SAMHD exam

Nikolai Len

2025-02-23

Exercise 1: general questions

1.1 Describe some techniques allowing to select the number of clusters when clustering with Gaussian mixture models.

When using Gaussian Mixture Models (GMM) for clustering, the number of clusters (K) isn't directly estimated by the EM algorithm. Instead, we need to use model selection techniques are to determine the optimal K. Here are the main approaches:

1. Bayesian Information Criterion (BIC) This method balances model fit and complexity. It works by running the Expectation-Maximization (EM) algorithm for different values of K, computing the BIC score for each model, and selecting the K that yields the **highest BIC score**. This approach helps prevent overfitting by penalizing overly complex models, but it can be sensitive to noise, which may affect the quality of results.

2. Akaike Information Criterion (AIC) This method also penalizes model complexity but applies a different penalty than BIC. The process involves running EM for various values of K, computing the AIC score for each model, and selecting the one with the **lowest score**. AIC favors a larger number of clusters, which can sometimes lead to overfitting.

3. The Elbow Method
This is a heuristic approach based on log-likelihood. It involves running EM for a range of K values, plotting log-likelihood against K, and identifying the point where adding more clusters no longer significantly improves the likelihood. The optimal K is chosen at the "elbow point," where the curve starts to flatten.

4. Cross-Validation V-Fold Cross-Validation for GMM, implies that the dataset is divided into V equally sized folds. The model is trained on V−1 folds and evaluated on the remaining fold, with this process repeated V times, each time using a different fold as the validation set. The average log-likelihood across all folds is computed for each value of K, and the K that yields the highest average validation likelihood is selected. This approach provides a more stable estimate of model performance and helps prevent overfitting. However, it can be computationally expensive, especially for large datasets.

5.Stability-Based Methods This method assess the robustness of clustering results by measuring consistency across multiple EM runs. The process involves running GMM multiple times for each K, comparing cluster assignments across runs, and selecting the K where cluster stability is highest. This ensures that the clustering is reliable and not just a result of randomness, but it requires multiple EM runs which increases computational cost.

1.2 Describe the general setup and the goal of double cross-validation.

Double Cross-Validation (DCV) is a robust resampling method used to evaluate model performance while avoiding overfitting. DCV consists of two nested cross-validation loops:

1. Outer Loop (Model Evaluation): The dataset is divided into training and test sets multiple times. This loop evaluates the final model's generalization ability.
2. Inner Loop (Model Selection and Tuning): Within each training set from the outer loop, an inner cross-validation is performed to choose the best model parameters.

The primary goal of DCV is to provide a more reliable estimate of model performance while preventing overfitting that may arise from tuning model parameters on the same dataset used for evaluation.

Exercise 2: k-means clustering

2.1 how we can relate the quality of a clustering to variance?

Clustering quality is evaluated by how well data points are grouped together within clusters and how well-separated the clusters are. The notions of variance help assess clustering quality as follows:

1. Within-cluster variance (W): Measures how tightly the points in a cluster are grouped. Lower W means clusters are more compact, which indicates better clustering.

2.Between-cluster variance (B): Measures how far apart clusters are. Higher B indicates better separation between clusters.

3.Total variance (S): S=W+B

The goal of k-means is to minimize W and maximize B to achieve well-defined clusters.

2.2 Cluster dataset into 2 groups

Step 1: Define the dataset

```
data <- data.frame(
  var1 = c(0, 0, 1, 3, 3.5, 1, 3, 4),
  var2 = c(1, 2, 1, 1, 1, 5, 4, 5)
)
```

Step 2: Apply k-means clustering with k = 2

```
# Setting nstart = 10 runs k-means 10 times with different initial centroids to get a better solution
set.seed(42)  # Ensures reproducibility
kmeans_result <- kmeans(data, centers = 2, nstart = 10)

print(kmeans_result)
```

```
## K-means clustering with 2 clusters of sizes 5, 3
##
## Cluster means:
##       var1     var2
## 1 1.500000 1.200000
## 2 2.666667 4.666667
##
## Clustering vector:
## [1] 1 1 1 1 1 2 2 2
##
## Within cluster sum of squares by cluster:
## [1] 11.800000  5.333333
##  (between_SS / total_SS =  59.4 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
print(kmeans_result$cluster)
```

```
## [1] 1 1 1 1 1 2 2 2
```

