Projet

0. Introduction

0.1. Objectives

Our objective is too try understand some key components of what differs from private insured to the non insured in the United States.

0.2. Importation of our database

We used data from the 2024 Current Population Survey - Annual and Economic Supplement (CPS-ASEC), a large cross-sectional survey conducted annually by the U.S. Census Bureau. We keep the following variables :

- NOW_PRIV : a dummy to say if people are insured or not
- A_AGE : age
- PTOT_VAL : total income
- PEARN_VAL : earnings
- POTH_VAL : all income not from earnings
- PMED_VAL : expenditures in medical care
- POTC_VAL : expenditures in over the counter medications
- DSAB_VAL : expenditures in over the counter medications

```
bdd <- read.csv("bdd_cluster.csv",sep = ",")

bdd <- bdd[,3:11]

bdd$NOW_PRIV[bdd$NOW_PRIV == 2] <- 0
```

1. About the projet

1.1. Contribution

Pierre : data, CART, Bagging, Bootstrap

Rémi : Clustering, discriminant analysis, Random forest, Hybrid Hierarchical Clustering

Anas : global structure of the report (explaination of the database, context,...), comprehension of the Hybrid Hierarchical Clustering

Célia : PCA, comprehension of the Hybrid Hierarchical Clustering

1.2. Difficulties

The first difficulty we encountered was the size of our dataset, which contains 177,000 observations, too many for the methods we intended to use. Additionally, the hybridHclust package did not work. Fortunately, thanks to Mathis, we were able to run a Hybrid Hierarchical Clustering, but only with 100 observations, making the conclusions unreliable.

For the discriminant analysis, we faced an issue with linearity between variables, which we resolved by removing one of them.

2. Exploratory Analysis

2.1. NA's

```
anyNA(bdd)
```

```
[1] FALSE
```

2.2. Summary

```
library(dplyr)
```

```
Attaching package: 'dplyr'
```

```
The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```r
#For continous variables
summary(select(bdd,-NOW_PRIV))
```

```
     A_AGE          PTOTVAL           PEARNVAL          POTHVAL
 Min.   : 0.00   Min.   : -20043   Min.   : -17048   Min.   :  -9999
 1st Qu.:18.00   1st Qu.:      0   1st Qu.:      0   1st Qu.:      0
 Median :37.00   Median :  23263   Median :   3500   Median :      2
 Mean   :37.98   Mean   :  42944   Mean   :  33722   Mean   :   9222
 3rd Qu.:56.00   3rd Qu.:  57039   3rd Qu.:  50000   3rd Qu.:   6214
 Max.   :85.00   Max.   :2108379   Max.   :2099999   Max.   :1249830
    PMED_VAL           POTC_VAL          DSAB_VAL          FPERSONS
 Min.   :     0.0   Min.   :    0.0   Min.   :     0.0   Min.   : 1.000
 1st Qu.:     0.0   1st Qu.:    0.0   1st Qu.:     0.0   1st Qu.: 1.000
 Median :   100.0   Median :   90.0   Median :     0.0   Median : 3.000
 Mean   :   744.9   Mean   :  180.1   Mean   :   126.2   Mean   : 2.921
 3rd Qu.:   500.0   3rd Qu.:  200.0   3rd Qu.:     0.0   3rd Qu.: 4.000
 Max.   :200000.0   Max.   :75000.0   Max.   :100000.0   Max.   :16.000
```

```r
#For binary variables
table(bdd$NOW_PRIV)
```

```
    0      1
66063 111425
```

3. Methods :

Set the seed :

```r
set.seed(1)
```

Reduce our database :

We reduce the size of our database because it is too large and would consume to many RAM in our computers. We reduce the number of observations to 10000, which is quite large and will permit us to have no problem of RAM.

```r
bdd <- bdd[sample(1:nrow(bdd), 10000), ]
```

3.1. PCA

We can perform PCA on quantitative variables. Here, we do it on the variables age (A_AGE), total income (PTOT_VAL), earnings (PEARN_VAL), all income not from earnings (POTH_VAL), expenditures in medical care (PMED_VAL), expenditures in over the counter medications (POTC_VAL) and aid perceived for a disability (DSAB_VAL).

We can expect income variables to be correlated. PCA is relevant to make easier to interprete our dataset.

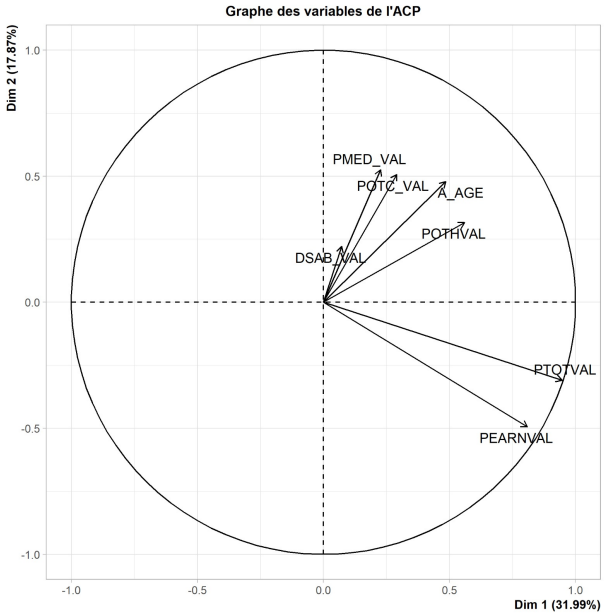
```r
library(Factoshiny)
#we scale our variables
bdd_quanti <- bdd[2:8]
bdd_scaled <- scale(bdd_quanti)
#we perform PCA with Factoshiny
resassur <- PCAshiny(bdd_scaled)
```

