Module 10: Recommended Exercises - Solutions

TMA4268 Statistical Learning V2020

Thiago G. Martins, Martina Hall, Stefanie Muff, Department of Mathematical Sciences, NTNU

March 14, 2020

Recommended exercise 1

Get the data by loading it

```
load("pca-examples.rdata")

# We will work with nyt.frame
nyt_data = nyt.frame
```

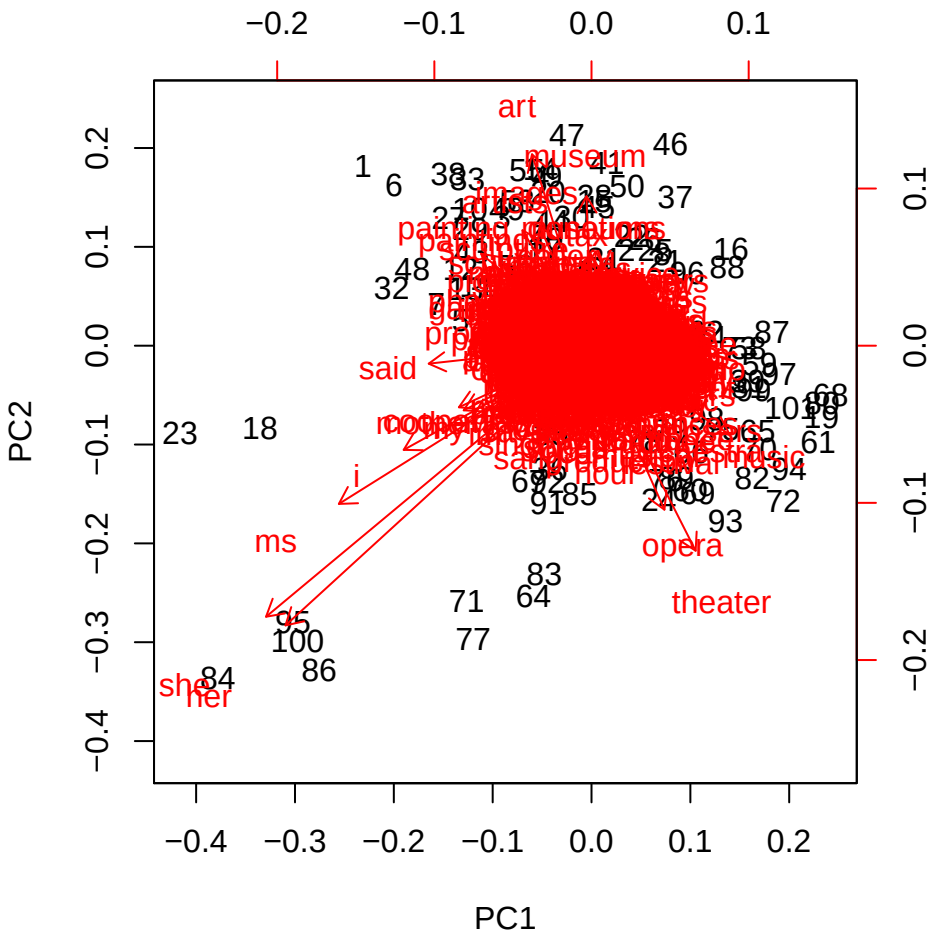
Compute the PCA

```
nyt_pca = prcomp(nyt_data[, -1])
```

Default biplot

Too much information on the graph, we should select only a few loading vectors to display.

