Customer Churn Prediction Using Random Forest Model

Customer churn is well known problem in business environment. By customer churn we mean situation when returning clients stop buying company's goods or services. We determine customer churn rate as a percentage of such customers to total number of them. To manage churn rate is important, because for company it costs less to retain customer then to find new one. Another crucial factor to consider is profit. Returning customers usually spend 67% more on company's product and services. If company increases customer retention (or decreases churn rate) by 5%, then it will result in 25% profit increase. If classify customers correctly, then we can take preemptive measures and try to retain customers. Therefore, churn rate prediction plays significant role in firm's profit maximization.

Brief data description and preparation

In this research we use data on clients of telecom company. Data contains characteristics of 7043 customers. Each customer has the following features: ID, gender, seniority status, partner, dependents, tenure term, phone service, multiple lines, internet service, online security, online backup, device protection, technical support, TV streaming, movies streaming, contract term, paperless billing, payment method, monthly charges, total charges, churn. Before proceeding to model's estimation we make data examination. We get reed of missing values: there were 11 observations, which contained null values in any column. We also transform categorical variables into factors, so R built-in models could easily process them. As tenure variable is integer and contains values between 1 and 72 (number of months for which client uses company's services) we decide to split this variable into folds. We split values into 5 folds: less or equal to 1 year, between a year and 2 years, between 2 and 3 years, between 3 and 5 years, more than 5 years. One would expect the more time customer sticks to company less likely it will churn. In this data set 1869 clients terminated their subscriptions, which is approximately 26.58% of total customers number. We provide more detailed data summary in Appendix 2.

Before splitting data into training and test samples we remove unnecessary variables from original dataset. These variables are customer ID (tells us nothing) and tenure (as we created new factor variable). As we get rid of these features out total number of variables becomes 20. We divide our data into training and test sets. 70% of original set is going to be used to train the model. Other 30% we will use to test our model's performance. In these data set churn is our feature we will try to predict. As we have dominance of one class (non-churn – 73.43%) over the other (churn – 26.57%) we should divide our data in training and test sets properly. We assume data set represents true distribution of this variable. Thus, we may want to divide data in such way that we keep the same proportion of churn and non-churn customers in both sets. We implement stratification, so we will not be caught by situation where we have disproportionally large share of churn customers in one set and nearly absence of them in another.

Model's selection rationale

We choose to use random forest model for this prediction problem for several reasons. First, such type of model usually demonstrates good performance in classification problems. Second, random forest model does not depend on variables' distribution. We do not need to satisfy those assumptions. And the most important advantage is the fact that this model does not suffer from overfitting if we set the only one parameter to recommended value (number of randomly selected variables at each split – square root of total number of variables). Random forest model is based on decision trees. Decision tree is series of questions with binary answers (yes / no), which in the end will lead to a certain prediction. We can think of random forest as ensemble of individual decision trees. Relatively low correlation or its absence between decision trees is key point in random forest model. This gives us another advantage of this model: many uncorrelated individual models gathered as ensemble perform better. By implementing random forest, we effectively build some number of decision trees, which use different samples. Then we split nodes in each tree considering number of randomly selected features. In the last step we obtain ensembled forecast.

Model's specification

For random forest model we must pay attention to one important parameter. We must choose optimal number of randomly selected features at each node split. The idea behind this process is to get least possible node's impurity. This means that the model could definitely classify each observation by using that set of variables. The rule of thumb is to use square root of total number of features for this parameter for classification problem. Random forest function in R sets this parameter's value using that rule by default. We use tuning tool in R package to find optimal value for it (effectively, we cross-validate parameter). By using tuneRF function we find that optimal number of randomly selected variables is 4 (Figure 1). Criteria for optimal value selection is out-of-bag error. We calculate this error by using observations in the training set, which we did not use for model's estimation. This error is just percentage of observations, which model did not predict correctly. We select number of features which leads to minimum value of this error. On Figure 1 we see average out-of-bag error for each number of features as the result of cross-validation. By using rule of thumb, we would get approximately the same result as the total number of features is 19.

