Homework 5: PCA, SVM & Clustering

Advanced Topics in Data Science II
Harvard University, Spring 2017

Tim Hagmann

March 30, 2017

Contents

Problem 1: Face recoginition

In this problem, the task is to build a facial recognition system using Principal Components Analysis (PCA) and a Support Vector Machine (SVM). We provide you with a collection of grayscale face images of three political personalities "George W. Bush", "Hugo Chavez" and "Ariel Sharon", divided into training and test data sets. Each face image is of size 250×250, and is flattened into a vector of length 62500. All the data for this problem is located in the file `CS109b-hw5-dataset_1.Rdata`. You can read this file using the `load()`

function, which will load four new variables into your environment. The vectorized images are available as rows in the arrays `imgs_train` and `imgs_test`. The identity of the person in each image is provided in the vectors `labels_train` and `labels_test`. The goal is to fit a face detection model to the training set, and evaluate its classification accuracy (i.e. fraction of face images which were recognized correctly) on the test set.

One way to perform face recognition is to treat each pixel in an image as a predictor, and fit a classifier to predict the identity of the person in the image.

Problem with high-dimensional data

The problem that could come out of this approach can be summarized under the term *curse of dimensionality*. The curse of dimensionality refers to the phenomena that arise when analyzing data in a very high-dimensional spaces. The problem is that when the dimensionality increases, the volume of the space increases so fast that the available data become sparse. More specifically, the amount of data needed to support the result grows exponentially with the dimensionality.

Dimensionality reduction with PCA

Instead we recommend working with low-dimensional representations of the face images computed using PCA. This can be done by calculating the top K principal components (PCs) for the vectorized face images in the training set, projecting each training and test image onto the space spanned by the PC vectors, and represent each image using the K projected scores. The PC scores then serve as predictors for fitting a classification model. Why might this approach of fitting a classification model to lower dimensional representations of the images be more beneficial?

The main linear technique for dimensionality reduction is principal component analysis (PCA). PCA performs a linear mapping of the data to a lower-dimensional space in such a way that the variance of the data in the low-dimensional representation is maximized. As long as the problem is linear or there is a linear hyperplane present in the feature space, the PCA methods helps reducing the amount of dimensions and therefore migigates the course of dimensionality problem. Furthermore, decreasing the dimensionality also decreases the compution time.

Initialize

In the following code chunk all the necessary setup for the modelling environment is done.

```
## Options
options(scipen = 10)                          # Disable scientific notation
update_package <- FALSE                       # Use old status of packages

## Init files (always execute, eta: 10s)
source("scripts/01_init.R")                   # Helper functions to load packages
source("scripts/02_packages.R")               # Load all necessary packages
source("scripts/03_functions.R")              # Load project specific functions
```

Load the data

```
## Read data
load("data/CS109b-hw5-dataset_1.Rdata")
```

Application of PCA

Apply PCA to the face images in `imgs_train`, and identify the top 5 principal components. Each PC has the same dimensions as a vectorized face image in the training set, and can be reshaped into a 250 x 250 image, referred to as an *Eigenface*.

Note: Applying the prcomp function trough *predict* on new data, the same mean/variance scaling is used derived from the training set.

```
# Fit the model
fit_pca <- prcomp(imgs_train, scale=TRUE, center=TRUE)

# Get the amount on variance
pca_variance <- fit_pca$sdev^2/sum(fit_pca$sdev^2)

# Print
print(paste0(sprintf("Top 5 principal components explain: %.1f",
                     sum(pca_variance[1:5]) * 100),
             "% of the variance"))
```

```
## [1] "Top 5 principal components explain: 42.2% of the variance"
```

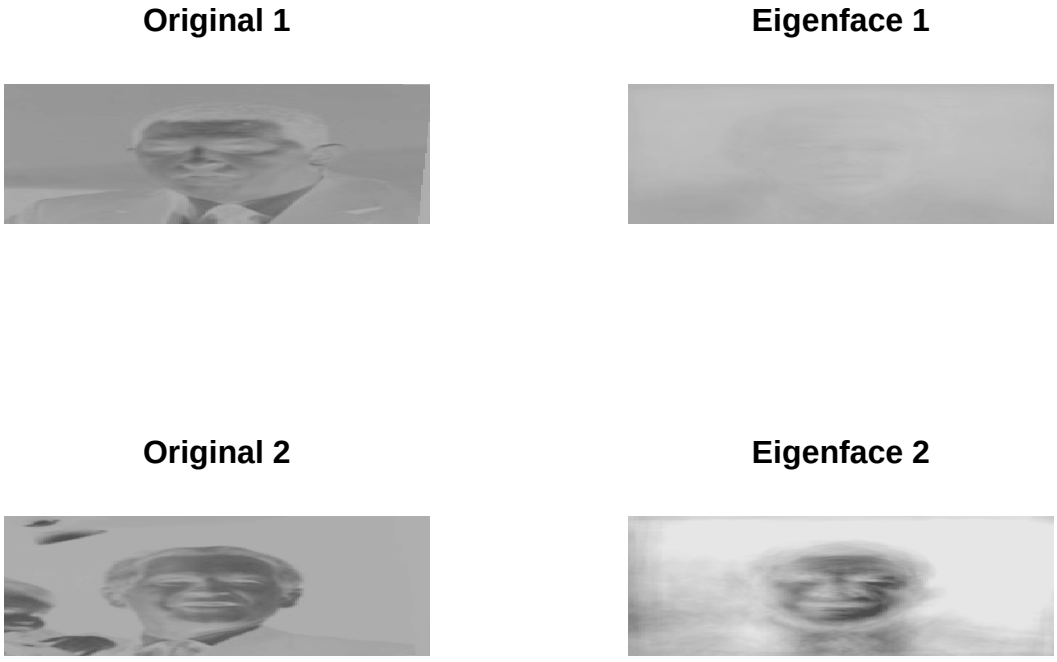
Extract top 5 pc's

```
eig_faces_5pc <- fit_pca$x[, 1:5] %*% t(fit_pca$rotation[, 1:5])
```

Visualize eigenface 1

```
# Visualize image 1
par(mfrow=c(2, 2))
plot.face(imgs_train[1, ], main="Original 1")
plot.face(eig_faces_5pc[1, ], main="Eigenface 1")

# Visualize image 2
plot.face(imgs_train[2, ], main="Original 2")
plot.face(eig_faces_5pc[2, ], main="Eigenface 2")
```



The above plots shows, that the amount of retained information has been reduced heavily through the application of pca. Especially the face of Hugo Chavez is barely noticable. Nevertheless, the outlines of the nose, eyes, mouth, head conture as well as other facial features are still visible.

Retain PC's with 90% varation

Retain the top PCs that contribute to 90% of the variation in the training data. How does the number of identified PCs compare with the total number of pixels in an image? Compute the PC scores for each image in the training and test set, by projecting it onto the space spanned by the PC vectors.

Calculate the pc's retaining 90% of the variation

```
top_pca <- 1:length(pca_variance)
top_pca <- top_pca[cumsum(pca_variance) <= 0.9]
print(max(top_pca))
```

```
## [1] 108
```

Instead of using 339 Columns with pixels only 108 are sufficient in order to retain 90% of the variability in the data. This helps migigating the curse of hyperdimensionality problem in the data.

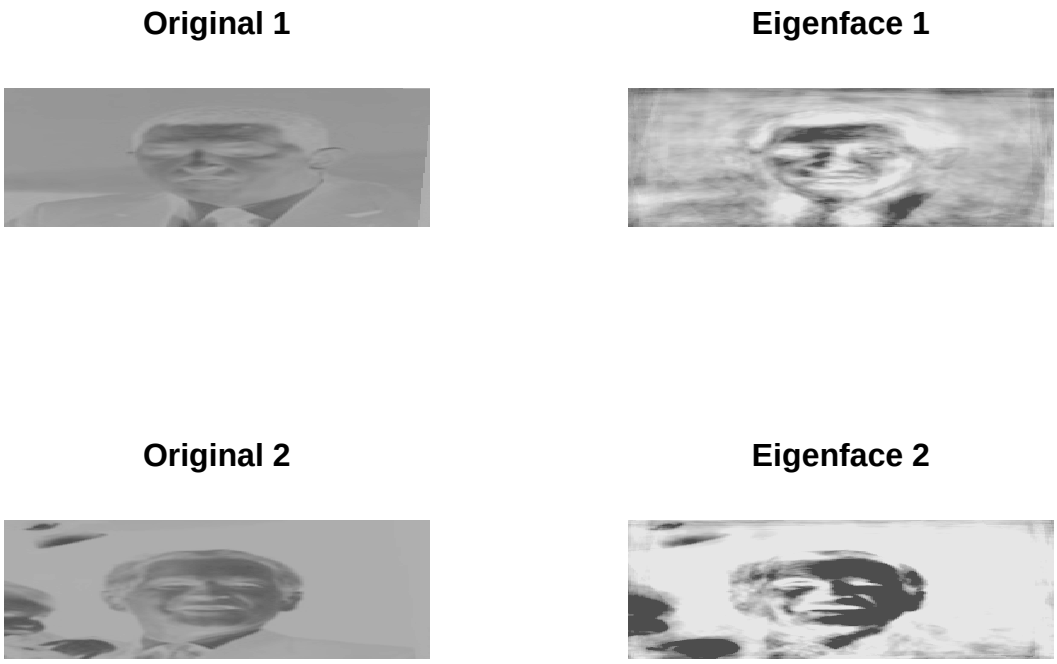
Extract 90% variance pc's

```
top_eig_faces <- fit_pca$x[, top_pca] %*% t(fit_pca$rotation[, top_pca])
```

Visualize the eigenfaces

```
# Visualize image 1
par(mfrow=c(2, 2))
plot.face(imgs_train[1, ], main="Original 1")
plot.face(top_eig_faces[1, ], main="Eigenface 1")

# Visualize image 2
plot.face(imgs_train[2, ], main="Original 2")
plot.face(top_eig_faces[2, ], main="Eigenface 2")
```



The above plot shows, that retainign 90% of the variabiltiy in the data preseves much more of the information than simply using the top 5 pc's. Nevertheless, the big advantage is, that the number of identified pc's is still much less than the number of columns/pixels. The svm can therefore perform it's calculations in a much lower dimensionality realm than with all the pixels.

Train a SVM model

Treating the PC scores as predictors, fit a SVM model to the the training set, and report the classification accuracy of the model on the test set. How does the accuracy of the fitted model compare to a naïve classifier

that predicts a random label for each image?

Transforming the test and training data

```
df_pca_train <- data.frame(labels_train, predict(fit_pca, imgs_train)[, top_pca])
df_pca_test <- data.frame(labels_test, predict(fit_pca, imgs_test)[, top_pca])
```

Train a svm with 5 fold cv

```
# Set parameters
costs <- 10^c(-10:10)
gammas <- 10^c(-10:10)

# Fit model
fit_svm_cv <- tune(svm, labels_train ~ .,
                   data=df_pca_train,
                   tunecontrol=tune.control(cross=5),
                   ranges=list(cost=costs, gamma=gammas,
                               kernel='radial'))
```

Predict accuracy

```
# Naive classifier
pred_naive <- sample(x=("George_W_Bush"), size=length(labels_test), replace=TRUE)
vec_union <- union(pred_naive, labels_test)

# Best SVM
pred_svm <- predict(fit_svm_cv$best.model, df_pca_test)

# Confusion matrix
cm_naive <- confusionMatrix(table(factor(pred_naive, vec_union),
                                  factor(labels_test, vec_union)))
cm_svm <- confusionMatrix(table(pred_svm, labels_test))

# Output
pander(cm_naive$table)
```

	George_W_Bush	Ariel_Sharon	Hugo_Chavez
George_W_Bush	264	38	37
Ariel_Sharon	0	0	0
Hugo_Chavez	0	0	0

```
pander(cm_naive$overall)
```

Table 2: Table continues below

Accuracy	Kappa	AccuracyLower	AccuracyUpper	AccuracyNull
0.7788	0	0.7308	0.8218	0.7788

AccuracyPValue	McnemarPValue
0.5309	NA

```
pander(cm_svm$table)
```

	Ariel_Sharon	George_W_Bush	Hugo_Chavez
Ariel_Sharon	29	3	0
George_W_Bush	8	260	9
Hugo_Chavez	1	1	28

```
pander(cm_svm$overall)
```

Table 5: Table continues below

Accuracy	Kappa	AccuracyLower	AccuracyUpper	AccuracyNull
0.9351	0.811	0.9034	0.9589	0.7788

AccuracyPValue	McnemarPValue
3.912e-15	0.02156

The naive classifier performs worse than the svm in terms of overall accuracy. This indicates, that the principal component analysis, with a 90% variation threshold, is able to preserve enough information to achive a high accuracy rate.

Problem 2: Analyzing Voting Patterns of US States

In this problem, we shall use unsupervised learning techniques to analyze voting patterns of US states in six presidential elections. The data set for the problem is provided in the file `CS109b-hw5-dataset_2.txt`. Each row represents a state in the US, and contains the logit of the relative fraction of votes cast by the states for Democratic presidential candidates (against the Republican candidates) in elections from 1960 to 1980. The logit transformation was used to expand the scale of proportions (which stay between 0 and 1) to an unrestricted scale which has more reliable behavior when finding pairwise Euclidean distances. Each state is therefore described by 6 features (years). The goal is to find subgroups of states with similar voting patterns.

