Université d'Ottawa
Faculté de génie

École de science informatique
et de génie électrique

uOttawa
L'Université canadienne
Canada's university

University of Ottawa
Faculty of Engineering

School of Electrical Engineering
and Computer Science

Fundamental of Data Science

Assignment THREE

Reyad Melies

Part A: Clustering

1) K-Means Clustering

Using only the Sex and Age fields

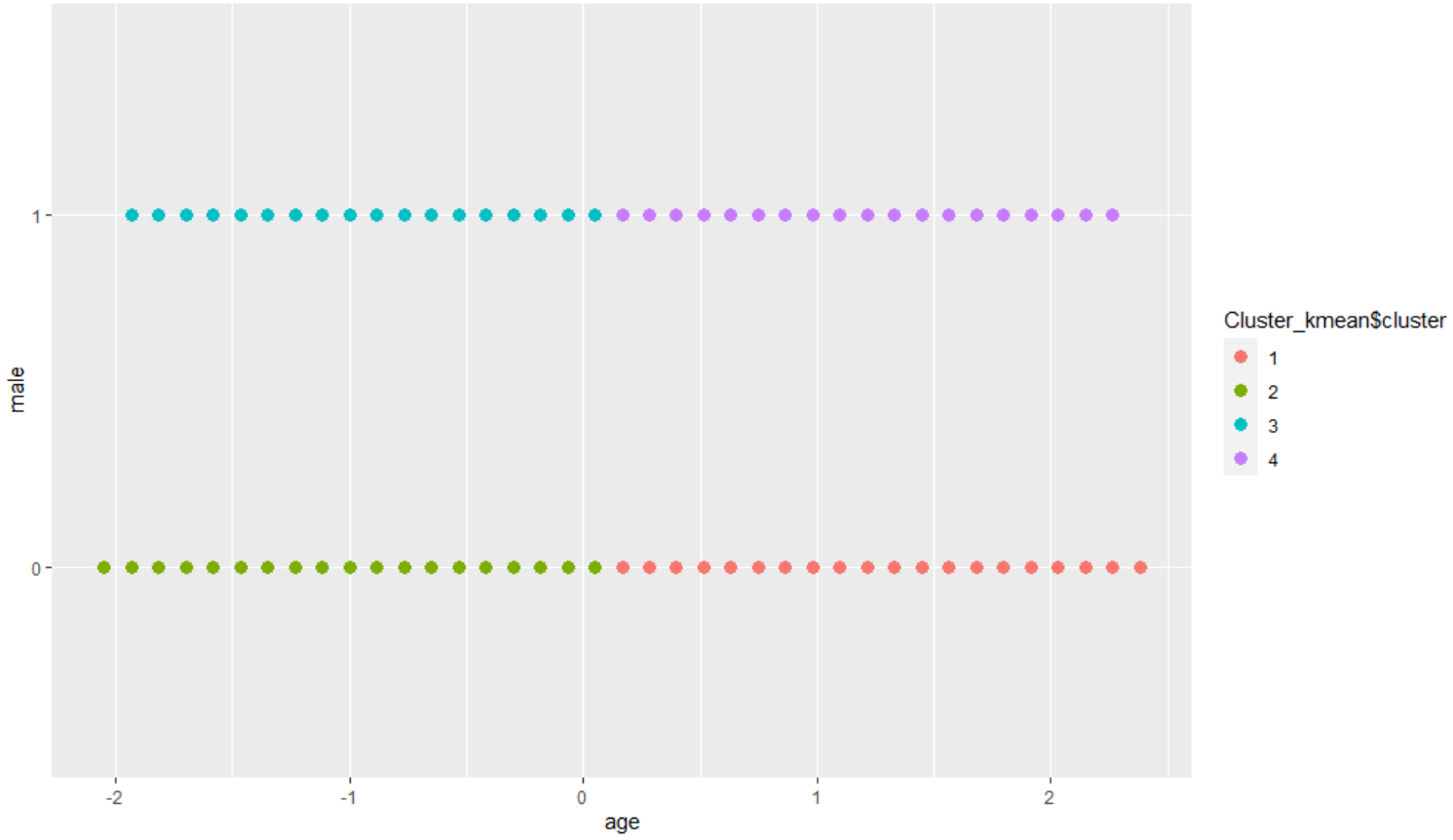
```
dataset = dataset[,1:2]
```

ensure you standardize Age.

```
#########data Standatization
dataset$age <- scale(dataset$age)
#########################################################
```

a) Perform k-means clustering on the selected attributes, specifying k = 4 clusters and plot.

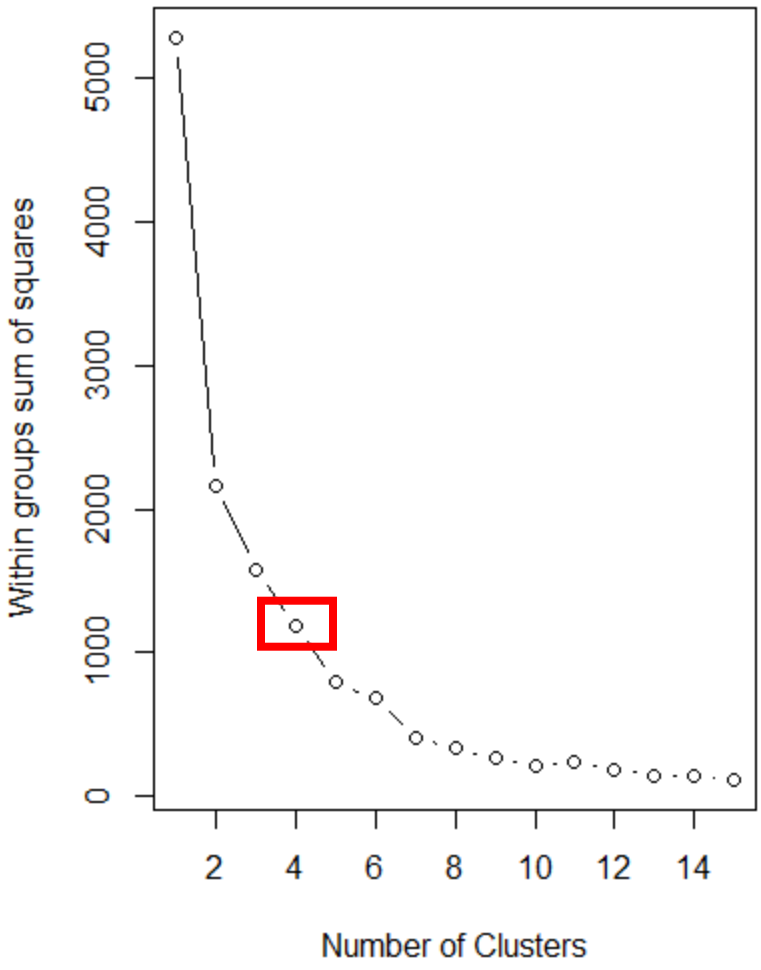
```
set.seed(917)
#Run k-means cluster of the dataset
Cluster_kmean <- kmeans(dataset, 4, nstart = 20)
#Tabulate the cross distribution
table(Cluster_kmean$cluster,dataset$male)
#Plot
Cluster_kmean$cluster <- factor(Cluster_kmean$cluster)
ggplot(dataset, aes(age,male, color = Cluster_kmean$cluster), ) +  geom_point(size=3)
```



