Midterm
Computer Science
Fall 2015
B565

Suhas Jagadish
Sunday, October 25, 9:00 p.m.

October 25, 2015

All the work herein is solely mine

Directions

For question 1, you'll provide as much explanation as you deem necessary to solve the problem of the client. For 2, you'll simply supply the algorithm and discussion. For 3, you'll provide the mathematics and answer. A bonus of *up to* 50pts *may* be added for presentation and readability. You cannot discuss the exam with anyone. If you think you've found something wrong with any of the problems, *fix them* and write about. No changes will be made to the exam while it's being worked on. You are free to use materials, but must *explicitly* site them if they contribute significantly to your answer. Most importantly, think creatively and positively as you work through this. Good luck!

1 Application [250pts]

Assume you work for `Filmflix.com`, an online movie rental business. The data from `Filmflix.com` is found in Table **??**. Read the dialogue and answer appropriately.

1. The client says to you, "We'd like to understand our customers broadly from their tastes in movies. What do you suggest?"
   "I'll do a k-means and show you the results!

   We first categorize the datasets into tables. The table used for kmeans is attached as "data.csv".
   This table contains customer to genre relation. Each row represents the set of movies(in terms of genre) the customer rents. For example, customer 1 rents 5 movies of the genre 'romance'. This representation makes sense when we calculate euclidean distance between 2 customers. If the distance is small, that means customers are closely related to each other in their taste of movies.
   Now I use my kmeans code which was submitted in assignment 2 and pass this dataset. The results for varying number of centroids is shown below -

   For k=2, we get 2 clusters where first cluster contains 11 customers and second cluster contains 3 customers.

Cluster	Customers
C1	2,3,4,5,6,7,9,11,12,13 and 14
C2	1,8 and 10

The results indicate that customers in each cluster watch similar genre of movies and hence have similar taste in movies.

For k=3, we get 3 clusters where first cluster contains 2 customers, second cluster contains 9 customers and third cluster contains 3 customers.

Cluster	Customers
C1	6 and 11
C2	1,3,4,5,7,8,12,13 and 14
C2	2,9 and 10

As we increase the "k" value, we refine our customers' taste in movies. The R code for kmeans is attached as "kmeans.R"

2. The client says to you, "I was reading about association rules. These rules could help us promote certain movies. What does the data suggest?"
"Well, we can do either genre or movies–or even both. Why don't I generate a couple of rules for each individually and suggest some promotions."

In order to promote movies and genres individually, we first create 2 tables, one with customer-movie relationship and other with customer-genre relationship. This time we just record 1s and 0s since the weight on the data is not needed. The files are attached as "data1.csv" and "data_knn.csv".
We install the "arules" package and run "apriori" algorithm in R which generates association rules.
We pass our tables to the algorithm and define the support and confidence.
Support of a rule indicates how frequently the items in the rule occur together. Confidence is the conditional probability that the consequent will occur given the occurrence of antecedent.

So with these parameters defined, we run the apriori algorithm, first for genres. The R code is attached as "apriori_genre.R". Based on the values of support and confidence, below are the 2 rules which can be used for promotion -

$$Action => Classic$$
$$\{ScienceFiction, Horror\} => Comedy$$

So the apriori algorithm suggests that whenever a customer rents movies of the genre "Action", he/she has a higher probability of renting "Classic" movies.
Similarly when the customer rents "Science Fiction" and "Horror" movies, he/she would rent "Comedy" movies.

Now we run the apriori algorithm for movies. The R code is attached as "apriori_movies.R". Below 2 rules can be used for promotion -

$$Movie1 => Movie3$$
$$\{Movie2, Movie3\} => Movie1$$

The above rules specify that if a customer rents movie 1, he/she is most probable to rent movie 3. And if the customer rents movie 2 and 3, he/she has a high probability of renting movie 1.
This gives the client a clear understanding on how to promote the movies, based on genre and movies itself.

3. The client says to you, "My favorite movies are 2,5,8, and 9." I'm curious, what customers is nearest to me in my choices?"
"I suggest a *knn*. I'll find out!"

knn algorithm takes the actual dataset, test dataset, class label and number of nearest neighbors as arguments and outputs the nearest class label.
So in this case, we use the client's entry as the test dataset against the actual dataset that contains customer-movie relationship.
In order to run the *knn* algorithm in R, we have to first import the package "class". In my example, I'm passing the label set to be 1 to 14(customers 1 to 14). So depending on the number of nearest neighbors chosen, I may get different label as the output. For example,

When I choose k=1, I get the output as label=14. This makes sense when we actually look at the data. Customer 14 has rented movies 5,2 and 8 and he/she will be the nearest customer for our client.

