Risco de Crédito

luiz felipe

8/11/2020

Mini Projeto 4 - Risco de Crédito

O projeto trata de fazer uma análise de crédito para conceder ou não crédito a um determinado cliente.

Etapa 1 - Coletando dados

Ler arquivo csv

```
# Coletando dados
risk <- read.csv("credit_dataset.csv", header = TRUE)
```

Etapa 2 - Criando função para separar dados categóricos de quantitativos

```
# Separando variáveis categóricas pelo número de de elementos únicos
categoricalData <- function(dataFrame, greaterLevel) {
  vectorTypes <- sapply(dataFrame,
                        function(x) ifelse(max(x) >= greaterLevel,
                                           FALSE, TRUE))
  return(vectorTypes)
}

# Separando variáveis quantitativas pelo número de elementos únicos
quantitativeData <- function(dataFrame, greaterLevel) {
  vectorTypes <- sapply(dataFrame,
                        function(x) ifelse(max(x) >= greaterLevel,
                                           TRUE, FALSE))
  return(vectorTypes)
}

categoricas <- categoricalData(risk, 5)
quantitativas <- quantitativeData(risk, 5)
```

Etapa 3 - Verificando se há dados faltantes

```
# Existem valores NA
sapply(risk, function(x) sum(is.na(x)))
```

```
##                  credit.rating                 account.balance
##                              0                               0
##         credit.duration.months  previous.credit.payment.status
##                              0                               0
##                 credit.purpose                   credit.amount
```

```
##                             0                             0
##                       savings           employment.duration
##                             0                             0
##              installment.rate                marital.status
##                             0                             0
##                    guarantor            residence.duration
##                             0                             0
##                current.assets                           age
##                             0                             0
##                 other.credits               apartment.type
##                             0                             0
##                  bank.credits                    occupation
##                             0                             0
##                    dependents                     telephone
##                             0                             0
##                foreign.worker
##                             0
```

Etapa 4 - Novo data frame com fator e quantitativas

```
# Transformando variáveis categóricas em fatores
ToFactor <- function(dataFrame) {
  listaVectors <- lapply(dataFrame, factor)
  df <- as.data.frame(listaVectors)
  return(df)
}

riskCat <- ToFactor(risk[,categoricas])
riskQuant <- risk[,quantitativas]

# Criando uma nova coluna com a idade elevada ao quadrado para verificar sua influência no modelo
risK <- cbind(riskCat, riskQuant)
risK$age2 <- risK$age^2
```

Etapa 5 - Normalizando dados quantitativos

```
# Normalizando variáveis quantitativas
Normalizar <- function(x) {
  x <- (x - min(x))/(max(x) - min(x))
}

riskQuant <- as.data.frame(lapply(riskQuant, Normalizar))
riskCat <- ToFactor(risk[,categoricas])

# Data frame com variáveis tipo fator e quantitativas normalizadas
RiskNeural <- cbind(riskCat, riskQuant)
```

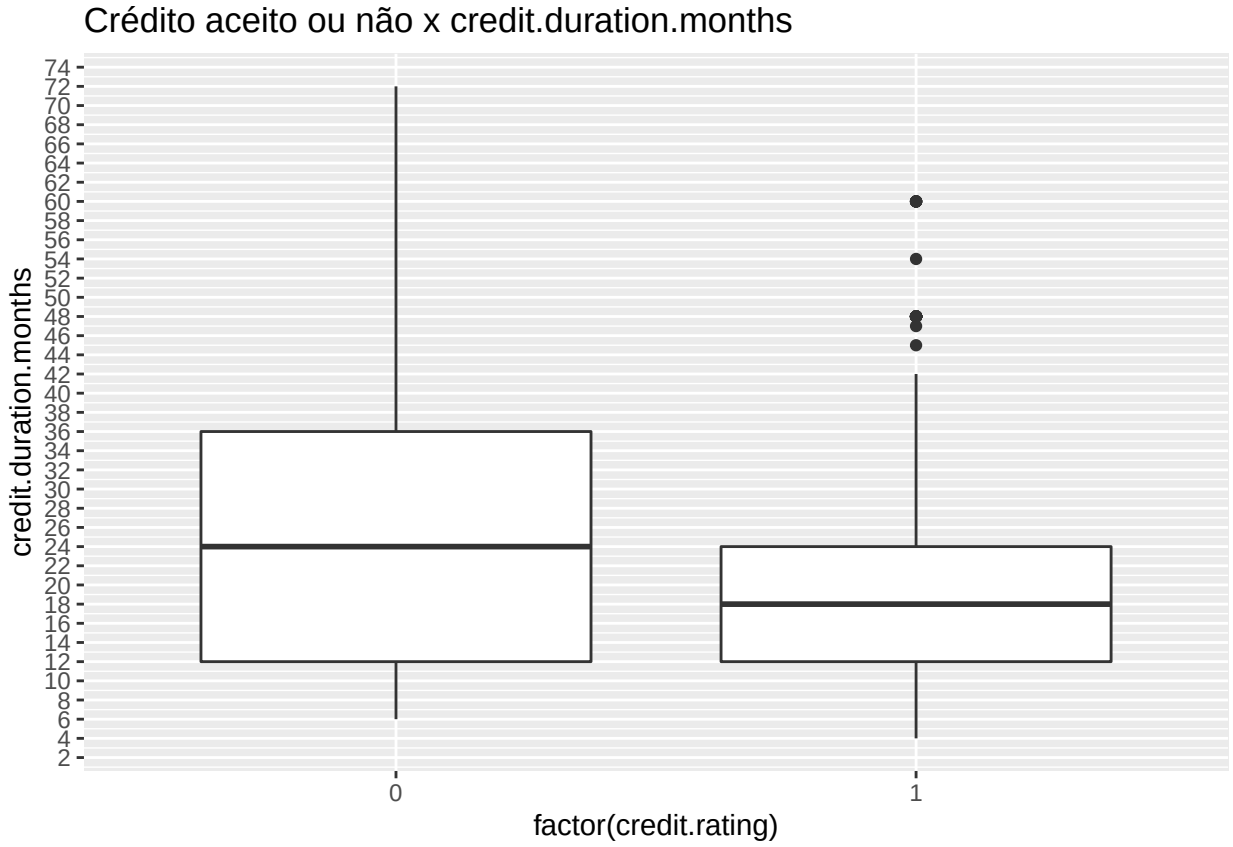
Etapa 6 - Plotando gráficos

```
# Plotando gráficos de variáveis quantitativas
library(ggplot2)
boxplotQuantitative <- function(dataFrame, vectorQuantitativeTrue) {
  df <- dataFrame[,vectorQuantitativeTrue]
```

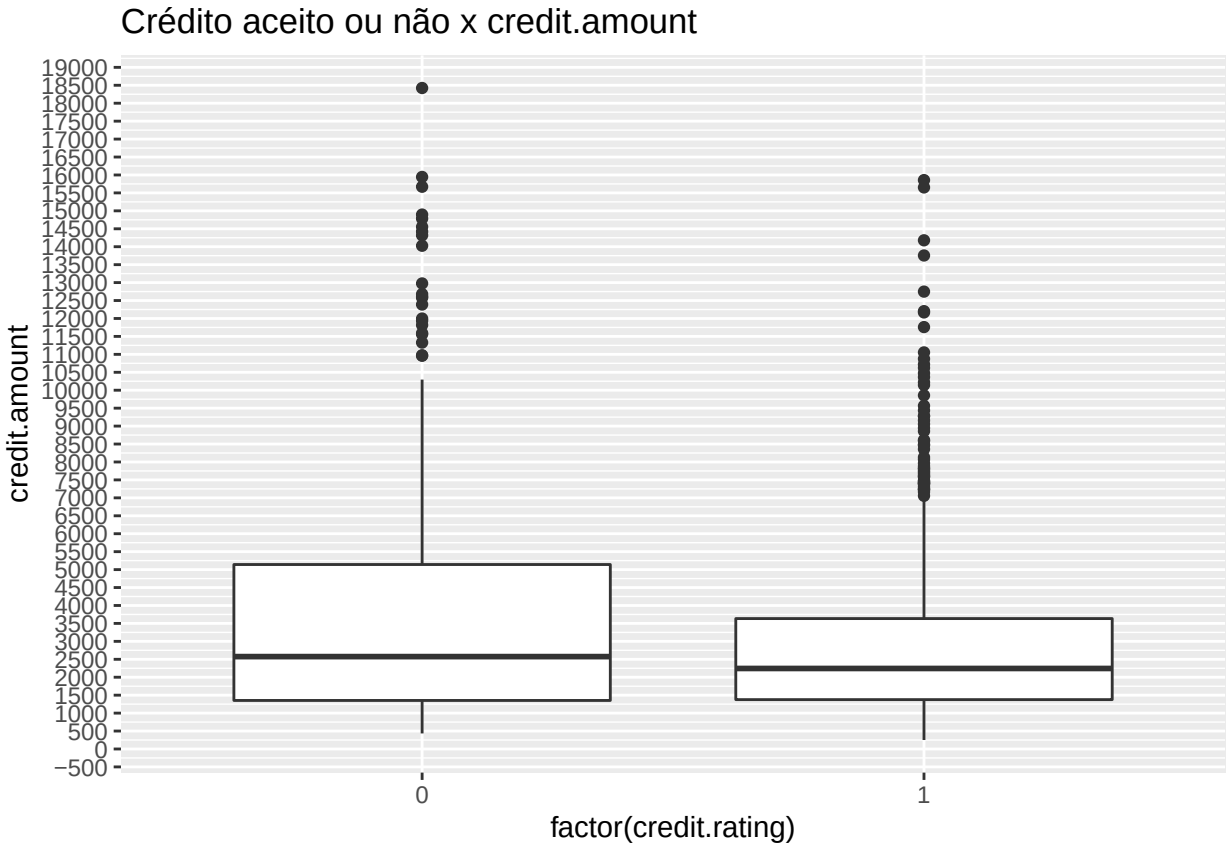
```
  plot.graphic <- list()
  for(i in 1:ncol(df)) {
    plot.graphic[[i]] <- ggplot(data = risk, mapping = aes(x = factor(credit.rating), group = factor(cr
      geom_boxplot(aes_string(y = names(df)[i])) +
      scale_y_continuous(breaks = scales::extended_breaks(50)) +
      ggtitle(paste("Crédito aceito ou não x", sep = " ", names(df)[i]))
  }
  return(plot.graphic)
}

boxplotQuantitative(risk, quantitativas)
```

