Rui_Peng_Week_8.R

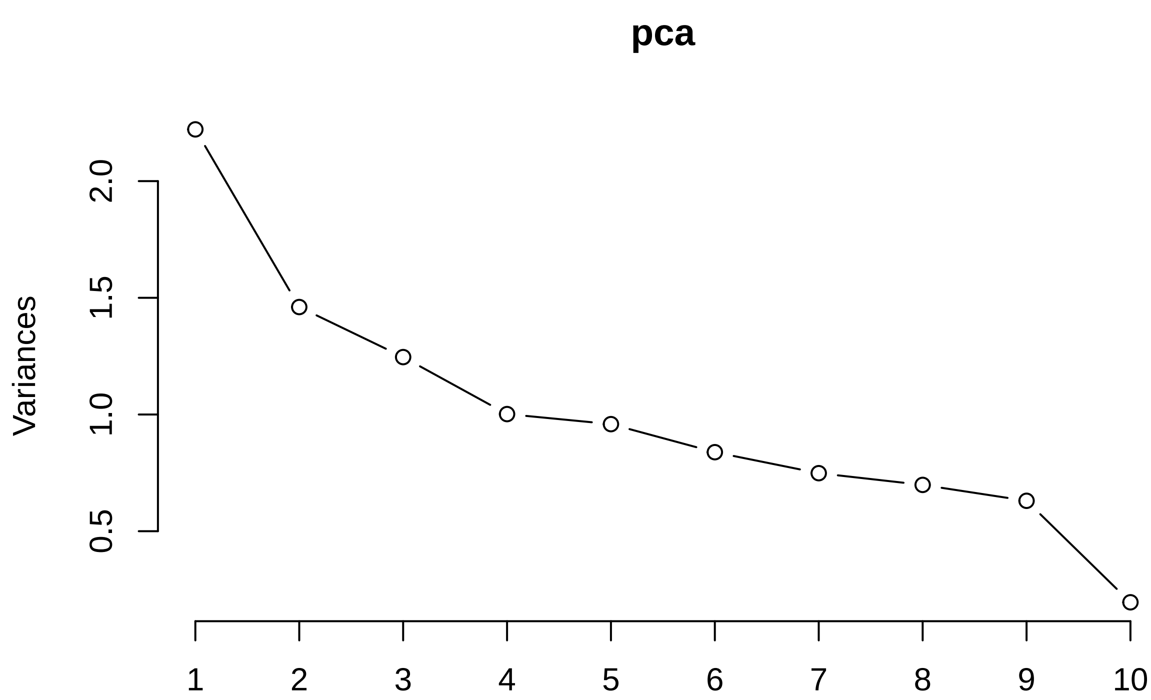
raypeng

2025-05-02

```r
#Step 1: Use the code from Week 7 as a Starting Point
library( ggplot2 )
library( flexclust )

SEED = 1
set.seed( SEED )

TARGET = "TARGET_BAD_FLAG"

PATH = "/Users/raypeng/Documents/IS 5213 Data science and big data/HMEQ_Scrubbed"
FILE_NAME = "HMEQ_Scrubbed.csv"

INFILE = paste(PATH, FILE_NAME, sep = "/")

setwd(PATH)
df = read.csv(FILE_NAME)

#Step 2: PCA Analysis
#Use only the input variables. Do not use either of the target variables.
df_pca = df
df_pca$TARGET_BAD_FLAG = NULL
df_pca$TARGET_LOSS_AMT = NULL

#Use only the continuous variables. Do not use any of the flag variables.
#Select at least 4 of the continuous variables.
