Homework 4

Chen Yi-Ju(Ernie)

2020/6/9

Question 9.1

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.

Answer: Using an estimation using both 4 and 5 PCAs(assumption based on scree plot), the results were worse than the original regression. This may be due to overtraining on the original regression.

```
setwd("D:/ernie/self-study/GTxMicroMasters/Introduction to Analytics Modeling/week4/homework")
library(tidyverse)
crime <- read.table("uscrime.txt",header = T) %>%
  data.frame()
```

```
#original model from 8.2 with 9 variables
model <- lm (data = crime , Crime ~ Ed + Ineq + LF + M + M.F +Po1 + Pop + Prob  + Time )
summary(model)
```

```
##
## Call:
## lm(formula = Crime ~ Ed + Ineq + LF + M + M.F + Po1 + Pop + Prob +
##     Time, data = crime)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -468.62 -100.73   -6.44  139.91  520.35
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -5189.5782  1460.9341  -3.552 0.001063 **
## Ed            140.9730    57.8900   2.435 0.019823 *
## Ineq           68.7477    15.8765   4.330 0.000109 ***
## LF           -609.2340  1065.0117  -0.572 0.570751
## M              68.3357    35.3331   1.934 0.060784 .
## M.F            17.8666    15.2790   1.169 0.249738
## Po1           126.5215    17.3893   7.276 1.22e-08 ***
## Pop            -0.6526     1.2716  -0.513 0.610833
## Prob        -4006.6838  2033.8562  -1.970 0.056359 .
## Time            1.7858     6.6248   0.270 0.788995
```

