Projet\_Classification

Bernice AGOSSOUVO

13/02/2022

## Installation des bibliothèques

library(readxl)

## Warning: package 'readxl' was built under R version 4.0.5

library(FactoMineR)

## Warning: package 'FactoMineR' was built under R version 4.0.5

library (factoextra)

## Warning: package 'factoextra' was built under R version 4.0.5

## Loading required package: ggplot2

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

library(questionr)

## Warning: package 'questionr' was built under R version 4.0.5

library(prettyR)

## Warning: package 'prettyR' was built under R version 4.0.3

##   
## Attaching package: 'prettyR'

## The following objects are masked from 'package:questionr':  
##   
## describe, freq

library(corrplot)

## Warning: package 'corrplot' was built under R version 4.0.5

## corrplot 0.89 loaded

library(PerformanceAnalytics)

## Warning: package 'PerformanceAnalytics' was built under R version 4.0.5

## Loading required package: xts

## Warning: package 'xts' was built under R version 4.0.5

## Loading required package: zoo

## Warning: package 'zoo' was built under R version 4.0.5

##   
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':  
##   
## as.Date, as.Date.numeric

##   
## Attaching package: 'PerformanceAnalytics'

## The following object is masked from 'package:graphics':  
##   
## legend

library(ggpubr)

## Warning: package 'ggpubr' was built under R version 4.0.5

library(cluster)  
library(doBy)

## Warning: package 'doBy' was built under R version 4.0.5

## 1) Chargement de jeu de Données

BDD <- read.csv("C:/Users/berni/Desktop/cours\_Eiffel/Cours S3/travaux pratiques/BDD.csv", row.names=1, sep=";")

## 2) Analyse elementaire et commentaire

BDD <- BDD[,1:9]  
  
attach(BDD)

## The following object is masked from package:base:  
##   
## T

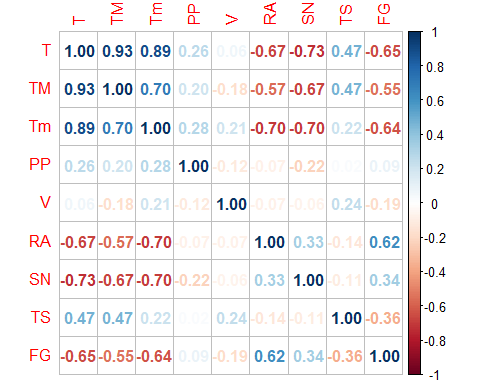
cv=function(x, na.rm=F){  
 sd(x, na.rm = na.rm)/mean(x,na.rm = na.rm)  
}  
  
describe(BDD[,c(1:9)], num.desc = c("mean","cv"))

## Description of BDD[, c(1:9)]

##   
## Numeric   
## mean cv  
## T 13.74 0.13  
## TM 19.04 0.09  
## Tm 8.19 0.26  
## PP 777.79 0.19  
## V 11.76 0.24  
## RA 151.24 0.16  
## SN 8.22 0.71  
## TS 15.41 0.74  
## FG 44.12 0.66

## Correlation entre les variables

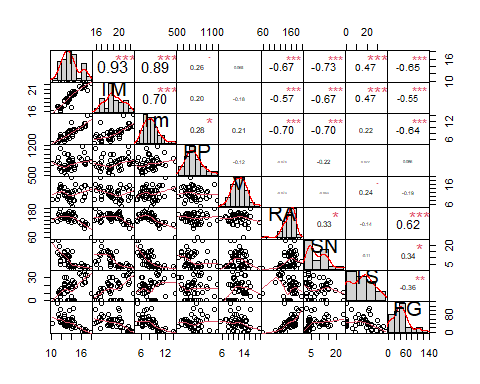
mcor <-cor(BDD)  
corrplot(mcor, method = "number")



mcor

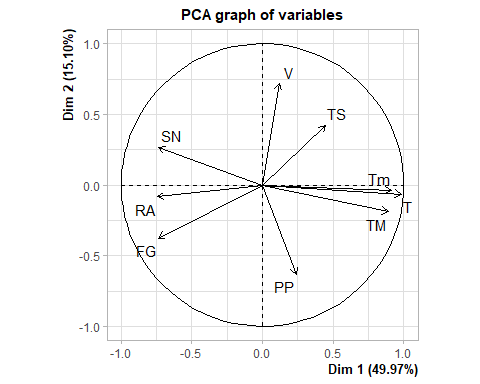
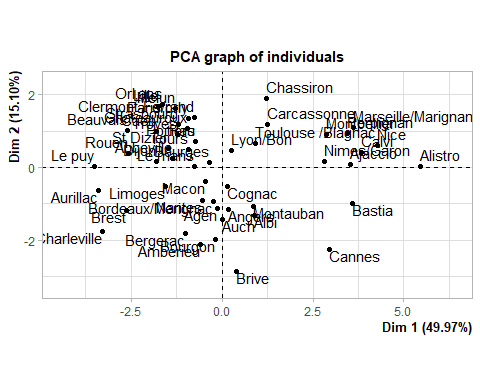
## T TM Tm PP V RA  
## T 1.00000000 0.9316945 0.8899967 0.26043509 0.06482408 -0.67131257  
## TM 0.93169454 1.0000000 0.6959166 0.20207495 -0.17697636 -0.56894098  
## Tm 0.88999671 0.6959166 1.0000000 0.28340995 0.20574953 -0.70272396  
## PP 0.26043509 0.2020749 0.2834099 1.00000000 -0.11988508 -0.07445743  
## V 0.06482408 -0.1769764 0.2057495 -0.11988508 1.00000000 -0.07323418  
## RA -0.67131257 -0.5689410 -0.7027240 -0.07445743 -0.07323418 1.00000000  
## SN -0.73371441 -0.6736773 -0.6976487 -0.22260222 -0.06288628 0.33461009  
## TS 0.47052768 0.4695803 0.2176329 0.02190411 0.24046581 -0.13627145  
## FG -0.65149882 -0.5466510 -0.6390787 0.08621030 -0.18794873 0.61861088  
## SN TS FG  
## T -0.73371441 0.47052768 -0.6514988  
## TM -0.67367727 0.46958025 -0.5466510  
## Tm -0.69764866 0.21763287 -0.6390787  
## PP -0.22260222 0.02190411 0.0862103  
## V -0.06288628 0.24046581 -0.1879487  
## RA 0.33461009 -0.13627145 0.6186109  
## SN 1.00000000 -0.10903701 0.3410303  
## TS -0.10903701 1.00000000 -0.3621080  
## FG 0.34103029 -0.36210804 1.0000000

chart.Correlation(BDD, histogram=TRUE, pch=19)



## III / Analyse en Composantes Principales

res\_pca <- PCA(BDD, scale.unit = TRUE,ncp=5, graph = TRUE)



print (res\_pca)

