ISYE6501 HW4

June 13, 2018

Question 9.1

Using the same crime data set uscrime.txt as in Question 8.2, apply Principal Component Analysis and then create a regression model using the first few principal components. Specify your new model in terms of the original variables (not the principal components), and compare its quality to that of your solution to Question 8.2. You can use the R function prcomp for PCA. (Note that to first scale the data, you can include scale=TRUE to scale as part of the PCA function. Don't forget that, to make a prediction for the new city, you'll need to unscale the coefficients (i.e., do the scaling calculation in reverse)!)

```
#Set seed so results are reproducible
set.seed(1)

#Read in data

crime<-read.csv("5.1uscrimeSummer2018.txt", stringsAsFactors = FALSE, header=TRUE, sep='\t')


#crime<-read.csv("http://www.statsci.org/data/general/uscrime.txt",
#                 stringsAsFactors = FALSE, header=TRUE, sep='\t')

#Get eigenvalues / eigenvectors
matrixCrime<-as.matrix(crime)

XTX<-t(matrixCrime)%*%matrixCrime
eig<-eigen(XTX)


#Run PCA on matrix of scaled data
pca <- prcomp(crime[,1:15], scale=TRUE)
summary(pca)
```

```
## Importance of components:
##                           PC1    PC2    PC3     PC4     PC5     PC6
## Standard deviation     2.4534 1.6739 1.4160 1.07806 0.97893 0.74377
## Proportion of Variance 0.4013 0.1868 0.1337 0.07748 0.06389 0.03688
## Cumulative Proportion  0.4013 0.5880 0.7217 0.79920 0.86308 0.89996
##                            PC7     PC8     PC9    PC10    PC11    PC12
## Standard deviation     0.56729 0.55444 0.48493 0.44708 0.41915 0.35804
## Proportion of Variance 0.02145 0.02049 0.01568 0.01333 0.01171 0.00855
## Cumulative Proportion  0.92142 0.94191 0.95759 0.97091 0.98263 0.99117
##                           PC13   PC14    PC15
## Standard deviation     0.26333 0.2418 0.06793
## Proportion of Variance 0.00462 0.0039 0.00031
## Cumulative Proportion  0.99579 0.9997 1.00000
```

```
#pca$x
```

We need to decide how many variables to use in our model. We would like the variables to explain roughly 90% of the variance in the data; from the summary of the pca we see that variables 1 through 6 explain 89.996% of the variance. We build a regression model with these factors and view the output:

```r
#Need to decide how many vars - 6
#Make regression model based on the pca to predict crime
PCcrime <- cbind(pca$x[,1:6],crime[,16])

#Make regression model
lm1 <- lm(PCcrime[,7] ~ ., data=as.data.frame(PCcrime[,1:6]))

#View summary output
summary(lm1)
```

```
##
## Call:
## lm(formula = PCcrime[, 7] ~ ., data = as.data.frame(PCcrime[,
##     1:6]))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -377.15 -172.23   25.81  132.10  480.38
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   905.09      35.35  25.604  < 2e-16 ***
## PC1            65.22      14.56   4.478 6.14e-05 ***
## PC2           -70.08      21.35  -3.283  0.00214 **
## PC3            25.19      25.23   0.998  0.32409
## PC4            69.45      33.14   2.095  0.04252 *
## PC5          -229.04      36.50  -6.275 1.94e-07 ***
## PC6           -60.21      48.04  -1.253  0.21734
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 242.3 on 40 degrees of freedom
## Multiple R-squared:  0.6586, Adjusted R-squared:  0.6074
## F-statistic: 12.86 on 6 and 40 DF,  p-value: 4.869e-08
```

```r
#Specify your new model in terms of the original variables (not the PC's) and
#compare to your solution from 8.2.

#Translate coefficients
coef <- lm1$coefficients[2:length(lm1$coefficients)]%*%t(pca$rotation[,1:(length(lm1$coefficients)-1)])

#Unscale coefficients and intercept
intercept <- lm1$coefficients[1]-sum(coef*sapply(crime[,1:15],mean)/sapply(crime[,1:15],sd))
coef <- coef/sapply(crime[,1:15],sd)

#Compare estimates and actuals
estimates <- as.matrix(crime[,1:15])%*%t(coef)+intercept

#estimates
#t(estimates)

#actual
#crime[,16]
```

```r
sum(estimates-crime[,16])
```

```
## [1] -2.728484e-12
```

