Sorbonne Univérsité - ISUP

TP 1 – ACP, Kmeans, Classification hiérarchique ascendante

Adnane EL KASMI - Sébastien IBRAHIM - Hmida SEBAA

21 Novembre 2021

TP 1 Données, Kmeans, CHA

November 21, 2021

L'objectif de ce TP est de vous mettre en garde contre une application trop systématique ou aveugle de l'ACP dans une étude de clustering, et d'explorer et comparer le comportement des Kmeans et de la classification hiérarchique ascendante. Outre les questions posées ci-dessous, n'hésitez pas à laisser libre cours à votre curiosité.

1 Données, ACP

[13]:
```
# Chargement des packages

library(dplyr)
library(FactoMineR)
library(tidyr)
library(rgl)
library(factoextra)
library(cluster)
library(cowplot)
library(stats)
library(ggplot2)
library(ggfortify)
```

1- Chargez les données (data(crabs, package = "MASS")), consultez l'aide associée. L'espèce et le sexe peuvent être considérés comme des classifications (possiblement à croiser pour obtenir quatre classes). Nous choisissons en tout cas de ne pas en tenir compte ici : supprimez les !

[5]:
```
# Chargement des données

data(crabs, package = "MASS")

# On choisit de conserver les variables crabs, FL, RW, CL, CW, BD (Supprimer␣
↪espèce et sexe)

crabs <- select(crabs,FL,RW,CL,CW,BD)


head(crabs)
```

FL	RW	CL	CW	BD
8.1	6.7	16.1	19.0	7.0
8.8	7.7	18.1	20.8	7.4
9.2	7.8	19.0	22.4	7.7
9.6	7.9	20.1	23.1	8.2
9.8	8.0	20.3	23.0	8.2
10.8	9.0	23.0	26.5	9.8

2- Étudiez successivement :

(a) La proportion de variance de ces données expliquée respectivement par leur première et deuxième composante principale

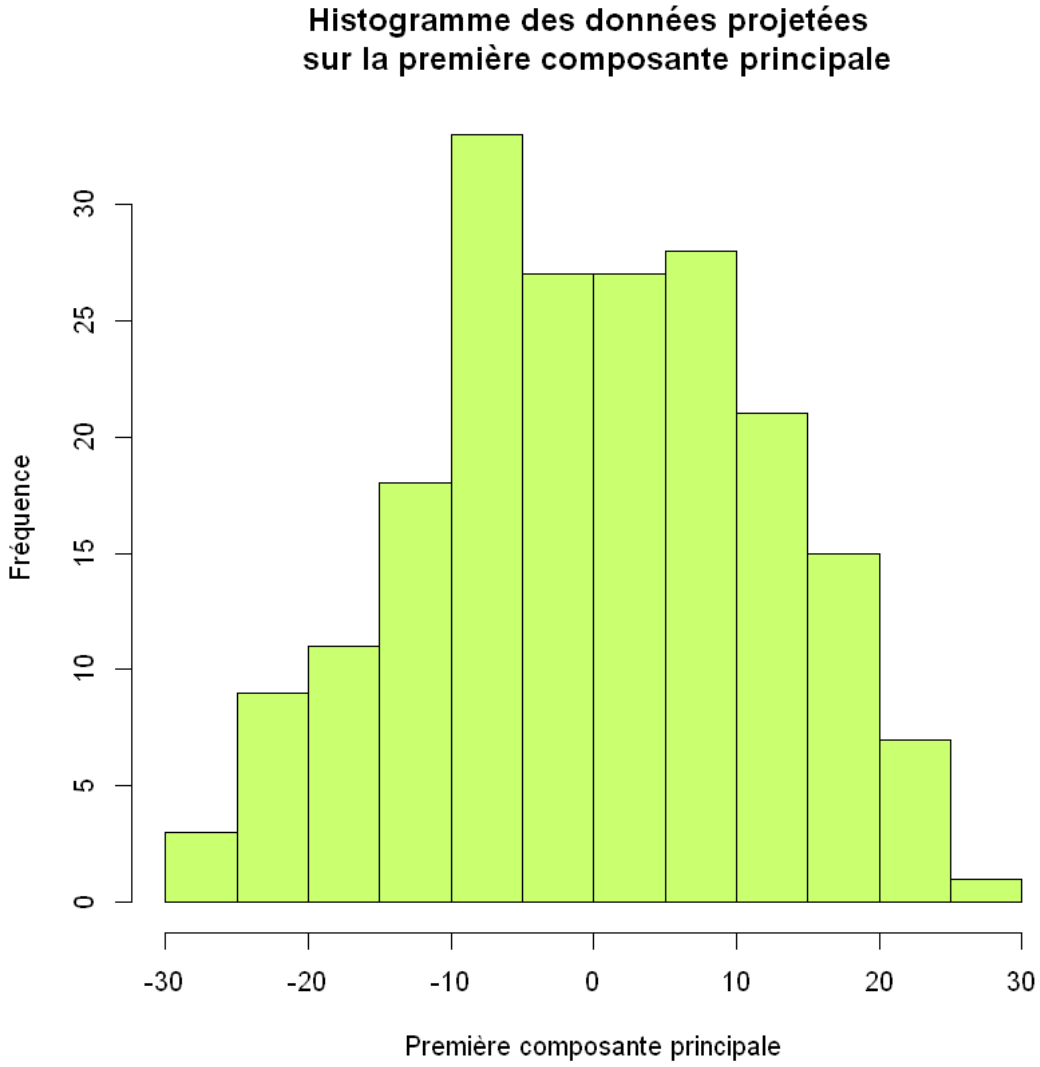
[6]:
```
pca_crabs<-PCA(crabs,scale.unit = FALSE,graph = F)
pca_crabs$eig
```

	eigenvalue	percentage of variance	cumulative percentage of variance
comp 1	140.00219017	98.24717995	98.24718
comp 2	1.29035257	0.90551084	99.15269
comp 3	0.99526778	0.69843374	99.85112
comp 4	0.13462282	0.09447218	99.94560
comp 5	0.07752466	0.05440328	100.00000

On applique une ACP sur l'ensemble des variables sélectionnées. La première composante de l'ACP explique 98.24% de la variance. Alors que la deuxième composante explique 0.90% de la variance. Ainsi, les deux composantes principales expliquent 99.14% de la variance.

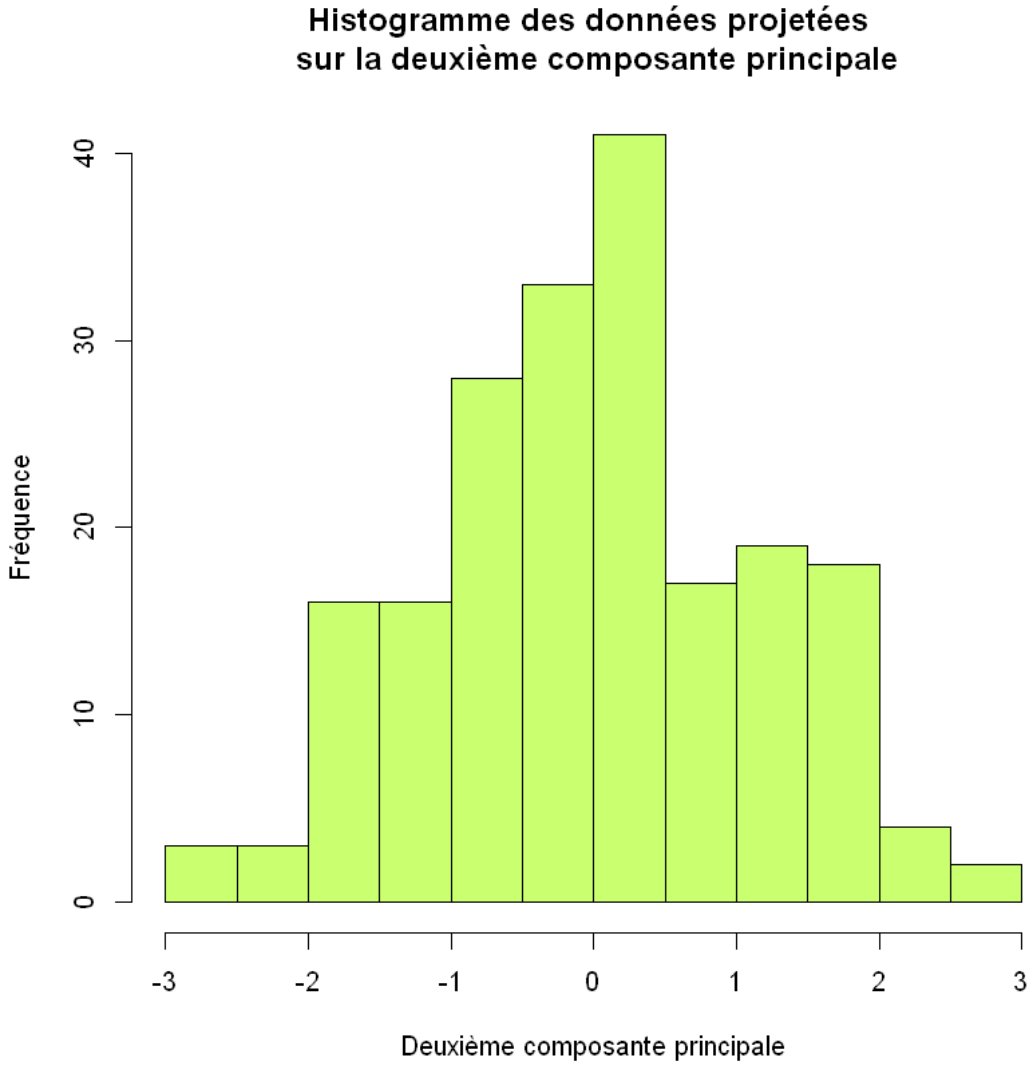
(b) La projection de ces données sur leur première composante principale

[8]:
```
hist(pca_crabs[["ind"]][["coord"]][,1],col = "darkolivegreen1",main␣
↪="Histogramme des données projetées \n sur la première composante␣
↪principale",xlab = "Première composante principale",ylab="Fréquence")
```



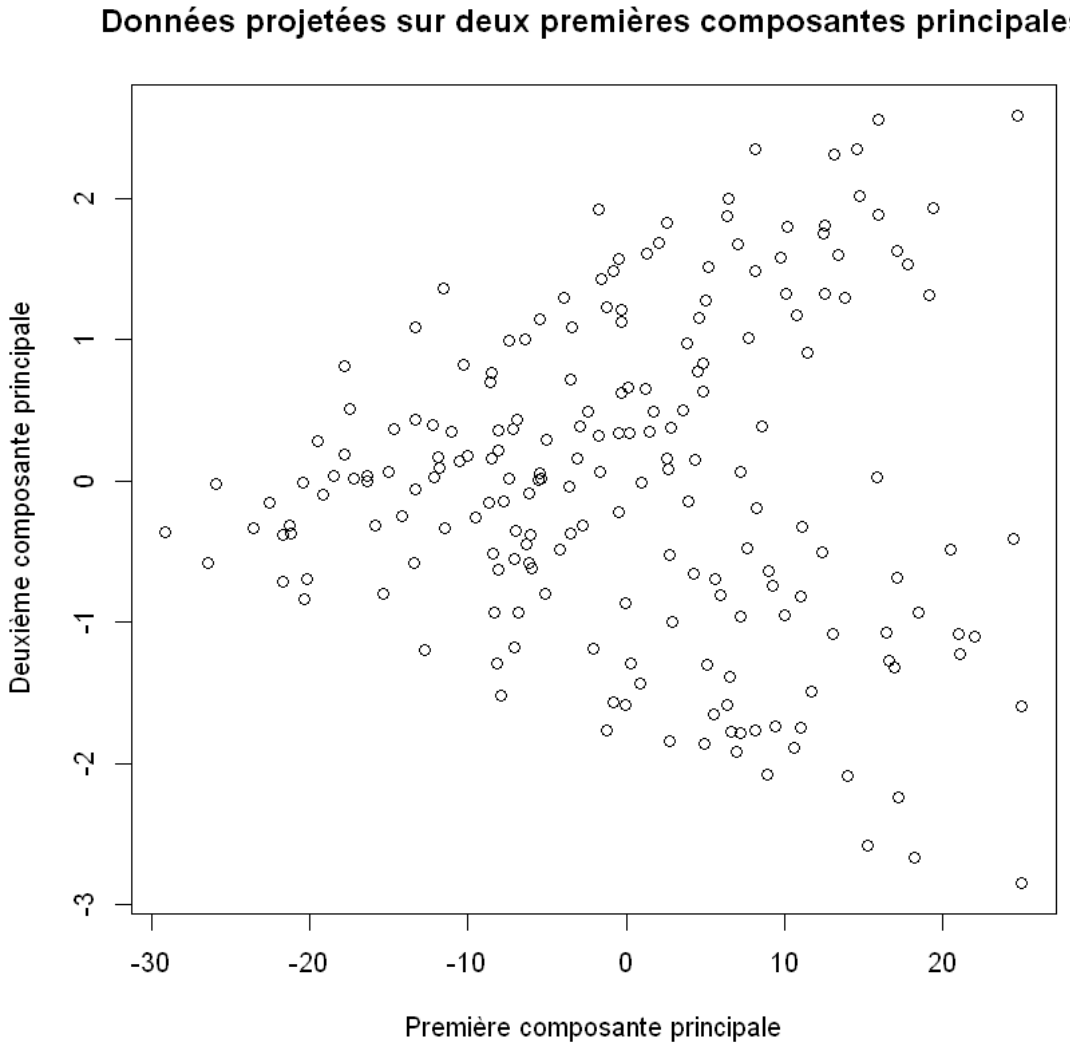
(c) La projection de ces données sur leur deuxième composante principale.

[10]:
```
hist(pca_crabs[["ind"]][["coord"]][,2],col = "darkolivegreen1",main␣
↪="Histogramme des données projetées \n sur la deuxième composante␣
↪principale",xlab = "Deuxième composante principale",ylab="Fréquence")
```



(d) La projection de ces données sur leurs deux premières composantes principales.

[11]:
```
plot(pca_crabs[["ind"]][["coord"]][,1],pca_crabs[["ind"]][["coord"]][,2],main
↪="Données projetées sur deux premières composantes principales",xlab =
↪"Première composante principale",ylab="Deuxième composante principale")
```



3- Concluez !

On peut remarquer qu'aprés la projection sur les composantes principales données par l'ACP, on ne pas vraiment conclure à l'existence de groupes au sein de notre échantillons de crabes, ce qui est problématique sachant qu'on sait qu'il existe au moins 2 espèces de crabes.

Pour conclure, dans certains cas l'ACP peut-étre un bon outil, mais ce n'est pas une formule magique, comme le démontre notre étude.

L'analyse en composante principale est une méthode efficace même s'il s'agit d'un travail de synthèse qui demande de considérer simultanement plusieurs des pistes.

