Rapport du projet d'analyse de données

Partie 1

Projet Régression Linéaire Multiple

On peut vouloir expliquer les points obtenus dans une compétition de jeux olympiques, en fonction des performances des athlètes.

Notre jeu de données Décathlon contient les performances réalisées par des athlètes lors d'une compétition.

Dans ce jeu de données, il y a 55 lignes et 11 colonnes :

- Les colonnes 2 à 11 sont des variables quantitatives (les variables explicatives), correspondent aux performances des athlètes pour les dix épreuves du Décathlon (Longueur, 100m, Poids, Hauteur, 400m, 110m.haies, Perche, Javelot, 1500m et Disque)
- La 1ère colonne est une variable quantitative (variable dépendante) qui correspond au nombre de points obtenus par chaque athlète.

Nous allons faire une régression linéaire multiple sur notre jeu de données.

1) Calculer le modèle de régression linéaire multiple incluant toute les variables explicatives

On va commencer tout d'abord par importer et lire les données :

```
Decathlon <- read_excel("C:/Users/hp/Desktop/Decathlon.xlsx")
```

Il est important de s'assurer que l'importation a bien été effectuée, et notamment que les variables quantitatives sont bien considérées comme quantitatives :

```
> summary(Decathlon)
     Points        Longueur          100m           Poids          Hauteur           400m      
 Min.   :7230   Min.   :6.580   Min.   :10.28   Min.   :12.84   Min.   :1.630   Min.   :46.07  
 1st Qu.:7978   1st Qu.:7.130   1st Qu.:10.70   1st Qu.:13.92   1st Qu.:1.905   1st Qu.:47.99  
 Median :8237   Median :7.440   Median :10.91   Median :14.65   Median :2.050   Median :48.81  
 Mean   :8218   Mean   :7.384   Mean   :10.90   Mean   :14.66   Mean   :2.031   Mean   :48.83  
 3rd Qu.:8458   3rd Qu.:7.665   3rd Qu.:11.14   3rd Qu.:15.41   3rd Qu.:2.185   3rd Qu.:49.70  
 Max.   :9126   Max.   :7.960   Max.   :11.43   Max.   :16.36   Max.   :2.330   Max.   :52.67  
   110m.haies        Perche         Javelot          1500m           Disque     
 Min.   :13.30   Min.   :4.350   Min.   :49.45   Min.   :254.6   Min.   :35.30  
 1st Qu.:14.04   1st Qu.:4.755   1st Qu.:57.44   1st Qu.:267.8   1st Qu.:42.13  
 Median :14.56   Median :4.890   Median :63.19   Median :274.2   Median :44.75  
 Mean   :14.49   Mean   :4.909   Mean   :62.21   Mean   :274.9   Mean   :43.93  
 3rd Qu.:14.93   3rd Qu.:5.115   3rd Qu.:66.19   3rd Qu.:280.4   3rd Qu.:45.52  
 Max.   :15.63   Max.   :5.480   Max.   :72.32   Max.   :304.5   Max.   :51.65  
```

Maintenant on va aborder le problème de sélection des variables explicatives, on a toujours un souci d'obtenir le meilleur modèle de régression linéaire multiple, pour cela nous allons commencer par calculer le modèle de régression linéaire multiple qui comporte toutes les variables explicatives dont on dispose (les 10 épreuves de Decathlon),donc on va exécuter la commande lm( ) :

```
myregmult <- lm(Points~.,data=Decathlon)
summary(myregmult)
```

c'est le **modèle global** : avec Points c'est la variables dépendante qui a été choisis. (toutes les variables explicatives qui existent dans Decathlon vont être intégrer dans le modèle de régression linéaire multiple).

```
> myregmult <- lm(Points~.,data=Decathlon)
> summary(myregmult)

Call:
lm(formula = Points ~ ., data = Decathlon)

Residuals:
    Min      1Q  Median      3Q     Max 
-641.99 -142.85    1.77  130.24  546.21 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) 4182.798   2802.478   1.493 0.142695    
Longueur     491.480    115.358   4.260 0.000106 ***
`100m`      -127.931    137.018  -0.934 0.355565    
Poids        141.804     42.197   3.361 0.001617 ** 
Hauteur       -4.576    226.850  -0.020 0.983998    
`400m`        46.753     37.315   1.253 0.216848    
`110m.haies` -195.059    75.920  -2.569 0.013658 *  
Perche       499.796    139.495   3.583 0.000845 ***
Javelot      -13.595      7.252  -1.875 0.067486 .  
`1500m`      -11.384      4.960  -2.295 0.026532 *  
Disque        40.867     11.676   3.500 0.001079 ** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 264.8 on 44 degrees of freedom
Multiple R-squared:  0.7165,	Adjusted R-squared:  0.6521 
F-statistic: 11.12 on 10 and 44 DF,  p-value: 3.814e-09
```

Donc là on a les différentes caractéristiques de cet objet « myregmult », nous avons des statistiques sur le résidu, le minimum, le 1er quartile, la médiane, le 3ème quartile et le maximum. Ces statistiques nous donnent une idée sur la dispersion du résidu.

Ensuite nous avons les coefficients, avec les paramètres estimées, et les p_values, donc les variables explicatives qui ont un p_value < 0.05, sont significatives, et ceux qui ont un p_value > 0.05 ne sont pas significatives.

Donc on remarque bien que les variables explicatives Longueur, Perche, Poids, Disque, 110m.haies et 1500m sont des variables explicatives significatives.

1.1) Existe-t-il des variables explicatives non significatives ?

Il existe des variables non significatives : 100m, Hauteur, 400m et Javelot, car ils ont un p_value > 0.05, et leurs intervalle de confiance contient le 0.

```
> #Intervalles de confiance des variables explicatives
> confint(myregmult)
                   2.5 %      97.5 %
(Intercept)   -1465.22636 9830.821446
Longueur        258.99062  723.968972
`100m`         -404.07386  148.211292
Poids            56.76119  226.846220
Hauteur        -461.76107  452.609291
`400m`          -28.45029  121.956818
`110m.haies`   -348.06621  -42.051502
Perche          218.66349  780.929481
Javelot         -28.21042    1.020320
`1500m`         -21.37987   -1.389091
Disque           17.33507   64.399265
```

1.2) Donner la valeur de R^2 et R^2_{ajus}

En regardant l'exécution de la même commande summary(muregmult), on trouve :

- Multiple R-squared qui est le R^2= 0.7165. On a une qualité de modélisation très faible de 7.2% (la somme des carrés expliqués occupe 7.2% de la somme des carrés totaux, c'est-à-dire la somme des carrés résiduels est à la voisine des 92.8%)
- Adjusted R-squared qui est le R^2_{ajus}= 0.6521.

1.3) Le test de Fisher est-il significatif ? Que signifie ce test ?

En regardant toujours l'exécution de la même commande summary(myregmult), on trouve : le p_value du test de Fisher = 3.814 e-9, qui est inférieur à 0.05. Donc le test de Fisher est globalement significatif. On a ce résultat parce que nous avons le nombre d'observation plus grand que les nombre de variables, alors le nombre grand d'observations fait en sorte un F qui est significatif bien qu'un R^2 qui est à peine égale à 7.2%.Il signifie qu'on rejette l'hypothèse nulle H0 et accepter l'hypothèse H1, c'est-à-dire il existe une variation de la variable dépendante en fonction des variables explicatives.

2) Améliorer le modèle initiale par la procédure step, que remarquez-vous ?

Maintenant on va faire une procédure de sélection, step ( ), c'est une procédure qui existe sur R, qui est basé sur le critère AIC, donc on met step (modèle de regression qui a été déjà calculé), on exécute la commande :

```
> #Amélioration du modèle
> step(myregmult)
Start:  AIC=623.42
Points ~ Longueur + `100m` + Poids + Hauteur + `400m` + `110m.haies` +
    Perche + Javelot + `1500m` + Disque

               Df Sum of Sq     RSS    AIC
- Hauteur       1        29 3085486 621.42
- `100m`        1     61131 3146589 622.50
- `400m`        1    110084 3195541 623.35
<none>                      3085458 623.42
- Javelot       1    246444 3331901 625.64
- `1500m`       1    369490 3454948 627.64
- `110m.haies`  1    462895 3548353 629.11
- Poids         1    791915 3877373 633.98
- Disque        1    859020 3944478 634.93
- Perche        1    900198 3985656 635.50
- Longueur      1   1272862 4358320 640.41
```

Ici, il nous a donné le critère AIC du modèle global, c'est-à-dire lorsqu'il ne retire rien.

Ensuite, il nous donne le critère AIC lorsqu'il retire la variable Hauteur, puis lorsqu'il retire la variable 100m, puis lorsqu'il retire la variable 400m. Donc le faite de retirer ces 3 variables c'est là où on obtient le critère AIC le plus faible (ce sont les mêmes variables qui ne sont pas significatives à l'exception de la variable Javelot, elle avait une p. value de 0.067486 qui reste très proche de 0.05).

```
Step:  AIC=620.09
Points ~ Longueur + Poids + `110m.haies` + Perche + Javelot +
    `1500m` + Disque

               Df Sum of Sq     RSS    AIC
<none>                      3238911 620.09
- Javelot       1    247519 3486430 622.14
- `1500m`       1    336862 3575773 623.53
- `110m.haies`  1    472082 3710992 625.57
- Poids         1    672897 3911808 628.47
- Perche        1    936974 4175885 632.06
- Disque        1   1171942 4410852 635.07
- Longueur      1   1444674 4683584 638.37

Call:
lm(formula = Points ~ Longueur + Poids + `110m.haies` + Perche +
    Javelot + `1500m` + Disque, data = Decathlon)
```

En retirant les 3 variables : 100m, 400m et Hauteur, il va calculer un nouveau modèle sans ces 3 variables. Donc le meilleur modèle ça va être ce modèle, Points comme variables dépendante et 7 variables explicatives à savoir : Longueur, Poids, 110m.haies, Perche, Javelot, 1500m et Disque.

2.1) faites les tests de validation pour le modèle amélioré de la procédure de step : test d'homoscédasticité, test de normalité (shapiro et ks) et recherche de valeurs aberrantes

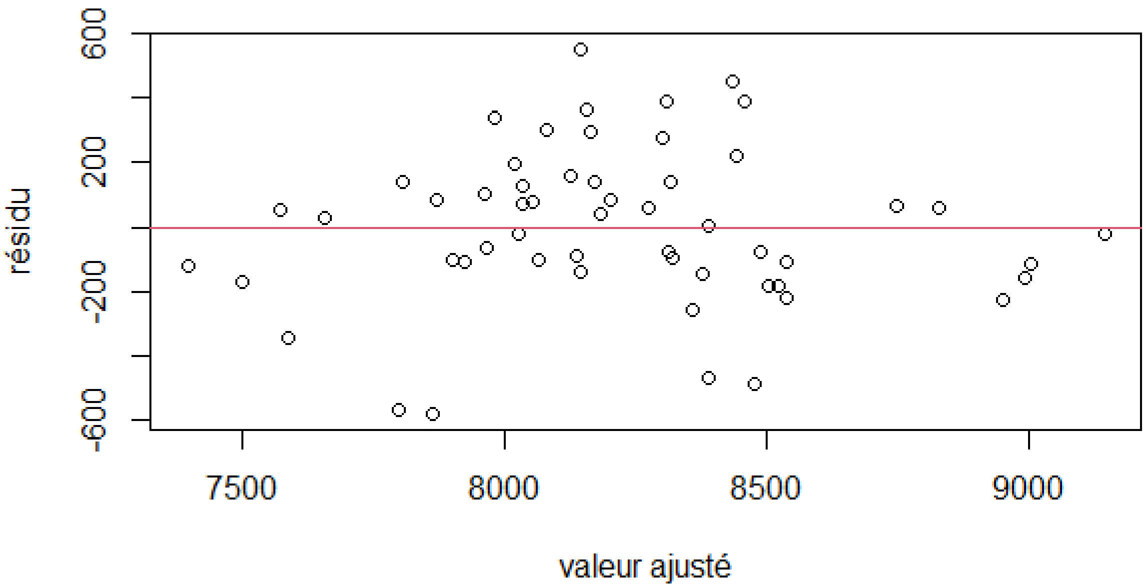
Maintenant le nouveau modèle « myregmult1 », c'est le modèle issu de la procédure de la variable explicative :

```
> myregmult1<-lm(Points~Longueur + Poids + `110m.haies` + Perche + Javelot + `1500m` +
 Disque, data = Decathlon)
> summary(myregmult1)

Call:
lm(formula = Points ~ Longueur + Poids + `110m.haies` + Perche +
    Javelot + `1500m` + Disque, data = Decathlon)

Residuals:
    Min      1Q  Median      3Q     Max
-583.38 -128.66    2.22  138.86  554.64

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  4572.488   2307.954   1.981 0.053438 .
Longueur      483.056    105.502   4.579 3.44e-05 ***
Poids         118.215     37.831   3.125 0.003046 **
`110m.haies` -190.231     72.681  -2.617 0.011884 *
Perche        500.520    135.740   3.687 0.000586 ***
Javelot       -13.501      7.124  -1.895 0.064226 .
`1500m`        -9.125      4.127  -2.211 0.031942 *
Disque         45.360     10.999   4.124 0.000151 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 262.5 on 47 degrees of freedom
Multiple R-squared:  0.7024,	Adjusted R-squared:  0.6581
F-statistic: 15.85 on 7 and 47 DF,  p-value: 1.773e-10
```

Donc on remarque que toutes les variables ici sont assez significatives au seuil de 5%, on remarque également qu'il un R² 70.24% (c'est-à-dire que la somme des carrés expliqués occupe 70.24% de la somme des carrés totaux), ce qui est bien coté qualité de modélisation, et également on a le test de Fisher est globalement significatif (p. value <0.05).

Maintenant on peut faire la validation de ce modèle qui a été obtenu par la procédure step, on va commencer par test d'homoscédasticité : c'est le graphique dont nous avons résidu en ordonné et valeur ajustée en abscisse.

```
> ## Validation de modèle par procédure step :
> ## test d'homoscédasticité
> plot(predict(myregmult1),resid(myregmult1), xlab="valeur ajusté", ylab="résidu")
> abline(h=c(600,0,600), lty=c(2,1,2), col=c(1,2,1))
```



On remarque qu'il y a une absence de structure conique, donc on accepte l'hypothèse homoscédasticité.

Passons maintenant aux tests de normalités : shapiro et ks (les 2 tests existent sur R), on exécute les deux tests :

```
> ## test de normalité (shapiro et ks)
> ## SHAPIRO
> shapiro.test(resid(myregmult1)) ## p-value = 0.4448 > 0.05 Donc On ne rejette
  pas la normalité

        Shapiro-Wilk normality test

data:  resid(myregmult1)
W = 0.97896, p-value = 0.4448
```

Le test de shapiro ne rejette pas la normalité, parce que nous avons p-value de 0.448>0.05.

```
> ## KS
> ks.test(resid(myregmult1), pnorm) ## p-value = 5.251e-13 < 0.05 Donc on rejette
 la normalité

        One-sample Kolmogorov-Smirnov test

data:  resid(myregmult1)
D = 0.49571, p-value = 5.251e-13
alternative hypothesis: two-sided
```

Le test ks rejette la normalité, parce que le p-value est de 5.251 e-13 < 0.05

On passe maintenant à la recherche des points aberrants, donc on parcourue toutes les données, on a 55 données, M c'est la matrice qui contient toutes les variables explicatives, on a calculé également les intervalles de prévisions, on a repris la formule qui est dans le cours, avec 2.0117 est tout simplement le test de Student qu'on a calculé.

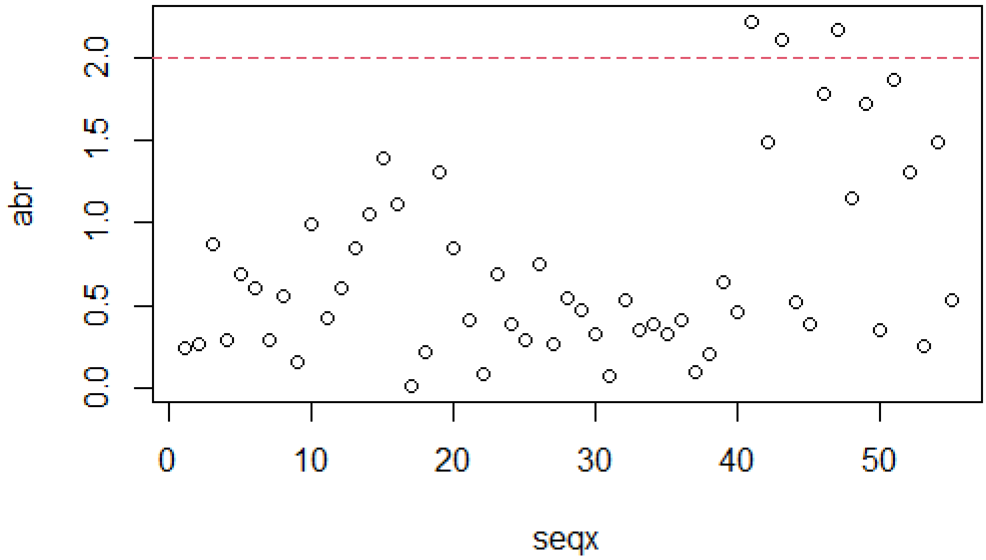
Initialement dans le vecteur M, on a que les 0, et on exécute le code pour tester est ce que la variable dépendante se situe à l'intérieur ou à l'extérieur de l'intervalle de prévision.

```
## recherche de valeurs aberrantes
sd=sqrt(deviance(myregmult1)/df.residual(myregmult1))  ## sd = 262.5128
x0=rep(1,55)
M=matrix(c(x0,Decathlon$Javelot,Decathlon$`1500m`,Decathlon$`110m.haies`,Decathlon$Poids,De
ul=matrix(0,55,1)
ll=matrix(0,55,1)
m=matrix(0,55,4)
j=1
for (i in 1:55) {
  ##  l'abscisse de la loi de Student à n-p-1 degré de libertés. Dans notre cas n=55 et p=7
  # Le calcul donne : 2,0117
  ul[i]=predict(myregmult1)[i]+2.0117*262.5128*sqrt(1+M[i,]%*%solve(t(M)%*%M)%*%M[i,])
  ll[i]=predict(myregmult1)[i]-2.0117*262.5128*sqrt(1+M[i,]%*%solve(t(M)%*%M)%*%M[i,])
  if (Decathlon$Points[i]>ul[i] | Decathlon$Points[i]<ll[i]) {
    m[j,]<-c(i,Decathlon$Points[i],ul[i],ll[i])
    j<-j+1
  }
}
View(m)
```

En exécutant, on trouve l'absence de valeurs aberrantes.