To get the prediction for the new city given in the previous homework, we have to perform one final calculation:

```r
#Model estimates times new data should yield the prediction for the Crime value:

newobs=data.frame(M = 14.0,
So = 0,
Ed = 10.0,
Po1 = 12.0,
Po2 = 15.5,
LF = 0.640,
M.F = 94.0,
Pop = 150,
NW = 1.1,
U1 = 0.120,
U2 = 3.6,
Wealth = 3200,
Ineq = 20.1,
Prob = 0.04,
Time = 39.0)

#Dot product to get prediction for new city
as.matrix(coef)%*%t(as.matrix(newobs))
```

```
##          [,1]
## [1,] 7172.074
```

The result is 7172.074 - obviously this is not a good prediction since it is way outside of any reasonable value. We may have overfitting in our model due to the small dataset; thus we prefer the more reasonable prediction of 1038 we achieved in the previous homework.

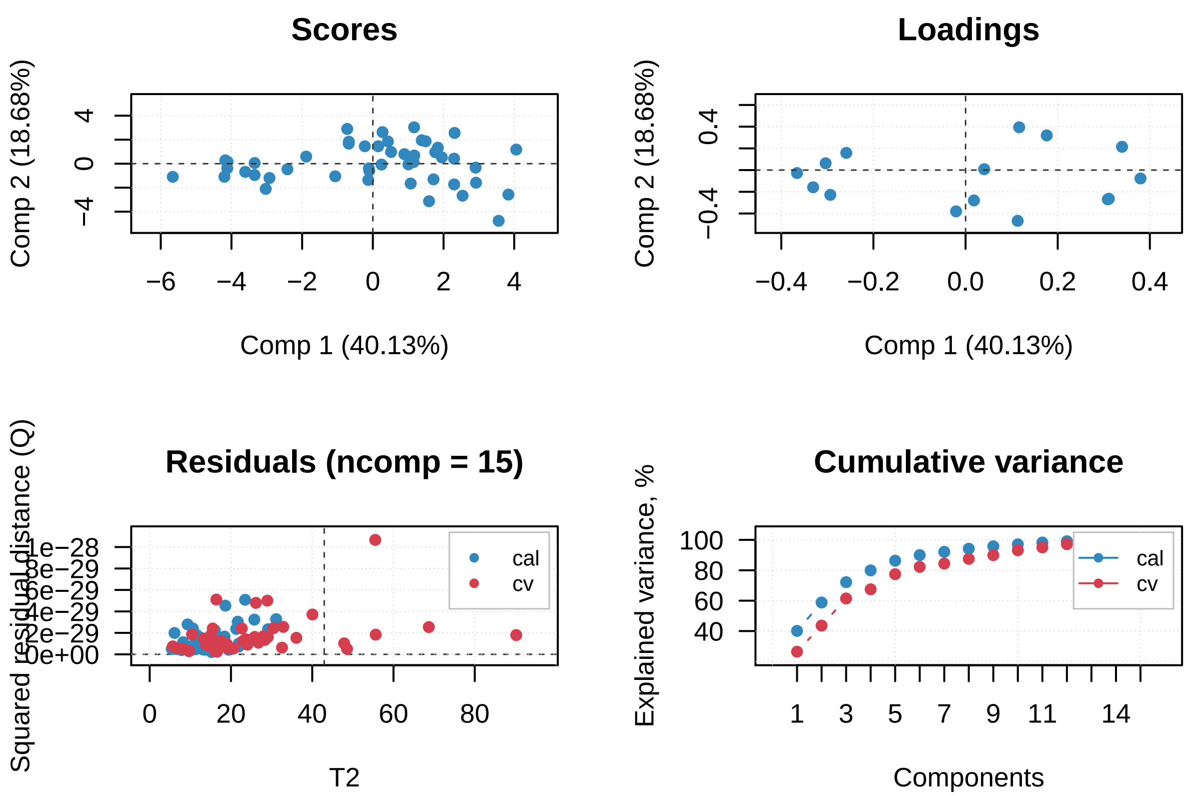
For 'extra credit', we will briefly investigate the use of the mdatools package for their pca function with the built-in cross validation feature. We have to choose the number of folds to be small because of the tiny dataset; we tried both 3 and 4 folds and got identical results:

```r
#EXTRA: perform cross validation and evaluate a larger number or all of the PCA models

#Cross-validated pca with 3&4 fold cv
pca_cv_3 <- pca(as.matrix(crime[,1:15]), scale=TRUE, cv=3)
#pca_cv_4 <- pca(as.matrix(crime[,1:15]), scale=TRUE, cv=4)

#Setting the model to select the first 6 components
model<-selectCompNum(pca_cv_3,6)

#Summary output of these models
summary(pca_cv_3)
```

```
##
## PCA model (class pca) summary
##
## Info:
##
##
##         Eigvals Expvar Cumexpvar
```

```
## Comp 1     6.019  40.13      40.13
## Comp 2     2.802  18.68      58.81
## Comp 3     2.005  13.37      72.17
## Comp 4     1.162   7.75      79.92
## Comp 5     0.958   6.39      86.31
## Comp 6     0.553   3.69      90.00
## Comp 7     0.322   2.15      92.14
## Comp 8     0.307   2.05      94.19
## Comp 9     0.235   1.57      95.76
## Comp 10    0.200   1.33      97.09
## Comp 11    0.176   1.17      98.26
## Comp 12    0.128   0.85      99.12
## Comp 13    0.069   0.46      99.58
## Comp 14    0.058   0.39      99.97
## Comp 15    0.005   0.03     100.00
```

```
#summary(pca_cv_4)

#Cross-validation results for folds=3&4
plot(pca_cv_3)
```




```
#plot(pca_cv_4)
```

