ISYE6501 HOMEWORK WEEK FOUR

Ritwik Bhatia

6/11/2018

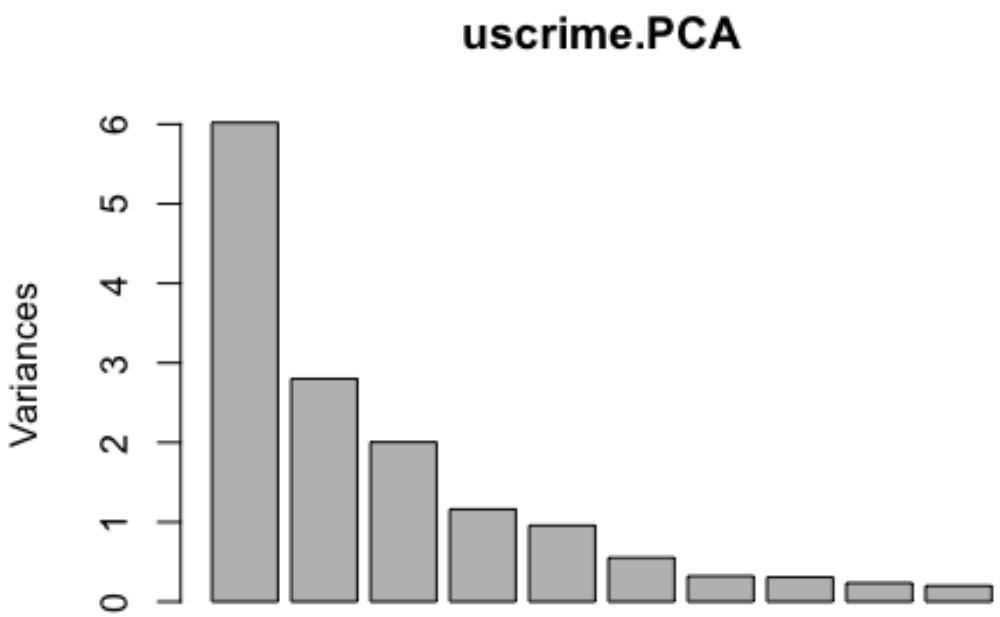
Question 9.1

In order to answer this question, I first conducted a PCA and plotted the results, as seen below. I then ran a regression using the PC predictors and crime response, and I found a weak model with an R^2 value of only 0.3091.

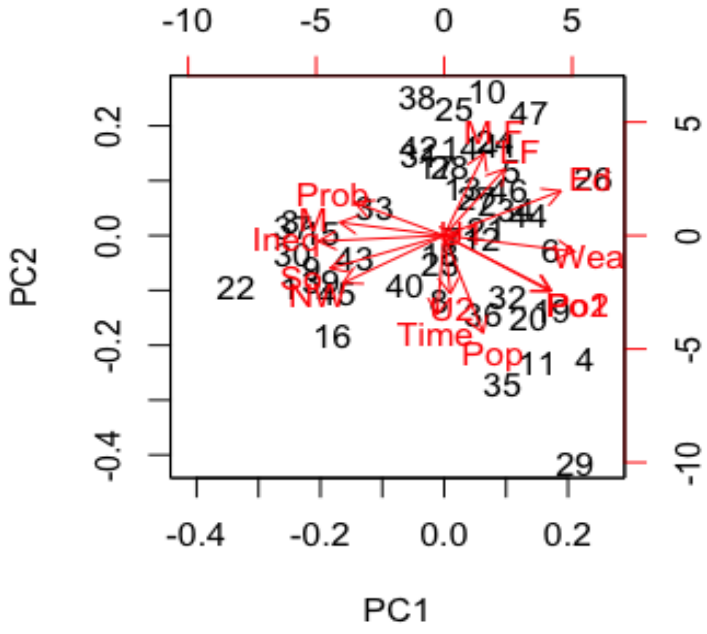
Next, I expressed the above model in terms of the original coefficients. I made sure to unscale the data, as I had originally scaled the data in the first step.