## \*\*Results for the Principal Component Analysis (PCA)\*\*  
## The analysis was performed on 51 individuals, described by 9 variables  
## \*The results are available in the following objects:  
##   
## name description   
## 1 "$eig" "eigenvalues"   
## 2 "$var" "results for the variables"   
## 3 "$var$coord" "coord. for the variables"   
## 4 "$var$cor" "correlations variables - dimensions"  
## 5 "$var$cos2" "cos2 for the variables"   
## 6 "$var$contrib" "contributions of the variables"   
## 7 "$ind" "results for the individuals"   
## 8 "$ind$coord" "coord. for the individuals"   
## 9 "$ind$cos2" "cos2 for the individuals"   
## 10 "$ind$contrib" "contributions of the individuals"   
## 11 "$call" "summary statistics"   
## 12 "$call$centre" "mean of the variables"   
## 13 "$call$ecart.type" "standard error of the variables"   
## 14 "$call$row.w" "weights for the individuals"   
## 15 "$call$col.w" "weights for the variables"

res\_pca$ind$dist

## Dijon Clermont-Ferrand Cognac Bourgon   
## 2.262677 2.792782 1.583957 2.281053   
## Le puy Lille Lyon/Bon Macon   
## 4.611748 2.536883 1.737749 1.685018   
## Marseille/Marignane Limoges Le mans Montpellier   
## 3.977278 2.639532 1.372138 3.265678   
## Nantes Nice Nimes/Garon Montauban   
## 1.945629 4.580999 3.227375 2.646132   
## Melun Orl‚ans Pau Toulouse /Blagnac   
## 2.286731 2.537579 1.109350 1.718498   
## St Dizier Tours Troyes Strasbourg   
## 1.976341 1.497332 1.731941 2.512737   
## Rouen Poitiers Perpignan Paris/orly   
## 2.800031 1.109350 3.706235 2.173894   
## Abbeville Agen Ajaccio Albi   
## 2.605583 1.742009 3.928408 2.349008   
## Amberieu Alistro Angers Auch   
## 2.359981 6.215405 1.984539 2.802600   
## Aurillac Bastia Beauvais Bergerac   
## 4.071082 4.016421 2.979707 3.001520   
## Brest Brive Bourges Bordeaux/Merignac   
## 4.767806 3.123557 1.940605 2.892451   
## Caen Cannes Chassiron Calvi   
## 2.414259 4.250902 4.050250 4.023650   
## Charleville Carcassonne Chateauroux   
## 4.460241 2.240938 2.246436

### les valeurs propres de l'ACP

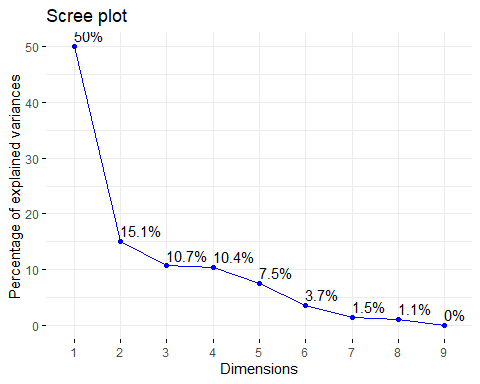
eig\_val <- get\_eigenvalue(res\_pca)  
eig\_val

## eigenvalue variance.percent cumulative.variance.percent  
## Dim.1 4.497009441 49.96677157 49.96677  
## Dim.2 1.359193823 15.10215359 65.06893  
## Dim.3 0.962626821 10.69585356 75.76478  
## Dim.4 0.934897312 10.38774791 86.15253  
## Dim.5 0.674099567 7.48999518 93.64252  
## Dim.6 0.330052590 3.66725100 97.30977  
## Dim.7 0.135685875 1.50762083 98.81739  
## Dim.8 0.102097092 1.13441213 99.95181  
## Dim.9 0.004337481 0.04819423 100.00000

copie(eig\_val)

### le graphique des eboulis de valeurs propres

fviz\_eig(res\_pca, geom = c("line"),linecolor = "blue",addlabels = TRUE, ylim = c(0, 50))



### Les variables

var <- get\_pca\_var(res\_pca)  
var

## Principal Component Analysis Results for variables  
## ===================================================  
## Name Description   
## 1 "$coord" "Coordinates for the variables"   
## 2 "$cor" "Correlations between variables and dimensions"  
## 3 "$cos2" "Cos2 for the variables"   
## 4 "$contrib" "contributions of the variables"

#### Coordonnées

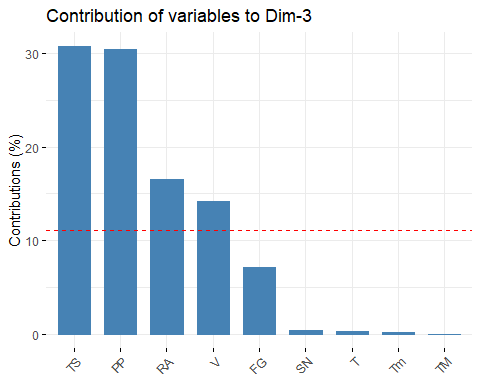
head(var$coord)

## Dim.1 Dim.2 Dim.3 Dim.4 Dim.5  
## T 0.9834393 -0.06389934 0.059229455 0.06806077 -0.05519658  
## TM 0.8891562 -0.18999914 0.004236464 0.32733085 -0.12137582  
## Tm 0.9121588 -0.03746051 -0.041132669 -0.29182126 0.01822712  
## PP 0.2416396 -0.63417877 0.540830398 -0.24507000 0.41274089  
## V 0.1177526 0.72156262 0.369977135 -0.55485632 -0.05541513  
## RA -0.7426251 -0.08378377 0.399601861 0.13279701 -0.38146677

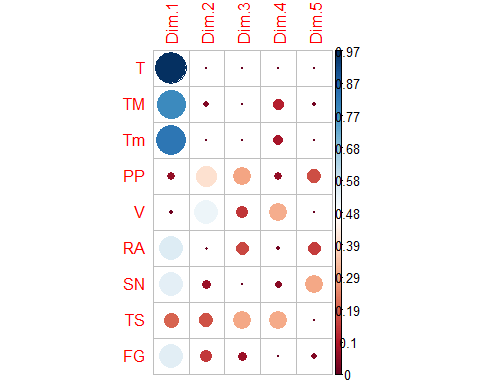
#### Cos2: qualité de répresentation  
head(var$cos2)

## Dim.1 Dim.2 Dim.3 Dim.4 Dim.5  
## T 0.96715285 0.004083126 3.508128e-03 0.004632269 0.0030466623  
## TM 0.79059881 0.036099672 1.794763e-05 0.107145488 0.0147320887  
## Tm 0.83203375 0.001403290 1.691896e-03 0.085159647 0.0003322279  
## PP 0.05838971 0.402182716 2.924975e-01 0.060059305 0.1703550383  
## V 0.01386568 0.520652615 1.368831e-01 0.307865537 0.0030708366  
## RA 0.55149204 0.007019720 1.596816e-01 0.017635046 0.1455168949

