Data Mining Principles - Assignment 2: Part 1

Ryan Durfey

Saturday, January 31, 2015

```
library(caret)
library(poLCA)
library(knitr)
```

```
set.seed(312435)

data(GermanCredit)
GC<-GermanCredit

## separate training and test indices to encompass 60% and 40% of the data, respectively.
## these will be used to subset the data into train and test
train_indices<-sample(1:nrow(GC),as.integer(nrow(GC)*.6),replace=FALSE)
length(train_indices)
```

```
## [1] 600
```

Data Preparation

Before using poLCA(), we need to prepare the data that will be passed to it. The function takes categorical variables with values 1,2,3,…K, so we need to combine the multiple fields in the original dataset that make up each variable. The function below does just that, and after defining it, we proceed to create vectors representing each categorical variable observed in the GermanCredit dataset.

```
#######################################################
## function to combine categorical variables from GC ##
#######################################################
combine.cat.vars<-function(cols,dataframe){
        combined<-dataframe[,cols]
        levels<-1:length(cols)
        combined<-data.frame(mapply("*",combined,levels))
        combined.1<-combined[,1]
        for(i in 2:max(levels)){
                combined.1<-combined.1+combined[,i]
        }
        combined.1
}

######################################################################
## Create single-vector, combined categorical variables with levels ##
######################################################################
checking<-combine.cat.vars(11:14,GC)
cred.hist<-combine.cat.vars(15:19,GC)
purpose<-combine.cat.vars(20:30,GC)
savings<-combine.cat.vars(31:35,GC)
emp.dur<-combine.cat.vars(36:40,GC)
per.stats<-combine.cat.vars(41:45,GC)
debtor.gaur<-combine.cat.vars(46:48,GC)
real.estate<-combine.cat.vars(49:52,GC)
install.plans<-combine.cat.vars(53:55,GC)
housing<-combine.cat.vars(56:58,GC)
job<-combine.cat.vars(59:62,GC)
```

Finding the Best LCA Models:

In order to find the best LCA model, we will need to run poLCA() under varying conditions and multiple times. The function below will create 100 models per class number in attempt to avoid a local optima problem. On each iteration, it compares each BIC and upon completion outputs the model that has the lowest BIC.

```r
#####################################
## function to find best LCA model ##
#####################################
LCA_best_models<-function(data,formula,max.class=14){
        ret<-NULL
        min_aic<-100000
        min_bic<-100000
        clust_bic<-c()
        clust_aic<-c()
        for(i in 2:max.class){
                for(j in 1:100){
                        res<-poLCA(formula,data,nclass=i,maxiter=1000,tol=.001,
                                   verbose=FALSE)
                        clust_bic<-rbind(clust_bic,c(i,res$bic))
                        clust_aic<-rbind(clust_aic,c(i,res$aic))
                        if(res$bic < min_bic){
                                min_bic<-res$bic
                                LCA_best_model_BIC<-res
                        }
                        if(res$aic < min_aic){
                                min_aic<-res$aic
                                LCA_best_model_AIC<-res
                        }
                }
        }
        ret$LCA_best_model_BIC<-LCA_best_model_BIC
        ret$min_bic<-min_bic
        ret$LCA_best_model_AIC<-LCA_best_model_AIC
        ret$min_aic<-min_aic
        ret$clust_bic<-as.data.frame(clust_bic)
        ret$clust_aic<-as.data.frame(clust_aic)
        return(ret)
}
```

Selected Variables Applied to LCA Model Function

The variables that are of interest to us in this analysis are those surrounding a loan recipient’s money and income. Thus, we’ll choose to look at the variables regarding Checking Account, Savings Account, Employment Duration, and Job Type.

```
###########################################################################
## try LCA_best_models with 4 variables denoting money and employment ##
###########################################################################
mj<-data.frame(checking,savings,emp.dur,job)
mj.train<-mj[train_indices,]
formula<-cbind(checking,savings,emp.dur,job)~1
mj.model<-LCA_best_models(data=mj.train,formula=formula,max.class=7)

## are the models with lowest AIC and BIC the same?
identical(mj.model$LCA_best_model_AIC,mj.model$LCA_best_model_BIC)
```

```
## [1] FALSE
```

