Navarro_Assignment_2

John Navarro

January 16, 2017

komeans code from Anil

```
fun.okc.2= function (data = data, nclust = nclust, lnorm = lnorm, tolerance = tolerance)
{
    M = nrow(data)
    N = ncol(data)
    K = nclust
    niterations = 50
#    datanorm = apply(data, 2, fun.normalize)
    datanorm = scale(data)
    S = matrix(sample(c(0, 1), M * K, replace = TRUE), M, K)
    S = cbind(S, rep(1, M))
    W = matrix(runif(N * K), K, N)
    W = rbind(W, rep(0, N))
    sse = rep(0, niterations)
    oprevse = exp(70)
    opercentse = 1
    i = 1
    while ((i <= niterations) & (opercentse > tolerance)) {
        for (k in 1:K) {
            sminusk = S[, -k]
            wminusk = W[-k, ]
            s = as.matrix(S[, k])
            w = t(as.matrix(W[k, ]))
            dstar = datanorm - sminusk %*% wminusk
            prevse = exp(70)
            percentse = 1
            l = 1
            while ((l <= niterations) & (percentse > tolerance)) {
                for (m in 1:N) {
                  if (lnorm == 2) {
                    w[1, m] = mean(dstar[s == 1, m], na.rm = TRUE)
                  }
                  if (lnorm == 1) {
                    w[1, m] = median(dstar[s == 1, m], na.rm = TRUE)
                  }
                }
                for (m in 1:M) {
                  if (lnorm == 2) {
                    ss1 = sum((dstar[m, ] - w[1, ])^2, na.rm = TRUE)
                    ss0 = sum((dstar[m, ])^2, na.rm = TRUE)
                  }
                  if (lnorm == 1) {
                    ss1 = sum(abs(dstar[m, ] - w[1, ]), na.rm = TRUE)
                    ss0 = sum(abs(dstar[m, ]), na.rm = TRUE)
                  }
                  if (ss1 <= ss0) {
```

```
                    s[m, 1] = 1
                  }
                  if (ss1 > ss0) {
                    s[m, 1] = 0
                  }
                }
                if (sum(s) == 0) {
                  s[sample(1:length(s), 2)] = 1
                }
                if (lnorm == 2) {
                  se = sum((dstar - s %*% w)^2, na.rm = TRUE)
                }
                if (lnorm == 1) {
                  se = sum(abs(dstar - s %*% w), na.rm = TRUE)
                }
                percentse = 1 - se/prevse
                prevse = se
                l = l + 1
            }
            S[, k] = as.vector(s)
            W[k, ] = as.vector(w)
        }
        if (lnorm == 2)
            sse[i] = sum((datanorm - S %*% W)^2, na.rm = TRUE)/sum((datanorm -
                mean(datanorm, na.rm = TRUE))^2, na.rm = TRUE)
        if (lnorm == 1)
            sse[i] = sum(abs(datanorm - S %*% W), na.rm = TRUE)/sum(abs(datanorm -
                median(datanorm, na.rm = TRUE)), na.rm = TRUE)
        if (lnorm == 2) {
            ose = sum((datanorm - S %*% W)^2, na.rm = TRUE)
        }
        if (lnorm == 1) {
            ose = sum(abs(datanorm - S %*% W), na.rm = TRUE)
        }
        opercentse = (oprevse - ose)/oprevse
        oprevse = ose
        i = i + 1
    }
    if (lnorm == 2)
        vaf = cor(as.vector(datanorm), as.vector(S %*% W), use = "complete.obs")^2
    if (lnorm == 1)
        vaf = 1 - sse[i - 1]
     rrr = list(Data = data, Normalized.Data = datanorm, Tolerance = tolerance,
        Groups = S[, 1:K], Centroids = round(W[1:K, ], 2), SSE.Percent = sse[1:i -
            1], VAF = vaf)

    return(rrr)
}

komeans=function (data = data, nclust = nclust, lnorm = lnorm, nloops = nloops, tolerance = tolerance,
{
    prevsse = 100
    set.seed(seed)
```

```
    for (i in 1:nloops) {
        z = fun.okc.2(data = data, nclust = nclust, lnorm = lnorm,
            tolerance = tolerance)
        if (z$SSE.Percent[length(z$SSE.Percent[z$SSE.Percent >  0])] < prevsse) {
            prevsse = z$SSE.Percent[length(z$SSE.Percent[z$SSE.Percent >  0])]
            ind = i
            z.old = z
        }
    }
    return(list(data = z.old$Data, Normalized.Data = z.old$Normalized.Data,
        Group = z.old$Group %*% as.matrix(2^(0:(nclust-1)) ), Centroids = z.old$Centroids, Tolerance = z
        SSE.Pecent = z.old$SSE.Percent, VAF = z.old$VAF, iteration = ind,
        seed = seed))
}
```