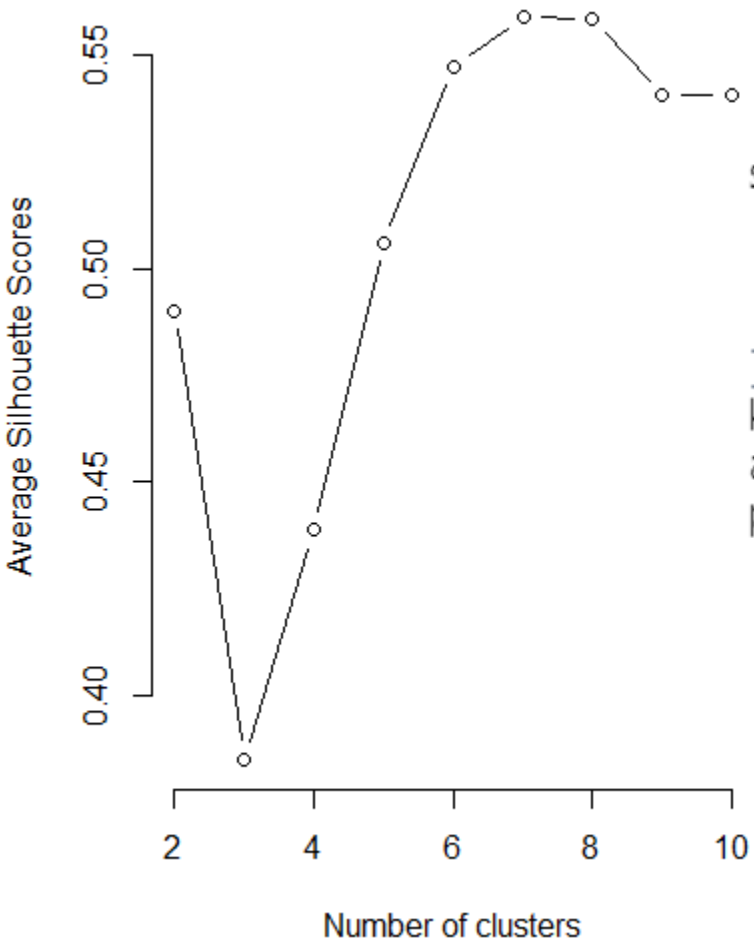
b) Apply the elbow method to determine the best k and plot

```
wss <- (nrow(dataset))*sum(apply(dataset,2,var))
for (i in 2:15) {
  wss[i] <- sum(kmeans(dataset,centers=i)$withinss)
}
plot(1:15, wss, type="b", xlab="Number of Clusters",ylab="Within groups sum of squares")
```

Best K is equal to 4 same as the figure above



c) Evaluate the quality of the clusters using the Silhouette Coefficient method.

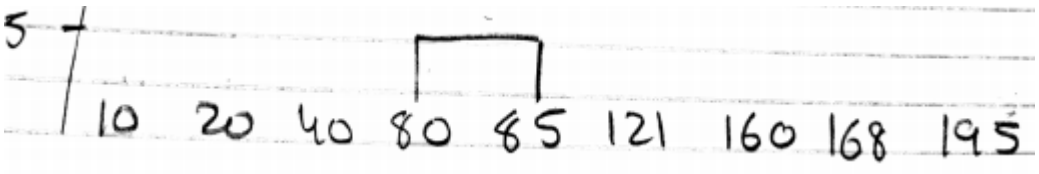



```
silhouette_score <- function(k){
  km <- kmeans(dataset, centers = k, nstart=25)
  ss <- silhouette(km$cluster, dist(dataset))
  mean(ss[, 3])
}
k <- 2:10
avg_sil <- sapply(k, silhouette_score)
plot(k, type='b', avg_sil, xlab='Number of clusters',
     ylab='Average Silhouette Scores', frame=FALSE)
```

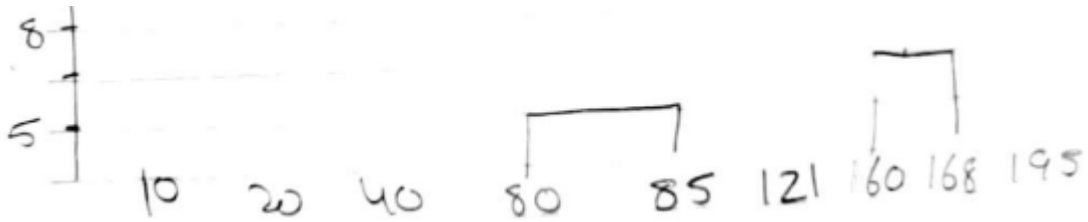
For 4 clusters silhouette score:
0.4387456

hierarchical agglomerative clustering with single linkage:

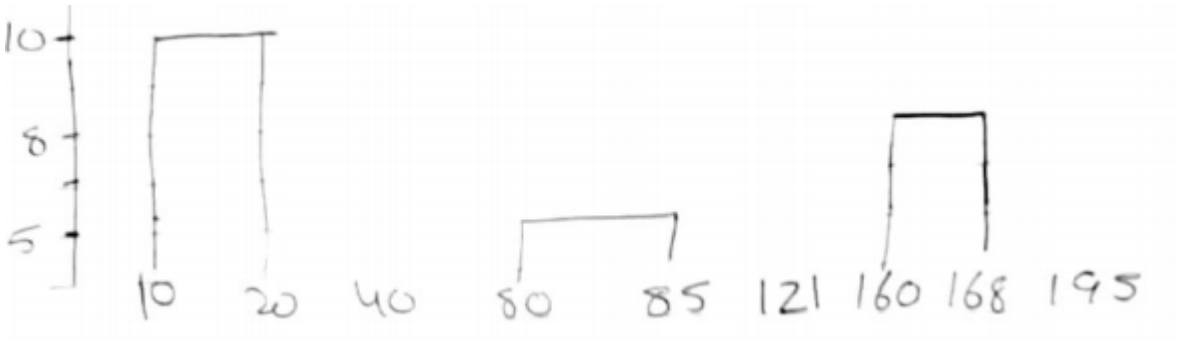
	10	20	40	80	85	121	160	168	195
10	0								
20	10	0							
40	30	20	0						
80	70	60	40	0					
85	75	65	45	5	0				
121	111	101	81	41	36	0			
160	150	140	120	80	75	39	0		
168	158	148	128	88	83	47	8	0	
195	185	175	155	115	110	74	35	27	0



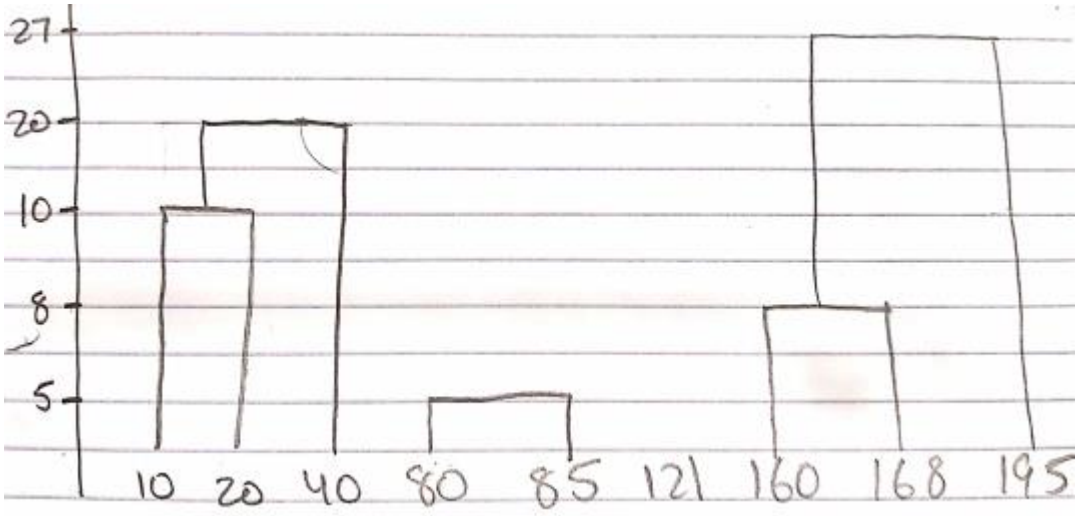
	10	20	40	80-85	121	160	168	195
10	0							
20	10	0						
40	30	20	0					
80-85	70	60	40	0				
121	111	101	81	36	0			
160	150	140	120	75	39	0		
168	158	148	128	83	47	8	0	
195	185	175	155	110	74	35	27	0



