How and When to Use Principal Component Analysis in R

Anjali Nadeswaran

10/30/2017

Introduction

Principal component analysis (PCA) is a multivariate technique that is used to emphasize variation and identify key patterns in a data table. In effect, it identifies the significant information in a data table and presents it as a set of new variables referred to as principal components.

This is especially important for visualization purposes, as patterns of similarity of the observations and the variables can be displayed as points on a graph.

PCA is often used as a dimensionality-reduction technique. When looking at graphs in more than two dimensions, it can become quite difficult to make observations about a cloud of data. PCA allows us to transform the data and project it into a lower dimension.

Historical Background

Principal component analysis was invented by Karl Pearson in 1901. Pearson developed PCA as an analog of the principal axis theorem, and it was later expanded on and named by Harold Hotelling in the 1930s.

The method is referred to by many different names depending on the field of application. For example, it is called the discrete Karhunen–Loève transform (KLT) in signal processing, the Hotelling transform in multivariate quality control, and empirical modal analysis in structural dynamics.

Background Mathematics

Variance is a measure of the spread between numbers in a data set. The formula for variance is:
$$s^2 = \dfrac{\sum_{i=1}^n (X_i - \bar{X})^2}{(n - 1)}$$

Covariance is a measure of the joint variability of two or more sets of random variables. The formula for covariance is: $$cov(X, Y) = \dfrac{\sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y})}{(n - 1)}$$

Eigenvalues are a set of scalars associated with a linear system of equations.

PCA in R

There are many methods of principal component analysis in R. For our purposes, however, we will stick to breaking down the function **prcomp**, a function from the R base package.

The function **prcomp** is mathematically based on the algorithm for singular value decomposition. Using SVD, we can break a matrix X of size $n x p$ into three components, $$X = U\Sigma V^T$$ where U is the matrix with the eigenvectors of XX^T, Σ is the diagonal matrix with the singular values and V^T is the matrix with the eigenvectors of X^TX of sizes $n x n$, $n x p$, and $p x p$, respectively.

Principal component analysis reduces the p dimensions of the data set X to k principal components. By multiplying the first k columns of U with the $k x k$ upper-left submatrix of Σ, we get the scores from the first k PCs. The loading factors of the kth PC are given in the kth row in V^T. Therefore, multiplying the scores by the loadings results in the matrix X.

We can use both **prcomp** and **svd** to obtain these values. We can check that this is true with an example.

```r
# generate a 5 x 6 matrix with random numbers
set.seed(123)
matrix <- scale(matrix(rnorm(30), 5, 6))

# perform PCA
PCA <- prcomp(matrix, scale. = FALSE, center = FALSE)
PCA$rotation # loadings
```

```
##             PC1        PC2        PC3         PC4         PC5
## [1,] -0.4239130 -0.4285286  0.2387928  0.21637888  0.07706823
## [2,]  0.4889788  0.1547436  0.4238470 -0.43690236  0.51830913
## [3,]  0.4066735 -0.3409601  0.6345803  0.38003792 -0.32907430
## [4,]  0.4978883  0.2243717 -0.3134427  0.09473586 -0.53532847
## [5,] -0.1002247  0.6771571  0.1904181  0.65391033  0.25364626
## [6,] -0.3973366  0.4089982  0.4755356 -0.42578643 -0.51595857
```

```r
PCA$x # scores
```

```
##             PC1        PC2        PC3         PC4           PC5
## [1,]  2.9000254 -0.7561224  0.1786175 -0.30992957 -2.220446e-16
## [2,]  0.1537815  1.6467590  0.6685952  0.41007382 -1.665335e-15
## [3,] -1.8996452 -1.7515494  0.4501920  0.09176618 -3.747003e-16
## [4,]  0.1983451 -0.1440491 -0.9803363  0.48393067  6.661338e-16
## [5,] -1.3525069  1.0049620 -0.3170684 -0.67584111  1.554312e-15
```

