Machine Learning Project 2 : Segmentation of the French territory in homogeneous climate region using clustering algorithm

Thomas Marcoux PÃ©pin and OdÃ©lia Guedj

Loading the data

The "weatherdata.Rdata" data set provides temperature and wind temporal evolution for $n = 259$ gridpoints at an hourly sampling rate for a given year ($p = 8760$ hours).

`temp` denotes the time series for the temperature and `wind` denotes the time series for the wind. The GPSpos variable contains the GPS positions (longitude and latitude) of the time series grid points.

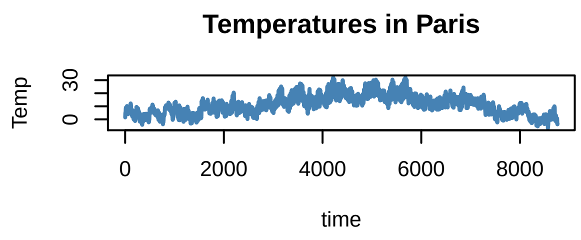
The goal of this project is to find a good segmentation of the french territory based on the temperatures and the wind time series.

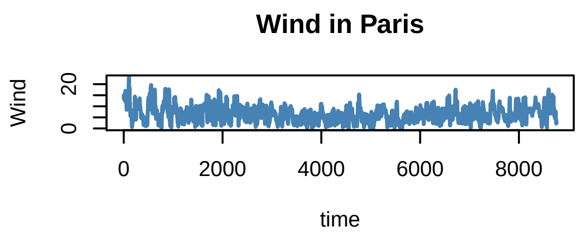
1. Preliminary

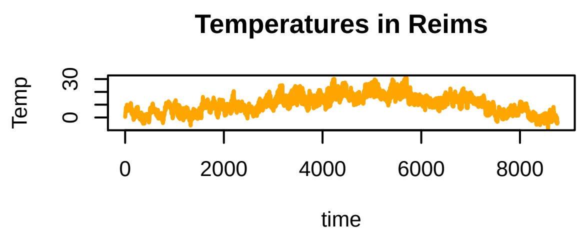
We chose 3 cities in France: Paris, Reims and Lyon, noted their GPS position, and affected them to the closest observations the name of these cities in the dataframe `Temp` and `Wind`.

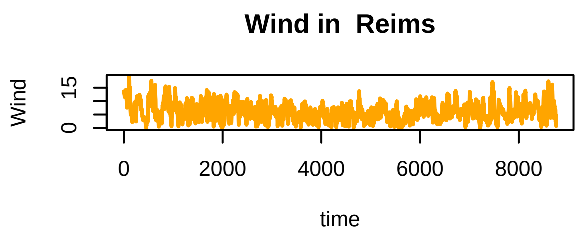
For every chosen city, we compute the distance to the 259 gridpoints of the map. The closest point is considered as the chosen city.

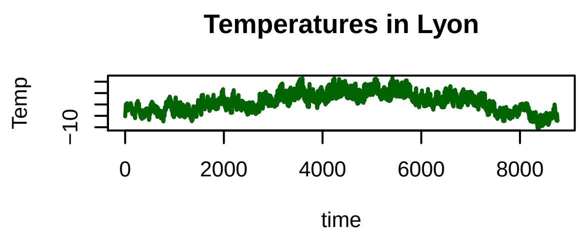
Let's display the time series of temperature and wind for the 3 chosen cities:

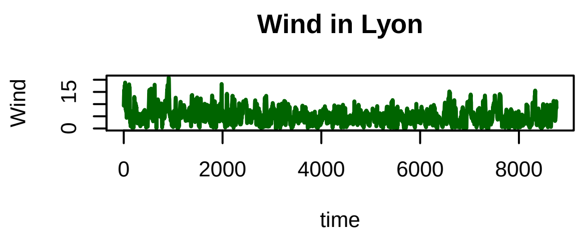






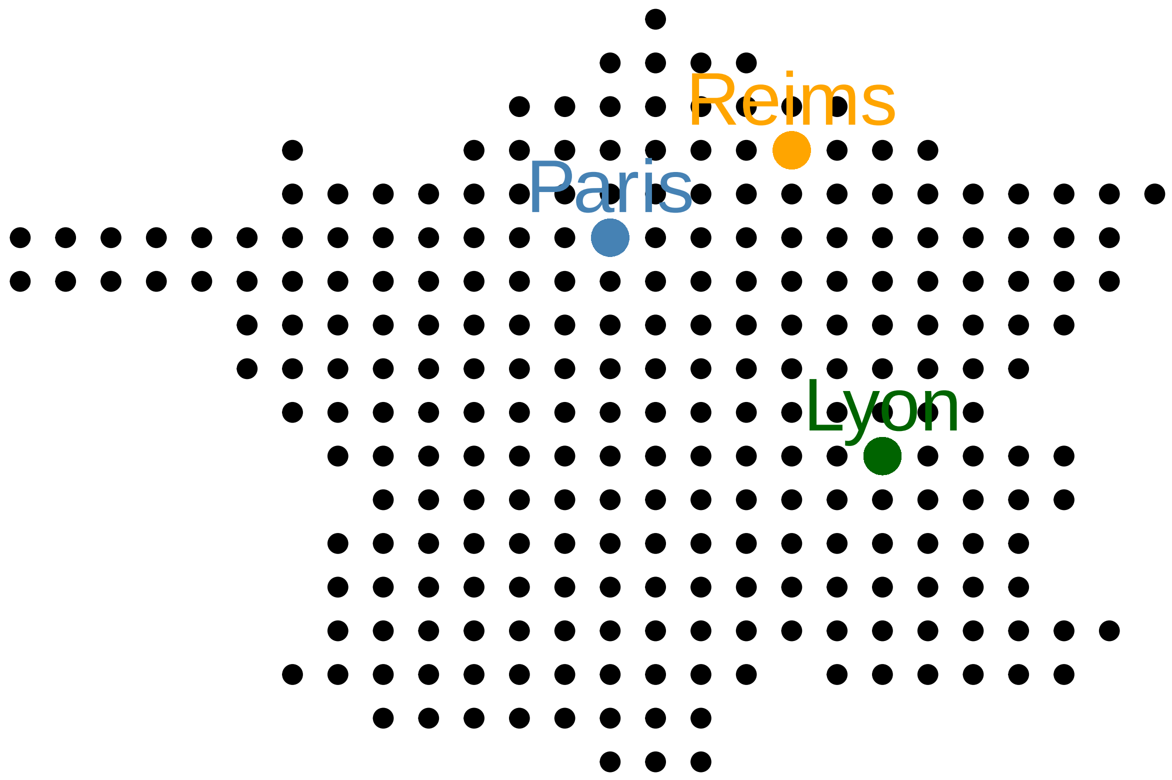






Even if the chosen cities are not that close, the evolution of `wind` and `temperature` through time seems to be quite the same. We can also display the map of France with the 259 gridpoints and particularly, the 3 chosen cities.

The 259 pointgrids of the French map with our 3 chosen cities



2. Wind Clustering

This section aims to cluster wind data.

2.1 Raw Data

First we work on the raw data which is the hourly meseaurs of the wind (for 8760 hours). We perform a clustering of the 259 points with differents methods:

- First the k-means algorithm. It's a very common algorithm for clustering data. The idea behind it is very simple. First, we have to choose a number of clusters. Choosing this number is usually is a difficult part but in our case the number of cluster k is fixed to 4. At the initialization state, we randomly choose the k centers of our k clusters. Then we attribute a cluster to each observation by minimizing a certain criterion. When we are working on the raw data we could use the distance between the observation and the clusters: so that we include an observation in his closest cluster.
- Second, the Hierarchical Clustering algorithm: it's quite different of the k-means algorithm because it provides a structured clustering. First they are as many clusters than observations. Then we aggregate the two closest clusters basing on an aggregation rule (to be defined). We repeat this step until a certain criterion is satisfied. The clusters will be different if we use different aggregate rules. The `hclust` function of R provides us 8 different methods of aggregation. We will try some of them and comment the results.

K-means

We run the k-means algorithm with 4 clusters. We choose to perform the algorithm 50 times (`nstart` = 50), for more stability in the results (since the centers are randomly chosen at the initialisation step).

We'll show this clustering on a map at the end of the next part.

Hierarchical Clustering

Overview of three aggregation methods

We chose to perform 3 differents methods of Hierarchical Clustering which are used the most: the ward criterion, the complete criterion and the single criterion.

- The single method proposed by the `hclust` function refers to the single-link criterion: the distance between 2 clusters is the minimum of the distances between all pairs of elements picked in two clusters.
- The complete method refers to the complete-link criterion: the distance between 2 clusters is the maximum of the distances between all pairs of elements picked in two clusters.
- The ward.D criterion consist in grouping clusters such as the raise of within inertia is maximum.

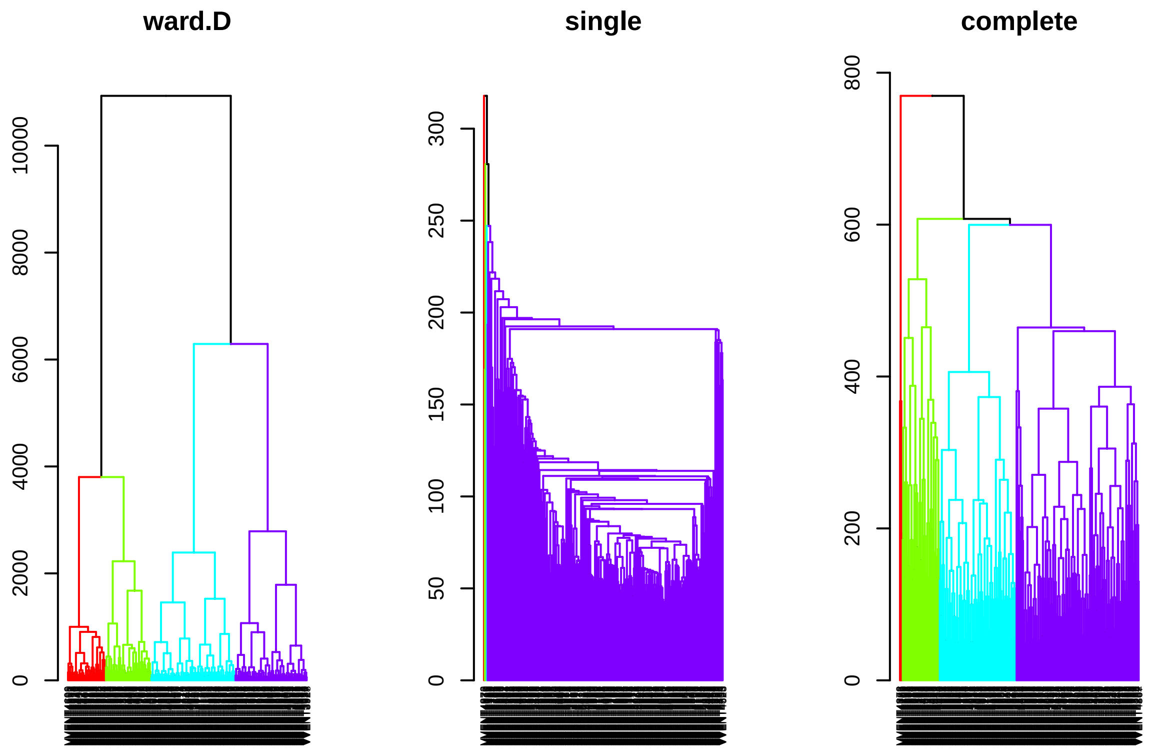
Computing the distance matrix of the wind raw data using the euclidian method:

Let display the dendrograms obtained by the different methods:

```
##
## ---------------------
## Welcome to dendextend version 1.13.2
## Type citation('dendextend') for how to cite the package.
##
## Type browseVignettes(package = 'dendextend') for the package vignette.
## The github page is: https://github.com/talgalili/dendextend/
##
## Suggestions and bug-reports can be submitted at: https://github.com/talgalili/dendextend/issues
## Or contact: <tal.galili@gmail.com>
##
##  To suppress this message use:  suppressPackageStartupMessages(library(dendextend))
## ---------------------
```