2 Kmeans et clustering hiérarchique ascendant

3 Claustering

Objectif: c'est de separer les trois variétés de blé

Analyse des données Seeds

[24]:
```
# On charge les données dans un data set

coll <- read.table("seeds_dataset.txt", col.names =␣
↪c('area','perimeter','compactness','lengthOfKernel','widthOfKernel','asymmetryCoefficient',

head(coll)
```

area	perimeter	compactness	lengthOfKernel	widthOfKernel	asymmetryCoefficient	lengthOfKernelGr
15.26	14.84	0.8710	5.763	3.312	2.221	5.220
14.88	14.57	0.8811	5.554	3.333	1.018	4.956
14.29	14.09	0.9050	5.291	3.337	2.699	4.825
13.84	13.94	0.8955	5.324	3.379	2.259	4.805
16.14	14.99	0.9034	5.658	3.562	1.355	5.175
14.38	14.21	0.8951	5.386	3.312	2.462	4.956

[25]:
```
# On enlève les NA du data set

coll <- drop_na(coll)
```

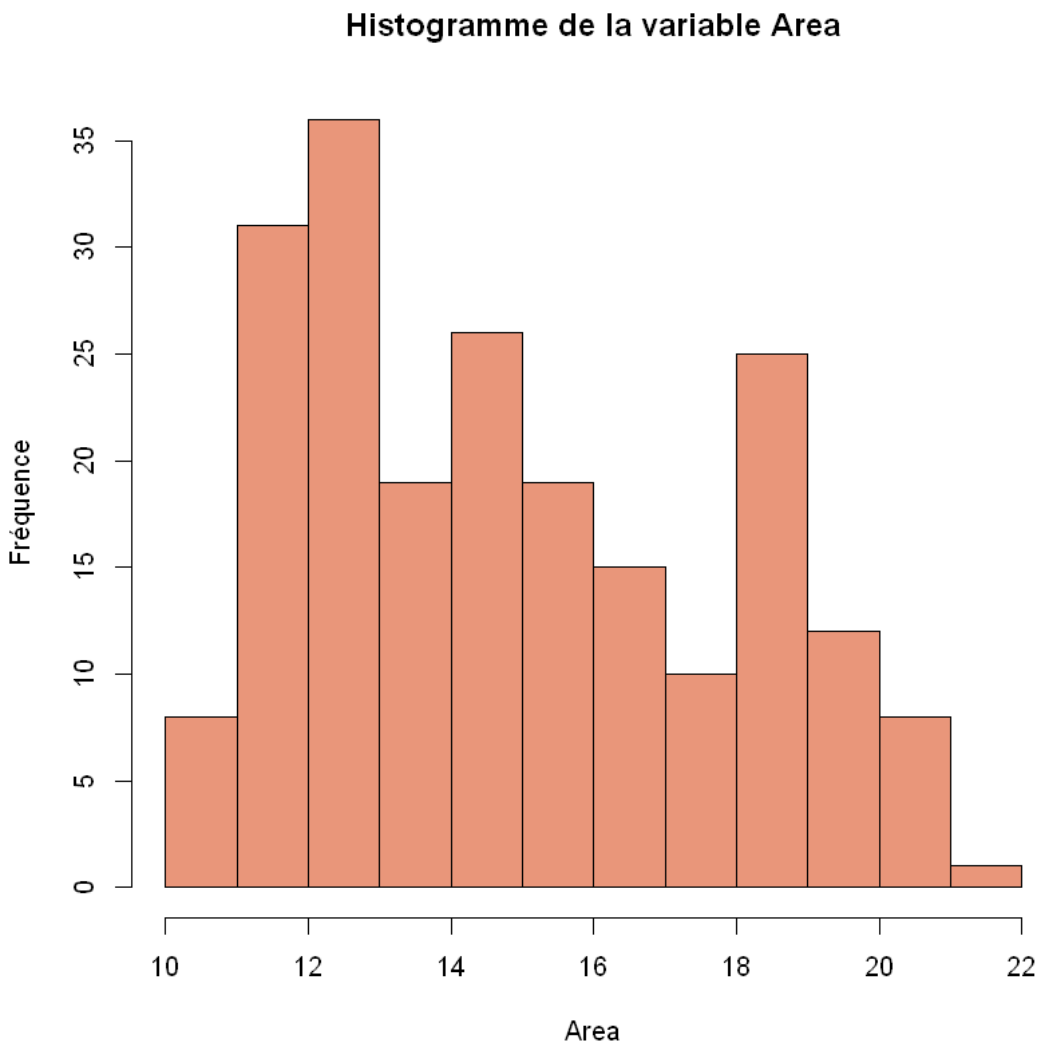
[26]:
```
# On enleve la colonne seedType qui ne nous sert pas ici

data_seeds␣
↪<-select(coll,area,perimeter,compactness,lengthOfKernel,widthOfKernel,asymmetryCoefficient,
```

On trace les histogrammes des différentes variables:

Dans le meilleur des mondes, on souhaiterait obtenir des lois normales multivariées, ce qui traduirait l'existence de différents groupes au sein de notre échantillon de graines.

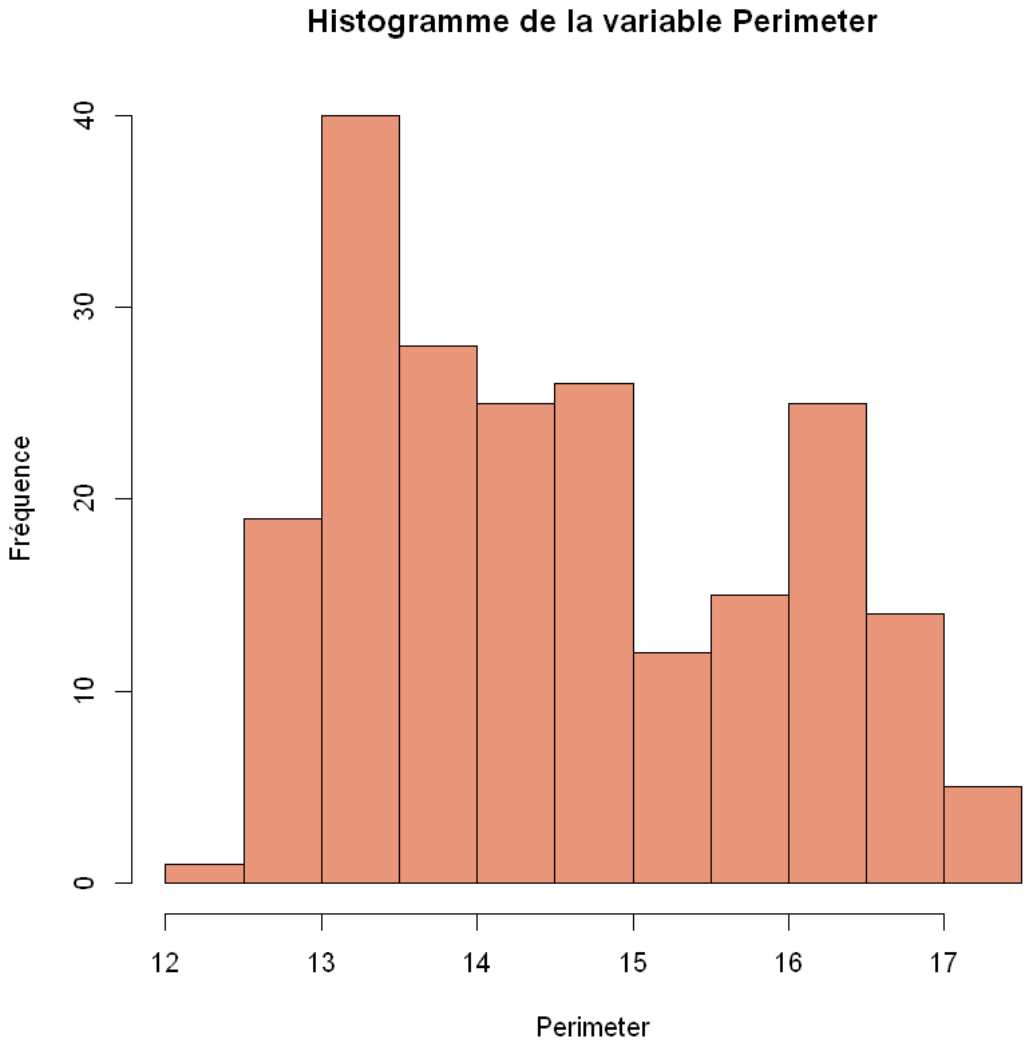
L'histogramme 1 nous laisse penser qu'il y au moins 2 graines.

[28]:
```
hist(data_seeds$area,col = "darksalmon",main ="Histogramme de la variable␣
↪Area",xlab = "Area",ylab="Fréquence")
```



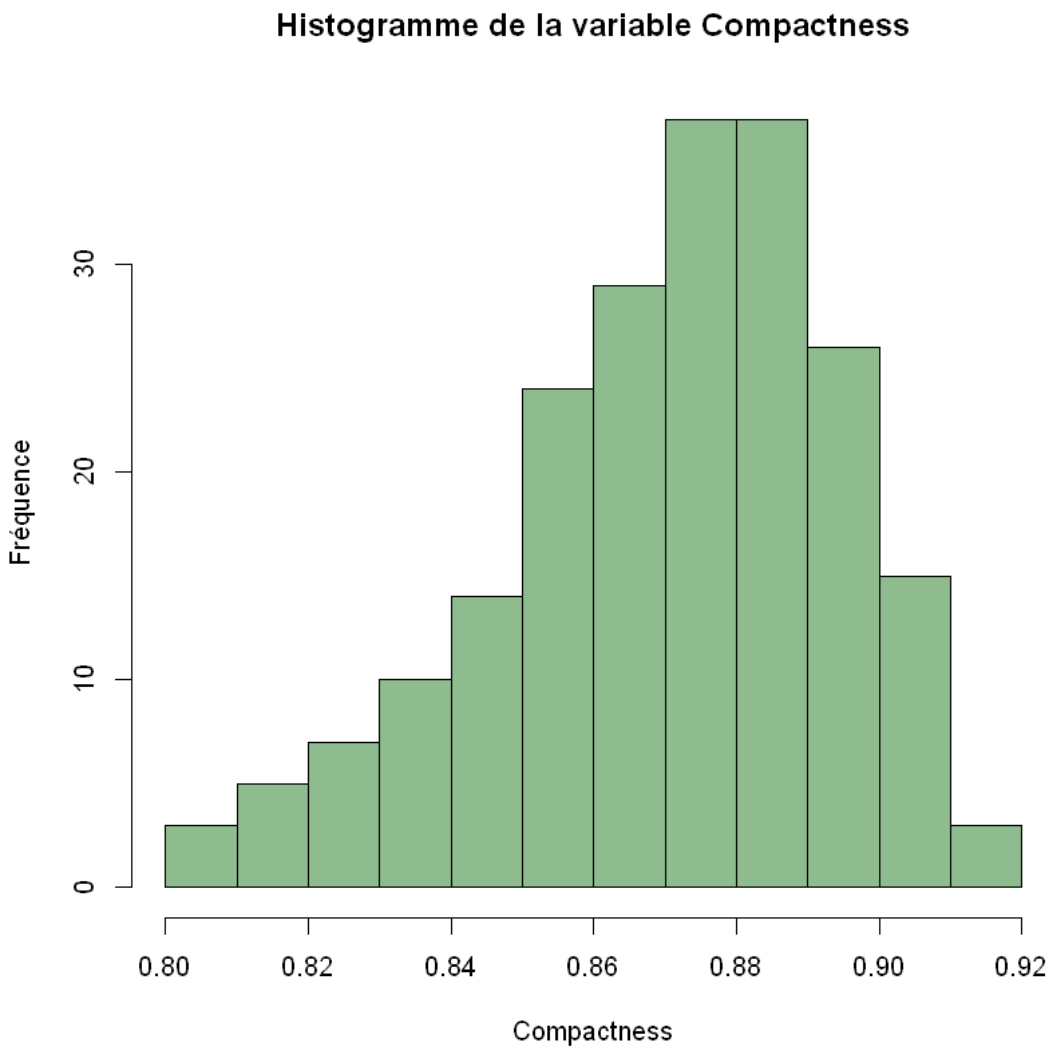
L'histogramme 2 semble representer deux lois normales ce qui viens appuyer l'hypothèse selon laquelle il y aurait au moins deux graines.

[29]:
```
hist(data_seeds$perimeter,col = "darksalmon",main ="Histogramme de la variable␣
↪Perimeter",xlab = "Perimeter",ylab="Fréquence")
```



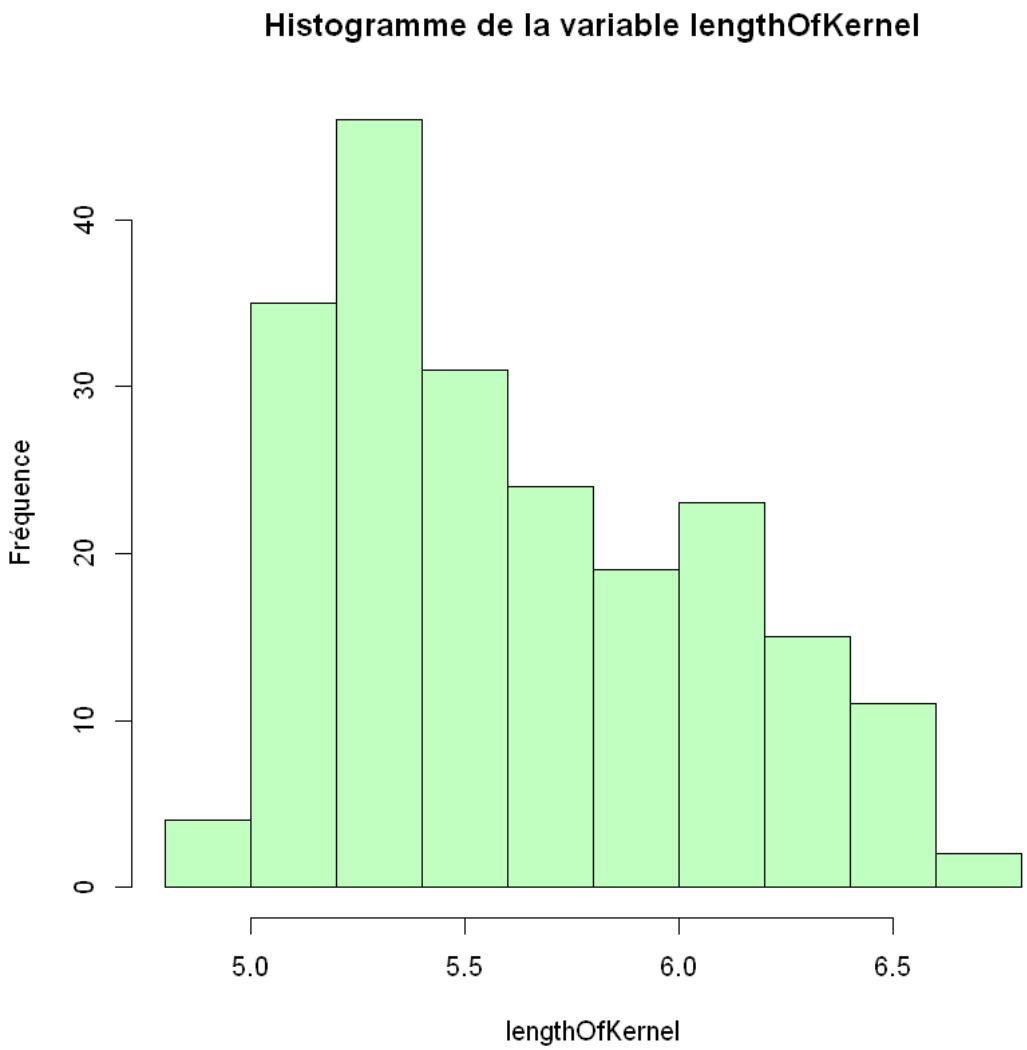
L'histogramme 3 ressemble à l'histogramme d'une loi normale.

[31]:
```
hist(data_seeds$compactness,col = "darkseagreen",main ="Histogramme de la␣
↪variable Compactness",xlab = "Compactness",ylab="Fréquence")
```



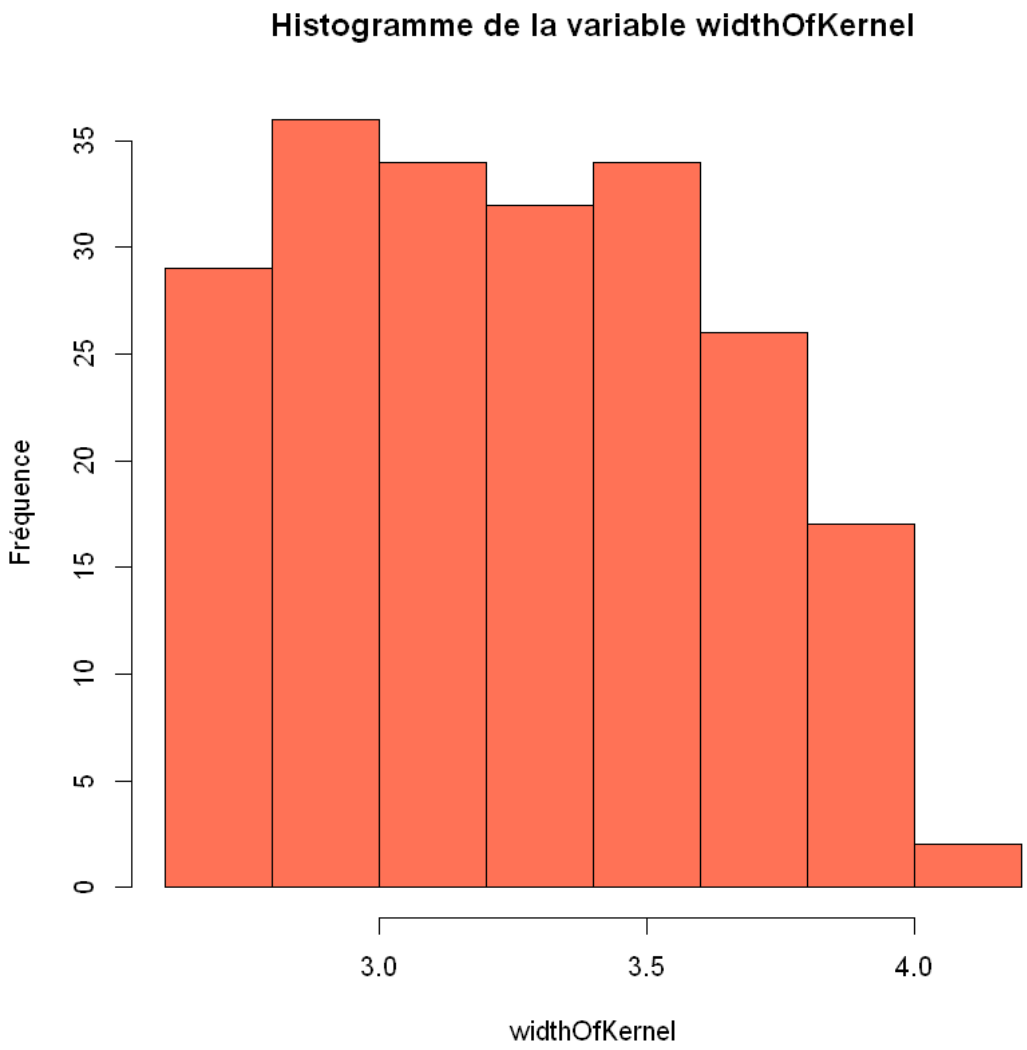
L'histogramme 4 semble représenter deux lois normales donc ce qui appuie l'hypothèse selon laquelle il y aurait au moins deux graines.

