T. D. n° 3
Classification ascendante hiérarchique et la méthode des K-moyennes

Résumé

Ce document est le T.D. n° 3 du module Analyse exploratoire. Il reprend rapidement des éléments du cours et propose une mise en pratique de deux approches : la classification ascendante hiérarchique (CAH) avec *hclust()* et la méthode des K-moyenne (k-means) avec *kmeans()*.

1 Football

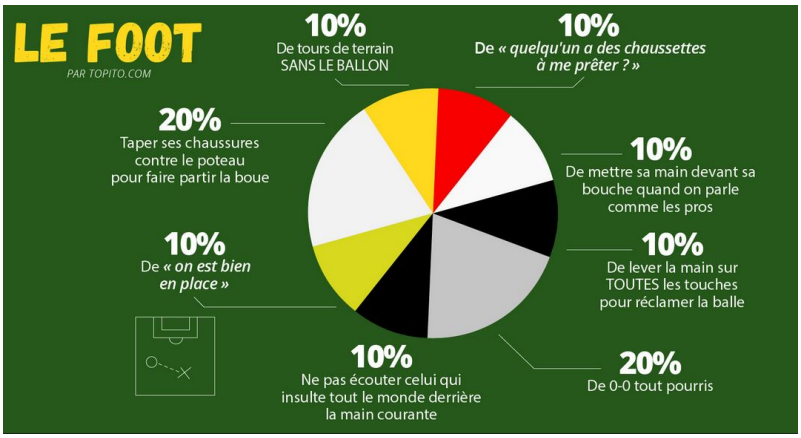


FIGURE 1 – Infographie sur mon appréciation du football
source :`www.topito.com`

1.1 Chargement des données

Commencez par charger les données du fichier *events_France.csv*. Associez ces données à un dataframe et appliquez la fonction *summary()*. Il est également possible de télécharger le fichier *events_France.csv* depuis la plate-forme Kaggle : `https://www.kaggle.com/secareanualin/football-events` [1]. Ce fichier de données fournit une vue granulaire de 208 446 matchs de football jouées en France dans la saison 2011/2012 à la saison 2016/2017 à partir du 25.01.2017. Certains matchs qui ont été joués pendant ces saisons contiennent cependant des données manquantes (environ 10%). Parmi les informations disponibles nous avons : [2]

1. j'ai cependant "nettoyé" les données
2. je n'aime pas le football donc ne râlez pas pour mon vocabulaire d'amateur

— *fthg* but marqué par l'équipe
— *ftag* but marqué par l'équipe adverse
— *odd_h* côte de l'équipe
— *sort_order* ordre chronologique de l'évènement.
— *time* temps du match.

