

2019/2020



Faculté des sciences et techniques MARRAKECH

Master SDAD

Enseignant: Pr. A. Ouabarab

[PROJET ADD: TEXT MINING]

Réalisé par:

ELMRHARI Yassine

Objectif

L'objectif de ce projet est d'appliquer la notion du « text mining » pour analyser des grandes quantités d'informations textuelles exprimées en langage naturel de sorte à coder le texte, à le quantifier et à fournir des informations thématiques, des structures, des résumés qui ensuite, seront analyser et représenté graphiquement pour faire les études souhaitées.

Introduction

En gros, ce projet consiste à faire du « text mining » sur les différents chapitres de cours de l'analyse des données suivants: Régression multiple, Analyse en composantes principales, Analyse factorielle de correspondances, Analyse de correspondances multiples, Analyse discriminante, Classification. Pour générer un tableau lexical rempli par la totalité des mots ayant été prétraité de ces chapitres et ensuite faire appliquer toutes les notions vues dans le cours et interprétation utiles a la description des données.



Bibliothèques nécessaires

```
#Installation
install.packages("pdftools")
install.packages("tm")
install.packages("SnowballC")
install.packages("reshape")

#Importation
library(pdftools)
library("tm")
library("SnowballC")
library(dplyr)
library(reshape2)

library(FactoMineR)
library(ggplot2)
library(factoextra)
```

Pdftools :

Utilitaire d'extraction du texte, des polices, des pièces jointes et métadonnées d'un fichier PDF. Prend également en charge le rendu de haute qualité des documents PDF en PNG, JPEG, TIFF ou dans des vecteurs bitmap bruts pour un traitement ultérieur dans R.

Tm (Text Mining) :

Un framework pour les applications d'exploration de texte dans R.

SnowballC :

Utilitaire pour faire le texte stemming (réduire les mots à leurs racines). En d'autres termes, ce processus supprime les suffixes des mots pour les rendre simples et pour obtenir l'origine commune.

Dplyr :

Un outil rapide et cohérent pour travailler avec des data frames comme des objets, à la fois en mémoire et hors de la mémoire.

Reshape2 :

Un package R qui facilite la transformation des données entre les formats larges et longs.

FactoMineR :

Un package R dédié à l'analyse exploratoire multidimensionnelle de données.

Ggplot2 :

Une librairie R de visualisation de données.

factoextra :

Un package R facilitant l'extraction et la visualisation de la sortie d'analyses exploratoires de données multi variées

Phase 1 : prétraitement de données

1. Extraction de données :

On utilise la fonction `pdf_text()` du package «`pdftools`» pour charger les fichiers pdf des cours souhaitées :

```
RM <- pdf_text("C:/Users/PC Gamer/Desktop/Cours/Regression multiple.pdf")
ACP <- pdf_text("C:/Users/PC Gamer/Desktop/Cours/Analyse en Composantes Principales.pdf")
AFC1 <- pdf_text("C:/Users/PC Gamer/Desktop/Cours/AFC1.pdf")
AFC2 <- pdf_text("C:/Users/PC Gamer/Desktop/Cours/AFC2.pdf")
ACM <- pdf_text("C:/Users/PC Gamer/Desktop/Cours/Analyse des correspondances multiples.pdf")
AD <- pdf_text("C:/Users/PC Gamer/Desktop/Cours/analyse discriminante.pdf")
C <- pdf_text("C:/Users/PC Gamer/Desktop/Cours/Classification.pdf")
```

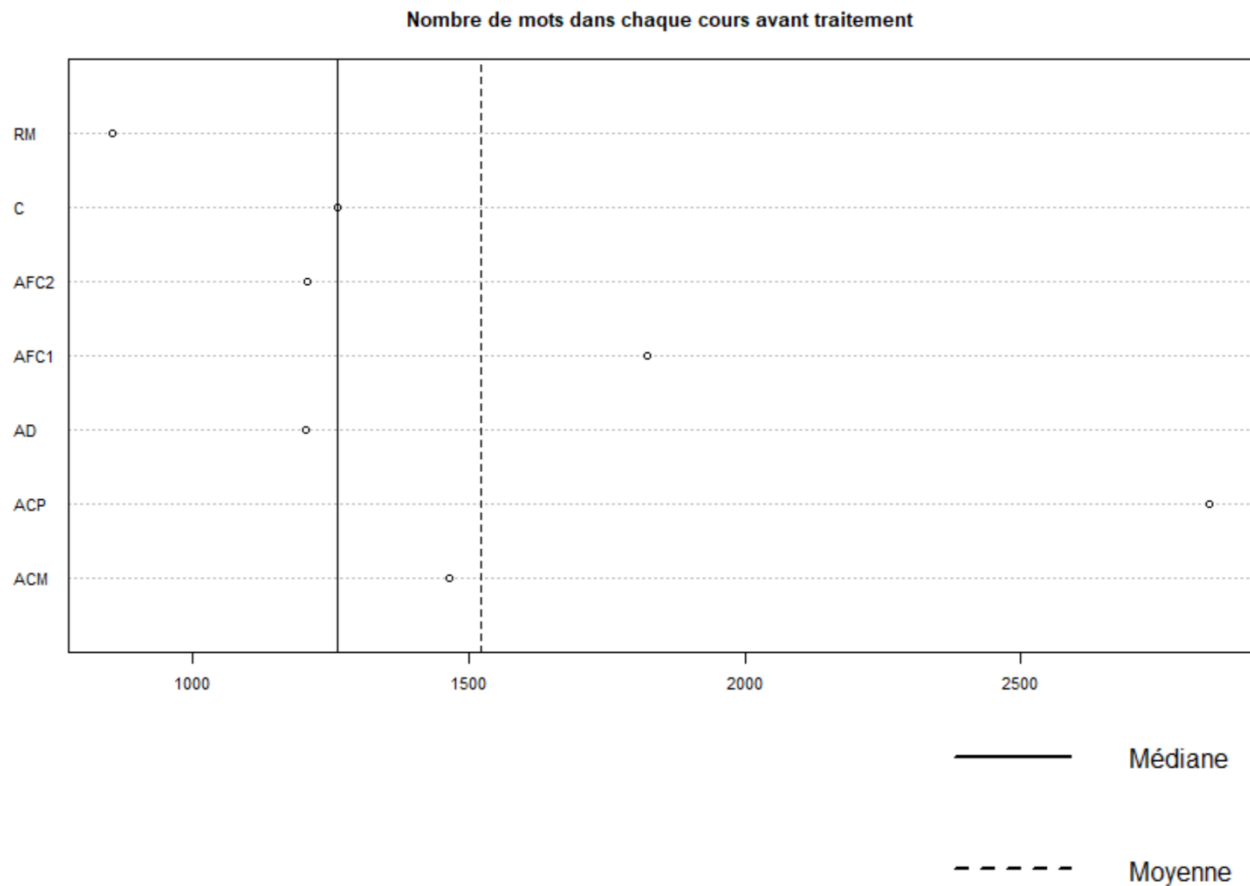
2. Chargement des données :