Question 10.1

Using the same crime data set uscrime.txt as in Questions 8.2 and 9.1, find the best model you can using (a) a regression tree model, and (b) a random forest model. In R, you can use the tree package or the rpart package, and the randomForest package. For each model, describe one or two qualitative takeaways you get from analyzing the results (i.e., don't just stop when you have a good model, but interpret it too).

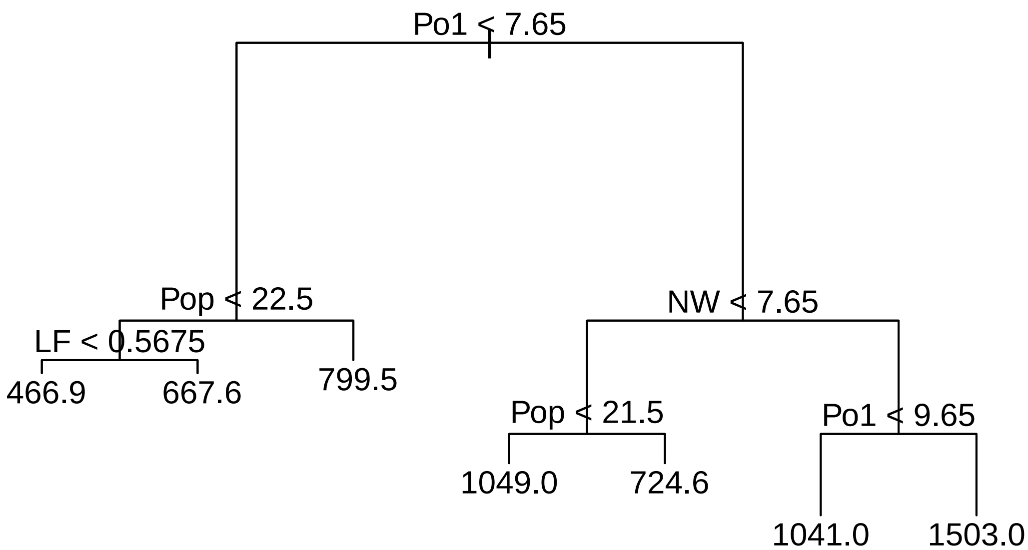
```
#Read in data
#crime<-read.csv("http://www.statsci.org/data/general/uscrime.txt",
#                  stringsAsFactors = FALSE, header=TRUE, sep='\t')

tree.data<-tree(Crime ~ ., data=crime)

summary(tree.data)
```

```
##
## Regression tree:
## tree(formula = Crime ~ ., data = crime)
## Variables actually used in tree construction:
## [1] "Po1" "Pop" "LF"  "NW"
## Number of terminal nodes:  7
## Residual mean deviance:  47390 = 1896000 / 40
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -573.900  -98.300   -1.545    0.000  110.600  490.100
```

```
#Notice that only 4 predictors were used in the construction of this tree

# More information about the way the tree was split
tree.data$frame
```

```
##        var  n        dev      yval splits.cutleft splits.cutright
## 1      Po1 47 6880927.66  905.0851          <7.65           >7.65
## 2      Pop 23  779243.48  669.6087          <22.5           >22.5
## 4       LF 12  243811.00  550.5000        <0.5675         >0.5675
## 8   <leaf>  7   48518.86  466.8571
## 9   <leaf>  5   77757.20  667.6000
## 5   <leaf> 11  179470.73  799.5455
## 3       NW 24 3604162.50 1130.7500          <7.65           >7.65
## 6      Pop 10  557574.90  886.9000          <21.5           >21.5
## 12 <leaf>  5  146390.80 1049.2000
## 13 <leaf>  5  147771.20  724.6000
## 7      Po1 14 2027224.93 1304.9286          <9.65           >9.65
## 14 <leaf>  6  170828.00 1041.0000
## 15 <leaf>  8 1124984.88 1502.8750
```