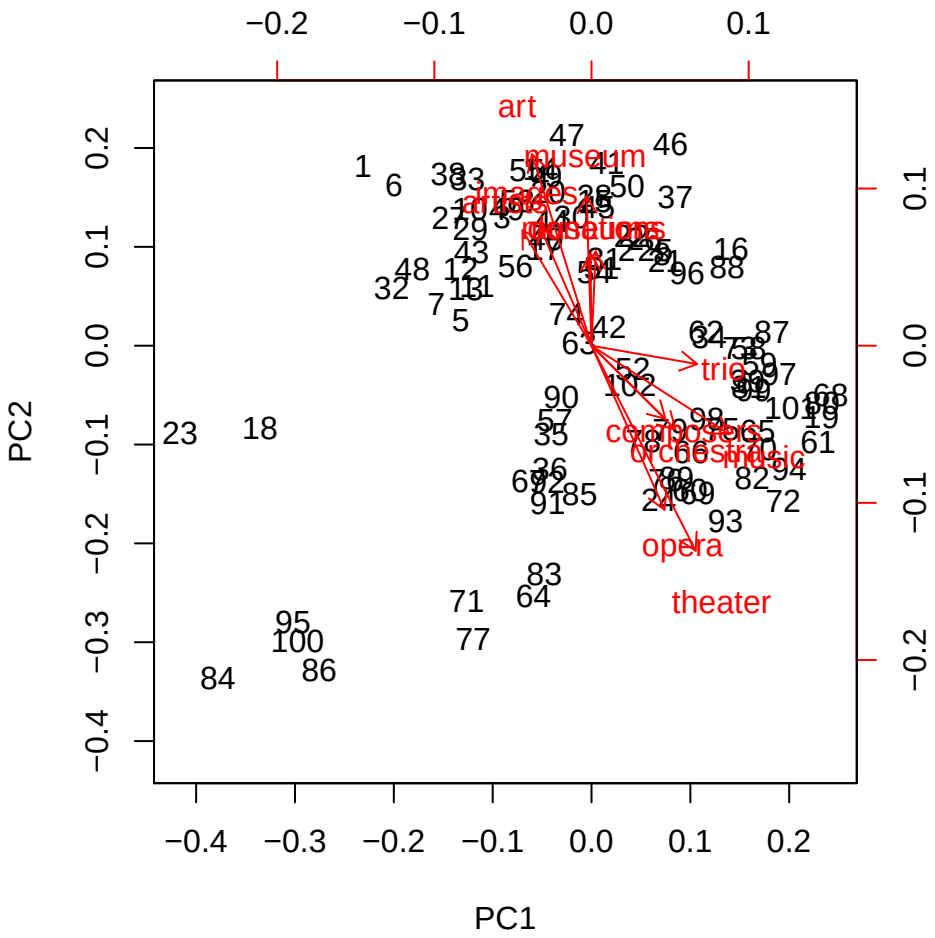
```
biplot(nyt_pca, scale = 0)
```



Lets pick some words with high PC1 weight and some words with high PC2 weight. Only looking at the graph we can see that PC1 is associated with music and PC2 with art.

```
nyt_loading = nyt_pca$rotation[, 1:2]
informative_loadings = rbind(head(nyt_loading[order(nyt_loading[, 1], decreasing = TRUE),
    ]), head(nyt_loading[order(nyt_loading[, 2], decreasing = TRUE), ]))

biplot(x = nyt_pca$x[, 1:2], y = informative_loadings, scale = 0)
```



Proportion of variance explained (PVE)

The numbers below show that although the graphs based on PC1 and PC2 give some insight about document types, the first two PCs explain only small portion of the variability contained in the data.

```
pr.var = nyt_pca$sdev^2
pr.var
```

```
##   [1] 1.988751e-02 1.862759e-02 1.724294e-02 1.537218e-02 1.440484e-02
##   [6] 1.393311e-02 1.313569e-02 1.285084e-02 1.228344e-02 1.207883e-02
##  [11] 1.191209e-02 1.178849e-02 1.169569e-02 1.159828e-02 1.140559e-02
##  [16] 1.133632e-02 1.124601e-02 1.113017e-02 1.110012e-02 1.095067e-02
##  [21] 1.086651e-02 1.072844e-02 1.060126e-02 1.058186e-02 1.056528e-02
##  [26] 1.045440e-02 1.035742e-02 1.032494e-02 1.030728e-02 1.022799e-02
##  [31] 1.010669e-02 1.004629e-02 1.001430e-02 9.977470e-03 9.945853e-03
##  [36] 9.885270e-03 9.732177e-03 9.679161e-03 9.636696e-03 9.578010e-03
##  [41] 9.535481e-03 9.466407e-03 9.437146e-03 9.399834e-03 9.307165e-03
##  [46] 9.203604e-03 9.179709e-03 9.133426e-03 9.102611e-03 9.041575e-03
##  [51] 8.998882e-03 8.925813e-03 8.884005e-03 8.848814e-03 8.799356e-03
##  [56] 8.747191e-03 8.717555e-03 8.643816e-03 8.559859e-03 8.529536e-03
##  [61] 8.469763e-03 8.467367e-03 8.393981e-03 8.376228e-03 8.251717e-03
##  [66] 8.235627e-03 8.149720e-03 8.123239e-03 8.044627e-03 8.015822e-03
##  [71] 7.982089e-03 7.888293e-03 7.810759e-03 7.795754e-03 7.774335e-03
##  [76] 7.683500e-03 7.631978e-03 7.615316e-03 7.548048e-03 7.482993e-03
```

```
##  [81] 7.419521e-03 7.366235e-03 7.340519e-03 7.288173e-03 7.269157e-03
##  [86] 7.175837e-03 7.160695e-03 7.109025e-03 7.036169e-03 6.984169e-03
##  [91] 6.892596e-03 6.805417e-03 6.670563e-03 6.566518e-03 6.466879e-03
##  [96] 6.420743e-03 6.337557e-03 6.233789e-03 6.191162e-03 6.025576e-03
## [101] 5.283112e-03 3.582418e-32
```

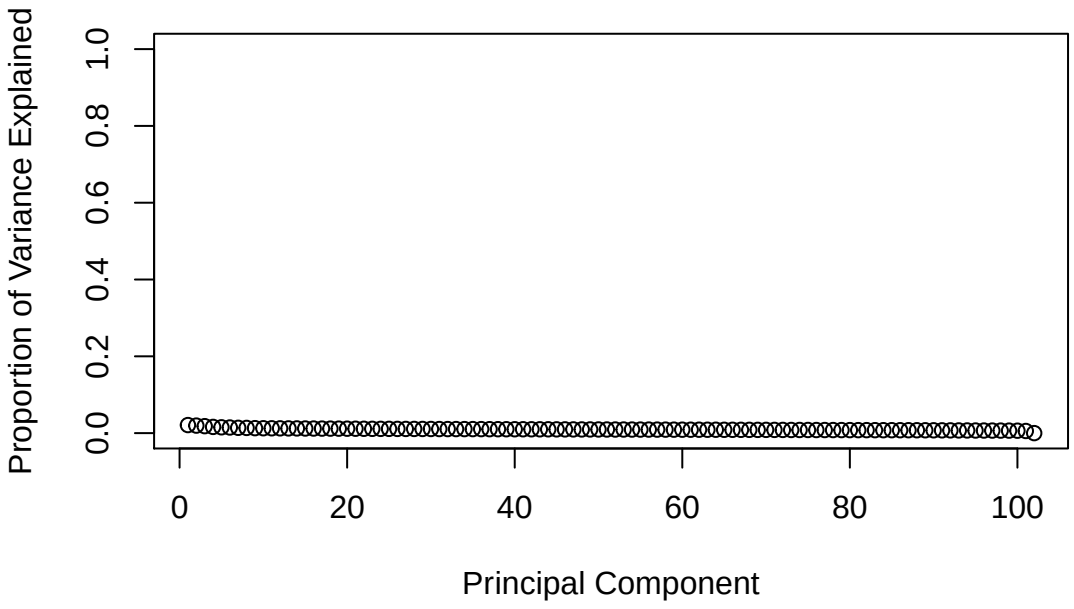
We can then compute the proportion of variance explained by each principal component

```
pve = pr.var/sum(pr.var)
pve
```

