Segmentation of The French Territory Based on Temperature and Wind Time Series

NIANG Mohamed
KAINA Mohamed Abdellah

16 December 2019

Contents

1 Context

1.1 Weather Segmentation

The aim of this project is to perform a segmentation of the French territory based on Temperature and Wind time series gathered at n = 259 grid points using several clustering methods.

1.2 Weather Data

The "weatherdata.Rdata" data set provides temperature and wind temporal evolution for n = 259 grid points at an hourly sampling rate for a given year (p = 8760 hours). Temp denotes the time series for the temperature. Wind denotes the time series for the wind. The GPSpos variable contains the GPS positions (longitude and latitude) of the time series grid points.

2 Loading Packages and Data

2.1 Load Library

```
library(knitr)
library(ggplot2)
library(kernlab)
```

```
##
## Attaching package: 'kernlab'
```

```
## The following object is masked from 'package:ggplot2':
##
##     alpha
```

```
library(FactoMineR)
library(kableExtra)
library(cluster)
library(mclust)
```

```
## Package 'mclust' version 5.4.5
## Type 'citation("mclust")' for citing this R package in publications.
```

```
library(NbClust)
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------------------------- tidyvers
```

```
## v tibble  2.1.3     v purrr   0.3.3
## v tidyr   1.0.0     v dplyr   0.8.3
## v readr   1.3.1     v stringr 1.4.0
## v tibble  2.1.3     v forcats 0.4.0
```

```
## -- Conflicts ------------------------------------------------------------------------- tidyverse_con
## x kernlab::alpha()    masks ggplot2::alpha()
## x purrr::cross()      masks mclust::cross(), kernlab::cross()
## x dplyr::filter()     masks stats::filter()
## x dplyr::group_rows() masks kableExtra::group_rows()
## x dplyr::lag()        masks stats::lag()
## x purrr::map()        masks mclust::map()
```

```
library(factoextra)
```

```
## Welcome! Related Books: `Practical Guide To Cluster Analysis in R` at https://goo.gl/13EFCZ
```

```
library(fpc)
library(maps)
```

```
##
## Attaching package: 'maps'
```

```
## The following object is masked from 'package:purrr':
##
##     map
```

```
## The following object is masked from 'package:mclust':
##
##     map
```

```
## The following object is masked from 'package:cluster':
##
##     votes.repub
```

2.2 Load Data

```
rm(list=ls())
load("weatherdata.Rdata")
ls()
```

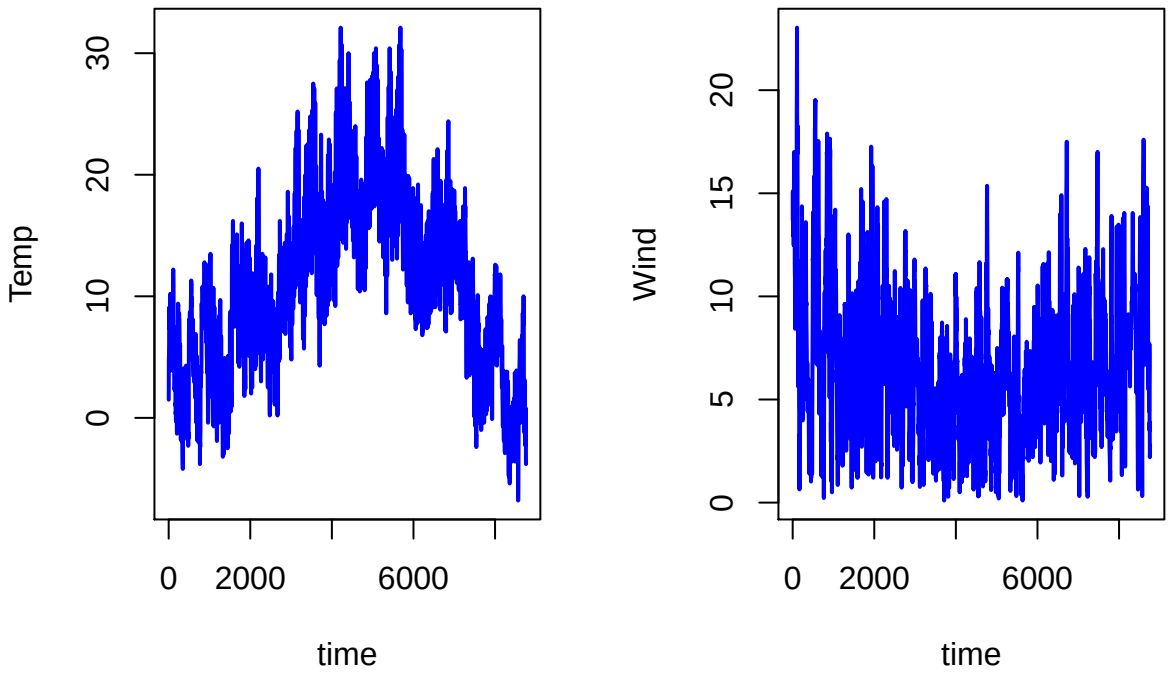
```
## [1] "GPSpos" "Temp"   "Wind"
```

3 Preliminary

Paris city is located at a latitude of 48.51 and a longitude of 2.20 and corresponds to the point i = 59 in the data base.

3.1 Display The Temperature and Wind Data for Paris City

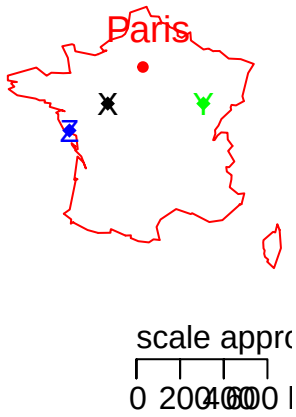
```
CityLat <- 48.51
CityLong <- 2.20
tabpos <- (GPSpos$Lon-CityLong)^2+(GPSpos$Lat-CityLat)^2
i <- which.min(tabpos)
par(mfrow=c(1,2))
plot(Temp[i,],type='l',lwd=2,xlab='time',ylab='Temp',col="blue")
plot(Wind[i,],type='l',lwd=2,xlab='time',ylab='Wind',col="blue")
```



3.2 Representation of The Results of The Clustering Instances on a Map

```
set.seed(1234)
N <- 259
```

```
alea <- sample(1:N,3)
ville1 <- c(GPSpos[[1]][alea[1]], GPSpos[[2]][alea[1]])
ville2 <- c(GPSpos[[1]][alea[2]], GPSpos[[2]][alea[2]])
ville3 <- c(GPSpos[[1]][alea[3]], GPSpos[[2]][alea[3]])

map("world", "France",col="red", xlim=c(-5,10), ylim=c(35,55))
title("Clustering Instances on a Map")
map.scale(2,38, metric=T, relwidth=0.4)
map.cities(country='France', capitals=1, pch=20, col='red')

points(ville1[1],ville1[2], pch=18)
text(ville1[1],ville1[2], label="X")

points(ville2[1],ville2[2], pch=18,col="green")
text(ville2[1],ville2[2], label="Y",col="green")

points(ville3[1],ville3[2], pch=18,col="blue")
text(ville3[1],ville3[2], label="Z",col="blue")
```



4 Wind Clustering

This section aims to cluster wind data.

4.1 Raw Data

In this section, we will study and compare the results of the kmeans and the hierarchical clustering to provide a segmentation into 4 groups of the Wind using the raw time series.

```
winddata <- as.matrix(Wind)
```

```
# Data Dimension : High Dimensional Data
dim(winddata)
```

```
## [1]  259 8760
```

4.1.1 Kmeans Algorithm

```
set.seed(1234)
windkmeans <- kmeans(winddata, centers = 4, nstart = 25)
```

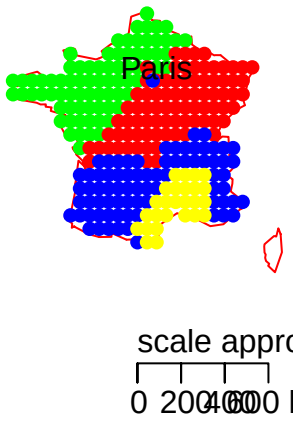
```
table(windkmeans$cluster)
```

```
##
##  1  2  3  4
## 76 82 75 26
```

```
res <- cbind((as.vector(windkmeans$cluster)),as.vector(seq(1:259)))
```

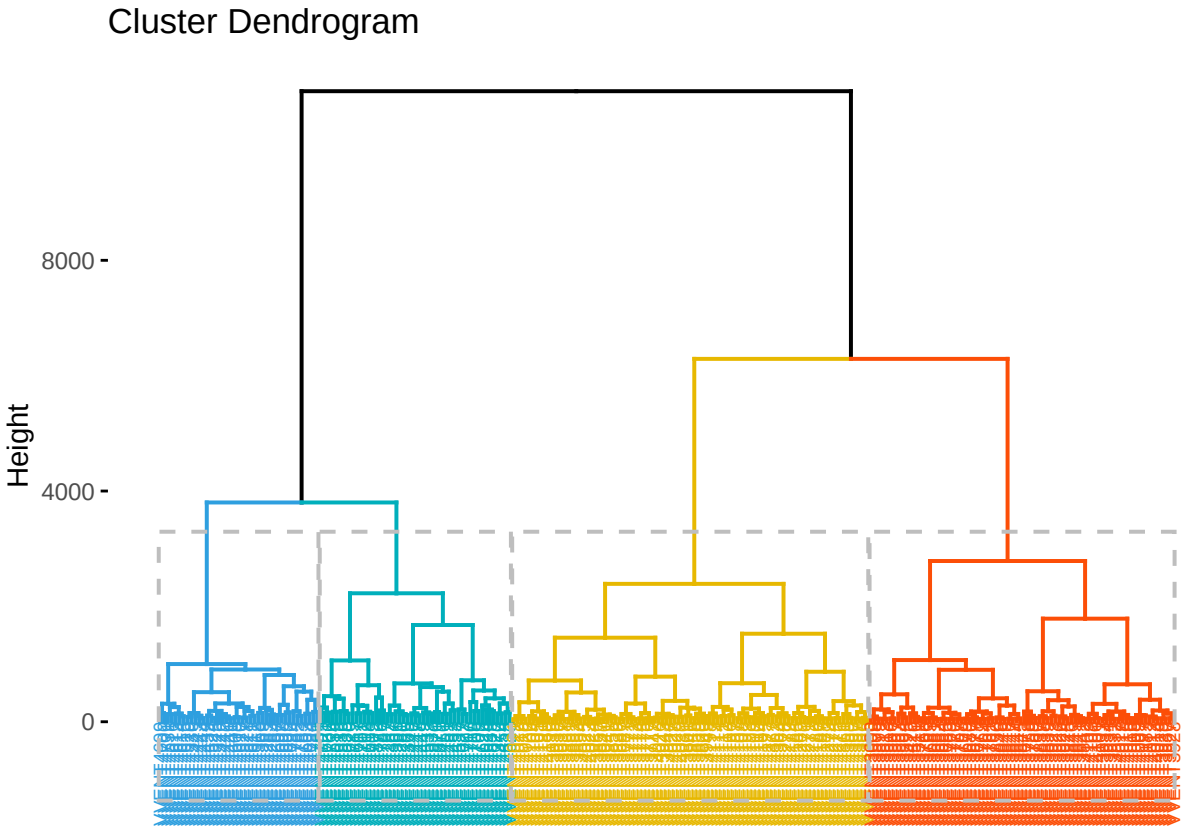
```
map("world", "France", col="red", xlim=  c(-5,10), ylim = c(35,55))
title("Clustering Instances on a Map Using Kmeans")
map.scale(2, 38, metric = T, relwidth = 0.3)
points(GPSpos[[1]][res[res[,1] == 1,2]], GPSpos[[2]][res[res[,1] == 1,2]], pch=16, col = "green")
points(GPSpos[[1]][res[res[,1] == 2,2]], GPSpos[[2]][res[res[,1] == 2,2]], pch=16, col = "red")
points(GPSpos[[1]][res[res[,1] == 3,2]], GPSpos[[2]][res[res[,1] == 3,2]], pch=16, col = "blue")
points(GPSpos[[1]][res[res[,1] == 4,2]], GPSpos[[2]][res[res[,1] == 4,2]], pch=16, col = "yellow")
points(2.8, 48.51, pch=16, col = "blue")
text(3, 49, label = "Paris")
```

