PCA on Wind Turbine Noise Data

Thileepan Paulraj

20 joulukuuta 2018

UNDERSTANDING PCA (work reproduced from here:https://goo.gl/Wgeieb)

In any real life data set, often variables vary with each other and some of the variation in one variable is actually duplicated in the other. Principal component Analysis (PCA) is a technique in which numeric variables covary.

How does PCA reduce dimension of data?

PCA combines multiple features in the data set into a smaller set of features which are weighted linear combinations of the original set of features. These smaller set of features are called principal components and they represent most of the variability present in the full set of features. Since we have reduced the number of features, we have effectively reduced the dimension of the data. Maximum variability is captured when the mean squared error between the original dataset and it's weighted linear combination (Principal Component) is minimum.

Reading data

Let's use a real data set that I have. I have one full year of environmental noise data from a wind farm in Finland and I'm going to use only 0.1% of that entire data set because it's really huge.

The actual data has about 71 features and I'll randomly choose only 10 of them.

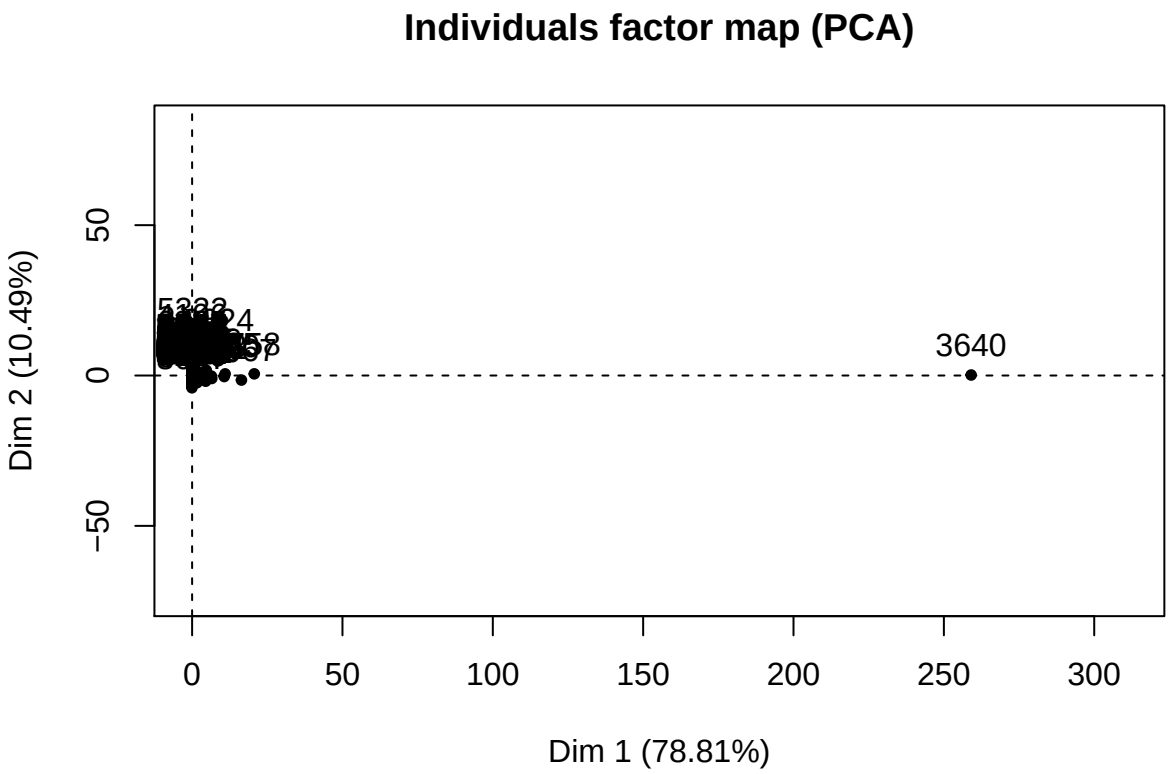
I will also scale and center the data.