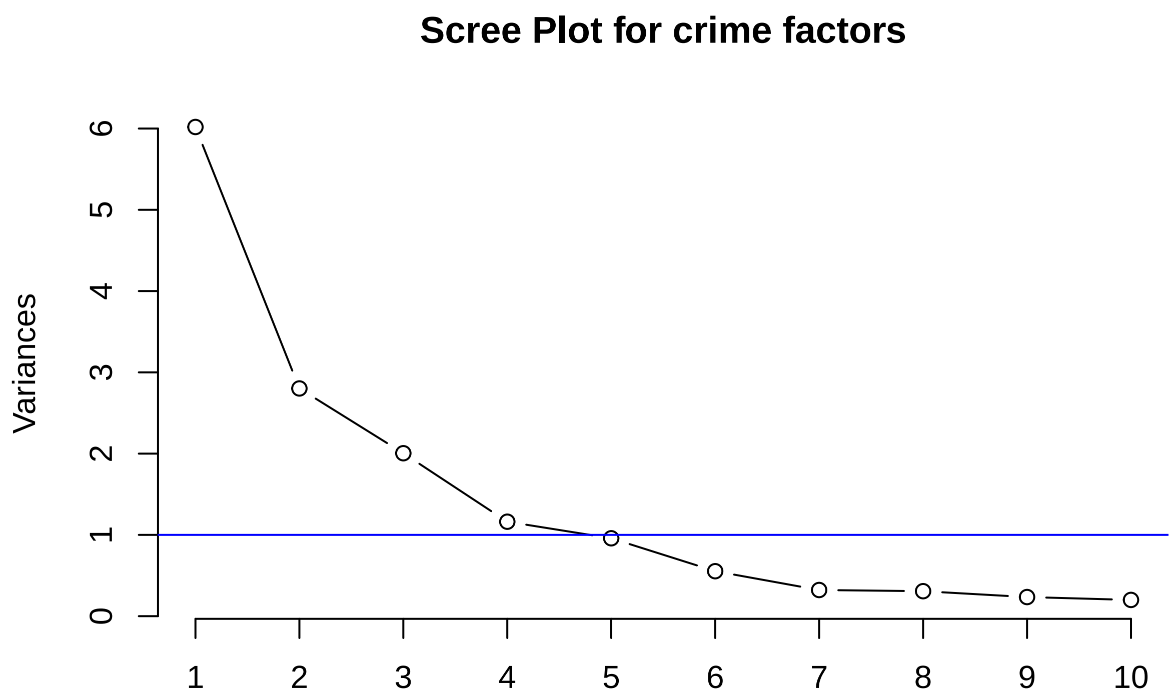
```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 213.8 on 37 degrees of freedom
## Multiple R-squared:  0.7543, Adjusted R-squared:  0.6945
## F-statistic: 12.62 on 9 and 37 DF,  p-value: 7.275e-09
```

```
pca <- prcomp(formula =  ~  So + Ed + Ineq + LF + M + M.F +Po1 + Pop + Prob  + Time  + Po2 + NW + U1 +U
pca
```

```
## Standard deviations (1, .., p=15):
##  [1] 2.45335539 1.67387187 1.41596057 1.07805742 0.97892746 0.74377006
##  [7] 0.56729065 0.55443780 0.48492813 0.44708045 0.41914843 0.35803646
## [13] 0.26332811 0.24180109 0.06792764
##
## Rotation (n x k) = (15 x 15):
##                PC1         PC2           PC3         PC4         PC5
## So     -0.33088129  0.15837219 -0.0155433104 -0.29247181  0.12061130
## Ed      0.33962148 -0.21461152 -0.0677396249 -0.07974375  0.02442839
## Ineq   -0.36579778  0.02752240  0.0002944563  0.08066612  0.21672823
## LF      0.17617757 -0.31943042 -0.2715301768  0.14326529  0.39407588
## M      -0.30371194 -0.06280357 -0.1724199946  0.02035537  0.35832737
## M.F     0.11638221 -0.39434428  0.2031621598 -0.01048029  0.57877443
## Po1     0.30863412  0.26981761 -0.0506458161 -0.33325059  0.23527680
## Pop     0.11307836  0.46723456 -0.0770210971  0.03210513  0.08317034
## Prob   -0.25888661 -0.15831708  0.1176726436 -0.49303389 -0.16562829
## Time   -0.02062867  0.38014836 -0.2235664632  0.54059002  0.14764767
## Po2     0.31099285  0.26396300 -0.0530651173 -0.35192809  0.20473383
## NW     -0.29358647  0.22801119 -0.0788156621 -0.23925971  0.36079387
## U1      0.04050137 -0.00807439  0.6590290980  0.18279096  0.13136873
## U2      0.01812228  0.27971336  0.5785006293  0.06889312  0.13499487
## Wealth  0.37970331  0.07718862 -0.0100647664 -0.11781752 -0.01167683
##                 PC6         PC7         PC8         PC9        PC10        PC11
## So     -0.100500743  0.19649727 -0.22734157  0.65599872 -0.06141452 -0.23397868
## Ed     -0.008571367 -0.23943629  0.14644678  0.44326978 -0.51887452  0.11821954
## Ineq    0.272027031  0.37483032 -0.07184018  0.02494384  0.01390576  0.18278697
## LF      0.504234275 -0.15931612 -0.25513777 -0.14393498 -0.03077073 -0.38532827
## M      -0.449132706 -0.15707378  0.55367691 -0.15474793  0.01443093 -0.39446657
## M.F    -0.074501901  0.15548197  0.05507254  0.24378252  0.35323357  0.28029732
## Po1    -0.095776709  0.08011735 -0.04613156 -0.19425472  0.14320978  0.13042001
## Pop     0.547098563  0.09046187  0.59078221  0.20244830  0.03970718 -0.05849643
## Prob    0.283535996 -0.56159383  0.08598908  0.05306898  0.42530006  0.08978385
## Time   -0.148203050 -0.44199877 -0.19507812  0.23551363  0.29264326  0.26363121
## Po2    -0.119524780  0.09518288 -0.03168720 -0.19512072  0.05929780  0.13885912
## NW      0.051219538 -0.31154195 -0.20432828 -0.18984178 -0.49201966  0.20695666
## U1      0.017385981 -0.17354115  0.20206312 -0.02069349 -0.22765278  0.17857891
## U2      0.048155286 -0.07526787 -0.24369650 -0.05576010  0.04750100 -0.47021842
## Wealth -0.154683104 -0.14859424 -0.08630649  0.23196695  0.11219383 -0.31955631
##               PC12        PC13        PC14          PC15
## So      0.05753357 -0.29368483  0.29364512  0.0084369230
## Ed     -0.47786536  0.19441949 -0.03964277 -0.0280052040
## Ineq   -0.43762828 -0.12181090 -0.59279037  0.0177570357
## LF     -0.02705134 -0.27742957  0.15385625  0.0336823193
## M      -0.16580189 -0.05142365 -0.04901705  0.0051398012
## M.F     0.23925913  0.31624667  0.04125321  0.0097922075
```

```
## Po1    -0.22611207 -0.18592255  0.09490151 -0.6894155129
## Pop     0.18350385  0.12651689  0.05326383  0.0001496323
## Prob   -0.15567100 -0.03547596 -0.04761011  0.0293376260
## Time   -0.13536989 -0.05738113  0.04488401  0.0376754405
## Po2    -0.19088461 -0.13454940  0.08259642  0.7200270100
## NW      0.36671707  0.22901695 -0.13227774 -0.0370783671
## U1      0.09314897 -0.59039450  0.02335942  0.0111359325
## U2     -0.28440496  0.43292853  0.03985736  0.0073618948
## Wealth  0.32172821 -0.14077972 -0.70031840 -0.0025685109
```