When I choose k=3, I get the different labels as output such as 4 and 8. Since I've defined my labels to be 1 through 14, 4 and 8 indicates customer 4 and 8.
This makes sense, because customer 4 and 8 belong to the same cluster in our initial kmeans result(with k=3, we had customers 4 and 8 belong to cluster 2).

Interpreting the results of k=3, we say that our client's favorite movies are 2,5,8 and 9 and the customers nearest to our client are customer 4(movie 2) and 8(movie 5,4 and 9).

In a similar way, we can find the result altering the number of nearest neighbors. The R code is attached as "knn.R".

4. The client says to you, "We have a recommendation process–we ask the genres you like and we suggest the movie; but, it's not very scientific. If I like *Action* and *Drama* what are the three best recommendations for movies?"
"We can use a Näive Bayes for this. I'll let you know."

Näive Bayes classifier uses the Bayes rule to calculate the posterior probability of an event(s) given the likelihood and prior probability. Mathematically,

$$P(outcome|evidence) = (P(likelihood\ of\ evidence) * Prior\ probability\ of\ outcome)/P(evidence)$$

So in our case, we need to find $P(M_1|(Action \cap Drama))$.....$P(M_{10}|(Action \cap Drama))$ and pick the 3 highest probabilities for 3 best recommendations.
We use our movies-genre table for calculating each probability. In order to avoid 0 values, we assume 0=0.01 in our calculations. The calculation is shown below-

$P(M_1|(Action \cap Drama)) = P((Action \cap Drama) * P(M_1) = (0.01 * 0.01) * (1/10) =$ **0.00001**
(Movie 1 is mapped to only romance and science fiction, hence probability of action and drama is 0.01. Probability of Movie 1 is 1/10 and it will be same for all the movies)

$P(M_2|(Action \cap Drama)) = P((Action \cap Drama) * P(M_2) = (0.3 * 0.01) * (1/10) =$ **0.0003**
$P(M_3|(Action \cap Drama)) = P((Action \cap Drama) * P(M_3) = (0.3 * 0.01) * (1/10) =$ **0.0003**
$P(M_4|(Action \cap Drama)) = P((Action \cap Drama) * P(M_4) = (0.25 * 0.01) * (1/10) =$ **0.00025**
$P(M_5|(Action \cap Drama)) = P((Action \cap Drama) * P(M_5) = (0.3 * 0.3) * (1/10) =$ **0.009**
$P(M_6|(Action \cap Drama)) = P((Action \cap Drama) * P(M_6) = (0.3 * 0.01) * (1/10) =$ **0.0003**
$P(M_7|(Action \cap Drama)) = P((Action \cap Drama) * P(M_7) = (0.25 * 0.25) * (1/10) =$ **0.00625**
$P(M_8|(Action \cap Drama)) = P((Action \cap Drama) * P(M_8) = (0.01 * 0.01) * (1/10) =$ **0.00001**

$P(M_9|(Action \cap Drama)) = P((Action \cap Drama) * P(M_9) = (0.01 * 0.5) * (1/10) =$ **0.0005**

$P(M_{10}|(Action \cap Drama)) = P((Action \cap Drama) * P(M_{10}) = (0.01 * 0.01) * (1/10) =$ **0.00001**

As we can see from the above probabilities, movie 5,7 and 9 have higher probabilities of being Action and Drama. Hence I would suggest movies 5,7 and 9 to the client where his taste lies in Action and Drama.

