COMP 3340 Assignment 1 Part 2)

Question 1:

Question 1A)

There is not strong correlation between variables within the students academic performance dataset, however the correlation between raisedHands and VisITedResources is the strongest with a 0.692 pearsons coefficient, which is almost defined as a strong correlation.

Table:

	raisedhands	VisITedResources	AnnouncementsView	Discussion
raisedhands	1.0000000	0.6915717	0.6439178	0.3393860
VisITedResources	0.6915717	1.0000000	0.5945000	0.2432918
AnnouncementsView	0.6439178	0.5945000	1.0000000	0.4172900
Discussion	0.3393860	0.2432918	0.4172900	1.0000000

Code:

```
question_1a <- function(datafile){
  #function takes a dataframe as input and uses the cor method which calculates the pearsons correlation for all the variable pairs.
  numeric_columns <- datafile[ names(datafile) %in% c("raisedhands","VisITedResources","AnnouncementsView","Discussion")]
  pearsons <- cor(numeric_columns, method = "pearson")
}
```

Question 1B)

(F) Students "raised hands" are more actively involved in study-related works

```
a <- cor(datafile$raisedhand, datafile$VisITedResources, method = "pearson")
b <- cor(datafile$raisedhand, datafile$AnnouncementsView, method = "pearson")
c <- cor(datafile$raisedhand, datafile$Discussion, method = "pearson")
print(a)
print(b)
print(c)
```

```
[1] 0.6915717
[1] 0.6439178
[1] 0.339386
```

(F) No apparent gender bias when it comes to subject/topic choices

```
        Pearson's Chi-squared test

data:  datafile$gender and datafile$Topic
X-squared = 23.04, df = 11, p-value = 0.01745
```

	Arabic	Biology	Chemistry	English	French	Geology	History	IT	Math	Quran	Science	Spanish
F	16	10	12	19	30	10	8	32	5	9	23	1
M	43	20	12	26	35	14	11	63	16	13	28	24

```
gender_topic <- chisq.test(datafile$gender, datafile$Topic)
print(gender_topic)
```

(T) Girls seem to have better overall performance than boys

```
        Pearson's Chi-squared test

data:  datafile$gender and datafile$Class
X-squared = 33.326, df = 2, p-value = 5.798e-08
```

(F) Boys are generally a bit more open to discussions, visiting resources, and raising hands

Means in Contribution by Gender.

```
gender_contribution <- split(X <- datafile[ names(datafile) %in% c("raisedhands","Discussion","VisITedResources","gender")], X$gender)
i = 1
while(i < 3){
  X <- colMeans(gender_contribution[[i]][sapply(gender_contribution[[i]], is.numeric)], na.rm=TRUE)
  print(gender_contribution[[i]]$gender[1])
  print(X)
  i = i +1
}
```

```
[1] "F"
     raisedhands VisITedResources       Discussion
        52.86286         64.00000         47.82857
[1] "M"
     raisedhands VisITedResources       Discussion
        43.28197         49.51803         40.67541
```

(T) Those who participated more (higher counts in Discussion, Announcement Views,

Raised Hands), usually perform better.

Means in contribution values by Results.

```
[1] "H"                 "70.2887323943662" "53.3802816901408" "53.6619718309859"
[1] "L"                 "16.8897637795276" "15.5748031496063" "30.8346456692913"
[1] "M"                 "48.9383886255924" "40.9620853080569" "43.7914691943128"
```

```
# Determines mean values for class contribution by results
numeric_columns <- datafile[ names(datafile) %in% c("raisedhands","AnnouncementsView","Discussion","Class")]
X <- split(numeric_columns, numeric_columns$Class)
i =1;
while(i < 4)
{
  out <- c(X[[i]]$Class[1],mean(X[[i]]$raisedhands),mean(X[[i]]$AnnouncementsView),mean(X[[i]]$Discussion))
  print(out)
  i = i + 1
}
```

Question 2:

Explanations:

To generate the MSTs and RNGs the euclidean distance measure was used to generate a distance matrix, this was achieved by looping through the matrixes comparing columns or rows and determining their distances from each other by that measure. The following code determines the euclidean distance of two objects.

```
euclidean_dist <- function(x, y) {
  sqrt(sum((x - y)^2))
}
```

Code for MSTs:

```
generate_distances <- function(datafile, filename1,filename2){
  row_matrix <- get_distance_rows(datafile)
  cols_matrix <- get_distance_cols(datafile)
  name1 = paste("Results/",filename1,"distancematrix.csv", sep = "")
  name2 = paste("Results/",filename2,"distancematrix.csv", sep = "")
  write.csv(row_matrix,file = name1)
  write.csv(cols_matrix,file = name2)
  relative_neighbourhoods(row_matrix,filename1)
  relative_neighbourhoods(cols_matrix,filename2)

  rows_graph <- mst(graph_from_adjacency_matrix(row_matrix, mode = "undirected",weighted = TRUE))
  cols_graph <- mst(graph_from_adjacency_matrix(cols_matrix, mode = "undirected",weighted = TRUE))

  V(rows_graph)$label <- colnames(row_matrix)
  E(rows_graph)$label <- round(E(rows_graph)$weight, 3)
  V(cols_graph)$label <- colnames(cols_matrix)
  E(cols_graph)$label <- round(E(cols_graph)$weight, 3)
  print(sum(E(rows_graph)$weight))
  print(sum(E(cols_graph)$weight))
  name1 = paste("Results/",filename1,"MST.gml", sep = "")
  name2 = paste("Results/",filename2,"MST.gml", sep = "")

  write_graph(rows_graph, name1,format = 'gml')
  write_graph(cols_graph,name2,format = 'gml')
}
```

