STAT 154 Lab 3: Principal Component Analysis

Yuansi Chen and Raaz Dwivedi

Feb 4, 2019

The goal of this lab is to go over the various options and steps required to perform a Principal Components Analysis (PCA). You will also learn about the functions **prcomp()** and **princomp()**.

```r
library(ggplot2)
```

Remark: You need to set the eval=TRUE on line 8 of the Rnw file to Run the code when you compile the pdf. No change is needed if you only run code chunks.

1 Relationship between PCA and SVD

Given a matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$. Suppose SVD gives that

$$X = UDV^\top.$$

The minimal output of a PCA procedure should consists of eigenvalues, principal vectors, and principal components.

1. How do you find eigenvalues of $X^\top X$ from D?
2. How do you find principal vectors from U or V?
3. How do you find the principal components (or scores) directly from U, D or V?

2 PCA in Action

We now see PCA in action on NBA dataset using different methods in R and our own understanding from the previous question. The primary goal in PCA is to summarize the systematic patterns of variation in a data set, which is done via the principal components (PCs). Derived from this idea, the most common uses of the PCA results is to visualize multivariate data and/or to perform a dimension reduction (for other analytical purposes).

First we try to understand the dataset, and then we build on our understanding of PCA to get a better insight of the dataset.

2.1 NBA teams dataset

We will use the data set about NBA teams, containing statistics per game during the regular season 2016-2017. The data is stored in a csv file.

1. Spend some time explore the basic structure of the dataset, start with the 'View' function followed by: (a) descriptive statistics with **summary()**, (b) univariate plots: boxplots, histograms, density curves, (c) compute the correlation matrix, (d) get a scatterplot matrix with **pairs()**. Discuss your observations (e.g. pattern, trend, variability).

```
dataset <- read.csv('nba-teams-2017.csv', stringsAsFactors = FALSE)
# uncomment get some info
# str(dataset, vec.len = 1)

# To do, use boxplot, density, summary etc.
```

2. Create a new data frame **dat** that contains the following columns

```
variables <- c(
  'wins',
  'losses',
  'points',
  'field_goals',
  'points3',
  'free_throws',
  'off_rebounds',
  'def_rebounds',
  'assists',
  'steals',
  'blocks',
  'personal_fouls')

dat <- dataset[ ,variables]
```

2.2 PCA in R

R provides two built-in functions to perform Principal Components Analysis: (a) **prcomp()** and (b) **princomp()**. Both functions are part of the **stats** package which comes in the default distribution of R. We first discuss the package**prcomp()**.

The main input of **prcomp()** is a data frame or a data matrix. In order to perform PCA on standardized data (mean = 0, variance = 1), we use the argument "scale. = TRUE". The object ‘pca_prcomp’ is an object of class **prcomp**, basically a list that contains the following results:

(a) **sdev**: corresponds to the standard deviations of the principal components (i.e. the square roots of the eigenvalues of $\frac{1}{n-1}\mathbf{X}^\top\mathbf{X}$).

(b) **rotation** is the rotation matrix or loadings (i.e. eigenvectors of $\frac{1}{n-1}\mathbf{X}^\top\mathbf{X}$)

(c) **center** is the vector of means of the raw data (i.e. the centroid).

(d) **scale** is the vector of standard deviations of the raw data.

(e) **x** is the matrix of principal components (aka scores).

```r
pca_prcomp <- prcomp(dat, scale. = TRUE)
```

2.2.1 Basic Manipulations

Create the following objects:

(i) ‘eigenvalues’: vector of eigenvalues of $\frac{1}{n-1}\mathbf{X}^\top\mathbf{X}$ (i.e. $\lambda_1, \lambda_2, \dots$)

(ii) ‘loadings’: matrix of eigenvectors (i.e. $\mathbf{V}$)

(iii) ‘scores’: matrix of principal components (i.e. $\mathbf{Z} = \mathbf{XV}$)

Questions: How many eigenvalues are almost zero (or zero)? What about the loading associated to the 12th PC? What about the 12th PC score? Can you guess what’s going on with the values of the 12th dimension?

```r
# prcomp
# To do
# start with ?prcomp
```

2.2.2 Proportion of Variance Explained and Screeplot

The primary goal in PCA is to summarize the systematic patterns of variation in a data set, which is done via the principal components (PCs). Derived from this idea, the most common uses of the PCA results is to visualize multivariate data and/or to perform a dimension reduction (for other analytical purposes).

Scree plot: The first step when examining the results of a PCA is to look at how much variability is captured by each PC. This is done by examining the eigenvalues. A Scree Plot is a simple line segment plot that shows the fraction of total variance in the data as explained or represented by the first PCs.