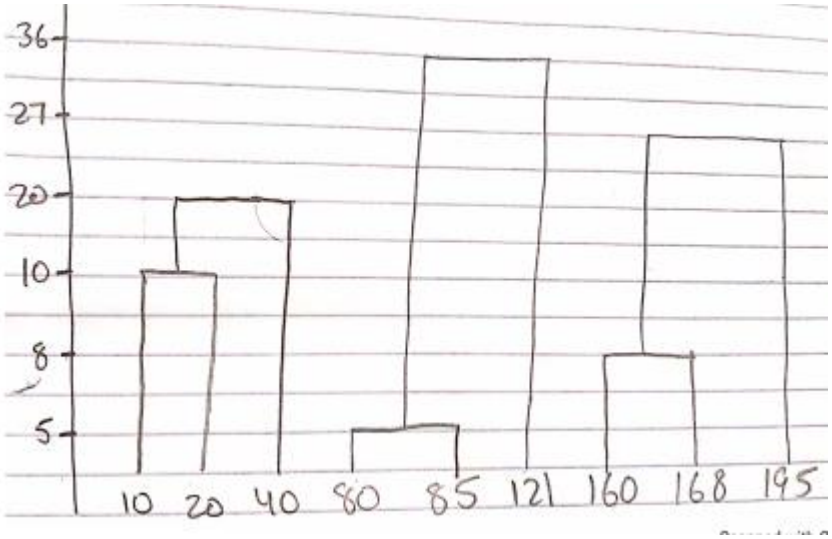
	10	20	40	80-85	121	160-168	195
10	0						
20	10	0					
40	30	20	0				
80-85	70	60	40	0			
121	111	101	81	36	0		
168-160	158	148	128	83	47	8	0
195	185	175	155	110	74	27	0



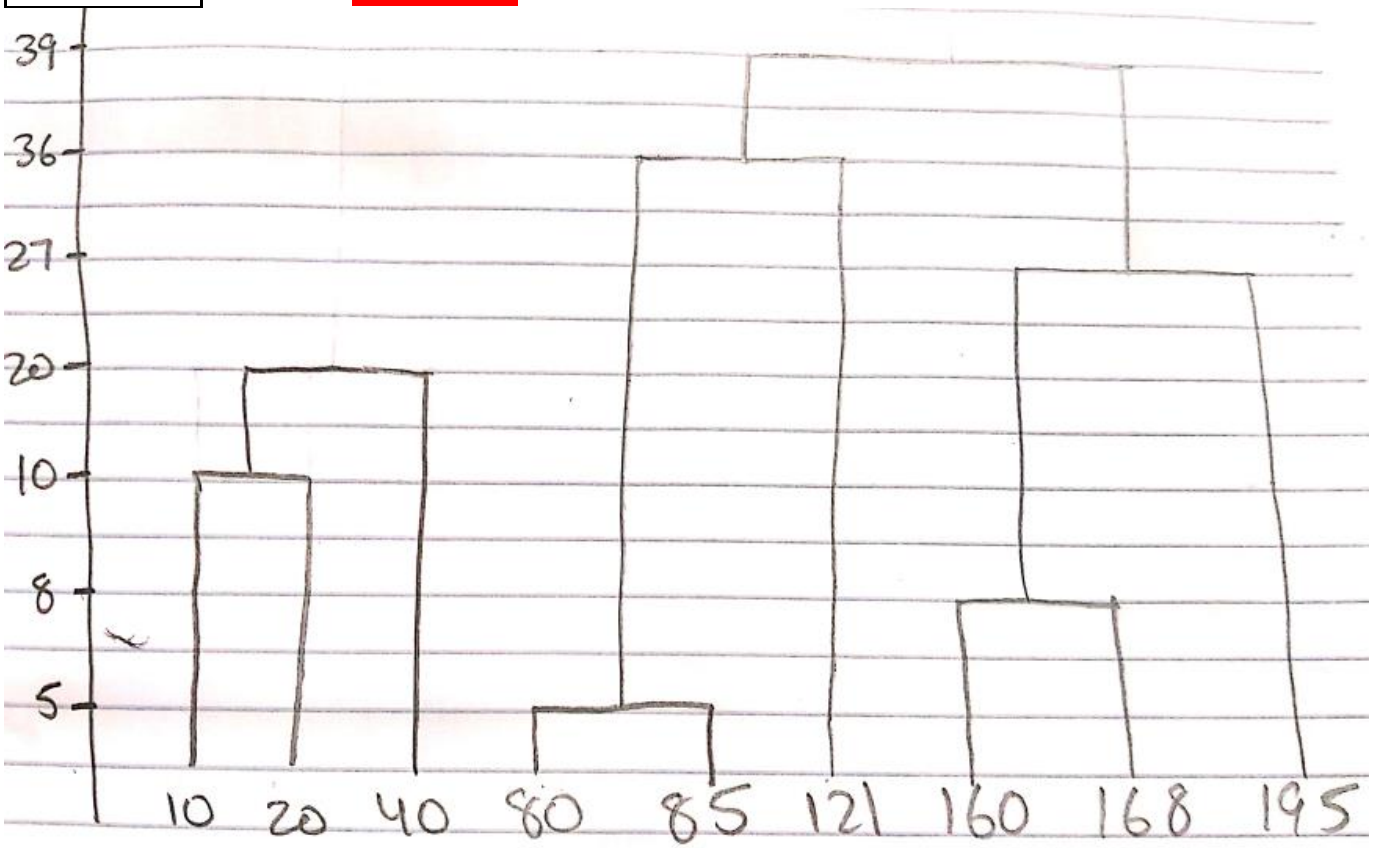
	10-20-40	80-85	121	160-168	195
10-20-40	0				
80-85	40	0			
121	81	36	0		
160-168	120	75	39	0	
195	155	110	74	27	0



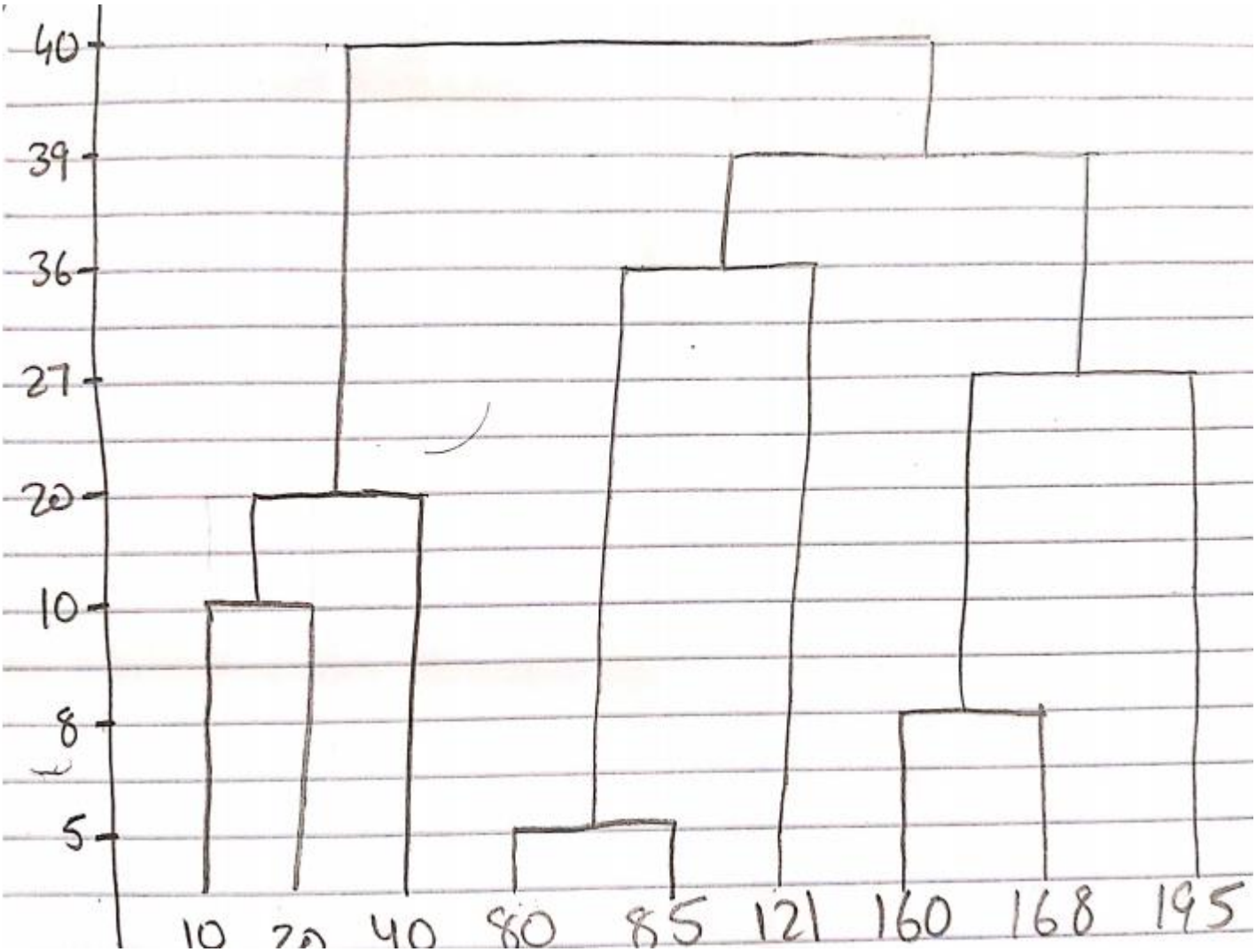
	10-20-40	80-85	121	160-168-195
10-20-40	0			
80-85	40	0		
121	81	36	0	
160-168-195	120	75	39	0