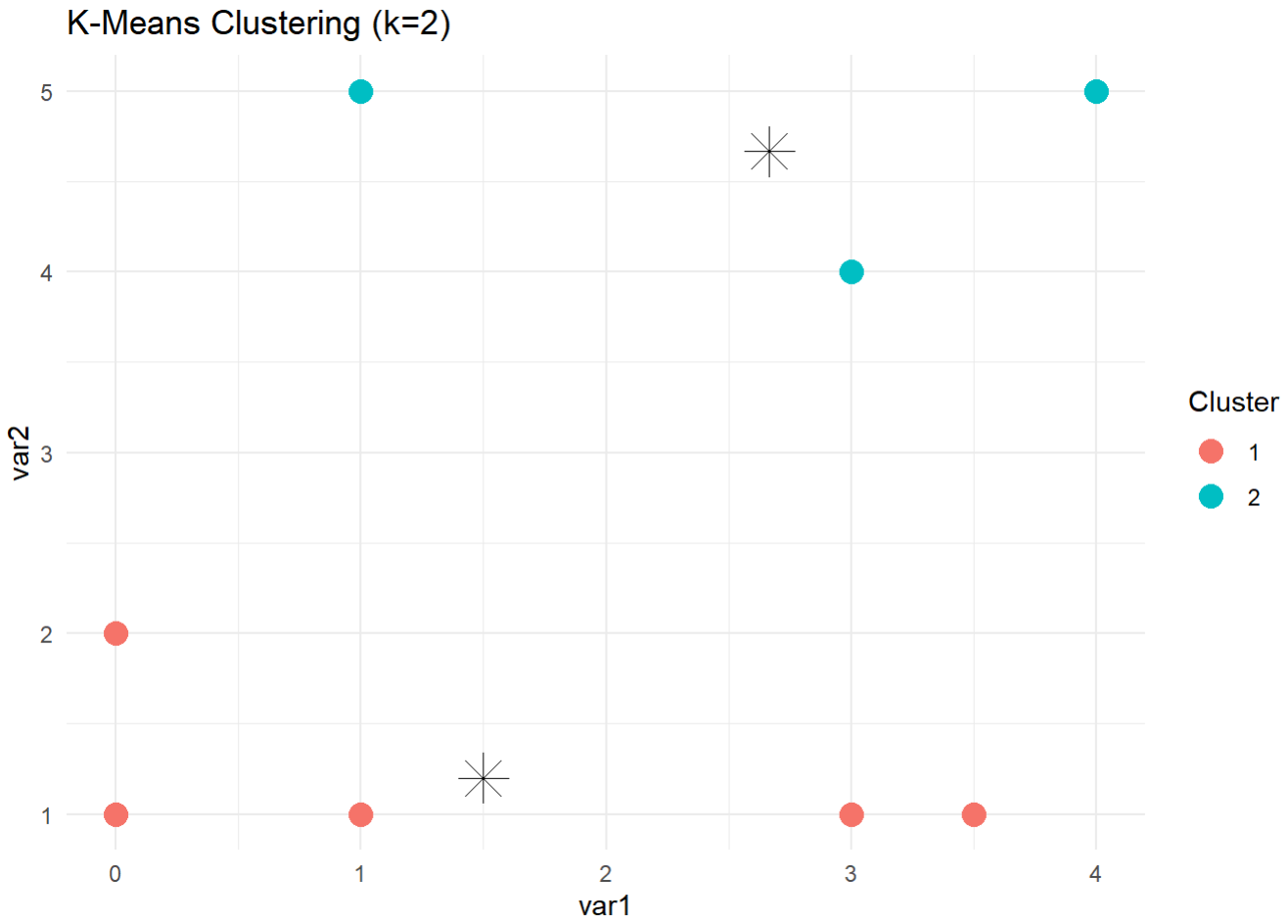
```
print(kmeans_result$centers)
```

```
##       var1     var2
## 1 1.500000 1.200000
## 2 2.666667 4.666667
```

Step 3: Visualizing the clusters

```
library(ggplot2)
# Convert cluster centers to a dataframe
centers_df <- as.data.frame(kmeans_result$centers)

# Visualizing the clusters
ggplot(data, aes(x = var1, y = var2, color = factor(kmeans_result$cluster))) +
  geom_point(size = 4) +  # Plot individual points
  geom_point(data = centers_df, aes(x = var1, y = var2),
             color = "black", size = 6, shape = 8) +  # Marking centroids
  labs(title = "K-Means Clustering (k=2)", color = "Cluster") +
  theme_minimal()
```



Exercise 3: the Vélib data

3.1 Loading the data

```
# Loading the dataset
load("velib.Rdata")
```

3.2 Pretreatment et descriptive analysis

Inspect the data

```
summary(velib)
```

```
##          Length Class      Mode
## data      181   data.frame list
## position    2   data.frame list
## dates     181   -none-     character
## bonus    1189   -none-     numeric
## names    1189   -none-     character
```

Extract the relevant data frame

```
X <- velib$data  # The main dataset
position <- velib$position  # GPS coordinates of stations
```

Check the structure of the main dataset

```
# Display number of features and observations
num_features <- ncol(X)
num_observations <- nrow(X)
cat("Number of features:", num_features, "\n")
```

```
## Number of features: 181
```

```
cat("Number of observations:", num_observations, "\n")
```

```
## Number of observations: 1189
```

Check for missing values in X

```
missing_values <- sum(is.na(X))
cat("Total missing values in X:", missing_values, "\n")
```

```
## Total missing values in X: 0
```

Check for missing values in position

```
#test for missing values
sum(is.na(position) == TRUE)
```

```
## [1] 0
```