My model and the coefficients of the original variable can be found at the end of the code below.

```
setwd("~/Desktop/ISYE/Week 2")
CRIMEHW<-read.table("CRIMEHW.txt",header=TRUE)

#we use the prcomp() to create principal components of the uscrime dataframe
with the exception of the dependent variable: "Crime"
uscrime.PCA <- prcomp(CRIMEHW[,1:15], scale. = TRUE)

summary(uscrime.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

plot(uscrime.PCA)
```




```
biplot(uscrime.PCA)
```



```r
uscrime.PCA.4<-uscrime.PCA$x[,1:4]
#use the column bind (cbind()) to append
pccrime <- cbind(uscrime.PCA.4, CRIMEHW[,16])

#Fit linear model with the PC predictors and crime response
lm.model.pca <- lm(pccrime[,5]~., data = as.data.frame(pccrime[,1:4]))
summary(lm.model.pca)
```

```
##
## Call:
## lm(formula = pccrime[, 5] ~ ., data = as.data.frame(pccrime[,
##     1:4]))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -557.76 -210.91  -29.08  197.26  810.35
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   905.09      49.07  18.443  < 2e-16 ***
## PC1            65.22      20.22   3.225  0.00244 **
## PC2           -70.08      29.63  -2.365  0.02273 *
## PC3            25.19      35.03   0.719  0.47602
## PC4            69.45      46.01   1.509  0.13872
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 336.4 on 42 degrees of freedom
## Multiple R-squared:  0.3091, Adjusted R-squared:  0.2433
## F-statistic: 4.698 on 4 and 42 DF,  p-value: 0.003178
```

```r
last<-t(t(uscrime.PCA$x %*% t(uscrime.PCA$rotation)) * uscrime.PCA$scale + us
crime.PCA$center)
colMeans(last)
```

```
##            M           So           Ed          Po1          Po2
## 1.385745e+01 3.404255e-01 1.056383e+01 8.500000e+00 8.023404e+00
##           LF          M.F          Pop           NW           U1
## 5.611915e-01 9.830213e+01 3.661702e+01 1.011277e+01 9.546809e-02
##           U2       Wealth         Ineq         Prob         Time
## 3.397872e+00 5.253830e+03 1.940000e+01 4.709138e-02 2.659792e+01
```

```r
last<-t(t(uscrime.PCA$x %*% t(uscrime.PCA$rotation)) * uscrime.PCA$scale + us
crime.PCA$center)

lm(pccrime[,5]~.,data=as.data.frame(last))
```

```
##
## Call:
## lm(formula = pccrime[, 5] ~ ., data = as.data.frame(last))
##
```

```
## Coefficients:
## (Intercept)            M           So           Ed          Po1
##  -5.984e+03    8.783e+01   -3.803e+00    1.883e+02    1.928e+02
##         Po2           LF          M.F          Pop           NW
##  -1.094e+02   -6.638e+02    1.741e+01   -7.330e-01    4.204e+00
##          U1           U2       Wealth         Ineq         Prob
##  -5.827e+03    1.678e+02    9.617e-02    7.067e+01   -4.855e+03
##        Time
##  -3.479e+00
```

Question 10.1(a)

The code below is rather simple and easy to follow. I will now present some qualitative facts about the tree model, which has been visualized on the following page.

We can see that the tree utilizes 4 variables: "Po1", "Pop", "LF", and "NW". In addition, there are a total of 7 terminal nodes, or branches. Interestingly, Po1 is used to branch off at the very top, and again at the bottom in the righter bottom branch.

