Data Mining Pset2

yunzhi wang

1/21/2018

GermanCredit Data from caret package

```
library(caret)

## Warning: package 'caret' was built under R version 3.3.2

## Loading required package: lattice

## Warning: package 'lattice' was built under R version 3.3.2

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 3.3.2

## Warning in as.POSIXlt.POSIXct(Sys.time()): unknown timezone
'zone/tz/2017c.
## 1.0/zoneinfo/America/Chicago'

data("GermanCredit")
my.data <- GermanCredit
```

1.Select the numerical variables that you think are useful

```
#1.1 Extract all numerical variables in the dataset
my.data.numerical <- my.data[, 1:7]

#1.2 Perform stepwise method and select top predictors that contribute to r^2
most.
y <- "Amount"
available.x <- colnames(my.data.numerical)[-2]
chosen.x <- NULL
r2 <- NULL

while (length(available.x) > 0) {
  best.r2 <- 0
  for (this.x in available.x) {
    rhs <- paste(c(chosen.x, this.x), collapse=" + ")
    f <- as.formula(paste(y, rhs, sep=" ~ "))
    this.r2 <- summary(lm(f, data=my.data.numerical))$r.square
    if (this.r2 > best.r2) {
      best.r2 <- this.r2
      best.x <- this.x
    }
  }
```

```r
  chosen.x <- c(chosen.x, best.x)
  available.x <- available.x[available.x != best.x]
  r2 <- c(r2, best.r2)
}
chosen.x <- c("(Intercept)", chosen.x)
r2 <- c(summary(lm(Amount ~ 1, data=my.data.numerical))$r.square, r2)

(cum.r2 <- cbind(chosen.x, r2))

##      chosen.x                   r2
## [1,] "(Intercept)"              "0"
## [2,] "Duration"                 "0.390605248125923"
## [3,] "InstallmentRatePercentage" "0.492318248181089"
## [4,] "Age"                      "0.497941569820031"
## [5,] "NumberExistingCredits"    "0.498528870877817"
## [6,] "NumberPeopleMaintenance"  "0.498531458286788"
## [7,] "ResidenceDuration"        "0.498533828611173"
```

From the table of cumulative r^2, we will select “Duration”, “InstallmentRatePercentage”, “Age” as our numerical variables for analysis.

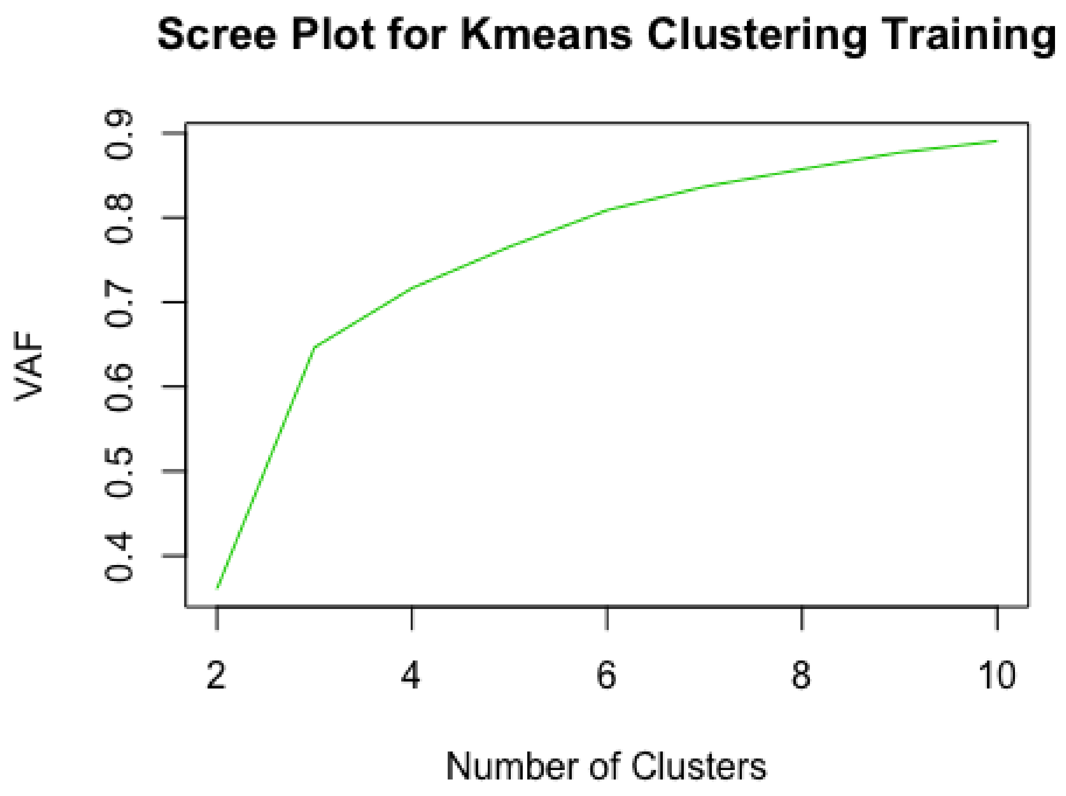
2.3.Use kmeans and komeans and Generate kmeans solution with K = 2-10, produce VAF.

```
##                 VAF
## kmclus.2  0.3605915
## kmclus.3  0.6464667
## kmclus.4  0.7165431
## kmclus.5  0.7656181
## kmclus.6  0.8087911
## kmclus.7  0.8366103
## kmclus.8  0.8572825
## kmclus.9  0.8770184
## kmclus.10 0.8905664
```

