GeoMatch Data Scientist Technical Assessment

This assessment contains 2 problems. Problem 1 contains multiple sections.

Problem 1

Note: For this question, you need to use R but can use any package you might need.

Introduction:

Refugees arriving in a country are assigned to locations by a refugee agency, with a placement officer determining the most suitable location for each refugee family (or "case").

Data Description:

The dataset `cohort_dat_assess.RDS` contains information on 1,740 adult refugees from 1,029 cases placed in 50 locations. The objective is to explore how an algorithmic-assisted placement would differ from the current placements in the dataset ('status-quo placements').

Please note that some of the variables below are at the individual level (each person_id will have the same unique value for this variable), while others are at the case level (each case_id will have the same unique value for this variable). Individual level variables are in `orange` below, case level in `purple`.

- `person_id`: Unique identifier for each adult refugee.
- `case_id`: Unique identifier for each case (usually a family).
- `arrival_location_id`: Unique identifier for the location where the case was placed by the placement officer.
- `case_size`: Total number of persons in the case (including minors not listed in `cohort_dat_assess.RDS`).

- `ArrivalDate`: Arrival date at the location.
- `Nationality_norm`: Nationality.
- `sex`: Sex.
- `free_case`: indicates whether a case has family ties in the country, and has to be placed in the location of their tie (0) or can be placed anywhere (1).
- `hard_singles_male`: 1 if the case requires a location accommodating single males; 0 otherwise.
- `hard_singles_female`: 1 if the case requires a location accommodating single females; 0 otherwise.
- `hard_spf`: 1 if the case requires a location accommodating single parents; 0 otherwise.
- `case_placement_order`: Order in which each case was placed (1 for the first placement, 2 for the second, etc.).
- `arrival_location_order`: Order of locations to be used to break ties (see more details in question 3 below). 1 should be favored over 2 in case of a tie, etc.

Your task is to code an algorithm that smooths out arrivals across locations. We want to avoid one location receiving too many people in a short period of time, and also avoid locations going too long without receiving any people. Moreover, your algorithm will have to accommodate some location- and case- specific constraints: locations have a capacity quota that describes how many persons they can take in total, and some locations can't accommodate certain types of cases.

Q1: Location Capacity and Rate.

When doing the algorithmic placements, we will assume that each location's capacity quota is equal to the total number of refugees they received under the status quo placements.

- Load `cohort_dat_assess.RDS`.
- Compute each location's capacity quota based on the total number of refugees received. Note that you only have adults, not minors, in the dataset, and therefore you should use the `'case_size'` variable to compute the total number of refugees received.
- Calculate each location's "rate" as the total capacity across all locations divided by its capacity quota. Conceptually, a rate of 10 means that in a perfect,

constraint-free algorithmic placement, the location would receive 1 individual every 10 individuals placed elsewhere.

A. **What is the average capacity across all locations?**
B. **What is the average rate across all locations?**

Q2: Eligibility Constraints

Load `Aff_Nationality.csv` and `Aff_CaseType.csv`. They contain information about the constraints that apply.

- A location can accommodate a nationality only if it is listed in the `Aff_Nationality.csv` file under this location.
- `Aff_CaseType.csv` provides information on 3 constraints: `SingleParentFamilies`, `SingleIndividualMale` and `SingleIndividualFemale`. The file indicates whether each location can accommodate such cases. If `Accepted=="Yes"`, this location can accommodate such cases.

Note: These constraints only apply to cases with no existing family tie in the country, i.e., cases that can be placed anywhere (`free_case == 1`). So for this question, please focus only on free cases.

Prepare a dataset that indicates, for each free case and location combination, whether this location can accommodate this case. In order to accommodate a case, a location needs to be able to accommodate *every* nationality in the case, and, if the case has a specific family type (single male, single female, or single parent), the location needs to be able to accommodate it as well. Single parents and singles cases are already identified in the dataset (`hard_singles_male, hard_singles_female, hard_spf`). We call a location "eligible" for a given case if it can accommodate all of the case's characteristics.

For each free case, compute the number of eligible locations and then answer the following questions.

A. **What is the average number of locations eligible across all free cases?**
B. **What is the minimum number of locations eligible across all free cases?**
C. **What is the maximum number of locations eligible across all free cases?**

Q3: Algorithmic Placement

You are now ready to code your placement algorithm and simulate a counterfactual algorithmic-assisted placement.