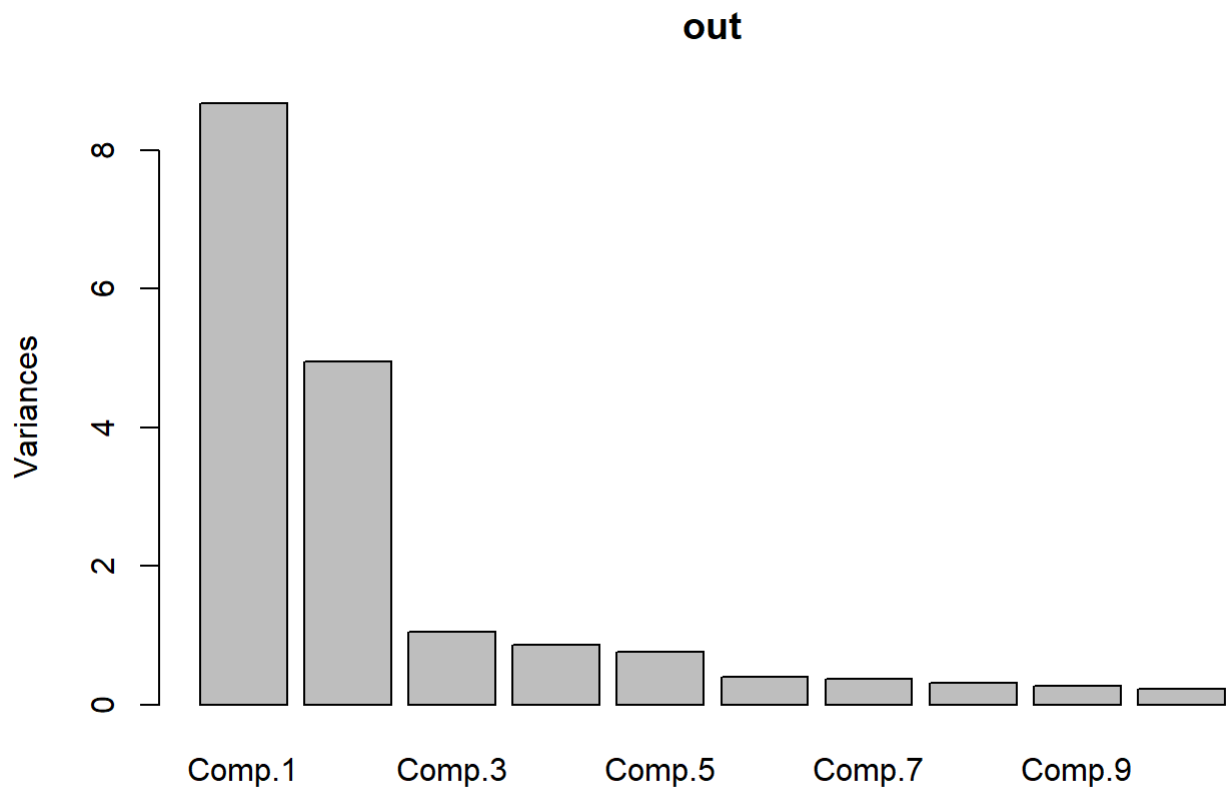
3.3 Data visualization

PCA

```
out = princomp(X)
```

Visually find the most interesting components

```
screeplot(out)
```



Define the cattell scree-test function

```
cattell <- function(x,thd=0.1){
  sc = abs(diff(x))
  p = length(x)
  d = p-1
  for (j in 1:(p-2)){
    if (prod(sc[(j+1):length(sc)] < thd * max(sc))){
      d = j
      break
    }
  }
  d
}
```

Find the most interesting components by calculation

```
d.cattell = cattell(out$sdev)
d.cattell
```

```
## [1] 5
```

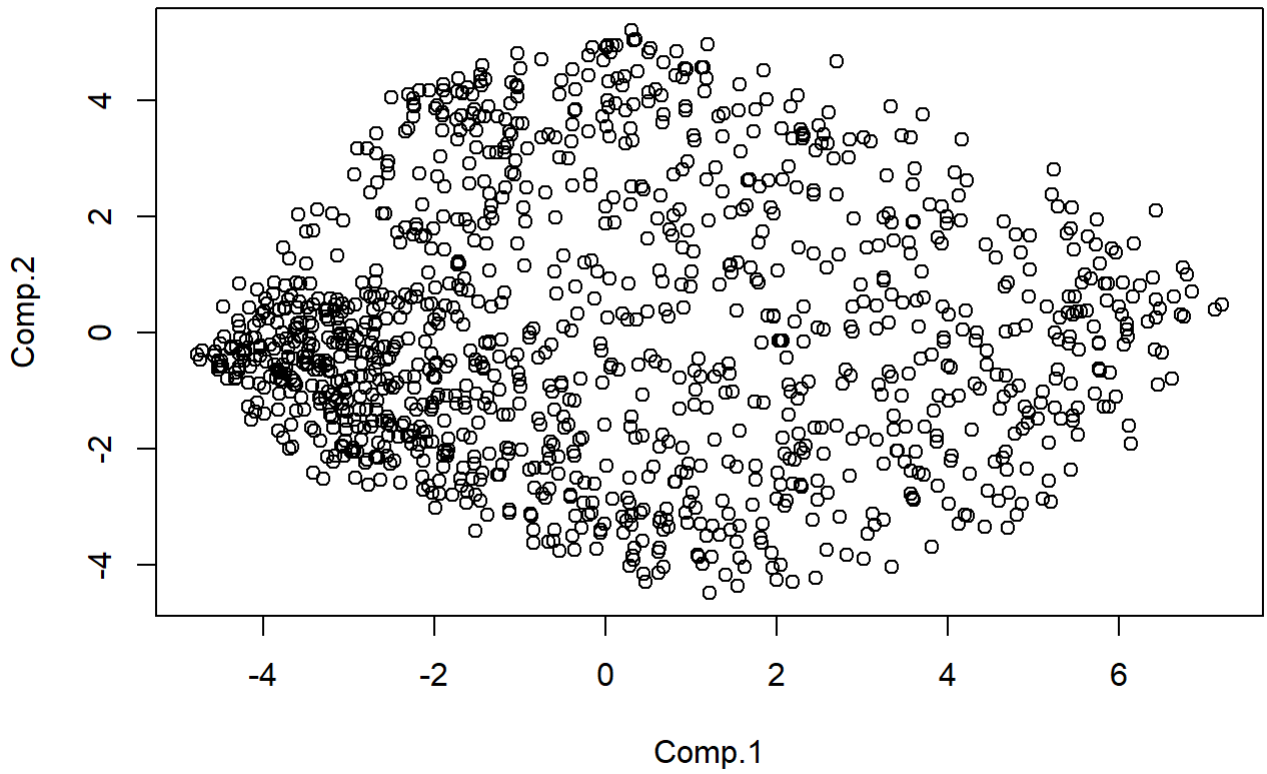
Select the top 5 components

```
d.star = 5
Y = predict(out)[,1:d.star]
Y[1:10,]
```

