A Data Science Lab Project

Zhiyu Quan[*]

03 April 2018

Abstract

Data science lab projects.

Contents

1 Introduction

The fourth actuarial pricing game is a Kaggle-style competition in insurance pricing. In this iteration of the game, the game organizers will provide the players with real historical home insurance data1 and ask the players to give the game organizers their most profitable model for offering insurance premium prices according to the data.

The game organizers will then use previously unseen data and submitted pricing models to simulate a competitive insurance market where players will compete for customers that choose the cheapest premium offered to them. In each market, the player that makes the most money will win.

The game will occur in two stages during each player will submit a pricing model. In the first stage, the player can submit only simple models, while in the second stage player can submit any model you wish. There will be some criteria that models should satisfy.

The first stage of the game, involves some limitations on model complexity. The aim for this stage is for the models to be simple which means the player should use only GLMs for offering premium prices to contracts.

The second stage of the game starts shortly after the first. There are no requirements on the models in this case.

[*]zhiyu.quan@uconn.edu; Ph.D. Candidate at Department of Mathematics, University of Connecticut.

2 Dataset and Explanatory Analysis

The dataset contains approximately 2 million real historical contracts across 3 consecutive years. These are contracts concerning home insurance policies from this decade in France.

For the purposes of the game, the player will each receive a separate and disjoint set of contracts from the first two years of the dataset. This will be training data during the game.

In this iteration of the game, there are more than 200 registered players. However, each market, the game organizers had simulated, will contain a random selection of 20 players. This is so that each player obtains a sizeable part of the dataset.

The remaining contracts from the third year (about 716, 000) will be used in the market simulation and will not be provided to players in advance (like a real market). However, players are provided with a fake, randomly generated, dataset (100 contracts) for which players should offer premiums. The purpose of this fake dataset is to ensure players' model is reproducible.

```r
# setwd('/Users/ZhiyuQuan/Downloads/DataSclass/Pricing
# Game/PricingGame/data') setwd('D:/Github research/PricingGame/data')

set.seed(1)
data_year1 = read.csv("data_year1.csv")  # load data
data_year2 = read.csv("data_year2.csv")

names = c("RISK_NATURE_3", "RISK_FAMILY_STRUCT", "RISK_PROT_1", "RISK_PROT_2")  # change to fa
data_year1[, names] = lapply(data_year1[, names], factor)
data_year2[, names] = lapply(data_year2[, names], factor)

data_year1$IND = data_year1$AMOUNT != 0  # make indicator for claim happen or not
data_year2$IND = data_year2$AMOUNT != 0

data_year1$DEPT = substr(as.character(data_year1$INSEE_CODE), 1, 2)
data_year2$DEPT = substr(as.character(data_year2$INSEE_CODE), 1, 2)

print(str(data_year1))
```

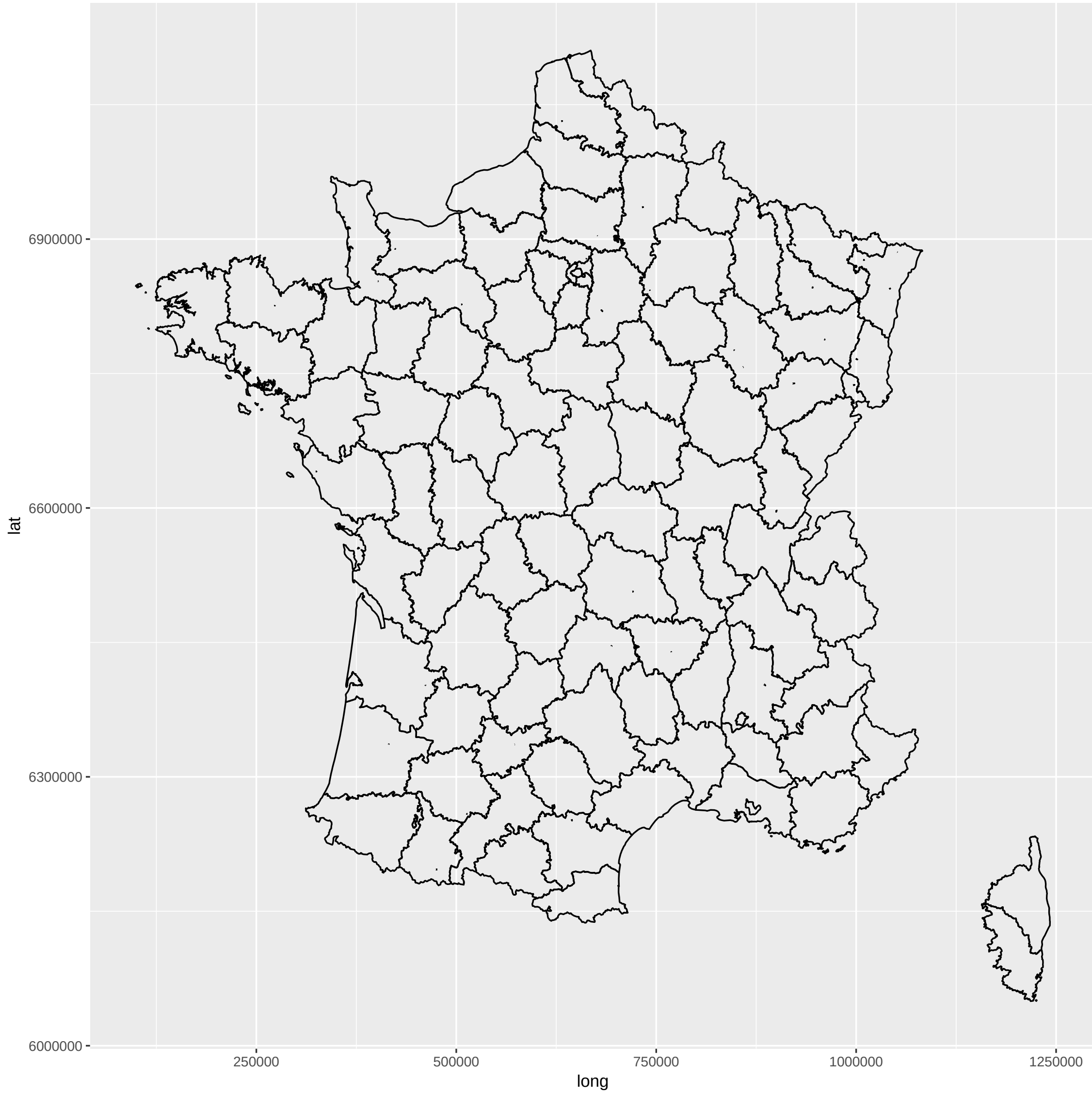
```
## 'data.frame':    34065 obs. of  27 variables:
##  $ IDCNT             : int  1001050 1001964 1002118 1003122 1003822 1004284 1005129 1005343
##  $ RISK_NATURE_1     : Factor w/ 2 levels "PH","SH": 1 1 1 1 1 1 1 1 1 1 ...
##  $ RISK_NATURE_2     : Factor w/ 4 levels "FO","FT","HO",..: 2 3 1 2 2 3 1 2 3 1 ...
##  $ RISK_NATURE_3     : Factor w/ 5 levels "1","2","3","4",..: 2 1 1 2 2 1 5 2 3 1 ...
##  $ RISK_FAMILY_STRUCT: Factor w/ 5 levels "0","1","2","3",..: 5 5 5 2 5 5 3 2 2 5 ...
##  $ RISK_ROOMS_NB     : int  7 4 5 3 3 5 5 2 6 5 ...
##  $ RISK_PROT_1       : Factor w/ 4 levels "1","2","3","4": 1 4 4 1 1 4 1 1 4 4 ...
##  $ RISK_PROT_2       : Factor w/ 3 levels "0","1","2": 2 1 1 2 2 1 2 2 1 1 ...
##  $ SURF_HOUSE        : int  20 8 12 6 5 10 12 3 11 10 ...
##  $ SURF_VER          : int  0 0 0 0 0 0 0 0 3 0 ...
##  $ SURF_OUTB         : int  0 1 0 0 0 1 0 0 1 0 ...
##  $ SURF_GR           : int  0 1 0 0 0 1 0 0 5 0 ...
```

```
##  $ OPTION_1           : Factor w/ 2 levels "False","True": 1 1 1 1 1 1 1 1 1 1 ...
##  $ OPTION_2           : Factor w/ 2 levels "False","True": 1 1 1 1 1 1 1 1 1 1 ...
##  $ OPTION_3           : Factor w/ 2 levels "False","True": 1 1 1 1 1 1 1 1 1 1 ...
##  $ OPTION_4           : Factor w/ 2 levels "False","True": 1 1 1 1 1 1 1 1 1 1 ...
##  $ OPTION_5           : Factor w/ 2 levels "False","True": 1 1 1 1 1 1 1 1 1 1 ...
##  $ OPTION_6           : Factor w/ 2 levels "False","True": 1 1 1 1 1 1 1 1 1 1 ...
##  $ OPTION_7           : Factor w/ 2 levels "False","True": 1 1 1 1 1 1 1 1 1 1 ...
##  $ OPTION_8           : Factor w/ 2 levels "False","True": 1 2 1 1 1 1 1 1 1 1 ...
##  $ INSEE_CODE         : Factor w/ 10215 levels "01002","01004",..: 3359 7485 7239 9663 7190 
##  $ ZONE_1             : Factor w/ 19 levels "B","C","D","E",..: 14 11 8 3 10 5 15 16 5 14 ..
##  $ ZONE_2             : Factor w/ 16 levels "B","C","D","E",..: 10 11 7 4 10 7 16 16 4 14 ..
##  $ ZONE_3             : Factor w/ 9 levels "C","E","G","H",..: 9 2 2 1 3 2 1 1 6 1 ...
##  $ AMOUNT             : num  3506 0 0 0 0 ...
##  $ IND                : logi  TRUE FALSE FALSE FALSE FALSE FALSE ...
##  $ DEPT               : chr  "34" "69" "67" "87" ...
## NULL
```

