ISYE 6501 - Homework 6

Justin Lewis

February 14, 2019

The first section is hidden to avoid packge imports giving long output, but begin by importing all packages and clearing the environment to begin fresh using rm(list=ls()).

```
## Warning: package 'kernlab' was built under R version 3.5.2
```

```
## Warning: package 'ggplot2' was built under R version 3.5.2
```

```
##
## Attaching package: 'ggplot2'
```

```
## The following object is masked from 'package:kernlab':
##
##     alpha
```

```
## Warning: package 'factoextra' was built under R version 3.5.2
```

```
## Welcome! Related Books: `Practical Guide To Cluster Analysis in R` at https://goo.gl/13EFCZ
```

```
## Warning: package 'GGally' was built under R version 3.5.2
```

Question 9.1

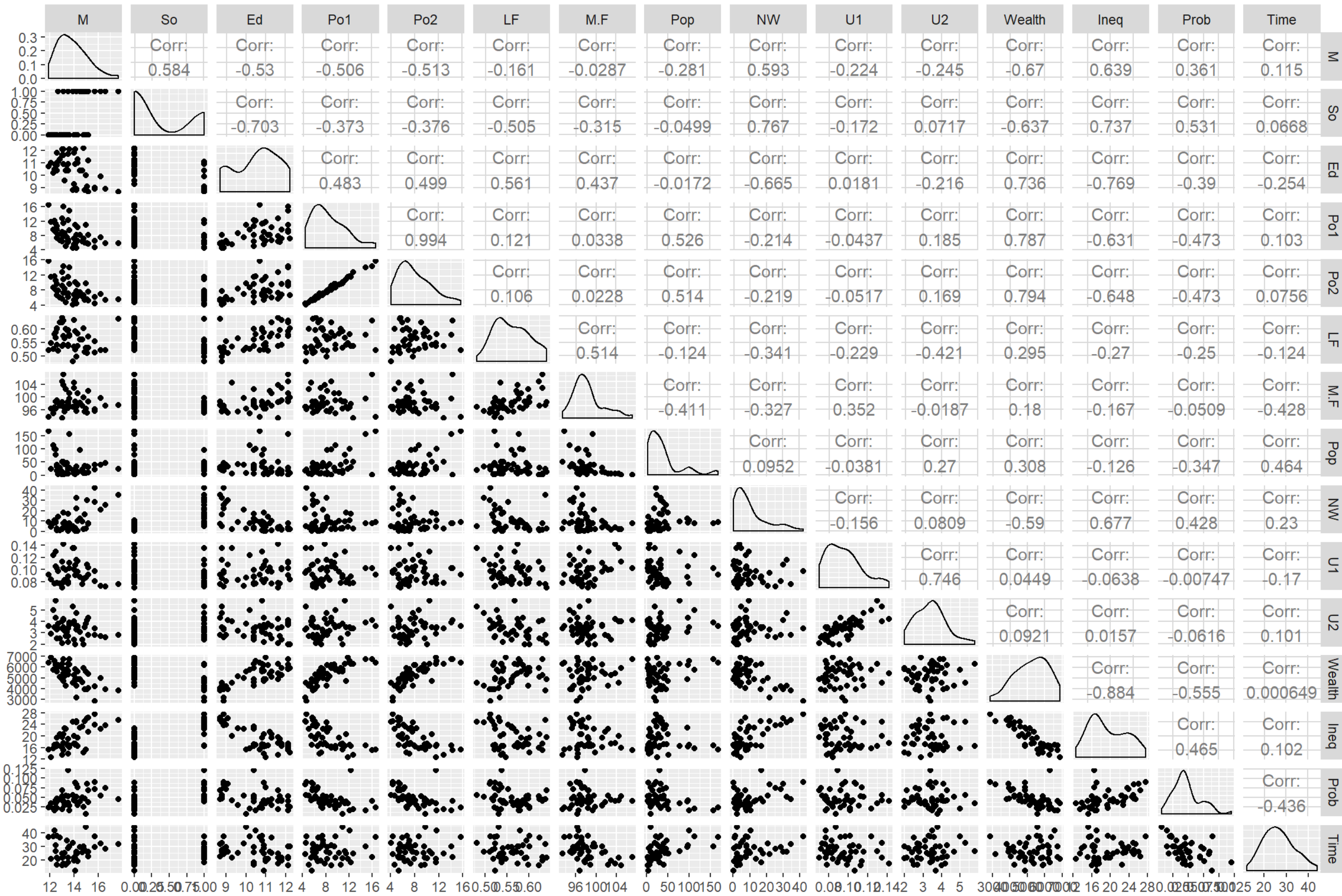
Using the same crime data set uscrime.txt as in Question 8.2, apply Principal Component Analysis and then create a regression model using the first few principal components. Specify your new model in terms of the original variables (not the principal components), and compare its quality to that of your solution to Question 8.2.

