

# Fouille de données massives - Projet final

CASANOVA MARROQUIN Stephanya

03/12/2021

## Données

Le projet a été basé sur une base de données décrivant **6224 individus américains** décrits par 15 variables :

- **Age** : âge
- **CSP** : catégorie socio-professionnelle
- **ScoreDemo** : un score démographique
- **Diplome** : le type de diplôme
- **ScoreDiplome** : un score construit en fonction du type de diplôme
- **StatutMarital** : le statut marital
- **Profession** : la profession
- **SituationFamiliale** : la situation familiale
- **Ethnie** : l'origine ethnique
- **Genre** : le genre
- **Economies** : le montant des économies
- **Dettes** : le montant des dettes
- **HeureSemaine** : le nombre d'heures travaillées par semaine
- **PaysOrigine** : le pays d'origine
- **Revenus** : montant des revenus (supérieur ou inférieur à 50k\$)

## Import des données dans notre environnement

Pour charger les données dans notre environnement, nous allons utiliser la commande :

```
data = read.table("adult_sample.data", sep=",", header=TRUE, strip.white=TRUE, na.strings="?", stringsAsFactors = TRUE)
```

Dans cette commande nous pouvons observer les paramètres **sep** : référence au séparateur, **strip.white** : supprime les espaces en blanc après d'un séparateur, **na.strings**: substitution d'un string pour NA et **stringsAsFactors**.

## Travail effectué

### 1. Identification des valeurs nulls et imputation des valeurs

Après d'avoir importé les données, nous allons identifier les valeurs nulls et nous allons leur imputer une valeur par défaut. Initialement, nous identifions les variables catégorielles et quantitatives :

```
ind_categ_feature = c(2,4,6,7,8,9,10,14,15)
data_categ = data[, ind_categ_feature]
data_continuous = data[, -ind_categ_feature]
```

Avec le commande **summary()** nous allons regarder les détails des données :

```
summary(data_categ)
```

```
##              CSP              Diplome              StatutMarital
## Private      :4301  HS-grad      :2003  Divorced      : 850
## Self-emp-not-inc: 489  Some-college:1418  Married-AF-spouse : 5
## Local-gov     : 400  Bachelors   :1020  Married-civ-spouse :2848
## State-gov     : 242  Masters     : 324  Married-spouse-absent: 86
## Self-emp-inc   : 229  Assoc-voc   : 271  Never-married      :2051
## (Other)       : 175  11th        : 231  Separated          : 194
## NA's          : 388  (Other)     : 957  Widowed            : 190
##              Profession              SituationFamiliare              Ethnie
## Craft-repair   : 783  Husband     :2499  Amer-Indian-Eskimo: 63
## Exec-managerial: 766  Not-in-family :1603  Asian-Pac-Islander: 189
## Prof-specialty : 766  Other-relative: 188  Black              : 617
## Sales          : 735  Own-child    : 972  Other              : 45
## Adm-clerical   : 714  Unmarried    : 646  White              :5310
## (Other)       :2071  Wife         : 316
## NA's          : 389
##              Genre              PaysOrigine              Revenus
## Female:2025  United-States:5561  <=50K:4715
## Male :4199  Mexico      : 126  >50K :1509
##              Canada      : 34
##              Germany     : 28
##              Philippines  : 28
##              (Other)     : 326
##              NA's        : 121
```

Dans cette information, nous pouvons observer que les variables **CSP**, **Profession** et **PaysOrigine** ont des valeurs **nulls**.

```
attWithNA<-subset(data_categ, select=c(CSP, Profession, PaysOrigine))
summary(attWithNA)
```

```
##              CSP              Profession              PaysOrigine
## Private      :4301  Craft-repair   : 783  United-States:5561
## Self-emp-not-inc: 489  Exec-managerial: 766  Mexico      : 126
## Local-gov     : 400  Prof-specialty : 766  Canada      : 34
## State-gov     : 242  Sales          : 735  Germany     : 28
## Self-emp-inc   : 229  Adm-clerical   : 714  Philippines : 28
## (Other)       : 175  (Other)        :2071  (Other)     : 326
## NA's          : 388  NA's           : 389  NA's        : 121
```

Ces valeurs nulls vont être remplacés par de valeurs aléatoires en fonction de la distribution de fréquences dans chaque attribut :

```

naCSP=length(data_categ$CSP[is.na(data_categ$CSP)])
data_categ$CSP[is.na(data_categ$CSP)]=sample(levels(data_categ$CSP),naCSP,
prob=table(data_categ$CSP),replace=TRUE)
data_categ$CSP=as.factor(data_categ$CSP)

nbNA=length(data_categ$Profession[is.na(data_categ$Profession)])
data_categ$Profession[is.na(data_categ$Profession)]=sample(levels(data_categ$Profession),nbNA,
prob=table(data_categ$Profession),replace=TRUE)
data_categ$Profession=as.factor(data_categ$Profession)

nbNA=length(data_categ$PaysOrigine[is.na(data_categ$PaysOrigine)])
data_categ$PaysOrigine[is.na(data_categ$PaysOrigine)]=sample(levels(data_categ$PaysOrigine),nbNA,
prob=table(data_categ$PaysOrigine),replace=TRUE)
data_categ$PaysOrigine=as.factor(data_categ$PaysOrigine)

```

De nouveau **Summary** pour valider que effectivement les donnés ont été remplacés :

```

attWithNA<-subset(data_categ, select=c(CSP, Profession, PaysOrigine))
summary(attWithNA)

```

```

##              CSP              Profession              PaysOrigine
## Private      :4597  Craft-repair      : 836  United-States:5675
## Self-emp-not-inc: 518  Exec-managerial: 819  Mexico          : 129
## Local-gov     : 416  Prof-specialty   : 814  Canada           :  34
## State-gov     : 259  Sales             : 780  Germany           :  28
## Self-emp-inc   : 247  Adm-clerical     : 750  Philippines       :  28
## Federal-gov   : 184  Other-service    : 664  Puerto-Rico       :  22
## (Other)       :   3  (Other)           :1561  (Other)           : 308

```

Maintenant, nous allons observer les détails des attributs quantitatifs :

```
summary(data_continuous)
```

```

##      Age      ScoreDemo      ScoreDiplome      Economies
## Min.   :17.00  Min.    : 19302  Min.    : 1.00  Min.    :    0
## 1st Qu.:28.00  1st Qu.: 118253  1st Qu.: 9.00  1st Qu.:    0
## Median :37.00  Median : 179312  Median :10.00  Median :    0
## Mean   :38.62  Mean    : 191041  Mean    :10.06  Mean    : 1028
## 3rd Qu.:48.00  3rd Qu.: 241204  3rd Qu.:12.00  3rd Qu.:    0
## Max.   :90.00  Max.    :1184622  Max.    :16.00  Max.    :99999
##      Dettes      HeureSemaine
## Min.    :  0.00  Min.    : 1.00
## 1st Qu.:  0.00  1st Qu.:40.00
## Median :  0.00  Median :40.00
## Mean    : 89.81  Mean    :40.48
## 3rd Qu.:  0.00  3rd Qu.:45.00
## Max.    :3004.00  Max.    :99.00

```

Dans les résultat, nous ne trouvons pas des valeurs nulls.