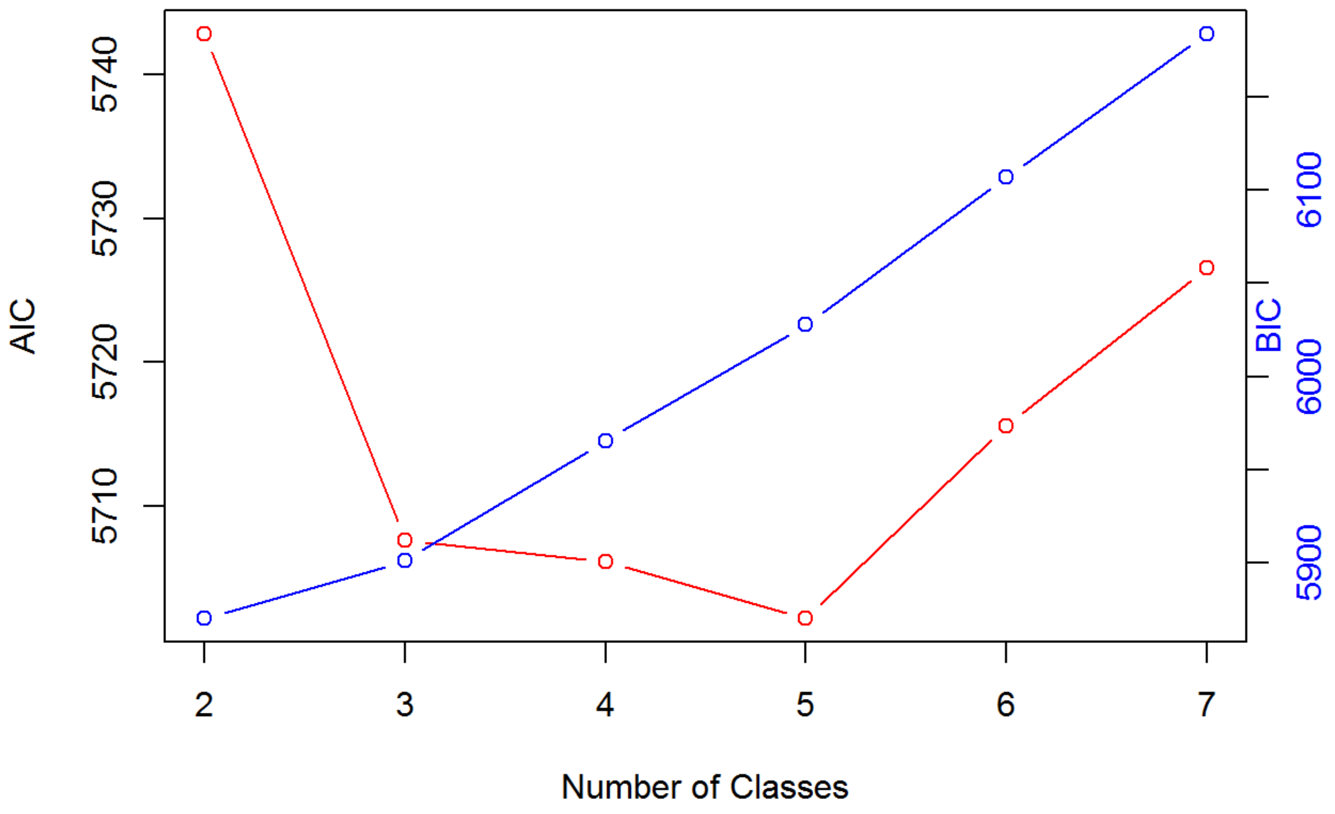
```
## no? ok, let's look at them
mj.model$LCA_best_model_AIC
```

```
## Conditional item response (column) probabilities,
##  by outcome variable, for each class (row)
##
## $checking
##            Pr(1)  Pr(2)  Pr(3)  Pr(4)
## class 1:  0.2667 0.4000 0.2000 0.1333
## class 2:  0.0997 0.0002 0.0612 0.8389
## class 3:  0.3175 0.4127 0.0627 0.2070
## class 4:  0.0779 0.5202 0.0000 0.4019
## class 5:  0.3850 0.2624 0.0607 0.2919
##
## $savings
##            Pr(1)  Pr(2)  Pr(3)  Pr(4)  Pr(5)
## class 1:  0.6000 0.0000 0.1333 0.0667 0.2000
## class 2:  0.2671 0.0132 0.1883 0.1445 0.3869
## class 3:  0.7628 0.1127 0.0000 0.0000 0.1245
## class 4:  0.0015 0.5723 0.0393 0.0401 0.3469
## class 5:  0.8962 0.0000 0.0274 0.0014 0.0750
##
## $emp.dur
##            Pr(1)  Pr(2)  Pr(3)  Pr(4)  Pr(5)
## class 1:  0.3333 0.0667 0.0000 0.0000 0.6000
## class 2:  0.0002 0.3828 0.1798 0.4372 0.0000
## class 3:  0.1232 0.1640 0.0988 0.3378 0.2763
## class 4:  0.2109 0.3801 0.2294 0.1533 0.0263
## class 5:  0.2465 0.3636 0.2031 0.1869 0.0000
##
## $job
##           Pr(1)  Pr(2)  Pr(3)  Pr(4)
## class 1:      1 0.0000 0.0000 0.0000
## class 2:      0 0.1800 0.7035 0.1165
## class 3:      0 0.0000 0.1709 0.8291
## class 4:      0 0.2076 0.7912 0.0011
## class 5:      0 0.2806 0.7194 0.0000
##
## Estimated class population shares
##  0.025 0.2011 0.1444 0.13 0.4994
##
## Predicted class memberships (by modal posterior prob.)
##  0.025 0.1783 0.1333 0.1267 0.5367
##
## =========================================================
## Fit for 5 latent classes:
## =========================================================
## number of observations: 600
## number of estimated parameters: 74
## residual degrees of freedom: 325
## maximum log-likelihood: -2777.105
##
## AIC(5): 5702.21
## BIC(5): 6027.583
## G^2(5): 202.7758 (Likelihood ratio/deviance statistic)
## X^2(5): 237.9649 (Chi-square goodness of fit)
##
```

```
mj.model$LCA_best_model_BIC
```

```
## Conditional item response (column) probabilities,
##  by outcome variable, for each class (row)
##
## $checking
##            Pr(1)  Pr(2)  Pr(3)  Pr(4)
## class 1:  0.2742 0.2480 0.0501 0.4277
## class 2:  0.2811 0.4313 0.1098 0.1778
##
## $savings
##            Pr(1)  Pr(2)  Pr(3)  Pr(4)  Pr(5)
## class 1:  0.6179 0.0941 0.0636 0.0389 0.1855
## class 2:  0.6967 0.0871 0.0313 0.0190 0.1659
##
## $emp.dur
##            Pr(1)  Pr(2)  Pr(3)  Pr(4)  Pr(5)
## class 1:  0.1764 0.3669 0.2043 0.2466 0.0058
## class 2:  0.1790 0.0639 0.0000 0.2774 0.4797
##
## $job
##            Pr(1)  Pr(2)  Pr(3)  Pr(4)
## class 1:  0.0000 0.2287 0.6944 0.0769
## class 2:  0.2257 0.0000 0.0980 0.6764
##
## Estimated class population shares
##  0.8892 0.1108
##
## Predicted class memberships (by modal posterior prob.)
##  0.8867 0.1133
##
## =========================================================
## Fit for 2 latent classes:
## =========================================================
## number of observations: 600
## number of estimated parameters: 29
## residual degrees of freedom: 370
## maximum log-likelihood: -2842.421
##
## AIC(2): 5742.841
## BIC(2): 5870.352
## G^2(2): 333.407 (Likelihood ratio/deviance statistic)
## X^2(2): 560.1415 (Chi-square goodness of fit)
##
```

```
## plot of AIC & BIC and numbers of classes
par(mar=c(5,4,4,4)+0.3)
plot(aggregate(V2~V1,data=mj.model$clust_aic,function(x)min(x)),type="b",col="red",
     ylab="AIC",xlab="Number of Classes")
par(new=TRUE)
plot(aggregate(V2~V1,data=mj.model$clust_bic,function(x)min(x)),type="b",axes=FALSE,
     xlab="",ylab="",col="blue")
axis(4,at=pretty(range(aggregate(V2~V1,data=mj.model$clust_bic,function(x)min(x))[,2])),
     col.axis="blue")
mtext("BIC",side=4,col="blue")
```