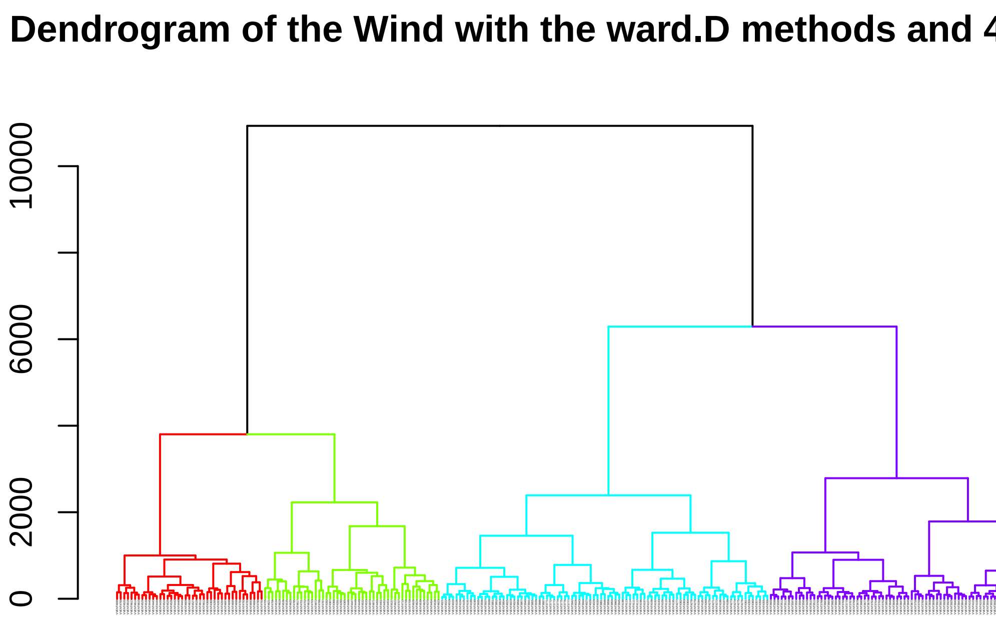
```
##
## Attaching package: 'dendextend'
```

```
## The following object is masked from 'package:stats':
##
##     cutree
```



With the ward.D method

The ward.D methods is described as the most able to give homegenous clusters. The 3 dendrograms above confirm this hypothesis.



Displaying the final dendrogram

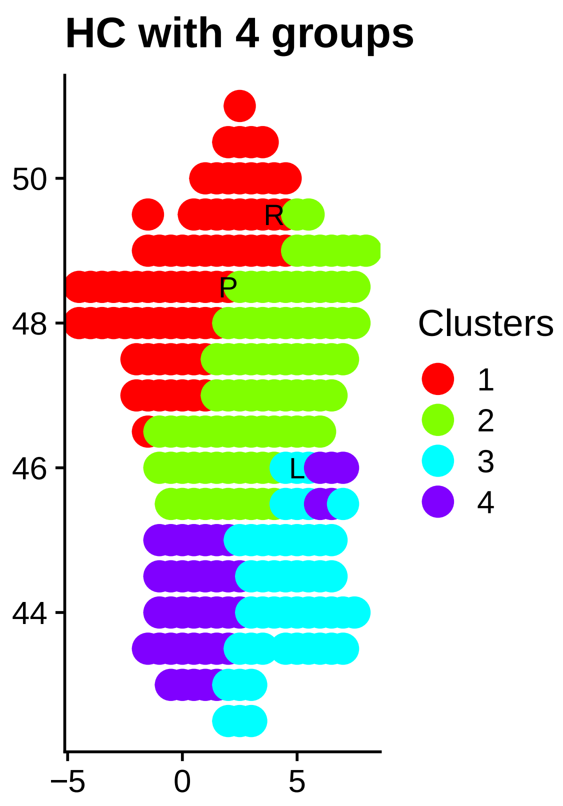
kmeans and HC : Comparison of the clusters

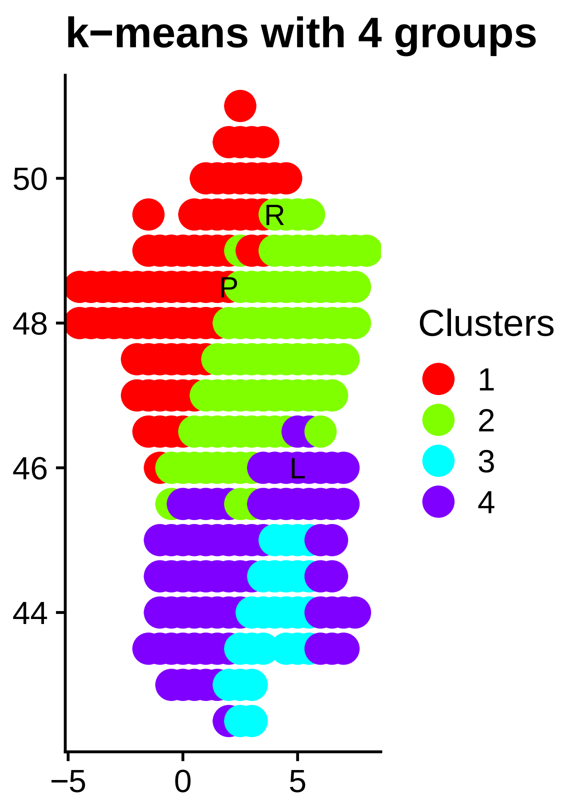
We can display the clustering performed by the k-means algorithm and the HC algorithm (k = 4) on an actual map. Since the attribution of a number (1,2,3,4) for each cluster is random, we have to be sure that the same numbers are chosen for the same clusters in each clustering algorithm. That is to say we can compare the clusters more or less a permutation.

Here in our example, the red cluster on the HC algorithm corresponds to the green cluster on the k-means. Let's fix it.

It could be interesting to check if our three cities are on the same clusters for the two methods performed.

```
##
## ********************************************************

## Note: As of version 1.0.0, cowplot does not change the

##   default ggplot2 theme anymore. To recover the previous

##   behavior, execute:
##   theme_set(theme_cowplot())

## ********************************************************
```





The 4 clusters chosen by our algorithms are quite the same. Moreover, we can deduce French weather information from those choices. We can clearly see that cluster 3 is associated with mediterranean climate while cluster 1 is associated with the North part of France, cluster 2 is part of France under the continental climate and cluster 4 is associated with the South part of France.

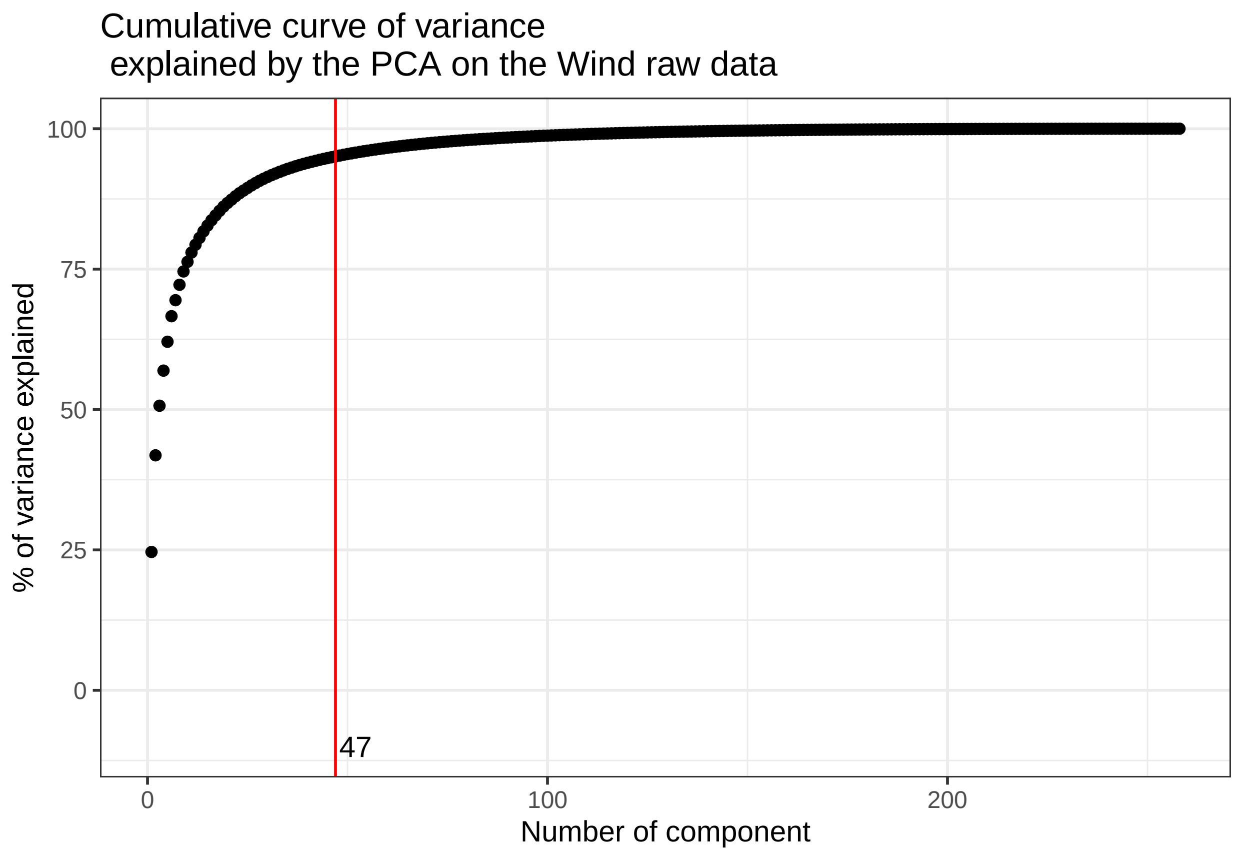
The algorithms clusters differ on certain points of the data. Yet, the two clustering results are both interpretable.

2.2 Principal Component Analysis

We want to extract the most important features in order to perform the same clustering algorithms than above on a smallest dataset. The question is: are we able to get the same results with less variables ?

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```
res.pca.wind = PCA(Wind, graph = F)
```



In order to get 95 % of the variance explained we have to keep the first 47 principal components. So our new dataset has p = 47 features.

2.3 Clustering on the PCA

```
print(eig.val.wind[1:10,3])
```

```
##    Dim.1    Dim.2    Dim.3    Dim.4    Dim.5    Dim.6    Dim.7    Dim.8
## 24.62957 41.84011 50.66891 56.92455 62.06401 66.61574 69.46286 72.20729
##    Dim.9   Dim.10
## 74.58062 76.29390
```

If we keep 10 components we'd get 76.29% of explained variance.

Hierarchical Clustering

Since we used the PCA function from the FactoMineR package, it's easier to also use a fonction from this package to performe the Hierarchical Clustering : the HCPC function.

```
wind.hc.wardD.pca = HCPC(PCA(Wind, ncp = 10, graph = F, scale.unit = F), nb.clust = 4, method = "ward",
wind.hc.wardD.pca =  wind.hc.wardD.pca$data.clust[,8761]
```

Again, we have to make sure that the figures used for the clusters are the same for the HC algorithm on the PCA.