#It would be preferable if there were a theme to the variables selected.
df_pca = df_pca[c(1,2,4,6,8,10,12,14,16,18)]

#Do a Principal Component Analysis (PCA) on the continuous variables.
pca = prcomp(df_pca,center=TRUE, scale=TRUE)
summary(pca)
```

```
## Importance of components:
##                           PC1    PC2    PC3    PC4     PC5     PC6     PC7
## Standard deviation     1.4905 1.2085 1.1163 1.0009 0.97918 0.91572 0.86520
## Proportion of Variance 0.2222 0.1461 0.1246 0.1002 0.09588 0.08385 0.07486
## Cumulative Proportion  0.2222 0.3682 0.4928 0.5930 0.68889 0.77274 0.84760
##                            PC8     PC9    PC10
## Standard deviation     0.83568 0.79387 0.44203
## Proportion of Variance 0.06984 0.06302 0.01954
## Cumulative Proportion  0.91744 0.98046 1.00000
```

```r
#Display the Scree Plot of the PCA analysis.
plot(pca, type = "l")
```




```r
df_new = data.frame( predict( pca, df_pca ) )

#Using the Scree Plot, determine how many Principal Components you wish to use.
#Note, you must use at least two. You may decide to use more. Justify your decision.
#I decide to use 4 PCs with PC4 has a standard deviation above 1.
#This means the first 4 PCs contain most of the information of the imputed dataset.

#Print the weights of the Principal Components.
print(pca$rotation)
```

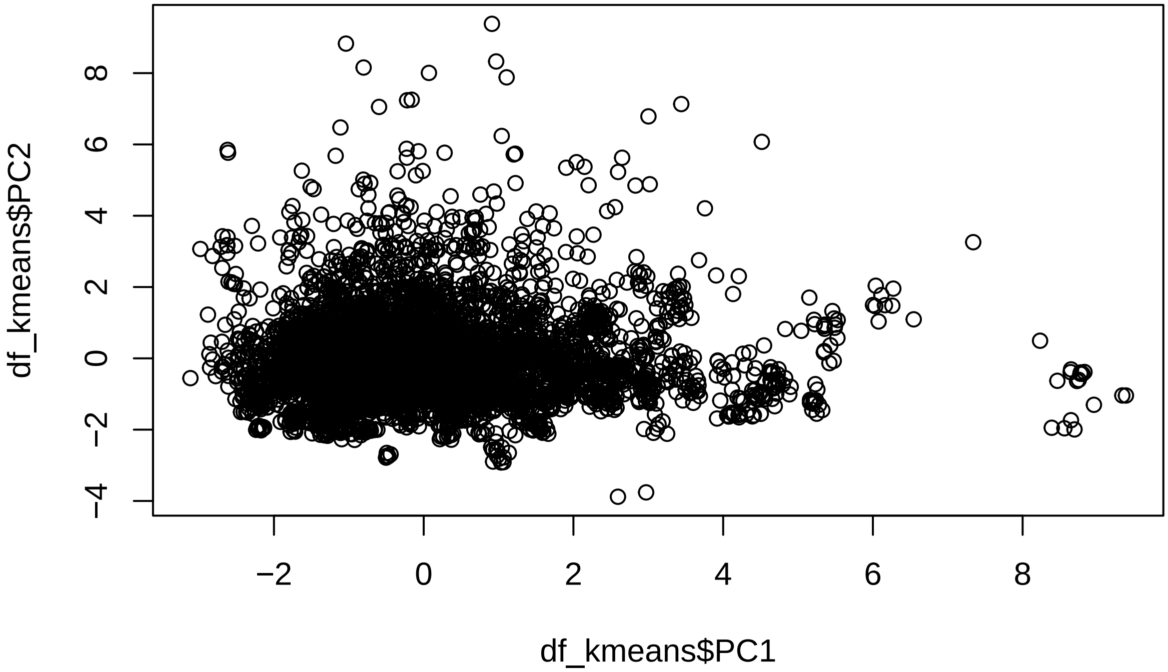
```
##                     PC1          PC2         PC3         PC4          PC5
## LOAN         0.31425517 -0.104598465  0.05295727 -0.53771580  0.419827766
## IMP_MORTDUE  0.57476524  0.001640244  0.19466925  0.22040956  0.098110092
## IMP_VALUE    0.58633796 -0.078601929  0.15458274  0.10039762  0.186199448