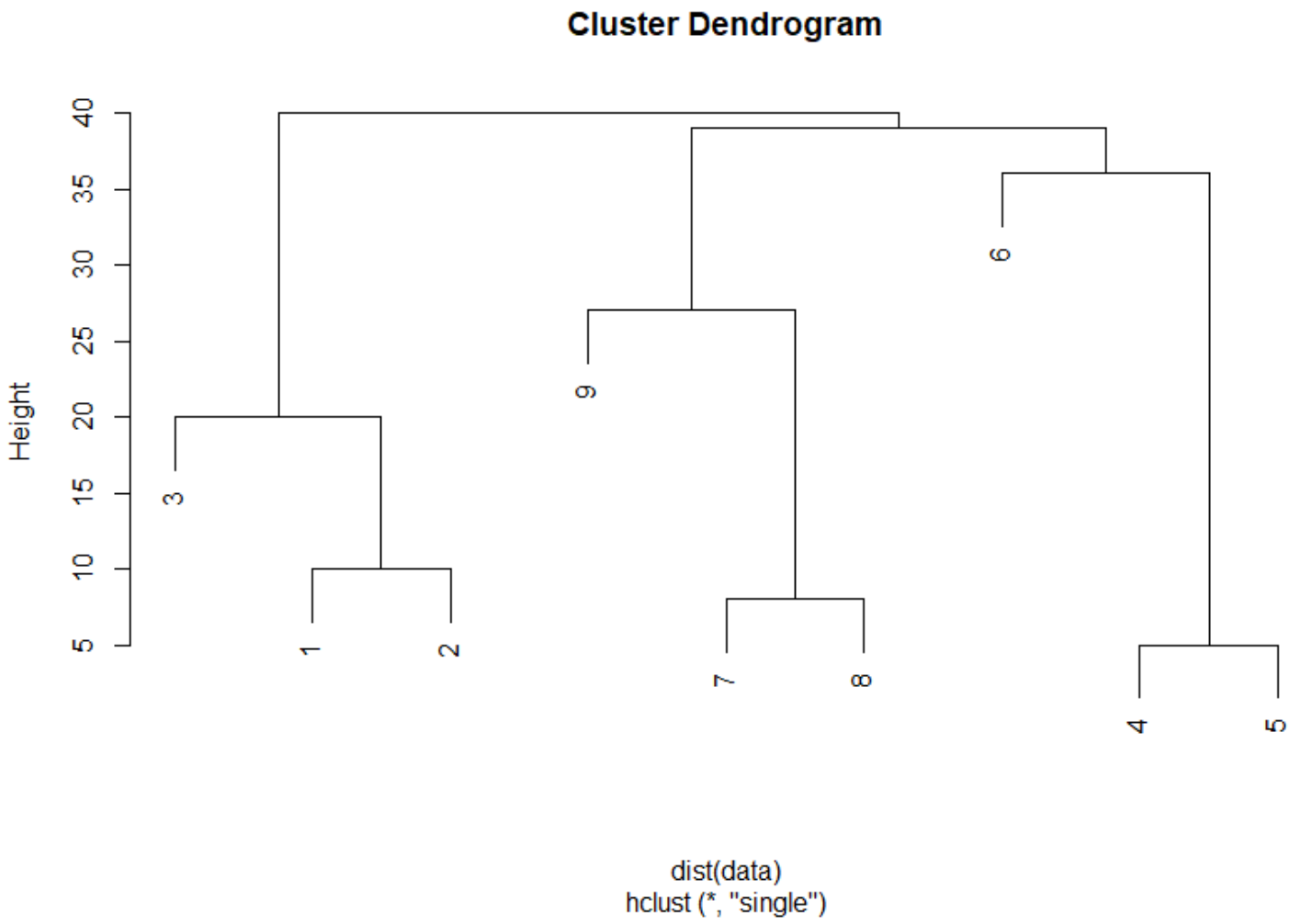


	10-20-40	121-80-85	168-168-195
10-20-40	0		
80-121-85	40	0	
160-168-195	120	39	0



	10-20-40	160-168-195-121-80-85
10-20-40	0	
160-168-195-121-80-85	40	0



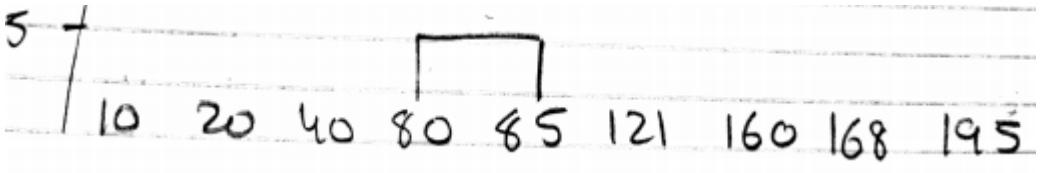


5 8 10 20 27 36 39 40

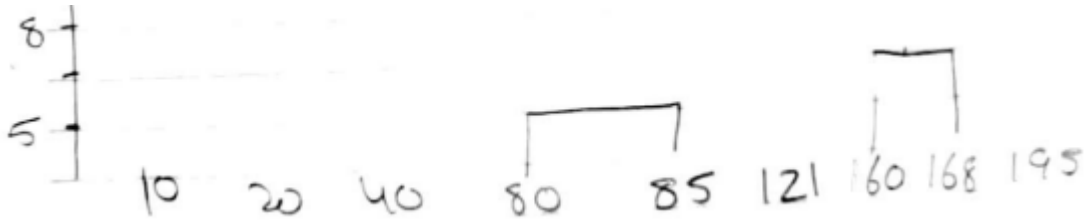
hierarchical agglomerative clustering with complete linkage

	10	20	40	80	85	121	160	168	195
10	0								
20	10	0							
40	30	20	0						

80	70	60	40	0					
85	75	65	45	5	0				
121	111	101	81	41	36	0			
160	150	140	120	80	75	39	0		
168	158	148	128	88	83	47	8	0	
195	185	175	155	115	110	74	35	27	0

