Project Feedback

Priyank Thakkar

18/11/2021

INTRODUCTION

This data set is all about the students who graduate from the Universities of USA. I am curious to know that does your Major in your study matter or not for your economic success. What is the general trend right now. And, what are the steps we can take before selecting the Major to boost your odds for the same. And mostly what is the share of the women in the different categories.

DATA

This data is from the GitHub repository, from Fivethirtyeight.com. All data is from American Community Survey 2010-2012 Public Use Micro data series. All Three files in the repository contains basic earnings and labor force information. "recent-grads.csv" contains a more detailed breakdown, including by sex and by the type of job they got. "grad-students.csv" contains details on graduate school attendees. Here, we have used the data from the file of "recent-grades.csv". And here is the bifurcation of all the Headers and their Descriptions. Our data contains 174x21 size of entries.

Where, we can see that 174 are rows, they represent distinct 174 majors, from all Major Categories available in the data. Moreover, all 21 columns and their representation are below.

Header	**Description**
Rank	: Rank by median earnings
Major_code	: Major code
Major	: Major description
Major_category	: Category of major
Total	: Total number of people with major
Sample_size	: Sample size (unweighted) of full-time, year-round ONLY
Men	: Male graduates
Women	: Female graduates
ShareWomen	: Women as share of total
Employed	: Number employed
Full_time	: Employed 35 hours or more
Part_time	: Employed less than 35 hours
Full_time_year_round	: Employed at least 50 weeks and at least 35 hours
Unemployed	: Number unemployed
Unemployment_rate	: Unemployed / (Unemployed + Employed)
Median	: Median earnings of full-time, year-round workers (Normalized)

P25th	: 25th percentile of earnings
P75th	: 75th percentile of earnings
College_jobs	: Number with job requiring a college degree
Non_college_jobs	: Number with job not requiring a college degree
Low_wage_Jobs	: Number in low-wage service jobs

For our variable study, we may need all the variables later on for more explorations. But here are few of the variables that we can focus on for PCA and EDA.

- Total, Men, Women
- ShareWomen
- Mean
- Employed
- Unemployed
- Unemployment_rate
- College_jobs
- Non_college_jobs
- Low_wage_jobs

Principal Components Analysis

Step 1: Load the libraries and Data

```
library(tidyverse)

## -- Attaching packages --------------------------------------- tidyverse 1.
3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.4     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.1     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflict
s() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()

setwd("D:\\SJSU_HW\\GitHubSJSU\\RStudio_Learning\\Data_set")
data_grade <- read.csv("recent_grads.csv")
recent_grade <- as_tibble(data_grade)
is_tibble(recent_grade)

## [1] TRUE

str(recent_grade)

## tibble [173 x 21] (S3: tbl_df/tbl/data.frame)
##  $ Rank                : int [1:173] 1 2 3 4 5 6 7 8 9 10 ...
##  $ Major_code          : int [1:173] 2419 2416 2415 2417 2405 2418 6202 50
```

```
01 2414 2408 ...
##  $ Major               : chr [1:173] "PETROLEUM ENGINEERING" "MINING AND M
INERAL ENGINEERING" "METALLURGICAL ENGINEERING" "NAVAL ARCHITECTURE AND MARIN
E ENGINEERING" ...
##  $ Total               : int [1:173] 2339 756 856 1258 32260 2573 3777 179
2 91227 81527 ...
##  $ Men                 : int [1:173] 2057 679 725 1123 21239 2200 2110 832
80320 65511 ...
##  $ Women               : int [1:173] 282 77 131 135 11021 373 1667 960 109
07 16016 ...
##  $ Major_category      : chr [1:173] "Engineering" "Engineering" "Engineer
ing" "Engineering" ...
##  $ ShareWomen          : num [1:173] 0.121 0.102 0.153 0.107 0.342 ...
##  $ Sample_size         : int [1:173] 36 7 3 16 289 17 51 10 1029 631 ...
##  $ Employed            : int [1:173] 1976 640 648 758 25694 1857 2912 1526
76442 61928 ...
##  $ Full_time           : int [1:173] 1849 556 558 1069 23170 2038 2924 108
5 71298 55450 ...
##  $ Part_time           : int [1:173] 270 170 133 150 5180 264 296 553 1310
1 12695 ...
##  $ Full_time_year_round: int [1:173] 1207 388 340 692 16697 1449 2482 827
54639 41413 ...
##  $ Unemployed          : int [1:173] 37 85 16 40 1672 400 308 33 4650 3895
...
##  $ Unemployment_rate   : num [1:173] 0.0184 0.1172 0.0241 0.0501 0.0611 ..
.
##  $ Median              : int [1:173] 110000 75000 73000 70000 65000 65000
62000 62000 60000 60000 ...
##  $ P25th               : int [1:173] 95000 55000 50000 43000 50000 50000 5
3000 31500 48000 45000 ...
##  $ P75th               : int [1:173] 125000 90000 105000 80000 75000 10200
0 72000 109000 70000 72000 ...
##  $ College_jobs        : int [1:173] 1534 350 456 529 18314 1142 1768 972
52844 45829 ...
##  $ Non_college_jobs    : int [1:173] 364 257 176 102 4440 657 314 500 1638
4 10874 ...
##  $ Low_wage_jobs       : int [1:173] 193 50 0 0 972 244 259 220 3253 3170
...
```

Step 2: Remove Missing values and create with our Variables.

