Unsupervised Learning

Lucas Ben

In this project I'll analyse a data set of acoustic properties of the voices of ~3000 individuals.

```
library(ggplot2)
theme_set(theme_bw())
voice_data = read.csv("voice_data.csv")
```

Principal Components Analysis

I'm creating a matrix, *X*, for the data variables where each row is an observation and each column is a variable. The matrix will include all variables from the original data set except for the last variable which is qualitative. Also, I'm identifying the three variables that most strongly contribute to the first principal component which are *meanfreq*, *Q25*, and *centroid*.

```
X = scale(voice_data[, -ncol(voice_data)]) # excluding the last column and scaling the v
ariables
Sigma = cov(X) # retrieving covariance matrix
eig = eigen(Sigma) # retrieving eigenvalues and eigenvectors
eig$vectors[,1] # loading for the first principal component
```

```
##  [1]  0.31336136 -0.27977246  0.27930171  0.30339858  0.18852307 -0.24104888
##  [7] -0.13079792 -0.13187105 -0.22261647 -0.27442864  0.24312987  0.31336136
## [13]  0.18770729  0.15814609  0.11054344  0.22603264  0.08893811  0.22917517
## [19]  0.22764859 -0.08568381
```

```
tail(order(abs(eig$vectors[,1])), 3)
```

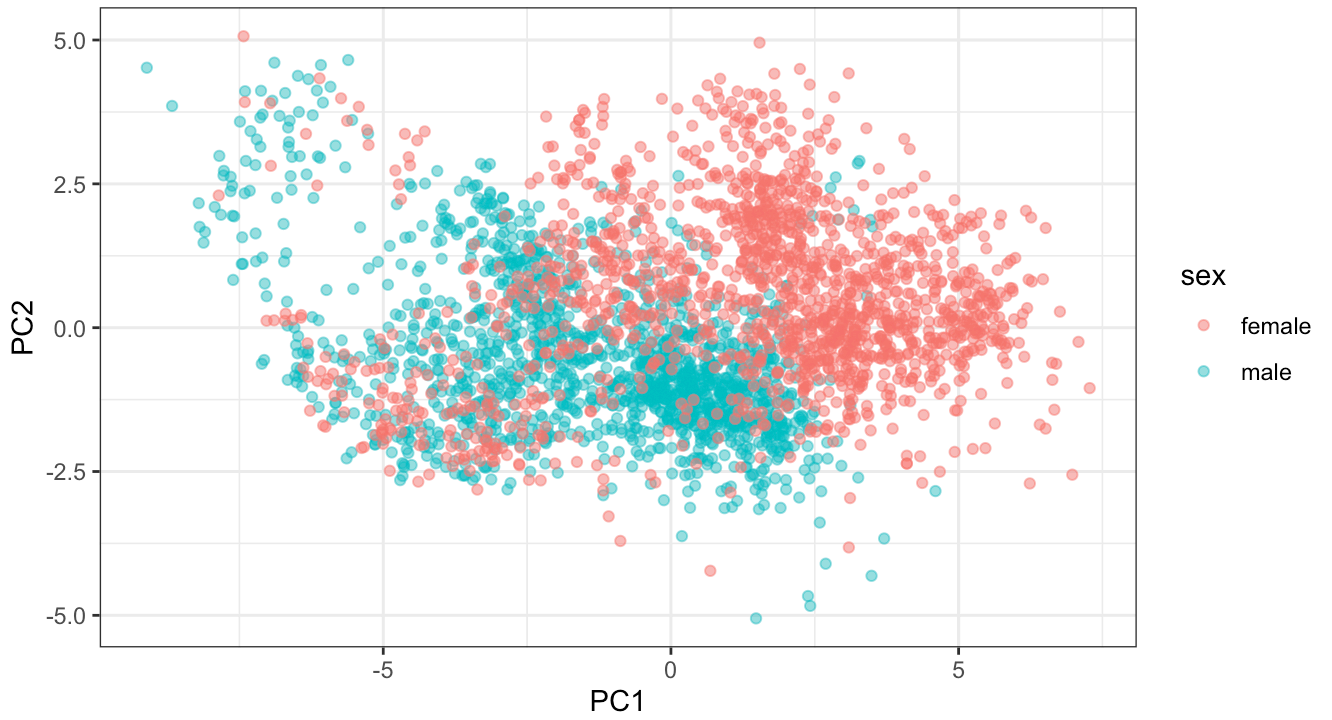
```
## [1]  4  1 12
```

```
colnames(X)[c(1,4,12)]
```

```
## [1] "meanfreq" "Q25"      "centroid"
```

