Principal Component Analysis: Theory, Visualization, and Application

1. Introduction

Principal Component Analysis (PCA), a statistical method based on orthogonal transformation and singular value decomposition, is a powerful tool to reduce the dimension of the data while capturing the most of its variations. We have learned some general mathematical ideas behind PCA decomposition so far in class, and we also did a PCA ranking of the NBA teams in one of our homework assignments.

Since PCA is such a useful technique, the primary motivation is to inform the audience what it is about and how it can be visualized and applied. The contents of this post include the mathematical theory behind PCA decomposition, the different ways to visualize the results (including screeplots and biplots), as well as some examples of its applications to different fields of science.

2. Mathematical Theory behind PCA Decomposition

The general idea behind PCA is simple. It is equivalent to a change of basis problem in linear algebra, where we select a new coordinate system (a new set of basis vectors) such that the greatest variation of the original data is projected onto the first coordinate axis (PC1), the second greatest variation onto the second coordinate axis (PC2), and so on. We therefore want a new coordinate system that better represents our data.

Mathematically speaking, suppose that we have a matrix X with m objects and n variables, and we have a linear transformation T that acts on $M_{m\times n(F)}$ for some field F. Define $Y = TX$ in the standard basis e. Then

$$y_i = \sum_i a_i T(x_i)$$

for some coefficients a_i, where y_i represents the i^{th} column of Y. In other words, each column of Y is a linear combination of the columns of X for some coefficients.

Next, we want to choose an appropriate X. Since we are most interested in capturing the variations of our data, a natural choice would be to use a covariance matrix. In order to convert x into a square matrix, we define

$$S_X = \frac{1}{n-1} XX^T$$

where each entry of S_X corresponds to the correlation between an object and a variable.

Finally, we want to choose a new basis for T that best represents S_X. We do so simply by diagonalizing S_X. The eigenvectors that we get after diagonalization will constitute the new basis β such that $[S_X]_\beta$ is a diagonal matrix. In other words, Vectors in β are the principal components that we seek. The corresponding eigenvalues are the contributions of each variable to the principal components (the weights a_i in the first equation).

3. Visualization of PCA Results

we have seen in class that it is easy to generate a histogram based on the rank of PC1. However, that doesn't tell us much information since it only includes one principal component. I present some other ways to visualize PCA results in this post.

Let's use the NBA team example in homework-3:

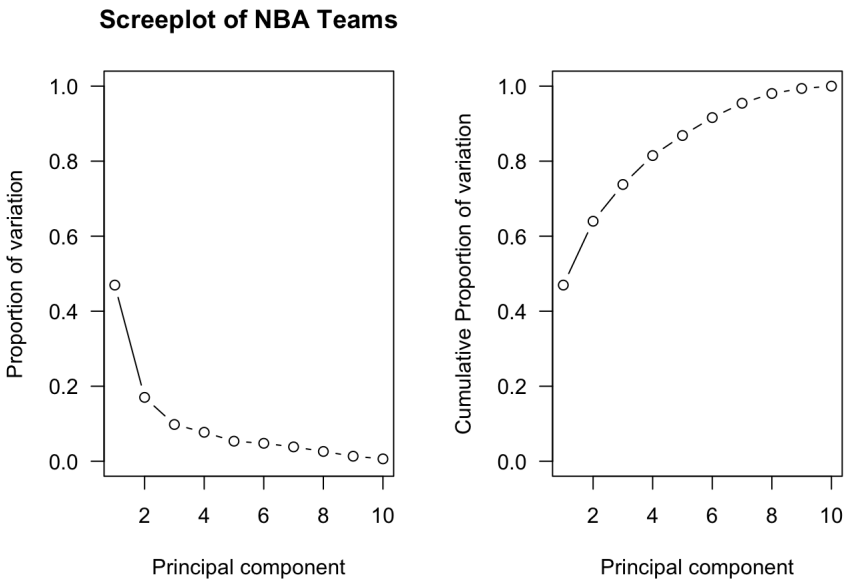
```
library(readr)
library(dplyr)
library(ggplot2)
teams <- read.csv(file = "data/nba2017-teams.csv")  #load data file of teams
```