First, we look at the number of dimensions we want to keep. There are two criteria:

- we can choose to keep enough dimensions so that they explain at least 80% of the variance. Here it is suggested to keep 4 or 5 dimensions
- we can use the scree test criterion, which is to retain the k principal components so the difference between the k and k+1th eigenvalue is large and then for k+1 and k+2, etc.. is small. Here it is more suggested to keep 1 dimension. It is a bit small.

Let us have a look at what variables are well enough correlated to each component. We only consider the variables that contribute enough to a component. We must have contr $>$ 100/p = 14,3

- Total income and earnings are positively correlated to dim1, which makes sense since there is an inclusion relation between the two variables.



- Age, expenditures in over the counter medications, earnings and expenditures in medical care are positively correlated to dim2.
- All income not from earnings is positively correlated to dim 3 whereas expenditures in medical care, expenditures in over the counter medications and aid perceived for a disability are negatively correlated to dim3

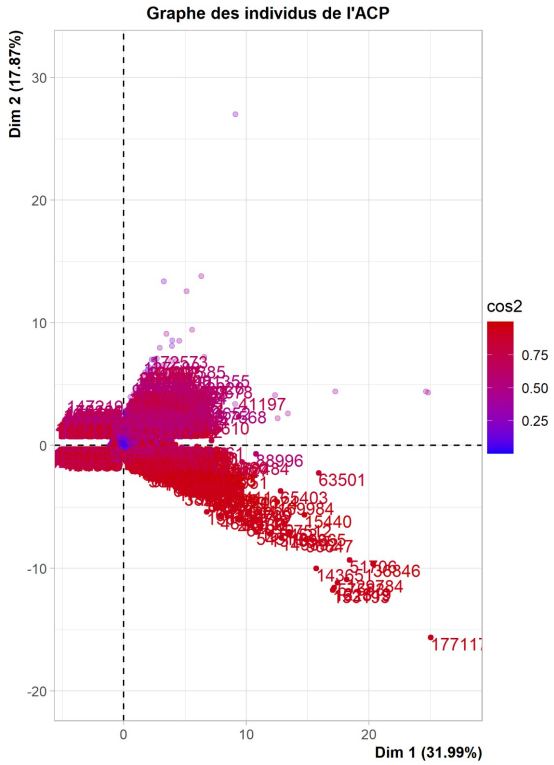
  Dim3 can be a measure of vulnerability : to be represented on the negative side of this axis mean that the person has not a lot of income despite aid for disability, and expenditures for medical care.
- We only have one variable contributing enough to dim4 : it is the aid perceived for a disability.
- Expenditures in medical care is positively correlated to dim5, contrary to expenditures in over the counter medications.

  Dim5 can be a measure of access to medical care : those who are on the positive side may be those who have access to proper medical care and do not only consume medications that everybody can buy, conversely to those on the negative side.

Interpretation:

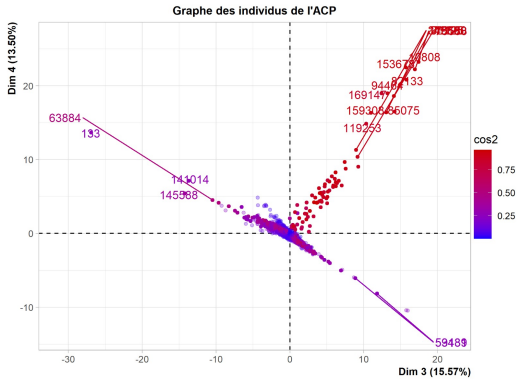
Here we have 10 000 individuals so we are not going to give interpretation for all of them, and even not for all of those who are well represented. But we can try to look at outliers or groups formed with respect to their representation on certain variables.

We can keep the outliers for later to see if they match our further analysis. The labels appear only for well enough represented individuals ($\cos^2 > 0.5$).



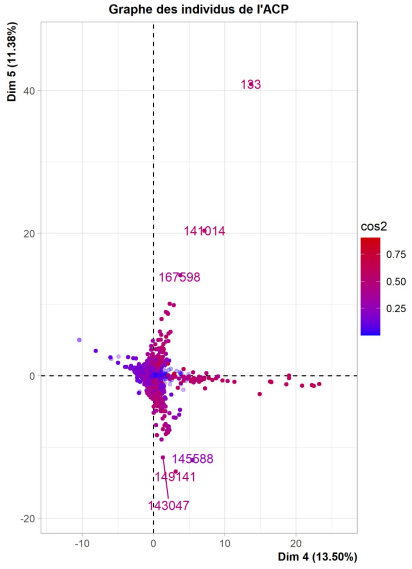
This gives a vision of the level of representation of the individuals : the higher the $\cos^2$ is, the better the representation is on dim1 and dim2.

We isolate less individuals for dim3 and dim4 ($\cos^2 > 0.25$) :



There is a small number of individuals well-represented. We can keep in mind that the individuals at the left are vulnerable. The individuals at the top are those who perceive the most aid for disability.