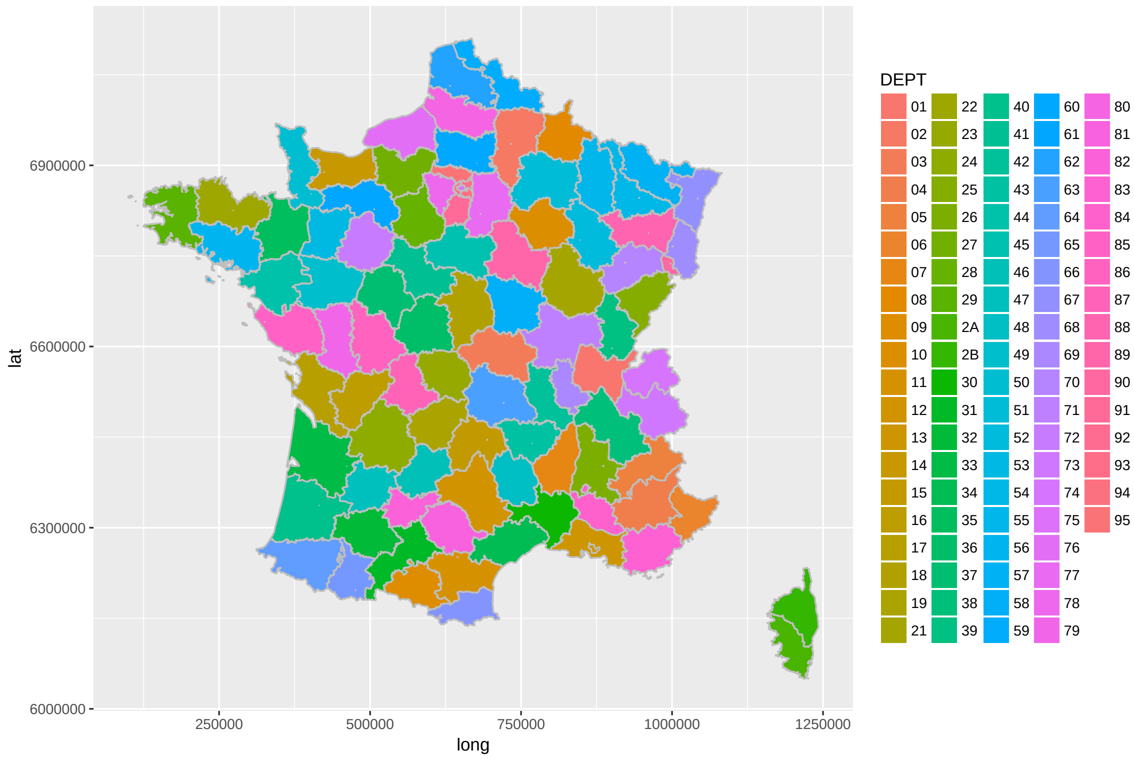
```r
print(str(data_year2))
```

```
## 'data.frame':    35142 obs. of  27 variables:
##  $ IDCNT              : int  2000018 2000704 2002227 2002478 2002704 2002713 2003048 2003104
##  $ RISK_NATURE_1      : Factor w/ 2 levels "PH","SH": 1 2 1 1 1 1 1 1 1 1 ...
##  $ RISK_NATURE_2      : Factor w/ 4 levels "FO","FT","HO",..: 1 1 4 1 3 3 1 3 2 3 ...
##  $ RISK_NATURE_3      : Factor w/ 5 levels "1","2","3","4",..: 1 5 2 1 1 1 5 1 2 1 ...
##  $ RISK_FAMILY_STRUCT: Factor w/ 5 levels "0","1","2","3",..: 2 1 2 2 2 2 2 5 3 5 ...
##  $ RISK_ROOMS_NB      : int  5 1 3 5 2 3 3 4 3 7 ...
##  $ RISK_PROT_1        : Factor w/ 4 levels "1","2","3","4": 4 4 4 4 4 4 4 1 4 1 ...
##  $ RISK_PROT_2        : Factor w/ 3 levels "0","1","2": 1 1 1 1 1 1 1 2 1 2 ...
##  $ SURF_HOUSE         : int  10 1 6 9 6 6 6 10 5 13 ...
##  $ SURF_VER           : int  0 0 2 0 0 0 0 0 0 0 ...
##  $ SURF_OUTB          : int  0 0 0 0 1 1 0 1 0 1 ...
##  $ SURF_GR            : int  0 0 1 0 1 2 0 3 0 2 ...
##  $ OPTION_1           : Factor w/ 2 levels "False","True": 1 1 1 1 1 1 1 1 1 1 ...
##  $ OPTION_2           : Factor w/ 2 levels "False","True": 1 1 1 1 1 1 1 1 1 1 ...
##  $ OPTION_3           : Factor w/ 2 levels "False","True": 1 1 1 1 1 1 1 1 1 1 ...
##  $ OPTION_4           : Factor w/ 2 levels "False","True": 1 1 1 1 1 1 1 1 1 1 ...
##  $ OPTION_5           : Factor w/ 2 levels "False","True": 1 1 1 1 1 1 1 1 1 1 ...
##  $ OPTION_6           : Factor w/ 2 levels "False","True": 1 1 1 1 1 2 2 1 1 1 ...
##  $ OPTION_7           : Factor w/ 2 levels "False","True": 1 1 1 1 1 1 1 2 1 1 ...
##  $ OPTION_8           : Factor w/ 2 levels "False","True": 1 1 1 1 1 2 1 1 1 1 ...
##  $ INSEE_CODE         : Factor w/ 10394 levels "01004","01005",..: 1162 8477 1694 8741 10221
##  $ ZONE_1             : Factor w/ 20 levels "B","C","D","E",..: 18 15 6 12 16 4 5 5 10 2 ...
##  $ ZONE_2             : Factor w/ 14 levels "B","C","D","E",..: 9 14 4 9 11 4 4 7 9 3 ...
##  $ ZONE_3             : Factor w/ 10 levels "C","E","G","H",..: 6 1 8 6 1 6 5 1 6 6 ...
##  $ AMOUNT             : num  0 0 0 0 0 ...
##  $ IND                : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ DEPT               : chr  "13" "75" "18" "77" ...
## NULL
```