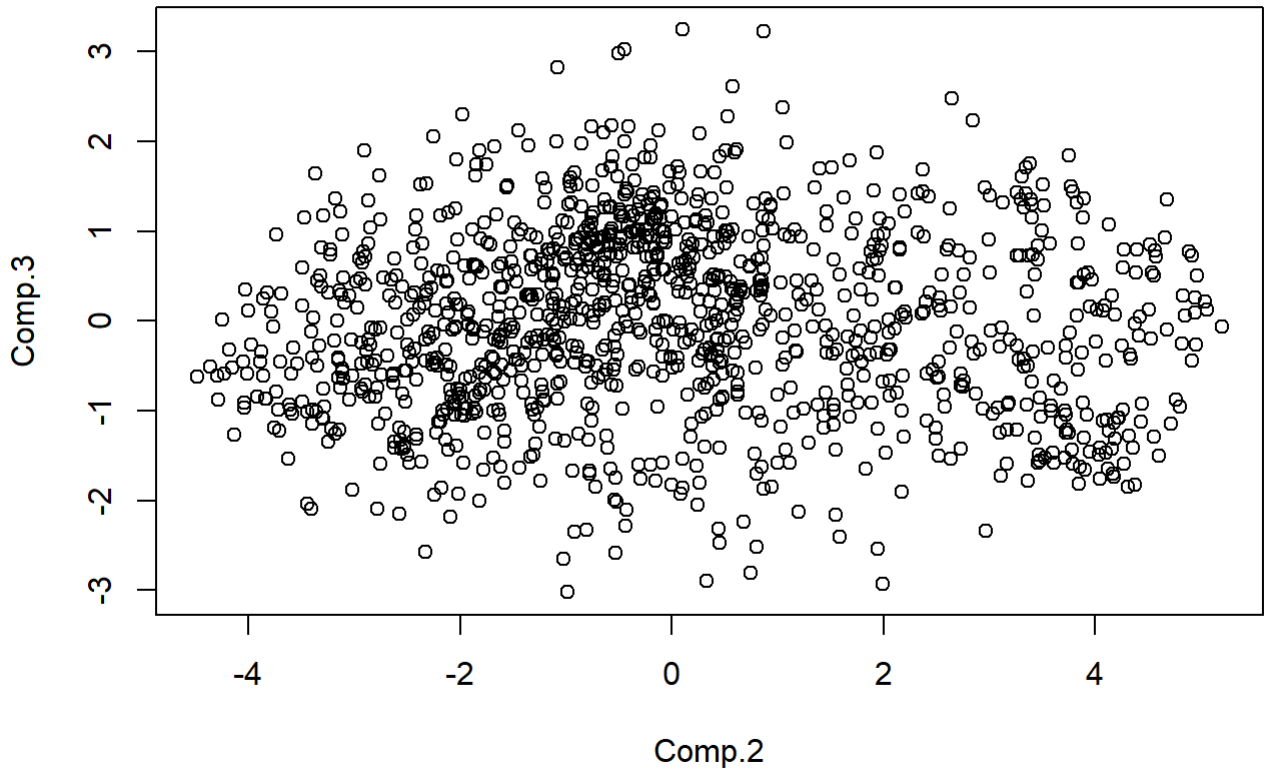
```
##           Comp.1     Comp.2     Comp.3     Comp.4     Comp.5
## 19117 -1.2040281 -2.2743248  0.3765529 -1.5113583 -0.7208404
## 17111  1.9966378 -4.2517544  0.0101666 -1.1582747 -0.6775315
## 6103   0.8860784  0.8326818  0.4468908 -1.0874062 -1.5082802
## 15042  0.6769466  3.7618830 -0.1271212  2.1575871  2.0098358
## 12003  2.1348525  2.8547899 -0.3616034 -0.8162485  1.2103656
## 13038 -0.3032946 -1.8535891  0.6121352 -0.9463538 -0.9531223
## 17041  0.6261200 -1.9621200  0.6162480 -0.1535873 -0.7289564
## 41203  0.4445372 -3.0455680  0.2804429  0.3607209 -0.4484349
## 43401  3.6341017  0.5580006  0.4430034  0.5438336  1.6503185
## 5015   0.9915129  0.7844612 -1.4838054 -0.1780015  3.4782944
```

