CSCI-B 565 DATA MINING
Homework 3
Morning Class
Computer Science Core
Fall 2013
Indiana University

Debpriya Seal
debseal@indiana.edu

October 15, 2013

All the work herein is solely mine.

Problem One

Assume you have a data series $\Delta = a, b, a, b, b, a, ?, a, b, b$ where the ? is missing data. Explain how to replace this using uniform distribution, random distribution imputed from the data, and $B(p, n)$ .What are the sigificant differences between these three approaches?
Answer:

1. Uniform Distribution.
   Answer: Since we are given that the above distribution is unifrom. Then that means the chances of each is random variable is equal. Now, in the given distribution we have $P(a) = 4/9$ and $P(b) = 5/9$. So for the uniform distribution to hold the missing number should be a.

2. Random Distribution.
   Answer: Since we are given that the above distribution is random. However, looking at the data, clearly b appeared more number of times than a. So we can keeping that in mind we can say that the missing number is b.

3. Bernoulli Distribution.
   Answer: Since we are given that the above distribution is Bernoulli. Clearly, the data gives us the probability of getting a is $P(a) = 4/9$ and $P(b) = 5/9$. Hence, clearly based on the probability we can easily say that the missing value is b.

Problem Two: Medical Data

1. Write a reasonably formal statement that uses Δ. What is the size of Δ?
   Answer: The size of delta is 699 tuples. Below is the formal statement:

This is data is about the breast cancer. And looking at the data we few things pop up straight away. Firstly that the data has been binned. As all the feautres apart from SCN are between 1 to 10. This rarely happens in real life. Also, the data look pretty clean.The task in our hand is to cluster these data into malignant or benign. The size of data is not huge by any standard. Also, the number of attributes we have is 10, however, not considering SCN we have 9 attributes. Compared to the size, these many attributes is a great thing. For clustering I intend to use k-means algorithm as it one of most popular algorithm.Reason being is realitvely efficient with the running time complexity of $O(irck)$ where k is the number of clusters, i is the number of iterations, c is the number of features and r is the number of datapoints.

(a) Is this a large data set?
Answer: Well what is large is relative. What is large to me maynot be large to someone else. However, with huge RAM coming by. The defination of large has changed over time. Like the 3rd question given to us would be an example of large dataset. Since we have approximatly a million datapoints and 7K centroids. But I will go forward and say that it is not a large dataset by any means.

(b) Discuss the number of attributes with the datas size
Answer: For this problen statement the number of attributes is 9 and number of datapoints is 699. Looking deeper inot k-means algorithm(which we are going to implement). The heart of k-means has a $C \times \Delta$ operation, where C is the number of centroids(which again is 2). So if we see the total operation approximatly is $2 \times 699 = 1398$. And this in current computer speed is nothing. However, features does effect the result of the clustering algorithm. And you can see this clearly in the 5^thquestion. If the number of features decreasing the accuracy will also start to decrease.
Lastly, For our astronomy data it would be 700×0.7 million $=$ 70 million. Now that's huge by any standards.

2. Ignoring the Sample code number (SCN), how many attributes does Δ have?
Answer: Ignoring the SCN, there are 9 attributes. The last one Class is not an attribute as it is the Label y.

3. How many missing values are there? Give the SCNs for that have missing data. Remove the tuples that have missing data. Let Δ^* be a cleaned Δ: the tuples with the missing values are removed. R offers several ways to remove unknown data, though you are free to write your own code. Let $\Delta^m = \Delta - \Delta^*$. For each $d \in \Delta^m$, replace the unknown data using one of the techniques we discussed in class; alternatively, you may employ your won approach. No matter how you decide to replace the unknowns, explain fully. The final data should be presented as $(SCN, A_i, data)$ where SCN is the tuple key, A_i is the attribute, and data is the new data.