[32]:
```
hist(data_seeds$lengthOfKernel,col = "darkseagreen1",main ="Histogramme de la␣
↪variable lengthOfKernel",xlab = "lengthOfKernel",ylab="Fréquence")
```



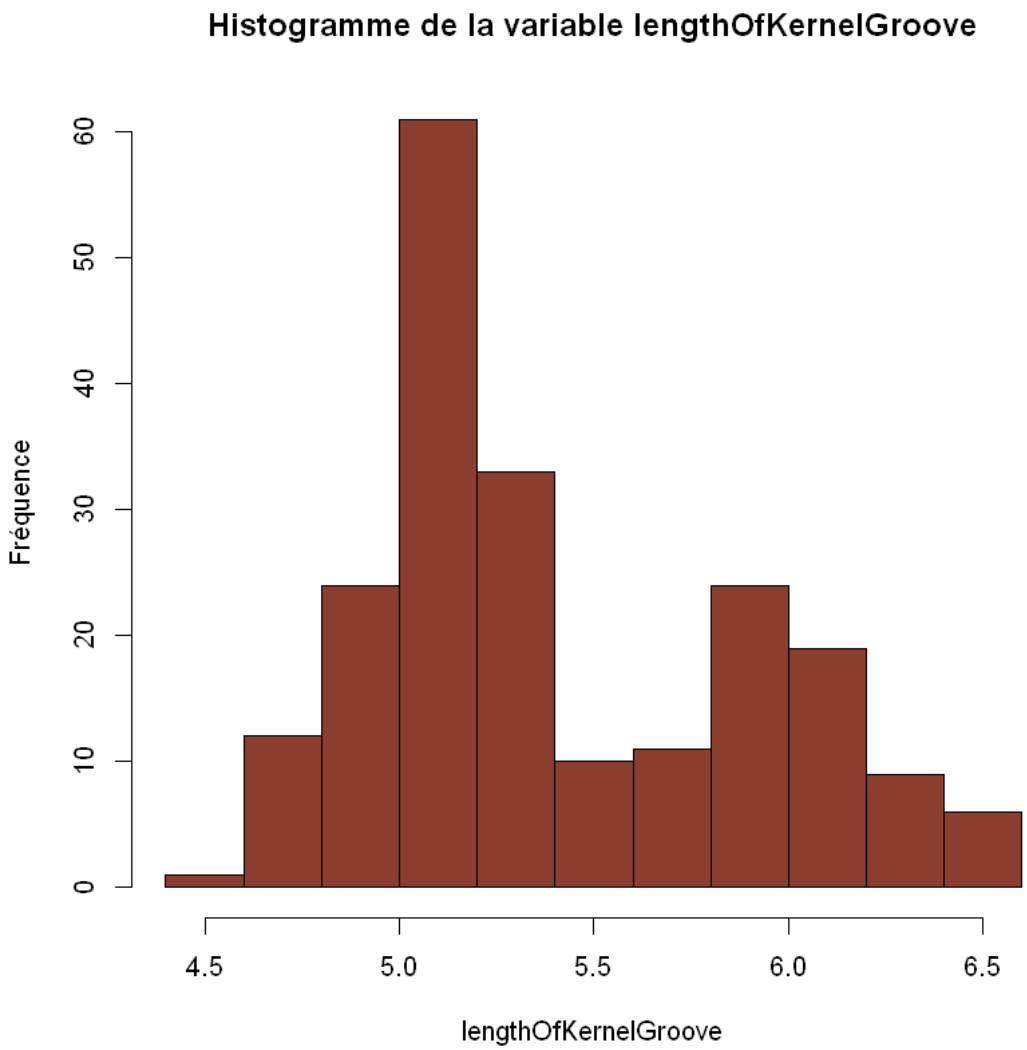
Histogramme 5: Il est difficile de distinguer quoique ce soit.

[33]:
```
hist(data_seeds$widthOfKernel,col = "coral1",main ="Histogramme de la variable␣
↪widthOfKernel",xlab = "widthOfKernel",ylab="Fréquence")
```

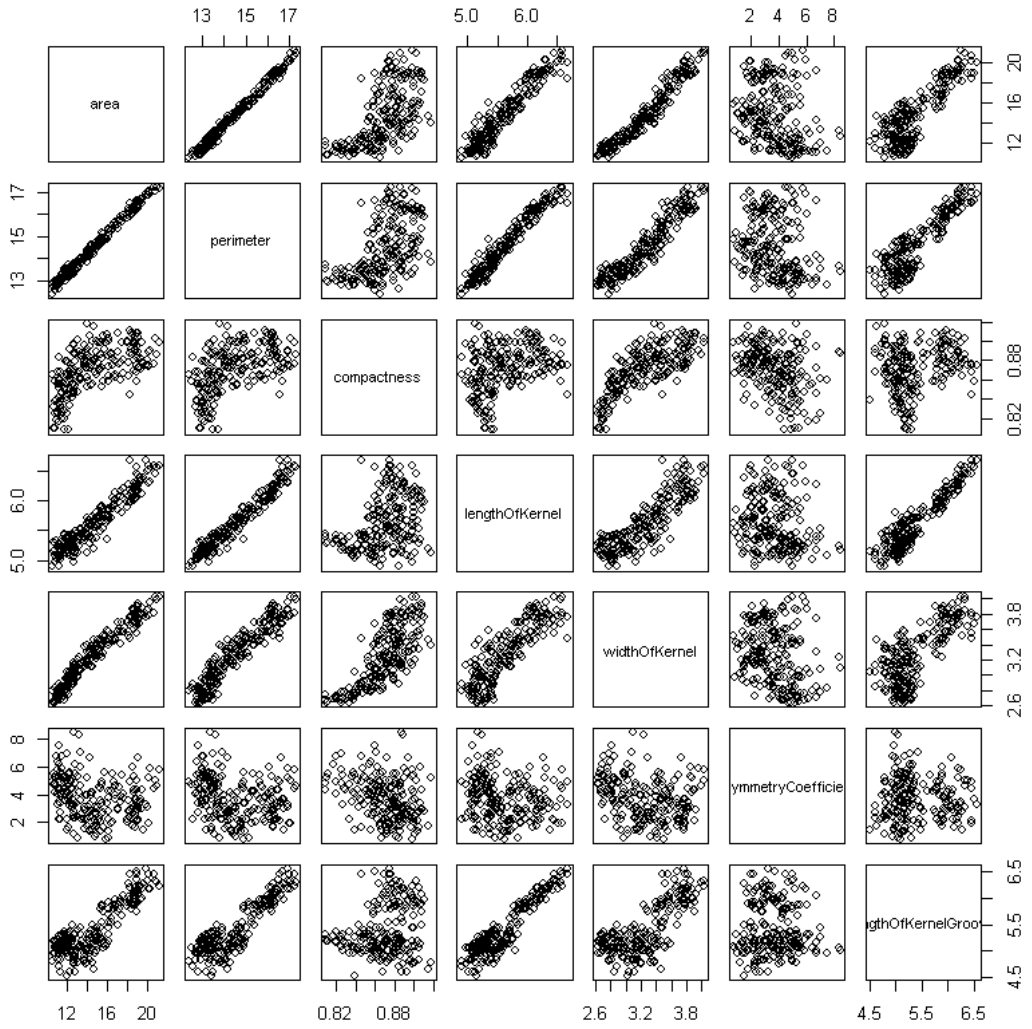


Histogramme 6 Une fois de plus l'histogramme 5 semble représenter deux lois normales donc ce qui appuie l'hypothèse selon laquelle il y aurait au moins deux graines.

[35]:
```
hist(data_seeds$lengthOfKernelGroove,col = "coral4",main ="Histogramme de la␣
↪variable lengthOfKernelGroove",xlab =␣
↪"lengthOfKernelGroove",ylab="Fréquence")
```



[36]:
```
plot(data_seeds)
```



Voyons si une ACP pourrait nous apporter d'avantages d'informations.

[44]:
```
pca_seeds<-PCA(data_seeds,scale.unit = FALSE,graph = F)
```

On regarde la proportion de variance expliquée pour les axes principaux

[46]:
```
pca_seeds$eig
```

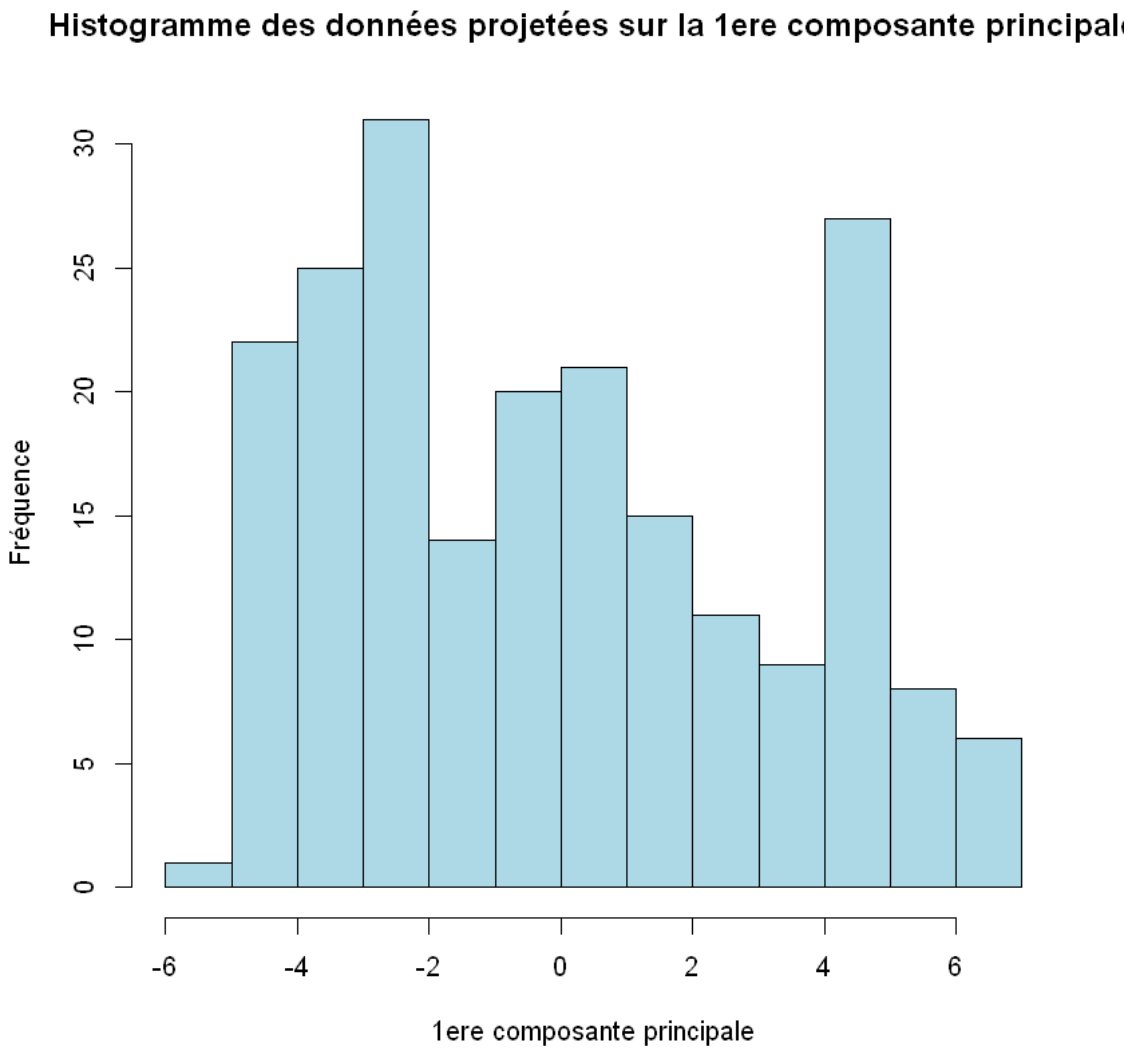
	eigenvalue	percentage of variance	cumulative percentage of variance
comp 1	1.074193e+01	8.293852e+01	82.93852
comp 2	2.119315e+00	1.636325e+01	99.30176
comp 3	7.327941e-02	5.657909e-01	99.86756
comp 4	1.282613e-02	9.903061e-02	99.96659
comp 5	2.735140e-03	2.111803e-02	99.98770
comp 6	1.562971e-03	1.206771e-02	99.99977
comp 7	2.951423e-05	2.278796e-04	100.00000

On remarque que la composante 1 de l'acp explique 83% de la variance.

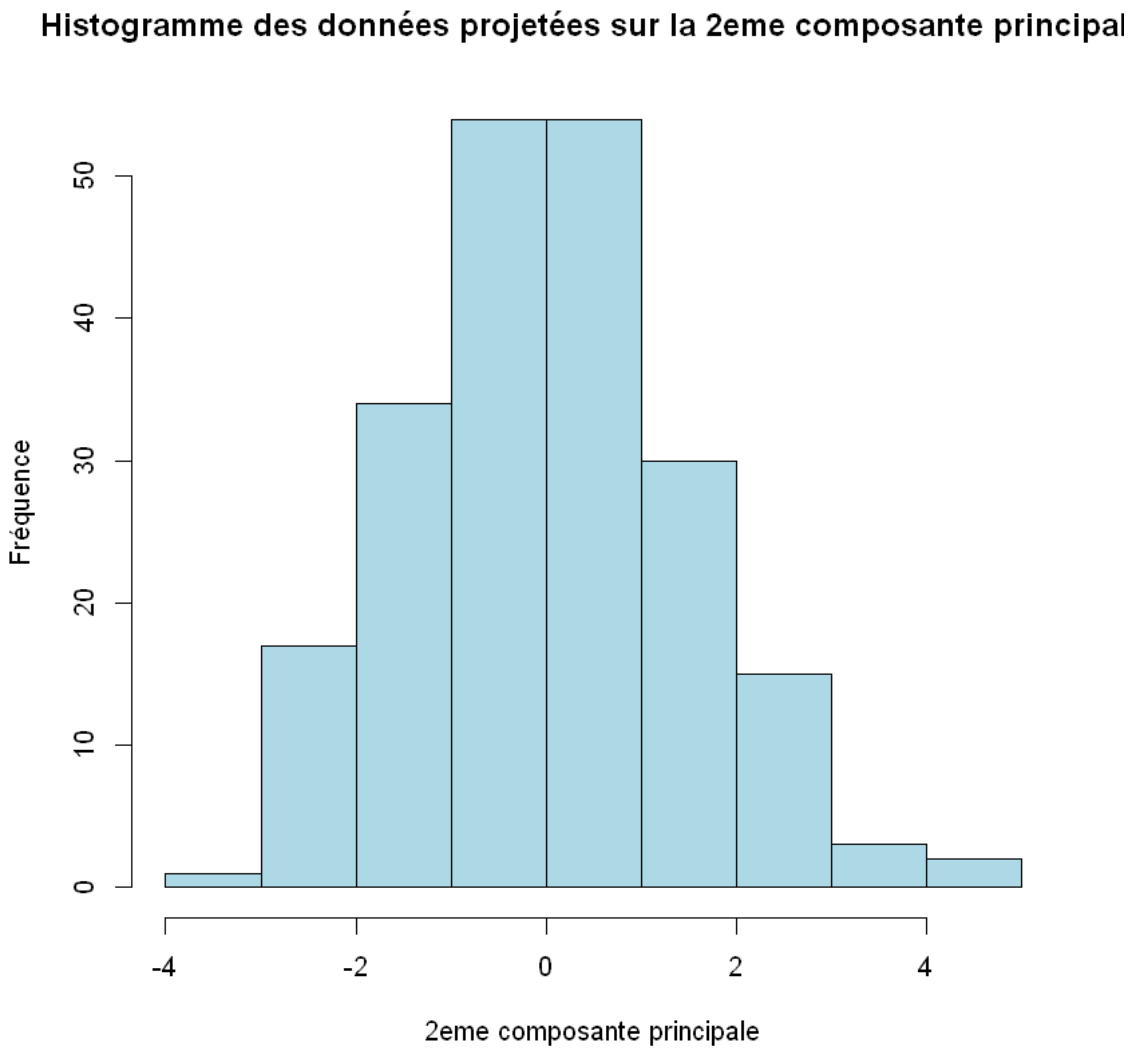
On remarque que la composante 2 de l'acp explique 16% de la variance.