Plots of the successive components

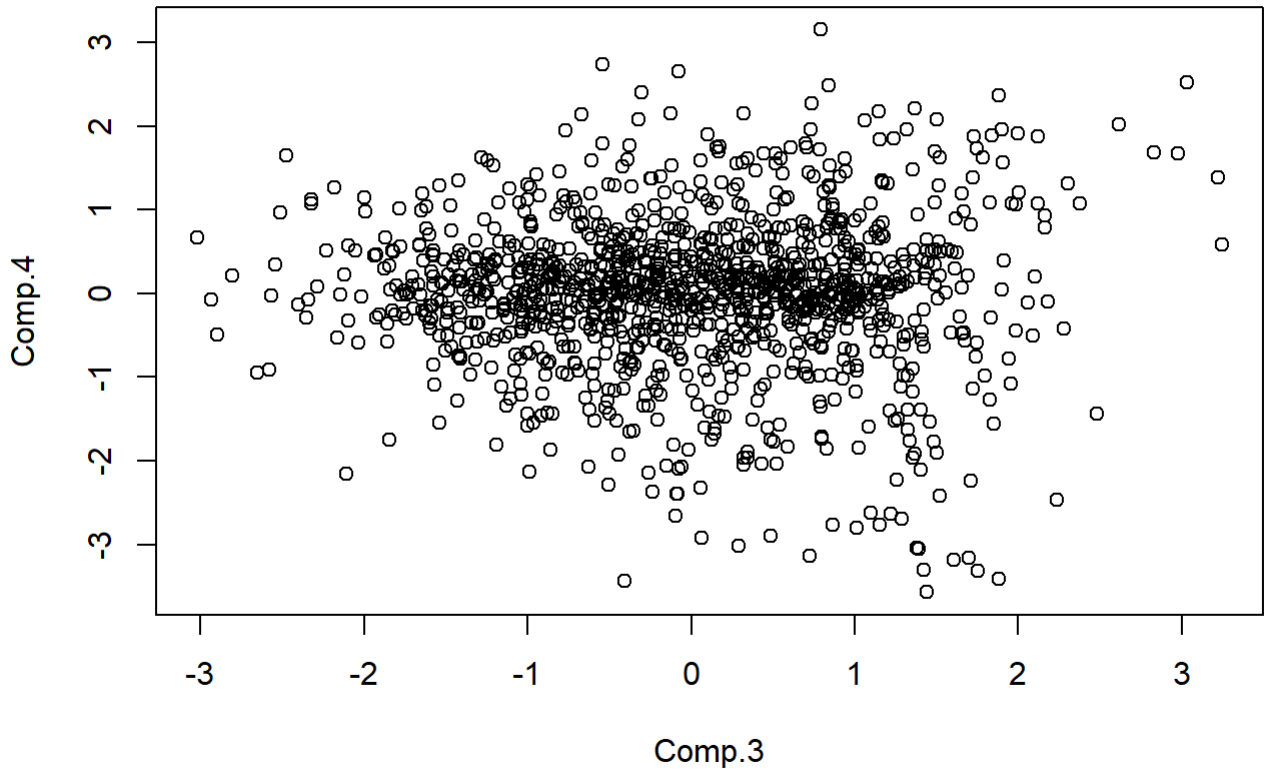
```
plot(Y[,1:2])
```



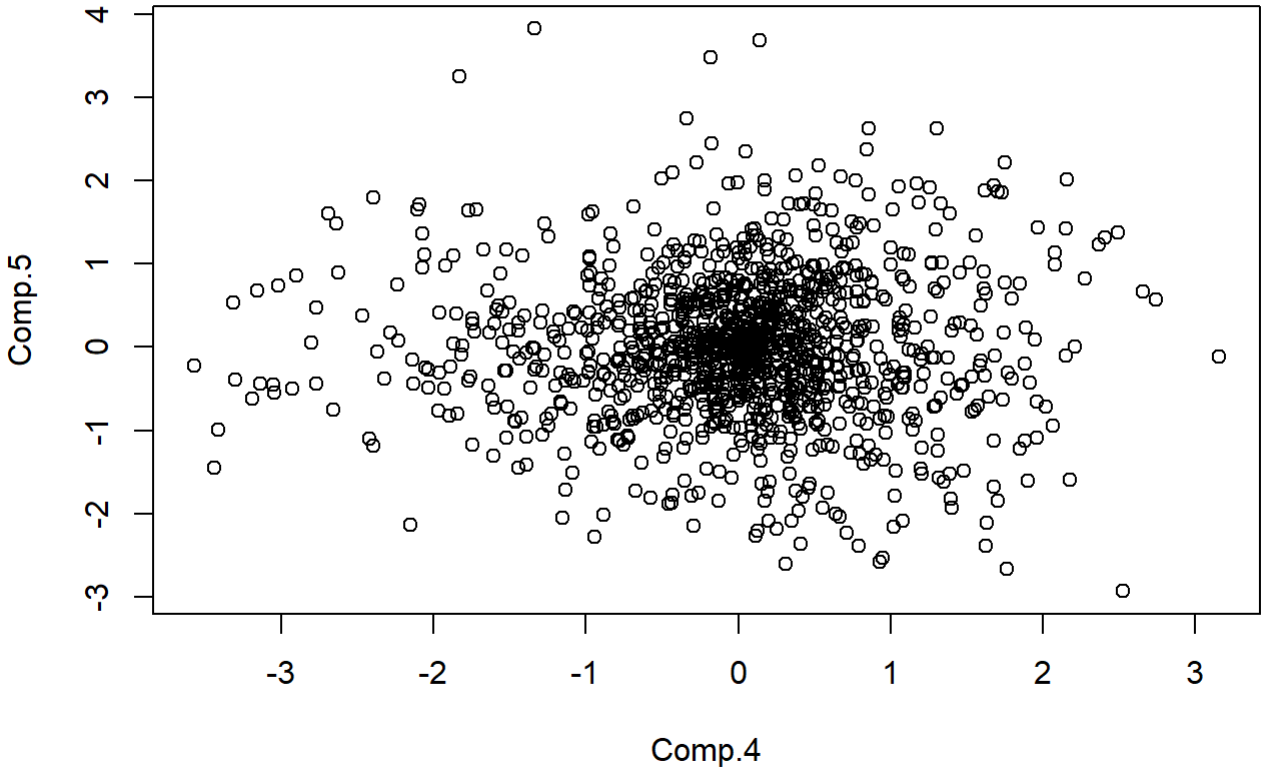

```
plot(Y[,2:3])
```




```
plot(Y[,3:4])
```