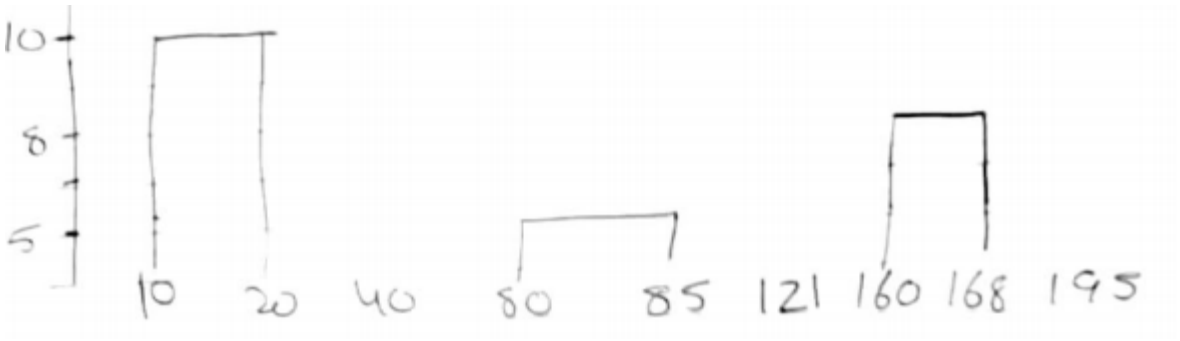


	10	20	40	80-85	121	160	168	195
10	0							
20	10	0						
40	30	20	0					
80-85	75	65	45	0				
121	111	101	81	41	0			
160	150	140	120	80	39	0		
168	158	148	128	88	47	8	0	
195	185	175	155	115	74	35	27	0

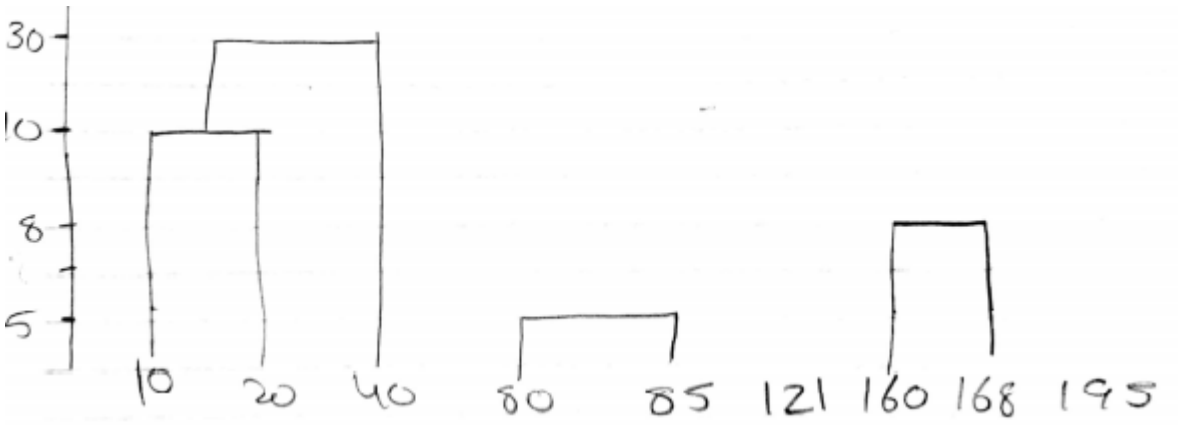


	10	20	40	80-85	121	160-168	195
10	0						

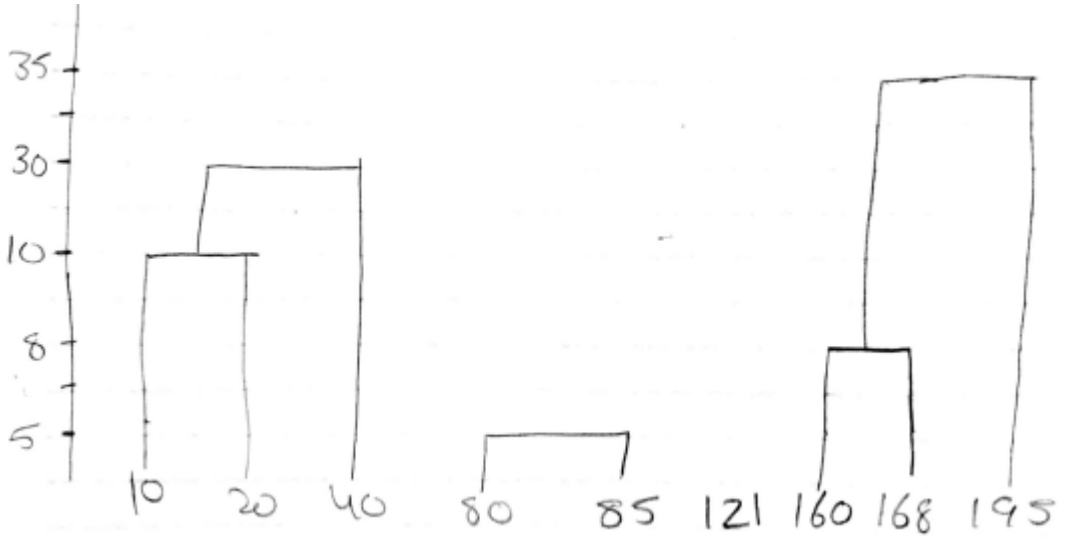
20	10	0					
40	30	20	0				
80-85	75	65	45	0			
121	111	101	81	41	0		
160-168	158	148	128	88	47	0	
195	185	175	155	115	74	35	0



	10-20	40	80-85	121	160-180	195
10-20	0					
40	30	0				
80-85	75	45	0			
121	111	81	41	0		
160-168	158	128	88	47	0	
195	185	155	115	74	35	0

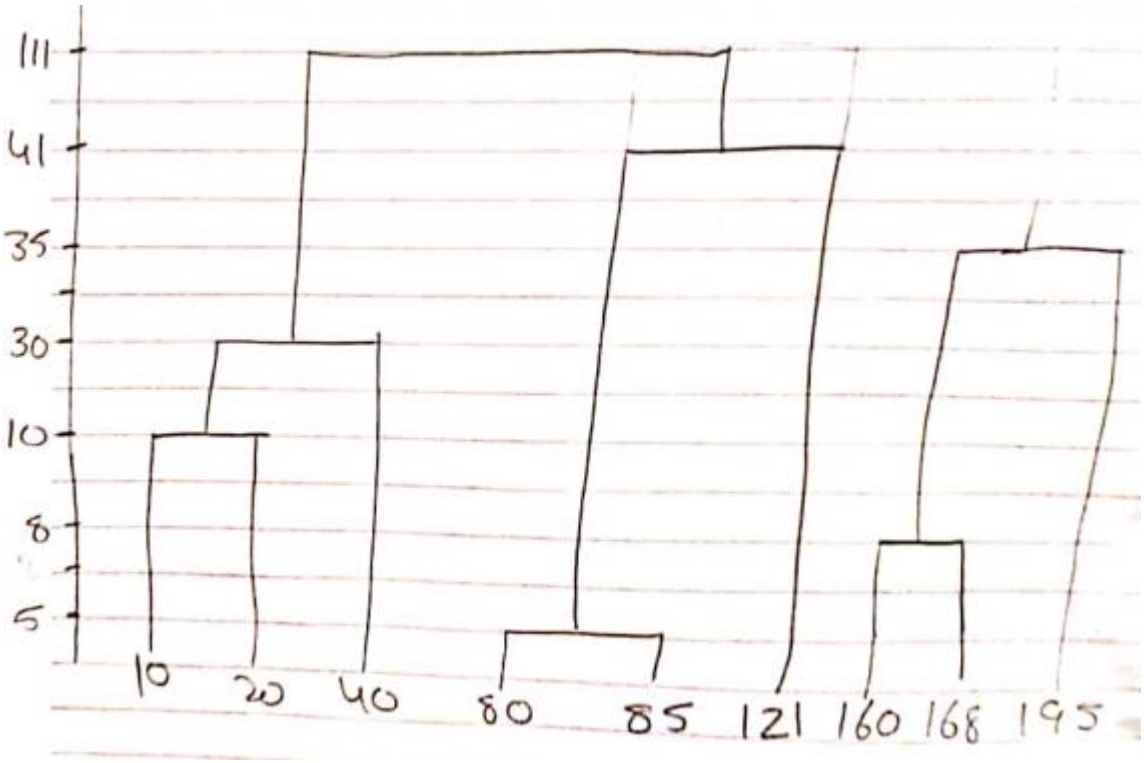


	10-20-40	80-85	121	160-180	195
10-20-40	0				
80-85	75	0			
121	111	41	0		
160-168	158	88	47	0	
195	185	115	74	35	0