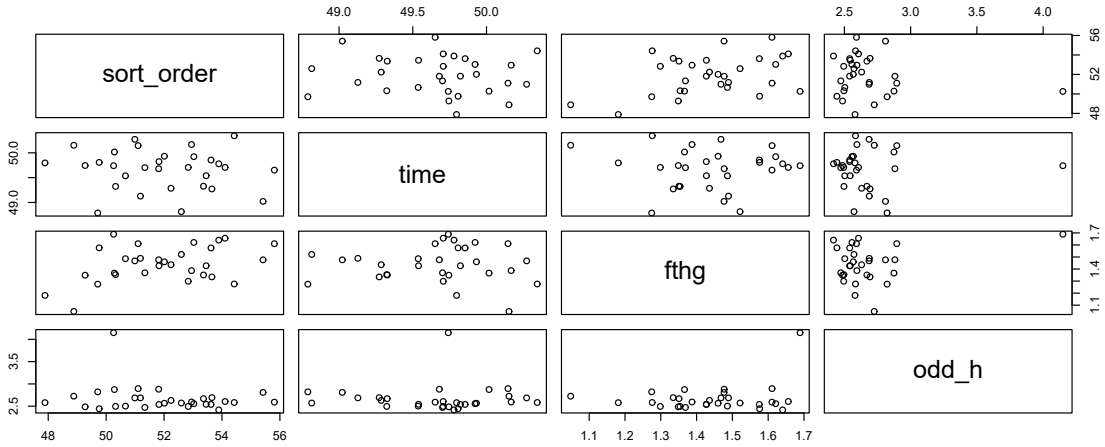
Commencez par sélectionner les colonnes suivantes dans un dataframe : *fthg*, *ftag* , *odd_h*, *sort_order* et *time*. Les équipes ont réalisés plusieurs évènements, il y a donc des doublons. Vous allez concaténer les données en utilisant la moyenne. Affichez ensuite un croisement deux à deux pour chaque variable quantitative grâce à la fonction *pairs()*. Créez un sous ensemble de données composées uniquement des variables quantitatives.

```
> data_event <- read.csv("C:/Users/claey/Documents/Cours/cour ESIEA
   /5A 2017-2018 DTE/Analyse exploratoire/TD3/data/football-events/
   football-events/events_France.csv")

> print(head(data_event))
    X   id_odsp  id_event sort_order time
1 321 00nmICd9/ 00nmICd91          1    3
2 322 00nmICd9/ 00nmICd92          2    3
3 323 00nmICd9/ 00nmICd93          3    7
4 324 00nmICd9/ 00nmICd94          4    7
5 325 00nmICd9/ 00nmICd95          5    7
6 326 00nmICd9/ 00nmICd96          6    7
                                                                  text
                                                                     event_type
1                                  Foul by Juan Manuel FalcA  n (
   Metz).          3
2 TiemouA   Bakayoko (Monaco) wins a free kick in the defensive
   half.          8
3                                     Foul by Anthony Martial (Monaco
   ).          3
4     Sylvain Marchal (Metz) wins a free kick in the defensive
   half.          8
5                                       Foul by Cheick Doukoure (Metz
   ).          3
6 TiemouA   Bakayoko (Monaco) wins a free kick in the defensive
   half.          8
  event_type2 side event_team  opponent             player player2
     player_in
1          NA    1       Metz AS Monaco juan manuel falcon    <NA>
        <NA>
2          NA    2  AS Monaco      Metz   tiemoue bakayoko    <NA>
        <NA>
3          NA    2  AS Monaco      Metz    anthony martial    <NA>
        <NA>
4          NA    1       Metz AS Monaco    sylvain marchal    <NA>
        <NA>
5          NA    1       Metz AS Monaco    cheick doukoure    <NA>
        <NA>
6          NA    2  AS Monaco      Metz   tiemoue bakayoko    <NA>
```

```
        <NA>
  player_out shot_place shot_outcome is_goal location bodypart
     assist_method