For PCA we are taking only 6 variables. We, choose these variables because we want to predict that what will the Mean salary looks like for any Major category. And these variables help us to predict such.

- Total
- Employed
- Full_time
- Unemployed
- Median

- College_jobs

```
# Check if any NA values
recent_grade %>% summarise(Total_Count = n())

## # A tibble: 1 x 1
##   Total_Count
##         <int>
## 1         173

filter(recent_grade, is.na(Total) | is.na(Men) | is.na(Women) | is.na(ShareWo
men)) %>%
  summarise(Missing_Count = n())

## # A tibble: 1 x 1
##   Missing_Count
##           <int>
## 1             1

# Drop the NA
new_recent_grade <- drop_na(recent_grade)

# Generate our new Data only from the variables that are in need.
batch <- select(new_recent_grade,
                Total,
                Employed,
                Full_time,
                Unemployed,
                Median,
                College_jobs)
is_tibble(batch)

## [1] TRUE

#check for all the variables, they should be numeric for PCA calculation.
str(batch)

## tibble [172 x 6] (S3: tbl_df/tbl/data.frame)
##  $ Total       : int [1:172] 2339 756 856 1258 32260 2573 3777 1792 91227
81527 ...
##  $ Employed    : int [1:172] 1976 640 648 758 25694 1857 2912 1526 76442 6
1928 ...
##  $ Full_time   : int [1:172] 1849 556 558 1069 23170 2038 2924 1085 71298
55450 ...
##  $ Unemployed  : int [1:172] 37 85 16 40 1672 400 308 33 4650 3895 ...
##  $ Median      : int [1:172] 110000 75000 73000 70000 65000 65000 62000 62
000 60000 60000 ...
##  $ College_jobs: int [1:172] 1534 350 456 529 18314 1142 1768 972 52844 45
829 ...
```

Step 3: Calculate Cumulative Percent Variance.

- Calculate Correlation matrix.
- Calculate Scaled Covariance for later use of Principal component score.
- Get Eigen Values and Vectors from Correlation matrix.
- For Percent Variance apply below formula.
- $PercentVariance = EigenValues/sum(EigenValues)$

```
# Calculate the Correlation
cor <- cor(batch)

# Calculate the Scaled Covariance = correlation
scaled <- scale(batch)
ScaleCov <- cov(scaled)
ScaleCov

##                   Total   Employed   Full_time Unemployed      Median
## Total         1.0000000  0.9962140  0.98933921  0.9747684 -0.10673767
## Employed      0.9962140  1.0000000  0.99583083  0.9688554 -0.10439869
## Full_time     0.9893392  0.9958308  1.00000000  0.9600422 -0.07903094
## Unemployed    0.9747684  0.9688554  0.96004220  1.0000000 -0.12362234
## Median       -0.1067377 -0.1043987 -0.07903094 -0.1236223  1.00000000
## College_jobs  0.8004648  0.7971931  0.77213457  0.7133619 -0.04706015
##              College_jobs
## Total          0.80046477
## Employed       0.79719311
## Full_time      0.77213457
## Unemployed     0.71336190
## Median        -0.04706015
## College_jobs   1.00000000

eigenCor<-  eigen(cor)
eigenCor

## eigen() decomposition
## $values
## [1] 4.614797653 0.991472970 0.347380474 0.039531910 0.005367193 0.00144980
0
##
## $vectors
##             [,1]        [,2]       [,3]        [,4]        [,5]        [,6
]
## [1,] -0.46331349  0.01386713 -0.1244759  0.04525717  0.81834628  0.3129140
8
## [2,] -0.46307459  0.01630305 -0.1313057  0.28272048  0.01905015 -0.8293136
7
## [3,] -0.45907317  0.03992598 -0.1906968  0.54170715 -0.49567161  0.4606034
0
```

```
## [4,] -0.45038717 -0.01222966 -0.3365600 -0.77899337 -0.27537768  0.0326443
1
## [5,]  0.05760302  0.99585631 -0.0631648 -0.02632888  0.01190222 -0.0112889
5
## [6,] -0.39241259  0.07790044  0.9020182 -0.13057721 -0.09100319  0.0312254
4
```