1. Select the numeric variables that you think are appropriate and useful

```
#Load the German data
AssignmentData<- read.csv("C:/Users/JohntheGreat/Documents/MSCA/DataMining/germandata.csv", header=FALSE
head(AssignmentData)
```

```
##    V1 V2  V3  V4   V5  V6  V7 V8  V9  V10 V11  V12 V13  V14  V15 V16  V17
## 1 A11  6 A34 A43 1169 A65 A75  4 A93 A101   4 A121  67 A143 A152   2 A173
## 2 A12 48 A32 A43 5951 A61 A73  2 A92 A101   2 A121  22 A143 A152   1 A173
## 3 A14 12 A34 A46 2096 A61 A74  2 A93 A101   3 A121  49 A143 A152   1 A172
## 4 A11 42 A32 A42 7882 A61 A74  2 A93 A103   4 A122  45 A143 A153   1 A173
## 5 A11 24 A33 A40 4870 A61 A73  3 A93 A101   4 A124  53 A143 A153   2 A173
## 6 A14 36 A32 A46 9055 A65 A73  2 A93 A101   4 A124  35 A143 A153   1 A172
##   V18  V19  V20 V21
## 1   1 A192 A201   1
## 2   1 A191 A201   2
## 3   2 A191 A201   1
## 4   2 A191 A201   1
## 5   2 A191 A201   2
## 6   2 A192 A201   1
```

The problem is to cluster the participants in the German credit study. Due to the difficulty with clustering and categorical data. I am choosing to only consider the numeric categories.

```
num.cat <- AssignmentData[,c(2,5,8,11,13,16,18)]
head (num.cat)
```

```
##   V2   V5 V8 V11 V13 V16 V18
## 1  6 1169  4   4  67   2   1
## 2 48 5951  2   2  22   1   1
## 3 12 2096  2   3  49   1   2
## 4 42 7882  2   4  45   1   2
## 5 24 4870  3   4  53   2   2
## 6 36 9055  2   4  35   1   2
```

```
cor(num.cat)
```

```
##              V2          V5          V8        V11         V13         V16
## V2   1.00000000  0.62498420  0.07474882 0.03406720 -0.03613637 -0.01128360
## V5   0.62498420  1.00000000 -0.27131570 0.02892632  0.03271642  0.02079455
## V8   0.07474882 -0.27131570  1.00000000 0.04930237  0.05826568  0.02166874
## V11  0.03406720  0.02892632  0.04930237 1.00000000  0.26641918  0.08962523
## V13 -0.03613637  0.03271642  0.05826568 0.26641918  1.00000000  0.14925358
## V16 -0.01128360  0.02079455  0.02166874 0.08962523  0.14925358  1.00000000
## V18 -0.02383448  0.01714215 -0.07120694 0.04264343  0.11820083  0.10966670
##             V18
## V2  -0.02383448
## V5   0.01714215
## V8  -0.07120694
## V11  0.04264343
## V13  0.11820083
## V16  0.10966670
## V18  1.00000000
```

After looking at the 7 numeric variables, I believe that the following attributes are most appropriate: 5 - Credit Amount (numerical) 8 - Installment rate in percentage of disposable income(numerical) 11 - Present residence since(numerical) 13 - Age (numerical)

2. Use kmeans and komeans

3. Generate the K-means solution.

```
#scale the data
attributes <- AssignmentData[, c(5, 8, 11, 13)]
scaled_attributes <- scale(attributes)
head(scaled_attributes)
```

```
##              V5          V8        V11         V13
## [1,] -0.7447588  0.91801781  1.0464631  2.76507291
## [2,]  0.9493418 -0.86974813 -0.7655942 -1.19080809
## [3,] -0.4163541 -0.86974813  0.1404344  1.18272051
## [4,]  1.6334296 -0.86974813  1.0464631  0.83108664
## [5,]  0.5663801  0.02413484  1.0464631  1.53435438
## [6,]  2.0489838 -0.86974813  1.0464631 -0.04799802
```

```
# need to split the data into train/test portions
set.seed(555)
train_ind <- sample(seq_len(nrow(scaled_attributes)), size = 700)
# separate into two data frames: train and test
train_data <- scaled_attributes[train_ind, ]
test_data<- scaled_attributes[-train_ind, ]
```