Answer: There are 16 missing values and all for 6^{th} attribute.Below is the output from my Java code.

```
Cleansing data .....
================================================================

1. For SCN 1057013 attribute 6 for record 24 cleaned with 3
```

```
 2. For SCN 1096800 attribute 6 for record 41 cleaned with 3
 3. For SCN 1183246 attribute 6 for record 140 cleaned with 3
 4. For SCN 1184840 attribute 6 for record 146 cleaned with 3
 5. For SCN 1193683 attribute 6 for record 159 cleaned with 3
 6. For SCN 1197510 attribute 6 for record 165 cleaned with 3
 7. For SCN 1241232 attribute 6 for record 236 cleaned with 3
 8. For SCN 169356 attribute 6 for record 250 cleaned with 3
 9. For SCN 432809 attribute 6 for record 276 cleaned with 3
10. For SCN 563649 attribute 6 for record 293 cleaned with 3
11. For SCN 606140 attribute 6 for record 295 cleaned with 3
12. For SCN 61634 attribute 6 for record 298 cleaned with 3
13. For SCN 704168 attribute 6 for record 316 cleaned with 3
14. For SCN 733639 attribute 6 for record 322 cleaned with 3
15. For SCN 1238464 attribute 6 for record 412 cleaned with 3
16. For SCN 1057067 attribute 6 for record 618 cleaned with 3

============================================================
```

Explanation over replacing data: I decided to use mean of the attributes for which we have missing values. And it is a lot better than getting rid of missing data. Particularly, with small data, getting rid of data has substantial effect.

R code for getting rid of data with missing values:

```
>myData <- read.table("WolBergsBreastCancerData.csv", sep=",",
+ header = FALSE, colClasses=c("character", "character",
+ "character","character","character","character", +"character",
+ "character", "character", "character", "character"),
+ na.strings=c("NA","?"))
>cleanData <- na.omit(myData)
```

(a) Is the amount of missing data significant?
Answer: The total amount of datapoints we have is 699. ANd among them 16 are missing. So it comes about 2.3%, and missing value within 1-5% is considered to be managebale. Although significant is a relative word. And its defination differ person to person. However, for me it is not that significant.

(b) Assess the significance of either keeping or removing the tuples with unknown data.
Answer: It is always good to keep the data by replacing with some other values rather than just getting rid of it. And it becomes more significant if the amount of missing data is huge. Then getting rid of them is not a prudent choice.Moreover, even though an attribute is missing in a datapoint but other attributes are still present.With insufficient data, removal might lead to biased results.
On the other hand, imputing the mean/median for missing values is generally common practice. Although, it is better than just throwing away the data. But it has its own problems. By replacing the missing values with a mean/median we distort the actual distribution.And this happen to work for me.

Below you can see how much adverse effect throwing the data out has on small dataset.

Table 1: PPV by Number of attribute with **Mean** Imputation

$C_{km=2}(\Delta^*)$	# of Iteration to Converge	PPV
$A_1, ..., A_9$	3	0.9570815450643777
$A_1, ..., A_7$	3	0.9570815450643777
$A_1, ..., A_5$	6	0.9356223175965666
$A_1, ..., A_3$	5	0.932761087267525
A_1, A_2	3	0.8869814020028612

*The missing value was replaced with 3

[h]

Table 2: PPV by Number of attribute with **Removal** of records with missing values

$C_{km=2}(\Delta^*)$	# of Iteration to Converge	PPV
$A_1, ..., A_9$	4	0.7130307467057101
$A_1, ..., A_7$	4	0.677891654465593
$A_1, ..., A_5$	6	0.7569546120058566
$A_1, ..., A_3$	4	0.7818448023426061
A_1, A_2	4	0.7569546120058566

*The missing value records were thrown away

[h]

4. Assume the attribute Clump Thickness is A_1, Uniformity of Cell Size is A_2 and so on. Attribute A_10 has only two domain values and is the classifier. For Δ^* and the attributes $A_i, 1 \leq i \leq 9$.

   (a) Which A_i has the greatest variance? You will write an R function that takes a list of numbers and returns the variance.
   Answer: I wrote a R function which takes variance or entropy as argument and returns the attribute.