## IMP_YOJ      0.03435411 -0.260508848 -0.53332969 -0.51783332  0.018198368
## IMP_DEROG   -0.03192356  0.555370079 -0.18904164  0.05750818  0.383539825
## IMP_DELINQ   0.02493396  0.459862520 -0.43274648  0.17040014  0.147886344
## IMP_CLAGE    0.23297961 -0.242635491 -0.53339115  0.09355940 -0.172544558
## IMP_NINQ     0.04386120  0.461243097  0.14867761 -0.46490354  0.006637185
## IMP_CLNO     0.36929326  0.217794624 -0.29861623  0.14890732 -0.349807190
## IMP_DEBTINC  0.17802968  0.275775654  0.17937104 -0.32348860 -0.676730173
##                     PC6          PC7         PC8         PC9         PC10
```

```
## LOAN        -0.213495057 -0.44043005 -0.35336106  0.21745400 -0.101959313
## IMP_MORTDUE  0.006342209  0.23976652  0.13640577 -0.13289245 -0.692636273
## IMP_VALUE   -0.050919084  0.18362254  0.13777813 -0.15610907  0.708322628
## IMP_YOJ     -0.106699879  0.47595145  0.36416419  0.06189337 -0.060587595
## IMP_DEROG   -0.040965995 -0.34472757  0.61903640  0.02468630 -0.008212326
## IMP_DELINQ  -0.397005325  0.27884949 -0.51269531 -0.22668592  0.010743180
## IMP_CLAGE    0.314965078 -0.46591623 -0.05424561 -0.49387254 -0.026149737
## IMP_NINQ     0.655283652  0.20913489 -0.15817225 -0.22004875  0.021740265
## IMP_CLNO     0.225736964 -0.03028645 -0.09454951  0.72011437  0.055669736
## IMP_DEBTINC -0.448947365 -0.17308851  0.14821557 -0.20966727  0.005391156
```

```r
#Use the weights to tell a story on what the Principal Components represent.
#PC1 is more about MORTDUE, VALUE and CLNO. I call this "Financial Capacity".
#PC2 is more about DEROG, NINQ and DELINQ. I call this "Credit Risk".