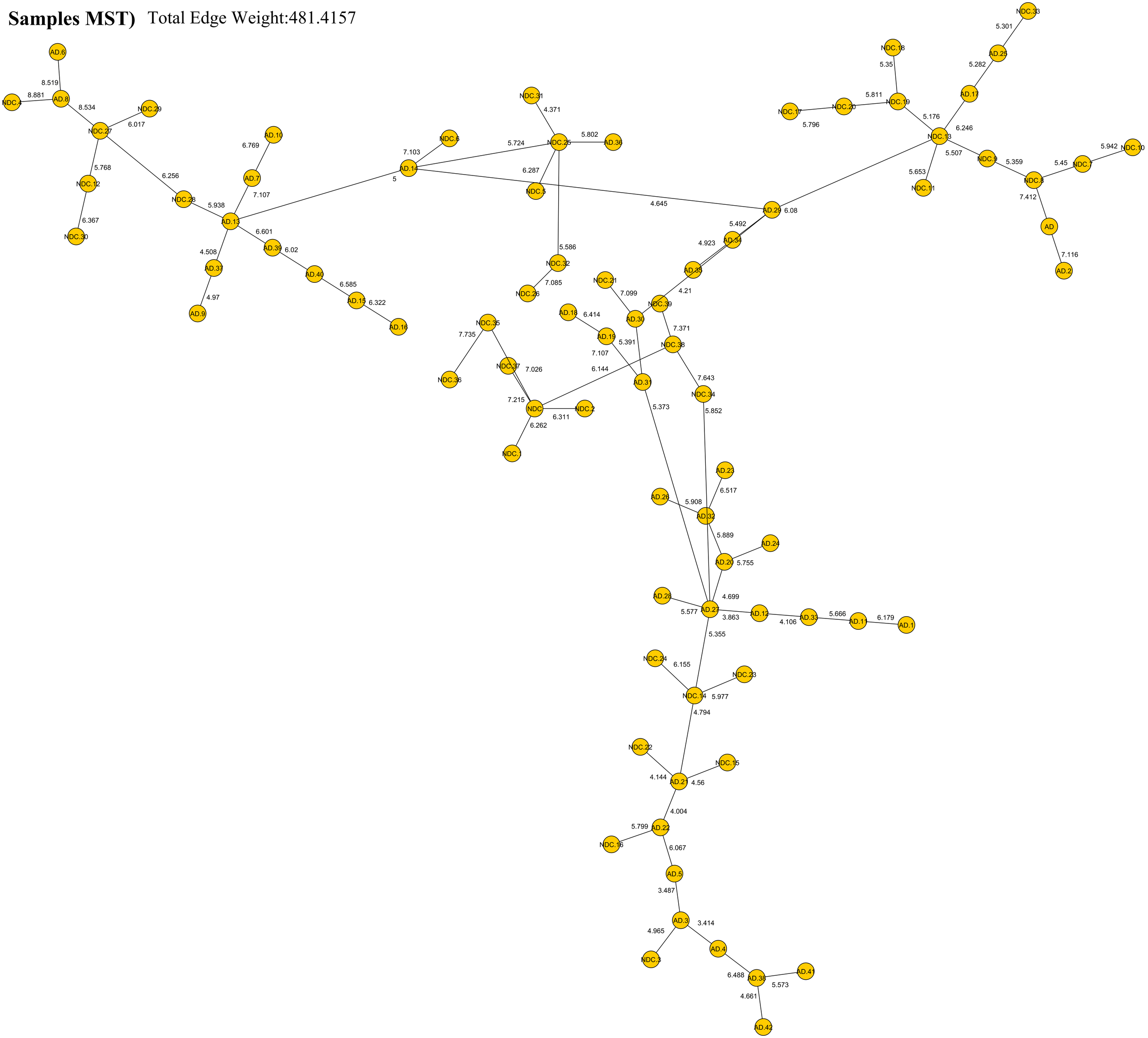
The mst() function in R uses prims algorithm to find the path between nodes with the lowest total edge weight by chossing a random vertex, adding the lowest weighted edge to the minimum spanning tree and repeating the process, however if the lowest weighted edge forms a loop it is ignored.

Code for RNGs:

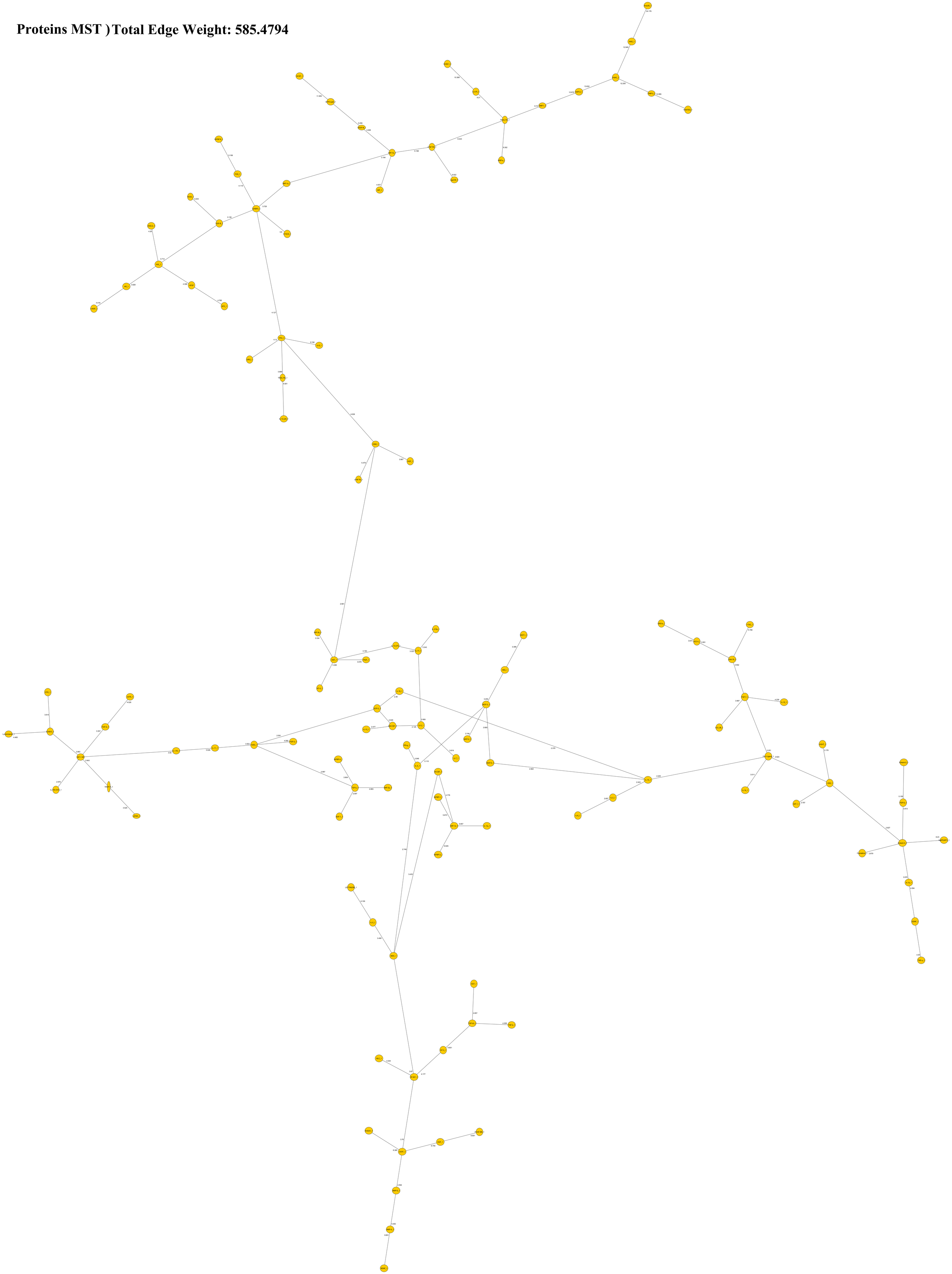
```
relative_neighbourhoods <- function(matrix,title)
{
  matrix1 <- as.matrix(matrix)
  rng_graph <- rng(dx = matrix1,r = 1, algorithm = 'cover_tree')
  print(rng_graph)
  rng_graph <- as.undirected(rng_graph)
  #adds labels
  V(rng_graph)$label <- colnames(matrix1)
  E(rng_graph)$weight <- apply(get.edges(rng_graph, 1:gsize(rng_graph)),1,function(x)matrix1[x[1],x[2]])
  E(rng_graph)$label <- round(E(rng_graph)$weight, 3)
  name <- paste("Results/",title,"RNG.gml", sep ="")
  plot(rng_graph)
  write_graph(rng_graph, name, format = 'gml')
}
```

The rng() function takes a distance matrix as input and uses the cover tree algorithm to generate a relative neighbourhood graph. The apply function is used to apply edge weights to the new graph by looping through the original matrix and finding the cell that had the edge as the current edge within the graph and applying the weight of that edge as a label within the graph.