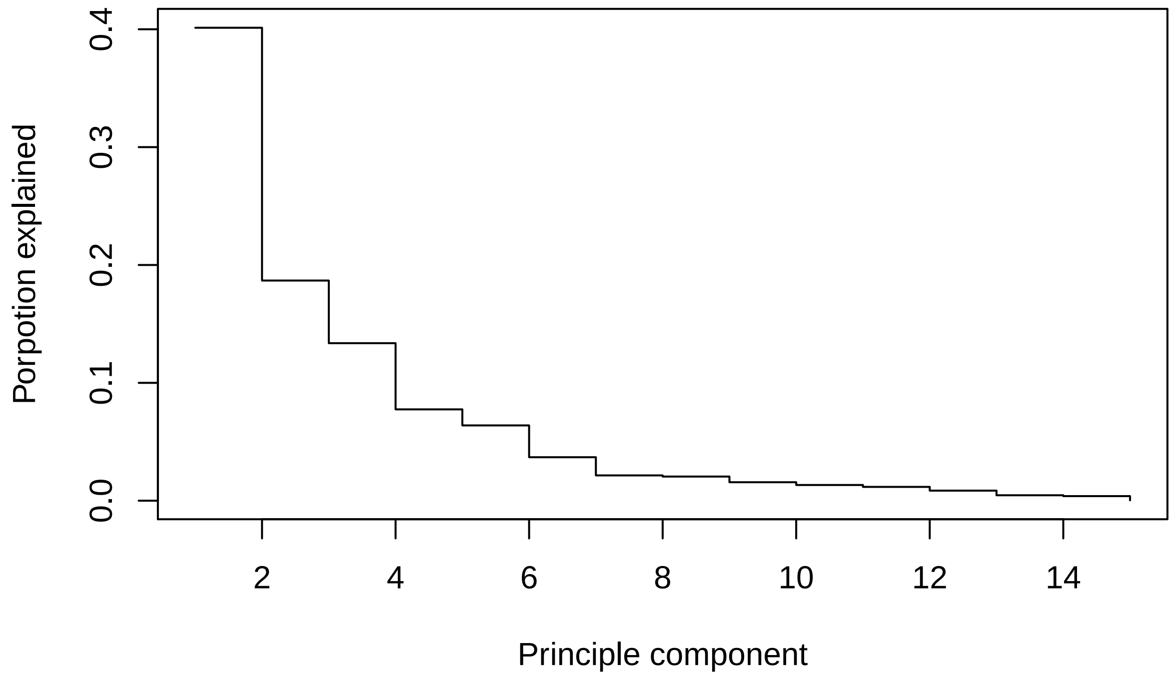
```
#Plotting Scree plot
# Kaiser eigenvalue-greater-than-one rule
Scree <- plot(pca,
              type="line",
              main="Scree Plot for crime factors")%>%
              abline(h=1, col="blue")
```