```
table(PCA = GPSpos$wind.hc.wardD.pca, "raw data" = GPSpos$wind.hc.wardD)
```

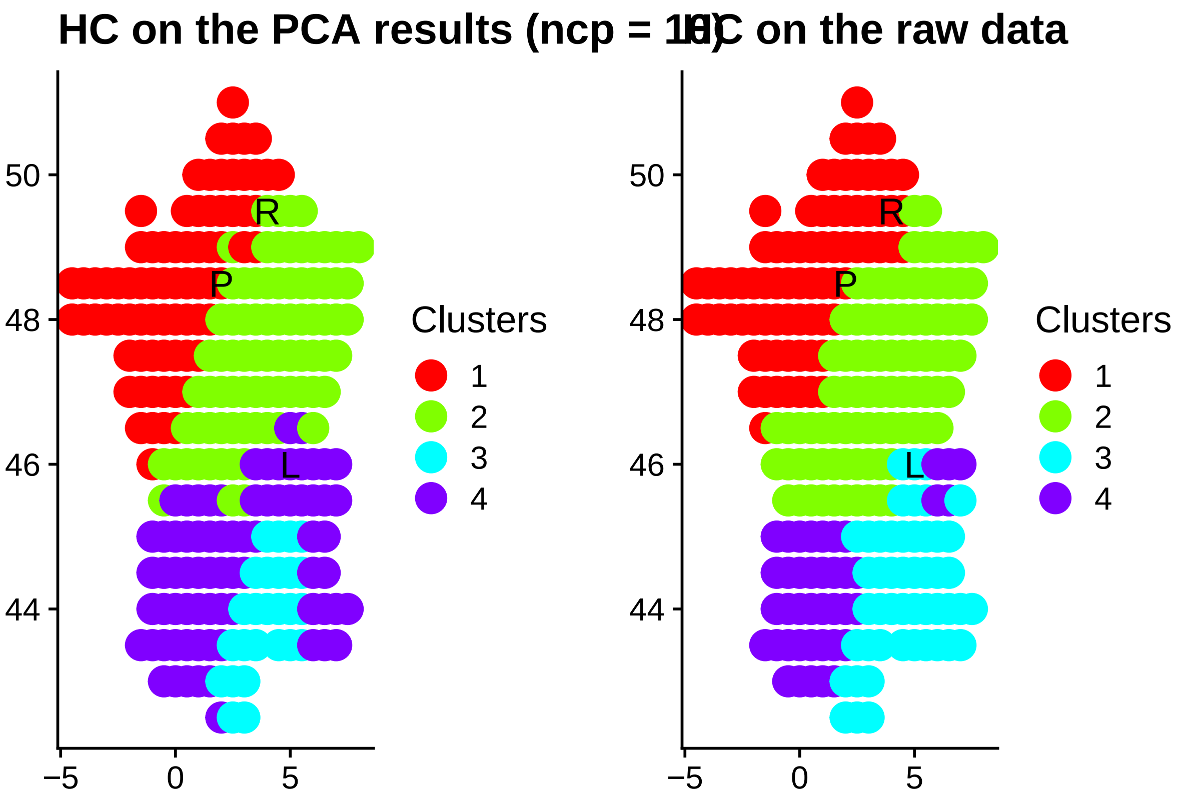
```
##    raw data
## PCA  1  2  3  4
##   1 72  4  0  0
##   2  6 76  0  0
##   3  0  0 26  0
##   4  0 11 23 41
```

Since we only kept the first 10 components, the results of the two clusterings (on the raw data and on the results of the PCA) can not be the same. The interesting question is to know how “wrong”" is the clustering basing on the results of the PCA ? In order to know that, we sum up the number of mistakes (meaning the number of differences between the 2 clusterings) and we divide it by 259 which is the total number of raws in our initial data set.

```
(length(which(GPSpos$wind.hc.wardD.pca != GPSpos$wind.hc.wardD))/ 229)*100
```

```
## [1] 19.21397
```

So, we observe an “error” of 19.2%. But this “error” is calculated on the assumption that the HC performed on the raw data is more close to the “truth” (that is to say the right segmentation of France) that the HC performed in the raw data, so we should analyse this number carrefully. The only thing we can say about it is : with significantly less signal (10 components over 259) we get around 80% of the results obtained with the entire information.



There we observe the same results than with the previous method. The clusters are quite the same. * Cluster 1: North and West part of France * Cluster 2: Center and East part of France * Cluster 3: Mediterranean climate * Cluster 4: Southern part of France where the climate might not be mediterranean

K-means

2.4 Conclusion

In this section, for our segmentation of the French territory, we used 2 algorithms. First on the raw data and second on the 10 first components of a Principal Composant Analysis. Our 4 segmentation gives globally the same results. Since we chose to make 4 groups, it seems that they correspond to 4 zones with very different climates: North-West with oceanic climate, North-East with continental climate, South-West with oceanic climate too but hotter than North-West region and South-East for the medditeranean climate. The South-East and the South-West clusters are the ones with the most variations. Before making any hypothesis about this results, we have to check if they are the same when we base our study on the temperature data instead of the wind data.

3. Temparature clustering

This section aims to cluster temperature data.

3.1 Raw data

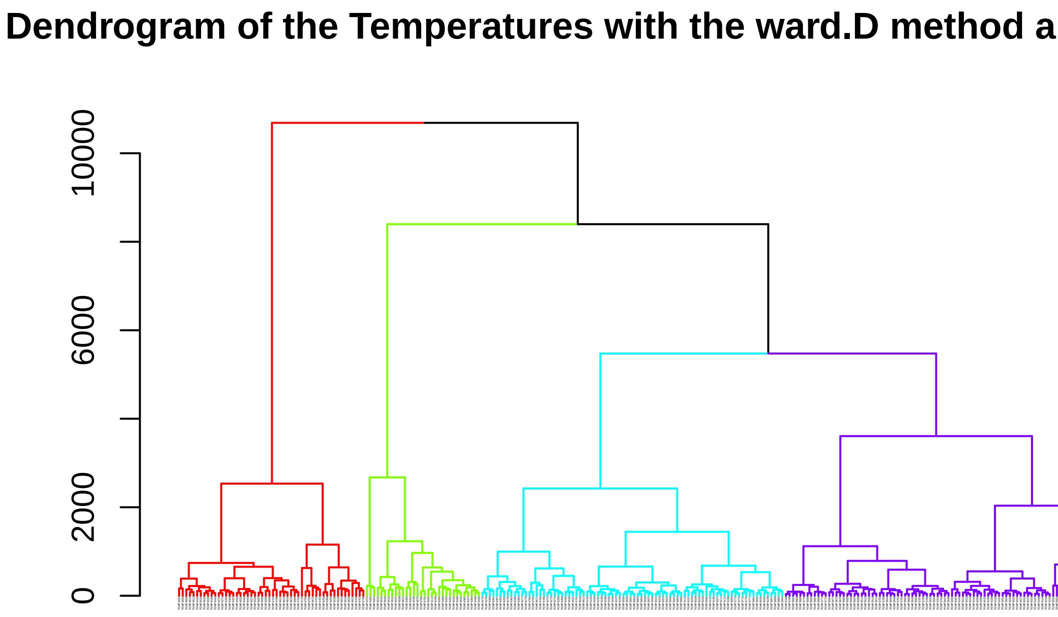
In this part we study and compare the use of k-means and Hierarchical Clustering to provide a segmentation into 4 groups of the temperature using the raw time series.

K-means

```r
set.seed(3)
temp.kmeans = kmeans(x = Temp, centers = 4, nstart = 50)
temp.kmeans = as.factor(temp.kmeans$cluster)
```

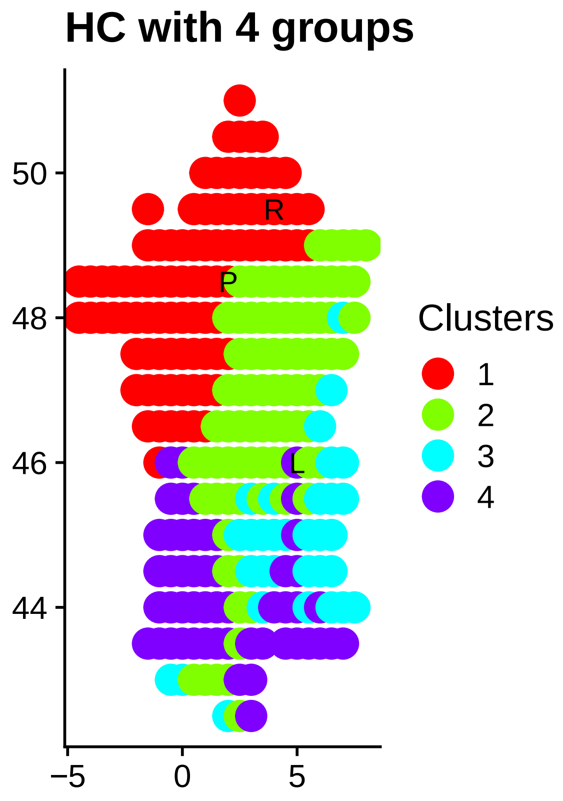
Hierarchical clustering with the ward.D method

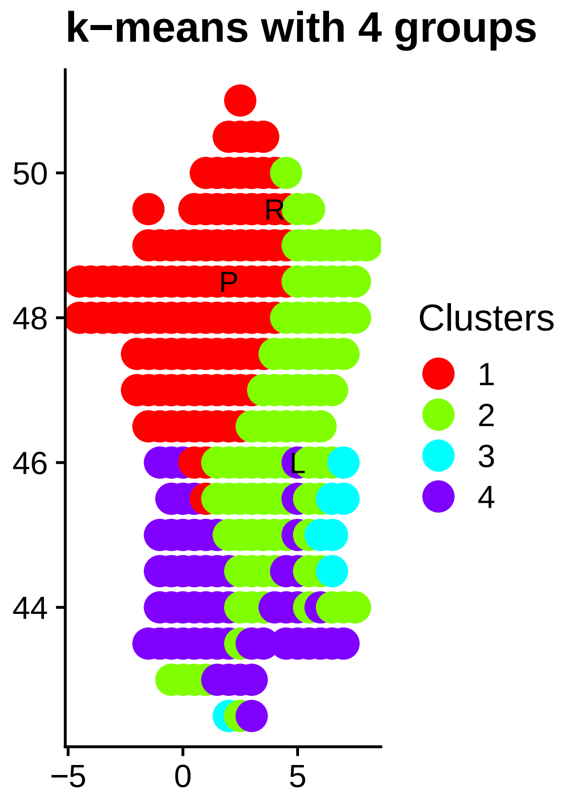
Computing the distance matrix of the temperature raw data using the euclidian method:



Displaying the final dendrogram

kmeans and HC : Comparison of the clusters

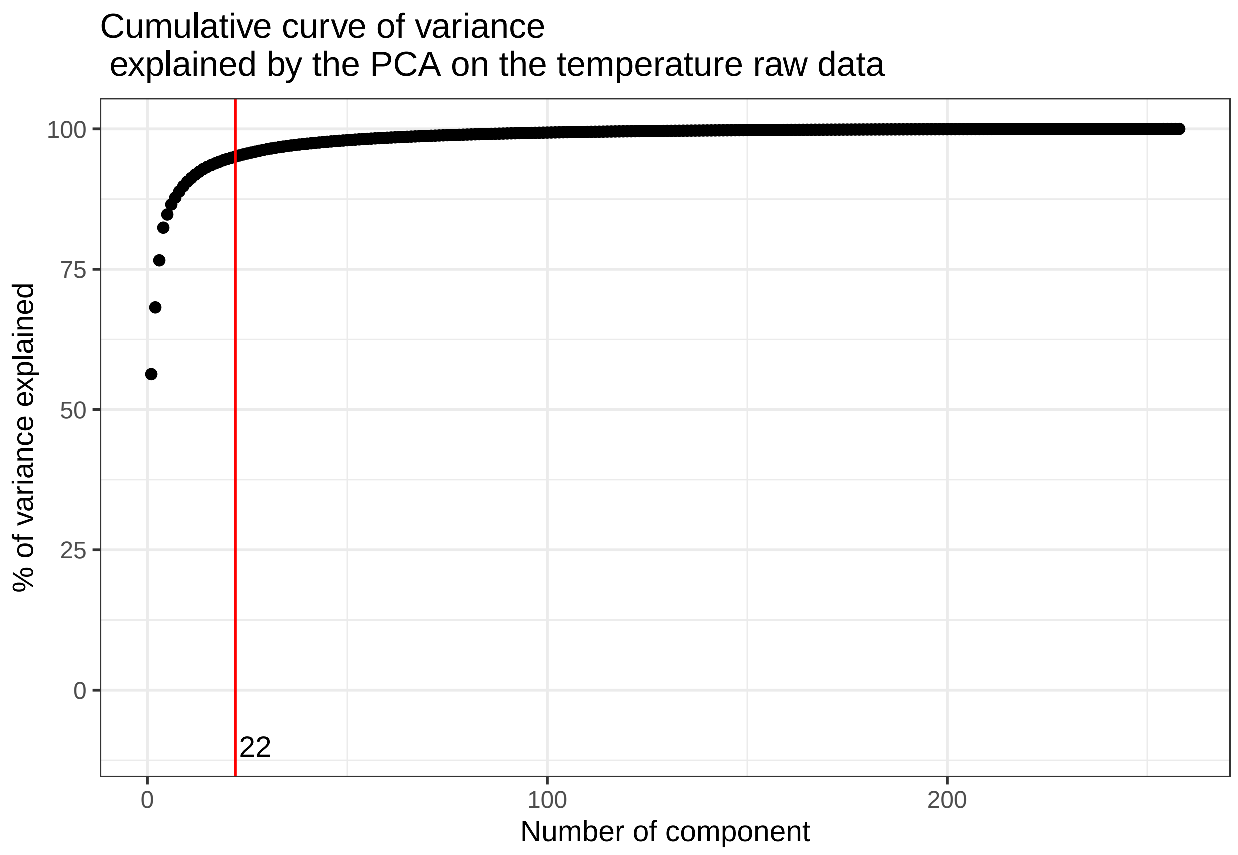




First, its important to notice that, by clustering with the temperature, we don't get the same results than clustering with wind. It seems like clustering this way, the algorithms were able to notice small details in the France landscape. For example, if we look at cluster 4, we can see that it found a Lyon-Marseille line, which is contained between two moutainous areas. Those areas are indeed contained in the cluster 3 chosen by HC algorithm. Moreover, we can notice that, the cities we choose, Paris and Reims, are now assiocated with the same cluster while they were not before.

2.2 Principal Compenent Analysis

We want to extract the most important features in order to perform the same clustering algorithms than above on a smallest dataset.



In order to get 95% of the variance explained we have to keep te first 22 principal components.