1        <NA>          NA            NA        0        NA        NA
                 0
2        <NA>          NA            NA        0         2        NA
                 0
3        <NA>          NA            NA        0        NA        NA
                 0
4        <NA>          NA            NA        0         2        NA
                 0
5        <NA>          NA            NA        0        NA        NA
                 0
6        <NA>          NA            NA        0         2        NA
                 0
  situation fast_break
                                                          link_odsp
1        NA          0 /soccer/france/ligue-1-2014-2015/metz-monaco
   -00nmICd9/
2        NA          0 /soccer/france/ligue-1-2014-2015/metz-monaco
   -00nmICd9/
3        NA          0 /soccer/france/ligue-1-2014-2015/metz-monaco
   -00nmICd9/
4        NA          0 /soccer/france/ligue-1-2014-2015/metz-monaco
   -00nmICd9/
5        NA          0 /soccer/france/ligue-1-2014-2015/metz-monaco
   -00nmICd9/
6        NA          0 /soccer/france/ligue-1-2014-2015/metz-monaco
   -00nmICd9/
  adv_stats       date league season country   ht         at fthg
     ftag odd_h odd_d
1      TRUE 2014-12-20     F1   2015  france Metz AS Monaco    0
      1  4.45  3.55
2      TRUE 2014-12-20     F1   2015  france Metz AS Monaco    0
      1  4.45  3.55
3      TRUE 2014-12-20     F1   2015  france Metz AS Monaco    0
      1  4.45  3.55
4      TRUE 2014-12-20     F1   2015  france Metz AS Monaco    0
      1  4.45  3.55
5      TRUE 2014-12-20     F1   2015  france Metz AS Monaco    0
      1  4.45  3.55
6      TRUE 2014-12-20     F1   2015  france Metz AS Monaco    0
      1  4.45  3.55
  odd_a odd_over odd_under odd_bts odd_bts_n
1  2.14       NA        NA      NA        NA
2  2.14       NA        NA      NA        NA
3  2.14       NA        NA      NA        NA
4  2.14       NA        NA      NA        NA
5  2.14       NA        NA      NA        NA
6  2.14       NA        NA      NA        NA
>


> list_col <- c("event_team","sort_order","time","fthg","odd_h")
```

```
> data_event.x <- data_event[,list_col]

#Consolidate
> data_event.c <- aggregate(.~event_team,data=data_event.x,FUN=mean
    )
#Change index
> rownames(data_event.c ) <- data_event.c$event_team
> data_event.c$event_team <- NULL


> pairs(data_event.c)
```