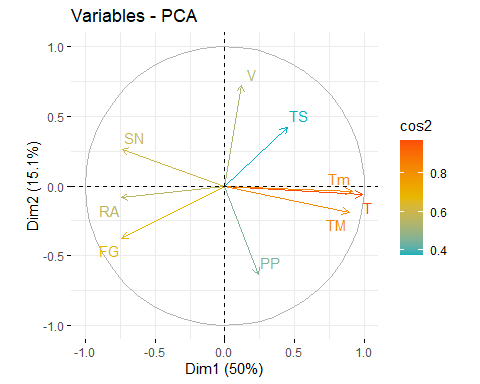
#### Contributions aux composantes principales  
cos = head(var$contrib)  
copie (cos)  
fviz\_contrib(res\_pca, choice = "var", axes = 3)



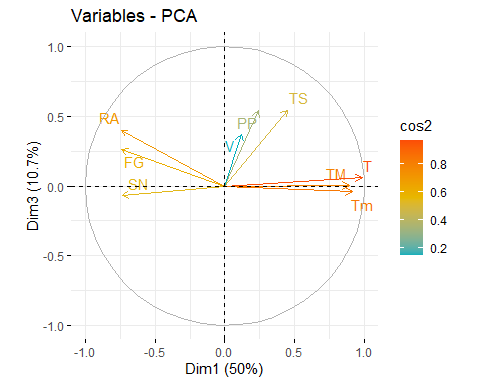
### correlation entre variables et chaque dimension  
corrplot(var$cos2,is.corr=FALSE)



### cercle de correlation   
#### Sur les axes 1 et 2  
fviz\_pca\_var(res\_pca, col.var = "cos2",  
 gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),  
 repel = TRUE # Évite le chevauchement de texte  
)



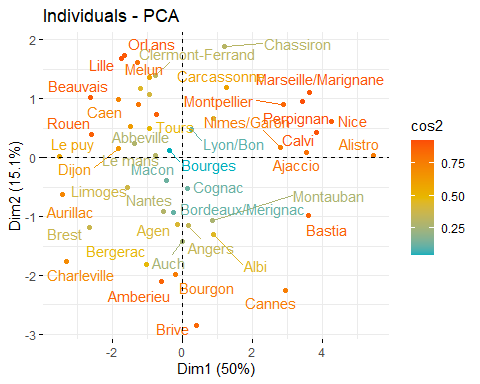
### Sur les axes 1 et 3  
fviz\_pca\_var(res\_pca, axes = c(1,3), col.var = "cos2",  
 gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),  
 repel = TRUE # Évite le chevauchement de texte  
)



### Qualité de représentation des individus  
  
#### Sur les axes 1 et 2

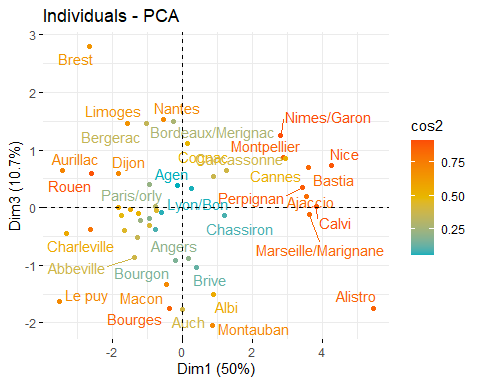
fviz\_pca\_ind (res\_pca, col.ind = "cos2",  
 gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),  
 repel = TRUE # Évite le chevauchement de texte  
)

## Warning: ggrepel: 8 unlabeled data points (too many overlaps). Consider  
## increasing max.overlaps

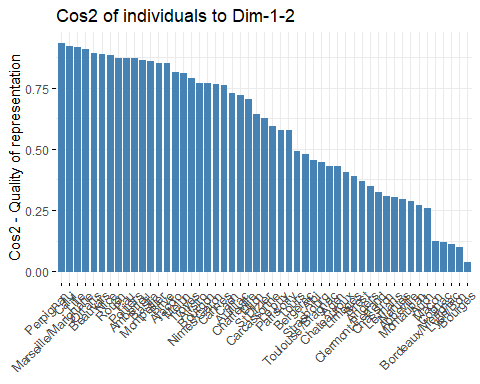


#### Sur les axes 1 et 3  
fviz\_pca\_ind (res\_pca,axes = c(1,3), col.ind = "cos2",  
 gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),  
 repel = TRUE # Évite le chevauchement de texte  
)

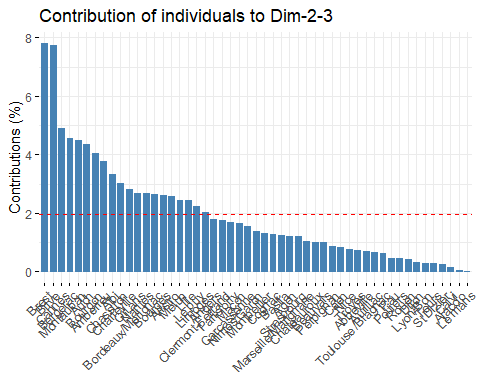
## Warning: ggrepel: 16 unlabeled data points (too many overlaps). Consider  
## increasing max.overlaps



### Contribution   
fviz\_cos2(res\_pca, choice = "ind", axes = 1:2)

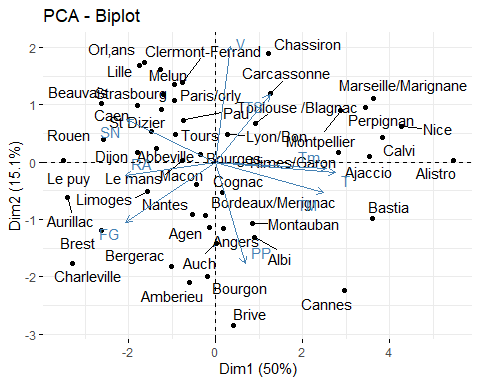


fviz\_contrib(res\_pca, choice = "ind", axes = 2:3)



### Biplot   
  
fviz\_pca\_biplot (res\_pca,  
 gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),  
 repel = TRUE # Évite le chevauchement de texte  
)

## Warning: ggrepel: 3 unlabeled data points (too many overlaps). Consider  
## increasing max.overlaps



## respca <- PCA(BDDX[,2:11], scale.unit = TRUE, graph = TRUE)  
  
### fviz\_pca\_ind (respca,axes = 2:3,  
 ### gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),  
 ## repel = TRUE # Évite le chevauchement de texte)####  
### le plan pour les axes 1 et 3  
  
# plot(rescap,choix = "ind",axes = c(1,3))  
  
# plot(rescap,choix = "var",axes =c(1,3))

## CLASSIFICATION PAR LA METHODE K MEANS  
# Calculer k-means avec k = 4  
set.seed(123) # garde la classification fixe  
res.km <- kmeans(scale(BDD),4, nstart = 25)  
res.km