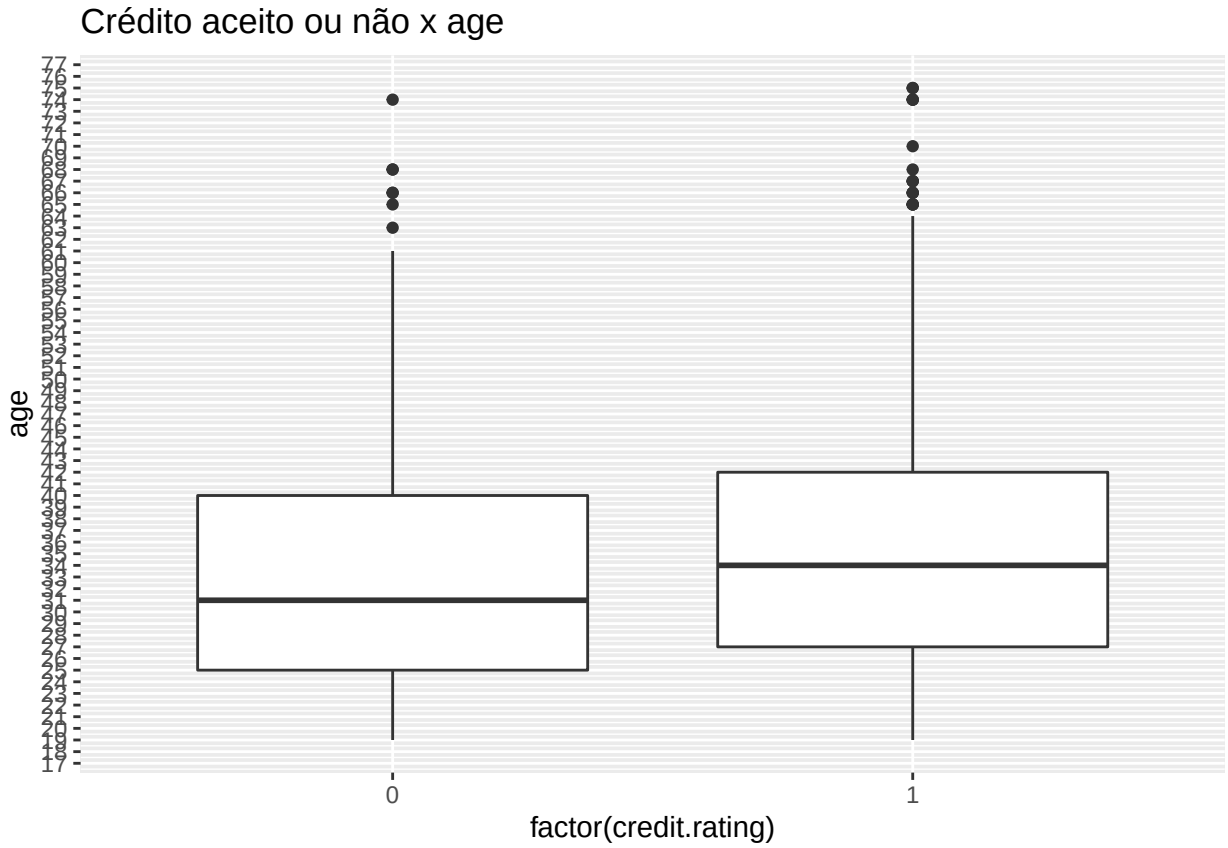
```
## [[1]]
```




```
##
## [[2]]
```


Crédito aceito ou não x credit.amount
credit.amount
19000
18500
18000
17500
17000
16500
16000
15500
15000
14500
14000
13500
13000
12500
12000
11500
11000
10500
10000
9500
9000
8500
8000
7500
7000
6500
6000
5500
5000
4500
4000
3500
3000
2500
2000
1500
1000
500
0
−500
0
1
factor(credit.rating)

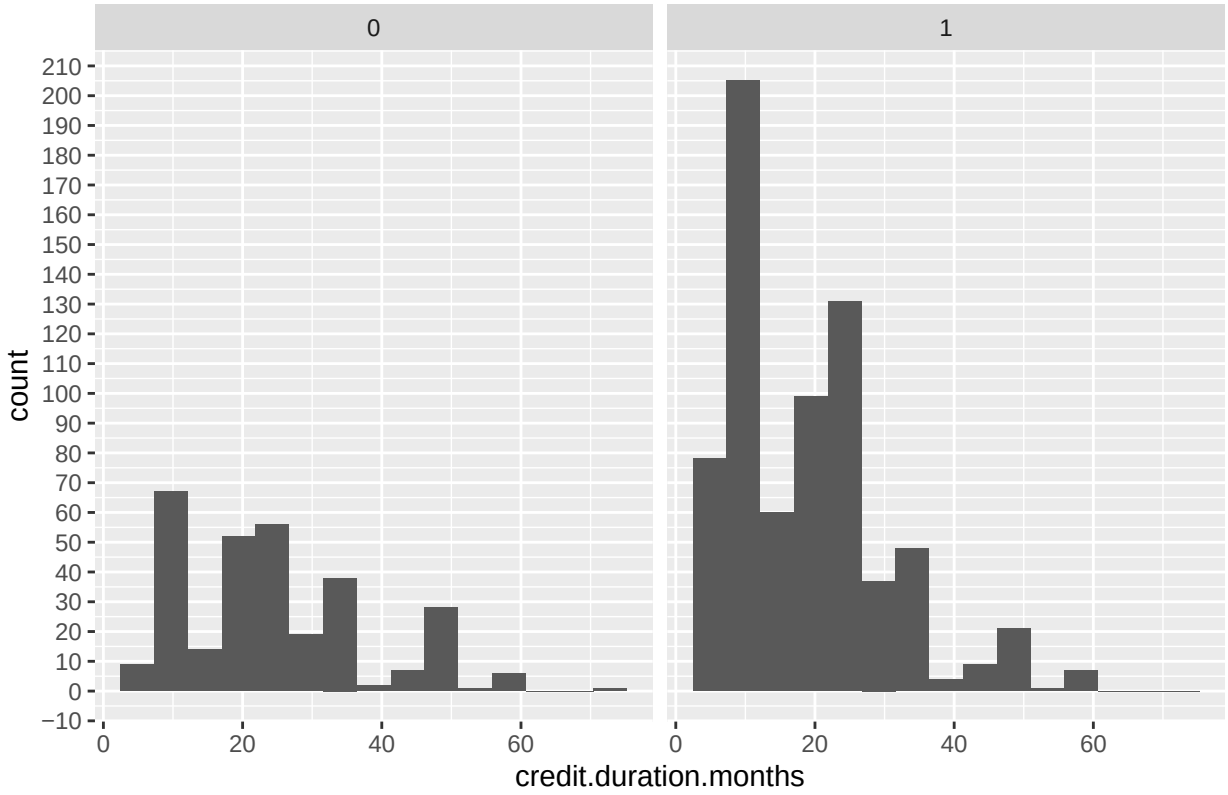
```
##
## [[3]]
```



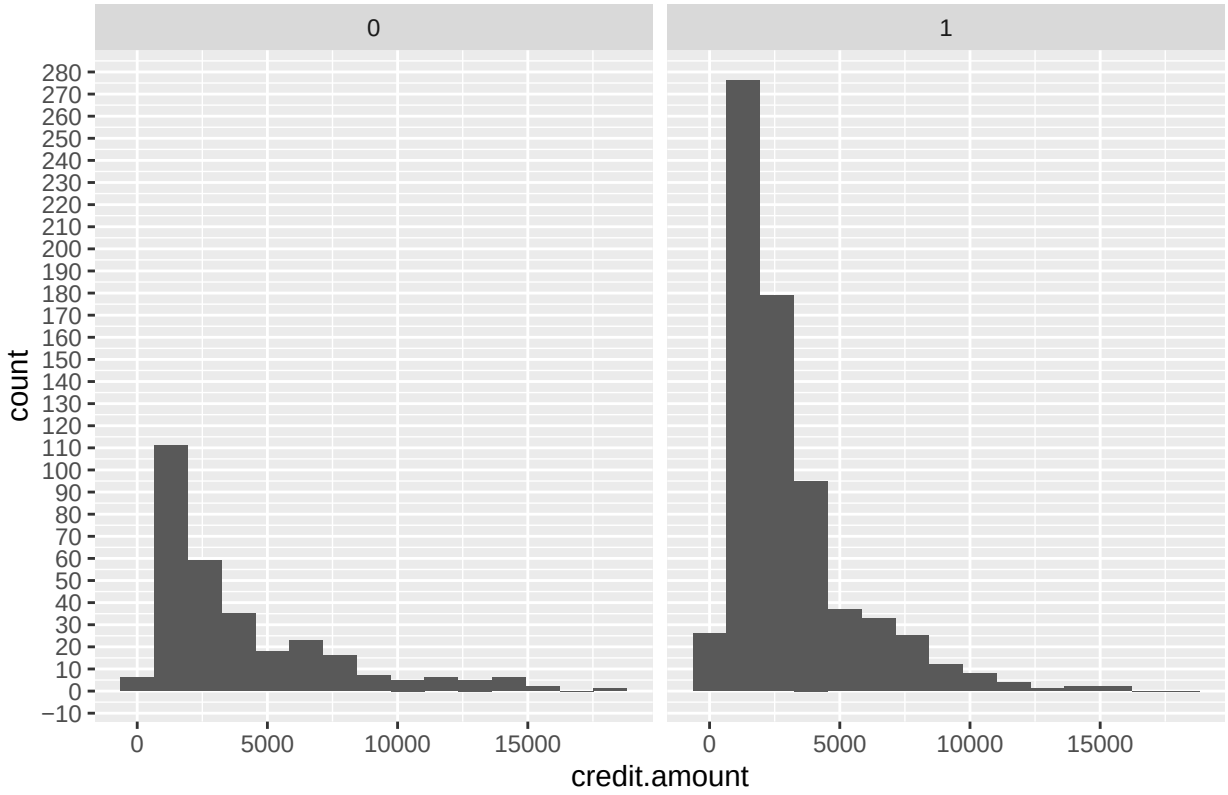

```
histogramQuantitative <- function(dataFrame, vectorQuantitativeTrue) {
  df <- dataFrame[,vectorQuantitativeTrue]
  plot.graphic <- list()
  for(i in 1:ncol(df)) {
    plot.graphic[[i]] <- ggplot(data = risk) +
      geom_histogram(aes_string(x = names(df)[i]), bins = 15) +
      facet_grid(cols = vars(credit.rating)) +
      scale_y_continuous(breaks = scales::extended_breaks(30)) +
      ggtitle(paste("Histograma de", sep = " ", names(df)[i]))
  }
  return(plot.graphic)
}

histogramQuantitative(risk, quantitativas)
```