FIGURE 2 – Croisement deux à deux des variables quantitatives



À vous !

a) Omettez les variables avec NA au dataframe *royau.x*.

b) Centrez et réduisez les variables grâce à la fonction *scale()*, stockez ce résultat dans *events_France.csv*.

c) Est-il nécessaire d'utiliser un centrage-réduction sur les variables ? Justifiez.

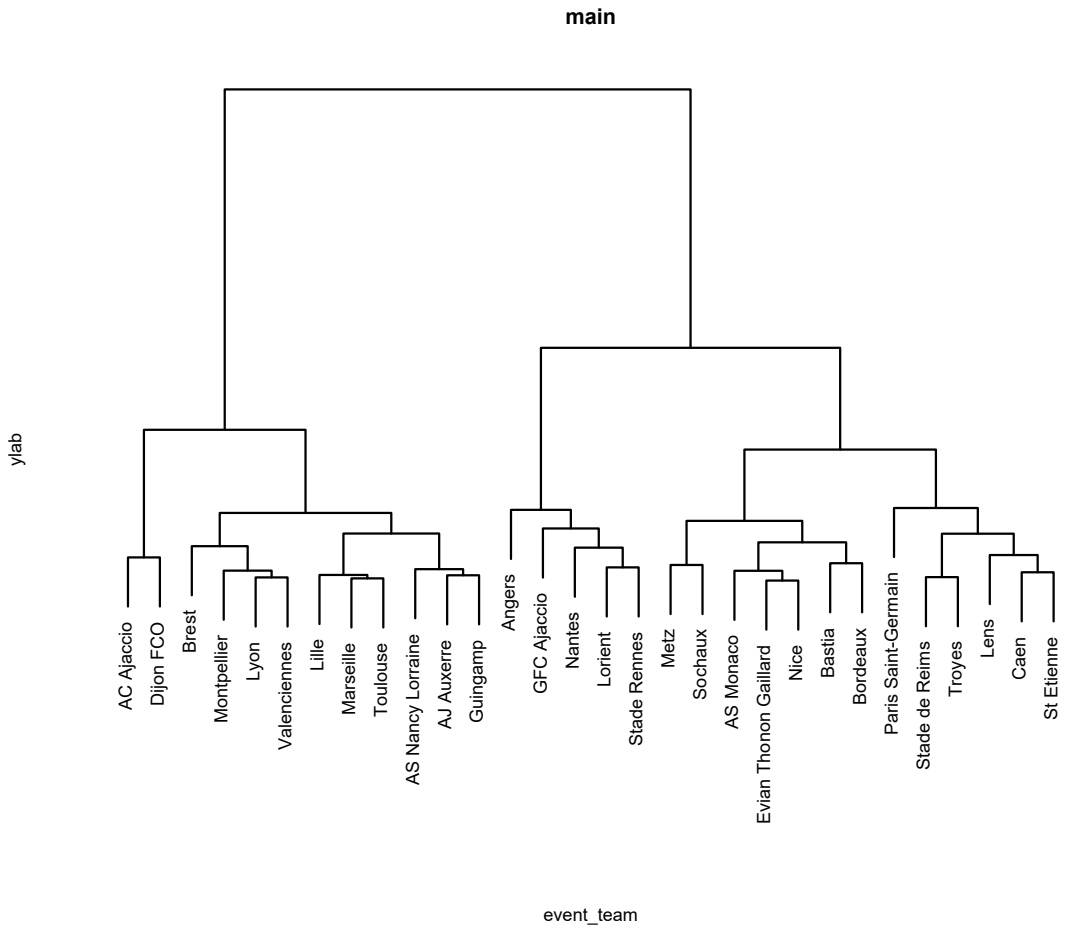
1.2 CAH

La fonction *dist()* calcule et renvoie la matrice des distances calculées en utilisant une mesure de distance spécifiée (ou par défaut la distance L2) pour calculer les distances entre les lignes d'une matrice de données. Appliquez cette fonction à votre dataframe. *events_France.csv*.
La méthode suppose qu'on dispose d'une mesure de dissimilarité entre les individus, dans le cas de points situés dans un espace L2, on peut utiliser la distance comme mesure de dissimilarité. La classification ascendante hiérarchique est dite ascendante car elle part d'une situation où tous les individus sont seuls dans une classe, puis sont rassemblés en classes de plus en plus grandes. Le qualificatif hiérarchique vient

du fait qu'elle produit une hiérarchie H à l'ensemble des classes à toutes les étapes de l'algorithme. Dans ce TD nous utilisons la méthode de Ward pour séparer des individus en classes. La méthode de Ward propose qu'à chaque pas, on cherche à obtenir un minimum local de l'inertie intraclasse ou un maximum de l'inertie interclasse. En d'autres termes, elle consiste à réunir les deux clusters dont le regroupement fera le moins baisser l'inertie interclasse. On suppose tout de même l'existence de distances euclidiennes entre observations. Cette technique tend à regrouper les petites classes entre elles.

```
> d.data_event <- dist(data_event.c)
> cah.ward <- hclust(d.data_event,method="ward.D2")
> par(cex=0.5)
> plot(cah.ward, xlab="", ylab="", main="", sub="", axes=FALSE)
> title(xlab="event_team", ylab="ylab", main="main")
```

FIGURE 3 – Classification ascendante hiérarchique



On décide constituer quatre regroupements de 8 classes.

```
groupes.cah <- cutree(cah.ward,k=8)
```

À vous !

a) Proposez un label à la place de "*ylab*".

b) Proposez une autre visualisation de *cah.ward*.

c) Affichez les 8 groupes en utilisant la fonction *rect.hclust()*.

d) Affichez le contenue de chaque groupe.