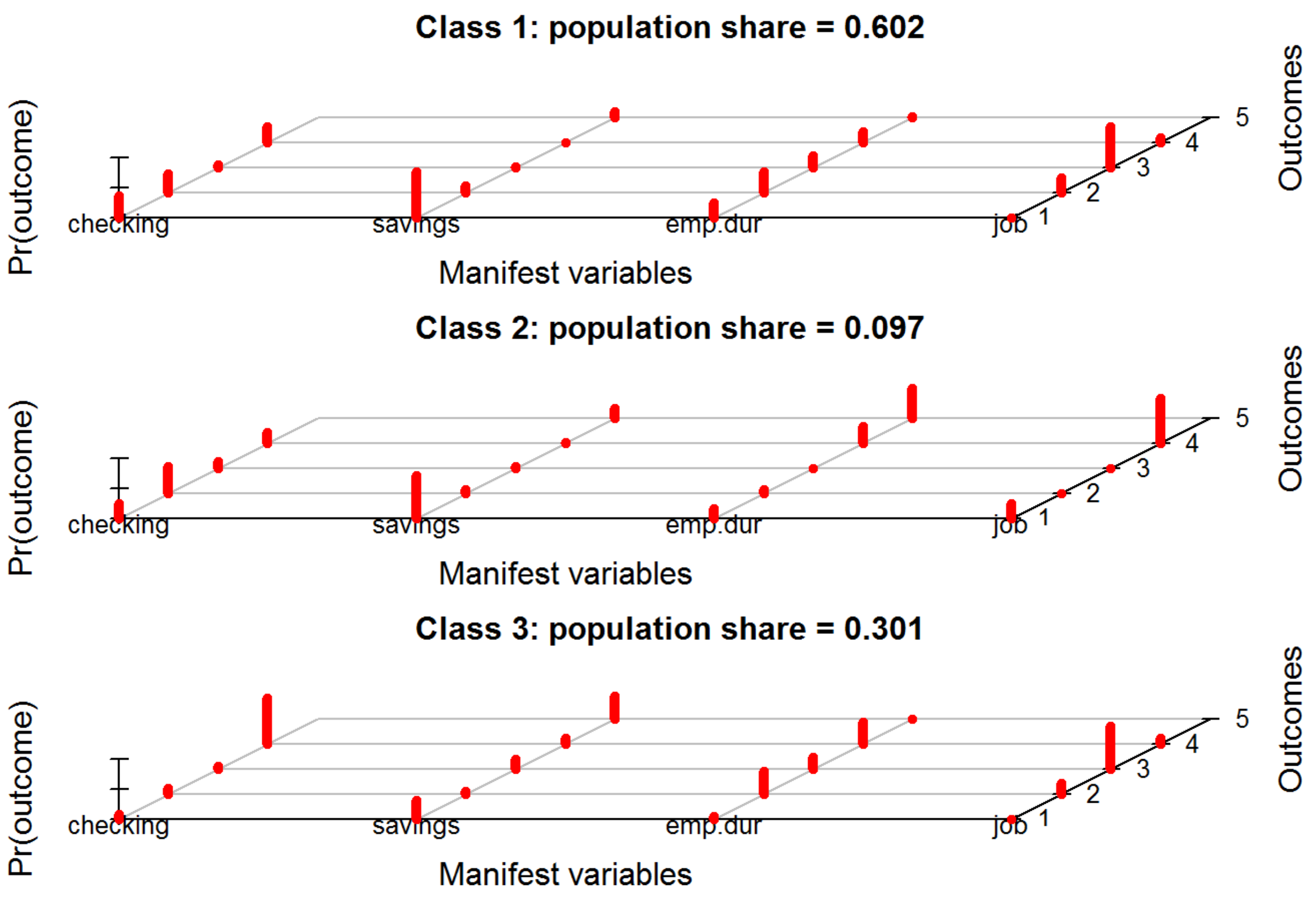
Interpretation of Model and Class Selection

From the above model summaries and graph, we see that selecting by AIC would give us 5 classes whereas selecting by BIC would give us 2 classes. We'll choose the 3 class model, which is in-between the BIC-only and AIC-only selection. Choosing 3 classes also gives us the lowest possible number of AIC+BIC, so it feels like the best compromise.

In order to see the graphs produced by poLCA(), we need to rerun the function with the parameter graphs=TRUE (unfortunately, we cannot call these graphs after we get the model, so we cannot just pull them from the models just created). In order to make sure we don't get stuck in a local optimum, we use nrep=500, which will cause the function to repeat itself and output the model with the maximum log-likelihood, which would correspond to the model we observed above (or at least be very close to it).

Note: Earlier, in the LCA_best_models function, we kept nrep=1 in order to be able to record the AIC and BIC values for each run of poLCA().

```
## recreate 3-class model in order to see graphs
mj.AIC.model<-poLCA(formula,mj.train,nclass=3,maxiter=1000,tol=.001,verbose=FALSE,graphs=TRUE,nrep=50
0)
```



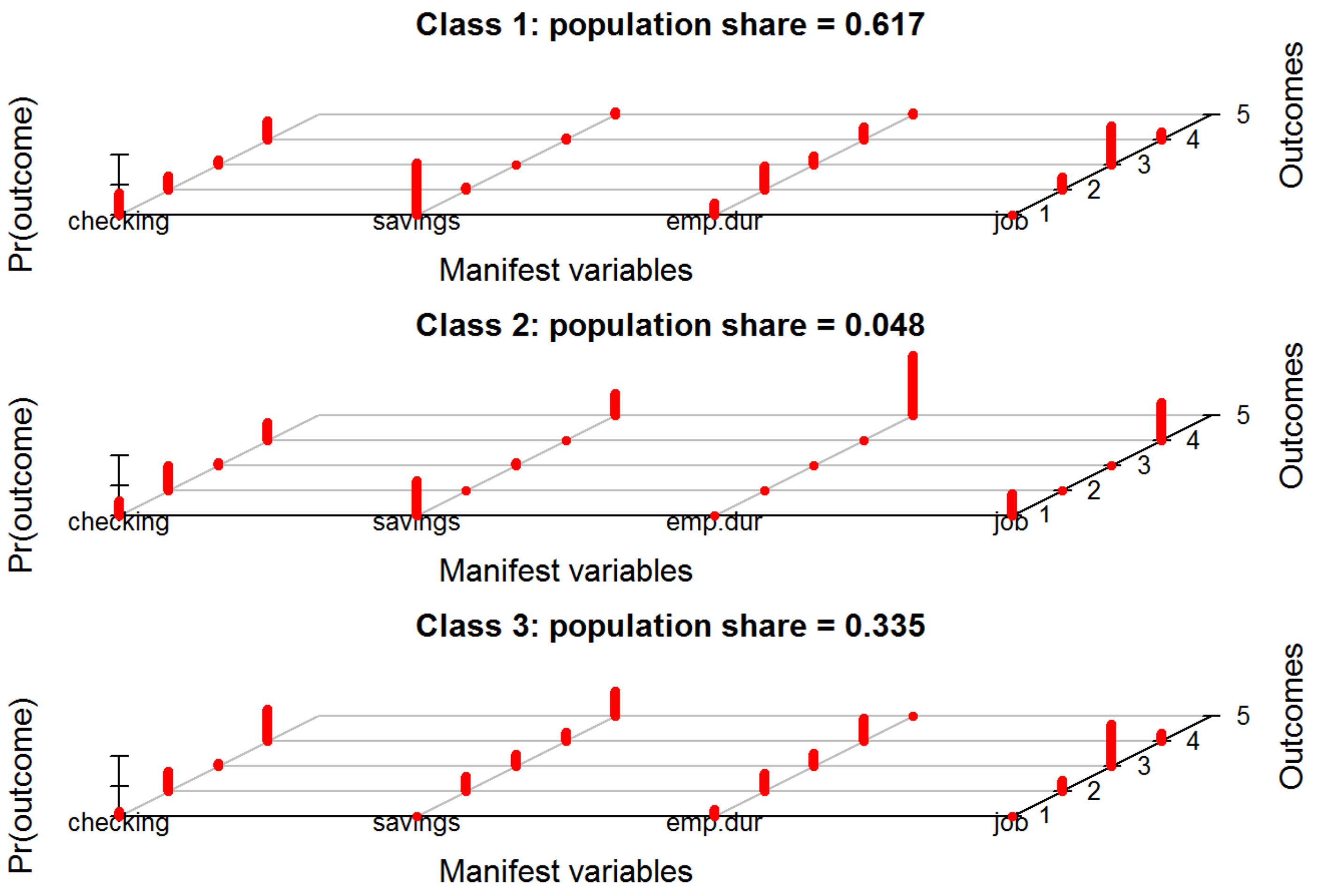
Unfortunately, the outputted graphs aren’t extremely helpful. Each class is made up of a mixture of many of the variables and levels. However, we can see that Class 1 features observations with checking=1, savings=1, and job=3 more prominently than the other variables & levels. Class 2 shows a prominence of observations where savings=1, emp.dur=5, and job=4. Lastly, Class 3 favors checking=4, savings=5, and job=3. We interpret this to mean that observations that exhibit these values would likely be classified into each of these latent classes.