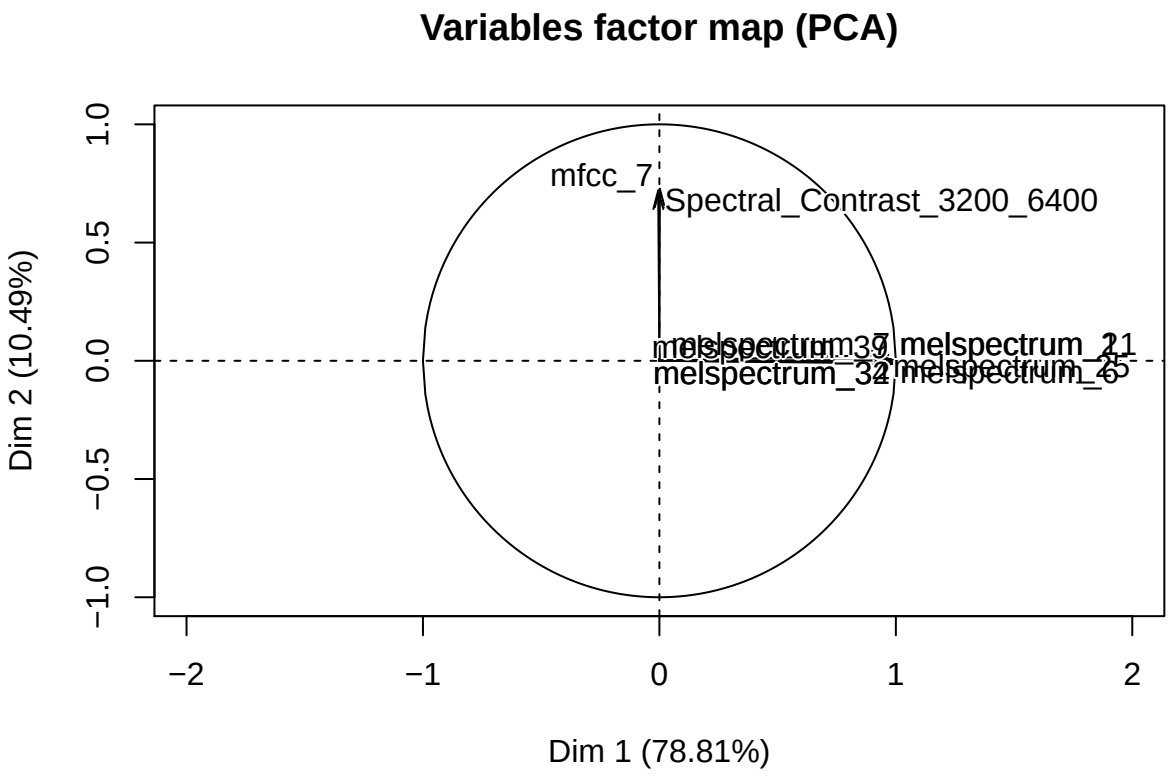
```
pca_test_data = read.csv('pca_test_data')
list_of_selected_columns = sample(colnames(pca_test_data), 10)
pca_test_data = pca_test_data[, list_of_selected_columns]
pca_test_data.scaled = scale(pca_test_data, scale = TRUE, center = TRUE)
```

Visualizing principal components

Now, let's apply PCA to the data and see how and in which direction are each variables in the data set represented.

```
library(FactoMineR)
pca = PCA(pca_test_data)
```


Individuals factor map (PCA)
Dim 2 (10.49%)
50
0
−50
3640
0
50
100
150
200
250
300
Dim 1 (78.81%)



Real life data is messy and has outliers. As we can see in the first image above, we can find outliers in the data. Outlier detection and removal can also be done in the PCA domain. I can manually remove the outliers from the data set using row numbers but that is not the focus of this work, so I will not do that now.

As we see in the second image, all the 'Melspectrum' related variables form a principal component in one direction and they all represent 29.91% of the variance in the data. 'Spectral Centroid', 'Spectral Rolloff' and 'Zero Crossing Rate' form a principal component along the y-axis and the remaining features form a principal component along the third axis.

COMPUTING PRINCIPAL COMPONENT ANALYSIS STEP BY STEP (work recreated from here https://goo.gl/vCuGqt)

Reading, Scaling, and centering the data

```
pca_test_data = read.csv('pca_test_data')
list_of_selected_columns = sample(colnames(pca_test_data), 10)
pca_test_data = pca_test_data[, list_of_selected_columns]
pca_test_data.scaled = scale(pca_test_data, scale = TRUE, center = TRUE)
```

Computing the correlation matrix

The correlation matrix finds how each variable in our dataset is correlated with the other variables. The diagonals of the correlation matrix will always contain 1s because it's the value of correlation of the variable

with itself.

The dimension of the correlation matrix will depend on the number of variables/ features present in our data set. If we have 4 variables then the correlation matrix will be [4,4] in dimension. If we have 16 variables then the correlation matrix will be [16,16] in dimension.

```
res.cor = round(cor(pca_test_data.scaled), 2)
```

Calculating Eigen Values and Eigen Vectors

Eigen values

In the dataframe below, eigen values indicate how much variance each principal component explains. By doing this, the eigen values indicate which principal component is the most important. If we sort the eigen values and the corresponding eigen vectors in decending order, then we will know which principal components capture maximum variance.