1.3 K-mean

La méthode dite *K-means* propose de choisir aléatoirement un point comme le barycentre de chaque groupe puis de le recalculer à chaque nouvel individu introduit dans le groupe. Cette technique est intéressante car elle s'adapte lorsqu'on injecte de nouveaux individus. *K-means*, à la différence de la CAH, ne fournit pas d'outil d'aide à la détection du nombre de classes. Nous devons les programmer sous R ou utiliser des procédures proposées par des packages dédiés. En faisant varier le nombre de groupes on observe l'évolution d'un indicateur de l'aptitude des individus à être plus proches entre eux dans un même groupe, que des individus des autres groupes. Cet indicateur traduit de la qualité d'une solution.

```
> groupes.kmeans <- kmeans(data_event.c,centers=8,nstart=5)
> print(groupes.kmeans)
K-means clustering with 8 clusters of sizes 3, 9, 4, 2, 3, 6, 2, 1

Cluster means:
  sort_order     time     fthg    odd_h
1   49.58147 49.44663 1.399864 2.583780
2   53.60890 49.76797 1.474341 2.578264
3   50.87762 49.42233 1.424485 2.539700
4   55.61308 49.33683 1.543583 2.700434
5   50.79262 50.14490 1.481729 2.818470
6   52.21951 49.54022 1.436725 2.613912
7   48.38580 49.97474 1.114779 2.652461
8   50.25739 49.74082 1.689065 4.149199

Clustering vector:
           AC Ajaccio             AJ Auxerre                  Angers
                       AS Monaco     AS Nancy Lorraine
                    4                      2                      7
                                           6                      6
                Bastia              Bordeaux                 Brest
                                    Caen             Dijon FCO
                    3                      3                      2
                                           3                      4
Evian Thonon Gaillard           GFC Ajaccio               Guingamp
                    Lens                 Lille
                    6                      1                      2
                                           5                      2
              Lorient                   Lyon             Marseille
                                  Metz           Montpellier
```

```
                    1                         2                         2
                                              6                         2
              Nantes                      Nice    Paris Saint-Germain
                                  Sochaux               St Etienne
                    7                         6                         8
                                              6                         3
       Stade de Reims            Stade Rennes                 Toulouse
                           Troyes            Valenciennes
                    5                         1                         2
                                              5                         2

Within cluster sum of squares by cluster:
[1] 0.9305152 3.0960453 0.8861214 0.3077104 0.4822524 1.8824301
    0.5782352 0.0000000
 (between_SS / total_SS =  93.1 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "
    tot.withinss" "betweenss"    "size"
[8] "iter"         "ifault"
> print(table(groupes.cah,groupes.kmeans$cluster))

groupes.cah 1 2 3 4 5 6 7 8
          1 0 0 0 2 0 0 0 0
          2 0 5 0 0 0 1 0 0
          3 0 0 0 0 0 0 1 0
          4 0 0 2 0 0 5 0 0
          5 0 4 0 0 0 0 0 0
          6 0 0 2 0 3 0 0 0
          7 3 0 0 0 0 0 1 0
          8 0 0 0 0 0 0 0 1
```

Le groupe 4 de la CAH coïncide avec le groupe 6 des K-Means. Il y a certes des correspondances, mais elles ne sont pas toujours exactes, c'est pourquoi l'on peut avoir des résultats différents entre la méthode CAH et la méthode K-Means.

À vous !

a) Cherchez la signification du paramètre *centers* de la fonction *kmeans()*.

b) Cherchez la signification du paramètre *nstart* de la fonction *kmeans()*.

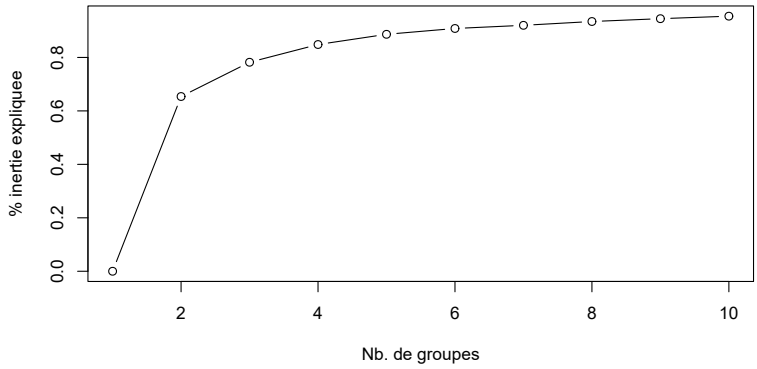
1.4 Détection des groupes pour le K-mean