Holdout Validation

And now to test our model via Holdout Validation! The test_indices vector that we created at the beginning will now allow us to select out a test dataset.

```
## create test dataset. everything EXCEPT for the train_indices
mj.test<-mj[-train_indices,]

## create 3-class model on the test data, with the probs.start inputted from the
## previously generated model (again, with nclass=3 and nrep=500)
probs<-mj.AIC.model$probs
mj.test<-poLCA(formula,mj.test,nclass=3,maxiter=1000,tol=.001,verbose=FALSE,
               graphs=TRUE,nrep=500,probs.start=probs)
```



```
## comparing test model with train model - class sizes
a<-table(mj.AIC.model$predclass)/length(mj.AIC.model$predclass)
b<-table(mj.test$predclass)/length(mj.test$predclass)
kable(rbind(train_set=a,test_set=b),col.names=c("Class 1","Class 2","Class 3"),
      caption="Class Size Proportions")
```

Class Size Proportions

	Class 1	Class 2	Class 3
train_set	0.6516667	0.1083333	0.24
test_set	0.5700000	0.0500000	0.38

Interpretation of Holdout Validation

To judge the holdout validation, we compare the class sizes of the train and test models. The class size proportions we see in the train and test models, while not nearly identical, are still fairly close. Thus, we'll consider the holdout a success.

Comparison to K-Means/K-Overlapping-Means Clustering Assignment

The previous clustering solutions we found in Assignment 1 used entirely different variables than those used here, and so it is not very useful to compare them directly. However, there are similarities between the clustering and class discovery methods in both cases. Within LCA, the classes that are found are analogous to the clusters found in K-means/Ko-means clustering. Running multiple iterations of each method then allows us to decide the best number of clusters or classes to use in our model.

However, I will note that LCA seems to be a much more difficult analysis to implement and interpret. Knowledge of the data and where each observation and variable come from and correspond to is key in predicting which variables to consider inputting to the LCA.

Data Mining Principles - Assignment 2: Part 2

Ryan Durfey

Friday, February 06, 2015

Introduction

The dataset on which the following Principle Components Analysis will be performed is from the GermanCredit dataset located in the caret package. The first seven variables will be considered since they are numerical.

```
library(caret)
library(dplyr)
library(knitr)

set.seed(12345)
```

```
data(GermanCredit)
GC<-GermanCredit
GC2<-GC[,1:7]

## separate training and test datasets to encompass 60% and 40% of the data, respectively
GC_train<-sample_frac(GC2,.60)
GC_test<-setdiff(GC2,GC_train)
dim(GC_train)
```

```
## [1] 600   7
```

```
dim(GC_test)
```

```
## [1] 400   7
```

```
## making sure none of the rows intersect, confirming the dataset separation was successful
nrow(intersect(GC_train,GC_test))
```

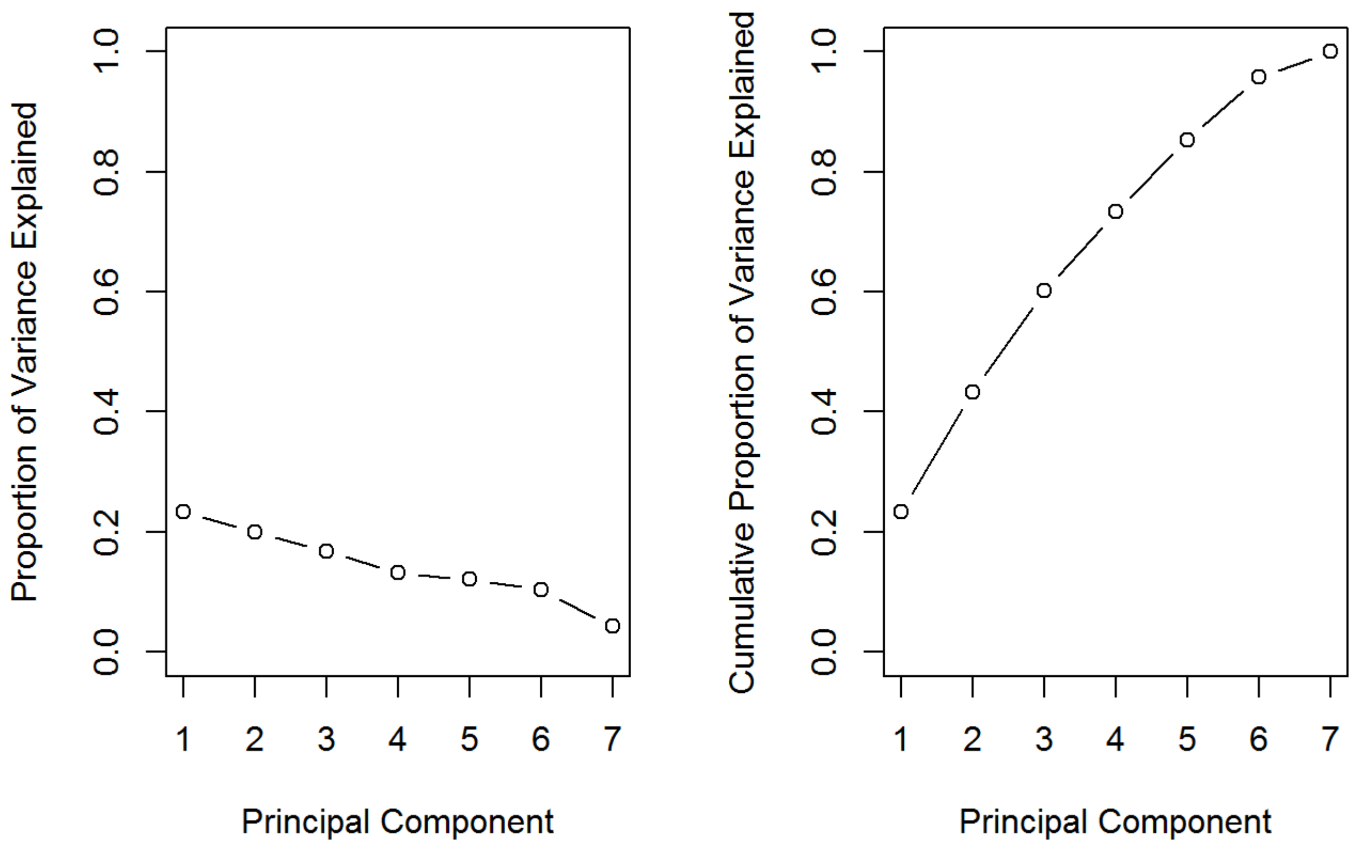
```
## [1] 0
## scale the data
GC_train_unscaled<-GC_train
GC_test_unscaled<-GC_test
GC_train<-scale(GC_train)
GC_test<-scale(GC_test)
```