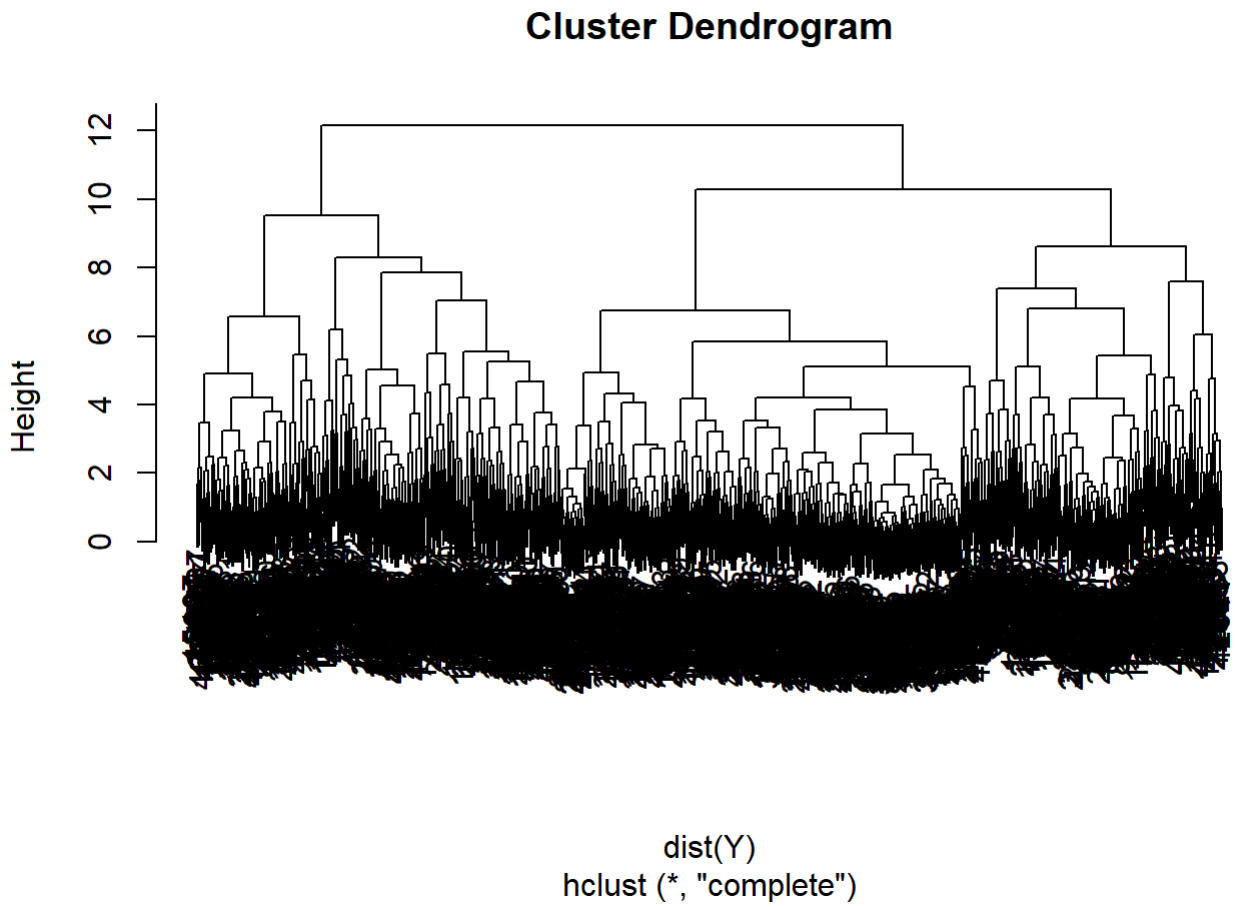
```
plot(Y[,4:5])
```



3.4 Clustering

3.4.1 Hierarchical clustering

```
# hierarchical clustering
out.hc = hclust(dist(Y), method = "complete")
plot(out.hc)
```