```
print(eig.val.temp[1:10,3])
```

```
##    Dim.1    Dim.2    Dim.3    Dim.4    Dim.5    Dim.6    Dim.7    Dim.8
## 56.29860 68.19479 76.57533 82.40447 84.74606 86.52574 87.75564 88.85403
##    Dim.9   Dim.10
## 89.78512 90.58647
```

2.3 Clustering on the PCA

If we kept 10 components we'd get 90.59% of explained variance.

Hierarchical Clustering

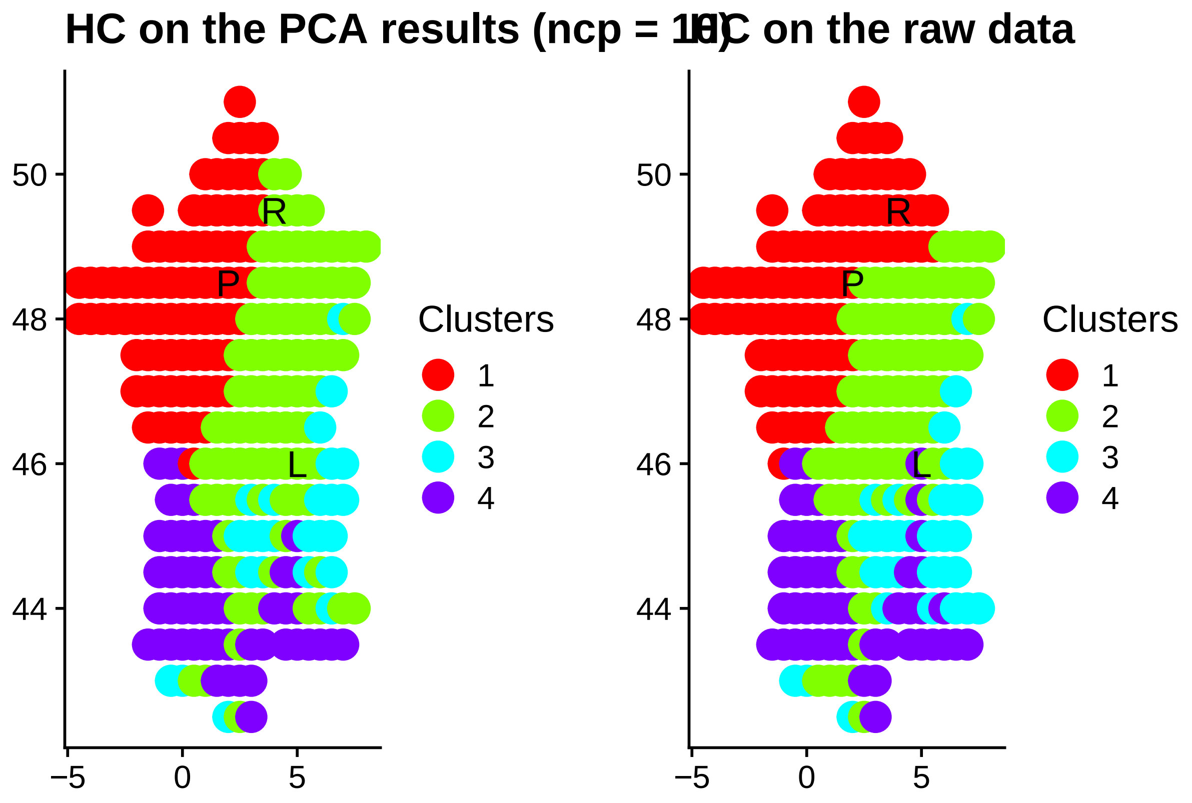
Since we used the PCA function from the FactoMineR package, it's easier to also use a fonction from this package to performe the Hierarchical Clustering : the HCPC function.

Again, we have to make sure that the figures used for the clusters are the "same" for the HC algorithm on the PCA.

```
table(PCA = GPSpos$temp.hc.wardD.pca.bis, "raw data" = GPSpos$temp.hc.wardD)
```

