Assignment 4 – Serosurvey

Q1 - Likelihood

A likelihood function was created. This took in the state s, the test suite t, and a given outcome y, and returned the probability p of getting that outcome.

s and t were each python lists corresponding to the RAT, RT-PCR and IgG tests. y was taken as a tuple. This was done so that it could contain the string “NA” which denoted no outcome.

Initial probability value was initialized to 1. By running a while loop, the likelihood was calculated independently for each test (namely RAT, RT_PCR and IgG) and multiplied to get overall likelihood.

The following was done for each test-

- Firstly, outcomes incompatible with the test suite were eliminated – if the test had not been conducted, but the outcome displayed an integer, or if the test had been conducted but the outcome was “NA”. These cases were assigned probability 0.
- For the remaining compatible cases, the probability was calculated based on the state and the test outcome, using the sensitivity and specificity values of that particular test
- The following values of sensitivity and specificity were used –
  RAT: sensitivity 0.5, specificity 0.975
  RT-PCR: sensitivity 0.95, specificity 0.97
  IgG: sensitivity 0.921, specificity 0.977
  (These values were taken from the reference given in the discussion forum)
- Thus, depending on the sensitivity and specificity values of each type of test, the following conditional probability expressions were implemented using if statements.
    1. $P(y=1|s=1)$=sensitivity
    2. $P(y=0|s=1)$=1-sensitivity
    3. $P(y=0|s=0)$=specificity
    4. $P(y=0|s=1)$=1-specificity

Q2– Creating a population and assigning test outcomes

The 4 possible states were named as 1,2,3,4. Using the random library of numpy, the states array was generated which contained the states assigned to each of the 70,000,000 people based on the probabilities given in the disease prevalence vector p = (0.1, 0.32, 0.01, 0.57). Now, from this array of 70 million people, 16000 entries were randomly sampled and stored in the state_index array.

Tuples s and t were initialized. These would contain lists corresponding to the state and test suite of each of the 16000 individuals. (Note, these are different from the s and t used in the likelihood function).

A loop was run to iterate for each individual. Based on the probability given they were assigned into one of two test suites – [0,1,1] (40%) or [0,0,1] (60%).

Based on the value in the state index array, the following state vectors were assigned:

- State 1 – [1,1,0]
- State 2 – [0,0,1]
- State 3 – [1,1,1]

- State 4 – [0,0,0]

This order was according to the data in the excel file provided.

Next, the tuple y was initialized. This would be a tuple of tuples corresponding to the test outcome of each of the 16000 individuals.

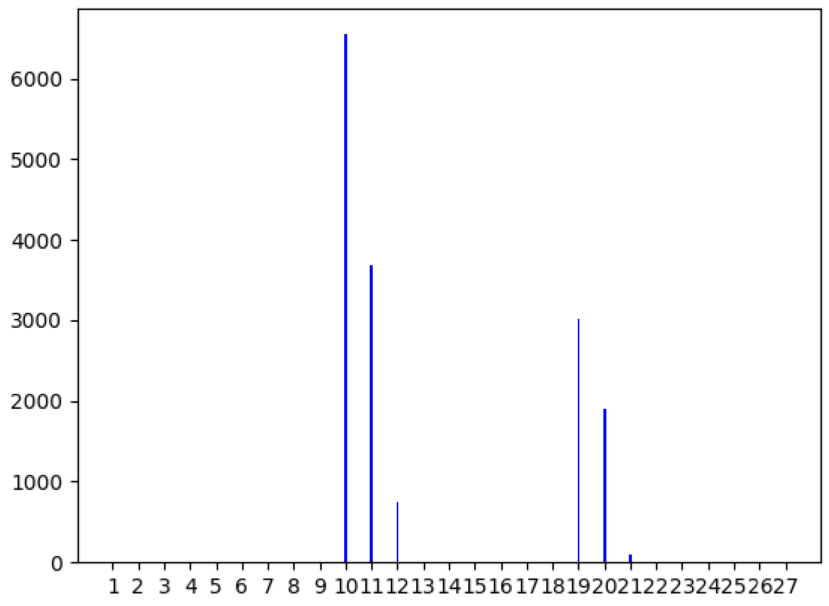
A loop was run, and for each individual the likelihood of all 27 outcomes were calculated based on the state and test suite found earlier. They were then assigned an outcome based on these probabilities, which was stored in y.