Extract 2-10 k-means clusters using the variable set. Present the VAF. Run them from at least 50-100 random starts

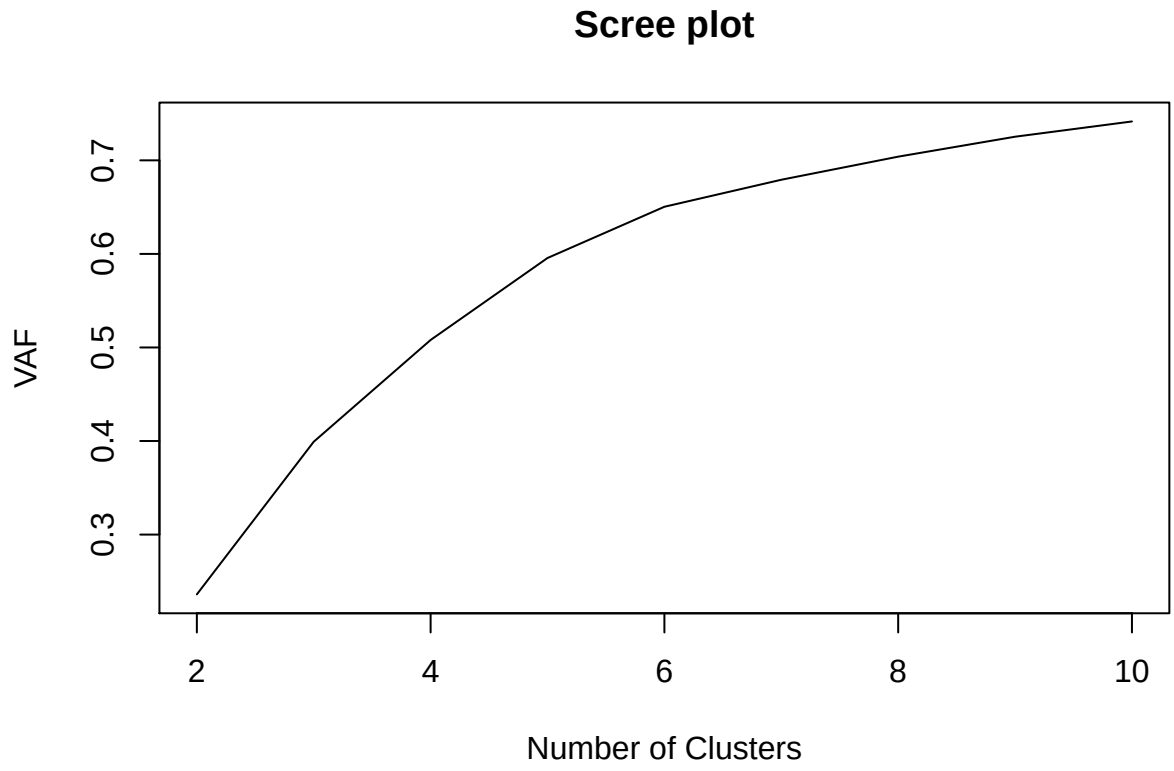
```r
#create VAF.vector where VAF's will be stored
VAF.vector <- numeric(0)
#set the seed for reproducability
set.seed(202)
# Use a for loop to extract the VAF for each number of clusters
for (i in 2:10){
  km.output <- kmeans(train_data, centers=i, nstart=50)
  VAF.vector[i-1] <- km.output$betweenss/km.output$totss
}
print(VAF.vector)
```

```
## [1] 0.2360957 0.3992483 0.5080213 0.5956196 0.6503913 0.6792559 0.7039342
## [8] 0.7252735 0.7415598
```

4. Perform Scree tests to choose appropriate number of k-means clusters

5. Show the scree plot

```r
plot(c(2:10), VAF.vector, xlab = 'Number of Clusters', ylab = 'VAF', type= "l", main='Scree plot')
```



6. Choose 1 K-means solution to retain

6a. Use the Criteria of VAF

Based on this plot, there appears to be a bend in at either 5 or 6 clusters. Since these are high numbers to separate the customer base into, I will go with the lesser of the two. Going forward I will investigate using 5 clusters.