Need to begin by importing the data:

```
data <- as.data.frame(read.csv('uscrime.txt', header=TRUE, sep='\t'))
atts <- data[1:ncol(data)-1]
resp <- data[ncol(data)]
```

If we are going to use PCA, it should be to reduce the number of features used, or to address multi-collinearity in the data. Let’s see if this is present in our data.

```
ggpairs(atts, columns=1:ncol(atts))
```



From this we can see there are certainly some correlations within the data. Particularly PO1 and PO2, as well as Wealth with each of the previously mentioned. PCA could be used well here to remove this collinearity.

Now apply PCA, making sure to scale as well:

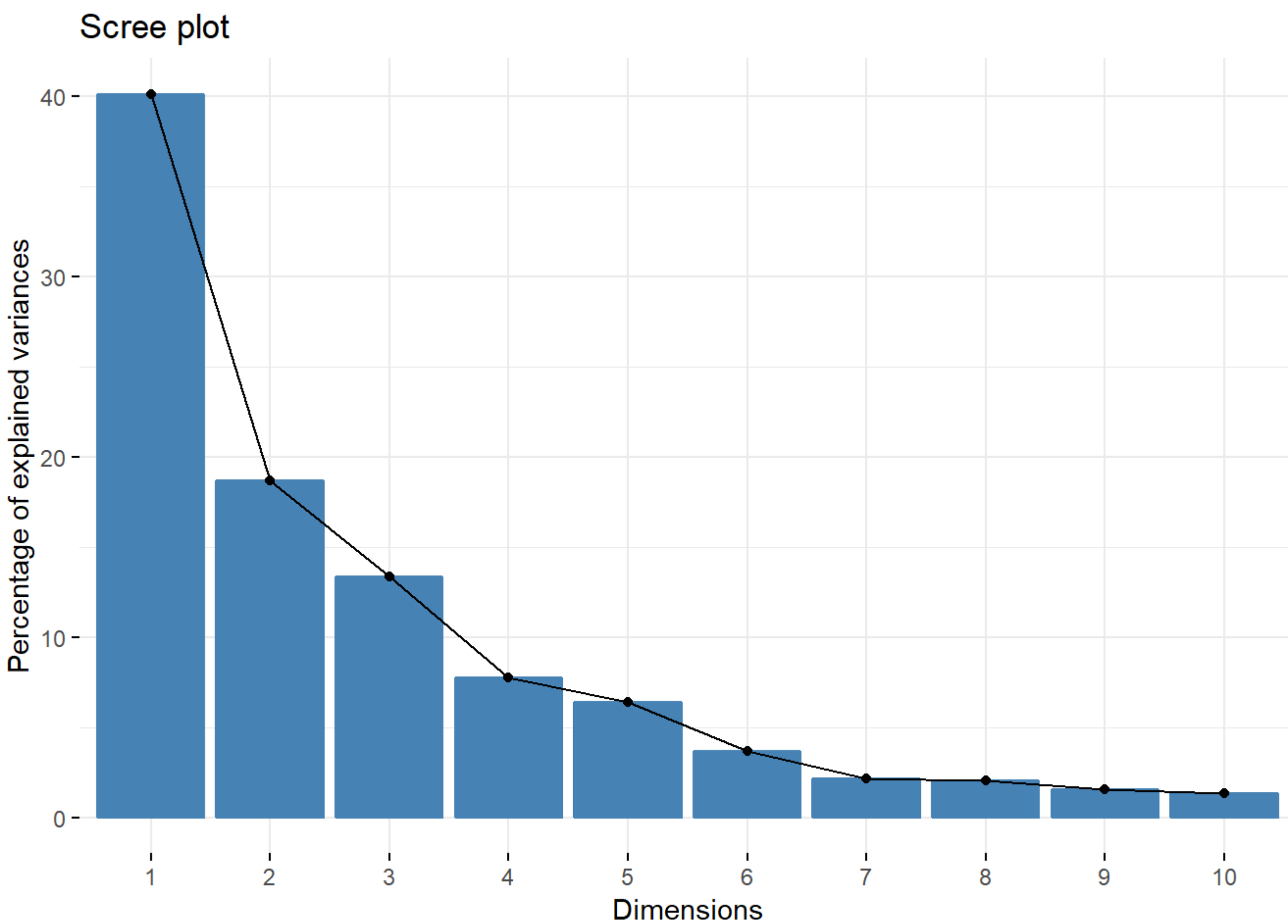
```
#Perform the PCA
pca_atts <- prcomp(~., atts, scale=TRUE)

#Check the summary
summary(pca_atts)
```

```
## 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
```

```
#Grab the coefficients
pca_evectors <- as.data.frame(pca_atts[2])

#Visualize how much variance is explained in each PC
fviz_eig(pca_atts)
```



