Principal Component Analysis

Eduardo Abdala

5/8/2020

Principal component analysis is a way to reduce your data's dimensions and still get the essence of the dataset This is dont by choosing dimensions that explain most of the data's variability, as you will experience some data loss with this technique. We need a hyperplane that is close to the data, and then project the data onto it.

We identify the axis with the most data variability and build another axis, orthogonal to the first. A data set will have k principal components, where k is the amount of features (columns) you have in your data. To find these components there exists a matrix factorization $U\Sigma V^T$ where V contains all of the principal components.

To do PCR, you can use the `prcomp()` function.

```
# generating 3 columns of normally distributed data (mean = 0, sd = 1)
data.matrix <- matrix(nrow=100, ncol=3)
colnames(data.matrix) <- c(
  paste("x", 1:3, sep="")
)
rownames(data.matrix) <- seq(1,100,1)
for (i in 1:100) {
  x.values <- rnorm(3, 0, 1)
  data.matrix[i,] <- c(x.values)
}

head(data.matrix)
```

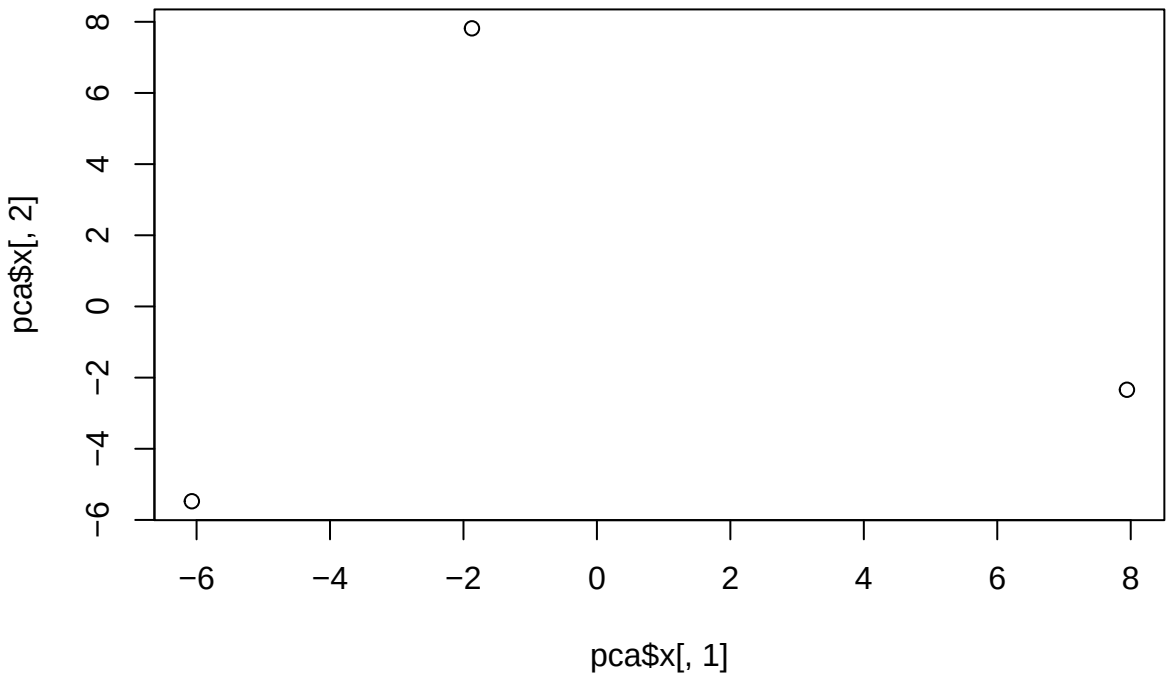
```
##           x1           x2         x3
## 1  0.9180201  0.005553872  1.1874751
## 2 -0.1596624  1.407939194  0.6369953
## 3 -0.9495391 -0.907609949  0.7437450
## 4  1.0774415 -1.364032228 -0.4227582
## 5  0.2798194 -0.277864246  0.9421761
## 6  1.2954705 -1.411138363  1.4149648
```

```
dim(data.matrix)
```

```
## [1] 100   3
```