6b. Interpretability of the segments

We can look at the centers of the 5 clusters to describe our 5 segments

```
km.output.5 <- kmeans(train_data, centers=5, nstart=50)
km.output.5$centers
```

```
##           V5         V8         V11         V13
## 1 -0.3364722  0.7268405  0.80031655 -0.35542359
## 2  2.2951135 -0.7565984  0.09455957  0.07885723
## 3 -0.2618186  0.4686340 -1.10226055 -0.38137767
## 4 -0.2557656  0.3750048  0.78396878  1.66498472
## 5 -0.1417463 -1.2544573  0.08309085 -0.39740636
```

Cluster 1 is a younger person with an above average residence rate who has a high rate on an small loan amount. Cluster 2 is an average aged person with a average residence length who has a low rate on a very high loan amount. Cluster 3 is a younger person with a very short residence rate who has a moderately high rate on a below average loan amount. Cluster 4 is an elderly person with a long residence who has an above average rate on a below average loan amount. Cluster 5 is a younger person with a long residence length who has a verylow rate on an average loan amount.

These clustered segments all seem to be separate groups of people with different traits. The 2 that seem to be closest together are clusters 1 and 3 which both have young people with moderately high rates on low loan amounts, the only difference is the length of residence, but in that regard, the two groups are very different.

6c. Performance of the holdout

```
TestCluster <- kmeans(test_data, centers=km.output.5$centers)
TestVAF <- TestCluster$betweenss/TestCluster$totss
TestClustersizes <- TestCluster$size
print(c(VAF.vector[4], TestVAF))
```

```
## [1] 0.5956196 0.6141321
```

```
TrainClustersizes <- km.output.5$size
scaled.train <- scale(TrainClustersizes)
print(scaled.train[1:5])
```

```
## [1]  0.7346041 -1.3579046  0.9572114 -0.7346041  0.4006932
```

```
scaled.test <- scale(TestClustersizes)
print(scaled.test[1:5])
```

```
## [1]  0.2057596 -1.3991651  1.1111017 -0.5761268  0.6584306
```

It appears that the VAFs are close in both the train and the test set. Since the two sets are different sizes, I scaled each of the cluster sizes. They seem similarly sized, the only difference is cluster 4, which seems to be slightly bigger in the training set than the test set.

7. Generate 3-5 komeans clusters on the training data

```
ko.output.3 <- komeans(data=train_data, nclust=3, lnorm=2, nloops=100, tolerance=0.001, seed=5)
ko.output.4 <- komeans(data=train_data, nclust=4, lnorm=2, nloops=100, tolerance=0.001, seed=5)
ko.output.5 <- komeans(data=train_data, nclust=5, lnorm=2, nloops=100, tolerance=0.001, seed=5)
ko.VAF.5 <- ko.output.5$VAF
ko.VAF.4 <- ko.output.4$VAF
```