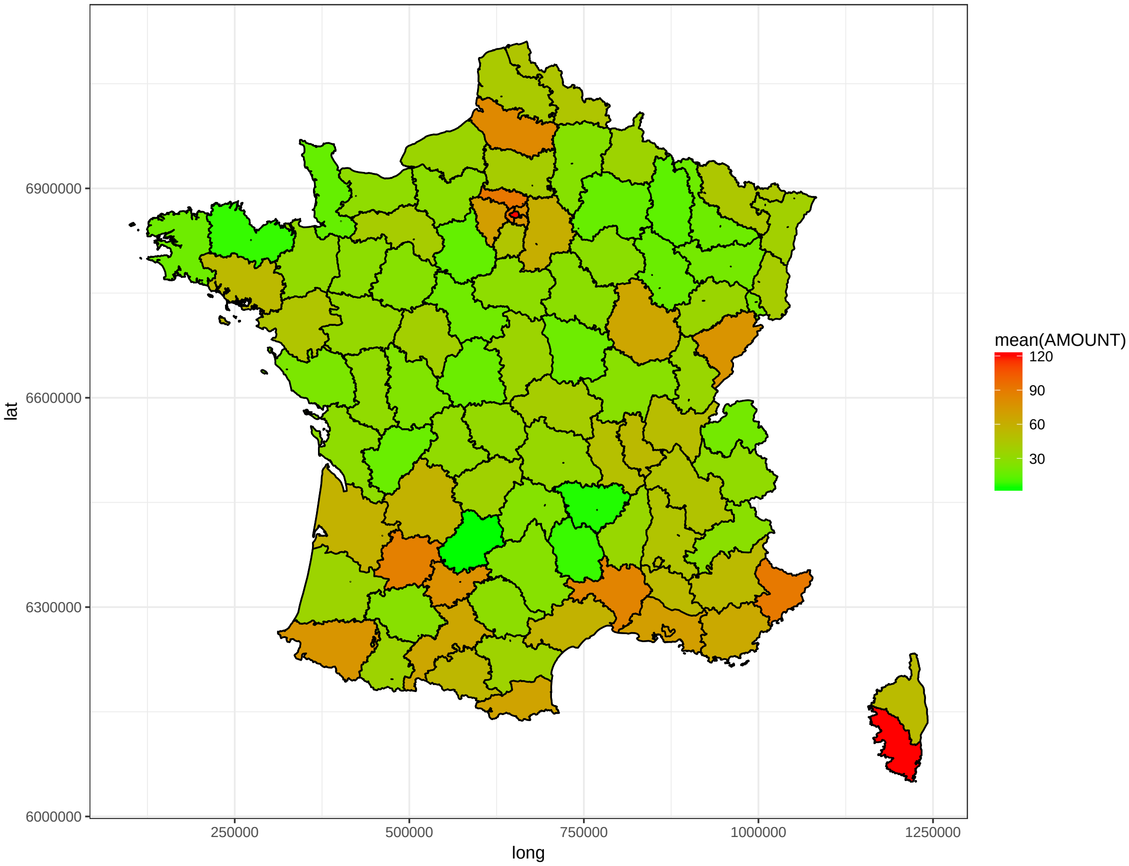
```r
load("spatial.RData")

CombinedData = rbind.data.frame(data_year1, data_year2)
library(dplyr)
MeanAmount = CombinedData %>% group_by(DEPT) %>% summarise(mean(AMOUNT))

DEPARTMENTS.spatial = dplyr::inner_join(MeanAmount, DEPARTMENTS.df, by = "DEPT")

library(ggplot2)
ggplot(data = DEPARTMENTS.df, aes(x = long, y = lat, group = group)) +
    geom_path()
```