## K-means clustering with 4 clusters of sizes 13, 10, 20, 8  
##   
## Cluster means:  
## T TM Tm PP V RA  
## 1 0.0889064 0.3081096 0.06205413 0.2549638 -0.90394772 0.3441468  
## 2 1.6450977 1.4737896 1.42567223 0.7434610 0.29813327 -1.4894334  
## 3 -0.3953042 -0.4365704 -0.33755160 -0.7692619 0.43590326 0.3197564  
## 4 -1.2125845 -1.2514890 -1.03904925 0.5795124 0.00649031 0.5031622  
## SN TS FG  
## 1 -0.6283096 -0.7113022 0.5051824  
## 2 -0.9253522 1.1136586 -1.2426222  
## 3 0.3219636 0.1042926 -0.1985064  
## 4 1.3727844 -0.4969385 1.2286223  
##   
## Clustering vector:  
## Dijon Clermont-Ferrand Cognac Bourgon   
## 4 3 1 1   
## Le puy Lille Lyon/Bon Macon   
## 4 3 3 1   
## Marseille/Marignane Limoges Le mans Montpellier   
## 2 4 3 2   
## Nantes Nice Nimes/Garon Montauban   
## 1 2 2 1   
## Melun Orl‚ans Pau Toulouse /Blagnac   
## 3 3 3 3   
## St Dizier Tours Troyes Strasbourg   
## 3 3 3 3   
## Rouen Poitiers Perpignan Paris/orly   
## 4 3 2 3   
## Abbeville Agen Ajaccio Albi   
## 3 1 2 1   
## Amberieu Alistro Angers Auch   
## 1 2 1 1   
## Aurillac Bastia Beauvais Bergerac   
## 4 2 4 1   
## Brest Brive Bourges Bordeaux/Merignac   
## 4 1 3 1   
## Caen Cannes Chassiron Calvi   
## 3 2 3 2   
## Charleville Carcassonne Chateauroux   
## 4 3 3   
##   
## Within cluster sum of squares by cluster:  
## [1] 43.50586 38.50786 67.34299 40.36065  
## (between\_SS / total\_SS = 57.8 %)  
##   
## Available components:  
##   
## [1] "cluster" "centers" "totss" "withinss" "tot.withinss"  
## [6] "betweenss" "size" "iter" "ifault"

## Ajout de la variable res.km$cluster dans le jeu de données.  
BDD\_NEW=cbind(BDD,res.km$cluster) #ajout la variable res.km$cluster  
View(BDD\_NEW)

# Renommer la variable cla.km en classes  
BDD\_NEW = rename.variable(BDD\_NEW, "res.km$cluster", "Kmean")  
  
as.factor(BDD\_NEW$kmean)

## factor(0)  
## Levels:

attach(BDD\_NEW)

## The following objects are masked from BDD:  
##   
## FG, PP, RA, SN, T, Tm, TM, TS, V

## The following object is masked from package:base:  
##   
## T

# Clustering K-means montrant le groupe de chaque individu  
res.km$cluster # repartition des observations dans la parties

## Dijon Clermont-Ferrand Cognac Bourgon   
## 4 3 1 1   
## Le puy Lille Lyon/Bon Macon   
## 4 3 3 1   
## Marseille/Marignane Limoges Le mans Montpellier   
## 2 4 3 2   
## Nantes Nice Nimes/Garon Montauban   
## 1 2 2 1   
## Melun Orl‚ans Pau Toulouse /Blagnac   
## 3 3 3 3   
## St Dizier Tours Troyes Strasbourg   
## 3 3 3 3   
## Rouen Poitiers Perpignan Paris/orly   
## 4 3 2 3   
## Abbeville Agen Ajaccio Albi   
## 3 1 2 1   
## Amberieu Alistro Angers Auch   
## 1 2 1 1   
## Aurillac Bastia Beauvais Bergerac   
## 4 2 4 1   
## Brest Brive Bourges Bordeaux/Merignac   
## 4 1 3 1   
## Caen Cannes Chassiron Calvi   
## 3 2 3 2   
## Charleville Carcassonne Chateauroux   
## 4 3 3

res.km$size # répartition des données dans chaque classe

## [1] 13 10 20 8

res.km$centers #coordonnées des centres de classes( qui sont aussi les moyennes dans chaque classe pour une variables

## T TM Tm PP V RA  
## 1 0.0889064 0.3081096 0.06205413 0.2549638 -0.90394772 0.3441468  
## 2 1.6450977 1.4737896 1.42567223 0.7434610 0.29813327 -1.4894334  
## 3 -0.3953042 -0.4365704 -0.33755160 -0.7692619 0.43590326 0.3197564  
## 4 -1.2125845 -1.2514890 -1.03904925 0.5795124 0.00649031 0.5031622  
## SN TS FG  
## 1 -0.6283096 -0.7113022 0.5051824  
## 2 -0.9253522 1.1136586 -1.2426222  
## 3 0.3219636 0.1042926 -0.1985064  
## 4 1.3727844 -0.4969385 1.2286223

res.km$withinss # somme des carrées des ecart intra classes pour chaque classe

## [1] 43.50586 38.50786 67.34299 40.36065

# sum(res.km$withinss/51)  
# res.km$tot.withinss  
# res.km$betweenss  
# res.km$totss  
# copie(round(res.km$centers),3)  
  
## Calcul de la moyenne et ecart-type dans chaque classe pour chaque variable  
summaryBy(T+TM+Tm+PP+V+RA+SN+TS+FG ~ BDD\_NEW$Kmean, data=BDD\_NEW, FUN=mean) # moyennes

## Kmean T.mean TM.mean Tm.mean PP.mean V.mean RA.mean SN.mean  
## 1 1 13.900 19.59231 8.323077 814.7023 9.230769 159.5385 4.538462  
## 2 2 16.680 21.67000 11.200000 885.4300 12.590000 115.3000 2.800000  
## 3 3 13.035 18.26500 7.480000 666.4085 12.975000 158.9500 10.100000  
## 4 4 11.575 16.81250 6.000000 861.6925 11.775000 163.3750 16.250000  
## TS.mean FG.mean  
## 1 7.307692 58.92308  
## 2 28.100000 7.70000  
## 3 16.600000 38.30000  
## 4 9.750000 80.12500

apply(BDD, 2, mean)

## T TM Tm PP V RA SN   
## 13.741176 19.043137 8.192157 777.787059 11.756863 151.235294 8.215686   
## TS FG   
## 15.411765 44.117647

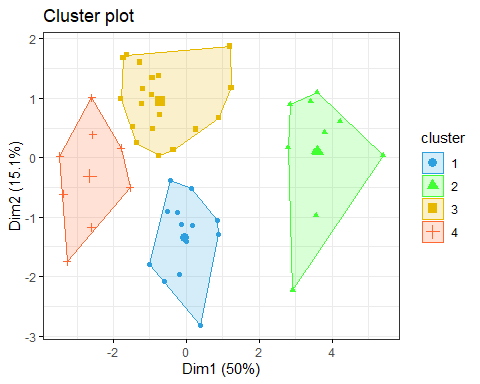
summaryBy(T+TM+Tm+PP+V+RA+SN+TS+FG ~ BDD\_NEW$Kmean, data=BDD\_NEW, FUN=sd) # ecart-type