Clustering Instances on a Map Using Kmeans



4.1.2 Hierarchical Clustering

```
# We apply the agglomeration method
# List of methods
m <- c( "average", "single", "complete", "ward")
names(m) <- c( "average", "single", "complete", "ward")

# function to compute coefficient
ac <- function(y) {
  agnes(winddata, method = y)$ac
}

map_dbl(m, ac)

##   average    single  complete      ward
## 0.8326556 0.7192021 0.8738594 0.9641170

# We remark that the most powerful method is the Ward method

dist <- dist(winddata, method = 'euclidean')
windHclust <- hclust(dist, method = "ward.D")

# Cut in 4 groups and color by groups
fviz_dend(windHclust, k = 4, # Cut in four groups
    cex = 0.5, # label size
    k_colors = c("#2E9FDF", "#00AFBB", "#E7B800", "#FC4E07"),
    color_labels_by_k = TRUE, # color labels by groups
    rect = TRUE # Add rectangle around groups
)
```



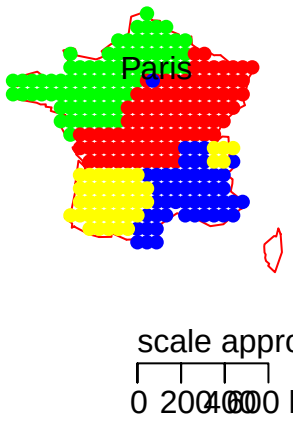

```
# Cut tree into 4 groups
grp <- cutree(windHclust, k = 4)
```

```
table(grp)
```

```
## grp
##  1  2  3  4
## 78 91 49 41
```

```
res <- cbind((as.vector(grp)),as.vector(seq(1:259)))
```

```
map("world", "France",col="red", xlim=c(-5,10), ylim=c(35,55))
title("Clustering Instances on a Map Using Hclust")
map.scale(2,38,metric=T, relwidth=0.3)
points(GPSpos[[1]][res[res[,1] == 1,2]],GPSpos[[2]][res[res[,1] == 1,2]], pch=16,col="green")
points(GPSpos[[1]][res[res[,1] == 2,2]],GPSpos[[2]][res[res[,1] == 2,2]], pch=16,col="red")
points(GPSpos[[1]][res[res[,1] == 3,2]],GPSpos[[2]][res[res[,1] == 3,2]], pch=16,col="blue")
points(GPSpos[[1]][res[res[,1] == 4,2]],GPSpos[[2]][res[res[,1] == 4,2]], pch=16,col="yellow")
points(2.8, 48.51, pch=16,col="blue")
text(3, 49, label="Paris")
```



4.1.3 Conclusion : Comparison of The Results of The Two Algorithms (Kmeans and Hclust)

To compare the two models, we use the *cluster.stats*() function of the fpc library. Among the values returned by the function *cluster.stats*(), there are two indices to compare the performance of two clusters, namely the within.cluster.ss and the means.silwidth.

Most often, we focus on using within.cluster.ss and avg.silwidth to validate the clustering method. The within.cluster.ss measurement stands for the within clusters sum of squares, and avg.silwidth represents the average silhouette width.

- within.cluster.ss measurement shows how closely related objects are in clusters; the smaller the value, the more closely related objects are within the cluster.
- avg.silwidth is a measurement that considers how closely related objects are within the cluster and how clusters are separated from each other. The silhouette value usually ranges from 0 to 1; a value closer to 1 suggests the data is better clustered.

```
# Stats Kmeans
stat_kmeans <- cluster.stats(dist, windkmeans$cluster)
within_kmeans <- stat_kmeans$within.cluster.ss
avg_kmeans <- stat_kmeans$avg.silwidth

# Stats Hclust
stat_hclust <- cluster.stats(dist, grp)
within_hclust <- stat_hclust$within.cluster.ss
avg_hclust <- stat_hclust$avg.silwidth

statsmodelsKmeans <- c(within_kmeans, avg_kmeans)
statsmodelsHclust <- c(within_hclust, avg_hclust)
```

```
resultsKmeans <- data.frame("Kmeans" = c("within.cluster.ss","avg.silwidth"), "Stats Kmeans" = statsmod
resultsHclust <- data.frame("Hclust" = c("within.cluster.ss","avg.silwidth"), "Stats Hclust" = statsmod
resultfinal <- cbind(resultsKmeans,resultsHclust)

# Comparison Table
kable(arrange(resultfinal,desc(statsmodelsKmeans),desc(statsmodelsHclust)), digits = 3) %>%
  kable_styling(bootstrap_options = c("striped", "hover"),
                full_width = F,
                font_size = 18,
                position = "center")
```

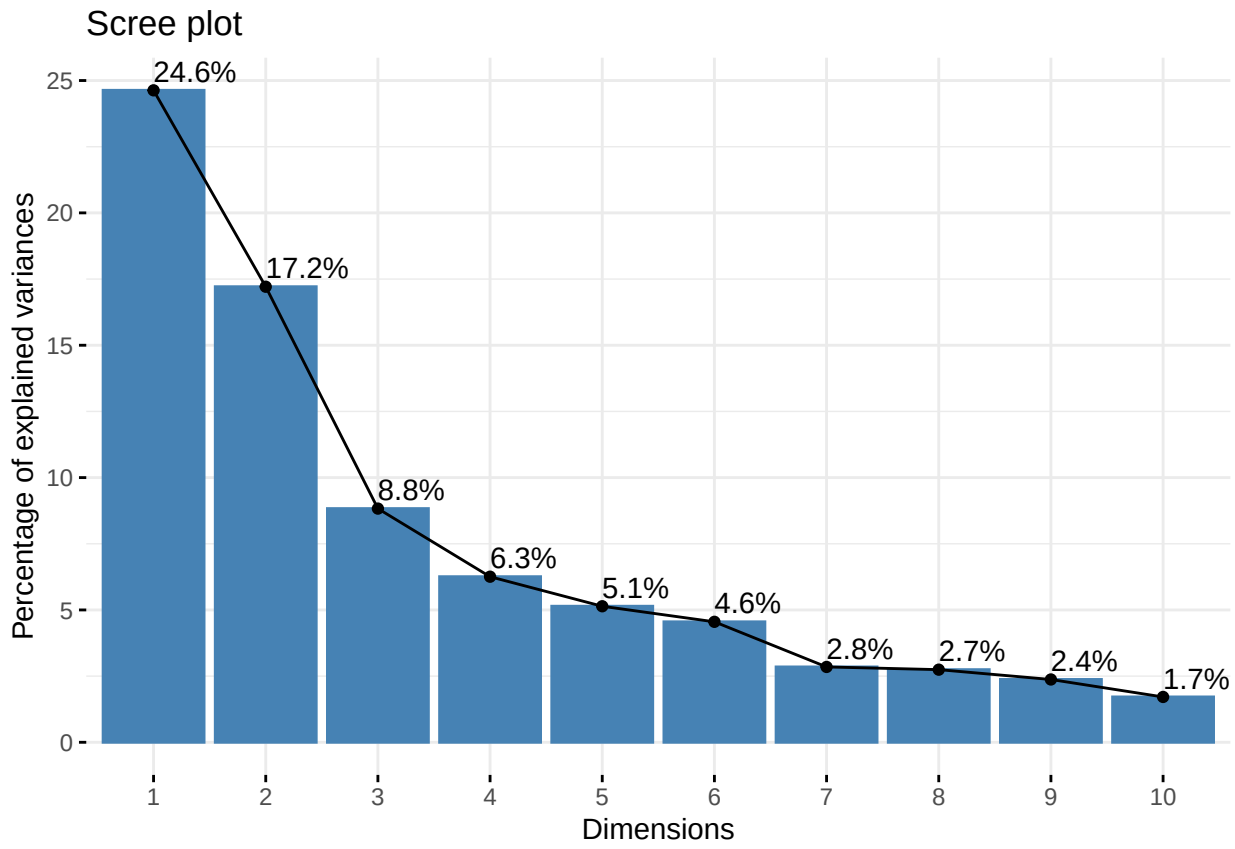
Kmeans	Stats.Kmeans	Hclust	Stats.Hclust
within.cluster.ss	8181826.681	within.cluster.ss	8744705.954
avg.silwidth	0.198	avg.silwidth	0.182

Comments: Based on the results of the above table, we conclude that the Kmeans is **the best model because it has within.cluster.ss smaller and one avg.silwidth larger than the Hclust**.

4.2 Feature Extraction

In this section, We use a Principal Component Analysis (PCA) to reduce the dimension of the n = 259 time series for the Wind data.

```
windPca <- PCA(winddata, graph = FALSE)

fviz_eig(windPca, addlabels = TRUE)
```



Comments: The number of component is determined at the point, beyond which the remaining eigenvalues are all relatively small and of comparable size. From the plot above, we might want to stop at the sixth principal component. 66.61 % of the information (variances) contained in the data are retained by the first six principal components.