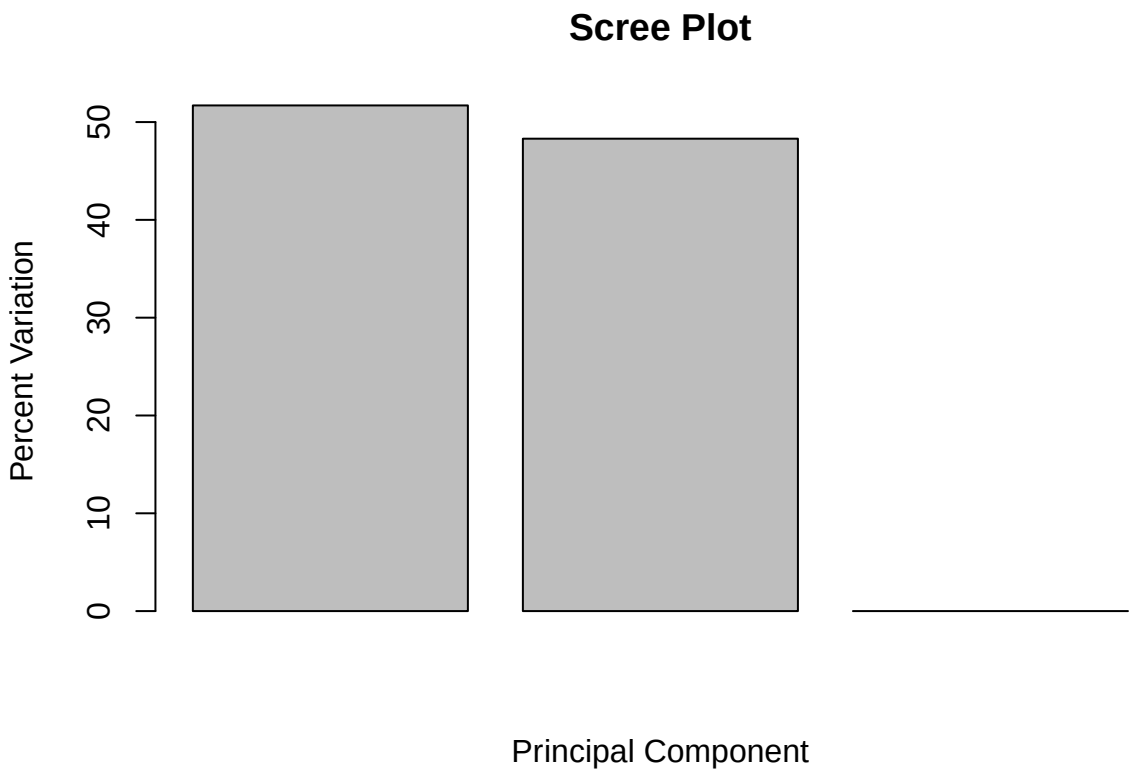
```
# R as a function to return principal components, standard deviation, and rotation
pca <- prcomp(t(data.matrix), scale=TRUE)

## graph the first two principal components
plot(pca$x[,1], pca$x[,2])
```




```
## scree plot showing which components explain the majority of the data. We see that the first two comp
pca.var <- pca$sdev^2
pca.var.per <- round(pca.var/sum(pca.var)*100, 1)

barplot(pca.var.per, main="Scree Plot", xlab="Principal Component", ylab="Percent Variation")
```