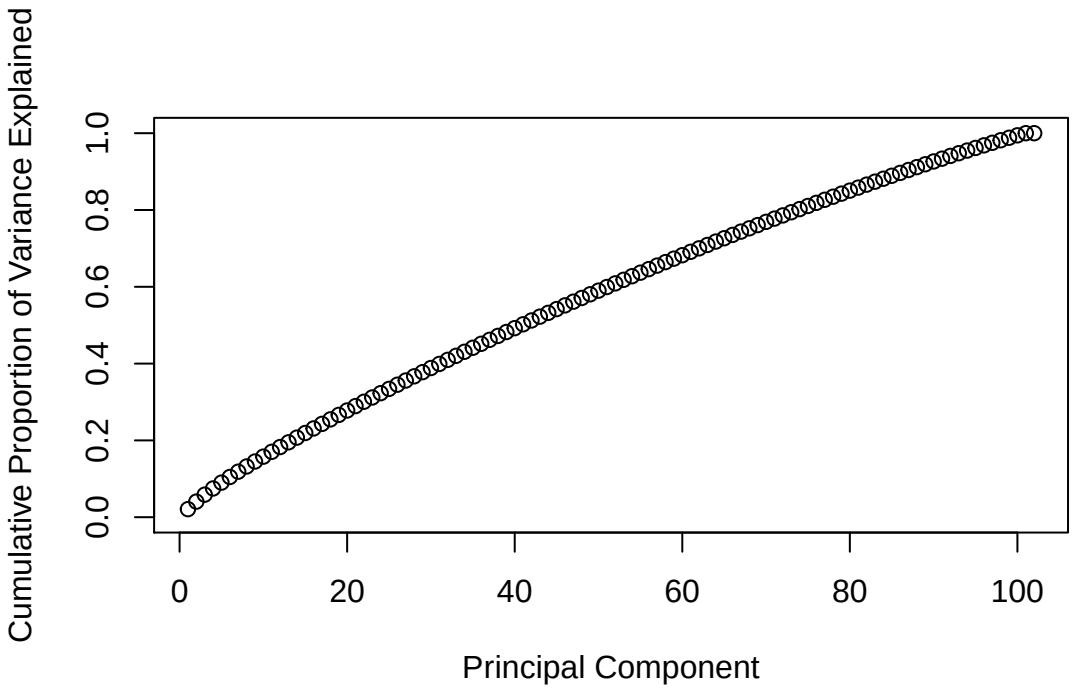
```
##   [1] 2.093766e-02 1.961121e-02 1.815344e-02 1.618390e-02 1.516547e-02
##   [6] 1.466884e-02 1.382931e-02 1.352942e-02 1.293206e-02 1.271664e-02
##  [11] 1.254110e-02 1.241097e-02 1.231328e-02 1.221072e-02 1.200786e-02
##  [16] 1.193492e-02 1.183985e-02 1.171789e-02 1.168626e-02 1.152892e-02
##  [21] 1.144031e-02 1.129495e-02 1.116105e-02 1.114063e-02 1.112318e-02
##  [26] 1.100644e-02 1.090434e-02 1.087014e-02 1.085155e-02 1.076808e-02
##  [31] 1.064037e-02 1.057678e-02 1.054310e-02 1.050432e-02 1.047104e-02
##  [36] 1.040726e-02 1.024608e-02 1.019026e-02 1.014556e-02 1.008377e-02
##  [41] 1.003900e-02 9.966275e-03 9.935469e-03 9.896187e-03 9.798624e-03
##  [46] 9.689594e-03 9.664439e-03 9.615711e-03 9.583269e-03 9.519011e-03
##  [51] 9.474062e-03 9.397135e-03 9.353120e-03 9.316071e-03 9.264001e-03
##  [56] 9.209081e-03 9.177880e-03 9.100248e-03 9.011858e-03 8.979933e-03
##  [61] 8.917003e-03 8.914481e-03 8.837220e-03 8.818530e-03 8.687444e-03
##  [66] 8.670504e-03 8.580061e-03 8.552182e-03 8.469419e-03 8.439092e-03
##  [71] 8.403578e-03 8.304829e-03 8.223201e-03 8.207404e-03 8.184854e-03
##  [76] 8.089223e-03 8.034980e-03 8.017438e-03 7.946618e-03 7.878128e-03
##  [81] 7.811304e-03 7.755204e-03 7.728131e-03 7.673021e-03 7.653000e-03
##  [86] 7.554753e-03 7.538811e-03 7.484412e-03 7.407710e-03 7.352964e-03
##  [91] 7.256556e-03 7.164773e-03 7.022798e-03 6.913259e-03 6.808359e-03
##  [96] 6.759786e-03 6.672208e-03 6.562961e-03 6.518083e-03 6.343753e-03
## [101] 5.562084e-03 3.771586e-32
```

We can plot the PVE explained by each component, as well as the cumulative PVE, as follows:

```
plot(pve, xlab = "Principal Component", ylab = "Proportion of Variance Explained",
    ylim = c(0, 1), type = "b")
```



```r
plot(cumsum(pve), xlab = "Principal Component", ylab = "Cumulative Proportion of Variance Explained",
    ylim = c(0, 1), type = "b")
```



Recommended exercise 2

The answer is on page 388 of the book, with the explanation around Equation (10.12).