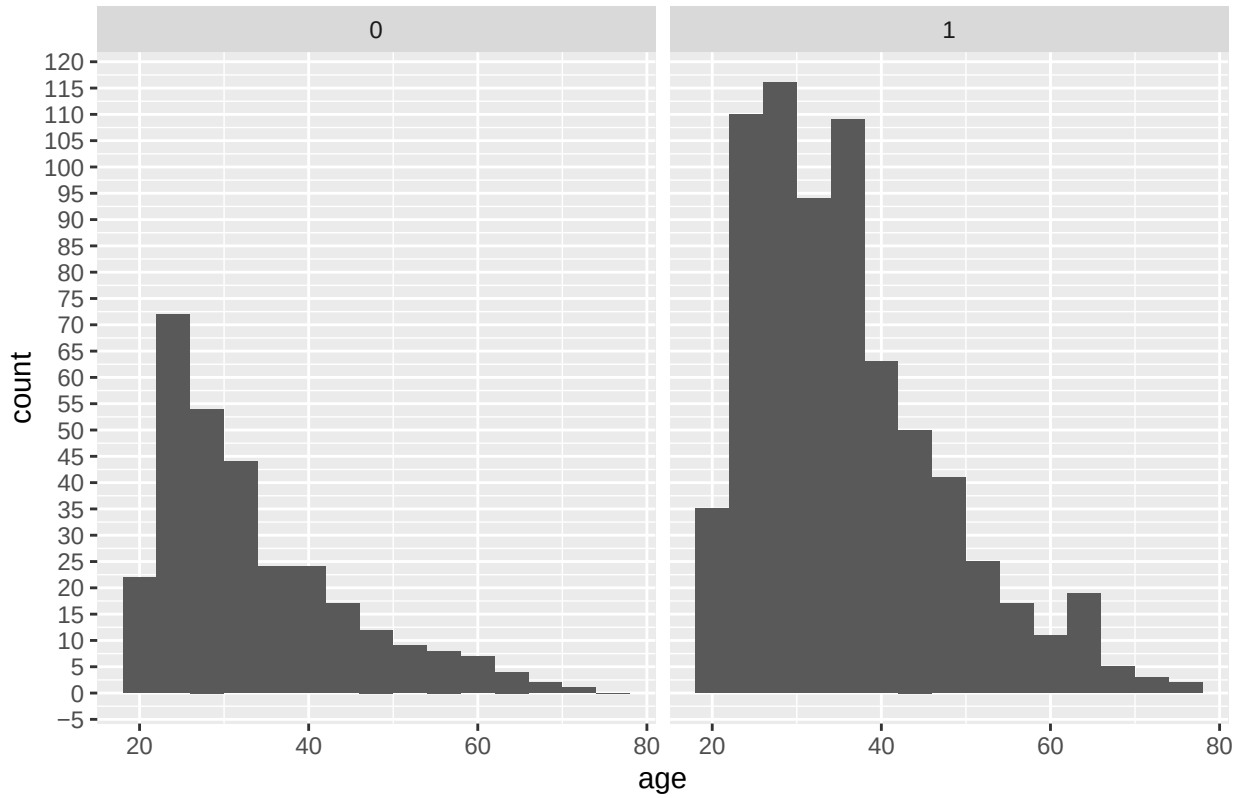
```
## [[1]]
```


Histograma de credit.duration.months
0
1
count
credit.duration.months
210
200
190
180
170
160
150
140
130
120
110
100
90
80
70
60
50
40
30
20
10
0
−10
0
20
40
60

```
##
## [[2]]
```


Histograma de credit.amount
0
1
count
credit.amount
0
5000
10000
15000

```
##
## [[3]]
```




```
# Plotando gráficos de variáveis categóricos
table(risk$credit.rating)
```

```
##
##   0   1
## 300 700
```

```
barsCategorical <- function(dataFrame, vectorQuantitativeTrue) {
  df <- dataFrame[,vectorQuantitativeTrue]
  plot.graphic <- list()
  for(i in 1:ncol(df)) {
    if(names(df)[i] != "credit.rating") {
      plot.graphic[[i]] <- ggplot(data = df, aes(fill = factor(credit.rating))) +
        geom_bar(aes_string(names(df)[i]), position = position_dodge()) +
        scale_y_continuous(breaks = scales::extended_breaks(30)) +
        ggtitle(paste("Gráfico de barras de", sep = " ", names(df)[i]))
    }
    else {
      plot.graphic[[i]] <- ggplot(data = df) +
        geom_bar(aes_string(names(df)[i])) +
        scale_y_continuous(breaks = scales::extended_breaks(30)) +
        ggtitle(paste("Gráfico de barras de", sep = " ", names(df)[i]))
    }
  }
  return(plot.graphic)
}

barsCategorical(risk, categoricas)
```

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

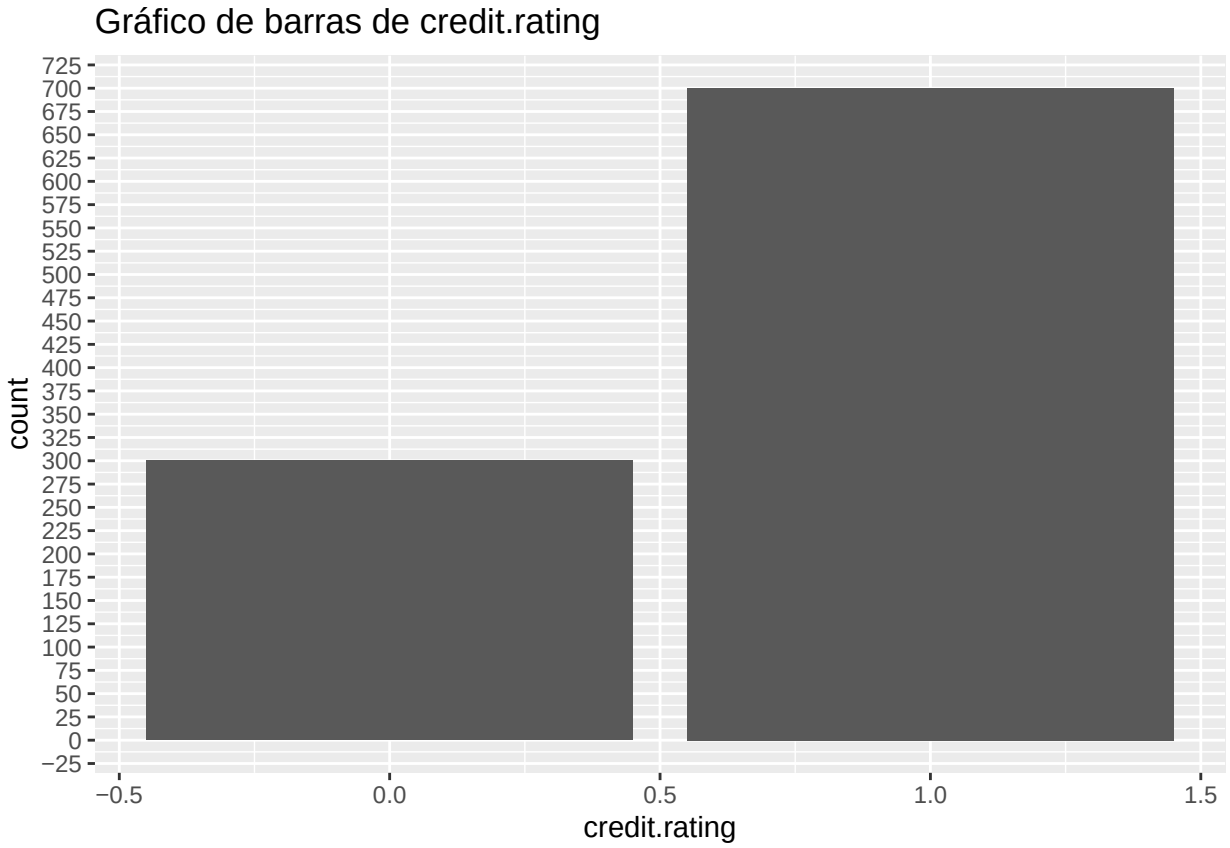


Gráfico de barras de credit.rating
count
725
700
675
650
625
600
575
550
525
500
475
450
425
400
375
350
325
300
275
250
225
200
175
150
125
100
75
50
25
0
−25
−0.5
0.0
0.5
1.0
1.5
credit.rating

```
##
## [[2]]
```