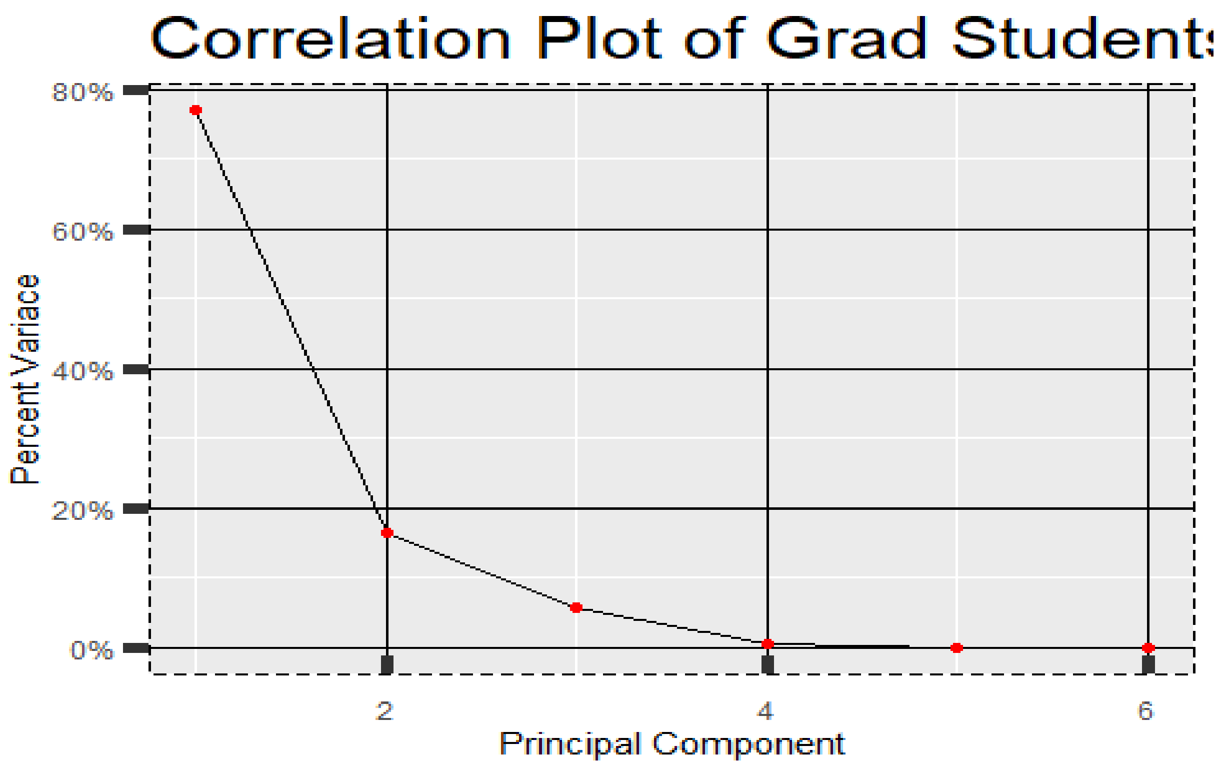
Now, we will calculate the Percent Variance, and that will give us the information about, what proportion of total variance is explained by the First, second and till the end of principal component.

We can calculate Cumulative percent variance just to see that how many columns represents major portion of the data information.

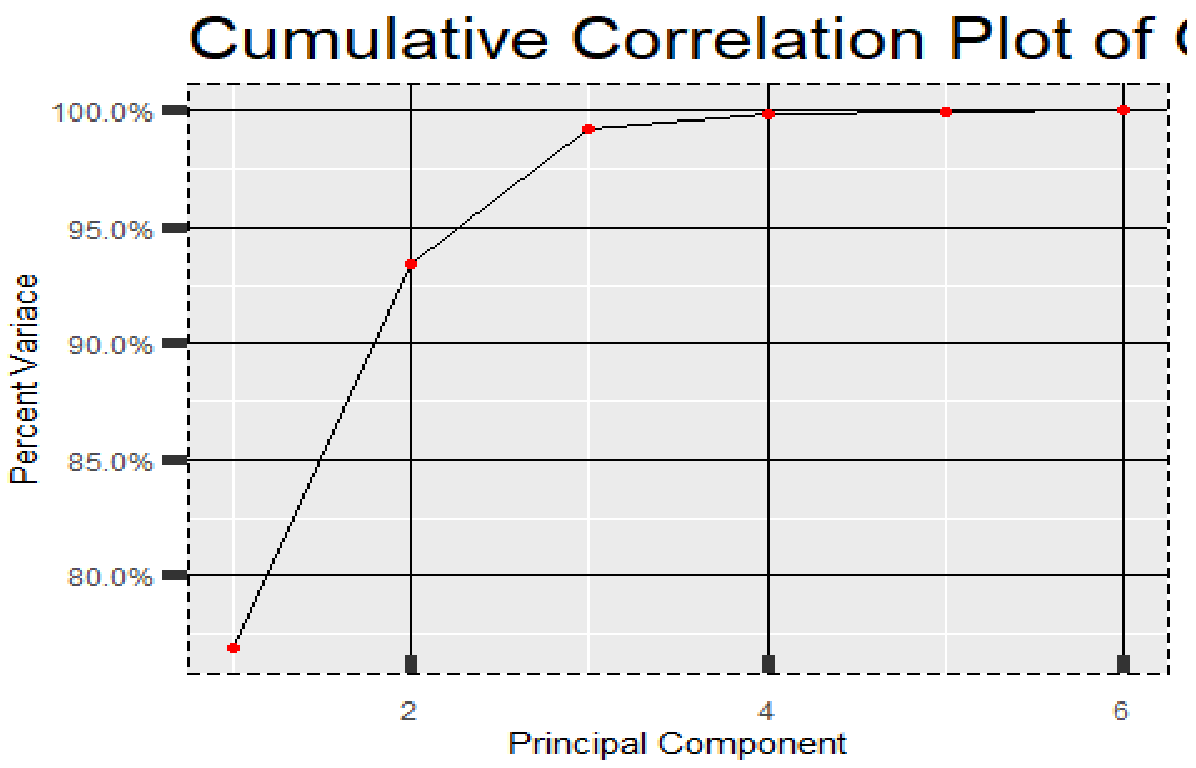
```
# Percent Variance
PV <- eigenCor$values/sum(eigenCor$values)
PV

## [1] 0.7691329422 0.1652454949 0.0578967456 0.0065886516 0.0008945322
## [6] 0.0002416334

# Cumulative Percent Variance
cumsum(PV)

## [1] 0.7691329 0.9343784 0.9922752 0.9988638 0.9997584 1.0000000
```