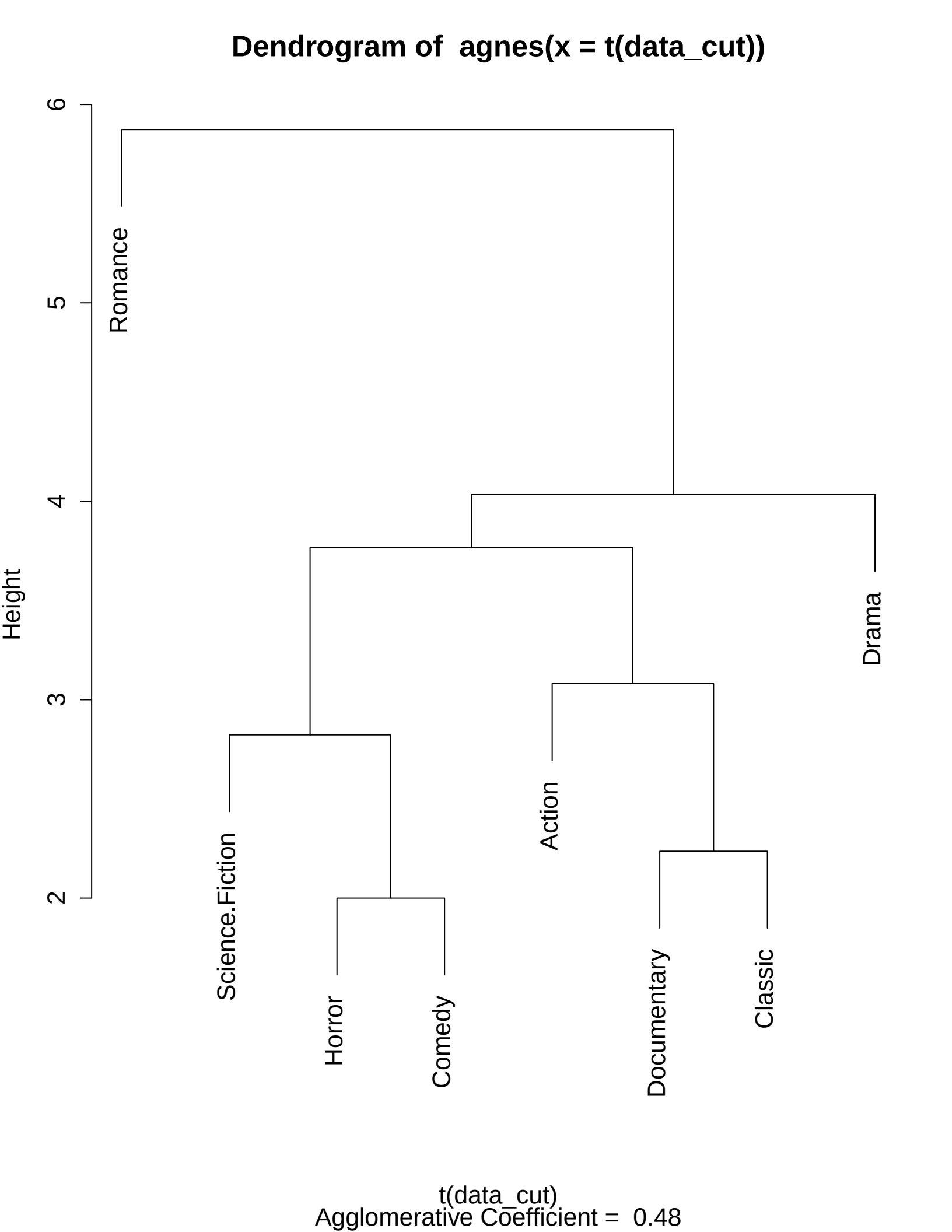
The same result can be obtained by running naive bayes in R using the "e1071" package. I wanted to express it mathematically so I chose to write down all the probabilities.

5. The client says to you, "Right now, the genres are kind of unrelated to one another. I wonder if you could build a tree that shows how they are related from viewers' points of view?"
"I'll do aglommerative clustering, and we'll see if we can interpret the tree."

Agglomerative clustering helps us to visualize the customer-movie(genre) hierarchically. From customer's point of view, we need to depict what genres are related to one another and this helps the client to understand their customer needs better.
We use the package "cluster" and run our agglomerative clustering on our genre-movie table. The resulting tree is shown below.

The tree at the leaf nodes contains horror-comedy and documentary-classic pair. This means that our customers watch horror and comedy together or documentary and classic together.
As we go up the tree, we get other genres related to one another. For example, customers who watch horror and comedy, most probably watch Science Fiction.
As we reach the top of the tree we get "romance" which indicates that most of our customers watch romantic movies, no matter what other genre they watch. This also makes sense from the data, since the number of romantic movies rented is high in number.