## Kmean T.sd TM.sd Tm.sd PP.sd V.sd RA.sd SN.sd  
## 1 1 0.5212165 0.7064719 0.8407872 120.40357 2.132051 14.63224 2.470337  
## 2 2 0.6338594 0.4922736 1.6904963 136.59105 2.644260 23.12791 1.475730  
## 3 3 0.8683166 1.0698131 1.4741992 72.93667 2.381922 12.05459 4.153629  
## 4 4 0.7704359 0.7642690 1.0542431 154.54504 2.552730 19.93516 5.800246  
## TS.sd FG.sd  
## 1 9.724408 24.102080  
## 2 8.774331 7.944949  
## 3 8.786353 10.618256  
## 4 7.995534 29.526925

apply(BDD, 2, sd)

## T TM Tm PP V RA SN   
## 1.786413 1.782387 2.109772 144.786262 2.794513 24.126822 5.852568   
## TS FG   
## 11.393290 29.307096

# graphique schématisant l'affectation dans chaque classe  
  
fviz\_cluster(res.km, data = BDD,  
 palette = c("#2E9FDF", "#42ff33", "#E7B800","#ff6833"),   
 geom = "point",  
 labels=TRUE,  
 ellipse.type = "convex",   
 ggtheme = theme\_bw()  
)



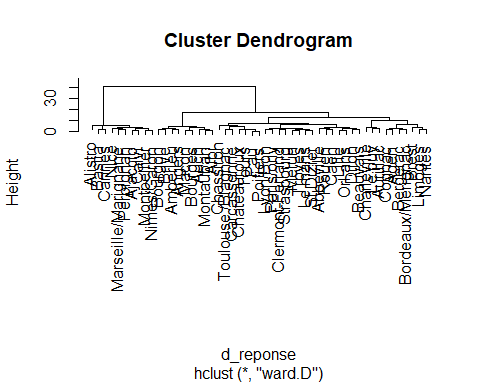
# AUtre méthode pour la moyenne  
meanClass <- aggregate(scale(BDD[ , 1: 9]),  
 by = list(res.km$cluster),  
 FUN = mean)  
colnames(meanClass)[ 1] <- "CLUSTER"  
  
meanClass

## CLUSTER T TM Tm PP V RA  
## 1 1 0.0889064 0.3081096 0.06205413 0.2549638 -0.90394772 0.3441468  
## 2 2 1.6450977 1.4737896 1.42567223 0.7434610 0.29813327 -1.4894334  
## 3 3 -0.3953042 -0.4365704 -0.33755160 -0.7692619 0.43590326 0.3197564  
## 4 4 -1.2125845 -1.2514890 -1.03904925 0.5795124 0.00649031 0.5031622  
## SN TS FG  
## 1 -0.6283096 -0.7113022 0.5051824  
## 2 -0.9253522 1.1136586 -1.2426222  
## 3 0.3219636 0.1042926 -0.1985064  
## 4 1.3727844 -0.4969385 1.2286223

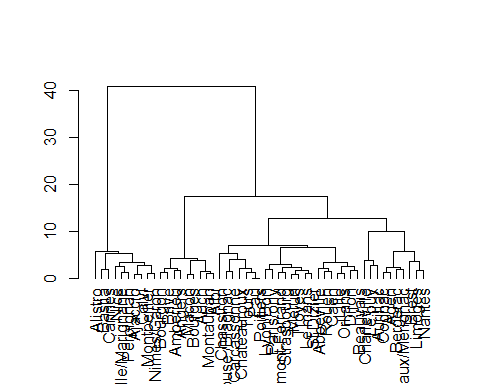
#clusplot(BDD,res.km$cluster, color=TRUE,shade = TRUE,labels=2,lines=0)  
  
# plotcluster(BDD,res.km$cluster)  
  
# Pour Comparer deux clusters   
# cluster.stats(BDD, res.km$cluster,res.km$cluster)  
  
### CLASSIFICATION ASCENDANTE HIERACHIQUE  
reponse <- scale(BDD)  
#matrice des distances entre individus  
d\_reponse<- dist(reponse)  
  
#### Dendogramme de CAH par la méthode Ward avec la fontion hclust  
  
cah.ward <- hclust(d\_reponse,method="ward")

## The "ward" method has been renamed to "ward.D"; note new "ward.D2"

plot(cah.ward)



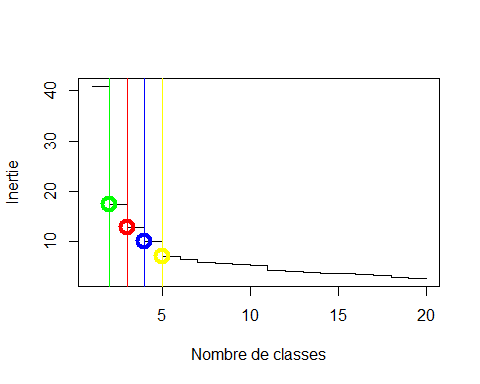
#Schémas Dendrogramme   
cah.ward\_Den <- as.dendrogram(cah.ward)  
plot(cah.ward\_Den)



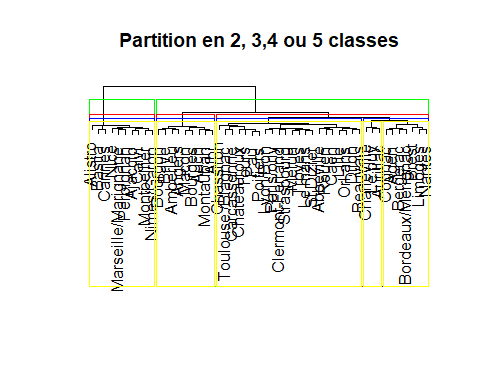
# Découpage du dendogramme, A quel niveau le coupons nous ?  
inertie <- sort(cah.ward$height, decreasing = TRUE)  
inertie

## [1] 40.8976250 17.4041742 12.8246264 9.9544905 7.1112506 6.5303234  
## [7] 5.8182228 5.6410075 5.3936424 5.2969641 4.2077154 4.1201729  
## [13] 3.7907731 3.7374719 3.6464232 3.4846431 3.2843002 2.9381191  
## [19] 2.6764795 2.6730416 2.5815633 2.4533135 2.3947167 2.3781526  
## [25] 2.1365262 2.1309336 2.0345081 1.8604098 1.8259331 1.8187212  
## [31] 1.7759159 1.7175028 1.6955979 1.6947878 1.5003898 1.4414085  
## [37] 1.4280438 1.3250338 1.3094667 1.2993311 1.2642596 1.2347171  
## [43] 1.1158962 1.0701308 1.0469246 1.0033677 0.9129233 0.8777791  
## [49] 0.8546933 0.0000000

plot(inertie[1:20], type = "s", xlab = "Nombre de classes", ylab = "Inertie")  
points(c(2, 3, 4,5), inertie[c(2, 3, 4,5)], col = c("green", "red", "blue", "yellow"), cex = 2, lwd = 4)  
abline(v=c(2, 3, 4,5),col=c("green", "red", "blue", "yellow"))