Now, using the Graphs we will can see that how many variables we need to represent the data and what are the variables we can reduce.

```
SwCorPlot <- qplot(c(1:6),PV) +
  geom_line()+
  geom_point(shape = 20,colour = "red", fill = NA , size = 2, stroke = 1 ) +
  xlab("Principal Component") +
  ylab("Percent Variace") +
  ggtitle("Correlation Plot of Grad Students") +
  scale_y_continuous(labels = scales::percent)+
  theme(plot.title = element_text(size = rel(2))) +
  theme(panel.grid.major = element_line(colour = "black")) +
  theme(panel.border = element_rect(linetype = "dashed", fill = NA)) +
  theme(axis.ticks = element_line(size = 2)) +

  theme(
    axis.ticks.length.y = unit(.25, "cm"),
    axis.ticks.length.x = unit(-.25, "cm"),
    axis.text.x = element_text(margin = margin(t = .3, unit = "cm"))
  )
SwCorPlot
```



```
SwCumCorPlot <- qplot(c(1:6),cumsum(PV)) +
  geom_line()+
  geom_point(shape = 20,colour = "red", fill = NA , size = 2, stroke = 1 ) +
  xlab("Principal Component") +
  ylab("Percent Variace") +
  ggtitle("Cumulative Correlation Plot of Grad Students") +
  scale_y_continuous(labels = scales::percent) +
  theme(plot.title = element_text(size = rel(2))) +
  theme(panel.grid.major = element_line(colour = "black")) +
  theme(panel.border = element_rect(linetype = "dashed", fill = NA)) +
  theme(axis.ticks = element_line(size = 2)) +

  theme(
    axis.ticks.length.y = unit(.25, "cm"),
    axis.ticks.length.x = unit(-.25, "cm"),
    axis.text.x = element_text(margin = margin(t = .3, unit = "cm"))
  )
SwCumCorPlot
```



Now, We will count the Principal Components Score.

The sample principal components are defined as those linear combinations which have maximum sample variance. If we project the 172 data points onto the first eigen vectors, the projected values are called the first principal component.

From above graph, we can say that. 1st component contains 76% data, 2nd component contains 93% and if we include 3rd component the total data will be 99%. So no need to add more component, they have very negligent data available which does not bother us.

```
selectedEigenValues <- eigenCor$vectors[,1:3]
colnames(selectedEigenValues) = c("PC1", "PC2", "PC3")
row.names(selectedEigenValues) = colnames(batch)
selectedEigenValues

##                      PC1         PC2        PC3
## Total        -0.46331349  0.01386713 -0.1244759
## Employed     -0.46307459  0.01630305 -0.1313057
## Full_time    -0.45907317  0.03992598 -0.1906968
## Unemployed   -0.45038717 -0.01222966 -0.3365600
## Median        0.05760302  0.99585631 -0.0631648
## College_jobs -0.39241259  0.07790044  0.9020182

# Principal component scores for batch data
PC1 <- as.matrix(scaled) %*% selectedEigenValues[,1]
PC2 <- as.matrix(scaled) %*% selectedEigenValues[,2]
PC3 <- as.matrix(scaled) %*% selectedEigenValues[,3]
```