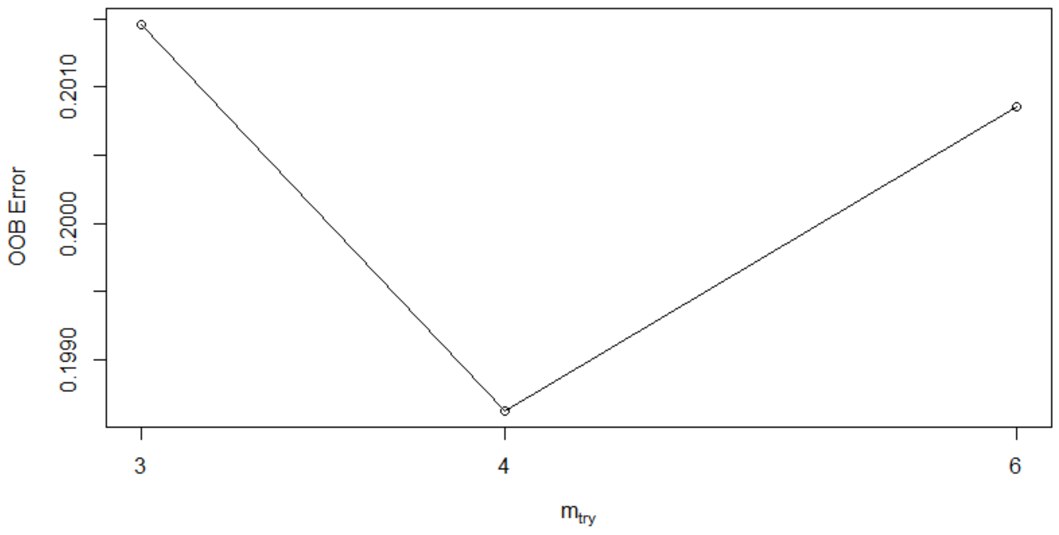


Figure 1. Number of randomly selected features optimization

Another parameter of our interest is number of trees. This parameter differs from previous one in the sense that the more trees we grow the better performance model shows. The only cost of increasing this parameter is computational time. This does not lead to any noticeable slowdown in model's estimation for our problem's dimensionality. Thus, we set this parameter to be equal to 1000. These are two parameters, which usually researches pay attention to. But we deal with imbalanced classification problem. Number of churned customers is much less then number of retained customers. For us it is more important to predict churn class than non-churn. In other words, we consider misclassification of non-churn class less harmful than misclassification of churn class. We can think of it as an over optimistic forecast of company's revenue. When in fact greater number of churned clients will result in less revenue. We should let the model know that despite disproportion between classes, churn class should prevail. We implement this by using "strata" optional argument in random forest function in R. The next step is to identify the proportion by using which model will draw a sample from training set. We choose this to be 45 and 55 respectively. This means that model will draw samples where number of churn class will be greater than non-churn. We estimate such model and obtain the following result:

```
random_forest<-
randomForest(x=x1,y=y1,ntree=1000,mtry=4,sampsize=c(45,55),strata=y1,metric="Accuracy",replace=T)

Call:

randomForest(x = x1, y = y1, ntree = 1000, mtry = 4, replace = T,     strata = y1, sampsize = c(45, 55), metric
= "Accuracy")

Type of random forest: classification

Number of trees: 1000

No. of variables tried at each split: 4

OOB estimate of  error rate: 26.89%

Confusion matrix:      No       Yes   class.error

                 No   2513 1102   0.3048409

                 Yes   222    1087    0.1695951
```

We reach out-of-bag error of 26.86%. For each class errors are 30.48% and 16.96% respectively. Then we run estimated model to obtain prediction for test data and get the following confusion matrix:

```
testLabel    No    Yes

        No  1078  470

        Yes   90    470
```

We can easily derive corresponding errors: 26.57% for out-of-bag error, 30.36% for non-churn class, 16.07% for churn class. As we can see model demonstrates almost the same performance for both training and test sets. For comparison let's see what would have happened if we did not account for disproportion between classes:

```
random_forest1 <- randomForest(x = x1,y = y1, ntree = 1000, mtry = 4,metric="Accuracy",replace=T)

Call:

 randomForest(x = x1, y = y1, ntree = 1000, mtry = 4, replace = T,      metric = "Accuracy")

Type of random forest: classification
```

```
              Number of trees: 1000
No. of variables tried at each split: 4
OOB estimate of  error rate: 19.88%
Confusion matrix:    No      Yes     class.error
                  No   3275 340   0.09405256
                  Yes   639  670   0.48815890
```