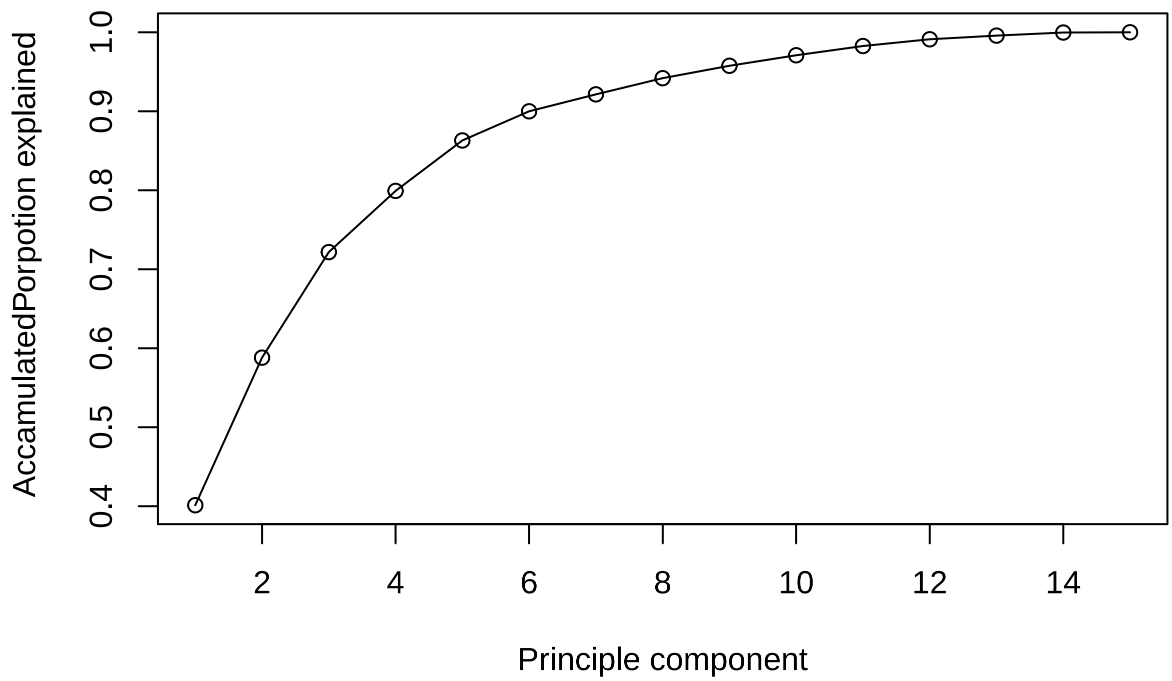

```
# Calculate  and Plot the variances and proportion of variances

var <- pca$sdev^2
propvar <- var/sum(var)
```

```
#plotting
por.explained <- plot(propvar, xlab = "Principle component" , ylab = "Porpotion explained" , type = "s")
```




```
acc.por.explained <- plot(cumsum(propvar) , xlab = "Principle component" , ylab = " AccamulatedPorpotion
```




```r
#choose 4 new variables(variance > 1)
pca.chosen <- pca$x[, 1:4]
pca.chosen
```

```
##           PC1         PC2         PC3         PC4
## 1  -4.1992835  1.09383120  1.11907395 -0.67178115
## 2   1.1726630 -0.67701360  0.05244634  0.08350709
## 3  -4.1737248 -0.27677501  0.37107658 -0.37793995
## 4   3.8349617  2.57690596 -0.22793998 -0.38262331
## 5   1.8392999 -1.33098564 -1.27882805 -0.71814305
## 6   2.9072336  0.33054213 -0.53288181 -1.22140635
## 7   0.2457752  0.07362562  0.90742064 -1.13685873
## 8  -0.1301330  1.35985577 -0.59753132 -1.44045387
## 9  -3.6103169  0.68621008 -1.28372246 -0.55171150
## 10  1.1672376 -3.03207033 -0.37984502  0.28887026
## 11  2.5384879  2.66771358 -1.54424656  0.87671210
## 12  1.0065920  0.06044849 -1.18861346  1.31261964
## 13  0.5161143 -0.97485189 -1.83351610  1.59117618
## 14  0.4265556 -1.85044812 -1.02893477  0.07789173
## 15 -3.3435299 -0.05182823  1.01358113 -0.08840211
## 16 -3.0310689  2.10295524  1.82993161 -0.52347187
## 17 -0.2262961 -1.44939774  1.37565975 -0.28960865
## 18 -0.1127499  0.39407030  0.38836278 -3.97985093
## 19  2.9195668  1.58646124 -0.97612613 -0.78629766
## 20  2.2998485  1.73396487  2.82423222  0.23281758
## 21  1.1501667 -0.13531015 -0.28506743  2.19770548
```

```
## 22 -5.6594827  1.09730404 -0.10043541  0.05245484
## 23 -0.1011749  0.57911362 -0.71128354  0.44394773
## 24  1.3836281 -1.95052341  2.98485490  0.35942784
## 25  0.2727756 -2.63013778 -1.83189535 -0.05207518
## 26  4.0565577 -1.17534729  0.81690756 -1.66990720
## 27  0.8929694 -0.79236692 -1.26822542  0.57575615
## 28  0.1514495 -1.44873320 -0.10857670  0.51040146
## 29  3.5592481  4.76202163 -0.75080576 -0.64692974
## 30 -4.1184576  0.38073981 -1.43463965 -0.63330834
## 31 -0.6811731 -1.66926027  2.88645794  1.30977099
## 32  1.7157269  1.30836339  0.55971313  0.70557980
## 33 -1.8860627 -0.59058174 -1.43570145 -0.18239089
## 34  1.9526349 -0.52395429  0.75642216 -0.44289927
## 35  1.5888864  3.12998571  1.73107199  1.68604766
## 36  1.0709414  1.65628271 -0.79436888  1.85172698
## 37 -4.1101715 -0.15766712 -2.36296974  0.56868399
## 38 -0.7254706 -2.89263339  0.36348376  0.50612576
## 39 -3.3451254  0.95045293 -0.19551398  0.27716645
## 40 -1.0644466  1.05265304 -0.82886286  0.12042931
## 41  1.4933989 -1.86712106 -1.81853582  1.06112429
## 42 -0.6789284 -1.83156328  1.65435992 -0.95121379
## 43 -2.4164258  0.46701087 -1.42808323 -0.41149015
## 44  2.2978729 -0.41865689  0.64422929  0.63462770
## 45 -2.9245282  1.19488555  3.35139309  1.48966984
## 46  1.7654525 -0.95655926 -0.98576138 -1.05683769
## 47  2.3125056 -2.56161119  1.58223354 -0.59863946
```

```r
#Combining PCAs  with crime
new.crime <- cbind(pca.chosen, crime[,16]) %>%
  data.frame()
colnames(new.crime)[5] <- "Crime"
new.model <- lm ( Crime ~ PC1 + PC2 + PC3 + PC4 ,data = new.crime )
summary(new.model)
```

```
##
## Call:
## lm(formula = Crime ~ PC1 + PC2 + PC3 + PC4, data = new.crime)
##
## 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
#Transforming new data back to original variable
PCAs <- new.model$coefficients[2:5]
intercept <- new.model$coefficients[1]
original <- pca$rotation[,1:4] %*% PCAs
original
```

```
##              [,1]
## So      10.223091
## Ed      14.352610
## Ineq   -27.536348
## LF     -14.005349
## M      -21.277963
## M.F    -24.437572
## Po1     63.456426
## Pop     39.830667
## Prob     3.295707
## Time    -6.612616
## Po2     64.557974
## NW      15.434545
## U1     -27.222281
## U2       1.425902
## Wealth  38.607855
```

```r
PCAs
```

```
##       PC1       PC2       PC3       PC4
##  65.21593  70.08312 -25.19408 -69.44603
```