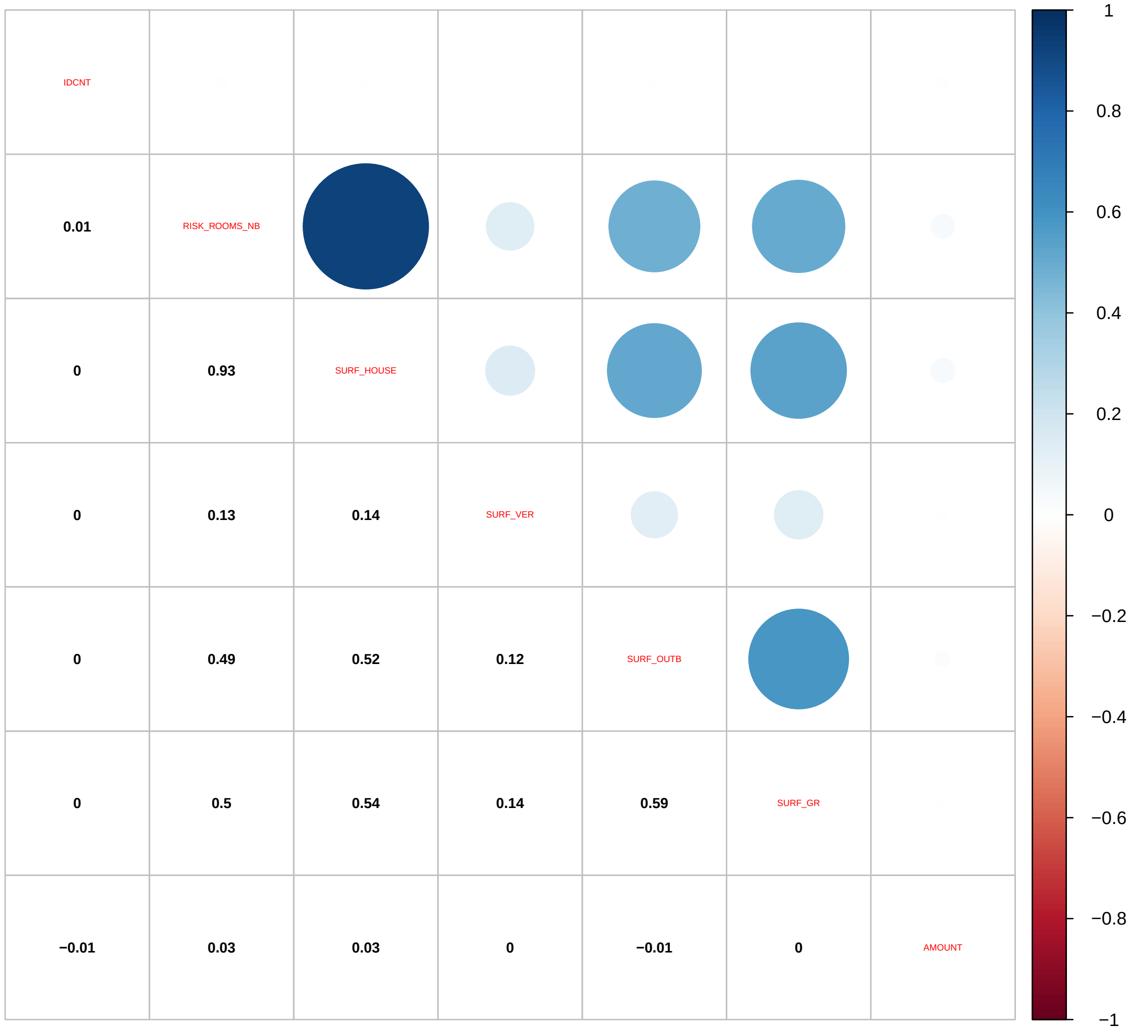

```r
ggplot(DEPARTMENTS.df, aes(x = long, y = lat, group = group, fill = DEPT)) +
    geom_polygon() + geom_path(color = "gray") + coord_equal() + scale_colour_brewer("Departmen
```




```
ggplot(data = DEPARTMENTS.spatial, mapping = aes(x = long, y = lat, group = group)) +
    coord_equal() + geom_polygon(color = "black", fill = "white") + geom_polygon(data = DEPARTI
    aes(fill = `mean(AMOUNT)`), color = "black") + theme_bw() + scale_fill_gradient(low = "gree
    high = "red")
```




```r
NumericVariables <- sapply(CombinedData, is.numeric)
correlation = cor(CombinedData[, NumericVariables])

library(corrplot)
corrplot.mixed(correlation, lower.col = "black", number.cex = 0.7, tl.cex = 0.4)
```




```r
print(str(CombinedData))
```

```
## 'data.frame':    69207 obs. of  27 variables:
##  $ IDCNT            : int  1001050 1001964 1002118 1003122 1003822 1004284 1005129 1005343
##  $ RISK_NATURE_1    : Factor w/ 2 levels "PH","SH": 1 1 1 1 1 1 1 1 1 1 ...
##  $ RISK_NATURE_2    : Factor w/ 4 levels "FO","FT","HO",..: 2 3 1 2 2 3 1 2 3 1 ...
##  $ RISK_NATURE_3    : Factor w/ 5 levels "1","2","3","4",..: 2 1 1 2 2 1 5 2 3 1 ...
##  $ RISK_FAMILY_STRUCT: Factor w/ 5 levels "0","1","2","3",..: 5 5 5 2 5 5 3 2 2 5 ...
##  $ RISK_ROOMS_NB    : int  7 4 5 3 3 5 5 2 6 5 ...
##  $ RISK_PROT_1      : Factor w/ 4 levels "1","2","3","4": 1 4 4 1 1 4 1 1 4 4 ...
##  $ RISK_PROT_2      : Factor w/ 3 levels "0","1","2": 2 1 1 2 2 1 2 2 1 1 ...
##  $ SURF_HOUSE       : int  20 8 12 6 5 10 12 3 11 10 ...
##  $ SURF_VER         : int  0 0 0 0 0 0 0 0 3 0 ...
##  $ SURF_OUTB        : int  0 1 0 0 0 1 0 0 1 0 ...
##  $ SURF_GR          : int  0 1 0 0 0 1 0 0 5 0 ...
##  $ OPTION_1         : Factor w/ 2 levels "False","True": 1 1 1 1 1 1 1 1 1 1 ...
##  $ OPTION_2         : Factor w/ 2 levels "False","True": 1 1 1 1 1 1 1 1 1 1 ...
##  $ OPTION_3         : Factor w/ 2 levels "False","True": 1 1 1 1 1 1 1 1 1 1 ...
##  $ OPTION_4         : Factor w/ 2 levels "False","True": 1 1 1 1 1 1 1 1 1 1 ...
```

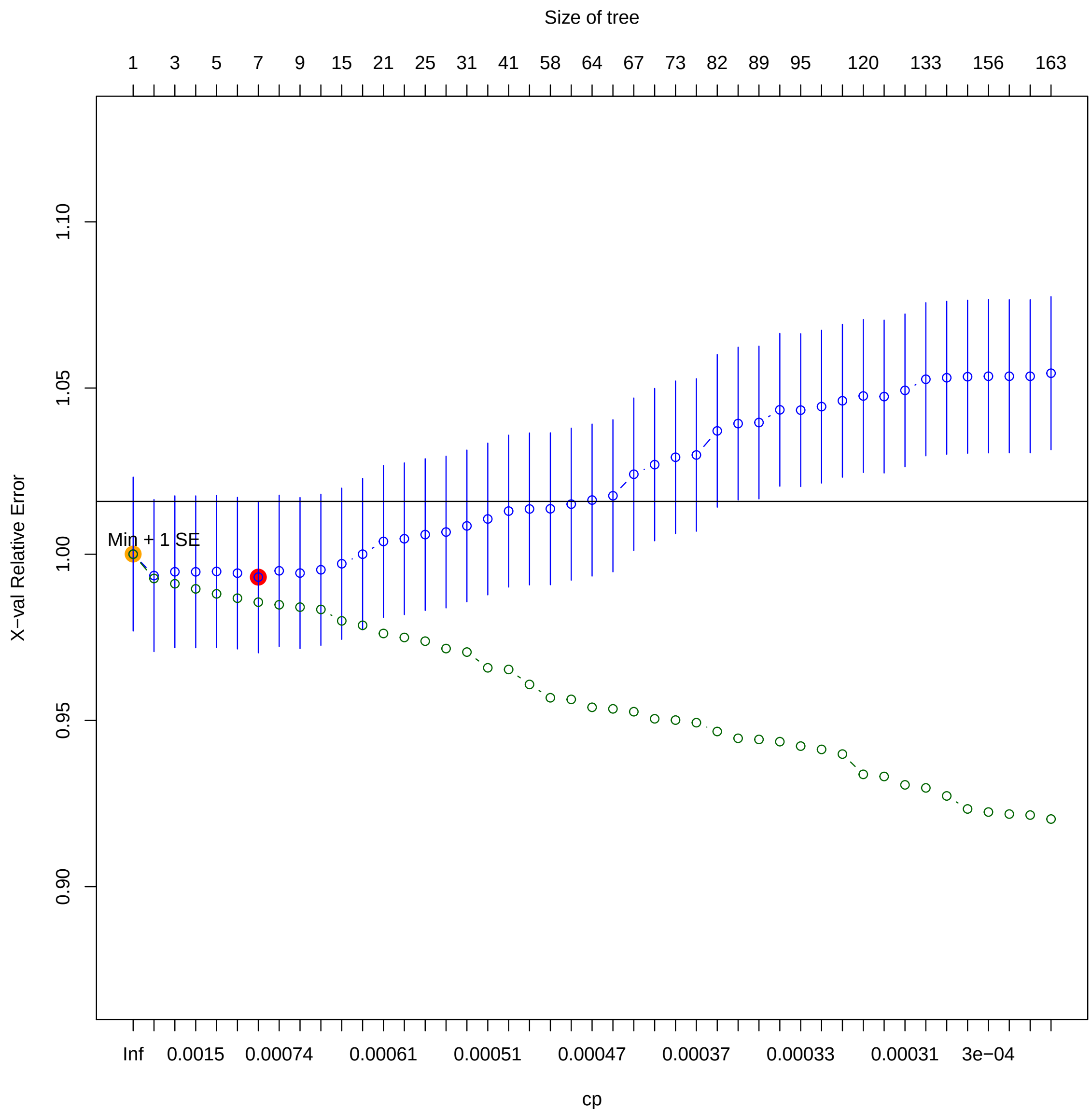
```
##  $ OPTION_5           : Factor w/ 2 levels "False","True": 1 1 1 1 1 1 1 1 1 1 ...
##  $ OPTION_6           : Factor w/ 2 levels "False","True": 1 1 1 1 1 1 1 1 1 1 ...
##  $ OPTION_7           : Factor w/ 2 levels "False","True": 1 1 1 1 1 1 1 1 1 1 ...
##  $ OPTION_8           : Factor w/ 2 levels "False","True": 1 2 1 1 1 1 1 1 1 1 ...
##  $ INSEE_CODE         : Factor w/ 14690 levels "01002","01004",..: 3359 7485 7239 9663 7190 3
##  $ ZONE_1             : Factor w/ 20 levels "B","C","D","E",..: 14 11 8 3 10 5 15 16 5 14 ..
##  $ ZONE_2             : Factor w/ 16 levels "B","C","D","E",..: 10 11 7 4 10 7 16 16 4 14 ..
##  $ ZONE_3             : Factor w/ 10 levels "C","E","G","H",..: 9 2 2 1 3 2 1 1 6 1 ...
##  $ AMOUNT             : num  3506 0 0 0 0 ...
##  $ IND                : logi  TRUE FALSE FALSE FALSE FALSE FALSE ...
##  $ DEPT               : chr  "34" "69" "67" "87" ...
## NULL
```

```r
CombinedData$DEPT = as.factor(CombinedData$DEPT)
# paste(names(CombinedData), collapse=' + ')

library(rpart)
# grow tree
FitRpart = rpart(log(AMOUNT + 1) ~ RISK_NATURE_1 + RISK_NATURE_2 + RISK_NATURE_3 +
    RISK_FAMILY_STRUCT + RISK_ROOMS_NB + RISK_PROT_1 + RISK_PROT_2 + SURF_HOUSE +
    SURF_VER + SURF_OUTB + SURF_GR + OPTION_1 + OPTION_2 + OPTION_3 + OPTION_4 +
    OPTION_5 + OPTION_6 + OPTION_7 + OPTION_8 + ZONE_1 + ZONE_2 + ZONE_3,
    method = "anova", data = data_year1, control = rpart.control(minsplit = 13,
        cp = 3e-04))

printcp(FitRpart)  # display the results
```