Initial Principal Components Analysis using prcomp()

```
## initial PCA
PCA_train<-prcomp(GC_train,scale=TRUE,retx=TRUE)
## note: scaled=TRUE has failed in the past, so it is done manually above.
summary(PCA_train)
```

```
## Importance of components:
##                           PC1    PC2    PC3    PC4    PC5    PC6     PC7
## Standard deviation     1.2796 1.1827 1.0841 0.9592 0.9185 0.8533 0.54491
## Proportion of Variance 0.2339 0.1998 0.1679 0.1314 0.1205 0.1040 0.04242
## Cumulative Proportion  0.2339 0.4337 0.6016 0.7331 0.8536 0.9576 1.00000
```

```
## Scree plots for selecting components via Variance Explained (Variance Accounted For)
pve<-PCA_train$sdev^2/sum(PCA_train$sdev^2)
opar<-par(no.readonly=TRUE)
par(mfrow=c(1,2))
plot(pve,xlab="Principal Component",ylab="Proportion of Variance Explained",ylim=c(0,1),
     type="b")
plot(cumsum(pve),xlab="Principal Component",
     ylab ="Cumulative Proportion of Variance Explained",ylim=c(0,1),type="b")
```

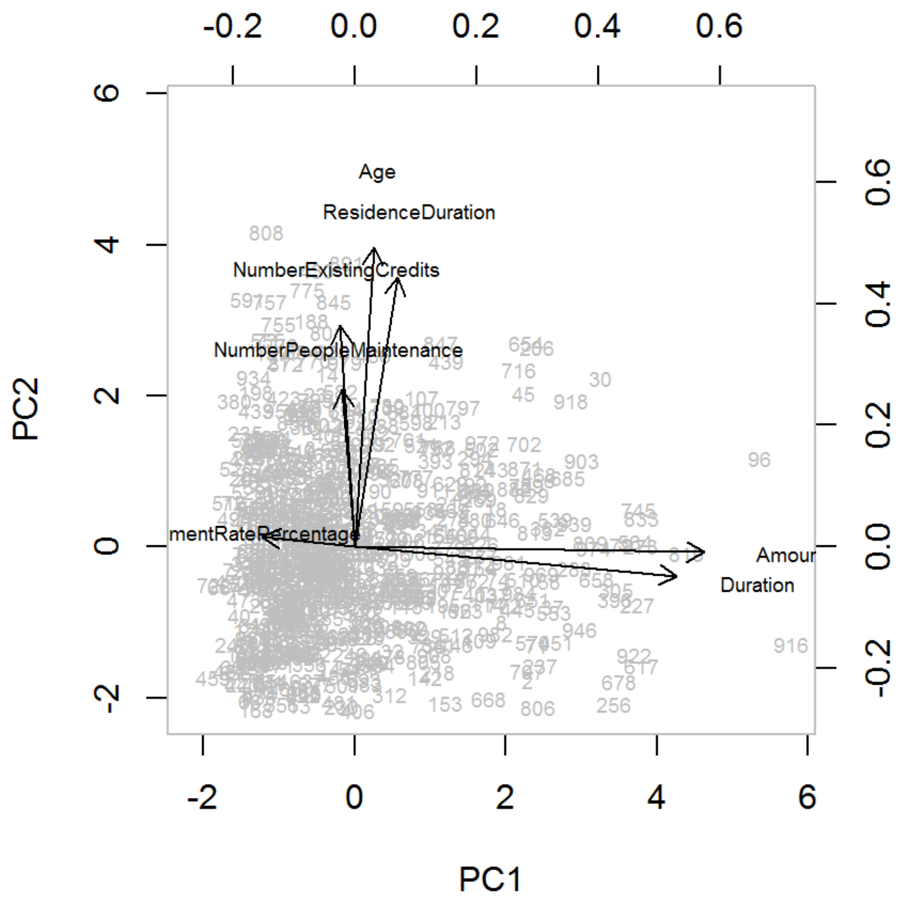


```
par(opar)
```

From these plots, there appears to be no obvious elbow with which to help us decide which components to retain. Thus, we will use the 85% rule and include the principal components that cumulatively comprise at least 85% of the Variance Explained, which in this case are the first 5 components.

Plots and Interpretations of Principal Components

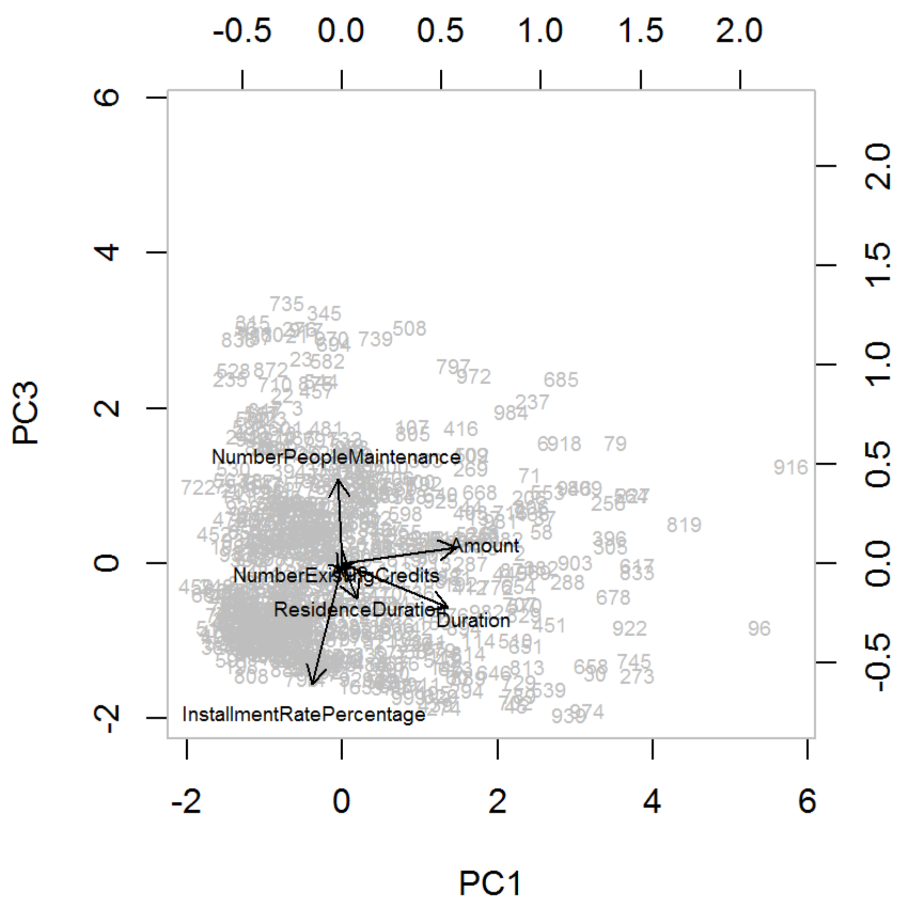
```
## plot of Component 1 vs. Component 2
biplot(PCA_train,scale=0,cex=0.6,col=c(8,1))
```