Recommended exercise 3

k-means clustering:

```r
km.out = kmeans(nyt_data[, -1], 2, nstart = 20)
```

Cluster assignments

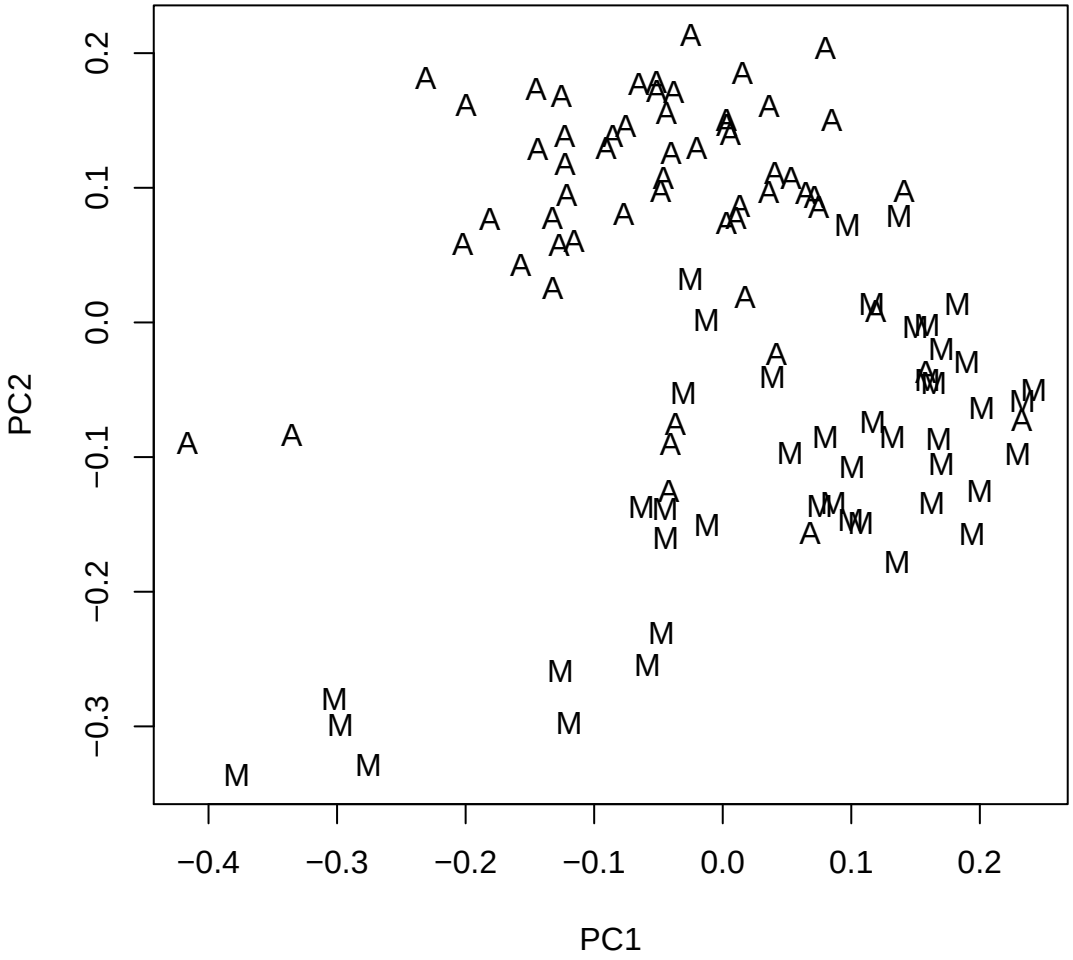
```r
km.out$cluster
```

```
##   [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 1 2 2 2 2 2 2 2 2 1 1
##  [36] 1 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 1 1 1 1 1 1 2 1 1 1 1 1 1 1
##  [71] 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 2 1 1 1 1 1 1
```

Plot the data

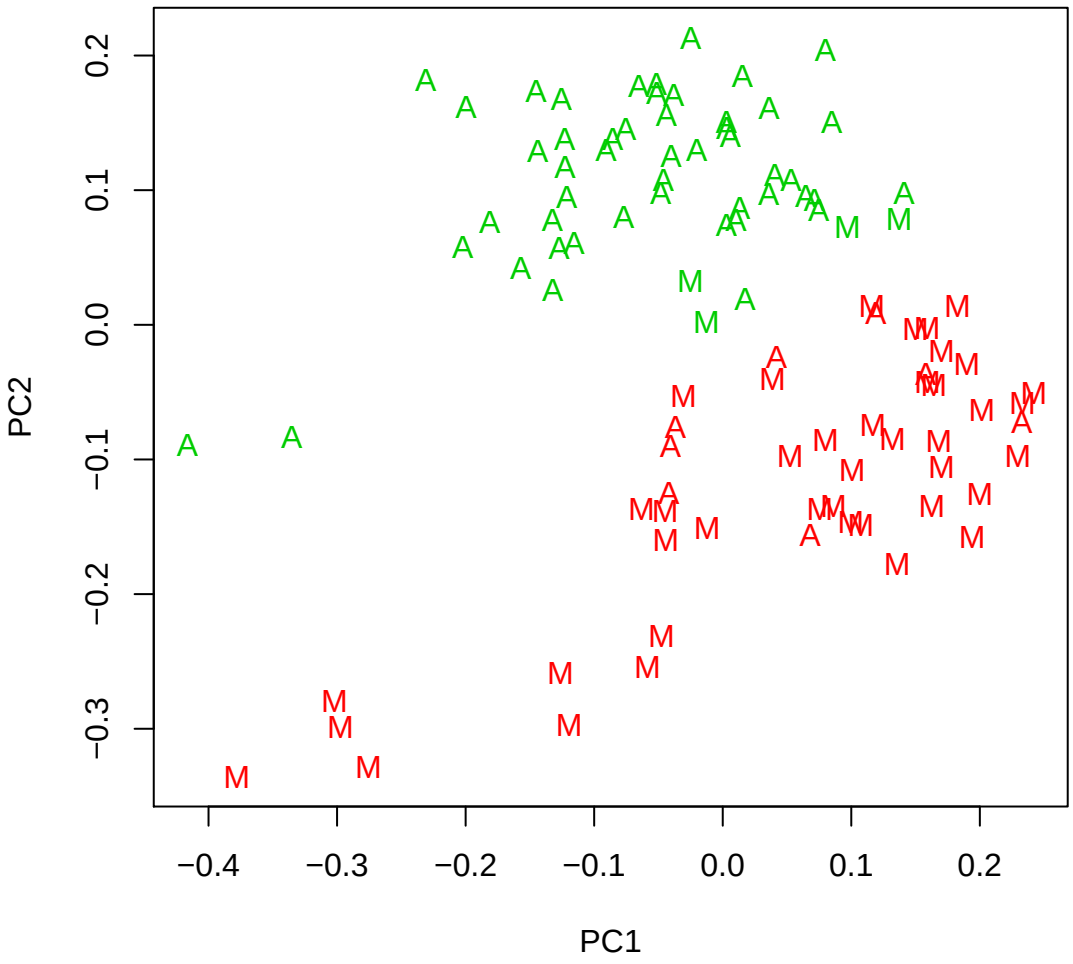
To plot the data we need to use the PCA projections. Below we will use the plot based on PCA with true labels (A for art and M for music) and compare with the plot that color the points according to the k-means clustering.

```r
# PCA with true labels
plot(nyt_pca$x[, 1:2], type = "n")
points(nyt_pca$x[nyt_data[, "class.labels"] == "art", 1:2], pch = "A")
points(nyt_pca$x[nyt_data[, "class.labels"] == "music", 1:2], pch = "M")
```