This model has lower OOB error, but the error for churn class is 48.82%. It is nearly 3 times higher than for our selected model (16.07%). By taking into consideration class disproportion we reduced error by 66.67% for churn class and worsen our out-of-bag error only by approximately 25%. For test data it shows almost the same difference in confusion matrix:

```
testLabel      No    Yes
            No  1391  157
          Yes    286    274
```

After we estimated model and checked its performance on test data, we may be interested in features importance for model (Figure 2):

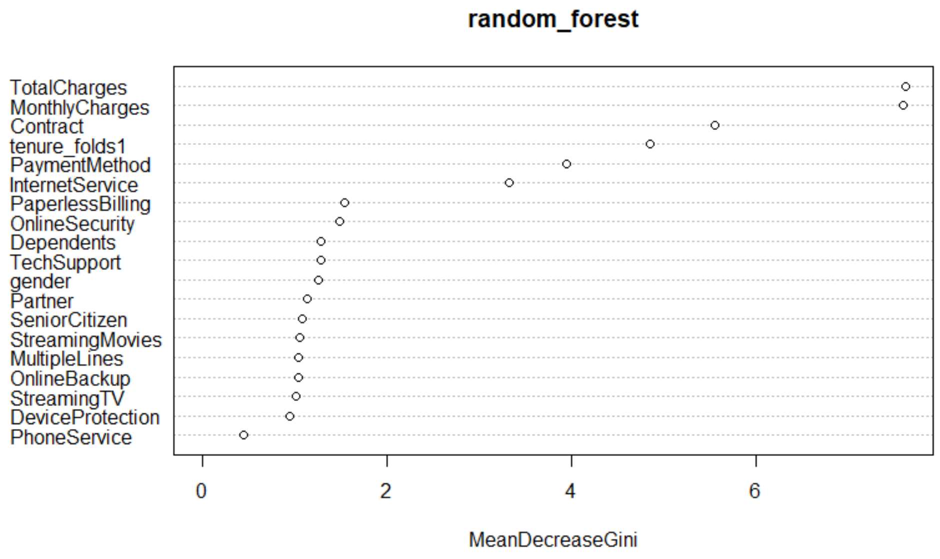


Figure 2. Feature importance

In the graph above we see features and mean decrease in Gini. We use Gini to measure node impurity:

$$G = \sum_{i=1}^{C} p(i) * (1 - p(i))$$

where p(i) is probability of picking a sample of class I, C is the total number of classes (2 in our case)

The less this measure the more precisely we can classify our observations. In this graph features are sorted by mean decrease in GINI. We can clearly see 6 most important features: total charges, monthly charges, contract, tenure_folds, payment method, internet service.

In addition, we may want to draw ROC (Receiver Operating Characteristic) curve (Figure 3):

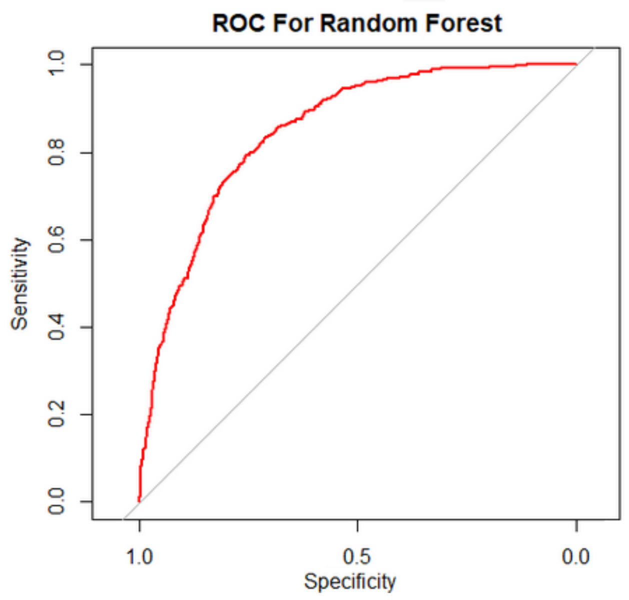


Figure 3. ROC curve

The area under this curve represents model's accuracy. In our case, the accuracy of the model is 84.78%. In fact, we increased model's accuracy by handling imbalanced classification. For plane random forest model accuracy is 82.99%.