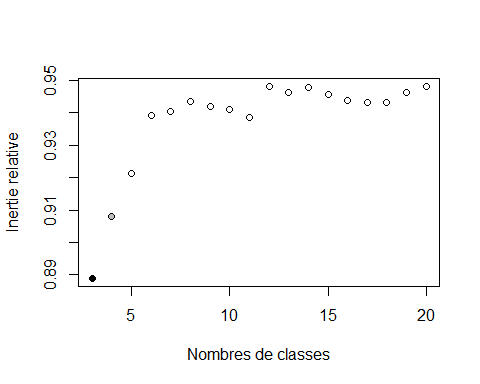
#dendrogramme avec matérialisation des groupes de 3 à 6  
plot(cah.ward, main = "Partition en 2, 3,4 ou 5 classes", xlab = "", ylab = "", sub = "", axes = FALSE)  
rect.hclust(cah.ward,2, border ="green")  
rect.hclust(cah.ward,3, border = "red")  
rect.hclust(cah.ward,4, border = "blue")  
rect.hclust(cah.ward,5, border = "yellow")



#La fonction best.cutree permet de déterminer le nombre de classes optimal(la perte d'inertie est minimale)  
  
best.cutree <- function(hc, min=3, max=20, loss=FALSE, graph=FALSE, ...){  
 if (class(hc)!="hclust") hc <- as.hclust(hc)  
 max <- min(max, length(hc$height))  
 inert.gain <- rev(hc$height)  
 intra <- rev(cumsum(rev(inert.gain)))  
 relative.loss = intra[min:(max)]/intra[(min - 1):(max - 1)]  
 best = which.min(relative.loss)  
 names(relative.loss) <- min:max  
 if (graph) {  
 temp <- relative.loss  
 temp[best] <- NA  
 best2 <- which.min(temp)  
 pch <- rep(1, max-min+1)  
 pch[best] <- 16  
pch[best2] <- 21  
 plot(min:max, relative.loss, pch=pch, bg="grey75", ...)  
 } else {  
 if (loss)  
 relative.loss  
 else  
 best + min - 1  
 }  
}  
## On applique best.cutree à cah.ward en affichant la liste des inerties  
best.cutree(cah.ward)

## [1] 3

# graphique des pertes d'inerties   
best.cutree(cah.ward,graph =TRUE,xlab= "Nombres de classes", ylab = "Inertie relative")



#découpage en 3 groupes  
Kclust <- cutree(cah.ward,k=2)  
table(res.km$cluster)

##   
## 1 2 3 4   
## 13 10 20 8

# ajout de la variable Kclust au jeu de données  
BDD\_NEW = cbind(BDD\_NEW,Kclust)  
# Calcul des moyennes et ecart-types par cluster  
summaryBy(T+TM+Tm+PP+V+RA+SN+TS+FG ~ BDD\_NEW$Kclust, data=BDD\_NEW, FUN=mean) # moyennes

## Kclust T.mean TM.mean Tm.mean PP.mean V.mean RA.mean SN.mean  
## 1 1 13.02439 18.40244 7.458537 751.5327 11.55366 160.0 9.536585  
## 2 2 16.68000 21.67000 11.200000 885.4300 12.59000 115.3 2.800000  
## TS.mean FG.mean  
## 1 12.31707 53.0  
## 2 28.10000 7.7

apply(BDD, 2, mean)

## T TM Tm PP V RA SN   
## 13.741176 19.043137 8.192157 777.787059 11.756863 151.235294 8.215686   
## TS FG   
## 15.411765 44.117647

summaryBy(T+TM+Tm+PP+V+RA+SN+TS+FG ~ BDD\_NEW$Kclust, data=BDD\_NEW, FUN=sd) # ecart-type

## Kclust T.sd TM.sd Tm.sd PP.sd V.sd RA.sd SN.sd  
## 1 1 1.1013130 1.3306930 1.451719 135.6566 2.82357 14.34225 5.762367  
## 2 2 0.6338594 0.4922736 1.690496 136.5910 2.64426 23.12791 1.475730  
## TS.sd FG.sd  
## 1 9.740223 25.436195  
## 2 8.774331 7.944949

apply(BDD, 2, sd)

## T TM Tm PP V RA SN   
## 1.786413 1.782387 2.109772 144.786262 2.794513 24.126822 5.852568   
## TS FG   
## 11.393290 29.307096

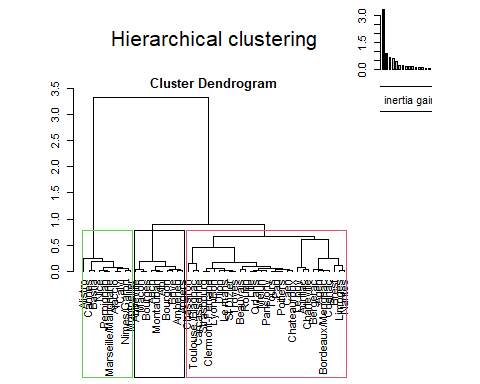
#liste des groupes  
print(sort(Kclust))

## Dijon Clermont-Ferrand Cognac Bourgon   
## 1 1 1 1   
## Le puy Lille Lyon/Bon Macon   
## 1 1 1 1   
## Limoges Le mans Nantes Montauban   
## 1 1 1 1   
## Melun Orl‚ans Pau Toulouse /Blagnac   
## 1 1 1 1   
## St Dizier Tours Troyes Strasbourg   
## 1 1 1 1   
## Rouen Poitiers Paris/orly Abbeville   
## 1 1 1 1   
## Agen Albi Amberieu Angers   
## 1 1 1 1   
## Auch Aurillac Beauvais Bergerac   
## 1 1 1 1   
## Brest Brive Bourges Bordeaux/Merignac   
## 1 1 1 1   
## Caen Chassiron Charleville Carcassonne   
## 1 1 1 1   
## Chateauroux Marseille/Marignane Montpellier Nice   
## 1 2 2 2   
## Nimes/Garon Perpignan Ajaccio Alistro   
## 2 2 2 2   
## Bastia Cannes Calvi   
## 2 2 2

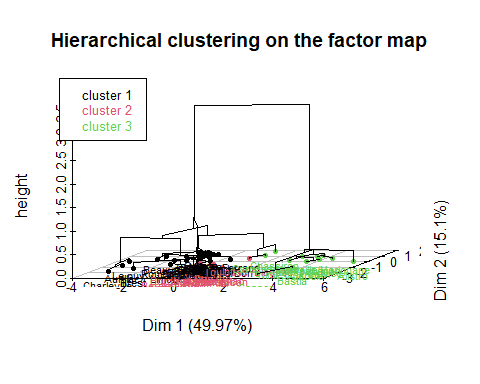
### Classification CAH avec le package HCPC  
  
res.hcpc <- HCPC(res\_pca, graph = FALSE)  
table (res.hcpc$data.clust$clust)