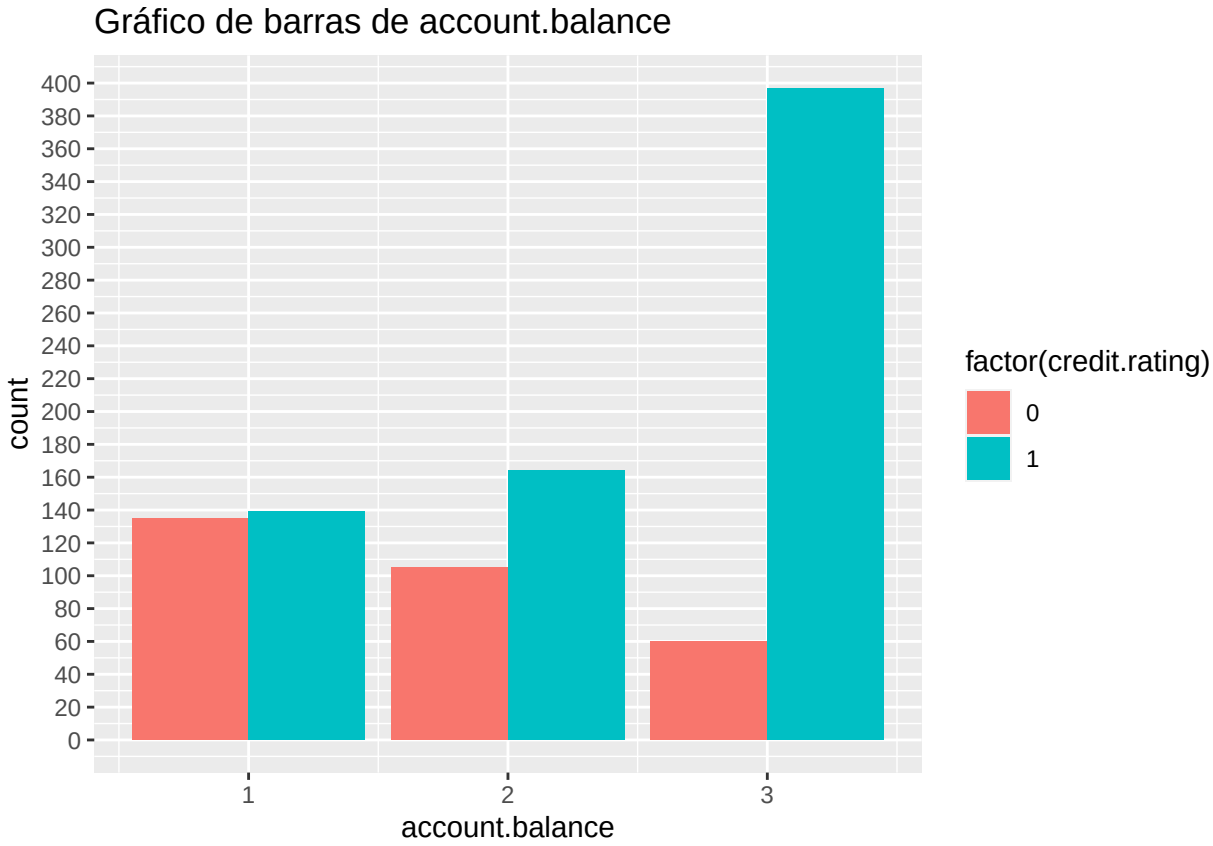


Gráfico de barras de account.balance
count
account.balance
factor(credit.rating)
0
1
400
380
360
340
320
300
280
260
240
220
200
180
160
140
120
100
80
60
40
20
0
1
2
3

```
##
## [[3]]
```

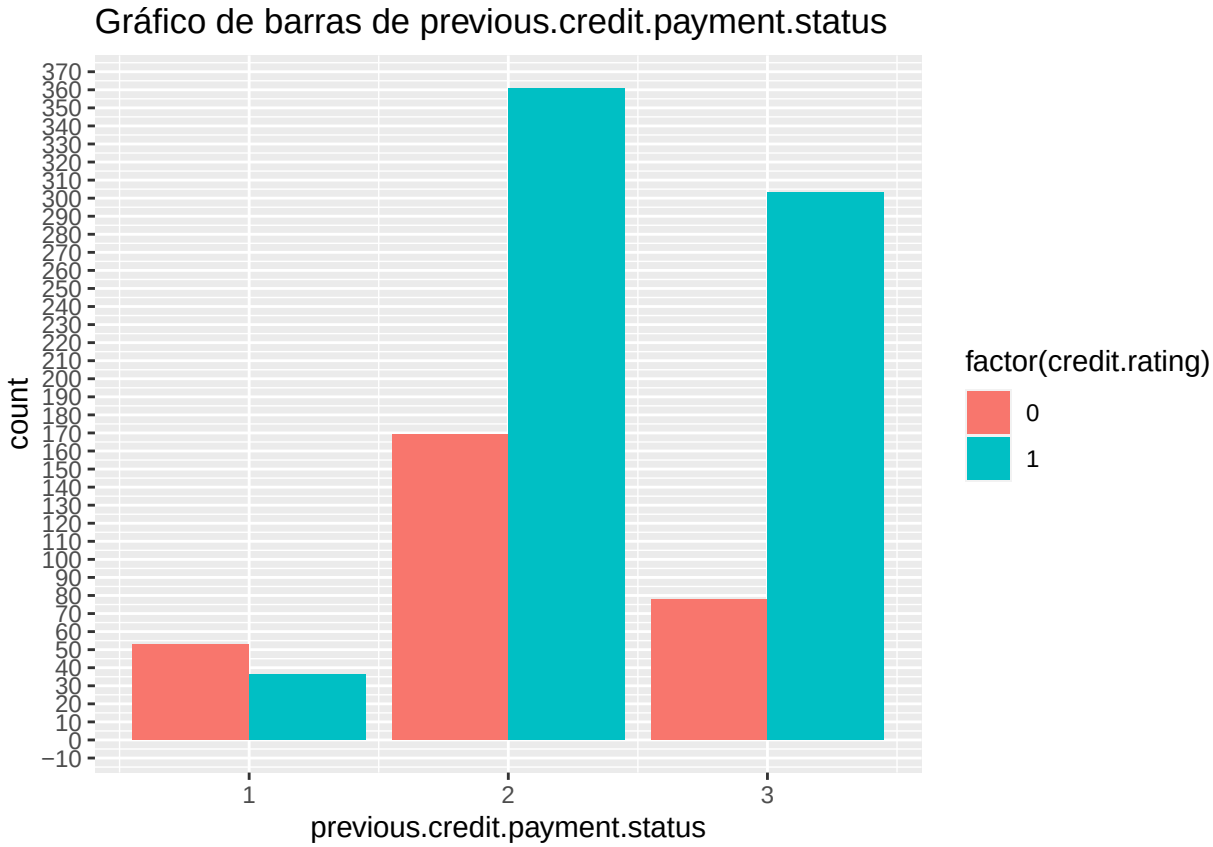


Gráfico de barras de previous.credit.payment.status
count
previous.credit.payment.status
factor(credit.rating)
0
1

```
##
## [[4]]
```

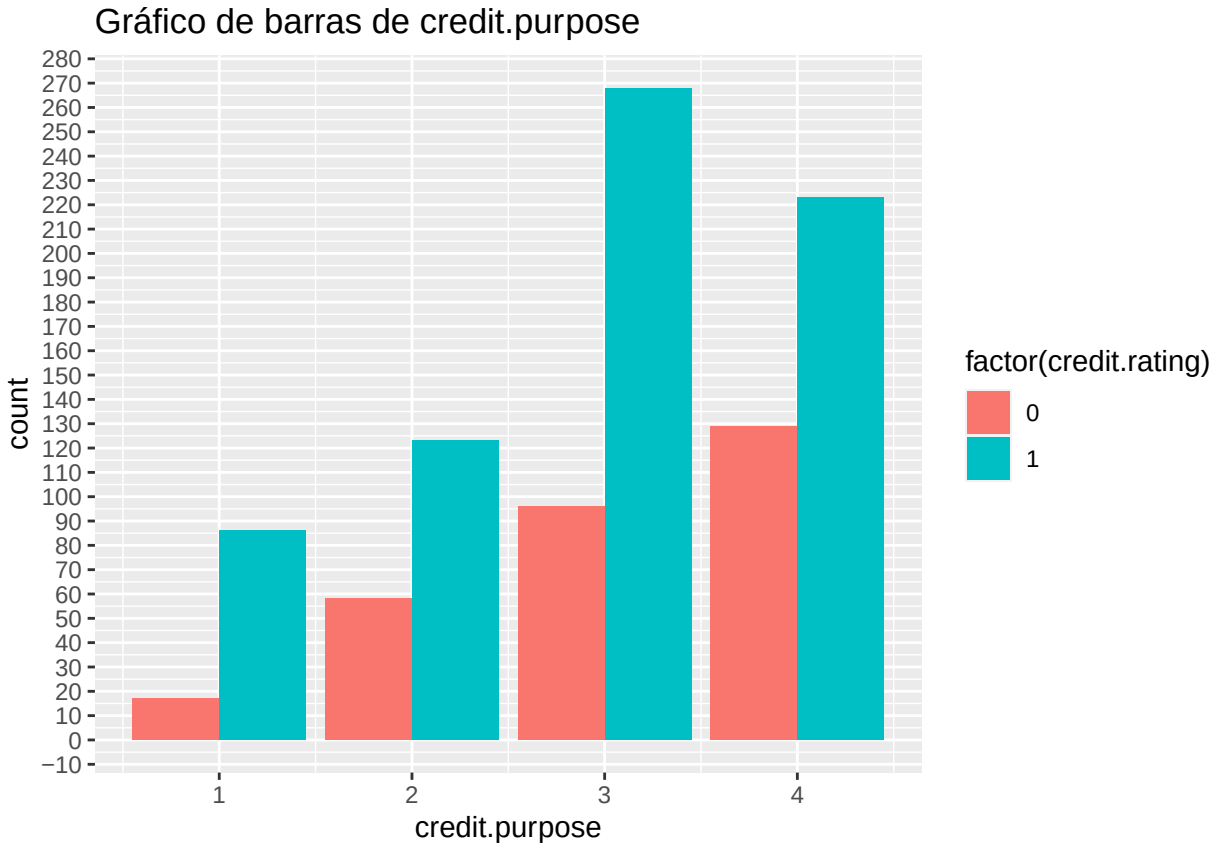


Gráfico de barras de credit.purpose
count
credit.purpose
factor(credit.rating)
0
1

```
##
## [[5]]
```