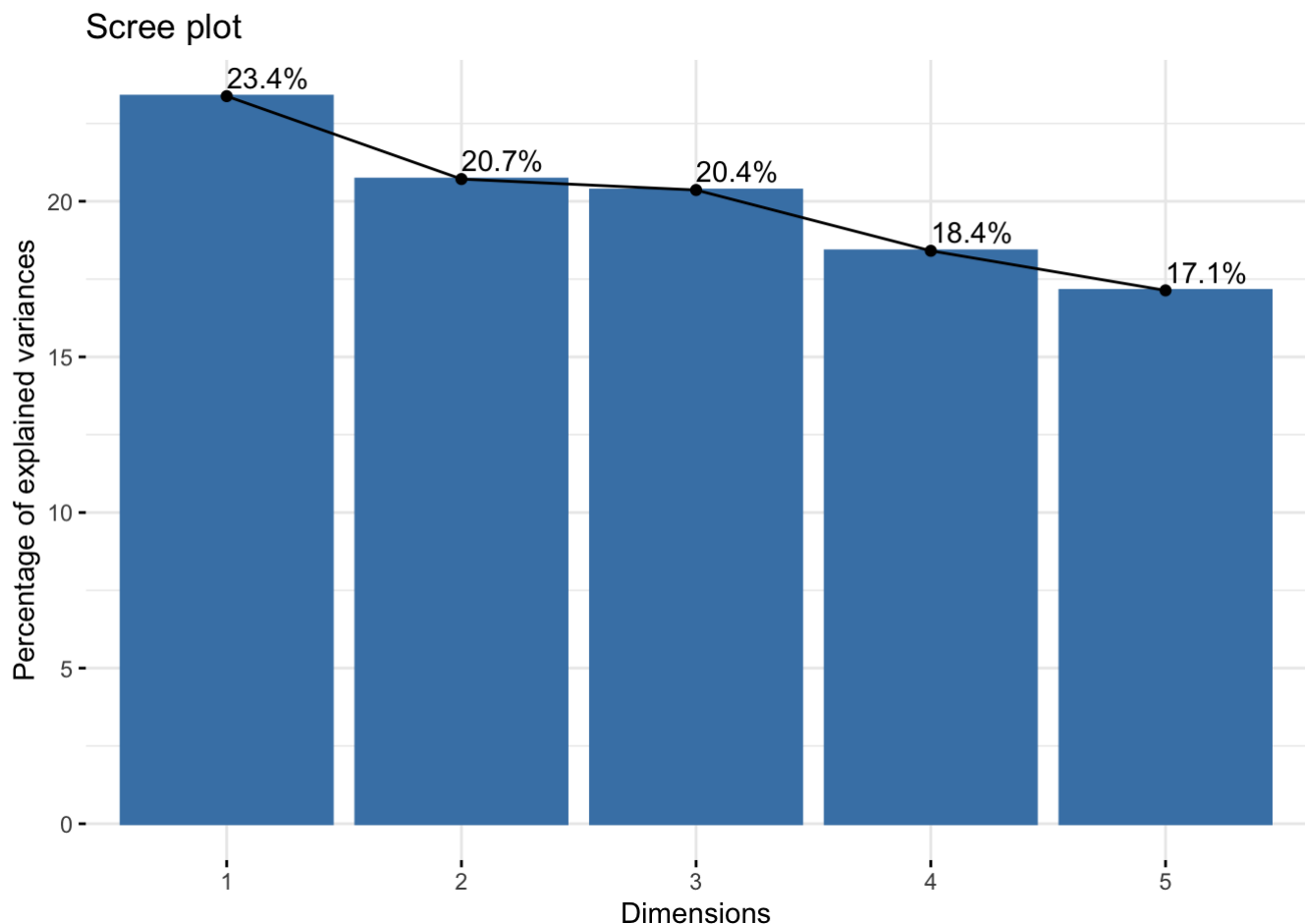
Nous allons supprimer la variable redondante ScoreDiplome puisque est le recodage de la variable Diplome :

```
data_continuous$ScoreDiplome = NULL
```

## 2. Analyse Factorielle

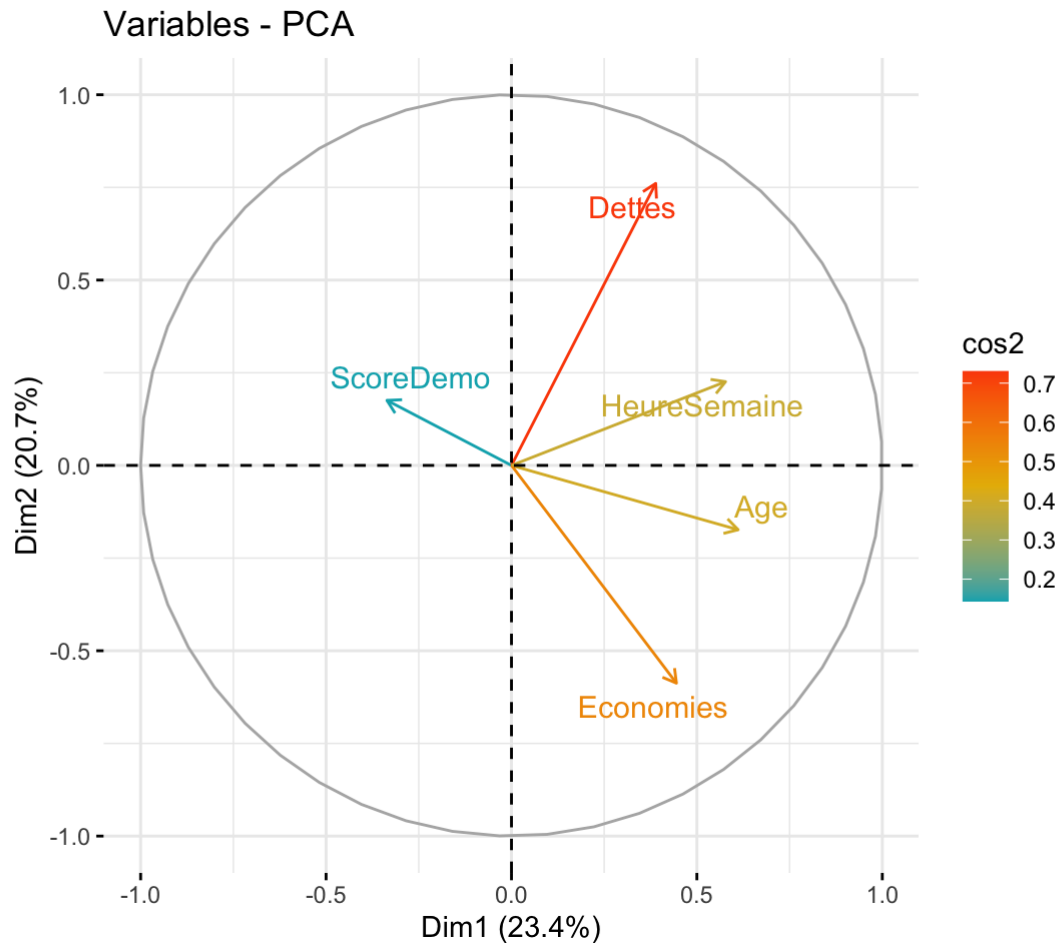
Maintenant avec la data traitée, nous allons faire une analyse en composant principaux sur les variables continues :

```
res.pca <- PCA(data_continuous, graph = FALSE)
fviz_eig(res.pca, addlabels = TRUE)
```



Du graphique ci-dessus, nous observons que le 82.9% (> 72%) des informations (variances) contenues dans les données sont conservées par les quatre premières composantes principales.

```
# Colorer en fonction du cos2: qualité de représentation
fviz_pca_var(res.pca, col.var = "cos2",
              gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
              repel = TRUE # Évite le chevauchement de texte
)
```



Le graphique de corrélation des variables ci-dessus montre les relations entre toutes les variables. Il peut être interprété comme suit:

- \* Les variables : **Dettes et HeureSemaine** et **Age et Economies** sont positivement corrélées.
- \* Les variables : **ScoreDemo et (Age, Economies)** sont négativement corrélées.
- \* Les variables **ScoreDemo, Age et HeureSemaine** ne sont pas bien représentés pour l'ACP ( $\cos^2$ ) à différence de **Dettes et Economies**