4.3 Clustering With The Wind Data

In this section, we Compute and study a segmentation in 4 groups of the wind in France, based on the PCA representation keeping only 10 principal components using kmeans and hierarchical clustering.

```
windPca_10 <- PCA(winddata, ncp = 10, graph = FALSE)
windPca_coord_10 <- as.matrix(windPca_10$ind$coord)
```

4.3.1 Kmeans Algorithm With PCA

```
set.seed(1234)
windkmeansPca <- kmeans(windPca_coord_10, centers = 4, nstart = 25)
```

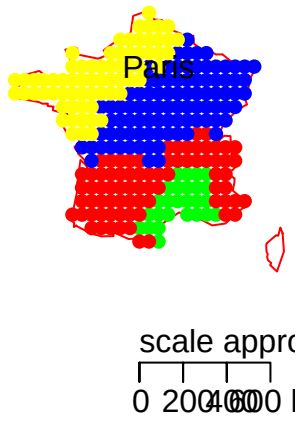
```
table(windkmeansPca$cluster)
```

```
##
##  1  2  3  4
## 26 75 91 67
```

```
res <- cbind((as.vector(windkmeansPca$cluster)),as.vector(seq(1:259)))
```

```
map("world", "France", col="red", xlim=  c(-5,10), ylim = c(35,55))
title("Clustering Instances on a Map Using Kmeans With PCA")
map.scale(2, 38, metric = T, relwidth = 0.3)
```

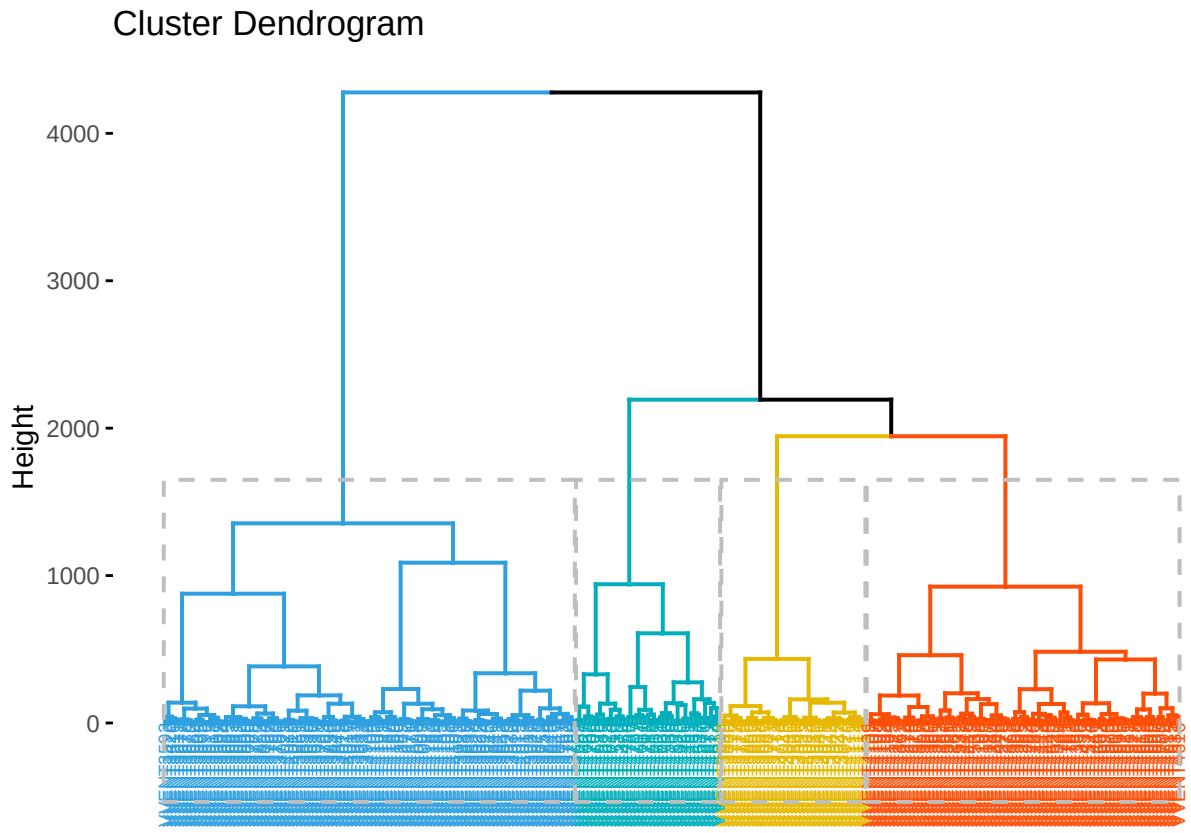
```
points(GPSpos[[1]][res[res[,1] == 1,2]], GPSpos[[2]][res[res[,1] == 1,2]], pch=16, col = "green")
points(GPSpos[[1]][res[res[,1] == 2,2]], GPSpos[[2]][res[res[,1] == 2,2]], pch=16, col = "red")
points(GPSpos[[1]][res[res[,1] == 3,2]], GPSpos[[2]][res[res[,1] == 3,2]], pch=16, col = "blue")
points(GPSpos[[1]][res[res[,1] == 4,2]], GPSpos[[2]][res[res[,1] == 4,2]], pch=16, col = "yellow")
points(2.8, 48.51, pch=16, col = "blue")
text(3, 49, label = "Paris")
```



4.3.2 Hierarchical Clustering With PCA

```
# function to compute coefficient
ac <- function(y) {
  agnes(windPca_coord_10, method = y)$ac
}

map_dbl(m, ac)

##   average    single  complete      ward
## 0.9091479 0.7600996 0.9346522 0.9787904

# We remark that the most powerful method is the Ward method

dist <- dist(windPca_coord_10, method = 'euclidean')
windHclustPca <- hclust(dist, method = "ward.D")

# Cut in 4 groups and color by groups
fviz_dend(windHclustPca, k = 4, # Cut in four groups
    cex = 0.5, # label size
    k_colors = c("#2E9FDF", "#00AFBB", "#E7B800", "#FC4E07"),
    color_labels_by_k = TRUE, # color labels by groups
```

```
    rect = TRUE # Add rectangle around groups
)
```




```
# Cut tree into 4 groups
grp <- cutree(windHclustPca, k = 4)
```

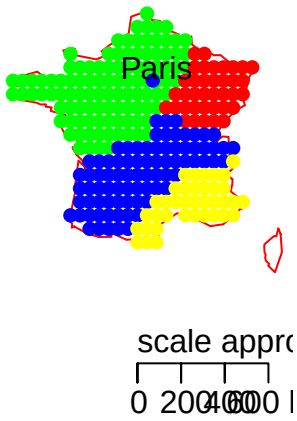
```
table(grp)
```

```
## grp
##   1   2   3   4
## 105  37  80  37
```

```
res <- cbind((as.vector(grp)),as.vector(seq(1:259)))
```

```
map("world", "France",col="red", xlim=c(-5,10), ylim=c(35,55))
title("Clustering Instances on a Map Using Hclust With PCA")
map.scale(2,38,metric=T, relwidth=0.3)
points(GPSpos[[1]][res[res[,1] == 1,2]],GPSpos[[2]][res[res[,1] == 1,2]], pch=16,col="green")
points(GPSpos[[1]][res[res[,1] == 2,2]],GPSpos[[2]][res[res[,1] == 2,2]], pch=16,col="red")
points(GPSpos[[1]][res[res[,1] == 3,2]],GPSpos[[2]][res[res[,1] == 3,2]], pch=16,col="blue")
points(GPSpos[[1]][res[res[,1] == 4,2]],GPSpos[[2]][res[res[,1] == 4,2]], pch=16,col="yellow")
points(2.8, 48.51, pch=16,col="blue")
text(3, 49, label="Paris")
```

Clustering Instances on a Map Using Hclust With PCA



4.3.3 Conclusion : Comparison of The Results of The Two Algorithms With (Kmeans and Hclust)

```
# Stats Kmeans With PCA
stat_kmeansPca <- cluster.stats(dist, windkmeansPca$cluster)
within_kmeansPca <- stat_kmeansPca$within.cluster.ss
avg_kmeansPca <- stat_kmeansPca$avg.silwidth

# Stats Hclust With PCA
stat_hclustPca <- cluster.stats(dist, grp)
within_hclustPca <- stat_hclustPca$within.cluster.ss
avg_hclustPca <- stat_hclustPca$avg.silwidth

statsmodelsKmeansPca <- c(within_kmeansPca, avg_kmeansPca)
statsmodelsHclustPca <- c(within_hclustPca, avg_hclustPca)
resultsKmeansPca <- data.frame("Kmeans With PCA" = c("within.cluster.ss","avg.silwidth"), "Stats Kmeans
resultsHclustPca <- data.frame("Hclust With PCA" = c("within.cluster.ss","avg.silwidth"), "Stats Hclust
resultfinalPca <- cbind(resultsKmeansPca,resultsHclustPca)

# Comparison Table
kable(arrange(resultfinalPca,desc(statsmodelsKmeansPca),desc(statsmodelsHclustPca)), digits = 3) %>%
  kable_styling(bootstrap_options = c("striped", "hover"),
                full_width = F,
                font_size = 10,
                position = "center")
```

Kmeans.With.PCA	Stats.Kmeans.With.PCA	Hclust.With.PCA	Stats.Hclust.With.PCA
within.cluster.ss	845826.669	within.cluster.ss	919679.678
avg.silwidth	0.256	avg.silwidth	0.226

Comments: Based on the results of the above table, we conclude that the Kmeans with PCA is **the best model because it has within.cluster.ss smaller and one avg.silwidth larger than the Hclust with PCA**. However, the results obtained with the principal component analysis (PCA) are better than those obtained previously.

5 Temperature Clustering

This section aims to cluster temperature data.

5.1 Raw Data

In this section, we will study and compare the results of the kmeans and the hierarchical clustering to provide a segmentation into 4 groups of the temperature using the raw time series.

```
tempdata <- as.matrix(Temp)
```

```
# Data Dimension : High Dimensional Data
dim(tempdata)
```

```
## [1]  259 8760
```

5.1.1 Kmeans Algorithm

```
set.seed(1234)
tempkmeans <- kmeans(tempdata, centers = 4, nstart = 25)
```

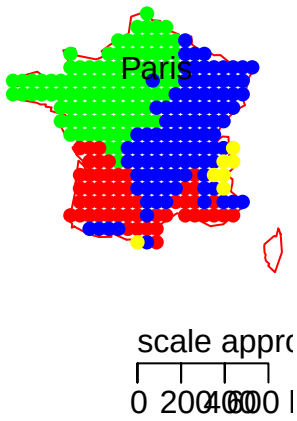
```
table(tempkmeans$cluster)
```

```
##
##   1   2   3   4
##  99  53 100   7
```

```
res <- cbind((as.vector(tempkmeans$cluster)),as.vector(seq(1:259)))
```