Le texte peut être chargé en utilisant la fonction `Corpus()` du package «`tm`». `Corpus` est une liste de documents prêts à être traités.

```
RM <- Corpus(VectorSource(RM))
ACP <- Corpus(VectorSource(ACP))
AFC1 <- Corpus(VectorSource(AFC1))
AFC2 <- Corpus(VectorSource(AFC2))
ACM <- Corpus(VectorSource(ACM))
AD <- Corpus(VectorSource(AD))
C <- Corpus(VectorSource(C))
```

3. Nombre de mots de chaque pdf avant le prétraitement :

Voici un « dotchart » du nombre des mots contenu dans chaque chapitre avant le prétraitement :



4. Etapes du text minning ou traitement de données:

a. Suppression des caractères spéciaux et quelques nombres :

On utilise la fonction `tm_map()` du package «tm» avec l'argument «specialchars» qui référence a la fonction suivante :

```
specialchars <- content_transformer(function(x) gsub("[^[:alnum:]]+", " ", x))
```

Pour remplacer les caractères spéciaux ainsi que quelques nombres des corpus par des espaces vides.

```
RM <- tm_map(RM, specialchars)
C <- tm_map(C, specialchars)
AFC1 <- tm_map(AFC1, specialchars)
AFC2 <- tm_map(AFC2, specialchars)
AD <- tm_map(AD, specialchars)
ACP <- tm_map(ACP, specialchars)
ACM <- tm_map(ACM, specialchars)
```

b. Suppression des nombres :

On utilise la fonction `tm_map()` du package «tm» avec l'argument «`removeNumbers`» pour supprimer tout les nombres des corpus.

```
RM <- tm_map(RM, removeNumbers)
C <- tm_map(C, removeNumbers)
AFC1 <- tm_map(AFC1, removeNumbers)
AFC2 <- tm_map(AFC2, removeNumbers)
AD <- tm_map(AD, removeNumbers)
ACP <- tm_map(ACP, removeNumbers)
ACM <- tm_map(ACM, removeNumbers)
```

c. Suppression des mots outils français :

On utilise la fonction `tm_map()` du package «tm» avec l'argument «`removeWords, stopwords('french')`» pour supprimer les mots outils français des corpus.


```

RM <- tm_map(RM, removeWords, stopwords("french"))
C <- tm_map(C, removeWords, stopwords("french"))
AFC1 <- tm_map(AFC1, removeWords, stopwords("french"))
AFC2 <- tm_map(AFC2, removeWords, stopwords("french"))
AD <- tm_map(AD, removeWords, stopwords("french"))
ACP <- tm_map(ACP, removeWords, stopwords("french"))
ACM <- tm_map(ACM, removeWords, stopwords("french"))

```

d. Suppression des espaces vides :

On utilise la fonction `tm_map()` du package «tm» avec l'argument «`stripWhitespace`» pour supprimer les espaces vides générés à cause des suppressions précédentes des corpus. Cette étape est juste un plus car les espaces vides sont supprimés automatiquement dans la génération des matrices des mots.

```

RM <- tm_map(RM, stripWhitespace)
C <- tm_map(C, stripWhitespace)
AFC1 <- tm_map(AFC1, stripWhitespace)
AFC2 <- tm_map(AFC2, stripWhitespace)
AD <- tm_map(AD, stripWhitespace)
ACP <- tm_map(ACP, stripWhitespace)
ACM <- tm_map(ACM, stripWhitespace)

```

e. Texte stemming :

On utilise la fonction `tm_map()` du package «tm» avec l'argument «`stemDocument`» pour réduire les mots des corpus à leurs racines.

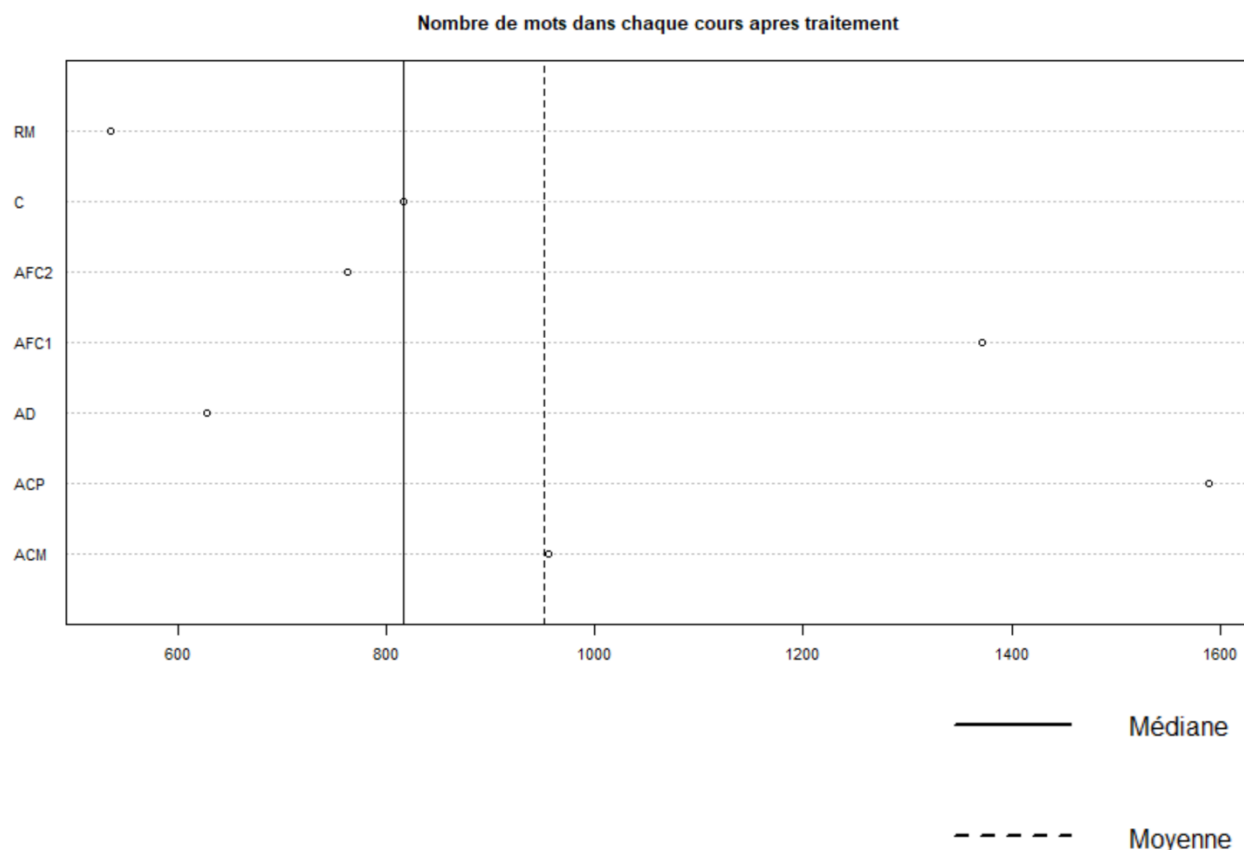
```

RM <- tm_map(RM, stemDocument)
C <- tm_map(C, stemDocument)
AFC1 <- tm_map(AFC1, stemDocument)
AFC2 <- tm_map(AFC2, stemDocument)
AD <- tm_map(AD, stemDocument)
ACP <- tm_map(ACP, stemDocument)
ACM <- tm_map(ACM, stemDocument)

