Clinical Trial Matching Engine, v0.1

Tiffany Huang, Jack Jacobs, Sara Khoshhal, Andy Nunn

Updated: October 14, 2021

Table of Contents

Program Structure

The main script of the clinical trial matching engine, `_clinical_trial_matching_engine.py`, will only operate if the following requirements are met.

Programs required

- Google Chrome
- ChromeDriver
    - Ensure that the version of ChromeDriver present in this program is compatible with your current version of Google Chrome. To find your current version of Chrome:
        i. Click on the 3 vertical dots at the far right side of your Chrome toolbar
        ii. Hover mouse pointer over the 'Help' menu
        iii. Click 'About Google Chrome'
        iv. Find Google Chrome version in the tab that opens
    - Save file as '`chromedriver.exe`' and save in primary directory (see 'File structure' section)
- Anaconda3
    - Spyder 4.2.5+

File structure required

- [Primary Directory]
    - **_clinical_trial_matching_engine.py**
    - clinical_trial_search.py
    - hospital_score.py
    - census.py
    - hospitals.py
    - cdc.py
    - census_score.py
    - chromedriver.exe
    - prostate_r.csv
    - nonhodgkinlymphoma_r.csv
    - leukemia_r.csv
    - corpus_r.csv
    - colonandRectum_r.csv
    - lungbroh_r.csv
    - pancrease_r.csv
    - femalebreast_r.csv
    - liverandbillduct_r.csv
    - ovary_r.csv
    - Hospitals.csv

Python packages required

- pandas
- json
- requests
- datetime
- selenium
  - To install, open Anaconda Prompt and execute `conda install selenium` in the command line

Script documentation

The main script, `_clinical_trial_matching_engine.py`, references 6 component scripts containing the functions necessary to execute this program. Each component script is documented here, along with the functions those scripts contain.

census_score.py

Inputs desired demographic information and outputs a “Census Score” based on demographic match to desired conditions for each county in the United States

- Libraries required
  - pandas

Functions

- **`census_score`**`(census, age, race, condition)`
  - This function provides a score to each county based off the age, race, and condition selection of the user.
    - A specific county population column is selected by the age and race input of the user.
    - EX: if 'white' and '<18 years' of age is chosen, the census column 'White.Child' will be chosen.
    - The selected county population column is scored using the scorer() function below.
  - Parameters
    - *census* : DataFrame - The county census dataframe returned from census.py
    - *age* : int - An int from 0-3 that represents the age grouping the user chooses in script.py
    - *race* : int - An int from 0-5 that represents the race/ethnicity grouping the user chooses in script.py

        - *condition* : int - An int from 0-9 that represents the cancer type the user chooses in script.py
    - Returns
        - *Dateframe* : DataFrame - A dataframe with 'County, State' name, state name, COUNTYFIPS, and newly added census score.
- **`scorer`**`(x)`
    - Scoring criteria:
        - First round clinical trials average between 20 to 80 participants
        - Each cancer type has a rate of new singular cases (i.e. 1 new lung cancer case per 12500 people)
        - For a county to be likely to have 20 cancer participants, it must have a population of at least 20 * (the denominator of the rate of new singular cases)
        - The participant range of 20-80 is divided into 11 buckets of populations
        - Each county will fall in one of the buckets and receive a score from 0 to 1, incremented by 0.1 (The larger buckets receiving the highest score)
        - EX: if a county meets the minimum population needed for the likely hood of 80 participants, then it would recieve a score of 1
        - The county population that is scored against the buckets is determined by the age and race the user selected. Only the resulting column will used for scoring.
    - Parameters
        - *x* : Dataframe column value - The population of a subgroup of a county.
    - Returns
        - *numeric* : float - A score from 0 to 1 (incremented by 0.1)

census.py

Retreives data from US Census API and packages it for scoring by **`census_score.py`**

- Libraries required
    - pandas
    - requests

Functions

- **`census`**`()`
    - This function accesses the ACS5 2019 CENSUS data for U.S. county population counts based on age and race/ethnicity.
    - Returns
        - *census* : DataFrame - contains aggregated and disaggreagted population counts by race/ethinicity and age for each county.
            - Race/ethnicity groups include : white, Black, American Indian or Alaskan Native, Asian or Pacific Islander, Hispanic/Latino

        - Age groups: child (<18 years), adult (18 to 64 years), older adult (65+ years)
        - For each county, there is also a grand population total.
    - Note: The CENSUS API only allows for 50 variables per requests, so we had to run requests several times. The age variables were not pre-grouped according to our age needs so we had to perform calculations to get our age groupings. We also were going to include additional racial groups but they did not match with CDC data, so the code is commented out.
- **`extract_state`**`(a)`
    - Takes a str and splits it on the comma. Then takes the index 0 value and strips it of the left white space.
    - Parameters
        - *a* : str
    - Returns
        - *state_name* : str - The second word in a "Word, Word" pair.