PC1 appears to be the variables Duration and Amount. These both have to do with properties of the loan itself, as opposed to properties of the person receiving the loan, so we could interpret PC1 as "Loan Properties."

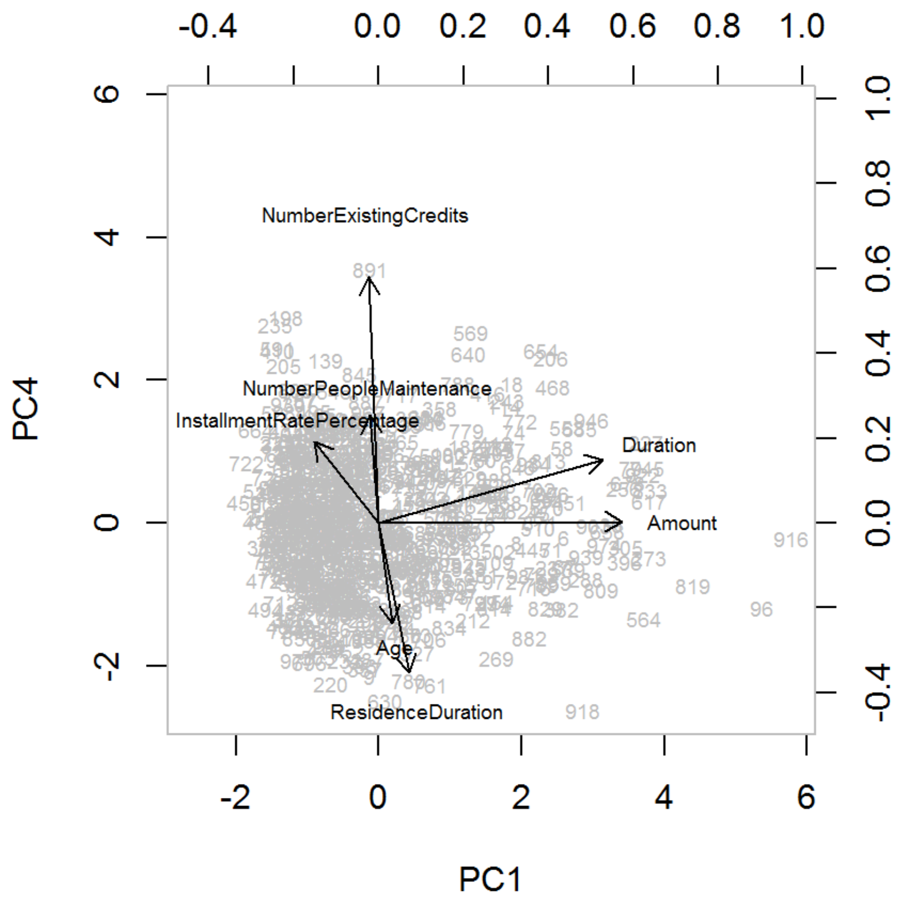
PC2 is more difficult to interpret, since the remaining 5 variables appear to have an effect. However, since they are all variables describing the loan recipient, we could interpret PC2 at "Loan Recipient Properties."

```
## PC1 vs. PC3
biplot(PCA_train,choices=c(1,3),scale=0,cex=0.6,col=c(8,1))
```



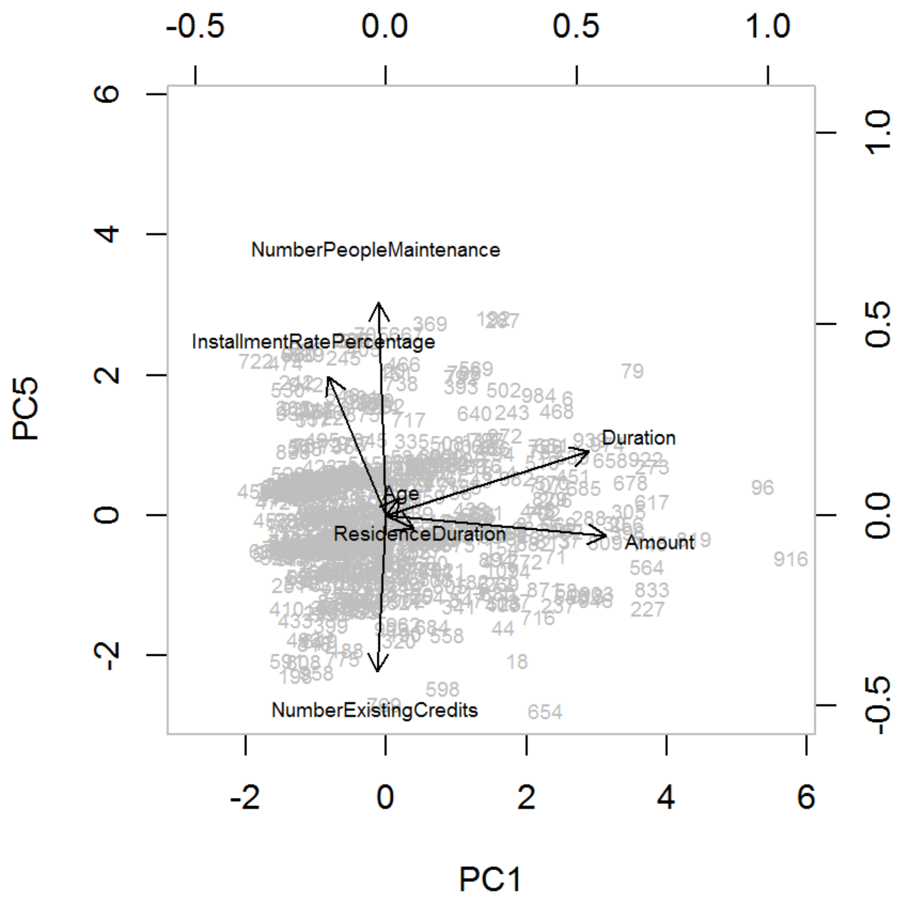
PC3 draws the most impact from InstallmentRatePercentage. So, we could interpret PC3 as “Loan Repayment Impact.”

```
## PC1 vs. PC4
biplot(PCA_train,choices=c(1,4),scale=0,cex=0.6,col=c(8,1))
```