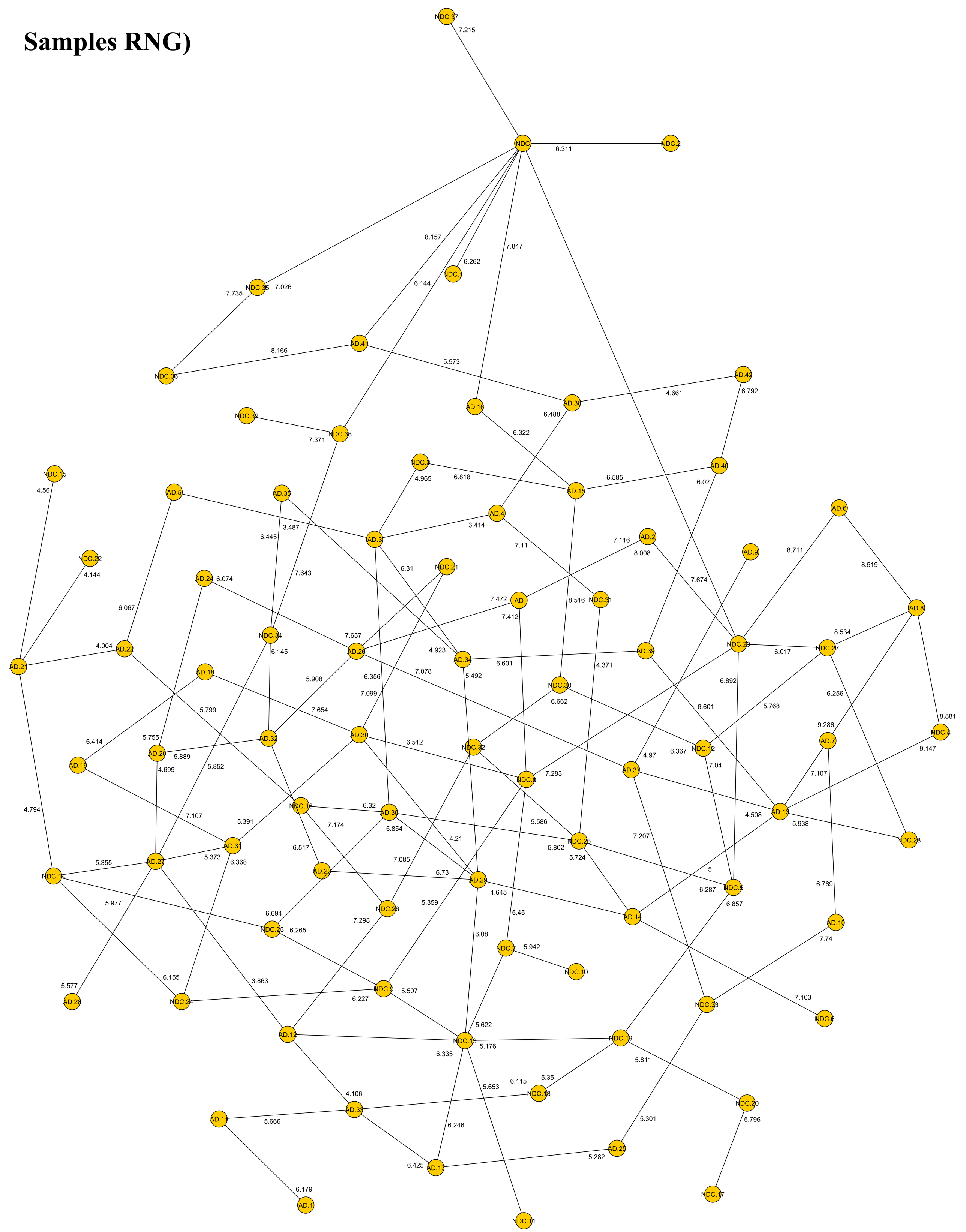
Samples MST) Total Edge Weight:481.4157



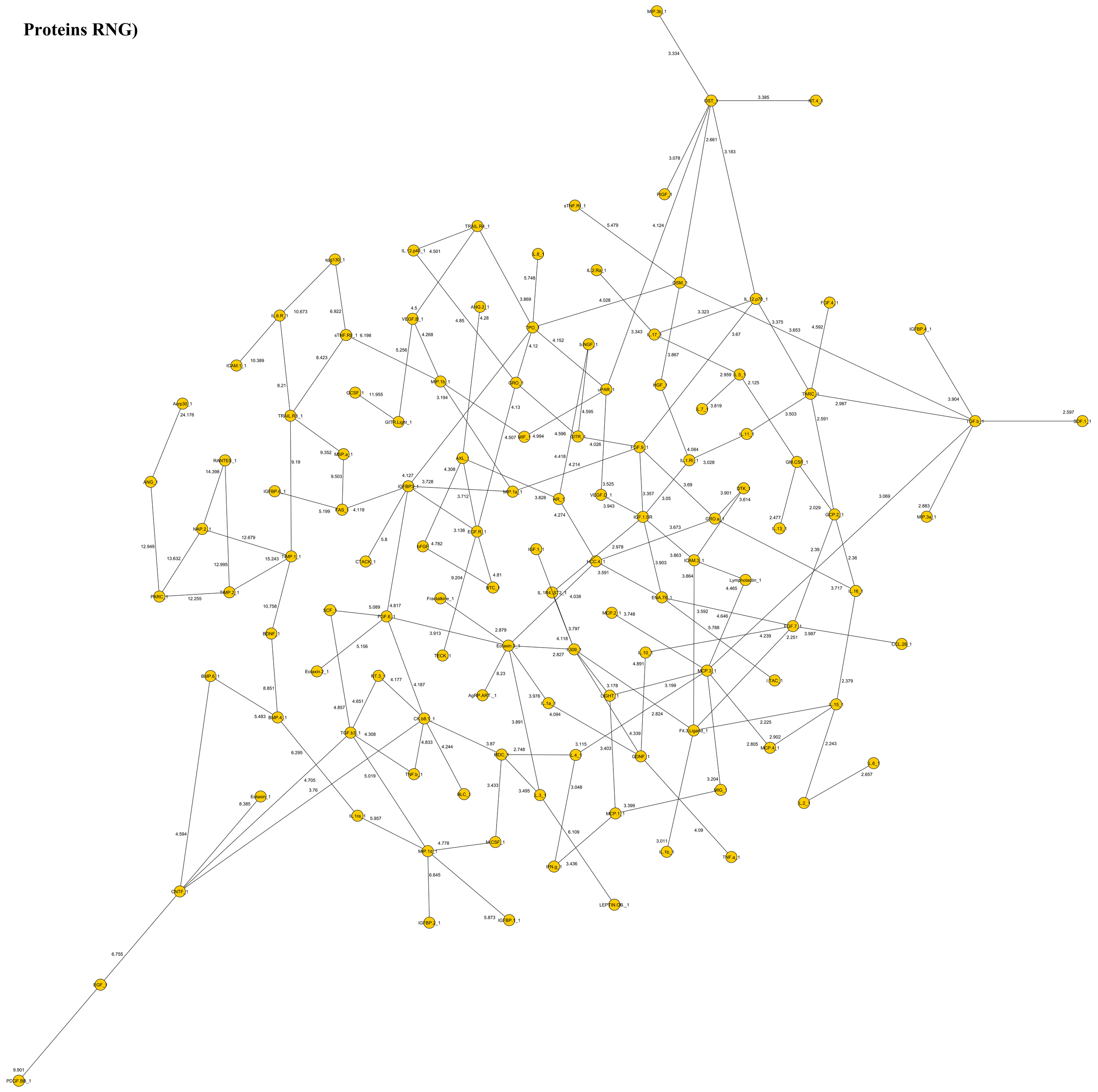
Proteins MST)Total Edge Weight: 585.4794



Samples RNG)


NDC.37
7.215
NDC
6.311
NDC.2
8.157
7.847
6.262
NDC.1
6.144
7.735
NDC.35
7.026
AD.41
8.166
NDC.36
5.573
AD.42
4.661
6.792
AD.38
AD.16
6.488
NDC.39
6.322
NDC.38
7.371
NDC.3
4.965
6.818
AD.40
6.585
6.02
AD.15
NDC.15
4.56
AD.5
AD.35
3.487
6.445
AD.4
3.414
AD.3
7.11
AD.6
8.711
8.519
AD.2
7.116
8.008
AD.9
NDC.22
4.144
6.31
NDC.21
7.643
AD.24
6.074
7.674
7.472
AD
8.516
NDC.31
7.412
AD.8
6.067
NDC.34
7.657
8.534
4.004
AD.22
6.145
AD.26
4.923
AD.34
6.601
AD.39
NDC.29
6.017
NDC.27
AD.21
4.371
AD.18
5.908
6.356
7.078
5.492
NDC.30
6.662
6.892
7.099
6.256
5.799
7.654
6.601
5.768
8.881
9.286
6.414
5.755
AD.32
AD.30
6.512
NDC.4
AD.7
AD.19
AD.20
5.889
NDC.32
6.367
NDC.12
9.147
4.97
7.04
4.699
5.852
7.283
AD.37
NDC.8
7.107
4.794
NDC.16
6.32
AD.36
4.508
AD.13
5.938
7.107
5.391
7.174
5.854
5.586
NDC.25
7.207
NDC.28
4.21
6.517
AD.31
5.802
5.724
5.355
AD.27
5.373
6.368
7.085
AD.23
6.73
AD.29
5
6.287
NDC.5
6.769
NDC.14
4.645
6.857
5.977
6.694
NDC.26
5.359
5.45
AD.14
AD.10
7.298
NDC.23
6.265
6.08
7.74
NDC.7
5.942
NDC.10
5.577
6.155
3.863
NDC.9
5.507
7.103
AD.28
NDC.24
6.227
NDC.33
NDC.6
5.622
AD.12
NDC.13
NDC.19
5.176
6.335
5.811
6.115
5.35
5.653
NDC.18
4.106
NDC.20
AD.33
AD.11
5.666
5.301
5.796
6.246
AD.25
5.282
6.425
AD.17
6.179
AD.1
NDC.17
NDC.11