Note: Since the outcomes have been implemented as “NA” for no outcome, the set of outcomes were not directly imported from the csv file because they have been represented as empty entries there. However, the same order of the outcomes 1 to 27 in the data file has been maintained.

Q3 – Generating the Histogram

The number of individuals having each of the 27 outcomes were found by counting from the tuple y.

The following histogram was generated



Here 1 to 27 denote all the possible test outcomes. They are in the same order as the data in the excel file:

- 1=("NA","NA","NA")
- 2=("NA",0,"NA")
- 3=("NA",1,"NA")
- 4=(0,"NA","NA")
- 5=(1,"NA","NA")
- 6=(0,0,"NA")
- 7=(0,1,"NA")
- 8= (1,0,"NA")

- 9=(1,1,"NA")
- 10=("NA","NA",0)
- 11=("NA",0,0)
- 12=("NA",1,0)
- 13=(0,"NA",0)
- 14=(1,"NA",0)
- 15=(0,0,0)
- 16=(0,1,0)
- 17=(1,0,0)
- 18=(1,1,0)
- 19=("NA","NA",1)
- 20=("NA",0,1)
- 21=("NA",1,1)
- 22=(0,"NA",1)
- 23=(1,"NA",1)
- 24=(0,0,1)
- 25=(0,1,1)
- 26=(1,0,1)
- 27=(1,1,1)

As we can see from the histogram, only 6 of the 27 outcomes can occur. This is as expected based on the test suites prescribed and the population state. In the test suite vectors [0,1,1] and [0,0,1], we see that the RAT test is never done and the IgG test is always done. Thus the outcome of the Rat has to be “NA”, while that of IgG has to be either 0 or 1. This leaves us with 1*3*2=6 possible outcomes. These are:

- 10=("NA","NA",0)
- 11=("NA",0,0)
- 12=("NA",1,0)
- 19=("NA","NA",1)
- 20=("NA",0,1)
- 21=("NA",1,1)

Furthermore, the probability of 10 and 19 are higher since the RT-PCR is more likely to be not conducted (60%) than conducted (40%).

Q4 – Estimating Disease Prevalence using MLE

Based on the generated outcomes in Q2, and the test suites assigned, the likehihood expression was found. This depended on the disease prevalence vector [p1,p2,p3,1-p1-p2-p3].

This expression was maximized using gradient ascent.

Firstly, the likelihood function was vectorized. This was done because for each iteration of the gradient ascent, the likelihood had to be found for each of the 16000 individuals. Hence it took a lot of time to run the gradient ascent as a loop. Thus it was vectorized to shorten runtime.

The new function likelihood2 is a vectorized version of the likelihood function in q1. It takes in 3 arguments: s-state vector, tuple_t – tuple of lists of test suites for each of the 16000 individuals, and tuple_s – tuple of tuples containing outcomes of each of the 16000 individuals. The tuples y and t were converted into numpy arrays. The string NA was replaced by -1 to allow for the use of the equality operator. Then, the same if conditions used in the earlier likelihood function were implemented using numpy array indexing. Thus, this likelihood 2 function takes in the state, test suite, and outcomes for all 16000 individuals and returns a vector containing the likelihood for each of those 16000 individuals.

Now, gradient ascent was implemented using Pytorch. Using the likelihood2 function, the vector of likelihoods of all 16000 individuals was computed for each of the 4 states. These were converted to torch tensors and stored as l1, l2, l3 and l4. For example, l1 is a tensor of likelihoods of the individuals being in state1- [1,1,0]

An initial estimate of disease prevalence vectors p1=0.2, p2=0.1, p3=0.4 was chosen. These were also implemented as tensors.

The overall likelihood was found from the expression p1*l1 +p2*l2 + p3*l3 + (1-p1-p2-p3)*l4.

The log was taken and then it was summed. This gave the expression needed to be maximized. Backpropagation was used to compute the gradients. A learning rate of 10^-7 was chosen based on trial and error. 5000 runs of gradient ascent were conducted to converge to the following result of disease prevalence vector –

[p1=0.1032, p2=0.3166, p3=0.0112, p4=0.569]

We can see that this is quite close to the disease prevalence estimate we used in Q2 – (0.1, 0.32, 0.01, 0.57). Thus, we can conclude that the MLE procedure was successful.

Note: Different runs of the code gave slightly different results due to inherent randomness of the gradient ascent, however convergence always occurred, and the result was always close to the expected value.