cdc.py

See functional description

- Libraries required
    - pandas
- Files required
    - lungbroh_r.csv
    - femalebreast_r.csv
    - prostate_r.csv
    - colonandRectum_r.csv
    - pancrease_r.csv
    - liverandbillduct_r.csv
    - ovary_r.csv
    - leukemia_r.csv
    - nonhodgkinlymphoma_r.csv
    - corpus_r.csv

Functions

- **`findstate`**`(indication, gender, race)`
    - Generating a disease prevalence score for each state based on age-adjusted rate of cancer death (death per 100,000 people), and cancer death count (data source: https://gis.cdc.gov/Cancer/USCS/#/AtAGlance/).
        - The death rate and case count will be normalized by minus its minima value of all states then divided by the difference between maxima and minima value. The weight of cancer dealth rate and case count are considered equal (wieght = 0.5).

    - The formula for the scoring (CDC_Score):
        - Case Count = cancer death count
        - age-adjusted rate of cancer death = rate
    - CDC_Score = (0.5 * ((case count - case count minima value) \ (case count maxima value - case count minima value))) + (0.5 * ((rate - rate minima value) \ (rate maxima value - rate minima value)))
    - Parameters
        - *indication* : int
            - 1. Female Breast Cancer (Top 2)
            - 2. Corpus and Uterus, NOS (Top 10)
            - 3. Ovary (Top 7)
            - 4. Prostate (Top 3)
            - 5. Lung and Bronchus Cancer (Top 1)
            - 6. Colon and Rectum (Top 4)
            - 7. Non-Hodgkin Lymphoma (Top 9)
            - 8. Leukemias (Top 8)
            - 9. Pancreas (Top 5)
            - 10. Liver (Top 6)
        - *gender* : int - Gender options:
            - 1. Male and Female
            - 2. Male only
            - 3. Female only
        - *race* : int - Race options:
            - 1. All Races and Ehnicities
            - 2. White
            - 3. Black
            - 4. American Indian and Alaska Native
            - 5. Asian and Pacific Islander
            - 6. Hispanic
    - Returns
        - *cdc_f* : DataFrame - A dataframe containing final socring result of each state

hospitals.py

Receives a CSV dataset of hospitals as input, returns a scored version of this dataset as a DataFrame.

- Libraries required
    - pandas
- Files required
    - Hospitals.csv

Functions

- **`hospital`**`()`
  - This function creates a dataframe of hospitals and applies a **`bed_scorer`**`()` function that gives each hospital a normalized score (the more beds the higher score)
  - Returns
    - *Dataframe* : DataFrame - A datafram of hospitals with a score based of bed capacity.
- **`bed_scorer`**`(b)`
  - This function is applied to the hospital 'BEDS' column and returns a normalized score based off the number of beds.
  - Parameters
    - *b* : int - hospital ‘BEDS’ value, represents the number of beds each hospital has.
  - Returns
    - *score* : float - A normalized score.

hospital_score.py

See functional description

- Libraries required
  - pandas

Functions

- **`hospital_scorer`**`(df, condition, age)`
  - This function provides an additional score to hospitals based off the age and condition chosen.
    - If the cancer chosen is one of the first three, hospitals of type 'WOMEN' receive an additional score of 100.
    - If the cancer chosen is 'Prostate', hospitals of type 'WOMEN' receive a score of -100.
    - If the age group '<18 years' is chosen, hospitals of type 'CHILDREN' receive an additional score of 100.
  - Parameters
    - df : DataFrame - Hospital dataframe.
    - *condition* : int - An integer representing a cancer type selected by the user.
    - *age* : int - An integer representing an age group selected by the user.
  - Returns
    - *df* : DataFrame - The hospital dataframe returned with an additional score column.

clinical_trial_search.py

Receives a list of hospitals as input, returns the hospital list along with each hospital's count of search results returned for a given condition in the ClinicalTrials.gov database

- Libraries required
    - pandas
    - json
    - selenium
- Files required
    - Chrome driver for user's version of Google Chrome
    - Hospitals.csv
- Programs required
    - Google Chrome

Functions

- **`expr_string`**`(condition, hospital)`
    - Generates a search ("expr") string for use in the clinicaltrials.gov API demo page, based on the desired condition and hospital 'https://clinicaltrials.gov/api/gui/demo/simple_study_fields'
    - Parameters
        - *condition* : string - medical condition to search.
        - *hospital* : string - Hospital (at which search result clinical trials are based) to search for.
    - Returns
        - *expr_string* : string - A formatted search string for input into the "expr" field of the clinicaltrials.gov API demo page.

- **`obtain_study_count`**`(clinical_trial_search_results)`
    - Obtains the number of search results found from a clinicaltrials.gov JSON API search result string
    - Parameters
        - *clinical_trial_search_results* : string - Raw returned JSON-format search results.
    - Returns
        - *study_count* : int - The number of search results found.