Proteins RNG)



Question 3)

Explanations:

The Boruta package was used to select the features that have the largest impact on the determining the class result. Boruta is a wrapper function that applies the random forest algorithm to the dataset its given, requiring specification of the evalutated attributes and the attribute that is determined by the others, and returns the important attributes in an boruta object. The tentative rough fix function is another method of the Boruta package that tries to identify if features on the edge of important or un important should be kept or not. The getSelectedAttributes function is used to get the names of the features selected in a vector, which is then used to loop through the dataset and remove all other proteins. The CLASS column was looped through replacing any values that were not “AD” with “NON_AD”, the matrix was then written to a file for Weka to use.

For Classfication the Weka software was used to generate a J48 pruned tree using the features selected. The collapseTree setting within the classifier in Weka was set to true as this ensured that unneccesary checks were not being formed. Other values were mainly set to default as they didn’t have a significant impact on the classifiers perforamance.

a)

The following proteins were chosen as the most important for determining the outcome of the CLASS value.

```
[1] "CLASS"    "EGF_1"    "IL.1a_1"  "IL.3_1"   "MIG_1"    "RANTES_1" "TNF.a_1"  "GCSF_1"
```

Code**:**

```
feature_selection <- function(data){
  set.seed(111)
  bor <- Boruta(as.factor(data[,1]), x = data)
  bor <- TentativeRoughFix(bor, averageOver = Inf)
  x <- getSelectedAttributes(bor, withTentative = FALSE)
  print(x)
  x <- as.data.frame(x)
  a = 1
  while(a <= nrow(x))
  {
    x[a,1] <- gsub(".","-",x[a,1], fixed = TRUE)
    a = a + 1
  }
  matrix1 <- matrix(data = data[,colnames(data) %in% x[,1]], nrow = nrow(data), ncol = nrow(x))
  colnames(matrix1) <- x[,1]
  rownames(matrix1) <- rownames(data)
  a = 1
  while(a <= nrow(matrix1))
  {
    if(matrix1[a,  nrow(x)  'AD'){
      matrix1[a,1]  NON_AD"
    }
    a = a + 1
  }
  write.csv(matrix1, file = "Results/features.csv")
  x
}
```

b)

Weka outputs the classification system below, with the pruned tree being the actual system that will be used to test on the different datasets.

```
=== Run information ===

Scheme:       weka.classifiers.trees.J48 -B -C 0.25 -M 2
Relation:     features
Instances:    83
Attributes:   9

              CLASS
              EGF_1
              IL-1a_1
              IL-3_1
              MIG_1
              RANTES_1
              TNF-a_1
              GCSF_1
Test mode:    evaluate on training data

=== Classifier model (full training set) ===

J48 pruned tree
------------------

IL-1a_1 <= -0.145044532
|   IL-3_1 <= 0.113499427
|   |   GCSF_1 <= -0.415675342
|   |   |   EGF_1 <= 0.568856941: AD (3.0)
|   |   |   EGF_1 > 0.568856941: NON_AD (2.0)
|   |   GCSF_1 > -0.415675342: AD (31.0)
|   IL-3_1 > 0.113499427: NON_AD (4.0)
IL-1a_1 > -0.145044532: NON_AD (43.0/9.0)

Number of Leaves  :     5

Size of the tree :      9


Time taken to build model: 0 seconds
```

c)

Weka outputs with the classification system.
Training Set Results:

```
Time taken to test model on training data: 0 seconds

=== Summary ===

Correctly Classified Instances          74               89.1566 %
Incorrectly Classified Instances         9               10.8434 %
Kappa statistic                          0.7845
Mean absolute error                      0.1715
Root mean squared error                  0.2928
Relative absolute error                 34.3391 %
Root relative squared error             58.6005 %
Total Number of Instances               83

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure  MCC      ROC Area  PRC Area  Class
               0.791     0.000     1.000       0.791   0.883      0.803    0.911     0.908     AD
               1.000     0.209     0.816       1.000   0.899      0.803    0.911     0.844     NON_AD
Weighted Avg.  0.892     0.101     0.911       0.892   0.891      0.803    0.911     0.877

=== Confusion Matrix ===

  a  b   <-- classified as
 34  9 |  a = AD
  0 40 |  b = NON_AD
```

Test Set AD Results:

```
Time taken to test model on supplied test set: 0.15 seconds

=== Summary ===

Correctly Classified Instances          82               89.1304 %
Incorrectly Classified Instances        10               10.8696 %
Kappa statistic                          0.7784
Mean absolute error                      0.202
Root mean squared error                  0.3265
Relative absolute error                 40.2707 %
Root relative squared error             65.0694 %
Total Number of Instances               92

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.810    0.040    0.944      0.810    0.872      0.785    0.864     0.850     AD
                 0.960    0.190    0.857      0.960    0.906      0.785    0.864     0.834     NON_AD
Weighted Avg.    0.891    0.122    0.897      0.891    0.890      0.785    0.864     0.842

=== Confusion Matrix ===

  a  b   <-- classified as
 34  8 |  a = AD
  2 48 |  b = NON_AD
```