For example if we divide the first eigen value 4.763 by the total of all other eigen values then we will get a percentage of variance of 68.055. This indicates that the first prinipal component will capture 68.055% of total variance in the data.

```
res.eig = eigen(res.cor)

res.eig$values
```

```
##  [1]  5.994561e+00  1.367074e+00  1.005434e+00  9.352039e-01  6.913742e-01
##  [6]  1.372479e-02 -3.340698e-18 -3.098456e-05 -1.222641e-04 -7.219611e-03
```

Eigen Vectors

Eigen vectors transform our original data into principal components. Matrix multiplication of our scaled and centred original dataset with the first eigen vector will generate data for principal component 1. Each of these components are projected in a different direction in the 3-D space. The principal components are orthoginal to each other.

```
res.eig$vectors
```

```
##                [,1]          [,2]          [,3]         [,4]          [,5]
##  [1,] -0.4072639956 -0.0102077972 -0.0003959616  0.008214488 -9.773924e-03
##  [2,] -0.4086539049 -0.0007169928 -0.0000485859  0.002295456 -1.031868e-02
##  [3,]  0.0024426775 -0.6621080799  0.0784996148  0.207708300  7.156730e-01
##  [4,]  0.0001376675 -0.2324203294  0.8283038522 -0.482024473 -1.659654e-01
##  [5,] -0.4086498301 -0.0055602410  0.0007321674  0.004516451  3.277394e-05
##  [6,]  0.0147773728 -0.6368251128 -0.1116242944  0.348748953 -6.782035e-01
##  [7,] -0.4079704409 -0.0054855721  0.0007361056  0.004428615  1.741435e-04
##  [8,] -0.0045934099 -0.3191154829 -0.5434037608 -0.776368705 -1.027151e-02
##  [9,] -0.4086498301 -0.0055602410  0.0007321674  0.004516451  3.277394e-05
## [10,] -0.4079991670  0.0040159822  0.0010655621 -0.001521502 -3.678145e-04
##                [,6]          [,7]          [,8]         [,9]         [,10]
##  [1,]  0.7399200701  0.000000e+00  0.0024159698  0.002786783  5.351386e-01
##  [2,]  0.0662045337  2.682059e-13  0.5928523246 -0.561248051 -4.025338e-01
##  [3,] -0.0004816000  5.579191e-16  0.0027518828 -0.010309838 -1.226280e-04
##  [4,]  0.0004187736  2.359505e-17 -0.0001021173  0.001097698  6.787709e-05
##  [5,]  0.0658536358  7.071068e-01 -0.2952942299  0.281995327 -4.023639e-01
##  [6,] -0.0121622020 -3.084316e-15 -0.0057689267 -0.002228289 -1.870145e-03
```

```
##  [7,] -0.4732583897  2.898213e-13  0.4844392830  0.509926730  3.388652e-01
##  [8,]  0.0011484294  1.320053e-16  0.0002731920  0.001129420  1.497950e-04
##  [9,]  0.0658536358 -7.071068e-01 -0.2952942299  0.281995327 -4.023639e-01
## [10,] -0.4640460086 -2.932968e-13 -0.4892857678 -0.515568737  3.361050e-01
```

Calculating principal components

As mentioned above, under the heading 'Eigen Vector' above, to calculate principal components, we need to multiply the scaled and centred data set with eigen vectors. Multiplying the first eigen vector with the data matrix gives us the first principal component and multiplying the data matrix with the second eigen vector gives us the second principal component and so on and so forth.

First we have to transpose the eigen vectors and then the scaled and centred original data.

Transposing the eigen vectors

```
eigenvectors.t = t(res.eig$vectors)
```

Transposing the adjusted data

```
data.scaled.t = t(pca_test_data.scaled)
```

Now, we have to do matrix multiplication between the transpose eigen vector matrix and the transposed original data.

```
pc = eigenvectors.t %*% data.scaled.t
pc = t(pc)
dim(pc)
```

```
## [1] 8701   10
```

```
colnames(pc) <- c("PC1", "PC2", "PC3", "PC4", "pc5", "pc6", "pc7", "pc8", "pc9", "pc10")
head(pc)
```

