Let me show you full task.

1. Processing survey data

This data provide us with the list of students and for each student the data contains the semester number and the time the student made a survey in each listed semester, and then list of up to twenty nominations of partners.

Some students skipped making survey in some semesters, the max number of surveys is four (semesters numbers for them are 2, 3, 5, and 6),

From this we need to create a data structure with

* 1. Scalars: id of the students, number of semester she took surveys and then a few vectors with the length equal to the number of semesters
  2. Vectors: the first vector should contain the semester number in which survey was taken, then a vector with time at which survey was taken, and final vector should have the number of partners listed in the list of nominations for each semester.
  3. Matrix, finally there will be one vector for each semester with the list of partners student chose in survey

Example

id 10060 number of semesters 4,

Vectors semester numbers 2,3,5,6; times 250000, 500000, 700000, 900000, partners 2, 3, 4, 2

Matrix of lists [2] 1009, 1123; [3] 1123, 1184, 1890; [5] 1123;1009, 1184, 1890; [6] 1009, 1184

It’s not big structure below 100 number per id (at most 80 ids in lists plus additional data) so 18k data items so you can keep it.

Solution, you program goes first reading survey data and creates structure in memory for each student and then program moves to process cogsnet data

1. Processing cosgsnet data

First for each pair (student, partner) we want to compute the signal from calls.

So in a loop, go through the cogsnet data and read all the data about one student. It helps that the student are listed in the same order in survey and cogsnet data and all calls are listed chronologically for each pair of students as are times of surveys.

So now in a big loop you do

For each student:

* 1. Read all the data about the student and create a structure in memory that has scalar student id has vectors: vector of partner ids for call, vector of number of calls for this partner, and matrix of calls, for each id in partner vector you have a vector of times of call
  2. After you finish creating this data you stop reading the file and start processing signals. The code below needs to be nested in three loops setting parameters for finding solution.
  3. In a loop for each partner
     1. First read the time of the first survey for this student from survey data into real surverytime variable
     2. For each call this partner made do if calltime > surveytime then
        1. Produce signal for survey sigsurvey = signal \* decay between the time of surverytime and the previous call time and store in the structure with top twenty partners for this  student together with partner id (remember the smallest value in top twenty and only merge sort this result into top twenty if it is larger than smallest item there).
        2. If it was not the last survey time, replace current survey time with the nest survey time and continue, otherwise break from processing calls for this partner and move to next compute new signal by (i) compute decay of the old signal into signal, if it is less than theta, set signal to 0; and then (ii) compute signal as

mu+(1-mu)\*signal

                end of if

            end of for each call

     end of loop for each partner

* 1. Here you are ready to produce signals for top twenty partners for this student so we can check correctness for modeling.

            For true modeling run over large number of parameter points, we can only store top 20 list of id's for this student for each semester in which this student took survey

* 1. Now for each semester you compare the list obtained for this student with the list he provided in the survey and compute Jaccard metric for these two lists (for example using sort-merge of both lists and keeping duplicates and then Jaccard metric is the ratio of number of duplicates divided by the number of unique entries. If this is not the first student, you add this Jaccard metric to the Jaccard sum for all students, this sim is the only value you need to remember for further processing.

end of the loop for parameter values

end of the loop for student

You store the sums of Jaccard for each parameter setting, say 1 million numbers, so 4 or 8 Mbytes quite feasible.