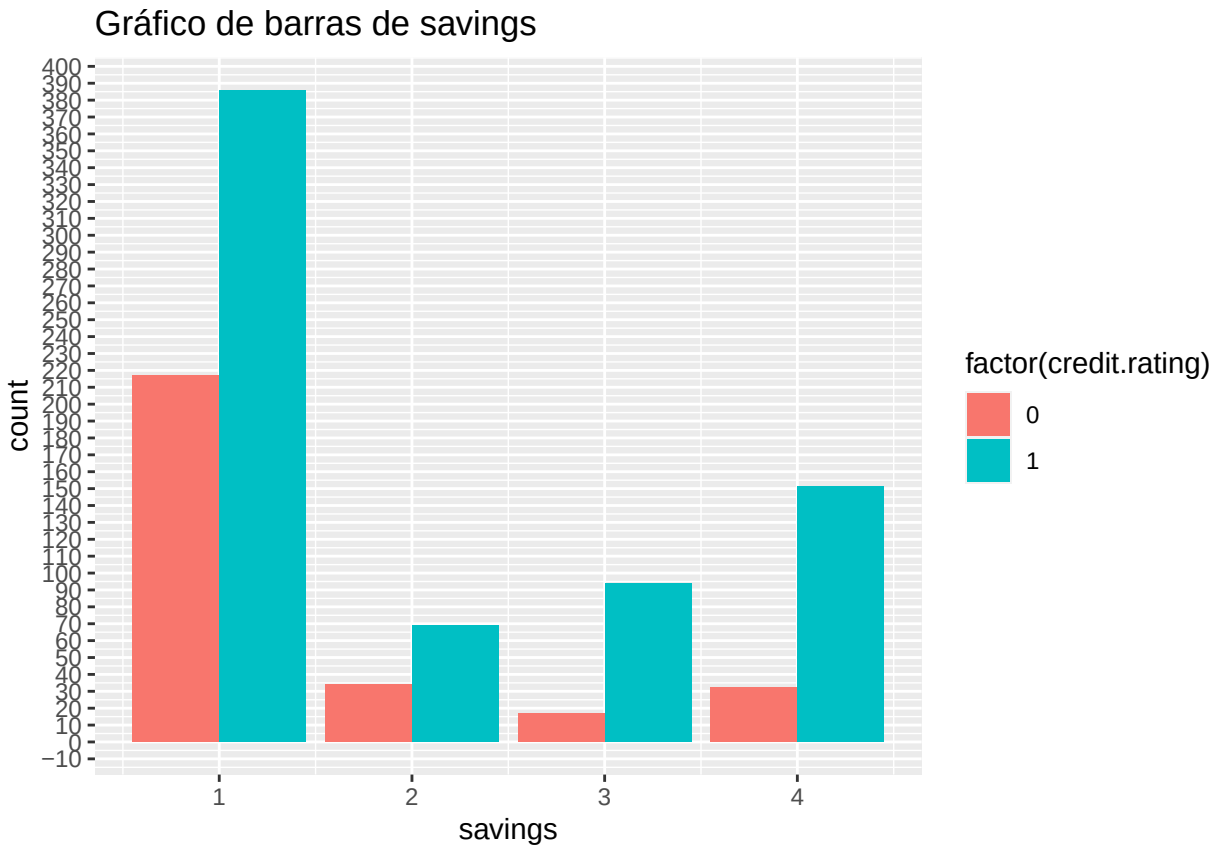


Gráfico de barras de savings
count
savings
1
2
3
4
factor(credit.rating)
0
1

```
##
## [[6]]
```

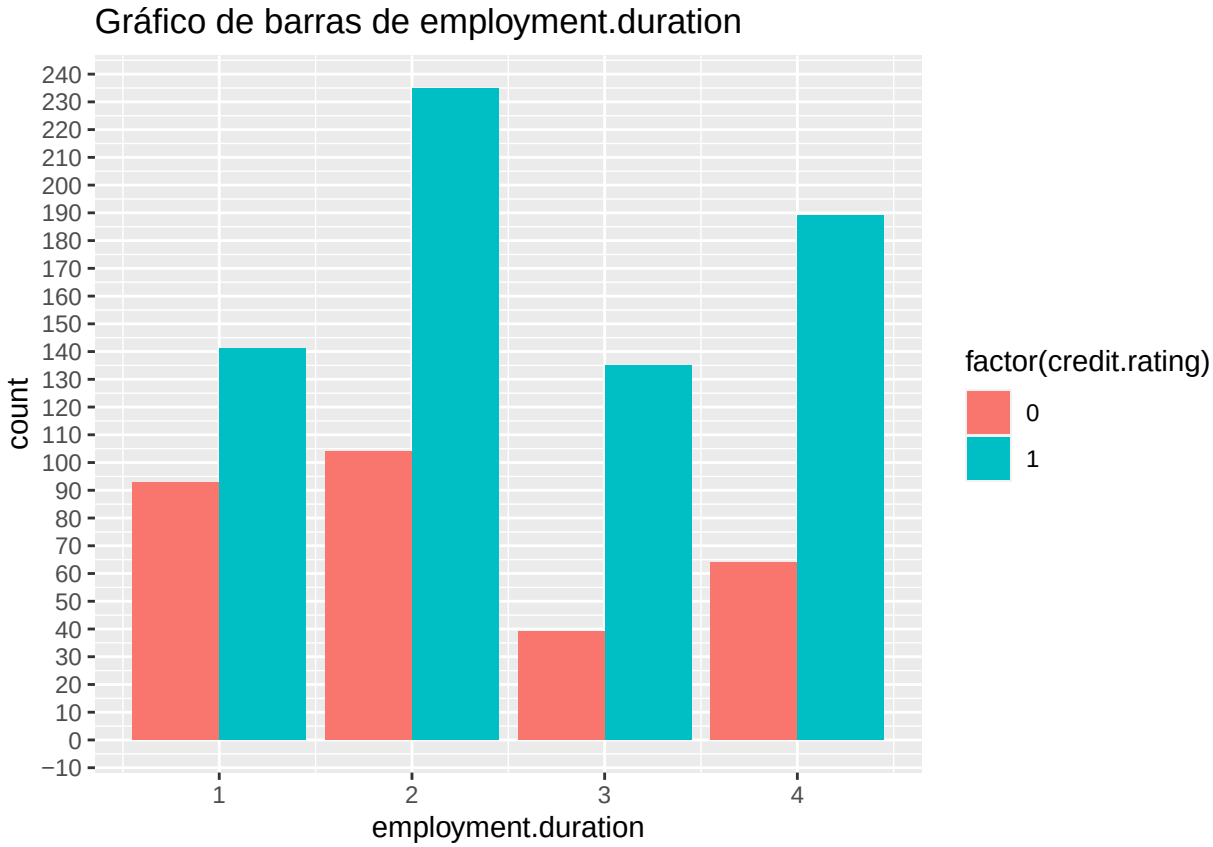


Gráfico de barras de employment.duration
count
employment.duration
factor(credit.rating)
0
1

```
##
## [[7]]
```

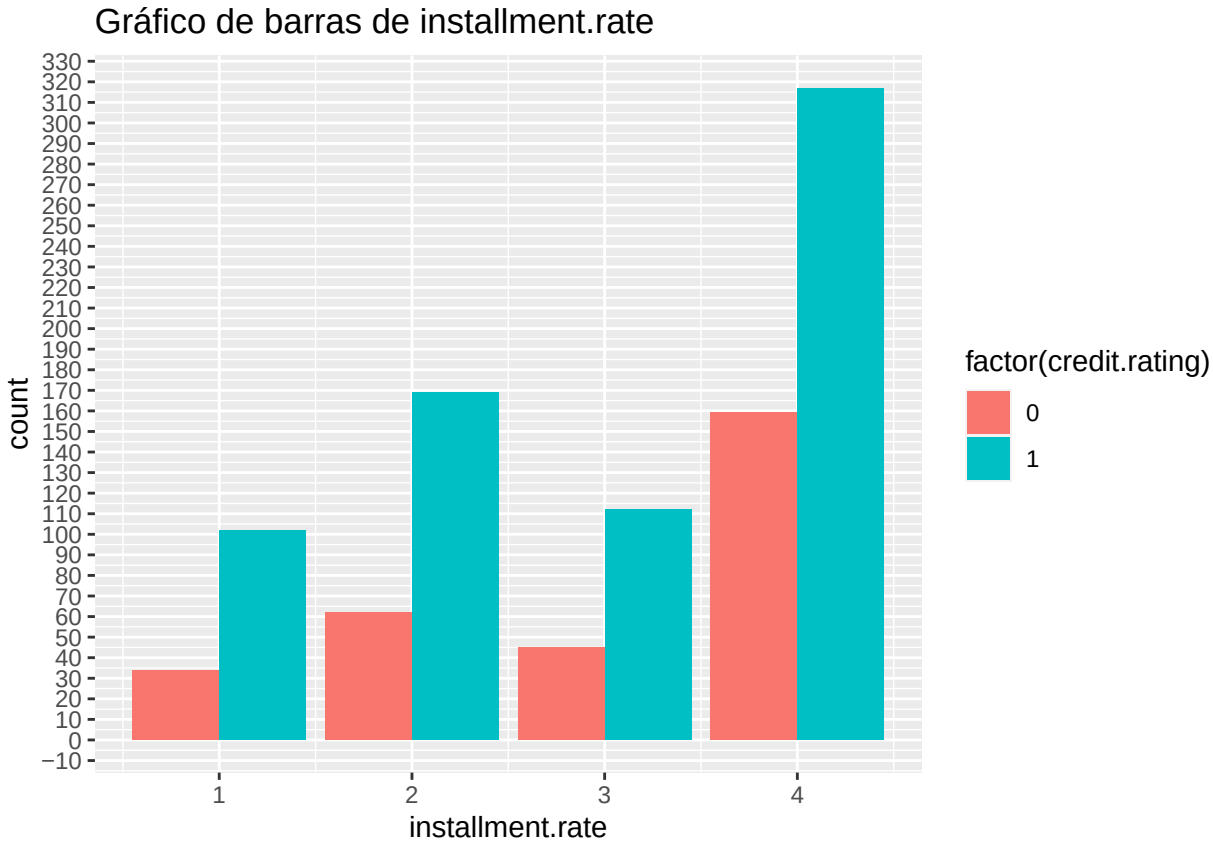


Gráfico de barras de installment.rate
count
installment.rate
factor(credit.rating)
0
1

```
##
## [[8]]
```