#PC3 is more about YOJ, CLAGE and DELINQ. I call this "Financial Responsibility".
#PC4 is more about LOAN, YOJ and NINQ. I call this "Borrowing Intensity".

#Perform a scatter plot using the first two Principal Components.
#Do not color the dots. Leave them black.
df_kmeans = df_new[1:2]
print( head( df_kmeans ) )
```

```
##          PC1        PC2
## 1 -2.4361630 -0.2914953
## 2 -1.2657133  0.3930930
## 3 -2.6621119 -0.1696773
## 4 -0.7828377  0.8659403
## 5 -0.5746093 -0.2924981
## 6 -2.3178901 -0.2111695
```

```r
plot( df_kmeans$PC1, df_kmeans$PC2 )
```

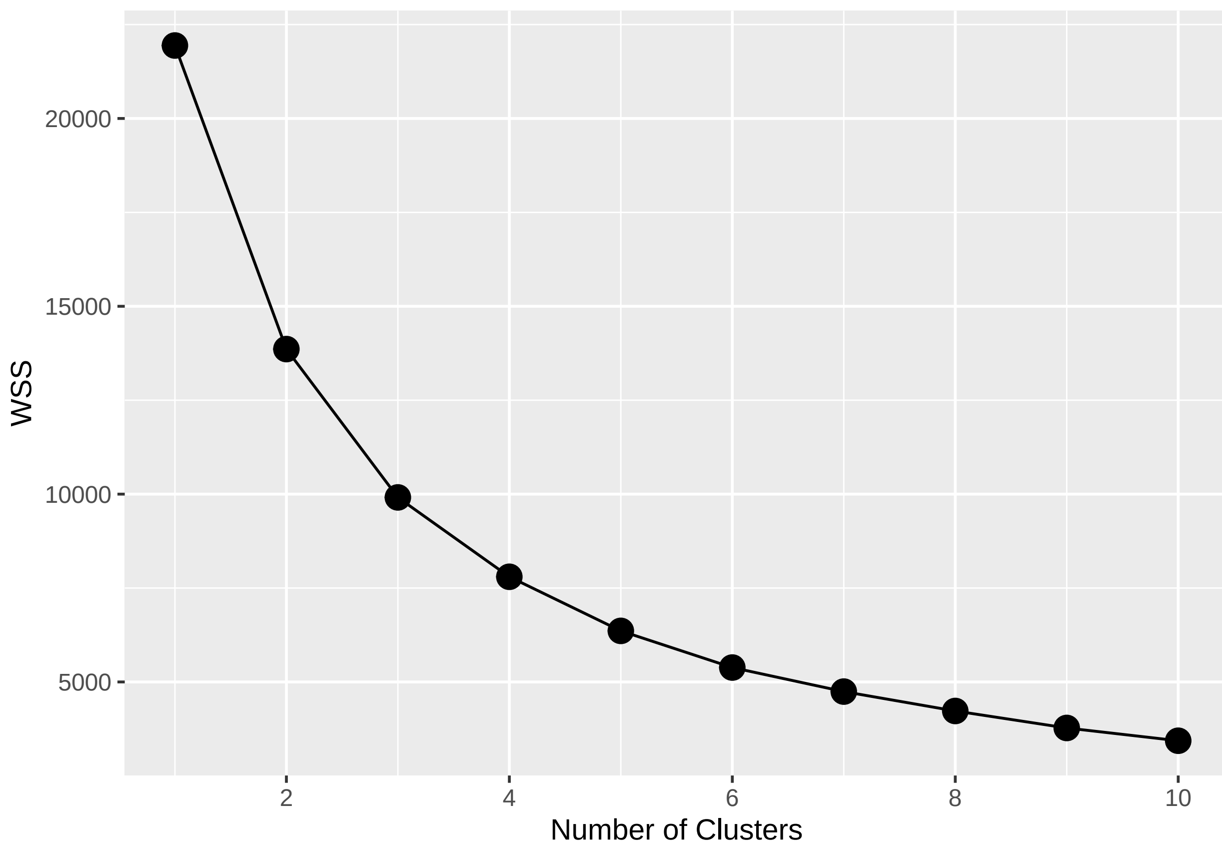



```
#Step 3: Cluster Analysis - Find the Number of Clusters
#Use the principal components from Step 2 for this step.
#Using the methods presented in the lectures, complete a KMeans cluster analysis for N=1 to at least N=
#Feel free to take the number higher.
#Print a scree plot of the clusters and determine how many clusters would be optimum. Justify your deci

# Maximum Clusters To Search
MAX_N = 10

# Set up an array to hold the Sum of Square Errors
WSS = numeric( MAX_N )

for ( N in 1:MAX_N )
{
  km = kmeans( df_kmeans, centers=N, nstart=20  )
  WSS[N] = km$tot.withinss
}
```

```
## Warning: did not converge in 10 iterations
```

```
df_wss = as.data.frame( WSS )
df_wss$clusters = 1:MAX_N

scree_plot = ggplot( df_wss, aes( x=clusters, y=WSS, group=1 )) +
  geom_point( size=4 ) +
```

```r
  geom_line() +
  scale_x_continuous( breaks=c(2,4,6,8,10)) +
  xlab("Number of Clusters")

scree_plot
```




```r
#Step 4: Cluster Analysis
#Using the number of clusters from step 3, perform a cluster analysis using the principle components fr
#Print the number of records in each cluster.