Dans cette partie, c'est l'approximation c'est-à-dire la recherche des valeurs aberrantes en faisant l'approximation : sd c'est somme des résidus au carré divisé par (N-k-1), sd dans notre cas c'est 262.5128, et les points aberrants c'est la valeur absolu du vecteur de la variable dépendante moins le vecteur de la variable ajustée divisé par le sd, puis on va tracer le nuage de points aberrants avec la droite horizontale.

```
seqx=seq(1,55,length=55)
sd=sqrt(deviance(myregmult1)/df.residual(myregmult1))
sd
abr=abs(Decathlon$Points-predict(myregmult1))/262.5128
plot(seqx,abr)
abline(h=2, lty=2,col=2)
```



On remarque la présence de 3 points aberrants, qui sont source de la normalité.

Le calcul exacte nous a donné aucun point aberrant alors que le calcul approché nous a donné 3 points aberrants.

3.A) Appliquer la méthode pas à pas de sélection des variables explicatives, basée sur le test de Fisher

On va appliquer maintenant la procédure de sélection, qui est basé sur le test de Fisher, c'est une approche pas à pas, en intégrant d'abord une première variable, et une deuxième variable, puis on va voir est ce qu'on peut retirer ou intégrer.

Etape1: On a la variable dépendante et on va voir qu'elle est la variable explicatives qui va entrer dans ce modèle, on va créer un vecteur Fish c'est là où on va mettre les Fishers qu'on a va calculer (c'est un vecteur où il y a que des 0).

```
#Etape1:
Fish = rep(0,11)
for (i in  2:ncol(Decathlon) ) {
  mod1<-lm(Points~as.matrix(Decathlon[,i]), data=Decathlon)
  Fish[i]=var(predict(mod1))*(nrow(Decathlon)-1)/(deviance(mod1)/df.residual(mod1))
}
Fish  #on remarque que la plus grande valeur c'est 21.142459 qui correspond à
#la variable "Longueur"
df2=nrow(Decathlon)-2
df2
1-pf(max(Fish),1,df2) #C'est égale à 2.683854e-05<max(Fish) donc il est significatif
## Introduction de la variable Longueur
```

En exécutant cette étape, on trouve le plus grand Fisher c'est égale à 21.142459, qui correspond à la variable Longueur, et p.value est egale à 2.683854e-05 Donc max(Fish) est significatif, donc il y a une introduction de la variable Longueur.

Etape 2: on va introduire la variable Longueur, on parcourt à partir de la 3ème variable jusqu'à la dernière, et chaque fois, on calcule le modèle de régression en intégrant la variable explicative avec la variable Points.

```
#Etape 2: -Inroduction- (on va introduire la variable Longueur)
Fish = rep(0,11)
SCR1<-deviance(lm(Points~Longueur, data=Decathlon))
for (i in 3:11) {
  mod<-lm(Points~Longueur+as.matrix(Decathlon[,i]),data=Decathlon)
  SCR2=deviance(mod)
  Fish[i]=(SCR1-SCR2)/(SCR2/(nrow(Decathlon)-3))
}
Fish #on remarque que la plus grande valeur c'est 17.00722446 qui correspond à
#la variable "Disque"
df2=nrow(Decathlon)-3
df2
1-pf(max(Fish),1,df2) #c'est égale à 0.0001345579 < max(Fish) donc c'est significatif
##Introduction de la variable Disque
summary(mod)
```

En exécutant cette étape, on trouve le plus grand Fisher c'est égale à 17.00722446, qui correspond à la variable Disque, et p.value est égale à 0.0001345579 Donc max(Fish) est significatif, donc il y a une introduction de la variable Disque.

```
> Fish #on remarque que la plus grande valeur c'est 17.00722446 qui correspond à
 [1]  0.00000000  0.00000000  4.10434861 13.57147406  0.07830561  1.18946534 12.38819398
 [8]  5.72624310  1.12334873  1.20996715 17.00722446
> #la variable "Disque"
> df2=nrow(Decathlon)-3
> df2
[1] 52
> 1-pf(max(Fish),1,df2) #c'est égale à 0.0001345579 < max(Fish) donc c'est significatif
[1] 0.0001345579
```

Maintenant on a la variable Longueur et la variable Disque, donc il est possible de retirer l'une des deux. Donc on va retirer soit la variable Longueur, soit la variable Disque, à chaque fois on calcule le F. Donc nous avons deux Fishers à calculer, un Fisher quand on retire la variable Longeur, et un autre lorsque on retire la variable Disque.

```
#Etape 2: -Retrait-
Fish = rep(0,2)
SCR2=deviance(lm(Points~Longueur+Disque, data=Decathlon))
mod<-lm(Points~Longueur, data=Decathlon)
SCR1<-deviance(mod)
Fish[1]=(SCR1-SCR2)/(SCR2/(nrow(Decathlon)-3))

mod<-lm(Points~Disque, data=Decathlon)
SCR1=deviance(mod)
Fish[2]=(SCR1-SCR2)/(SCR2/(nrow(Decathlon)-3))
Fish #on remarque que les 2 Fishers sont grands alors ils sont significatifs
df2=nrow(Decathlon)-3
df2
1-pf(min(Fish),1,df2) #C'est égale à 0.0001345579 donc il est significatif
## Aucune variable n'est retirée, les F sont significatifs
```

En exécutant le code, on a 2 Fishers qui sont significatifs, la variable qui peut être éliminé c'est celle qui a le plus petit Fisher, c'est pour cela on a calculé la p. value du plus petit Fisher, qui est significatif, donc on ne retire aucune des deux variables.

```
> Fish #on remarque que les 2 Fishers sont grands alors ils sont significatifs
[1] 17.00722 21.34985
> df2=nrow(Decathlon)-3
> df2
[1] 52
> 1-pf(min(Fish),1,df2) #C'est égale à 0.0001345579 donc il est significatif
[1] 0.0001345579
```

De la même manière, on continue les autres étapes.

A chaque étape :
La var entrante est celle qui présente le plus grand F avec pvalue<10%
La var sortante est celle qui présente le plus petit F avec pvalue>10%

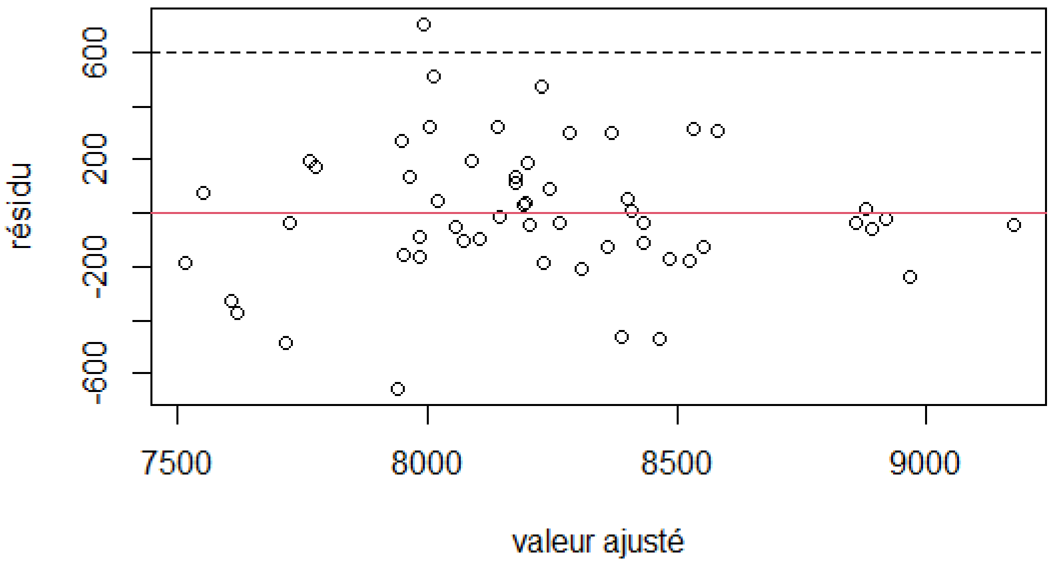
Arrêt :
Si les var entrantes ont p-values>10% et les var sortantes ont des pvalues<10%

Etape	Var entrée	Var sortie	R^2_{ajus}	F	pvalue
0	Longueur	Aucune	0.2258	21.142459	2.683854e-05
1	Disque	Aucune	0.4406	17.00722446	0.0001345579
2	Poids	Aucune	0.4463	10.4508742	0.002151312
3	Perche	Aucune	0.5287	9.820248599	0.002885973
4	110. haies	Aucune	0.6105	5.06510628	0.02893396
5	1500m	1500m	0.6396	2.74780466	0.1039105
6	Javelot	Javelot	0.6304	1.449006165	0.2347153
7	100m	100m	0.6191	1.057337021	0.3092001

Donc finalement c'est un modèle à 5 variables explicatives, qui sont Longeur, Disque, Poids, Perche et 110m.haies.

3.A.1) Faire les tests de validation pour le modèle obtenu par cette méthode

Maintenant on peut faire la validation de ce modèle qui a été obtenu par la méthode de séléction de Fisher, on va commencer par test d'homoscédasticité : c'est le graphique dont nous avons résidu en ordonné et valeur ajustée en abscisse.



On remarque qu'il y a une absence de structure conique, donc on accepte l'hypothèse homoscédasticité.

Passons maintenant aux tests de normalités : shapiro et ks (les 2 tests existent sur R), on exécute les deux tests :

```
> ## test de normalité (shapiro et ks)
> ## SHAPIRO
> shapiro.test(resid(myregmult2)) ## p-value = 0.5938 > 0.05 Donc On ne rejette pas la normal
ité

        Shapiro-Wilk normality test

data:  resid(myregmult2)
W = 0.98235, p-value = 0.5938
```

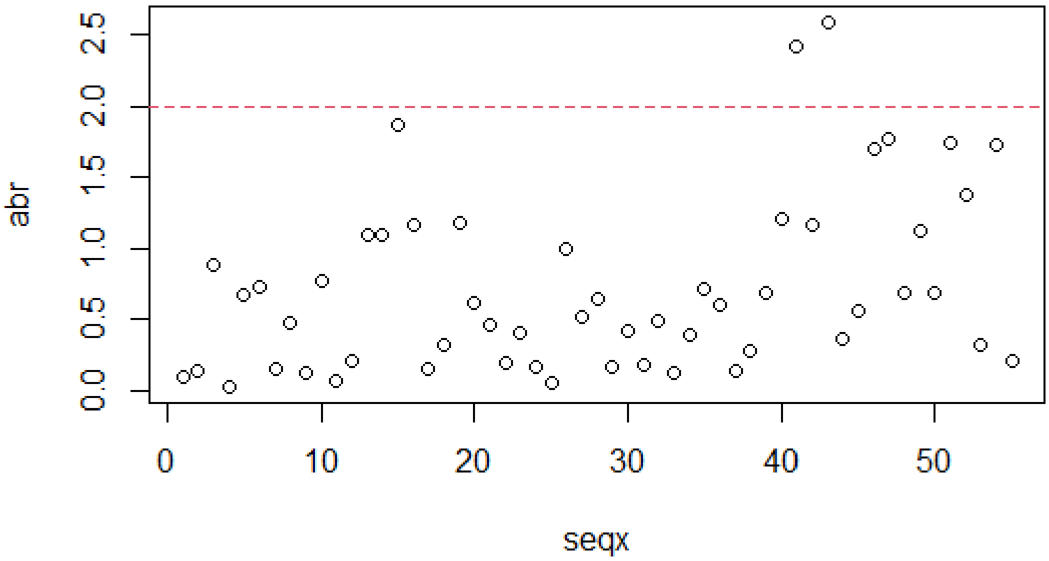
Le test de shapiro ne rejette pas la normalité, parce que nous avons p-value de 0.5938>0.05.

```
> ## KS
> ks.test(resid(myregmult2), pnorm) ## p-value = 1.332e-15 < 0.05 Donc on rejette la normalit
é

        One-sample Kolmogorov-Smirnov test

data:  resid(myregmult2)
D = 0.54545, p-value = 1.332e-15
alternative hypothesis: two-sided
```

Le test ks rejette la normalité, parce que le p-value est de 1.332 e-15 < 0.05

On passe maintenant à la recherche des points aberrants :

En exécutant le code : Par la méthode exacte, on a l'absence de valeurs abérrantes.



On remarque la présence de 2 points aberrants, qui sont source de la normalité.

Le calcul exacte nous a donné aucun point aberrant alors que le calcul approché nous a donné 2 points aberrants.

3.A.2) Calculer le critère AIC du modèle obtenu par cette méthode

On exécutant la commande :

```
Step:  AIC=623.2
Points ~ Longueur + Poids + `110m.haies` + Perche + Disque

               Df Sum of Sq     RSS    AIC
<none>                      3686014 623.20
- `110m.haies`  1    381021 4067035 626.61
- Poids         1    630787 4316801 629.89
- Perche        1    686041 4372054 630.59
- Disque        1   1040692 4726706 634.88
- Longueur      1   1664370 5350383 641.69

Call:
lm(formula = Points ~ Longueur + Poids + `110m.haies` + Perche +
    Disque, data = Decathlon)

Coefficients:
 (Intercept)      Longueur         Poids  `110m.haies`        Perche        Disque
     1287.17        514.83        112.80       -167.39        414.55         42.47
```

AIC= 632.2

4.A) Conclusion, il s’agit de comparer et choisir entre le modèle obtenu par step et celui obtenu par cette méthode de sélection

On a le critère AIC du modèle obtenu par step =620.09 et celui du modèle obtenu par la méthode de sélaction de Fisher = 623.2

Donc on conclut, que le modèle obtenu par step est le meilleur.

Partie 2

Projet Classification

La classification permet d'identifier des groupes homogènes au sein de la population du point de vue des variables étudiées.

Notre jeu de données Décathlon contient les performances réalisées par des athlètes lors d'une compétition.

Dans ce jeu de données, il y a 55 lignes et 11 colonnes :

- ❖ Les colonnes 2 à 11 sont des variables quantitatives (les variables explicatives), correspondent aux performances des athlètes pour les dix épreuves du Décathlon (Longueur, 100m, Poids, Hauteur, 400m, 110m.haies, Perche, Javelot, 1500m et Disque)
- ❖ La 1ère colonne est une variable quantitative (variable dépendante) qui correspond au nombre de points obtenus par chaque athlète.

Nous allons faire une classification sur ce jeu de données afin d'identifier des groupes homogène, autrement dit afin de catégoriser les individus en classe d'invidus.

Dans cette partie, nous aurons besoin de deux fichier Excel, le premier c'est le fichier Excel « Decathlon » (notre jeu de donnée), et le 2 ème c'est « Decathlon_CR » c'est là où nous avons les données centrées et réduites manuellement sur Excel, qui vont être utilisée pour la CAH.

1) Appliquer kmeans au tableau des variables quantitatives, le nombre de classes va varier de 1 à N et les variables doivent être centrées et réduites. N étant le plus petit entier tel que le taux d'inertie expliquée de la classification à N classes est supérieur à 0.95

On va commencer tout d'abord par importer et lire les données du premier fichier Excel, celui de notre jeu de données :

```
Decathlon <- read_excel("C:/Users/hp/Desktop/Decathlon.xlsx")
```

Ensuite, on peut le nombre d'observations qu'on a, en utilisant la commande nrow ( ) : en exécutant, on trouve 55 observation, on peut également voir les caractéristiques des variables, en utilisant la commande str( ) :

```
> str(Decathlon)
tibble [55 x 11] (S3: tbl_df/tbl/data.frame)
 $ Points    : num [1:55] 8894 8820 8725 8414 8343 ...
 $ Longueur  : num [1:55] 7.84 7.96 7.83 7.47 7.74 7.14 7.19 7.53 7.44 7.49 ...
 $ 100m      : num [1:55] 10.8 10.4 10.5 10.9 10.6 ...
 $ Poids     : num [1:55] 16.4 15.2 15.9 15.7 14.5 ...
 $ Hauteur   : num [1:55] 2.12 2.06 2.09 2.15 1.97 2.17 2.03 1.88 2.08 1.94 ...
 $ 400m      : num [1:55] 48.4 49.2 46.8 49 48 ...
 $ 110m.haies: num [1:55] 14.1 14.1 14 14.6 14 ...
 $ Perche    : num [1:55] 5.01 4.92 4.63 4.47 4.93 4.77 4.81 5.48 4.4 5.11 ...
 $ Javelot   : num [1:55] 70.5 69.7 55.5 58.5 55.4 ...
 $ 1500m     : num [1:55] 280 282 278 265 278 ...
 $ Disque    : num [1:55] 48.7 50.1 51.6 48.3 43.7 ...
```

Passons maintenant au centrage et réduction des données, en utilisant la commande scale( ), c'est une commande dans R qui prend 3 arguments, le 1er argument c'est le nom de l'objet « Decathlon », le 2ème c'est centre=T (centrage), et le 3ème argument, c'est scale=T (réduction).

```
#centrage réduction des données
Decathlon_cr<-scale(Decathlon,center=T,scale=T)
View(Decathlon_cr)
```

	Points	Longueur	100m	Poids	Hauteur	400m	110m.haies	Perche	Javelot	1500m	Disque
1	1.505197619	1.26187291	-0.16141930	1.703223784	0.470221930	-0.33243528	-0.772537363	0.366319048	1.46078264	0.55298462	1.363062973
2	1.340369833	1.59381238	-1.48988835	0.571323033	0.154186824	0.25101995	-0.630609831	0.039602059	1.31841146	0.76744961	1.758682427
3	1.128766595	1.23421129	-1.29547825	1.272500489	0.312204377	-1.42202036	-0.914464895	-1.013152684	-1.17220536	0.34821904	2.196994628
4	0.436044415	0.23839289	-0.03181256	1.072164073	0.628239483	0.09636917	0.132250655	-1.593982887	-0.65896605	-1.01939944	1.254908014
5	0.277898837	0.98525669	-0.90665804	-0.179938526	-0.319865835	-0.60658895	-0.843501129	0.075903947	-1.19857040	0.34175276	-0.057182406
6	0.153164296	-0.67444064	0.03299081	0.651457600	0.733584519	0.39864116	0.824147375	-0.504926256	0.21811073	-0.57538146	0.480746204
7	0.041794171	-0.53613253	0.22740091	-0.009652573	-0.003830729	-0.07234078	-0.453200415	-0.359718705	-0.78200287	-1.13471479	0.224589723
8	0.037339366	0.40436262	-0.32342772	-0.400308584	-0.793918494	-0.01610413	0.558033252	2.072507770	-0.15451507	0.15638603	-0.535341171
9	0.015065341	0.15540802	-0.67984625	0.140599739	0.259531860	0.20884247	-0.559646065	-1.848096100	-1.21966242	0.15423061	0.221743540

Showing 1 to 9 of 55 entries, 11 total columns

Avant de faire le kmeans, il faut portion d'inertie expliquée, c'est-à-dire déterminer le N de telle sorte qu'il soit le plus petit entier tel que le taux d'inertie expliquée de la classification à N classes est supérieur à 0.95. Donc pour cela nous allons exécuter la boucle for :

```
> #évaluer la proportion d'inertie expliquée
> N=41
> inertie.expl <- rep(0,times=N)
> for (k in 2:N){
+   clus <- kmeans(Decathlon_cr,centers=k,nstart=5)
+   inertie.expl[k] <- clus$betweenss/clus$totss
+ }
> #l'inertie explicative est nulle si k=1 (1 seule classe)
> max(inertie.expl)
[1] 0.9544727
```

Nous avons trouvé, **N=41**

Passons maintenant à la commande du kmeans ( ), parmis les paramètres, y a l'objet Decathlon_cr (résultat de la fonction scale( )), nous avons également le nombre de classe c'est « centers », donc on aura dans notre cas 41 classes, et il y aussi le « nstart » c'est le nombre d'essais (il est par défaut =5) c'est-à-dire le kmeans va être exécuter à 5 reprises, en principe, il va retenir la meilleur classification c'est-à-dire le meilleur taux d'inertie expliqué , pour que la kmeans démarre, il faut préciser une condition initiale.