By default, the **prcomp** and **svd** functions will both retrieve $min(n, p)$ PCs. In this example, we expect to have 5 PCs.

```
# perform SVD
SVD <- svd(matrix)
SVD # the diagonal of Sigma SVD$d is given as a vector
```

```
## $d
## [1] 3.729754e+00 2.717013e+00 1.320299e+00 9.816252e-01 3.386514e-16
##
## $u
##             [,1]        [,2]       [,3]        [,4]       [,5]
## [1,]  0.77753802 -0.27829176  0.1352857 -0.31573107 -0.4472136
## [2,]  0.04123101  0.60609165  0.5063970  0.41774990 -0.4472136
## [3,] -0.50932186 -0.64465991  0.3409774  0.09348393 -0.4472136
## [4,]  0.05317914 -0.05301746 -0.7425111  0.49298926 -0.4472136
## [5,] -0.36262631  0.36987749 -0.2401490 -0.68849202 -0.4472136
##
## $v
##            [,1]       [,2]       [,3]        [,4]        [,5]
## [1,] -0.4239130 -0.4285286  0.2387928  0.21637888  0.07706823
## [2,]  0.4889788  0.1547436  0.4238470 -0.43690236  0.51830913
## [3,]  0.4066735 -0.3409601  0.6345803  0.38003792 -0.32907430
## [4,]  0.4978883  0.2243717 -0.3134427  0.09473586 -0.53532847
## [5,] -0.1002247  0.6771571  0.1904181  0.65391033  0.25364626
## [6,] -0.3973366  0.4089982  0.4755356 -0.42578643 -0.51595857
```

```
sigma <- matrix(0, 5, 5) # generate an empty 5 x 5 matrix
diag(sigma) <- SVD$d # store the values of the diagonal of Sigma SVD$d in the new matrix
```

Now that we have created the PCA and SVD objects, we can compare the resulting scores and loadings. We will compare the scores from the PCA with the product of \(U\) and \(\Sigma\) from the SVD and the loadings from the PCA with \(V\) from the SVD.

```
# compare PCA scores with the SVD's U*Sigma
scores <- SVD$u %*% sigma
round(PCA$x, 5) == round(scores, 5)
```

```
##       PC1  PC2  PC3  PC4  PC5
## [1,] TRUE TRUE TRUE TRUE TRUE
## [2,] TRUE TRUE TRUE TRUE TRUE
## [3,] TRUE TRUE TRUE TRUE TRUE
## [4,] TRUE TRUE TRUE TRUE TRUE
## [5,] TRUE TRUE TRUE TRUE TRUE
```

```
# compare PCA loadings with the SVD's V
round(PCA$rotation, 5) == round(SVD$v, 5)
```

```
##       PC1  PC2  PC3  PC4  PC5
## [1,] TRUE TRUE TRUE TRUE TRUE
## [2,] TRUE TRUE TRUE TRUE TRUE
## [3,] TRUE TRUE TRUE TRUE TRUE
## [4,] TRUE TRUE TRUE TRUE TRUE
## [5,] TRUE TRUE TRUE TRUE TRUE
## [6,] TRUE TRUE TRUE TRUE TRUE
```

The values we obtained using the **prcomp** and **svd** functions are the same.

Examples

Let’s take a look at the data set “Iris” to see how we can calculate and visualize PCA in R.

```
# load packages
library(ggplot2)
# load the data set
iris <- read.table('http://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data', sep = ',', col.names
= c('sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'species'))
#inspect data
dim(iris)
```

```
## [1] 150   5
```

```
head(iris, 5)
```

```
##   sepal_length sepal_width petal_length petal_width     species
## 1          5.1         3.5          1.4         0.2 Iris-setosa
## 2          4.9         3.0          1.4         0.2 Iris-setosa
## 3          4.7         3.2          1.3         0.2 Iris-setosa
## 4          4.6         3.1          1.5         0.2 Iris-setosa
## 5          5.0         3.6          1.4         0.2 Iris-setosa
```