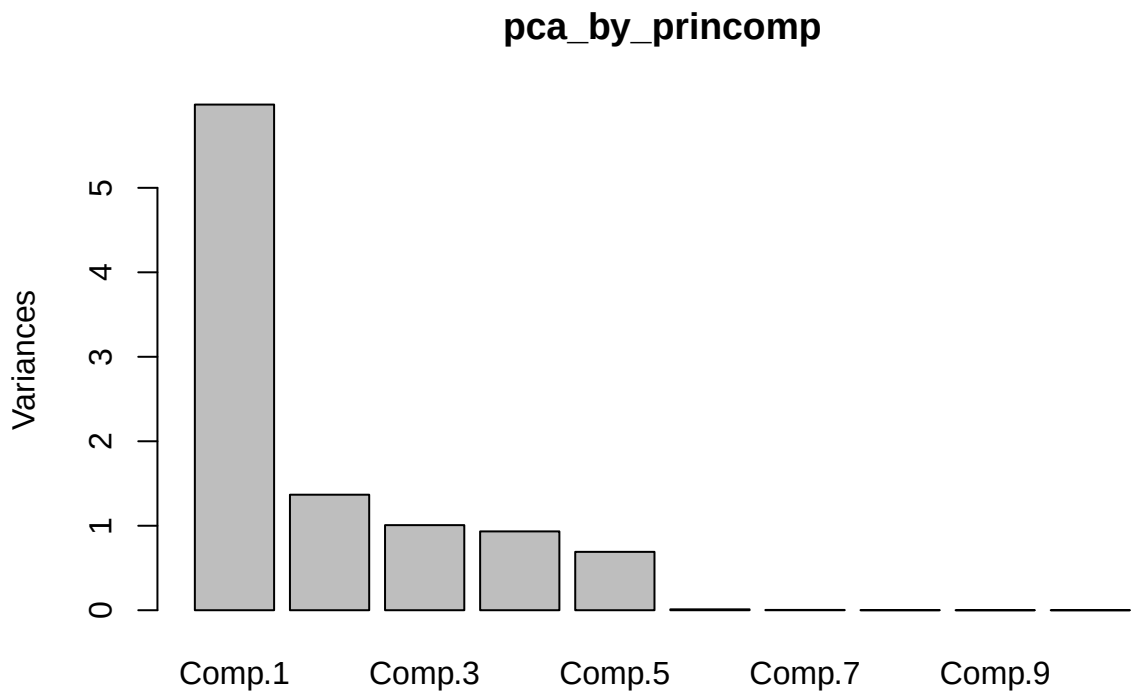
```
##              PC1        PC2        PC3        PC4         pc5          pc6
## [1,]  0.04514191 -1.5910067 -1.1698246 -2.2198304 -0.04589371  0.007451132
## [2,]  0.05740662 -0.2754893  0.8949205  1.0621316  1.02019252  0.010662267
## [3,]  0.04805597  0.3766408  0.3712534  0.4915676  0.60250191  0.014472181
## [4,] -0.02414953  4.5672384 -1.1794662 -2.5581186  0.79614992  0.062864966
## [5,]  0.07322360 -0.4333066  0.6078581  0.5573823 -1.63002664 -0.010259957
## [6,]  0.08280646 -2.0353027 -0.8520167  1.3114929 -0.03153060 -0.014493936
##              pc7           pc8           pc9          pc10
## [1,] 0.001916055 -0.0006319719  0.0004838003  0.0010044003
## [2,] 0.001983662  0.0041336501 -0.0105721002  0.0014865800
## [3,] 0.001852426  0.0033521393 -0.0004586591  0.0022671091
## [4,] 0.001538662  0.0160801818  0.0430182428  0.0092381268
## [5,] 0.001955529 -0.0123853720  0.0039973224 -0.0005203226
## [6,] 0.002078781 -0.0065622967 -0.0178573029 -0.0021761131
```

The following two plots are recreated from the book 'Practical Statistics for Data Scientists by Peter Bruce and Andrew Bruce

Screeplot

Screeplot is used to visualize the percentage of variance explained by each principal component. This plot can directly be visualized if we use the 'princomp' command in R.

```
pca_by_princomp = princomp(pca_test_data.scaled)
screeplot(pca_by_princomp)
```