PCA with true labels but colored by k-means clustering

```r
plot(nyt_pca$x[, 1:2], type = "n")
points(nyt_pca$x[nyt_data[, "class.labels"] == "art", 1:2], pch = "A", col = (km.out$cluster +
    1)[nyt_data[, "class.labels"] == "art"])
points(nyt_pca$x[nyt_data[, "class.labels"] == "music", 1:2], pch = "M", col = (km.out$cluster +
    1)[nyt_data[, "class.labels"] == "music"])
```



Recommended exercise 4

Perform hierarchical clustering

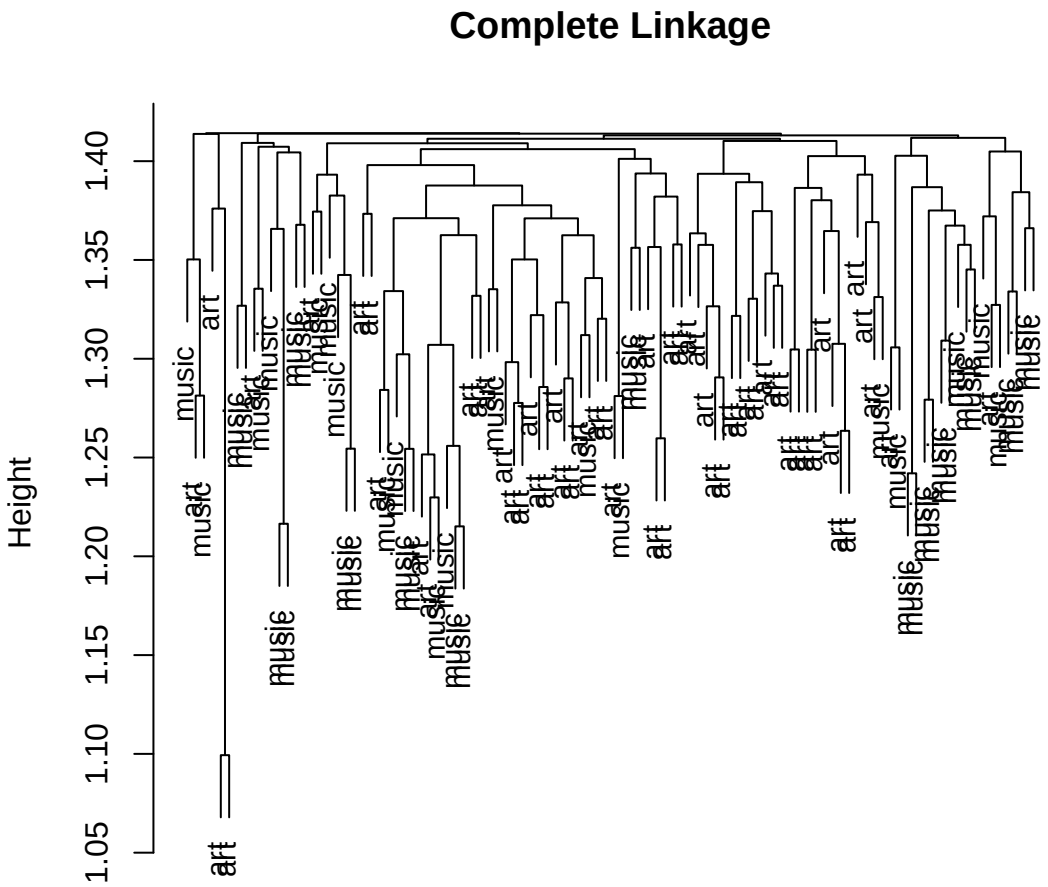
I will use euclidean distance and complete linkage

```
hc.complete = hclust(dist(nyt_data[, -1]), method = "complete")
str(hc.complete)
```

```
## List of 7
##  $ merge      : int [1:101, 1:2] -46 -84 -68 -18 -65 -23 -71 -64 -95 -37 ...
##  $ height     : num [1:101] 1.1 1.22 1.22 1.23 1.24 ...
##  $ order      : int [1:102] 94 24 93 16 46 55 59 87 39 90 ...
##  $ labels     : NULL
##  $ method     : chr "complete"
##  $ call       : language hclust(d = dist(nyt_data[, -1]), method = "complete")
##  $ dist.method: chr "euclidean"
##  - attr(*, "class")= chr "hclust"
```