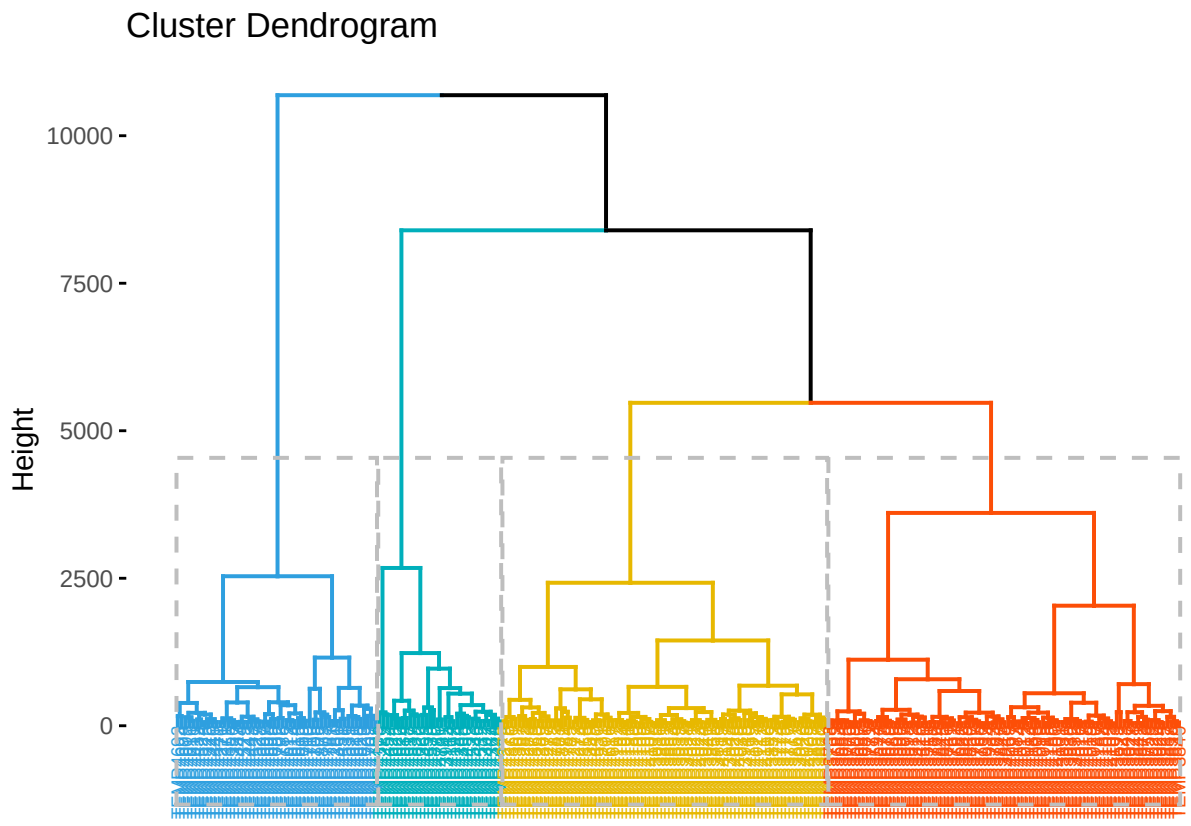
```
map("world", "France", col="red", xlim=  c(-5,10), ylim = c(35,55))
title("Clustering Instances on a Map Using Kmeans")
map.scale(2, 38, metric = T, relwidth = 0.3)
points(GPSpos[[1]][res[res[,1] == 1,2]], GPSpos[[2]][res[res[,1] == 1,2]], pch=16, col = "green")
points(GPSpos[[1]][res[res[,1] == 2,2]], GPSpos[[2]][res[res[,1] == 2,2]], pch=16, col = "red")
points(GPSpos[[1]][res[res[,1] == 3,2]], GPSpos[[2]][res[res[,1] == 3,2]], pch=16, col = "blue")
points(GPSpos[[1]][res[res[,1] == 4,2]], GPSpos[[2]][res[res[,1] == 4,2]], pch=16, col = "yellow")
points(2.8, 48.51, pch=16, col = "blue")
text(3, 49, label = "Paris")
```

Clustering Instances on a Map Using Kmeans

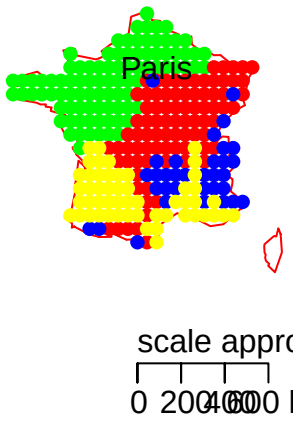


5.1.2 Hierarchical Clustering

```
# function to compute coefficient
ac <- function(y) {
  agnes(tempdata, method = y)$ac
}

map_dbl(m, ac)

##   average    single  complete      ward
## 0.8852868 0.8017066 0.9473666 0.9689150

# We remark that the most powerful method is the Ward method

dist <- dist(tempdata, method = 'euclidean')
tempHclust <- hclust(dist, method = "ward.D")

# Cut in 4 groups and color by groups
fviz_dend(tempHclust, k = 4, # Cut in four groups
    cex = 0.5, # label size
    k_colors = c("#2E9FDF", "#00AFBB", "#E7B800", "#FC4E07"),
    color_labels_by_k = TRUE, # color labels by groups
    rect = TRUE # Add rectangle around groups
)
```




```
# Cut tree into 4 groups
grp <- cutree(tempHclust, k = 4)

table(grp)

## grp
##  1  2  3  4
## 91 84 32 52

res <- cbind((as.vector(grp)),as.vector(seq(1:259)))

map("world", "France",col="red", xlim=c(-5,10), ylim=c(35,55))
title("Clustering Instances on a Map Using Hclust")
map.scale(2,38,metric=T, relwidth=0.3)
points(GPSpos[[1]][res[res[,1] == 1,2]],GPSpos[[2]][res[res[,1] == 1,2]], pch=16,col="green")
points(GPSpos[[1]][res[res[,1] == 2,2]],GPSpos[[2]][res[res[,1] == 2,2]], pch=16,col="red")
points(GPSpos[[1]][res[res[,1] == 3,2]],GPSpos[[2]][res[res[,1] == 3,2]], pch=16,col="blue")
points(GPSpos[[1]][res[res[,1] == 4,2]],GPSpos[[2]][res[res[,1] == 4,2]], pch=16,col="yellow")
points(2.8, 48.51, pch=16,col="blue")
text(3, 49, label="Paris")
```



5.1.3 Conclusion : Comparison of The Results of The Two Algorithms (Kmeans and Hclust)

```
# Stats Kmeans
stat_kmeans <- cluster.stats(dist, tempkmeans$cluster)
within_kmeans <- stat_kmeans$within.cluster.ss
avg_kmeans <- stat_kmeans$avg.silwidth

# Stats Hclust
stat_hclust <- cluster.stats(dist, grp)
within_hclust <- stat_hclust$within.cluster.ss
avg_hclust <- stat_hclust$avg.silwidth

statsmodelsKmeans <- c(within_kmeans, avg_kmeans)
statsmodelsHclust <- c(within_hclust, avg_hclust)
resultsKmeans <- data.frame("Kmeans" = c("within.cluster.ss","avg.silwidth"), "Stats Kmeans" = statsmod
resultsHclust <- data.frame("Hclust" = c("within.cluster.ss","avg.silwidth"), "Stats Hclust" = statsmod
resultfinal <- cbind(resultsKmeans,resultsHclust)

# Comparison Table
kable(arrange(resultfinal,desc(statsmodelsKmeans),desc(statsmodelsHclust)), digits = 3) %>%
  kable_styling(bootstrap_options = c("striped", "hover"),
                full_width = F,
                font_size = 18,
                position = "center")
```

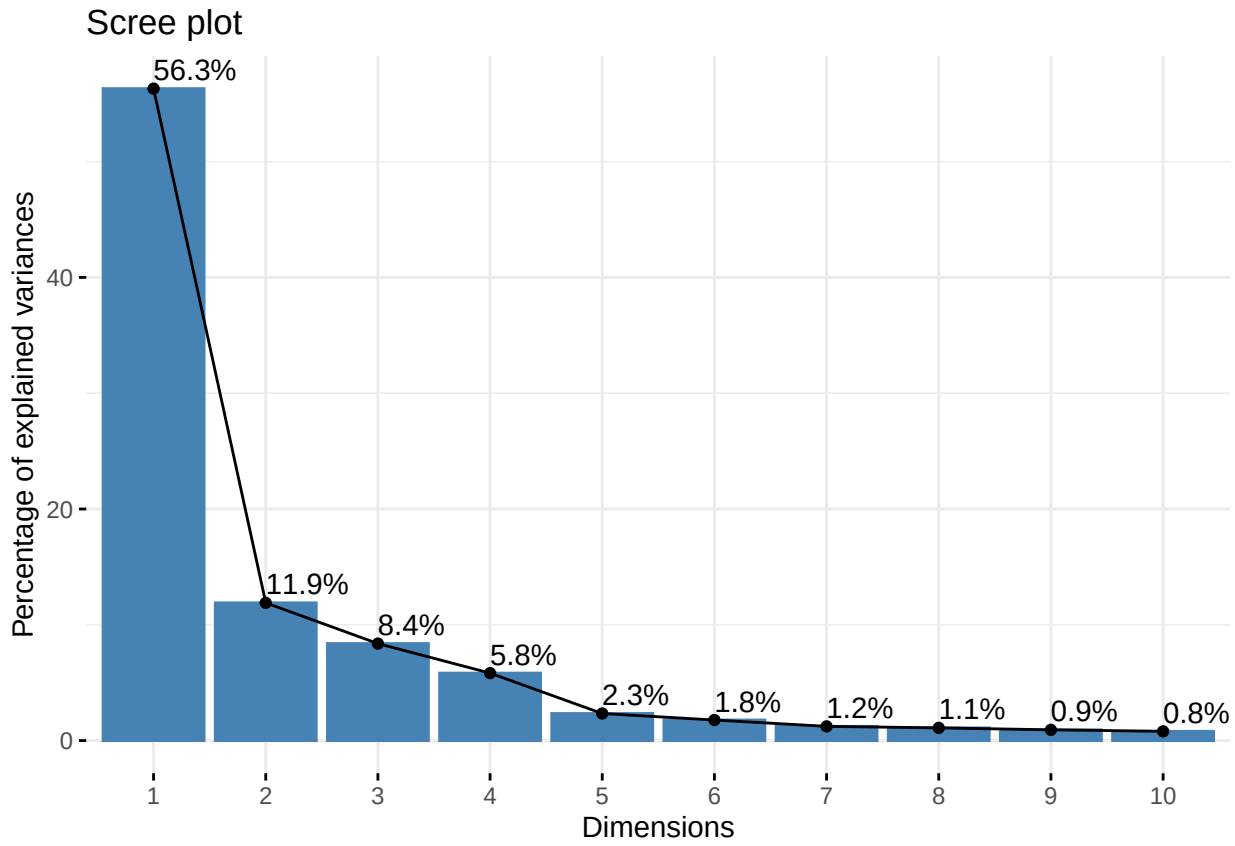
Kmeans	Stats.Kmeans	Hclust	Stats.Hclust
within.cluster.ss	8094522.932	within.cluster.ss	8535801.008
avg.silwidth	0.243	avg.silwidth	0.204

Comments: Based on the results of the above table, we conclude that the Kmeans is **the best model because it has within.cluster.ss smaller and one avg.silwidth larger than the Hclust**.

5.2 Feature Extraction

In this section, We use a Principal Component Analysis (PCA) to reduce the dimension of the n = 259 time series for the temperature data.

```
tempPca <- PCA(tempdata, graph = FALSE)
fviz_eig(tempPca, addlabels = TRUE)
```



Comments: The number of component is determined at the point, beyond which the remaining eigenvalues are all relatively small and of comparable size. From the plot above, we might want to stop at the fourth principal component. 84.74 % of the information (variances) contained in the data are retained by the first four principal components.

