Untitled

June Namgung
10/29/2017

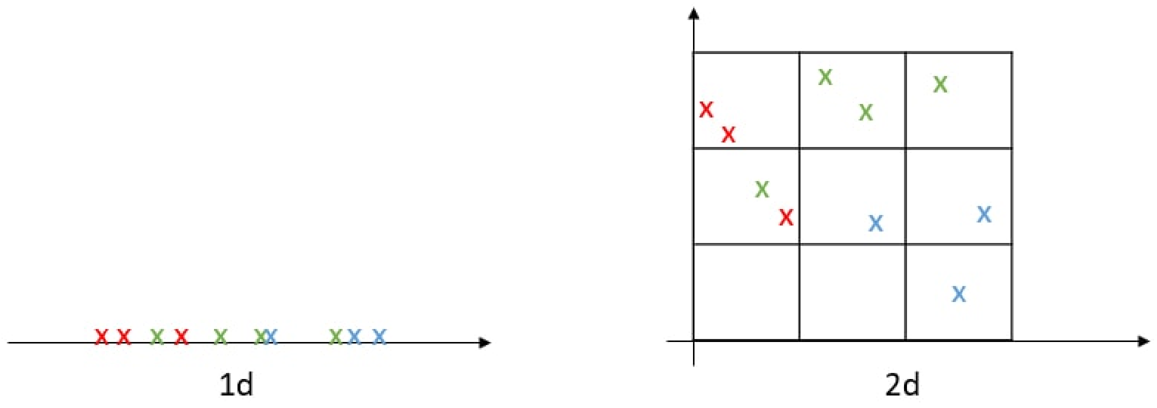
PCA Practice

Introduction

6 weeks already passed from the start of this semester. I had no prior experience with language 'R' at first, but I'm getting used to class materials. There were many interesting things that we've covered from the beginning of the class. Among them, the thing that really caught up my mind was PCA. When I first saw PCA, it was kind of confusing. I know nothing about principal components at all. But as I practice, I could manipulate the matrix in a same way that linear algebra does. I could transpose the matrix, get an eigenvalue from it, and even analyze data with them. It really caught my attention because I had fun when I took linear algebra class before. And.. I won't lie. To be honest, I could not understand PCA that much while I was doing the last homework. These are the reason why I'm reviewing some of PCA basics and do PCA practice for Y'all who had the same experience with me.

PCA Basic

PCA, Principal Component Analysis is a dimension reducing technique that is widely used in data analysis. PCA deals with the "curse of dimensionality". The curse of dimensionality usually refers to what happens when you add more variables to a multivariate model. It gets harder to predict certain values if we add more dimensions to a data set.



As dimension of data gets higher, the volume,so called sparcity, increases exponentially. So, the basic concept of PCA starts from the idea: It is more efficient to use less-dimensional compressed data, called principal components, rather than original high-dimensional data if the two methods do not differ much in prediction.

We use PCA instead of using linear regression or decision tree model when there exists multicollinearity, a state of very high intercorrelations/inter-associations among the independent variables. It is a type of disturbance in the data, and if present in the data the statistical inferences made about the data may not be reliable. PCA can deal with multicollinearity since the principal components are orthogonal.

Process

i> Get some raw data

ii> Adjust the raw data, by subtracting the mean from each value

iii> Calculate the covariance matrix with the result

iv> Get the eigenvalues and eigenvectors

v> Choosing components and form a feature vector

vi> Get a final data, multiplying adjusted raw data and featured raw data

Analyzing Data

First, We will find a correlation between salary and many other basketball stats by using linear regression and lowess line. I will recall the data we've already dealt with, nba 2016-17 player statistics.

```r
#install.packages("gridExtra")
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 3.4.2

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(ggplot2)
library(grid)#
library(gridExtra)#These two libraries are for putting multiple scatterplots on one page
```

```
##
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':
##
##     combine
```

```r
pool <- read.csv('nba2017-player-statistics.csv')
```