```
##    raw data
## PCA  1  2  3  4
##   1 79  6  0  0
##   2 11 76  7  3
##    3  0   0 25   0
##    4  1   2   0 49
```

Since we only kept the first 10 components, the results of the two clusterings (on the raw data and on the results of the PCA) can not be the same.



Even though we reduced the dimension of the data set through PCA, we still get a lot of information with HC. Our cities are not in the same clusters on the two maps but we still get most of the information, and the clusters are still the same: * Cluster 1: North-West part of France, with oceanic climate * Cluster 2: East-Central part of France, with continental climate * Cluster 3: Moutainous area of France (Massif central, Alpes, Pyrénnées) * Cluster 4: Southest part of France, mostly close to the coasts

2.4 Clustering using model based

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

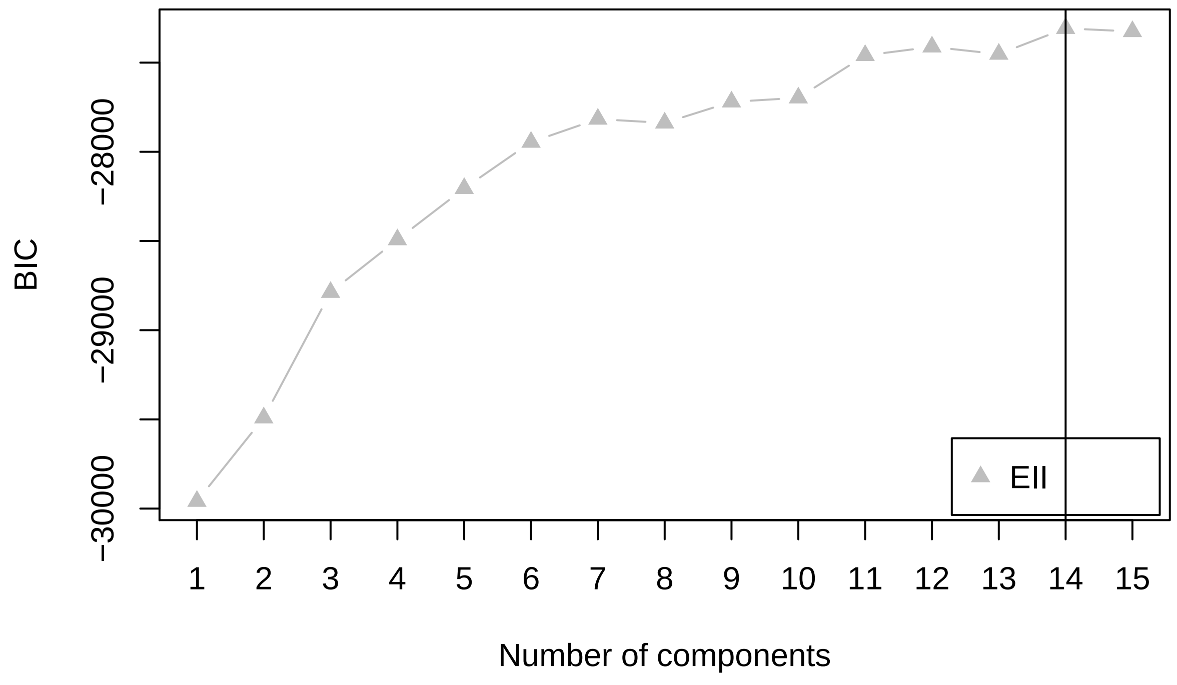
In this section we intend to compute a segmentation of the temperature time series , based on the PCA representation keeping only 10 principal components using a model based clustering method.

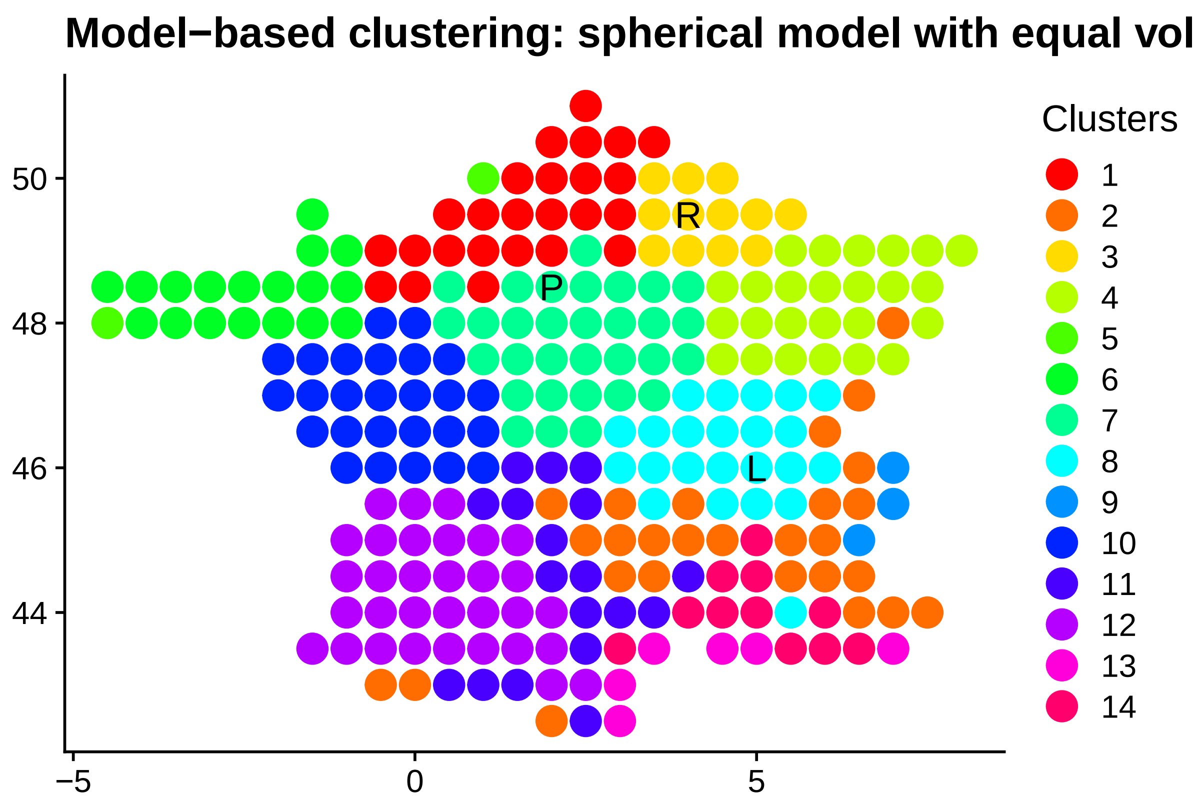
Mclust function from the mclust package compute model-based clustering based on parameterized finite Gaussian mixture models. This function allows us to make several hypothesis on the variance covariance matrix of the data.

In this work, 4 assumptions will be studied: * spherical model with equal volume : “EII” * spherical model with unequal volume : “VII” * diagonal, varying volume and shape : “VVI” * ellipsoidal, varying volume, shape, and orientation : “VVV”

Assumption 1: Spherical model with equal volume

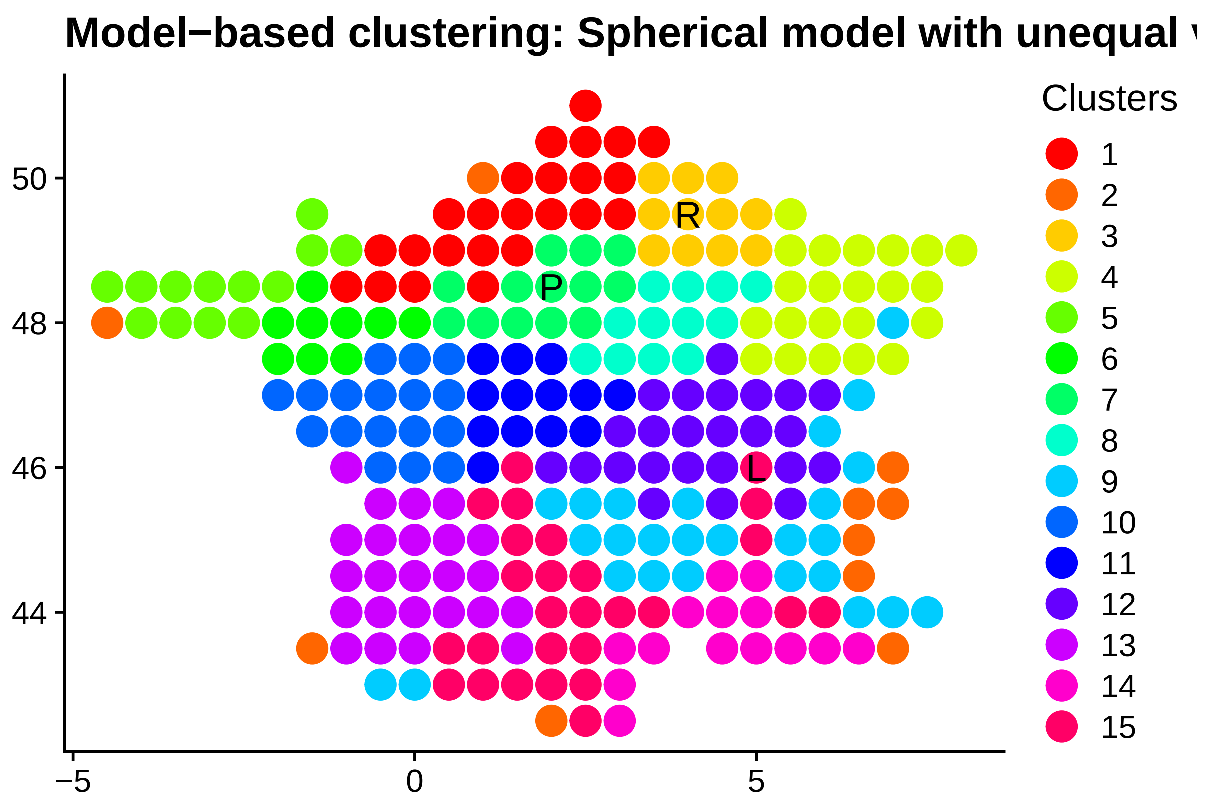
```
mclust_EII = Mclust(pca_temp, modelNames="EII", G=1:15)
```





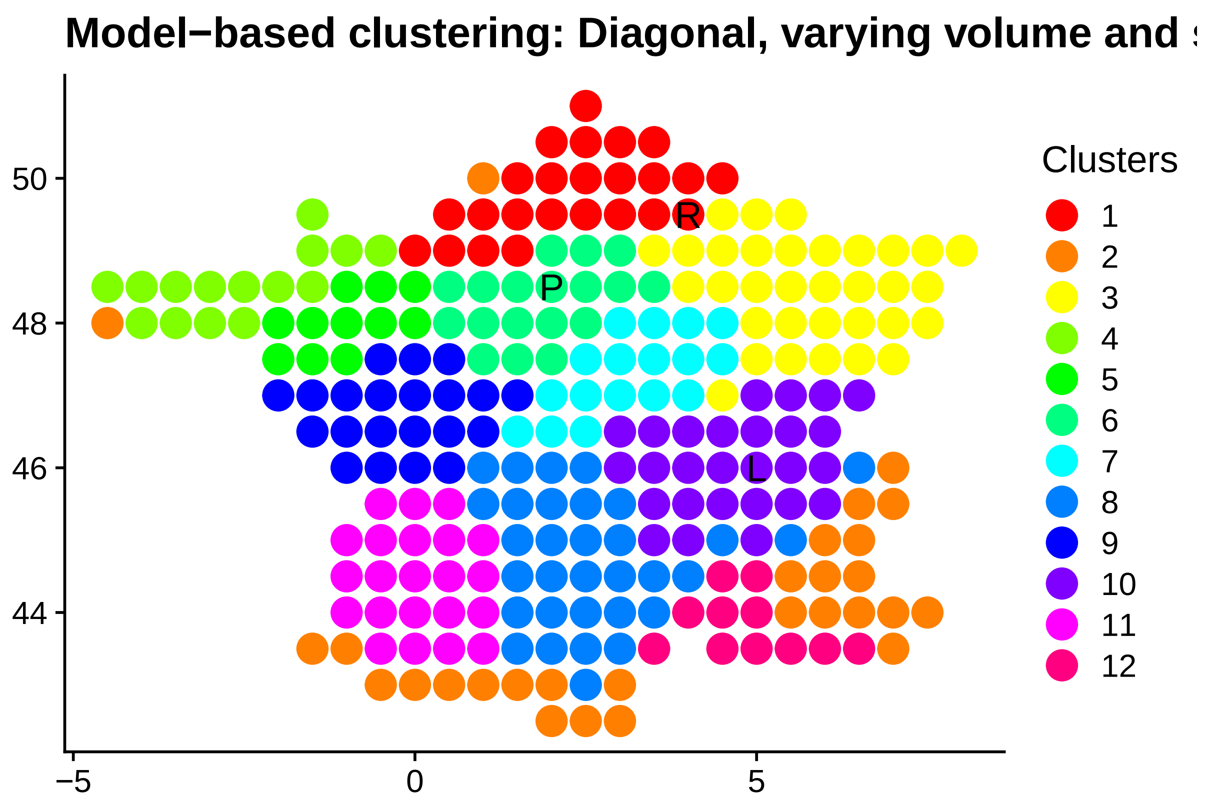
Assumption 2: Spherical model with unequal volume

```
mclust_VII = Mclust(pca_temp, modelNames="VII", G=1:15)
table(mclust_VII$classification)
GPSpos$mclust.VII = as.factor(mclust_VII$classification)
```



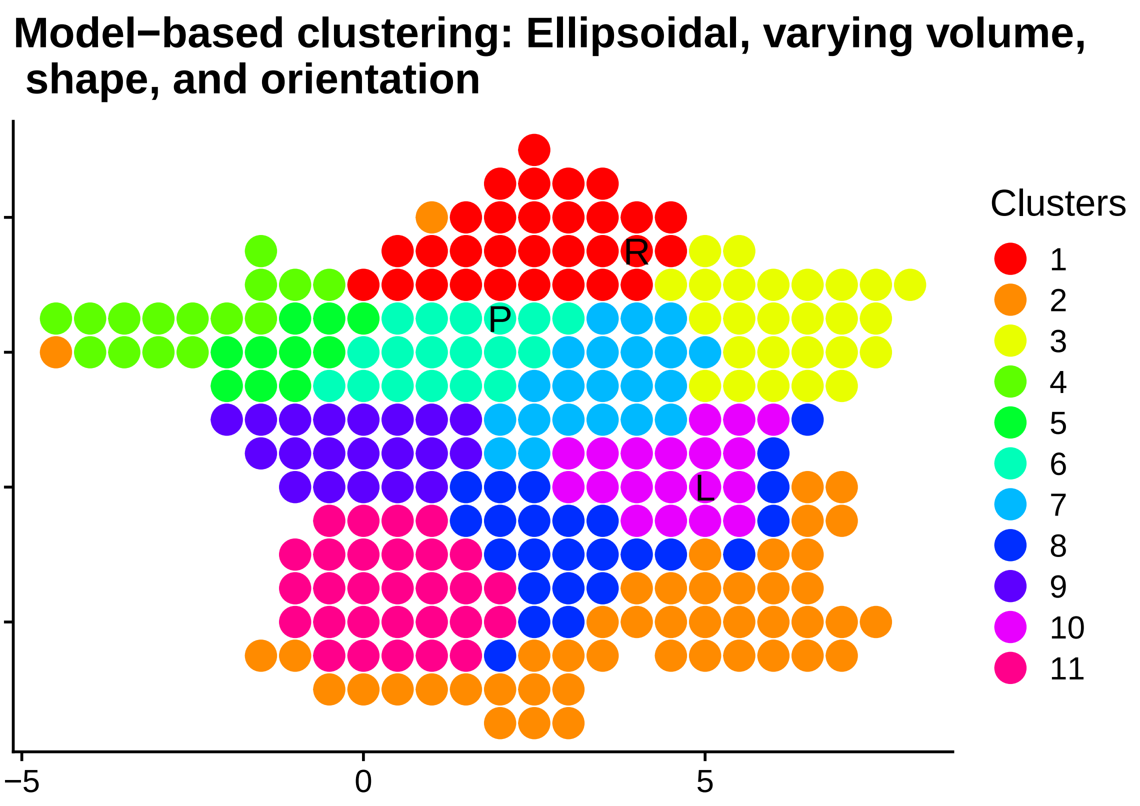
Assumption 3: Diagonal, varying volume and shape