On propose d'observer l'évolution de la proportion d'inertie pour différentes partitions, on cherche le point où la tangente devient horizontale dans le graphique (plus d'évolution).

```
> inertie.expl <- rep(0,times=10)
> for (k in 2:10){
+   clus <- kmeans(data_event.c,centers=k,nstart=5)
+   inertie.expl[k] <- clus$betweenss/clus$totss
```

```
+ }
> par(cex=1)
> plot(1:10,inertie.expl,type="b",xlab="Nb. de groupes",ylab="%
    inertie expliquee")
```

FIGURE 4 – Pourcentage d'inertie selon les groupes



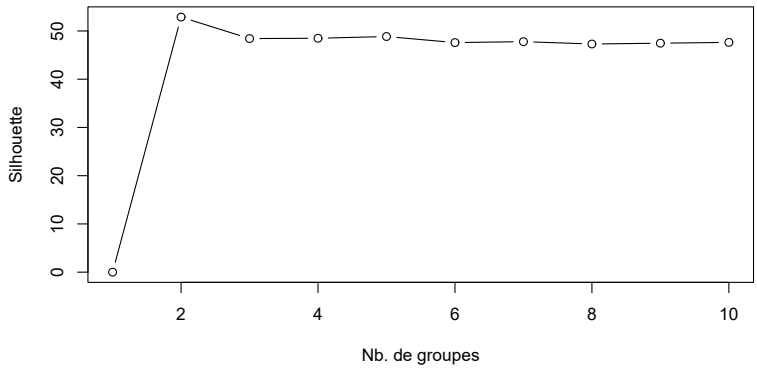
La largeur moyenne de la silhouette est la moyenne de s_i de tous les objets i dans les données, c'est à dire la largeur moyenne de la silhouette pour k classes. Ceci peut être utilisé pour sélectionner le "meilleur" nombre de classes, en choisissant le k qui donne la valeur la plus élevée de la moyenne de s_i.

À vous !

a) Utilisez la largeur moyenne de silhouette maximale avec la fonction *kmeansruns()* de la librairie *fpc*. Affichez ce second critère de façon graphique.

b) Interprétez ces résultats.

FIGURE 5 – Silhouette selon les groupes



1.5 Statistiques comparatives

Comparez les moyennes des variables actives conditionnellement aux groupes. Pour cela, quantifiez globalement l'amplitude des écarts avec la proportion de variance expliquée en créant la fonction *stat.com()* renvoyant :

- Le nombre de groupes.
- Le nombre d'observations.
- La moyenne globale.
- La variabilité totale.
- Les effectifs conditionnels.
- La moyennes conditionnelles.
- La variabilité expliquée.
- Les éléments du vecteur.

```
> stat.comp <- function(x,y){
+   K <- length(unique(y))
+   n <- length(x)
+   m <- mean(x)
+   TSS <- sum((x-m)^2)
+   nk <- table(y)
+   mk <- tapply(x,y,mean)
+   BSS <- sum(nk * (mk - m)^2)
+   result <- c(mk,100.0*BSS/TSS)
+   names(result) <- c(paste("G",1:K),"% epl.")
+   return(result)
+ }
```

À vous !

a) Appliquez la fonction *stat.comp()* aux variables de la base originale *data_ event.c* et non pas aux variables centrées réduites.

b) Interprétez ces résultats.

1.6 ACP et CAH

Complétez l'analyse CAH avec l'ACP pour tenir compte des liaisons entre les variables. Affichez sur le graphique les groupes trouvés grâce à la classification ascendante hiérarchique précédente.

```
> acp <- princomp(data_event.c,cor=T,scores=T)
> plot(1:4,acp$sdev^2,type="b",xlab="Nb. de facteurs",ylab="Val.
   Propres")
> biplot(acp,cex=0.65)
> plot(acp$scores[,1],acp$scores[,2],type="n")
> text(acp$scores[,1],acp$scores[,2],col=c("red","green","blue","
   black")[groupes.cah],cex=0.65,labels=rownames(data_event.c))
```