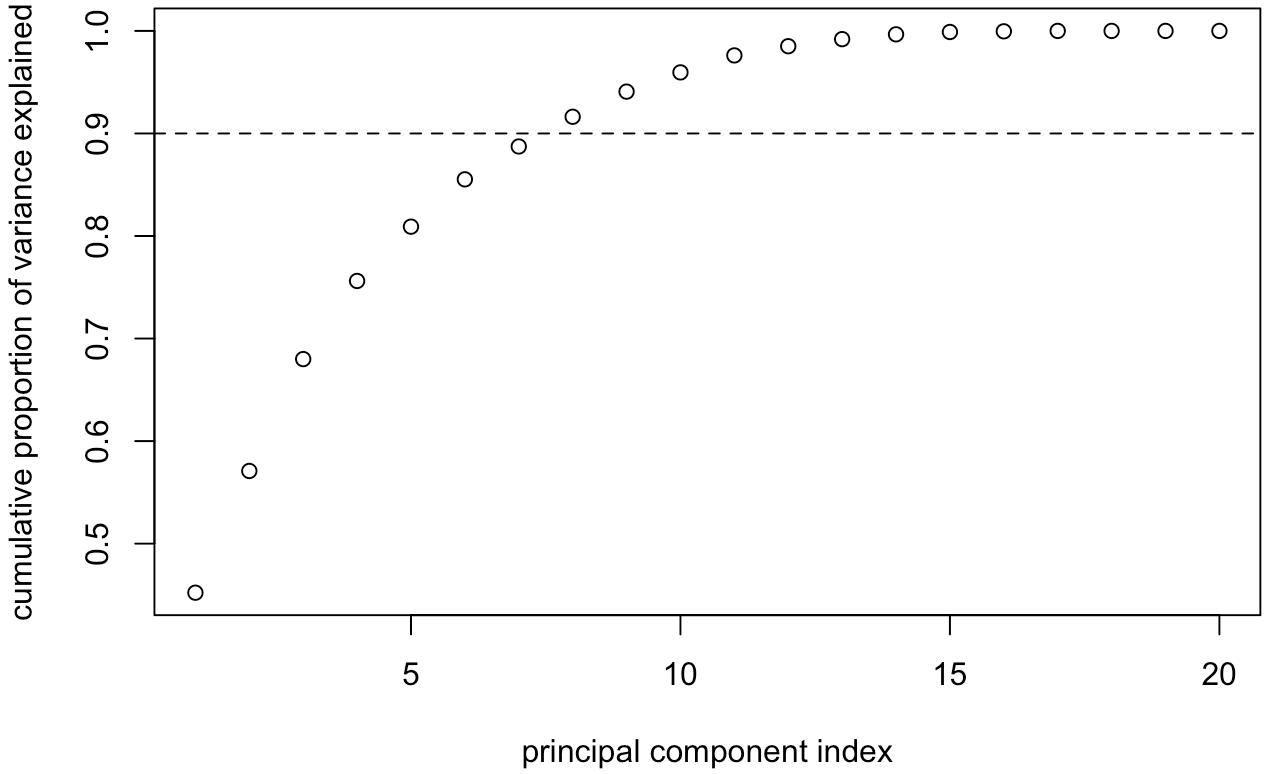
Now I'm plotting the scores for PC1 and PC2, coloured by the qualitative variable *sex* in the original data frame.

```
Z = X %*% eig$vectors # computing scores
df = data.frame(PC1 = Z[,1], PC2 = Z[,2], sex = voice_data$label)
ggplot(df, aes(PC1, PC2, col = sex)) +
  geom_point(alpha = 0.5) +
  coord_equal()
```



And here is the cumulative proportion of variance explained by the principal components, against the number of principal components. To explain at least 90% of the variance in the data, the minimum number of principal components to keep is 8.

```
pc_var = eig$values # principal component variance
pc_prop_var = pc_var/sum(pc_var) # proportion of variance explained
pc_cumul_prop_var = cumsum(pc_prop_var) # cumulative proportion of variance explained
plot(pc_cumul_prop_var, xlab = "principal component index",
     ylab = "cumulative proportion of variance explained")
abline(h = 0.9, lty = 2)
```