Maintenant, nous allons regarder la contribution de chaque variable aux axes principaux :

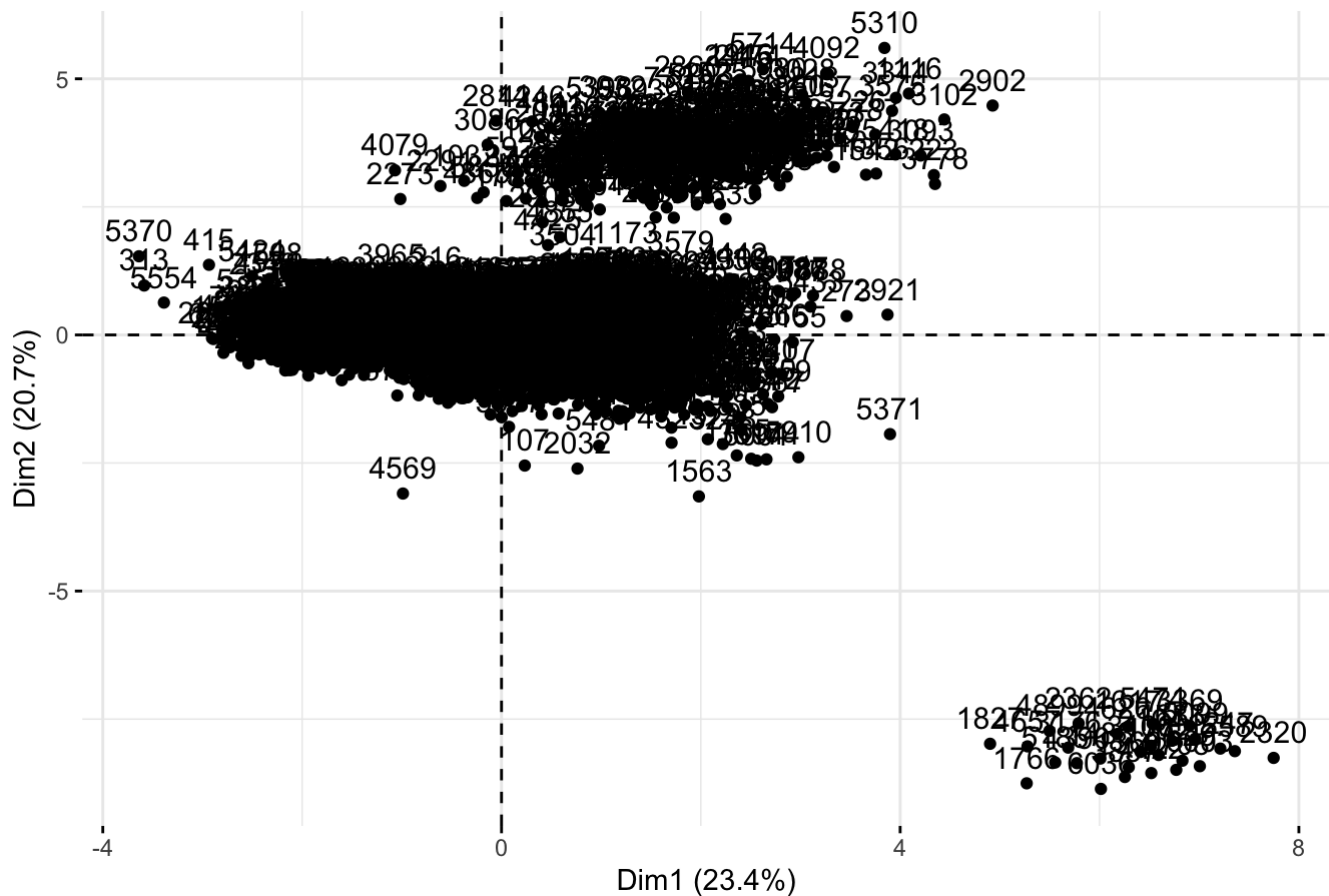
```
res.pca$var$contrib
```

##	Dim.1	Dim.2	Dim.3	Dim.4	Dim.5
## Age	31.965557	2.908096	7.80407230	40.28041948	17.041856
## ScoreDemo	9.649666	2.974066	56.24750717	26.77278503	4.355976
## Economies	16.927830	33.315762	19.02590595	0.09252983	30.637973
## Dettes	12.944193	55.874475	0.04812468	4.30854503	26.824662
## HeureSemaine	28.512754	4.927602	16.87438990	28.54572063	21.139533

Maintenant, pour les individus :

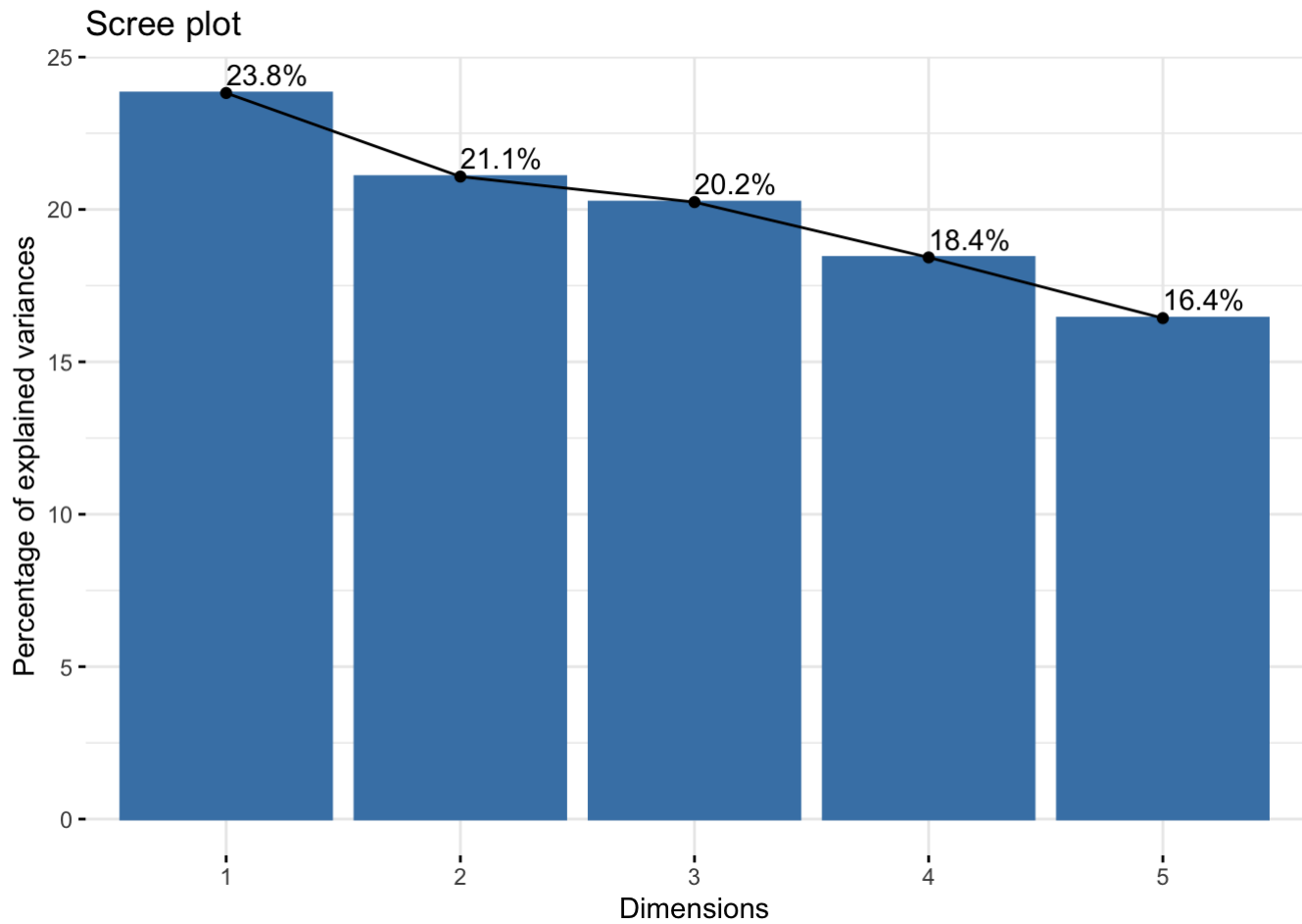
```
fviz_pca_ind(res.pca)
```

## Individuals - PCA



Le graphique ci-dessus permet d'observer le comportement de chaque individu en considérant les variables : **age**, **economies**, **heureSemaine**, **dettes** et **scoreDemo**. Nous pouvons observer alors les gens ayant le plus de dettes et les plus d'heures travaillées en haut et les gens qui ont plus des économies et d'âge en bas. Le groupe en bas à droite représente un groupe avec des économies et des revenus élevés. Ces individus vont être mis de côté pour l'analyse.