Test Set MCI Results:

```
Time taken to test model on supplied test set: 0.07 seconds

=== Summary ===

Correctly Classified Instances          27               57.4468 %
Incorrectly Classified Instances        20               42.5532 %
Kappa statistic                          0.136
Mean absolute error                      0.4478
Root mean squared error                  0.5998
Relative absolute error                 89.3583 %
Root relative squared error            119.618  %
Total Number of Instances               47

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.455    0.320    0.556      0.455    0.500      0.138    0.558     0.506     AD
                 0.680    0.545    0.586      0.680    0.630      0.138    0.558     0.562     NON_AD
Weighted Avg.    0.574    0.440    0.572      0.574    0.569      0.138    0.558     0.536

=== Confusion Matrix ===

  a  b   <-- classified as
 10 12 |  a = AD
  8 17 |  b = NON_AD
```

	Training	Test AD	Test MCI
Sensitivity	0.791	0.81	0.455
Specificity	1.0	0.96	0.68
Accuracy	89.16%	89.13%	57.45%
F1-Score	0.891	0.89	0.569
MCC	0.803	0.785	0.138
Youden’s J	0.791	0.77	0.135

The classifier algorithm showed promise based on its performance with the first Test set with strong scores all around, specifically in its specificity, however the second test set performed very poorly using the same classifier system, with its accuracy only being slightly better then a coin flip. These results could indicate a flaw in the classifier system however the MCI test set also has the lowest sample size and could have been a statistical outlier.

Question 4:

Explanation:

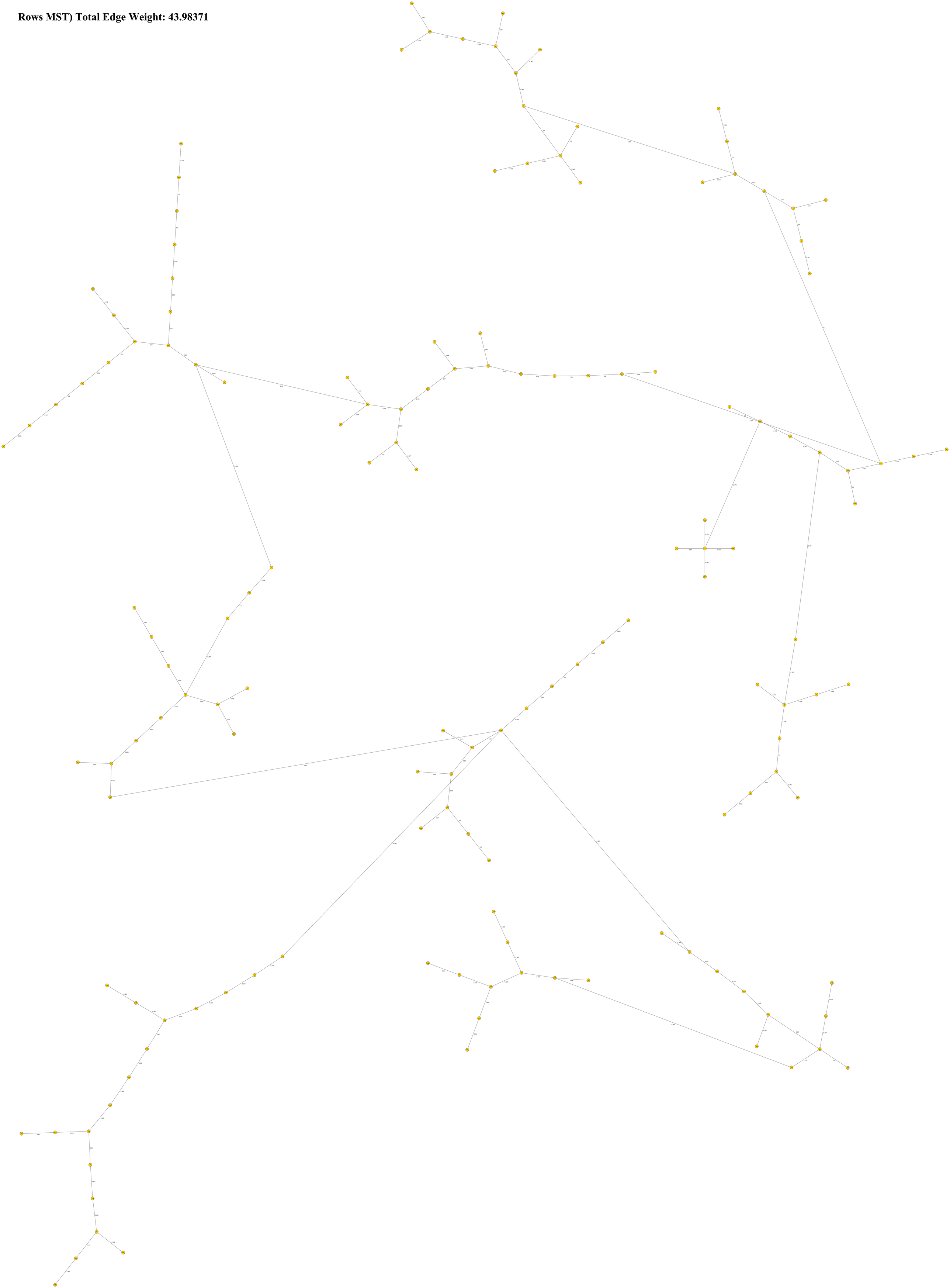
The Iris dataset was chosen as it has the most similar data types to the Alzheimers dataset being composed of numeric data with a nominal class variable. It details the measurements of the Sepals and Petals of samples of 3 different species of Iris flowers.

Code for Graphs:

```
question_4 <- function(){
  datafile <- read.csv("../Datasets/Iris.csv", header = TRUE)
  rownames(datafile) <- datafile[,1]
  datafile[,1] <- NULL
  checking <- datafile[1]
  class <-  datafile[5]
  datafile[,5] <- NULL
  generate_distances(datafile,"IrisRowsMatrix","IrisAttributesMatrix")
```

This function formats the data into a useable format to use the same code used for question2. It generates the MSTs and RNGs for the Iris Dataset. For details on the specific code refer to question 2.