PC4 is most affected by NumberExistingCredits, so we’ll interpret this component as mainly “Credits at this Bank.”

```
## PC1 vs. PC5
biplot(PCA_train,choices=c(1,5),scale=0,cex=0.6,col=c(8,1))
```



PC5 appears somewhat similar to PC3, but draws more impact from NumberPeopleMaintenance, which we might translate into the interpretation of “Number of People Liable for this Loan.”

Orthogonality of Loadings and Scores

```
## Loadings are Orthogonal
round(t(PCA_train$rotation)%*%PCA_train$rotation,2)
```

```
##     PC1 PC2 PC3 PC4 PC5 PC6 PC7
## PC1   1   0   0   0   0   0   0
## PC2   0   1   0   0   0   0   0
## PC3   0   0   1   0   0   0   0
## PC4   0   0   0   1   0   0   0
## PC5   0   0   0   0   1   0   0
## PC6   0   0   0   0   0   1   0
## PC7   0   0   0   0   0   0   1
```

```
#round(PCA_train$rotation%*%t(PCA_train$rotation),2)

## Scores are Orthogonal
#round(t(PCA_train$x) %*% PCA_train$x,2)
round(cor(PCA_train$x),2)
```

```
##     PC1 PC2 PC3 PC4 PC5 PC6 PC7
## PC1   1   0   0   0   0   0   0
## PC2   0   1   0   0   0   0   0
## PC3   0   0   1   0   0   0   0
## PC4   0   0   0   1   0   0   0
## PC5   0   0   0   0   1   0   0
## PC6   0   0   0   0   0   1   0
## PC7   0   0   0   0   0   0   1
```

Holdout Validation

Using predict() and the PCA output from the training set, we calculate the predicted component scores of the test dataset. Then, by comparing its Correlation and Variance Accounted For (VAF) with those from the training set model, we determine if the holdout is a success or not.

```
PCA_test_predict<-predict(PCA_train,newdata=GC_test)

## comparison of correlation
cor_train<-round(cor(as.vector(scale(GC_train)),
                     as.vector(PCA_train$x[,1:5]%*%t(PCA_train$rotation)[1:5,])),2)
cor_test<-round(cor(as.vector(scale(GC_test)),
                    as.vector(PCA_test_predict[,1:5]%*%t(PCA_train$rotation)[1:5,])),2)
kable(rbind(cor_train,cor_test),col.names=c("Cor"))
```

	Cor
cor_train	0.92
cor_test	0.93

```
##PCA_train Variance Accounted For (i.e., Variance Explained)
pve
```

```
## [1] 0.23392483 0.19981531 0.16789601 0.13142720 0.12051006 0.10400887
## [7] 0.04241771
```

```
##PCA_test Variance Accounted For (i.e., Variance Explained)
a<-c(var(PCA_test_predict[,1]),var(PCA_test_predict[,2]),var(PCA_test_predict[,3]),
     var(PCA_test_predict[,4]),var(PCA_test_predict[,5]),var(PCA_test_predict[,6]),
     var(PCA_test_predict[,7]))
var_total<-sum(a)
pve_test<-a/var_total
pve_test
```

```
## [1] 0.2399685 0.2033941 0.1454641 0.1396475 0.1315557 0.1018779 0.0380922
```

```
## comparison of PVE/VAF
b<-(pve_test-pve)/pve
kable(rbind(VAF_Train=pve,VAF_Test=pve_test,Percent_Diff=b),
      col.names=c("VAF.PC1","VAF.PC2","VAF.PC3","VAF.PC4","VAF.PC5","VAF.PC6","VAF.PC7"))
```

	VAF.PC1	VAF.PC2	VAF.PC3	VAF.PC4	VAF.PC5	VAF.PC6	VAF.PC7
VAF_Train	0.2339248	0.1998153	0.1678960	0.1314272	0.1205101	0.1040089	0.0424177
VAF_Test	0.2399685	0.2033941	0.1454641	0.1396475	0.1315557	0.1018779	0.0380922
Percent_Diff	0.0258361	0.0179103	-0.1336057	0.0625461	0.0916575	-0.0204885	-0.1019741

The correlation and VAF values are not very different between the test and training datasets. So we will consider the Holdout Validation complete and the Principal Components solution a success.

Varimax Rotation

```
## rotate the loadings from the PCA training set using varimax
rotated<-varimax(PCA_train$rotation[,1:5])
rotated
```

