HW6

Question 9.2

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)!)

All my steps are outlined in my report below.

```
install.packages("caret")
set.seed(3)
library(caret)

## Loading required package: ggplot2

## Loading required package: lattice
```

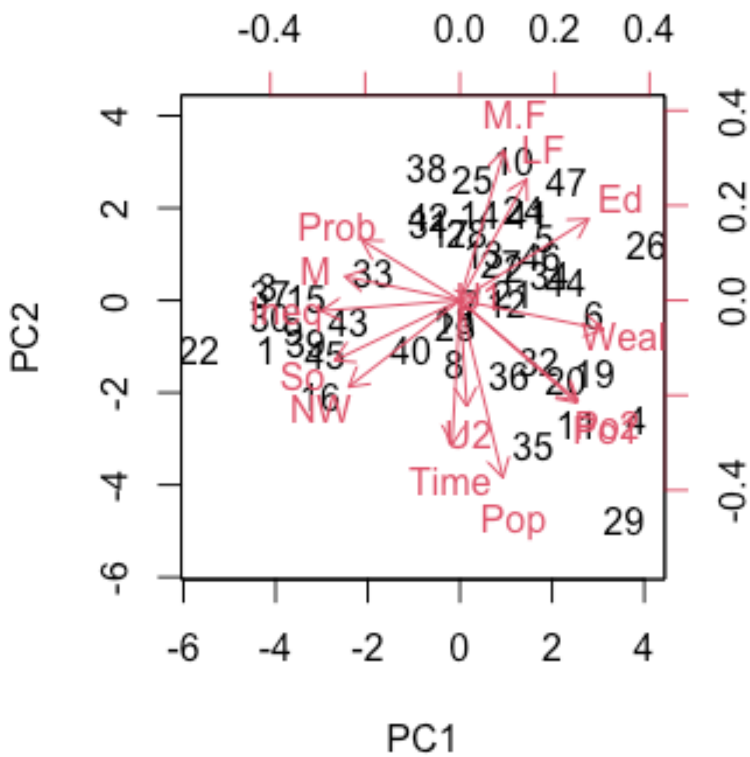
Step 1: load data

```
crime_data <- read.delim("uscrime.txt")
str(crime_data)

## 'data.frame':    47 obs. of  16 variables:
##  $ M     : num  15.1 14.3 14.2 13.6 14.1 12.1 12.7 13.1 15.7 14 ...
##  $ So    : int  1 0 1 0 0 0 1 1 1 0 ...
##  $ Ed    : num  9.1 11.3 8.9 12.1 12.1 11 11.1 10.9 9 11.8 ...
##  $ Po1   : num  5.8 10.3 4.5 14.9 10.9 11.8 8.2 11.5 6.5 7.1 ...
##  $ Po2   : num  5.6 9.5 4.4 14.1 10.1 11.5 7.9 10.9 6.2 6.8 ...
##  $ LF    : num  0.51 0.583 0.533 0.577 0.591 0.547 0.519 0.542 0.553 0.632
...
##  $ M.F   : num  95 101.2 96.9 99.4 98.5 ...
##  $ Pop   : int  33 13 18 157 18 25 4 50 39 7 ...
##  $ NW    : num  30.1 10.2 21.9 8 3 4.4 13.9 17.9 28.6 1.5 ...
##  $ U1    : num  0.108 0.096 0.094 0.102 0.091 0.084 0.097 0.079 0.081 0.1
...
##  $ U2    : num  4.1 3.6 3.3 3.9 2 2.9 3.8 3.5 2.8 2.4 ...
##  $ Wealth: int  3940 5570 3180 6730 5780 6890 6200 4720 4210 5260 ...
##  $ Ineq  : num  26.1 19.4 25 16.7 17.4 12.6 16.8 20.6 23.9 17.4 ...
##  $ Prob  : num  0.0846 0.0296 0.0834 0.0158 0.0414 ...
```

```
##  $ Time  : num  26.2 25.3 24.3 29.9 21.3 ...
##  $ Crime : int  791 1635 578 1969 1234 682 963 1555 856 705 ...
```

Step 2: Perform PCA and analyze

```
pca<-prcomp(crime_data[,-16],scale. =TRUE)
#Biplotting to get an idea of attributes contributing to construct PC1, PC2 e
tc..
biplot(pca,scale=0)
```