These are the variables used, expressed in original form.

```r
# un-scaling data

origi.var <- original/sapply(crime[,1:15],sd)
origi.inter <-  intercept - sum(original*sapply(crime[,1:15],mean)/sapply(crime[,1:15],sd))

origi.var
```

```
##                  [,1]
## So         8.13445978
## Ed        29.96524916
## Ineq     -24.61459872
## LF        -4.71259516
## M         -7.60978529
## M.F     -604.71355665
## Po1       21.53447549
## Pop        1.04621551
## Prob       0.32050426
## Time    -366.78104113
## Po2       76.44113075
## NW         0.01599585
## U1        -6.82330056
## U2        62.71293220
## Wealth     5.44778133
```

```r
#Trying with 5 PCAs
#choose 4 new variables(variance > 1)
pca.chosen2 <- pca$x[, 1:5]
pca.chosen2
```

```
##            PC1         PC2         PC3         PC4          PC5
## 1  -4.1992835  1.09383120  1.11907395 -0.67178115 -0.055283376
## 2   1.1726630 -0.67701360  0.05244634  0.08350709  1.173199821
## 3  -4.1737248 -0.27677501  0.37107658 -0.37793995 -0.541345246
## 4   3.8349617  2.57690596 -0.22793998 -0.38262331  1.644746496
## 5   1.8392999 -1.33098564 -1.27882805 -0.71814305 -0.041590320
## 6   2.9072336  0.33054213 -0.53288181 -1.22140635 -1.374360960
## 7   0.2457752  0.07362562  0.90742064 -1.13685873 -0.718644387
## 8  -0.1301330  1.35985577 -0.59753132 -1.44045387  0.222781388
## 9  -3.6103169  0.68621008 -1.28372246 -0.55171150  0.324292990
## 10  1.1672376 -3.03207033 -0.37984502  0.28887026  0.646056610
## 11  2.5384879  2.66771358 -1.54424656  0.87671210  0.324083561
## 12  1.0065920  0.06044849 -1.18861346  1.31261964 -0.358087724
## 13  0.5161143 -0.97485189 -1.83351610  1.59117618 -0.599881946
## 14  0.4265556 -1.85044812 -1.02893477  0.07789173 -0.741887592
## 15 -3.3435299 -0.05182823  1.01358113 -0.08840211 -0.002969448
## 16 -3.0310689  2.10295524  1.82993161 -0.52347187  0.387454246
## 17 -0.2262961 -1.44939774  1.37565975 -0.28960865 -1.337784608
## 18 -0.1127499  0.39407030  0.38836278 -3.97985093 -0.410914404
## 19  2.9195668  1.58646124 -0.97612613 -0.78629766 -1.356288600
## 20  2.2998485  1.73396487  2.82423222  0.23281758  0.653038858
## 21  1.1501667 -0.13531015 -0.28506743  2.19770548 -0.084621572
## 22 -5.6594827  1.09730404 -0.10043541  0.05245484  0.689327990
## 23 -0.1011749  0.57911362 -0.71128354  0.44394773 -0.689939865
## 24  1.3836281 -1.95052341  2.98485490  0.35942784  0.744371276
## 25  0.2727756 -2.63013778 -1.83189535 -0.05207518 -0.803692524
## 26  4.0565577 -1.17534729  0.81690756 -1.66990720  2.895110075
## 27  0.8929694 -0.79236692 -1.26822542  0.57575615 -1.830793964
## 28  0.1514495 -1.44873320 -0.10857670  0.51040146  1.023229895
## 29  3.5592481  4.76202163 -0.75080576 -0.64692974 -0.309946510
## 30 -4.1184576  0.38073981 -1.43463965 -0.63330834  0.254715638
## 31 -0.6811731 -1.66926027  2.88645794  1.30977099  0.470913997
## 32  1.7157269  1.30836339  0.55971313  0.70557980 -0.331277622
## 33 -1.8860627 -0.59058174 -1.43570145 -0.18239089 -0.291863659
## 34  1.9526349 -0.52395429  0.75642216 -0.44289927 -0.723474420
## 35  1.5888864  3.12998571  1.73107199  1.68604766 -0.665406182
## 36  1.0709414  1.65628271 -0.79436888  1.85172698 -0.020031154
## 37 -4.1101715 -0.15766712 -2.36296974  0.56868399  2.469679496
## 38 -0.7254706 -2.89263339  0.36348376  0.50612576 -0.028157162
## 39 -3.3451254  0.95045293 -0.19551398  0.27716645 -0.487259213
## 40 -1.0644466  1.05265304 -0.82886286  0.12042931  0.645884788
## 41  1.4933989 -1.86712106 -1.81853582  1.06112429 -0.009855774
## 42 -0.6789284 -1.83156328  1.65435992 -0.95121379 -2.115630145
## 43 -2.4164258  0.46701087 -1.42808323 -0.41149015  0.867397522
## 44  2.2978729 -0.41865689  0.64422929  0.63462770  0.703116983
## 45 -2.9245282  1.19488555  3.35139309  1.48966984 -0.806659622
## 46  1.7654525 -0.95655926 -0.98576138 -1.05683769 -0.542466034
## 47  2.3125056 -2.56161119  1.58223354 -0.59863946  1.140712406
```

```r
#Combining PCAs  with crime
new.crime2 <- cbind(pca.chosen2, crime[,16]) %>%
  data.frame()
colnames(new.crime2)[6] <- "Crime"
new.model2 <- lm ( Crime ~ PC1 + PC2 + PC3 + PC4 + PC5 ,data = new.crime2 )
summary(new.model2)
```

```
##
## Call:
## lm(formula = Crime ~ PC1 + PC2 + PC3 + PC4 + PC5, data = new.crime2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -420.79 -185.01   12.21  146.24  447.86
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   905.09      35.59  25.428  < 2e-16 ***
## PC1            65.22      14.67   4.447 6.51e-05 ***
## PC2            70.08      21.49   3.261  0.00224 **
## PC3           -25.19      25.41  -0.992  0.32725
## PC4           -69.45      33.37  -2.081  0.04374 *
## PC5           229.04      36.75   6.232 2.02e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 244 on 41 degrees of freedom
## Multiple R-squared:  0.6452, Adjusted R-squared:  0.6019
## F-statistic: 14.91 on 5 and 41 DF,  p-value: 2.446e-08
```