Screeplot

The first thing that we would like to know is how much variations each principal component explains. As a rule of thumb, we want to consider all principal components up to which 90% of variations are captured. The easiest way to do this is by making a screeplot:

```r
# Creates a function screeplot() whose input is a prcomp() result
# This function is taken from
# http://rstudio-pubs-static.s3.amazonaws.com/27823_dbc155ba66444eae9eb0a6bacb36824f.html
# with some modifications

screeplot <- function(x) {

    # calculate the variation based on standard deviation
    x.var <- x$sdev ^ 2

    # the proportion of variation each PC explains
    x.pvar <- x.var/sum(x.var)

    print("proportions of variance:")
    print(x.pvar)

    # creates a matrix that combines the plots
    par(mfrow = c(1,2))

    # plot the proportion of variation each PC explains
    plot(x.pvar, xlab = "Principal component", ylab = "Proportion of variation",
         main = "Screeplot of NBA Teams", ylim = c(0,1), type = 'b', las = 1)

    # plot the cumulative variation
    plot(cumsum(x.pvar), xlab = "Principal component", ylab = "Cumulative Proportion of variation",
         ylim = c(0,1), type = 'b', las = 1)
}

# use prcomp() to perform the PCA analysis
teams_select <- teams %>%
  select(points3, points2, free_throws, off_rebounds, def_rebounds,
         assists, steals, blocks, turnovers, fouls)
pca_team <- prcomp(teams_select, scale. = T)

screeplot(pca_team)
```

```
## [1] "proportions of variance:"
##  [1] 0.469588631 0.170201009 0.097952464 0.077171938 0.053408824
##  [6] 0.047801622 0.038220374 0.026026243 0.013359274 0.006269622
```



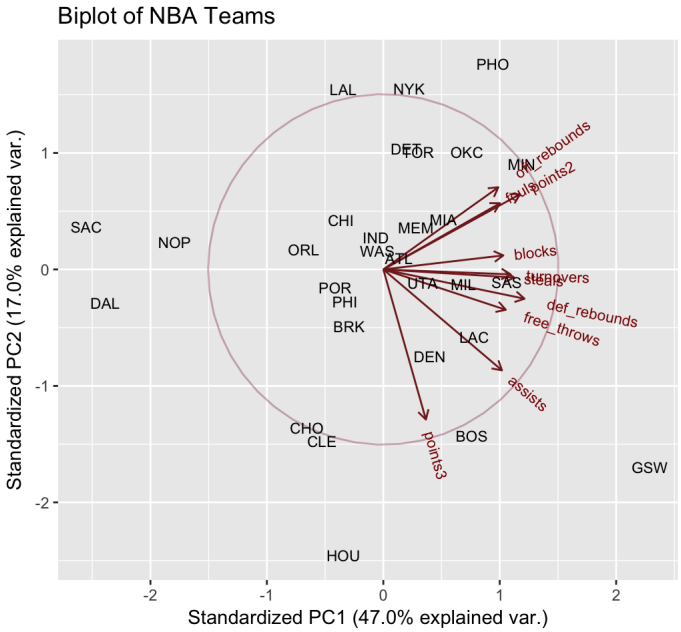
Each dot in this plot represents a principal component. The first plot shows how much variations each PC explains, and the second plot shows the cumulative variations explained up to each PC. The shapes of the two plots don't have to look the same.

Based on these two plots, it is clear that the proportion of variations explained decreases significantly at PC2. The cumulative variation reaches the 90% mark at PC6. This is generally to be avoided because we need to take into account of six PC's. In most cases, we want at most three PC's to explain over 90% of the variations. In this case, what we can do is to modify our chosen variables or add/delete some of them, so that they represent our data in a better way.

Biplot

In addition to making a screetplot, we would also like to visualize how the objects and variables are related. A good way to do this is by making a biplot, which plots the position of objects relative to the PC's, as well as the correlations of variables. The function that does this is called "ggbiplot", and is shown in https://github.com/vqv/ggbiplot/blob/master/R/ggbiplot.r. Since this function is quite long, I've decided to hide this code chunk in my final knitted document. If you are interested, you are welcomed to go on this website and see how the function works.

```
# The parameter circle determines whether to draw a circle around the arrows

ggbiplot(pca_team, labels = teams$team, circle = T)
```



The x-axis of this plot is the first principal component, and the y-axis is the second principal component. The arrows represent the variables (free throws, blocks, turnovers, etc), and each team has a unique coordinate based on the two PC's. One way to think about such a biplot is to treat each arrow as a vector. Then each object is represented as the linear combination of those vectors. Since the variables are all standardized, the origin represents the average of all statistics. For instance, Houston Rockets (HOU) has the lowest y-value, which means that it makes a lot of three-point shots each game. Golden State Warriors occupies the south-east cornor, which means that it makes many assists. In this way, we can easily identify the relationship between each object and all the variables.