Plot dendograms

```
plot(hc.complete, main = "Complete Linkage", labels = as.character(nyt_data[,
    1]), xlab = "", sub = "", cex = 0.9)
```



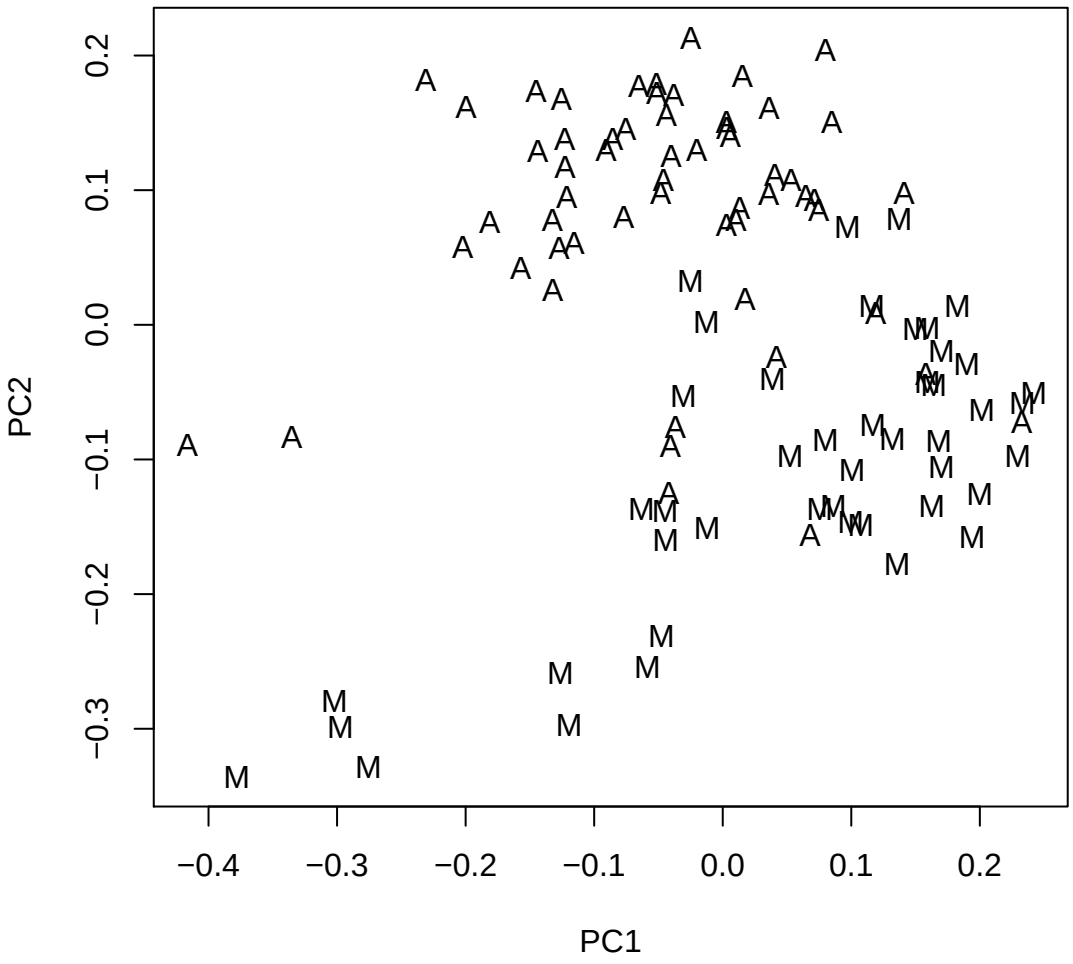
Cluster assignment

```
hc.clusters = cutree(hc.complete, 2)
```