Rows MST) Total Edge Weight: 43.98371



Attributes MST)

Total Edge Weight:

80.5007

SepalLengthCm

28.954

PetalLengthCm

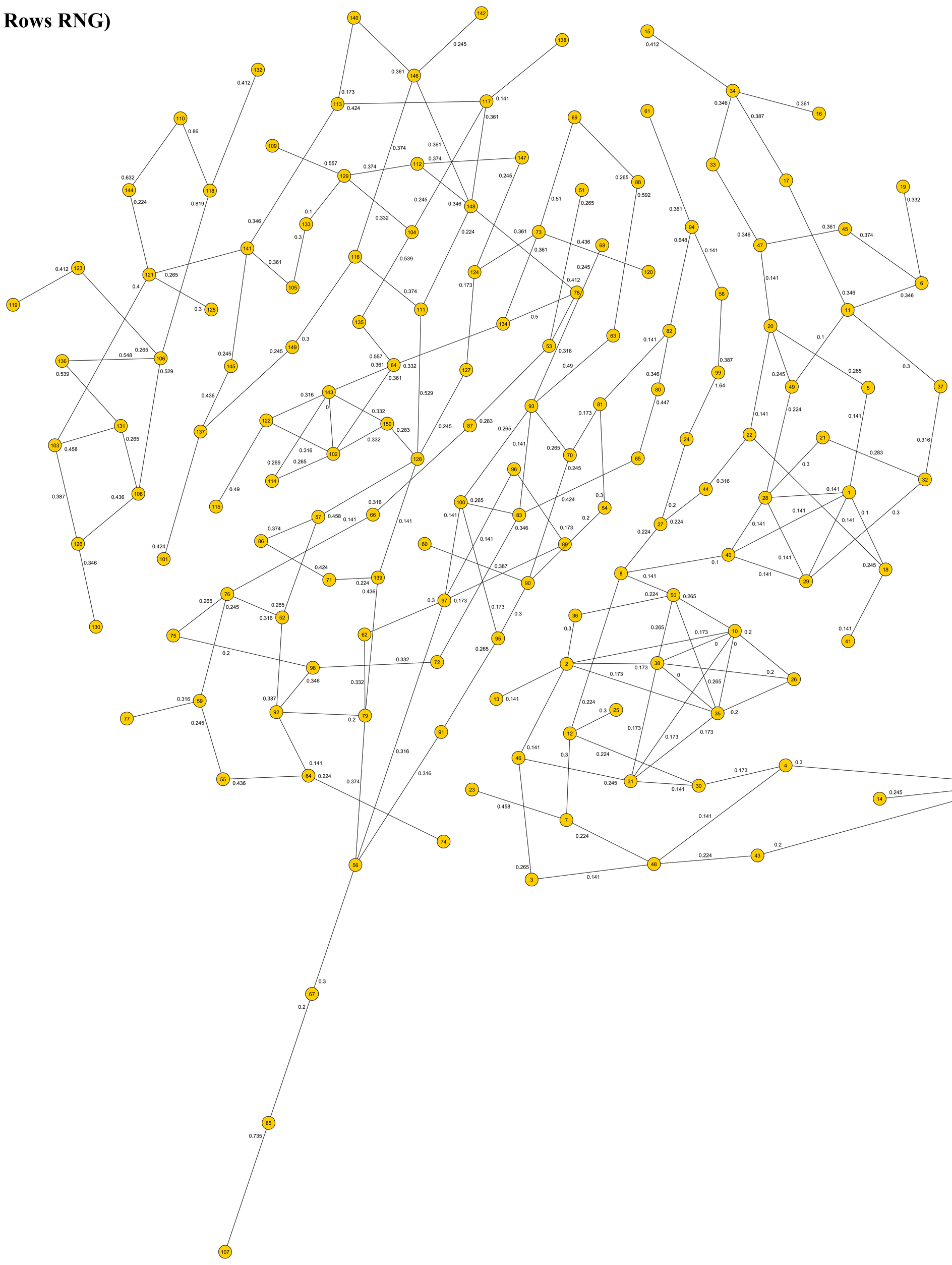
25.734

SepalWidthCm

25.813

PetalWidthCm

Rows RNG)


140
142
0.245
138
0.361
146
132
0.412
0.173
113
0.424
117
0.141
15
0.412
34
0.346
0.361
16
110
0.86
0.361
61
69
109
0.557
0.374
0.361
112
0.374
147
0.374
129
0.245
33
17
0.632
144
0.224
118
0.819
0.265
88
0.592
51
0.265
19
0.332
0.1
133
0.3
0.346
148
0.245
0.51
0.361
94
0.361
45
0.374
0.346
47
0.332
104
0.224
0.361
73
0.436
68
0.361
0.648
0.141
6
0.346
0.412
123
0.539
116
0.141
0.346
11
0.412
121
0.265
0.4
105
0.374
124
0.173
120
0.245
78
58
119
0.3
125
111
0.5
135
134
20
0.1
82
63
0.141
53
0.316
0.3
0.265
106
0.548
0.245
149
0.3
0.245
145
0.557
0.361
84
0.332
0.49
0.387
99
0.346
80
0.447
1.64
136
0.539
0.529
0.436
127
0.316
143
0.361
0.529
93
0.173
81
0.245
49
0.224
5
37
0.141
0.141
0.316
0
0.332
122
150
0.283
87
0.283
0.265
0.265
24
22
21
0.283
131
0.265
103
0.458
0.332
102
0.316
0.265
0.265
114
0.141
70
0.265
65
0.245
0.3
32
0.316
44
0.141
1
0.387
0.436
108
0.49
115
128
96
0.424
0.2
28
0.141
0.141
0.1
0.3
100
0.265
0.141
83
0.346
0.173
54
0.2
27
0.224
0.224
57
0.458
0.141
66
0.374
86
0.424
60
0.387
89
40
0.1
0.141
0.141
0.141
0.245
18
126
0.346
130
101
71
139
0.224
0.436
90
8
0.141
29
76
0.265
0.245
0.265
52
0.316
97
0.3
0.173
0.173
0.3
50
0.224
0.265
36
0.3
10
0.2
0.265
0.173
0
0
95
0.265
75
0.2
62
72
0.332
98
0.346
0.332
2
0.173
0.173
38
0
0.2
26
0.265
41
0.141
13
0.141
0.224
0.3
25
35
0.2
59
0.316
0.387
92
79
0.2
12
0.173
0.173
0.173
77
0.245
91
0.141
46
0.3
0.224
0.141
64
0.224
0.374
0.316
0.316
55
0.436
4
0.3
42
0.624
9
0.141
39
0.245
14
23
0.458
7
0.224
0.141
74
0.2
43
0.224
48
0.265
3
0.141
56
0.3
67
0.2
85
0.735
107