Remarque : le nombre de classe içi c'est une **input** : dans kmeans, il faut indiquer le nombre de classe, pour qu'il nous fait la répartition.

```
> #k-means avec les données centrées et réduites
> #center = 41 - nombre de groupes demandés
> #nstart = 5 - nombre d'essais avec différents individus de départ
> groupes.kmeans <- kmeans(Decathlon_cr,centers=41,nstart=5)
> #affichage des résultats
> print(groupes.kmeans)
K-means clustering with 41 clusters of sizes 1, 2, 1, 1, 2, 1, 1, 1, 1, 2, 2, 1, 2, 1, 1, 1, 1, 2,
 1, 1, 3, 1, 2, 1, 1, 1, 1, 1, 1, 1, 2, 1, 3, 1, 1, 2, 1, 2, 1, 1, 1


Cluster means:
         Points    Longueur        100m       Poids     Hauteur        400m    110m.haies
1   0.473910257  1.17888805 -0.12901762  0.98201269 -0.53055591 -0.57847062 -0.967687720
2   0.246715201  0.62565560 -0.66364540  0.21071748 -0.45154713 -0.54683751 -0.683832655
3  -0.129715822  0.34903938 -0.80945298 -1.05140194 -0.05650325  0.72903147  0.114509713
4   1.128766595  1.23421129 -1.29547825  1.27250049  0.31220438 -1.42202036 -0.914464895
5   0.835863165  0.58416317 -1.21447404 -0.94121691  0.36487690 -0.82099117 -1.100744781
6  -0.931580726 -0.97871849 -1.78150351 -0.59062818 -1.63667878  0.67982440  0.859629258
7   0.805793231  0.84694858 -0.38823109 -0.46040951  0.36487690 -1.46419785  0.238696304
8  -2.089830030 -0.17653144 -0.64744456 -1.73254575  1.31298221  0.92585974 -2.103107978
9   0.003928328 -0.50847091  1.49106659 -0.81099824  0.78625704  0.31428618  0.398364778
10  0.128662869  0.29371613  0.51901607 -0.02467780 -0.18818454  0.26859391  0.478199015
11 -0.105214395 -0.45314767  0.81063123 -0.22501422  0.28586812 -0.14615138 -0.178215822
12  1.371553468  0.59799398 -0.29102604  1.10221454  1.04961962 -0.38867193 -0.825760187
13  1.422783726  1.42784265 -0.82565382  1.13727341  0.31220438 -0.04070767 -0.701573597

Clustering vector:
 [1] 13 13  4 23  2 36 11 28 23 25 40 12  5  7  5 17 14 31 10  2  1 21 31 21 38  9 11 20  3 36
[31] 22 18 10 21 27 33 33 33 34 30  8 39 15 16  6 38 32 37 24 18 35 41 26 29 19

Within cluster sum of squares by cluster:
 [1] 0.000000 1.162993 0.000000 0.000000 2.993114 0.000000 0.000000 0.000000 0.000000 1.998155
[11] 1.584307 0.000000 1.986673 0.000000 0.000000 0.000000 0.000000 1.559953 0.000000 0.000000
[21] 3.539141 0.000000 2.461488 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
[31] 1.347308 0.000000 6.584286 0.000000 0.000000 1.355504 0.000000 3.258243 0.000000 0.000000
[41] 0.000000
 (between_SS / total_SS =  95.0 %)
```

En exécutant le kmeans, nous avons les résultats qui sont donnés, on la taille du groupe : K-means clustering with 41 clusters of sizes, donc au totale nous avons 55 individus qui ont été regroupées en 41 groupes (groupes de 1 individus, 2 et 3), puis il nous a donné la moyenne par groupe : c'est Cluster means, également il nous a donné les affectations : Clustering vecteur : la 1ère classe c'est 2 effectifs donc 13 va être répété 2 fois, la 2ème classe c'est 1 effectid donc 4 va être répété une seule fois…

Nous trouvons également « within cluster sum of squares by cluster » c'est les interties intra par classe, donc là nous avons 41 classes, et dans chaque classe y a l'inertie intra, et la somme de ces 41 inerties intra, va nous donner l'inertie intra total.

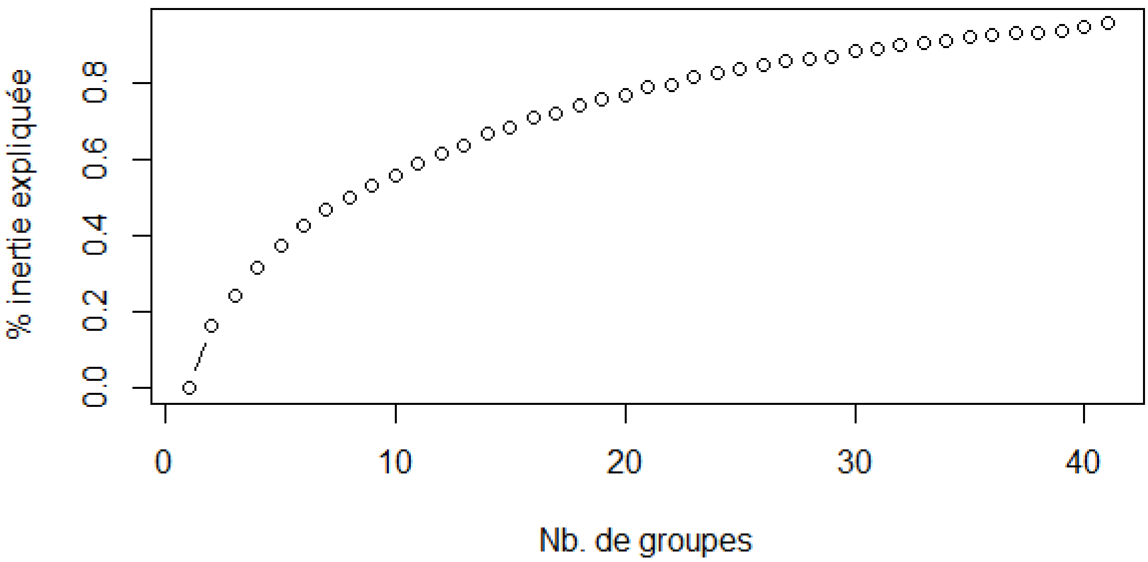
Et finalement nous avons le taux d'inertie, qui est l'inertie inter sur l'inertie total, qui est de 95%.

Donc voilà les principaux résultats qu'on peut obtenir en exécutant la commande kmean( ).

2) Déterminer Nc le nombre de classes à retenir en utilisant la méthode de la diapositive 22 du cours : $\frac{var(I_2)}{var(I)} < 0{,}05$, I étant le vecteur de taille N des taux d'inertie expliquée et I_2 étant le vecteur des (N-Nc) dernières valeurs des taux d'inertie expliquée.

En exécutant la commande :

```
#graphique
plot(1:N,inertie.expl,type="b",xlab="Nb. de groupes",ylab="% inertie expliquée")
```



On remarque que c'est difficile de spécifier le nombre de classes à partir du plot, par ce que nous n'avons pas un grand nombre d'observations (nous avons que 55 observations), c'est pour cela qu'on va déterminer le Nc à partir du rapport des variances $\frac{var(I_2)}{var(I)} < 0,05$

```
#rapport des variances
var(inertie.expl[17:N])*(N-17)*100/(var(inertie.expl)*(N-1))
#5.21719 > 5%
var(inertie.expl[18:N])*(N-18)*100/(var(inertie.expl)*(N-1))
#4.266346 < 5%
```

Si on élimine de 17 jusqu'à N (c'est-à-dire Nc=16), on a un rapport de 5.21719>5%

Si on élimine de 18 jusqu'à N (c'est-à-dire Nc=17), on a un rapport de 4.266346<5%

Donc on va retenir 17 classes, **Nc=17**.

3) Faire une CAH sur le tableau des variables quantitatives. Les variables doivent au préalable être centrées et réduites.

On va charger tout d'abord le package « FactoMineR» c'est un package du logiciel R dédié à l'analyse de données.

```
library(FactoMineR)
```

Puis nous allons passer à importer et lire les données du deuxième fichier Excel « Decathlon_CR », dont nous avons fait le centrage et réduction des données manuellement sur Excel.

```
library(FactoMineR)
#Les données sont centrées et réduites sur Excel
library(readxl)
Decathlon_CR <- read_excel("C:/Users/hp/Desktop/Decathlon_CR.xlsx")
View(Decathlon_CR)
```

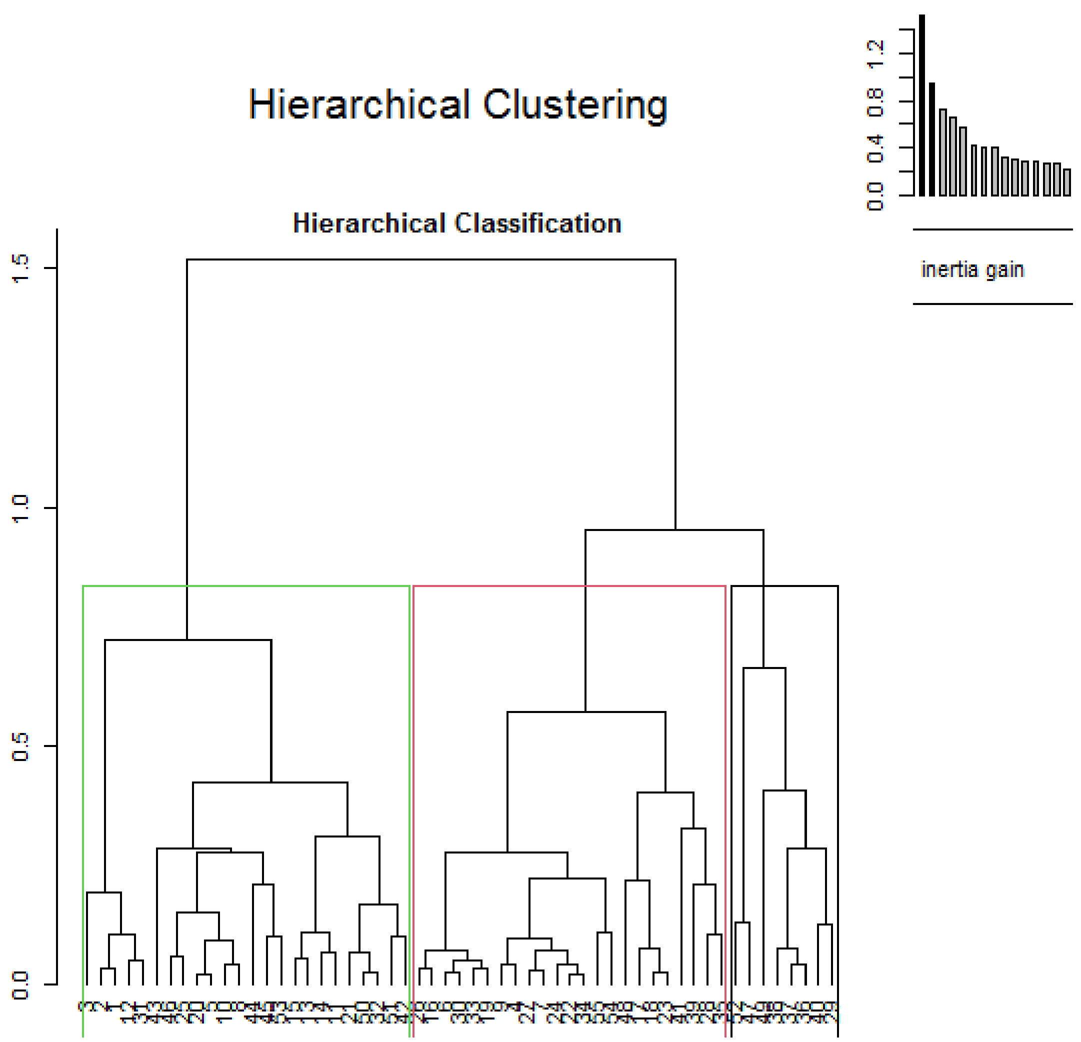
	Points_CR	Longueur_CR	100m_CR	Poids_CR	Hauteur_CR	400m_CR	110m.haies_CR	Perche_CR	Javelot_CR	1500m_CR	Disque_CR
1	1.519070701	1.27350332	-0.16290707	1.718922030	0.474555865	-0.33549927	-0.779657673	0.369695332	1.47424636	0.55808136	1.375626031
2	1.352723733	1.60850221	-1.50362033	0.576588795	0.155607931	0.25333355	-0.636422026	0.039967063	1.33056297	0.77452303	1.774891824
3	1.139170194	1.24558675	-1.30741839	1.284228852	0.315081898	-1.43512682	-0.922893321	-1.022490694	-1.18300933	0.35142850	2.217243855
4	0.440063342	0.24059011	-0.03210577	1.082045979	0.634029831	0.09725738	0.133469581	-1.608674284	-0.66503959	-1.02879503	1.266474231
5	0.280460170	0.99433759	-0.91501451	-0.181596981	-0.322813969	-0.61217975	-0.851275497	0.076603537	-1.20961736	0.34490262	-0.057709444
6	0.154575978	-0.68065682	0.03329487	0.657461944	0.740345809	0.40231535	0.831743364	-0.509580053	0.22012101	-0.58068463	0.485177138
7	0.042179379	-0.54107395	0.22949682	-0.009741538	-0.003866036	-0.07300753	-0.457377466	-0.363034155	-0.78921042	-1.14517321	0.226659718
8	0.037683515	0.40808955	-0.32640868	-0.403998142	-0.801235869	-0.01625256	0.563176524	2.091609628	-0.15593921	0.15782741	-0.540275296
9	0.015204195	0.15684039	-0.68611224	0.141895617	0.261923909	0.21076732	-0.564804202	-1.865129604	-1.23090379	0.15565212	0.223787302

Showing 1 to 10 of 55 entries, 11 total columns

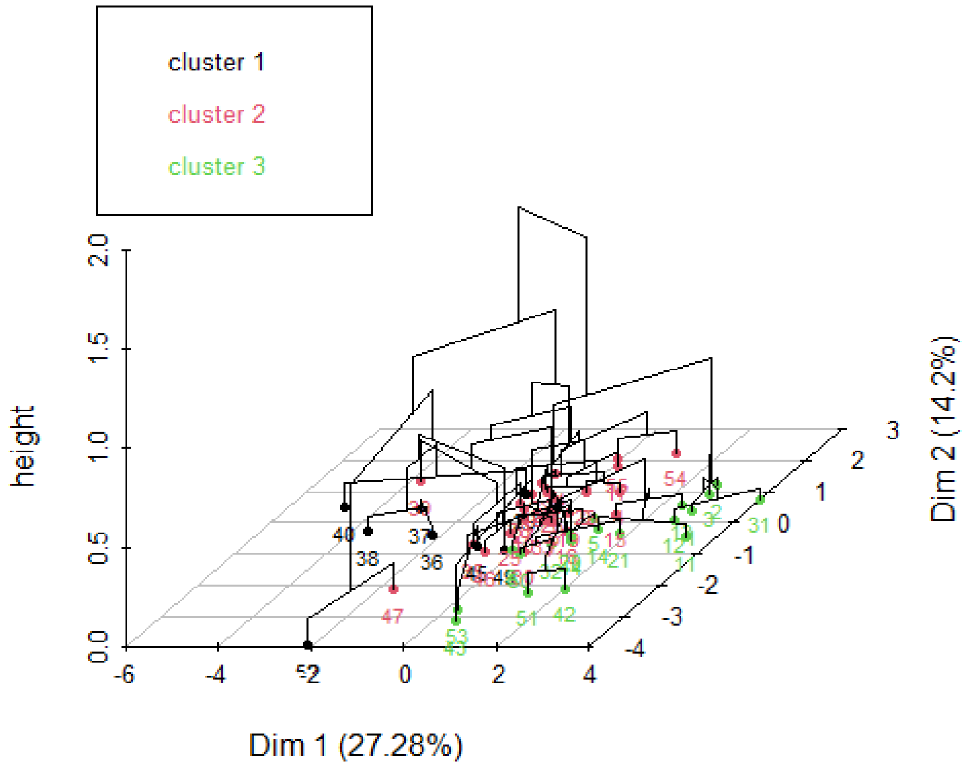
Remarque : Il ne faut pas appliquer la CAH sur le résultat de la fonction scale( ), c'est pour cela qu'on a fait le centrage et réduction des données manuellement sur Excel.

Ensuite, exécuter la commande HCPC( ) : la mention "nb.clust=-1" pour avoir une coupe systématique.