#Print the cluster center points for each cluster

# 4 would be optimum since from here the line starts to get flat.
BEST_N = 4
km = kmeans( df_kmeans, centers=BEST_N, nstart=20  )

print( km$size )
```

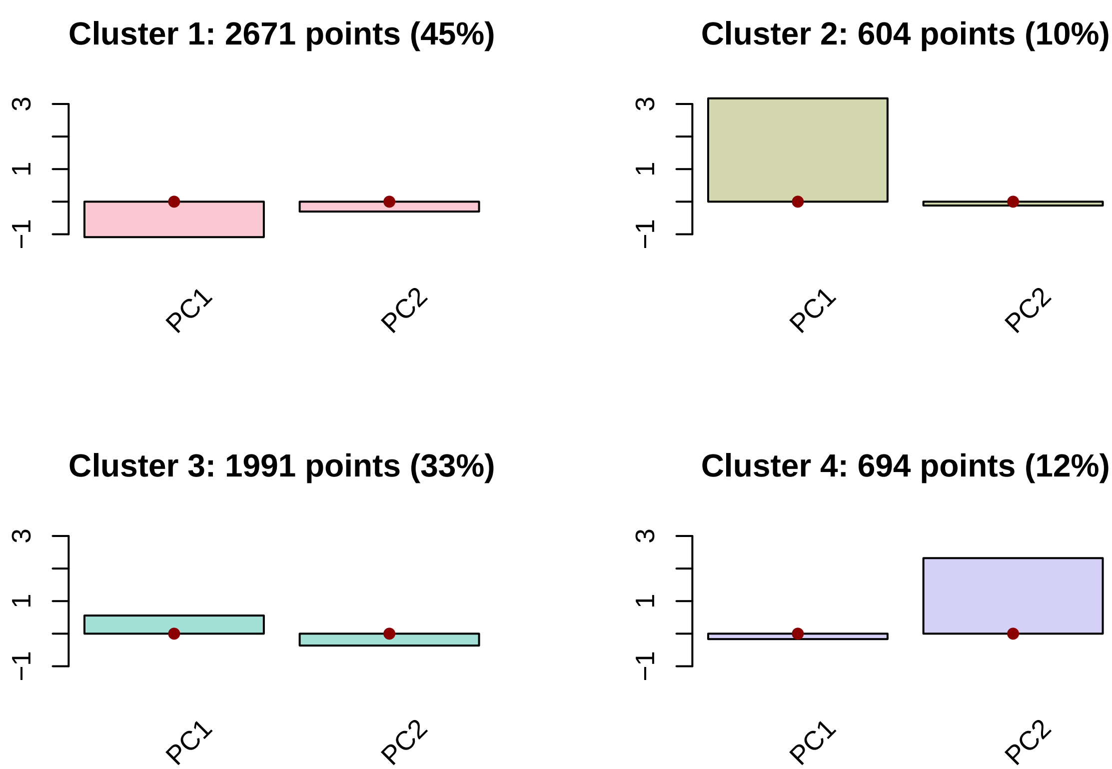
```
## [1] 2671  604 1991  694
```

```r
print( km$centers )
```

```
##          PC1        PC2
## 1 -1.0891828 -0.3035157
## 2  3.1726984 -0.1185246
```

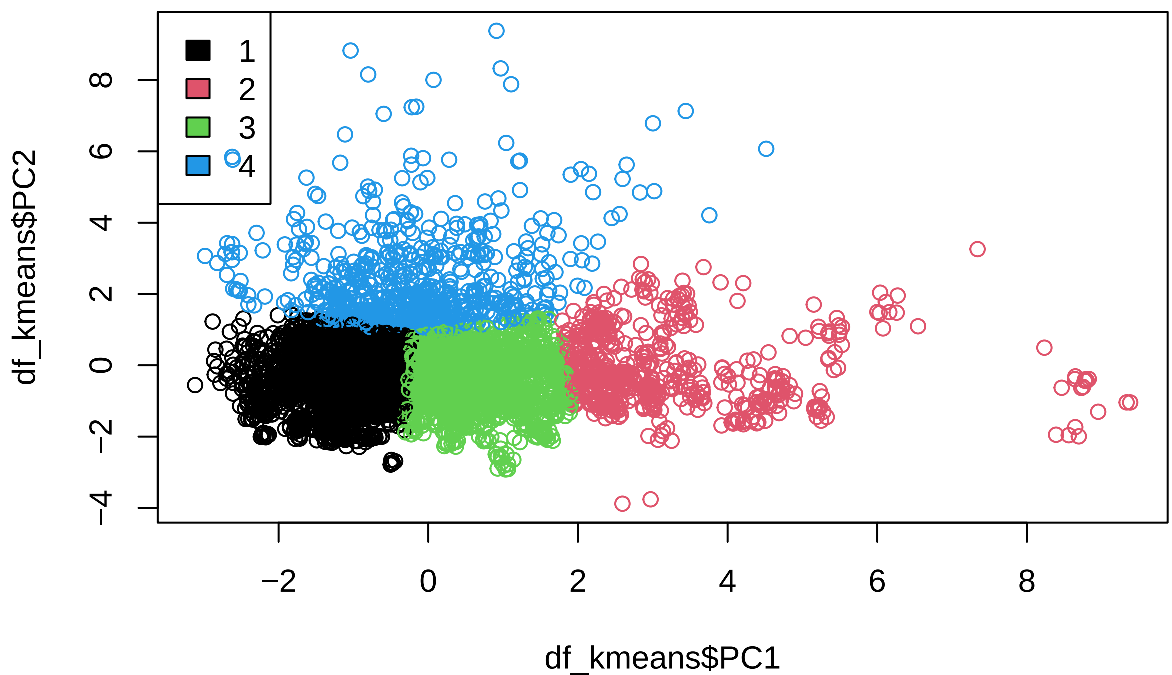
```
## 3  0.5570299 -0.3656450
## 4 -0.1673616  2.3202858
```

```r