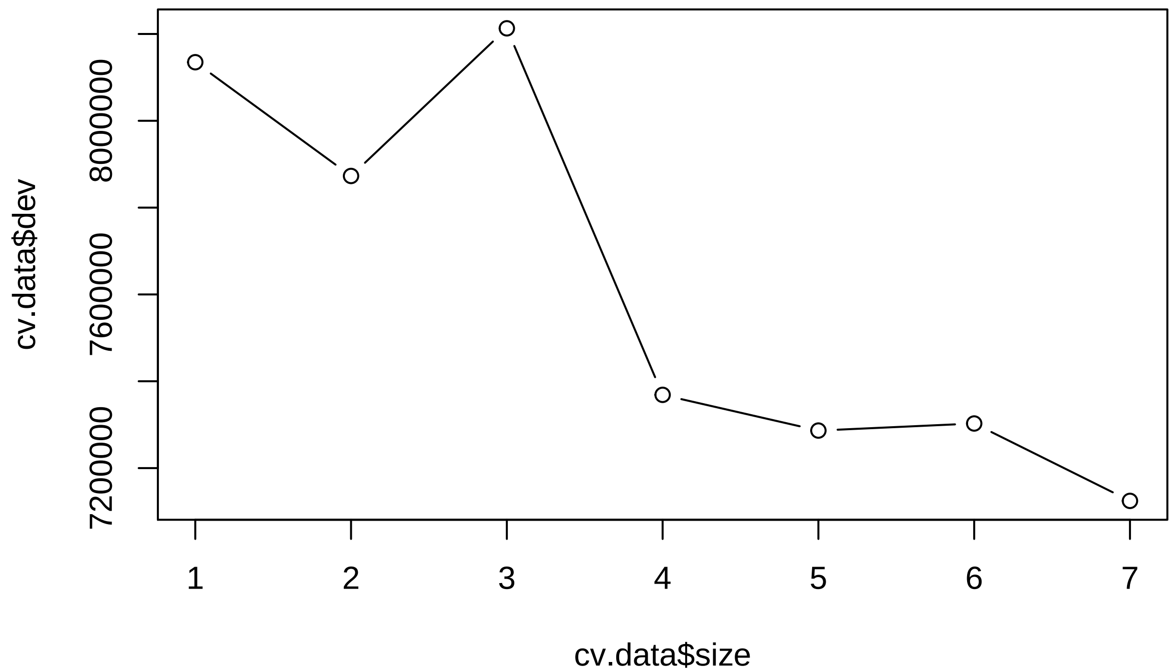
```
# Plot the regression tree
plot(tree.data)
text(tree.data)
```




```r
#Calculate R^2
yhat <- predict(tree.data)
SSres <- sum((yhat-crime$Crime)^2)
SStot <- sum((crime$Crime-mean(crime$Crime))^2)
R2tree <- 1-SSres/SStot

#Print value of R^2 obtained using this tree
R2tree
```

```
## [1] 0.7244962
```

```r
#Determine if pruning the tree will improve performance through cross-validation
# by looking at the deviance of trees with a different number of terminal nodes
# Deviance is a quality-of-fit statistic.

cv.data <- cv.tree(tree.data)
plot(cv.data$size, cv.data$dev, type="b")
```




```r
cv.data$dev
```

```
## [1] 7124512 7302733 7286447 7368726 8212980 7872920 8134948
```

```r
#Consider pruning tree
k=4
prune.data <- prune.tree(tree.data, best=k)

#Compare to results from previous homework

#(b) Random Forest

#Grow the random tree and set the number of predictors that we
# want to consider at each split of the tree (npred)

numpred <- 4
rf.data <- randomForest(Crime ~ ., data=crime, mtry=numpred, importance=TRUE)
rf.data
```

```
##
## Call:
##  randomForest(formula = Crime ~ ., data = crime, mtry = numpred,      importance = TRUE)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##           Mean of squared residuals: 81549.89
```

```
##                      % Var explained: 44.3
```