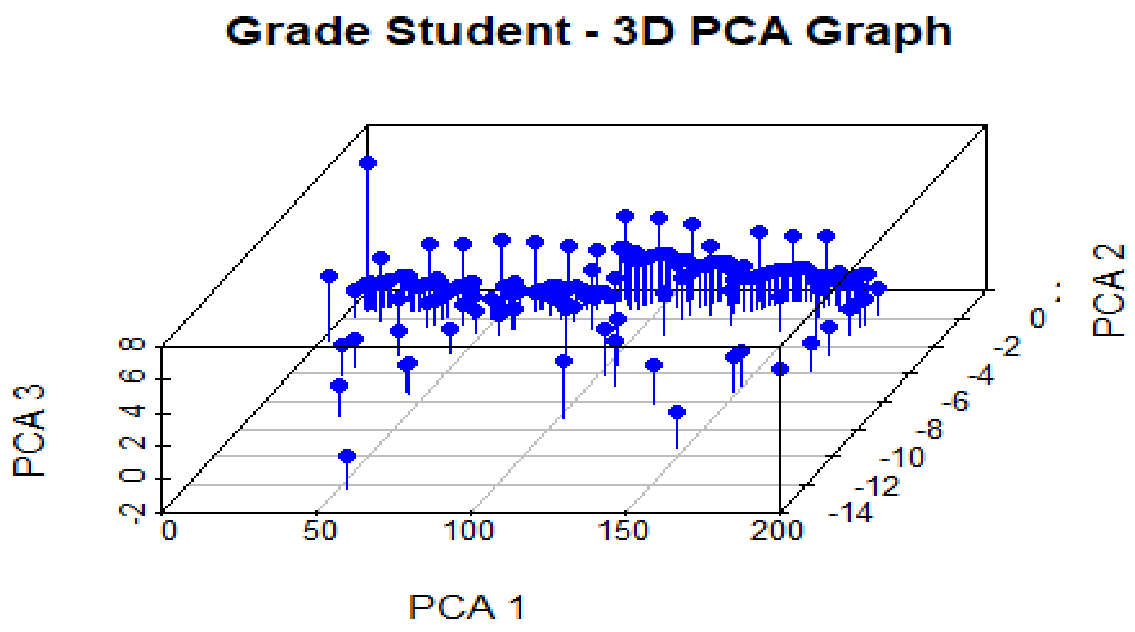
```
# get it into one data frame and see the head
PC <- data.frame(grade_student = row.names(batch), PC1,PC2,PC3)
head(PC)
```

```
##   grade_student       PC1      PC2        PC3
## 1             1 1.6103298 6.002849 -0.3922069
## 2             2 1.4885026 2.955332 -0.2409742
## 3             3 1.4832178 2.782174 -0.2200640
## 4             4 1.4547778 2.522303 -0.2057466
## 5             5 0.2344976 2.183249  0.2167440
## 6             6 1.3490662 2.090573 -0.1914041
```

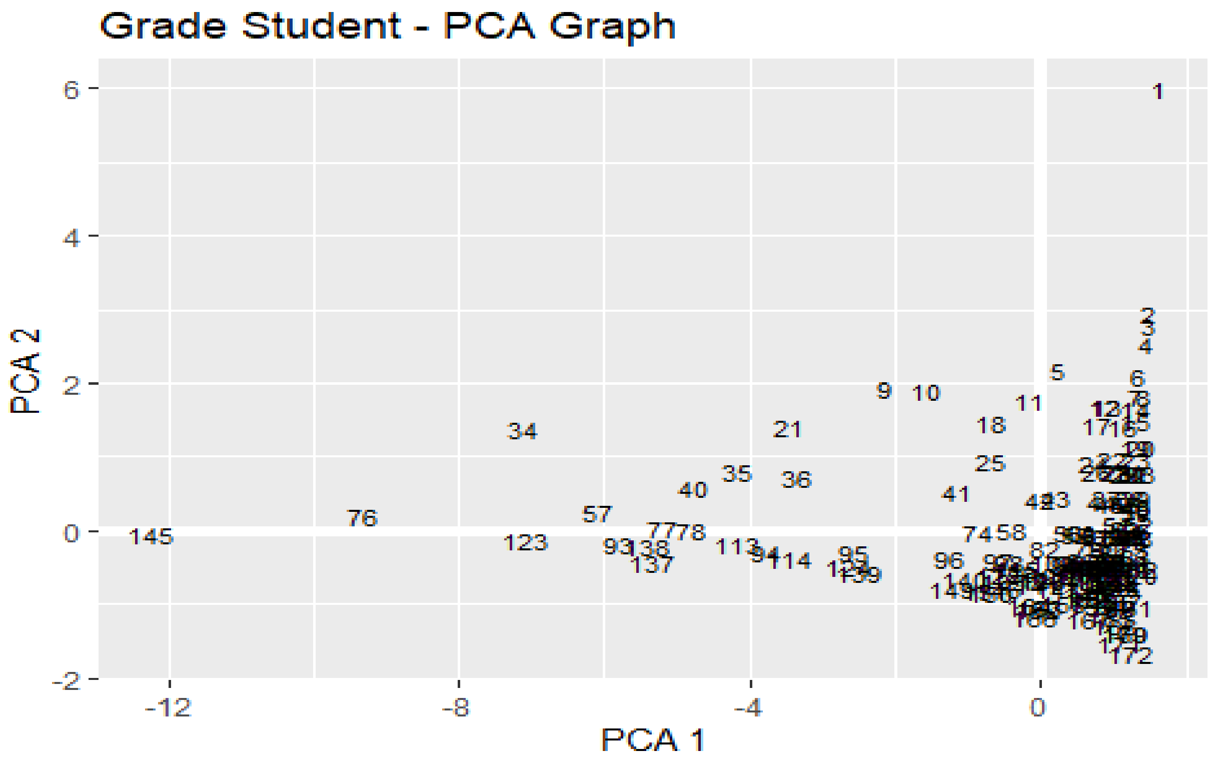
Step 4: Visualization of the PC

We, have selected three principal components so lets visualize them. 1. Visualize PC1, PC2 and PC3 together. 2. Visualize PC1 and PC2 only.

```
library(scatterplot3d)

scatterplot3d(PC[,1:3], angle = 75, pch = 16,
              main = "Grade Student - 3D PCA Graph",
              xlab = "PCA 1",
              ylab = "PCA 2",
              zlab = "PCA 3",
              color = "Blue",
              type = "h"
              )
```