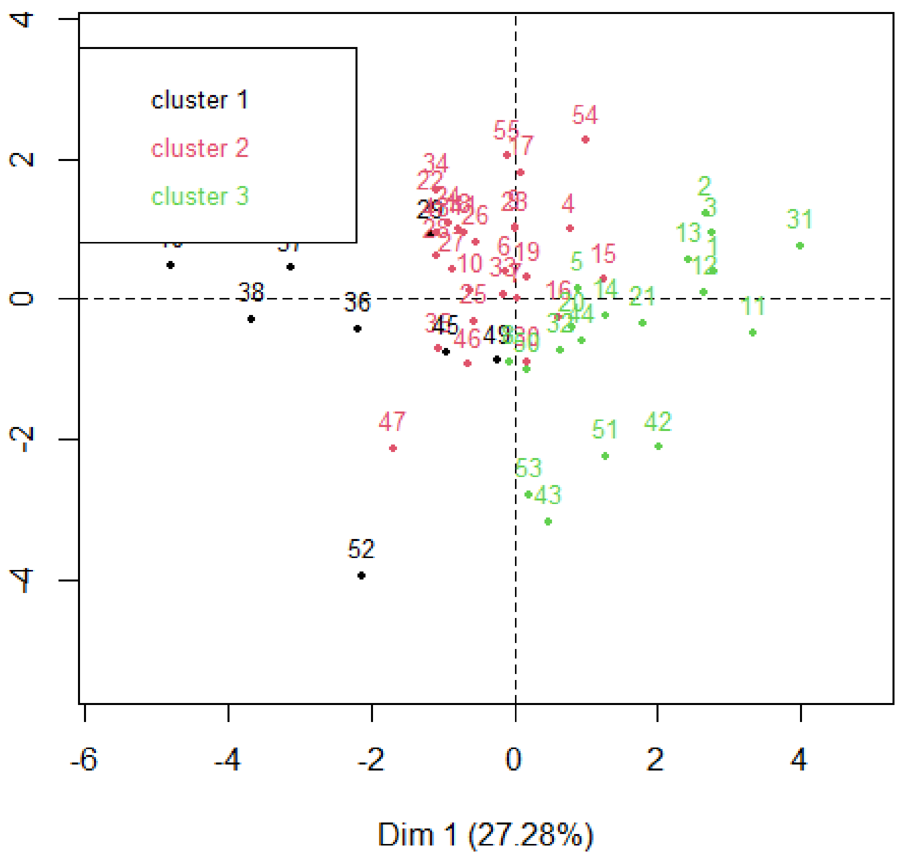
```
#CAH
Res<-HCPC(Decathlon_CR,nb.clust=-1)
```



Hierarchical clustering on the factor map



Factor map



La CAH nous a donné 3 classes d'individus, nous avons 55 individus qui ont été catégorisé en 3 classes.

3.1) Quelles sont les variables quantitatives les plus corrélées avec la variable classification

Il suffit d'exécuter la commande : Res$desc.var

```
> Res$desc.var

Link between the cluster variable and the quantitative variables
================================================================
                  Eta2      P-value
400m_CR       0.4611459 1.042918e-07
Hauteur_CR    0.3551379 1.112087e-05
1500m_CR      0.3275033 3.310880e-05
Longueur_CR   0.3214617 4.177664e-05
100m_CR       0.2977001 1.022323e-04
Points_CR     0.2854054 1.605290e-04
110m.haies_CR 0.2459374 6.495188e-04
Poids_CR      0.1761140 6.494501e-03
Perche_CR     0.1152145 4.147519e-02
```

On a comme 1er résultat les coefficients de corrélations entre les variables quantitatives et la variable classification, et ils sont classés par ordre décroissant.

On remarque qu'il n'y a aucune variable quantitative corrélée avec la variable classification, car toutes les variables ont un coefficient de corrélation <0.5.

3.2) Faire la description des classes retenues par variables

En exécutant la même commande de la question précédente, on a comme 2ème résultat la description des 3 classes.

```
Description of each cluster by quantitative variables
=====================================================
$`1`
               v.test Mean in category  Overall mean sd in category Overall sd      p.value
1500m_CR     4.187733        1.3812939 -4.871797e-15      1.1556527          1 2.817542e-05
400m_CR      2.986091        0.9849406  2.752848e-16      1.0711352          1 2.825689e-03
100m_CR      2.083724        0.6873013  1.225484e-14      1.2218943          1 3.718530e-02
Points_CR   -2.766010       -0.9123487 -1.571565e-15      1.1071510          1 5.674671e-03
Hauteur_CR  -3.456549       -1.1401180  1.148545e-15      0.6153123          1 5.471390e-04
Longueur_CR -3.830355       -1.2634153  3.069514e-15      0.7710532          1 1.279583e-04

$`2`
                 v.test Mean in category  Overall mean sd in category Overall sd      p.value
Hauteur_CR     3.892221        0.5201199  1.148545e-15      0.7297947          1 9.933086e-05
400m_CR        2.373579        0.3171829  2.752848e-16      0.6860717          1 1.761660e-02
100m_CR        2.233029        0.2984011  1.225484e-14      0.7832168          1 2.554702e-02
110m.haies_CR  2.123309        0.2837391 -1.405675e-14      0.8760952          1 3.372795e-02
1500m_CR      -2.108619       -0.2817760 -4.871797e-15      0.7918400          1 3.497750e-02
Poids_CR      -2.272150       -0.3036288  9.493668e-16      0.9066117          1 2.307746e-02
Perche_CR     -2.481703       -0.3316315 -1.788263e-16      1.0219219          1 1.307563e-02

$`3`
                 v.test Mean in category  Overall mean sd in category Overall sd      p.value
Points_CR      3.486833        0.6531436 -1.571565e-15      0.7991163          1 4.887768e-04
Poids_CR       3.083823        0.5776529  9.493668e-16      0.7944925          1 2.043594e-03
Longueur_CR    2.711986        0.5080015  3.069514e-15      0.9665203          1 6.688145e-03
Perche_CR      2.004515        0.3754800 -1.788263e-16      0.9433270          1 4.501493e-02
110m.haies_CR -3.563578       -0.6675192 -1.405675e-14      0.7474376          1 3.658345e-04
100m_CR       -3.892537       -0.7291390  1.225484e-14      0.7073408          1 9.920137e-05
400m_CR       -4.709335       -0.8821392  2.752848e-16      0.6189502          1 2.485265e-06
```

La **classe 1** est composée d'individus tels que 36,38 et 52. Ce groupe a un v. test de 4.187733 (c'est un v. test assez grand pour la variable 1500m_CR), et un v. test de 2.986091 (c'est un v. test assez grand pour la variable 400m_CR), aussi y a un v. test de 2.083724 (c'est un v. test assez grand pour la variable 100m_CR). Donc cette classe est caractérisée par des individus qui ont bien réussi les 3 épreuves 1500m, 400m et 100m, mais qui n'ont pas bien réussir les 2 épreuves Hauteur et Longueur (v. tests assez grands et négatifs), également ces individus n'ont pas pu réaliser plusieurs points.

La **classe 2** est composée d'individus tels que 4, 15 et 54. Cette classe est caractérisée par des individus qui ont bien réussi les 4 épreuves Hauteur, 400m, 100m et 110m.haies (v. tests assez grands et positifs), mais qui n'ont pas pu bien réussir les 3 épreuves 1500m, Poids et Perche (v. tests assez grands et négatifs).

La **classe 3** est composée d'individus tels que 2, 3, 31 et 42. Cette classe est caractérisée par des individus qui ont réussi à collecter plusieurs points (v. test assez grand et positif), également ont bien réussi les 3 épreuves Poids, Longueur et Perche (v. tests assez grands et positifs), mais qui n'ont pas pu bien réussir les 3 épreuves 110m.haies, 100m et 400m (v. tests assez grands et négatifs).

3.3) Calculer les taux d'inertie : Inertie Inter/Inertie total, avant et après la consolidation de la CAH.

Il suffit d'exécuter la commande Res$call :

```
> ##Calcul du taux d'inertie
> I<-Res$call
> I
$t

$t$nb.clust
[1] 3

$t$within
 [1] 11.00000000  9.48169868  8.52988219  7.80663405  7.14319289  6.57043126  6.14489498
 [8]  5.73553037  5.33269317  5.00343179  4.69067007  4.40234826  4.11772149  3.83870285
[15]  3.56084866  3.33669833  3.11848969  2.90722987  2.69770273  2.50453677  2.33643830
[22]  2.18408545  2.05155725  1.92440178  1.81490043  1.70606405  1.59959609  1.49481581
[29]  1.39183193  1.28895351  1.19188495  1.09977929  1.02082688  0.94436189  0.86952717
[36]  0.79493829  0.72487870  0.65768630  0.59734848  0.54192044  0.48956173  0.43775533
[43]  0.39217221  0.34807053  0.30472631  0.26174749  0.22474462  0.18795437  0.15125399
[50]  0.12191497  0.09302695  0.06792502  0.04297488  0.02143797

$t$inert.gain
 [1] 1.51830132 0.95181648 0.72324814 0.66344117 0.57276163 0.42553628 0.40936461
 [8] 0.40283721 0.32926138 0.31276172 0.28832180 0.28462677 0.27901864 0.27785419
[15] 0.22415033 0.21820864 0.21125982 0.20952714 0.19316596 0.16809847 0.15235285
[22] 0.13252820 0.12715547 0.10950134 0.10883639 0.10646795 0.10478029 0.10298388
[29] 0.10287842 0.09706856 0.09210566 0.07895241 0.07646500 0.07483471 0.07458888
[36] 0.07005958 0.06719240 0.06033783 0.05542804 0.05235870 0.05180641 0.04558312
[43] 0.04410168 0.04334421 0.04297882 0.03700288 0.03679025 0.03670038 0.02933902
[50] 0.02888802 0.02510193 0.02495014 0.02153690 0.02143797

$t$quot
[1] 0.8996154 0.9152101 0.9150157 0.9198171 0.9352347 0.9333814 0.9297646 0.9382561
```

Nous avons l'inertie intra ($t$nb.clust), donc nous avons pour la première classe, l'inertie intra = 11, quand nous passant à 2 classes : l'inertie intra= 9.18, quand nous passant à 3 classes : l'inertie intra = 8.53….Plus nous augmente de classes, plus l'inertie intra est nulle, quand on arrive à 55 classes (puisque on a que 55 individus), l'inertie intra est nulle.

Nous avons également le gain d'inertie ($t$inert.gain) c'est l'inertie inter, quand je passe d'une seule classe à 2 classes, nous avons un gain d'inertie de 1.52, quand je passe de 2 classes à 3classes, nous avons un gain d'inertie de 0.95…Quand je passe de 54 classes à 55 classes, donc là il n'y a pas de gain d'inertie.

Et bien sur la somme de l'inertie intra et l'inertie inter, c'est l'inertie total, qui est dans notre cas égale à 11.

```
$bw.before.consol
[1] 2.470118

$bw.after.consol
[1] 2.681894
```

Ici nous avons l'inertie inter avant consolidation et après consolidation, alors avant consolidation c'est des résultats de la CAH, et après consolidation c'est le résultat lorsque la solution de la CAH est solution initiale du kmeans.

Donc on a amélioré le taux d'inertie, on est passé de 2.470118 à 2.681894.

4) Comparer les classifications faites par kmeans et CAH

Le kmeans nécessite une connaissance préalable du clusters (nombre de classes), alors que la CAH, on peut s'arrêter à n'importe quel nombre de groupes, on le trouve approprié en interprétant le dendrogramme.

Les méthodes utilisées dans kmeans sont normalement moins gourmandes en calculs et sont adaptées à de très grands ensembles de données. Pour la CAH, les méthodes de division fonctionnent dans la direction opposées : en commençant par un cluster qui inclut tous les enregistrements, et les méthodes hiérarchiques sont particulièrement utiles lorsque l'objectif est d'organiser les clusters dans une hiérarchie naturelle.

Coté avantages, pour le kmeans, y a toujours la convergence qui est garanti, et pour la CAH, elle facilite le traitement de toute forme de similitude ou de distance, par conséquent, elle est applicables à tous les types d'attributs.

Coté désavantages, pour le kmeans, c'est parfois difficile de prévoir le nombre de classe, et pour la CAH, le clustering hiérarchique nécessite le calcul et le stockage d'une matrice de distance n × n. Pour les très grands ensembles de données, cela peut être coûteux et lent.

Partie 3

Projet ACP

Notre jeu de données Décathlon contient les performances réalisées par des athlètes lors d'une compétition.

Dans ce jeu de données, il y a 55 lignes et 11 colonnes :

- ❖ Les colonnes 2 à 11 sont des variables quantitatives (les variables explicatives), correspondent aux performances des athlètes pour les dix épreuves du Décathlon (Longueur, 100m, Poids, Hauteur, 400m, 110m.haies, Perche, Javelot, 1500m et Disque)
- ❖ La $1^{ère}$ colonne est une variable quantitative (variable dépendante) qui correspond au nombre de points obtenus par chaque athlète.

Nous allons faire une ACP (Analyse en composantes principales) sur ce jeu de données afin de le décrire, de le résumer, d'en réduire la dimensionnalité.

L'ACP réalisée sur les individus du tableau de données répond à différentes questions :

1. Etude des individus (i.e. des athlètes) : deux athlètes sont proches s'ils ont des résultats similaires. On s'intéresse à la variabilité entre individus. Y a-t-il des similarités entre les individus pour toutes les variables ? Peut-on établir des profils d'athlètes ? Peut-on opposer un groupe d'individus à un autre ?
2. Etude des variables (i.e. des performances) : on étudie les liaisons linéaires entre les variables. Les objectifs sont de résumer la matrice des corrélations et de chercher des variables synthétiques: peut-on résumer les performances des athlètes par un petit nombre de variables ?
3. Lien entre les deux études : peut-on caractériser des groupes d'individus par des variables ?

1) Faire une ACP normée sur le tableau des variables quantitatives

On va commencer tout d'abord par importer et lire les données :

```
Decathlon <- read_excel("C:/Users/hp/Desktop/Decathlon.xlsx")
```

Il est important de s'assurer que l'importation a bien été effectuée, et notamment que les variables quantitatives sont bien considérées comme quantitatives :

```
> summary(Decathlon)
     Points        Longueur          100m           Poids          Hauteur           400m      
 Min.   :7230   Min.   :6.580   Min.   :10.28   Min.   :12.84   Min.   :1.630   Min.   :46.07  
 1st Qu.:7978   1st Qu.:7.130   1st Qu.:10.70   1st Qu.:13.92   1st Qu.:1.905   1st Qu.:47.99  
 Median :8237   Median :7.440   Median :10.91   Median :14.65   Median :2.050   Median :48.81  
 Mean   :8218   Mean   :7.384   Mean   :10.90   Mean   :14.66   Mean   :2.031   Mean   :48.83  
 3rd Qu.:8458   3rd Qu.:7.665   3rd Qu.:11.14   3rd Qu.:15.41   3rd Qu.:2.185   3rd Qu.:49.70  
 Max.   :9126   Max.   :7.960   Max.   :11.43   Max.   :16.36   Max.   :2.330   Max.   :52.67  
   110m.haies        Perche          Javelot          1500m           Disque     
 Min.   :13.30   Min.   :4.350   Min.   :49.45   Min.   :254.6   Min.   :35.30  
 1st Qu.:14.04   1st Qu.:4.755   1st Qu.:57.44   1st Qu.:267.8   1st Qu.:42.13  
 Median :14.56   Median :4.890   Median :63.19   Median :274.2   Median :44.75  
 Mean   :14.49   Mean   :4.909   Mean   :62.21   Mean   :274.9   Mean   :43.93  
 3rd Qu.:14.93   3rd Qu.:5.115   3rd Qu.:66.19   3rd Qu.:280.4   3rd Qu.:45.52  
 Max.   :15.63   Max.   :5.480   Max.   :72.32   Max.   :304.5   Max.   :51.65  
```

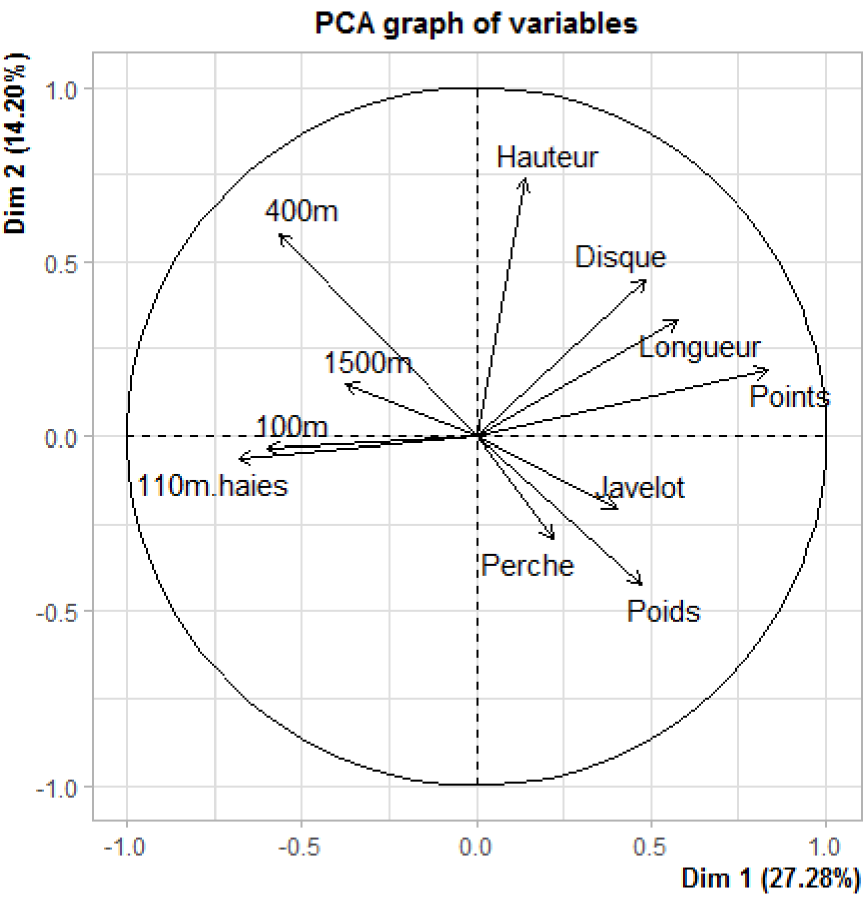
Donc nous avons 11 variables et 55 observations c'est-à-dire un tableau de taille 55x11 (11colonnes et 55lignes).