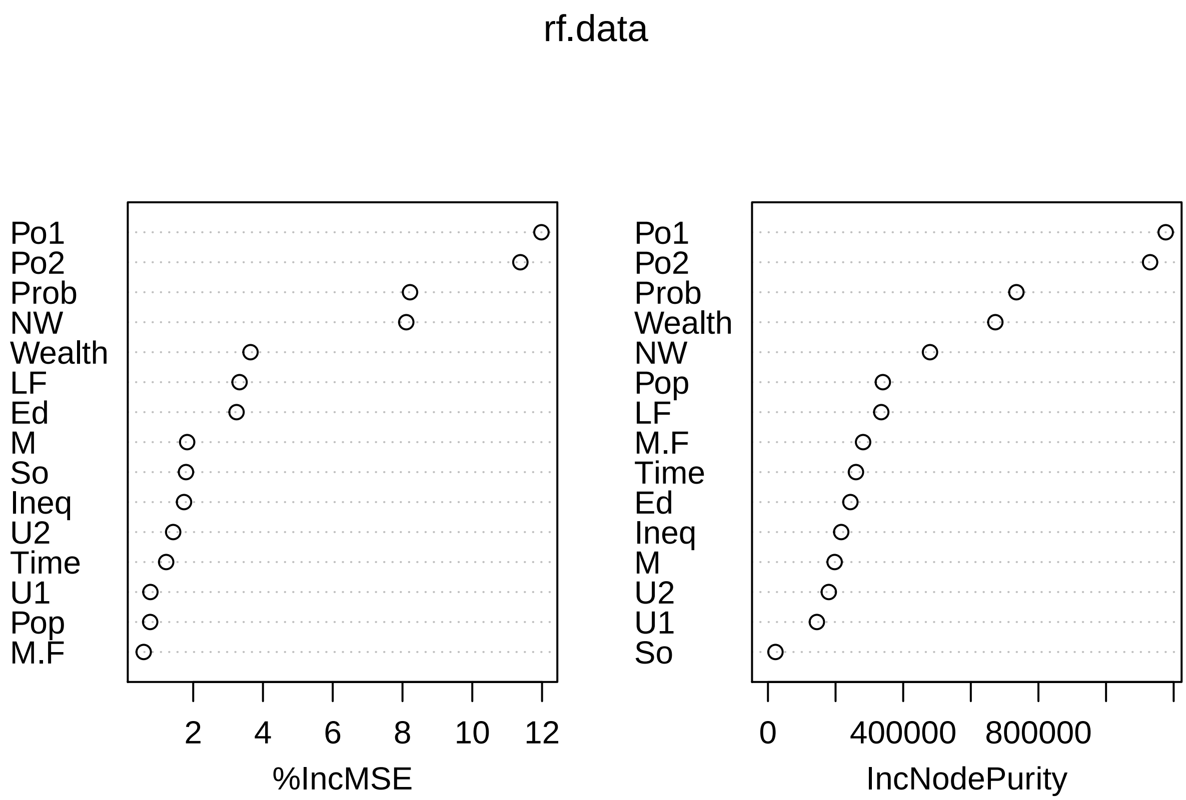
```r
#Calculate R^2
yhat.rf <- predict(rf.data)
SSres <- sum((yhat.rf-crime$Crime)^2)
SStot <- sum((crime$Crime - mean(crime$Crime))^2)
R2 <- 1 - SSres/SStot
R2
```

```
## [1] 0.4429756
```

```r
#We can't see a real model because there are many different trees, but we can see
#which variables are most important to the branching overall

importance(rf.data)
```

```
##           %IncMSE IncNodePurity
## M       1.8266196      197185.4
## So      1.7932215       22198.8
## Ed      3.2421415      243821.5
## Po1    11.9821724     1176416.1
## Po2    11.3781762     1130283.8
## LF      3.3286589      335126.7
## M.F     0.5807620      281379.8
## Pop     0.7665355      339786.2
## NW      8.1076292      479325.9
## U1      0.7731340      144622.5
## U2      1.4257463      179939.1
## Wealth  3.6431924      672539.2
## Ineq    1.7364944      216670.5
## Prob    8.2157115      734793.2
## Time    1.2249474      260245.8
```

```r
#Plot these importance measures
varImpPlot(rf.data)
```