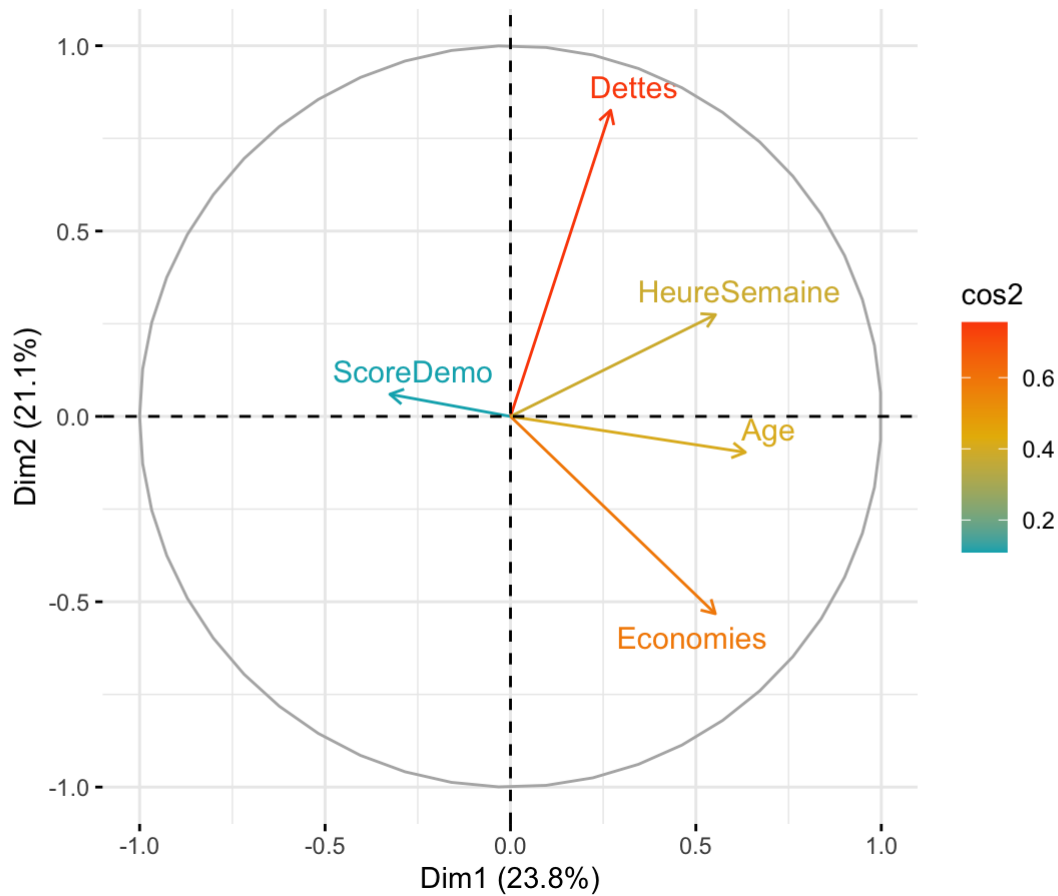
```
data_categ$Economies = data_continuous$Economies
data_continuous<-filter(data_continuous, Economies != 99999)
data_categ<-filter(data_categ, Economies != 99999)
data_categ$Economies = NULL
res.pca <- PCA(data_continuous, graph = FALSE)
fviz_eig(res.pca, addlabels = TRUE)
```



Après d'avoir filtrés les individus avec **Economies** de 99999 :

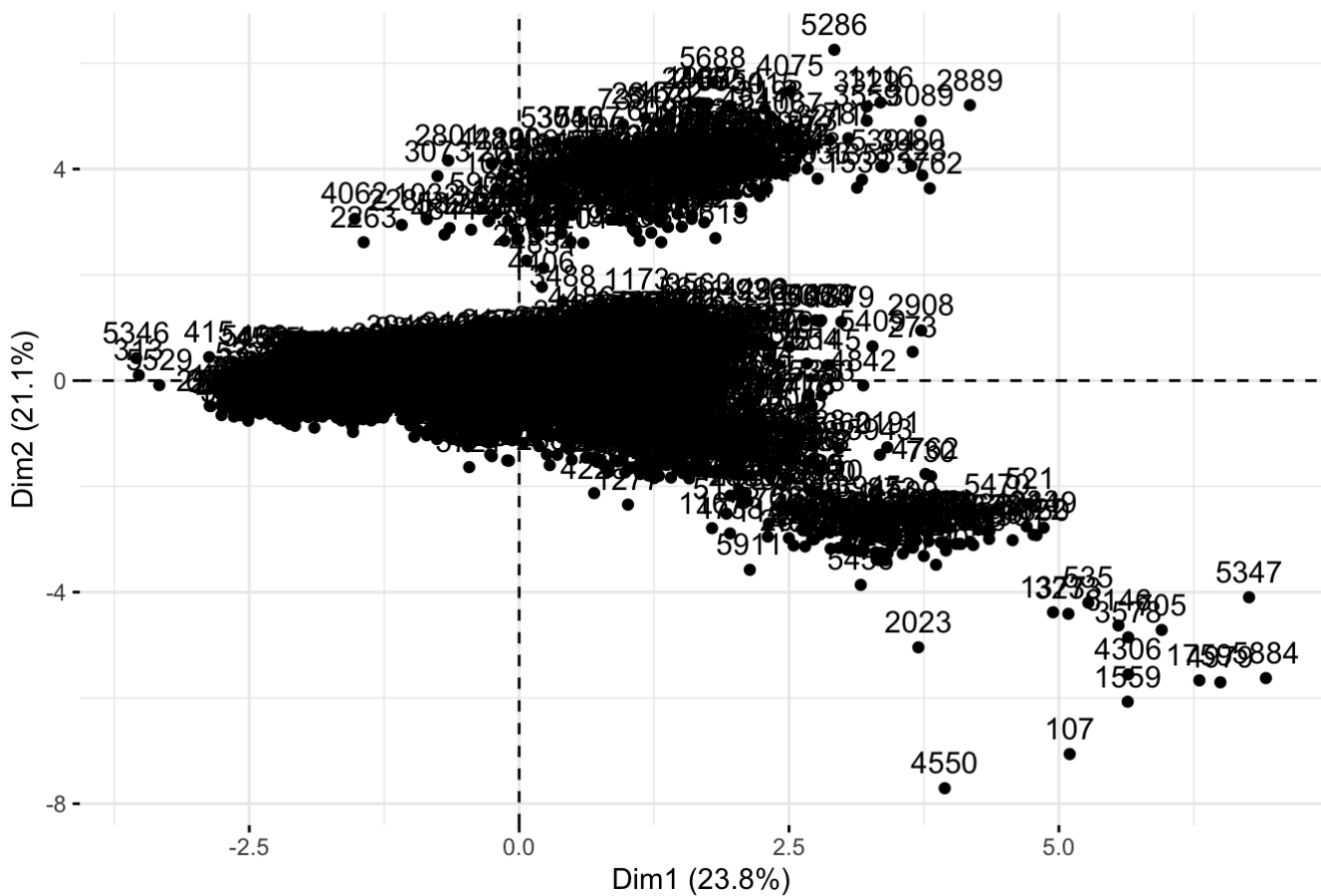
```
fviz_pca_var(res.pca, col.var = "cos2",  
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),  
             repel = TRUE # Évite le chevauchement de texte  
             )
```