```

5. Nombre de mots de chaque pdf après le prétraitement :

Voici un « dotchart » du nombre des mots contenu dans chaque chapitre après le prétraitement :



6. Création des matrices des mots:

a. Génération des matrices des mots:

Cette étape ci présente la génération des matrices des mots à partir des corpus prétraitées, on transforme le corpus en TermDocumentMatrix qui est tout simplement une table contenant la fréquence des mots :

```

dtmRM <- TermDocumentMatrix(RM)
mRM <- as.matrix(dtmRM)
vRM <- sort(rowSums(mRM),decreasing=TRUE)
dRM <- data.frame(freq=vRM)
dRM <- setNames(cbind(rownames(dRM), dRM, row.names = NULL),c("Nom", "Frequence"))

dtmC <- TermDocumentMatrix(C)
mC <- as.matrix(dtmC)
vC <- sort(rowSums(mC),decreasing=TRUE)
dC <- data.frame(freq=vC)
dC <- setNames(cbind(rownames(dC), dC, row.names = NULL),c("Nom", "Frequence"))

dtmAFC1 <- TermDocumentMatrix(AFC1)
mAFC1 <- as.matrix(dtmAFC1)
vAFC1 <- sort(rowSums(mAFC1),decreasing=TRUE)
dAFC1 <- data.frame(freq=vAFC1)
dAFC1 <- setNames(cbind(rownames(dAFC1), dAFC1, row.names = NULL),c("Nom", "Frequence"))

dtmAFC2 <- TermDocumentMatrix(AFC2)
mAFC2 <- as.matrix(dtmAFC2)
vAFC2 <- sort(rowSums(mAFC2),decreasing=TRUE)
dAFC2 <- data.frame(freq=vAFC2)
dAFC2 <- setNames(cbind(rownames(dAFC2), dAFC2, row.names = NULL),c("Nom", "Frequence"))

dtmAD <- TermDocumentMatrix(AD)
mAD <- as.matrix(dtmAD)
vAD <- sort(rowSums(mAD),decreasing=TRUE)
dAD <- data.frame(freq=vAD)
dAD <- setNames(cbind(rownames(dAD), dAD, row.names = NULL),c("Nom", "Frequence"))

dtmACP <- TermDocumentMatrix(ACP)
mACP <- as.matrix(dtmACP)
vACP <- sort(rowSums(mACP),decreasing=TRUE)
dACP <- data.frame(freq=vACP)
dACP <- setNames(cbind(rownames(dACP), dACP, row.names = NULL),c("Nom", "Frequence"))

dtmACM <- TermDocumentMatrix(ACM)
mACM <- as.matrix(dtmACM)
vACM <- sort(rowSums(mACM),decreasing=TRUE)
dACM <- data.frame(freq=vACM)
dACM <- setNames(cbind(rownames(dACM), dACM, row.names = NULL),c("Nom", "Frequence"))

```

On obtient comme résultat pour chaque chapitre :

dACM x	dACP x	dAD x
Filter	Filter	Filter
Nom	Nom	Nom
Frequence	Frequence	Frequence
1 modalité	1 variabl	1 site
2 ouaarab	2 axe	2 ouaarab
3 variabl	3 individus	3 variabl
4 individus	4 composant	4 group
5 inert	5 matric	5 discrimin
6 total	6 valeur	6 individus
7 dimens	7 propr	7 axe
8 oui	8 représent	8 propr
9 donné	9 inert	9 class
10 tableau	10 principal	10 matric
11 non	11 les	11 varianc
12 plus	12 donné	12 vecteur
13 colonn	13 acp	13 centr
14 propr	14 etude	14 poid
15 valeur	15 tableau	15 les
16 acm	16 varianc	16 gravité
17 entr	17 qualité	17 inert
18 les	18 vecteur	18 valeur
19 question	19 corrélat	19 mieux
20 deux	20 premier	20 modalité
Showing 1 to 21 of 333 entries, 2 total columns	Showing 1 to 21 of 497 entries, 2 total columns	Showing 1 to 21 of 237 entries, 2 total columns

b. Suppression des mots cités moins de 17 fois:

Comme on remarque, il y'a des mots cites moins de 17 fois présents dans la plupart des chapitres, donc on va enlever ceux-là a l'aide de la fonction simple suivante pour chaque chapitre :

```
dRM <- dRM[!(dRM$Frequence<=17),]  
dC <- dC[!(dRM$Frequence<=17),]  
dAFC1 <- dAFC1[!(dRM$Frequence<=17),]  
dAFC2 <- dAFC2[!(dRM$Frequence<=17),]  
dAD <- dAD[!(dRM$Frequence<=17),]  
dACP <- dACP[!(dRM$Frequence<=17),]  
dACM <- dACM[!(dRM$Frequence<=17),]
```