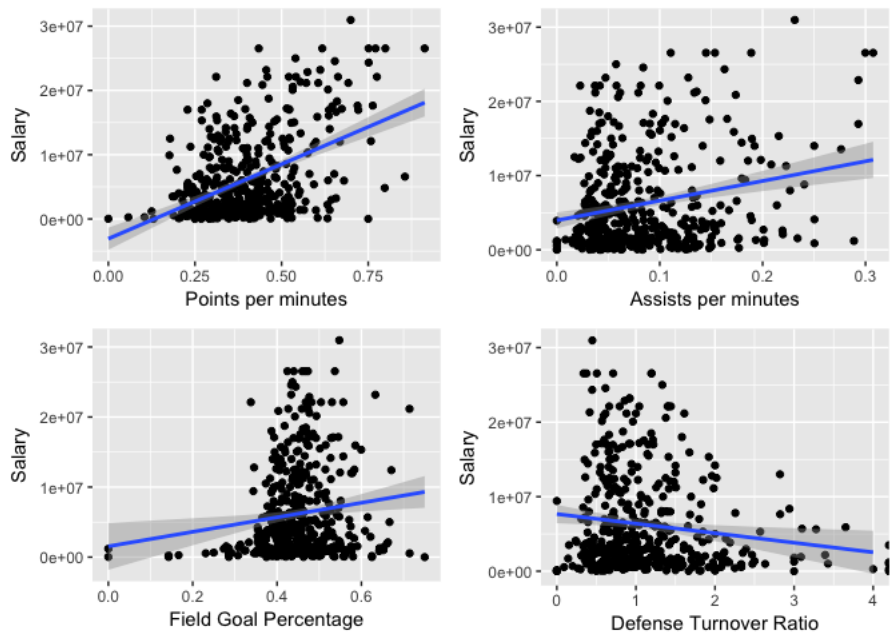
First, let's filter the data. I want to focus on salary, played minutes, rebounds, assists, field goal percentage, and def/tov rate of players to get efficiency. (FYI, I assigned def/tov rate with turnovers / (steal+block))

```r
pool <- pool %>%
  mutate(fg_pct = FGM / FGA,
         def_tov = (BLK + STL) / TO,
         PTS = Points3 * 3 + Points2 * 2 + FTM,
         PTS_MIN = PTS / MIN,
         AST_MIN = AST / MIN)
pool <- pool[, c("Player", "Salary", "PTS_MIN", "AST_MIN", "fg_pct", "def_tov")]
pool <- na.omit(pool) #excluding missing data
```

So, we have enough data set right now. We will compare four stats with salary, PTS/MIN, AST/MIN, fg_pct, def_tov by creating scatterplots.

```r
#Visualizing data with scatterplot/regression line
pm_sal <- ggplot(pool, aes(x = PTS_MIN, y = Salary, label = pool$Player)) +
  labs(x = "Points per minutes", y = "Salary") +
  geom_point() +
  geom_smooth(method = lm)

as_sal <- ggplot(pool, aes(x = AST_MIN, y = Salary, label = pool$Player)) +
  labs(x = "Assists per minutes", y = "Salary") +
  geom_point() +
  geom_smooth(method = lm)

fg_sal <- ggplot(pool, aes(x = fg_pct, y = Salary, label = pool$Player)) +
  labs(x = "Field Goal Percentage", y = "Salary") +
  geom_point() +
  geom_smooth(method = lm)

dt_sal <- ggplot(pool, aes(x = def_tov, y = Salary, label = pool$Player)) +
  labs(x = "Defense Turnover Ratio", y = "Salary") +
  geom_point() +
  geom_smooth(method = lm)

grid.arrange(pm_sal, as_sal, fg_sal, dt_sal) #putting multiple scatterplots on one page
```