5.3 Clustering Using Model Based : MClust

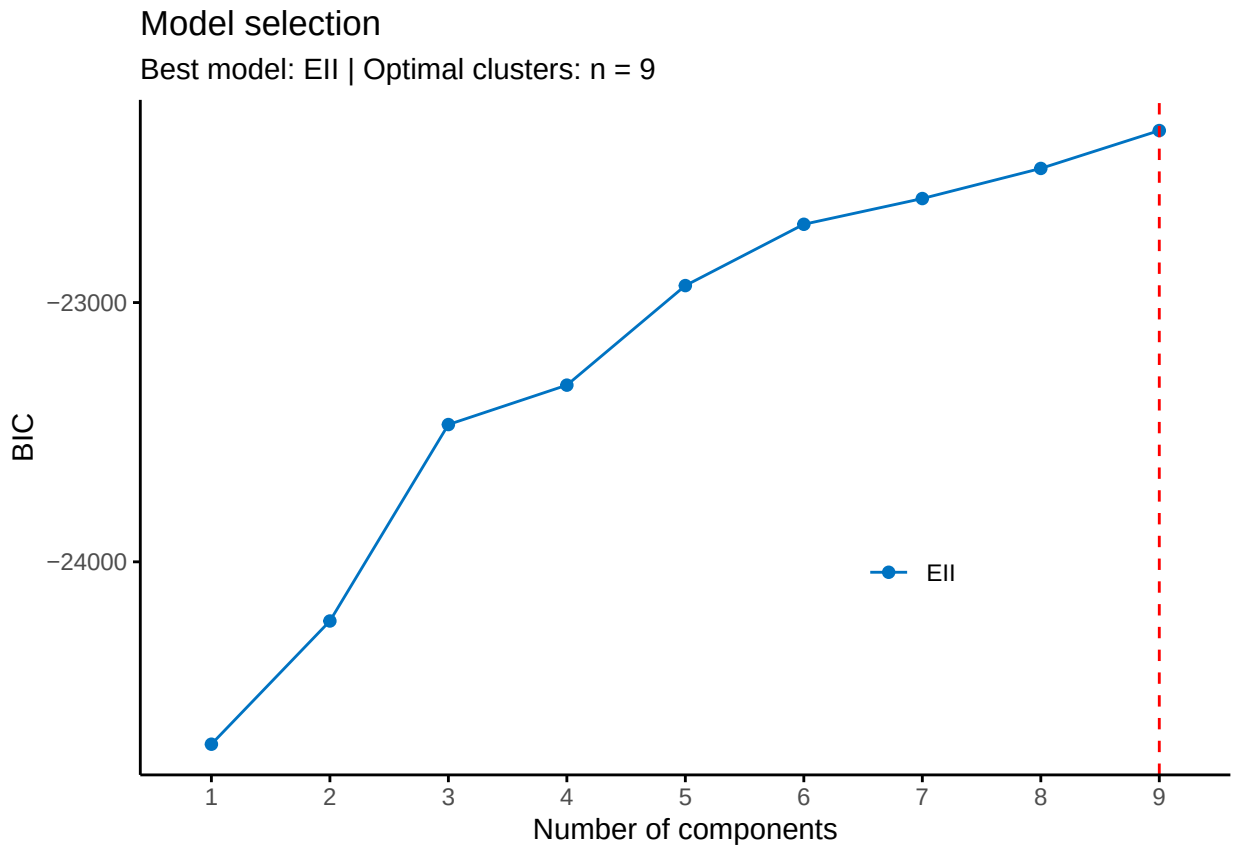
In this section, We use a Principal Component Analysis (PCA) to reduce the dimension of the n = 259 time series for the temperature data and we compute and study a segmentation of the temperature time

series, based on the PCA representation keeping only 10 principal components using a model based clustering method : Mclust.

In this work, 3 assumptions will be studied:

- spherical model with equal volume (EII)
- spherical model with unequal volume (VII)
- diagonal, varying volume and shape (VVI)

```
tempPca_10 <- PCA(tempdata, ncp = 10, graph = FALSE)
tempPca_coord_10 <- as.matrix(tempPca_10$ind$coord)

tempmclust_EII <- Mclust(tempPca_coord_10, modelNames = "EII")

summary(tempmclust_EII) # Model = EII  and Number of Components = 9

## ----------------------------------------------------
## Gaussian finite mixture model fitted by EM algorithm
## ----------------------------------------------------
##
## Mclust EII (spherical, equal volume) model with 9 components:
##
##  log-likelihood   n df       BIC       ICL
##       -10893.21 259 99 -22336.55 -22361.08
##
## Clustering table:
##  1  2  3  4  5  6  7  8  9
## 40 24 18 57 21  7 43 35 14

# BIC values used for choosing the number of clusters
fviz_mclust(tempmclust_EII, "BIC", palette = "jco")
```



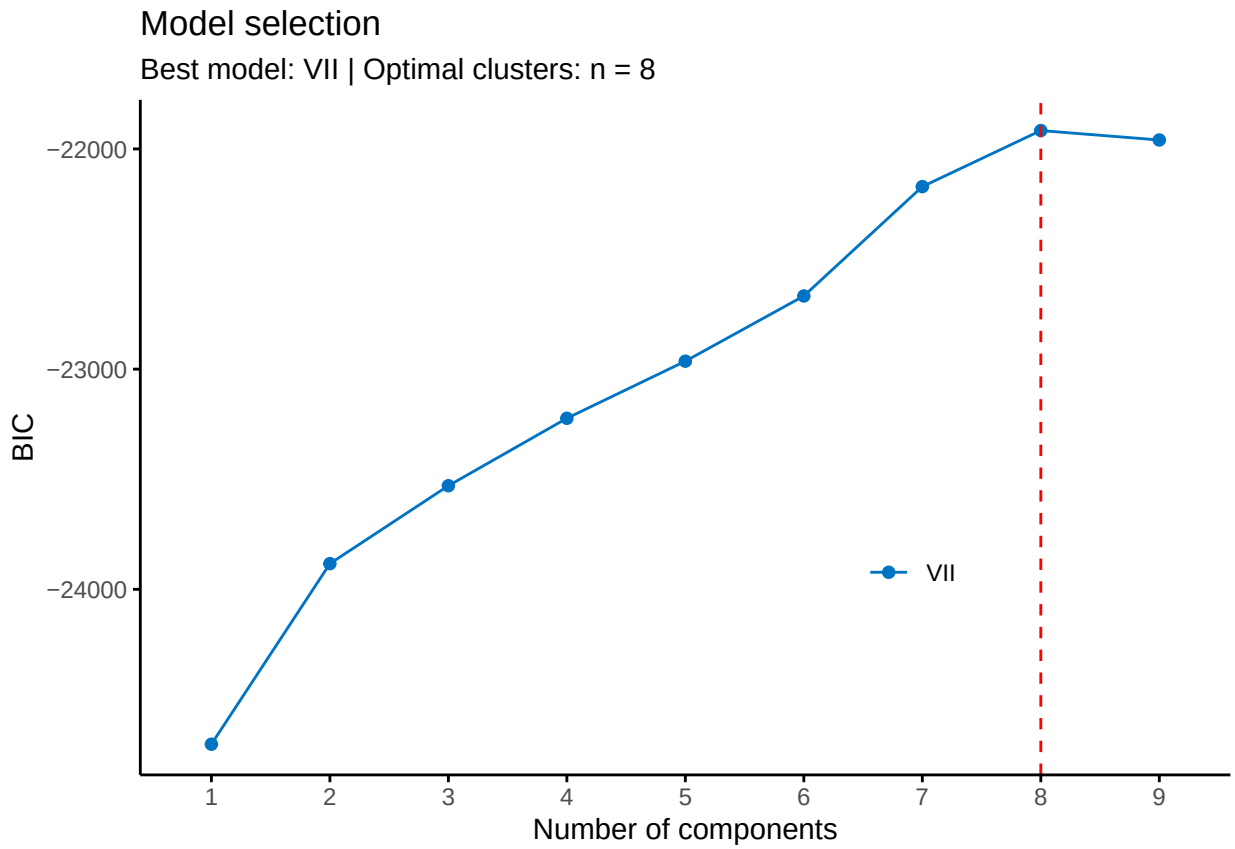
Comments: A higher value of the BIC is observed in the 9th component. Therefore, we take $K = 9$ as the number of classes.

```
tempmclust_VII <- Mclust(tempPca_coord_10, modelNames = "VII")
```

```
summary(tempmclust_VII) # Model = VII  and Number of Components = 8
```

```
## ----------------------------------------------------
## Gaussian finite mixture model fitted by EM algorithm
## ----------------------------------------------------
##
## Mclust VII (spherical, varying volume) model with 8 components:
##
##  log-likelihood   n df       BIC       ICL
##       -10694.37 259 95 -21916.64 -21928.94
##
## Clustering table:
##  1  2  3  4  5  6  7  8
## 34  7 22 51 41 48 24 32
```

```
# BIC values used for choosing the number of clusters
fviz_mclust(tempmclust_VII, "BIC", palette = "jco")
```



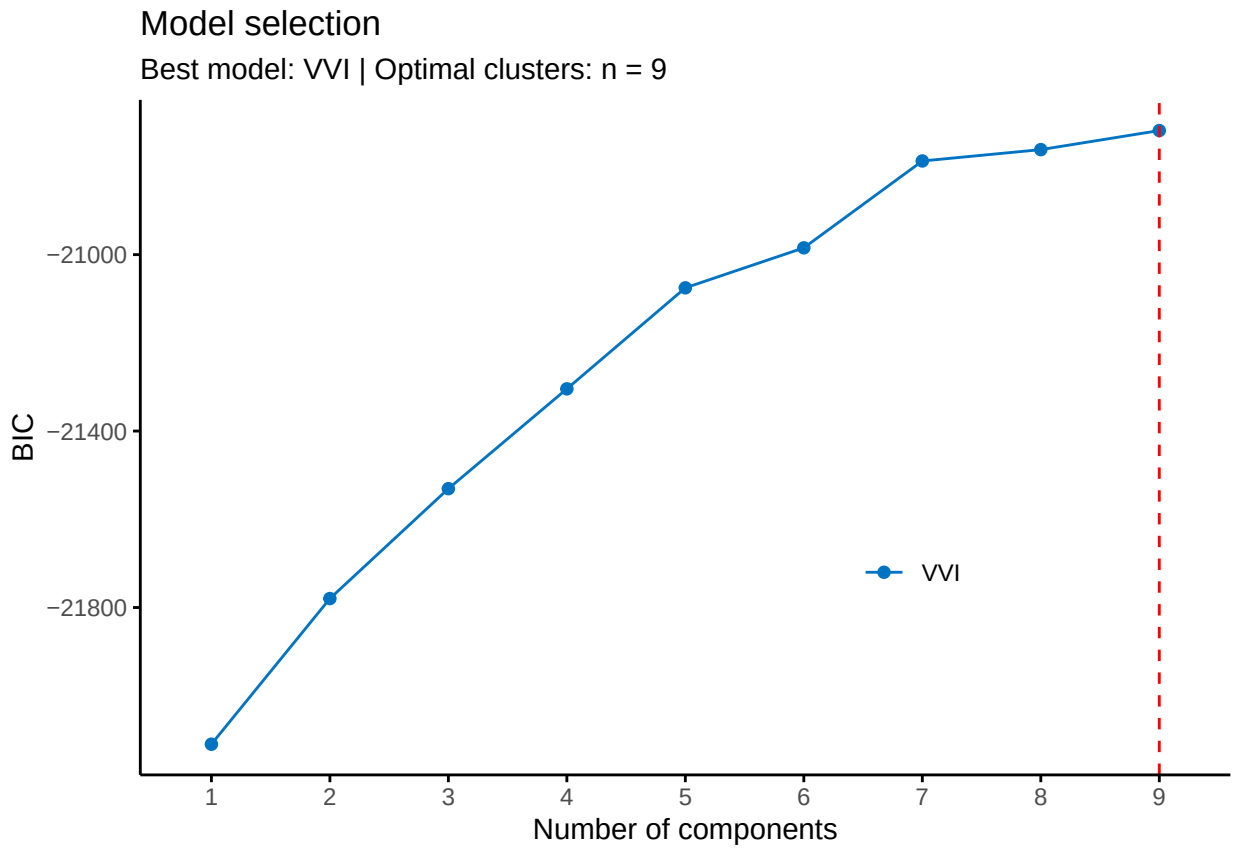
Comments: A higher value of the BIC is observed in the 8th component. Therefore, we take $K = 8$ as the number of classes.

```
tempmclust_VVI <- Mclust(tempPca_coord_10, modelNames = "VVI")
```

```
summary(tempmclust_VVI) # Model = VVI  and Number of Components = 9
```

```
## ----------------------------------------------------
## Gaussian finite mixture model fitted by EM algorithm
## ----------------------------------------------------
##
## Mclust VVI (diagonal, varying volume and shape) model with 9 components:
##
##  log-likelihood   n  df       BIC       ICL
##       -9836.996 259 188 -20718.67 -20734.58
##
## Clustering table:
##  1  2  3  4  5  6  7  8  9
## 23 43 21 34 32 30 26 37 13
```

```
# BIC values used for choosing the number of clusters
fviz_mclust(tempmclust_VVI, "BIC", palette = "jco")
```