##   
## 1 2 3   
## 25 14 12

HCPC <- res.hcpc$data.clust$clust  
plot(res.hcpc, choice = "tree")

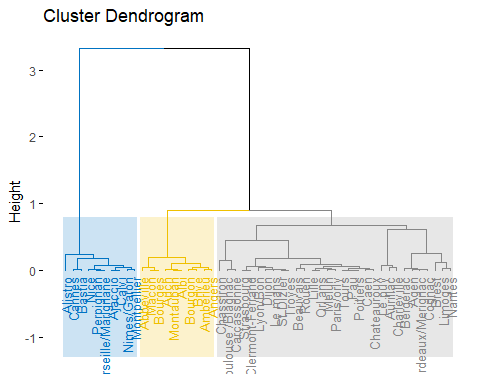


plot(res.hcpc, choice = "3D.map")



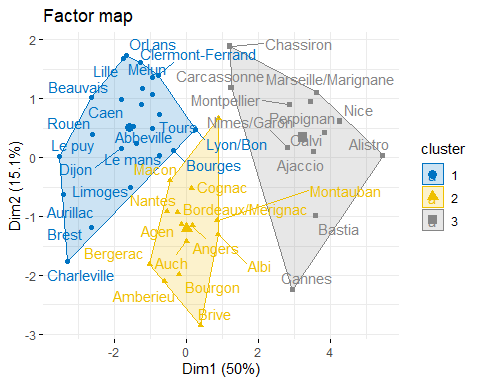
### Dendogramme des donnees   
  
fviz\_dend(res.hcpc,   
 cex = 0.7, # Taille du text  
 palette = "jco", # Palette de couleur ?ggpubr::ggpar  
 rect = TRUE, rect\_fill = TRUE, # Rectangle autour des groupes  
 rect\_border = "jco", # Couleur du rectangle  
 labels\_track\_height = 0.8 # Augment l'espace pour le texte  
)

## Warning: `guides(<scale> = FALSE)` is deprecated. Please use `guides(<scale> =  
## "none")` instead.



### Visualiser les individus et colorier par groupe   
fviz\_cluster(res.hcpc,  
 repel = TRUE, # Evite le chevauchement des textes  
 show.clust.cent = TRUE, # Montre le centre des clusters  
 palette = "jco", # Palette de couleurs, voir ?ggpubr::ggpar  
 ggtheme = theme\_minimal(),  
 main = "Factor map"  
)

## Warning: ggrepel: 8 unlabeled data points (too many overlaps). Consider  
## increasing max.overlaps



# Caractérisation des groupes   
  
BDD\_NEW = cbind(BDD\_NEW,HCPC)  
table(HCPC)

## HCPC  
## 1 2 3   
## 25 14 12

summaryBy(T+TM+Tm+PP+V+RA+SN+TS+FG ~ BDD\_NEW$HCPC, data=BDD\_NEW, FUN=mean) # moyennes

## HCPC T.mean TM.mean Tm.mean PP.mean V.mean RA.mean SN.mean TS.mean  
## 1 1 12.38800 17.72000 6.708000 722.2252 12.184000 161.20 12.880000 14.44  
## 2 2 13.95714 19.61429 8.385714 805.8543 9.514286 160.00 4.500000 8.50  
## 3 3 16.30833 21.13333 11.058333 860.7958 13.483333 120.25 2.833333 25.50  
## FG.mean  
## 1 53.280000  
## 2 57.357143  
## 3 9.583333

apply(BDD, 2, mean)

## T TM Tm PP V RA SN   
## 13.741176 19.043137 8.192157 777.787059 11.756863 151.235294 8.215686   
## TS FG   
## 15.411765 44.117647

summaryBy(T+TM+Tm+PP+V+RA+SN+TS+FG ~ BDD\_NEW$HCPC, data=BDD\_NEW, FUN=sd) # ecart-type

## HCPC T.sd TM.sd Tm.sd PP.sd V.sd RA.sd SN.sd  
## 1 1 0.8671409 1.0492060 1.0696261 142.0990 1.982145 14.13624 4.710980  
## 2 2 0.5445030 0.6837196 0.8411139 120.3241 2.306798 14.16388 2.377782  
## 3 3 1.0509375 1.5251428 1.6983727 136.3275 3.250967 24.53615 1.403459  
## TS.sd FG.sd  
## 1 9.129074 25.723724  
## 2 10.353372 23.886292  
## 3 10.370412 8.436483

apply(BDD, 2, sd)

## T TM Tm PP V RA SN   
## 1.786413 1.782387 2.109772 144.786262 2.794513 24.126822 5.852568   
## TS FG   
## 11.393290 29.307096

res.hcpc$desc.var$quanti

## $`1`  
## v.test Mean in category Overall mean sd in category Overall sd  
## SN 5.580970 12.8800 8.215686 4.6157990 5.794906  
## RA 2.892230 161.2000 151.235294 13.8506318 23.889113  
## FG 2.189287 53.2800 44.117647 25.2039997 29.018349  
## PP -2.687310 722.2252 777.787059 139.2280249 143.359762  
## Tm -4.926208 6.7080 8.192157 1.0480153 2.088985  
## TM -5.198421 17.7200 19.043137 1.0280078 1.764826  
## T -5.304458 12.3880 13.741176 0.8496211 1.768812  
## p.value  
## SN 2.391803e-08  
## RA 3.825181e-03  
## FG 2.857601e-02  
## PP 7.202999e-03  
## Tm 8.384082e-07  
## TM 2.009890e-07  
## T 1.130081e-07  
##   
## $`2`  
## v.test Mean in category Overall mean sd in category Overall sd  
## FG 1.984481 57.357143 44.117647 23.017407 29.018349  
## TS -2.664944 8.500000 15.411765 9.976759 11.281038  
## SN -2.788951 4.500000 8.215686 2.291288 5.794906  
## V -3.525247 9.514286 11.756863 2.222887 2.766980  
## p.value  
## FG 0.0472022664  
## TS 0.0077001219  
## SN 0.0052879118  
## V 0.0004230887  
##   
## $`3`  
## v.test Mean in category Overall mean sd in category Overall sd  
## T 5.692644 16.308333 13.741176 1.006196 1.768812  
## Tm 5.381593 11.058333 8.192157 1.626068 2.088985  
## TM 4.645458 21.133333 19.043137 1.460213 1.764826  
## TS 3.507595 25.500000 15.411765 9.928914 11.281038  
## V 2.447352 13.483333 11.756863 3.112564 2.766980  
## PP 2.271120 860.795833 777.787059 130.523692 143.359762  
## SN -3.643084 2.833333 8.215686 1.343710 5.794906  
## FG -4.667899 9.583333 44.117647 8.077317 29.018349  
## RA -5.087435 120.250000 151.235294 23.491577 23.889113  
## p.value  
## T 1.250873e-08  
## Tm 7.382959e-08  
## TM 3.393229e-06  
## TS 4.521767e-04  
## V 1.439101e-02  
## PP 2.313972e-02  
## SN 2.693913e-04  
## FG 3.042954e-06  
## RA 3.629378e-07