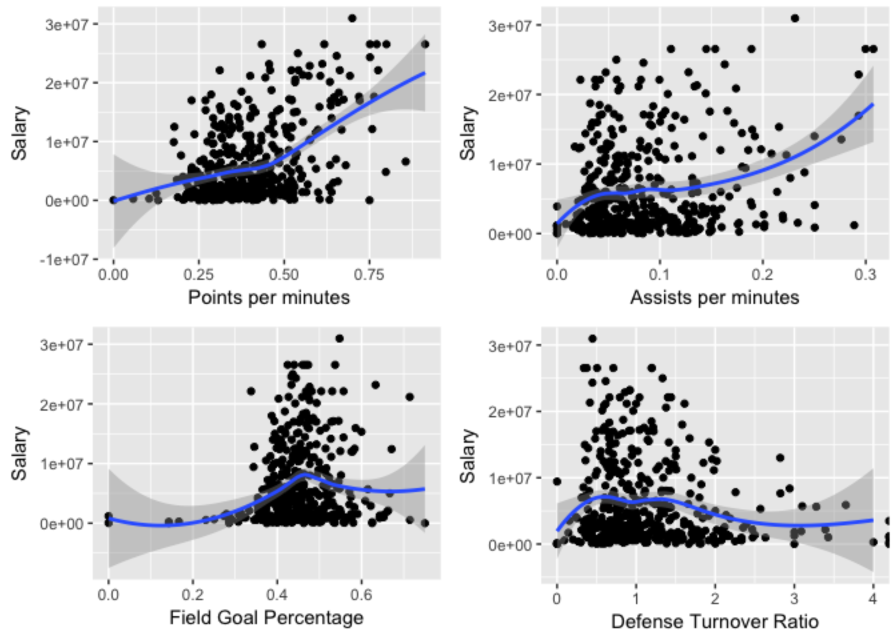
```
## Warning: Removed 5 rows containing non-finite values (stat_smooth).
```



By using regression line, it looks like scoring is the most related factor with salary. Assist, field goal percentage and defensive stats come in order.

```r
#Visualizing data with scatterplot/loess line
pm_sal_l <- ggplot(pool, aes(x = PTS_MIN, y = Salary, label = pool$Player)) +
  labs(x = "Points per minutes", y = "Salary") +
  geom_point() +
  geom_smooth()

as_sal_l <- ggplot(pool, aes(x = AST_MIN, y = Salary, label = pool$Player)) +
  labs(x = "Assists per minutes", y = "Salary") +
  geom_point() +
  geom_smooth()

fg_sal_l <- ggplot(pool, aes(x = fg_pct, y = Salary, label = pool$Player)) +
  labs(x = "Field Goal Percentage", y = "Salary") +
  geom_point() +
  geom_smooth()

dt_sal_l <- ggplot(pool, aes(x = def_tov, y = Salary, label = pool$Player)) +
  labs(x = "Defense Turnover Ratio", y = "Salary") +
  geom_point() +
  geom_smooth()

grid.arrange(pm_sal_l, as_sal_l, fg_sal_l, dt_sal_l) #putting multiple scatterplots on one page
```

```
## `geom_smooth()` using method = 'loess'
## `geom_smooth()` using method = 'loess'
## `geom_smooth()` using method = 'loess'
## `geom_smooth()` using method = 'loess'

## Warning: Removed 5 rows containing non-finite values (stat_smooth).
```



By using lowess line, it looks like scoring is the most related factor with salary. Assist, field goal percentage and defensive stats come in order, just like regression lines.

Now, let's use PCA. Remember, i> Get some raw data ii> Adjust the raw data, by subtracting the mean from each value iii> Calculate the covariance matrix with the result iv> Get the eigenvalues and eigenvectors v> Choosing components and form a feature vector vi> Get a final data, multiplying adjusted raw data and featured raw data

We already got raw data, so we will adjust the data.

```
pts_s <- scale(pool$PTS_MIN) # subtracting the mean from each value
ast_s <- scale(pool$AST_MIN)
fg_s <- scale(pool$fg_pct)
def_s <- scale(pool$def_tov)
sal_s <- scale(pool$Salary)
```

We have a problem here, def_s has non-valid value. This is because there were infinite values in def_tov column. We should clean this data before we standardize the data.

```
is.na(pool) <- sapply(pool, is.infinite) # replacing infinite value to NA
pool$def_tov[is.na(pool$def_tov)] <- 0 #change NA to 0
```