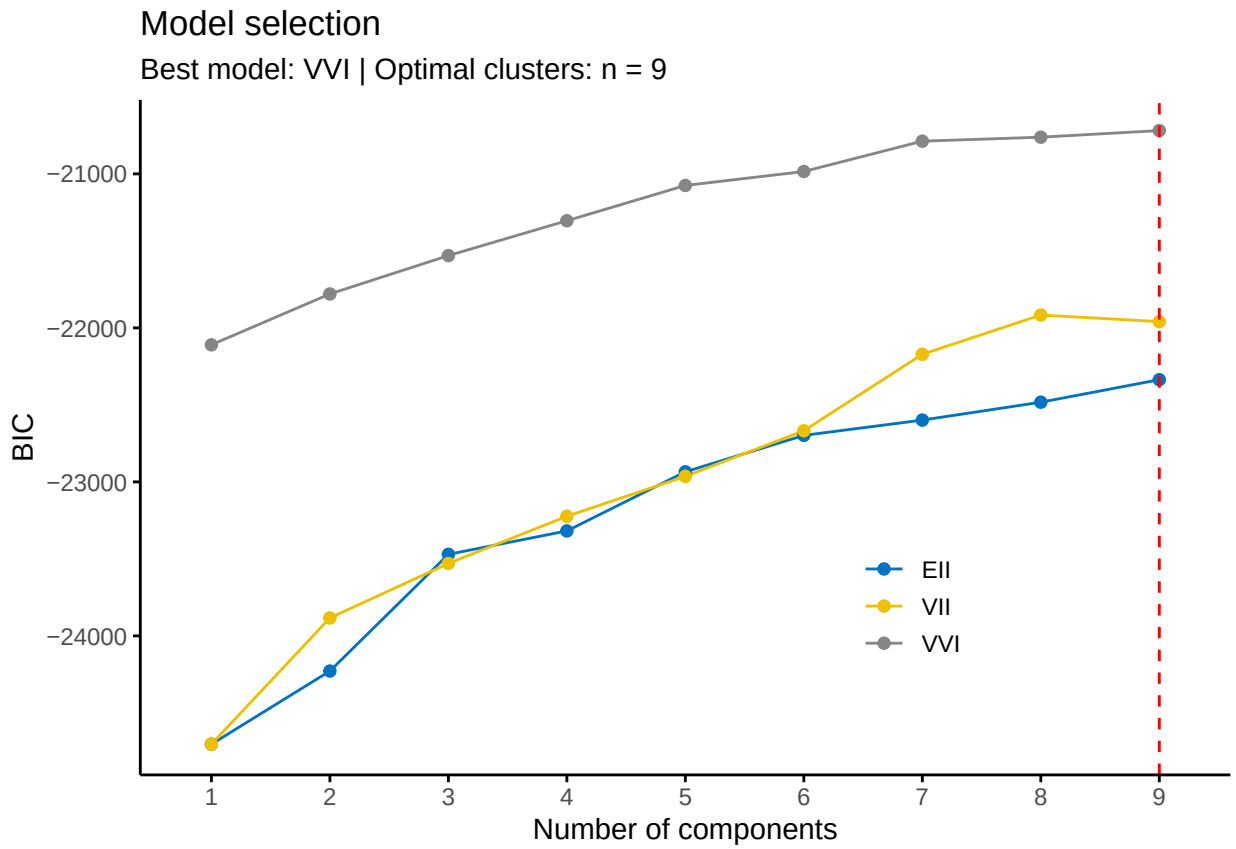
Comments: A higher value of the BIC is observed in the 9th component. Therefore, we take $K = 9$ as the number of classes.

```
tempmclust <- Mclust(tempPca_coord_10, modelNames = c("EII","VII","VVI"))
```

```
summary(tempmclust) # Model = VVI  and Number of Components = 9
```

```
## ----------------------------------------------------
## Gaussian finite mixture model fitted by EM algorithm
## ----------------------------------------------------
##
## Mclust VVI (diagonal, varying volume and shape) model with 9 components:
##
##  log-likelihood   n  df       BIC       ICL
##       -9836.996 259 188 -20718.67 -20734.58
##
## Clustering table:
##  1  2  3  4  5  6  7  8  9
## 23 43 21 34 32 30 26 37 13
```

```
# BIC values used for choosing the number of clusters
fviz_mclust(tempmclust, "BIC", palette = "jco")
```



Comments: The model given by 'Mclust' object is $(VVI, 9)$. Thus the analysis of the results shows that the best model is the VVI because it has the largest BIC. And a higher value of the BIC is observed in the 9th component. Therefore, we take $K = 9$ as the number of classes.

```
tempmclust_VVI <- Mclust(tempPca_coord_10, modelNames = "VVI", G=9)
```

```
grp <-  tempmclust_VVI$classification
```

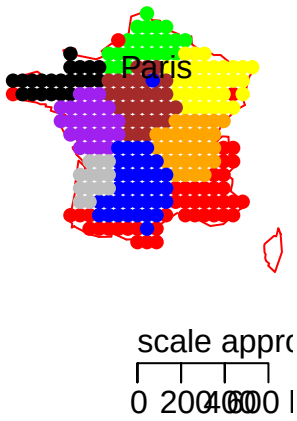
```
table(tempmclust_VVI$classification)
```

```
##
##  1  2  3  4  5  6  7  8  9
## 23 43 21 34 32 30 26 37 13
```

```
res <- cbind((as.vector(grp)),as.vector(seq(1:259)))
```

```
map("world", "France", col = "red", xlim = c(-5,10), ylim = c(35,55))
title("Clustering Instances on a Map Using Mclust")
map.scale(2, 38, metric = T, relwidth = 0.3)
points(GPSpos[[1]][res[res[,1] == 1,2]],GPSpos[[2]][res[res[,1] == 1,2]], pch=16,col="green")
points(GPSpos[[1]][res[res[,1] == 2,2]],GPSpos[[2]][res[res[,1] == 2,2]], pch=16,col="red")
points(GPSpos[[1]][res[res[,1] == 3,2]],GPSpos[[2]][res[res[,1] == 3,2]], pch=16,col="black")
points(GPSpos[[1]][res[res[,1] == 4,2]],GPSpos[[2]][res[res[,1] == 4,2]], pch=16,col="yellow")
points(GPSpos[[1]][res[res[,1] == 5,2]],GPSpos[[2]][res[res[,1] == 5,2]], pch=16,col="brown")
points(GPSpos[[1]][res[res[,1] == 6,2]],GPSpos[[2]][res[res[,1] == 6,2]], pch=16,col="Orange")
points(GPSpos[[1]][res[res[,1] == 7,2]],GPSpos[[2]][res[res[,1] == 7,2]], pch=16,col="Purple")
points(GPSpos[[1]][res[res[,1] == 8,2]],GPSpos[[2]][res[res[,1] == 8,2]], pch=16,col="blue")
points(GPSpos[[1]][res[res[,1] == 9,2]],GPSpos[[2]][res[res[,1] == 9,2]], pch=16,col="gray")
points(2.8, 48.51, pch = 16, col = "blue")
```