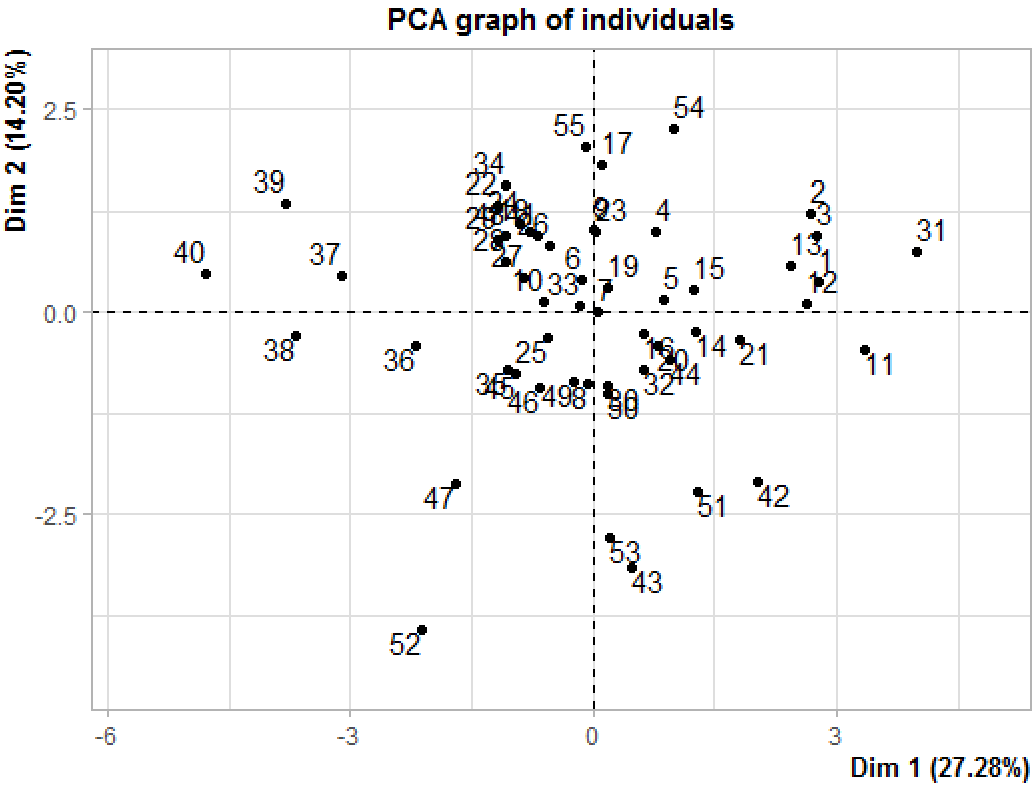
Avant d'exécuter la commande de l'ACP, il faut charger le package « FactoMineR » c'est un package du logiciel R dédié à l'analyse de données.

```
library(FactoMineR)
```

Ensuite on applique l'ACP normée, c'est la commande PCA( ) : qui comporte 3 arguments : le 1er argument est le nom de l'objet « Decathlon », le 2ème argument est le nombre de composantes : quand on fera l'ACP, R va calculer les projections, les contributions et cos2 que ça soit pour les variables ou les individus, sur un certain nombre d'axes (ncp=5), et le 3ème argument est le plan de projection (axes=c(1,2)).

```
#L'application de l'ACP
res <- PCA(Decathlon,ncp=5,axes = c(1,2))
```





Toujours l'axe qui correspond à la plus grande valeur propre, il est mis en abscisse et l'axe qui correspond à la plus petite valeur propre, il est mis en ordonné. Il est indiqué également le pourcentage d'inertie de 27.28% du 1er axe, c'est la valeur propre du 1er axe, divisée par la somme des valeurs propres (la somme des valeurs propres = nombre des variables, parce que nous somme dans l'ACP normée), et le pourcentage d'inertie de 14.20% du 2ème axe, c'est la valeur propre du 2ème axe, divisée par la somme des valeurs propres.

Les deux premiers axes de l'analyse expriment 41.48% de l'inertie totale du jeu de données ; cela signifie que 41.48% de la variabilité totale du nuage des individus (ou des variables) est représentée dans ce plan. Le premier plan représente donc seulement une part de la variabilité contenue dans l'ensemble du jeu de donnée.

Les variables qui sont proche de la circonférence, nous avons les variables Points, Hauteur, 400m et 110m.haies, elles sont bien représentées. Les variables Longueur, Disque, Poids et 100m, sont moyennement représentées, parce qu'elles sont à mi-distance entre le centre du repère et la circonférence du cercle. Et les variables Javelot, Perche et 1500m sont mal représentées, car la distance de projection est inférieure à la moitié du rayon.

2) Justifiez les raisons pour centrer et réduire les variables

Les raisons pour centrer et réduire les variables et de rendre les variables comparables.

C'est tout simplement l'application d'un centrage et d'une réduction : La distribution obtenue aura une moyenne 0 et ecart-type1

3) Calculer l'indice KMO et les indices MSAI, conclure.

Tous d'abord nous calculons les coefficients de corrélation 2 à 2, par la commande cor( ) : nous avons 11 variables (c'est une matrice 11 x11 =121, et si on élimine les « 1 » des colonnes : 110 coefficients de corrélation qui sont calculées, on peut diviser par 2 parce que c'est une matrice symétrique, donc 55 coefficients de corrélation).

```
> cor(Decathlon) #Calcul des corrélations entres les variables
               Points     Longueur        100m        Poids      Hauteur        400m
Points      1.0000000  0.534003684 -0.39449496  0.381751059  0.14937389 -0.25528986
Longueur    0.5340037  1.000000000 -0.33530315 -0.005409433  0.21983198 -0.24874106
100m       -0.3944950 -0.335303147  1.00000000 -0.083039972 -0.08741756  0.36975251
Poids       0.3817511 -0.005409433 -0.08303997  1.000000000 -0.15027006 -0.38514326
Hauteur     0.1493739  0.219831984 -0.08741756 -0.150270057  1.00000000  0.24283835
400m       -0.2552899 -0.248741065  0.36975251 -0.385143257  0.24283835  1.00000000
110m.haies -0.5218820 -0.319132631  0.32202063 -0.208000136 -0.01315481  0.26318323
Perche      0.2817157  0.029100618 -0.12962824  0.078825571 -0.14431616 -0.08409617
Javelot     0.1403008  0.032619418 -0.05137345  0.239323386  0.05857515 -0.16811832
1500m      -0.1761942 -0.092212830  0.16976286 -0.073813257 -0.19485370  0.44025902
Disque      0.4900765  0.139401878 -0.16542064  0.208810465  0.22921721  0.04473877
             110m.haies      Perche     Javelot       1500m      Disque
Points      -0.52188197  0.28171573  0.14030077 -0.17619424  0.49007653
Longueur    -0.31913263  0.02910062  0.03261942 -0.09221283  0.13940188
100m         0.32202063 -0.12962824 -0.05137345  0.16976286 -0.16542064
Poids       -0.20800014  0.07882557  0.23932339 -0.07381326  0.20881046
Hauteur     -0.01315481 -0.14431616  0.05857515 -0.19485370  0.22921721
400m         0.26318323 -0.08409617 -0.16811832  0.44025902  0.04473877
110m.haies   1.00000000 -0.09163436 -0.27308382  0.06252703 -0.31275284
Perche      -0.09163436  1.00000000  0.17452733  0.12781384 -0.07141747
Javelot     -0.27308382  0.17452733  1.00000000 -0.28949015  0.17571192
1500m        0.06252703  0.12781384 -0.28949015  1.00000000 -0.01738037
Disque      -0.31275284 -0.07141747  0.17571192 -0.01738037  1.00000000
```

On peut remarquer des variables assez corrélées, en regardant par exemple le coefficient de corrélation entre la variable « Points » et la variable « Longueur », et des variables faiblement corrélées, en regardant par exemple le coefficient de corrélation entre la variable « Longueur » et la variable « Perche ».

Il est important de faire le test de sphéricité de Bartlett, afin d'éviter la situation extrême : la matrice de corrélation soit proche de la matrice identité (si c'est le cas, y aura aucune relation entre les variables), ce test de Bartlett fait voir si on est dans ce cas ou pas.

```
> #Test de sphéricité de Bartlett
> bartlett.test(Decathlon) #On a le p-value < 2.2e-16 (inférieur à 5%)

        Bartlett test of homogeneity of variances

data:  Decathlon
Bartlett's K-squared = 5152.3, df = 10, p-value < 2.2e-16
```

On a le p-value < 2.2 e-16 (inférieur à 0.05) donc la matrice de corrélation est différente de l'identité.

Avant de calculer l'indice KMO et des MSAi, on doit télécharger le package «psych », puis exécuter la commande KMO( ), qui est appliquée au tableau de coefficients de corrélation (c'est-à-dire cor ( ) de tableau de données).

```
> #Calcul de l'indice KMO et des MSAi
> library(psych)
> KMO(cor(Decathlon))
Kaiser-Meyer-Olkin factor adequacy
Call: KMO(r = cor(Decathlon))
Overall MSA =  0.47
MSA for each item = 
   Points  Longueur      100m     Poids   Hauteur      400m 110m.haies    Perche
     0.50      0.46      0.78      0.40      0.45      0.47       0.69      0.22
  Javelot     1500m    Disque 
     0.38      0.31      0.53 
```

En regardant les MSAi de chaque variable, on remarque qu'il y a des variables qui ont des MSAi assez faible notamment les variables : Perche, 1500m, Javelot, Poids, Hauteur, Longueur et 400m, qui ont des MSAi < 0.5.

4) Calculer les valeurs propres, le pourcentage d'inertie de chaque valeur propre ainsi que le cumul des pourcentages d'inertie

On exécute la commande : res$eig

```
> #calcul des valeurs propres et la matrice de corrélation
> res$eig
        eigenvalue percentage of variance cumulative percentage of variance
comp 1   3.0011116              27.282833                          27.28283
comp 2   1.5617918              14.198107                          41.48094
comp 3   1.3286182              12.078348                          53.55929
comp 4   1.2070236              10.972941                          64.53223
comp 5   0.9909887               9.008988                          73.54122
comp 6   0.7397638               6.725125                          80.26634
comp 7   0.6818340               6.198491                          86.46483
comp 8   0.5355775               4.868887                          91.33372
comp 9   0.4830787               4.391624                          95.72534
comp 10  0.3229338               2.935761                          98.66111
comp 11  0.1472784               1.338895                         100.00000
```

La première colonne « eigenvalue » correspond aux valeurs propres, la deuxième colonne « percentage of variance » correspond au pourcentage d'inertie de chaque valeur propre et la troisième colonne « cumulative percentage of variance » correspond au cumul des pourcentages d'inertie.

On remarque qu'il y a 4 valeurs propres supérieurs à 1, pour chaque valeur propre nous avons le taux d'inertie qui est le rapport de la valeur propre sur la somme des valeurs propres (la somme des valeurs propres = le nombre des variables car là ce sont des valeurs propres de la matrice de corrélation qui n'a que des « 1 » dans la diagonale, donc la trace c'est le nombre de variables)

Nous avons également le pourcentage d'inertie et le cumul des pourcentages d'inertie, donc le 1^{er} axe a un taux d'inertie de 27.28% et le $2^{ème}$ axe a un taux d'inertie de 14.20%, et l'espace constitué du 1^{er} et du $2^{ème}$ axe a un taux d'inertie de 41.48%, l'espace constitué du 1^{er} et du $2^{ème}$ et du $3^{ème}$ axe a un taux d'inertie de 53.56% … Et si on prend tout l'espace (les 11 axes), nous avons 100% d'inertie.

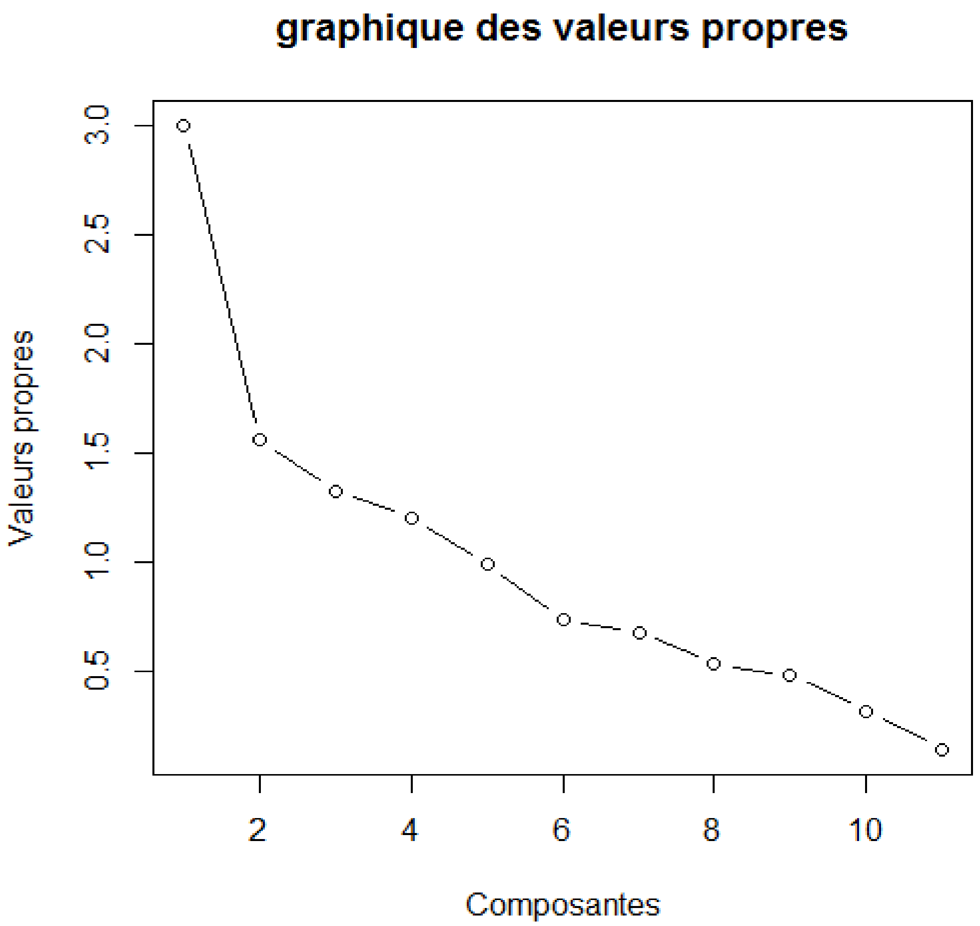
5) Tracer le graphique des valeurs propres

```
#Graphique des valeurs propres
plot(1:11,res$eig[,1],type="b",ylab="Valeurs propres",xlab="Composantes",main="graphique des valeurs propres")
```

res$eig[,1] correspond à la 1ère colonne celle des valeurs propres, nous avons spécifié le nom de chaque axe et le nom du graphe.

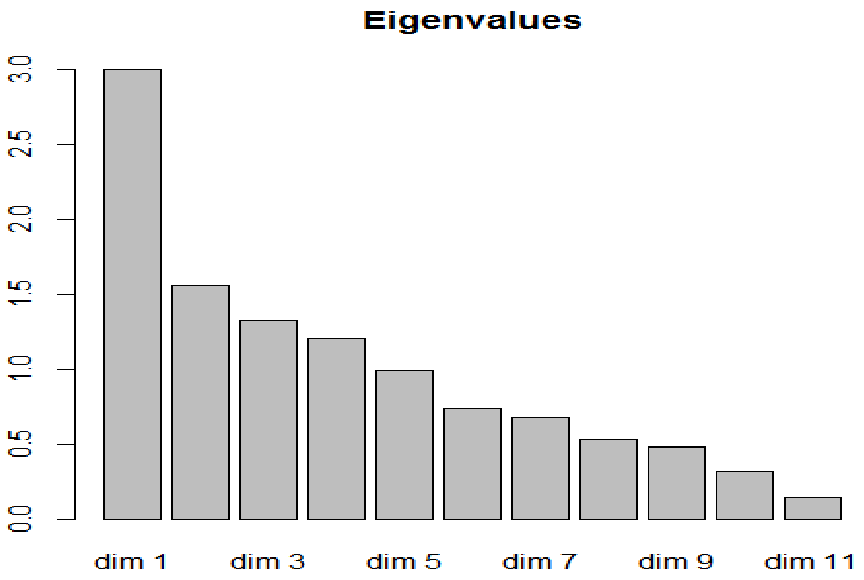
On également faire un graphe en barre :

```
barplot(res$eig[,1],main="Eigenvalues", names.arg=paste("dim",1:nrow(res$eig)))
```



^

On remarque une décroissante assez importante entre la 1ère et la 2ème valeur propre (donc la 1ère et plus de deux fois la 2ème), donc là il va falloir choisir la dimension du sous espace. Si on applique la règle des valeurs propres supérieures à 1, donc on va choisir les 4 valeurs propres > 1 (un sous espace de dimension 4), mais en remarque qu'il existe une valeur propre = 0.990988, c'est très proche de 1.



6) Déterminer la dimension du sous espace en utilisant la règle $\frac{var(V_2)}{var(V)} < 0{,}05$

Pour le choix de dimension du sous-espace, on opte pour la règle du rapport de variances :

```
> #Règle rapport des variances : 5 axes
> var(res$eig[5:11,1])*6/(var(res$eig[,1])*10) #c'est égale à 0.07385619>0.05
[1] 0.07385619
> var(res$eig[6:11,1])*5/(var(res$eig[,1])*10) #c'est égale à 0.03907991<0.05
[1] 0.03907991
```

Ici nous avons calculé deux rapport de variance, le premier, on élimine les 7 dernières valeurs propres, donc choisir que les 4 premières valeurs propres (celles qui sont supérieures à 1), c'est-à-dire un sous-espace de dimensions 4, on a un rapport de 0.07385619. Et le deuxième, on élimine les 6 dernières valeurs propres, donc choisir que les 5 premières valeurs propres (un sous-espace de dimension 5), on a un rapport de 0.03907991.

L'écart entre les deux rapports est large, c'est presque la moitié donc c'est bien significatif (c'est presque divisé par 2). Donc on va retenir les 5 plus grandes valeurs propres, c'est-à-dire un sous-espace de dimension 5.

On remarque que le choix de dimension 5, ne respecte pas la règle des valeurs propres supérieurs à 1, mais on remarque que 0.9909887 c'est assez proche de 1 et si on combine le tout, cette règle se justifie devant un seuil de 5%.