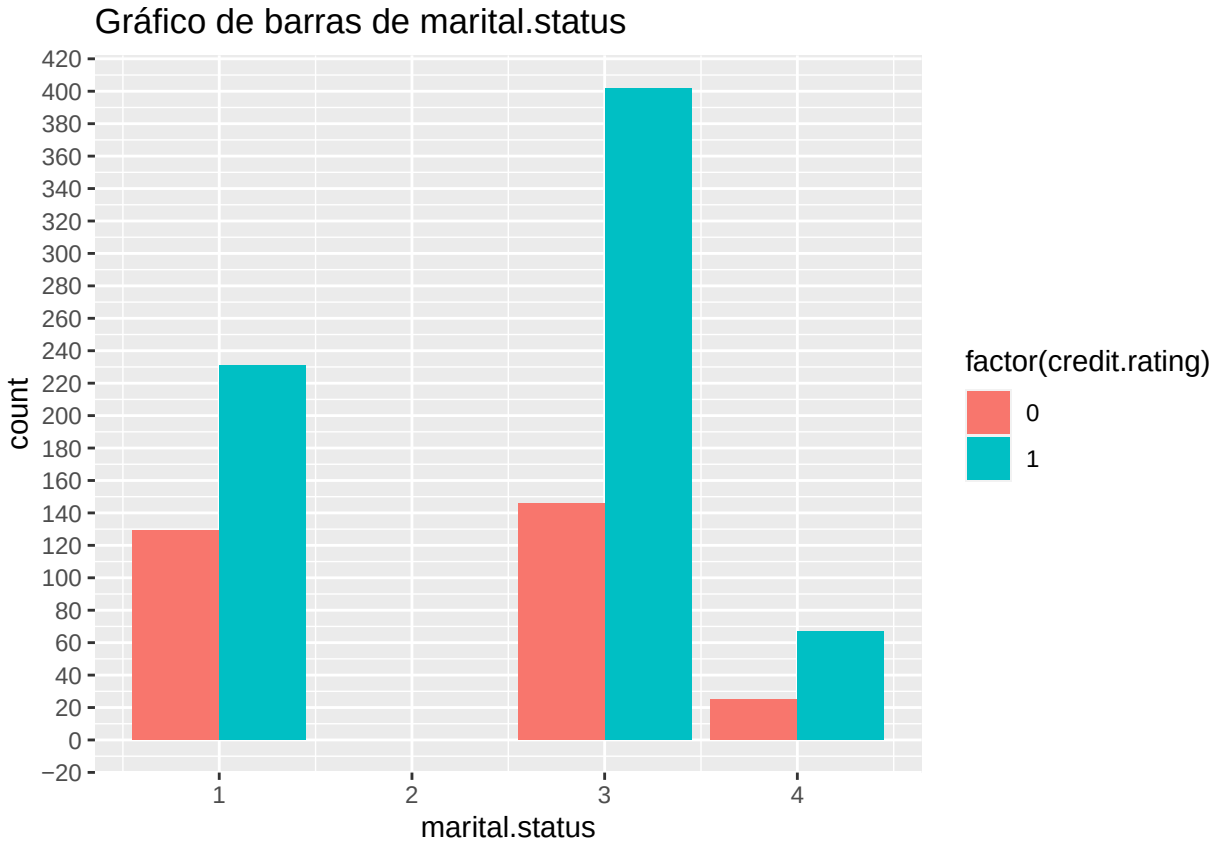


Gráfico de barras de marital.status
count
marital.status
factor(credit.rating)
0
1
420
400
380
360
340
320
300
280
260
240
220
200
180
160
140
120
100
80
60
40
20
0
−20
1
2
3
4

```
##
## [[9]]
```

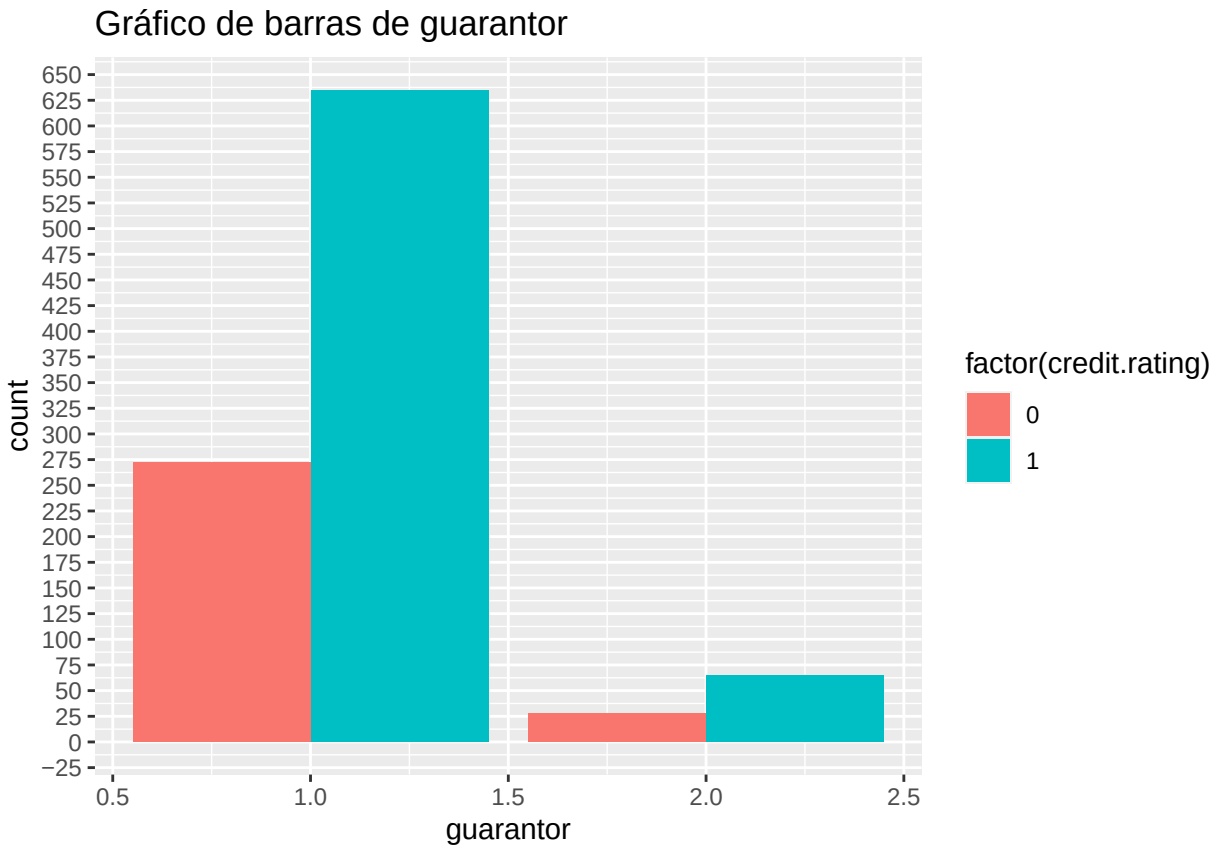


Gráfico de barras de guarantor
650
625
600
575
550
525
500
475
450
425
400
375
350
325
300
275
250
225
200
175
150
125
100
75
50
25
0
−25
count
0.5
1.0
1.5
2.0
2.5
guarantor
factor(credit.rating)
0
1

```
##
## [[10]]
```