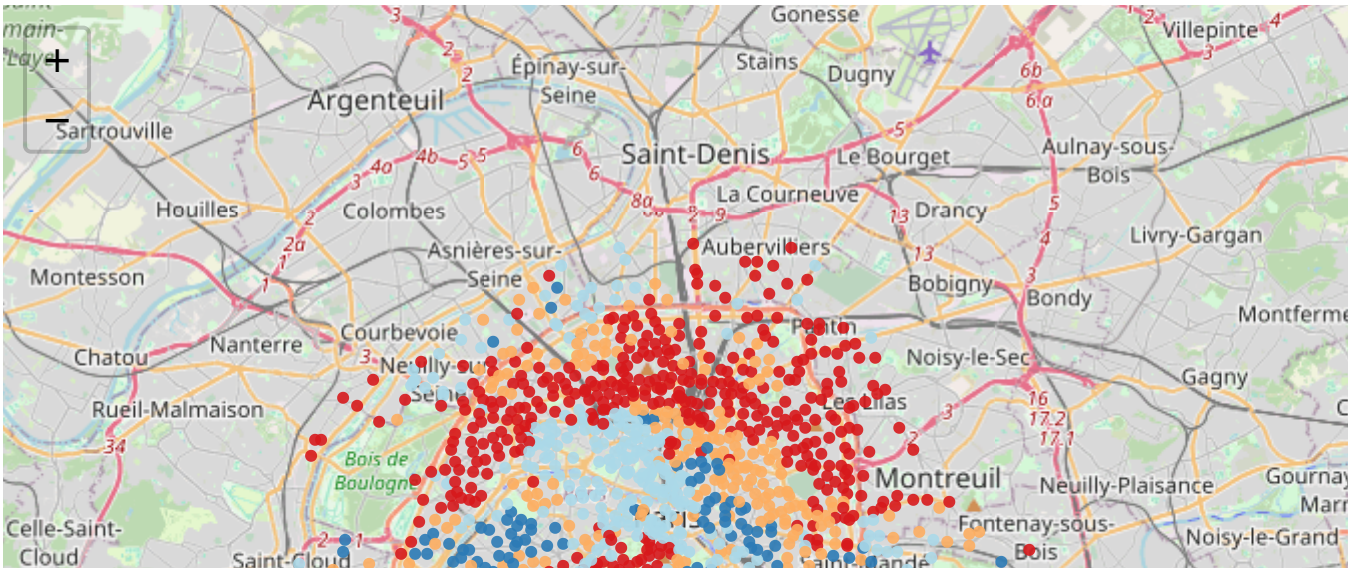
Select gap for k = 4

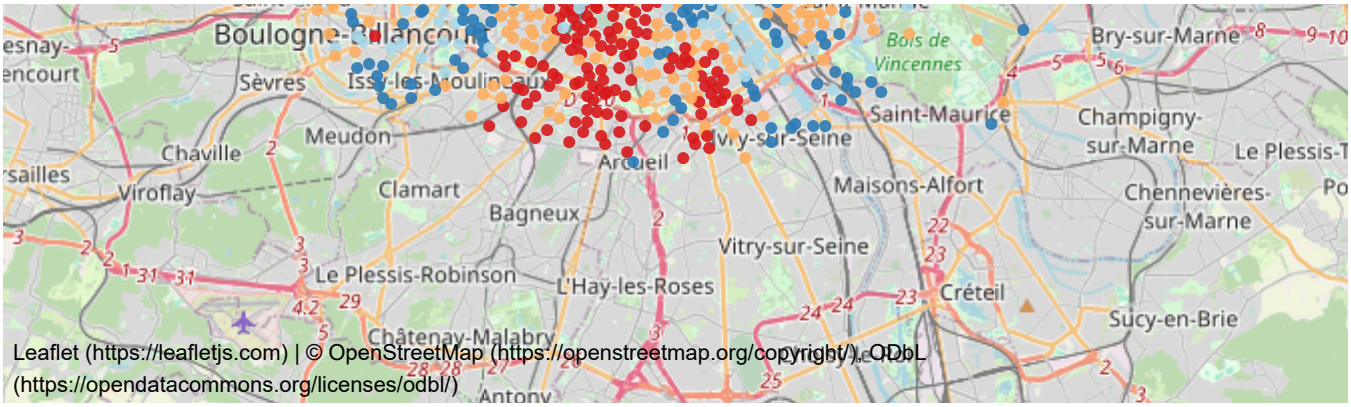
```
hc.clust = cutree(out.hc, k=4)
```

Plot the clusters

```
library("leaflet")

palette = colorFactor("RdYlBu", domain = NULL)
leaflet(position) %>% addTiles() %>%
addCircleMarkers(radius = 3,
color = palette(hc.clust),
stroke = FALSE, fillOpacity = 0.9)
```