Plot the data

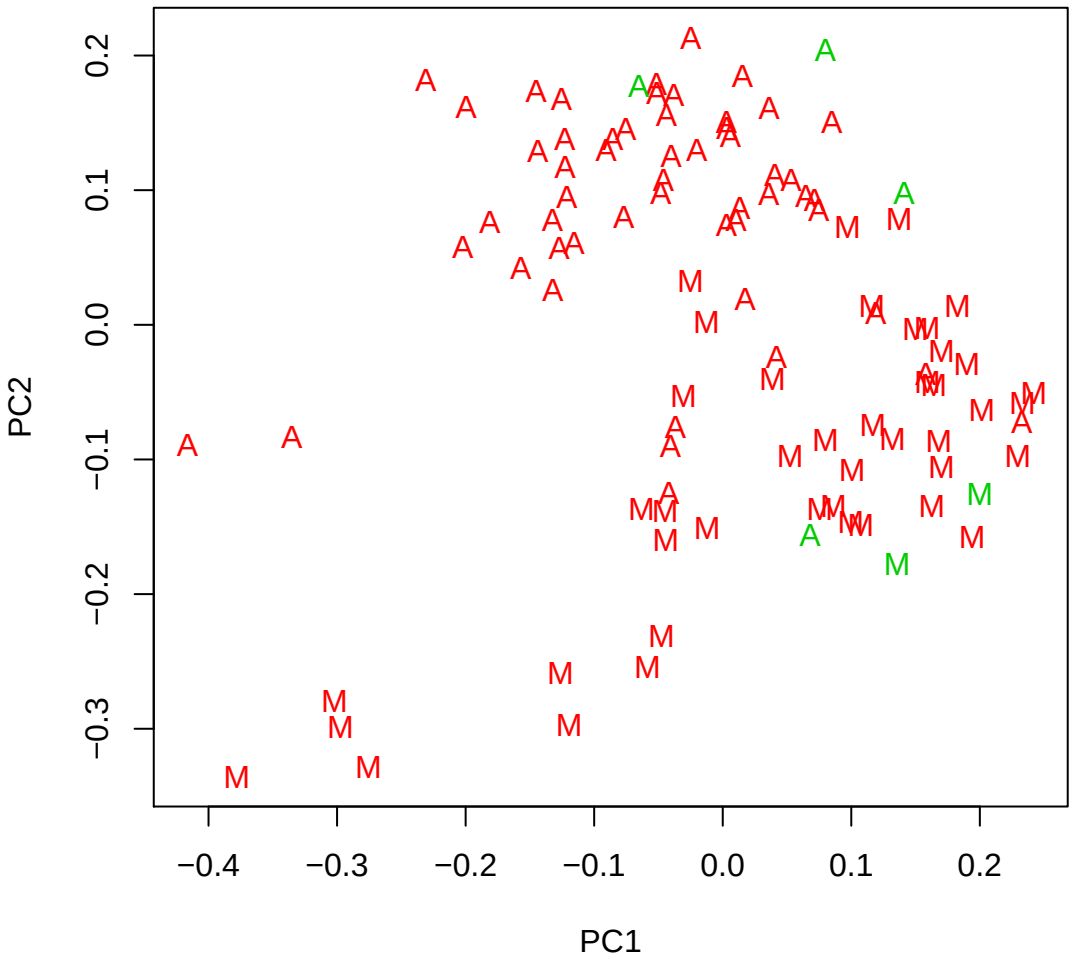
Based on the cluster assignment above and the plots below we see that hierarchical clustering performs worse than k-means for the same number of clusters (K=2)

```
# PCA with true labels
plot(nyt_pca$x[, 1:2], type = "n")
points(nyt_pca$x[nyt_data[, "class.labels"] == "art", 1:2], pch = "A")
points(nyt_pca$x[nyt_data[, "class.labels"] == "music", 1:2], pch = "M")
```



PCA with true labels but colored by k-means clustering

```
plot(nyt_pca$x[, 1:2], type = "n")
points(nyt_pca$x[nyt_data[, "class.labels"] == "art", 1:2], pch = "A", col = (hc.clusters +
    1)[nyt_data[, "class.labels"] == "art"])
points(nyt_pca$x[nyt_data[, "class.labels"] == "music", 1:2], pch = "M", col = (hc.clusters +
    1)[nyt_data[, "class.labels"] == "music"])
```

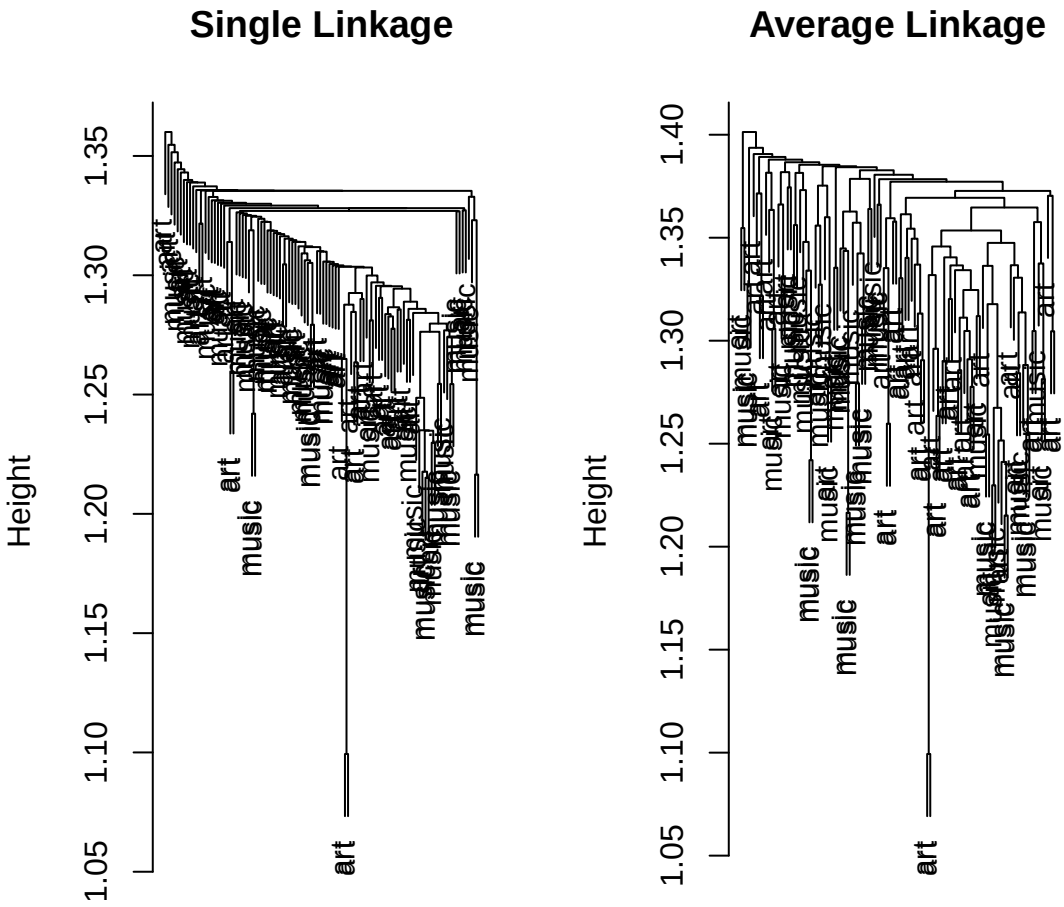


Lets see if single linkage or average linkage performs better.

```
# hierachical clustering
hc.single = hclust(dist(nyt_data[, -1]), method = "single")
hc.average = hclust(dist(nyt_data[, -1]), method = "average")
```