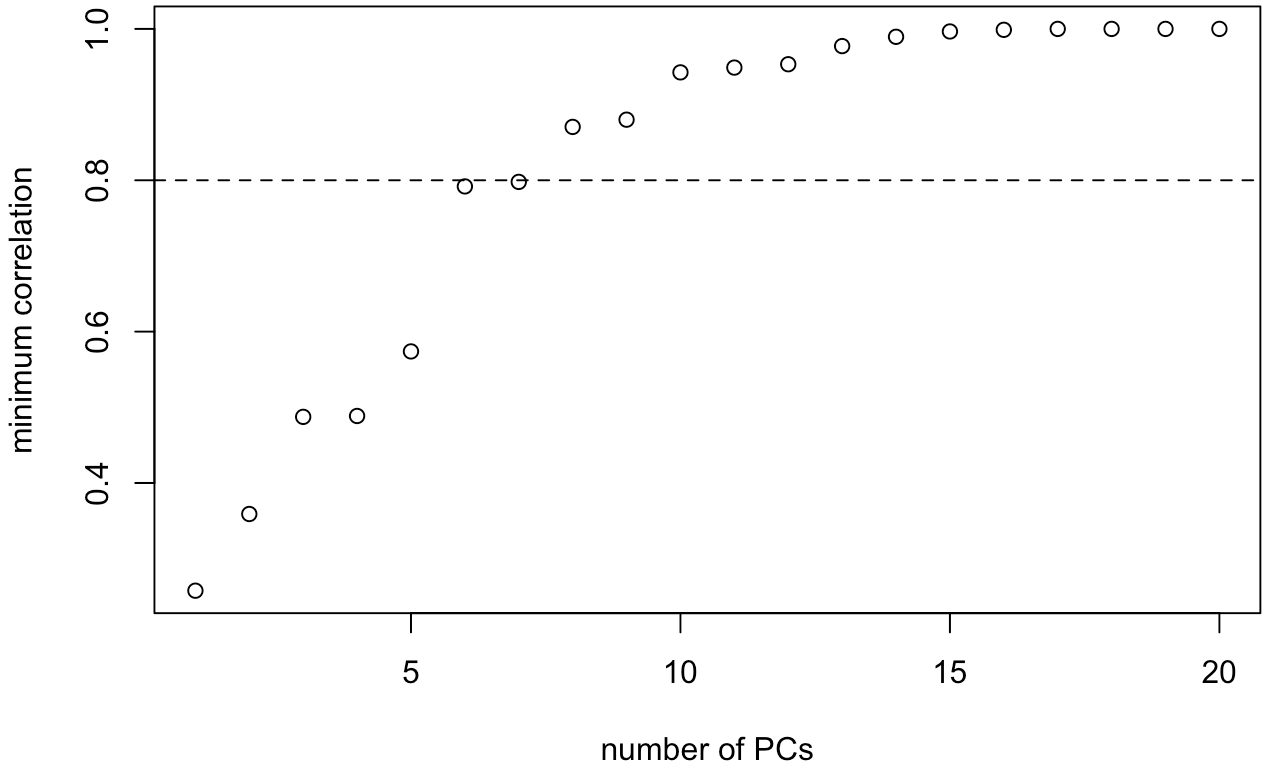

```
pc_cumul_prop_var[1:8]
```

```
## [1] 0.4521639 0.5708700 0.6799694 0.7561670 0.8091064 0.8552561 0.8873009
## [8] 0.9162848
```

Next I'm reconstructing the original variables from p' principal components, for p' = 1,2,,20. For each p', I'm computing the correlations between the original variables and the reconstructed variables and storing the smallest correlation.

To achieve a minimum correlation of 0.8 between original and reconstructed variables, eight principal components should be kept.

```
min_cors = rep(NA, length = ncol(Z))
for(n_pc in 1:ncol(Z)) {
  X_approx = Z[,1:n_pc] %*% t(eig$vectors[,1:n_pc]) # getting reconstructed variables
  cors = sapply(1:ncol(X), function(i) cor(X[,i], X_approx[,i])) # correlation between d
ata variables and reconstructed variables
  min_cors[n_pc] = min(cors) # saving minimum correlation
}
plot(min_cors,
     xlab = "number of PCs",
     ylab = "minimum correlation")
abline(h = 0.8, lty = 2)
```




```
min_cors[1:8]
```

```
## [1] 0.2576687 0.3590043 0.4873630 0.4885433 0.5738333 0.7918810 0.7978494
## [8] 0.8705046
```

Clustering

Here I'm applying the K-means algo using the function **kmeans**, with three clusters and 100 initial configurations.