4.Perform Scree Tests to choose appropriate number of k-mean clusters

First we produce a scree plot (as shown in part 5 below) of number of clusters versus VAF (variance accounted for), where VAF is equivalent to r^2 and is the sum of squares between clusters as a percentage of total sum of squares. Usually, the closer VAF to 1 the better, but to maintain simplicity, we usually identify the “elbow” point where the rate of increase drops. In our case, the “elbow” point is at 3-clusters.
5.Show the scree plot

```r
par(mfrow=c(1,1))
plot(2:10, kmclus.vaf[1:9, 1], main = "Scree Plot for Kmeans Clustering Training",
     xlab = "Number of Clusters", ylab = "VAF", type = "l", col = "11")
```



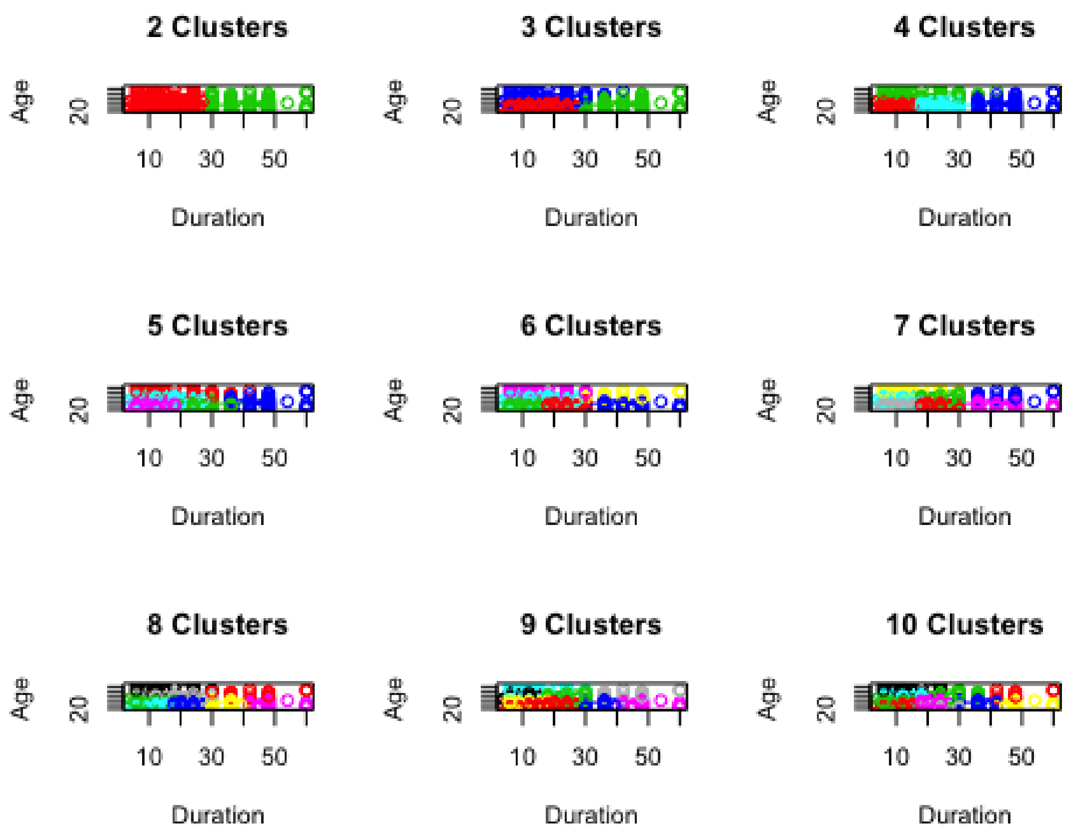
6.Choose 1 K-means solution by a) VAF cretiria; b) Intepretability of segments; c) Goodness of fit in holdout.