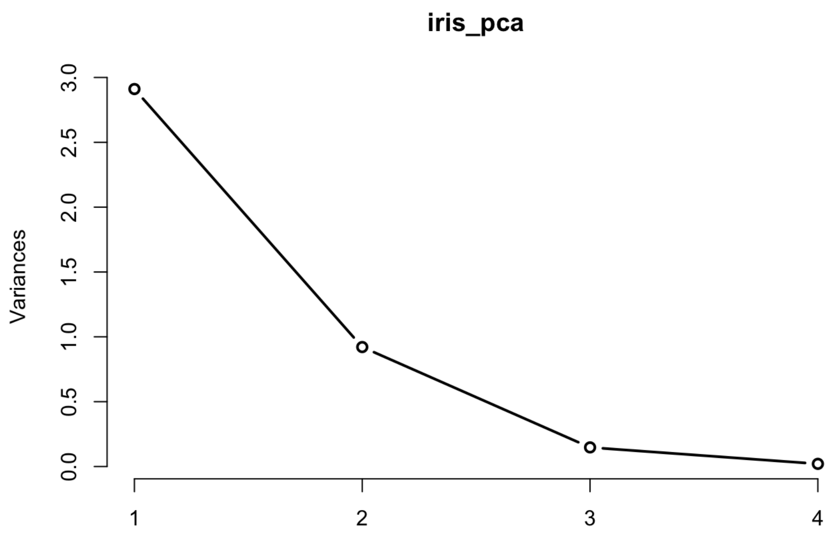
We will use the function **prcomp** to calculate the scores and loadings for the data set. If we set center and scale. equal to TRUE, then we can shift the variables to be zero centered and scaled to have unit variance.

```
# perform PCA
iris_pca <- prcomp(iris[ , 1:4], center = TRUE, scale. = TRUE)
iris_pca
```

```
## Standard deviations:
## [1] 1.7061120 0.9598025 0.3838662 0.1435538
##
## Rotation:
##                     PC1         PC2        PC3        PC4
## sepal_length  0.5223716 -0.37231836  0.7210168  0.2619956
## sepal_width  -0.2633549 -0.92555649 -0.2420329 -0.1241348
## petal_length  0.5812540 -0.02109478 -0.1408923 -0.8011543
## petal_width   0.5656110 -0.06541577 -0.6338014  0.5235463
```

It may be difficult to interpret the output of **prcomp**. So, it is helpful to plot the principal components to better visualize the results of our analysis.

```
# visualize variance by plotting the PCs
plot(iris_pca, type = 'l', lwd = 2)
```



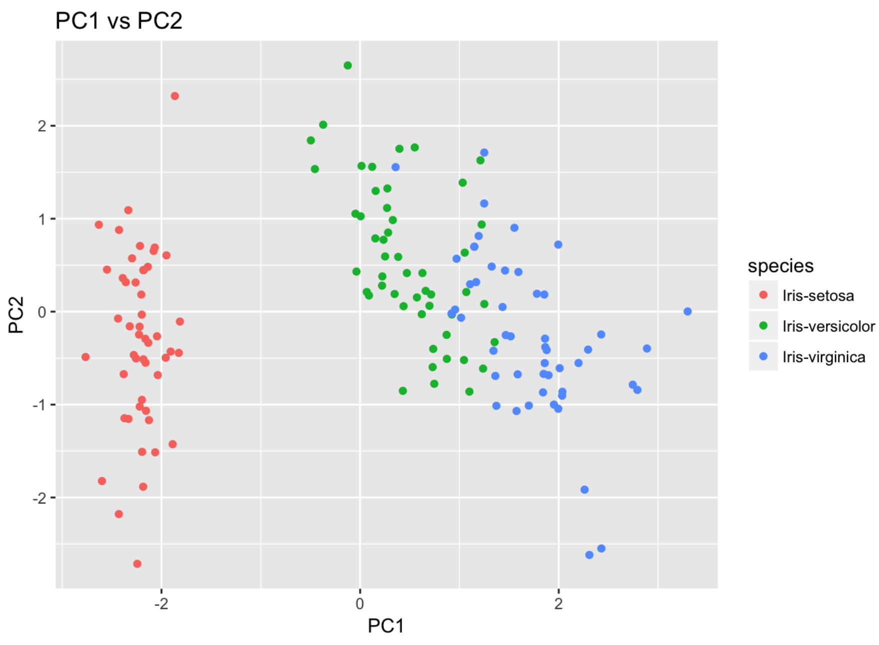
Graphing the PCs can be informative in determining the number of PCs to retain for further analysis. From this graph, we can see that the first two principal components explain most of the variability in the data; the first principal component has a variance of almost 3.0 and the second principal component has the second greatest variance at almost 1.0.

We can use PCA to create various other plots, as well.

```
# create a data frame with the principal components and species name
PCi <- data.frame(iris_pca$x, species = iris$species)
head(PCi, 5)
```

```
##         PC1        PC2         PC3         PC4     species
## 1 -2.256981 -0.5040154  0.12153619  0.02299628 Iris-setosa
## 2 -2.079459  0.6532164  0.22649206  0.10286364 Iris-setosa
## 3 -2.360044  0.3174139 -0.05130774  0.02773232 Iris-setosa
## 4 -2.296504  0.5734466 -0.09853036 -0.06609005 Iris-setosa
## 5 -2.380802 -0.6725144 -0.02135630 -0.03727242 Iris-setosa
```

```
ggplot(PCi, aes(x = PC1, y = PC2, col = species)) +
  geom_point() +
  ggtitle('PC1 vs PC2')
```



This basic graph is beneficial in that it allows us to see the amount of variation explained by each PC. PC1 is graphed along the x-axis and PC2 along the y-axis. From this plot, we can see that PC1 best explains variance; observations with class setosa all have negative values, while versicolor is mostly between -1 and 1, and virginica is primarily between 1 and 2. PC2 is the next best explanatory variable, though significantly less so than PC1. If we distinguish the different species by color, we can also see how the points in each class form elliptical shapes. Red represents the setosa class, green the versicolor, and blue the virginica.