Based on both the Scree plot and the summary, I am going to base my model on the first 5 principal components. This is because from the summary we can see anything after this is accounting for less than 5% of the variance seen in the data. Additionally, the scree plot shows the same reduction in amount of explained variance after the 5th principal component.

```
#Grab the principal component values from the pca_atts
pcavalues <- as.data.frame(pca_atts[['x']])

#Build the model, using the crime column as the response, and the top 5 PCs as mentioned earlier
pca_model <- lm(data$Crime ~ ., pcavalues[1:5])

#Check the summary from our model
summary(pca_model)
```

```
##
## Call:
## lm(formula = data$Crime ~ ., data = pcavalues[1:5])
##
## 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
```

Now grab the coefficents from each PC, and use them to work backwards and get our implied coefficients for the original factors in the data

```
#Grab all the coefficients
mcoefs <- as.data.frame(summary(pca_model)$coefficients)

#Skip the intercept coefficient, get only coefficients relevant to components
pccoefs <- mcoefs$Estimate[2:length(mcoefs$Estimate)]

#Make the transverse matrix of the first 5 predictor eivenvectors
#This is done so that we can multiply the two 'vectors' of numbers together to get the final coe
fficients. Transpose is used to get the variables as columns
pca_evectors_trans <- t(pca_evectors[1:5])
pca_evectors_trans
```

```
##                        M          So          Ed         Po1         Po2
## rotation.PC1 -0.30371194 -0.33088129  0.33962148  0.30863412  0.31099285
## rotation.PC2  0.06280357 -0.15837219  0.21461152 -0.26981761 -0.26396300
## rotation.PC3  0.17241999  0.01554331  0.06773962  0.05064582  0.05306512
## rotation.PC4 -0.02035537  0.29247181  0.07974375  0.33325059  0.35192809
## rotation.PC5 -0.35832737 -0.12061130 -0.02442839 -0.23527680 -0.20473383
##                      LF         M.F         Pop          NW          U1
## rotation.PC1  0.1761776  0.11638221  0.11307836 -0.29358647  0.04050137
## rotation.PC2  0.3194304  0.39434428 -0.46723456 -0.22801119  0.00807439
## rotation.PC3  0.2715302 -0.20316216  0.07702110  0.07881566 -0.65902910
## rotation.PC4 -0.1432653  0.01048029 -0.03210513  0.23925971 -0.18279096
## rotation.PC5 -0.3940759 -0.57877443 -0.08317034 -0.36079387 -0.13136873
##                       U2      Wealth          Ineq       Prob        Time
## rotation.PC1  0.01812228  0.37970331 -0.3657977826 -0.2588866 -0.02062867
## rotation.PC2 -0.27971336 -0.07718862 -0.0275223960  0.1583171 -0.38014836
## rotation.PC3 -0.57850063  0.01006477 -0.0002944563 -0.1176726  0.22356646
## rotation.PC4 -0.06889312  0.11781752 -0.0806661240  0.4930339 -0.54059002
## rotation.PC5 -0.13499487  0.01167683 -0.2167282285  0.1656283 -0.14764767
```

```
#Calculate the implied original coefficent in each of the components by multiplying
original_calc <- pca_evectors_trans*pccoefs
original_calc
```

```
##                      M          So          Ed       Po1       Po2
## rotation.PC1 -19.806857 -21.5787314  22.148731 20.127861 20.281688
## rotation.PC2  -4.401470  11.0992170 -15.040645 18.909660 18.499350
## rotation.PC3   4.343963   0.3915994   1.706637  1.275975  1.336927
## rotation.PC4  -1.413599  20.3110062   5.537887 23.142930 24.440009
## rotation.PC5  82.072312  27.6251516   5.595146 53.888461 46.892813
##                     LF         M.F       Pop         NW          U1
## rotation.PC1  11.489584   7.5899743  7.374511 -19.146515   2.6413344
## rotation.PC2 -22.386680 -27.6368772 32.745255  15.979735  -0.5658784
## rotation.PC3   6.840952  -5.1184833  1.940476   1.985688 -16.6036305
## rotation.PC4  -9.949206   0.7278145 -2.229574  16.615637 -12.6941068
## rotation.PC5  90.260251 132.5641295 19.049569  82.637245  30.0890646
##                     U2     Wealth          Ineq       Prob       Time
## rotation.PC1   1.181861 24.7627042 -23.855842638 -16.883531  -1.345318
## rotation.PC2  19.603185  5.4096192   1.928855344 -11.095354  26.641983
## rotation.PC3 -14.574790  0.2535725  -0.007418555  -2.964654   5.632551
## rotation.PC4  -4.784354  8.1819594  -5.601942128  34.239247 -37.541831
## rotation.PC5  30.919606 -2.6744930  49.640045057 -37.935972  33.817638
```