Analysis of these models: in part (a), the manual calculation of R2 yields 0.7244962, whereas in part (b) it yields 0.4429756. We prefer the model in part (a) due to less overfitting.

Question 10.2

Describe a situation or problem from your job, everyday life, current events, etc., for which a logistic regression model would be appropriate. List some (up to 5) predictors that you might use.

Answer: In the health sciences, we would like to classify patients as at risk for a heart attack or not. Typical predictors or risk factors would include high blood pressure, cholesterol level, resting heart rate and family history of heart disease.

Question 10.3

1. Using the GermanCredit data set germancredit.txt from http://archive.ics.uci.edu/ml/machine-learning-databases/statlog/german / (description at http://archive.ics.uci.edu/ml/datasets/Statlog+%28German+Credit+Data%29), use logistic regression to find a good predictive model for whether credit applicants are good credit risks or not. Show your model (factors used and their coefficients), the software output, and the quality of fit. You can use the glm function in R. To get a logistic regression (logit) model on data where the response is either zero or one, use family=binomial(link="logit") in your glm function call.

```r
german<-read.table("http://archive.ics.uci.edu/ml/machine-learning-databases/statlog/german/german.data
                   sep=" ", stringsAsFactors = FALSE, header=FALSE)

#Convert response variable to 0-1
german$V21[german$V21==1]<-0
```

```r
german$V21[german$V21==2]<-1

#70/30 train/test split
set.seed(99)
ind<-sample(1:nrow(german), size=round(0.3*nrow(german)))
test = german[ind,]
train = german[-ind,]


#Part 1

log_reg_1 <- glm(V21 ~ ., family=binomial(link="logit"), data=train)

summary(log_reg_1)
```

```
##
## Call:
## glm(formula = V21 ~ ., family = binomial(link = "logit"), data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.6427  -0.6652  -0.3812   0.6246   2.7909
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.068e+00  1.326e+00   0.805 0.420687
## V1A12       -4.673e-01  2.712e-01  -1.723 0.084832 .
## V1A13       -1.143e+00  4.482e-01  -2.549 0.010790 *
## V1A14       -1.702e+00  2.885e-01  -5.899 3.65e-09 ***
## V2           2.661e-02  1.092e-02   2.436 0.014838 *
## V3A31        3.040e-01  7.032e-01   0.432 0.665462
## V3A32       -6.263e-01  5.310e-01  -1.180 0.238179
## V3A33       -1.095e+00  5.939e-01  -1.843 0.065336 .
## V3A34       -1.339e+00  5.276e-01  -2.537 0.011182 *
## V4A41       -1.393e+00  4.157e-01  -3.352 0.000803 ***
## V4A410      -1.955e+00  9.537e-01  -2.050 0.040374 *
## V4A42       -1.116e+00  3.347e-01  -3.333 0.000858 ***
## V4A43       -9.348e-01  3.085e-01  -3.030 0.002443 **
## V4A44       -1.358e+00  1.005e+00  -1.351 0.176554
## V4A45       -5.308e-01  7.429e-01  -0.714 0.474920
## V4A46        1.887e-01  5.027e-01   0.375 0.707341
## V4A48       -2.161e+00  1.336e+00  -1.618 0.105710
## V4A49       -5.940e-01  3.957e-01  -1.501 0.133349
## V5           1.684e-04  5.241e-05   3.212 0.001316 **
## V6A62       -5.339e-01  3.585e-01  -1.489 0.136460
## V6A63       -5.909e-01  5.031e-01  -1.175 0.240170
## V6A64       -1.711e+00  6.526e-01  -2.622 0.008741 **
## V6A65       -1.102e+00  3.270e-01  -3.370 0.000751 ***
## V7A72       -3.105e-01  5.412e-01  -0.574 0.566181
## V7A73       -2.169e-01  5.097e-01  -0.426 0.670460
## V7A74       -9.372e-01  5.527e-01  -1.696 0.089957 .
## V7A75       -3.662e-01  5.134e-01  -0.713 0.475624
## V8           4.412e-01  1.109e-01   3.978 6.96e-05 ***
## V9A92       -2.223e-01  5.163e-01  -0.431 0.666814
```

```