Conclusion

In this paper we analyze data on clients of one telecom company. Data includes different clients' features and response variable. Response variable indicates customer's status – churn / non-churn. We choose random forest to predict this response variable on test data. In addition, we noticed that churn class is more important for us to predict accurately. By applying stratified sampling, we improved model's predictive accuracy for positive class (churn). Thus, we achieved classification error for churn class on test data of 16.07%. Genral performance of model is 84.78% (calculated as area under ROC curve).

References

G. James, D. Witten, T. Hastie, R. Tibshirani. "An Introduction to Statistical Learning with Applications in R". Springer Science+Business Media New York, 2013.

Appendix 1 R Script

```
###installing packages and libraries

install.packages("randomForest")

install.packages("unbalanced")

install.packages("sqldf")

install.packages("ROCR")

install.packages("caret")

install.packages("AUC")

install.packages("mlbench")

install.packages("splitstackshape")

install.packages("mlr")

install.packages("corrplot")

install.packages("gridExtra")

install.packages("ggthemes")

install.packages("party")

install.packages("e1071")


library(pROC)

library(readr)

library(e1071)

library(plyr)

library(corrplot)

library(ggplot2)

library(gridExtra)

library(ggthemes)

library(caret)

library(MASS)

library(randomForest)

library(party)
library(unbalanced)

library(sqldf)

library(MASS)

library(ROCR)

library(caret)

library(AUC)

library(mlbench)

library(splitstackshape)

library(mlr)


###load data

data <- read.csv('Customer-Churn.csv')

head(data)

str(data)


### make categorical variables factors if necessary

sapply(data, function(x) sum(is.na(x)))

data <- data[complete.cases(data),]


unique(data['OnlineSecurity'])

f2_col <- c(10:15)

for (i in 1:ncol(data[, f2_col])) {

  data[, f2_col][, i] <- as.factor(mapvalues(data[, f2_col][, i], from = c("No internet service"), to =
c("No")))

}


data$MultipleLines <- as.factor(mapvalues(data$MultipleLines, from = c("No phone service"), to =
c("No")))

data$SeniorCitizen <- as.factor(mapvalues(data$SeniorCitizen, from = c("0", "1"), to = c("No", "Yes")))
```

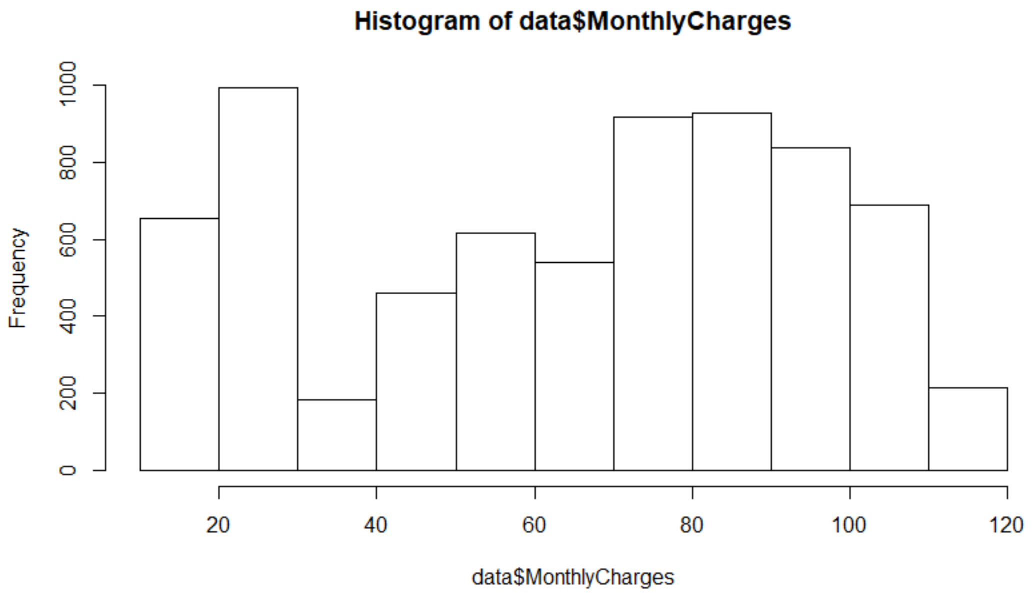
```
#may be create even folds with other intervals
tenure_folds <- function(tenure){
  if(tenure >= 0 & tenure <= 12){
    return('0 - 12 Month')
  }else if(tenure > 12 & tenure <= 24){
    return('12 - 24 Month')
  }else if(tenure > 24 & tenure <= 36){
    return('24 - 36 Month')
  }else if(tenure > 36 & tenure <= 50){
    return('36 - 60 Month')
  }else if(tenure > 60){
    return('> 60 Month')
  }
}

data$tenure_folds1 <- sapply(data$tenure, tenure_folds)
data$tenure_folds1 <- as.factor(data$tenure_folds1)

### EDA
hist(data$MonthlyCharges)
hist(data$TotalCharges)
hist(data$tenure)
summary(data)
 set.seed(100)


### split dataset
train.index <- createDataPartition(data$Churn, p = .7, list = FALSE)
```