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.

```r
#Question 10.1
library(rpart)
library(rpart.plot)
library(randomForest)
library(caret)
library(modelr)
```

```r
#CART
# set up train and testing split
train <- createDataPartition(crime$Crime, p = .85, list = F)
# set up test and train datasets
crime.train <- crime[train,]
crime.test <- crime[-train,]
# check splits
dim(crime.train); dim(crime.test)
```

```
## [1] 43 16
```

```
## [1]  4 16
```

```r
#Regression Tree
crime.tree <- train(
  Crime ~ .,
  data = crime.train,
  method = 'rpart',
  trControl = trainControl(method = 'boot_all', number = 10),
  metric = 'RMSE'
)
```

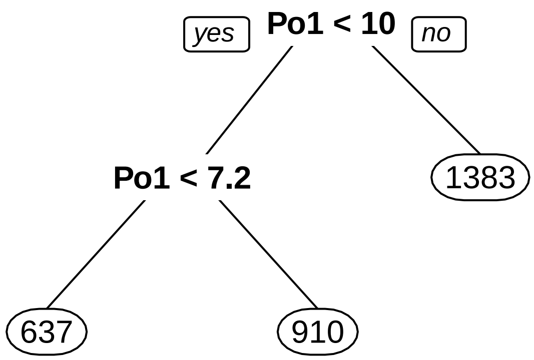
```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo, :
## There were missing values in the apparent performance measures.
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo, :
## There were missing values in resampled performance measures.
```

```r
crime.tree$finalModel
```

```
## n= 43
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
## 1) root 43 6601051.0  910.2791
##   2) Po1< 10 32 1534630.0  747.8438
##     4) Po1< 7.15 19  654085.2  637.2105 *
##     5) Po1>=7.15 13  308103.2  909.5385 *
##   3) Po1>=10 11 1765868.0 1382.8180 *
```

```r
prp(crime.tree$finalModel)
```




```
#Random Forest
crime.forest <- train(
  Crime ~ .,
  data = crime.train,
  method = 'rf',
  trControl = trainControl(method = 'boot_all', number = 10),
  metric = 'RMSE')
crime.forest$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 8
##
##           Mean of squared residuals: 82700.7
##                     % Var explained: 46.13
```

```
#Testing
crime.res1 <- crime.test %>%
  add_predictions(., crime.tree) %>%
  select('observations' = Crime, pred) %>%
  as.data.frame()
crime.res2 <- crime.test %>%
  add_predictions(., crime.forest) %>%
  select('observations' = Crime, pred) %>%
```

```
  as.data.frame()
```

```
crime.res1
```

```
##    observations      pred
## 19          750 1382.8182
## 28         1216  909.5385
## 34          923  909.5385
## 46          508 1382.8182
```

```
crime.res2
```

```
##    observations      pred
## 19          750 1245.2786
## 28         1216  993.0923
## 34          923 1021.4021
## 46          508 1131.4758
```

```
postResample(obs = crime.res1$observations, pred = crime.res1$pred)
```

```
##        RMSE    Rsquared         MAE
## 561.2186717   0.7287986 456.8898601
```

```
postResample(obs = crime.res2$observations, pred = crime.res2$pred)
```

```
##        RMSE    Rsquared         MAE
## 416.3513502   0.4659483 360.0160583
```

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.

I would consider logisitc regession useful in predicting customer behavior on EC sites. The results would be buy (0) and don't buy(1). As for predictors, age, occupation, time spent on site would be considered good predictors.