```
mclust_VVI = Mclust(pca_temp, modelNames="VVI", G=1:15)
table(mclust_VVI$classification)
GPSpos$mclust.VVI = as.factor(mclust_VVI$classification)
```



Assumption 4: Our own :

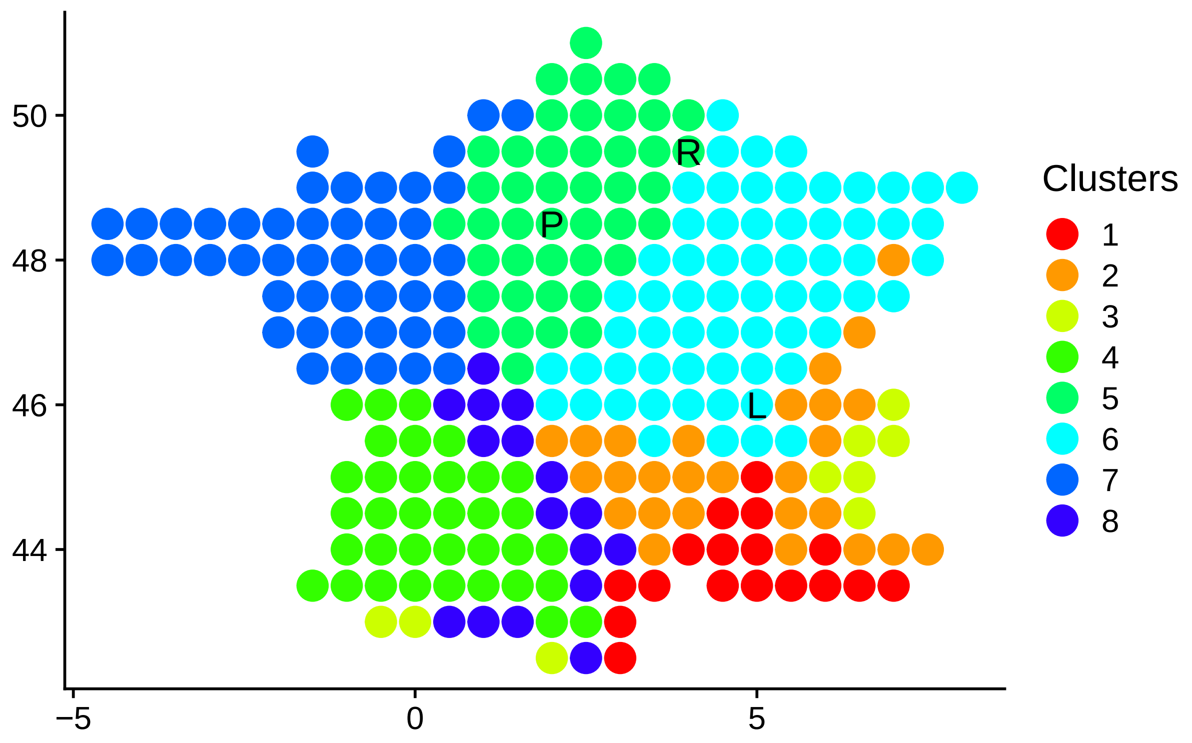
```
mclust_VVV = Mclust(pca_temp, modelNames="VEE", G=1:15)
table(mclust_VVI$classification)
GPSpos$mclust.VVV = as.factor(mclust_VVV$classification)
```



4. Clustering using spectral clustering

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

We tried to chose the adequate number of clusters to our data with the within-cluster sum of squares for each cluster (`withinss`), but it keeps decreasing as the number of clusters chosen. We'd need an other argument to make such a choice. So we choose a number of cluster that make decrease `withinss` enough without having too much clusters.



5. Wind and Temp variables at the same time

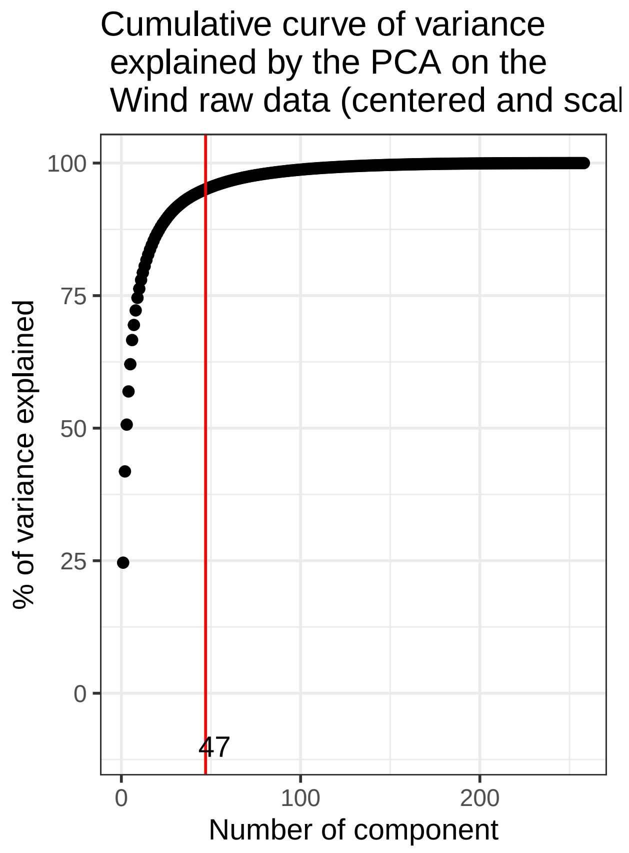
Our 3 cities

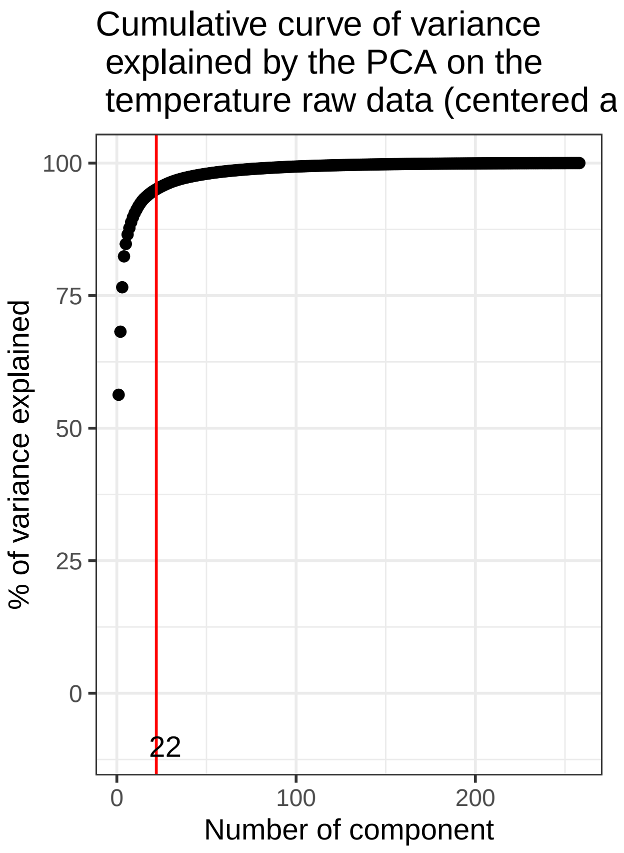
```
ParisLat = 48.51
ParisLong = 2.20

ReimsLat = 49.25
ReimsLong = 4.03

LyonLat = 45.75
LyonLong = 4.9
```

```
distToParis = (GPSpos$Lon-ParisLong)^2 + (GPSpos$Lat-ParisLat)^2
distToReims = (GPSpos$Lon-ReimsLong)^2 + (GPSpos$Lat-ReimsLat)^2
distToLyon = (GPSpos$Lon-LyonLong)^2 + (GPSpos$Lat-LyonLat)^2

paris = which.min(distToParis)
reims = which.min(distToReims)
lyon = which.min(distToLyon)
```

First approach: separated PCA

Our first approach in the segmentation of the french territory with both variables was to perform two PCA, one on each of the variable separately. Then we apply k-means and HC to the juncture of all principal components we kept from our PCAs.

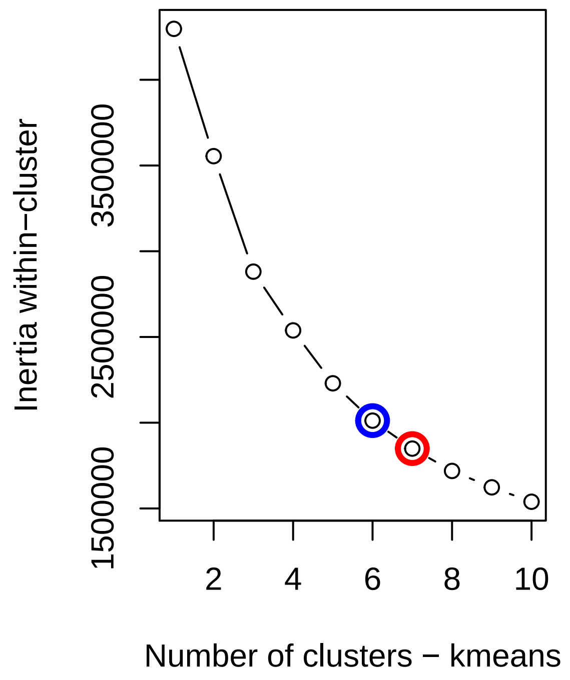


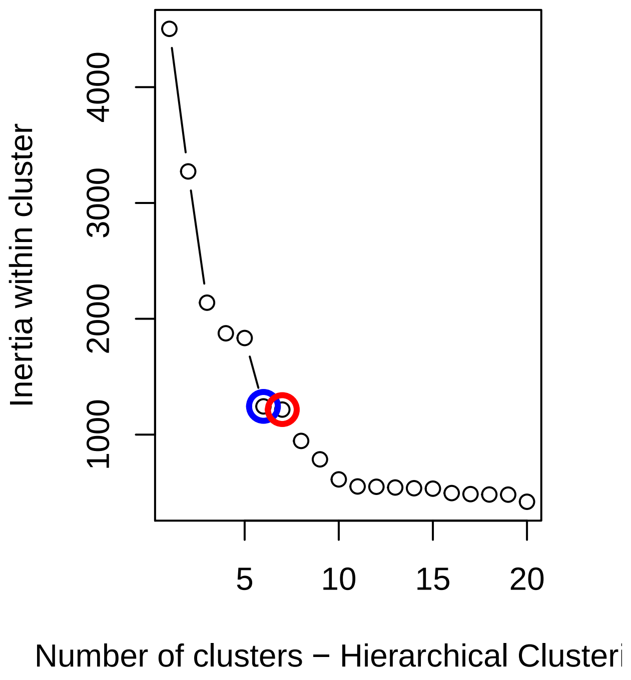


So, we choose to keep de 47 first components on the wind data and the 22 first on the temperature data.