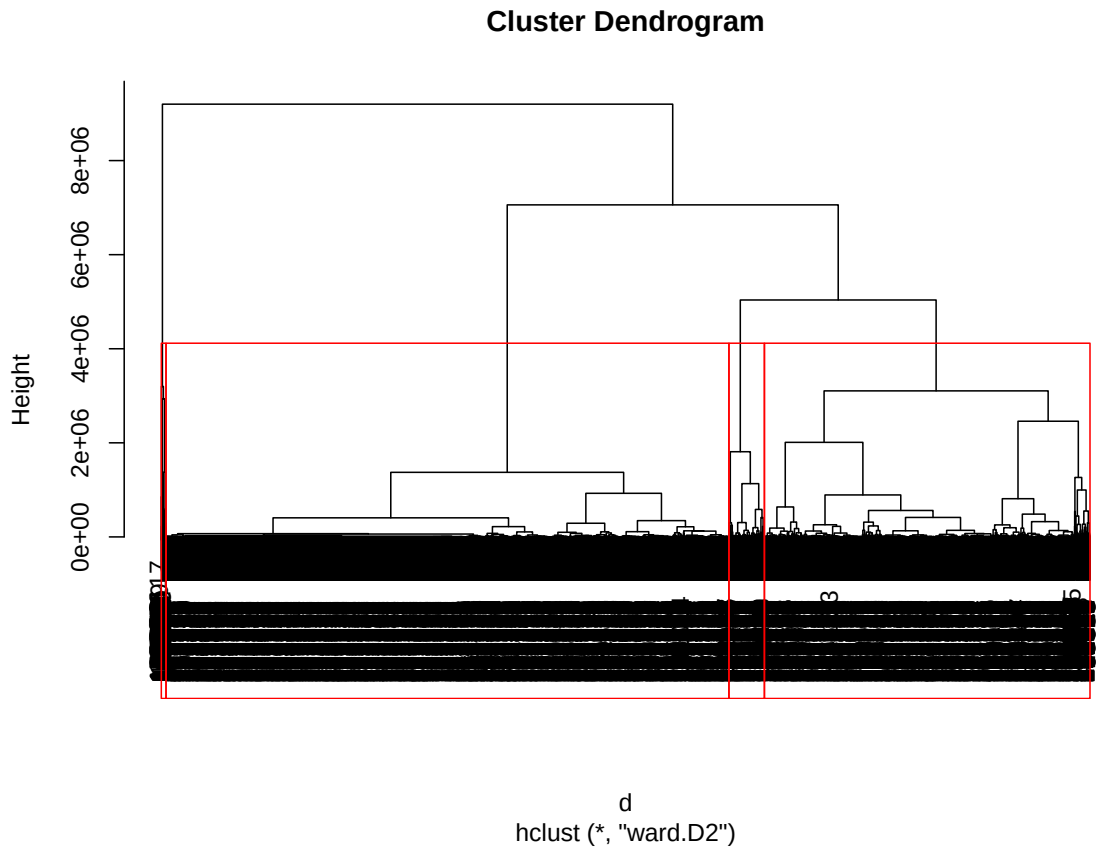
Now for dim5 ($\cos^2$>0.1):



Now only 6 individuals are represented here. It is a limit of having 5 dimensions : we have less and less individuals well represented of the axes.

3.2. Clustering

```
d <- dist(bdd, method = "euclidean")
fit <- hclust(d, method="ward.D2")

plot(fit)
groups <- cutree(fit, k=4)
rect.hclust(fit, k=4, border="red")
```




```
# We take the mean for each variables in each groups to interpret
cluster_means <- aggregate(. ~ groups, data = bdd, FUN = mean)
cluster_means
```

```
  groups  NOW_PRIV    A_AGE    PTOTVAL   PEARNVAL    POTHVAL  PMED_VAL POTC_VAL
1      1 0.4985976 32.01600   8653.249   4677.827   3975.422  508.6040 136.3821
```

```
2      2 0.7957787 46.98517  73481.340  56653.472  16827.868 1141.3137 250.8454
3      3 0.9790576 45.34031 206338.445 194995.374  11343.071 1307.6440 297.3037
4      4 0.9215686 45.47059 734829.765 610333.255 124496.510  938.3137 328.6275
   DSAB_VAL FPERSONS
1  75.86702 3.139086
2 240.33885 2.511979
3 118.58639 2.793194
4   0.00000 2.705882
```

The first group probably represents people on low incomes, with moderate medical expenses, and receiving low disability benefits. The second group corresponds to a middle class with a more stable income. However, medical and drug expenses are rising, suggesting a greater need for care. The third group represents individuals with very high incomes, but who also have higher medical expenses, which may indicate a more frequent need for care. The last group is the most affluent people, with great financial stability. They seem less concerned by medical expenses and social benefits, which could indicate better access to preventive care or better overall health.