#Convert the KMeans clusters into "flexclust" clusters
kf = as.kcca( object=km, data=df_kmeans, save.data=TRUE )
kfi = kcca2df( kf )
agg = aggregate( kfi$value, list( kfi$variable, kfi$group ), FUN=mean )

#Print the bar plot of the cluster. Describe the clusters from the bar plot.
barplot(kf)
```



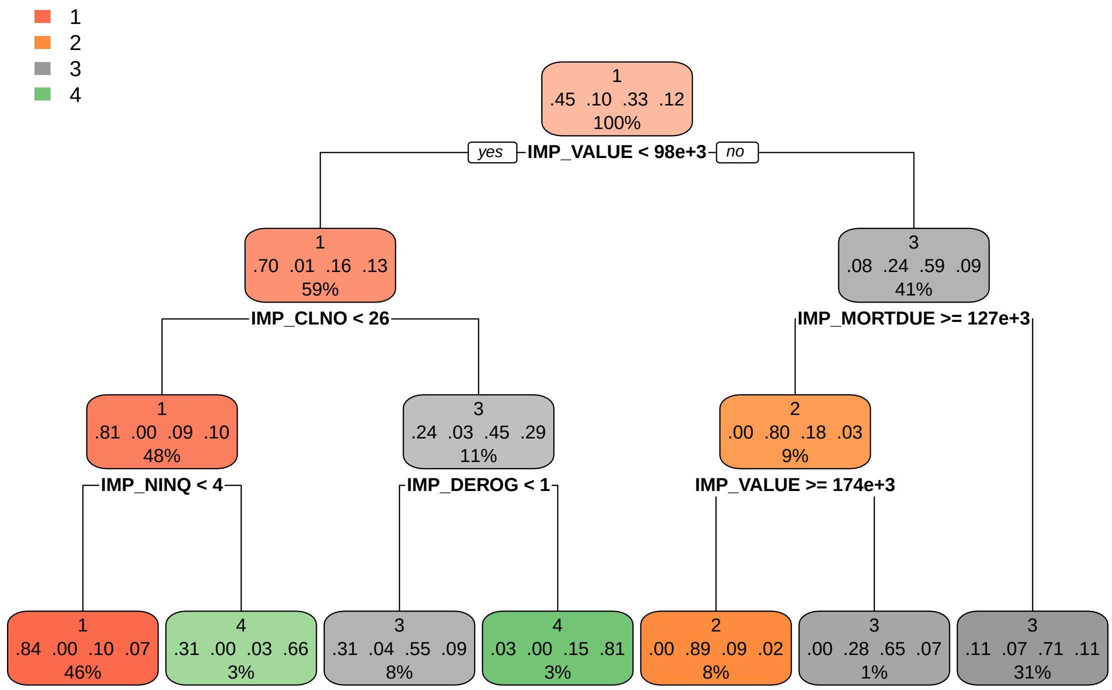

```r
#Cluster 1 has very low PC1(low house value) and slightly low PC2(low risk),
#they are majority responsible people and I think they probably default on loans but not so much.