## Variables - PCA



```
fviz_pca_ind(res.pca)
```

## Individuals - PCA



## 3. Clustering

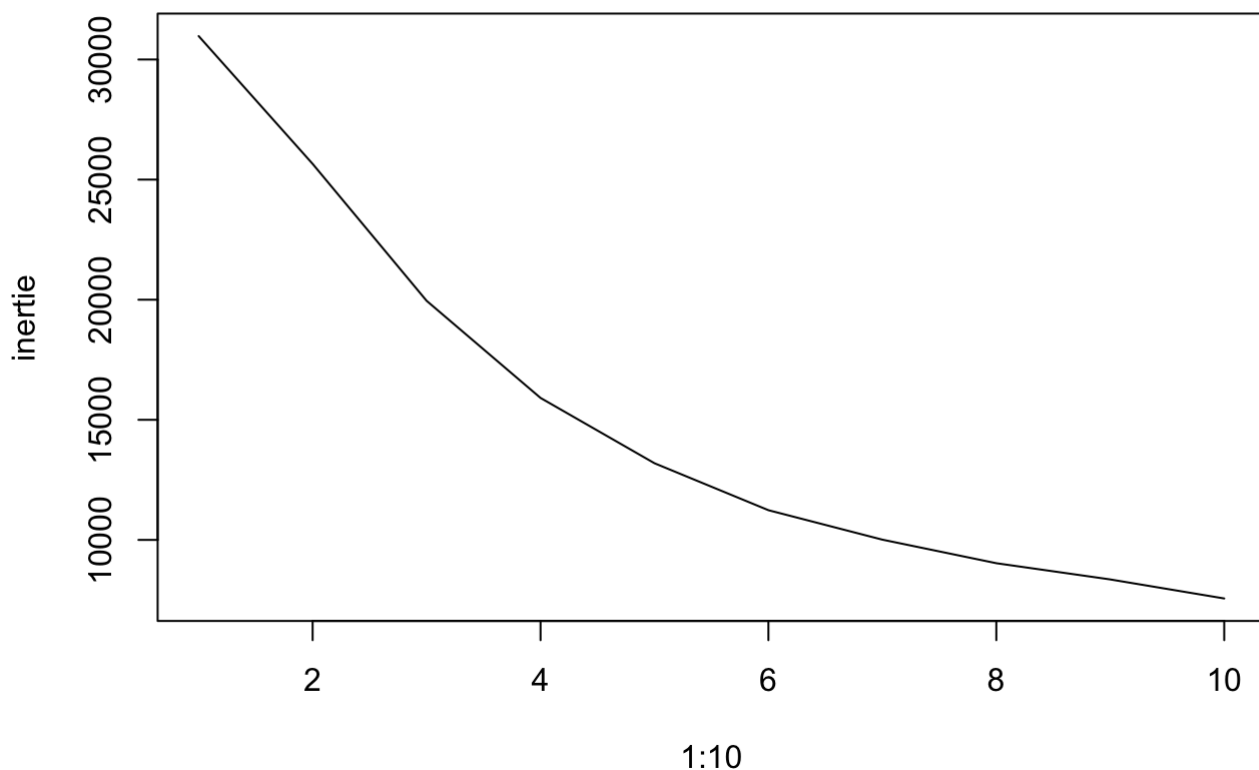


Comme les unités de mesure sont très différentes, nous allons normaliser les données :

```
data_continuous2=scale(data_continuous)
```

Maintenant, nous allons utiliser l'inertie intra-classe pour choisir un nombre de clusters adéquat :

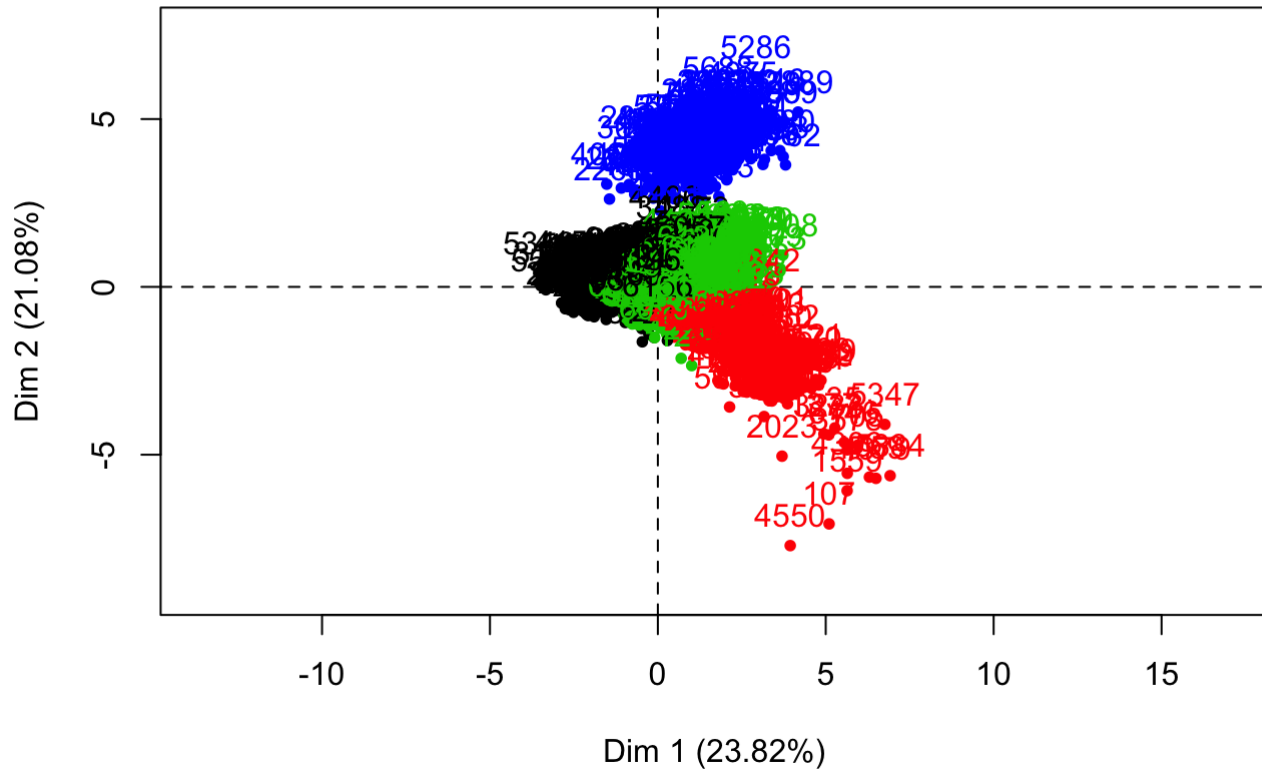
```
inertie=c()  
for (k in 1:10){  
  res=kmeans(data_continuous2,centers=k,nstart = 10)  
  inertie[k]=res$tot.withinss  
}  
plot(1:10,inertie,type='l')
```