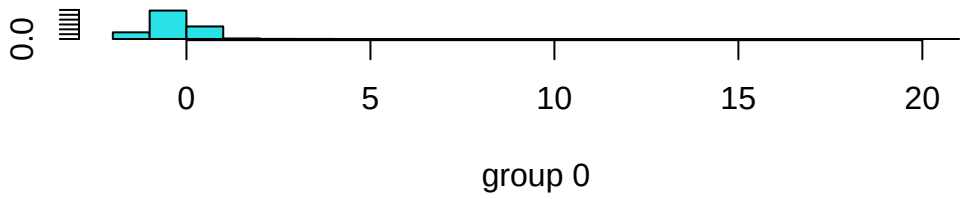
3.3. DA

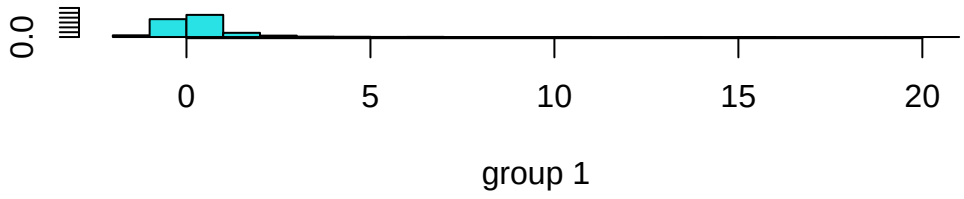
We choose to remove PTOTVAL for the DA because it's a linear combinationh of PEARNVAL and POTHVAL.

```
id <- sample(1:nrow(bdd), size = 5000)
bdd.train <- bdd[id, ]
bdd.test <- bdd[-id, ]

library(MASS)
```

```
Attaching package: 'MASS'

The following object is masked from 'package:dplyr':

    select
```

```
fit <- lda(NOW_PRIV ~  A_AGE + PEARNVAL + POTHVAL + PMED_VAL + POTC_VAL + DSAB_VAL + FPERSONS
plot(fit)
```





Predict with the NOW_PRIV for the test sample

```
pred <- predict(fit,bdd.test)
ct <- table(bdd.test$NOW_PRIV, pred$class)
print(ct)
```

```
   0    1
0  455 1426
1  204 2915
```

```
#frequence of good answer per type of NOW_PRIV
diag(prop.table(ct, 1))
```

```
        0         1
0.2418926 0.9345944
```

```
#total frequence of good answer
sum(diag(prop.table(ct)))
```

```
[1] 0.674
```

```
#66.34% of good answer
```

We can see that is accurate to predict private insuree with discriminant analysis but not when they are not insured. It could be useful to perform other methods as CART, Bagging and Random forest.

3.4. CART

Now, we want to predict if a person has a private insurance based on the age (A_AGE), total income (PTOT_VAL), earnings (PEARN_VAL), all income not from earnings (POTH_VAL), expenditures in medical care (PMED_VAL), expenditures in over the counter medications (POTC_VAL), aid perceive for a disability (DSAB_VAL) and number of people in the family (FPERSONS).

In order to do that we are going to use a classification tree.

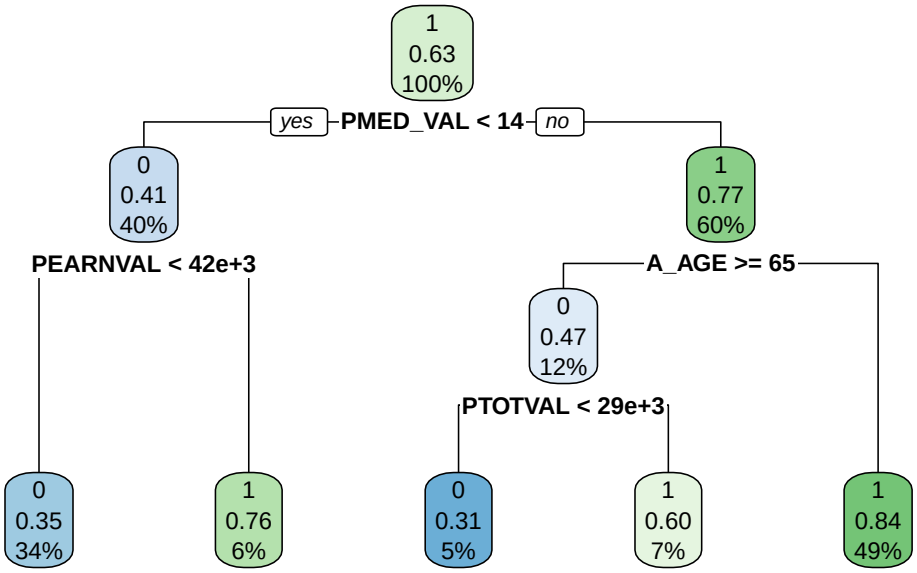
First we need to install following libraries

```
library(rpart)
library(rpart.plot)
```

Now, we shuffle our database in order to be sure that we don't create the training and test set on a sorted dataframe. This part will be use again for bagging and random forest too.

```
# Shuffle the databases
data_shuffled <- bdd[sample(nrow(bdd)), ]

# Creation of training and testing sets following the following proportion : 70% in the testi
train_indices <-floor(0.7*nrow(data_shuffled))
training_set <- data_shuffled[1:train_indices,]
testing_set <- data_shuffled[-c(1:train_indices),]
```

```
candidates_fit <- rpart(NOW_PRIV ~ .,
                        data = training_set, method = "class",
                        parms = list(split = "gini"), control = rpart.control(minsplit = 2))
rpart.plot(candidates_fit)
```




```
# confusion table and accuracy
candidates_predict <- predict(candidates_fit, testing_set, type = "vector")
conf_matrix <- table(candidates_predict, testing_set$NOW_PRIV)
#frequence of good answer per type of NOW_PRIV
diag(prop.table(conf_matrix, 1))
```

```
[1] 0.6513158 0.7976457
```

```
#total frequence of good answer
sum(diag(prop.table(conf_matrix)))
```

```
[1] 0.7383333
```

```
#73.73% of good answer
```