#plot dendogram

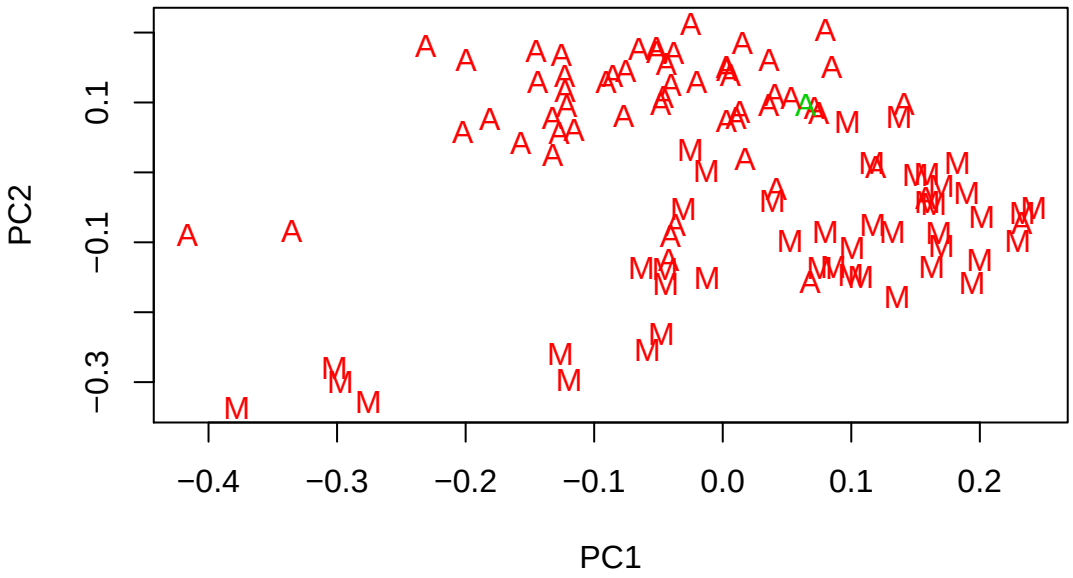
```
par(mfrow = c(1, 2))
plot(hc.single, main = "Single Linkage", labels = as.character(nyt_data[, 1]),
    xlab = "", sub = "", cex = 0.9)
plot(hc.average, main = "Average Linkage", labels = as.character(nyt_data[,
    1]), xlab = "", sub = "", cex = 0.9)
```

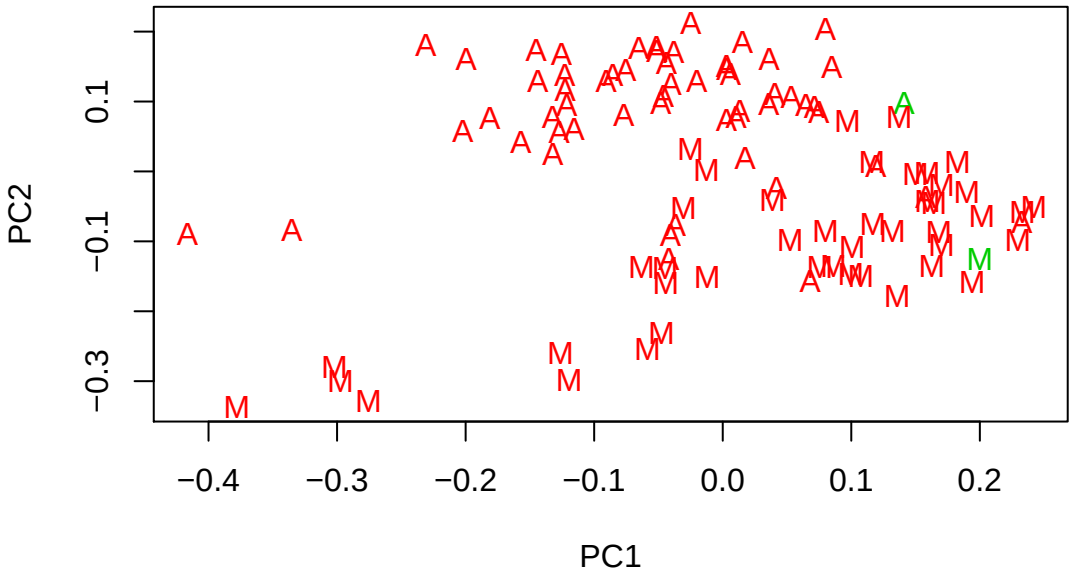



```
# divide into clusters
hc.clustersSingle = cutree(hc.single, 2)
hc.clustersAverage = cutree(hc.average, 2)
```

Plot clusters in PC1 and PC2-dimension

```
par(mfrow = c(2, 1))
plot(nyt_pca$x[, 1:2], type = "n")
points(nyt_pca$x[nyt_data[, "class.labels"] == "art", 1:2], pch = "A", col = (hc.clustersSingle +
    1)[nyt_data[, "class.labels"] == "art"])
points(nyt_pca$x[nyt_data[, "class.labels"] == "music", 1:2], pch = "M", col = (hc.clustersSingle +
    1)[nyt_data[, "class.labels"] == "music"])
plot(nyt_pca$x[, 1:2], type = "n")
points(nyt_pca$x[nyt_data[, "class.labels"] == "art", 1:2], pch = "A", col = (hc.clustersAverage +
    1)[nyt_data[, "class.labels"] == "art"])
points(nyt_pca$x[nyt_data[, "class.labels"] == "music", 1:2], pch = "M", col = (hc.clustersAverage +
    1)[nyt_data[, "class.labels"] == "music"])
```





Doesn't seem to provide a better fit than complete.