Nuage des variables :

7) Calculer le cos2 des variables sur le sous espace

Nous allons exécuter la commande res$var :

```
> res$var
$coord
                Dim.1       Dim.2       Dim.3       Dim.4       Dim.5
Points      0.8336822  0.18931385  0.25260148 -0.02766561 -0.03535797
Longueur    0.5768403  0.33132476 -0.02718865 -0.46588868  0.01668859
100m       -0.5979026 -0.03641299  0.05720007  0.44540580 -0.01620542
Poids       0.4729398 -0.42277937  0.15405436  0.40066750 -0.36420340
Hauteur     0.1395882  0.73694019 -0.33972587  0.12517611  0.25163933
400m       -0.5672659  0.58094679  0.32031032  0.23546891  0.16354241
110m.haies -0.6825048 -0.06025988 -0.20531815 -0.04375433  0.05203028
Perche      0.2218296 -0.29220218  0.53726652 -0.12636500  0.65873915
Javelot     0.4008398 -0.20472507 -0.15244803  0.56868571  0.49170897
1500m      -0.3758928  0.14884280  0.79152764 -0.08854523 -0.14969029
Disque      0.4836494  0.44994195  0.19618178  0.45830583 -0.25623329

$cor
                Dim.1       Dim.2       Dim.3       Dim.4       Dim.5
Points      0.8336822  0.18931385  0.25260148 -0.02766561 -0.03535797
Longueur    0.5768403  0.33132476 -0.02718865 -0.46588868  0.01668859
100m       -0.5979026 -0.03641299  0.05720007  0.44540580 -0.01620542
Poids       0.4729398 -0.42277937  0.15405436  0.40066750 -0.36420340
Hauteur     0.1395882  0.73694019 -0.33972587  0.12517611  0.25163933
400m       -0.5672659  0.58094679  0.32031032  0.23546891  0.16354241
110m.haies -0.6825048 -0.06025988 -0.20531815 -0.04375433  0.05203028
Perche      0.2218296 -0.29220218  0.53726652 -0.12636500  0.65873915
Javelot     0.4008398 -0.20472507 -0.15244803  0.56868571  0.49170897
1500m      -0.3758928  0.14884280  0.79152764 -0.08854523 -0.14969029
Disque      0.4836494  0.44994195  0.19618178  0.45830583 -0.25623329
```

```
$cos2
                Dim.1       Dim.2        Dim.3        Dim.4        Dim.5
Points     0.69502598 0.035839734 0.0638075071 0.0007653862 0.0012501860
Longueur   0.33274472 0.109776094 0.0007392227 0.2170522651 0.0002785091
100m       0.35748756 0.001325906 0.0032718479 0.1983863263 0.0002626156
Poids      0.22367209 0.178742395 0.0237327445 0.1605344455 0.1326441187
Hauteur    0.01948488 0.543080849 0.1154136696 0.0156690593 0.0633223514
400m       0.32179059 0.337499176 0.1025987036 0.0554456062 0.0267461208
110m.haies 0.46581274 0.003631253 0.0421555423 0.0019144410 0.0027071504
Perche     0.04920836 0.085382113 0.2886553082 0.0159681125 0.4339372637
Javelot    0.16067251 0.041912355 0.0232404008 0.3234034321 0.2417777140
1500m      0.14129540 0.022154180 0.6265159997 0.0078402577 0.0224071838
Disque     0.23391675 0.202447759 0.0384872905 0.2100442294 0.0656555003

$contrib
                Dim.1       Dim.2       Dim.3       Dim.4       Dim.5
Points     23.1589517  2.29478305  4.80254639  0.06341104  0.12615542
Longueur   11.0873826  7.02885579  0.05563845 17.98243813  0.02810416
100m       11.9118384  0.08489646  0.24625944 16.43599451  0.02650037
Poids       7.4529747 11.44470044  1.78627267 13.30002584 13.38502819
Hauteur     0.6492553 34.77293476  8.68674435  1.29815687  6.38981559
400m       10.7223800 21.60974163  7.72221100  4.59358110  2.69893293
110m.haies 15.5213403  0.23250559  3.17288602  0.15860842  0.27317672
Perche      1.6396711  5.46693309 21.72597818  1.32293296 43.78831542
Javelot     5.3537666  2.68360702  1.74921585 26.79346472 24.39762539
1500m       4.7081022  1.41851046 47.15545687  0.64955299  2.26109374
Disque      7.7943371 12.96253171  2.89679077 17.40183341  6.62525207
```

Donc nous avons la contribution, le cos 2, le coefficient de corrélation et la projection, on a démontré dans le cours que la projection c'est le coefficient de corrélation, donc on a les mêmes valeurs qui se représentent dans « coord » et « cor ». On remarque que nous avons des valeurs positives et négatives, et pour le cos2 c'est en fait « coord » ou « cor » élevé au carré, parce que la distance de la variable c'est 1, car nous sommes dans l'ACP normée, donc nous avons que les variables positives.

Et pour la contribution, c'est la projection au carré divisée par la valeur propre multiplié par 100, donc là si on fait la somme des contributions pour un seul axe, nous allons trouver 100, parce que c'est exprimé en pourcentage (ou tout simplement le cos2 divisé par la variable propre multiplié par 100).

8) Distinguer les variables bien représentées, moyennement représentées et faiblement représentées sur le sous espace

Pour le cos2, on va raisonner sur la répartition des variables sur tout le sous espace de dimension 5, donc on va faire le cumul des cos2 sur tout le sous espace :

```
> #Cumul des cos2
> print(t(apply(res$var$cos2,1,cumsum)),digit=2)
            Dim.1 Dim.2 Dim.3 Dim.4 Dim.5
Points      0.695  0.73  0.79  0.80  0.80
Longueur    0.333  0.44  0.44  0.66  0.66
100m        0.357  0.36  0.36  0.56  0.56
Poids       0.224  0.40  0.43  0.59  0.72
Hauteur     0.019  0.56  0.68  0.69  0.76
400m        0.322  0.66  0.76  0.82  0.84
110m.haies  0.466  0.47  0.51  0.51  0.52
Perche      0.049  0.13  0.42  0.44  0.87
Javelot     0.161  0.20  0.23  0.55  0.79
1500m       0.141  0.16  0.79  0.80  0.82
Disque      0.234  0.44  0.47  0.68  0.75
```

En regardant la dimension 5, on remarque que toutes les variables sont bien représentées à part les variables « 100m », « 110.haies » et Longueur qui sont moyennement représentées.

Si on compare la qualité de projection entre la dimension 4 et la dimension 5, donc nous avons les variables « Poids », « Perche » et Javelot qui étaiement moyennement représentées à la dimension 4, et passent à une bonne qualité de projection en dimension 5.

Les variables qui exigent plus de dimension par rapport aux autres, on peut remarquer par exemple les variables « Perche » (qui lui a fallu 5 dimension pour qu'elle passe à bonne représentation), « Javelot » (c'est à partir de la 4ème dim.) et « 1500m » (c'est à partir de la 3ème dim.), c'est les mêmes variables qui ont des MSAi faibles.

Donc nous avons vu la qualité de représentation des variables sur le sous espace de projection, qui est de dimension 5.

Pour distinguer les variables bien représentées, moyennement représentées et faiblement représentées sur le sous espace, on opte un kmeans de centers 3, pour avoir 3 classes :

```
> cat1<-kmeans(cum1[,4],centers=3,nstart=5)
> cat1
K-means clustering with 3 clusters of sizes 3, 5, 3

Cluster means:
       [,1]
1 0.8035262
2 0.5298220
3 0.6796189

Clustering vector:
    Points   Longueur       100m      Poids    Hauteur       400m 110m.haies     Perche
         1          3          2          2          3          1          2          2
   Javelot      1500m     Disque
         2          1          3

Within cluster sum of squares by cluster:
[1] 0.0002887890 0.0130248234 0.0005974211
 (between_SS / total_SS =  91.3 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"
[7] "size"         "iter"         "ifault"
```

On remarque que la 1ère classe, ce sont les variables bien représentées (il y a 3 variables : Ponits, 400m et 1500m), en regardant le Cluster means de cette classe, on a 0.8035262.

La 2ème classe, ce sont les variables faiblement représentées (il a y 5 variables : 100m, Poids, 110m.haies, Perche et Javelot) en regardant le Cluster means de cette classe, on a 0.5298220.

La 3ème classe, ce sont les variables moyennement représentées (il y a 3 variables : Longeur, Hauteur et Disque), en regardant le Cluster means de cette classe, on a 0.6796189.

9) Calculer la contribution des variables dans chaque axe du sous espace

Pour avoir les contributions, il suffit d'exécuter la commande :

```
#Contribution des variables au sous espace
cont<-res$var$contrib
```

```
> #Contribution des variables au sous espace
> res$var$contrib
                Dim.1       Dim.2       Dim.3       Dim.4       Dim.5
Points     23.1589517  2.29478305  4.80254639  0.06341104  0.12615542
Longueur   11.0873826  7.02885579  0.05563845 17.98243813  0.02810416
100m       11.9118384  0.08489646  0.24625944 16.43599451  0.02650037
Poids       7.4529747 11.44470044  1.78627267 13.30002584 13.38502819
Hauteur     0.6492553 34.77293476  8.68674435  1.29815687  6.38981559
400m       10.7223800 21.60974163  7.72221100  4.59358110  2.69893293
110m.haies 15.5213403  0.23250559  3.17288602  0.15860842  0.27317672
Perche      1.6396711  5.46693309 21.72597818  1.32293296 43.78831542
Javelot     5.3537666  2.68360702  1.74921585 26.79346472 24.39762539
1500m       4.7081022  1.41851046 47.15545687  0.64955299  2.26109374
Disque      7.7943371 12.96253171  2.89679077 17.40183341  6.62525207
```

10) Appliquer la CAH au tableau des contributions des variables aux axes du sous espace, interpréter les résultats

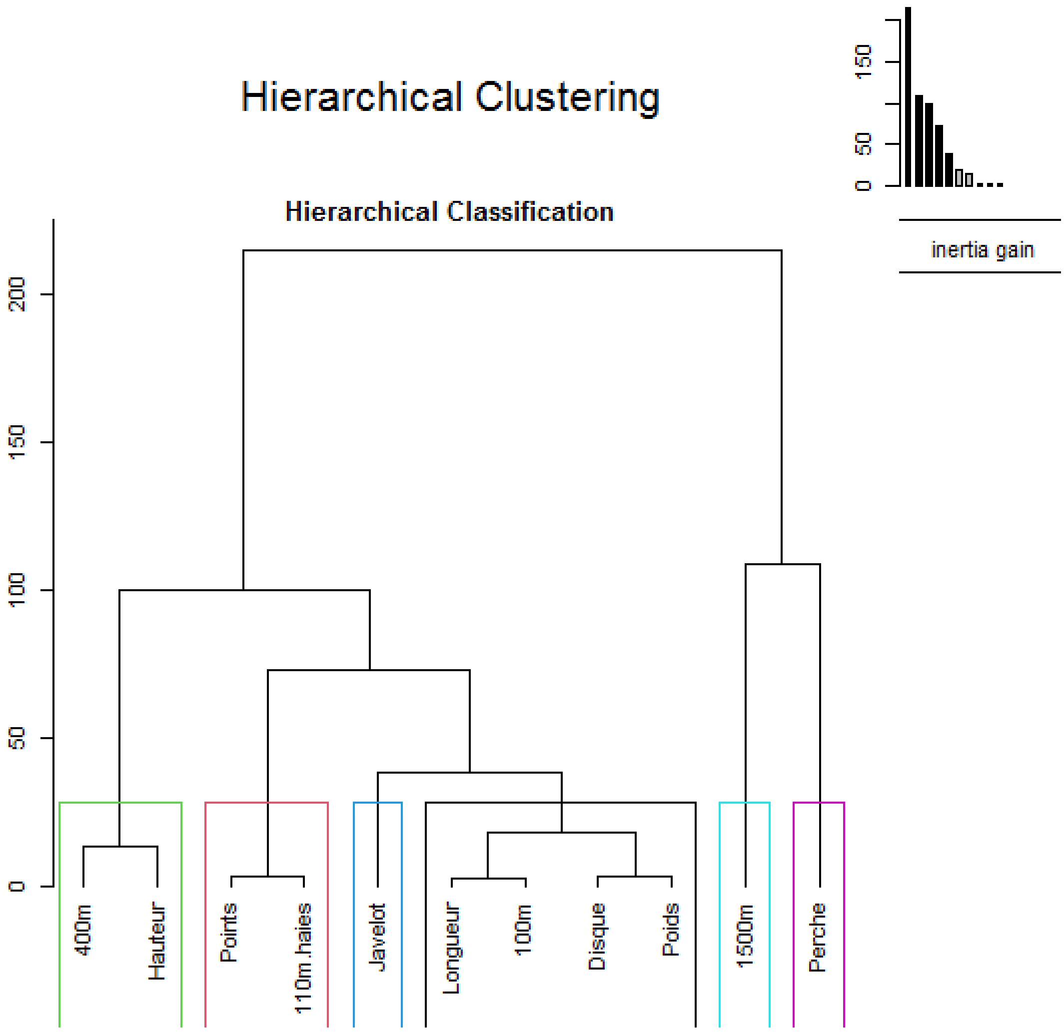
Le faite d'applique le CAH directement sur le vecteur «cont », ça ne va pas marcher, car nous avons le nom d'une variable qui est vide, pour cela nous allons exécuter la commande write.table ( ) ça va nous créer un fichier Excel qui contient les contributions des variables, puis nous allons remplir la case vide sur Excel manuellement ensuite lire le fichier.

```
write.table(cont,"Cont_Var.csv",sep=";",col.names=TRUE,dec=',', row.names=TRUE)
Don<-read.csv2("C:/Users/hp/Desktop/Cont_Var1.csv",row.names=1)
View(Don)
```

	Dim.1	Dim.2	Dim.3	Dim.4	Dim.5
Points	23.1589517	2.29478305	4.80254639	0.06341104	0.12615542
Longueur	11.0873826	7.02885580	0.05563846	17.98243813	0.02810416
100m	11.9118384	0.08489646	0.24625944	16.43599451	0.02650037
Poids	7.4529747	11.44470044	1.78627267	13.30002584	13.38502819
Hauteur	0.6492553	34.77293476	8.68674435	1.29815687	6.38981559
400m	10.7223800	21.60974163	7.72221100	4.59358110	2.69893293
110m.haies	15.5213403	0.23250559	3.17288602	0.15860842	0.27317672
Perche	1.6396711	5.46693309	21.72597818	1.32293296	43.78831542
Javelot	5.3537666	2.68360702	1.74921585	26.79346472	24.39762539
1500m	4.7081022	1.41851046	47.15545687	0.64955299	2.26109374
Disque	7.7943371	12.96253171	2.89679077	17.40183341	6.62525207

Ensuite, exécuter la commande HCPC( ) : la mention "nb.clust=-1" pour avoir une coupe systématique.

```
Cat_var<-HCPC(Don,nb.clust=-1)
```



La CAH nous a donné 6 classes de variables, nous avons 11 variables qui ont été catégorisé en 6 classes selon leur contribution aux 5 dimensions de notre sous espace.

Pour l’interprétation de ces 6 classes de variables, on utilise la commande cat_var$desc.var :

```
> Cat_var<-HCPC(Don,nb.clust=-1)
> Cat_var$desc.var #L'interprétatation des 6 classes de variables

Link between the cluster variable and the quantitative variables
================================================================
           Eta2      P-value
Dim.3 0.9963308 4.412895e-06
Dim.4 0.9796079 3.155708e-04
Dim.5 0.9312603 6.241073e-03
Dim.2 0.8372356 4.826512e-02

Description of each cluster by quantitative variables
=====================================================
$`1`
NULL

$`2`
        v.test Mean in category Overall mean sd in category Overall sd    p.value
Dim.1 2.477971         19.34015     9.090909       3.818806   6.165794 0.01321318

$`3`
NULL

$`4`
        v.test Mean in category Overall mean sd in category Overall sd     p.value
Dim.2 2.772841         28.19134     9.090909       6.581597   10.26861 0.005556919

$`5`
        v.test Mean in category Overall mean sd in category Overall sd    p.value
Dim.3 2.844721         47.15546     9.090909              0   13.38077 0.00444503

$`6`
        v.test Mean in category Overall mean sd in category Overall sd    p.value
Dim.5 2.654124         43.78832     9.090909              0   13.07302 0.00795146
```

La **classe 1** est composée de variables : *Longueur 100m, Disque et Poids*. Ce groupe est caractérisé par une classe nulle, qui signifie que pour chaque axe, la contribution moyenne dans cette classe est très proche de la contribution moyenne (tous les v. tests ne sont pas significatifs).

La **classe 2** est composée de variables Points et 110m.haies. Ce groupe a un v. test de 2.477971 (c'est un v. test assez grand pour la dim.1), donc ces variables contribuent à la dimension 1.

La **classe 3** est composée de variable 400m et Hauteur.Ce groupe est caractérisé par une classe nulle, qui signifie que pour chaque axe, la contribution moyenne dans cette classe est très proche de la contribution moyenne (tous les v. tests ne sont pas significatifs).

La **classe 4** composée d'une seule variable *Javelot*. Ce groupe a un v. test de 2.772841 (c'est un v. test assez grand pour la dim.2), donc cette variable contribue à la dimension 2.

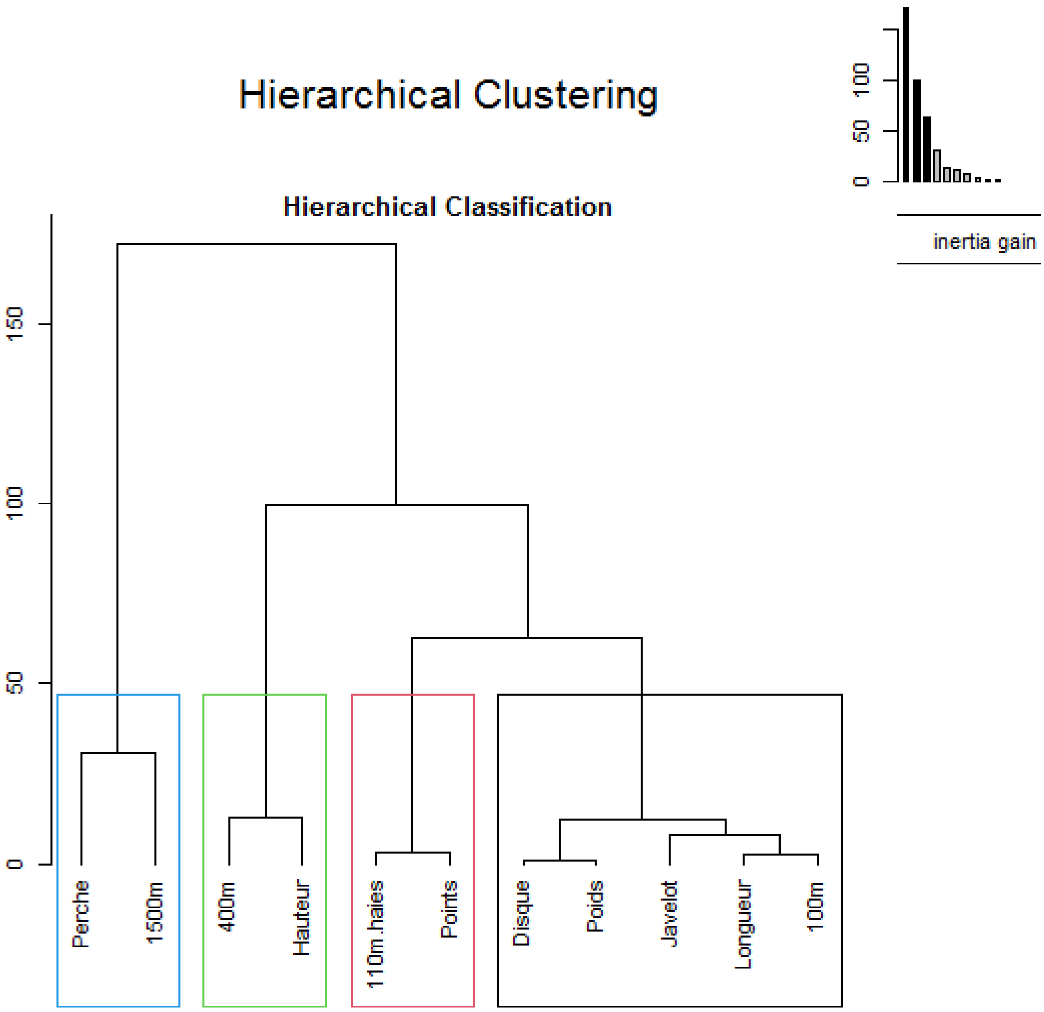
La **classe 5** est composée d'une seul variable *1500m*. Ce groupe a un v. test de 2.844721 (c'est un v. test assez grand pour la dim.3), donc cette variable contribue à la dimension 3.