Agglomerative clustering follows bottom-up approach where it clusters 2 nearest datasets(customers in our case) and moves up the tree to form one big cluster.
So the tree below suggests how the genres are related to one another from viewer's point of view.


Dendrogram of agnes(x = t(data_cut))
Height
6
5
4
3
2
Romance
Drama
Action
Science.Fiction
Horror
Comedy
Documentary
Classic
t(data_cut)
Agglomerative Coefficient = 0.48

2 Equivocation: k, ℓ-means Algorithm [100pts]

This problem asks you to modify the k-means algorithm to k, ℓ. The ℓ is the best number of centroids that the datum matches. So, $4, 2$-means each datum is matched to the 2 closest centroids. You can assume that we're using *average* for the best representative. Complete the algorithm below for k, ℓ-means. The final result *must* be an actual partition.

1: **ALGORITHM** `k,l-means`
2: **INPUT** (data Δ, distance $d : \Delta^2 \to \mathbb{R}_{\geq 0}$, centoid number k, Closest Matches ℓ, threshold τ)
3: **OUTPUT** (Set of centoids $c_1, c_2, \ldots, c_k$)
4: Assume centroid is structure $c = (v \in DOM(\Delta), B \subseteq \Delta)$
5: $c.v$ is the centroid value and $c.B$ is the set of nearest points.
6: τ is a percentage change from previous centroids
7: For example, $\{c_1, c_2, \ldots, c_k\}$ is previous and $\{d_1, d_2, \ldots, d_k\}$ is current
8: Total difference is $\Sigma_i \Sigma_j d(c_i, d_j)$
9: $Dom(\Delta)$ denotes domain of object.
10: $i = 0$ ▷ Initialize iterate where superscript is iteration
11: **for** $j = 1, k$ **do** ▷ Initialize Centroids
12: $c_j^i.v \leftarrow random(Dom(\Delta))$
13: $c_j^i.B \leftarrow \emptyset$
14: **end for**
15: $f_i = \Sigma_{j=1}^k \Sigma_{\ell=1}^k d(c_j^i.v, random(Dom(\Delta))$ ▷ Bootstrap difference between past centroids and current
16: **repeat**
17: $i \leftarrow i + 1$
18: **for** $\delta \in \Delta$ **do**
19: $c1_j^i.B \leftarrow c.B \cup \{\delta\}$, where $\min_{c1_j^i}\{d(\delta, c_j^i.v)\}$
20: $c2_j^i.B \leftarrow c.B \cup \{\delta\}$, where $\min_{c2_j^i}\{(d(\delta, c_j^i.v) \neq min_{c1_j^i})\}$ ▷ Minimum of the remaining entries.
21: $c_j^i.B \leftarrow random(c1_j^i.B, c2_j^i.B)$ ▷ Random centroid out of the 2 closest is assigned to datapoint
22: **end for**
23: **for** $j = 1, k$ **do**
24: $c_j^i.v \leftarrow ave(c_j^i.B)$ ▷ Update centroid to be *best* representative of nearest data
25: $c_j^i.B \leftarrow \emptyset$ ▷ ave is easiest representative
26: **end for**
27: $f_i \leftarrow \Sigma_{j=1}^k \Sigma_{\ell=1}^k d(c_j^i.v, c_\ell^{i-1})$ ▷ Find the difference between previous centroids and current centroids
28: **until** $(|f_i - f_{i-1}| < \tau(f_{i-1}))$
29: **return** $(c_1^i, c_2^i, \ldots, c_3^i)$

Modify your k-means code to allow $\ell = 2$ and rerun your analysis on the breast cancer data. Discuss your results with respect to your earlier results.
k, ℓ means algorithm differs from normal $k$means in that each dataset belong to ℓ best centroids.
For each data point, I intend to find the minimum and the next minimum distance to the centroids. So a datapoint can be mapped to the closest centroid and to the next closest centroid.
But we cannot assign a datapoint to 2 different centroids, this will form duplicate datapoints and hence will not result in actual partitions.

Hence we have to choose between the closest and the next closest centroid. This is the importance of k, ℓ means(or even called as Fuzzy clustering) where it doesn't assign a datapoint to a specific centroid but it assigns the weightage of each datapoint to the related centroids.