Les deux composantes principales expliquent donc 99% de la variance.

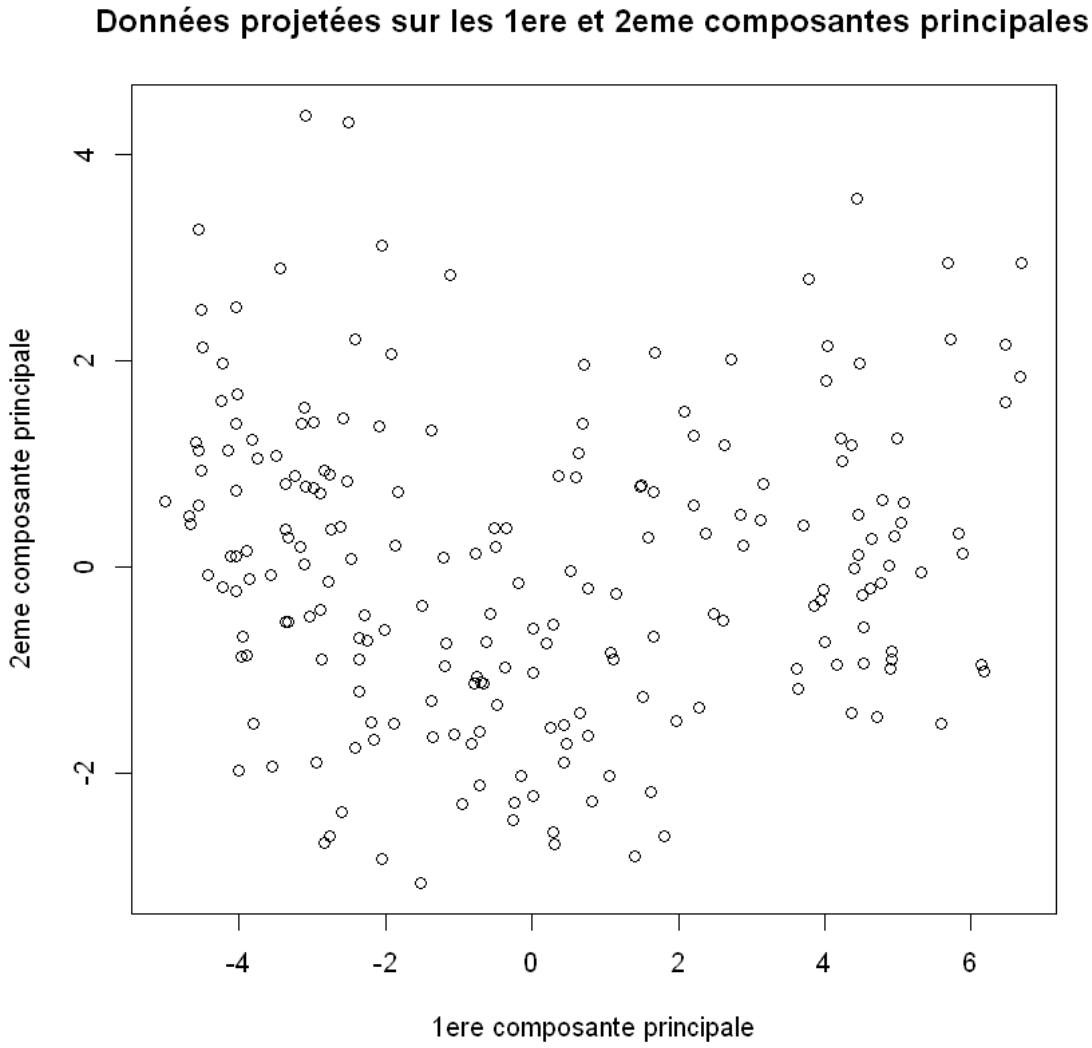
[50]:
```
hist(pca_seeds[["ind"]][["coord"]][,1],col = "lightblue",main ="Histogramme des␣
↪données projetées sur la 1ere composante principale",xlab = "1ere composante␣
↪principale",ylab="Fréquence")
```



[49]:
```
hist(pca_seeds[["ind"]][["coord"]][,2],col = "lightblue",main ="Histogramme des␣
↪données projetées sur la 2eme composante principale",xlab = "2eme composante␣
↪principale",ylab="Fréquence")
```



[51]:
```
plot(pca_seeds[["ind"]][["coord"]][,1],pca_seeds[["ind"]][["coord"]][,2],main
↪="Données projetées sur les 1ere et 2eme composantes principales",xlab =
↪"1ere composante principale",ylab="2eme composante principale")
```



La projection sur la premiére composante princiaple confirme nos supositions,au contraire, la projection sur la deuxiéme composante principale ne laisse en aucun cas penser à l'éxistence de différents groupes (l'histogramme répresente une loi normale centrée en 0) nous devons donc rester prudent quant à l'interprétation de nos résultats.

4 K-MEANS

Objectif : Arriver à séparer nos données en 3 clusters (on sait qu'il existe 3 types de graines).

Le but est de séparer le data set seeds en plusieurs clusters.

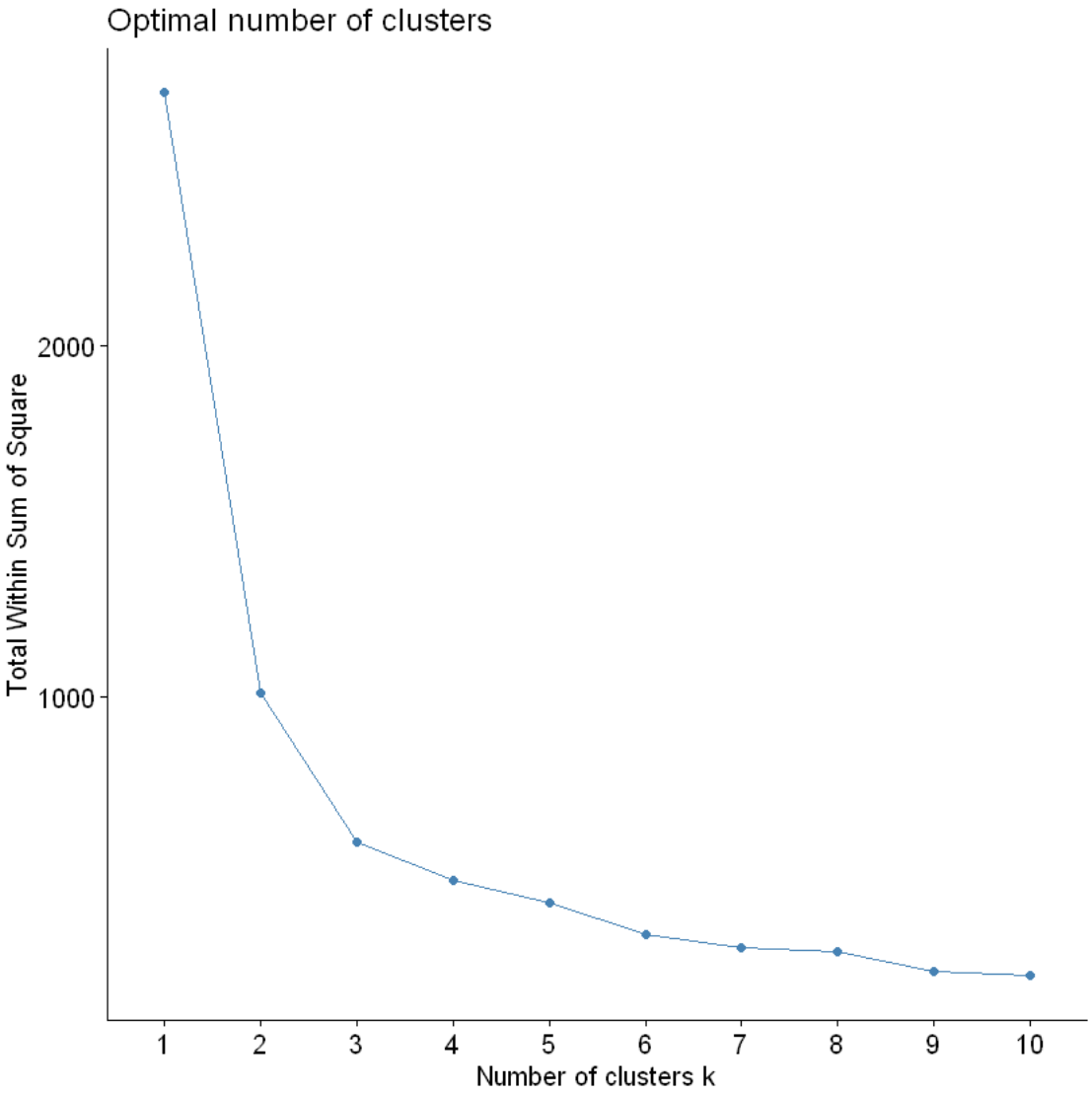
Bien evidement, le paramètre le plus important à déterminer pour le k-means est le nombre de cluster dans lequel l'algorithme va classer nos différentes données.

Pour cela, on peut s'aider de l'elbow méthode qui pourra nous donner une premiére intuition sur le nombre de cluster à choisir.

Elbow method

Le graphique suivant nous permet de dire que 3 clusters est un choix judicieux.

[38]:
```
fviz_nbclust(data_seeds, kmeans, method = "wss")
```



On applique un K_mean avec 3 centroides, 3 points de départ choisit de manière arbitraire et 100 points de départ.

D'aprés ce graphique, on peut raisonnablement choisir entre 3 clusters.

On effectue le K-means avec 3 centroides. Les points de départs de l'algorithme sont décidés aléatoirement, nsart =100 signifie que l'algo prend 100 points de départs différents et choisi le

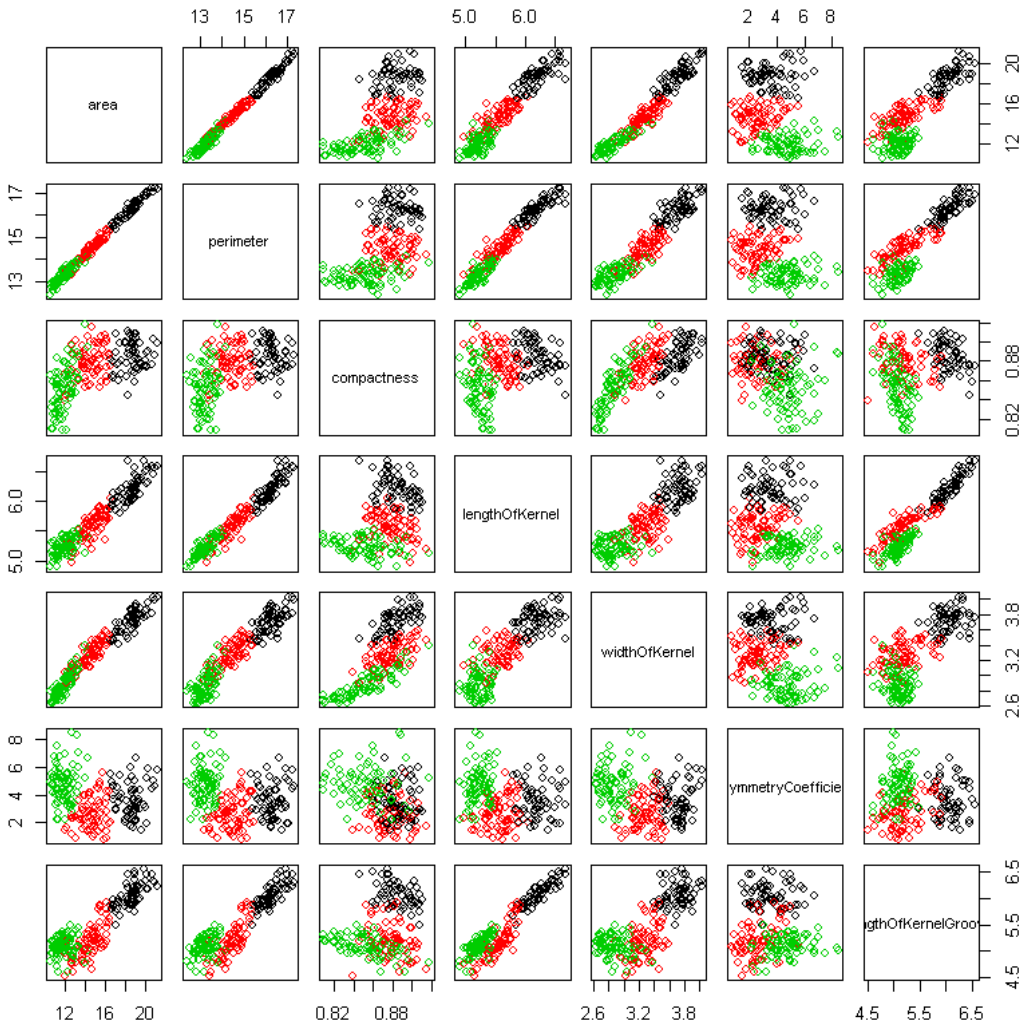
meilleur.

[40]:
```
kmeans_res<-kmeans(data_seeds, centers = 3, nstart = 100)

iteration<-kmeans_res$iter
```

L'algorithme converge assez vite au bout de 2 à 3 itérations.

[41]:
```
print(iteration)
plot(data_seeds,col=kmeans_res$cluster)
```

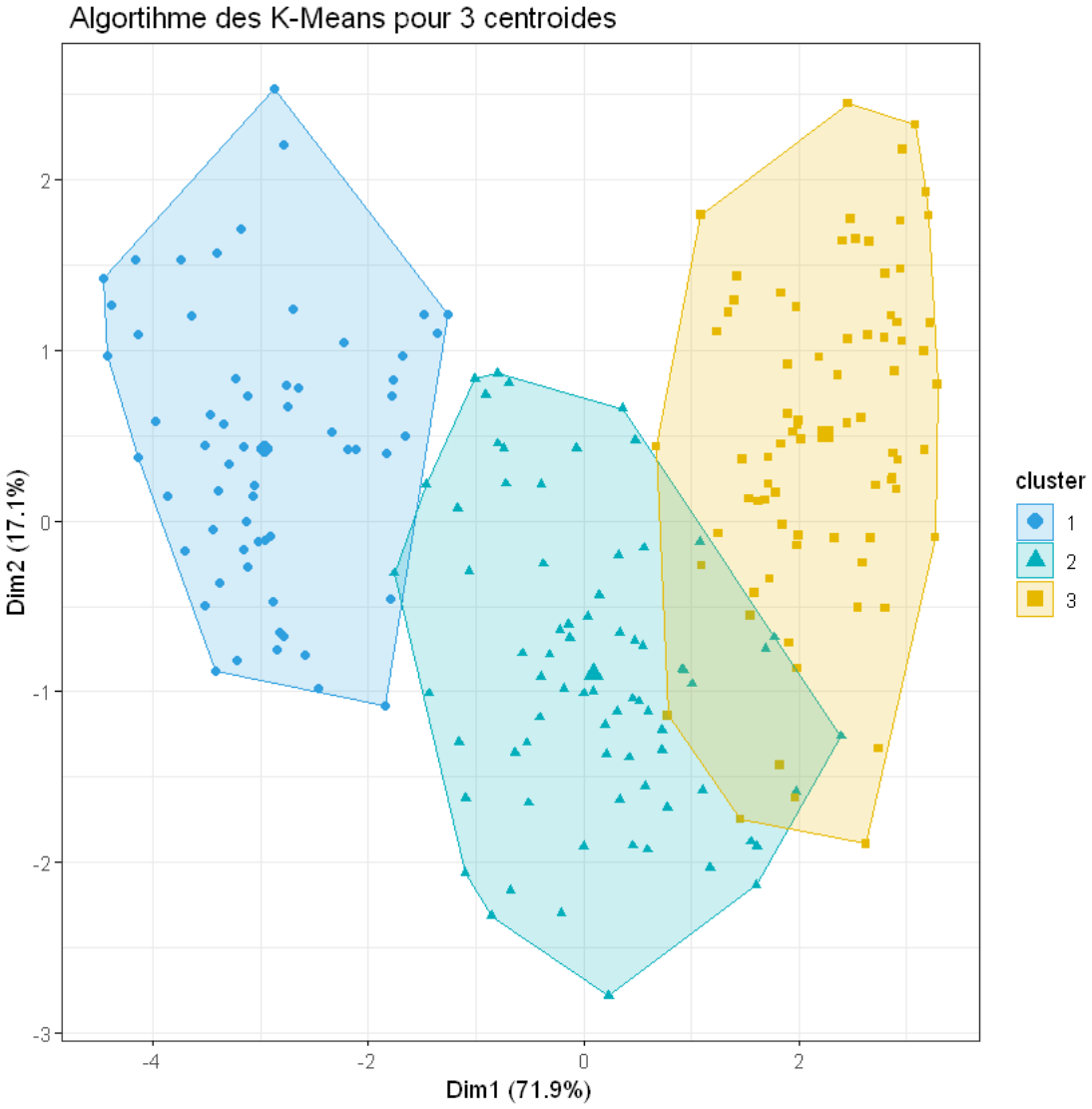
```
[1] 3
```



Les données sont séparées de manière assez distincte.