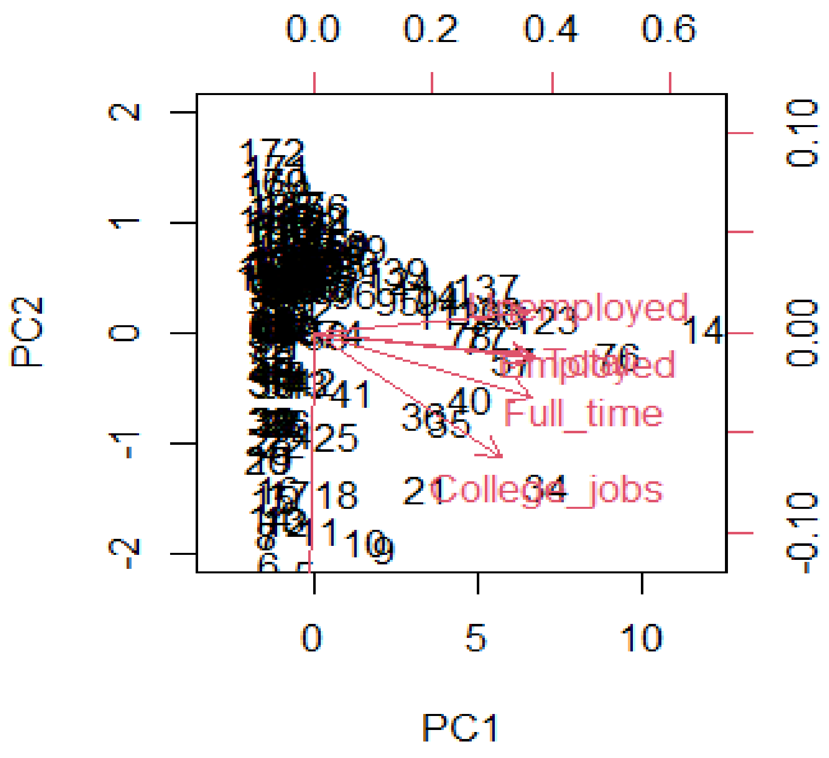


```
ggplot(PC, aes(PC1,PC2))+
  modelr::geom_ref_line(h=0) +
  modelr::geom_ref_line(v=0) +
  geom_text(aes(label = grade_student),size =3) +
  xlab("PCA 1") +
  ylab("PCA 2") +
  ggtitle("Grade Student - PCA Graph")
```



Now, get Visualization of the PC1 and PC2, from the inbuilt function who counts the PC. And show the vectors on the map.

```
results <- prcomp(batch, scale = TRUE)
results

## Standard deviations (1, .., p=6):
## [1] 2.14820801 0.99572736 0.58938992 0.19882633 0.07326113 0.03807624
##
## Rotation (n x k) = (6 x 6):
##                     PC1         PC2        PC3         PC4         PC5
## Total        0.46331349 -0.01386713  0.1244759 -0.04525717 -0.81834628
## Employed     0.46307459 -0.01630305  0.1313057 -0.28272048 -0.01905015
## Full_time    0.45907317 -0.03992598  0.1906968 -0.54170715  0.49567161
## Unemployed   0.45038717  0.01222966  0.3365600  0.77899337  0.27537768
## Median      -0.05760302 -0.99585631  0.0631648  0.02632888 -0.01190222
## College_jobs 0.39241259 -0.07790044 -0.9020182  0.13057721  0.09100319
##                     PC6
## Total        0.31291408
## Employed      -0.82931367
## Full_time      0.46060340
## Unemployed     0.03264431
## Median        -0.01128895
## College_jobs   0.03122544

biplot(results, scale = 0,  expand=3, xlim=c(-3.0, 12.0), ylim=c(-2.0, 2.0))
```



Principal Components Regression

Given a set of p predictor variables and a response variable, multiple linear regression uses a method known as least squares to minimize the sum of squared residuals (RSS):

- $RSS = \Sigma(y_i - \hat{y}_i)^2$

Where :

```
Σ: A greek symbol that means sum
yi: The actual response value for the ith observation
ŷi: The predicted response value based on the multiple linear regression mode
l
```

However, when the predictor variables are highly correlated then multicollinearity can become a problem. This can cause the coefficient estimates of the model to be unreliable and have high variance.

One way to avoid this problem is to instead use principal components regression, which finds M linear combinations (known as “principal components”) of the original p predictors and then uses least squares to fit a linear regression model using the principal components as predictors.