a. As described in 4 and 5, the K-means solution with 3 clusters is the best choice. Despite the VAF continues to increase up to 90% with 10 clusters, the rate of increase drops significantly after 3 clusters. To maintain simplicity and interpretability, I choose 3 clusters solution.

b.By generating pairwise plots among the 3 variables I choose, it is found the plot between age and duration best visualized the data. Look at the 9 plots with cluster labeled below:

```
par(mfrow = c(3,3))
plot(my.data.analysis[index.train,c(1,3)],
     col = (kmclus.2$cluster + 1), main = "2 Clusters")
plot(my.data.analysis[index.train,c(1,3)],
     col = (kmclus.3$cluster + 1), main = "3 Clusters")
plot(my.data.analysis[index.train,c(1,3)],
     col = (kmclus.4$cluster + 1), main = "4 Clusters")
plot(my.data.analysis[index.train,c(1,3)],
     col = (kmclus.5$cluster + 1), main = "5 Clusters")
plot(my.data.analysis[index.train,c(1,3)],
     col = (kmclus.6$cluster + 1), main = "6 Clusters")
plot(my.data.analysis[index.train,c(1,3)],
     col = (kmclus.7$cluster + 1), main = "7 Clusters")
```

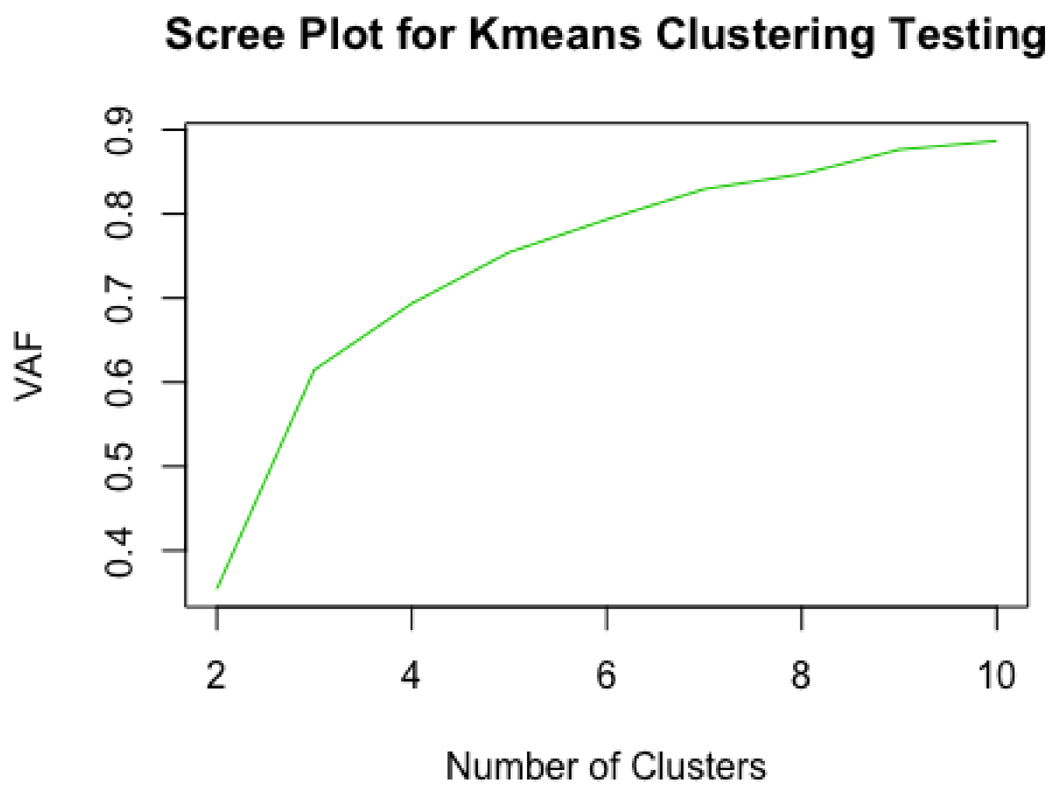
```r
plot(my.data.analysis[index.train,c(1,3)],
     col = (kmclus.8$cluster + 1), main = "8 Clusters")
plot(my.data.analysis[index.train,c(1,3)],
     col = (kmclus.9$cluster + 1), main = "9 Clusters")
plot(my.data.analysis[index.train,c(1,3)],
     col = (kmclus.10$cluster + 1), main = "10 Clusters")
```