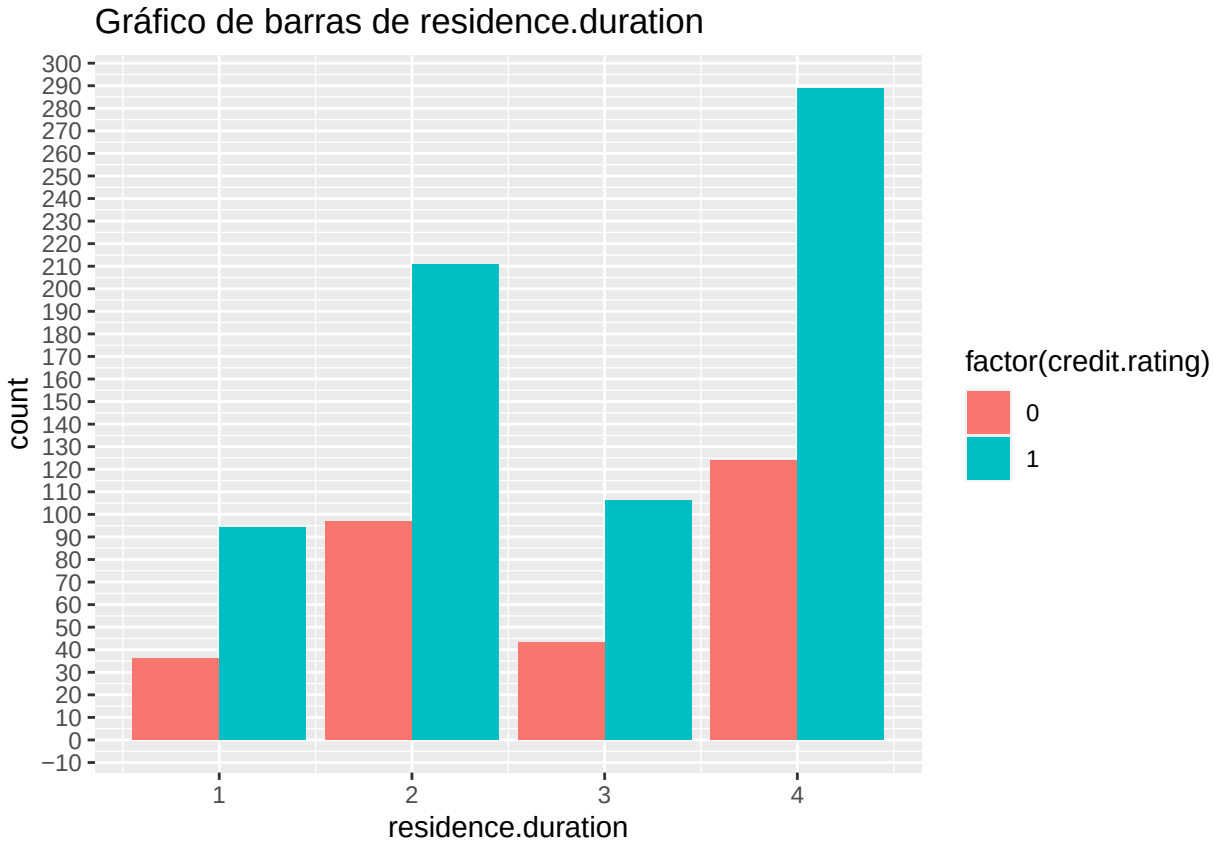


Gráfico de barras de residence.duration
count
residence.duration
factor(credit.rating)
0
1

```
##
## [[11]]
```

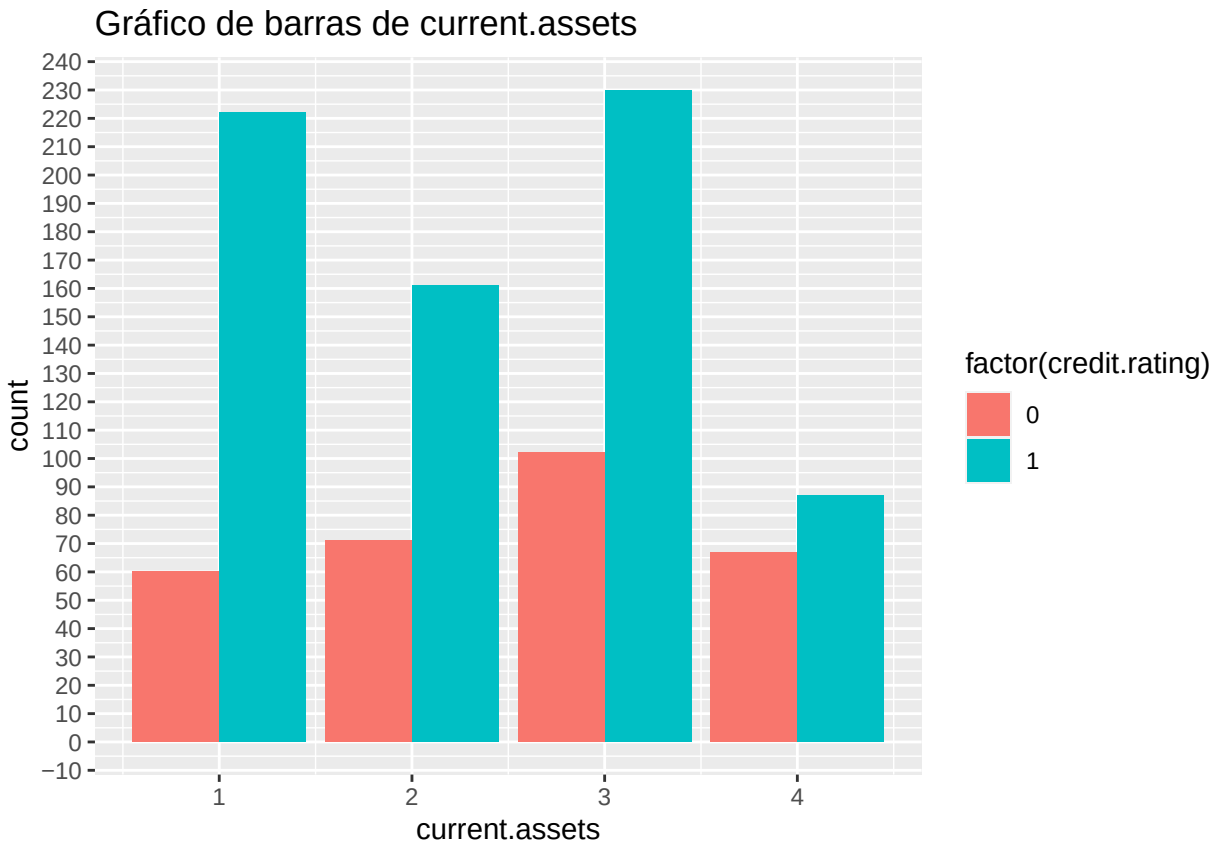


Gráfico de barras de current.assets
count
current.assets
factor(credit.rating)
0
1
240
230
220
210
200
190
180
170
160
150
140
130
120
110
100
90
80
70
60
50
40
30
20
10
0
−10
1
2
3
4

```
##
## [[12]]
```

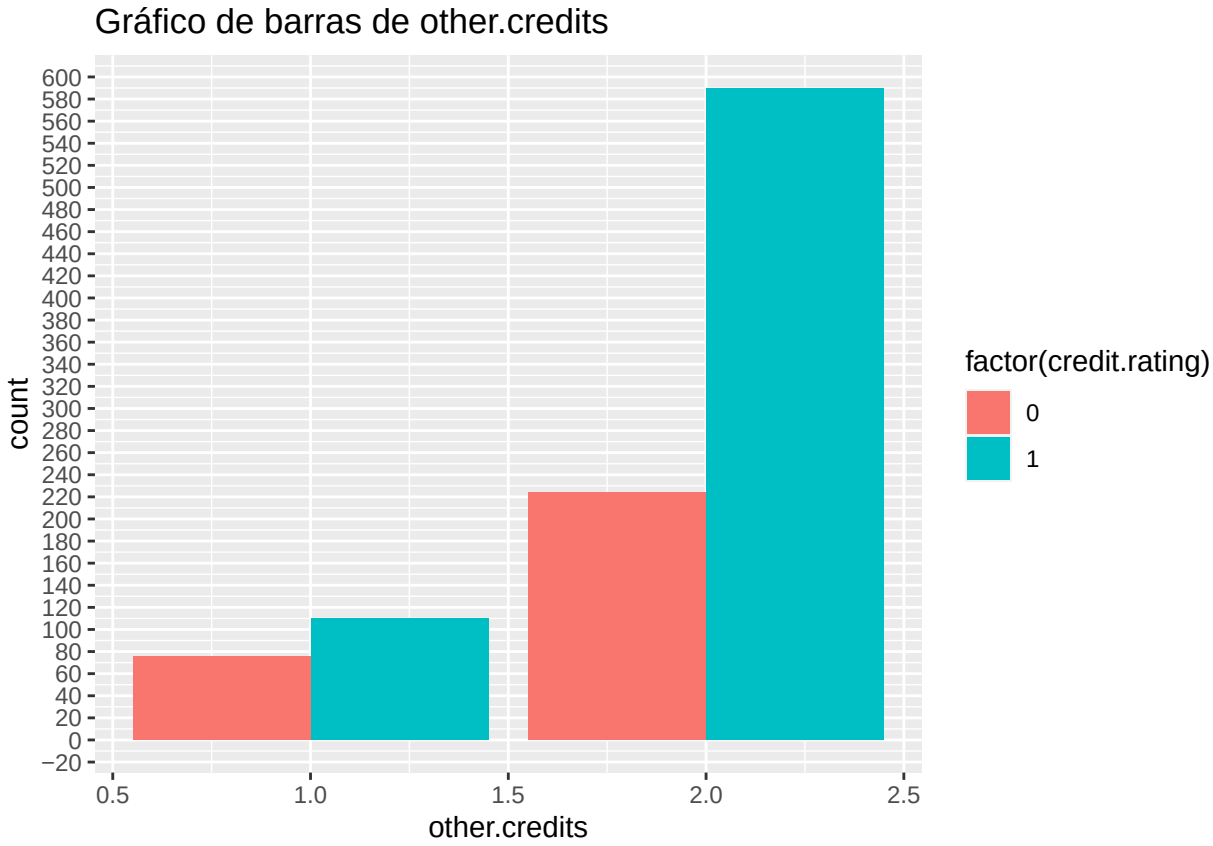


Gráfico de barras de other.credits
count
other.credits
0.5
1.0
1.5
2.0
2.5
factor(credit.rating)
0
1

```
##
## [[13]]
```


Gráfico de barras de apartment.type
count
apartment.type
factor(credit.rating)
0
1

```
##
## [[14]]
```