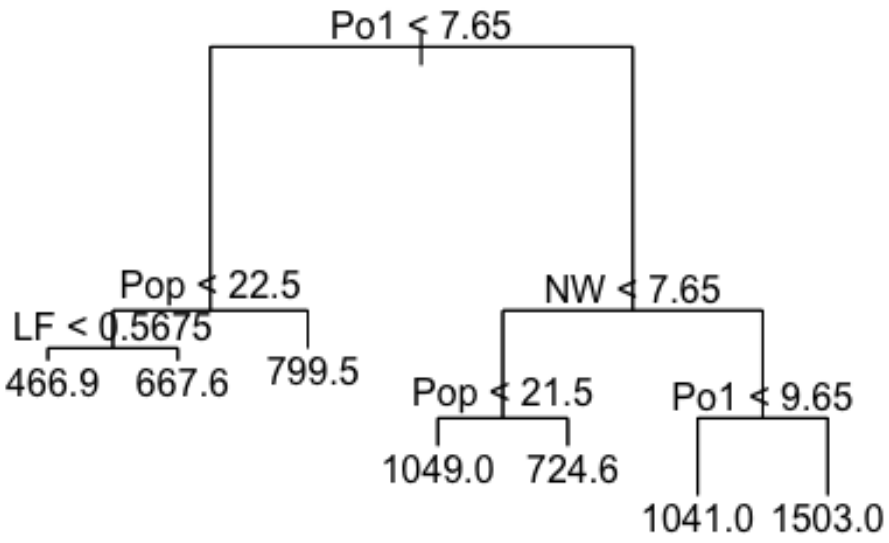
```
library('tree')
res <- tree(Crime ~ M + So + Ed + Po1 + Po2 + LF + M.F + Pop + NW + U1 + U2 +
Wealth + Ineq + Prob + Time, CRIMEHW)
summary(res)
```

```
##
## Regression tree:
## tree(formula = Crime ~ M + So + Ed + Po1 + Po2 + LF + M.F + Pop +
##     NW + U1 + U2 + Wealth + Ineq + Prob + Time, data = CRIMEHW)
## 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
```

```
res$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(res)
text(res)
```



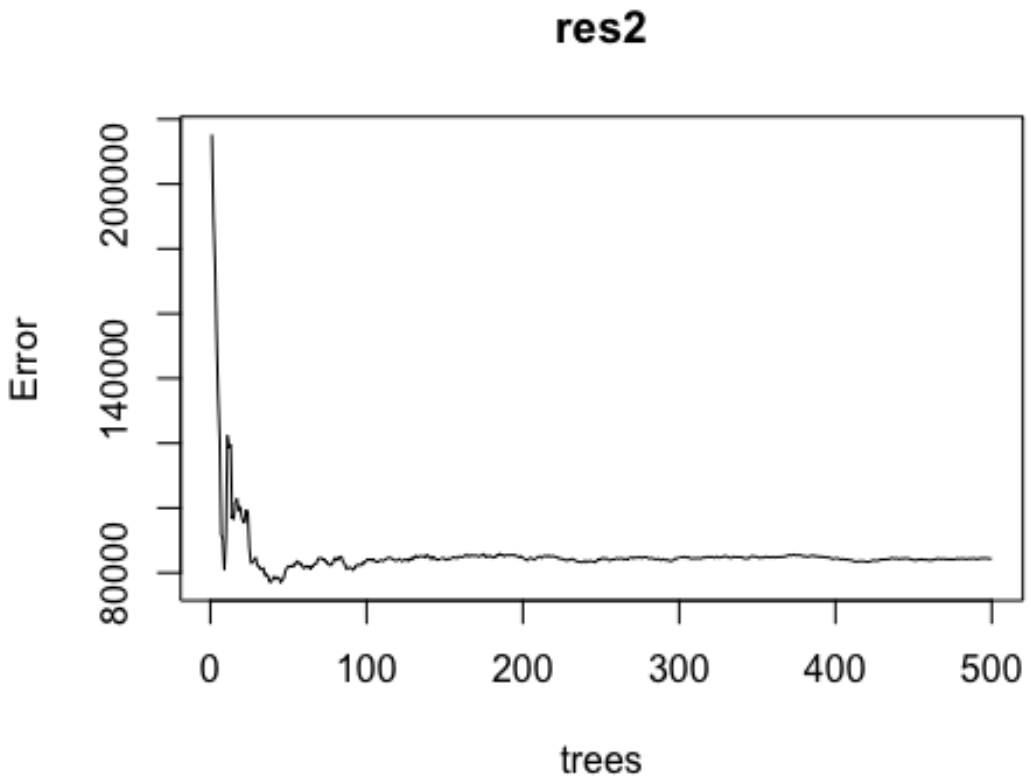

```
res

## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 47 6881000  905.1
##    2) Po1 < 7.65 23  779200  669.6
##      4) Pop < 22.5 12  243800  550.5
##        8) LF < 0.5675 7   48520  466.9 *
##        9) LF > 0.5675 5   77760  667.6 *
##      5) Pop > 22.5 11  179500  799.5 *
##    3) Po1 > 7.65 24 3604000 1131.0
##      6) NW < 7.65 10  557600  886.9
##       12) Pop < 21.5 5  146400 1049.0 *
##       13) Pop > 21.5 5  147800  724.6 *
##      7) NW > 7.65 14 2027000 1305.0
```

```
##       14) Po1 < 9.65 6  170800 1041.0 *
##       15) Po1 > 9.65 8 1125000 1503.0 *

yhat<-predict(res)
SSres<-sum((yhat-CRIMEHW$Crime)^2)
SStot<-sum((CRIMEHW$Crime-mean(CRIMEHW$Crime))^2)
R2<-1-SSres/SStot
R2

## [1] 0.7244962
```