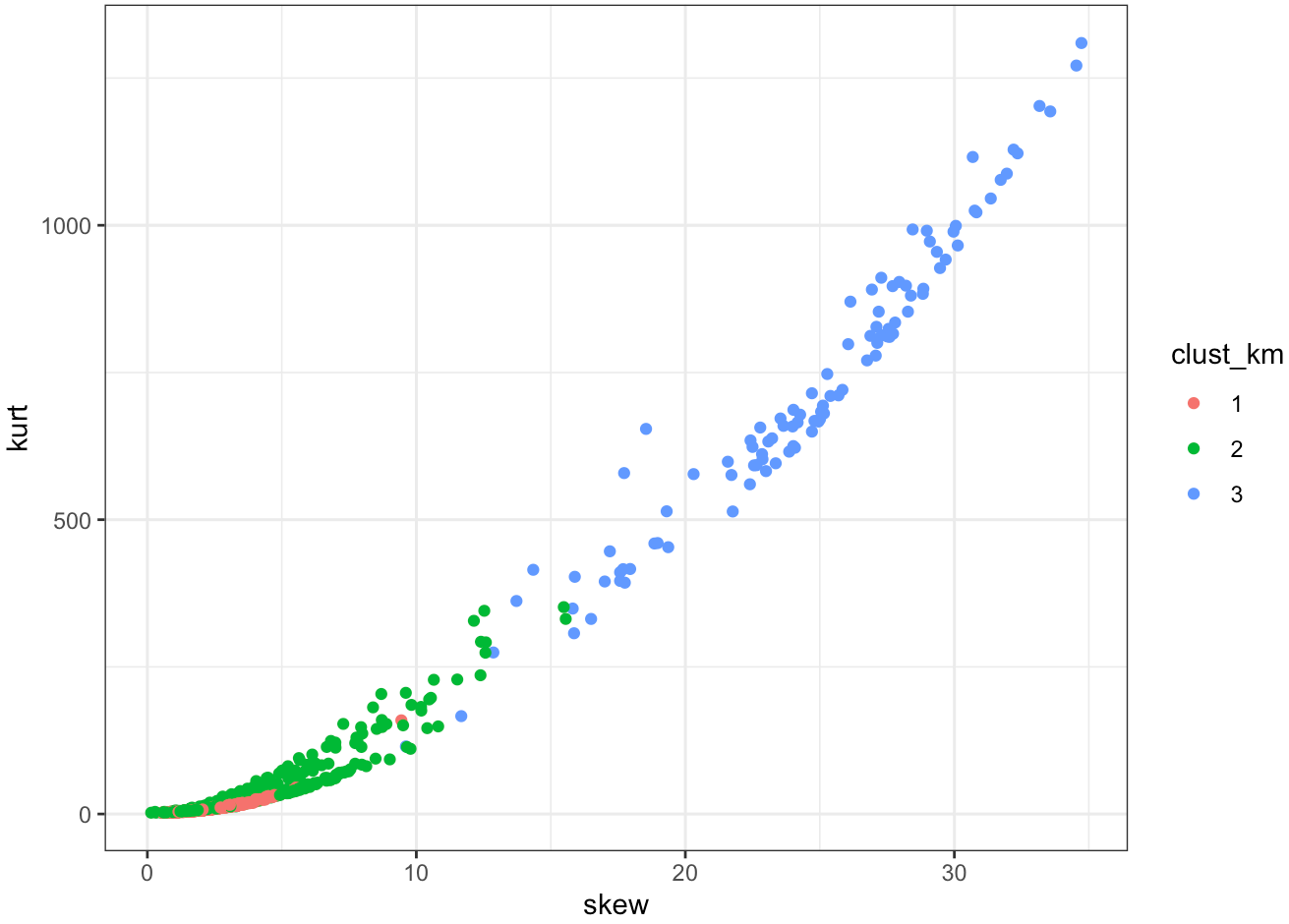
```
set.seed(2002)
km = kmeans(x = X, centers = 3, nstart = 100) # applying K-means with K = 3
km$tot.withinss / nrow(X) # mean within-cluster squared distance
```

```
## [1] 11.75278
```

Now I'm extracting the cluster labels from the K-means algo so I can plot the data and visually inspect it to find two data variables for which the smallest cluster is distinct from the other two clusters.