On obtient comme résultat pour chaque chapitre :

dACP		
	Nom	Frequence
1	variabl	65
2	axe	48
3	individus	45
4	composant	39
5	matric	36
6	valeur	33
7	propr	32
8	représent	26
9	inerti	26
10	principal	25
11	les	21
12	donné	20
13	acp	20
14	etude	18

dACM		
	Nom	Frequence
1	modalité	56
2	ouaarab	39
3	variabl	36
4	individus	28
5	inerti	24

dAD		
	Nom	Frequence
1	site	33
2	ouaarab	31
3	variabl	22
4	group	21
5	discrimin	20

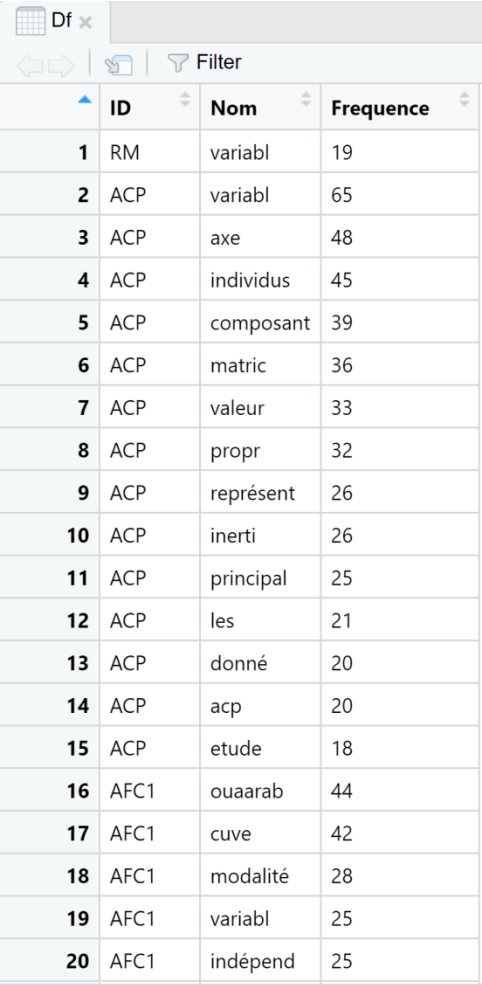
7. Création du tableau lexical avec les données prétraitées:

a. Concaténation des matrices des mots :

On concatène les matrices des mots suivant un ID pour chacune en utilisant la fonction `bind_rows()` dans un premier tableau :

```
Df <- bind_rows(list(RM=dRM, ACP=dACP, AFC1=dAFC1, AFC2=dAFC2, ACM=dACM, AD=dAD, C=dC), .id="ID")
```

Pour obtenir le resultat suivant :



ID	Nom	Frequence
1	RM	variabl
2	ACP	variabl
3	ACP	axe
4	ACP	individus
5	ACP	composant
6	ACP	matric
7	ACP	valeur
8	ACP	propr
9	ACP	représent
10	ACP	inerti
11	ACP	principal
12	ACP	les
13	ACP	donné
14	ACP	acp
15	ACP	etude
16	AFC1	ouaarab
17	AFC1	cuve
18	AFC1	modalité
19	AFC1	variabl
20	AFC1	indépend

Showing 1 to 21 of 43 entries, 3 total columns

b. Création du tableau lexical :

On commence la construction du tableau lexical à partir de la matrice des mots « Df » générée dans l'étape précédente.

Les principales étapes de constructions seront :

- ✓ Faire un reshape de la matrice des mots et groupage à travers « ID » tout en respectant le tri alphabétique.
- ✓ Remplacer les valeurs NULL par 0.
- ✓ Création d'une colonne Total des sommes des lignes.
- ✓ Faire une condition des valeurs ≤ 150 pour la colonne Total.
- ✓ Création d'une ligne Total des sommes des colonnes.

```
Resultat = dcast(Df, Nom~ID, value=Frequence) #Reshape avec tri alphabetique

Resultat[is.na(Resultat)] <- 0 #Remplacer les valeurs NULL par 0

Resultat <- cbind(Resultat, Total = rowSums(Resultat[-1])) #Creation de la colonne Total

Resultat <- Resultat[!(Resultat$Total>=150),] #La condition sur la colonne Total

Resultat <- rbind(Resultat, data.frame(Nom = "Total", t(colSums(Resultat[, -1])))) #Creation de la ligne Total

row.names(Resultat) = Resultat$Nom # Changement du nom des lignes 1
Resultat[1] = NULL # Changement du nom des lignes 2
```

Enfin et comme résultat, on obtient le tableau suivant :

	ACM	ACP	AD	AFC1	AFC2	C	RM	Total
acp	0	20	0	0	0	0	0	20
amer	0	0	0	0	32	0	0	32
axe	0	48	0	0	40	0	0	88
class	0	0	0	0	0	42	0	42
composant	0	39	0	0	0	0	0	39
cuve	0	0	0	42	0	0	0	42
dim	0	0	0	0	18	0	0	18
discrimin	0	0	20	0	0	0	0	20
distanc	0	0	0	0	0	18	0	18
donné	0	20	0	0	0	0	0	20
entr	0	0	0	20	0	0	0	20
etude	0	18	0	0	0	0	0	18
group	0	0	21	0	0	0	0	21
indépend	0	0	0	25	0	0	0	25
individus	28	45	0	0	0	20	0	93
inerti	24	26	0	0	0	0	0	50
les	0	21	0	0	0	0	0	21
matric	0	36	0	0	0	0	0	36
modalité	56	0	0	28	0	0	0	84
perçu	0	0	0	0	46	0	0	46

principal	0	25	0	0	0	0	0	25
profil	0	0	0	21	0	0	0	21
propr	0	32	0	0	0	0	0	32
représent	0	26	0	0	0	0	0	26
site	0	0	33	0	0	0	0	33
sucré	0	0	0	0	28	0	0	28
system	0	0	0	19	0	0	0	19
valeur	0	33	0	0	0	0	0	33
Total	108	389	74	155	164	80	0	970

Phase 2 : analyse de données

1. Analyse factorielle, analyse en correspondances principales :

Pour analyser ce tableau de données, on va utiliser l'analyse en correspondances principales ou ACP et ce car les variables sont quantitatives et non pas qualitatives.

a. Application de l'ACP sur les données :

```
#Application de l'ACP sur les donnees  
res.pca<-PCA(Resultat[-29,-8], scale.unit=TRUE, graph=F)
```