Question 10.3

Using the GermanCredit data set germancredit.txt , use logistic regression to find a good predictive model for whether credit applicants are good credit risks or not.

```
set.seed(101)
credit <- read.table("germancredit.txt", header = FALSE)
str(credit)
```

```
## 'data.frame':    1000 obs. of  21 variables:
##  $ V1 : chr  "A11" "A12" "A14" "A11" ...
##  $ V2 : int  6 48 12 42 24 36 24 36 12 30 ...
##  $ V3 : chr  "A34" "A32" "A34" "A32" ...
##  $ V4 : chr  "A43" "A43" "A46" "A42" ...
##  $ V5 : int  1169 5951 2096 7882 4870 9055 2835 6948 3059 5234 ...
##  $ V6 : chr  "A65" "A61" "A61" "A61" ...
##  $ V7 : chr  "A75" "A73" "A74" "A74" ...
##  $ V8 : int  4 2 2 2 3 2 3 2 2 4 ...
##  $ V9 : chr  "A93" "A92" "A93" "A93" ...
##  $ V10: chr  "A101" "A101" "A101" "A103" ...
```

```
##  $ V11: int  4 2 3 4 4 4 4 2 4 2 ...
##  $ V12: chr  "A121" "A121" "A121" "A122" ...
##  $ V13: int  67 22 49 45 53 35 53 35 61 28 ...
##  $ V14: chr  "A143" "A143" "A143" "A143" ...
##  $ V15: chr  "A152" "A152" "A152" "A153" ...
##  $ V16: int  2 1 1 1 2 1 1 1 1 2 ...
##  $ V17: chr  "A173" "A173" "A172" "A173" ...
##  $ V18: int  1 1 2 2 2 2 1 1 1 1 ...
##  $ V19: chr  "A192" "A191" "A191" "A191" ...
##  $ V20: chr  "A201" "A201" "A201" "A201" ...
##  $ V21: int  1 2 1 1 2 1 1 1 1 2 ...
```

```
credit$V21[credit$V21==1] <- 0
credit$V21[credit$V21==2] <- 1
```

```
#Dividing data
credit.part <- createDataPartition(credit$V21, times = 1, p = 0.7, list=FALSE)
head(credit.part)
```

```
##      Resample1
## [1,]         1
## [2,]         2
## [3,]         3
## [4,]         4
## [5,]         5
## [6,]         6
```

```
credit.train <- credit[credit.part,]
credit.valid <- credit[-credit.part,]
```

```
#model
credit.log <- glm(V21 ~ ., data = credit.train, family=binomial(link="logit"))
summary(credit.log)
```

```
##
## Call:
## glm(formula = V21 ~ ., family = binomial(link = "logit"), data = credit.train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.6141  -0.6484  -0.3327   0.5806   2.5401
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -3.569e-01  1.342e+00  -0.266 0.790318
## V1A12       -2.958e-01  2.809e-01  -1.053 0.292252
## V1A13       -5.582e-01  4.807e-01  -1.161 0.245532
## V1A14       -1.507e+00  2.820e-01  -5.346    9e-08 ***
## V2           4.587e-02  1.215e-02   3.774 0.000160 ***
## V3A31        8.164e-02  6.777e-01   0.120 0.904113
## V3A32       -3.752e-01  5.371e-01  -0.699 0.484776
## V3A33       -1.110e+00  5.813e-01  -1.909 0.056260 .
## V3A34       -1.857e+00  5.579e-01  -3.329 0.000872 ***
## V4A41       -1.848e+00  4.817e-01  -3.836 0.000125 ***
## V4A410      -2.417e+00  9.679e-01  -2.497 0.012511 *
## V4A42       -9.125e-01  3.323e-01  -2.746 0.006029 **
```

```