On affiche le résultat de l'algorithme par rapport à chaque couples variables, on voit que les données on été plutot bien séparées.

[52]:
```
fviz_cluster(kmeans_res,data_seeds, palette = c("#2E9FDF", "#00AFBB",␣
↪"#E7B800"), geom = "point", ellipse.type = "convex",  ggtheme =␣
↪theme_bw(),main =" Algortihme des K-Means pour 3 centroides")
```

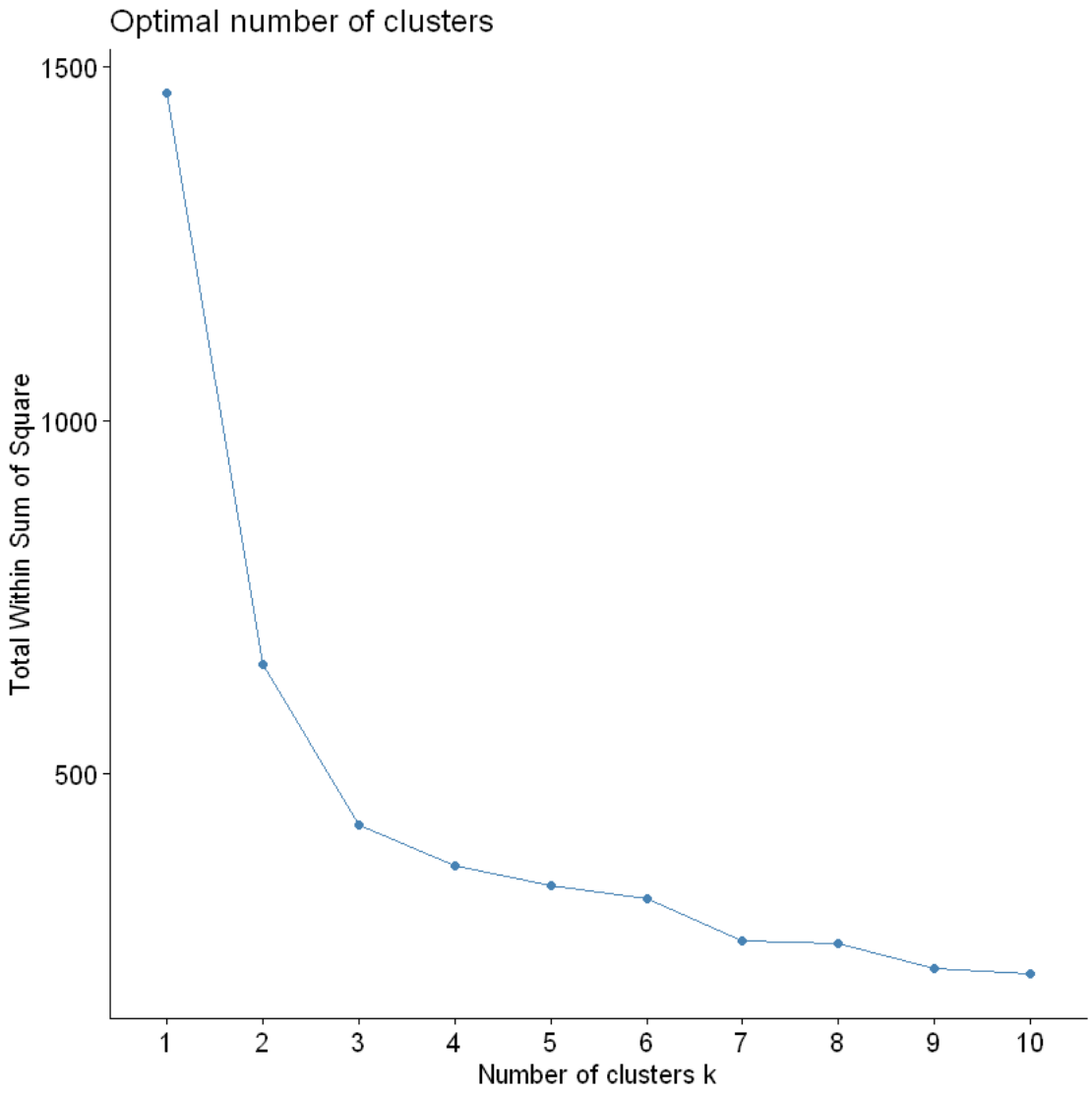


Même si les frontières entre les groupes se croisent, les résultats sont plutôt correctes.

Les unités de mesures de nos diffèrentes variables ne sont pas les mêmes afin de palier à ce problème, On standardise nos données afin d'assurer une interopérabilité optimale des données.

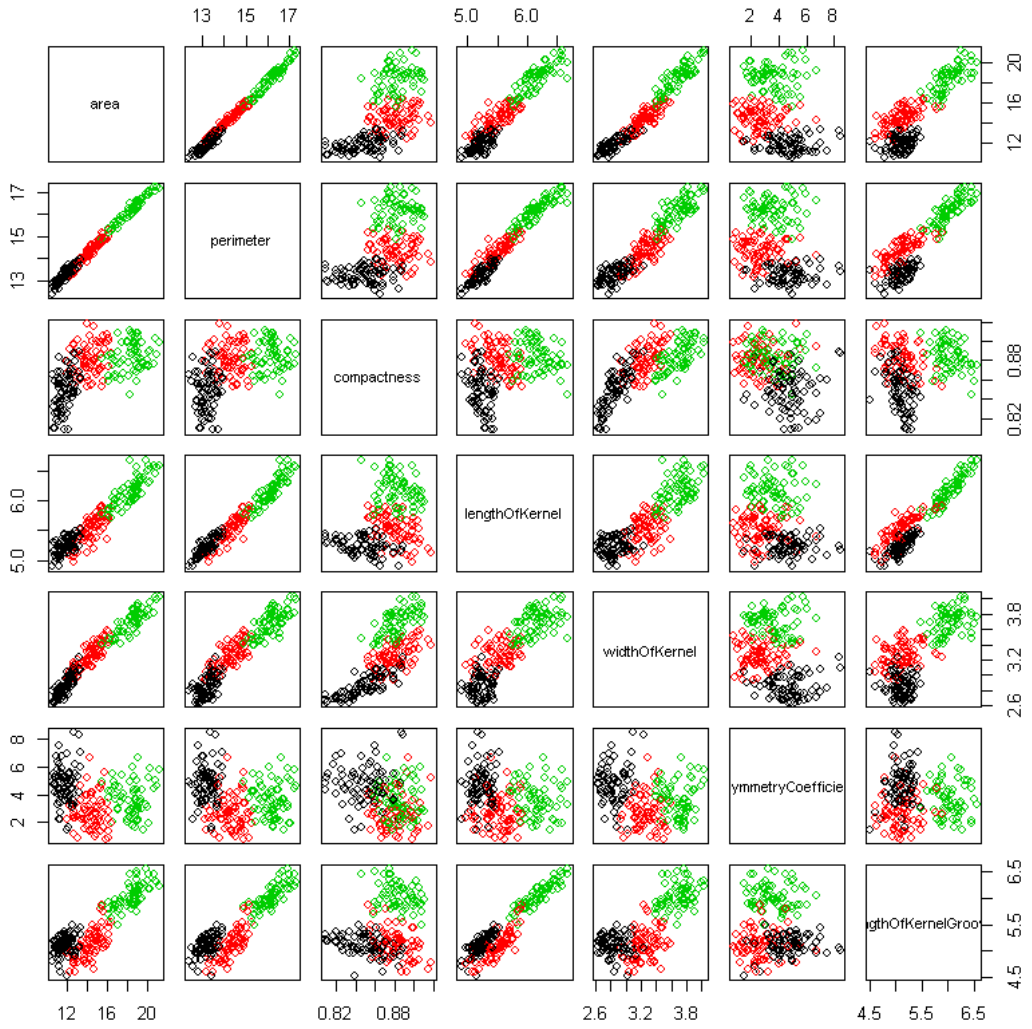
[53]:
```
seedtrain_scaled<-scale(data_seeds)
```

```
fviz_nbclust(seedtrain_scaled, kmeans, method = "wss")

kmeans_res_s<-kmeans(seedtrain_scaled, centers = 3, nstart = 100)
```

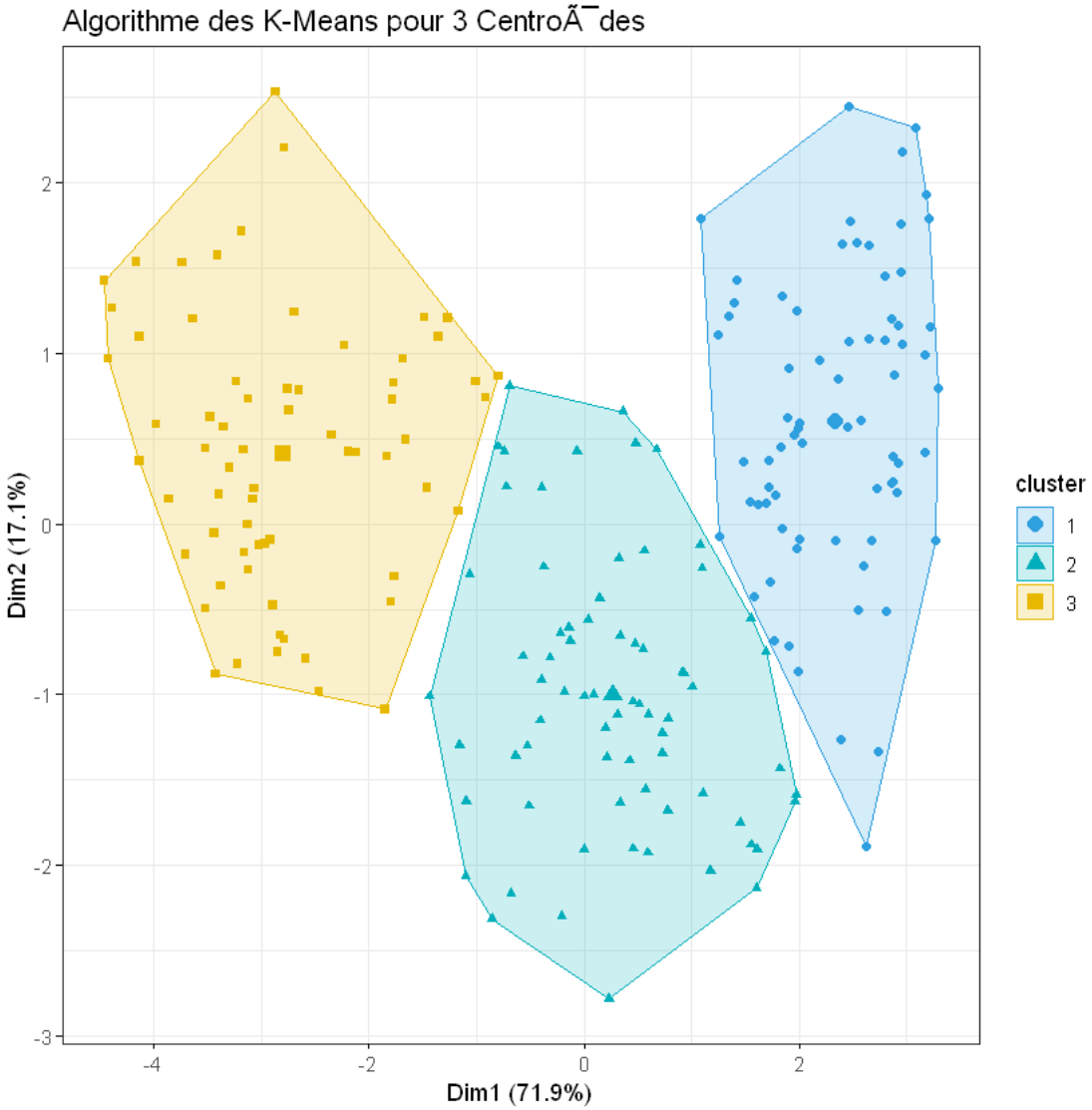


[54]:
```
iteration<-kmeans_res_s$iter
```

[55]:
```
plot(data_seeds,col=kmeans_res_s$cluster)
```



[56]:
```
fviz_cluster(kmeans_res_s,data_seeds, palette = c("#2E9FDF", "#00AFBB",␣
↪"#E7B800"), geom = "point", ellipse.type = "convex",  ggtheme =␣
↪theme_bw(),main ="Algorithme des K-Means pour 3 CentroÃ¯des")
```



On remarque que nos clusters ne s'intersectent plus, ce qui est plutot appreciable.

Dans notre problème on suppose donc que scale les données peut étre intéressant.

Etape 1

[58]:
```
step1 <- kmeans(seedtrain_scaled, centers = 3, nstart = 1 ,iter.max = 1)
step_1 <- fviz_cluster(step1,data_seeds, palette = c("#2E9FDF", "#00AFBB",␣
 ↪"#E7B800"), geom = "point", ellipse.type = "convex",  ggtheme =␣
 ↪theme_bw(),main ="K-Means Algorithme apres 1 itérations")
```

```
Warning message:
"did not converge in 1 iteration"
```

Etape 2

[59]:
```
step2 <- kmeans(seedtrain_scaled, centers = step1$center, nstart = 1 ,iter.max␣
↪= 1)
step_2 <- fviz_cluster(step2,data_seeds, palette = c("#2E9FDF", "#00AFBB",␣
↪"#E7B800"), geom = "point", ellipse.type = "convex",  ggtheme =␣
↪theme_bw(),main ="K-Means Algorithme apres 2 itérations")
```