Now, check def_s again, and it works!

```
def_s <- scale(pool$def_tov)
```

Next, we are creating a matrix with these standardized data, and then make a correlation matrix.

```
pool_s <- data.frame(list(
  Player = pool$Player,
  PTS_MIN = pts_s,
  AST_MIN = ast_s,
  FG_PCT = fg_s,
  DEF = def_s,
  SAL = sal_s
  ))
corr <- cor(pool_s[,2:6]) # getting covariance matrix from the result
corr
```

```
##            PTS_MIN    AST_MIN     FG_PCT        DEF        SAL
## PTS_MIN  1.0000000  0.2974236  0.2385666 -0.2906633  0.4727841
## AST_MIN  0.2974236  1.0000000 -0.1584944 -0.3782539  0.2287975
## FG_PCT   0.2385666 -0.1584944  1.0000000  0.2358523  0.1322968
## DEF     -0.2906633 -0.3782539  0.2358523  1.0000000 -0.1066308
## SAL      0.4727841  0.2287975  0.1322968 -0.1066308  1.0000000
```

By using correlation matrix, the stats that affect salary the most are points, assists, field goal percentage, defensive stats in order. This order is just the same with the one from linear regression and lowess line. Now, we will use prcomp of PCA to have a most efficient offensive player in the league.

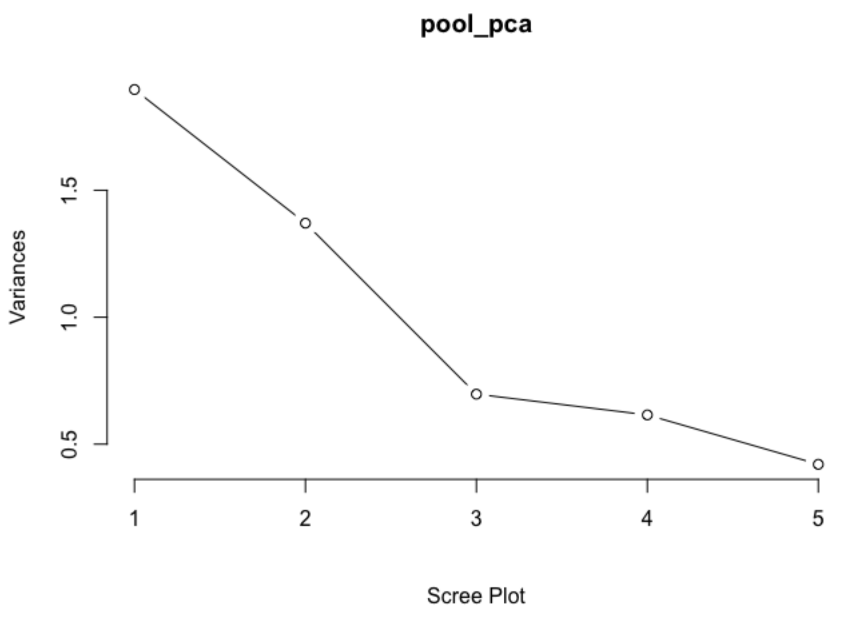
```
pool_pca <- prcomp(pool_s[,2:6])
summary(pool_pca)
```

```
## Importance of components%s:
##                           PC1    PC2    PC3    PC4     PC5
## Standard deviation     1.3772 1.1707 0.8349 0.7844 0.64853
## Proportion of Variance 0.3793 0.2741 0.1394 0.1231 0.08412
## Cumulative Proportion  0.3793 0.6534 0.7928 0.9159 1.00000
```

```
print(pool_pca)
```

```
## Standard deviations (1, .., p=5):
## [1] 1.3772056 1.1706643 0.8348863 0.7843656 0.6485253
##
## Rotation (n x k) = (5 x 5):
##                PC1        PC2        PC3        PC4         PC5
## PTS_MIN -0.56770918  0.3030869 -0.1859877  0.1876569 -0.71835787
## AST_MIN -0.49640128 -0.3028807  0.0333856 -0.8116731  0.04383189
## FG_PCT  -0.01698015  0.7196314 -0.5175557 -0.2587309  0.38345376
## DEF      0.44621800  0.4341084  0.5010409 -0.4365952 -0.41325788
## SAL     -0.48155293  0.3317863  0.6673740  0.2200332  0.40524298
```

```
 plot(pool_pca, type="l", sub = "Scree Plot")
```