To code your algorithm you need to keep track of location capacity and buildup.

Location Capacity: The algorithm should try to ensure that each location does not receive more persons than it received under the status quo placement (i.e. the capacity quotas you computed in Q1). So as you place cases, you need to keep track of the remaining number of available slots in each location. Note again that capacities refer to the total number of persons, adults and minors (e.g., if a location only has one slot left, a free case with `case_size == 2` should not be placed there).

Location Buildup: Buildup captures how overwhelmed or idle a location is. A buildup of 10 means that we should place 10 individuals elsewhere before we go back to this location. A buildup of -10 means that we should have placed an individual at this location 10 placements ago.

Set buildup to 0 for every location at the start of your algorithm. Then update the buildup after each placement using the following formula:

- For the location that received this placement: new buildup = max(0, (old buildup - size of the case)) + (size of the case * rate of the location)
- For the locations that did not receive the placement: new buildup = old buildup - size of the case

Your algorithm must also accommodate the following features:

- You must place cases in the same order as they were placed in the dataset (i.e. the order specified in the ‘case_placement_order’ variable).
- If the case is free (`free_case == 1`), you will place the case in the eligible location with the lowest buildup at the time of placement.

    - You can only place a free case in an eligible location, and in a location with enough remaining capacity for this case.
    - If multiple eligible locations are tied for the lowest buildup, then place the case in the location with the lowest `arrival_location_order` (i.e., place at 1 over 2 if they are tied for lowest buildup)
    - If there is no eligible location left, place the case at the location with the lowest buildup at the time of placement (i.e. ignore the constraints)
    - After placing a free case, you need to update capacities and buildup at all locations (using the formulas above).
- If a case has family ties in the country (`free_case == 0`), this case needs to be placed in the same location that it was actually placed in the status quo placement (`arrival_location_id`). This is because these refugees have family ties in that location and cannot be placed anywhere else.
    - You need to place a tie case in its status quo location even if this location is already at capacity.
    - For these placements you still need to update the capacity and buildup at all locations. In other words, cases with ties count for the capacity and buildup just like free cases.

After you run your algorithm, compute a table that compares the total number of people placed at each location for the status quo placements and your algorithmic placements.

A. **Which location gained the most extra people (adults and children) with your algorithmic placements, compared to the status quo placements? How many extra people did this location receive?**

B. **Which location lost the most extra people from the algorithmic placements, and how many fewer people did it receive?**

C. **Did your algorithmic placements meet the capacity quotas from Q1 for all locations or did locations go over their quotas? If so, can you explain the general idea for why this happened?**

D. **For location "`19334d00`" plot the cumulative sum of the number of arrived persons at the location against the case placement order and create two lines: one for the status quo placement and one for your algorithmic placements. Did your algorithmic placement smooth out the arrivals compared to the status quo?**

E. What is the average buildup across all locations and placements under your algorithmic placements? Include all the buildups you had before every placement you made: i.e., include the zeros from the first placement in the average, and do not update the buildup after placing the last case in the sequence.

F. How would you expect the algorithmic average buildup to compare to the status quo average buildup?

G. Can you think of elements in the composition of the cohort of cases that would increase the difference between the algorithmic and status quo average buildups?

H. Do you have any suggestions for how this algorithm could be improved? Do you have any suggestions for how the metric we proposed (average buildup) could be improved?

I. [Bonus only if you have time: what was the average buildup under the status quo placements?]

Problem 2

Note: For this question, you may ONLY use functions from the base, tidyverse, data.table and stats packages to complete this task; i.e. you may NOT load any additional packages in R or other software program that you may be using.

The data file `dem_dat.csv` contains hypothetical data on the arrival times of immigrants into a particular country. For each immigrant (each row in `dem_dat.csv`), you will need to compute the total number of months that they had at least one job over the course of their first 3 years in the country. To do this, you will also make use of `job_dat.csv`.

Info on the data:

- `dem_dat.csv` contains observations for unique immigrants. `YM_in` is their arrival year-month in the country, and `YM_out` is the time in which they leave the country.
- `job_dat.csv` contains job "episodes," with start and end year-months indicated. If an individual does not have any entries in `job_dat.csv`, then they have no job episodes and should be counted as zero months worked. Also if individuals have job spells before their arrival in the country those should not be counted.

A. **What are the mean, median, and maximum number of total months worked for this sample of immigrants in their first 3 years in the country?**