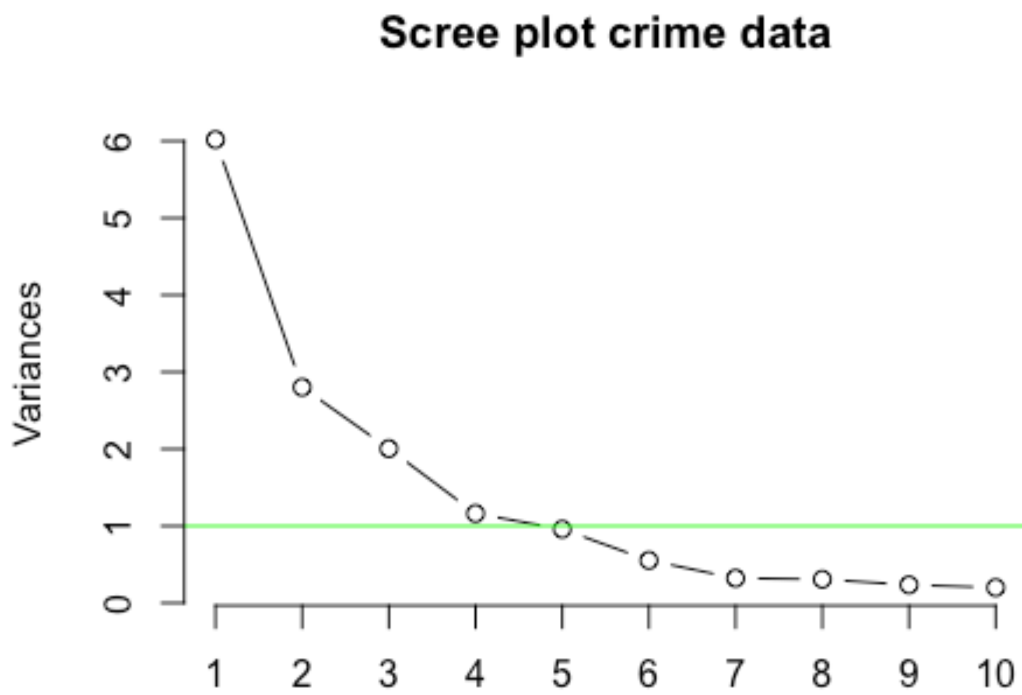

```
summary(pca)

## Importance of components:
##                           PC1    PC2    PC3     PC4     PC5     PC6     PC
7
## Standard deviation     2.4534 1.6739 1.4160 1.07806 0.97893 0.74377 0.5672
9
## Proportion of Variance 0.4013 0.1868 0.1337 0.07748 0.06389 0.03688 0.0214
5
## Cumulative Proportion  0.4013 0.5880 0.7217 0.79920 0.86308 0.89996 0.9214
2
##                            PC8     PC9    PC10    PC11    PC12    PC13    P
C14
## Standard deviation     0.55444 0.48493 0.44708 0.41915 0.35804 0.26333 0.2
```

```
418
## Proportion of Variance 0.02049 0.01568 0.01333 0.01171 0.00855 0.00462 0.0
039
## Cumulative Proportion  0.94191 0.95759 0.97091 0.98263 0.99117 0.99579 0.9
997
##                           PC15
## Standard deviation     0.06793
## Proportion of Variance 0.00031
## Cumulative Proportion  1.00000
```

```
#standard deviation
pca_sdev<-pca$dev
#variance
pca_var<-pca_sdev^2
#proportion of variance to the sum of variance
pca_proportion_var<-pca_var/sum(pca_var)
```