From the plot, the solution with 3 clusters appears to have the most clear boundaries among segments and similar cluster sizes. Also, these three clusters can be easily seperated by dividing age into adult and middle-aged and by dividing duration into short-term and long-term.

c. Goodness of fit in holdout

```r
kmclus.train.centers <- list(kmclus.2$centers, kmclus.3$centers,
kmclus.4$centers,
                             kmclus.5$centers, kmclus.6$centers,
kmclus.7$centers,
                             kmclus.8$centers, kmclus.9$centers,
kmclus.10$centers)

for (i in 2:10) {
  var.name <- paste("test.kmclus.", i, sep = "")
  assign(var.name, kmeans(my.data.analysis[-index.train, ],
                          centers = kmclus.train.centers[[i - 1]], i))
}

kmclus.btwss.test <- c(test.kmclus.2$betweenss, test.kmclus.3$betweenss,
                       test.kmclus.4$betweenss, test.kmclus.5$betweenss,
                       test.kmclus.6$betweenss, test.kmclus.7$betweenss,
                       test.kmclus.8$betweenss, test.kmclus.9$betweenss,
                       test.kmclus.10$betweenss)
kmclus.totss.test <- c(test.kmclus.2$totss, test.kmclus.3$totss,
test.kmclus.4$totss,
                       test.kmclus.5$totss, test.kmclus.6$totss,
test.kmclus.7$totss,
                       test.kmclus.8$totss, test.kmclus.9$totss,
test.kmclus.10$totss)

kmclus.vaf.test <- as.matrix(kmclus.btwss.test / kmclus.totss.test)
rownames(kmclus.vaf.test) <- paste("test.kmclus.", 2:10, sep = "")
colnames(kmclus.vaf.test) <- "VAF"
kmclus.vaf.test

##                      VAF
## test.kmclus.2  0.3543643
## test.kmclus.3  0.6147590
## test.kmclus.4  0.6934925
## test.kmclus.5  0.7543745
## test.kmclus.6  0.7933338
## test.kmclus.7  0.8293784
## test.kmclus.8  0.8470861
## test.kmclus.9  0.8762834
## test.kmclus.10 0.8864032

#Scree Plot
par(mfrow=c(1,1))
plot(2:10, kmclus.vaf.test[1:9, 1], main = "Scree Plot for Kmeans Clustering
Testing",
     xlab = "Number of Clusters", ylab = "VAF", type = "l", col = "11")
```



By applying K-means model to testing data and plotting VAF for each number of clusters, we can see the VAF at each level of cluster did not vary much between training and testing sample therefore the 3-cluster assignment works equally well in the holdout sample. From the scree plot, the elbow appears at number of cluster = 3 therefore we choose 3-cluster as our final K-means solution.

7. Generate 3-5 komeans

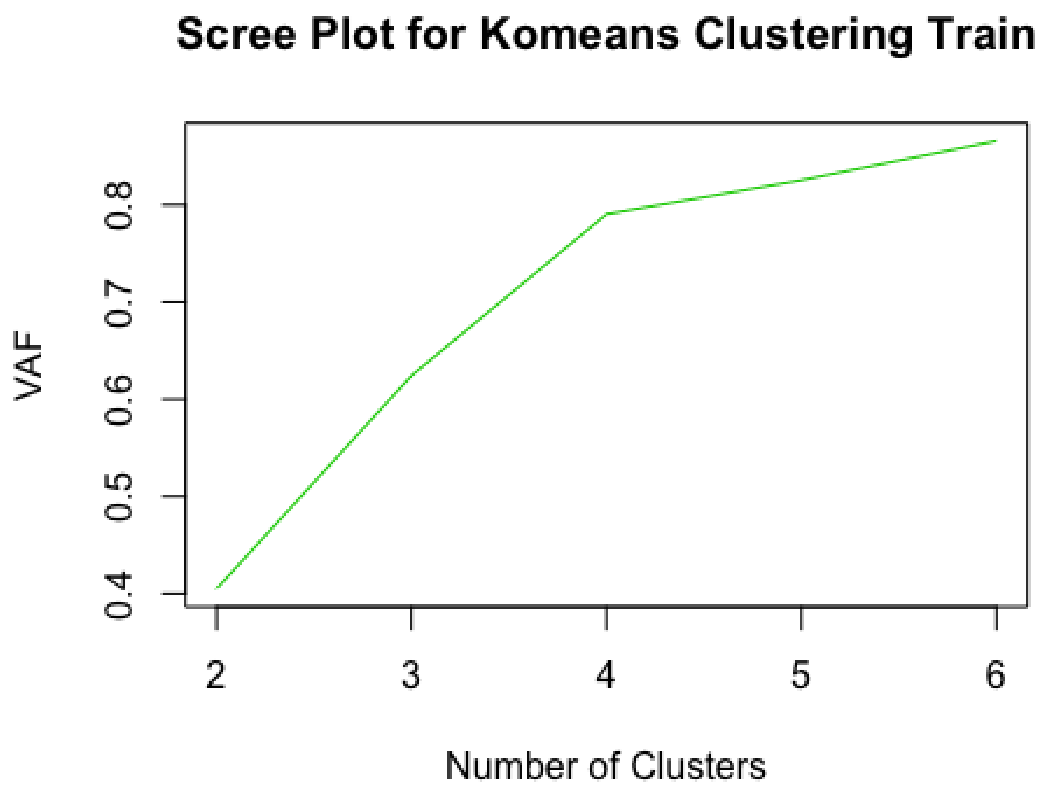
source code for komeans

```
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)
```

```r
    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, seed = seed)
{
    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.old$Tolerance,
        SSE.Pecent = z.old$SSE.Percent, VAF = z.old$VAF, iteration = ind,
        seed = seed))
}
```