FIGURE 6 – Valeurs propres selon le nombre de facteurs

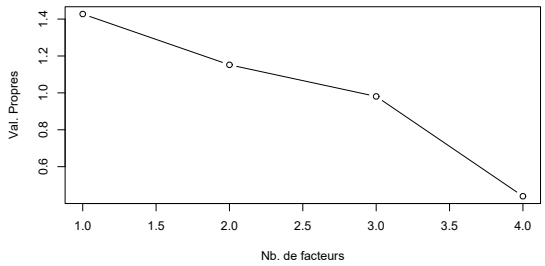


FIGURE 7 – Analyse en composantes principales

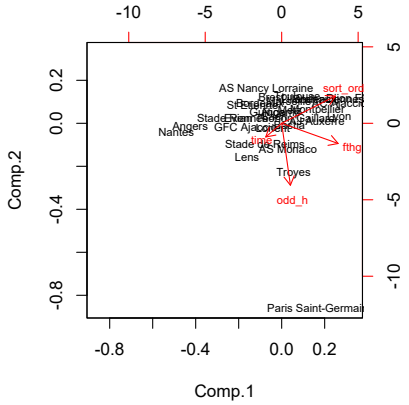
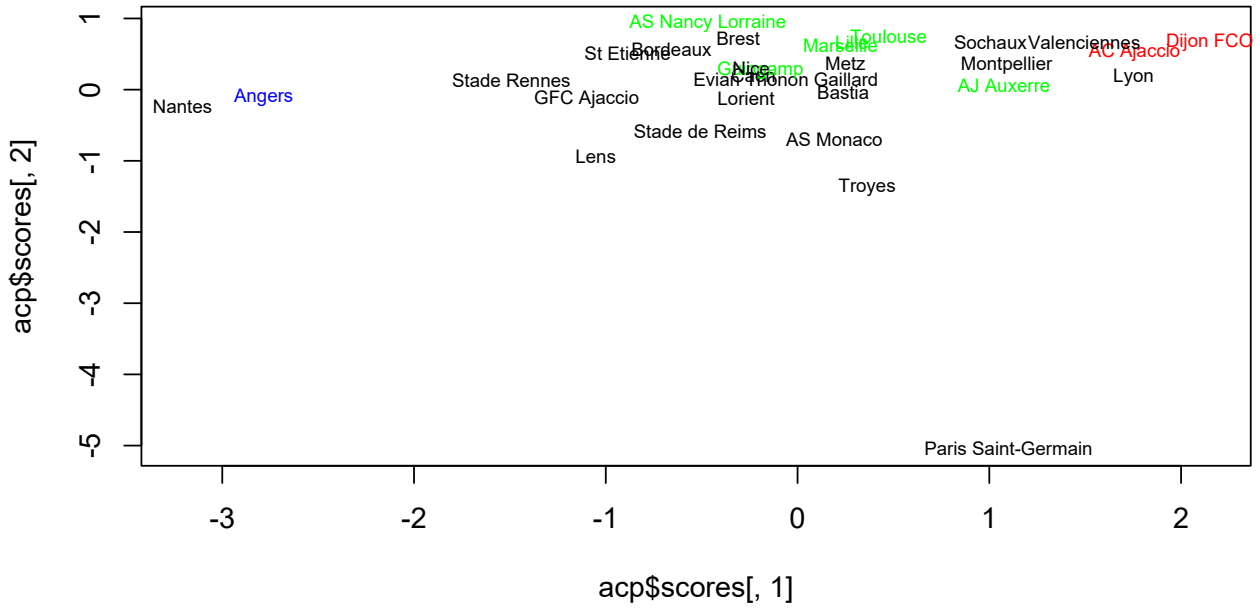


FIGURE 8 – Groupe de la CAH sur l'ACP



À vous !

a) Que pouvez-vous dire sur l'équipe d'Angers ? Celle du P.S.G (d'un point de vue mathématique) ?

b) Recommencer l'analyse de ce TD en intégrant les variables *season*,*ftag*,*odd_ d*,*odd_ a*. Présentez vos résultats.

c) Concluez sur l'interprétation des données.