Step 3: Scree plot and choose number of components to comput e regression model

```
#Scree plot, this plot is used to access components or factors which explains
the most variability in the data. It represents values in descending order.
screeplot(pca,main="Scree plot crime data", type= "line")
abline(h=1, col="green")
```




```
#Determine number of compponents to use from elbow plot. The point where the
slope of the curve is clearly leveling off (the “elbow) indicates the number
of factors that should be generated by the analysis. Since there seems to be
an ambiguity between 4 and 5, I will build the model with both and select the
one that gives me a better Rsquared.

pca_data_4<-cbind(pca$x[,1:4],crime_data[,16])
head(pca_data_4)

##            PC1        PC2         PC3         PC4     
## [1,] -4.199284 -1.0938312 -1.11907395  0.67178115  791
## [2,]  1.172663  0.6770136 -0.05244634 -0.08350709 1635
## [3,] -4.173725  0.2767750 -0.37107658  0.37793995  578
## [4,]  3.834962 -2.5769060  0.22793998  0.38262331 1969
## [5,]  1.839300  1.3309856  1.27882805  0.71814305 1234
## [6,]  2.907234 -0.3305421  0.53288181  1.22140635  682

model_pca_4<-lm(V5~.,data=as.data.frame(pca_data_4))
summary(model_pca_4)

## 
## Call:
## lm(formula = V5 ~ ., data = as.data.frame(pca_data_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
```

```
pca_data_5<-cbind(pca$x[,1:5],crime_data[,16])
head(pca_data_5)
```

```
##            PC1        PC2         PC3         PC4         PC5     
## [1,] -4.199284 -1.0938312 -1.11907395  0.67178115  0.05528338  791
## [2,]  1.172663  0.6770136 -0.05244634 -0.08350709 -1.17319982 1635
## [3,] -4.173725  0.2767750 -0.37107658  0.37793995  0.54134525  578
## [4,]  3.834962 -2.5769060  0.22793998  0.38262331 -1.64474650 1969
## [5,]  1.839300  1.3309856  1.27882805  0.71814305  0.04159032 1234
## [6,]  2.907234 -0.3305421  0.53288181  1.22140635  1.37436096  682
```

```
model_pca_5<-lm(V6~.,data=as.data.frame(pca_data_5))
summary(model_pca_5)
```

```
## 
## Call:
## lm(formula = V6 ~ ., data = as.data.frame(pca_data_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
```

```
#As seen above, the model_pca_5 has a better fit (larger R squared value:0.60
19 vs 0.2433) and I chose to make my prediction with this model.
```

Step 4: Compute equation using principal components

```
#get constant intercept
c<-model_pca_5$coefficients[1]
c
```

```
## (Intercept)
##    905.0851
#get other coefs (a1,a2,a3 etc...) responsible to make the equation(y=c+a1x1+
a2x2+a3x3 etc..), keeping in mind we are using 5 principal components.
coeffs<-model_pca_5$coefficients[2:6]
coeffs
```

```
##        PC1        PC2        PC3        PC4        PC5
##   65.21593  -70.08312   25.19408   69.44603 -229.04282
```

Step 5: Transform principal components to original variables fo r prediction (unscaling)

```
#get rotational matrix or eigenvector from PCA
eigen_vector<-pca$rotation[,1:5]
eigen_vector
```

```
##                PC1         PC2           PC3         PC4         PC5
## M      -0.30371194  0.06280357  0.1724199946 -0.02035537 -0.35832737
## So     -0.33088129 -0.15837219  0.0155433104  0.29247181 -0.12061130
## Ed      0.33962148  0.21461152  0.0677396249  0.07974375 -0.02442839
## Po1     0.30863412 -0.26981761  0.0506458161  0.33325059 -0.23527680
## Po2     0.31099285 -0.26396300  0.0530651173  0.35192809 -0.20473383
## LF      0.17617757  0.31943042  0.2715301768 -0.14326529 -0.39407588
## M.F     0.11638221  0.39434428 -0.2031621598  0.01048029 -0.57877443
## Pop     0.11307836 -0.46723456  0.0770210971 -0.03210513 -0.08317034
## 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
## Ineq   -0.36579778 -0.02752240 -0.0002944563 -0.08066612 -0.21672823
## Prob   -0.25888661  0.15831708 -0.1176726436  0.49303389  0.16562829
## Time   -0.02062867 -0.38014836  0.2235664632 -0.54059002 -0.14764767
```

```
#transforming our variables by multiplying the coeff values we got earlier by
this eigen vector
vars<-eigen_vector %*% coeffs
vars

##              [,1]
## M       60.794349
## So      37.848243
## Ed      19.947757
## Po1    117.344887
## Po2    111.450787
## LF      76.254902
## M.F    108.126558
## Pop     58.880237
## NW      98.071790
## U1       2.866783
## U2      32.345508
## Wealth  35.933362
## Ineq    22.103697
## Prob   -34.640264
## Time    27.205022

#Computing sigma for each column except Crime
sigma<-sapply(crime_data[,1:15],sd)
sigma

##            M           So           Ed          Po1          Po2
LF
##   1.25676339   0.47897516   1.11869985   2.97189736   2.79613186   0.04041
181
##          M.F          Pop           NW           U1           U2         Wea
lth
##   2.94673654  38.07118801  10.28288187   0.01802878   0.84454499 964.90944
200
##         Ineq         Prob         Time
##   3.98960606   0.02273697   7.08689519

#computing mean for each column except Crime
mu<-sapply(crime_data[,1:15], mean)
mu

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

```
#Compute original variables (unscaled) by dividing the variables we got earli
er (from the multiplication by eigen vectors) by sigma
main_vars<-vars/sigma
main_vars
```