Question 10.1(b)

This part of the question is similar to the regression tree model that we created earlier. We see that the random forest model here has a smaller R^2 value than the regression tree model. Similar details as mentioned in part (a) can be seen in the code below.

```
library('randomForest')

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

numpred<-4
res2 <- randomForest(formula = Crime ~ M + So + Ed + Po1 + Po2 + LF + M.F + P
op + NW + U1 + U2 + Wealth + Ineq + Prob + Time, data=CRIMEHW, mtry=numpred,
importance=TRUE)
summary(res2)

##                 Length Class  Mode
## call              5    -none- call
## type              1    -none- character
## predicted        47    -none- numeric
## mse             500    -none- numeric
## rsq             500    -none- numeric
## oob.times        47    -none- numeric
## importance       30    -none- numeric
## importanceSD     15    -none- numeric
## localImportance   0    -none- NULL
## proximity         0    -none- NULL
## ntree             1    -none- numeric
## mtry              1    -none- numeric
## forest           11    -none- list
## coefs             0    -none- NULL
## y                47    -none- numeric
## test              0    -none- NULL
## inbag             0    -none- NULL
## terms             3    terms  call

plot(res2)
```




```
res2

##
## Call:
##  randomForest(formula = Crime ~ M + So + Ed + Po1 + Po2 + LF +      M.F +
Pop + NW + U1 + U2 + Wealth + Ineq + Prob + Time, data = CRIMEHW,      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: 84370.9
##                     % Var explained: 42.37

yhat_res2<-predict(res2)
SSres2<-sum((yhat_res2-CRIMEHW$Crime)^2)
SStot2<-sum((CRIMEHW$Crime-mean(CRIMEHW$Crime))^2)
R2<-1-SSres2/SStot2
R2

## [1] 0.4237067
```

Question 10.2

A logistic regression model would be useful for a bank that is determining whether or not to grant people loans. The use of a regression model would help predict whether or not an applicant would pay back or default on their loan. If an applicant has paid back their loan, they receive a value of 1. If they haven't, they receive a value of 0. Some of the predictors that the bank could use to evaluate the applicant and determine the probability that an applicant will repay their loan are:

1. Credit Score
2. Income
3. Length of Loan
4. Outstanding Debts
5. Marital Status/# of Dependents

Question 10.3.1

To answer this question, I first ran a logistical regression using all the variables. After doing so, I ran another logistical regression, this time using only the variables that were marked as significant after my first step.