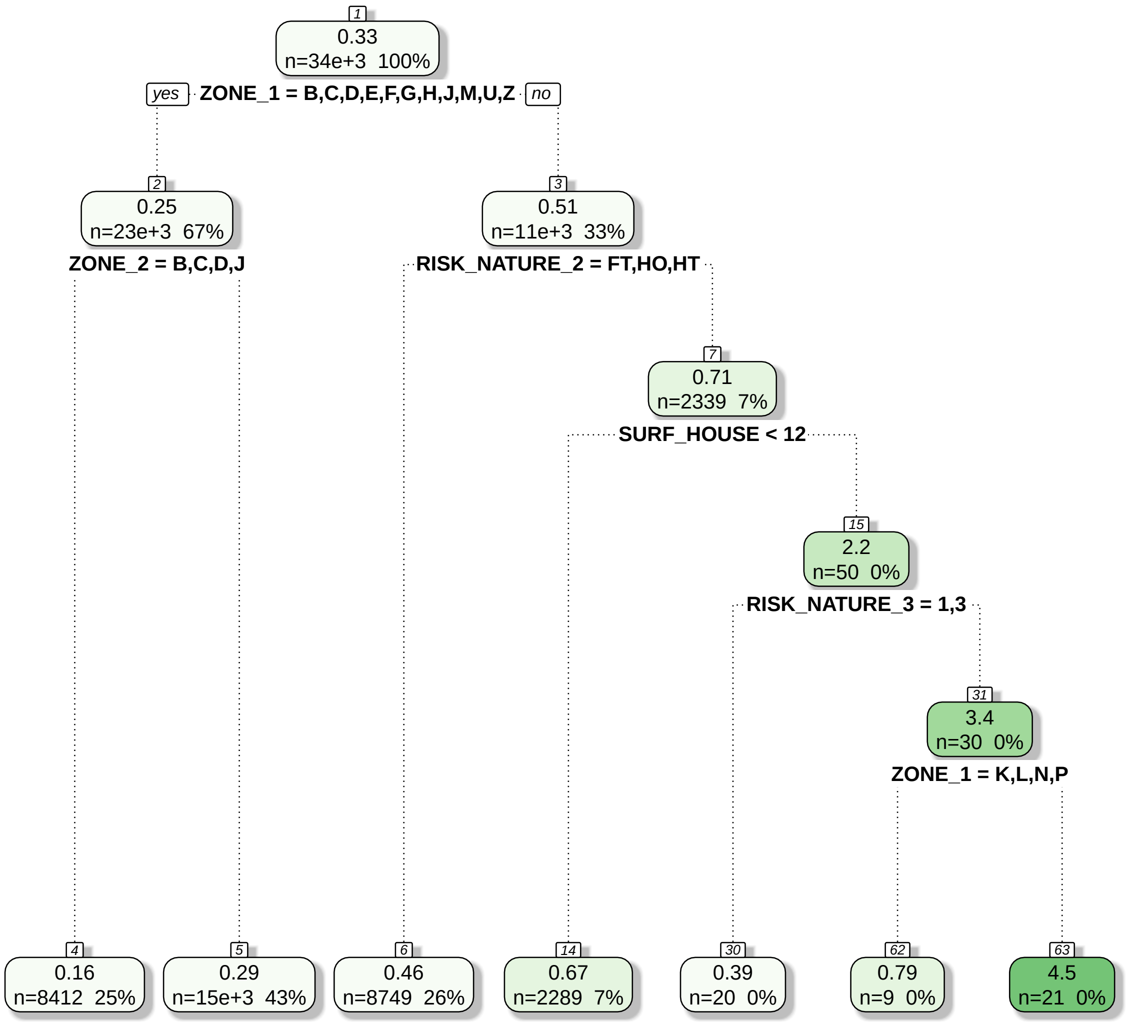
```
##
## Regression tree:
## rpart(formula = log(AMOUNT + 1) ~ RISK_NATURE_1 + RISK_NATURE_2 +
##     RISK_NATURE_3 + RISK_FAMILY_STRUCT + RISK_ROOMS_NB + RISK_PROT_1 +
##     RISK_PROT_2 + SURF_HOUSE + SURF_VER + SURF_OUTB + SURF_GR +
##     OPTION_1 + OPTION_2 + OPTION_3 + OPTION_4 + OPTION_5 + OPTION_6 +
##     OPTION_7 + OPTION_8 + ZONE_1 + ZONE_2 + ZONE_3, data = data_year1,
##     method = "anova", control = rpart.control(minsplit = 13,
##         cp = 3e-04))
##
## Variables actually used in tree construction:
##  [1] OPTION_3           OPTION_4           OPTION_5
##  [4] OPTION_6           OPTION_7           OPTION_8
##  [7] RISK_FAMILY_STRUCT RISK_NATURE_1      RISK_NATURE_2
## [10] RISK_NATURE_3      RISK_PROT_1        RISK_PROT_2
## [13] RISK_ROOMS_NB      SURF_GR            SURF_HOUSE
## [16] SURF_OUTB          SURF_VER           ZONE_1
## [19] ZONE_2             ZONE_3
##
## Root node error: 72068/34065 = 2.1156
##
```

```
## n= 34065
##
##            CP nsplit rel error  xerror     xstd
## 1  0.00730071      0   1.00000 1.00005 0.023158
## 2  0.00157669      1   0.99270 0.99358 0.022869
## 3  0.00154418      2   0.99112 0.99472 0.022848
## 4  0.00148907      3   0.98958 0.99471 0.022842
## 5  0.00131541      4   0.98809 0.99482 0.022841
## 6  0.00119492      5   0.98677 0.99431 0.022805
## 7  0.00078267      6   0.98558 0.99311 0.022774
## 8  0.00070471      7   0.98480 0.99502 0.022751
## 9  0.00070231      8   0.98409 0.99433 0.022724
## 10 0.00067470      9   0.98339 0.99533 0.022733
## 11 0.00067163     14   0.97996 0.99714 0.022749
## 12 0.00061459     16   0.97862 1.00001 0.022774
## 13 0.00059656     20   0.97616 1.00385 0.022808
## 14 0.00055769     22   0.97497 1.00466 0.022807
## 15 0.00055683     24   0.97385 1.00591 0.022823
## 16 0.00052936     28   0.97162 1.00667 0.022815
## 17 0.00052163     30   0.97056 1.00852 0.022820
## 18 0.00050430     39   0.96584 1.01060 0.022830
## 19 0.00050051     40   0.96533 1.01298 0.022836
## 20 0.00050019     49   0.96083 1.01362 0.022845
## 21 0.00049341     57   0.95683 1.01367 0.022845
## 22 0.00047539     58   0.95633 1.01507 0.022856
## 23 0.00046535     63   0.95396 1.01630 0.022867
## 24 0.00043066     64   0.95349 1.01759 0.022876
## 25 0.00042653     66   0.95263 1.02406 0.022928
## 26 0.00038813     71   0.95050 1.02696 0.022917
## 27 0.00038699     72   0.95011 1.02917 0.022929
## 28 0.00036182     74   0.94933 1.02986 0.022928
## 29 0.00034176     81   0.94668 1.03710 0.022949
## 30 0.00034105     87   0.94463 1.03931 0.022967
## 31 0.00033531     88   0.94429 1.03963 0.022970
## 32 0.00033307     90   0.94361 1.04345 0.022986
## 33 0.00033186     94   0.94228 1.04335 0.022981
## 34 0.00032812     97   0.94129 1.04441 0.022982
## 35 0.00032165    101   0.93988 1.04617 0.022994
## 36 0.00031786    119   0.93378 1.04760 0.022995
## 37 0.00031443    121   0.93314 1.04742 0.022986
## 38 0.00030312    129   0.93062 1.04929 0.022994
## 39 0.00030241    132   0.92971 1.05264 0.023018
## 40 0.00030233    140   0.92729 1.05310 0.023019
## 41 0.00030133    152   0.92337 1.05341 0.023021
## 42 0.00030093    155   0.92244 1.05353 0.023024
## 43 0.00030071    157   0.92184 1.05353 0.023024
## 44 0.00030048    158   0.92154 1.05353 0.023024
## 45 0.00030000    162   0.92034 1.05445 0.023031
```