   By removing the tuples with missing values:

```
> operFuncOnAttr("WolBergsBreastCancerData.csv", "var","y")
V7
 6
```

   By replacing attributes with missing values with Mean:

```
> operFuncOnAttr("WolBergsBreastCancerData.csv", "var","n")
V7
 6
```

   (b) Which A_i has the lowest entropy? You may use the R package entropy by Hausser and Strimmer.
   Answer: By removing the tuples with missing values:

```
> operFuncOnAttr("WolBergsBreastCancerData.csv", "entropy","n")
V9
 8
```

   By replacing attributes with missing values with Mean:

```
> operFuncOnAttr("WolBergsBreastCancerData.csv", "entropy","y")
V9
6
```

   R code for function operFuncOnAttr:

```
operFuncOnAttr <- function (fileName, funct, removeAttr) {

        #Reading the file and storing it in a Dataset
        myData <- read.table(fileName, sep=",", header = FALSE,
+ colClasses=c("character","character","character","character","character",
+ "character","character","character","character","character","character"),
+ na.strings=c("NA","?"))

        #Ignoring the SCN for the computation
        myData <- myData[, 2:(ncol(myData)-1)]

        # Converting string to numeric.
        cols <- c(1:(ncol(myData)))
        myData[,cols] = apply(myData[,cols], 2, function(x) as.numeric(x))

        #replacing the tuples with missing values with mean.
        if (removeAttr == "n"){
        myData[is.na(myData)] <- mean(myData[,6], na.rm=TRUE)
```

```
        }

        if (funct == "var") {
                ret <- which.max(sapply (myData, var,na.rm=TRUE))
        } else {
                # load entropy library
                if(is.element("entropy", installed.packages()[,1])) {
                        print("entropy package already installed")
                } else {
                        print("Installing Package entropy ....")
                        install.packages('entropy')
                }
                library("entropy")
                ret <- which.min(sapply(myData, entropy,na.rm=TRUE))
        }
        return (ret)
}
```

(c) Fill-in the table below with the KL distance for attribute pairs. For this we construct a mass function P_i over A_i by simple counting. For a cell whose row, column entries are A_i, A_j, find $d_{KL}(P_i||P_j)$. You may use an existing R function for this, but you need to provide sufficient package details for someone who would consider using that package.
Answer:

i. When throwing away the tuples the KL Distance Matrix looks like:

```
> computeKLDist("WolBergsBreastCancerData.csv","y")
[1] "seewave package already installed"
           [,1]       [,2]       [,3]      [,4]      [,5]      [,6]      [,7]
 [1,] 0.0000000 0.34942479 0.32526209 0.4602240 0.2598199 0.4825389 0.2953075
 [2,] 0.3303850 0.00000000 0.09345295 0.3017260 0.2293274 0.3406523 0.2406133
 [3,] 0.3131351 0.09296301 0.00000000 0.3226041 0.2367744 0.3262189 0.2470536
 [4,] 0.4488371 0.27668360 0.29780869 0.0000000 0.3332852 0.3821307 0.3154832
 [5,] 0.2791672 0.23689648 0.24479753 0.3526845 0.0000000 0.4329557 0.2250175
 [6,] 0.4430740 0.30147808 0.28623300 0.3850621 0.4006527 0.0000000 0.3421348
 [7,] 0.3017020 0.24714584 0.25332848 0.3138671 0.2118222 0.3761513 0.0000000
 [8,] 0.4493107 0.27416155 0.27801764 0.4282573 0.3354324 0.4702960 0.3271679
 [9,] 0.4898512 0.48282164 0.50148291 0.5492319 0.3456823 0.6821753 0.4828394
           [,8]      [,9]
 [1,] 0.4827846 0.4339470
 [2,] 0.3246805 0.5338144
 [3,] 0.3324193 0.5311455
 [4,] 0.4421757 0.5593252
 [5,] 0.3628460 0.3457609
 [6,] 0.5023946 0.6797457
 [7,] 0.3645522 0.4533546
 [8,] 0.0000000 0.5868865
 [9,] 0.5418213 0.0000000
```