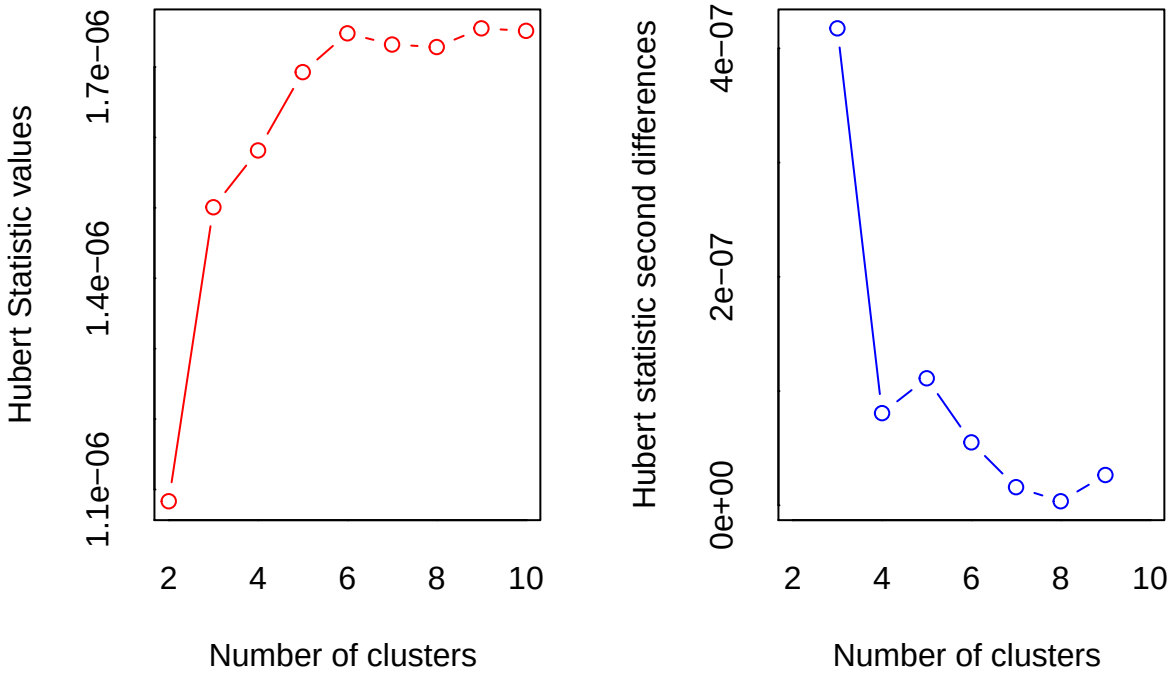
```
text(3, 49, label = "Paris")
```



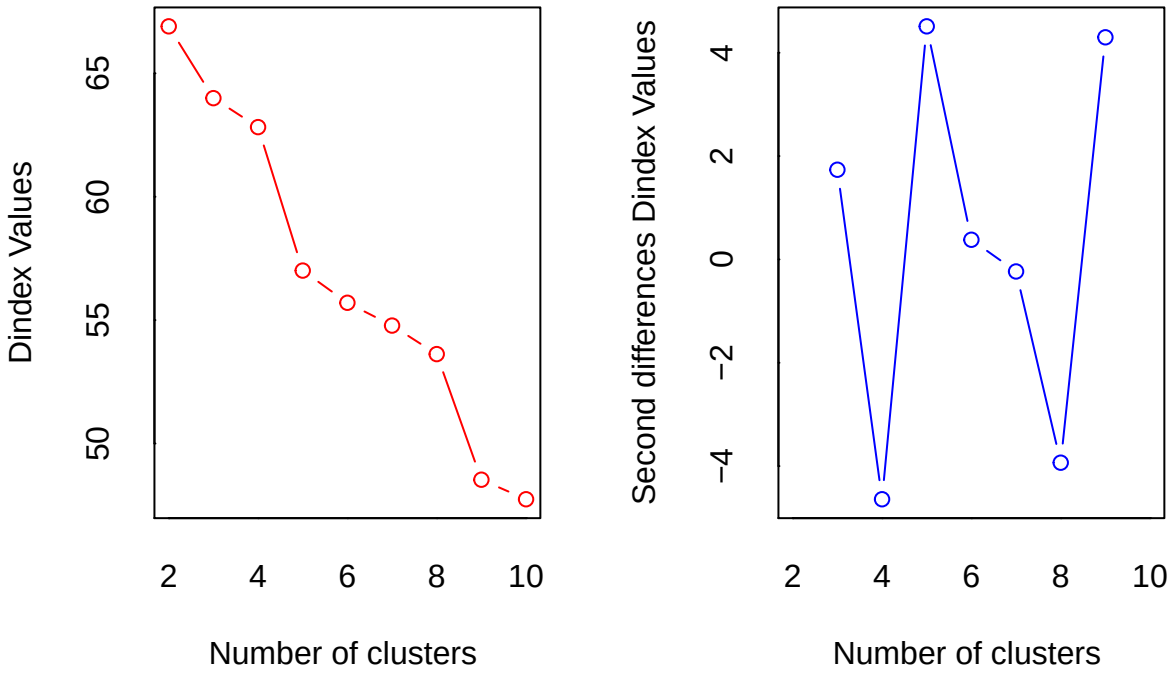
5.4 Clustering Using Spectral Clustering

In this section, we use the 'specc()' function of the 'kernlab' library for clustering the temperature observations using the 10 principal components of the PCA.

```
# Let's determine the number of clusters
res.nbclust <- tempPca_coord_10 %>%
  NbClust(distance = "euclidean",
          min.nc = 2, max.nc = 10,
          method = "complete", index ="all")
```




```
## *** : The Hubert index is a graphical method of determining the number of clusters.
##                 In the plot of Hubert index, we seek a significant knee that corresponds to a
##                 significant increase of the value of the measure i.e the significant peak in Hubert
##                 index second differences plot.
##
```

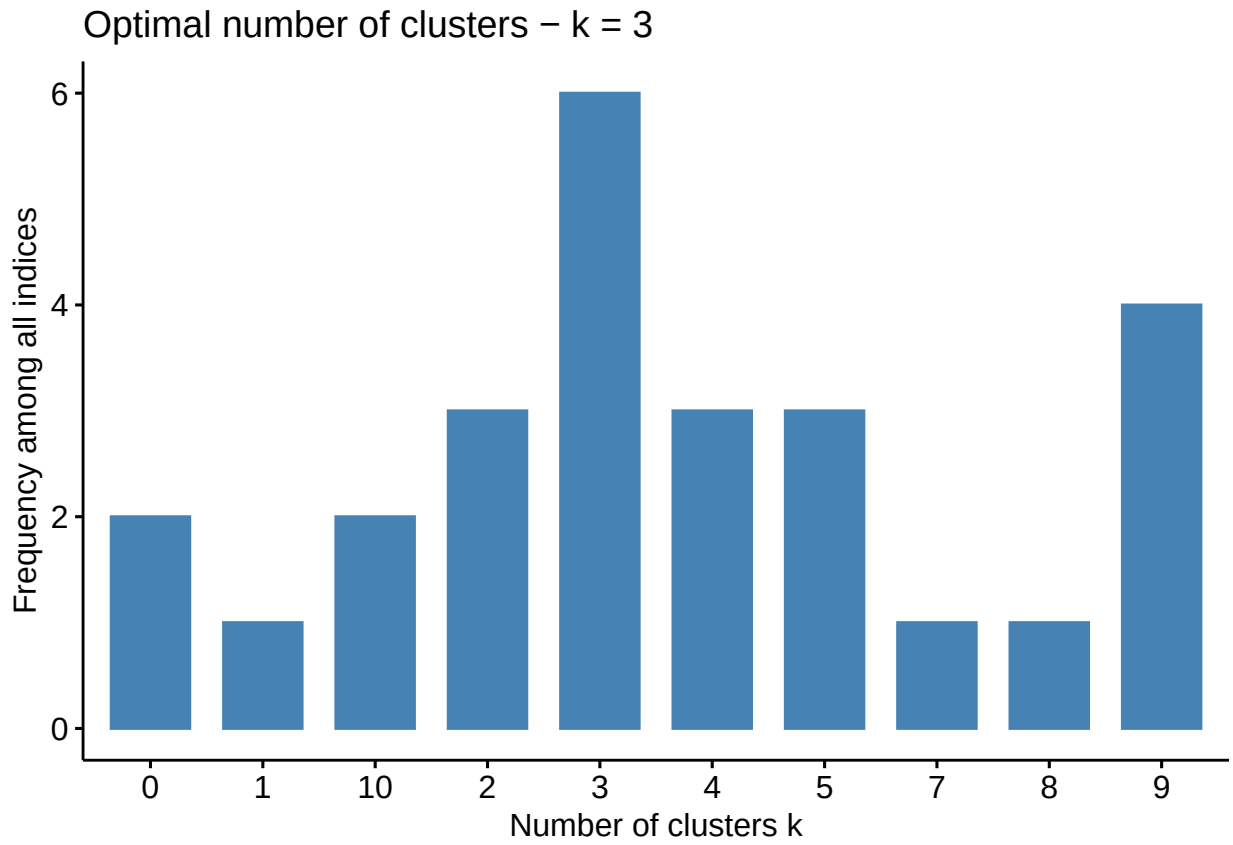



```
## *** : The D index is a graphical method of determining the number of clusters.
##                 In the plot of D index, we seek a significant knee (the significant peak in Dindex
##                 second differences plot) that corresponds to a significant increase of the value of
##                 the measure.
##
## *******************************************************************
## * Among all indices:
## * 3 proposed 2 as the best number of clusters
## * 6 proposed 3 as the best number of clusters
## * 3 proposed 4 as the best number of clusters
## * 3 proposed 5 as the best number of clusters
## * 1 proposed 7 as the best number of clusters
## * 1 proposed 8 as the best number of clusters
## * 4 proposed 9 as the best number of clusters
## * 2 proposed 10 as the best number of clusters
##
##                     ***** Conclusion *****
##
## * According to the majority rule, the best number of clusters is  3
##
##
## *******************************************************************
```

```
fviz_nbclust(res.nbclust, ggtheme = theme_minimal())
```

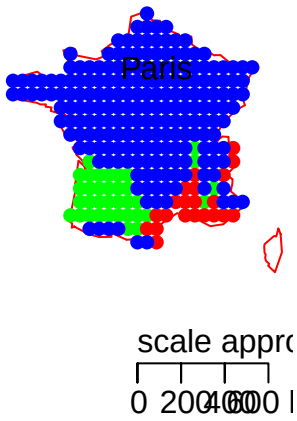
```
## Among all indices:
## ==================
```

```
## * 2 proposed  0 as the best number of clusters
## * 1 proposed  1 as the best number of clusters
## * 3 proposed  2 as the best number of clusters
## * 6 proposed  3 as the best number of clusters
## * 3 proposed  4 as the best number of clusters
## * 3 proposed  5 as the best number of clusters
## * 1 proposed  7 as the best number of clusters
## * 1 proposed  8 as the best number of clusters
## * 4 proposed  9 as the best number of clusters
## * 2 proposed  10 as the best number of clusters
##
## Conclusion
## =========================
## * According to the majority rule, the best number of clusters is  3 .
```




```
tempSpecc <- specc(tempPca_coord_10, centers = 3)

res <- cbind((as.vector(tempSpecc)),as.vector(seq(1:259)))

map("world", "France", col="red", xlim=c(-5,10), ylim=c(35,55))
title("Clustering Instances on a Map Using Spectral Clustering")
map.scale(2,38,metric=T, relwidth=0.3)
points(GPSpos[[1]][res[res[,1] == 1,2]],GPSpos[[2]][res[res[,1] == 1,2]], pch=16,col="green")
points(GPSpos[[1]][res[res[,1] == 2,2]],GPSpos[[2]][res[res[,1] == 2,2]], pch=16,col="red")
points(GPSpos[[1]][res[res[,1] == 3,2]],GPSpos[[2]][res[res[,1] == 3,2]], pch=16,col="blue")
points(2.8, 48.51, pch=16,col="blue")
text(3, 49, label="Paris")
```

Clustering Instances on a Map Using Spectral Clustering



6 Temperature and Wind Clustering

This section presents an approach to cluster wind and temperature data at the same time. To do this, since we have temperature and wind data, it is possible to calculate the wind chill index or felt temperature. Wind chill, sometimes also referred to as the temperature felt in popular parlance, refers to the feeling of cold produced by the wind on an organism that releases heat, while the actual temperature of the ambient air does not drop. The latter is a number without a unit.

