_0002422` will run an analysis for a disease
  - `python my_code_test.py --test` will run a suite of tests
- The suite of test should check the output for `my_code_test -t ENSG00000157764`

- ❏ It should also test the output for `my_code_test -d EFO_0002422`
- ❏ and it should test the output for `my_code_test -d EFO_0000616`

Problem B - parse a JSON dump

The goal of this problem is to parse quickly and efficiently large-ish data files and to calculate statistics on the data extracted. The works is a simplified and representative version of much of the Extract-Transform-Load work that we do at Open Targets.

First, you should download our _evidence_ data from:
https://storage.cloud.google.com/open-targets-data-releases/17.12/17.12_evidence_data.json.gz

This file contains a series of JSON objects, each representing an *evidence* linking a *target* to a *disease* or more than one diseases.

First part

We want you to write a python program that:

1. For each json object in the file, parse `target.id`, `disease.id` and `score.association_score`
2. For each `target.id`, `disease.id` pair, calculates the median and the top 3 `association_score`.
3. Outputs the resulting table in `csv` format, sorted in ascending order by the median value of the `association_score`

Second part

Each JSON object in the file defines a connection between a *target* and a *disease*. Since there are ~30,000 targets and ~8,000 diseases, different targets will be connected to the same disease.

You should expand your python program from the first part of the problem to use the same data and count how many `target-target` pairs share a connection to at least two diseases.

Notes

Better solutions will:

- ❏ take advantage of all cpus
- ❏ finish in less time
- ❏ use less memory

You should submit both code and outputs, if any.