ii. When repalcing the missing values with mean the KL Distance Matrix looks like:

```
> computeKLDist("WolBergsBreastCancerData.csv","n")
```

```
[1] "seewave package already installed"
           [,1]       [,2]       [,3]      [,4]      [,5]      [,6]      [,7]
 [1,] 0.0000000 0.34738153 0.32400762 0.4641627 0.2617773 0.4812723 0.2923631
 [2,] 0.3281002 0.00000000 0.09438032 0.3058258 0.2316805 0.3450263 0.2407317
 [3,] 0.3113454 0.09473766 0.00000000 0.3283380 0.2382599 0.3272561 0.2454988
 [4,] 0.4516002 0.27842612 0.30137452 0.0000000 0.3310688 0.3886677 0.3182790
 [5,] 0.2808857 0.23827180 0.24550698 0.3496033 0.0000000 0.4308768 0.2258272
 [6,] 0.4433625 0.30757052 0.28902232 0.3904466 0.3977981 0.0000000 0.3398343
 [7,] 0.2980436 0.24713276 0.25208530 0.3180932 0.2128848 0.3733125 0.0000000
 [8,] 0.4476825 0.26963252 0.27628375 0.4311402 0.3374148 0.4754893 0.3271676
 [9,] 0.4890809 0.48393551 0.50151244 0.5485056 0.3443245 0.6750230 0.4808986

           [,8]      [,9]
 [1,] 0.4815181 0.4329595
 [2,] 0.3202202 0.5343926
 [3,] 0.3305491 0.5294709
 [4,] 0.4421630 0.5608295
 [5,] 0.3633375 0.3445298
 [6,] 0.5062158 0.6672949
 [7,] 0.3639762 0.4505998
 [8,] 0.0000000 0.5919925
```

5. Implement k-means (see Algorithm1) so that you can cluster Δ^*. Call this program C_km. For this first iteration, you may assume that $1 \leq k \leq 10$ and that all domain values are numeric. Write $C_{km=i}$ for i blocks (or clusters). Assume there are k blocks $C_1, C_2, ..., C_k$. If an element of C_i is correctly clustered in C_i, then it is considered a True Positive (TP). If an element that correctly belongs to C_i is clustered in a different C_j , then the element is a False Positive (FP). The Positive Predictive Value (PPV) is
$PPV = \frac{TP}{TP+FP}$ In this problem we will investigate C_{kms} PPV varying the attributes used. Specifically, create a table that pairs $C_{km=2s}$ PPV using the attributes shown in the table below:
Answer: Since the first part of the questions says that the $1 \leq k \leq 10$. I ran k-means with 9 attributes for cluster 1 to n. And below are the accuracies. [h]

Table 3: K-means for clusters from 1 to 10

$C_{km=i}(\Delta^*)$	# of Iteration to Converge	PPV
$C_{km=1}$	1	0.6552217453505007
$C_{km=2}$	5	0.9570815450643777
$C_{km=3}$	10	0.9628040057224606
$C_{km=4}$	14	0.9570815450643777
$C_{km=5}$	9	0.9699570815450643
$C_{km=6}$	16	0.9599427753934192
$C_{km=7}$	11	0.9699570815450643
$C_{km=8}$	9	0.9713876967095851
$C_{km=9}$	14	0.9699570815450643
$C_{km=10}$	12	0.9656652360515021

*The missing value was replaced with 3

Analysis: I was amazed to see the k-means to cluster every single time.

Even though the PPV will not make sense here. But the fact that it converged every single time is amazing.