As a result, I had a tighter regression using now using only 10 variables.

```
german_credit <-as.data.frame(read.table("creditgerman.txt"))



german_credit$V21[german_credit$V21==1]<-0
german_credit$V21[german_credit$V21==2]<-1



#run logistical regression
fit<-glm(V21 ~ V1 + V2 + V3 + V4+V5+V6+V7+V8+V9+V10+V11+V12+V13+V14+V15+V16+V
17+V18+V19+V20, family=binomial(link="logit"), german_credit)

summary(fit)

##
## Call:
## glm(formula = V21 ~ V1 + V2 + V3 + V4 + V5 + V6 + V7 + V8 + V9 +
##     V10 + V11 + V12 + V13 + V14 + V15 + V16 + V17 + V18 + V19 +
##     V20, family = binomial(link = "logit"), data = german_credit)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3410  -0.6994  -0.3752   0.7095   2.6116
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  4.005e-01  1.084e+00   0.369 0.711869
```

```
## V1A12       -3.749e-01  2.179e-01  -1.720 0.085400 .
## V1A13       -9.657e-01  3.692e-01  -2.616 0.008905 **
## V1A14       -1.712e+00  2.322e-01  -7.373 1.66e-13 ***
## V2           2.786e-02  9.296e-03   2.997 0.002724 **
## V3A31        1.434e-01  5.489e-01   0.261 0.793921
## V3A32       -5.861e-01  4.305e-01  -1.362 0.173348
## V3A33       -8.532e-01  4.717e-01  -1.809 0.070470 .
## V3A34       -1.436e+00  4.399e-01  -3.264 0.001099 **
## V4A41       -1.666e+00  3.743e-01  -4.452 8.51e-06 ***
## V4A410      -1.489e+00  7.764e-01  -1.918 0.055163 .
## V4A42       -7.916e-01  2.610e-01  -3.033 0.002421 **
## V4A43       -8.916e-01  2.471e-01  -3.609 0.000308 ***
## V4A44       -5.228e-01  7.623e-01  -0.686 0.492831
## V4A45       -2.164e-01  5.500e-01  -0.393 0.694000
## V4A46        3.628e-02  3.965e-01   0.092 0.927082
## V4A48       -2.059e+00  1.212e+00  -1.699 0.089297 .
## V4A49       -7.401e-01  3.339e-01  -2.216 0.026668 *
## V5           1.283e-04  4.444e-05   2.887 0.003894 **
## V6A62       -3.577e-01  2.861e-01  -1.250 0.211130
## V6A63       -3.761e-01  4.011e-01  -0.938 0.348476
## V6A64       -1.339e+00  5.249e-01  -2.551 0.010729 *
## V6A65       -9.467e-01  2.625e-01  -3.607 0.000310 ***
## V7A72       -6.691e-02  4.270e-01  -0.157 0.875475
## V7A73       -1.828e-01  4.105e-01  -0.445 0.656049
## V7A74       -8.310e-01  4.455e-01  -1.866 0.062110 .
## V7A75       -2.766e-01  4.134e-01  -0.669 0.503410
## V8           3.301e-01  8.828e-02   3.739 0.000185 ***
## V9A92       -2.755e-01  3.865e-01  -0.713 0.476040
## V9A93       -8.161e-01  3.799e-01  -2.148 0.031718 *
## V9A94       -3.671e-01  4.537e-01  -0.809 0.418448
## V10A102      4.360e-01  4.101e-01   1.063 0.287700
## V10A103     -9.786e-01  4.243e-01  -2.307 0.021072 *
## V11          4.776e-03  8.641e-02   0.055 0.955920
## V12A122      2.814e-01  2.534e-01   1.111 0.266630
## V12A123      1.945e-01  2.360e-01   0.824 0.409743
## V12A124      7.304e-01  4.245e-01   1.721 0.085308 .
## V13         -1.454e-02  9.222e-03  -1.576 0.114982
## V14A142     -1.232e-01  4.119e-01  -0.299 0.764878
## V14A143     -6.463e-01  2.391e-01  -2.703 0.006871 **
## V15A152     -4.436e-01  2.347e-01  -1.890 0.058715 .
## V15A153     -6.839e-01  4.770e-01  -1.434 0.151657
## V16          2.721e-01  1.895e-01   1.436 0.151109
## V17A172      5.361e-01  6.796e-01   0.789 0.430160
## V17A173      5.547e-01  6.549e-01   0.847 0.397015
## V17A174      4.795e-01  6.623e-01   0.724 0.469086
## V18          2.647e-01  2.492e-01   1.062 0.288249
## V19A192     -3.000e-01  2.013e-01  -1.491 0.136060
## V20A202     -1.392e+00  6.258e-01  -2.225 0.026095 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1221.73  on 999  degrees of freedom
## Residual deviance:  895.82  on 951  degrees of freedom
## AIC: 993.82
##
## Number of Fisher Scoring iterations: 5
```

```
fit
```

```
##
## Call:  glm(formula = V21 ~ V1 + V2 + V3 + V4 + V5 + V6 + V7 + V8 + V9 +
##     V10 + V11 + V12 + V13 + V14 + V15 + V16 + V17 + V18 + V19 +
##     V20, family = binomial(link = "logit"), data = german_credit)
##
## Coefficients:
## (Intercept)        V1A12        V1A13        V1A14           V2
##   0.4005027   -0.3748534   -0.9656768   -1.7118880    0.0278633
##       V3A31        V3A32        V3A33        V3A34        V4A41
##   0.1433777   -0.5861136   -0.8531614   -1.4357716   -1.6664670
##      V4A410        V4A42        V4A43        V4A44        V4A45
##  -1.4887859   -0.7916104   -0.8915834   -0.5227827   -0.2163959
##       V4A46        V4A48        V4A49           V5        V6A62
##   0.0362838   -2.0594328   -0.7400868    0.0001283   -0.3577406
##       V6A63        V6A64        V6A65        V7A72        V7A73
##  -0.3760729   -1.3391988   -0.9466892   -0.0669104   -0.1828310
##       V7A74        V7A75           V8        V9A92        V9A93
##  -0.8310018   -0.2766245    0.3300898   -0.2754548   -0.8160779
##       V9A94      V10A102      V10A103          V11      V12A122
##  -0.3670719    0.4360476   -0.9786160    0.0047761    0.2814382
##     V12A123      V12A124          V13      V14A142      V14A143
##   0.1945347    0.7304477   -0.0145355   -0.1232006   -0.6463287
##     V15A152      V15A153          V16      V17A172      V17A173
##  -0.4436210   -0.6838602    0.2720759    0.5361304    0.5547175
##     V17A174          V18      V19A192      V20A202
##   0.4794752    0.2646714   -0.3000080   -1.3922159
##
## Degrees of Freedom: 999 Total (i.e. Null);  951 Residual
## Null Deviance:       1222
## Residual Deviance: 895.8     AIC: 993.8
```

```
fit2<-glm(V21 ~ V1 + V2 + V3 + V4+V5+V6+V8+V9+V10+V14, family=binomial(link="
logit"), german_credit)
summary(fit2)
```

```
##
## Call:
## glm(formula = V21 ~ V1 + V2 + V3 + V4 + V5 + V6 + V8 + V9 + V10 +
##     V14, family = binomial(link = "logit"), data = german_credit)
##
```

```
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1873  -0.7104  -0.3974   0.7754   2.7301
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.679412   0.630832   1.077 0.281476
## V1A12       -0.364776   0.207942  -1.754 0.079393 .
## V1A13       -1.037482   0.356786  -2.908 0.003639 **
## V1A14       -1.749674   0.225154  -7.771 7.79e-15 ***
## V2           0.029895   0.008813   3.392 0.000693 ***
## V3A31       -0.088694   0.515165  -0.172 0.863308
## V3A32       -0.817258   0.401282  -2.037 0.041689 *
## V3A33       -0.931562   0.458302  -2.033 0.042089 *
## V3A34       -1.570271   0.423629  -3.707 0.000210 ***
## V4A41       -1.513847   0.359198  -4.215 2.50e-05 ***
## V4A410      -1.571038   0.776281  -2.024 0.042991 *
## V4A42       -0.618489   0.246819  -2.506 0.012216 *
## V4A43       -0.854572   0.238088  -3.589 0.000332 ***
## V4A44       -0.502801   0.743140  -0.677 0.498666
## V4A45       -0.134902   0.534555  -0.252 0.800760
## V4A46        0.234448   0.384497   0.610 0.542025
## V4A48       -2.046610   1.219517  -1.678 0.093306 .
## V4A49       -0.763255   0.321175  -2.376 0.017480 *
## V5           0.000109   0.000041   2.659 0.007828 **
## V6A62       -0.273278   0.272730  -1.002 0.316341
## V6A63       -0.419210   0.392002  -1.069 0.284885
## V6A64       -1.330271   0.502794  -2.646 0.008151 **
## V6A65       -0.991523   0.253113  -3.917 8.95e-05 ***
## V8           0.312397   0.083775   3.729 0.000192 ***
## V9A92       -0.109894   0.367005  -0.299 0.764609
## V9A93       -0.769223   0.360808  -2.132 0.033012 *
## V9A94       -0.301929   0.434808  -0.694 0.487434
## V10A102      0.460928   0.392694   1.174 0.240491
## V10A103     -1.104043   0.409810  -2.694 0.007059 **
## V14A142     -0.120086   0.400392  -0.300 0.764237
## V14A143     -0.641435   0.232522  -2.759 0.005805 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1221.73  on 999  degrees of freedom
## Residual deviance:  926.22  on 969  degrees of freedom
## AIC: 988.22
##
## Number of Fisher Scoring iterations: 5

fit2
```

```
##
## Call:  glm(formula = V21 ~ V1 + V2 + V3 + V4 + V5 + V6 + V8 + V9 + V10 +
##     V14, family = binomial(link = "logit"), data = german_credit)
##
## Coefficients:
## (Intercept)        V1A12        V1A13        V1A14           V2
##    0.679412    -0.364776    -1.037482    -1.749674     0.029895
##       V3A31        V3A32        V3A33        V3A34        V4A41
##   -0.088694    -0.817258    -0.931562    -1.570271    -1.513847
##      V4A410        V4A42        V4A43        V4A44        V4A45
##   -1.571038    -0.618489    -0.854572    -0.502801    -0.134902
##       V4A46        V4A48        V4A49           V5        V6A62
##    0.234448    -2.046610    -0.763255     0.000109    -0.273278
##       V6A63        V6A64        V6A65           V8        V9A92
##   -0.419210    -1.330271    -0.991523     0.312397    -0.109894
##       V9A93        V9A94      V10A102      V10A103      V14A142
##   -0.769223    -0.301929     0.460928    -1.104043    -0.120086
##     V14A143
##   -0.641435
##
## Degrees of Freedom: 999 Total (i.e. Null);  969 Residual
## Null Deviance:       1222
## Residual Deviance: 926.2     AIC: 988.2
```