Voila un petit regard sur le sommaire :

```
Eigenvalues  
                Dim.1   Dim.2   Dim.3   Dim.4   Dim.5   Dim.6  
Variance         1.530    1.344    1.145    0.977    0.769    0.235  
% of var.        25.495   22.400   19.090   16.281   12.810    3.924  
Cumulative % of var. 25.495  47.895  66.986  83.266  96.076 100.000  
                Dim.7  
Variance         0.000  
% of var.        0.000  
Cumulative % of var. 100.000  
  
Individuals (the 10 first)  
                Dist   Dim.1   ctr   cos2   Dim.2   ctr   cos2  
acp             | 0.947 | -0.522 0.636 0.304 | 0.170 0.077 0.032 |  
amer            | 2.282 | -0.693 1.121 0.092 | -0.945 2.372 0.171 |  
axe             | 3.412 | -2.450 14.015 0.516 | 0.327 0.285 0.009 |  
class           | 4.514 | -0.034 0.003 0.000 | 0.907 2.187 0.040 |  
composant       | 1.773 | -1.137 3.017 0.411 | 0.726 1.400 0.168 |  
cuve            | 3.439 | 2.741 17.546 0.635 | -0.022 0.001 0.000 |  
dim             | 1.461 | -0.335 0.262 0.053 | -0.713 1.352 0.238 |  
discrimin       | 2.498 | 0.381 0.338 0.023 | -1.948 10.082 0.608 |  
distanc         | 2.056 | 0.057 0.007 0.001 | 0.151 0.061 0.005 |  
donné           | 0.947 | -0.522 0.636 0.304 | 0.170 0.077 0.032 |  
                Dim.3   ctr   cos2  
acp             | 0.085 0.022 0.008 |  
amer            | -1.395 6.070 0.374 |  
axe             | -1.662 8.613 0.237 |
```

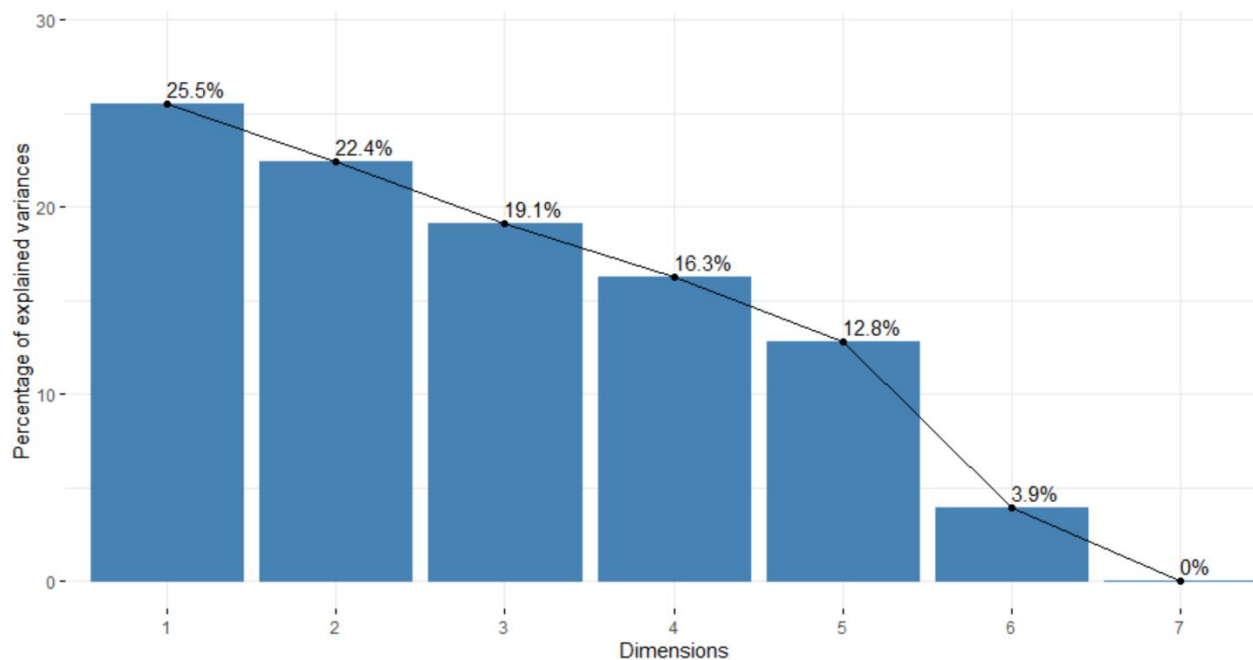

b. Calcul des valeurs propres :

```
#Calcul des valeurs propres  
eig.val<-get_eigenvalue(res.pca)
```

	eigenvalue	variance.percent	cumulative.variance.percent
Dim.1	1.5297264	25.49544	25.49544
Dim.2	1.3439994	22.39999	47.89543
Dim.3	1.1454213	19.09035	66.98578
Dim.4	0.9768337	16.28056	83.26635
Dim.5	0.7685996	12.80999	96.07634
Dim.6	0.2354196	3.92366	100.00000
Dim.7	0.0000000	0.00000	100.00000

c. Affichage du pourcentage des variances :