Part 2a: Visualize data

Generate the visualizations to analyze important characteristics of the data set

Read data

```
df_2 <- read.table("data/CS109b-hw5-dataset_2.txt", header=TRUE)
```

Rescaling

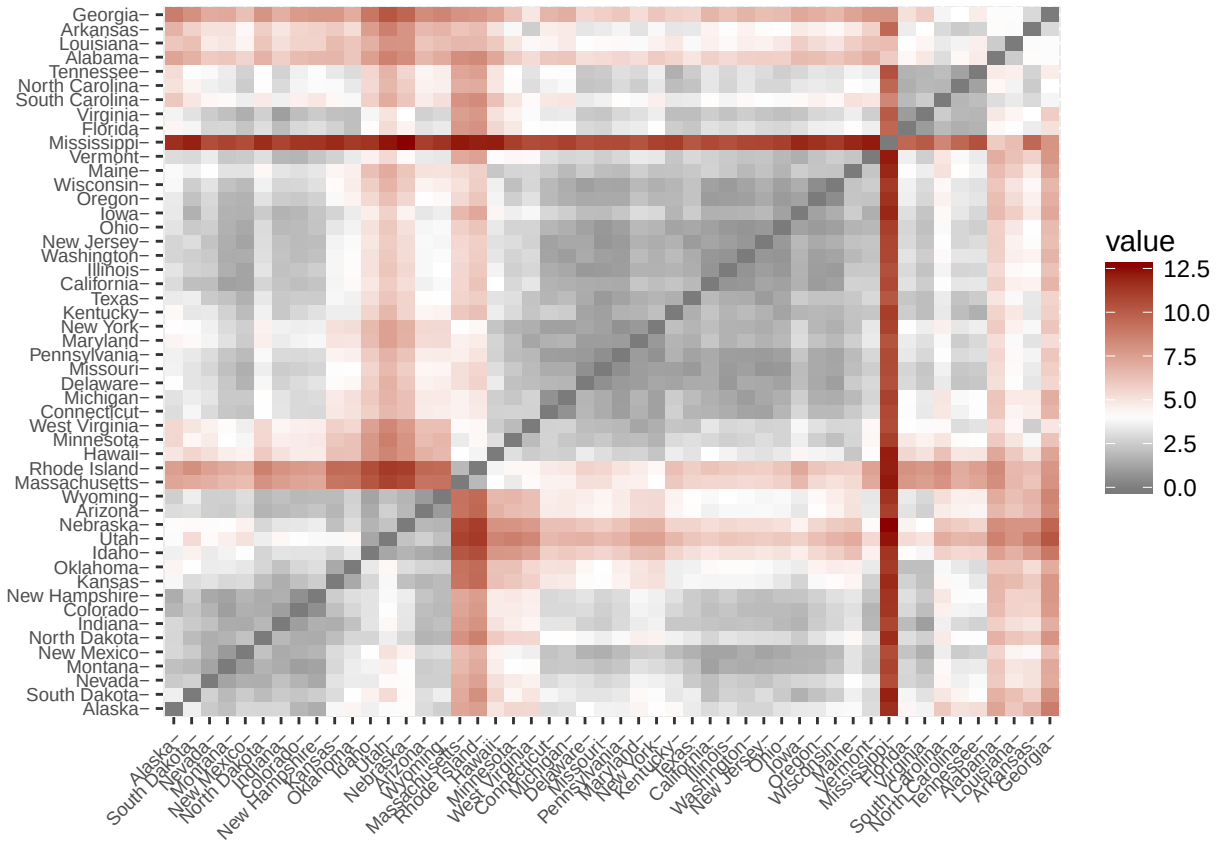
Rescale the data, and compute the Euclidean distance between each pair of states. Generate a heat map of the pair-wise distances.

Calculate euclidean distance

```
dist_euclidean <- daisy(df_2, metric="euclidean", stand=TRUE)
print("Plot I: Heat map of the pair-wise distances")
```

```
## [1] "Plot I: Heat map of the pair-wise distances"
```

```
fviz_dist(dist_euclidean,
          gradient=list(low="black", mid="white", high="darkred"),
          lab_size=7)
```



The above plot shows, that there are different Republican and Democratic Clusters present.

Apply MDS

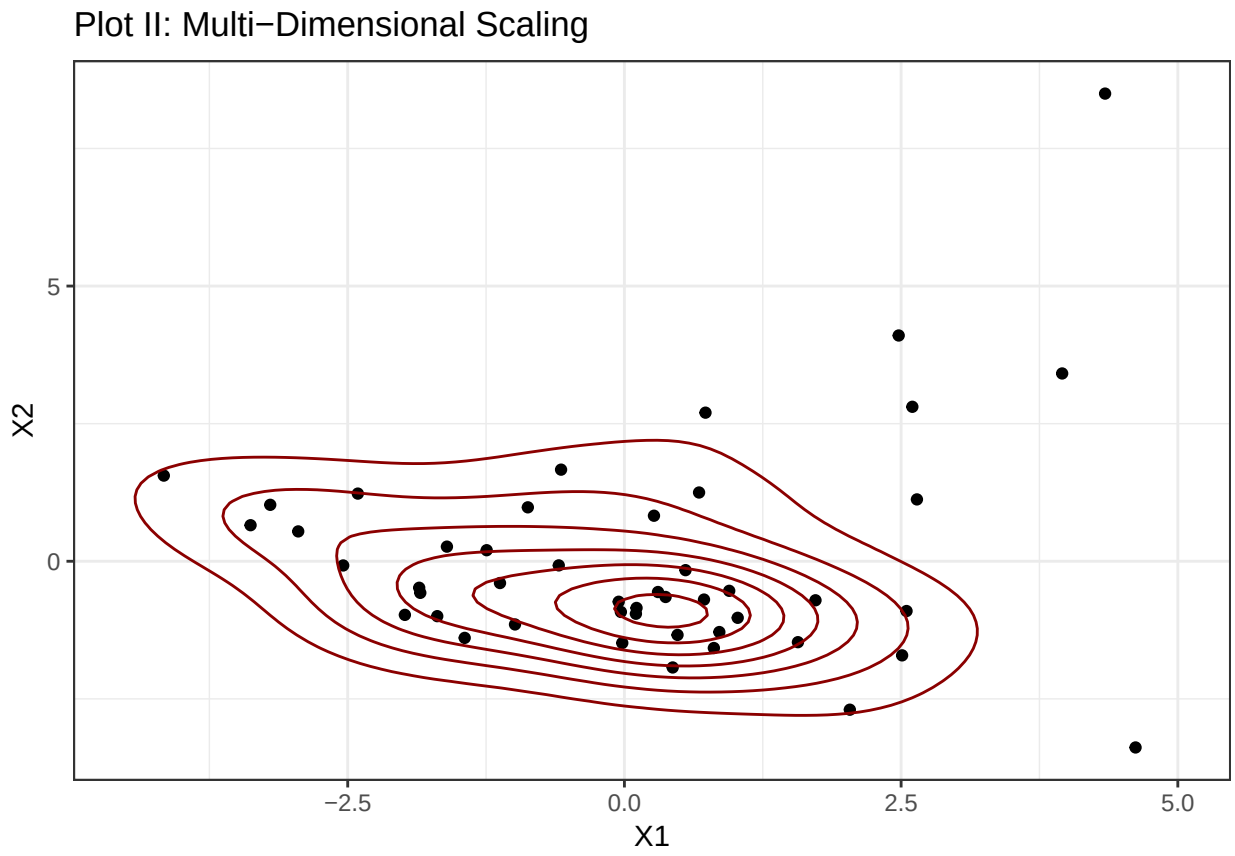
Apply multi-dimensional scaling to the pair-wise distances, and generate a scatter plot of the states in two dimension.

Calculate multi-dimensional scaling (MDS)

```
vec_mds <- cmdscale(dist_euclidean)
colnames(vec_mds) <- c("X1", "X2")
```

Visualize

```
ggplot(data=vec_mds, aes(x=X1, y=X2)) +
  geom_point() +
  geom_point(size=0.9) +
  geom_density2d(color="darkred") +
  ggtitle("Plot II: Multi-Dimensional Scaling") +
  xlim(-4.5, 5) +
  theme_bw()
```



The above conture plot shows only one cluster, however a couple of outliers appear to be present in the right part of the plot.

Apply PCA

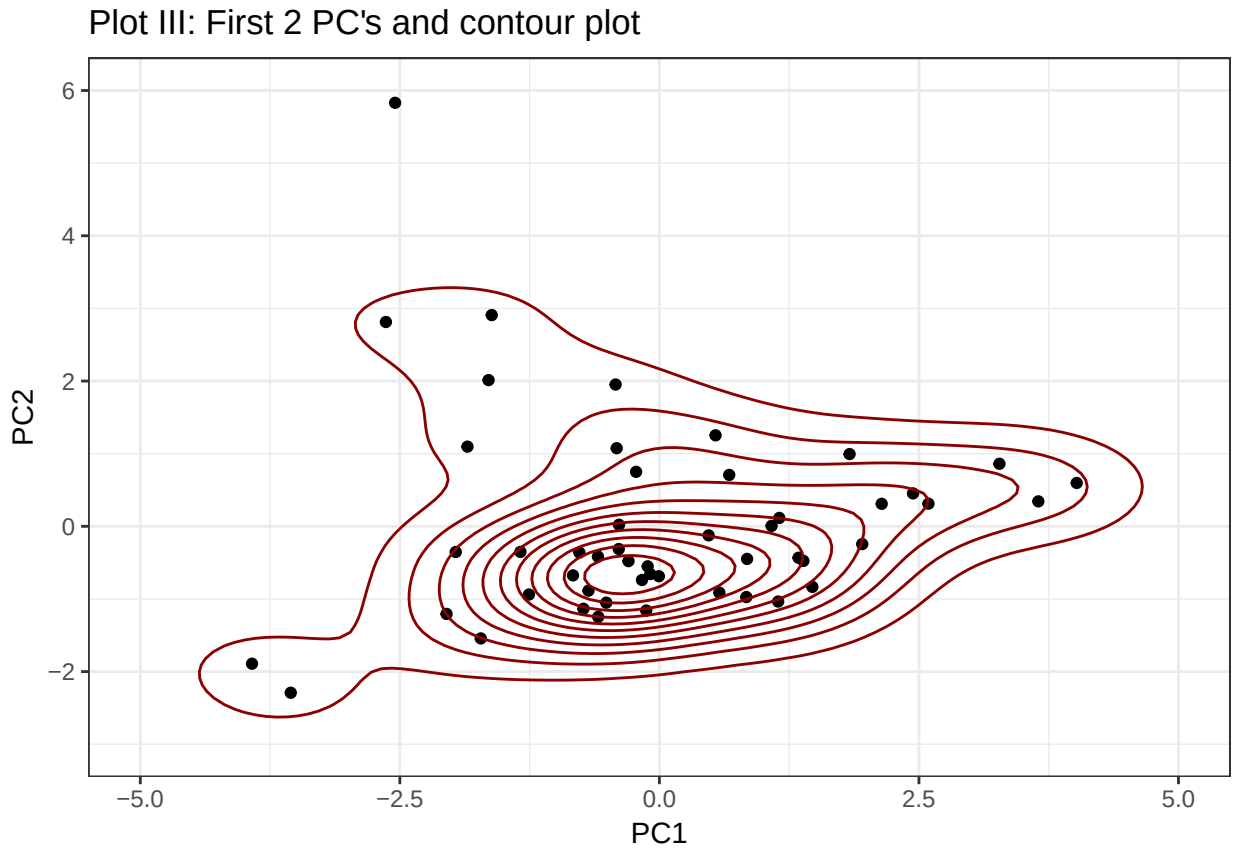
Apply PCA to the data, and generate a scatter plot of the states using the first two principal components. Add a 2d-density estimation overlay to the plot via the `geom_density2d` function.

Apply PCA using

```
fit_pca <- prcomp(df_2, scale=TRUE, center=TRUE)
pred_pca <- predict(fit_pca, df_2)[, 1:2]
```

Visualize

```
ggplot(data=pred_pca, aes(x=PC1, y=PC2)) +
  geom_point() +
  geom_point(size=0.9) +
  geom_density2d(color="darkred") +
  ggtitle("Plot III: First 2 PC's and contour plot") +
  xlim(-5, 5) +
  ylim(-3, 6) +
  theme_bw()
```



In the case of the PCA conture plot also only one cluster is visible. again some outliers appear to be visible.

Summary

Summarize the results of these visualizations. What can you say about the similarities and differences among the states with regard to voting patterns? By visual inspection, into how many groups do the states cluster?

The first visualizations shows some clustering on the edges as well as the middle. This makes sense as we

would expect that some states always vote republican while others always democrat. Then there are also flip-states which switch between republican and democrat between the years. For the MDA and PCA plot there are no clear clusters apparent. Visually inspecting the mca plot, there might be a cluster arround the outlier points and main points at around (2.5, -5) and (0.2, 0.2). For the PCA plot there might be a cluster at around (-0.2, -0.2), (2.5, 0.9) and (-2, 2).

Part 2b: Partitioning clustering

Apply the following partitioning clustering algorithms to the data:

- **K-means clustering**
- **Partitioning around medoids (PAM)**

In each case, determine the optimal number of clusters based on the Gap statistic, considering 2 to 10 clusters. Also determine the choice of the optimal number of clusters by producing elbow plots. Finally, determine the optimal number of clusters using the method of average silhouette widths. Do the choices of these three methods agree? If not, why do you think you are obtaining different suggested numbers of clusters?