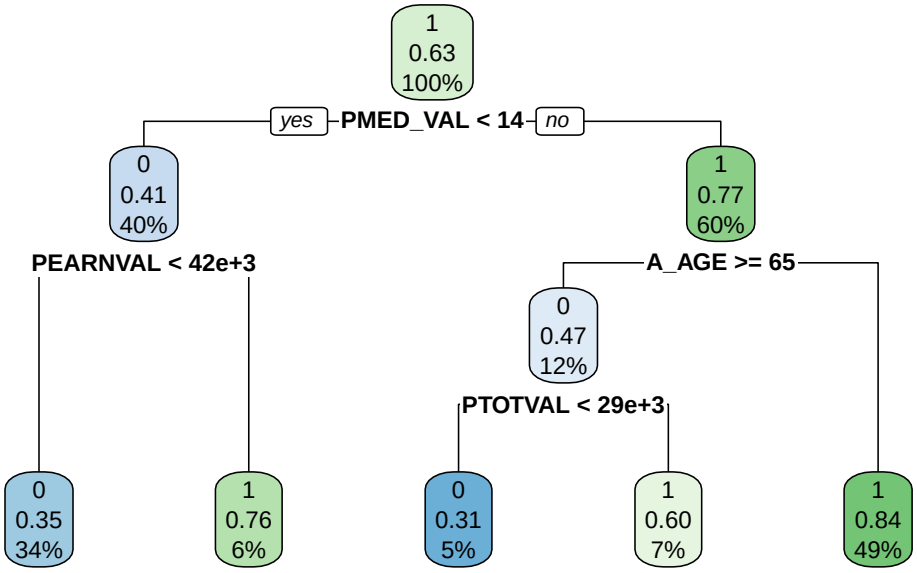
We see that we have a much better accuracy with the CART algorithm than with the DA, 73.73% against 66.34%. We specially see that to predict the non assurate people : 65.18% against 22.22%.

Comparaison of the main discriminant variable between the DA and the CART algorithm

```
fit$scaling
```

```
                  LD1
A_AGE    -1.691373e-02
PEARNVAL  1.400878e-05
POTHVAL   4.263775e-06
PMED_VAL  4.633156e-05
POTC_VAL  7.043226e-04
DSAB_VAL -1.489686e-05
FPERSONS  6.139489e-02
```

```
rpart.plot(candidates_fit)
```



The most disciminant variable for the DA are : POTC_VAL, PMED_VAL, FPERSONS and A_AGE

The main variable use in the CART algorithm are : PMED_VAL, PEARNVAL, A_AGE and PTOTVAL

We see that the variable used in the two method are not the same which explain the difference in the prediction result.

3.5. Bootstrap

We want to study repartition of income among the private insured.

```
# We just keep the private insured people
privateInsured <- bdd[bdd$NOW_PRIV==1,]
```

We first create our bootstrap function.

```
# We create a function (take it from Worksheet 4) that takes a given vector, generates a boot
bootstrap_mean <- function(X){
  boot_sample <- sample(X, size = length(X), replace = TRUE)
  return(mean(boot_sample))
}

cat("Average Income in the sample : ", mean(privateInsured$PTOTVAL))
```

```
Average Income in the sample :  55669.84
```

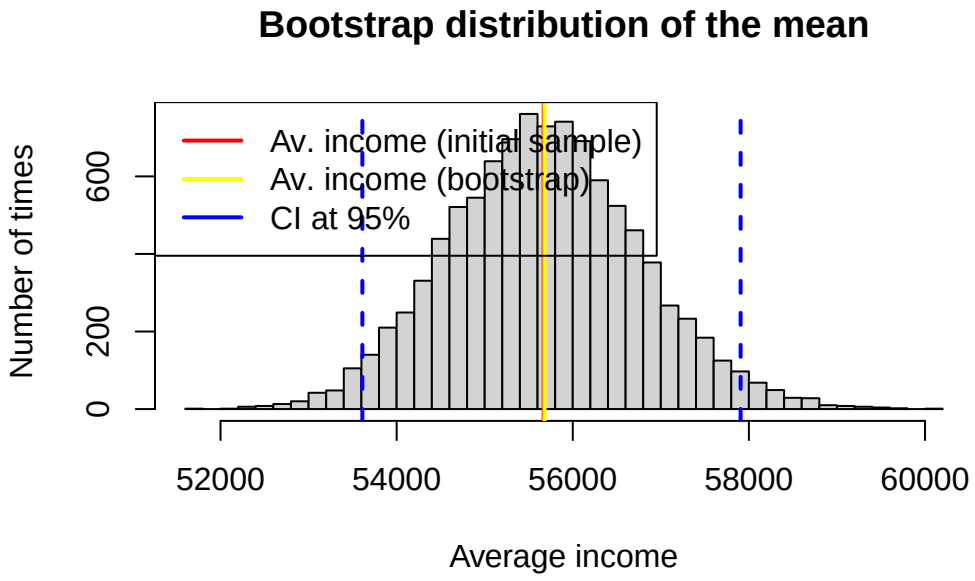
```
cat("Average Income by bootstraping (only once) : ",bootstrap_mean(privateInsured$PTOTVAL)) #
```

```
Average Income by bootstraping (only once) :  57968.36
```