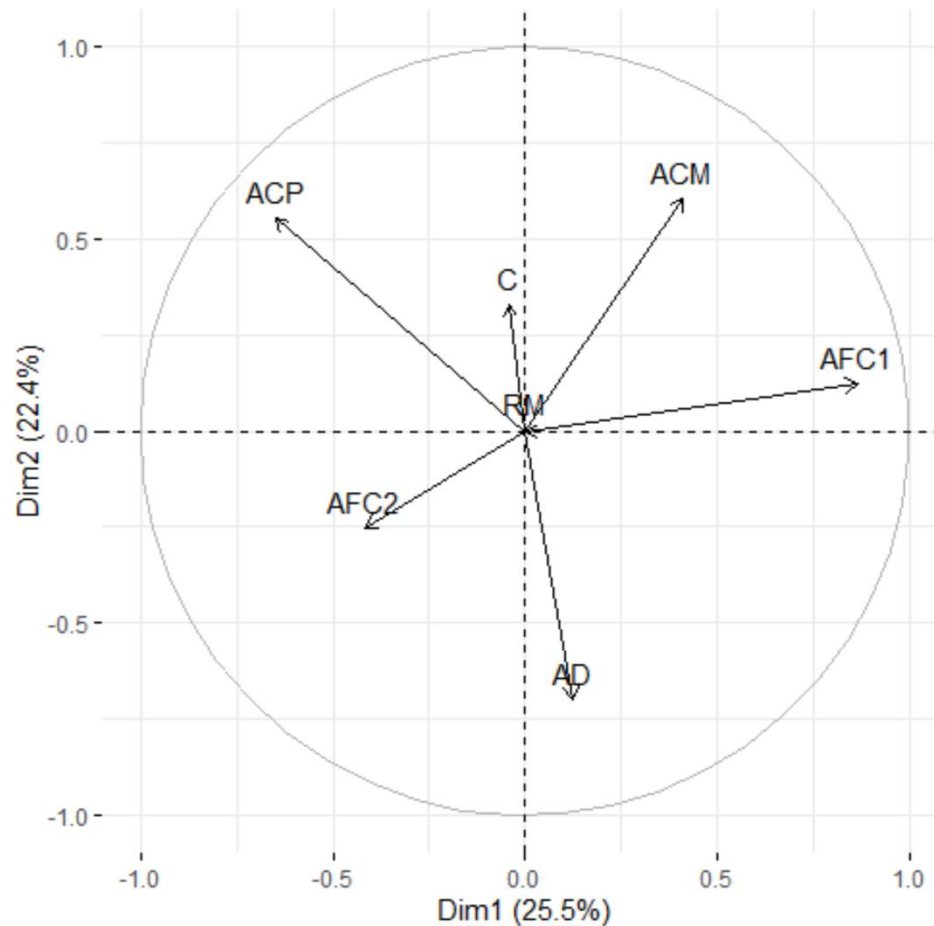
```
#Affichage du pourcentage des variances  
fviz_eig(res.pca, addlabels = TRUE, ylim = c(0, 30))
```



Comme on peut remarquer, les 4 premiers axes contiennent la majorité de l'information (83%) concernant notre tableau lexical.

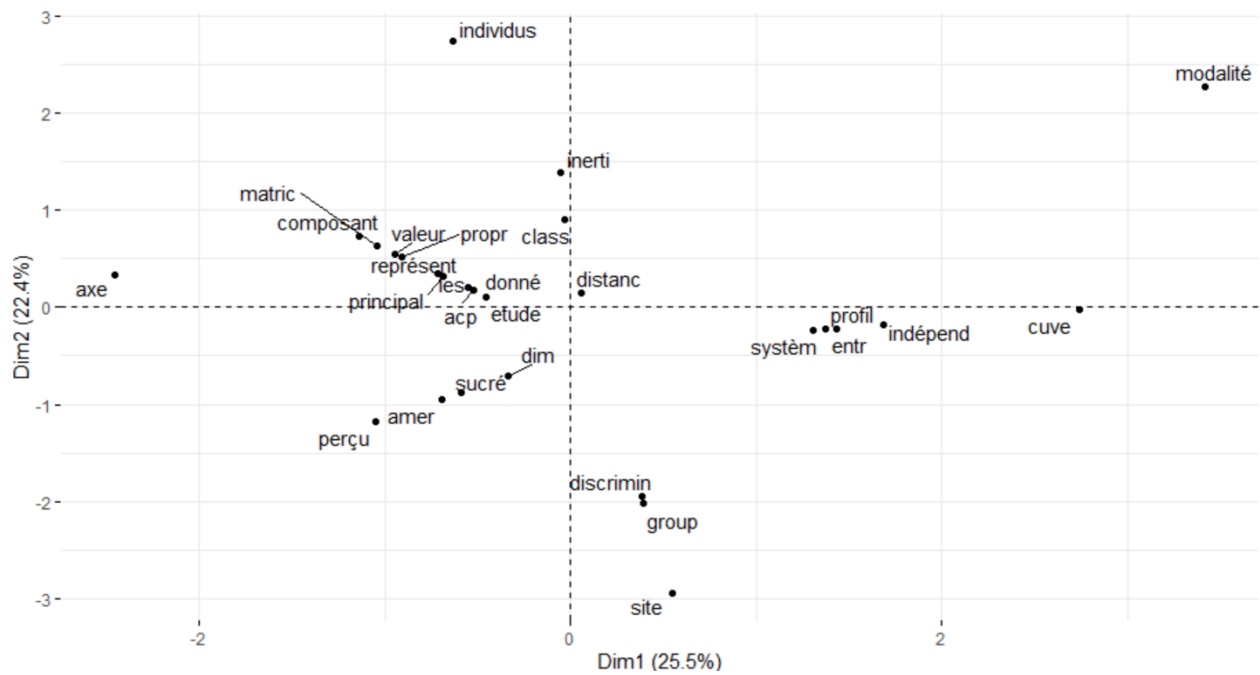
d. Affichage des variables :

```
#Affichage de la corrélation des variables  
fviz_pca_var(res.pca, col.var = "black")
```



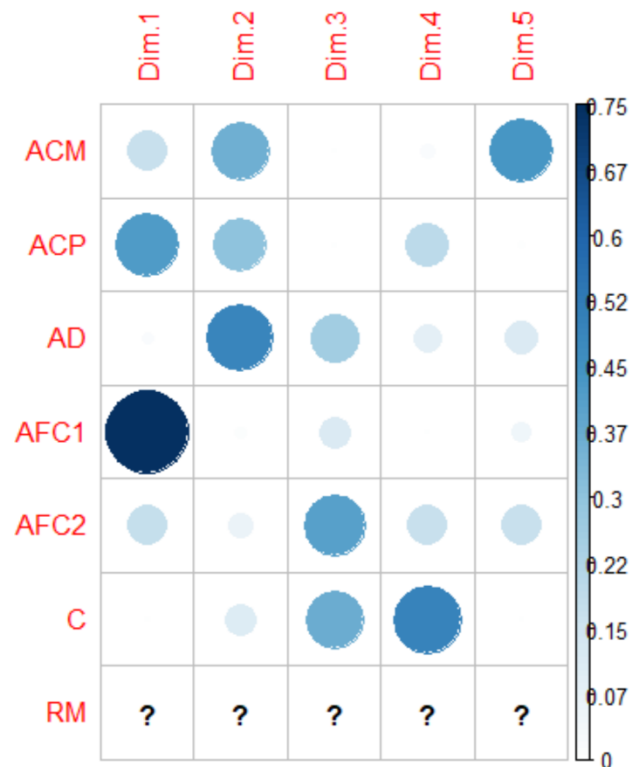
e. Affichage des individus :

```
#Affichage des individus  
fviz_pca_ind(res.pca, repel = TRUE )
```



f. Affichage de la qualité de représentation des variables :

