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)!)

Solution:

I. **I had some issues extracting the code and output from Jupyter notebook in VSCODE so snapshots are the best I could do. Sorry in advance.**

II. **Below will be snapshots of all code, outputs, markdown and results.**

III. **My conclusion is a markdown at the bottom of the last snapshot.**

HW 9.1 Solution.ipynb U

GTx_MM_in_Analytics > R_projects > HW 9.1 Solution.ipynb > library(DAAG)

+ Code + Markdown | Run All Clear Outputs of All Cells Go To Restart Interrupt | Outline ...

```r
library(DAAG)
library(magrittr)
```

[6] ✓ 0.7s

```r
data = read.table("C:/Users/ateje/Downloads/hw5_solutions-SP22/uscrime.txt", stringsAsFactors = FALSE, header = TRUE)

#optional check to make sure the data is read correctly
#head(data)

#Run principal component analysis and get summary.
data_pca = prcomp(data[,-16], center = TRUE, scale = TRUE)
summary(data_pca)

#plot using scree plot as shown on https://en.wikipedia.org/wiki/Scree_plot. This will determine how many principal components to keep.
plot(data_pca, type = "line")
attributes(data_pca)
data_pca$rotation
```

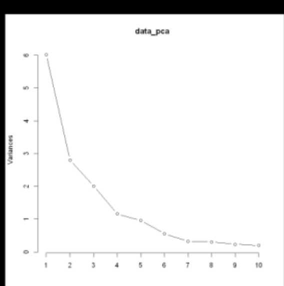
[7] ✓ 0.1s

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

$names
'sdev' · 'rotation' · 'center' · 'scale' · 'x'
$class
'prcomp'

A matrix: 15 × 15 of type dbl

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	PC11	PC12	PC13	PC14	PC15
M	-0.30371194	0.06280357	0.1724199946	-0.02035537	-0.35832737	-0.449132706	-0.15707378	-0.55367691	0.15474793	-0.01443093	0.39446657	0.16580189	-0.05142365	0.04901705	0.0051398012
So	-0.33088129	-0.15837219	0.0155433104	0.29247181	-0.12061130	-0.100500743	0.19649727	0.22734157	-0.65599872	0.06141452	0.23397868	-0.05753357	-0.29368483	-0.29364512	0.0084369230
Ed	0.33962148	0.21461152	0.0677396249	0.07974375	-0.02442839	-0.008571367	-0.23943629	-0.14644678	-0.44326978	0.51887452	-0.11821954	0.47786536	0.19441949	0.03964277	-0.0280052040
Po1	0.30863412	-0.26981761	0.0506458161	0.33325059	-0.23527680	-0.095776709	0.08011735	0.04613156	0.19425472	-0.14320978	-0.13042001	0.22611207	-0.18592255	-0.09490151	-0.6894155129
Po2	0.31099285	-0.26396300	0.0530651173	0.35192809	-0.20473383	-0.119524780	0.09518288	0.03168720	0.19512072	-0.05929780	-0.13885912	0.19088461	-0.13454940	-0.08259642	0.7200270100
LF	0.17617757	0.31943042	0.2715301768	-0.14326529	-0.39407588	0.504234275	-0.15931612	0.25513777	0.14393498	0.03077073	0.38532827	0.02705134	-0.27742957	-0.15385625	0.0336823193
M.F	0.11638221	0.39434428	-0.2031621598	0.01048029	-0.57877443	-0.074501901	0.15548197	-0.05507254	-0.24378252	-0.35323357	-0.28029732	-0.23925913	0.31624667	-0.04125321	0.0097922075
Pop	0.11307836	-0.46723456	0.0770210971	-0.03210513	-0.08317034	0.547098563	0.09046187	-0.59078221	-0.20244830	-0.03970718	0.05849643	-0.18350385	0.12651689	-0.05326383	0.0001496323
NW	-0.29358647	-0.22801119	0.0788156621	0.23925971	-0.36079387	0.051219538	-0.31154195	0.20432828	0.18984178	0.49201966	-0.20695666	-0.36671707	0.22901695	0.13227774	-0.0370783671
U1	0.04050137	0.00807439	-0.6590290980	-0.18279096	-0.13136873	0.017385981	-0.17354115	-0.20206312	0.02069349	0.22765278	-0.17857891	-0.09314897	-0.59039450	-0.02335942	0.0111359325
U2	0.01812228	-0.27971336	-0.5785006293	-0.06889312	-0.13499487	0.048155286	-0.07526787	0.24369650	0.05576010	-0.04750100	0.47021842	0.28440496	0.43292853	-0.03985736	0.0073618948
Wealth	0.37970331	-0.07718862	0.0100647664	0.11781752	0.01167683	-0.154683104	-0.14859424	0.08630649	-0.23196695	-0.11219383	0.31955631	-0.32172821	-0.14077972	0.70031840	-0.0025685109
Ineq	-0.36579778	-0.02752240	-0.0002944563	-0.08066612	-0.21672823	0.272027031	0.37483032	0.07184018	-0.02494384	-0.01390576	-0.18278697	0.43762828	-0.12181090	0.59279037	0.0177570357
Prob	-0.25888661	0.15831708	-0.1176726436	0.49303389	0.16562829	0.283535996	-0.56159383	-0.08598908	-0.05306898	-0.42530006	-0.08978385	0.15567100	-0.03547596	0.04761011	0.0293376260
Time	-0.02062867	-0.38014836	0.2235664632	-0.54059002	-0.14764767	-0.148203050	-0.44199877	0.19507812	-0.23551363	-0.29264326	-0.26363121	0.13536989	-0.05738113	-0.04488401	0.0376754405