Table 4: PPV by Number of attribute with **Mean** Imputation

$C_{km=2}(\Delta^*)$	# of Iteration to Converge	PPV
$A_1, ..., A_9$	3	0.9570815450643777
$A_1, ..., A_7$	3	0.9570815450643777
$A_1, ..., A_5$	6	0.9356223175965666
$A_1, ..., A_3$	5	0.932761087267525
A_1, A_2	3	0.8869814020028612

*The missing value was replaced with 3

[h]

Table 5: PPV by Number of attribute with **Median** Imputation

$C_{km=2}(\Delta^*)$	# of Iteration to Converge	PPV
$A_1, ..., A_9$	6	0.9585121602288984
$A_1, ..., A_7$	4	0.9570815450643777
$A_1, ..., A_5$	5	0.9356223175965666
$A_1, ..., A_3$	3	0.932761087267525
A_1, A_2	7	0.8869814020028612

*The missing value was replaced with 1

[h]

Table 6: PPV by Number of attribute with **Mode** Imputation

$C_{km=2}(\Delta^*)$	# of Iteration to Converge	PPV
$A_1, ..., A_9$	6	0.9585121602288984
$A_1, ..., A_7$	3	0.9570815450643777
$A_1, ..., A_5$	5	0.9356223175965666
$A_1, ..., A_3$	3	0.932761087267525
A_1, A_2	4	0.8841201716738197

*The missing value was replaced with 1

[h]

[h]

Table 7: PPV by Number of attribute with **Removal** of records with missing values

$C_{km=2}(\Delta^*)$	# of Iteration to Converge	PPV
$A_1, ..., A_9$	4	0.7130307467057101
$A_1, ..., A_7$	4	0.677891654465593
$A_1, ..., A_5$	6	0.7569546120058566
$A_1, ..., A_3$	4	0.7818448023426061
A_1, A_2	4	0.7569546120058566

*The missing value records were thrown away

6. One of the most common techniques in assessing function is using V -fold cross validation. The idea is simple. Suppose Δ^* = N. Partition Δ^* into V = 10 sets $D^* = d_1^*, d_2^*, ..., d_10^*$ such that each $|d_i^*| = N/10$ tuples and all d_i, d_j are pairwise disjoint. The task is to use V 1 sets to train and the remaining d to test. Fill-in the PPV for each fold. Create a weighted PPV, $(1/10)\sum_{i=1}^{10} PPV_i$
Answer: [h]

Table 8: PPV over V-fold with **Mean** Imputation

Train	# of Iteration to Converge	Test	PPV Results
$C_{km=2}(D^* - d_1^*)$	5	$C_{km=2}(d_1^*)$	0.8142857142857143
$C_{km=2}(D^* - d_2^*)$	5	$C_{km=2}(d_2^*)$	0.9571428571428572
$C_{km=2}(D^* - d_3^*)$	6	$C_{km=2}(d_3^*)$	0.9857142857142858
$C_{km=2}(D^* - d_4^*)$	7	$C_{km=2}(d_4^*)$	0.9428571428571428
$C_{km=2}(D^* - d_5^*)$	4	$C_{km=2}(d_5^*)$	0.9428571428571428
$C_{km=2}(D^* - d_6^*)$	5	$C_{km=2}(d_6^*)$	0.9714285714285714
$C_{km=2}(D^* - d_7^*)$	5	$C_{km=2}(d_7^*)$	0.9571428571428572
$C_{km=2}(D^* - d_8^*)$	5	$C_{km=2}(d_8^*)$	1.0
$C_{km=2}(D^* - d_9^*)$	6	$C_{km=2}(d_9^*)$	1.0
$C_{km=2}(D^* - d_1^*0)$	4	$C_{km=2}(d_1^*0)$	1.0

*Weighted PPV = 0.9571428571428573

[h]

[h]

[h]