K-Means clustering

K-Means gap statistic

```
set.seed(123)
gapstat_kmeans <- clusGap(scale(df_2),
                          FUN=kmeans,
                          K.max=10,
                          nstart=10,
                          B=500)
```

Show Tibshirani criteria

```
print(gapstat_kmeans, method = "Tibs2001SEmax")
```

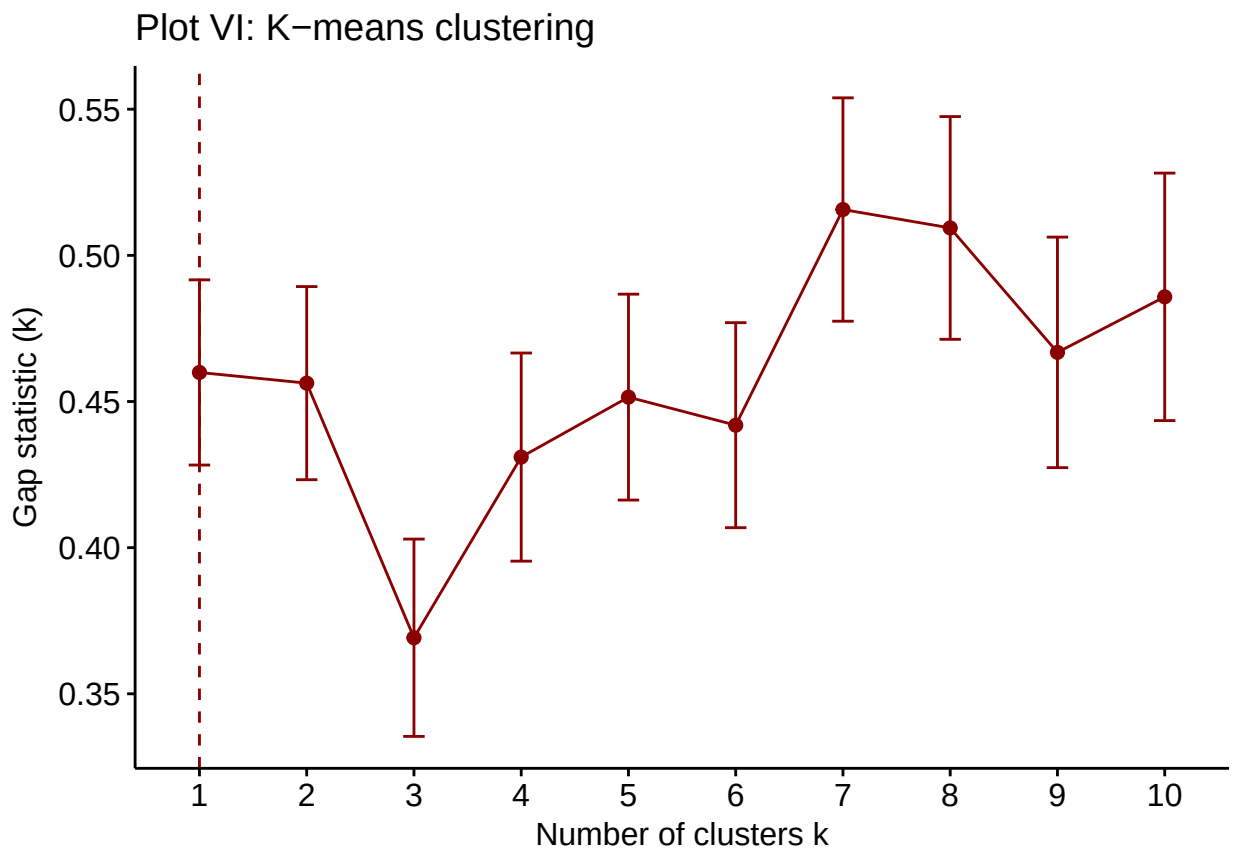
```
## Clustering Gap statistic ["clusGap"] from call:
## clusGap(x = scale(df_2), FUNcluster = kmeans, K.max = 10, B = 500,     nstart = 10)
## B=500 simulated reference sets, k = 1..10; spaceH0="scaledPCA"
##  --> Number of clusters (method 'Tibs2001SEmax', SE.factor=1): 1
##           logW   E.logW       gap     SE.sim
##  [1,] 3.620266 4.084185 0.4639189 0.03674221
##  [2,] 3.402462 3.851821 0.4493592 0.03625068
##  [3,] 3.235942 3.678870 0.4429286 0.03348415
##  [4,] 3.127166 3.555646 0.4284794 0.03151903
##  [5,] 3.020645 3.467041 0.4463965 0.03259480
##  [6,] 2.909188 3.389494 0.4803061 0.03309631
##  [7,] 2.819657 3.319197 0.4995398 0.03354795
##  [8,] 2.743882 3.254187 0.5103054 0.03409393
##  [9,] 2.675484 3.192643 0.5171590 0.03466202
## [10,] 2.611223 3.134713 0.5234902 0.03575866
```

The Tibshirani criterion has the idea that for a particular choice of K clusters, the total within cluster variation is compared to the expected within-cluster variation. According to this creterion one cluster is optimal.

K-Means gap stat

```
# Calculation
nbc_clust_elb <- fviz_nbclust(scale(df_2), kmeans,
                 method="gap_stat", linecolor="darkred",
                 k.max=10)

# Visualization
nbc_clust_elb + ggtitle("Plot VI: K-means clustering")
```

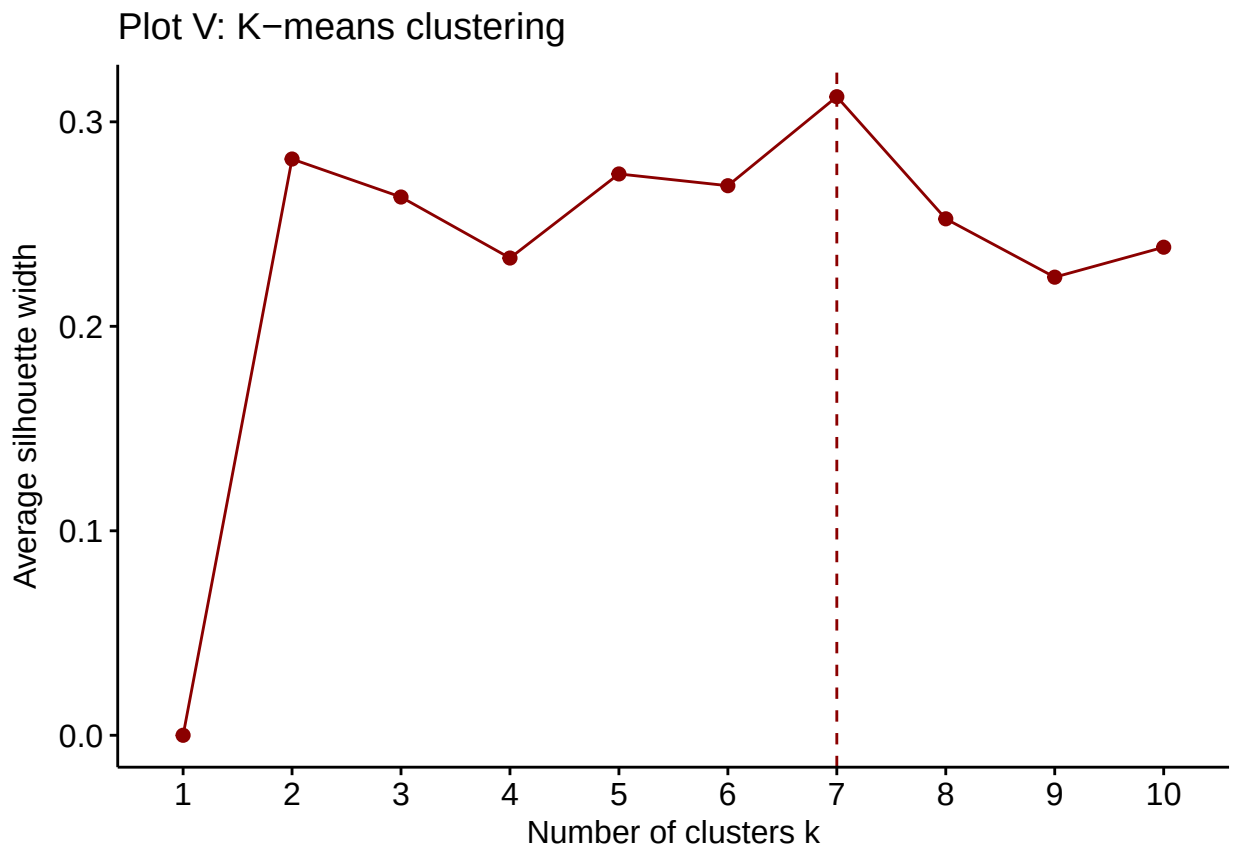


The above plot indicates a optimial amount of clusters at 1. However, there is also a elbow present at 2.

K-Means silouhette

```
# Calculation
nbc_clust_sil <- fviz_nbclust(scale(df_2), kmeans,
                 method="silhouette", linecolor="darkred",
                 k.max=10)
```

```
# Visualization
nbc_clust_sil + ggtitle("Plot V: K-means clustering")
```



The above plot shows, that the optimal number of clusters (siloutte) lies at seven.

Summary

The silouette method and the gap stat method do not agree on the optimal number of clusters. The gap stat methods suggest one cluster and the silhouette method seven clusters. This is because the computation for each method is different enough that inconsistent results can occur. However, it has to be noted, that both approaches measure global clustering characteristics only and are rather informal.

Partitioning around medoids (PAM)

```
gapstat_pam <- clusGap(scale(df_2),FUN=pam, K.max=10, B=500)
```

Print output

```
print(gapstat_pam, method = "Tibs2001SEmax")
```

```
## Clustering Gap statistic ["clusGap"] from call:
## clusGap(x = scale(df_2), FUNcluster = pam, K.max = 10, B = 500)
## B=500 simulated reference sets, k = 1..10; spaceH0="scaledPCA"
```

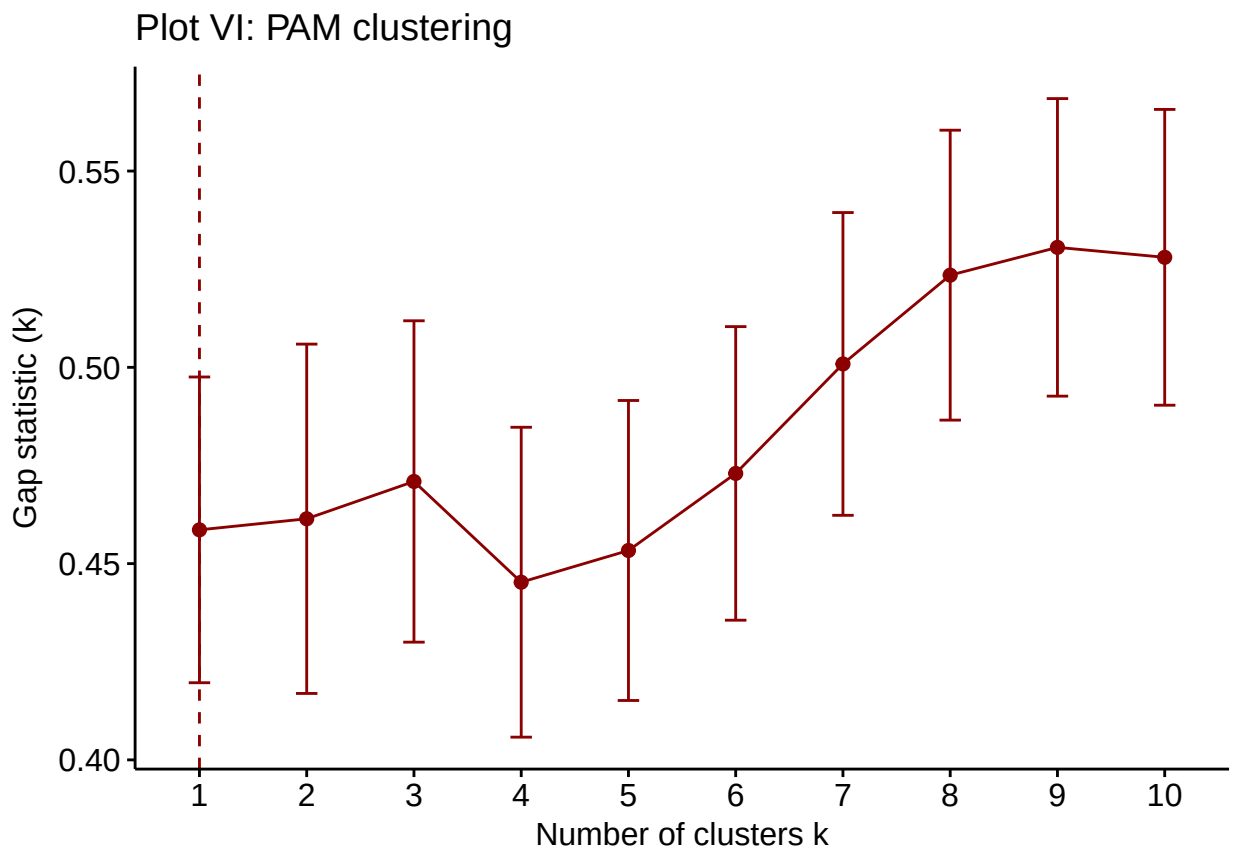
```
##  --> Number of clusters (method 'Tibs2001SEmax', SE.factor=1): 1
##           logW   E.logW       gap     SE.sim
##  [1,] 3.620266 4.079857 0.4595910 0.04036532
##  [2,] 3.405836 3.866350 0.4605143 0.04388669
##  [3,] 3.225686 3.696682 0.4709957 0.04203696
##  [4,] 3.130698 3.580510 0.4498125 0.03972230
##  [5,] 3.036721 3.492508 0.4557872 0.03727355
##  [6,] 2.940884 3.416789 0.4759048 0.03703718
##  [7,] 2.843135 3.347827 0.5046918 0.03752172
##  [8,] 2.754444 3.282813 0.5283691 0.03793385
##  [9,] 2.687968 3.220962 0.5329941 0.03909383
## [10,] 2.631513 3.162317 0.5308042 0.03914881
```

Again, the optimal amount of clusters for pam is set at 1.

PAM gap statistic

```
# Calculation
nbc_clust_gap <- fviz_nbclust(scale(df_2), pam,
                 method="gap_stat", linecolor="darkred",
                 k.max=10)

# Visualization
nbc_clust_gap + ggtitle("Plot VI: PAM clustering")
```

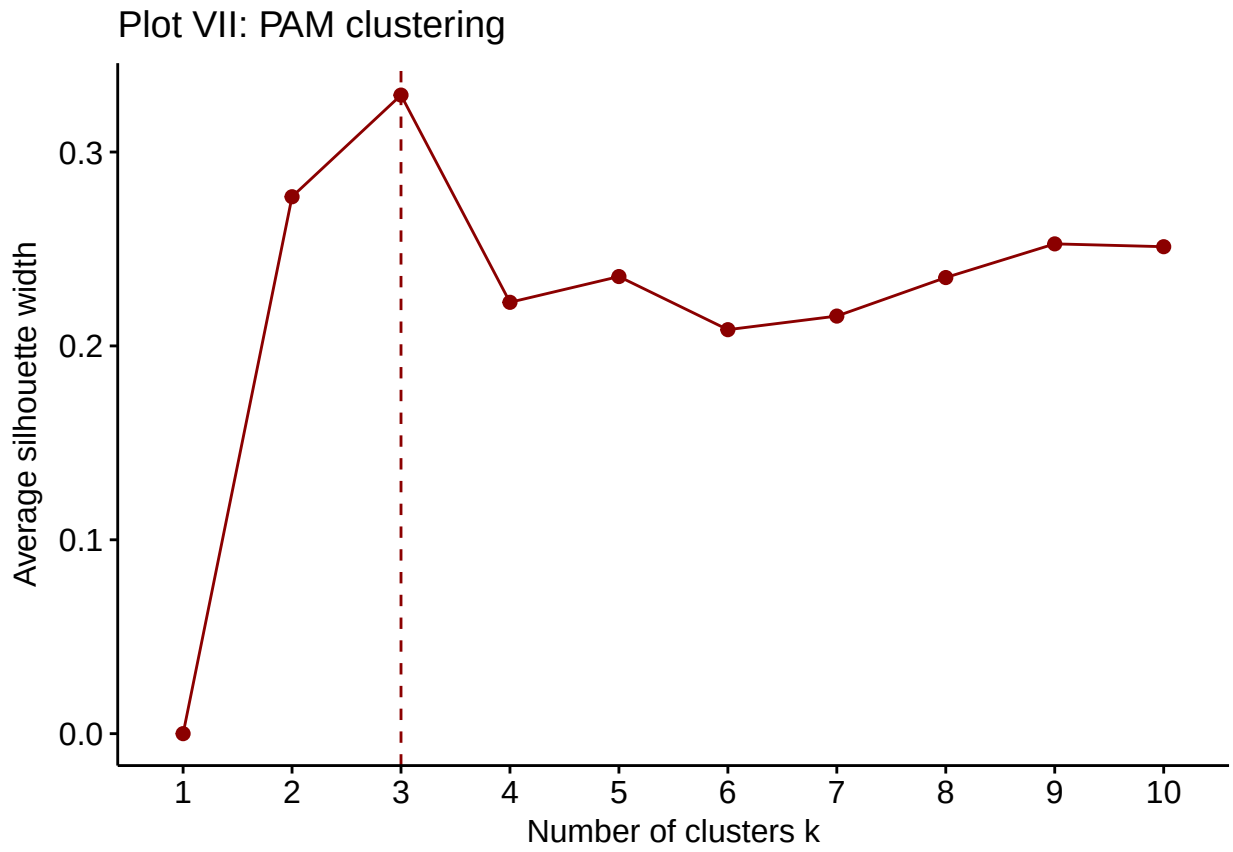


Optimal number of clusters is set at 1. However, there is also a clear elbow visible at 3.