Question 10.3.2

To solve this question, I tested different thresholds from 0.01 to 1.00, in steps of 0.01. I assigned a return of 0 to true negative (no cost) and a return of 0 to true positive (no cost). If the value returned was greated than my threshold value, but was predicted falsely to be good, then I returned a high cost of 5. Lastly, if I falsely predicted a good customer to be bad, I returned a low cost of just 1.

The final graph on the proceeding page shows that we can minimize costs by using a threshold between 0.05 and 0.15 (indices of 5 and 15).

```
library("boot")
cost_res <- vector(mode="list", length=100)
for (i in 1:100) {
  my_threshold<-i/100 #iterate over threshold 0.01, 0.02...0.99, 1.00

  mycost <- function(r, pi){
    if (r==1) { #observed bad customer
      if (pi > my_threshold) { #predicted good
        return(5)
      } else {
        return(0) #true negative, no cost
      }
    }

    if (r==0) { #observed good customer
```

```
      if (pi < my_threshold) { #predicted bad
        return(1)
      } else {
        return(0) #true positive, no cost
      }
    }
  }

  current_res <- cv.glm(data = german_credit, glmfit = fit2, cost = mycost, K
= 10)
  print(my_threshold)
  print(current_res$delta[1])

  cost_res[[i]] <- (current_res$delta[1])
}

plot(unlist(cost_res), type='l')
```