generate 3-5 komeans clusters

```
for (i in 2:6) {
  var.name <- paste("komclus.", i, sep = "")
  assign(var.name, komeans(data = my.data.analysis[index.train, ], nclust =
i,
                           lnorm = 2, nloops = 50, tolerance = 0.00001, seed
= 711))
}

komclus.vaf.train <- as.matrix(c(komclus.2$VAF, komclus.3$VAF, komclus.4$VAF,
                                 komclus.5$VAF, komclus.6$VAF))
rownames(komclus.vaf.train) <- paste("komclus.", 2:6, sep = "")
colnames(komclus.vaf.train) <- "VAF"
komclus.vaf.train

##                 VAF
## komclus.2 0.4047846
## komclus.3 0.6242792
## komclus.4 0.7903929
## komclus.5 0.8253752
## komclus.6 0.8657965

plot(2:6, komclus.vaf.train[1:5, 1], main = "Scree Plot for Komeans
Clustering Train",
     xlab = "Number of Clusters", ylab = "VAF", type = "l", col = "11")
```



8.Compare the 3-cluster kmeans and komeans solution. (According to scree plot for Komeans, 4-cluster is a better solution, but we choose to use 3-cluster for interpretibility and comparison purposes.)

```
compare <- as.data.frame(matrix(c(kmclus.btwss[2] / kmclus.totss[2],
komclus.3$VAF
), nrow = 2))
rownames(compare) <- c("kmeans.3", "komeans.3")
colnames(compare) < c("VAF")

## [1] TRUE

compare

##                  V1
## kmeans.3  0.6464667
## komeans.3 0.6242792

#By looking into the komeans code, groups (1,3,5,7) belongs to the first
cluster, (2,3,6,7) belongs to the second, (4,5,6,7) belongs to third cluster.

(clus.size.km <- kmclus.3$size)

## [1] 350 125 157
```

```
(clus.size.kom <-

c(sum(komclus.3$Group==1,komclus.3$Group==3,komclus.3$Group==5,komclus.3$Grou
p ==7),

sum(komclus.3$Group==2,komclus.3$Group==3,komclus.3$Group==6,komclus.3$Group=
=7),

sum(komclus.3$Group==4,komclus.3$Group==5,komclus.3$Group==6,komclus.3$Group=
=7)))

## [1] 225 171 161
```

These two methods results in similar VAFs (64.31% and 61.70%), but k overlapping means solution has generated more evenly splitted sample sizes (kom: 235, 164, 167 while km: 124, 343, 165).

9. By performing both kmeans and komeans analysis through a number of measures including scree plots for VAF, performance in test sample, uniformity of cluster sizes as well as interpretibility, I believe 3-cluster is the best solution to split the sample. In the 3-cluster final solution, these three clusters can be easily seperated by dividing age into adult and middle-aged and by dividing duration into short-term and long-term. So one cluster represents short-term&adult, one represents long-term&adult and the last one as middle-aged&short-term.

10. You are to recruit 30 people per segment for focus groups and follow-up AU.

a. What approach will you take to recruit people over telephone?
b. Assume consumers who are recruited will be reimbursed for either focus groups or AUU. Which of the consumers will you try to recruit?
c. How will you identify if a new recruit belongs to a group?

Answer:

When recruiting people, ask questions about the 3 variables identified earlier (Age, Installment, Duration).
Compare their answers with the center of each cluster and assign them to one of the cluster according to the Euclidean distance.
Randomly seperate each cluster into two subgroups of approximately same size for focus group and AUU respectively.