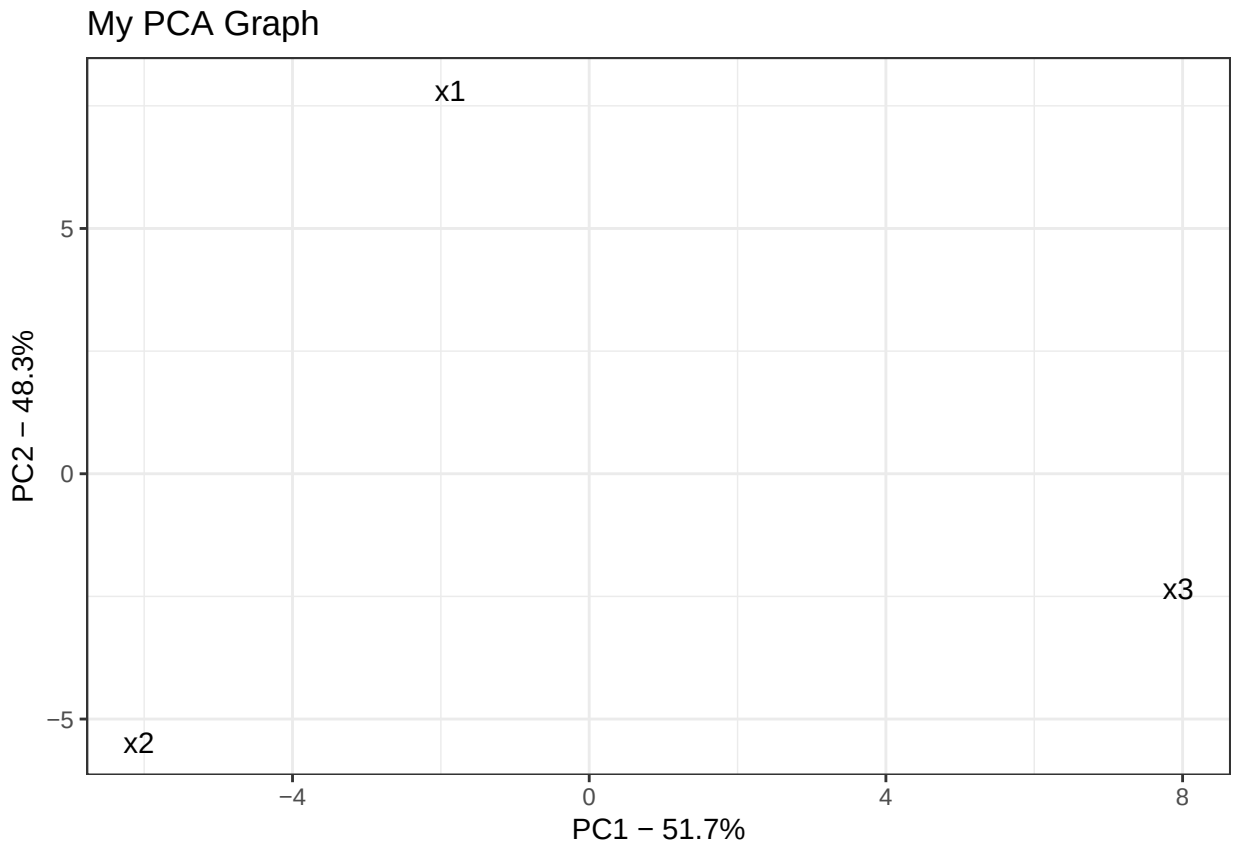



```
## With this plot, we can see the 3 components with the axes telling you how much each component accoun

pca.data <- data.frame(Sample=rownames(pca$x),
  X=pca$x[,1],
  Y=pca$x[,2])
pca.data
```

```
##    Sample         X         Y
## x1     x1 -1.872730  7.815785
## x2     x2 -6.070807 -5.474522
## x3     x3  7.943537 -2.341263
```

```
ggplot(data=pca.data, aes(x=X, y=Y, label=Sample)) +
  geom_text() +
  xlab(paste("PC1 - ", pca.var.per[1], "%", sep="")) +
  ylab(paste("PC2 - ", pca.var.per[2], "%", sep="")) +
  theme_bw() +
  ggtitle("My PCA Graph")
```




```
## what data original data points contribute most to the principal components?
## This is random data so it is not that useful, but something you can do with prcomp()
loading_scores <- pca$rotation[,1]
data_scores <- abs(loading_scores) ## get the magnitudes
data_score_ranked <- sort(data_scores, decreasing=TRUE)
inf_data <- names(data_score_ranked[1:10])

inf_data
```

```
##  [1] "49" "99" "7"  "9"  "74" "15" "19" "79" "13" "42"
```

```
pca$rotation[inf_data,1]
```

```
##         49         99          7          9         74         15         19
## -0.1390341  0.1390115  0.1388963  0.1387381 -0.1386696 -0.1385170  0.1381440
##         79         13         42
## -0.1379296  0.1378059  0.1377686
```

Covariance Matrix Approach

```
cov.mat <- cov(scale(t(data.matrix), center=TRUE))
dim(cov.mat)
```

```
## [1] 100 100
```

```
evect <- eigen(cov.mat, symmetric=TRUE)
dim(evect$vectors)
```

```
## [1] 100 100
```

```
head(evect$vectors[,1:2])
```

```
##             [,1]        [,2]
## [1,]  0.11983466  0.07298598
## [2,] -0.03931502 -0.13806023
## [3,]  0.13207059 -0.04498459
## [4,]  0.02280121  0.14198580
## [5,]  0.13683972  0.02547887
## [6,]  0.10510713  0.09421980
```

```
eigen.pcs <- t(t(evect$vectors) %*% t(scale(t(data.matrix), center=TRUE)))
eigen.pcs[,1:2]
```