## V4A43       -1.039e+00  3.116e-01  -3.334 0.000855 ***
## V4A44       -8.163e-01  9.009e-01  -0.906 0.364883
## V4A45       -3.390e-01  6.401e-01  -0.530 0.596388
## V4A46        1.099e-01  4.737e-01   0.232 0.816452
## V4A48       -1.674e+00  1.210e+00  -1.384 0.166426
## V4A49       -1.163e+00  4.208e-01  -2.764 0.005718 **
## V5           1.454e-04  5.535e-05   2.628 0.008594 **
## V6A62       -1.776e-01  3.428e-01  -0.518 0.604336
## V6A63        4.524e-01  4.577e-01   0.988 0.322956
## V6A64       -9.605e-01  6.456e-01  -1.488 0.136846
## V6A65       -8.524e-01  3.235e-01  -2.635 0.008419 **
## V7A72       -4.012e-01  5.588e-01  -0.718 0.472762
## V7A73       -6.514e-01  5.339e-01  -1.220 0.222457
## V7A74       -1.503e+00  5.815e-01  -2.585 0.009750 **
## V7A75       -8.686e-01  5.445e-01  -1.595 0.110686
## V8           3.831e-01  1.136e-01   3.372 0.000747 ***
## V9A92       -1.445e-01  5.564e-01  -0.260 0.795099
## V9A93       -6.374e-01  5.467e-01  -1.166 0.243694
## V9A94       -4.908e-01  6.419e-01  -0.765 0.444459
## V10A102      3.868e-01  4.966e-01   0.779 0.436073
## V10A103     -5.953e-01  4.893e-01  -1.217 0.223718
## V11         -2.731e-02  1.083e-01  -0.252 0.800898
## V12A122      5.001e-01  3.131e-01   1.597 0.110233
## V12A123      2.743e-01  2.899e-01   0.946 0.344022
## V12A124      6.561e-01  5.220e-01   1.257 0.208747
## V13         -1.923e-02  1.155e-02  -1.665 0.095899 .
## V14A142     -4.130e-01  5.260e-01  -0.785 0.432343
## V14A143     -8.298e-01  2.938e-01  -2.824 0.004744 **
## V15A152     -3.803e-01  2.869e-01  -1.326 0.184883
## V15A153     -6.739e-01  5.884e-01  -1.145 0.252124
## V16          7.446e-01  2.530e-01   2.944 0.003245 **
## V17A172      7.947e-01  8.275e-01   0.960 0.336845
## V17A173      5.865e-01  7.953e-01   0.737 0.460850
## V17A174      3.465e-01  8.183e-01   0.423 0.671940
## V18          1.495e-01  3.242e-01   0.461 0.644658
## V19A192     -3.794e-02  2.494e-01  -0.152 0.879062
## V20A202     -1.223e+00  9.110e-01  -1.343 0.179377
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 848.32  on 699  degrees of freedom
## Residual deviance: 592.81  on 651  degrees of freedom
## AIC: 690.81
##
## Number of Fisher Scoring iterations: 5
```

```r
#Confusion Matrix
creditPredict <- predict(credit.log, newdata=credit.valid[,-21], type="response")
Confusion.mat <- table(credit.valid$V21, round(creditPredict))
```

We can see that although sensitivity is quite high, Specifity isn't

```
Sensitivity <- Confusion.mat[1,1]/sum(Confusion.mat[1,])
Sensitivity
```

```
## [1] 0.8786408
```

```
Specfitity <-Confusion.mat[2,2]/sum(Confusion.mat[2,])
Specfitity
```

```
## [1] 0.4787234
```

Then we change the threshhold

```
#setting second thresh hold
threshold <- 0.7
thres <- as.matrix(table(round(creditPredict > threshold), credit.valid$V21))
names(dimnames(thres)) <- c("Predicted", "Observed")
thres
```

```
##          Observed
## Predicted   0   1
##         0 197  76
##         1   9  18
```

Below are the results for a different threshold, there is an obvious rise in specifity but also a slight loss in sensitivity.

```
Sensitivity2 <- thres[1,1]/sum(thres[1,])
Sensitivity2
```

```
## [1] 0.7216117
```

```
Specfitity2 <-thres[2,2]/sum(thres[2,])
Specfitity2
```

```
## [1] 0.6666667
```