However in my case, I'm picking 1 out of the 2 centroids randomly. So my modified algorithm first computes the 2 closest centroids for each datapoint and picks a centroid randomly. **This way we end up in an actual partition at the end**. Complete code has been updated in the above algorithm (line 17 to 27).

Survey	
Q1: Do you own your home?	Q2: Do you own your car?
Yes	No
No	Yes
Maybe	No
⋮	⋮

Now, I ran this algorithm on our breast cancer data to find the PPV values. I've chosen k=2 and l=4 and the resulting PPV was **0.95**.

The high PPV value suggests that the closest and the next closest centroid has similar data(benign or malignant). This inturn shows that the clusters formed are pure, even if a datapoint is assigned to the second closest centroid.

The major advantage in using k, ℓ means is that it is suitable in case of overlapping clusters.
In terms of computational time, k-means is better than k, ℓ means. The quality of clustering is found to be almost similar in my case, but I would say k-means produces better clustering quality since it maps the datapoints to the closest centroid.

The R code for k, ℓ means is attached as "klmeans.R".

3 Connections [75pts]

After examining the results of the survey, you find that there are only three kinds of responses in the same proportions: (Yes, No), (No, Yes), and (Maybe, No).

1. Are the questions Q1 and Q2 statistically dependent?
   We will first calculate the individual probabilities of Q1 and Q2 and formulate it as a table.

Q1/Q2	Yes	No	Maybe
Yes	0	0.3	0
No	0.3	0	0.3

   From the table, we can derive individual probabilities of Q1 and Q2.
   $P(Yes_{Q1}) = 0.3, P(No_{Q1}) = 0.3, P(Maybe_{Q1}) = 0.3$
   $P(Yes_{Q2}) = 0.3, P(No_{Q2}) = 0.6$

   Considering the first pair, $(Q1|Yes)$ and $(Q2|No)$ Now, we say that Q1 and Q2 are statistically independent if the below relation holds good,

   $P(Yes_{Q1}\ and\ No_{Q12}) = P(Yes_{Q1}) * P(No_{Q2})$
   We know from the above table that $P(Yes_{Q1})$=0.3 and $P(No_{Q2})$=0.6 and $P(Yes_{Q1}\ and\ No_{Q12})$=0.3.
   Since $0.3 \neq 0.3 + 0.6$, we say that Q1 and Q2 are **statistically dependent**.

   Conversely, we can prove that Q1 and Q2 are **statistically dependent** using the below rule,

   $P(Yes_{Q1}\ and\ No_{Q12}) = P(Yes_{Q1}) * P(No_{Q2}|Yes_{Q1})$
   $P(No_{Q2}|Yes_{Q1}) = P(Yes_{Q1}\ and\ No_{Q12})/P(Yes_{Q1}) = 0.3/0.3 = 1$

Now since 0.3 = 0.3 * 1, we prove that $P(Yes_{Q1}\ and\ No_{Q12}) = P(Yes_{Q1}) * P(No_{Q2}|Yes_{Q1})$ and hence can state that Q1 and Q2 are **statistically dependent**.

Since we have proved that the pair $(Q1|Yes)$ and $(Q2|No)$ are **statistically dependent**, there is no need to prove the other 2 pairs explicitly. However if the proof is considered, both the other pairs would be **statistically dependent**.

2. Are the questions Q1 and Q2 statistically correlated?

   We use Pearson's correlation coefficient to find out if Q1 and Q2 are statistically correlated or not.

   Correlation coefficient = $covariance(Q1, Q2)/\sqrt{(variance(Q2) * variance(Q1))}$

   By definition, if Q1 and Q2 are independent, then covariance(Q1,Q2) must be 0. However in our case, we have proved that Q1 and Q2 are statistically dependent and hence **covariance cannot be 0**. Hence our correlation coefficient can now range between -1 to 1. Correlation of negative value indicates that when Q1 increases, Q2 decreases and vice versa. Correlation of positive value indicates that when Q1 increases, Q2 also increases and vice versa.

   We do not have sufficient dataset to calculate the actual correlation coefficient.

   So given the above rules, we can say that **Q1 is statistically correlated to Q2**, but we do not know to what extent they are correlated(positively or negatively).

NOTE: The survey table is displayed at the top of page 7.