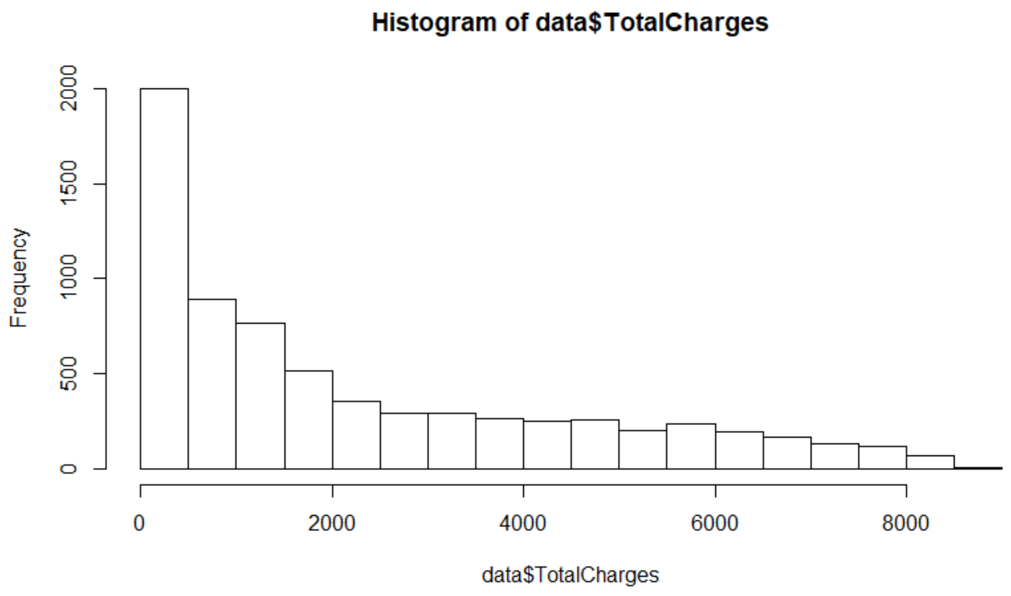
```
train <- data[ train.index,]

test  <- data[-train.index,]

trainLabel <- data$Churn[train.index]

testLabel <- data$Churn[-train.index]


summary(train$Churn)

summary(test$Churn)


### drop unneccessary columns

train$customerID <- NULL

train$tenure <- NULL

#train$TotalCharges<-NULL

test$customerID <- NULL

test$tenure <- NULL

#test$TotalCharges<-NULL


str(train)

str(test)


cols <- c(1:18, 20)

x1<-train[,cols]

y1<-train[,19]


### tune parameters on train data set

mtry_opt <- tuneRF(x1, y1, stepFactor=1.5, improve=1e-5, ntree=1000)

print(mtry_opt)


random_forest <- randomForest(x = x1,y = y1, ntree = 1000, mtry = 4,
sampsize=c(45,55),strata=y1,metric="Accuracy",replace=T)
```