Table 9: PPV over V-fold with **Median** Imputation

Train	# of Iteration to Converge	Test	PPV Results
$C_{km=2}(D^* - d_1^*)$	6	$C_{km=2}(d_1^*)$	0.8142857142857143
$C_{km=2}(D^* - d_2^*)$	4	$C_{km=2}(d_2^*)$	0.9571428571428572
$C_{km=2}(D^* - d_3^*)$	4	$C_{km=2}(d_3^*)$	0.9857142857142858
$C_{km=2}(D^* - d_4^*)$	8	$C_{km=2}(d_4^*)$	0.9428571428571428
$C_{km=2}(D^* - d_5^*)$	5	$C_{km=2}(d_5^*)$	0.9428571428571428
$C_{km=2}(D^* - d_6^*)$	5	$C_{km=2}(d_6^*)$	0.9714285714285714
$C_{km=2}(D^* - d_7^*)$	4	$C_{km=2}(d_7^*)$	0.9571428571428572
$C_{km=2}(D^* - d_8^*)$	4	$C_{km=2}(d_8^*)$	1.0
$C_{km=2}(D^* - d_9^*)$	3	$C_{km=2}(d_9^*)$	1.0
$C_{km=2}(D^* - d_1^*0)$	5	$C_{km=2}(d_1^*0)$	1.0

*Weighted PPV = 0.9571428571428573

Table 10: PPV over V-fold with **Mode** Imputation

Train	# of Iteration to Converge	Test	PPV Results
$C_{km=2}(D^* - d_1^*)$	5	$C_{km=2}(d_1^*)$	0.8142857142857143
$C_{km=2}(D^* - d_2^*)$	3	$C_{km=2}(d_2^*)$	0.9571428571428572
$C_{km=2}(D^* - d_3^*)$	4	$C_{km=2}(d_3^*)$	0.9857142857142858
$C_{km=2}(D^* - d_4^*)$	6	$C_{km=2}(d_4^*)$	0.9428571428571428
$C_{km=2}(D^* - d_5^*)$	5	$C_{km=2}(d_5^*)$	0.9428571428571428
$C_{km=2}(D^* - d_6^*)$	5	$C_{km=2}(d_6^*)$	0.9714285714285714
$C_{km=2}(D^* - d_7^*)$	5	$C_{km=2}(d_7^*)$	0.9571428571428572
$C_{km=2}(D^* - d_8^*)$	5	$C_{km=2}(d_8^*)$	1.0
$C_{km=2}(D^* - d_9^*)$	5	$C_{km=2}(d_9^*)$	1.0
$C_{km=2}(D^* - d_1^*0)$	5	$C_{km=2}(d_1^*0)$	1.0

*Weighted PPV = 0.9571428571428573

Problem Three: Astronomy

1. Prove or disprove that R is a metric.
   Answer:

2. Lets see whether the below distance is a metric or not
   $R(g, g^{'}) = \left((\frac{z_g - z_{g'}}{0.2596})^2 + (\frac{r_g - r_{g'}}{8.6})^2 + (\frac{(u_g - r_g) - (u_{g'} - r_{g'})}{11.84})^2 \right)^{1/2}$
   In order to prove it is metric, we need to make sure that the above 4 properties are met. Let us see them one by one,

   (i) $d(x, y) \geq 0$ for all $x, y \in X$:
   This is true since we are taking aquare of elements and adding them later on. Hence, it can never be less than 0.

   (ii) $d(x, y) = 0$ if and only if $x = y$:
   Clearly since we are doing subtraction operation here. For it to be 0, $x = y$.

   (iii) $d(x, y) = d(y, x)$ for all $x, y \in X$:
   Again taking square help us out here. Since, we are taking square, it really doesn't matter we take perform $x - y$ or $y - x$.

Table 11: PPV over V-fold with **Removal** of missing data tuple

Train	# of Iteration to Converge	Test	PPV Results
$C_{km=2}(D^* - d_1^*)$	3	$C_{km=2}(d_1^*)$	0.5294117647058824
$C_{km=2}(D^* - d_2^*)$	4	$C_{km=2}(d_2^*)$	0.5882352941176471
$C_{km=2}(D^* - d_3^*)$	5	$C_{km=2}(d_3^*)$	0.6911764705882353
$C_{km=2}(D^* - d_4^*)$	6	$C_{km=2}(d_4^*)$	0.5294117647058824
$C_{km=2}(D^* - d_5^*)$	5	$C_{km=2}(d_5^*)$	0.5
$C_{km=2}(D^* - d_6^*)$	5	$C_{km=2}(d_6^*)$	0.8529411764705882
$C_{km=2}(D^* - d_7^*)$	6	$C_{km=2}(d_7^*)$	0.8088235294117647
$C_{km=2}(D^* - d_8^*)$	6	$C_{km=2}(d_8^*)$	0.8823529411764706
$C_{km=2}(D^* - d_9^*)$	6	$C_{km=2}(d_9^*)$	0.7941176470588235
$C_{km=2}(D^* - d_1^*0)$	5	$C_{km=2}(d_1^*0)$	0.9705882352941176