Step 1: Load Necessary Packages

The set.seed() function sets the starting number used to generate a sequence of random numbers – it ensures that you get the same result if you start with that same seed each time you run the same process.

```
# load the package
library(pls)

##
## Attaching package: 'pls'

## The following object is masked from 'package:stats':
##
##     loadings

#make this example reproducible
set.seed(1)
```

Step 2: Fit PCR Model

By using PCR you can easily perform dimensionality reduction on a high dimensional dataset and then fit a linear regression model to a smaller set of variables, while at the same time keep most of the variability of the original predictors.

we’ll fit a principal components regression (PCR) model using Median as the response variable and the following variables as the predictor variables.

```
•  scale=TRUE: This tells R that each of the predictor variables should be
   scaled to have a mean of 0 and a standard deviation of 1. This ensures
   that no predictor variable is overly influential in the model if it hap
   pens to be measured in different units.

•  validation=”CV”: This tells R to use k-fold cross-validation to evaluat
   e the performance of the model. Note that this uses k=10 folds by defau
   lt.
```

```
model <- pcr(Median~Total+Employed+Full_time+Unemployed+College_jobs,
             data=batch,
             scale=TRUE,
             validation="CV")
```

Step 3: Choose the Number of Principal Components

Even though we have chosen the Principal Components from the previous method with the help of cumulative percent variance. We wanted to check it again with the model we just created. So, if there any confusion it will clarify it up for us.

```
#view summary of model fitting
summary(model)

## Data:    X dimension: 172 5
##  Y dimension: 172 1
## Fit method: svdpc
## Number of components considered: 5
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps
## CV           11495    11427    11433    11336    11420    11149
## adjCV        11495    11424    11428    11332    11411    11118
##
## TRAINING: % variance explained
##         1 comps  2 comps  3 comps  4 comps  5 comps
## X       92.0553   99.055   99.858    99.97   100.00
## Median   0.9448    1.421    2.988     5.52    12.61
```

There are two tables of interest in the output:

1. **VALIDATION: RMSEP**

This table tells us the test RMSE calculated by the k-fold cross validation. We can see the following:

```
If we only use the intercept term in the model, the test RMSE is 11495.
If we add in the first principal component, the test RMSE drops to 11427.
If we add in the second principal component, the test RMSE increase to 11433.
```

We can see that adding additional principal components actually leads to an increase in test RMSE. Thus, it appears that it would be optimal to only use two principal components in the final model.

2. **TRAINING: % variance explained**

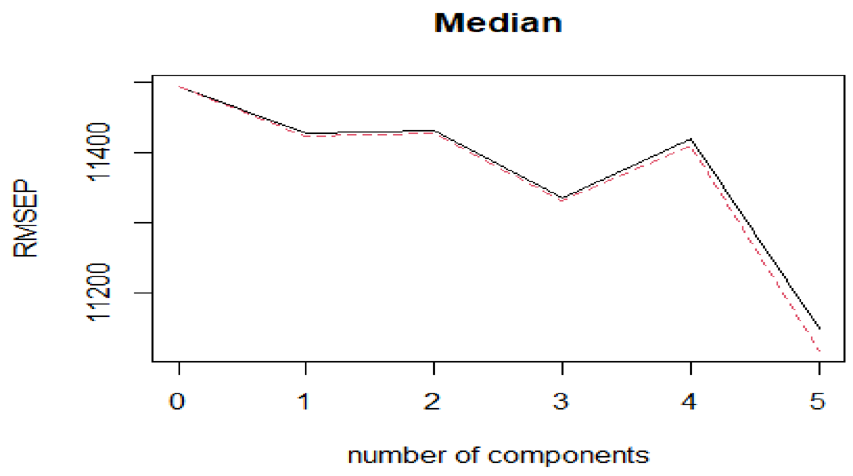
This table tells us the percentage of the variance in the response variable explained by the principal components. We can see the following:

```
By using just the first principal component, we can explain 92.05% of the var
iation in the response variable.
By adding in the second principal component, we can explain 99.05% of the var
iation in the response variable.
```

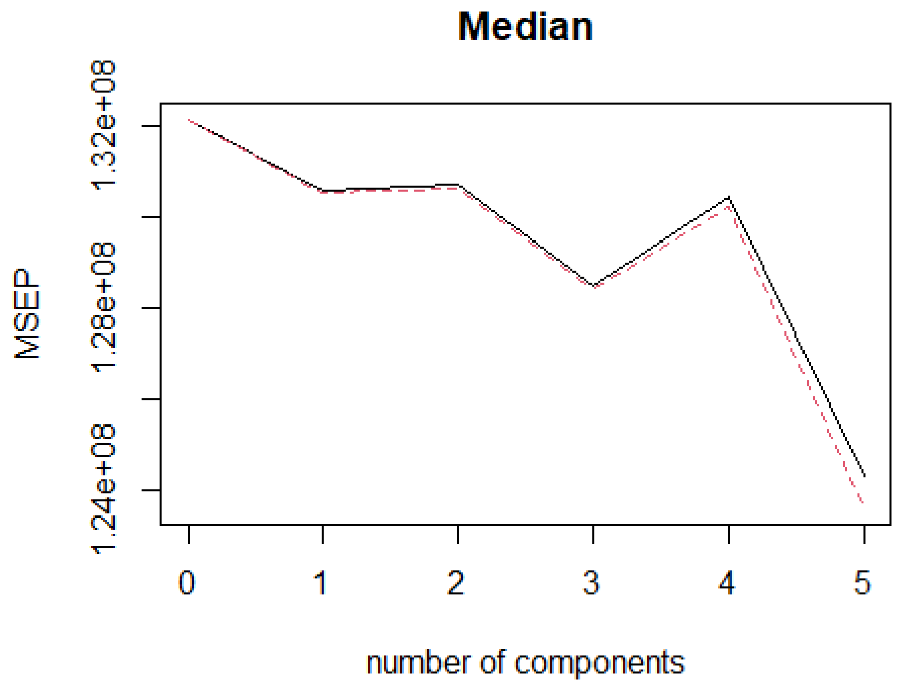
Note that we'll always be able to explain more variance by using more principal components, but we can see that adding in more than two principal components doesn't actually increase the percentage of explained variance by much.