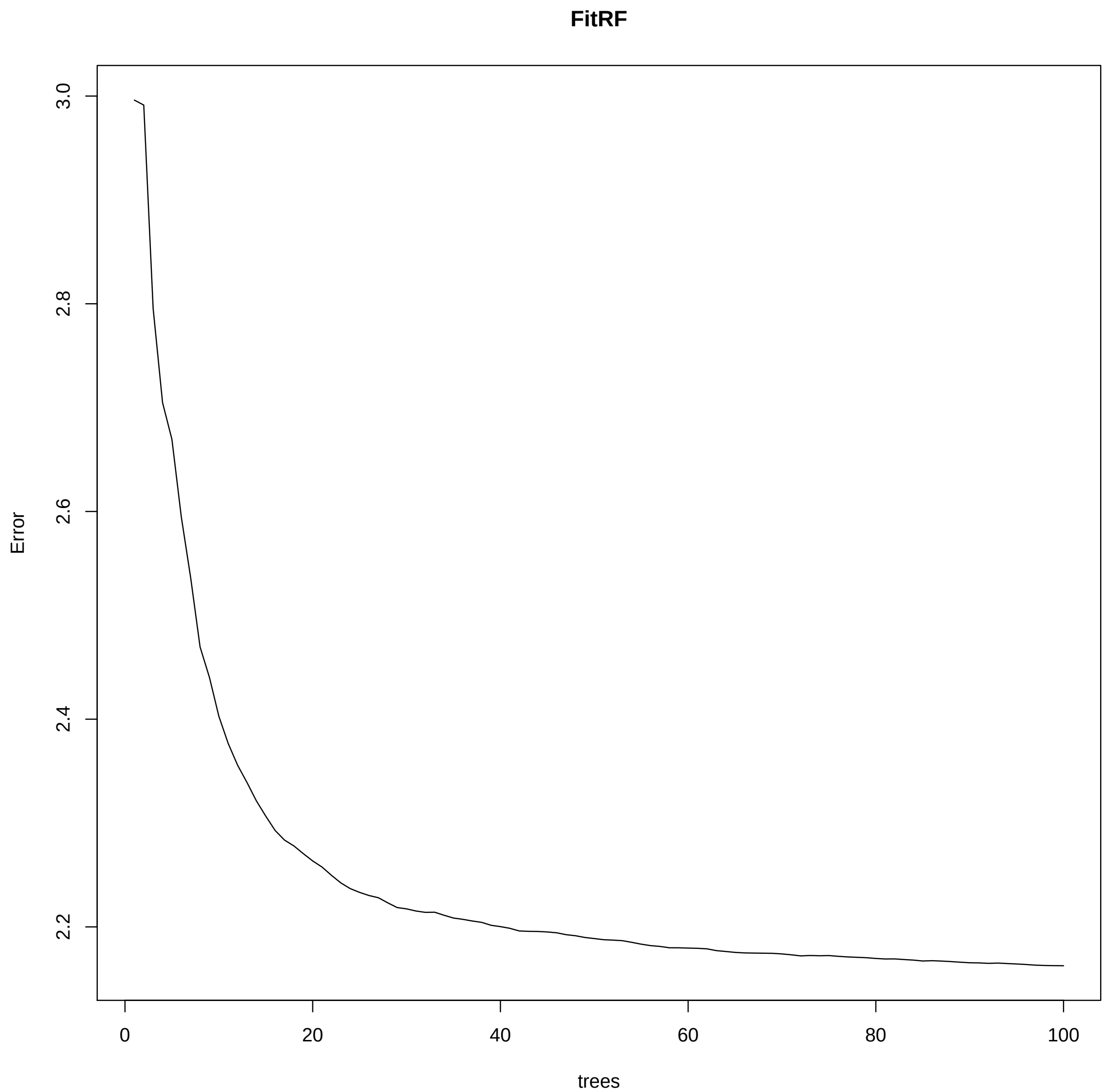
```r
mvpart::plotcp(FitRpart)  # display the results
```