Now, let's generate 10000 bootstrap sample to evaluate the distribution of income among insured people

```
#Generate B bootstrap samples
B <- 10000

bootstrap_means <- numeric(B)
for (i in 1:B) {
  bootstrap_means[i] <- bootstrap_mean(privateInsured$PTOTVAL)
}

# We compute the average income on the 10000 bootstraps
bootstrap_mean_estimate <- mean(bootstrap_means)
# We compute the 95% CI for the average income
ci <- quantile(bootstrap_means, c(0.025, 0.975))

hist(bootstrap_means, breaks = 30, main = "Bootstrap distribution of the mean", xlab = "Avera
```

```r
# We add the average income in the initial sample and in 10000 bootstraps
abline(v = mean(privateInsured$PTOTVAL) , col = "red", lwd = 2)
abline(v = bootstrap_mean_estimate, col = "yellow", lwd = 2)

# We add quantiles at 2,5% and 97,5%
abline(v = ci[1], col = "blue", lwd = 2, lty = 2)     # 2.5%
abline(v = ci[2], col = "blue", lwd = 2, lty = 2)     # 97.5%

legend("topleft", legend = c("Av. income (initial sample)","Av. income (bootstrap)", "CI at 9
```




```r
cat("2.5% quantile : ", ci[1])
```

```
2.5% quantile :  53610.83
```

```r
cat("97.5% quantile : ", ci[2])
```

```
97.5% quantile :  57906.29
```

We observe that the average income in our original sample is very close to the average of the 10,000 bootstrap samples. This suggests that our estimate is stable. Based on the bootstrap

distribution, we can construct a 95% confidence interval indicating that the average income for privately insured individuals lies between $53,583.47 and $57,852.57 per year. This is significantly lower than the average yearly income in the US, which is $65,470 (according to Sage.com), suggesting that a part of privately insured individuals may earn less than the national average. In fact, it makes sense because a lot of worker are insured by their firm and do not have to pay directly for their insurance.

3.6. Bagging

Now we want to go further on the classification tree part.

```
# We construct a function that takes a dataframe and make bootstrap on the classification tre
bootstrap_classification_tree <- function(x,d) {
  # bootstrap_sample with identifier d and dataset x
  bootstrap_sample <- x[d, ]

  # Trains the classification tree
  tree_model <- rpart(NOW_PRIV ~ ., data = bootstrap_sample,method = "class", parms = list(sp

  # Resulting prediction for the test set (encoding the prediction as 1 (for good) or 0(for b
  predictions <- predict(tree_model, testing_set, type = "class")

  # We return binary predictions
  return(as.numeric(predictions)-1)
}

indices <- sample(1:nrow(training_set), replace = TRUE)
training_set$NOW_PRIV <- as.factor(training_set$NOW_PRIV)
test_predictions <- bootstrap_classification_tree(training_set,indices) # we call the functio

#compute the accuracy
accuracy <- mean(test_predictions == testing_set$NOW_PRIV)

# print the accuracy
cat("Test Accuracy:", round(accuracy * 100, 2), "%")
```

```
Test Accuracy: 74.23 %
```

Now, we use a bagging approach in order to make a more robust classification tree than in the CART Part.

We need first to load the library boot

```r
library(boot)
```

```r
B <- 100 # We reduce the number of bootstrap to 100 (as in Worksheet 5 in order to avoid too
boot_results <- boot(data = training_set, statistic = bootstrap_classification_tree, R = B)

# get the predictions
boot_predictions <- boot_results$t

# final predictions with majority
final_predictions <- colMeans(boot_predictions) > 0.5 # It gives us True or False taking into

# compute the accuracy
accuracy_bagging <- mean(final_predictions == testing_set$NOW_PRIV)
predictions_error_bagging <- mean(final_predictions != testing_set$NOW_PRIV)
# print the accuracy
cat("Test Accuracy : ", round(accuracy_bagging * 100, 2), "%")
```

```
Test Accuracy :  73.93 %
```

```r
cat("Predicitons Error : ", round(predictions_error_bagging * 100, 2), "%")
```

```
Predicitons Error :  26.07 %
```

The accuracy increase slightly when bagging. However our accuracy in CART part where already higher than 70% meaning that our results where sufficiently stable to avoid bagging.

It is also possible to make bagging with the library ipred. Thanks to this library it will be possible to have access to the out of bag (OOB) error and compared bagging algorithm to random forest algorithm.

```r
library(ipred)

bagging_model <- bagging(NOW_PRIV ~ ., data = training_set, nbagg = B, coob = TRUE)
test_predictions_ipred <- predict(bagging_model, testing_set, type = "class")
predictions_error_ipred <- mean(test_predictions_ipred != testing_set$NOW_PRIV)
oob_error <- bagging_model$err

cat("Predicitons Error with ipred lib : ", round(predictions_error_ipred * 100, 2), "%")
```

```
Predicitons Error with ipred lib :  28.13 %
```

```
cat("Out of bag error : ", round(oob_error * 100, 2), "%")
```

```
Out of bag error :  26.66 %
```

It is a bit costly to use the ipred library since we have 0.7 points of percentage more of predictions error but thanks to it we know that OOB errors of the bagging method is 27.39%.