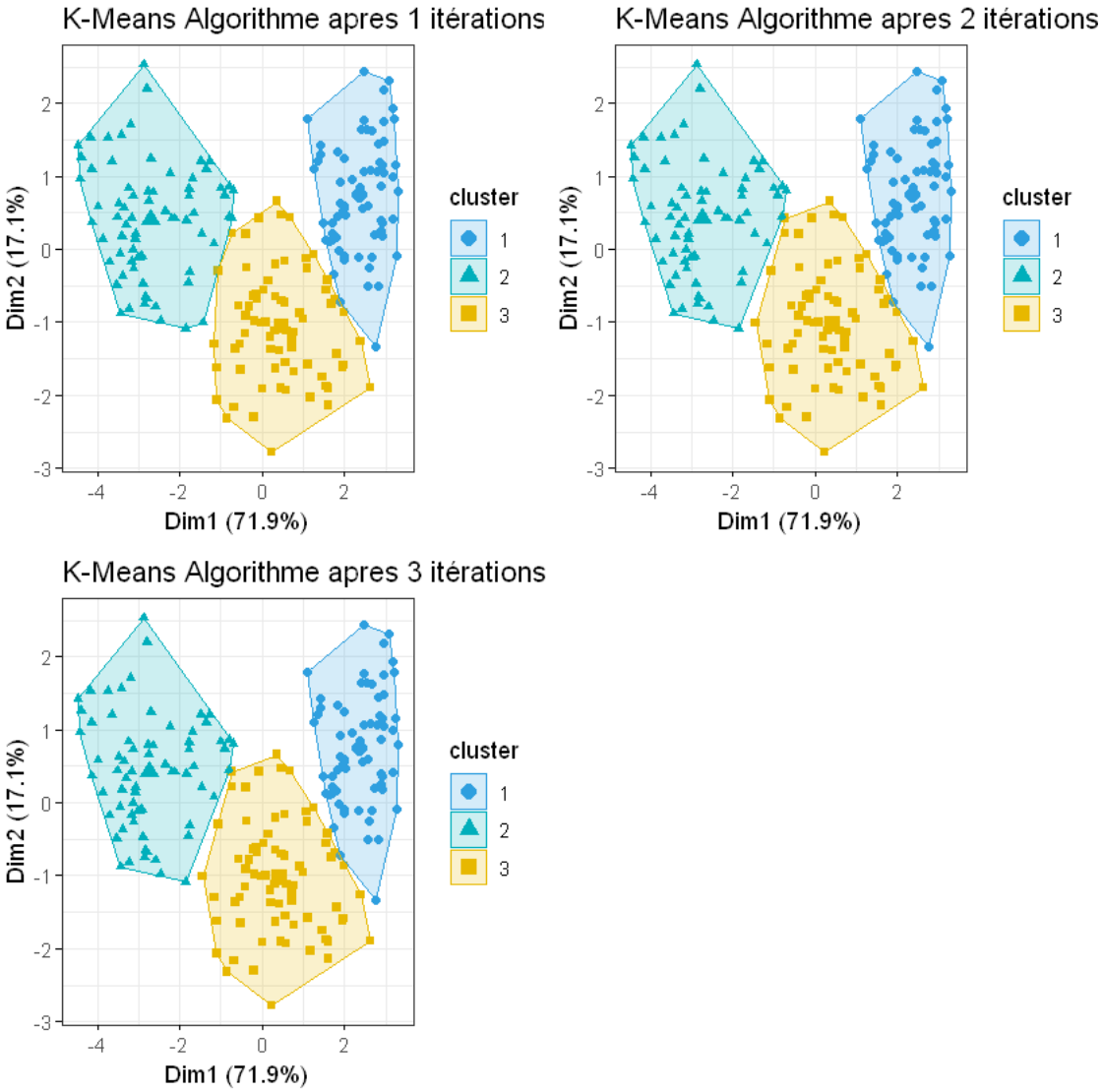
```
Warning message:
"did not converge in 1 iteration"
```

Etape 3

[60]:
```
step2<-kmeans(seedtrain_scaled, centers = step2$center, nstart = 1 ,iter.max =␣
↪1)
step_3<-fviz_cluster(step2,data_seeds, palette = c("#2E9FDF", "#00AFBB",␣
↪"#E7B800"), geom = "point", ellipse.type = "convex",  ggtheme =␣
↪theme_bw(),main ="K-Means Algorithme apres 3 itérations")
```

On observe les résultats du kmeans aprés chaque itérations, on voit qu'il progresse aprés la premiére itération mais on observe peu de différence entre l'étape 2 et 3, en effet l'algorithme converge trés vite (en fonction des centroides de départ il peut converger en 2 étapes)

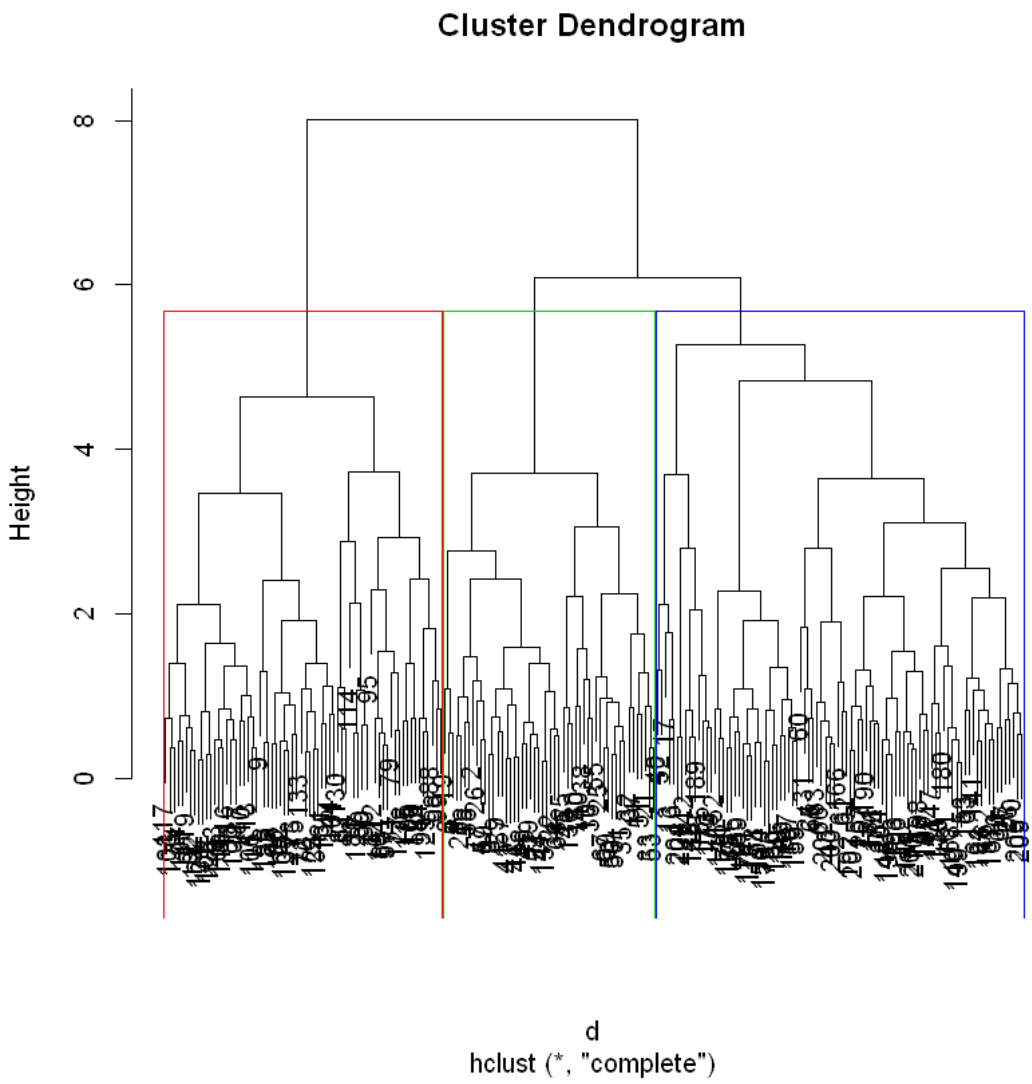
[61]:
```
plot_grid(step_1, step_2,step_3)
```



5 Clustering hiérarchique ascendant

Méthode Complete-Linkage

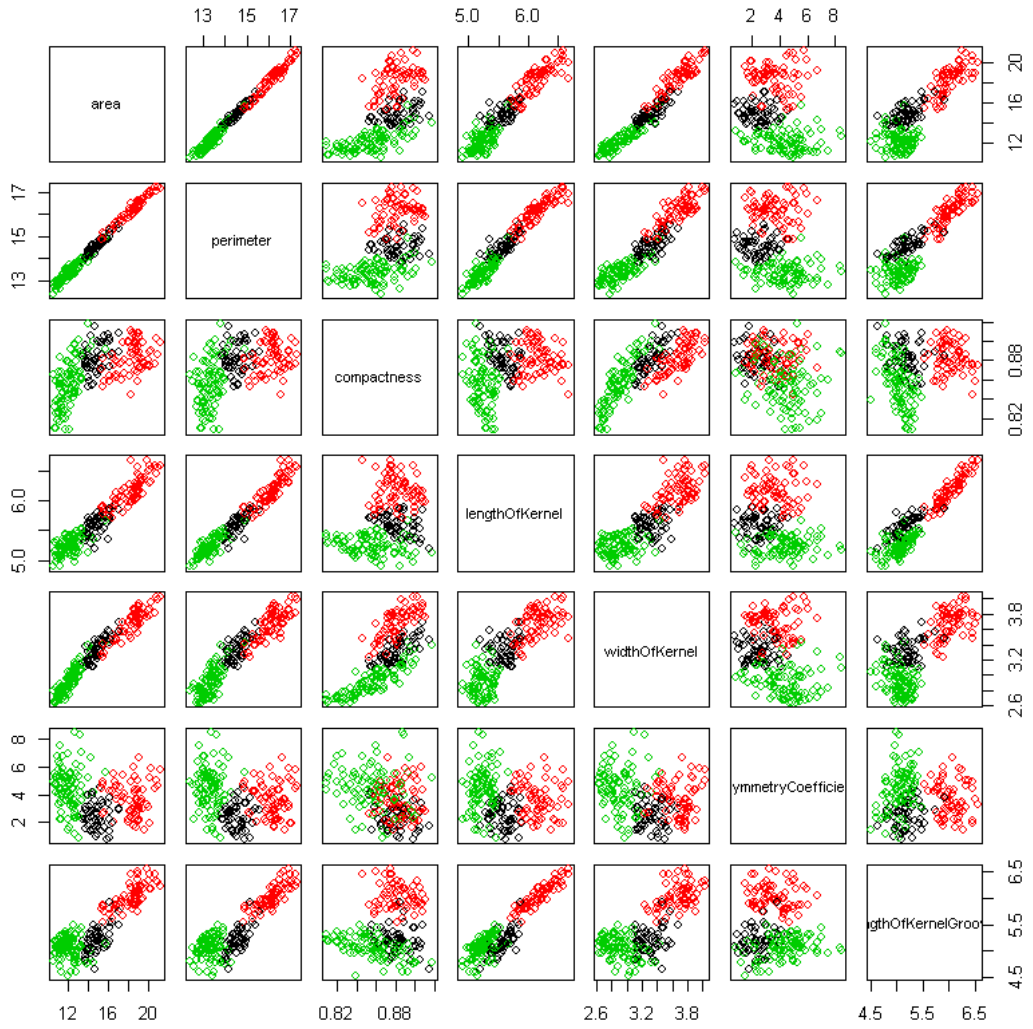
[66]:
```
d<-dist(seedtrain_scaled)
hc_c<-hclust(d,method='complete')
plot(hc_c)
rect.hclust(hc_c, k=3, border=2:5)
```



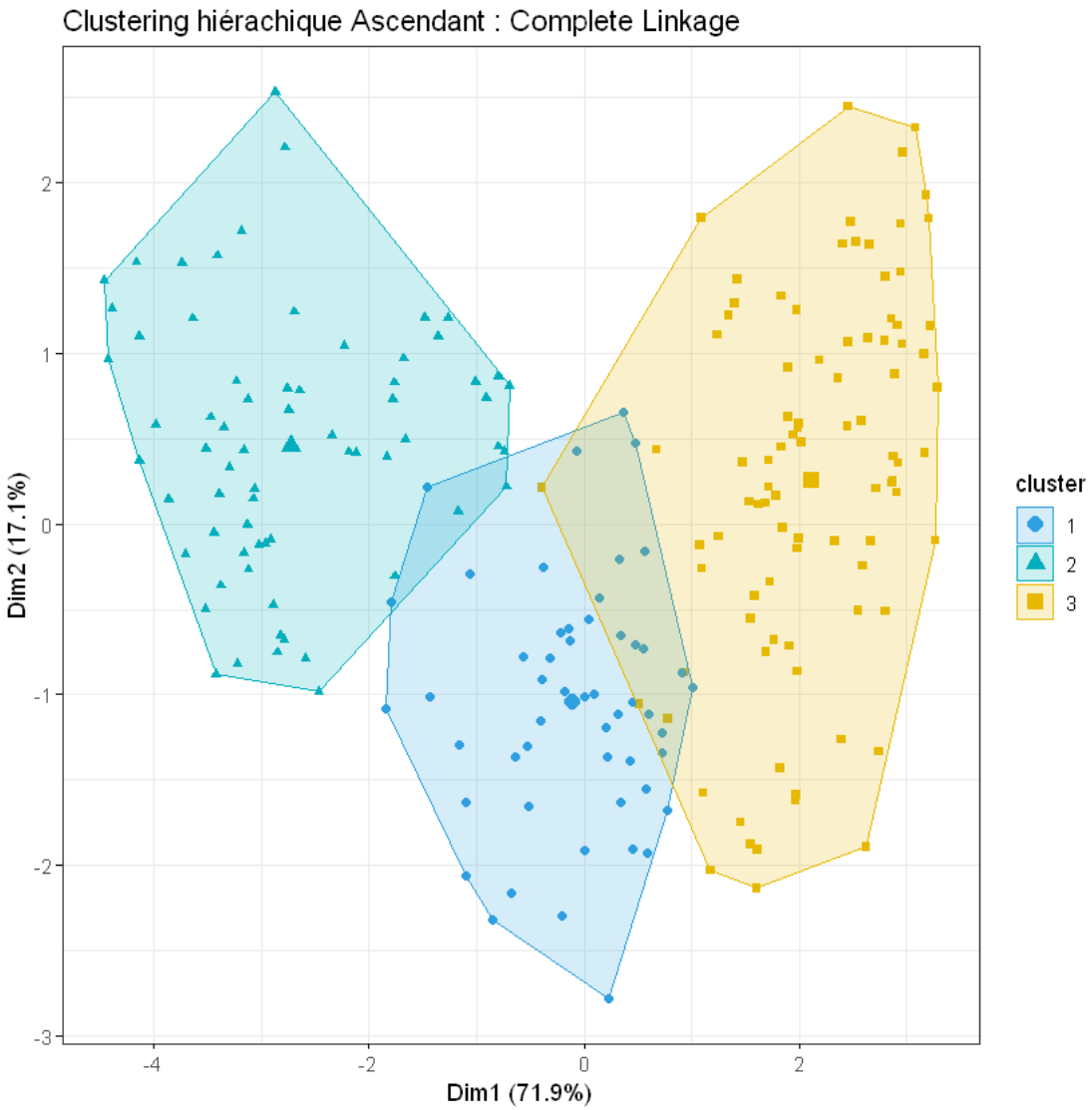
Visualisation du clustering Ascendant quand on découpe en 3 groupes

On récupere les clusters

[68]:
```
hc_clusters_c<-cutree(hc_c,k=3)
plot(data_seeds,col=hc_clusters_c)
```



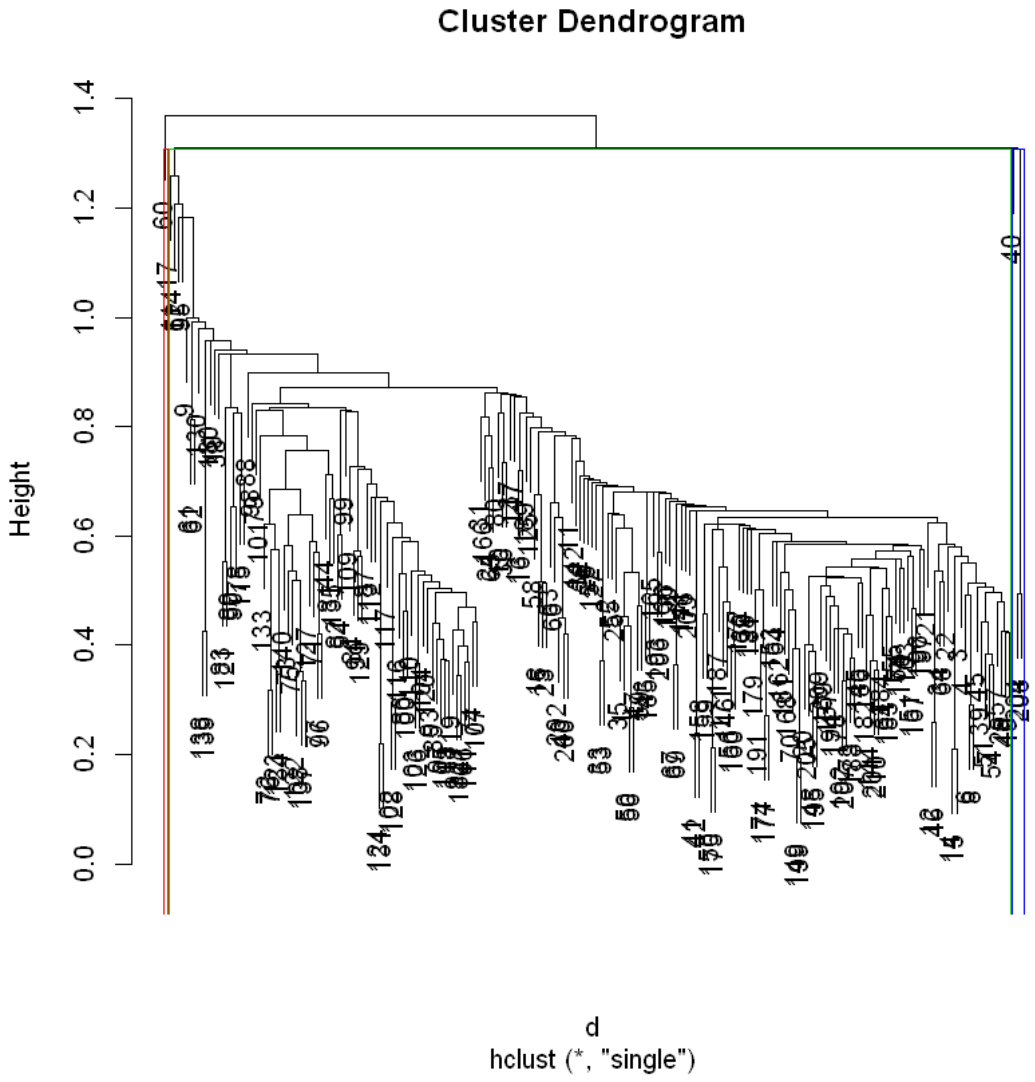
[69]:
```
fviz_cluster(object = list(data=data_seeds,cluster=hc_clusters_c), palette =␣
↪c("#2E9FDF", "#00AFBB", "#E7B800"), geom = "point", ellipse.type = "convex",␣
↪ ggtheme = theme_bw(),main ="Clustering hiérachique Ascendant : Complete␣
↪Linkage")
```