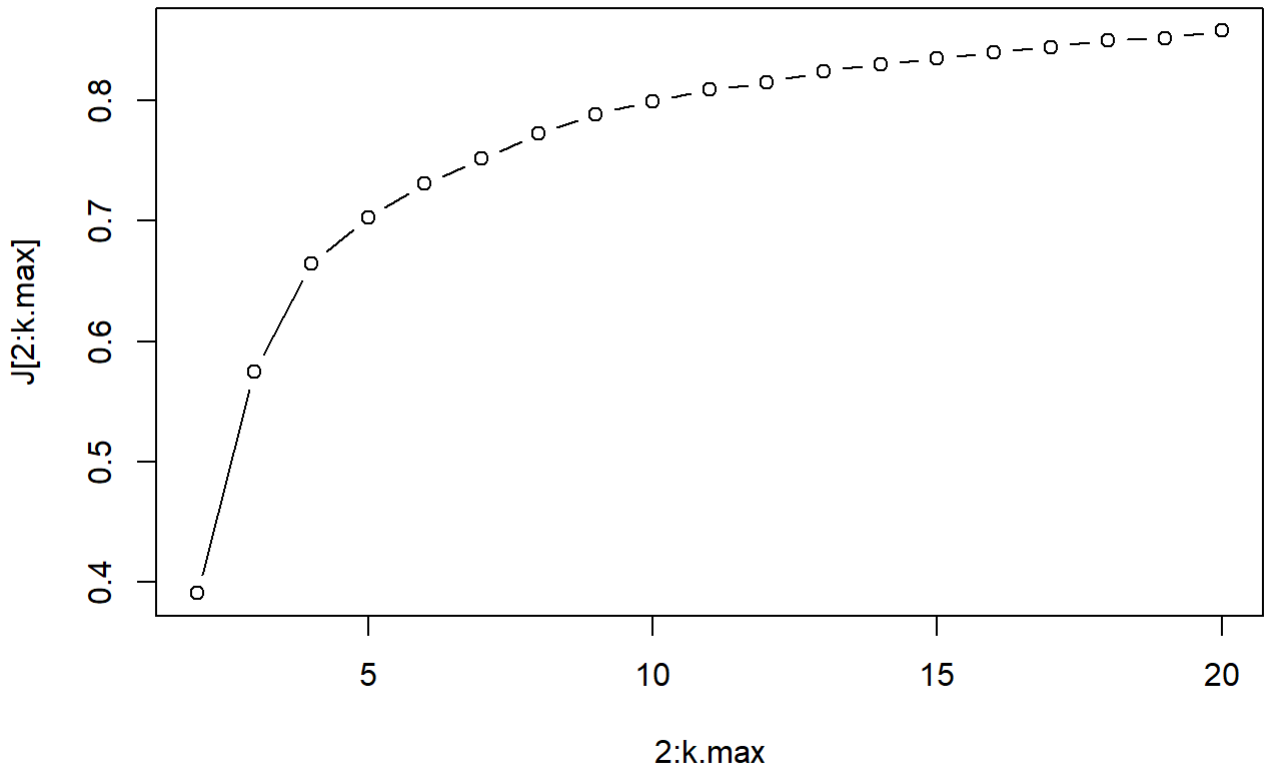




3.4.2 k-means

Let's find the best k

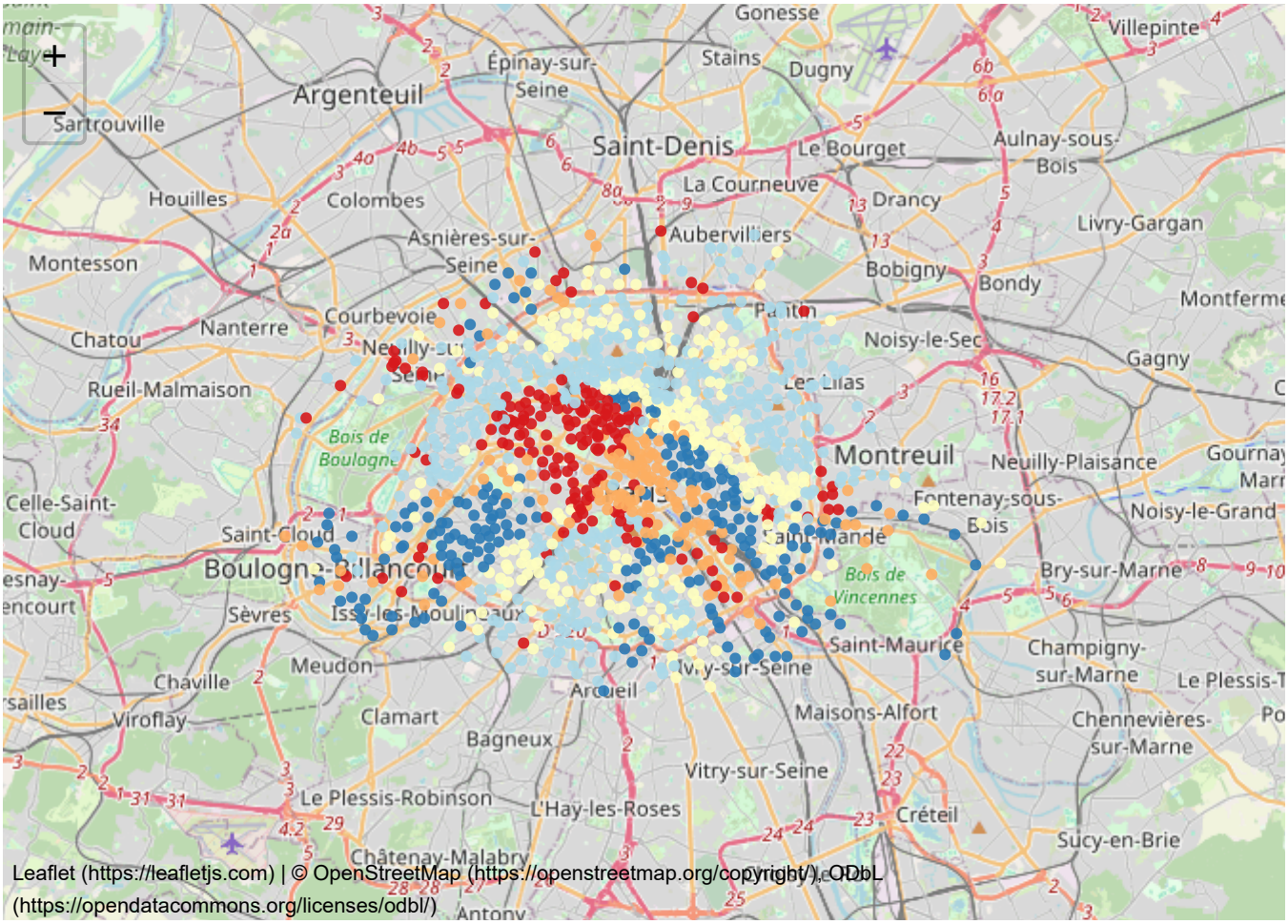
```
k.max = 20
J = rep(NA,k.max)
for (k in 1:k.max){
  out = kmeans(Y, k, nstart=10)
  J[k] = out$betweenss / out$totss
}
plot(2:k.max,J[2:k.max],type='b')
```



Let’s untilize k = 5 from elbow method

```
out.km = kmeans(Y, 5, nstart = 10)

palette = colorFactor("RdYlBu", domain = NULL)
leaflet(position) %>% addTiles() %>%
addCircleMarkers(radius = 3,
color = palette(out.km$cluster),
stroke = FALSE, fillOpacity = 0.9)
```



Comparison of heirarchical and k-means

- Hierarchical clustering identified 4 clusters, whereas k-means suggested between 4 and 6 clusters based on the elbow method.
- The cluster formations differ significantly between the two methods. For example, in hierarchical clustering, central stations are grouped together (light blue), whereas k-means divides them into three separate clusters (red, blue, and orange).
- An extra cluster in k-means appears to capture outliers from other groups, primarily stations located on the periphery of the dataset.

3.5 Summary

- Due to the high number of variables, clustering helps reveal patterns among individual locations.
- PCA reduces the dataset to 5 principal components, though these components do not exhibit strong correlations with each other.
- Both hierarchical clustering and k-means identify between 4 to 6 clusters, but k-means tends to create more clusters, potentially leading to overfitting.

- While both methods generally group nearby locations together, k-means and hierarchical clustering differ in how they handle central and peripheral stations.
- The cluster boundaries are not well-defined, particularly in k-means, where distinctions between clusters remain less clear.