If we wish to elaborate on this graph, we can call the function **ggbiplot**, downloaded from the github package "ggbiplot".

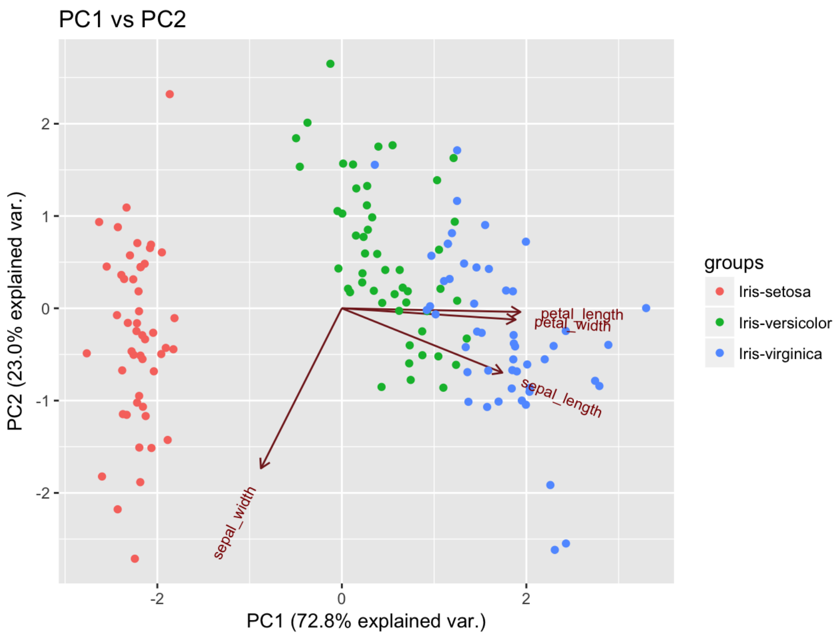
```
library(ggbiplot) # load the package ggbiplot
```

```
## Loading required package: plyr
```

```
## Loading required package: scales
```

```
## Loading required package: grid
```

```
ggbiplot(iris_pca, obs.scale = 1, var.scale = 1, groups = iris$species, ellipse = FALSE, circle = FALSE, varname.s
ize = 3) +
  ggtitle('PC1 vs PC2')
```



This new graph is a more detailed version of the scatter plot that we made previously. The arrows moving outwards from the origin describe the correlation between variables. For example, we can see that petal length and petal width are highly correlated. We also see that PC1 accounts for much of the variability in petal length, petal width, and sepal length. This information is very useful if we wish to perform further analysis about the correlation between floral elements and some other variable.

If we want to use PCA for predictive purposes, we now know that PC1, alone, can be used to discern Iris setosa from the other two species and PC1 and PC2, together, can be used to successfully identify the three species.

Take Home Message

As we have seen, PCA can be expressed in many different ways and utilized for many different purposes. Principal components are useful for describing the variability in a data set and reducing the dimensionality of more complicated data.

In R, specifically, we can use the base function **prcomp** to compute the scores and loadings of a given data set. We can then visualize these results by plotting the significant principal components, and observing how they describe variability; **ggbiplot** is a great function for this task.

Ultimately, we should tailor the way that we use PCA to fit our diverse purposes. R is a helpful language for doing exactly this.

Resources

https://www.utdallas.edu/~herve/abdi-awPCA2010.pdf
http://www.gastonsanchez.com/visually-enforced/how-to/2012/06/17/PCA-in-R/
http://setosa.io/ev/principal-component-analysis/
http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf
http://www.thetrophiclink.org/some-notes-on-pca/
https://medium.com/towards-data-science/a-one-stop-shop-for-principal-component-analysis-5582fb7e0a9c
https://www.r-bloggers.com/principal-component-analysis-in-r/
http://archive.ics.uci.edu/ml/datasets/Iris