*Weighted PPV = 0.7147058823529412

(iv) $d(x, y) \geq d(x, z) + d(z, y)$ for all $x, y, z \in X$:
Lets observe our distance function more closely. Firstly, lets get rid of the constants.Since constants are not really going to be a problem to prove this. Then we have the following equation.

$$R(g, g^{'}) = \Big((z_g - z_{g'})^2 + (r_g - r_{g'})^2 + ((u_g - r_g) - (u_{g'} - r_{g'}))^2\Big)^{1/2}$$

Now. Lets call $(u_{g'} - r_{g'}) = a_{g'}$, $(u_g - r_g) = a_g$; then we have :

$$R(g, g^{'}) = \Big((z_g - z_{g'})^2 + (r_g - r_{g'})^2 + (a_g - a_{g'})^2\Big)^{1/2}$$

Now what is this, this is nothing but eculidean distance in 3D space. And we know eculedian is a metric, so it should satisfy the transitivity rule.

Hence, proved that $R(g, g^{'})$ is a metric.

3. Run the control algorithm and compare the output to control 2.
Answer: I ran the control algorithm and compared it with mine. Below is the output of the same from my Java Code:

```
============================================================
Computing control Algortihm......
Number of datapoints present in both    :0
Number of datapoints present only in our computed control       :7246
Number of datapoints present only in our given control  :7246
============================================================
```

4. Run k means on candidate using main as the centroids. Because the galaxy type must be identical, you should first partition candidate into three blocks that represents galaxy types 0,1,2 and similarly for main. For each block in candidate, use the corresponding block in the partitioned main as the centroids.
Answer: Please see the .java file attached for this. Below is the few analysis i did before coding for the data.Although the plots did not came out to be really clear. Howver, it shows one important thing that we do have few outliers. Which stand apart in the 3d plot.
NOTE: I ahave not attached the candidate dataset plots. As they are upto 34MB in size and making latex to stop working. However, I plan to attach them along with other documents.