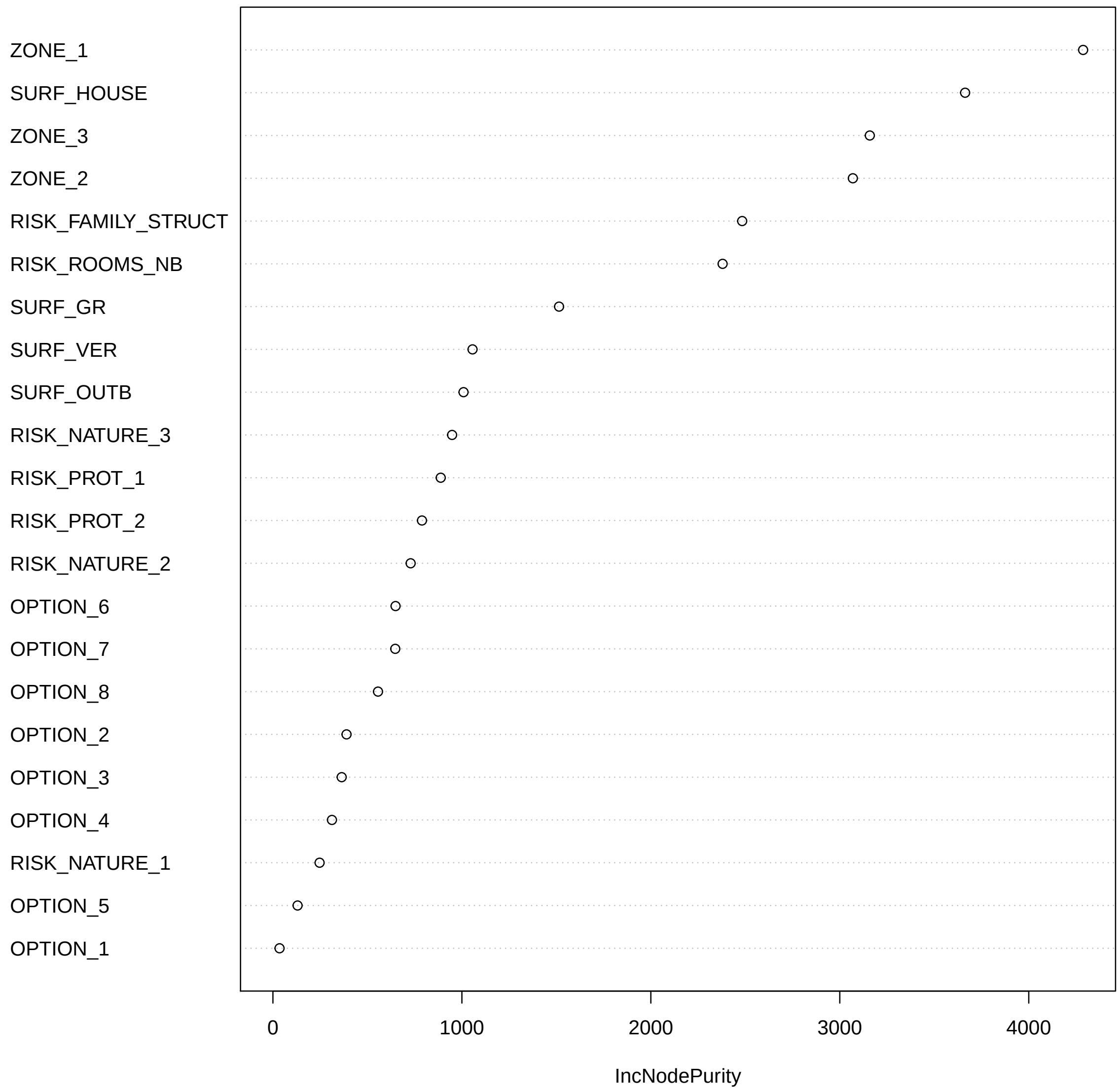

```r
# summary(FitRpart) # detailed summary of splits

# prune tree
PruneFit = rpart::prune(FitRpart, cp = FitRpart$cptable[which.min(FitRpart$cptable[,
    "xerror"]), "CP"])
library(rattle)
fancyRpartPlot(PruneFit, caption = NULL)
```



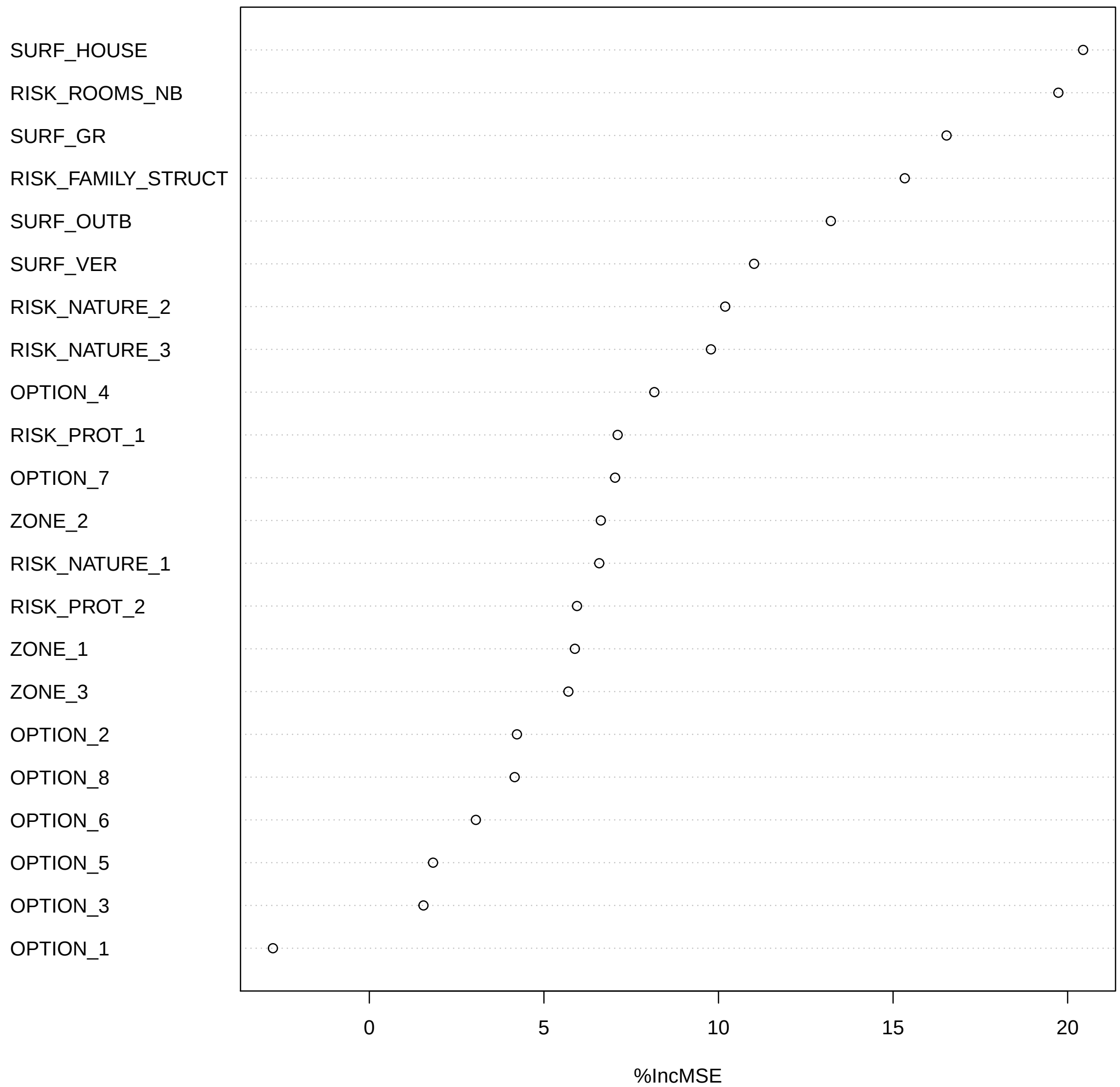

```r
library(randomForest)
FitRF = randomForest(log(AMOUNT + 1) ~ RISK_NATURE_1 + RISK_NATURE_2 +
    RISK_NATURE_3 + RISK_FAMILY_STRUCT + RISK_ROOMS_NB + RISK_PROT_1 +
    RISK_PROT_2 + SURF_HOUSE + SURF_VER + SURF_OUTB + SURF_GR + OPTION_1 +
    OPTION_2 + OPTION_3 + OPTION_4 + OPTION_5 + OPTION_6 + OPTION_7 + OPTION_8 +
    ZONE_1 + ZONE_2 + ZONE_3, method = "anova", data = data_year1, importance = TRUE,
    ntree = 100, nodesize = 10, mtry = 5)
plot(FitRF)
```




```
varImpPlot(FitRF, type = 2, main = NULL)  # IncNodePurity
```




```r
varImpPlot(FitRF, type = 1, main = NULL)   # IncMSE
```




```
library(statmod)
library(tweedie)
power = tweedie::tweedie.profile(CombinedData$AMOUNT~1,
                        xi.vec = seq(1, 2, length = 11),
                        #method="saddlepoint",
                        do.plot = TRUE,
                        do.points = TRUE)
power$xi.max

### gini test
giniTest <- function(y, py){
  # Calculate a gini score
  # Parameters:
  #     y: The real data from observation
  #     py: The predicted value from model
  # Return: gini score for this prediction
```

```r
  # Algorithm:
  #   1. Use a uniform random vector to break ties from predicted values
  #   2. Use predicted values to rank observed y
  #   3. Calculated gini score used ranked observed y.
  data <- as.data.frame(cbind(y, py))
  set.seed(1)
  n <- length(y)
  data$rand.unif <- runif(n)

  sorted.y <- data[order(data$py, data$rand.unif), ][, 1]
  i <- seq(n)
  giniIndex <- 1-2/(n-1)*(n-sum(sorted.y*i)/sum(sorted.y))
  return(giniIndex)
}


### binary function
binary.add <- function(vec, k){
  # Perform a binary addition. Add 1 to current value.
  # Parameters:
  #   vec, a vectory/1D array containing TRUE or FALSE
  #   k: the position to add 1
  # Return: The result of adding 1 to current value.
  # For example:  {TRUE, FALSE, FALSE} -> {TRUE, FALSE, TRUE}
  #               {TRUE, FALSE, TRUE}  -> {TRUE, TRUE, FALSE}
  if(k == 0) return(vec)
  if(vec[k] == FALSE){
    vec[k] = TRUE
    return(vec)
  }
  vec[k] = FALSE
  binary.add(vec, k-1)
}

varnames = names(CombinedData)[2:24][-20]  # All available variables may be used for fit
n = length(varnames)  # Number of available variables X's
index = array(FALSE, n)  # An indicator saying which x's will be used for fit
giniScore = array(0, 2^n-1)  # An array to collect gini scores
formulas = vector("list", 2^n-1)  # An list to collect formulas

# Loop all possible combinations of x's,
# Perform Tweedie fit
# Record gini score and fiting formula
# Note: y ~ 1 is not fitted. Thus, 2^n - 1 fittings are performed.
for (i in 1:(2^n - 1)) {
  index <- binary.add(index, n)
  fmla <- as.formula(paste("AMOUNT ~ ", paste(varnames[index], collapse = "+")))
```

```r
  fit1 <- glm(formula = fmla, family = tweedie(var.power = 1.4, link.power = 0),
              data = CombinedData)
  data_year2.py <- predict.glm(fit1, newdata = data_year2, type="response")
  formulas[[i]] <- fmla
  giniScore[i] <- giniTest(y = data_year2$AMOUNT, py = data_year2.py)
  print(i)
}

bestScore <- max(giniScore)
print(bestScore)
bestFormula <- formulas[[which(giniScore == bestScore)]]
print(bestFormula)


tweedie_fit <- glm(AMOUNT ~ OPTION_4 + OPTION_6 + OPTION_8 + ZONE_1 + ZONE_2 + ZONE_3,
                   family = tweedie(var.power = 1.4, link.power = 0),
                   data = CombinedData)
summary(tweedie_fit)


### Quantile regression
library(quantreg)
varnames = names(CombinedData)[2:24][-20] # All available variables may be used for fit
n <- length(varnames)       # Number of available variables X's
index <- array(FALSE, n)    # An indicator saying which x's will be used for fit
giniScore <- array(0, 2^n-1) # An array to collect gini scores
formulas <- vector("list", 2^n-1) # An list to collect formulas

for (i in 1:(2^n - 1)) {
  index <- binary.add(index, n)
  fmla <- as.formula(paste("AMOUNT ~ ", paste(varnames[index], collapse = "+")))
  fit1 <- rq(formula = fmla, data = CombinedData, tau = 0.95)
  data_year2.py <- predict.rq(fit1, newdata = data_year2)
  formulas[[i]] <- fmla
  giniScore[i] <- giniTest(y = data_year2$AMOUNT, py = data_year2.py)
  print(i)
}

bestScore <- max(giniScore)
print(bestScore)
bestFormula <- formulas[[which(giniScore == bestScore)]]
print(bestFormula)

quantile_fit <- rq(AMOUNT ~ ,
                   data = CombinedData, tau = 0.95)
```