Moreover, the rank of PC1 that we see in class is simply the projection of this biplot onto the x-axis. The circle in the graph, along with the length of the arrows, shows how well a variable is explained. For example, the tip of the points3 arrow is very close to the circle, which means that it is well-explained by the first two PCs. On the contrary, variables such as blocks and free throws are not quite well-explained.

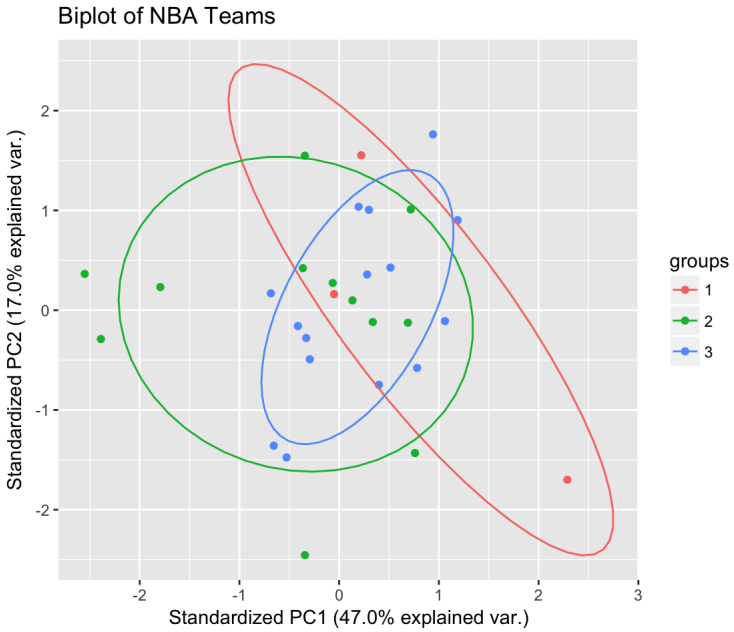
Cluster plotting

To better appreciate the power of biplots, we can group the teams by some criteria and plot them in clusters using different colors. For instance, I can group the teams by total salary and see how that affects the distribution. Just for the sake of illustration, I arbitrarily group salary into 3 groups: less than 80 million, greater than 80 but less than 95 million, and greater than 95 million.

```
# replicates the data frame
teams_salary = teams

# loops over all 30 teams
for (i in 1:30) {
  if (teams_salary[i,4] > 95) {    # salary is at the 4th column
    teams_salary[i,4] = "1"    # change salary > 80 to group 1
  }
  else if (teams_salary[i,4] < 95 & teams_salary[i,4] > 80) {
    teams_salary[i,4] = "2"    # cahnge 60 < salary < 80 to group 2
  }
  else if (teams_salary[i,4] < 80) {
    teams_salary[i,4] = "3"    # change salary < 60 to group 3
  }
}
```

Next, we would like to generate a biplot of the teams grouped by salary.

```
# The parameter ellipse generates ellipses based on groups
# The parameter var.axes determines whether include arrows for variables.

ggbiplot(pca_team, groups = teams_salary$salary, ellipse = T, var.axes = F)
```



Each group correspond to a range of salary (as defined in the function). Each ellipse in this plot can be understood as a two-dimensional “best-fit line” to all the points in that group. The more they overlap, the less differences there are between the groups.

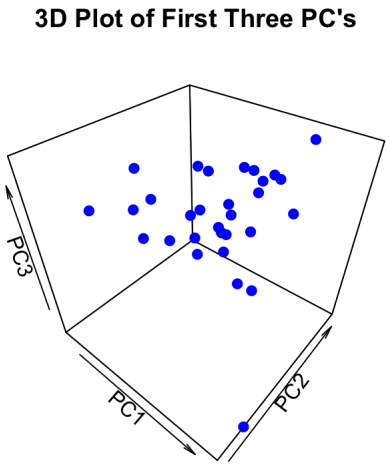
This biplot may seem a bit weird since the ellipses overlap extensively with each other. This means that salary does not really impact the distribution of teams based on the variables we’ve chosen.

However, if the data set is large enough, it is possible that we get ellipses with little or no overlap. This means that each group has distinct properties and that we can analyze the objects based on which groups they are in.