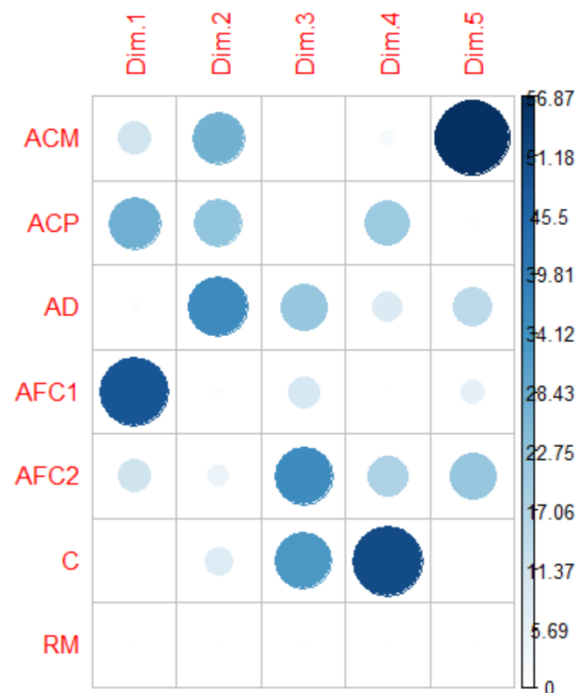
```
install.packages("corrplot")
library("corrplot")
corrplot(res.var$cos2, is.corr=FALSE)
```



On utilise le package « corrplot » pour afficher le graphique suivant qui indique la bonne représentation de la variable AFC1 sur l'axe 1, AD sur l'axe 2, AFC2 sur l'axe 3, C sur l'axe 4 et ACM sur l'axe 5. Pour RM, il n'y a pas suffisamment d'observations pour avoir une représentation.

g. Affichage de la contribution des variables:

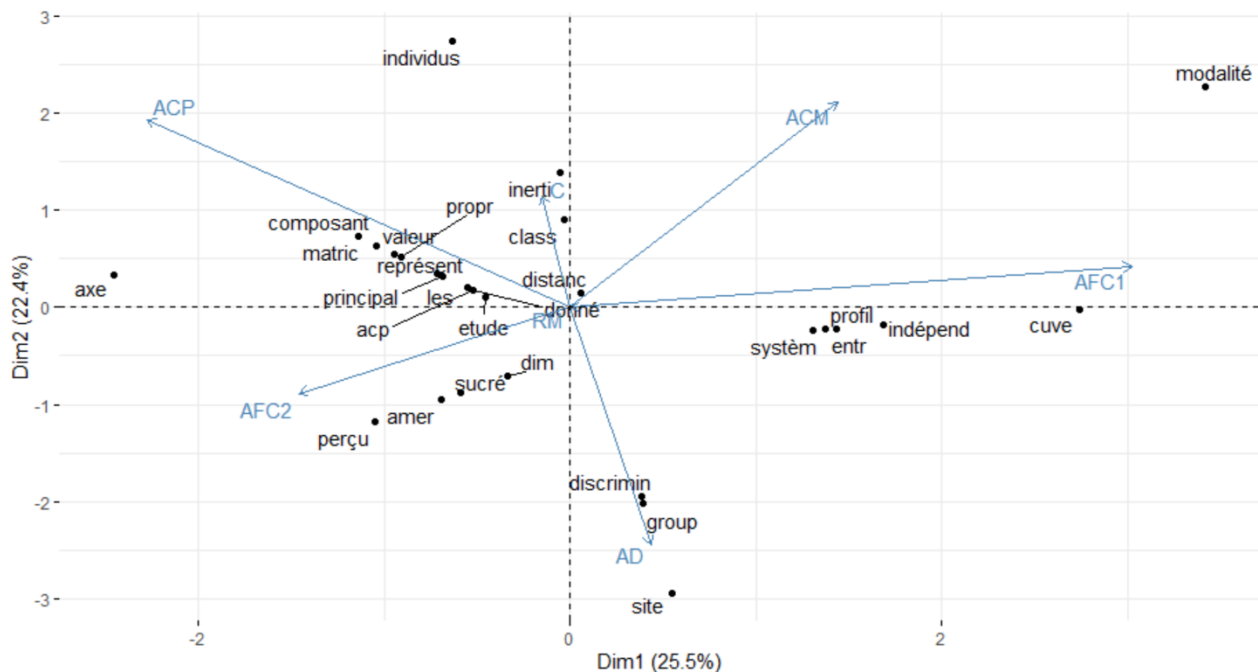
```
install.packages("corrplot")
library("corrplot")
corrplot(res.var$contrib, is.corr=FALSE)
```



Ici, le graphique suivant indique la contribution forte de la variable AFC1 sur l'axe 1, AD sur l'axe 2, AFC2 sur l'axe 3, C sur l'axe 4 et ACM sur l'axe 5. Pour RM, il n'y a pas suffisamment d'observations pour avoir une contribution.

h. Affichage des variables et des individus en même temps :

```
#Affichage des variables et des individus en meme temps  
fviz_pca_biplot(res.pca, repel=TRUE)
```



2. Classification horizontale hiérarchique :

On va appliquer une classification horizontale hiérarchique sur notre tableau lexical en calculant en premier temps la distance entre les individus ensuite faire une CAH selon une méthode complète et moyenne.