```
## Warning: did not converge in 10 iterations
```

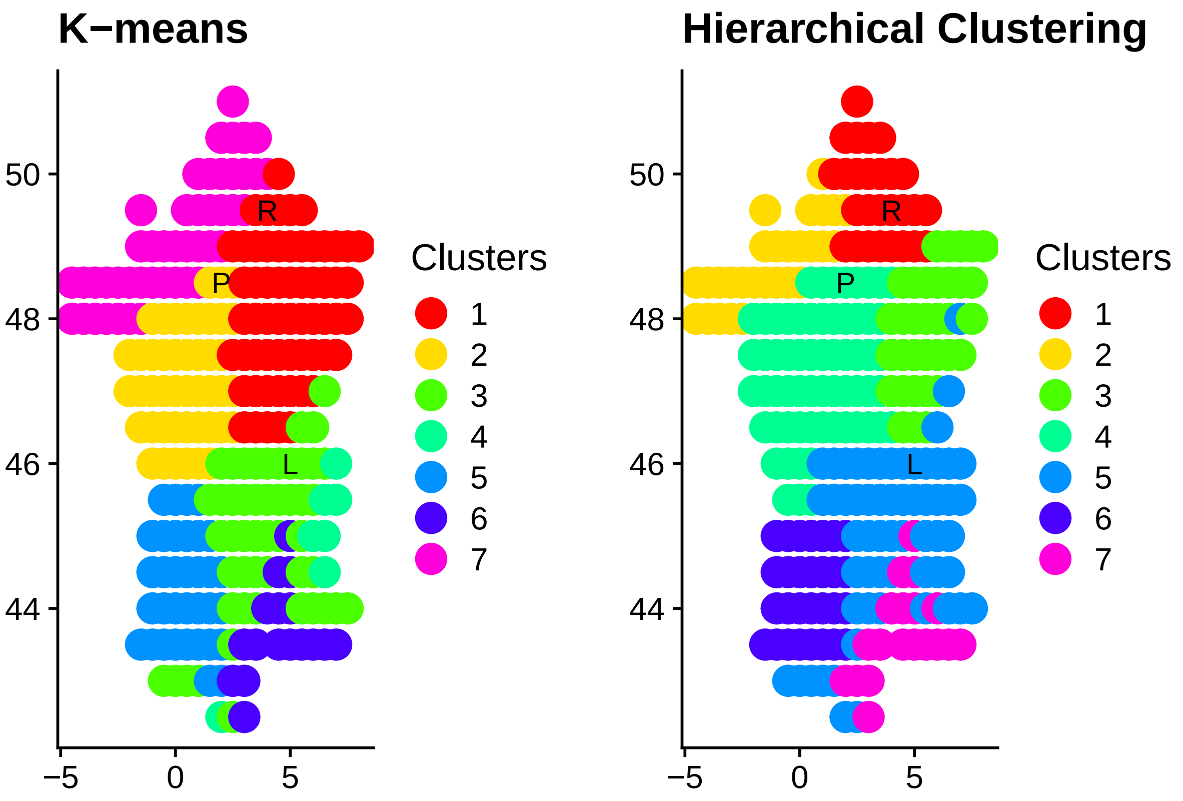
```
par(mfrow = c(1,2))
plot(tot_with1, type = "b", xlab = "Number of clusters - kmeans", ylab="Inertia within-cluster")
points(c(6,7), tot_with1[c(6,7)], col = c("blue","red"), cex = 2, lwd = 3)
plot(inertia.hc.pca.1[1:20], type = "b", xlab = "Number of clusters - Hierarchical Clustering", ylab =
points(c(6,7), inertia.hc.pca.1[c(6,7)], col = c("blue","red"), cex = 2, lwd = 3)
```





The last big gap of inertia is between 5 and 6 clusters. We choose 7 clusters for each algorithm to be able to compare their clusters.

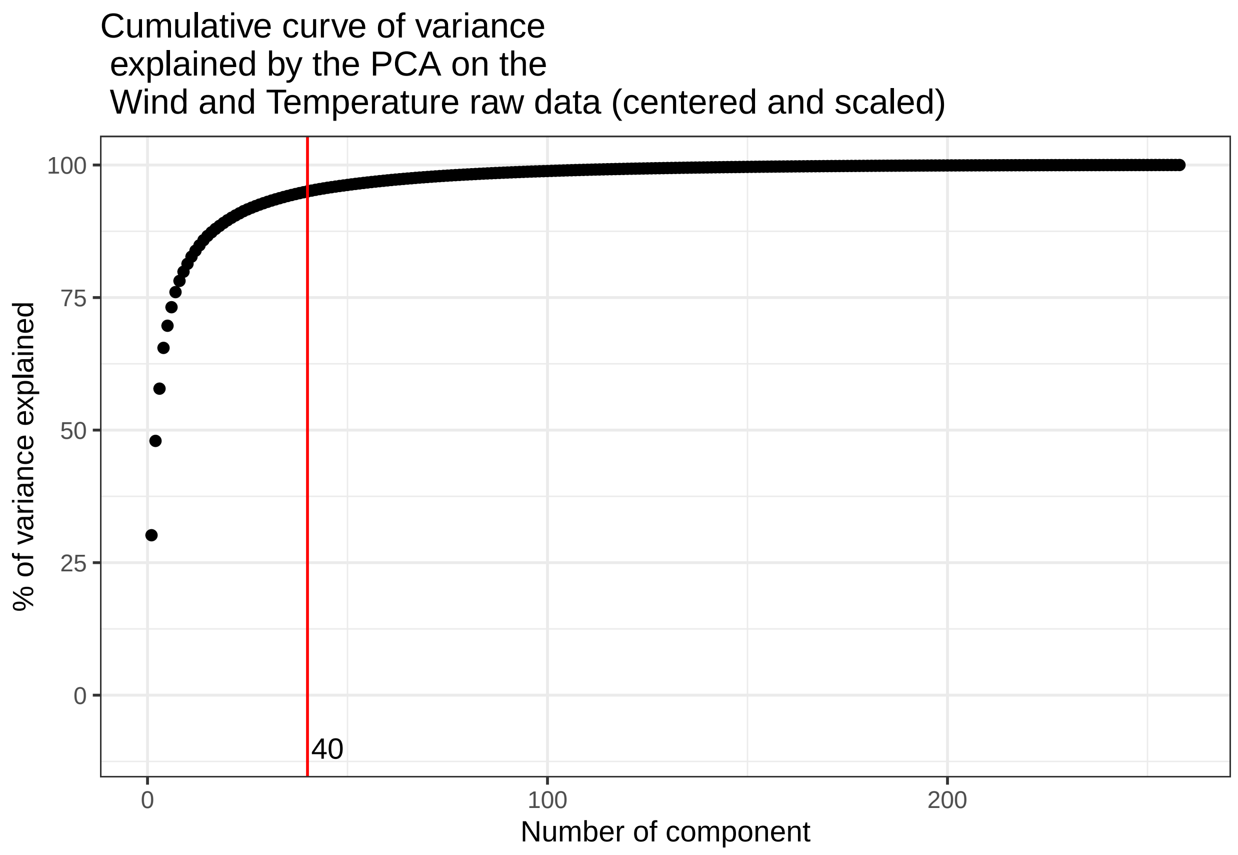
Displaying the maps



We can again find our previous clusters but with more precision as there is more clusters. Each one of them is associated with a special part of the French landscape such as, meditarrean coasts, Alpes, the Channel (La Manche) coasts, etc.

Second approach: Unique PCA

Our second approach was to perfom a PCA on both `wind` and `temp` variables at the same time.

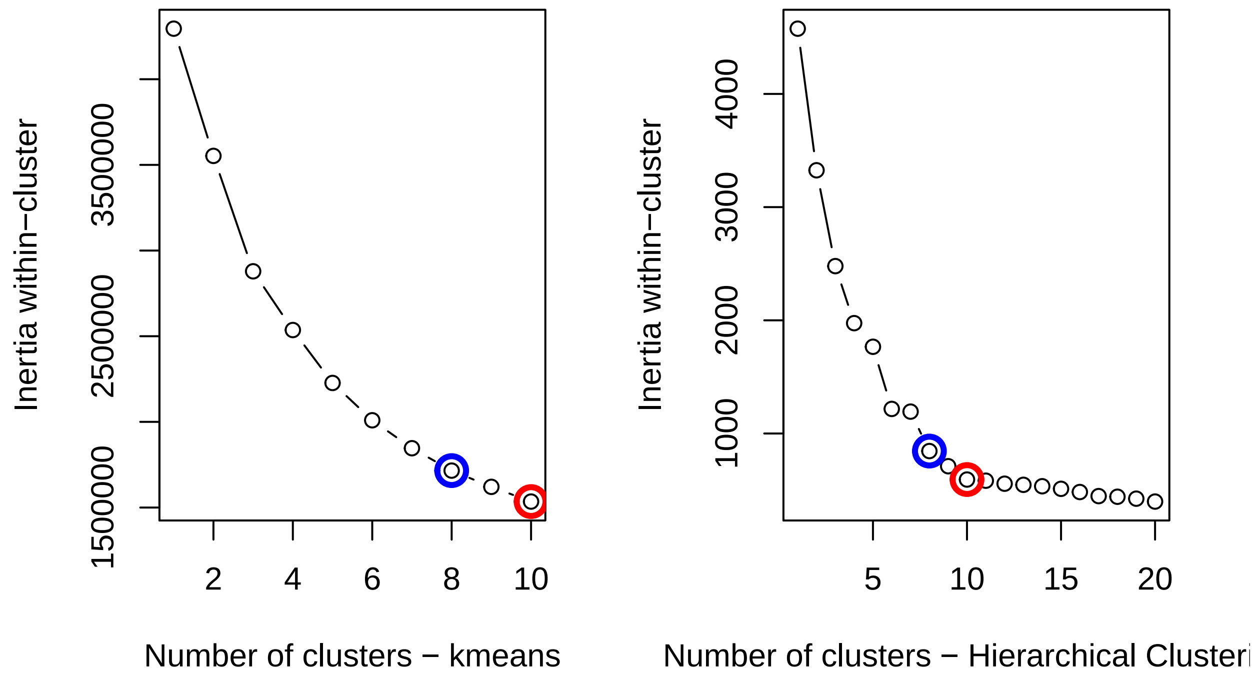


In order to have 95% of variance explained we'd have to keep the first 40 components of our PCA.

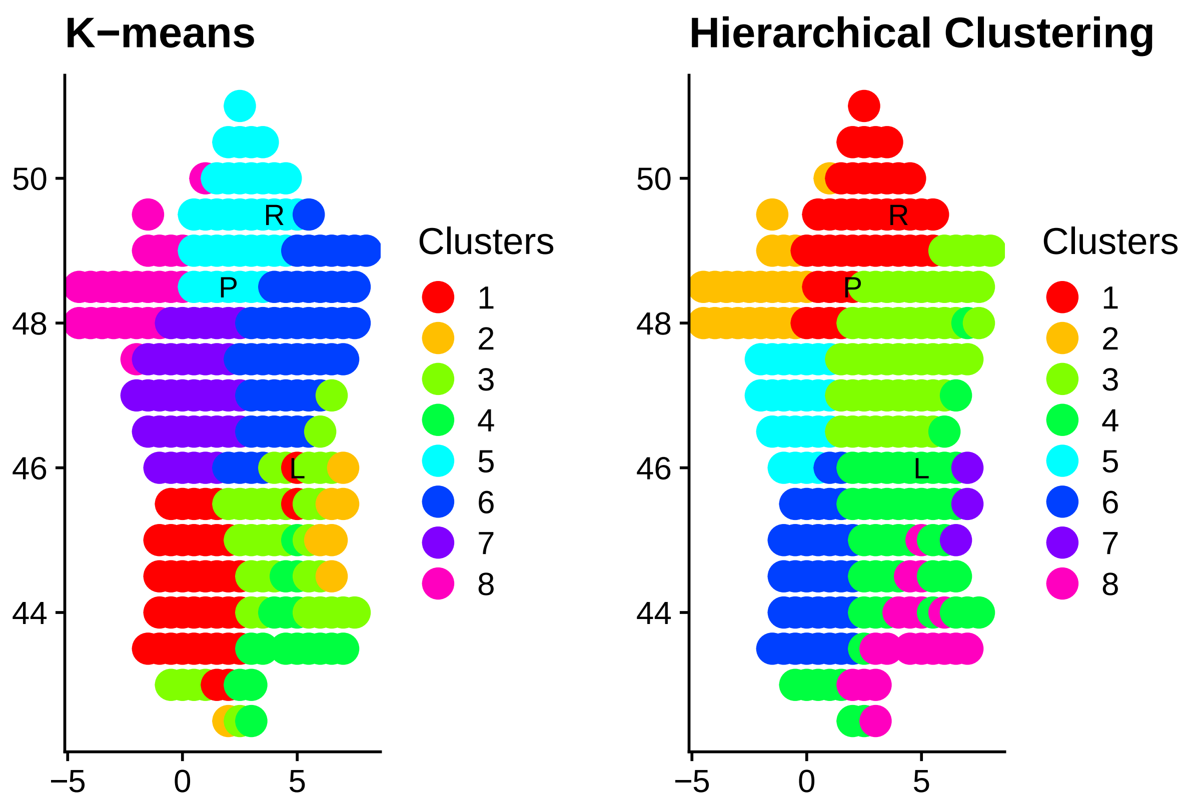
```
## Warning: did not converge in 10 iterations
```