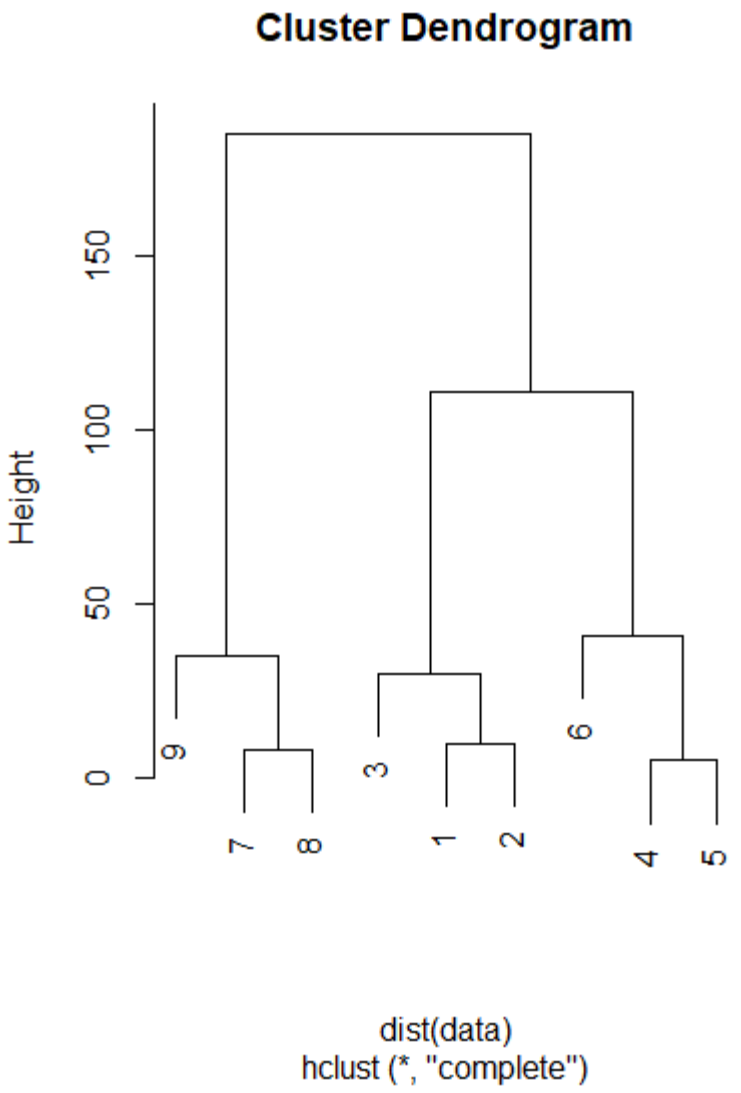


	10-20-40	80-85	121	160-168-195
10-20-40	0			
80-85	75	0		
121	111	41	0	
160-168-195	185	115	74	0

	10-20-40	80-85-121	160-168-195
10-20-40	0		
80-85-121	111	0	
160-168-195	185	115	0



	10-20-40-80-85-121	160-168-195
10-20-40-80-85-121	0	
160-168-195	185	0



5-8-10-30-35-41-111-185

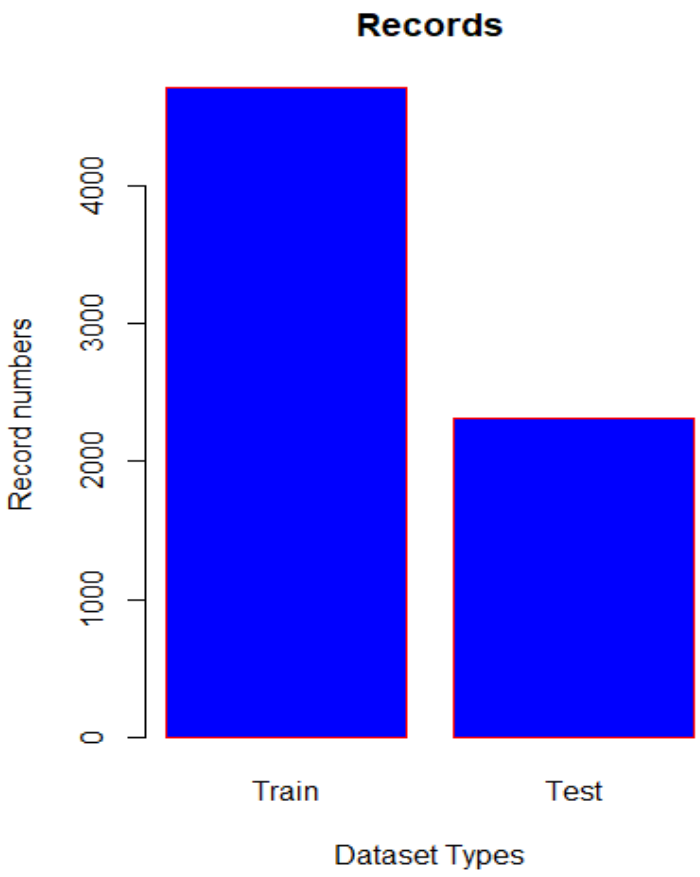
Part B: Model Evaluation & Performance Improvement

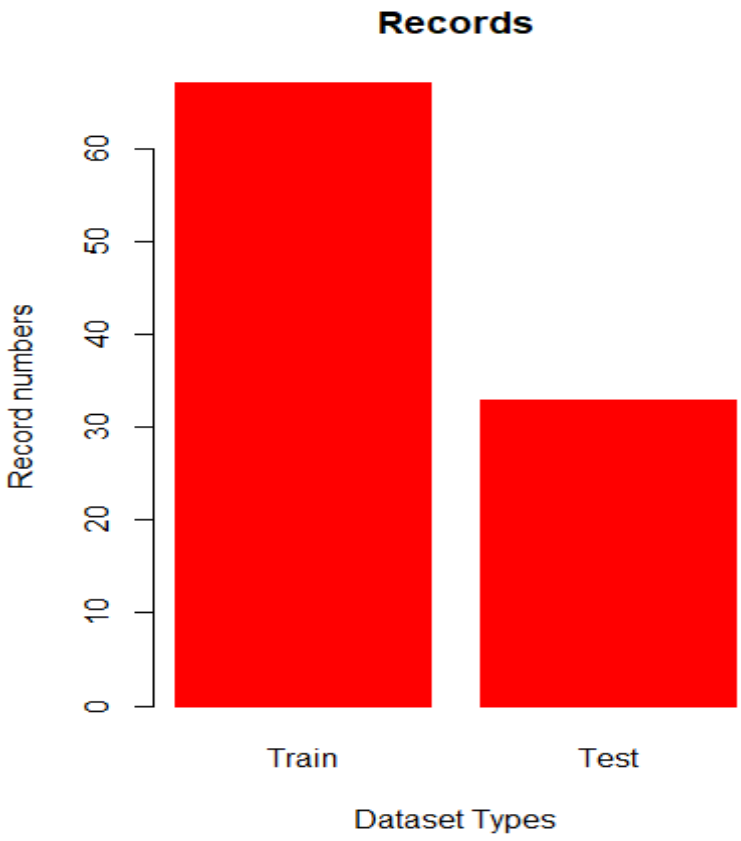
a) Partition the data set using the holdout method, so that 67% of the records are included in the training data set and 33% are included in the test data set. Use a bar graph to confirm your proportions.

```
# Holdout method
# using caret function
in_train <- createDataPartition(dataset$Churn, p = 0.67, list = FALSE)
train <- dataset[in_train, ]
test <- dataset[-in_train, ]

train_ <- nrow(train)
test_ <- nrow(test)
total<- c(nrow(train),nrow(test))
total_names=c('Train','Test')
barplot(total,names.arg=total_names,xlab="Dataset Types",ylab="Record numbers",col="blue",
        main="Records",border="red")
percentage=train_/(train_+test_)*100
percentage

total1<- c(train_/(train_+test_)*100,test_/(train_+test_)*100)
barplot(total1,names.arg=total_names,xlab="Dataset Types",ylab="Record numbers",col="red",
        main="Records",border="red")
```