This tell us that PC1:PC5 are the components to use since they account for 86% of the commulative proportion.

```r
principal_components = 5
```

Georgia Tech

```r
principal_components = 5


#Combine our principal components (PC1:PC5) with our initial "Crime" data in column 16
pca_data_matrix = cbind(data_pca$x[,1:principal_components],data[,16])
pca_data_matrix

#run lm() model of our V6 variable as a function of PC1:PC5
lm_model1 = lm(V6~., data = as.data.frame(pca_data_matrix))
summary(lm_model1)
```

[8] ✓ 0.1s

A matrix: 47 × 6 of type dbl

PC1	PC2	PC3	PC4	PC5	
-4.1992835	-1.09383120	-1.11907395	0.67178115	0.055283376	791
1.1726630	0.67701360	-0.05244634	-0.08350709	-1.173199821	1635
-4.1737248	0.27677501	-0.37107658	0.37793995	0.541345246	578
3.8349617	-2.57690596	0.22793998	0.38262331	-1.644746496	1969
1.8392999	1.33098564	1.27882805	0.71814305	0.041590320	1234
2.9072336	-0.33054213	0.53288181	1.22140635	1.374360960	682
0.2457752	-0.07362562	-0.90742064	1.13685873	0.718644387	963
-0.1301330	-1.35985577	0.59753132	1.44045387	-0.222781388	1555
-3.6103169	-0.68621008	1.28372246	0.55171150	-0.324292990	856
1.1672376	3.03207033	0.37984502	-0.28887026	-0.646056610	705
2.5384879	-2.66771358	1.54424656	-0.87671210	-0.324083561	1674
1.0065920	-0.06044849	1.18861346	-1.31261964	0.358087724	849
0.5161143	0.97485189	1.83351610	-1.59117618	0.599881946	511
0.4265556	1.85044812	1.02893477	-0.07789173	0.741887592	664
-3.3435299	0.05182823	-1.01358113	0.08840211	0.002969448	798
-3.0310689	-2.10295524	-1.82993161	0.52347187	-0.387454246	946
-0.2262961	1.44939774	-1.37565975	0.28960865	1.337784608	539
-0.1127499	-0.39407030	-0.38836278	3.97985093	0.410914404	929
2.9195668	-1.58646124	0.97612613	0.78629766	1.356288600	750
2.2998485	-1.73396487	-2.82423222	-0.23281758	-0.653038858	1225
1.1501667	0.13531015	0.28506743	-2.19770548	0.084621572	742
-5.6594827	-1.09730404	0.10043541	-0.05245484	-0.689327990	439
-0.1011749	-0.57911362	0.71128354	-0.44394773	0.689939865	1216
1.3836281	1.95052341	-2.98485490	-0.35942784	-0.744371276	968
0.2727756	2.63013778	1.83189535	0.05207518	0.803692524	523
4.0565577	1.17534729	-0.81690756	1.66990720	-2.895110075	1993
0.8929694	0.79236692	1.26822542	-0.57575615	1.830793964	342
0.1514495	1.44873320	0.10857670	-0.51040146	-1.023229895	1216
3.5592481	-4.76202163	0.75080576	0.64692974	0.309946510	1043
-4.1184576	-0.38073981	1.43463965	0.63330834	-0.254715638	696
-0.6811731	1.66926027	-2.88645794	-1.30977099	-0.470913997	373
1.7157269	-1.30836339	-0.55971313	-0.70557980	0.331277622	754
-1.8860627	0.59058174	1.43570145	0.18239089	0.291863659	1072

```
Call:
lm(formula = V6 ~ ., data = as.data.frame(pca_data_matrix))

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
```

After getting results for the linear regression we focus on transformation. We start by extracting intercept and beta vector before we calculate the alpha vector.

```r
lm_model1$coefficients
```

[9] ✓ 0.7s

(Intercept): 905.085106382979 PC1: 65.215930138666 PC2: -70.0831185497858 PC3: 25.1940780425772 PC4: 69.4460307968389 PC5: -229.042822001686

+ Code + Markdown


```r
lm_model1$coefficients
```

[9] ✓ 0.7s

(Intercept): 905.085106382979 PC1: 65.215930138666 PC2: -70.0831185497858 PC3: 25.1940780425772 PC4: 69.4460307968389 PC5: -229.042822001686


```r
test = 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)

test_newcity = data.frame(predict(data_pca, test))

test_newcity_model = predict(lm_model1, test_newcity)

test_newcity_model
```

[11] ✓ ✓ ✓ ✓ 0.5s

1: 1388.92569475604

A prediction of 1388 with an adjusted R-squared of 0.60 is a good prediction that stacks well against last week's cross validation. Overall we see that PCA can generate comparable results when observing less predictors while not risking losing valuable information when using cross validation couples with methods like RFE.