Selon le graphique, nous allons choisir 4 clusters.

```
res_ka=kmeans(data_continuous2,centers=4,nstart = 10)  
res.pca <- PCA(data_continuous2, graph = FALSE)  
plot(res.pca,choix = "ind",col.ind = res_ka$cluster,graph.type = "classic")
```

## PCA graph of individuals

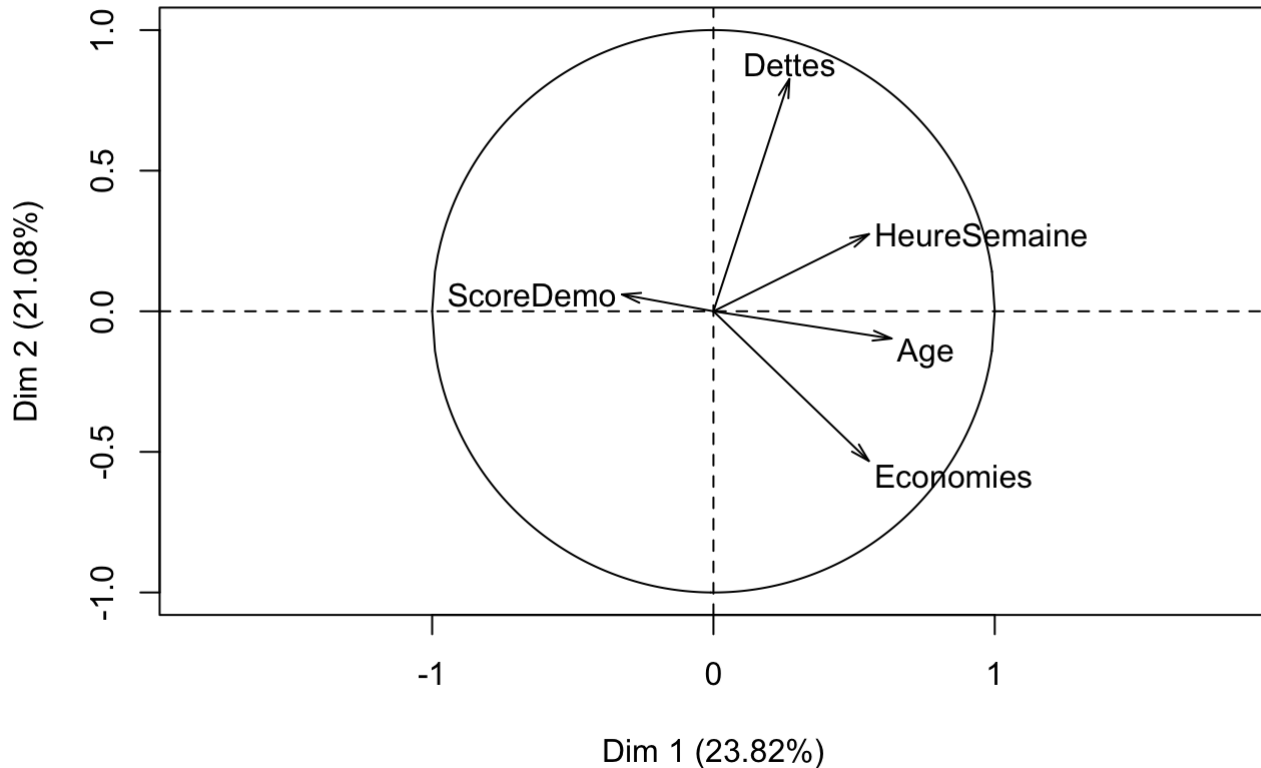


Ces 4 clusters pourraient correspondre à :

- \* **Cluster axe 1 centre** = Les gens qui travaillent le plus
- \* **Cluster axe 1 en haut** = Les gens avec plus de dettes
- \* **Cluster axe 2 centre** = Les gens avec le scoreDemo plus haut
- \* **Cluster axe 4** = Les gens avec plus des economies et agés

```
plot(res.pca,choix = "var",graph.type = 'classic')
```

## PCA graph of variables



## 4 Prediction

### 4.1 Ensemble d'entraînement et de testing

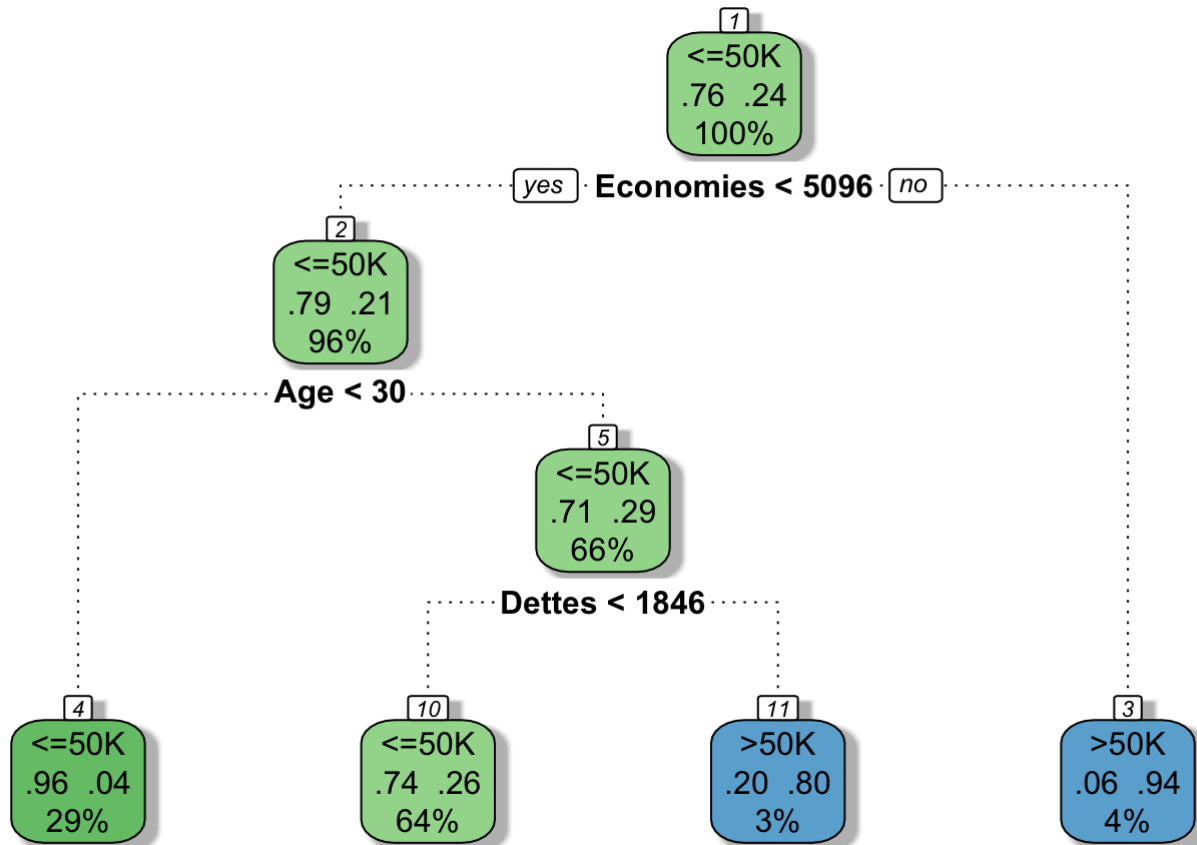
Pour la base de test, nous allons prendre 1000 individus au hasard :

```
set.seed(1)
ind_test=sample(1:nrow(data_continuous),1000)
dco_test=data_continuous[ind_test,]
dco_app=data_continuous[-ind_test,]
dca_test=data_categ[ind_test,]
dca_app=data_categ[-ind_test,]
```