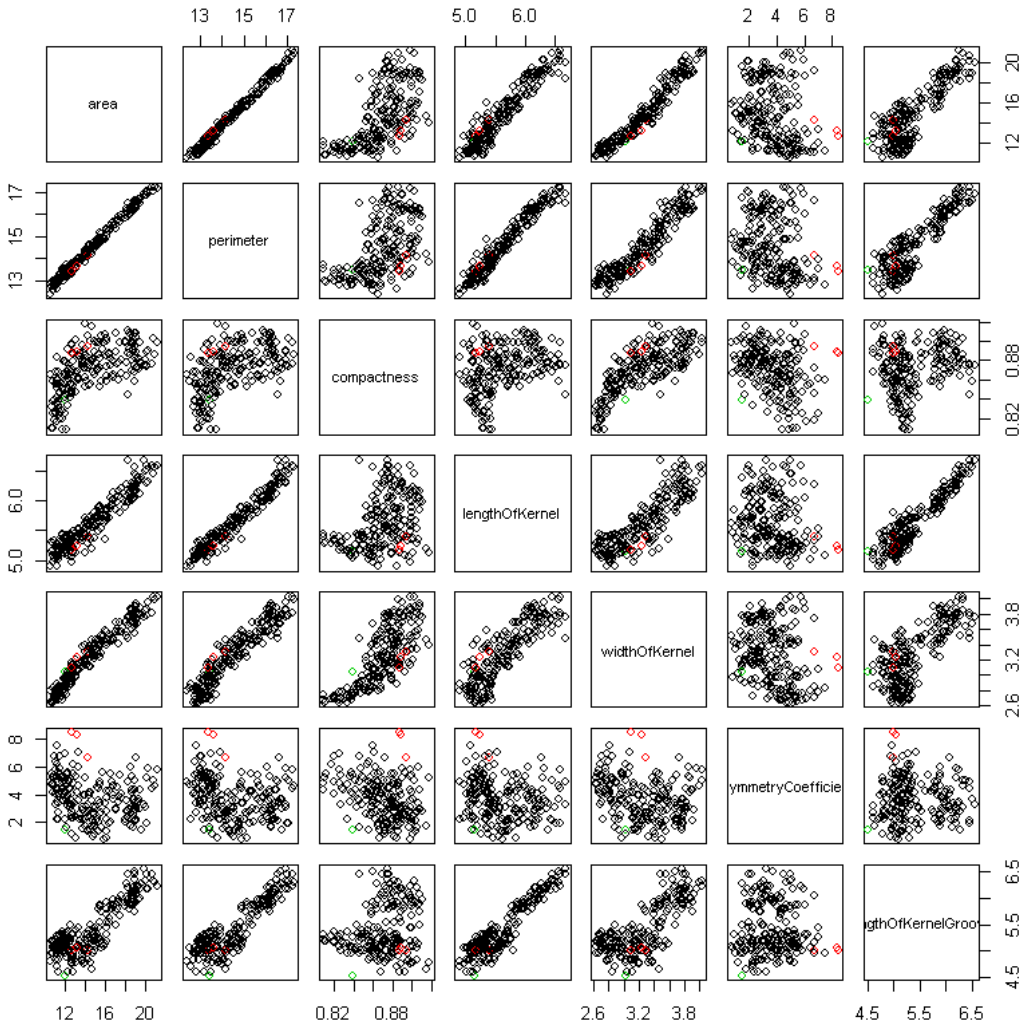
On affiche les clusters et leurs centroides sous forme d'ellispe grace à une ACP sur les deux axes principaux.

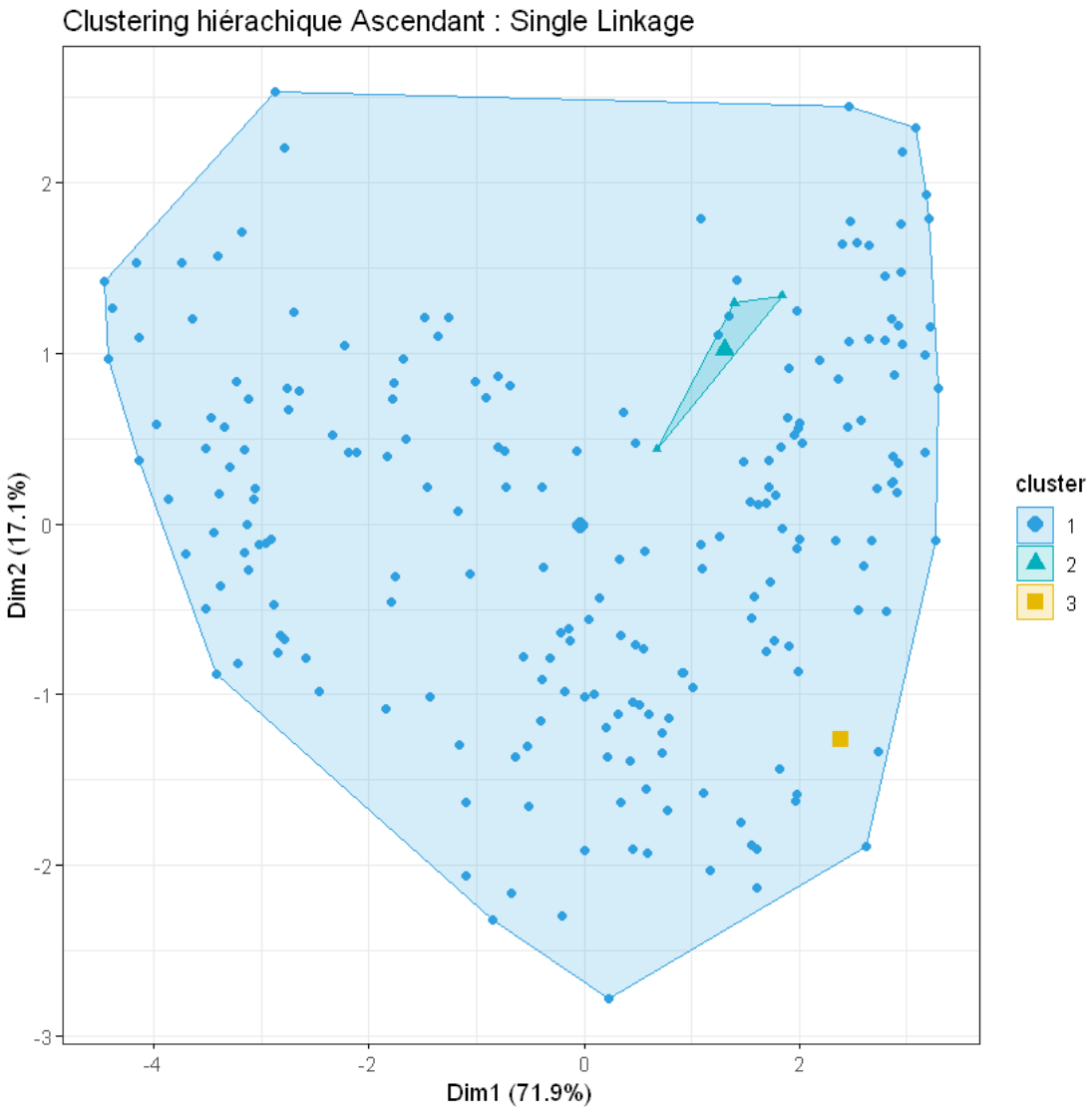
Méthode complete : on considère que la distance entre deux clusters est la distance entre leurs deux points les plus éloignés.

[71]:
```
hc_s<-hclust(d,method='single')
plot(hc_s)
rect.hclust(hc_s, k=3, border=2:5)
```



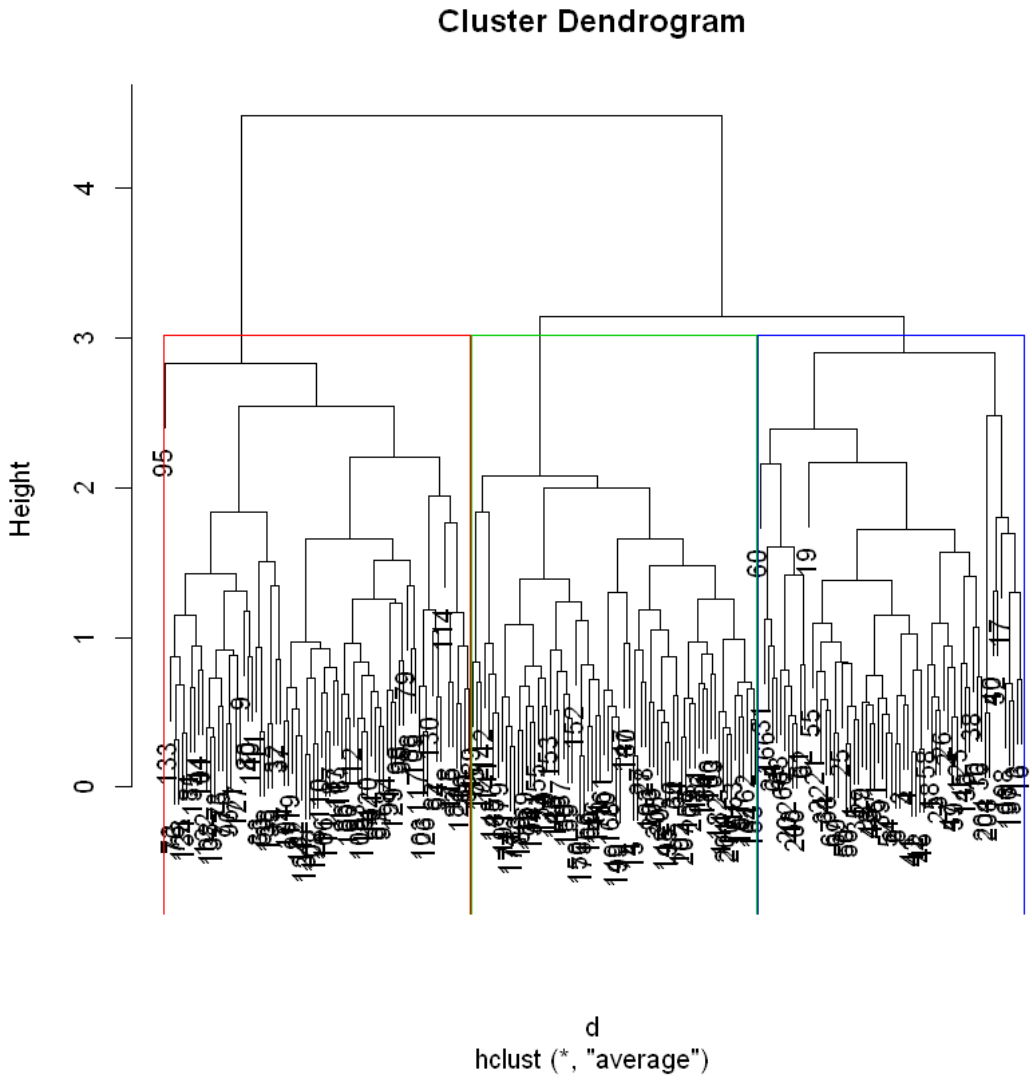
[88]:
```
hc_clusters_s<-cutree(hc_s,k=3)
plot(data_seeds,col=hc_clusters_s)
fviz_cluster(object = list(data=data_seeds,cluster=hc_clusters_s), palette =␣
↪c("#2E9FDF", "#00AFBB", "#E7B800"), geom = "point", ellipse.type = "convex",␣
↪ ggtheme = theme_bw(),main ="Clustering hiérachique Ascendant : Single␣
↪Linkage")
```


area
perimeter
compactness
lengthOfKernel
widthOfKernel
ymmetryCoefficie
gthOfKernelGroo