```
##                 [,1]
## M        4.837374e+01
## So       7.901922e+01
## Ed       1.783120e+01
## Po1      3.948484e+01
## Po2      3.985892e+01
## LF       1.886946e+03
## M.F      3.669366e+01
## Pop      1.546583e+00
## NW       9.537384e+00
## U1       1.590115e+02
## U2       3.829933e+01
## Wealth   3.724014e-02
## Ineq     5.540321e+00
## Prob    -1.523521e+03
## Time     3.838779e+00
```

```
#Compute original intercept (unscaled)
main_c<-c-sum(vars*mu/sigma)
main_c #intercept from original data
```

```
## (Intercept)
##   -5933.837
```

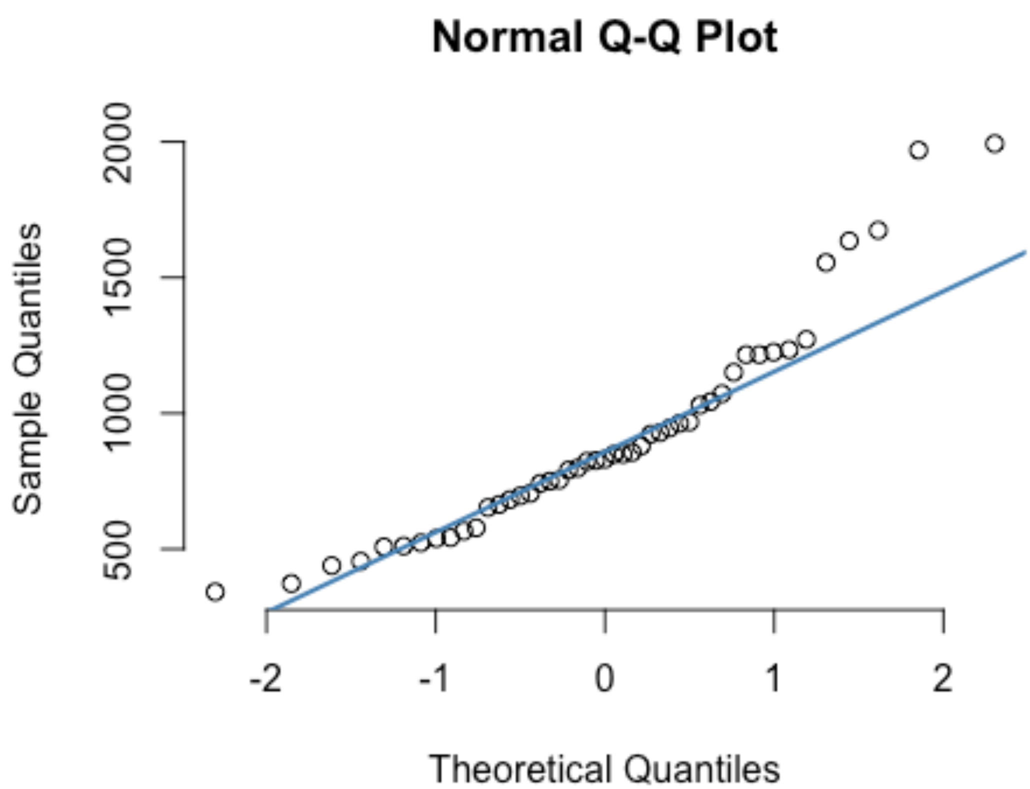
Step 6: Prepare test data and make prediction

```
test_data_frame<-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 = 0.040,Time = 39.0)

#PCA test data
test_data_pca<-data.frame(predict(pca,test_data_frame))
test_data_pca
```

```
##        PC1       PC2      PC3       PC4       PC5      PC6        PC7
PC8
## 1 1.224044 -2.767641 0.533605 -1.146837 -1.206098 2.333343 -0.1535916 -1.3
91625
##        PC9       PC10       PC11      PC12      PC13      PC14     PC15
## 1 1.460274 -0.4525158 -0.3466498 1.663782 -1.811307 -2.174071 1.288675
```

```
#prediction result using our PCA with 5 components
prediction_result<-predict(model_pca_5,test_data_pca)
prediction_result
```

```
##        1
## 1388.926

#prediction result is 1388.026, close to our prediction from last HW (1304), this result seems reasonable and falls within our crime range. Plotting a qq norm plot to see if this value could be an outlier.

qqnorm(crime_data$Crime,pch=1,frame=FALSE)
qqline(crime_data$Crime,col = "steelblue", lwd = 2)
```



1389 does not seem to be an outlier.