b) Identify the total number of records in the training data set and how many records in the training data set have a churn value of true (or 1). Calculate how many true churn records you need to resample to have 30% of the rebalanced data set have true churn values. (Changed to 30 % as the announcemnet)

```
train_ <- nrow(train)
total_records_train=c(nrow(train))
total_records_train
true_train=train[train$Churn == "Yes", ]
no.m_Of_Records_true=nrow(true_train)
no.m_Of_Records_true
rebalance_needed=no.m_Of_Records_true/total_records_train*100
rebalance_needed
rebalance_needed=rebalance_needed-30
rebalance_needed
total_records_train
rebalance_needed=rebalance_needed*total_records_train/100
rebalance_needed=abs(rebalance_needed)
(rebalance_needed+no.m_Of_Records_true)/total_records_train
(no.m_Of_Records_true-rebalance_needed)/total_records_train*100
```

Train records: 4713 records

Training data set have a churn value of true: 1253 records.

true churn records you need to resample to have 30%: 161 records.

c) Perform the rebalancing described in (b) and confirm that 30% of the records in the rebalanced data set have true churn values. (Changed to 30 % as the announcemnet)

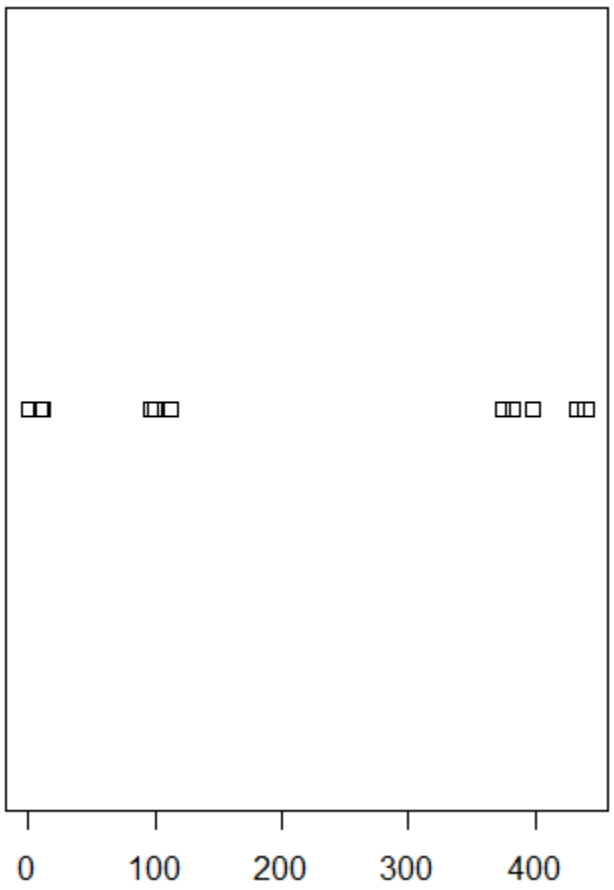
```
train<- ovun.sample(Churn~., data = train, method = "over", p = 0.3)$data
```

d) Use predictors you think are appropriate and obtain the predicted value.

Create a temp tree and use feature importance.

```
ctrl <- trainControl(selectionFunction='best',classProbs = TRUE,savePredictions = TRUE,
                     summaryFunction = twoClassSummary)
#specify method as class since we are dealing with classification
model <- rpart(Churn ~ ., data = train, method = "class")
# summarize importance
# estimate variable importance
importance <- varImp(model, scale=FALSE)
print(importance)
# plot importance
plot(importance)
```

```
                   Overall
Contract         397.348388
InternetService  112.891638
MonthlyCharges    97.017382
OnlineBackup      10.149199
OnlineSecurity   439.783713
PaperlessBilling  10.609630
PaymentMethod    373.907145
StreamingMovies    9.783706
TechSupport      432.263601
tenure           382.378571
TotalCharges     100.240513
gender             0.000000
SeniorCitizen      0.000000
Partner            0.000000
Dependents         0.000000
PhoneService       0.000000
MultipleLines      0.000000
DeviceProtection   0.000000
StreamingTV        0.000000
```

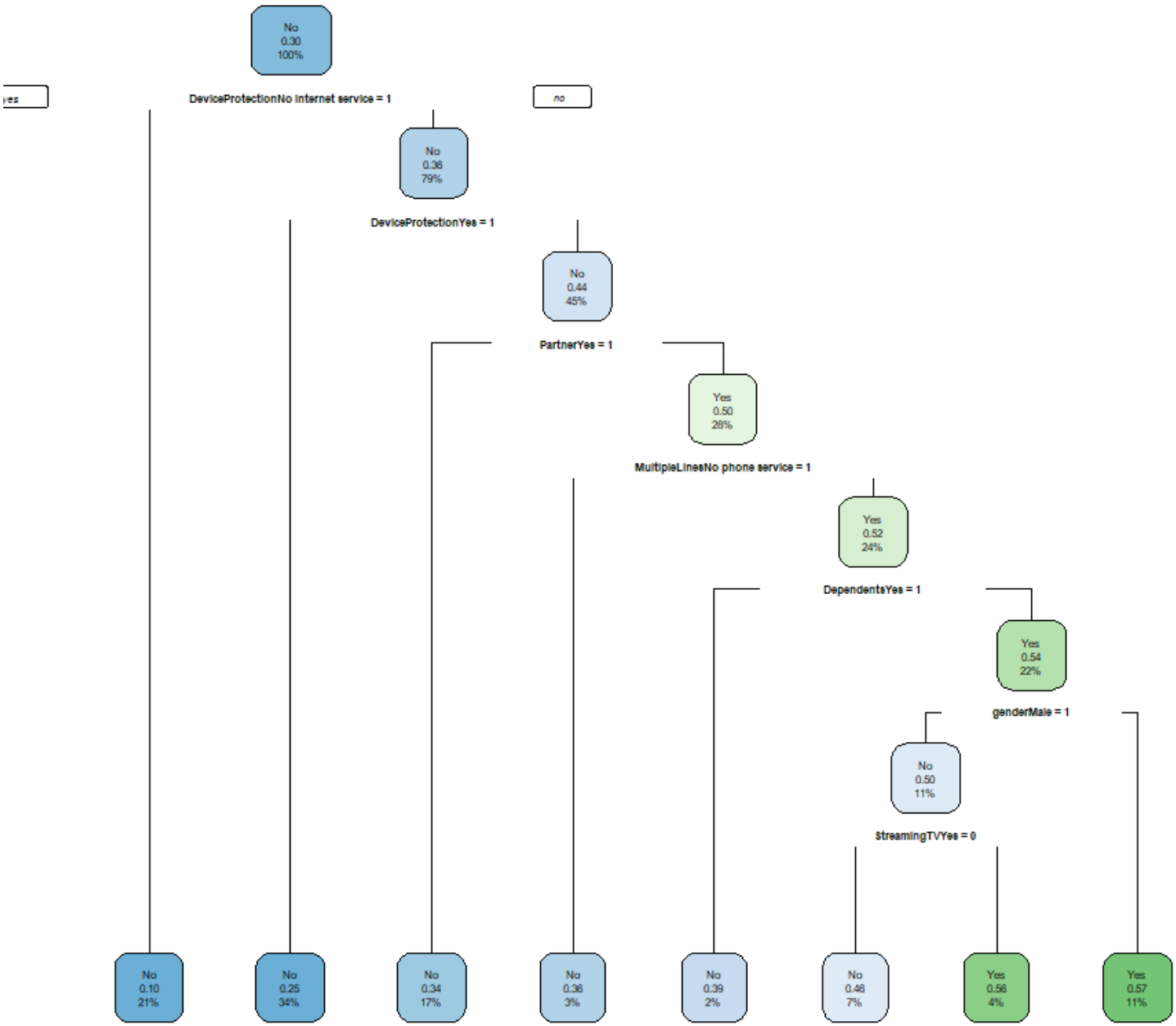


And dropped Id as it does not help the model.

```
drop <- c('gender','SeniorCitizen','Partner',
          'Dependents','PhoneService','MultipleLines'
          ,'DeviceProtection','StreamingTV','Churn')
```