PAM silouhette

```
# Calculation
nbc_clust_sil <- fviz_nbclust(scale(df_2), pam,
                 method="silhouette", linecolor="darkred",
                 k.max=10)

# Visualization
nbc_clust_sil + ggtitle("Plot VII: PAM clustering")
```



The above plot indicates a optimal number of clusters at 3.

Summary

The different methods for the pam algorithm do not agree on the optimal amount of clusters. The Gap Stat indicates 1 cluster and the silouhette statistic 3 clusters.

Principal Components Plot

With your choice of the number of clusters, construct a principal components plot the clusters for *K-means* and *PAM* using the `fviz_cluster` function. Are the clusterings the same? Summarize the results of the clustering including any striking features of the clusterings.

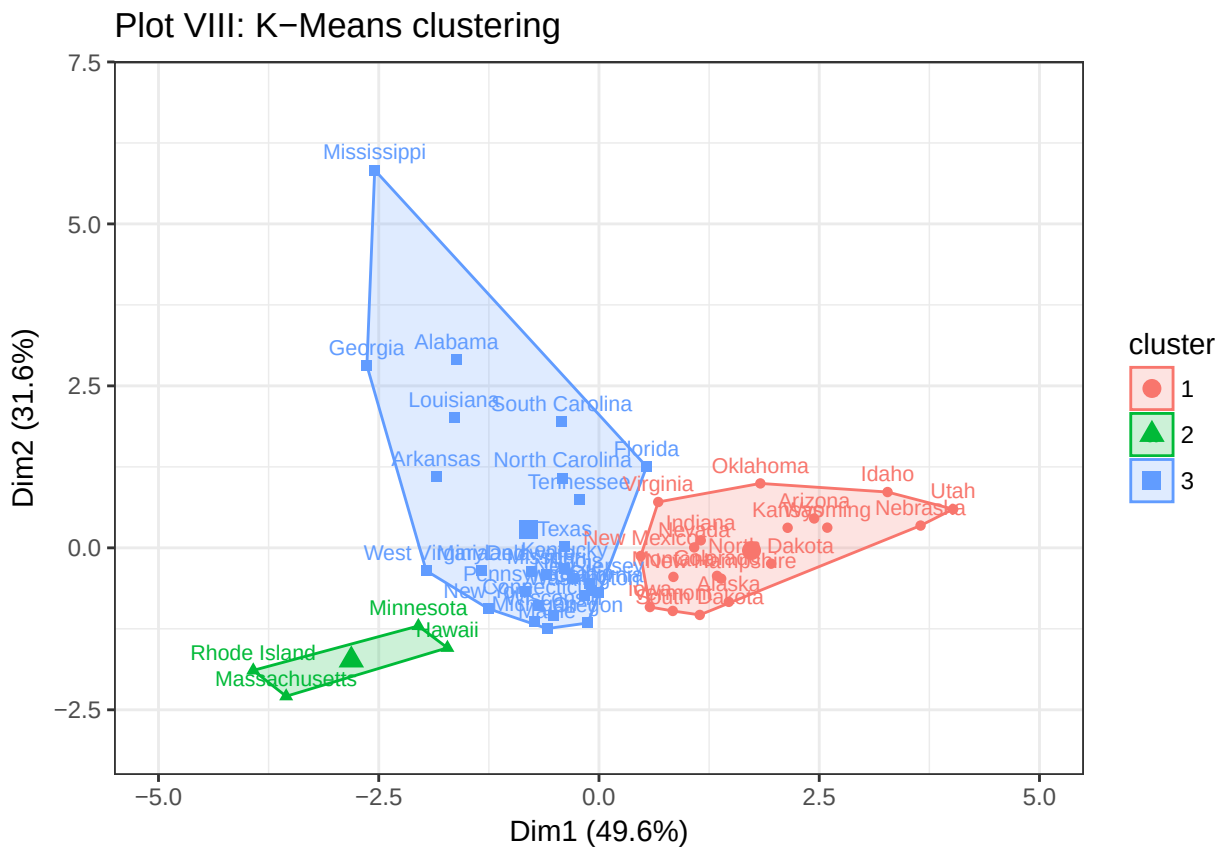
Calculate

```
# Options
opt_clust <- 3
sel_kmeans <- kmeans(scale(df_2), opt_clust)
sel_pam <- pam(scale(df_2), opt_clust)

# Calculation
fviz_clust_kmeans <- fviz_cluster(sel_kmeans,
                                  data=scale(df_2),
                                  linecolor="darkred",
                                  labelsize=8)
fviz_clust_pam <- fviz_cluster(sel_pam,
                               data=scale(df_2),
                               linecolor="darkred",
                               labelsize=8)
```

Visualize K-Mean

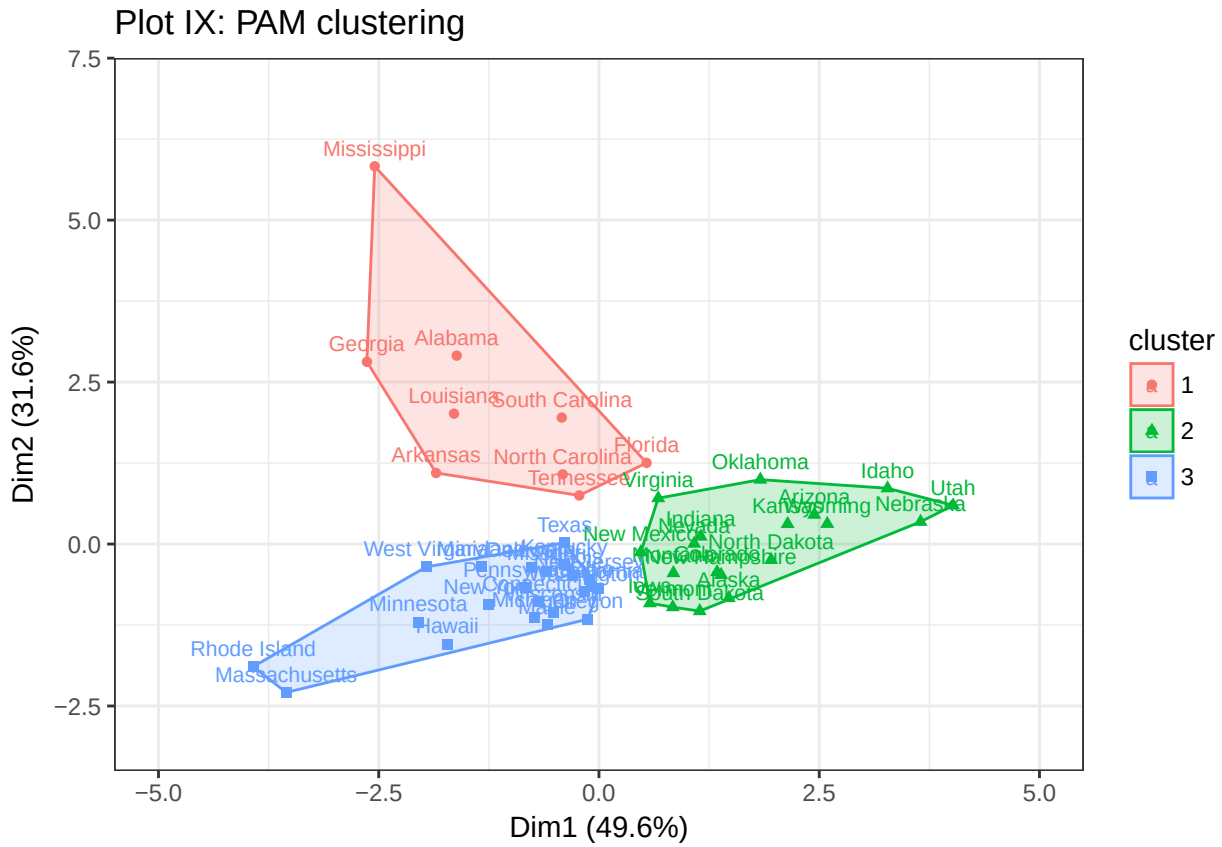
```
fviz_clust_kmeans +
  ggtitle("Plot VIII: K-Means clustering") +
  theme_bw() +
  xlim(-5, 5) +
  ylim(-3, 7)
```



The k-means clustering appears to cluster the 'mostly' Democratic states into their own cluster, while grouping the other Democratic states together with some of the most Republican states such as Alabama, Mississippi and Georgia.

Visualize PAM

```
fviz_clust_pam +
  ggtitle("Plot IX: PAM clustering") +
  theme_bw() +
  xlim(-5, 5) +
  ylim(-3, 7)
```



The pam cluste appears to cluster together the mostly Republican state together. On the other hand the left cluster also includes Democratic states along with Republican states such as Texas.

Overall: While definitly not congruent the clusters appear to be rather similar. The main difference is, that the center of the respective cluster lie at a different location. Furthermore, the top cluster appears to be much bigger for the k-means algorithm compared to the pam algorithm.

Silhouette Plots

Generate silhouette plots for the *K-means* and *PAM* clusterings with the optimal number of clusters. Identify states that may have been placed in the wrong cluster.

Missclassified states

```
# Calculate
sil_kmeans <- silhouette(sel_kmeans$cluster, dist(scale(df_2)))[, 3]
sil_pam <- silhouette(sel_pam)[, 3]

# Index
index_neg_sil_kmeans <- which(sil_kmeans < 0)

# Print missclassified states
print("K-Means misclassified states:")
```

```
## [1] "K-Means misclassified states:"
```

```
print(row.names(df_2)[index_neg_sil_kmeans])
```

```
## [1] "Florida"
```

```
print("PAM misclassified states:")
```

```
## [1] "PAM misclassified states:"
```

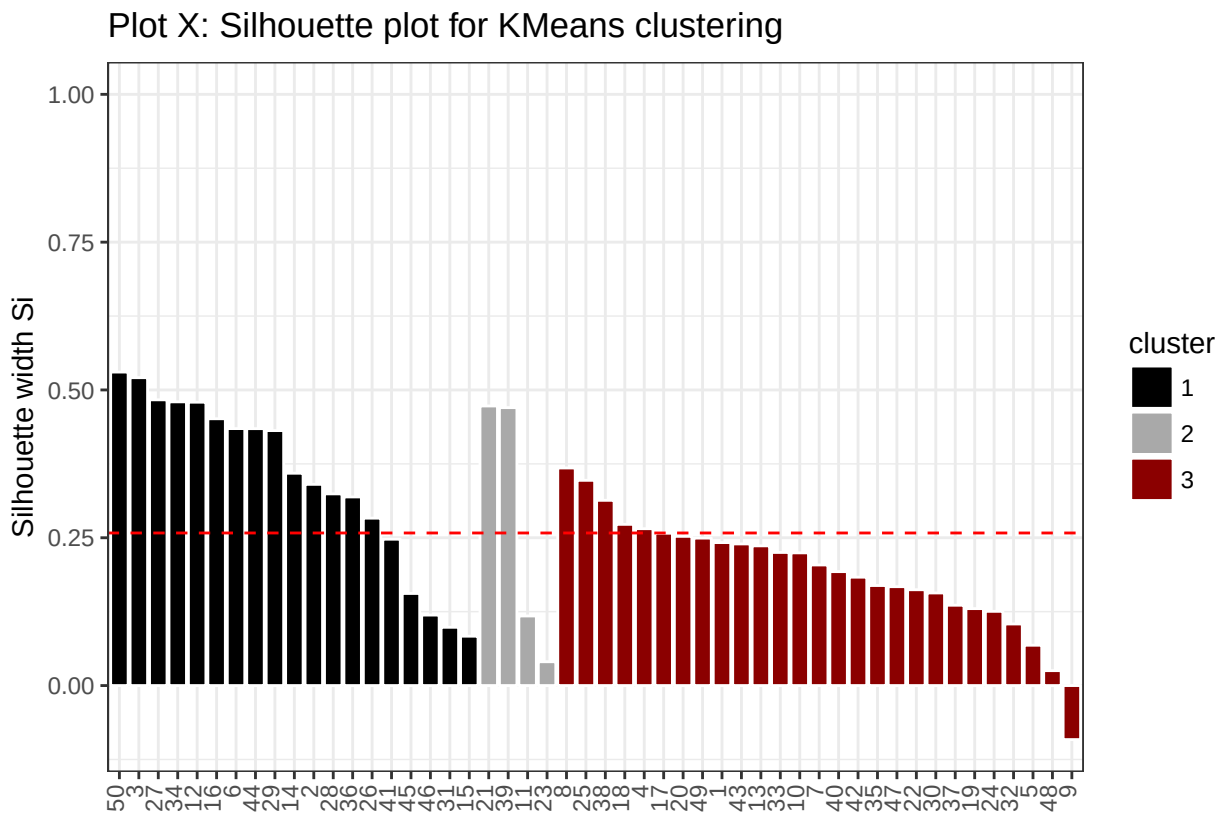
```
print(names(which(sil_pam < 0)))
```

```
## [1] "Tennessee" "Florida"   "Iowa"
```