8. Compare the chosen k-means solution with a komeans solution

Since I used 5 clusters for the k-means solution, we will compare that with the 5 cluster solution from komeans. We will compare cluster centroids

```
#Create the table
table(km.output.5$cluster, ko.output.5$Group)
```

```
##
##      0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16
##   1  6   1   7   0   0   0   0   0   0   0   0   0   0   0   0   0  12
##   2  0   0   0   0   0   0   0   0   5  14  11  12   5   7   3   2   0
##   3  3   7   0   0   5  20   0   0   0   0   0   0   0   2   0   0   1
##   4  0   0   6   0   3   0  17   0   0   0   1   0   0   0   4   0   0
##   5  1   8   8  49   3  48   3  29   0   1   0   3   0   3   0   0   0
##
##      17  18  19  20  21  22  23  24  25  26  27  28  29  30  31
##   1   0  35  64   0   0   0  34   1   0   4   9   0   0   0   0
##   2   0   0   0   0   0   0   0   3   4   4   2   1   2   0   4
##   3  10   0   0  17 102   0   1   0   8   0   0   2   5   0   0
##   4   0   6   0   8   0  51   1   0   0   4   0   0   0   4   2
##   5   0   0   0   0   0   0   0   0   0   0   0   0   0   0   2
```

```
#Choose the 5 largest clusters. 21/3 = 102, 19/1 = 64, 22/4 = 51, 3/5 = 49, 5/5 = 48
# use apply to take the mean of the ko cluster groups
ko.5.groupmeans.1 <- apply(train_data[ko.output.5$Group == 21,], 2, mean)
ko.5.groupmeans.2 <- apply(train_data[ko.output.5$Group == 19,], 2, mean)
ko.5.groupmeans.3 <- apply(train_data[ko.output.5$Group == 22,], 2, mean)
ko.5.groupmeans.4 <- apply(train_data[ko.output.5$Group == 3,], 2, mean)
ko.5.groupmeans.5 <- apply(train_data[ko.output.5$Group == 5,], 2, mean)

print(ko.5.groupmeans.1)
```

```
##         V5         V8        V11        V13
## -0.4519474  0.6814017 -1.0942516 -0.5073629
```

```
print(ko.5.groupmeans.2)
```

```
##         V5         V8        V11        V13
## -0.4566961  0.7923155  0.8765827 -0.9257091
```

```
print(ko.5.groupmeans.3)
```

```
##         V5         V8        V11        V13
## -0.5986622  0.8128551  0.8865757  1.7808428
```

```
print(ko.5.groupmeans.4)
```

```
##         V5         V8        V11        V13
## -0.1353560 -1.2710833  0.8245785 -0.9324648
```

```
print(ko.5.groupmeans.5)
```

```
##         V5         V8        V11        V13
## -0.3132053 -1.1720909 -1.1653127 -0.4487572
```

It seems like the results are different. The kmeans VAF is lower than the komeans VAF. Additionally, the komeans algorithm has identified different centroids than the results from the kmeans calculations.

9. Summarize and Interpret the results and segments

Here are demographic descriptions of the 5 segments that Komeans has created:

Cluster 1 is young with a very short residence time who has a high rate on a below average sized loan. Cluster 2 is a very young person who has a long residence length with a high rate on a below average sized loan. Cluster 3 is a very old person with a long residence who has a high rate on a small loan. Cluster 4 is a very young person with long residence who has a very low rate on an average sized loan. Cluster 5 is a young person with a very short residence who has a very low rate on a small loan.

Here are the deomgraphic descriptions of the 5 segments that Kmeans has created:

Cluster 1 is a younger person with an above average residence rate who has a high rate on an small loan amount. Cluster 2 is an average aged person with a average residence length who has a low rate on a very high loan amount. Cluster 3 is a younger person with a very short residence rate who has a moderately high rate on a below average loan amount. Cluster 4 is an elderly person with a long residence who has an above average rate on a below average loan amount. Cluster 5 is a younger person with a long residence length who has a verylow rate on an average loan amount.

I would chose the Komeans segmentation method. It gives us a higher VAF, and the Clusters seem more distinct. Despite there being 4 groups with below average sized loans, they are differentiated by age and by residence length.

The kmeans method gives us a lower VAF. More importantly, the clusters don't differentiate as well. Kmeans cluster 3 and cluster 4 are both young customers with high rates on small loans. The only difference is group 3 has a short residence and group 4 has a long residence. This could be interpreted as similarly aged people with similar loans and maybe one group still lives at home with their parents (cluster 4) and the other group has recently moved out on their own (cluster 3).

10 You are given the task of recruiting 30 people into these segments

a. What approach will you take to recruit people over the telephone?

I will call them and give them a vague explanation of what the focus group is for so they are not biased. But I would incentivize them by making it a paid market study. I guess the elderly groups might require a free lunch as well, maybe some giveaways.

b. Assume consumers will be reimbursed, which of the consumers will you try to recruit?

Regardless of the compensation, I would try to recruit consumers for all segments. I assume the lower income segments should fill up quicker than the higher income segments. Hopefully, they would be interested by the topic or by a free meal. Perhaps the higher income segments would be interested in a giveaway, like raffle tickets to a musical.

c. How will you identify if a new recruit belongs to a particular segment?

I can sort the recruits by age, which would put them in 1 of 3 categories. Then by loan amount and rate, which would take them into 1 of 5 categories.