Create a decision tree model that can predict Churn using the data set given.

```
train_features_selected = train[,(names(train) %in% drop)]
test_features_selected = test[,(names(test) %in% drop)]
model <- train(Churn ~ ., data = train_features_selected,
               method = "rpart", metric = "ROC",trControl = ctrl)
#plot the model
rpart.plot(model$finalModel)
```



```
          Reference
Prediction   No  Yes
       No  1516  187
       Yes  460  156

               Accuracy : 0.721
                 95% CI : (0.7023, 0.7392)
    No Information Rate : 0.8521
    P-Value [Acc > NIR] : 1

                  Kappa : 0.1671

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.7672
            Specificity : 0.4548
         Pos Pred Value : 0.8902
         Neg Pred Value : 0.2532
             Prevalence : 0.8521
         Detection Rate : 0.6537
   Detection Prevalence : 0.7344
      Balanced Accuracy : 0.6110

       'Positive' Class : No
```

e) Use an ensemble method (e.g., Random Forest, Adaboost) to obtain the predicted value of Churn.

```
rfpreds <- train(Churn ~ ., data = train , metric = "ROC" , method = "ranger",trControl = ctrl)
fpreds <- predict(rfpreds, newdata = test)
#Print the confusion Matrix
confusionMatrix(test$Churn, fpreds)
```

```
                Accuracy : 0.7831
                  95% CI : (0.7658, 0.7997)
     No Information Rate : 0.7943
     P-Value [Acc > NIR] : 0.9127

                   Kappa : 0.4009

  Mcnemar's Test P-Value : 7.598e-10

             Sensitivity : 0.8257
             Specificity : 0.6184
          Pos Pred Value : 0.8931
          Neg Pred Value : 0.4789
              Prevalence : 0.7943
          Detection Rate : 0.6559
    Detection Prevalence : 0.7344
       Balanced Accuracy : 0.7221

        'Positive' Class : No
```

f) Tune the hyper-parameters (e.g., node size, max depth, max terminal nodes, etc.) of the ensemble model and compare against the initial model.

```
grid_rf <- expand.grid(
  mtry = 2:7,
  splitrule="gini",
  min.node.size=2:5

  )
m_rftune <- train(Churn ~ ., data = train, method = "ranger",
                  tuneGrid = grid_rf,trControl = ctrl )

rfpredstune <- predict(m_rftune, newdata = test)

m_rftune$finalModel$min.node.size
m_rftune$finalModel$mtry
> #Print the confusion Matrix
> confusionMatrix(test$Churn, rfpredstune)
Confusion Matrix and Statistics

          Reference
Prediction   No  Yes
       No  1524  179
       Yes  297  319

               Accuracy : 0.7947
                 95% CI : (0.7777, 0.811)
    No Information Rate : 0.7853
    P-Value [Acc > NIR] : 0.1383

                  Kappa : 0.4396

 Mcnemar's Test P-Value : 8.199e-08

            Sensitivity : 0.8369
            Specificity : 0.6406
         Pos Pred Value : 0.8949
         Neg Pred Value : 0.5179
             Prevalence : 0.7853
         Detection Rate : 0.6572
   Detection Prevalence : 0.7344
      Balanced Accuracy : 0.7387

       'Positive' Class : No
```

g) Using a confusion matrix, compare the evaluation measures from the ensemble method with the decision tree model based on the following criteria: Accuracy, Sensitivity and Specificity. Identify the model that performed best and worst according to each criterion.

	Accuracy	Sensitivity	Specificity
decision tree	0.721	0.7672	0.4548
Random forest	0.721	0.7672	0.4548
Tuned Random Forest Best(min node size=2 &mtry=6)	0.7947	0.8369	0.6406

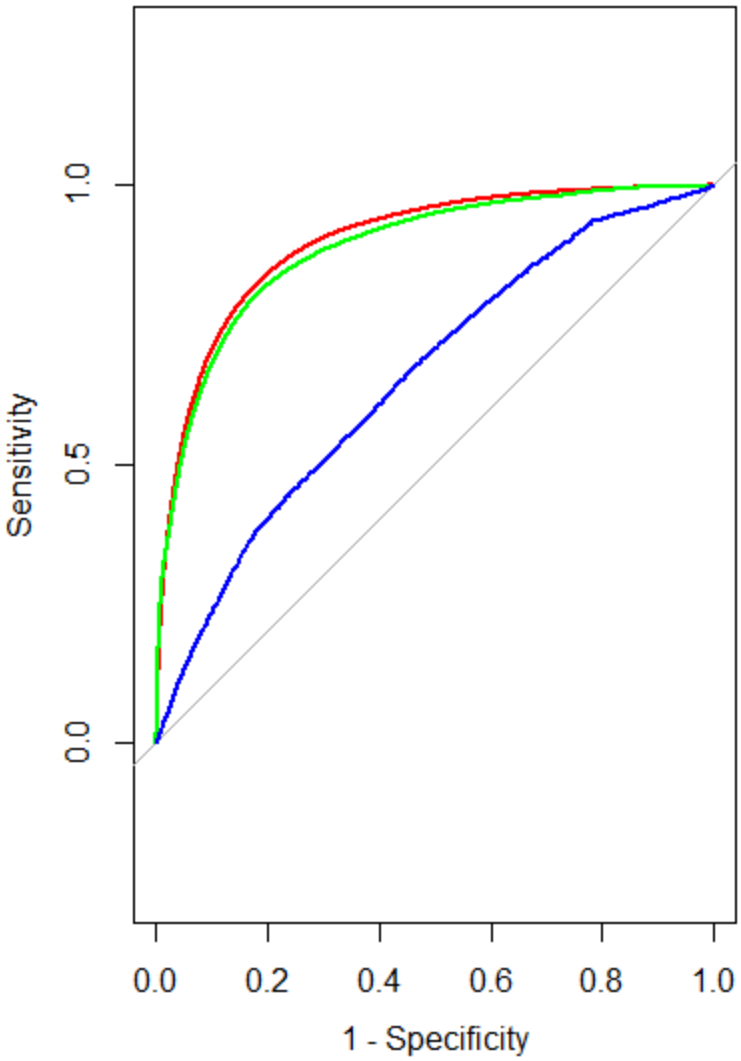
Model with best accuaracy : Tuned Random Forest Best(min node size=2 &mtry=6)
Model with worst accuaracy : decision tree and are equal.

Model with best Sensitivity: Tuned Random Forest Best(min node size=2 &mtry=6)
Model with worst Sensitivity: decision tree and are equal

Model with best Specificity: Tuned Random Forest Best(min node size=2 &mtry=6)
Model with worst Specificity: decision tree and are equal

g) Carry out a ROC analysis to compare the performance of the ensemble method with the decision tree technique. Plot the ROC graph of the models.

```
roc_rf_tune <- roc(m_rftune$pred$obs, m_rftune$pred$Yes)
roc_rf <- roc(rfpreds$pred$obs, rfpreds$pred$Yes)
tree <- roc(model$pred$obs, model$pred$Yes)
plot(roc_rf_tune, col = "red", legacy.axes = TRUE)
plot(roc_rf, col = "green", add = TRUE)
plot(tree, col = "blue", add = TRUE)
```



Best model is Tuned Random Forest

Best(min node size=2 &mtry=6) then normal random forest and then decision tree