Questions: Create such a plot for our data. How much of the variation in the data is captured by the first PC? By the second PC? By both of them together?

```
# scree_plot

# TO DO
#
# may be useful for plotting
# ggplot() + geom_point(aes(x = 1:length(eigenvalues), y=eigs_cum)) +
#   labs(x = "first PCs", y = "fraction of total variance explained")
```

2.2.3 Strategies for choosing the number of PCs

Choosing the number of PCs is largely based on a judgement call. Try the strategies below and think which one is more appropriate.

1. Retain just enough components to explain some specified, large percentage of the total variation of the original variables. For example, how many PCs would you retain to capture 70% of the total variation?
2. Exclude those PCs whose eigenvalues are less than the average: $\sum_{i=1}^{p} \lambda_i / p$. Using this criterion, how many PCs would you retain?
3. **Prove** that the average eigenvalue is one when the PCs are extracted from the correlation matrix (like in this case). Note that the rule to drop all components with eigenvalues under 1 is also known as Kaiser's Rule.
4. Just choose the first two or three PCs and show the total variation.

   ```
   # TO DO
   ```

2.2.4 Visualization with PCA

PCA is a powerful for EDA and can help identify useful low dimensional patterns in high-dimensional data. It also often provides a good visual summary of the data in two dimensions. This is often achieved using scatterplots with first few principal loadings and components.

1. **Visualizing the principal components/scores:** First we visualize the data in low dimensions:

(i) Visualize the projection scores on first two PCs. Label the points using the team names.

```
# scatterplot_2PC
# Make changes to the following code to get the plot
ggplot() + geom_point(aes(x = scores[, 1], y=scores[, 2])) +
  geom_text(aes(x = scores[, 1], y=scores[, 2], label=dataset$team)) +
  labs(x = "PC1", y = "PC2")
```

(ii) You may also want to see the plot PC1 - PC3, and then plot PC2 - PC3. If you want, continue visualizing other scatterplots. Do you notice any patterns?

```
# scatterplot_otherPCs

# To Do
```

(iii) Use **plotly** package to generate a 3D plot of the first three PCs.

```
# plotly3d
# might only be visible on R studio
# library(plotly)

# To Do
```

2. **Visualizing principal vectors/loadings:** The next stage consists of examining how principal vectors are formed. Recall that principal vectors are linear combinations of the input variables, in which the coefficients of such linear combinations are given by the loadings.The larger the loading of a variable in a given PC, the more associated the variable is with that PC.

   (i) Visualize the loadings of first two PCs, label the points using the feature (column) names.

```
# To Do
```

   (ii) Another way to examine how variables are associated with the PCs is to look at their correlations. How correlated is wins with PC1?

```
# variables_vs_PC

# To Do

# Use cor
```

3. Scroll to the end of the help menu for the princomp and notice the usage of biplot with it. Use the function biplot to produce the figures from the previous two parts on the same figure.

```
#
# start with ?prcomp
#
# To Do
```

2.3 Doing PCA manually using SVD

We will now see that we can obtain same results using our understanding of the relation between SVD and PCA from the previous question.

1. Scale the data 'dat' using the scale function and then compute its svd. Compare the 'v' vectors from svd with the loadings from the previous parts.

```
# To Do
```

2. Use the svd output to project the data onto the first two right singular vectors and scatter plot the projection scores and label the points with the corresponding team names.

```
# Hint: First two columns of XV
```

3. Reproduce the previous plot by using just the first two left singular vectors and the corresponding singular values.

```
# To Do

# Hint: first two columns of UD
```

4. Discuss the similarities between the projection plots that you have created.

2.4 Interpretations of PCA

Discuss the two different interpretations of PCA: (a) Maximum variance direction, and, (b) best representation of data.

2.5 PCA with princomp()

We now discuss another function of R that can be used to perform PCA is **princomp()**. The main input is a data frame or a data matrix. In order to perform PCA on standardized data (mean = 0, variance = 1), we use the argument **cor = TRUE**, which means that the analysis is performed using the correlation matrix.

```r
pca_princomp <- princomp(dat, cor = TRUE)
```

The object 'pca_princomp' is an object of class **princomp**, basically a list that contains various results:

(a) **sdev** corresponds to the standard deviations of the principal components.

(b) **loadings**: object of class loadings (or principal vectors)

(c) **center**: vector of variable means of the raw data.

(d) **scale**: vector of standard deviations of the raw data.

(e) **n.obs**: number of observations in the data.

(f) **scores**: matrix of principal components.

(g) **call**: function call.

Answer the following:

(i) Compare the results of **prcomp()** against those of **princomp()** in terms of eigenvalues, principal vectors (loadings), and scores.

```r
# princomp
# names(pca_princomp)
# To Do
```

(ii) What are the differences between **prcomp()** and **princomp()**?

(iii) Spend some time reading the help documentation of both functions to find out the main differences between them.

(iv) Are there any cases when it would be better to use one function or the other?