Since the results are not very satisfying it could be interesting to see what happens when we decorrelating our trees.

3.7. Random forest

```
library(randomForest)
```

```
randomForest 4.7-1.2
```

```
Type rfNews() to see new features/changes/bug fixes.
```

```

Attaching package: 'randomForest'
```

```
The following object is masked from 'package:dplyr':

    combine
```

```
library(caret)
```

```
Loading required package: ggplot2
```

```

Attaching package: 'ggplot2'
```

```
The following object is masked from 'package:randomForest':

    margin
```

```
Loading required package: lattice
```

```
Attaching package: 'lattice'

The following object is masked from 'package:boot':

    melanoma
```

```r
n <- nrow(bdd)

index <- sample(1:nrow(bdd), size = 7000)

train_sample <- bdd[index, ]
test_sample <- bdd[-index, ]

train_sample$NOW_PRIV <- as.factor(train_sample$NOW_PRIV)
test_sample$NOW_PRIV <- as.factor(test_sample$NOW_PRIV)

rf_model <- randomForest(NOW_PRIV ~ ., data = train_sample, ntree = 500, mtry = sqrt(ncol(tra

print(rf_model)
```

```
Call:
 randomForest(formula = NOW_PRIV ~ ., data = train_sample, ntree = 500,      mtry = sqrt(nco
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 3

        OOB estimate of  error rate: 25.83%
Confusion matrix:
     0    1 class.error
0 1693  936   0.3560289
1  872 3499   0.1994967
```

```r
rf_predictions <- predict(rf_model, test_sample)

cf_rf <- confusionMatrix(rf_predictions, test_sample$NOW_PRIV)

print(cf_rf)
```

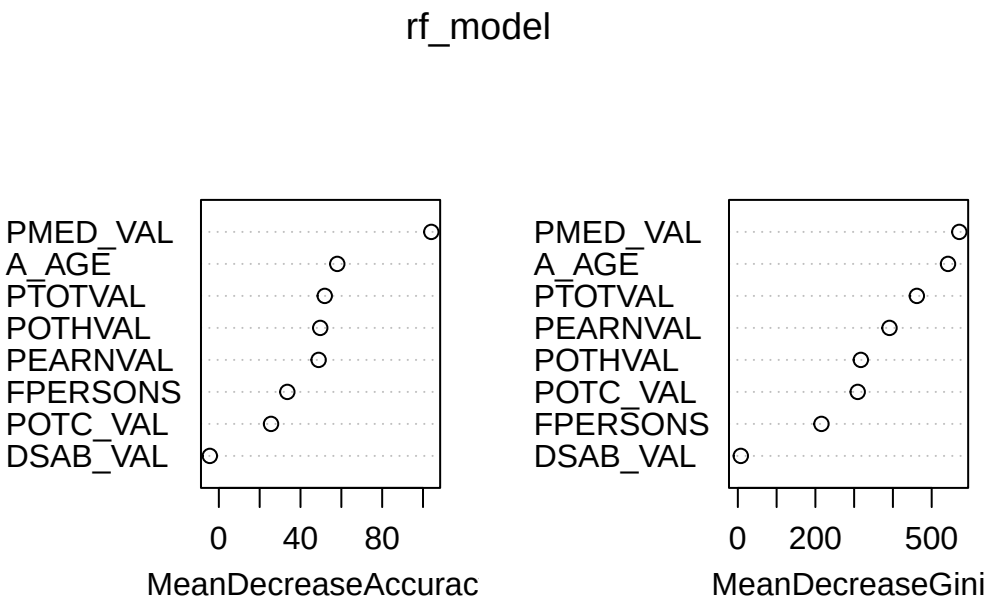
```
Confusion Matrix and Statistics
```

```
          Reference
Prediction    0    1
         0  712  362
         1  426 1500

               Accuracy : 0.7373
                 95% CI : (0.7212, 0.753)
    No Information Rate : 0.6207
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.436

 Mcnemar's Test P-Value : 0.02481

            Sensitivity : 0.6257
            Specificity : 0.8056
         Pos Pred Value : 0.6629
         Neg Pred Value : 0.7788
             Prevalence : 0.3793
         Detection Rate : 0.2373
   Detection Prevalence : 0.3580
      Balanced Accuracy : 0.7156

       'Positive' Class : 0
```

```r
# Affichage de l'importance des variables
importance(rf_model)
```

```
                0        1 MeanDecreaseAccuracy MeanDecreaseGini
A_AGE    -5.797515 61.26253            58.004466       542.107451
PTOTVAL  20.702399 33.50219            51.866542       461.728843
PEARNVAL 20.960567 35.48855            48.883997       391.190496
POTHVAL  25.948920 16.78658            49.667452       317.273357
PMED_VAL 99.326365 44.08338           104.057088       571.278399
POTC_VAL 14.347954 15.76974            25.569185       309.212362
DSAB_VAL  1.526752 -7.31005            -4.404025         7.642064
FPERSONS -5.063186 42.32040            33.550368       215.626647
```

```r
varImpPlot(rf_model)
```



We have 74,6% of total accuracy with an out of bag error of 26.43%.