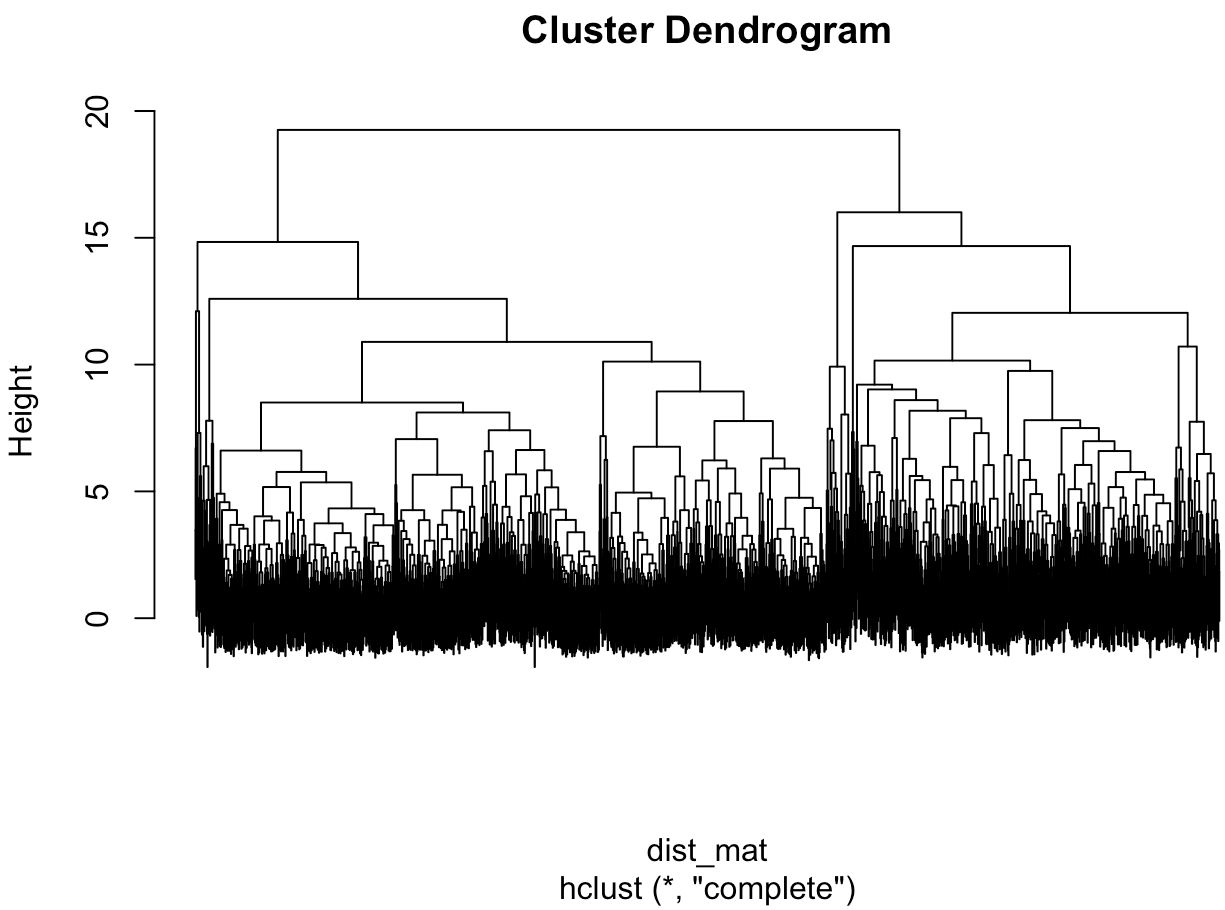
It appears that *skew* and *kurt* are the best variables to distinguish the smallest cluster.

```
voice_data$clust_km = factor(km$cluster)
ggplot(voice_data, aes(skew, kurt, col = clust_km)) +
  geom_point()
```



Now I'm applying complete linkage clustering using the function **hclust**, and plotting the dendrogram.

```
dist_mat = dist(X) # computing distance matrix
hc = hclust(d = dist_mat, method = "complete") # complete linkage
plot(hc, labels = FALSE)
```



Here I'm calculating the range of heights at which the dendorgram could be cut to obtain three clusters. I'm also extracting the cluster labels for the complete linkage algo, with K = 3.

```
heights = tail(hc$height, 3)[1:2]
cat("The range of heights is between 14.84 and 16.01")
```

```
## The range of heights is between 14.84 and 16.01
```

```
clust_hc = cutree(hc, k = 3)
```

To compare the two clusterings from earlier, using a contingency table helps determine which cluster in the first approach corresponds to each cluster in the second approach. The proportion of observations for which the cluster is different with the two approaches is ~8.7% ((44+209+24)/3168) which is low, as would be expected because complete linkage and K-means both tend to favour dense, circular clusters.

```
tab = table(hier = cutree(hc, k =3),
            kmeans = voice_data$clust_km) # comparing hierarchical clustering to K-means
tab
```

```
##     kmeans
## hier    1    2    3
##    1   44 1070   24
##    2    0    0   79
##    3 1742  209    0
```