K-Means silhouette

```
fviz_silhouette(silhouette(sel_kmeans$cluster, dist(scale(df_2))),
                main="Plot X: Silhouette plot for KMeans clustering") +
  theme_bw()  +
  scale_fill_manual(values=c("black", "darkgrey", "darkred")) +
  scale_color_manual(values=c("white", "white", "white")) +
  theme(axis.text.x=element_text(angle=90, hjust=1, vjust=0.25))
```

```
##   cluster size ave.sil.width
## 1       1   19          0.35
## 2       2    4          0.28
## 3       3   27          0.19
```

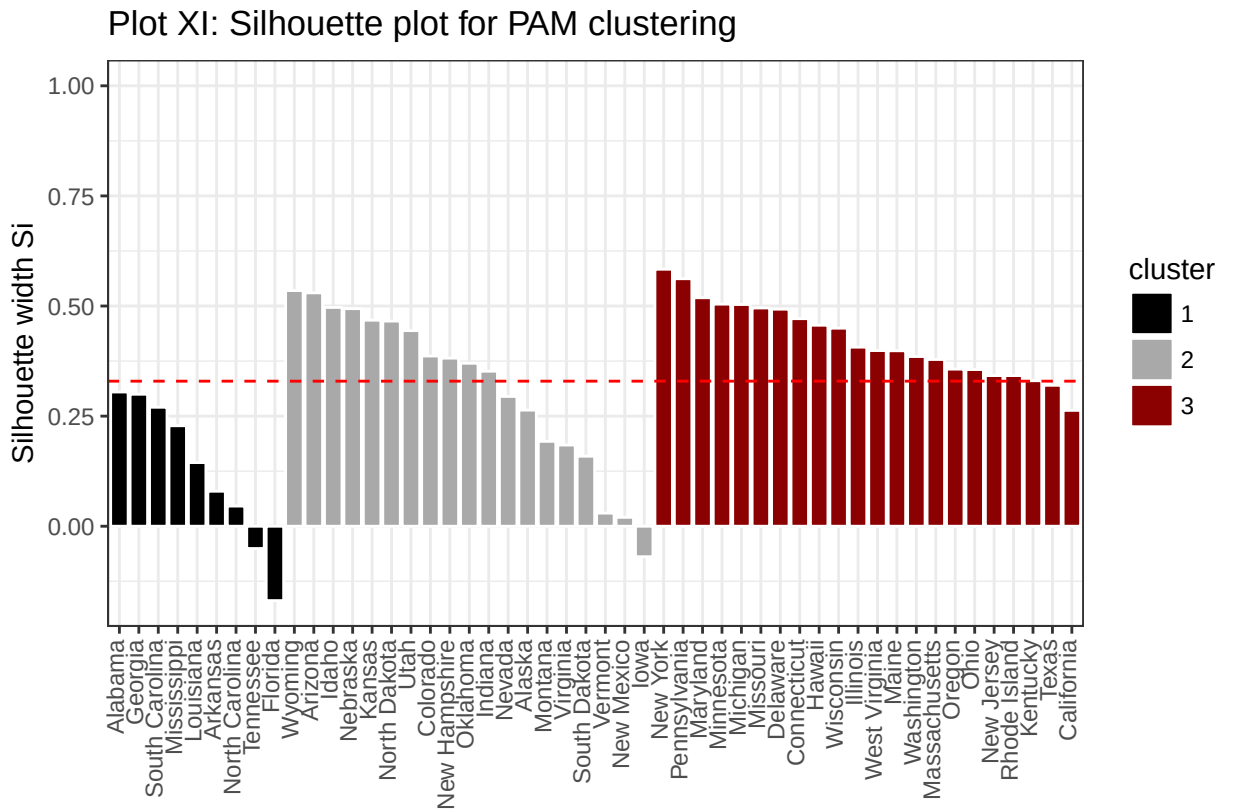


The k-means algorithm appears to have missclassified Florida wrongly into cluster 3.

PAM silhouette

```
fviz_silhouette(silhouette(sel_pam),
                main="Plot XI: Silhouette plot for PAM clustering") +
  theme_bw() +
  scale_fill_manual(values=c("black", "darkgrey", "darkred")) +
  scale_color_manual(values=c("white", "white", "white")) +
  theme(axis.text.x=element_text(angle=90, hjust=1, vjust=0.25))
```

```
##   cluster size ave.sil.width
## 1       1    9          0.13
## 2       2   19          0.32
## 3       3   22          0.42
```



The PAM algorithm misslassified ,Tennessee, Iowa as well as Florida into the wrong clusters.

Part 2c: Hierarchical clustering

Apply the following hierarchical clustering algorithms to the data:

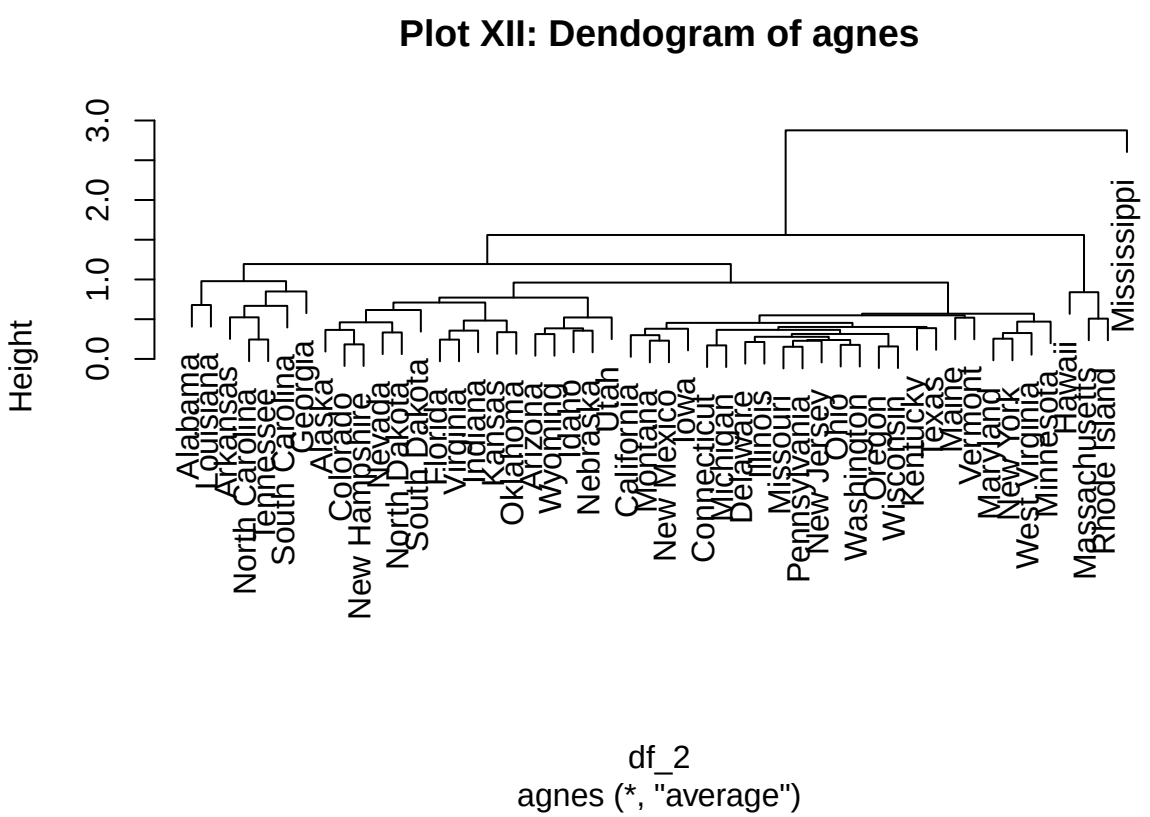
- **Agglomerative clustering** with Ward's method
- **Divisive clustering**

In each case, summarize the results using a dendogram. Determine the optimal number of clusters using Gap statistic, and add rectangles to the dendrograms sectioning off clusters. Do you find that states that predominantly vote for Republicans (e.g., Wyoming, Idaho, Alaska, Utah, Alabama) are closer together in the hierarchy? What can you say about states that usually lean towards Democrats (e.g. Maryland, New York, Vermont, California, Massachusetts)? Comment on the quality of clustering using Silhouette diagnostic plots.

Based on your choice of the optimal number of clusters in each case, visualize the clusters using a principal components plot, and compare them with the clustering results in Part 2b.

Agglomerative clustering

```
pltree(agnes(df_2), main="Plot XII: Dendogram of agnes")
```




```
gapstat_agnes <- clusGap(scale(df_2),
                         FUN=agnes.reformat,
                         K.max=10, B=500)
print(gapstat_agnes, method = "Tibs2001SEmax")
```

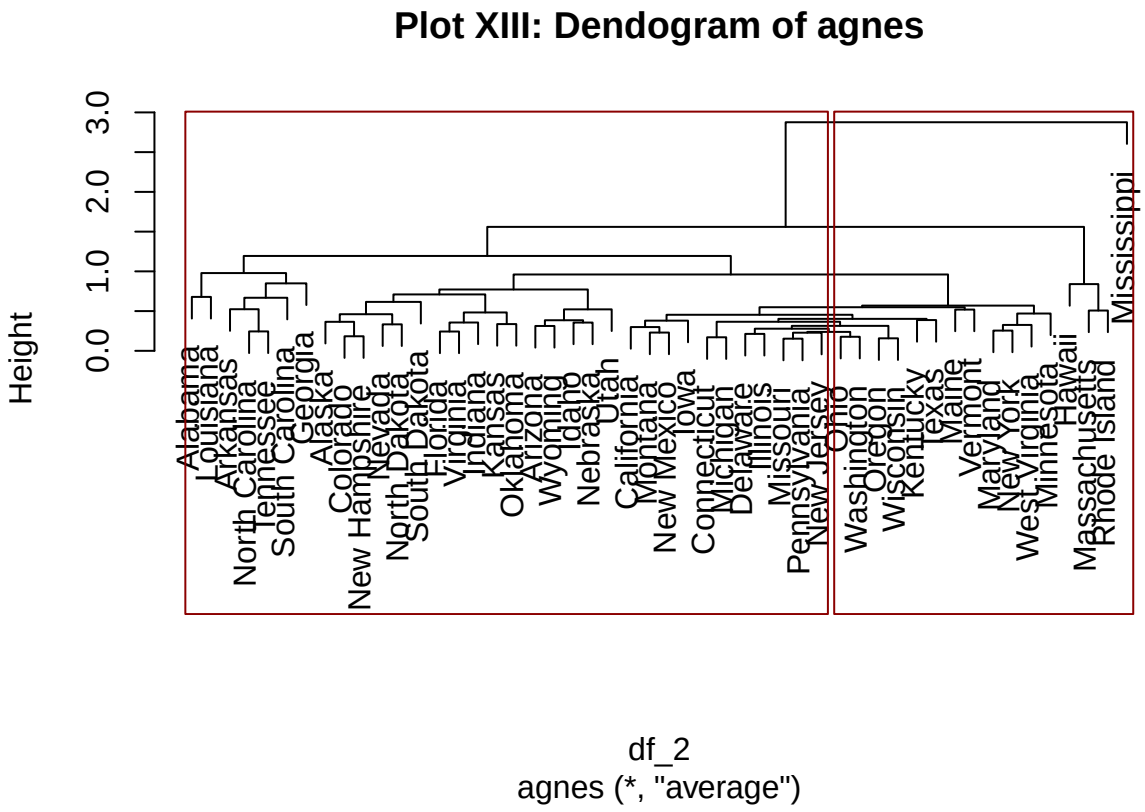
```
## Clustering Gap statistic ["clusGap"] from call:
## clusGap(x = scale(df_2), FUNcluster = agnes.reformat, K.max = 10,     B = 500)
## B=500 simulated reference sets, k = 1..10; spaceH0="scaledPCA"
##  --> Number of clusters (method 'Tibs2001SEmax', SE.factor=1): 1
##           logW   E.logW       gap     SE.sim
##  [1,] 3.620266 4.080172 0.4599059 0.03865442
##  [2,] 3.411249 3.877394 0.4661456 0.04475072
##  [3,] 3.230774 3.706042 0.4752678 0.04222003
##  [4,] 3.140818 3.581308 0.4404895 0.03858678
##  [5,] 3.023726 3.491944 0.4682179 0.03828579
##  [6,] 2.917142 3.414375 0.4972333 0.03745441
##  [7,] 2.828306 3.343336 0.5150303 0.03817637
##  [8,] 2.755878 3.276417 0.5205385 0.03828598
##  [9,] 2.693792 3.213740 0.5199480 0.03846854
## [10,] 2.625889 3.153631 0.5277422 0.03873774
```

According to the Tibshirani metric the optimal amount of clusters is again 1. However, in order to visualize a sensible amount of clusters on the dendogram, 2 is chosen.

Ward method

```
sel_agnes <- agnes(scale(df_2),
                   method="ward",
                   stand=T)

pltree(agnes(df_2), main="Plot XIII: Dendogram of agnes")
rect.hclust(sel_agnes, k=2, border="darkred")
```



The above plot shows, that the left box has mostly Republican states with some Democrate states such as Califorinia. While the right box right box selects mainly Democratic states with some outliers such as Missippi and Texas.