At this point we have all the coefficients for each original variable, separated into each of the prinicpal components we used. Now if we sum each column and unscale, we can find the final coefficients to put our model in terms of the original variables.

```
#Sum all the original coefficients together from each of the principal components
original_coefs <- as.data.frame(colSums(original_calc))
mu <- sapply(atts, mean)
sd <- sapply(atts, sd)
unscaled_original_coefs <- original_coefs/sd

#Make a table with all the implied coefficients with our original factors
kable(unscaled_original_coefs, col.names='Coefficient', caption='Implied Coefficients of Origina
l Factors')
```

Implied Coefficients of Original Factors

	Coefficient
M	48.3737430
So	79.0192180
Ed	17.8311962
Po1	39.4848384
Po2	39.8589169
LF	1886.9457724
M.F	36.6936631
Pop	1.5465826
NW	9.5373837
U1	159.0114753
U2	38.2993307
Wealth	0.0372401
Ineq	5.5403207
Prob	-1523.5214209
Time	3.8387787

Comparing these coefficients to the results of the final model in the previous homework assignment, the values all seem to be in a reasonable range. Additionally, the R-squared and adjusted R-squared values for the model appear to be reasonable. While they are lower than the model from the previous assignment, that is to be expected since we only used the first 5 principal components from the PCA method which accounted for ~80% of the variance in the data.

Now to finish getting our model in terms of the original coefficients, we need the intercept as well.

```
#Grab the estimate from the model using the first 5 principal components
pca_b <- mcoefs$Estimate[[1]]

#Subtract the total of all the other 'intercept' portions of the scaling results
orig_b <- pca_b - sum(original_coefs*mu/sd)

#Check the value
orig_b
```

```
## [1] -5933.837
```

Using the resulting intercept and our coefficients from above, we can attempt a prediction with the given new city data.

```
#Input data
input = as.vector(c(14.0, 0, 10.0, 12.0, 15.5, 0.640, 94.0, 150, 1.1, 0.120, 3.6, 3200, 20.1, 0.
04, 39.0))

#Now multiply each attibute by its relevant coefficient, and add our intercept
predicted_crime <- t(input) %*% unscaled_original_coefs[[1]]+orig_b
```

Our prediction is 1388.9256948 which is very comparable to my previous prediction from homework 5, of 1392.