We can also visualize the test RMSE (along with the test MSE and R-squared) based on the number of principal components by using the validationplot() function.

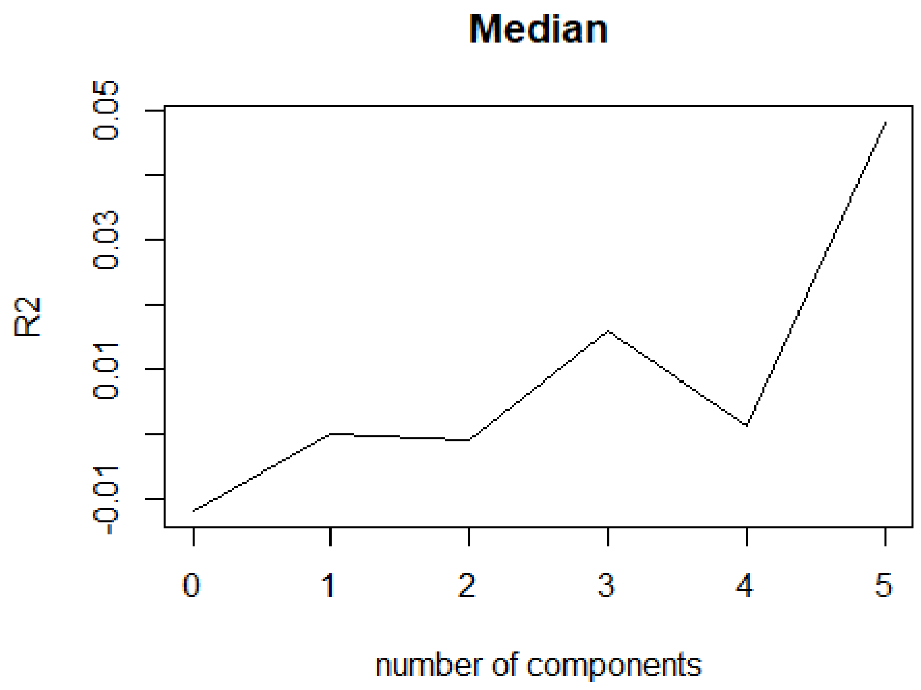
```
#visualize cross-validation plots
validationplot(model)
```



```
validationplot(model, val.type="MSEP")
```



```
validationplot(model, val.type="R2")
```



In each plot we can see that the model fit improves by adding in two principal components, yet it tends to get worse when we add more principal components.

Thus, the optimal model includes just the first two principal components.

Step 4: Use the Final Model to Make Predictions

We can use the final PCR model with two principal components to make predictions on new observations.

The following code shows how to split the original dataset into a training and testing set and use the PCR model with two principal components to make predictions on the testing set.

NOTE : Choose the number or **ncomp** as the number we selected for Principal components, at the time to predict the model, after training.

```
#define training and testing sets
train <- batch[1:130,c("Total","Employed","Full_time","Unemployed","Median","
College_jobs")]
y_test <- batch[131:nrow(batch), c("Median")]
test <- batch[131:nrow(batch),c("Total","Employed","Full_time","Unemployed","
College_jobs")]
```

```r
#use model to make predictions on a test set
modelT <-pcr(Median~Total+Employed+Full_time+Unemployed+College_jobs,
             data=train,
             scale=TRUE,
             validation="CV")
pcr_pred <- predict(modelT, test, ncomp=2)

#calculate RMSE
sq = (pcr_pred - y_test)^2
sqrt(mean(sq$Median))

## [1] 14124.03
```

Now, let's try to predict the value of the Median Income of some Major categories for which values look like below. It will predict the values from the train model and no of principal component as we selected before.

```r
nd = with(train, data.frame(Total=2000,
                            Employed=1976,
                            Full_time=1800,
                            Unemployed=37,
                            College_jobs=1534))
nd$pred = predict(modelT, newdata=nd, type="response", ncomp=2)
nd

##   Total Employed Full_time Unemployed College_jobs Median.2 comps
## 1  2000     1976      1800         37         1534       44256.55
```

Thank you for Reading. Let me know Professor if i am wrong somewhare, or have to add something in the report specifically. Will beautify later.