## V9A93       -5.910e-01  5.030e-01  -1.175 0.240014
## V9A94       -2.658e-01  5.893e-01  -0.451 0.651944
## V10A102      5.046e-01  4.870e-01   1.036 0.300133
## V10A103     -9.754e-01  5.012e-01  -1.946 0.051670 .
## V11         -4.595e-02  1.086e-01  -0.423 0.672149
## V12A122      3.339e-02  3.202e-01   0.104 0.916930
## V12A123     -1.417e-01  2.977e-01  -0.476 0.633984
## V12A124      6.683e-01  5.089e-01   1.313 0.189067
## V13         -1.358e-02  1.130e-02  -1.202 0.229554
## V14A142     -2.966e-01  5.692e-01  -0.521 0.602304
## V14A143     -7.517e-01  2.882e-01  -2.608 0.009109 **
## V15A152     -4.470e-01  2.932e-01  -1.525 0.127290
## V15A153     -7.951e-01  5.807e-01  -1.369 0.170920
## V16          1.827e-02  2.440e-01   0.075 0.940319
## V17A172      4.280e-01  8.151e-01   0.525 0.599513
## V17A173      3.424e-01  7.730e-01   0.443 0.657791
## V17A174      1.406e-01  7.676e-01   0.183 0.854700
## V18          2.006e-01  2.951e-01   0.680 0.496699
## V19A192     -1.541e-01  2.407e-01  -0.640 0.522024
## V20A202     -2.120e+00  8.993e-01  -2.358 0.018386 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 850.06  on 699  degrees of freedom
## Residual deviance: 610.66  on 651  degrees of freedom
## AIC: 708.66
##
## Number of Fisher Scoring iterations: 5
```

```r
#Now use automated variable selection process to determine best subset of predictors
#Backwards selection -output suppressed
backwards <- step(log_reg_1, trace=0)

#Formula of highest AIC model:
formula(backwards)
```

```
## V21 ~ V1 + V2 + V3 + V4 + V5 + V6 + V7 + V8 + V10 + V13 + V14 +
##     V20
```

```r
#Running a second model with the chosen predictors:
log_reg_2 <- glm(V21 ~ V1 + V2 + V3 + V4 + V5 + V6 + V8 + V9 + V10 + V13 +
                   V14 + V15 + V19 + V20, family=binomial(link="logit"), data=train)
#Summary of new model
summary(log_reg_2)
```

```
##
## Call:
## glm(formula = V21 ~ V1 + V2 + V3 + V4 + V5 + V6 + V8 + V9 + V10 +
##     V13 + V14 + V15 + V19 + V20, family = binomial(link = "logit"),
##     data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.5837  -0.6785  -0.3896   0.6413   2.7931
```

```
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.303e+00  9.181e-01   1.419 0.155831
## V1A12       -4.639e-01  2.640e-01  -1.757 0.078891 .
## V1A13       -1.162e+00  4.390e-01  -2.646 0.008142 **
## V1A14       -1.704e+00  2.804e-01  -6.076 1.23e-09 ***
## V2           2.440e-02  1.054e-02   2.315 0.020598 *
## V3A31        2.274e-01  6.678e-01   0.340 0.733497
## V3A32       -7.265e-01  4.927e-01  -1.474 0.140386
## V3A33       -1.168e+00  5.810e-01  -2.010 0.044417 *
## V3A34       -1.457e+00  5.142e-01  -2.833 0.004611 **
## V4A41       -1.422e+00  4.080e-01  -3.484 0.000493 ***
## V4A410      -1.881e+00  9.519e-01  -1.976 0.048200 *
## V4A42       -1.074e+00  3.249e-01  -3.305 0.000949 ***
## V4A43       -9.649e-01  3.025e-01  -3.190 0.001422 **
## V4A44       -1.252e+00  9.572e-01  -1.308 0.190918
## V4A45       -3.651e-01  7.208e-01  -0.507 0.612463
## V4A46        2.666e-01  4.922e-01   0.542 0.588043
## V4A48       -2.327e+00  1.337e+00  -1.740 0.081853 .
## V4A49       -6.342e-01  3.840e-01  -1.652 0.098617 .
## V5           1.606e-04  4.906e-05   3.272 0.001066 **
## V6A62       -5.759e-01  3.514e-01  -1.639 0.101237
## V6A63       -5.790e-01  4.892e-01  -1.184 0.236601
## V6A64       -1.646e+00  6.356e-01  -2.590 0.009589 **
## V6A65       -1.078e+00  3.165e-01  -3.408 0.000656 ***
## V8           4.244e-01  1.073e-01   3.957 7.59e-05 ***
## V9A92       -2.542e-01  4.942e-01  -0.514 0.607066
## V9A93       -6.497e-01  4.804e-01  -1.352 0.176295
## V9A94       -2.674e-01  5.737e-01  -0.466 0.641168
## V10A102      6.197e-01  4.736e-01   1.309 0.190679
## V10A103     -9.392e-01  4.970e-01  -1.890 0.058784 .
## V13         -1.110e-02  1.027e-02  -1.081 0.279792
## V14A142     -2.302e-01  5.468e-01  -0.421 0.673707
## V14A143     -7.777e-01  2.811e-01  -2.767 0.005658 **
## V15A152     -4.015e-01  2.772e-01  -1.449 0.147443
## V15A153     -8.577e-02  4.155e-01  -0.206 0.836445
## V19A192     -2.161e-01  2.214e-01  -0.976 0.329080
## V20A202     -1.961e+00  8.834e-01  -2.219 0.026466 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 850.06  on 699  degrees of freedom
## Residual deviance: 620.61  on 664  degrees of freedom
## AIC: 692.61
##
## Number of Fisher Scoring iterations: 5
```