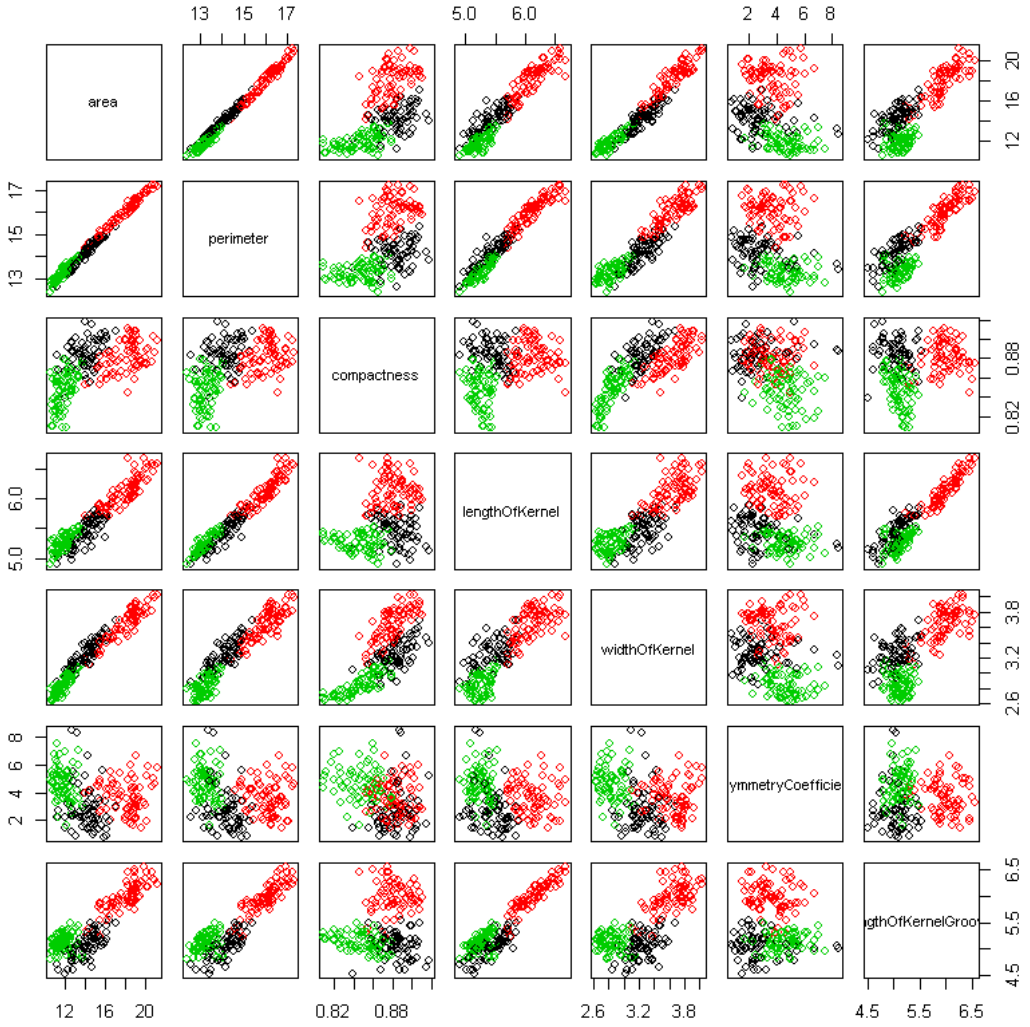


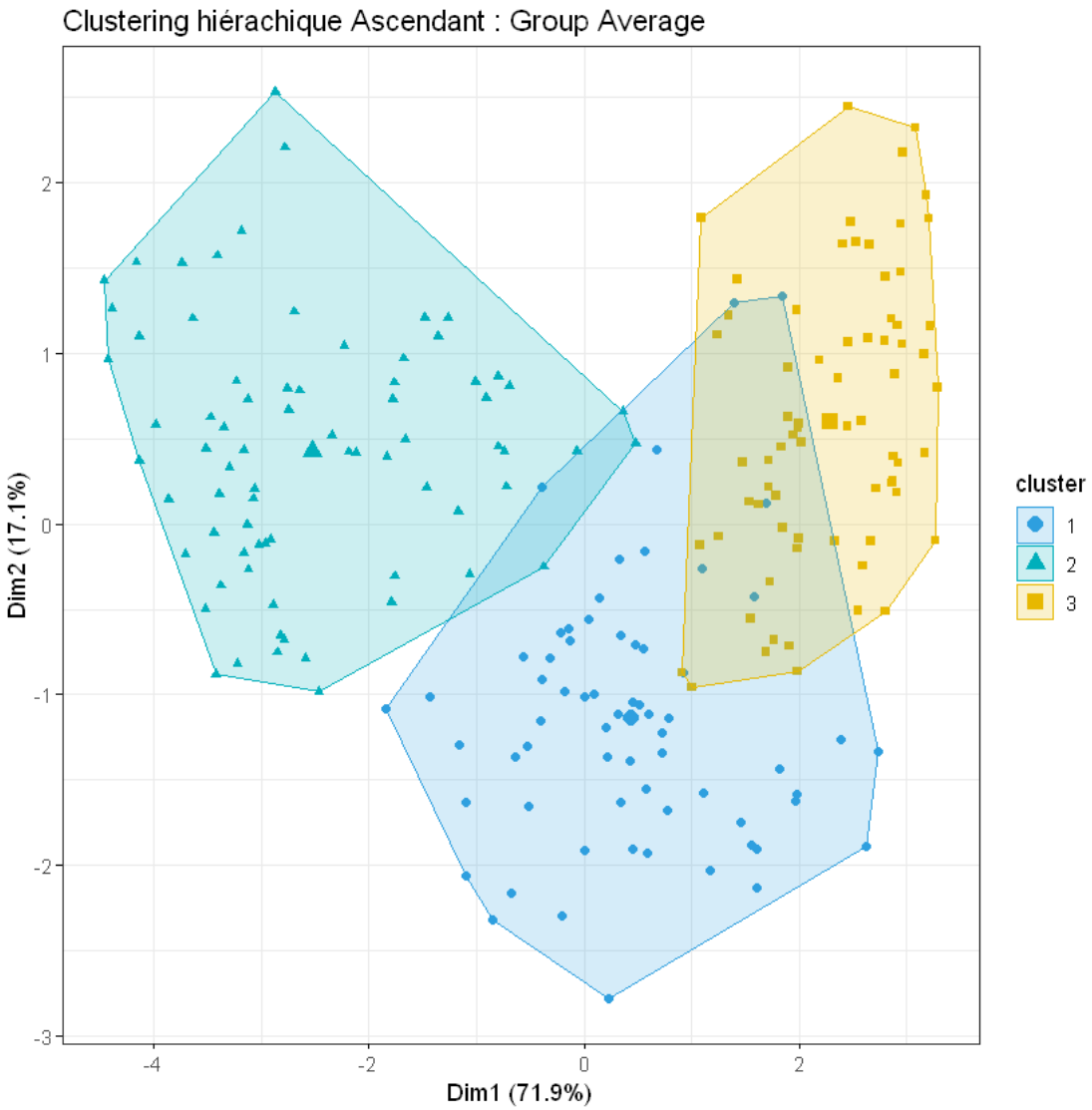
Méthode single : on considère que la distance entre deux clusters est la distance entre leurs deux points les plus proches.

[74]:
```
hc_a<-hclust(d,method='average')
plot(hc_a)
rect.hclust(hc_a, k=3, border=2:5)
```



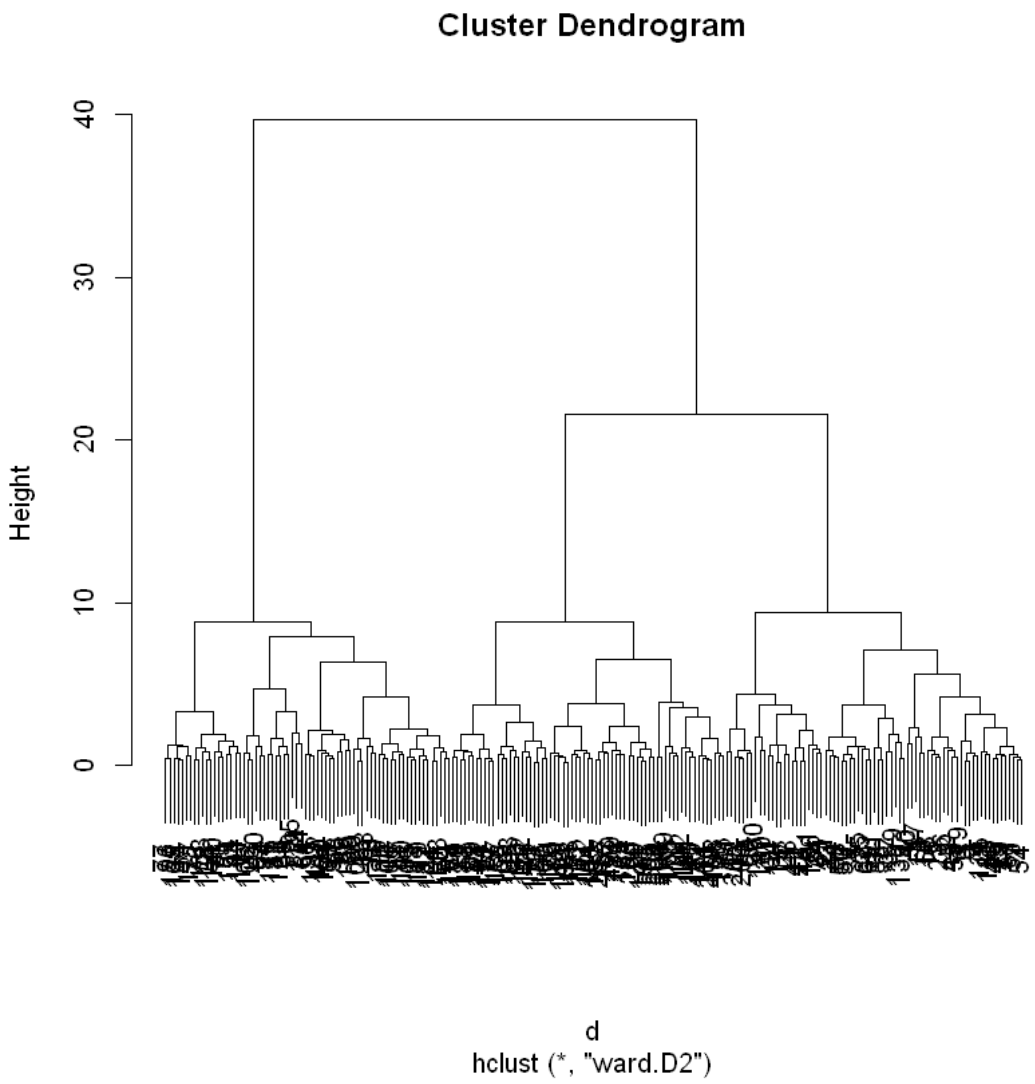
[75]:
```
hc_clusters_a<-cutree(hc_a,k=3)
plot(data_seeds,col=hc_clusters_a)
fviz_cluster(object = list(data=data_seeds,cluster=hc_clusters_a), palette =␣
↪c("#2E9FDF", "#00AFBB", "#E7B800"), geom = "point", ellipse.type = "convex",␣
↪ ggtheme = theme_bw(),main ="Clustering hiérachique Ascendant : Group␣
↪Average")
```


area
perimeter
compactness
lengthOfKernel
widthOfKernel
ymmetryCoefficie
gthOfKernelGroo

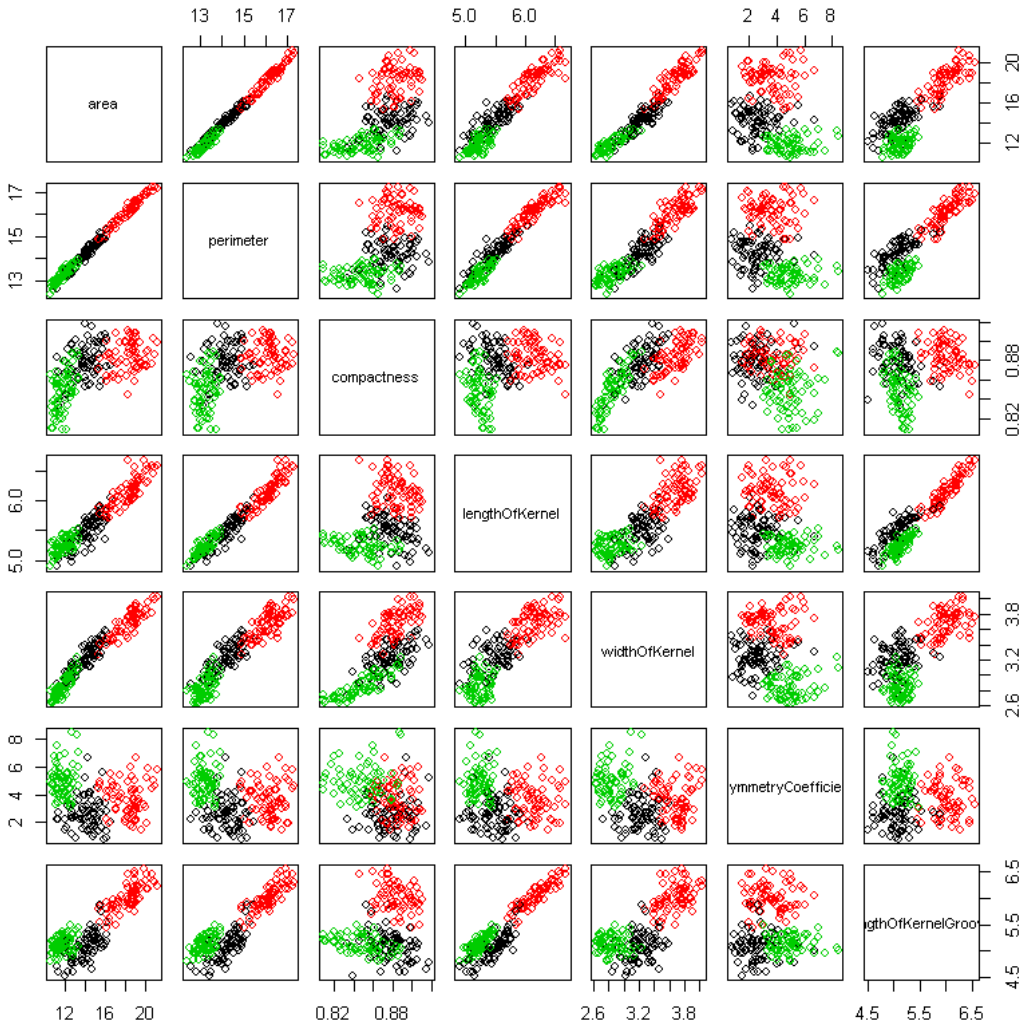


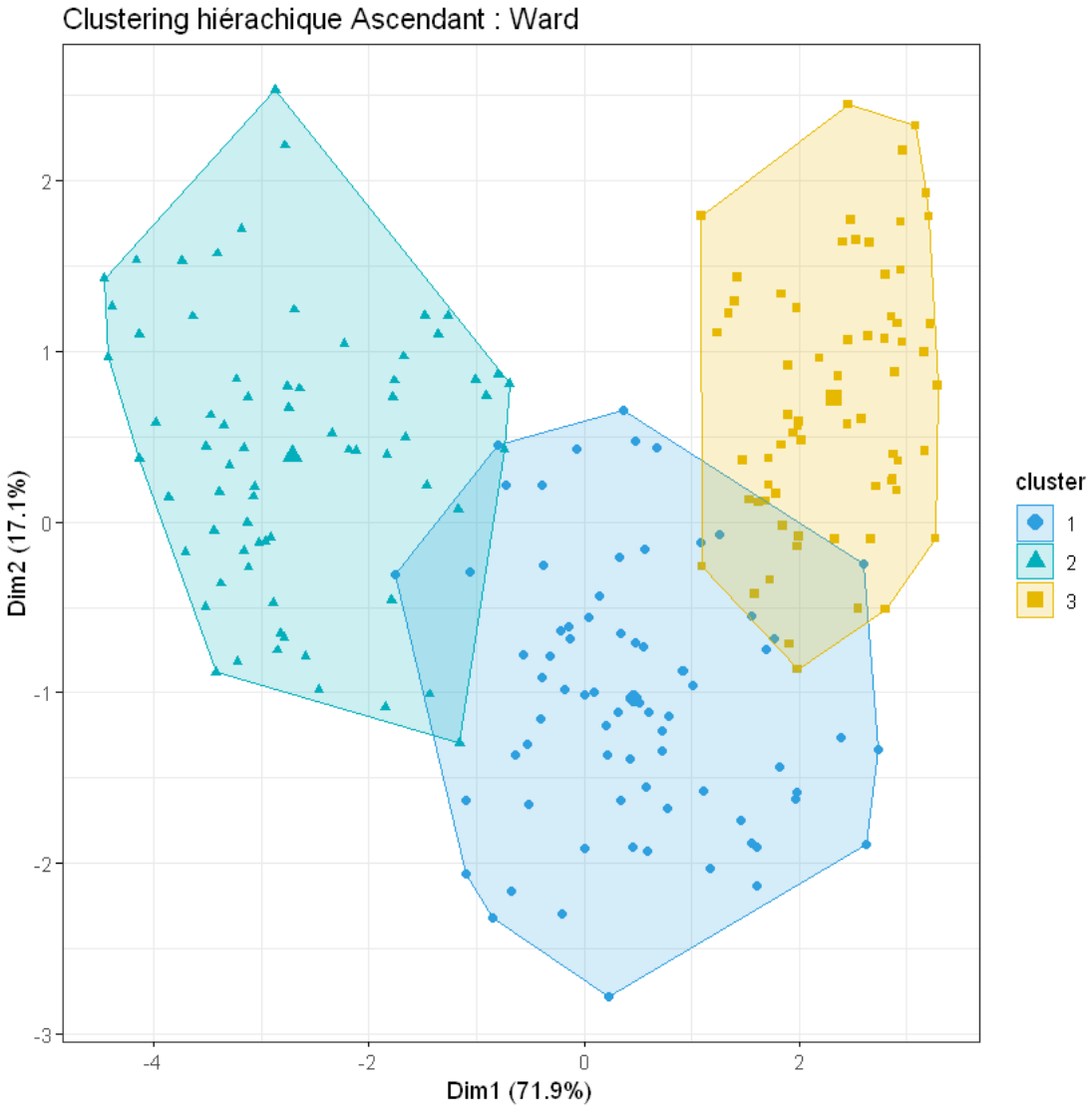
Méthode average : on considère que la distance entre deux clusters est la moyenne de toutes les distances entre les deux clusters.

[77]:
```
hc_w<-hclust(d,method='ward.D2')
plot(hc_w)
hc_clusters_w<-cutree(hc_w,k=3)
```



[78]:
```
plot(data_seeds,col=hc_clusters_w)
fviz_cluster(object = list(data=data_seeds,cluster=hc_clusters_w), palette =␣
↪c("#2E9FDF", "#00AFBB", "#E7B800"), geom = "point", ellipse.type = "convex",␣
↪ ggtheme = theme_bw(),main ="Clustering hiérachique Ascendant : Ward")
```


area
perimeter
compactness
lengthOfKernel
widthOfKernel
ymmetryCoefficie
gthOfKernelGroo



Méthode Ward : Cherche à minimiser l'inertie inter-classe due au regroupement de 2 clusters (gère mieux les valeurs extrèmes)

6 Comparaison

Le K-means semble avoir de meilleurs résultats pour notre problématique mais il est plux exigeant. En effet la classification hiérachique ne nécessite pas de déclarer au préalable le nombre de clusters. Cependant à chaque itérations il faut recalculer les distances de toutes les paires de points possibles entre les 2 clusters, donc plus couteux en compléxité.
Le CHA sera plus lent que le K-means avec de grandes bases de données , ce qui n'est pas le cas ici.

Au vu de la problématique , ou l'on connaissait le nombre de clusters à identifier il n'est pas

surprenant que le K-Means obtienne de meilleurs résultats.

En effet aucun algorithme ne peut prédire avec certitude ,les résultats sont donc à intrepréter avec précaution et recul.