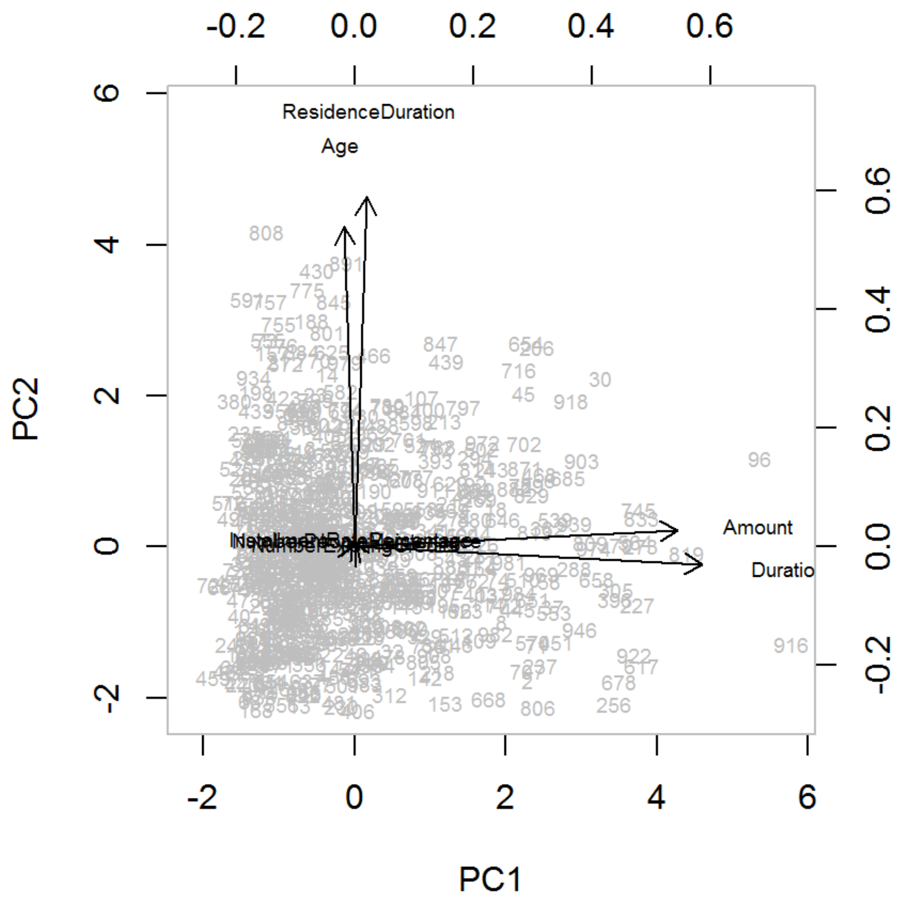
```
## $loadings
## 
## Loadings:
##                           PC1    PC2    PC3    PC4    PC5   
## Duration                   0.732        -0.242              
## Amount                     0.681         0.255              
## InstallmentRatePercentage               -0.935              
## ResidenceDuration                 0.736               -0.121
## Age                               0.674                0.121
## NumberExistingCredits                          0.999        
## NumberPeopleMaintenance                                0.985
## 
##                  PC1   PC2   PC3   PC4   PC5
## SS loadings    1.000 1.000 1.000 1.000 1.000
## Proportion Var 0.143 0.143 0.143 0.143 0.143
## Cumulative Var 0.143 0.286 0.429 0.571 0.714
## 
## $rotmat
##             [,1]         [,2]        [,3]        [,4]        [,5]
## [1,]  0.97496930  0.090818436  0.19825389 -0.03122273 -0.03012315
## [2,] -0.05028291  0.828524235 -0.01161911  0.45471070  0.32268003
## [3,] -0.13287292 -0.177337518  0.80830673 -0.05595723  0.54259120
## [4,]  0.13578301 -0.523275525 -0.26333610  0.72785488  0.32958622
## [5,]  0.10399157  0.005808088 -0.48770352 -0.50927339  0.70138285
```

```
## R-squared values / VAF
Five.PC.VAF<-cor(as.vector(scale(GC_train)),
                 as.vector(PCA_train$x[,1:5]%*%t(PCA_train$rotation[,1:5])))^2
Five.PC.rotated.VAF<-cor(as.vector(scale(GC_train)),
                         as.vector(PCA_train$x[,1:5]%*%rotated$rotmat%*%t(rotated$rotmat)%*%t(PCA_tra
in$rotation[,1:5])))^2
kable(rbind(Five.PC.VAF,Five.PC.rotated.VAF),col.names=c("VAF"))
```

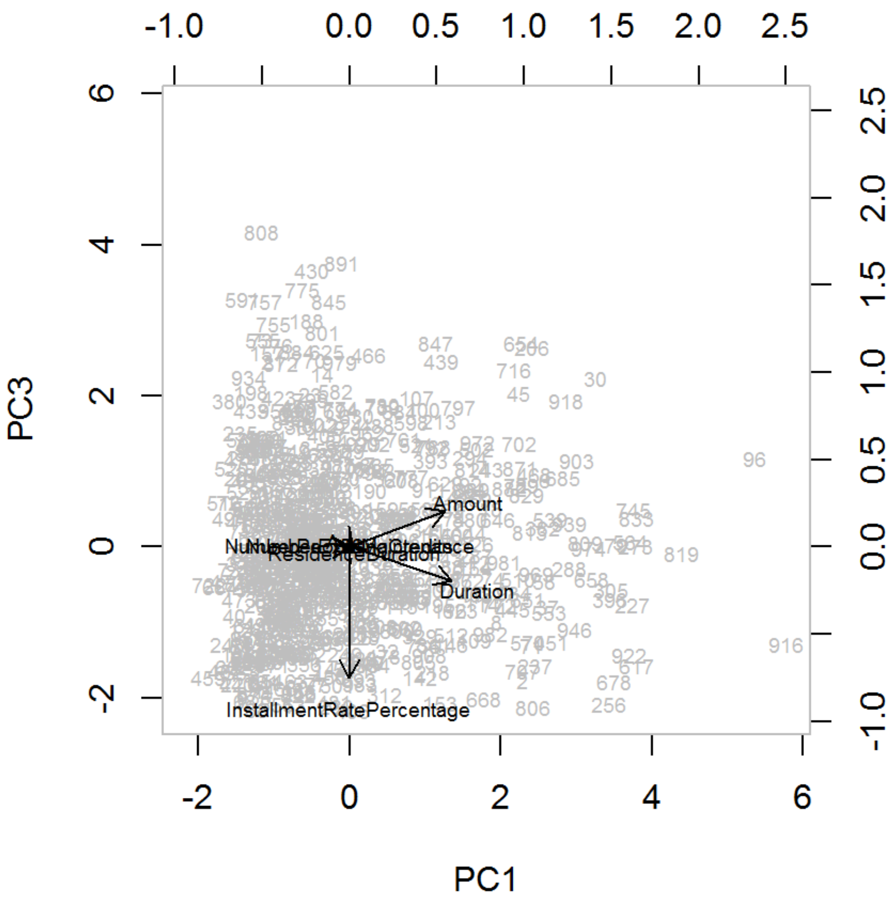
	VAF
Five.PC.VAF	0.8535734
Five.PC.rotated.VAF	0.8535734

From the table, we see that the VAF is identical in the non-rotated and rotated loadings. Thus we confirm that rotating the loadings does not alter the VAF.

```
## plot of rotated Loadings
biplot(PCA_train$x,rotated$loadings[,c(1,2)],cex=0.6,col=c(8,1))
```



```
biplot(PCA_train$x,rotated$loadings[,c(1,3)],cex=0.6,col=c(8,1),ylab="PC3")
```



After rotating the loadings, we can refine our interpretation of PC2. Here, it is now more clear that Age and ResidenceDuration have the biggest impacts by far. Thus, we could tweak our interpretation of PC2 to “Loan Recipient Age and Duration of Residence.” This also seems to make sense intuitively since the remaining variables show up more prominently in PC3, PC4, and PC5.

Similarly, with plotting rotated PC1 and PC3, we are reaffirmed in our previous interpretation that PC3 is most impacted by InstallmentRatePercentage. Thus, we’ll continue using our previous interpretation of “Loan Repayment Impact.”

Conclusion

After applying the Principle Component Analysis, it does not appear that it helped reduce the data very much. None of the principal components accounted for a large portion of the variance accounted for by themselves, which resulted in needing to retain a large number of components (reducing 7 down to 5 is not a very large jump). Therefore, in the end, using PCA on this dataset proved to be somewhat inefficient and rather unsatisfying.