2. Because the model gives a result between 0 and 1, it requires setting a threshold probability to separate between "good" and "bad" answers. In this data set, they estimate that incorrectly identifying a bad customer as good, is 5 times worse than incorrectly classifying a good customer as bad. Determine a good threshold probability based on your model.

```r
#Part 2
#You need to use the trained model to get predictions on the training/validation sets
# Then you calculate the loss based on the confusion matrix and what your model predicts
# one cost will be multiplied by 5 and the other by 1

#predictions
pred<-predict(log_reg_2,test, type="response")

# Borrowed from piazza post @348
# Optimal threshold probability
t_hold <- optimalCutoff(test$V21, pred)[1]
t_hold
```

```
## [1] 0.4806097
```

```r
#mis-classification error rate
misClassError(test$V21, pred, threshold = t_hold)
```

```
## [1] 0.2333
```

```r
# Sensitivity
sensitivity(test$V21, pred, threshold = t_hold)
```

```
## [1] 0.5698925
```

```r
#specificity
specificity(test$V21, pred, threshold = t_hold)
```

```
## [1] 0.8550725
```

```r
#confusion matrix. Cost can be calculated from this matrix . See lesson video for detail
cm<-confusionMatrix(test$V21, pred, threshold = t_hold)
cm
```

```
##     0  1
## 0 177 40
## 1  30 53
```

```r
#Calculate cost
cost<-cm[1,2]*1+cm[2,1]*5
cost
```

```
## [1] 190
```

References:

(9.1)

Using PCA for feature selection

https://stats.stackexchange.com/questions/27300/using-principal-component-analysis-pca-for-feature-selection/27310

(10.3)

Evaluating logistic regression models

https://www.r-bloggers.com/evaluating-logistic-regression-models/

Stepwise Logistic Regression in R

http://www.utstat.toronto.edu/~brunner/oldclass/appliedf11/handouts/2101f11StepwiseLogisticR.pdf

Evaluating cost-sensitive classification in the German dataset

https://mlr-org.github.io/mlr-tutorial/release/html/cost_sensitive_classif/index.html