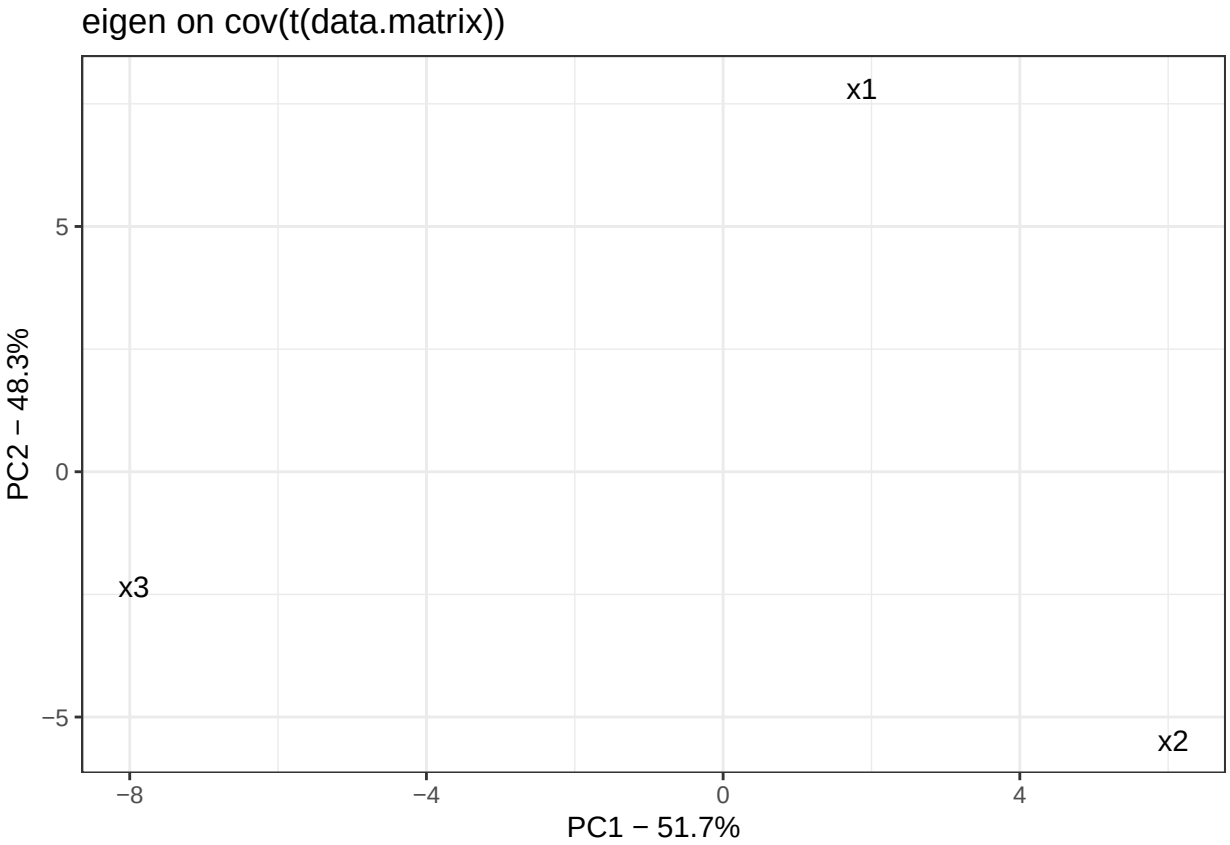
```
##         [,1]      [,2]
## x1 -1.872730  7.815785
## x2 -6.070807 -5.474522
## x3  7.943537 -2.341263
```

```
eigen.data <- data.frame(Sample=rownames(eigen.pcs),
  X=(-1 * eigen.pcs[,1]), ## eigen() flips the X-axis in this case, so we flip it back
  Y=eigen.pcs[,2]) ## X axis will be PC1, Y axis will be PC2
eigen.data
```

```
##    Sample         X         Y
## x1     x1  1.872730  7.815785
## x2     x2  6.070807 -5.474522
## x3     x3 -7.943537 -2.341263
```

```
eigen.var.per <- round(evect$values/sum(evect$values)*100, 1)

ggplot(data=eigen.data, aes(x=X, y=Y, label=Sample)) +
  geom_text() +
  xlab(paste("PC1 - ", eigen.var.per[1], "%", sep="")) +
  ylab(paste("PC2 - ", eigen.var.per[2], "%", sep="")) +
  theme_bw() +
  ggtitle("eigen on cov(t(data.matrix))")
```


eigen on cov(t(data.matrix))
x1
x2
x3
5
0
−5
−8
−4
0
4
PC2 − 48.3%
PC1 − 51.7%