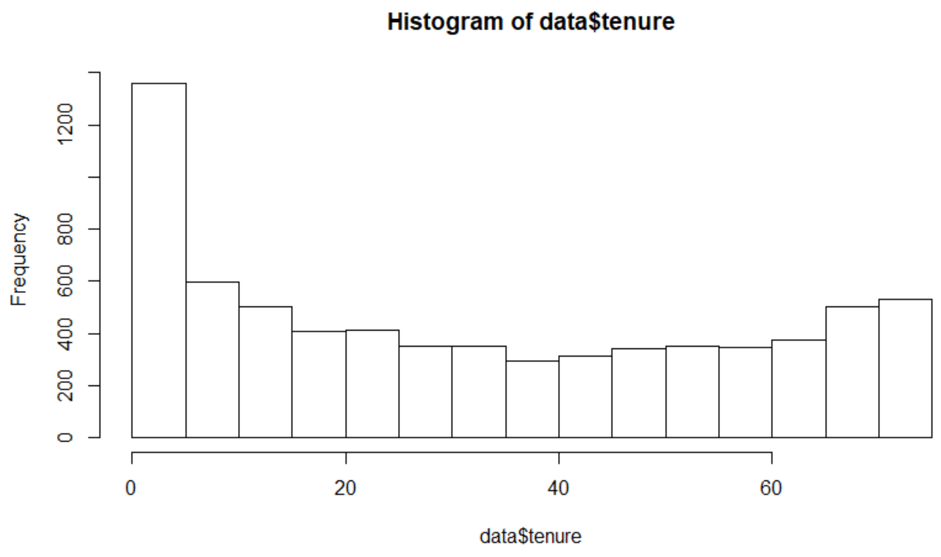
```
random_forest1 <- randomForest(x = x1,y = y1, ntree = 1000, mtry = 4,metric="Accuracy",replace=T)

### make prediction on test data set
print(random_forest)
print(random_forest1)
plot(random_forest)
table(testLabel, predict(object = random_forest, newdata = test))
 table(testLabel, predict(object = random_forest1, newdata = test))
 varImpPlot(random_forest)

### plot ROC
rf_prediction <- predict(random_forest, test, type="prob")
ROC_rf <- roc(test$Churn, rf_prediction[,2])
plot(ROC_rf, col = "red", main = "ROC For Random Forest")
print(ROC_rf)

rf_prediction1 <- predict(random_forest1, test, type = "prob")
ROC_rf1 <- roc(test$Churn, rf_prediction1[,2])
plot(ROC_rf1, col = "red", main = "ROC For Random Forest")
print(ROC_rf1)
```

Appendix 2 Data Summary

```
> summary(data)
      customerID       gender     SeniorCitizen Partner     Dependents     tenure      PhoneService MultipleLines
 0002-ORFBO:   1   Female:3483   No :5890      No :3639   No :4933   Min.   : 1.00   No : 680     No :4065
 0003-MKNFE:   1   Male  :3549   Yes:1142      Yes:3393   Yes:2099   1st Qu.: 9.00   Yes:6352     Yes:2967
 0004-TLHLJ:   1                                                     Median :29.00
 0011-IGKFF:   1                                                     Mean   :32.42
 0013-EXCHZ:   1                                                     3rd Qu.:55.00
 0013-MHZWF:   1                                                     Max.   :72.00
 (Other)   :7026
    InternetService OnlineSecurity OnlineBackup DeviceProtection TechSupport StreamingTV StreamingMovies           Contract
 DSL        :2416   No :5017       No :4607     No :4614          No :4992    No :4329    No :4301        Month-to-month:3875
 Fiber optic:3096   Yes:2015       Yes:2425     Yes:2418          Yes:2040    Yes:2703    Yes:2731        One year      :1472
 No         :1520                                                                                         Two year      :1685


 PaperlessBilling                   PaymentMethod MonthlyCharges    TotalCharges    Churn            tenure_folds1
 No :2864         Bank transfer (automatic):1542   Min.   : 18.25   Min.   :  18.8   No :5163   > 60 Month   :1407
 Yes:4168         Credit card (automatic)  :1521   1st Qu.: 35.59   1st Qu.: 401.4   Yes:1869   0 - 12 Month :2175
                  Electronic check         :2365   Median : 70.35   Median :1397.5              12 - 24 Month:1856
                  Mailed check             :1604   Mean   : 64.80   Mean   :2283.3              24 - 36 Month: 762
                                                   3rd Qu.: 89.86   3rd Qu.:3794.7              36 - 60 Month: 832
                                                   Max.   :118.75   Max.   :8684.8
```




Histogram of data$TotalCharges
Frequency
0
500
1000
1500
2000
0
2000
4000
6000
8000
data$TotalCharges


Histogram of data$tenure
Frequency
0
200
400
600
800
1200
0
20
40
60
data$tenure