### 4.2 Arbre de clasification

Pour la classification, nous allons utiliser un arbre binaire sur les données quantitatives. Il faut alors utiliser un dataframe et rapatrier la variable à prédire **Revenus** dans le même dataframe que les variables quantitatives :

```
dfco_app=data.frame(dco_app,Revenus=dca_app$Revenus)
mod2=rpart(Revenus~.,data=dfco_app)
p=predict(mod2,newdata = dco_test,type='class')
fancyRpartPlot(mod2,caption=NULL)
```



Pour évaluer les performances de l'arbre, nous allons observer le taux de bon classement et la matrice de confusion :

```
mean(p==dca_test$Revenus)
```

```
## [1] 0.831
```

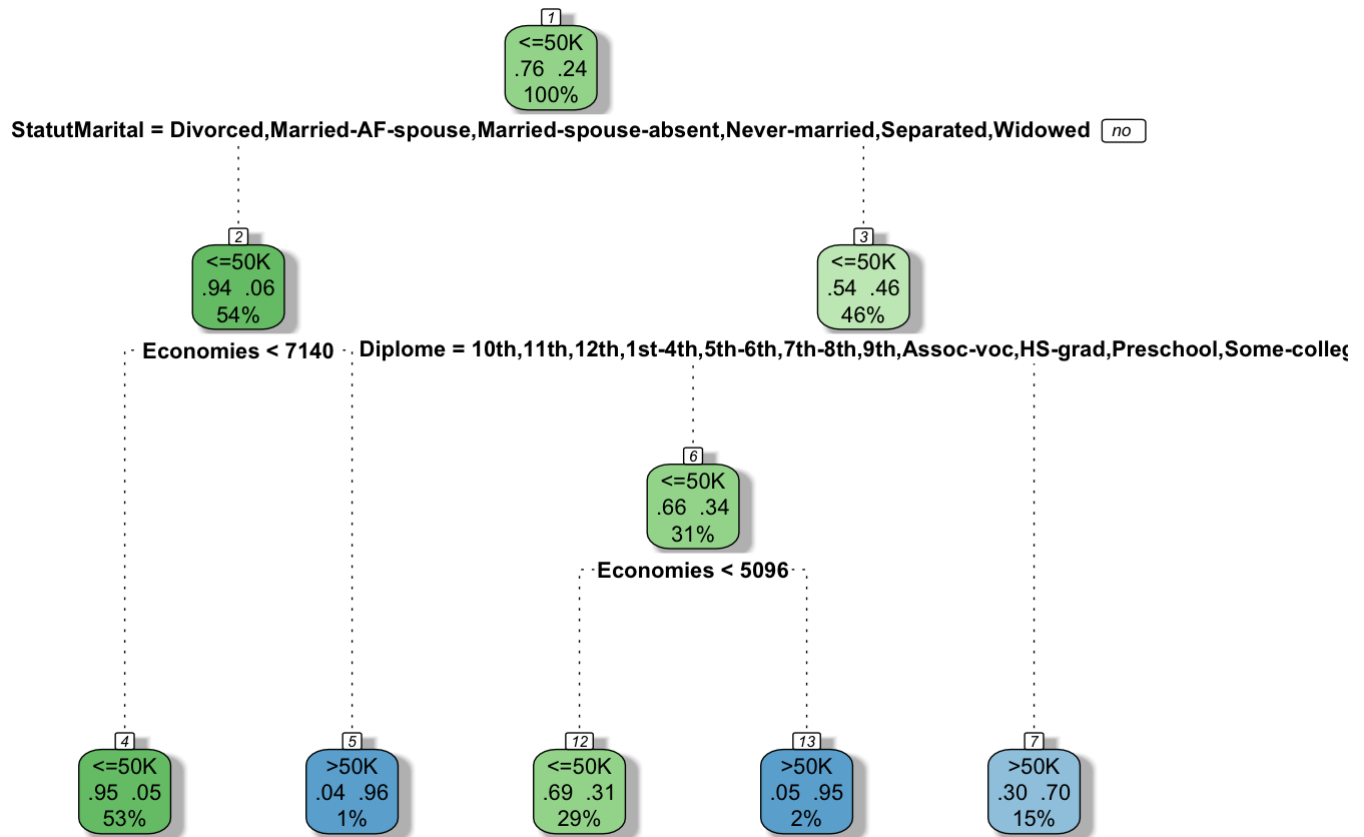
```
table(p, dca_test$Revenus)
```

```
##
## p      <=50K >50K
## <=50K   770  164
## >50K     5   61
```

On constate que la qualité de la prédiction dépend beaucoup de la classe. En effet, sur les 934 individus qui ont revenus  $\leq 50K$ , le taux de prévisions correctes est de 82.44 % environ, alors que sur les 66 individus avec  $>50K$ , il n'est que de 92.42 % environ.

Maintenant, si nous voudrions considérer toutes les variables quantitatives et catégorielles :

```
app=cbind(dco_app,dca_app)
test=cbind(dco_test,dca_test)
mod3=rpart(Revenus~.,data=app)
p=predict(mod3,newdata = test,type='class')
fancyRpartPlot(mod3,caption=NULL)
```



Taux de bon classement et la matrice de confusion :

```
mean(p==test$Revenus)
```

```
## [1] 0.828
```

```
table(p, test$Revenus)
```

```
##
## p      ≤50K >50K
## ≤50K    725  122
## >50K     50  103
```

### 4.3 Forêt aléatoire

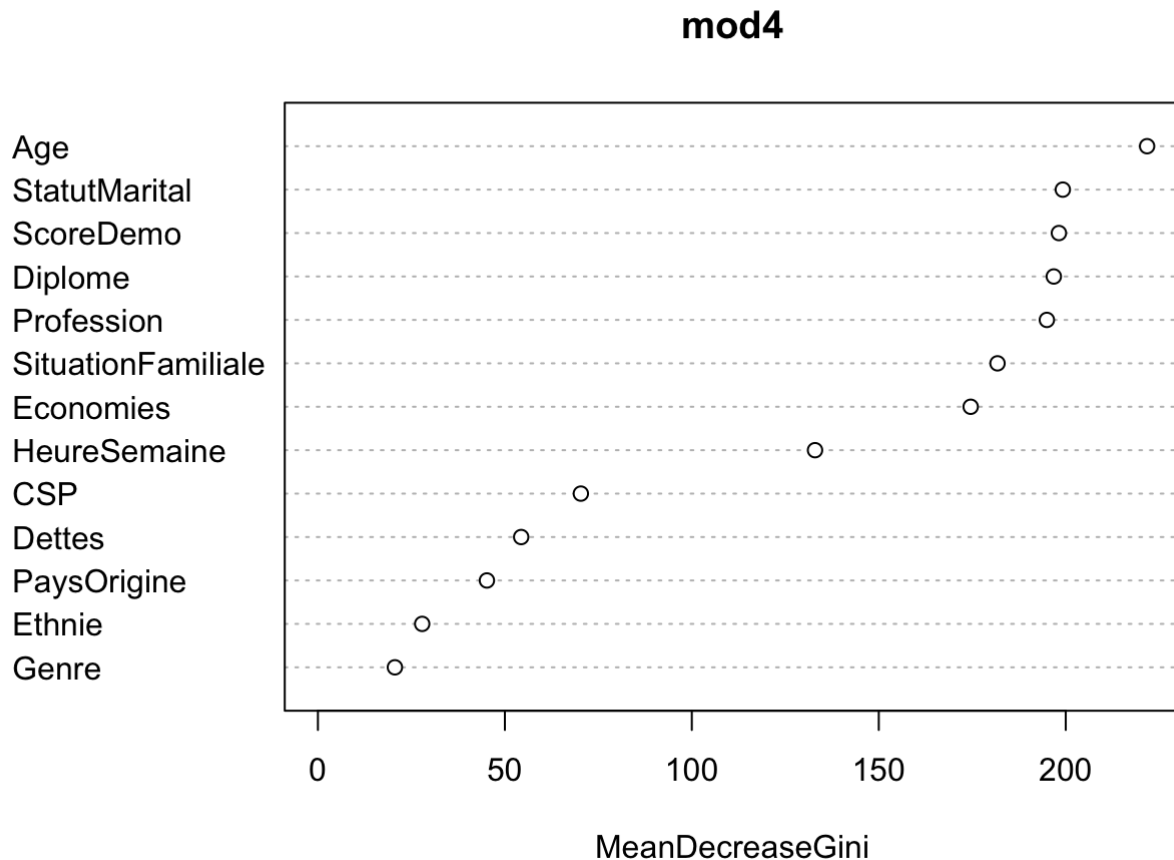
Réaliser une classification avec une forêt aléatoire. L'hyper-paramètre mtry sera réglé de façon intelligente, sans utiliser l'échantillon test. Évaluer la qualité de la prédiction sur les données tests et interpréter l'impact des différentes variables dans la prédiction.

```
set.seed(123)
mod4=randomForest(Revenus~.,data=app)
print(mod4)
```

```
##
## Call:
##  randomForest(formula = Revenus ~ ., data = app)
##                Type of random forest: classification
##                Number of trees: 500
## No. of variables tried at each split: 3
##
## OOB estimate of error rate: 15.43%
## Confusion matrix:
##      <=50K >50K class.error
## <=50K   3623   317  0.08045685
## >50K     485   771  0.38614650
```