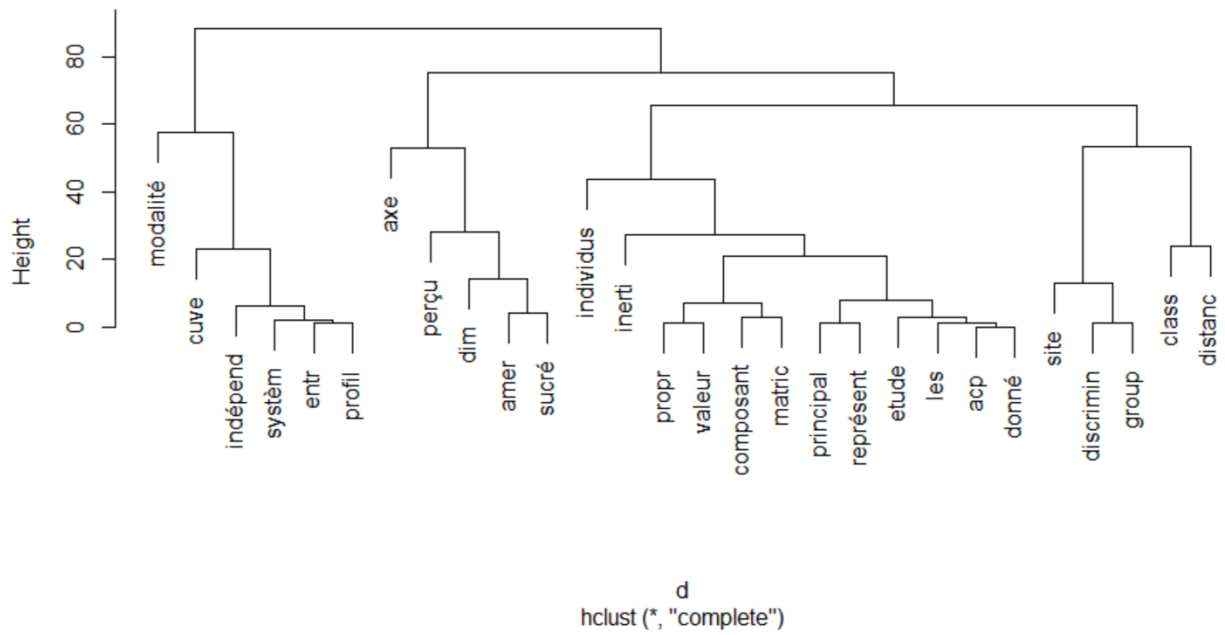
✚ Calcul de la distance :

```
d<-dist(Resultat[-29,-8])
```

	acp	amer	axe	class	composant	cuve
amer	37.73592					
axe	48.82622	48.66210				
class	46.51881	52.80152	75.28612			
composant	19.00000	50.44799	41.00000	57.31492		
cuve	46.51881	52.80152	75.28612	59.39697	57.31492	
dim	26.90725	14.00000	52.80152	45.69464	42.95346	45.69464
discrimin	28.28427	37.73592	65.60488	46.51881	43.82921	46.51881
distanç	26.90725	36.71512	65.02307	24.00000	42.95346	45.69464
donné	0.00000	37.73592	48.82622	46.51881	19.00000	46.51881
entr	28.28427	37.73592	65.60488	46.51881	43.82921	22.00000
etude	2.00000	36.71512	50.00000	45.69464	21.00000	45.69464
group	29.00000	38.27532	65.91661	46.95743	44.29447	46.95743
indépend	32.01562	40.60788	67.29785	48.87740	46.32494	17.00000
individus	42.53234	65.06151	52.84884	57.38467	34.92850	70.51950
inerti	24.73863	47.70744	51.57519	54.91812	27.29469	54.91812
les	1.00000	38.27532	48.25971	46.95743	18.00000	46.95743
matric	16.00000	48.16638	41.76123	55.31727	3.00000	55.31727
modalité	65.72671	70.31358	88.45338	75.39231	73.76313	57.72348
perçu	50.15974	14.00000	48.37355	62.28965	60.30755	62.28965
principal	5.00000	40.60788	46.14109	48.87740	14.00000	48.87740
profil	29.00000	38.27532	65.91661	46.95743	44.29447	21.00000
propr	12.00000	45.25483	43.08132	52.80152	7.00000	52.80152
représent	6.00000	41.23106	45.65085	49.39636	13.00000	49.39636
site	38.58756	45.96738	70.66116	53.41348	51.08816	53.41348
sucré	34.40930	4.00000	49.47727	50.47772	48.01042	50.47772
système	27.58622	37.21559	65.30697	46.09772	43.38202	23.00000

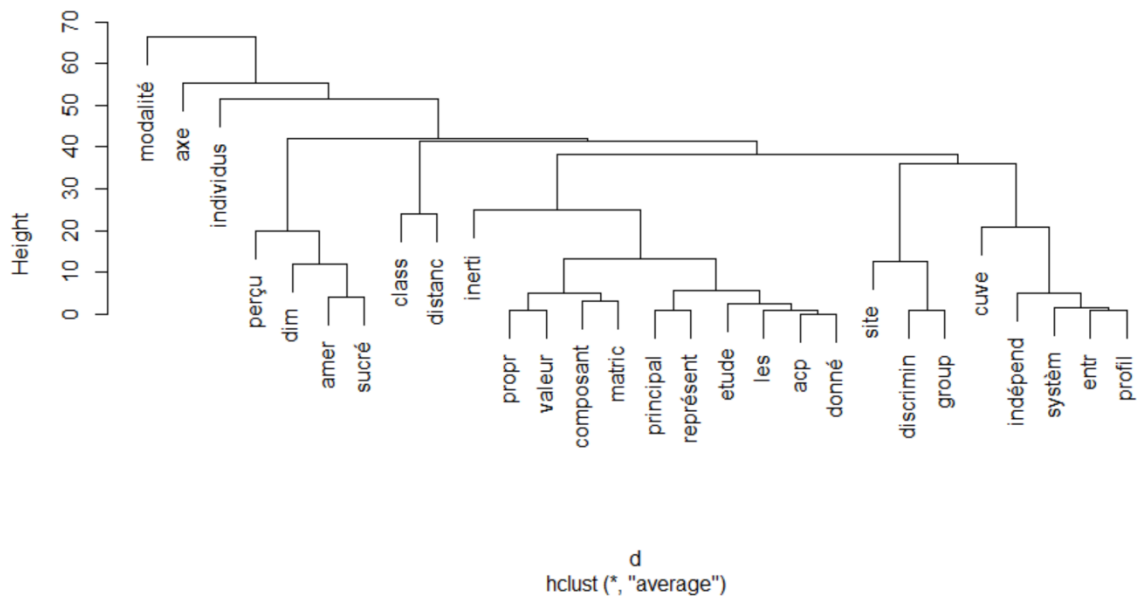
Classification horizontale hiérarchique complète :

```
#Complete
a = hclust(d, method='complete')
plot(a)
```



Classification horizontale hiérarchique moyenne :

```
#Moyenne
b = hclust(d,method='average')
plot(b)
```



Conclusion

Ce projet nous à permis de savoir les principes du text mining ainsi que ses utilités concernant l'extraction des informations voulu des textes comme le nombre d'occurrences des mots ici par exemple. Comme on a vu dans ce rapport, le text mining se trouve un outil puissant en ce qui concerne l'analyse de données car il facilite le processus de l'analyse en fournissant des données prêtent.