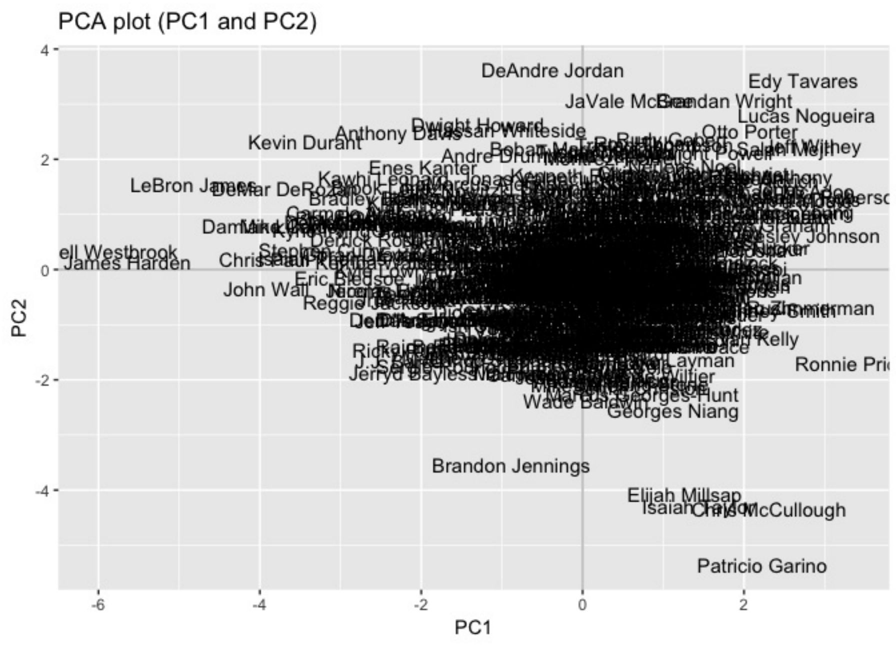


This method is called scree plot. We can see that those first 2 line of the graph are sloppy. This means these two PC1, PC2 components reduced a dimension of the data pretty much compare to PC3, PC4. So we will take PC1, PC2. From rotation of PCA, PC1 related to defensive stat, PC2 related to field goal percentage. In other words, PC1 represents players' defense, PC2 represents players' offense.

We will now draw a graph of players based PC1, PC2

```
eig_pca <- data.frame(
  eigenvalue = round(pool_pca$sdev^2,4),
  prop = round((pool_pca$sdev^2) / sum(pool_pca$sdev^2),4),
  cumprop = round(cumsum((pool_pca$sdev^2) / sum(pool_pca$sdev^2)),4)
) #getting eigenvalues and eigenvectors
```

```r
pca_df <- as.data.frame(pool_pca$x)
ggplot(data = pca_df, aes(x = PC1, y = PC2))+
  geom_hline(yintercept = 0, color = "gray") +
  geom_vline(xintercept = 0, color = "gray") +
  geom_text(label = pool_s$Player) +
  ggtitle("PCA plot (PC1 and PC2)")
```



By this plot, the most efficient offensive player is DeAndre Jordan! Adding some background information, DeAndre Jordan is a solid player but not a best offensive player because he cannot set up the game. He usually set screens and post up. We got DeAndre because we calculated these data with per minute data. If our data came from full game data, result may vary.

Conclusion

The idea of PCA is really interesting. It's interesting how we reduce dimension of high dimensional data. Even we can still have similar data from less-dimensional data. Remember, i> Get some raw data ii> Adjust the raw data, by subtracting the mean from each value iii> Calculate the covariance matrix with the result iv> Get the eigenvalues and eigenvectors v> Choosing components and form a feature vector vi> Get a final data, multiplying adjusted raw data and featured raw data

As long as you follow these steps, you cannot go that wrong with PCA.

Source

1> https://tgmstat.wordpress.com/2013/11/21/introduction-to-principal-component-analysis-pca/

2> http://www.statisticshowto.com/dimensionality/

3> http://www.statisticssolutions.com/multicollinearity/

4> https://www.statmethods.net/input/missingdata.html

5> http://rstudio-pubs-static.s3.amazonaws.com/2852_379274d7c5734f979e106dcf019ec46c.html

6> https://www.youtube.com/watch?v=a_vCQCOjdpk

7> http://rstudio-pubs-static.s3.amazonaws.com/27823_dbc155ba66444eae9eb0a6bacb36824f.html