#Cluster 2 has very high PC1(high financial capability) and near-zero PC2(average risk),
#they have much house loans and they may default.

#Cluster 3 has a high PC1(relatively high house value) and slightly low PC2(low risk),
#they are safe to have loans in my opinion and they tend to be responsible.

#Cluster 4 has a relatively low PC1(average house value) and very high PC2(high risk in defaulting),
#they are the most dangerous group!!!!!!

#Perform a scatter plot using the first two Principal Components. Color the plot by the cluster members
clus = predict( kf, df_kmeans )
```

```r
plot( df_kmeans$PC1, df_kmeans$PC2, col=clus )

#Add a legend to the plot.
legend( x="topleft", legend=c(1:BEST_N), fill=c(1:BEST_N) )
```




```r
#Score the training data using the flexclust clusters. In other words, determine which cluster they are
df$CLUSTER = clus
agg = aggregate( df$TARGET_BAD_FLAG, list( df$CLUSTER ), FUN=mean )

#Determine if the clusters predict loan default.
#I like this cluster analysis and I think it predicts loan default very well.
#As we can see from the chart:
#Black(C1) means "poor" but responsible;
#Green(C3) means "average income" and responsible;
#Red(C2) means "rich" but more possible to default because they have more loan amount.
#Blue(C4) means "average income" but highly possible to default on loans.


#Step 5: Describe the Clusters Using Decision Trees
#Using the original data from Step 2, predict cluster membership using a Decision Tree
#Display the Decision Tree
library( rpart )
library( rpart.plot )

df_tree = df_pca
```

```r
df_tree$CLUSTER = as.factor(clus)

dt = rpart( CLUSTER ~ . , data=df_tree )
dt = rpart( CLUSTER ~ . , data=df_tree, maxdepth=3 )

rpart.plot( dt )
```




```r
#Using the Decision Tree plot, describe or tell a story of each cluster. Comment on whether the cluster
#I feel those clusters make sense.
#Let's look at cluster 1 first and they are on the leftmost side,
#it means smaller house loan value, less credit card and less borrowing money inquiries.
#Cluster 1 is the biggest proportion and they are ordinary borrowers.

#Then let's look at the grey Cluster 3, high house loan value, less mortgage due,
#Which means they are responsible rich persons.

#The orange cluster 2 are close to cluster 3, but the difference is
#Cluster 2 people have more mortgage due and more house loan value.

#The there are interesting cluster 4:
#They don't have a high house value but they have many borrowing money inquiries,
#and they have defaulted on bills of the past years.

#Generally, I would rank their defaulting risk
#(combining default possibility and amount) in a descending order:
```

```
#Cluster 4(not responsible) > Cluster 2(large mortgage due) >
#Cluster 3(large loan) > Cluster 1(smaller loan)

#Step 6: Comment
#Discuss how you might use these clusters in a corporate setting.

#After the analysis of this loan default data set, I feel clusters are useful to distinguish groups.
#For example, in marketing or customer service sector the company can get data of all the customers
#and all the customers have different purchasing hobbies.
#Usually we can use a lot of metrics such as gender, age or regions to set data apart.
#However, if there isn't enough information that can be observed by those obvious metrics,
#we can try to use the cluster analysis.

#And just like this loan default analysis, in risk management sector,
#cluster analysis can be very important to identify which groups are high-risk
#and which ones are not.
```