The main important variable are : PMED_VAL, A_AGE and PTOTVAL

3.8. Hybrid Hierarchical Clustering

3.8.1. Context

The objective is to create and identify clusters that are really "close" to each others. First we proceed to do a descending clustering until a certain threshold. Then we introduce the concept of "mutual cluster", this means that we focus on points really close to each other inside the clusters to create new ones. By this algorithm we have smaller clusters but with a high homogeneity.

Contrary to the ascending clustering method, we don't need to specify another threshold so we loose the noise sensitivity.

To sum up, the mutual cluster algorithm focus on already existing clusters where the ascending clustering method restart from a single point to create new clusters.

3.8.2. Implementation

```
library("devtools")
```

```
Loading required package: usethis
```

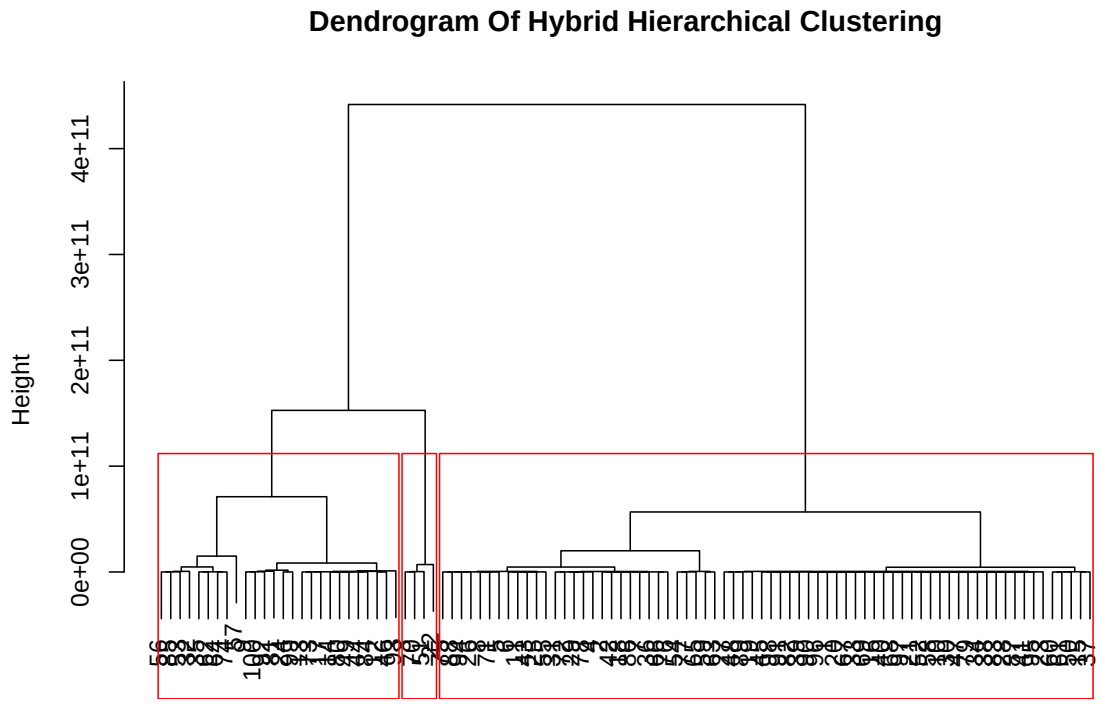
```
devtools::install_github("https://github.com/MathFreedom/hybridHclust_Patch")
```

```
Using GitHub PAT from the git credential store.
```

```
Skipping install of 'hybridHclust' from a github remote, the SHA1 (c0786d12) has not changed
  Use `force = TRUE` to force installation
```

```
library(hybridHclust)
```

```
bdd_100 <- bdd[sample(1:nrow(bdd), 100), ]

hc <- hybridHclust(bdd_100)

plot(hc, main = "Dendrogram Of Hybrid Hierarchical Clustering")

clusters <- cutree(hc, k = 3)

rect.hclust(hc, k=3, border="red")
```




```
clusters_mean <- aggregate(. ~ clusters, data = bdd_100, FUN = mean)
clusters_mean
```

```
  clusters  NOW_PRIV    A_AGE   PTOTVAL PEARNVAL    POTHVAL  PMED_VAL POTC_VAL
1        1 0.4714286 35.58571  17566.13 12276.73   5289.400  451.4857 131.2143
2        2 0.7500000 69.25000 119553.25     0.00 119553.250  771.2500 290.0000
3        3 0.8846154 48.15385  95320.62 89423.08   5897.538 1555.3846 432.5000
  DSAB_VAL FPERSONS
1 128.5714 2.614286
2   0.0000 2.000000
3   0.0000 2.423077
```

Cluster 1: Young people with modest incomes and a majority without private insurance.

Cluster 2: Middle-aged adults with high earned incomes but debts.

Cluster 3: Young people with very high but non-work-related incomes, all with private insurance.

Now, we want to use random forest in order to have uncorrelated trees and have better predictions error.

4. Conclusion

This project has enabled us to analyze the differences between the privately insured and the uninsured in the USA using a variety of statistical methods. Income, age and medical expenditure emerged as key variables. Tree models, in particular random forest, performed best, with an accuracy of 74.6%. Bootstrapping confirmed the robustness of our income estimates. Finally, Hybrid Hierarchical Clustering didn't work well for more than 100 observations leading to not significant result.