res.hcpc$desc.axes$quanti

## $`1`  
## v.test Mean in category Overall mean sd in category Overall sd  
## Dim.2 3.047095 0.5123401 1.864249e-15 0.8565158 1.165845  
## Dim.1 -5.110894 -1.5631131 -3.835712e-15 0.9520498 2.120615  
## p.value  
## Dim.2 2.310645e-03  
## Dim.1 3.206386e-07  
##   
## $`2`  
## v.test Mean in category Overall mean sd in category Overall sd  
## Dim.5 -3.465643 -0.6541807 -3.381010e-17 0.6087199 0.8210357  
## Dim.2 -4.514418 -1.2100248 1.864249e-15 0.8195145 1.1658447  
## p.value  
## Dim.5 5.289643e-04  
## Dim.2 6.349096e-06  
##   
## $`3`  
## v.test Mean in category Overall mean sd in category Overall sd  
## Dim.1 5.990642 3.238858 -3.835712e-15 1.124616 2.120615  
## p.value  
## Dim.1 2.090147e-09

res.hcpc$desc.ind$para

## Cluster: 1  
## Troyes Pau Poitiers St Dizier Dijon   
## 0.8167394 0.9277131 0.9277131 1.0568530 1.0694338   
## ------------------------------------------------------------   
## Cluster: 2  
## Agen Bourgon Angers Cognac Albi   
## 0.9783676 1.0959068 1.3169690 1.5208297 1.5592079   
## ------------------------------------------------------------   
## Cluster: 3  
## Marseille/Marignane Montpellier Perpignan Nice   
## 1.000831 1.030683 1.140735 1.207462   
## Nimes/Garon   
## 1.338148

sort(res.hcpc$data.clust$clust,decreasing = FALSE)

## [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2  
## [39] 2 3 3 3 3 3 3 3 3 3 3 3 3  
## Levels: 1 2 3

res.hcpc$data.clust

## T TM Tm PP V RA SN TS FG clust  
## Dijon 12.5 18.1 6.1 745.41 11.1 161 14 23 81 1  
## Clermont-Ferrand 12.9 18.9 6.2 513.07 10.0 168 10 31 27 1  
## Cognac 14.2 19.7 7.7 858.25 12.5 173 3 18 56 2  
## Bourgon 13.6 19.3 8.6 827.73 7.5 155 4 0 58 2  
## Le puy 10.1 15.7 4.7 784.99 8.9 139 24 0 39 1  
## Lille 12.0 16.8 7.0 596.53 14.9 156 12 16 56 1  
## Lyon/Bon 14.2 19.6 8.3 797.55 10.9 152 15 27 28 1  
## Macon 13.6 18.7 9.0 704.57 10.3 149 11 0 43 2  
## Marseille/Marignane 16.6 21.8 10.4 771.65 15.7 101 2 24 9 3  
## Limoges 12.5 17.3 7.8 1011.37 12.8 167 15 16 71 1  
## Le mans 13.3 18.6 7.0 749.21 9.8 167 11 20 37 1  
## Montpellier 16.1 21.2 9.9 865.58 14.6 120 6 33 15 3  
## Nantes 13.3 18.5 7.9 995.06 12.0 174 7 18 60 2  
## Nice 17.2 20.7 14.0 896.12 13.9 119 3 33 1 3  
## Nimes/Garon 15.7 21.6 9.7 915.78 13.0 129 3 36 25 3  
## Montauban 14.7 20.4 9.5 632.71 8.0 135 4 0 44 2  
## Melun 12.6 18.1 6.2 567.94 12.5 156 14 21 34 1  
## Orl‚ans 12.5 18.1 6.2 564.88 14.6 155 15 15 53 1  
## Pau 13.1 18.6 7.6 704.00 13.2 161 11 14 42 1  
## Toulouse /Blagnac 14.7 19.9 9.2 690.83 13.2 166 4 24 37 2  
## St Dizier 12.6 18.4 5.6 680.87 10.7 170 12 20 42 1  
## Tours 13.1 18.3 7.7 654.99 11.8 175 9 12 38 1  
## Troyes 12.6 18.4 6.4 677.76 12.0 157 14 20 44 1  
## Strasbourg 12.5 18.3 6.0 567.87 10.5 151 12 30 53 1  
## Rouen 11.6 16.6 6.1 792.30 13.9 177 13 9 68 1  
## Poitiers 13.1 18.6 7.6 704.00 13.2 161 11 14 42 1  
## Perpignan 16.8 21.8 11.3 758.67 15.6 130 1 24 12 3  
## Paris/orly 13.1 18.0 8.5 681.92 12.9 176 16 24 33 1  
## Abbeville 11.7 16.1 7.8 723.68 13.2 146 5 0 46 1  
## Agen 14.2 20.1 7.2 795.96 9.5 168 4 16 68 2  
## Ajaccio 16.7 22.0 10.4 802.57 9.7 123 3 38 4 3  
## Albi 14.5 20.2 9.3 747.17 8.3 147 3 0 28 2  
## Amberieu 13.1 19.0 7.8 992.61 9.1 155 7 0 60 2  
## Alistro 18.0 22.4 14.4 835.54 13.8 56 2 9 0 3  
## Angers 13.3 18.5 8.7 859.30 10.0 146 3 0 29 2  
## Auch 14.1 20.0 8.9 610.00 6.9 158 3 0 57 2  
## Aurillac 11.3 17.0 5.3 931.67 10.2 172 24 11 81 1  
## Bastia 16.6 21.5 11.6 1043.31 10.1 121 4 29 8 3  
## Beauvais 11.6 16.9 5.7 659.76 12.1 158 19 14 59 1  
## Bergerac 13.5 20.0 6.6 886.99 8.8 185 5 21 101 2  
## Brest 11.8 15.9 7.2 1133.96 16.3 197 7 5 107 1  
## Brive 14.2 20.4 8.5 900.05 5.2 152 3 0 51 2  
## Bourges 13.4 18.7 8.5 661.37 11.0 134 12 0 34 1  
## Bordeaux/Merignac 14.4 19.9 8.5 780.73 11.9 177 2 22 111 2  
## Caen 11.8 16.5 6.8 681.51 14.5 166 8 10 51 1  
## Cannes 16.2 21.5 10.0 1187.88 8.0 139 3 21 2 3  
## Chassiron 14.1 16.7 11.9 730.12 19.6 132 2 6 18 3  
## Calvi 16.9 22.2 10.3 777.20 11.5 115 1 34 1 3  
## Charleville 11.2 17.0 5.1 834.08 8.9 136 14 0 135 1  
## Carcassonne 14.8 20.2 8.8 745.13 16.3 158 4 19 20 3  
## Chateauroux 12.6 18.5 6.3 634.94 14.7 172 5 9 31 1