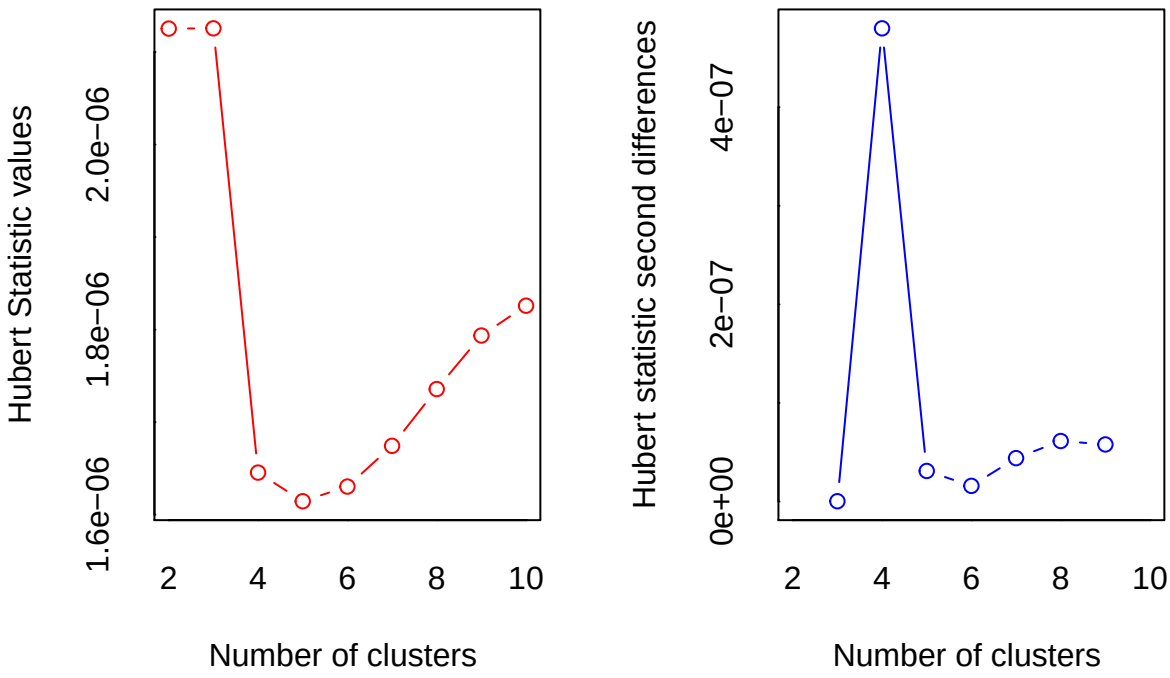
```
tempwinddata <- 13.12 + 0.6215*Temp + (0.3965*Temp - 11.37)*(Wind^0.16)
```

```
# Data Dimension : High Dimensional Data
dim(tempwinddata)
```

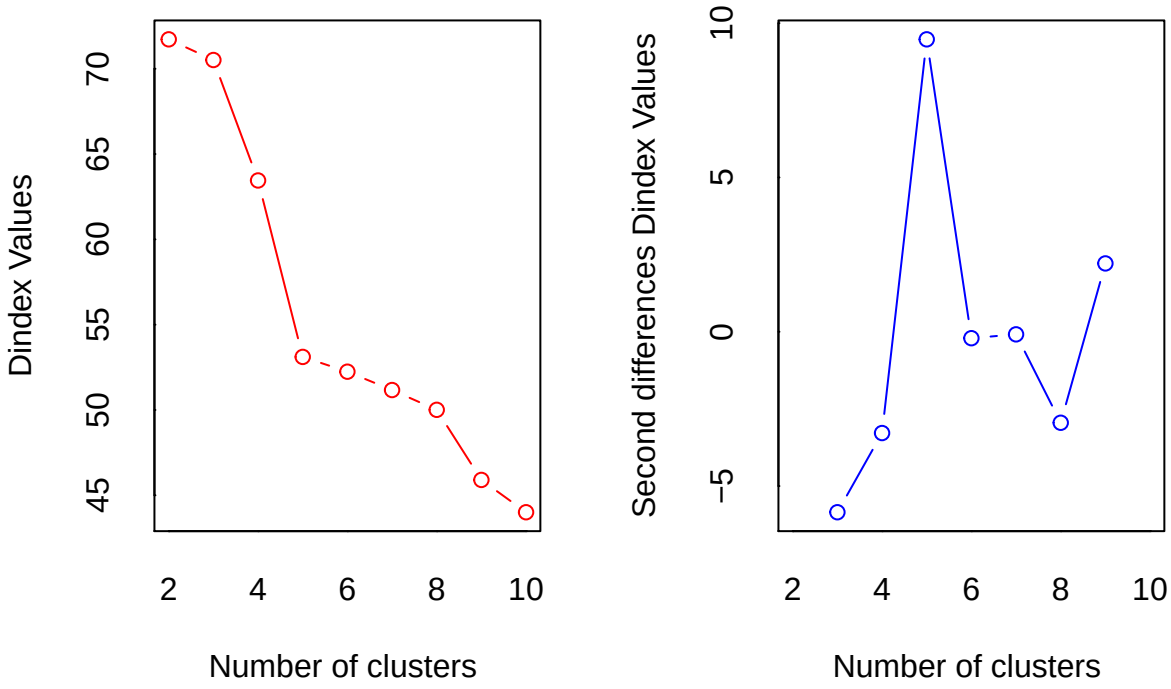
```
## [1]  259 8760
```

```
tempwindPca_10 <- PCA(tempwinddata, ncp = 10, graph = FALSE)
tempwindPca_coord_10 <- as.matrix(tempwindPca_10$ind$coord)
```

```
# Let's determine the number of clusters
res.nbclust <- tempwindPca_coord_10 %>%
  NbClust(distance = "euclidean",
          min.nc = 2, max.nc = 10,
          method = "complete", index ="all")
```




```
## *** : The Hubert index is a graphical method of determining the number of clusters.
##                 In the plot of Hubert index, we seek a significant knee that corresponds to a
##                 significant increase of the value of the measure i.e the significant peak in Hubert
##                 index second differences plot.
##
```

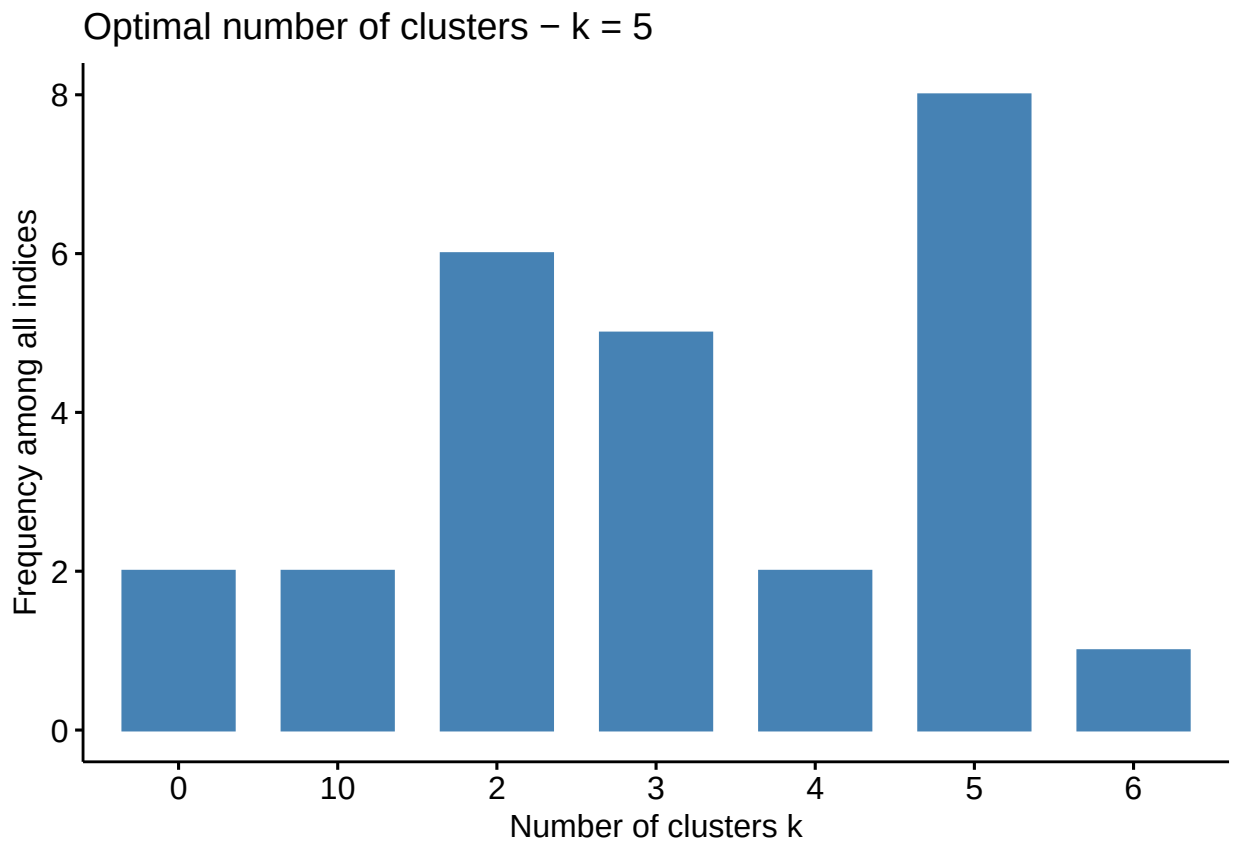



```
## *** : The D index is a graphical method of determining the number of clusters.
##                 In the plot of D index, we seek a significant knee (the significant peak in Dindex
##                 second differences plot) that corresponds to a significant increase of the value of
##                 the measure.
##
## *******************************************************************
## * Among all indices:
## * 6 proposed 2 as the best number of clusters
## * 5 proposed 3 as the best number of clusters
## * 2 proposed 4 as the best number of clusters
## * 8 proposed 5 as the best number of clusters
## * 1 proposed 6 as the best number of clusters
## * 2 proposed 10 as the best number of clusters
##
##                     ***** Conclusion *****
##
## * According to the majority rule, the best number of clusters is  5
##
##
## *******************************************************************
```

```
fviz_nbclust(res.nbclust, ggtheme = theme_minimal())
```

```
## Among all indices:
## ==================
## * 2 proposed  0 as the best number of clusters
## * 6 proposed  2 as the best number of clusters
```

```
## * 5 proposed  3 as the best number of clusters
## * 2 proposed  4 as the best number of clusters
## * 8 proposed  5 as the best number of clusters
## * 1 proposed  6 as the best number of clusters
## * 2 proposed  10 as the best number of clusters
##
## Conclusion
## ========================
## * According to the majority rule, the best number of clusters is  5 .
```




```
set.seed(1234)
tempwindkmeans <- kmeans(tempwindPca_coord_10, centers = 5, nstart = 25)

table(tempwindkmeans$cluster)

##
##   1   2   3   4   5
## 101  53   3  27  75

res <- cbind((as.vector(tempwindkmeans$cluster)),as.vector(seq(1:259)))

map("world", "France", col="red", xlim=  c(-5,10), ylim = c(35,55))
title("Clustering Instances on a Map Using Kmeans")
map.scale(2, 38, metric = T, relwidth = 0.3)
points(GPSpos[[1]][res[res[,1] == 1,2]], GPSpos[[2]][res[res[,1] == 1,2]], pch=16, col = "green")
points(GPSpos[[1]][res[res[,1] == 2,2]], GPSpos[[2]][res[res[,1] == 2,2]], pch=16, col = "red")
points(GPSpos[[1]][res[res[,1] == 3,2]], GPSpos[[2]][res[res[,1] == 3,2]], pch=16, col = "blue")
points(GPSpos[[1]][res[res[,1] == 4,2]], GPSpos[[2]][res[res[,1] == 4,2]], pch=16, col = "yellow")
```

```
points(GPSpos[[1]][res[res[,1] == 5,2]], GPSpos[[2]][res[res[,1] == 5,2]], pch=16, col = "brown")
points(2.8, 48.51, pch=16, col = "blue")
text(3, 49, label = "Paris")
```

Clustering Instances on a Map Using Kmeans