Calculations

```
out_agnes <- as.integer((agnes.reformat(scale(df_2), 2))[[1]])
sil_agnes <- silhouette(out_agnes, dist(scale(df_2)))[, 3]
index_agnes_neg_sil <- which(sil_agnes < 0)
print("Agnes Misclassified States:")
```

```
## [1] "Agnes Misclassified States:"
```

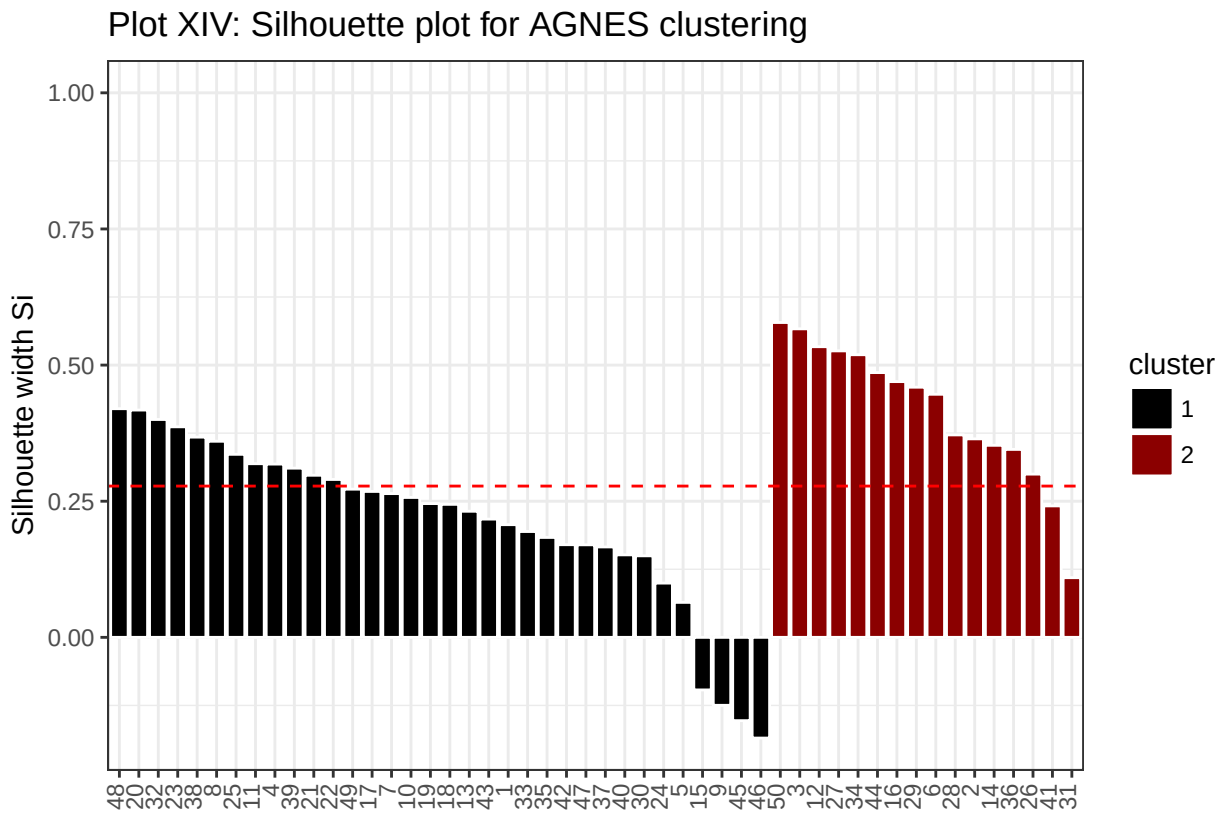
```
print(row.names(df_2)[index_agnes_neg_sil])
```

```
## [1] "Florida"  "Iowa"     "Vermont"  "Virginia"
```

Visualization

```
fviz_silhouette(silhouette(out_agnes, dist(scale(df_2))),
                main="Plot XIV: Silhouette plot for AGNES clustering") +
  theme_bw() +
  scale_fill_manual(values=c("black", "darkred")) +
  scale_color_manual(values=c("white", "white")) +
  theme(axis.text.x=element_text(angle=90, hjust=1, vjust=0.25))
```

```
##   cluster size ave.sil.width
## 1       1   34          0.21
## 2       2   16          0.42
```



The silhouette plot shows that 4 states (Florida, Iowa, Vermont, Virginia) are beeing misclassified into cluster 1.

Divisive Clustering

Gapstat for Diana

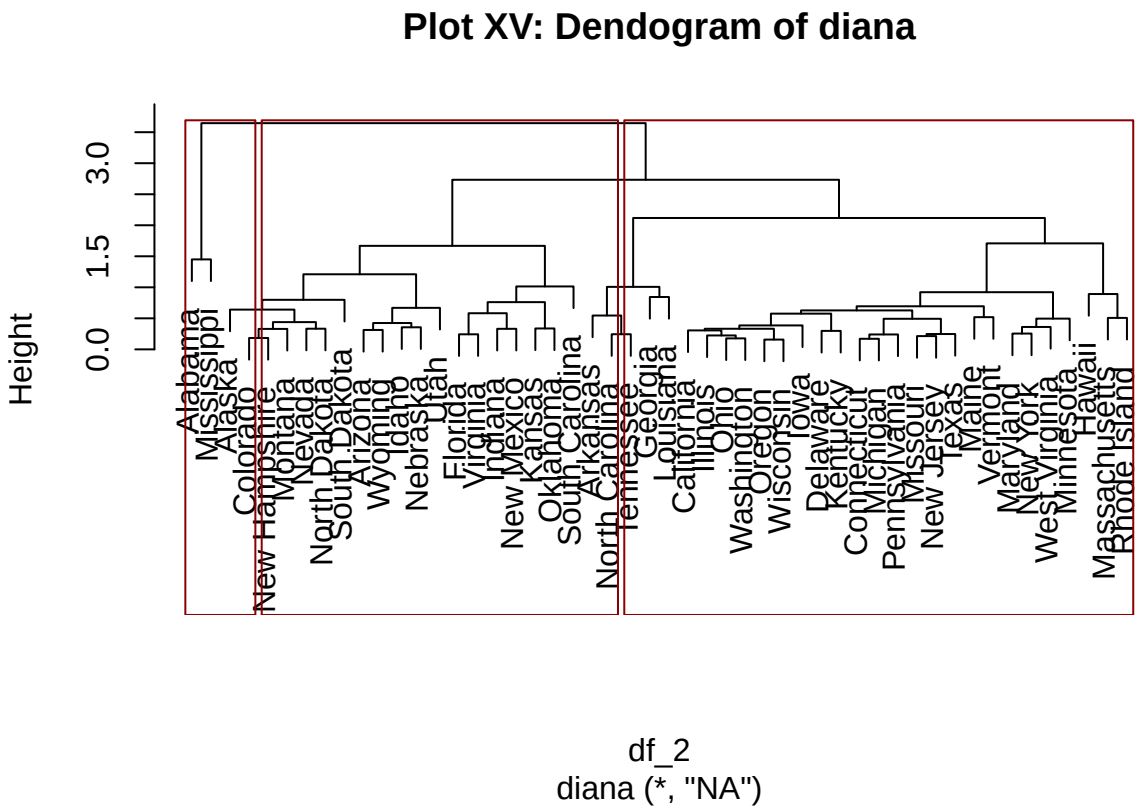
```
gapstat_diana <- clusGap(scale(df_2),
                         FUN=diana.reformat,
                         K.max=10,
                         B=500)
print(gapstat_diana, method = "Tibs2001SEmax")
```

```
## Clustering Gap statistic ["clusGap"] from call:
## clusGap(x = scale(df_2), FUNcluster = diana.reformat, K.max = 10,     B = 500)
## B=500 simulated reference sets, k = 1..10; spaceH0="scaledPCA"
##  --> Number of clusters (method 'Tibs2001SEmax', SE.factor=1): 1
##           logW   E.logW       gap     SE.sim
##  [1,] 3.620266 4.085415 0.4651484 0.03958772
##  [2,] 3.491409 3.877316 0.3859070 0.04251078
##  [3,] 3.254553 3.725549 0.4709955 0.04514482
##  [4,] 3.150653 3.575941 0.4252880 0.04128617
##  [5,] 3.088300 3.502241 0.4139418 0.03972635
##  [6,] 2.971592 3.434045 0.4624533 0.04079782
##  [7,] 2.844349 3.370147 0.5257978 0.04062140
##  [8,] 2.779890 3.308148 0.5282583 0.04143505
##  [9,] 2.717766 3.249910 0.5321435 0.04273237
## [10,] 2.665843 3.192913 0.5270695 0.04338928
```

The optimal number of clusters is again 1. However, I select 3 which is the next number satisfying the Tibshirani metric.

Dendogram for Diana

```
sel_diana <- diana(scale(df_2))
pltree(diana(df_2), main="Plot XV: Dendogram of diana")
rect.hclust(sel_diana, k=3, border="darkred")
```



The above plot show, that left and middle boxes lean towards Republican states. The box on the right leans

towards Democratic states with Georgia, Kentucky, Louisiana and Texas beeing the expeptions.

Calculations

```
out_diana <- as.integer((diana.reformat(scale(df_2), 2))[[1]])
sil_diana <- silhouette(out_diana, dist(scale(df_2)))[, 3]
print("Diana misclassified states:")
```

```
## [1] "Diana misclassified states:"
```

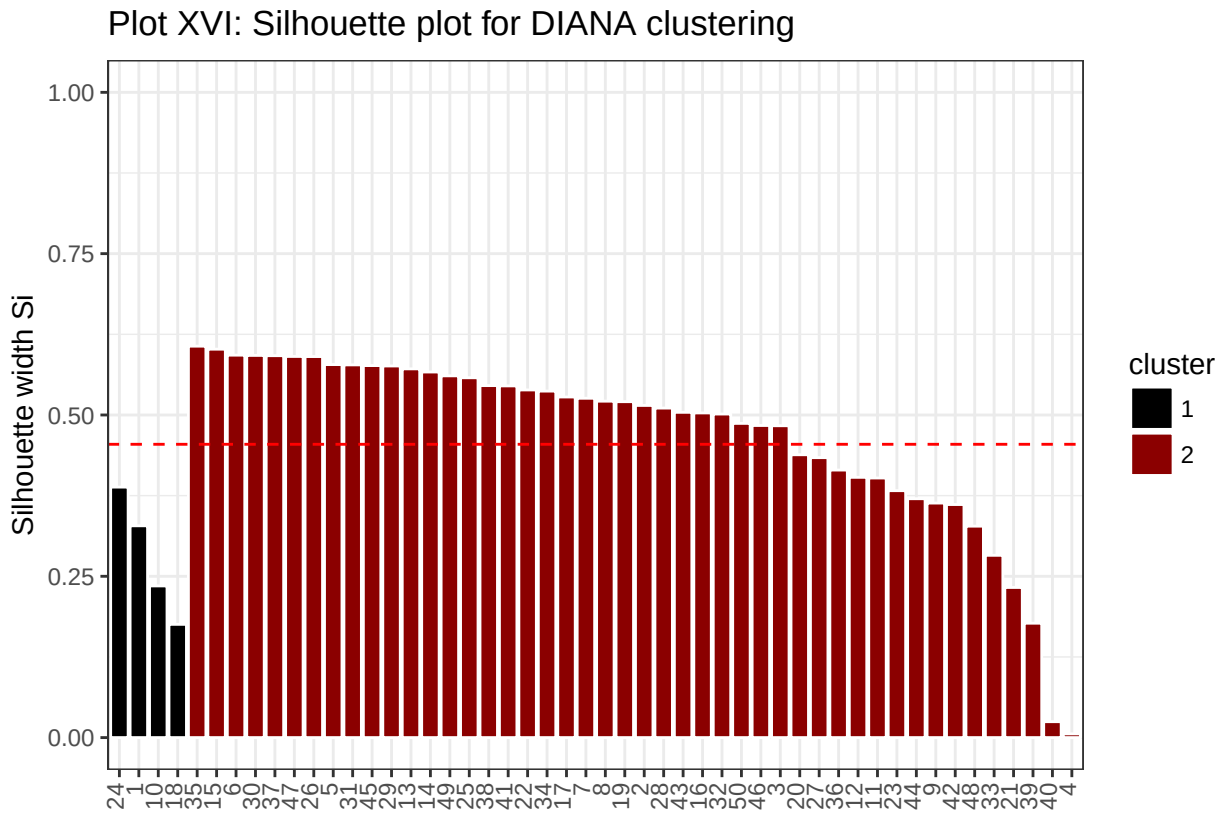
```
print(names(which(sil_diana < 0)))
```

```
## NULL
```

Silhouette for Diana

```
fviz_silhouette(silhouette(out_diana, dist(scale(df_2))),
                main="Plot XVI: Silhouette plot for DIANA clustering") +
  theme_bw() +
  scale_fill_manual(values=c("black", "darkred")) +
  scale_color_manual(values=c("white", "white")) +
  theme(axis.text.x=element_text(angle=90, hjust=1, vjust=0.25))
```

```
##   cluster size ave.sil.width
## 1       1    4          0.28
## 2       2   46          0.47
```



The above plot shows, that the state clusters appears to have no missclassified states.