3D plotting

The above plotting techniques can also be extended to three dimensions, where we take into account three PCs instead of two. This is needed when the first two PC’s do not explain most of the variations, as in our case. To do this, we can simply use the scatter3D() function in the “plot3D” package:

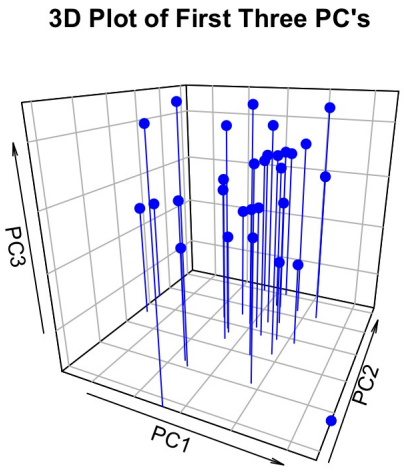
```
library(plot3D)

# generates the 3D plot. PC1 is the first column of pca_team$x. PC2 is the second column, and PC3 is the third col
umn.
# the colvar parameter determines whether the dots should be colored by groups
# the pch parameter determines the type of dot used

scatter3D(x = pca_team$x[,1], y = pca_team$x[,2], z = pca_team$x[,3], colvar = NULL, col = "blue",
  pch = 19, xlab = "PC1", ylab = "PC2", zlab = "PC3",
  main = "3D Plot of First Three PC's")
```



Each dot in this plot represents a team with some PC1, PC2, and PC3, as labeled by the axes. Its position determines how well it is explained by each of the PC’s. Imagine this as a three-dimensional biplot, where now we introduce a third dimension of PC3 (the z-axis).

We can further modify the plot so that it looks more informative.

```
# The parameters phi and theta determines the angle from which you view the plot
# type= "h"" generates a projection of each point onto the xy-plane
# bty = "b2" makes grid lines on each plane.

scatter3D(x = pca_team$x[,1], y = pca_team$x[,2], z = pca_team$x[,3], colvar = NULL, col = "blue",
  pch = 19, xlab = "PC1", ylab = "PC2", zlab = "PC3", main = "3D Plot of First Three PC's", phi = 20, bty = "b2",
theta = 25, type = "h")
```



The vertical lines in this plot are the projection of each point onto the xy-plane. This is actually the same as the biplot that we plotted above. Notice the the point in the lower-right corner does not seem to have a vertical line. This means that it is relatively poorly explained by PC3. In general, such a 3D plot is harder to visually interpret than 2D plots, which is why often when we need more than two PC's, we plot them by pairs in separate biplots.

4. Applications

Here I would like to briefly talk about some potential applications of PCA in other disciplines. As I've mentioned in the first section, the primary goal of PCA is to reduce the dimension of data, and discard those dimensions that do not contribute to the variations. This is most useful when the dimension of data is extremely large.

For instance, in the field of biological sciences, suppose that we are interested in some property of cells. Most likely there are many variables that might influence that property: temperature, moisture, nutrition, density, etc. Sometimes it is very hard to control for all those variables, and sometimes we might be interested in the cross-effects of them. In these cases, PCA decomposition can be handy because it tells us what variables have significant influences and what variables don't. In addition, we might want to group the cells based on their types, plot the results, and see if the groups differ significantly. Hence PCA can provide helpful insight in this scenario.

PCA decomposition can also be built into physical models. One article published in Nature discusses the possibility of quantum PCA. The spectural measurements can be decomposed in a similar manner, and the results provide useful insight to the quantum states. However, this topic is much beyond my knowledge, and I mention it to show that PCA can be used in many advanced settings.

5. Take Home Message

PCA is a technique to reduce the dimension of data while capturing most of its variations. The results can be easily visualized and can be applied to many advanced fields of science.

6. References

https://github.com/vqv/ggbiplot/blob/master/R/ggbiplot.r

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

https://www.cs.princeton.edu/picasso/mats/PCA-Tutorial-Intuition_jp.pdf

https://cran.r-project.org/web/packages/ggfortify/vignettes/plot_pca.html

http://www.sthda.com/english/wiki/ggfortify-extension-to-ggplot2-to-handle-some-popular-packages-r-software-and-data-visualization#plotting-cluster-package

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3298499/

http://www.nature.com/nphys/journal/v10/n9/full/nphys3029.html

http://www.sthda.com/english/wiki/impressive-package-for-3d-and-4d-graph-r-software-and-data-visualization