Dans la matrice de confusion, nous pouvons observer une meilleure classification pour la classe Revenus <=50K (8.09%) en comparaison à 38.69% d'erreur pour la classe >50K. Maintenant, nous allons observer quels sont les variables qui figurent dans notre modèle discriminant qui génèrent cette classification :

```
varImpPlot(mod4)
```

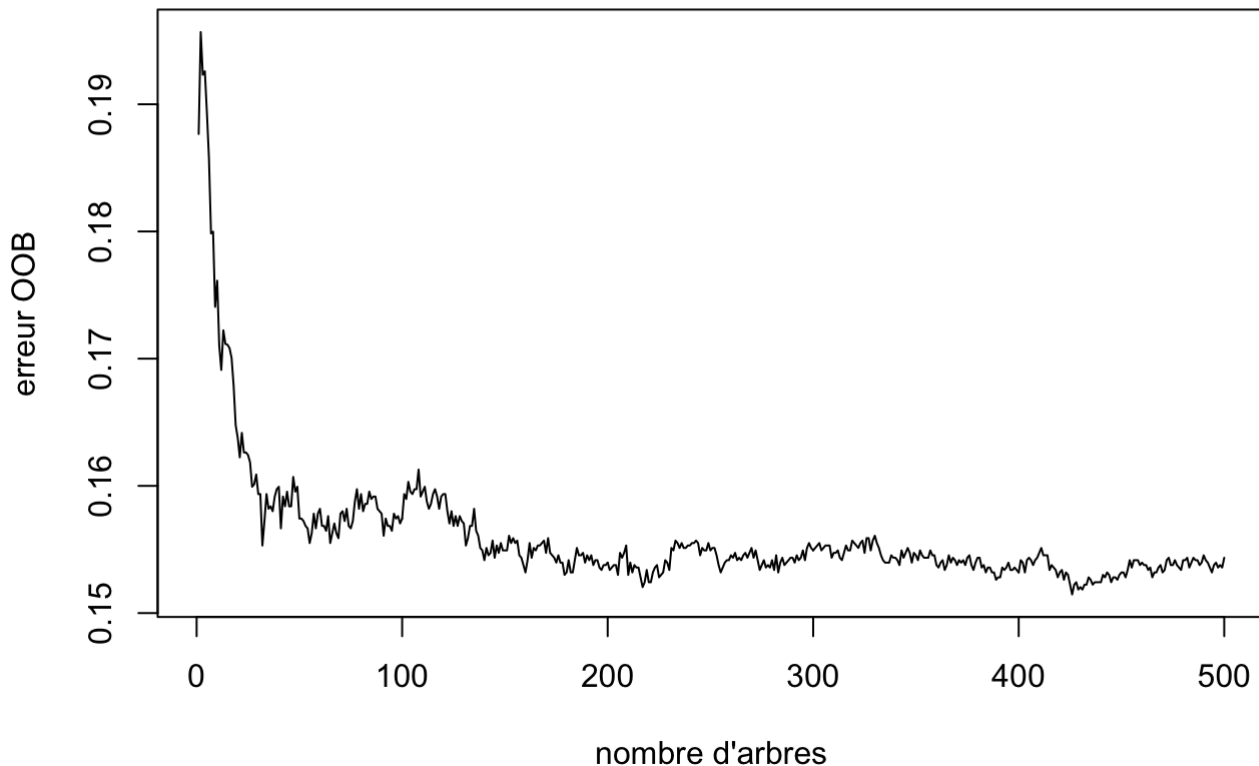


Dans le modèle que l'on a calculé, les 4 critères qui comptent le plus pour la classification sont **Age, Diplome, Profession et SituationFamiliale**.

Maintenant, si nous voulons améliorer la classification ou plus précisément minimiser l'**OOB** de 15.49% nous pourrions modifier 2 éléments : le nombre d'arbres construit par l'algorithme (**n<sub>tree</sub> = 500**) et le nombre de variables testées à chaque division (**m<sub>try</sub> = 3**).

Si nous voudrions choisir l'autre valeur de nTree, il faudrait choisir un nTree lorsque la valeur se stabilise au minimum :

```
plot(mod4$err.rate[, 1], type = "l", xlab = "nombre d'arbres", ylab = "erreur OOB")
```



Pour choisir le `mtry`, nous avons fait tourner Random Forest 10 fois avec dix valeurs de `mtry` différentes et nous avons choisi celle pour laquelle le `mtry` est minimal et se stabilise **`mtry = 2`**:

```
set.seed(123)
mod4 <- randomForest(Revenus~., data=app, mtry=2, importance=TRUE, ntree=500)
print(mod4)
```

```
##
## Call:
## randomForest(formula = Revenus ~ ., data = app, mtry = 2, importance = TRUE,
## ntree = 500)
##               Type of random forest: classification
##               Number of trees: 500
## No. of variables tried at each split: 2
##
## OOB estimate of error rate: 14.57%
## Confusion matrix:
##      <=50K >50K class.error
## <=50K  3667   273  0.06928934
## >50K    484   772  0.38535032
```

L'OOB obtenu **14.57%** est mieux qu'avant **15.24%** en 4.39%.

Maintenant, si nous faisons la prédiction avec le modèle amélioré :

```
set.seed(123)
p=predict(mod4,newdata = test,type='class')
```

Nous obtenons un taux de bon classement et la matrice de confusion :

```
mean(p==test$Revenus)
```

```
## [1] 0.842
```

```
table(p, test$Revenus)
```

```
##
## p      <=50K >50K
## <=50K    712   95
## >50K     63  130
```

## Références

Mars 14 2021

<http://www.sthda.com/french/wiki/fviz-pca-visualisation-de-l-analyse-en-composante-principale-logiciel-r-et-analyse-de-donn-es> (<http://www.sthda.com/french/wiki/fviz-pca-visualisation-de-l-analyse-en-composante-principale-logiciel-r-et-analyse-de-donn-es>)

[http://mehdikhaneboubi.free.fr/random\\_forest\\_r.html](http://mehdikhaneboubi.free.fr/random_forest_r.html) ([http://mehdikhaneboubi.free.fr/random\\_forest\\_r.html](http://mehdikhaneboubi.free.fr/random_forest_r.html))

<https://www.listendata.com/2014/11/random-forest-with-r.html> (<https://www.listendata.com/2014/11/random-forest-with-r.html>)