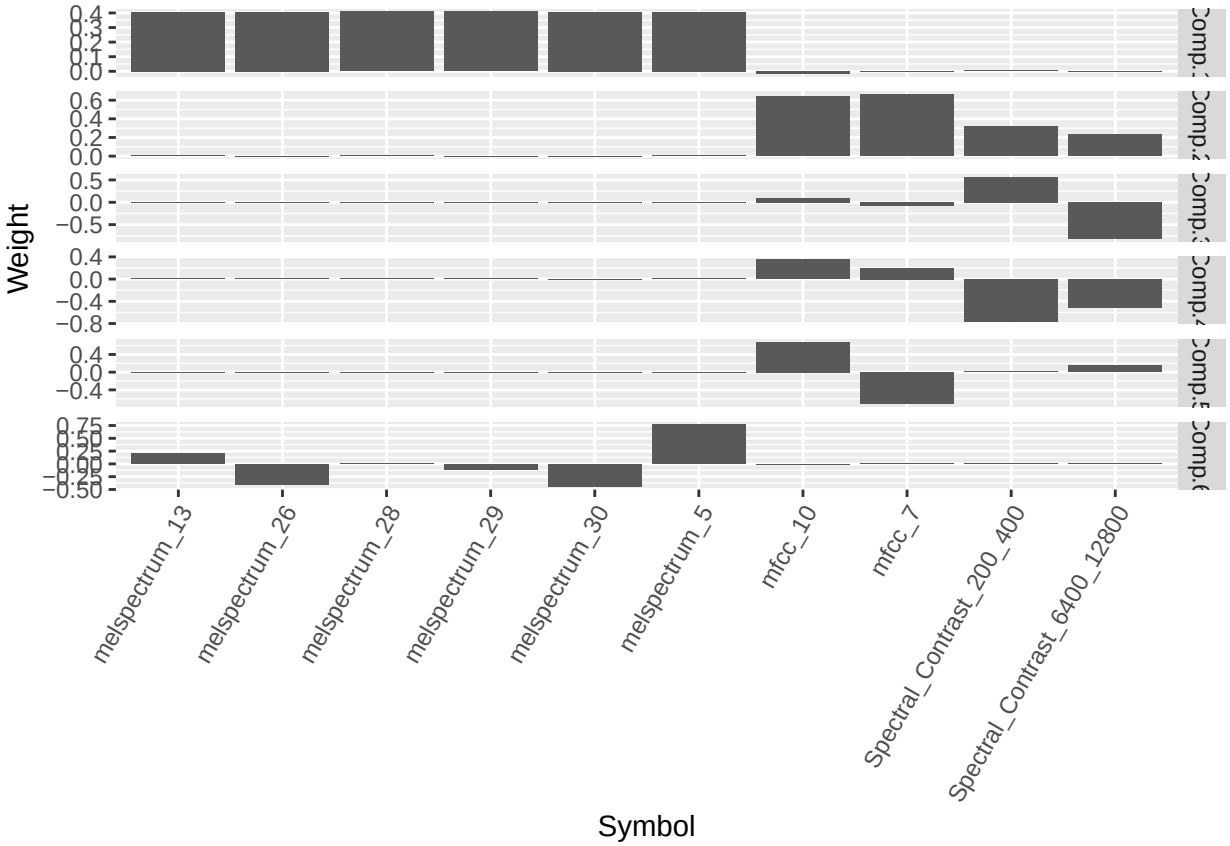


As we can infer from the previous image, majority of the variance is captured by the first principal component. Also, principal components 5 to 10 doesn't really capture any variance.

How much does each variable affect a principal component??

Since only the first 6 principal components capture some variance let's only visualize them and see how each variable/feature affects them.

```
library(tidyr)
library(ggplot2)
loadings <- pca_by_princomp$loadings[,1:6]
loadings <- as.data.frame(loadings)
loadings$Symbol <- row.names(loadings)
loadings <- gather(loadings, "Component", "Weight", -Symbol)
q = ggplot(loadings, aes(x = Symbol, y= Weight)) +
  geom_bar(stat = 'identity') +
  facet_grid(Component ~ ., scales = 'free_y')
q+theme(axis.text.x = element_text(angle = 60, hjust = 1))
```



The last 2 features, doesn't have any influence in principal component1. The first 7 features doesn't have any effect on principal components 2,3 and 4. This image gives a clear picture of how important each feature is to each principal component. If we look carefully, we can also see that some features affect the principal component in the positive direction and some affects it in the negative direction.

Summary

1. In this work, I first started off by visualizing how variables/features form principal components. For that I used thr **FactoMineR** library of R and the **PCA** command from that library.
2. Then I went through the step by step guide to perform principal component analysis where I did not use any library but I used commands like **cor** for calculating correlation matrix, **eigen** for calculating the eigen values and vectors, then I performed transpose of matrix and matrix multiplication.
3. Finally I used the **princomp** command to visualize variable importance and the weight of each variable in each principal component.