```
## HC
hc.pca.2 = hclust(dist(res.pca$ind$coord, "euclidian"), method = "ward.D")
GPSpos$hc.pca.2 = as.factor(cutree(hc.pca.2, 8))
inertia.hc.pca.2 = sort(hc.pca.2$height, decreasing = TRUE)
```

Results

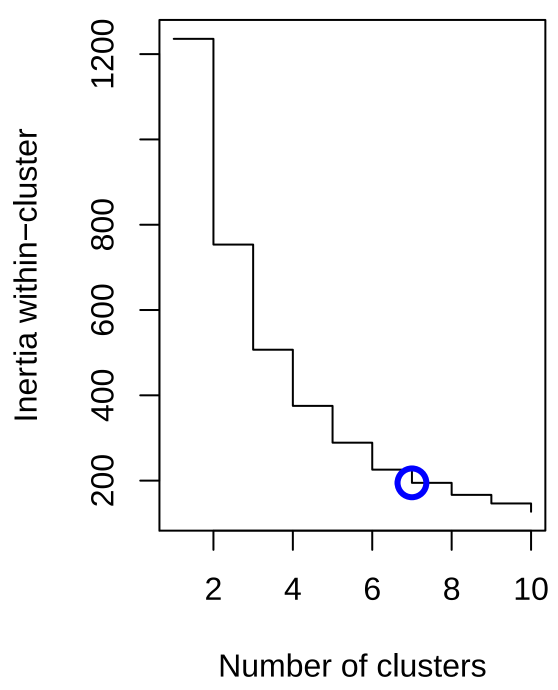


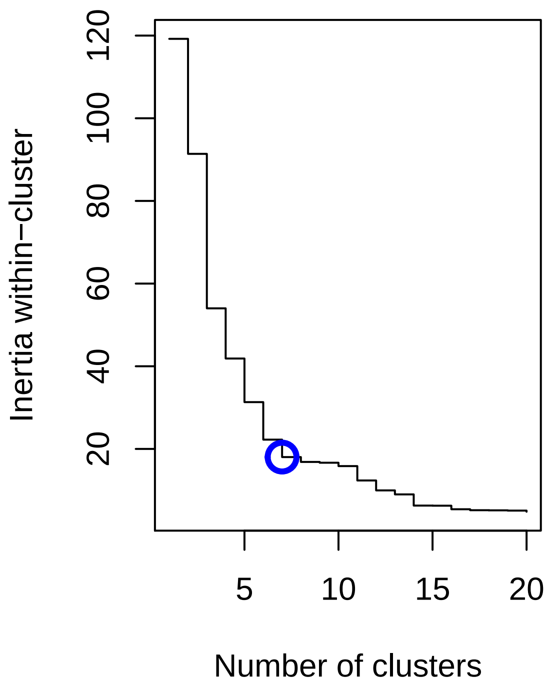
When we perform a PCA on the aggregated data of `wind` and `temp`, the number of clusters is increasing to 10 but to keep the results simple and interpretable we keep only 8 clusters.



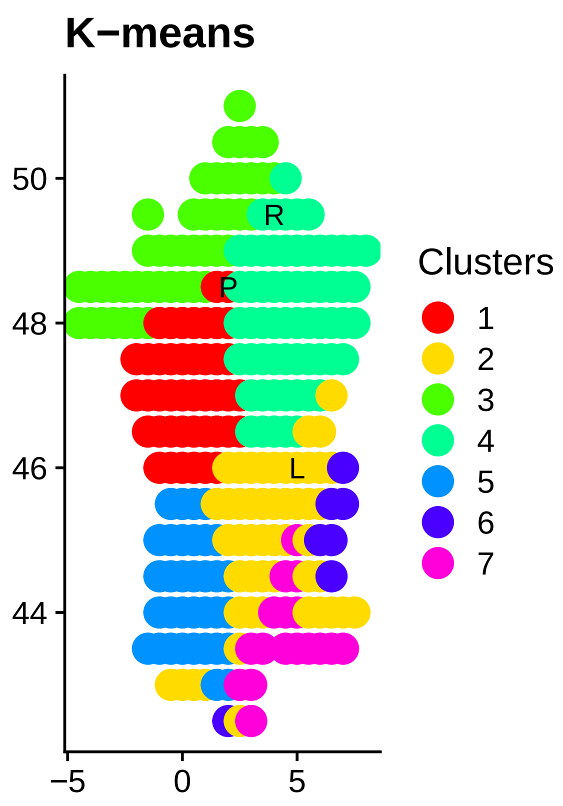
Third approach: Mean per year

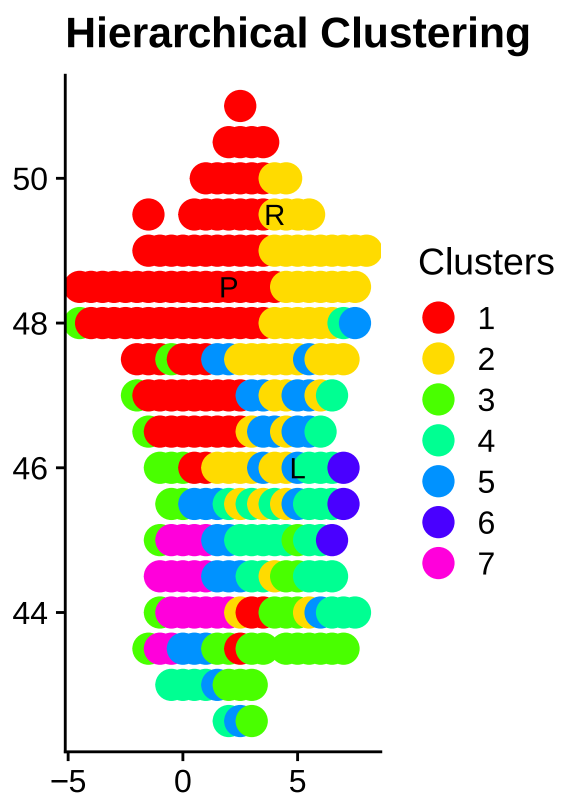
Our third approach was to replace the varibles at our disposition by their means, in order to know is only the annual wind force and temperature was enough information to perform a correct segmentation of a territory.





With the same argument as our other approaches we choose to keep 7 clusters in this approach.





We obtain two clearly different results there. On one hand, k-means gives us clusters that are similar as the ones we got before, and, on the other hand, Hierarchical Clustering gives us some new clusters. But we can still find information we already had, such as cluster 6 which is Alpes.

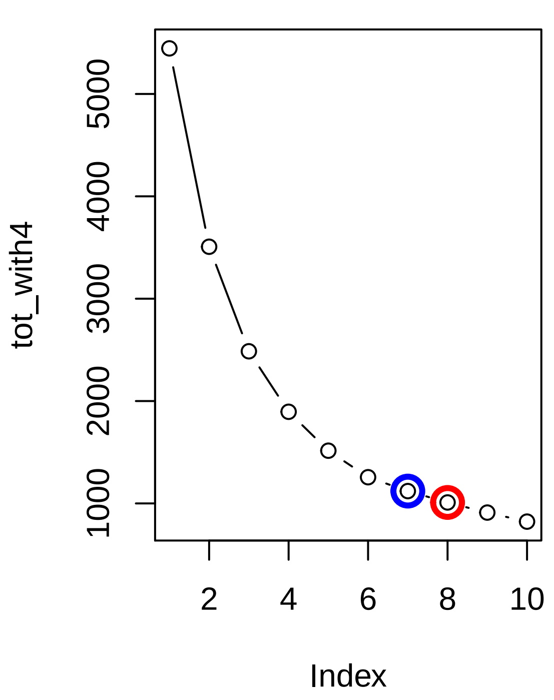
Fourth approach: Mean per season

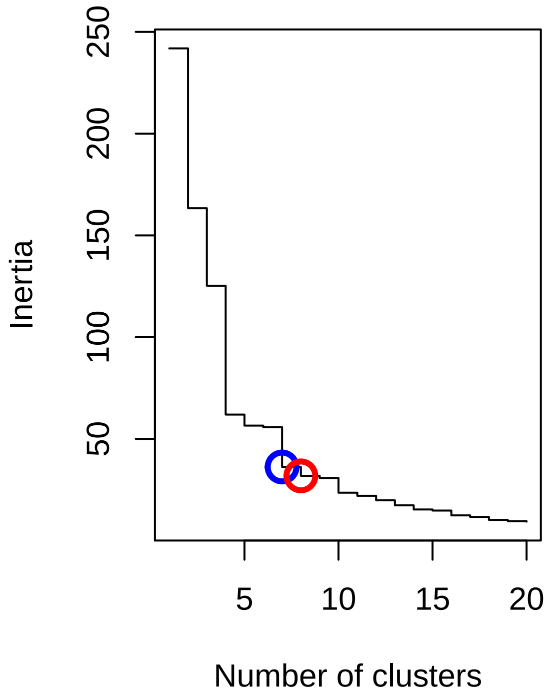
In our fourth and last approach, rather than doing a raw mean of the data we decided to separate the data by seasonand then apply this mean. So we now have $n = 8$ variables which are mean `wind` and mean `temp` in the four seasons.

```
df2 = data.frame(long = GPSpos$Lon, lat = GPSpos$Lat, W.H = W.H, W.P = W.P, W.E = W.E, W.A = W.A, T.H =
dim(df2)
```

```
## [1] 259  10
```

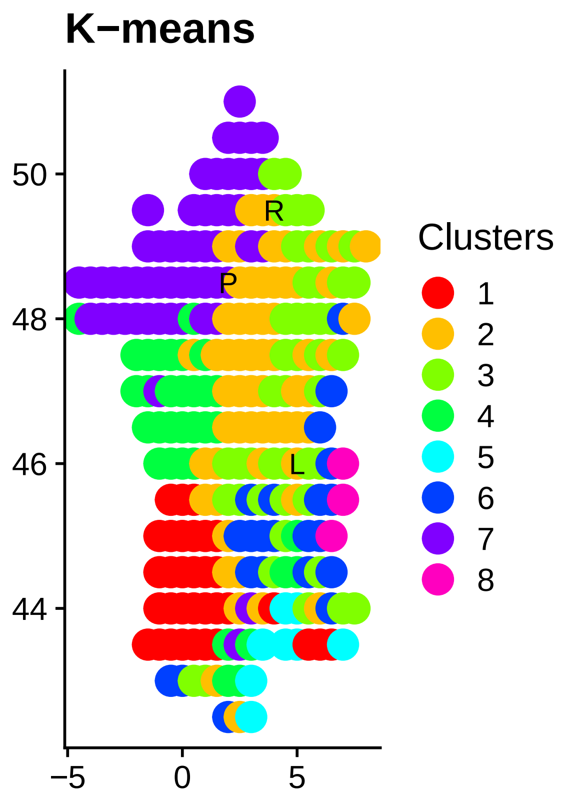
```
## k-means
tot_with4 = NULL
for(k in 1:10){
  tot_with4[k] = kmeans(df2[,c("W.H","W.P","W.E","W.A","T.H","T.P","T.E","T.A")], centers = k, nstart =
}
#plot(tot_with4, type = "b")
```

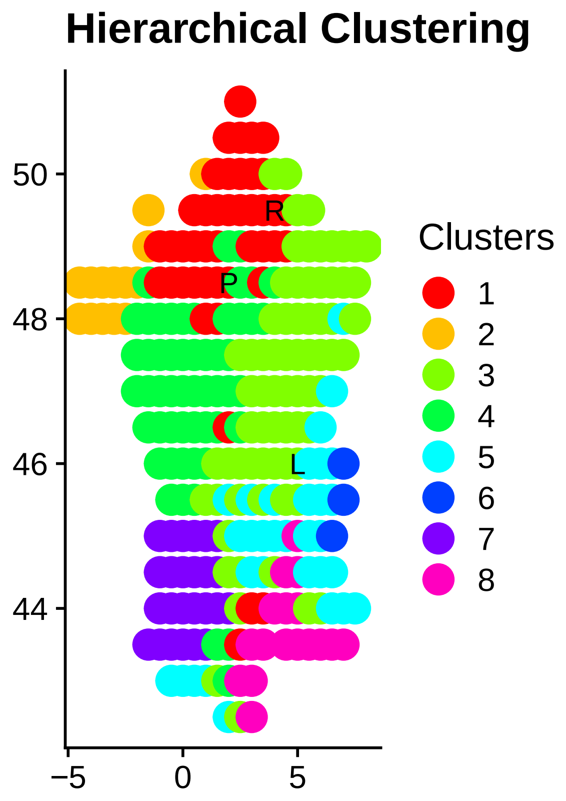




With the same argument as our other approaches we choose to keep 8 clusters in this approach.

Displaying the maps





This last reslust is really interesting. Indeed, by dividing our data in seasons, we tried to give to our k-means and Hierarchical algorithms more precise information, aiming for obtaining more precise clusters. But on one hand for the k-means we got, at least visually, less precise clusters and, on the other hand, hierachical clustering gave "better" results than it did with our previous approach.