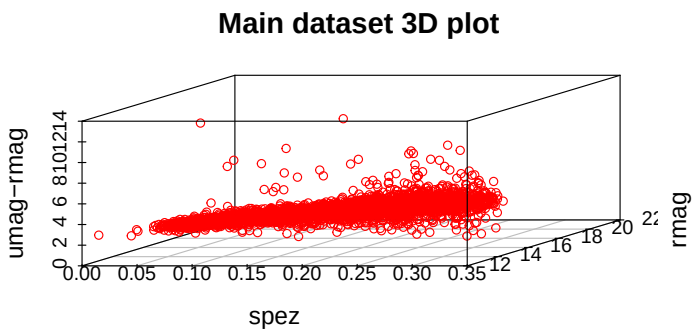

Figure 1: 3D scatter plot for data in *main* dataset

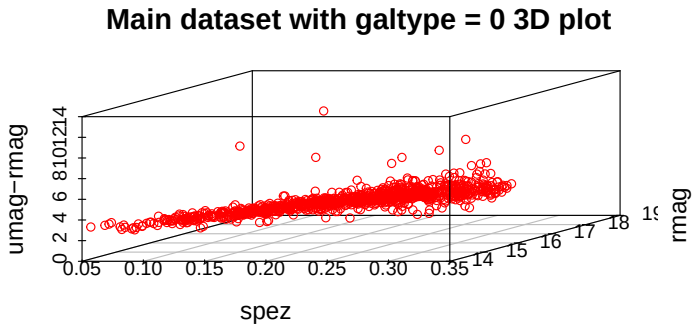

Figure 2: 3D scatter plot for data in *main (galtype= 0)* dataset

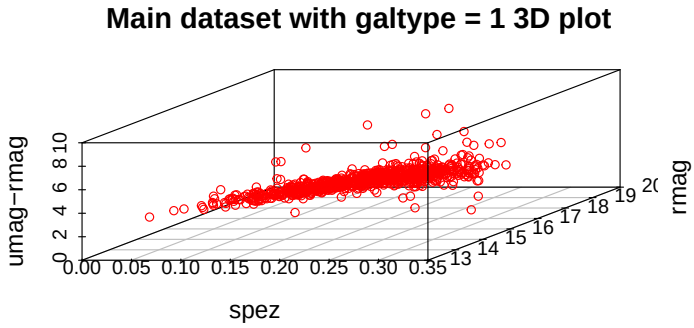

Figure 3: 3D scatter plot for data in *main (galtype= 1)* dataset

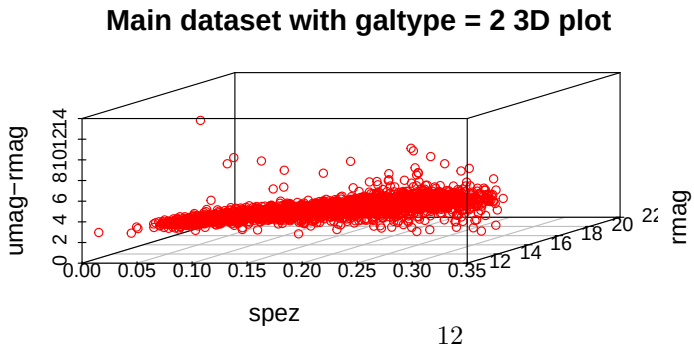

Figure 4: 3D scatter plot for data in *main (galtype= 2)* dataset

5. Perform data analysis on the features of main and candidate.
   Answer: Below is the correlation between all features in main and in candidate:

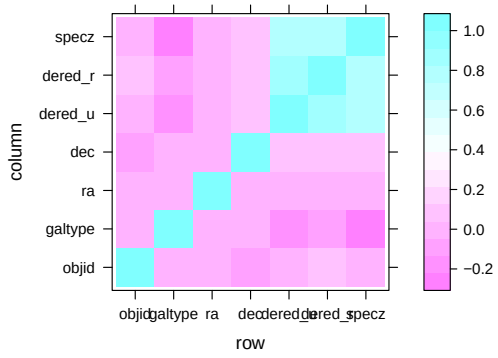

Figure 5: Feature correlation of all Main attributes.

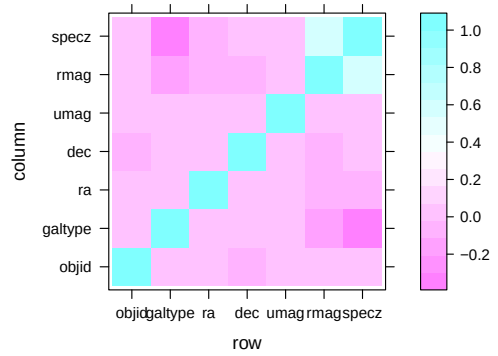

Figure 6: Feature correlation of all Candidate attributes.

Here cyan refer to positive correlation and pink refers to negative correlation. Moreover, denser the color means higher the correlation. That's why you will se that the diagonal elements are dark cyan. Because they are compared against themselves only.

My Clustering algorithm bash Shell script

```
now=$( date +%d%m%Y-%H%M%S )
outputFile="ClustAlgo_$now.out"
errFile="ClustAlgo_$now.err"

echo "Compiling Code...."
javac -d ./bin ./src/ClusteringAlgo.java

echo "Starting the program in background....."
echo "Execute ps -f the see the status of your code...."
java -cp ./bin ClusteringAlgo > $outputFile 2> $errFile &
```