Attributes

RNG)

SepalLengthCm

28.954

PetalLengthCm

25.734

SepalWidthCm

25.813

PetalWidthCm

Classification and Testing)

Explanations:

The feature selection and classification system used was the same as question 3 however there was not multiple sheets the data was randomised by row, and split into 3 sections, training, Test1, Test2 it was then saved to individual csv files to save for Weka use. The classification model was creating using the training set and then tested on all 3. The details of the code and functions used are in question 3. The following code shuffles the rows within the Iris dataset.

```
split_data <- function(datafile)
{
  set.seed(11)
  datafile <- datafile[sample(1:nrow(datafile)),]
  datafile <- datafile[sample(1:nrow(datafile)),]
  write.csv(datafile[1:50,],"Results/TrainingSet.csv")
  write.csv(datafile[51:100,],"Results/TestSet1")
  write.csv(datafile[101:150,],"Results/TestSet2.csv")

}
```

Features Selected:

```
Boruta performed 9 iterations in 0.1596181 secs.
 5 attributes confirmed important: CLASS, PetalLengthCm, PetalWidthCm, SepalLengthCm, SepalWidthCm;
 No attributes deemed unimportant.
             x
1        CLASS
2  SepalWidthCm
3 PetalLengthCm
4  PetalWidthCm
5 SepalLengthCm
```

Boruta determined that all the features within the dataset are important for determining the CLASS a flower belongs to, so the original dataset will be used for classification.
However Wekas classification makes use of only PetalWidth, and Pedal Length, so the features selected can be narrowed down to PetalWidth and Pedal Length.

Classification:

Weka outputs the classification system below, with the pruned tree being the actual system that will be used to test on the different datasets.

```
J48 pruned tree
------------------

PetalWidthCm <= 0.4: Iris-setosa (15.0)
PetalWidthCm > 0.4
|   PetalLengthCm <= 4.7: Iris-versicolor (15.0)
|   PetalLengthCm > 4.7: Iris-virginica (20.0/3.0)

Number of Leaves  :     3

Size of the tree :      5


Time taken to build model: 0 seconds
```

Testing:
Training Set:

```
=== Evaluation on training set ===

Time taken to test model on training data: 0 seconds

=== Summary ===

Correctly Classified Instances          47               94      %
Incorrectly Classified Instances         3                6      %
Kappa statistic                          0.9099
Mean absolute error                      0.068
Root mean squared error                  0.1844
Relative absolute error                 15.3405 %
Root relative squared error             39.1699 %
Total Number of Instances               50

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.833    0.000    1.000      0.833    0.909      0.873    0.956     0.919     Iris-versicolor
                 1.000    0.091    0.850      1.000    0.919      0.879    0.955     0.850     Iris-virginica
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     Iris-setosa
Weighted Avg.    0.940    0.031    0.949      0.940    0.940      0.913    0.969     0.920

=== Confusion Matrix ===

  a  b  c   <-- classified as
 15  3  0 |  a = Iris-versicolor
  0 17  0 |  b = Iris-virginica
  0  0 15 |  c = Iris-setosa
```

Test Set 1:

```
=== Evaluation on test set ===

Time taken to test model on supplied test set: 0 seconds

=== Summary ===

Correctly Classified Instances          47               94      %
Incorrectly Classified Instances         3                6      %
Kappa statistic                          0.9092
Mean absolute error                      0.072
Root mean squared error                  0.2035
Relative absolute error                 16.1239 %
Root relative squared error             42.9074 %
Total Number of Instances               50

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.923    0.054    0.857      0.923    0.889      0.849    0.953     0.822     Iris-versicolor
                 0.944    0.031    0.944      0.944    0.944      0.913    0.957     0.912     Iris-virginica
                 0.947    0.000    1.000      0.947    0.973      0.958    0.974     0.967     Iris-setosa
Weighted Avg.    0.940    0.025    0.943      0.940    0.941      0.914    0.962     0.910

=== Confusion Matrix ===

  a  b  c   <-- classified as
 12  1  0 |  a = Iris-versicolor
  1 17  0 |  b = Iris-virginica
  1  0 18 |  c = Iris-setosa
```

Test Set 2:

```
=== Evaluation on test set ===

Time taken to test model on supplied test set: 0 seconds

=== Summary ===

Correctly Classified Instances          47               94      %
Incorrectly Classified Instances         3                6      %
Kappa statistic                          0.9098
Mean absolute error                      0.066
Root mean squared error                  0.1926
Relative absolute error                 14.8809 %
Root relative squared error             40.8934 %
Total Number of Instances               50

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.895    0.032    0.944      0.895    0.919      0.872    0.957     0.902     Iris-versicolor
                 1.000    0.057    0.882      1.000    0.938      0.912    0.971     0.882     Iris-virginica
                 0.938    0.000    1.000      0.938    0.968      0.954    0.969     0.958     Iris-setosa
Weighted Avg.    0.940    0.029    0.944      0.940    0.940      0.910    0.965     0.914

=== Confusion Matrix ===

  a  b  c   <-- classified as
 17  2  0 |  a = Iris-versicolor
  0 15  0 |  b = Iris-virginica
  1  0 15 |  c = Iris-setosa
```

Results:

	Training	Test AD	Test MCI
Sensitivity	0.94	0.94	0.94
Specificity	0.97	0.98	0.97
Accuracy	94%	94%	94%
F1-Score	0.94	0.94	0.94
MCC	0.913	0.914	0.910
Youden’s J	0.91	0.92	0.91

Evaluation:

The classification system created by the training set was proven to be highly effective when tested on the test sets.