La **classe 6** est composée d'une seule variable *Perche*. Ce groupe a un v. test de 2.654124 (c'est un v. test assez grand pour la dim.5), donc cette variable contribue à la dimension 5.

Remarque : Le fait d'avoir 2 classes NULL est un problème dans les mesures où on a 2 groupes de variables qui ont le même comportement et cette séparation est difficile à justifier. Cet exemple montre que le seuil de 5%, n'est pas toujours pertinent, la règle des valeurs propres supérieures à 1 se justifie, bien qu'on aura du mal à justifier le rejeter de la valeur 0.9909887. On pourrait choisir un sous espace de dimension 4. Les 2 choix de dimensions restent valables.

Voilà l'intérêt de faire une CAH, elle nous aide à catégoriser les variables selon la contribution au sous espace de projection qui a été identifié par l'analyse factorielle.

Dans ce qui suit, nous avons fait une CAH, en retenant seulement 4 dimensions, on a éliminé la 5ème dimension (la contribution au 5ème axe), pour voir comment ça va se faire la catégorisation des variables, si on élimine le 5ème axe (dans ce cas, on ne respecte plus le seuil de 5% du rapport de variances).



Cette fois, La CAH nous a donné 4 classes de variables, nous avons 11 variables qui ont été catégorisé en 4 classes selon leur contribution aux 4 dimensions du sous espace.

```
> Don4<-Don[,1:4]
> Cat_var4<-HCPC(Don4, nb.clust=-1)
> Cat_var4$desc.var

Link between the cluster variable and the quantitative variables
================================================================
           Eta2      P-value
Dim.4 0.8818673 0.001257074
Dim.3 0.8320593 0.004213176
Dim.2 0.8115444 0.006248190
Dim.1 0.7269305 0.021975311

Description of each cluster by quantitative variables
=====================================================
$`1`
        v.test Mean in category Overall mean sd in category Overall sd     p.value
Dim.4 2.953938         18.38275     9.090909       4.505253   9.080502 0.003137474

$`2`
        v.test Mean in category Overall mean sd in category Overall sd    p.value
Dim.1 2.477971         19.34015     9.090909       3.818806   6.165794 0.01321318

$`3`
        v.test Mean in category Overall mean sd in category Overall sd     p.value
Dim.2 2.772841         28.19134     9.090909       6.581597   10.26861 0.005556919

$`4`
        v.test Mean in category Overall mean sd in category Overall sd     p.value
Dim.3 2.824148         34.44072     9.090909       12.71474   13.38077 0.004740647
```

La **classe 1** est composée de variables Disque, Poids, Javelot, Longueur et 100m. Ce groupe a un v. test de 2.953938 (c'est un v. test assez grand pour la dim.4), donc ces variables contribuent à la dimension 4.

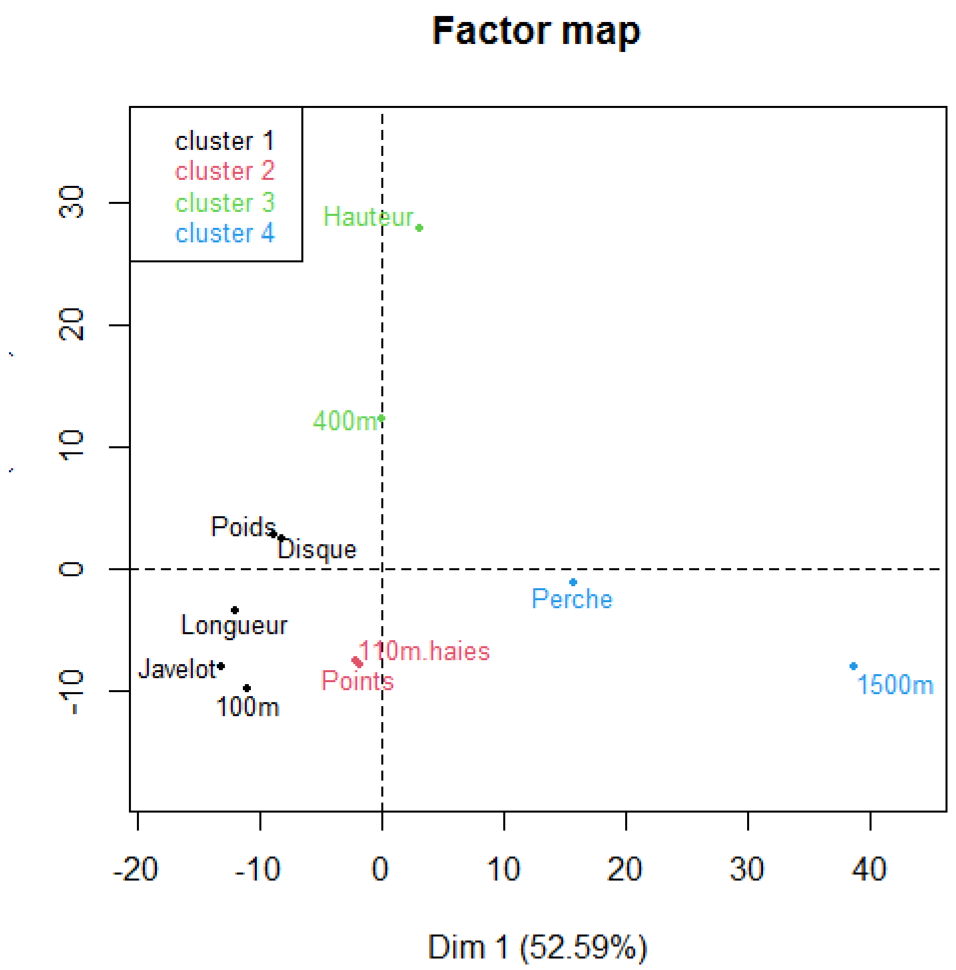
La **classe 2** est composée de variables Points et 110m.haies. Ce groupe a un v. test de 2.477971 (c'est un v. test assez grand pour la dim.1), donc ces variables contribuent à la dimension 1.

La **classe 3** est composée de variables 400m et Hauteur. Ce groupe a un v. test de 2.772841 (c'est un v. test assez grand pour la dim.2), donc ces variables contribuent à la dimension 2.

La **classe 4** est composée de variables Perche et 1500m. Ce groupe a un v. test de 2.824148 (c'est un v. test assez grand pour la dim.3), donc ces variables contribuent à la dimension 3.

Ici on a une classe par axe.

11) Tracer le nuage des variables projeté sur les 2 premiers axes.



12) Indiquer les variables qui sont relativement bien corrélées (positivement et négativement) avec les axes du 1er plan factoriel

On remarque que les variables telles que 1500m sont bien corrélées positivement avec l'axe 1, et les variables telles que Javelot et Longueur sont bien corrélées négativement avec l'axe 1.

On remarque également que les variables telles que Hauteur, sont bien corrélées positivement avec l'axe 2, et les variables telles que 100m et 1500m sont bien corrélées négativement avec l'axe 2.

Nuage des individus :

13) Calculer le cos2 des individus sur le sous espace

Nous allons exécuter la commande res$ind :

$cos2

	Dim.1	Dim.2	Dim.3	Dim.4	Dim.5
1	6.280246e-01	1.206042e-02	8.493158e-02	1.253461e-01	8.069704e-04
2	5.529138e-01	1.150674e-01	8.879101e-02	9.085521e-03	1.881880e-04
3	4.566614e-01	5.319492e-02	5.327786e-03	1.221267e-02	3.914390e-01
4	8.104125e-02	1.316101e-01	1.611306e-01	8.658756e-02	4.237093e-01
5	1.640374e-01	5.467255e-03	4.622074e-02	6.012139e-01	1.102764e-01
6	5.850679e-03	4.789282e-02	1.520718e-01	3.758557e-01	1.613146e-02
7	7.464275e-04	8.483865e-06	1.925347e-01	7.324654e-03	1.186449e-01
8	9.969033e-04	1.285077e-01	1.878929e-01	2.089764e-01	3.230251e-01
9	3.617372e-05	1.771674e-01	4.073940e-02	4.804259e-02	5.941213e-01
10	1.157561e-01	4.453556e-03	9.743890e-02	2.664669e-01	3.028398e-02
11	7.103578e-01	1.394663e-02	2.783212e-03	1.879877e-01	3.747529e-03
12	5.996967e-01	9.206460e-04	1.850968e-02	2.302362e-02	1.334824e-01
13	4.885947e-01	2.745925e-02	1.045640e-01	1.389373e-01	5.926589e-04
14	3.518185e-01	1.127345e-02	1.042201e-01	1.611996e-01	6.901153e-02
15	1.669206e-01	8.281354e-03	6.144704e-01	6.069296e-03	2.039546e-02
16	5.745199e-02	9.968356e-03	3.486605e-02	4.604471e-01	1.419010e-01
17	1.127220e-03	4.074251e-01	1.630782e-01	5.820546e-03	9.238045e-02
18	6.174095e-02	9.686094e-02	1.653247e-02	4.081779e-02	4.701843e-01
19	7.411926e-03	1.993917e-02	2.592895e-01	2.613787e-01	1.636199e-02
20	2.480135e-01	6.618237e-02	7.404833e-03	9.826471e-02	3.619916e-01
21	4.966271e-01	1.663743e-02	2.899210e-02	3.925479e-02	9.605275e-02
22	2.765747e-01	3.408012e-01	6.212305e-02	4.221072e-02	2.093760e-01
23	9.006289e-05	1.724635e-01	9.687870e-03	3.698680e-02	2.867672e-01
24	2.411593e-01	3.396762e-01	5.782555e-04	1.634938e-01	2.223711e-03
25	6.942150e-02	2.029499e-02	9.181696e-03	1.108608e-03	1.035876e-02
26	3.669831e-02	8.704329e-02	2.760968e-01	3.581091e-01	3.621972e-02
27	1.486846e-01	3.723241e-02	3.570588e-01	5.374913e-02	6.915596e-02
28	8.891981e-02	2.940653e-02	3.185160e-01	3.085598e-01	2.764417e-02
29	1.191520e-01	7.166416e-02	3.944834e-01	3.212137e-01	8.841243e-04
30	8.497503e-03	2.371844e-01	5.271525e-03	6.266624e-01	3.220512e-03
31	7.142689e-01	2.593745e-02	9.608215e-02	4.951933e-02	7.555938e-02
32	1.125318e-01	1.421054e-01	9.035703e-04	2.052009e-01	1.099205e-01
33	8.895913e-03	2.074262e-03	9.891930e-02	1.099311e-01	1.397368e-03
34	1.588662e-01	3.335598e-01	1.051651e-01	3.611360e-02	2.585339e-01
35	8.164742e-02	3.641456e-02	9.315687e-02	5.716837e-01	1.186943e-01
36	5.659020e-01	2.023854e-02	1.280420e-01	2.051108e-01	5.125524e-02
37	7.919479e-01	1.608045e-02	5.439169e-02	1.472678e-03	3.365622e-03
38	6.404390e-01	3.733722e-03	1.691975e-02	1.261287e-01	2.856758e-02
39	6.670581e-01	8.402641e-02	1.244051e-01	2.431011e-02	5.288150e-02
40	7.400835e-01	7.553426e-03	1.251502e-01	8.895402e-02	1.003021e-02
41	2.438266e-02	4.640706e-02	2.244758e-02	2.847624e-04	2.331500e-01
42	2.654509e-01	2.873835e-01	7.478293e-02	1.189958e-01	7.546787e-04
43	1.023163e-02	4.467972e-01	4.302052e-03	1.468591e-01	4.059023e-02
44	7.165171e-02	2.752586e-02	9.683800e-02	4.633953e-02	4.339059e-03
45	8.911155e-02	5.574183e-02	6.292607e-02	1.148681e-02	2.650112e-03
46	5.123758e-02	1.049643e-01	2.304235e-05	5.275780e-02	3.021962e-01
47	1.387239e-01	2.139705e-01	1.382686e-01	9.471810e-02	7.228510e-02
48	7.153652e-02	5.347585e-02	2.992968e-02	9.012617e-05	1.935772e-01
49	2.120618e-03	2.695353e-02	7.136147e-01	5.689782e-02	2.389787e-02
50	6.938834e-03	1.990706e-01	6.164683e-02	2.514820e-02	1.631653e-01
51	1.140386e-01	3.411162e-01	2.548638e-02	1.726588e-03	2.359509e-06
52	1.856380e-01	6.327873e-01	1.013410e-02	1.330947e-01	1.949669e-03
53	3.410056e-03	6.013948e-01	1.136582e-02	7.053038e-03	6.738487e-02
54	7.049271e-02	3.665135e-01	6.750777e-03	5.593388e-02	4.398604e-02
55	9.857271e-04	4.771435e-01	1.522882e-01	3.188043e-02	3.138180e-02

14) Distinguer les individus bien représentées, moyennement représentées et faiblement représentées sur le sous espace.

Pour le cos2, on va raisonner sur la répartition des individus sur tout le sous espace de dimension 5, donc on va faire le cumul des cos2 sur tout le sous espace :

```
> #Cumul des cos2
> cum<-print(t(apply(res$ind$cos2,1,cumsum)),digit=2)
     Dim.1   Dim.2 Dim.3 Dim.4 Dim.5
1  6.3e-01 0.64009 0.725 0.850  0.85
2  5.5e-01 0.66798 0.757 0.766  0.77
3  4.6e-01 0.50986 0.515 0.527  0.92
4  8.1e-02 0.21265 0.374 0.460  0.88
5  1.6e-01 0.16950 0.216 0.817  0.93
6  5.9e-03 0.05374 0.206 0.582  0.60
7  7.5e-04 0.00075 0.193 0.201  0.32
8  1.0e-03 0.12950 0.317 0.526  0.85
9  3.6e-05 0.17720 0.218 0.266  0.86
10 1.2e-01 0.12021 0.218 0.484  0.51
11 7.1e-01 0.72430 0.727 0.915  0.92
12 6.0e-01 0.60062 0.619 0.642  0.78
13 4.9e-01 0.51605 0.621 0.760  0.76
14 3.5e-01 0.36309 0.467 0.629  0.70
15 1.7e-01 0.17520 0.790 0.796  0.82
16 5.7e-02 0.06742 0.102 0.563  0.70
17 1.1e-03 0.40855 0.572 0.577  0.67
18 6.2e-02 0.15860 0.175 0.216  0.69
19 7.4e-03 0.02735 0.287 0.548  0.56
20 2.5e-01 0.31420 0.322 0.420  0.78
21 5.0e-01 0.51326 0.542 0.582  0.68
22 2.8e-01 0.61738 0.679 0.722  0.93
23 9.0e-05 0.17255 0.182 0.219  0.51
24 2.4e-01 0.58084 0.581 0.745  0.75
25 6.9e-02 0.08972 0.099 0.100  0.11
26 3.7e-02 0.12374 0.400 0.758  0.79
27 1.5e-01 0.18592 0.543 0.597  0.67
28 8.9e-02 0.11833 0.437 0.745  0.77
29 1.2e-01 0.19082 0.585 0.907  0.91
30 8.5e-03 0.24568 0.251 0.878  0.88

31 7.1e-01 0.74021 0.836 0.886  0.96
32 1.1e-01 0.25464 0.256 0.461  0.57
33 8.9e-03 0.01097 0.110 0.220  0.22
34 1.6e-01 0.49243 0.598 0.634  0.89
35 8.2e-02 0.11806 0.211 0.783  0.90
36 5.7e-01 0.58614 0.714 0.919  0.97
37 7.9e-01 0.80803 0.862 0.864  0.87
38 6.4e-01 0.64417 0.661 0.787  0.82
39 6.7e-01 0.75108 0.875 0.900  0.95
40 7.4e-01 0.74764 0.873 0.962  0.97
41 2.4e-02 0.07079 0.093 0.094  0.33
42 2.7e-01 0.55283 0.628 0.747  0.75
43 1.0e-02 0.45703 0.461 0.608  0.65
44 7.2e-02 0.09918 0.196 0.242  0.25
45 8.9e-02 0.14485 0.208 0.219  0.22
46 5.1e-02 0.15620 0.156 0.209  0.51
47 1.4e-01 0.35269 0.491 0.586  0.66
48 7.2e-02 0.12501 0.155 0.155  0.35
49 2.1e-03 0.02907 0.743 0.800  0.82
50 6.9e-03 0.20601 0.268 0.293  0.46
51 1.1e-01 0.45515 0.481 0.482  0.48
52 1.9e-01 0.81843 0.829 0.962  0.96
53 3.4e-03 0.60480 0.616 0.623  0.69
54 7.0e-02 0.43701 0.444 0.500  0.54
55 9.9e-04 0.47813 0.630 0.662  0.69
```

Pour distinguer les individus bien représentées, moyennement représentées et faiblement représentées sur le sous espace, on opte un kmeans de centers 3, pour avoir 3 classes :

```
> #kmeans (3 classes)
> cat3<-kmeans(cum[,4],centers=3,nstart=5) #les individus qui sont bien, moyennement
> #faiblement représenté
> cat3
K-means clustering with 3 clusters of sizes 22, 21, 12

Cluster means:
       [,1]
1 0.8302791
2 0.5567996
3 0.2027975

Clustering vector:
 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 
 1  1  2  2  1  2  3  2  3  2  1  2  1  2  1  2  2  3  2  2  2  1  3  1  3  1  2  1  1  1  1 
32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 
 2  3  2  1  1  1  1  1  1  3  1  2  3  3  3  2  3  1  3  2  1  2  2  2 

Within cluster sum of squares by cluster:
[1] 0.12123270 0.09387261 0.03949442
 (between_SS / total_SS =  92.4 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"   
[7] "size"         "iter"         "ifault"
```

On remarque que la $1^{ère}$ classe, ce sont les individus bien représentées (il y a 22 individus comme par exemple les individus 1, 2, 5, 11, 49…), en regardant le Cluster means de cette classe, on a 0.8302791>0.5.