3 Summary and Discussion

Appendix: Variable dictionary

The two training datasets (`data year1.csv` and `data year2.csv`) contain the following 25 variables:

1. `IDCNT` Unique contract ID
2. `RISK NATURE 1` An Indicator for whether this is a primary house PH or a secondary house SH.
3. `RISK NATURE 2` Nature of the ownership and house type. This indicates, both if this is a house H or a flat F, and, if the client is the owner O or a tenant T. Hence there are 4 possible values FO, FT, HO, HT.)
4. `RISK NATURE 3` The type of occupancy. This can be one of '001', '002', '003', '004', '005'. For example, '002' indicates a tenant while the others are different types of occupancy.
5. `RISK FAMILY STRUCT` The family structure. This can be one of '000', '001', '002', '003', '004' representing different combinations of adults and children in the client family.
6. `RISK ROOMS NB` Number of declared rooms. Note that '011' means 11 or more (can be up to 20 in real life).
7. `RISK PROT 1` Protection indicator 1. This can be one of '001', '002', '003', '004' and represents different house protection schemes.
8. `RISK PROT 2` Protection indicator 2. This can be one of '000', '001', '002'. This represents another class of house protection schemes.
9. `SURF HOUSE` Total house surface area. This ranges from '001' to '028', representing the real house surface area in ascending order.
10. `SURF VER` Veranda surface area. This ranges from '000' to '021', in ascending order of the declared veranda surface area. For example, '000' means no veranda present.
11. `SURF OUTB` Extra buildings surface area. This ranges from '000' to '011', in ascending order of area. This captures the total area of external buildings other than the main house. For example, '000' means no other building than the main house.
12. `SURF GR` Ground surface. Ranges from '000' to '014', in ascending order of the declared terrain surface. Again '000' means no free terrain surface.
13. `OPTION 1` Different options (true or false)
14. `OPTION 2` Different options (true or false)
15. `OPTION 3` Different options (true or false)
16. `OPTION 4` Different options (true or false)
17. `OPTION 5` Different options (true or false)
18. `OPTION 6` Different options (true or false)
19. `OPTION 7` Different options (true or false)

20. `OPTION 8` Different options (true or false)
21. `INSEE CODE` The INSEE code is a numerical index used by the French National Institute for Statistics and Economic Studies (INSEE) to identify various entities, including communes, departments.
22. `ZONE 1` Some geographical information. The zone scales, alphabetically ordered. Each zone is a combination of communes.
23. `ZONE 2` Other geographical zones.
24. `ZONE 3` Other geographical zones.
25. `AMOUNT` Total loss for that policy.

Please note: The `INSEE CODE` is a 5-digits alphanumeric code used by the French National Institute for Statistics and Economic Studies identify communes and departments in France. There are about 36,000 communes in France, but not every one of them is present in the dataset. The first 2 digits of INSEE CODE identifies the department. The `INSEE CODE` (or department code) can be used to merge external data to the datasets: population density, OpenStreetMap data, etc. If needed, two shapefiles are available online:

For French regional Departments https://pricing-game.dsi.ic.ac.uk/static/DEPARTEMENTS.zip

For communes https://pricing-game.dsi.ic.ac.uk/static/COMMUNES.zip

Be aware that, if you need to graph geographical information, the French reference system is RGF93 / Lambert-93 (EPSG: 2154) and not the common WGS84.

Reference