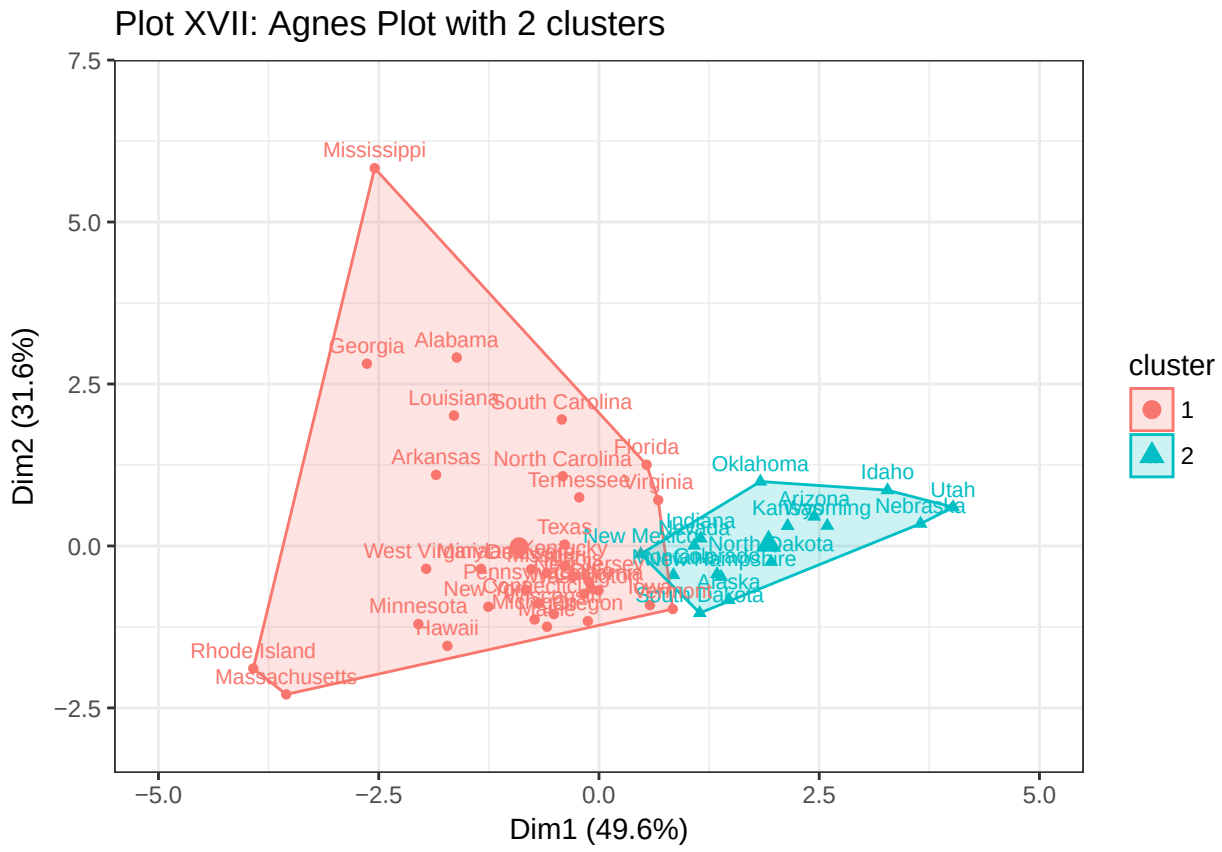
Principal Components Plots

Calculations

```
grp_agnes <- cutree(sel_agnes, k=2)
grp_diana <- cutree(sel_diana, k=3)
```

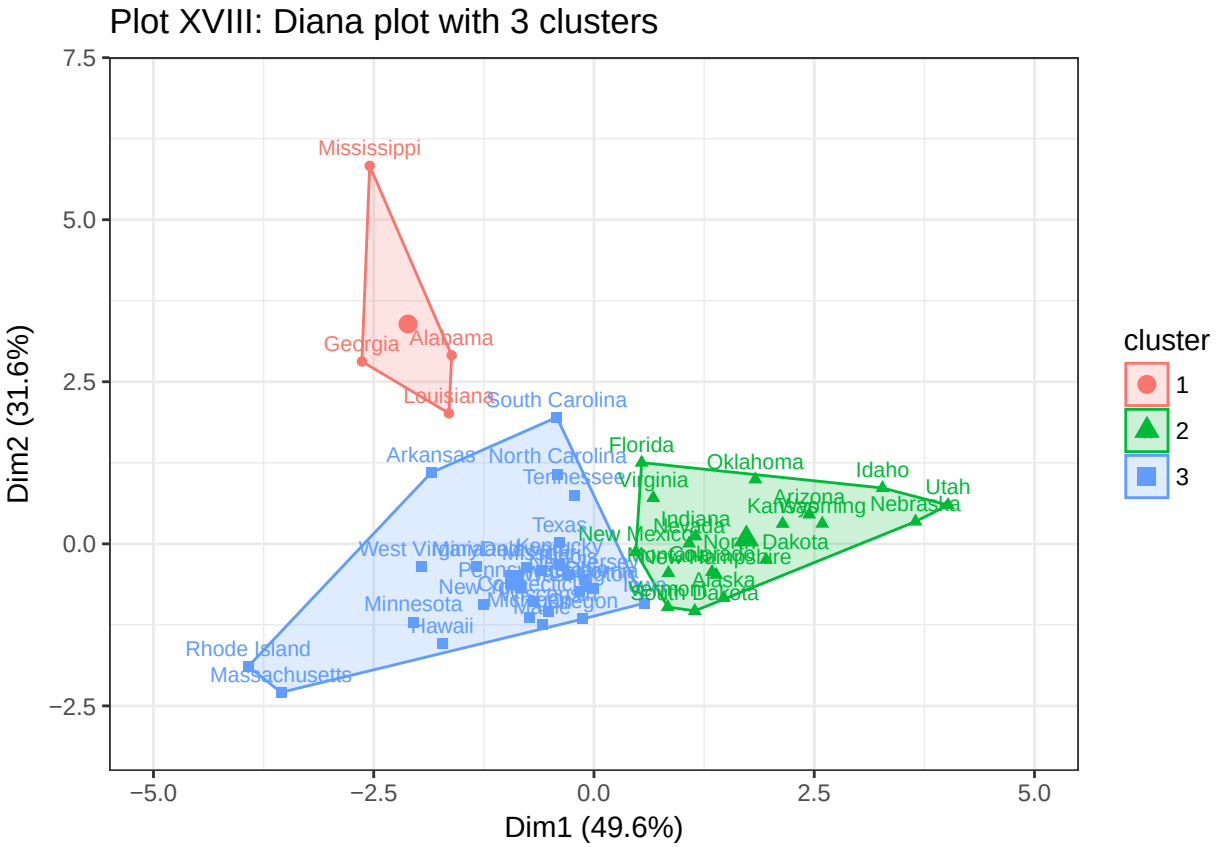
Visualize Agnes

```
fviz_cluster(list(data=scale(df_2), cluster=grp_agnes),
             main="Plot XVII: Agnes Plot with 2 clusters",
             labelsize=8) +
  theme_bw() +
  xlim(-5, 5) +
  ylim(-3, 7)
```



Visualize Diana

```
fviz_cluster(list(data=scale(df_2), cluster=grp_diana),
             main="Plot XVIII: Diana plot with 3 clusters",
             labelsize=8) +
  theme_bw() +
  xlim(-5, 5) +
  ylim(-3, 7)
```



The agnes plot with only two clusters is clearly different from the diana plot with 3 clusters. The plots appear to have some similarity with the already seen plots in the exercise 2b.

Part 2d: Soft clustering

We now explore if soft clustering techniques can produce intuitive grouping. Apply the following methods to the data:

- **Fuzzy clustering**
- **Gaussian mixture model**

For the fuzzy clustering, use the Gap statistic to choose the optimal number of clusters. For the Gaussian mixture model, use the internal tuning feature in `Mclust` to choose the optimal number of mixture components.

Summarize both sets of results using both a principal components plot, and a correlation plot of the cluster membership probabilities. Compare the results of the clusterings. Comment on the membership probabilities of the states. Do any states have membership probabilities approximately equal between clusters? For the fuzzy clustering, generate a silhouette diagnostic plot, and comment on the quality of clustering.

Fuzzy Clustering

Gap statistics fanny

```
gapstat_fanny <- clusGap(scale(df_2),
                         FUN=fanny,
```

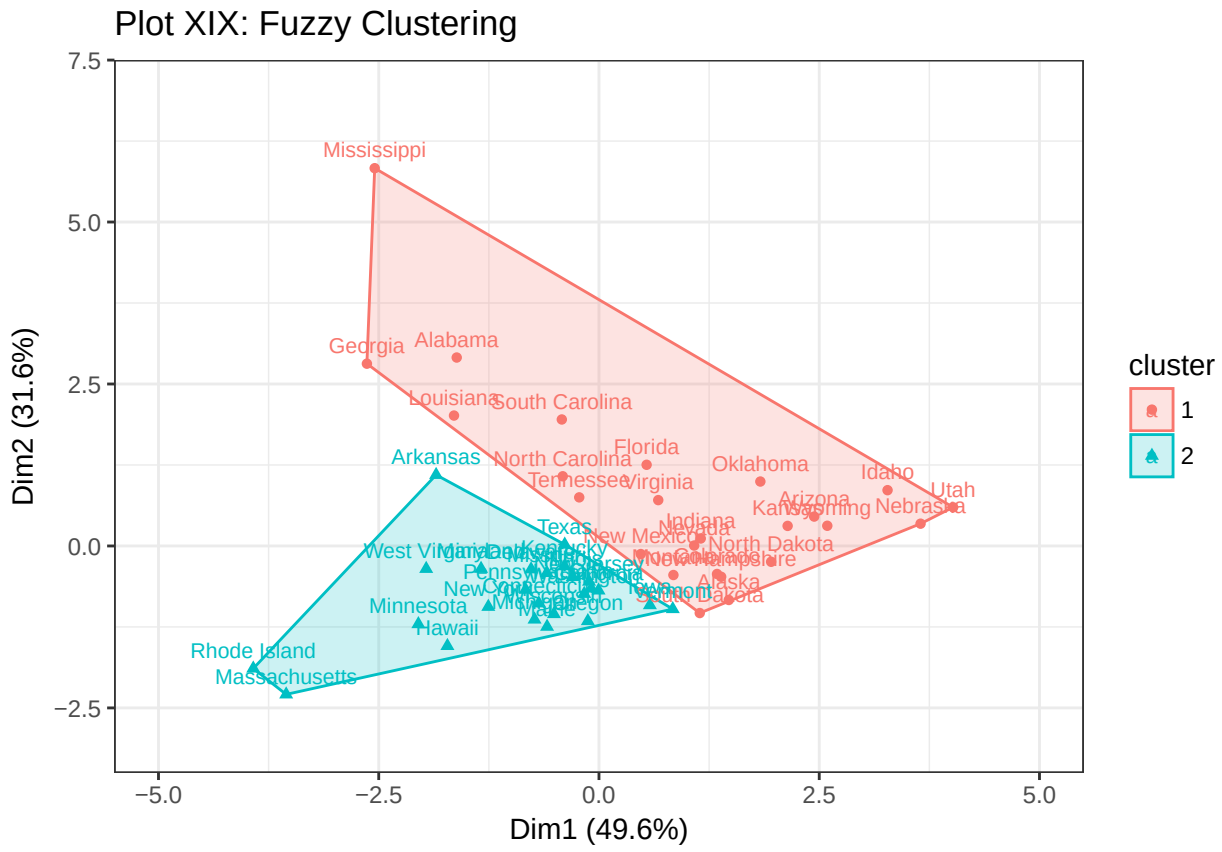
```
                        K.max=10,
                        B=500)
print(gapstat_fanny, method = "Tibs2001SEmax")
```

```
## Clustering Gap statistic ["clusGap"] from call:
## clusGap(x = scale(df_2), FUNcluster = fanny, K.max = 10, B = 500)
## B=500 simulated reference sets, k = 1..10; spaceH0="scaledPCA"
##  --> Number of clusters (method 'Tibs2001SEmax', SE.factor=1): 1
##           logW   E.logW       gap     SE.sim
##  [1,] 3.620266 4.083048 0.4627819 0.03837153
##  [2,] 3.420665 3.855493 0.4348279 0.03694546
##  [3,] 3.420665 3.824039 0.4033739 0.06532459
##  [4,] 3.420665 3.815313 0.3946472 0.07355559
##  [5,] 3.420665 3.809276 0.3886103 0.07956127
##  [6,] 3.216199 3.803679 0.5874793 0.08397052
##  [7,] 3.420665 3.795196 0.3745305 0.09131674
##  [8,] 3.323886 3.793606 0.4697205 0.09031415
##  [9,] 3.176778 3.784954 0.6081755 0.09770248
## [10,] 3.176560 3.777816 0.6012564 0.10015556
```

The optimal amount of clusters appears to be 1.

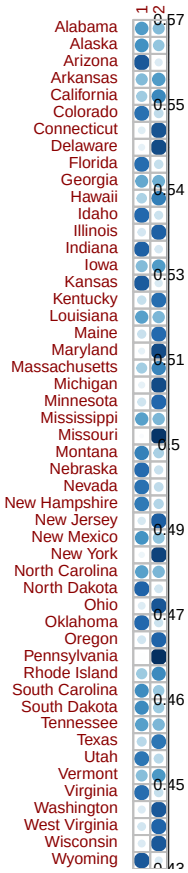
Visualization

```
sel_fanny <- fanny(df_2, 2)
fviz_cluster(sel_fanny,
             data=scale(df_2),
             main="Plot XIX: Fuzzy Clustering",
             labelsize=8) +
  theme_bw() +
  xlim(-5, 5) +
  ylim(-3, 7)
```



Correlation plot for fanny

```
corrplot(sel_fanny$membership,
         is.corr=FALSE,
         tl.cex=0.5, tl.col="darkred", cl.cex=0.5)
```



It appears that most states have a high probabilty of beeing in one of the groups. However, states like South and North Carolina as well as Tennesse are more undecided.