La $2^{ème}$ classe, ce sont les individus moyennement représentées (il a y 21 individus comme par exemple les individus 3, 4, 6, 8, 55…), en regardant le Cluster means de cette classe, on a 0.5567996.

La $3^{ème}$ classe, ce sont les individus faiblement représentées (il y a 12 individus comme par exemple les individus 7, 9, 18, 23, 50…), en regardant le Cluster means de cette classe, on a 0.2027975<0.5.

15) Calculer la contribution des individus dans chaque axe du sous espace

Pour avoir les contributions, il suffit d'exécuter la commande : resindcontrib[,1:5]

```
> #Contribution des individus au sous espace
> res$ind$contrib[,1:5]
          Dim.1        Dim.2        Dim.3        Dim.4        Dim.5
1  4.706586e+00 1.736804e-01 1.437740e+00  2.335642614 1.831472e-02
2  4.363392e+00 1.744931e+00 1.582770e+00  0.178271953 4.497511e-03
3  4.621099e+00 1.034380e+00 1.217810e-01  0.307275644 1.199578e+01
4  3.704495e-01 1.156035e+00 1.663730e+00  0.984112645 5.865490e+00
5  4.703214e-01 3.012176e-02 2.993442e-01  4.285952870 9.575204e-01
6  1.135333e-02 1.785856e-01 6.665727e-01  1.813445329 9.479907e-02
7  1.211607e-03 2.646227e-05 7.059359e-01  0.029561591 5.832258e-01
8  3.692056e-03 9.145414e-01 1.571837e+00  1.924327407 3.622972e+00
9  1.318402e-04 1.240788e+00 3.353909e-01  0.435359073 6.557576e+00
10 2.341337e-01 1.730956e-02 4.451786e-01  1.340077017 1.855010e-01
11 6.747681e+00 2.545691e-01 5.971806e-02  4.439901593 1.078042e-01
12 4.218308e+00 1.244395e-02 2.940949e-01  0.402667475 2.843441e+00
13 3.557858e+00 3.842268e-01 1.719904e+00  2.515504964 1.306948e-02
14 9.863728e-01 6.073490e-02 6.600179e-01  1.123706226 5.859459e-01
15 9.508872e-01 9.065244e-02 7.906823e+00  0.085965424 3.518569e-01
16 2.339432e-01 7.799879e-02 3.206934e-01  4.661777069 1.749864e+00
17 5.479181e-03 3.805518e+00 1.790543e+00  0.070345628 1.359880e+00
18 3.817239e-01 1.150757e+00 2.308850e-01  0.627468563 8.803546e+00
19 2.148816e-02 1.110796e-01 1.697990e+00  1.884103587 1.436540e-01
20 3.830525e-01 1.964194e-01 2.583332e-02  0.377352279 1.693148e+00
21 1.983852e+00 1.277098e-01 2.616017e-01  0.389886312 1.161989e+00
22 8.448674e-01 2.000489e+00 4.286578e-01  0.320601282 1.936941e+00
23 3.197724e-04 1.176661e+00 7.769719e-02  0.326518860 3.083458e+00
24 5.103230e-01 1.381228e+00 2.764031e-03  0.860218607 1.425059e-02
25 1.980990e-01 1.112848e-01 5.918246e-02  0.007865611 8.951782e-02
26 1.685583e-01 7.682424e-01 2.864490e+00  4.089646011 5.038051e-01
27 4.490737e-01 2.160886e-01 2.435978e+00  0.403635694 6.325498e-01
28 7.183173e-01 4.564787e-01 5.812072e+00  6.197599131 6.762927e-01
29 8.318363e-01 9.613851e-01 6.220811e+00  5.575666663 1.869232e-02
30 1.758198e-02 9.430215e-01 2.463739e-02  3.223862659 2.017971e-02
31 9.667589e+00 6.745944e-01 2.937522e+00  1.666470444 3.097121e+00
32 2.443970e-01 5.930481e-01 4.432660e-03  1.108067555 7.229572e-01
33 1.461319e-02 6.547524e-03 3.670435e-01  0.448994977 6.951506e-03
34 6.981874e-01 2.816912e+00 1.043985e+00  0.394618687 3.440895e+00
35 6.854258e-01 5.874244e-01 1.766505e+00 11.932739501 3.017596e+00
36 2.904857e+00 1.996281e-01 1.484629e+00  2.617812268 7.967742e-01
37 5.857415e+00 2.285423e-01 9.087071e-01  0.027082187 7.538556e-02
38 8.156960e+00 9.138008e-02 4.867730e-01  3.994211342 1.101888e+00
39 8.698479e+00 2.105496e+00 3.664373e+00  0.788193725 2.088319e+00
40 1.391527e+01 2.729065e-01 5.315258e+00  4.158559066 5.711292e-01
41 2.866177e-01 1.048250e+00 5.960374e-01  0.008322829 8.299862e+00
42 2.482102e+00 5.163644e+00 1.579501e+00  2.766518878 2.137032e-02
43 1.382898e-01 1.160419e+01 1.313418e-01  4.935288409 1.661423e+00
44 5.455110e-01 4.026956e-01 1.665347e+00  0.877192647 1.000428e-01
45 5.510890e-01 6.624117e-01 8.790231e-01  0.176625516 4.963241e-02
46 2.564781e-01 1.009631e+00 2.605376e-04  0.656621105 4.581040e+00
47 1.755185e+00 5.202171e+00 3.951640e+00  2.979690228 2.769708e+00
48 7.205283e-01 1.034999e+00 6.809375e-01  0.002257047 5.904604e+00
49 3.605962e-02 8.807108e-01 2.740971e+01  2.405584994 1.230641e+00
50 2.090607e-02 1.152530e+00 4.195451e-01  0.188390625 1.488769e+00
51 1.005771e+00 5.781074e+00 5.077352e-01  0.037861881 6.302055e-05
52 2.740450e+00 1.795031e+01 3.379265e-01  4.885199483 8.716242e-02
53 2.654885e-02 8.997105e+00 1.998788e-01  0.136529581 1.588766e+00
54 5.940929e-01 5.935525e+00 1.285126e-01  1.172064266 1.122635e+00
55 5.182938e-03 4.820892e+00 1.808704e+00  0.416782976 4.997015e-01
```

16) Appliquer la CAH au tableau des contributions des individus aux axes du sous espace, interpréter les résultats

La même démarche de celle des variables, rappelant que le faite d'applique le CAH directement sur le vecteur «cont », ça ne va pas marcher, car nous avons le nom d'une variable qui est vide, pour cela nous allons exécuter la commande write.table ( ) ça va nous créer un fichier Excel qui contient les contributions des individus, puis nous allons remplir la case vide sur Excel manuellement ensuite lire le fichier.

```
#Contribution des individus au sous espace
cont <- res$ind$contrib[,1:4]
write.table(cont,"Cont_Ind.csv",sep=";",col.names=TRUE,dec=',', row.names=TRUE)
Don<-read.csv2("Cont_Ind1.csv",row.names=1)
View(Don)
```

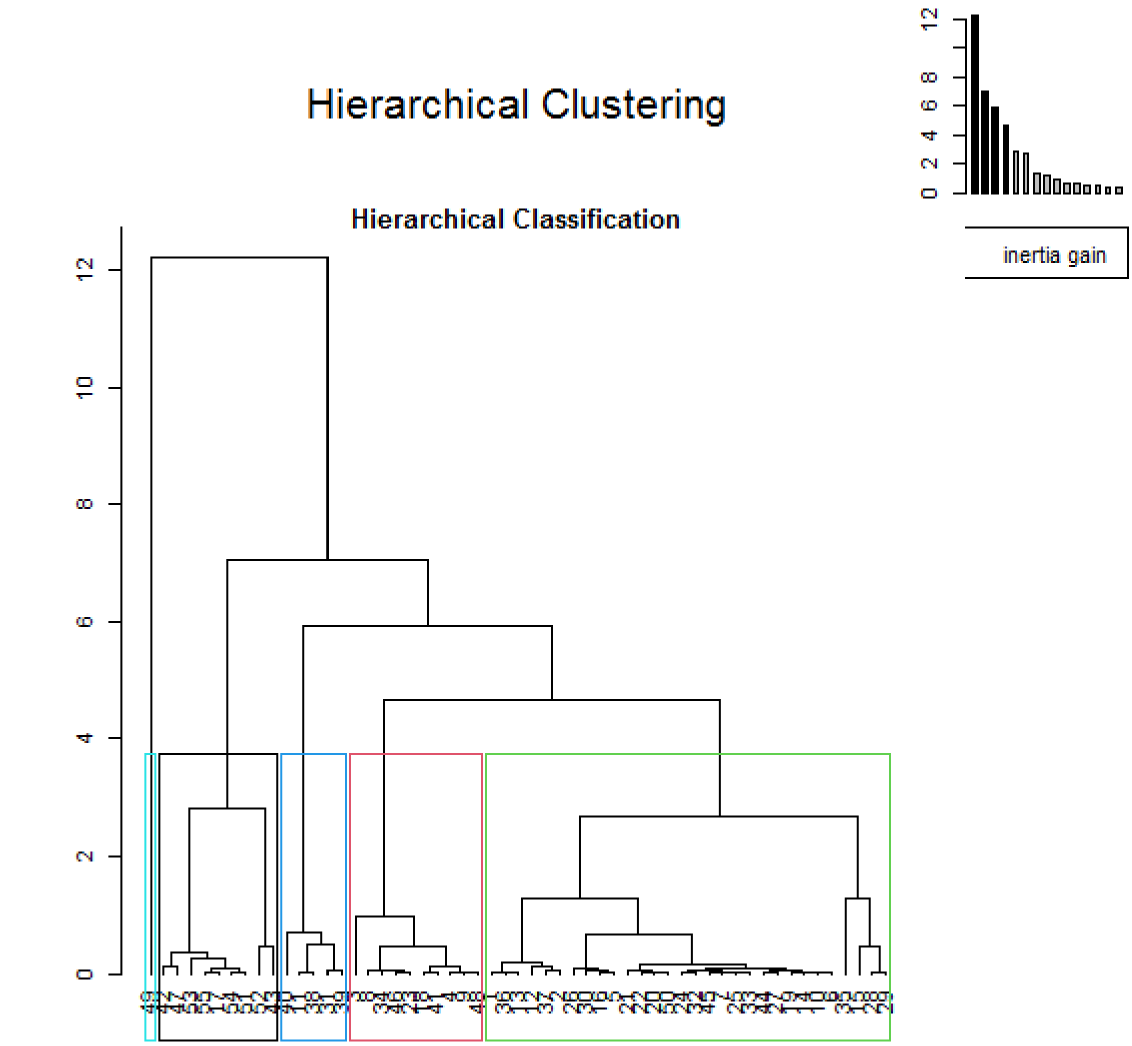
	Dim.1	Dim.2	Dim.3	Dim.4	Dim.5
1	4.706585704	0.173680407	1.437740171	2.335642614	0.018314722
2	4.363391667	1.744930732	1.582769617	0.178271953	0.004497511
3	4.621098788	1.034379892	0.121781033	0.307275644	11.995780800
4	0.370449524	1.156035318	1.663730495	0.984112645	5.865490105
5	0.470321407	0.030121759	0.299344163	4.285952870	0.957520362
6	0.011353327	0.178585558	0.666572713	1.813445329	0.094799065
7	0.001211607	0.000026500	0.705935898	0.029561591	0.583225802
8	0.003692056	0.914541402	1.571836991	1.924327407	3.622971637
9	0.000131840	1.240788335	0.335390917	0.435359073	6.557576226
10	0.234133718	0.017309556	0.445178649	1.340077017	0.185501034

Showing 1 to 10 of 55 entries, 5 total columns

Ensuite, exécuter la commande HCPC( ) : la mention "nb.clust=-1" pour avoir une coupe systématique.

```
Cat_Ind<-HCPC(Don, nb.clust =-1)
```

La CAH nous a donné 5 classes d'individus, nous avons 55 individus qui ont été catégorisé en 5 classes selon leur contribution aux 5 dimensions de notre sous espace.



Pour l’interprétation de ces 6 classes d’individus, on utilise la commande Cat_Ind$desc.var :

```
> Cat_Ind<-HCPC(Don, nb.clust =-1)
> Cat_Ind$desc.var

Link between the cluster variable and the quantitative variables
================================================================
           Eta2      P-value
Dim.3 0.8218825 3.990724e-18
Dim.5 0.7348683 7.476659e-14
Dim.1 0.7269936 1.538347e-13
Dim.2 0.6849496 5.220950e-12

Description of each cluster by quantitative variables
=====================================================
$`1`
        v.test Mean in category Overall mean sd in category Overall sd      p.value
Dim.5 6.296021           7.4297     1.818182       2.303471   2.501148 3.053821e-10

$`2`
        v.test Mean in category Overall mean sd in category Overall sd      p.value
Dim.2 6.072544         8.181863     1.818182       4.299616   3.177099 1.258995e-09

$`3`
         v.test Mean in category Overall mean sd in category Overall sd     p.value
Dim.1 -2.689540        0.9556332     1.818182      1.3518415   2.886345 0.007155065
Dim.5 -3.056525        0.9687571     1.818182      1.0991692   2.501148 0.002239187
Dim.2 -3.230902        0.6776380     1.818182      0.8540889   3.177099 0.001234002

$`4`
        v.test Mean in category Overall mean sd in category Overall sd      p.value
Dim.1 6.256203         8.840566     1.818182       2.588547   2.886345 3.944621e-10

$`5`
        v.test Mean in category Overall mean sd in category Overall sd     p.value
Dim.3 6.611473         27.40971     1.818182              0   3.870775 3.80515e-11
```

La **classe 1** est composée d'individus tels que 3 et 4. Ce groupe a un v. test de 6.296021 (c'est un v. test assez grand pour la dim.5), donc ces individus contribuent à la dimension 5.

La **classe 2** est composée d'individus tels que 43,47 et 53. Ce groupe a un v. test de 6.072544 (c'est un v. test assez grand pour la dim.2), donc ces individus contribuent à la dimension 2.

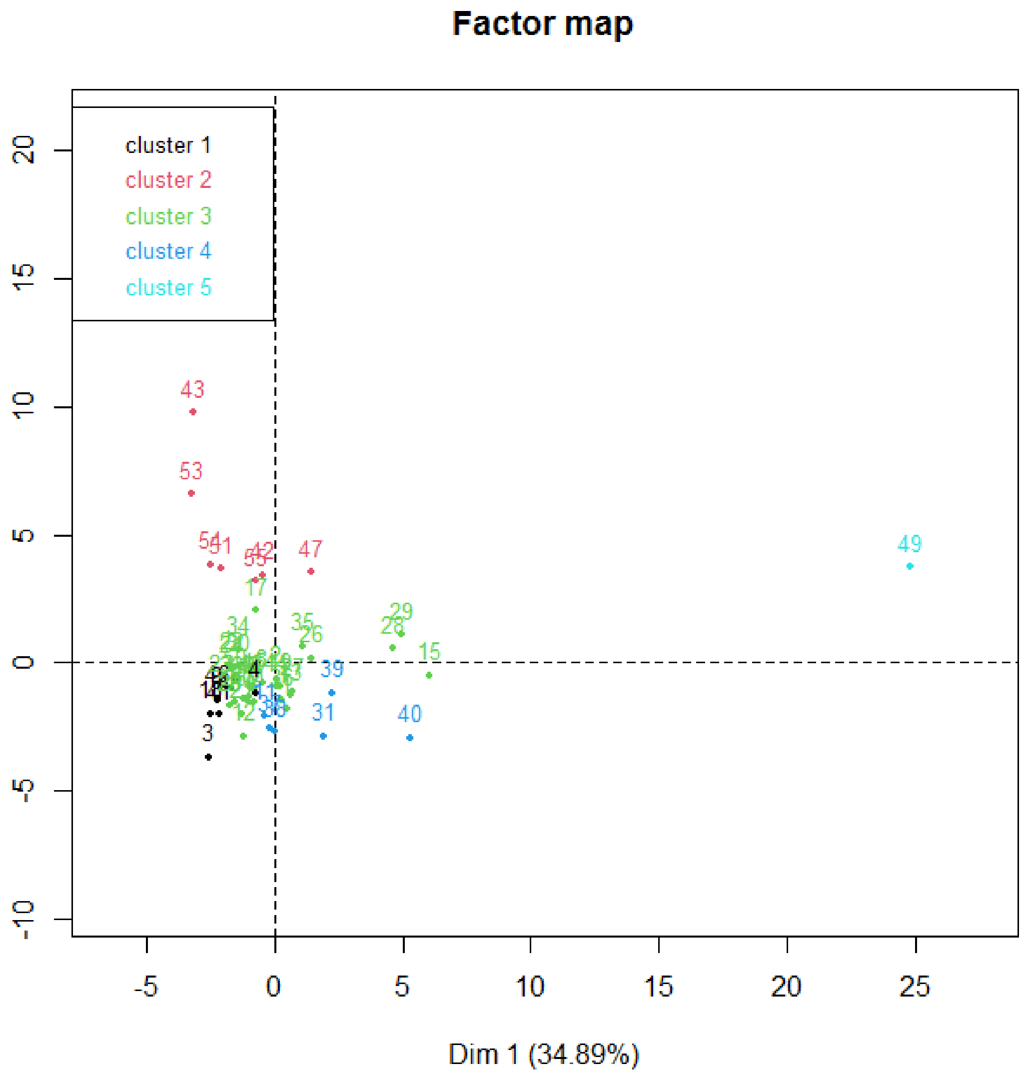
La **classe 3** est composée d'individus tels que 15, 28, 29 et 35. Ce groupe a un v. test de -2.689540 (c'est un v. test assez grand pour la dim.1 et négatif), donc ces individus ne contribuent pas à la dimension 1. Egalement des v.test de -3.056525 et -3.230902 (c'est des v.tests assez granf pour la dim.5 et dim.2), donc ces individus ne contribuent pas à la dimension 5 ni à la dimension 2.

Ce sont des individus qui ne contribuent carrément pas aux axes, donc on peut reprendre l'ACP et les mettent comme individus supplémentaires.

La **classe 4** est composée d'individus tels que31, 39 et 40. Ce groupe a un v. test de 6.256203 (c'est un v. test assez grand pour la dim.1), donc ces individus contribuent à la dimension 1.

La **classe 5** est composée d'individus tels que 49. Ce groupe a un v. test de 6.611473 (c'est un v. test assez grand pour la dim.3), donc ces individus contribuent à la dimension 3.

17) Tracer le nuage des individus projeté sur les 2 premiers axes



18) Conclusion

L’objectif de **l**’ACP était de condenser l'information contenu dans le tableau de données par une analyse des corrélations linéaires entre les variables et une visualisation graphique des distances entre les individus. Ça nous a permis de dégager les liaisons entre variables et les ressemblances entre individus.