Fanny silhouette plot

```
# Calculations
sil_fanny <- silhouette(sel_fanny)[, 3]
print("Fuzzy misclassified states:")
```

```
## [1] "Fuzzy misclassified states:"
```

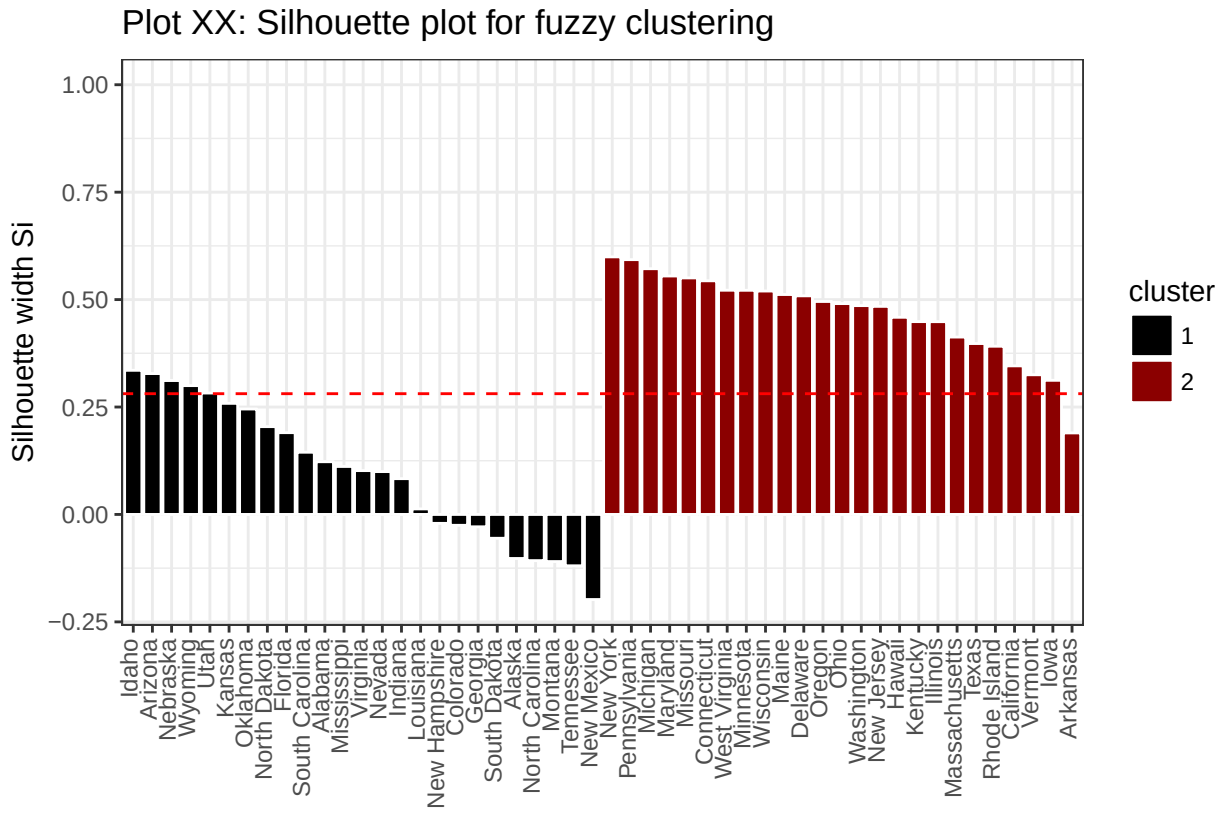
```
print(names(which(sil_fanny < 0)))
```

```
## [1] "New Hampshire"  "Colorado"       "Georgia"        "South Dakota"
## [5] "Alaska"         "North Carolina" "Montana"        "Tennessee"
## [9] "New Mexico"
```

Visualization

```
fviz_silhouette(silhouette(sel_fanny),
                main="Plot XX: Silhouette plot for fuzzy clustering") +
  theme_bw() +
  scale_fill_manual(values=c("black", "darkred")) +
  scale_color_manual(values=c("white", "white")) +
  theme(axis.text.x=element_text(angle=90, hjust=1, vjust=0.25))
```

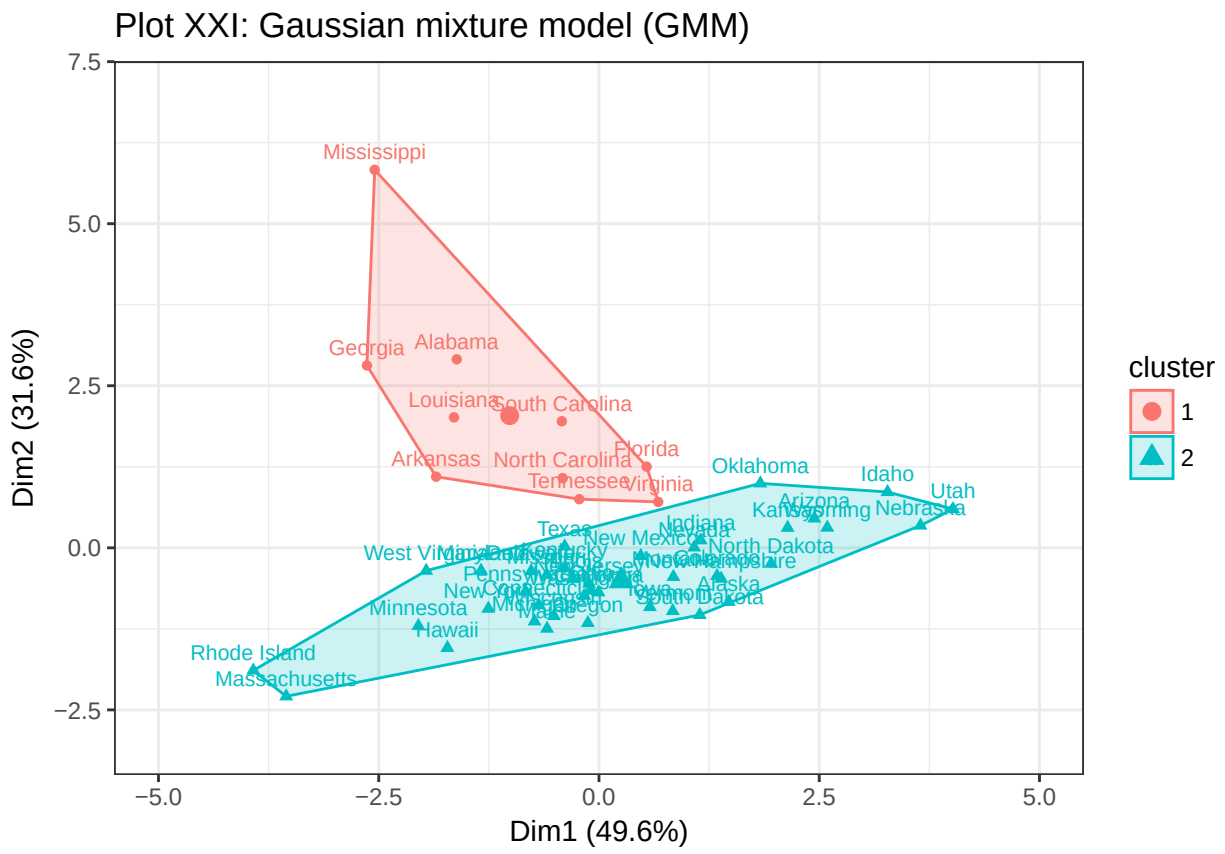
```
##   cluster size ave.sil.width
## 1       1   25          0.09
## 2       2   25          0.47
```



The above plot shows, that there are 9 missclassified states. Those are: New Hampshire, Colorado, Georgia, South Dakota, Alaska, North Carolina, Montana, Tennessee, New Mexico. This indicates that the clustering has potential to be improved.

Gaussian Mixture Model

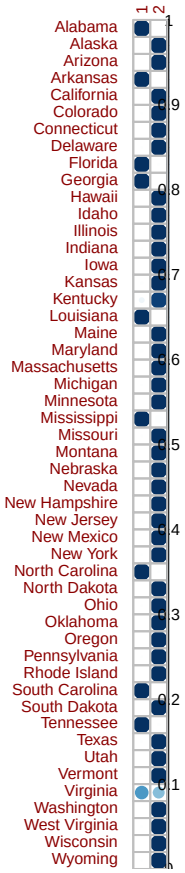
```
gmm <- Mclust(df_2)
fviz_cluster(gmm, data=scale(df_2),
             main="Plot XXI: Gaussian mixture model (GMM)",
             labelsize=8) +
  theme_bw() +
  xlim(-5, 5) +
  ylim(-3, 7)
```



The GMM has the same number of clusters as the fuzzy cluster algorithm. However, the cluster composition appears to be different between the two. The GMM has a large cluster of Democratic, Republican and split states and a smaller cluster with Republican states.

Correlation plot

```
corrplot(gmm$z,
         is.corr=FALSE,
         tl.cex=0.5,
         tl.col="darkred",
         cl.cex=0.5)
```



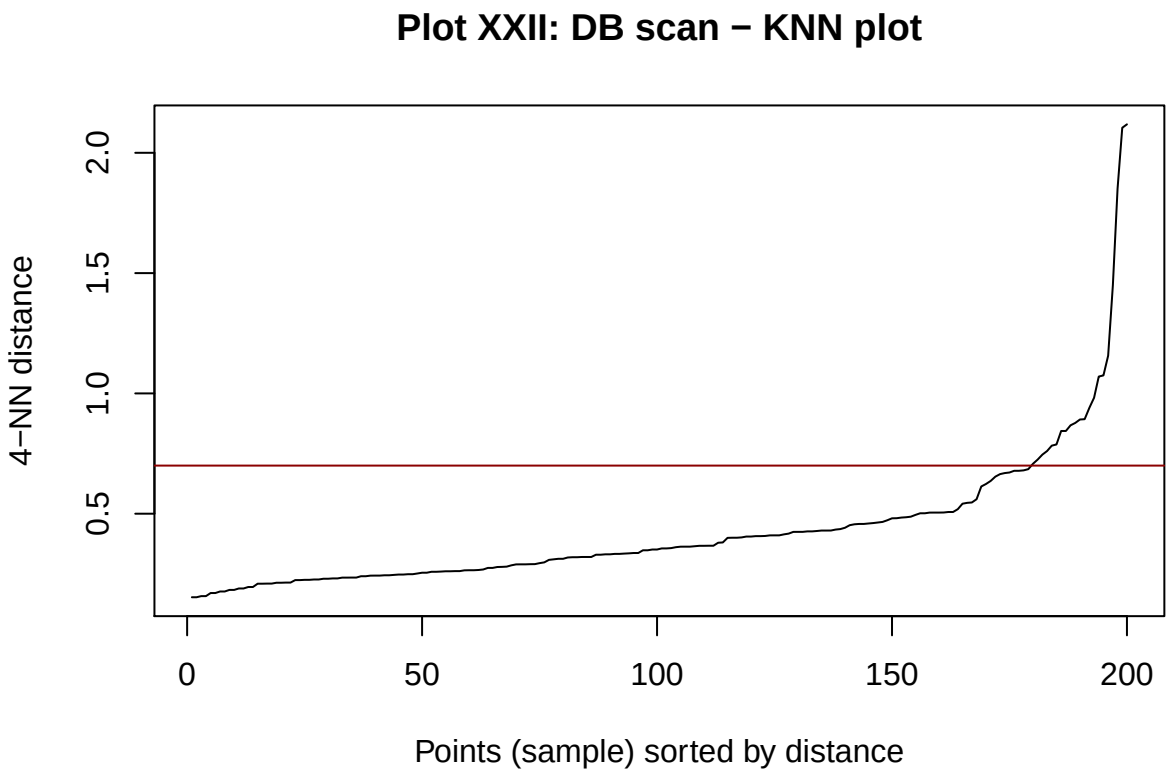
The GMM clustering is heavily different from the fuzzy clustering. Only the state of Virgina appears to have a similar probability of belonging to both clusters.

Part 2e: Density-based clustering

Apply DBSCAN to the data with `minPts = 5`. Create a knee plot. Summarize the results using a principal components plot, and comment on the clusters and outliers identified. How does the clustering produced by DBSCAN compare to the previous methods?

DB scan (KNN plot)

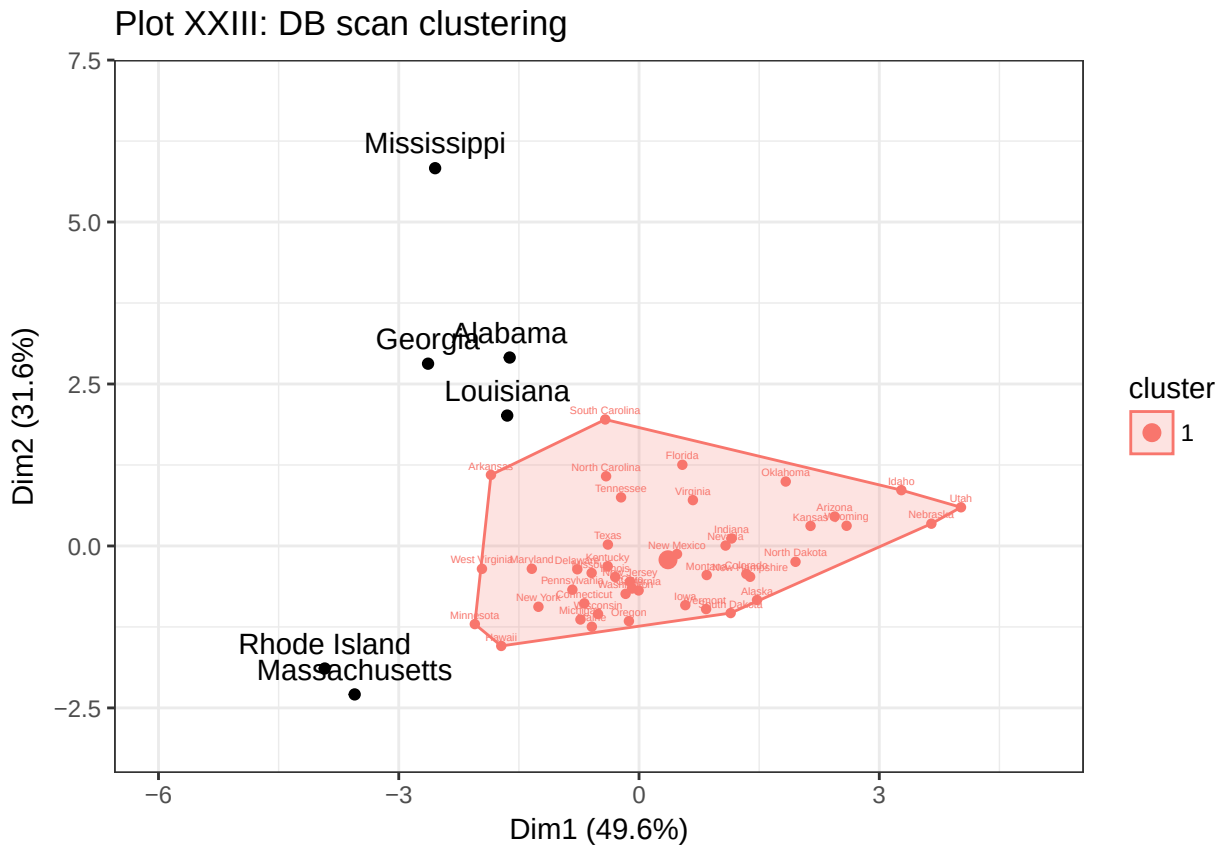
```
kNNdistplot(df_2)
title(main="Plot XXII: DB scan - KNN plot")
abline(0.7, 0, lty=1, lwd=1, col="darkred")
```



The *knee* appears to be at a eps of arround 0.7. The minimimum amount of points in a cluster is set at 5 (default).

DB scan cluster plot

```
chosen_dbscan <- dbscan(df_2, 0.7, minPts=5)
fviz_cluster(chosen_dbscan,
             data=scale(df_2),
             main="Plot XXIII: DB scan clustering",
             labelsize=4) +
  theme_bw() +
  xlim(-6, 5) +
  ylim(-3, 7)
```



The above plot shows, that the DBScan algorithm only selected 1 cluster as optimal. However, it also identified 6 outliers which are either very Democratic or very Republican: Georgia, Alabama, Louisiana, Rhode Island Masachusetts and Mississipii.