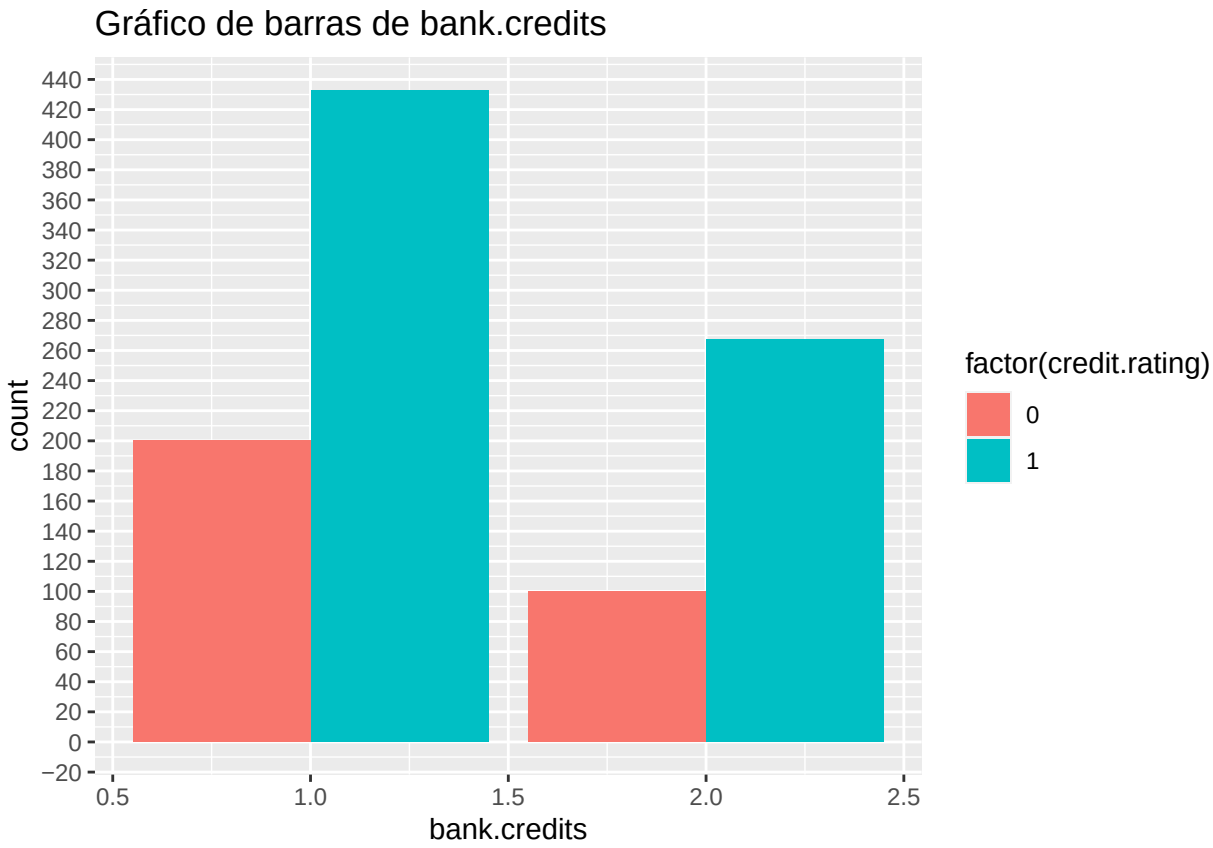


Gráfico de barras de bank.credits
count
bank.credits
factor(credit.rating)
0
1
440
420
400
380
360
340
320
300
280
260
240
220
200
180
160
140
120
100
80
60
40
20
0
−20
0.5
1.0
1.5
2.0
2.5

```
##
## [[15]]
```

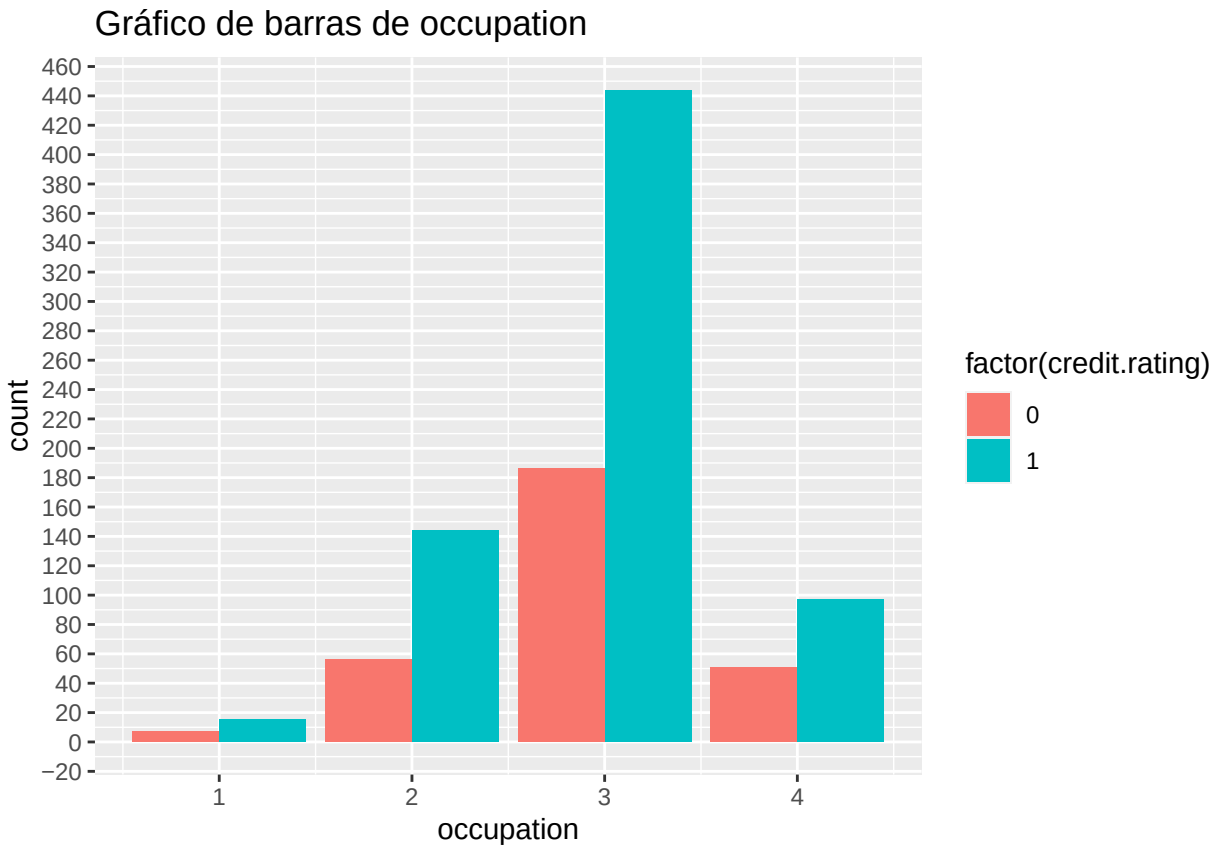


Gráfico de barras de occupation
460
440
420
400
380
360
340
320
300
280
260
240
220
200
180
160
140
120
100
80
60
40
20
0
−20
count
1
2
3
4
occupation
factor(credit.rating)
0
1

```
##
## [[16]]
```

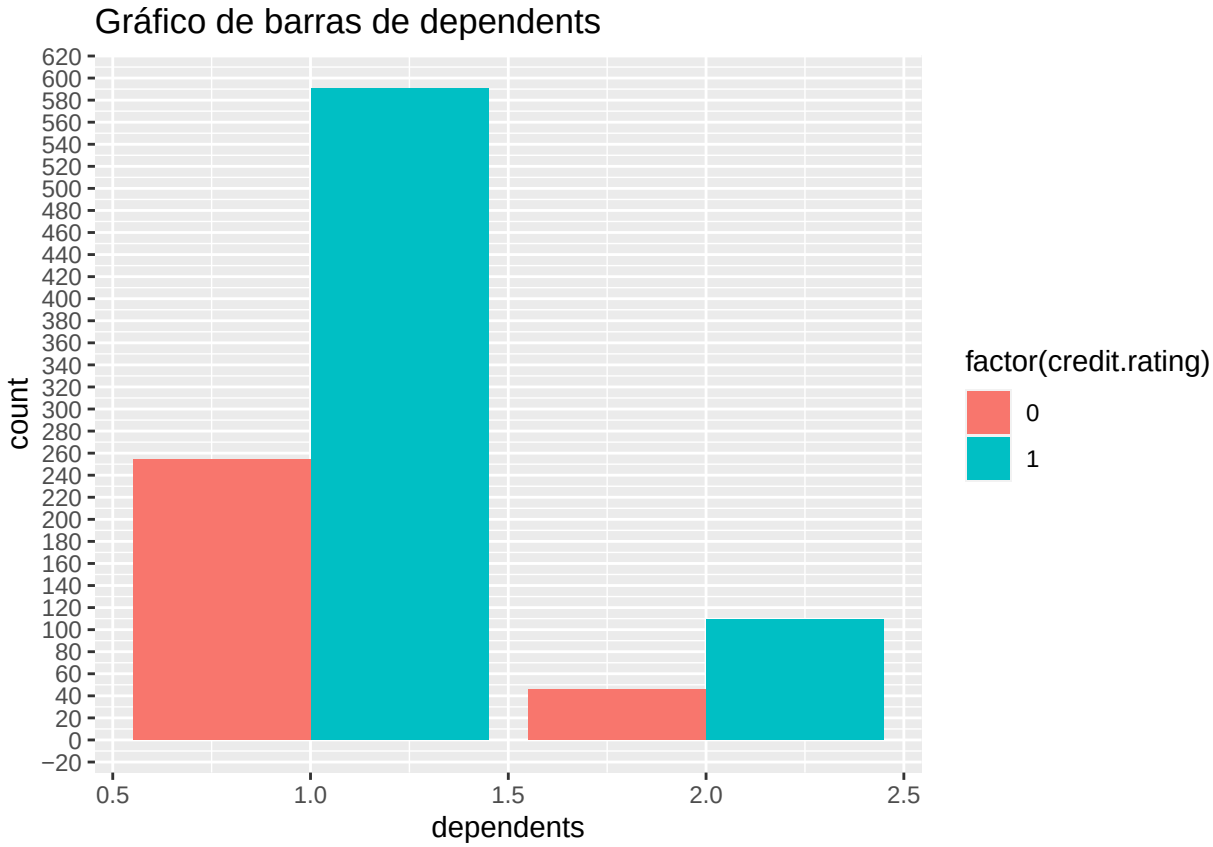


Gráfico de barras de dependents
count
dependents
factor(credit.rating)
0
1
620
600
580
560
540
520
500
480
460
440
420
400
380
360
340
320
300
280
260
240
220
200
180
160
140
120
100
80
60
40
20
0
−20
0.5
1.0
1.5
2.0
2.5

```
##
## [[17]]
```

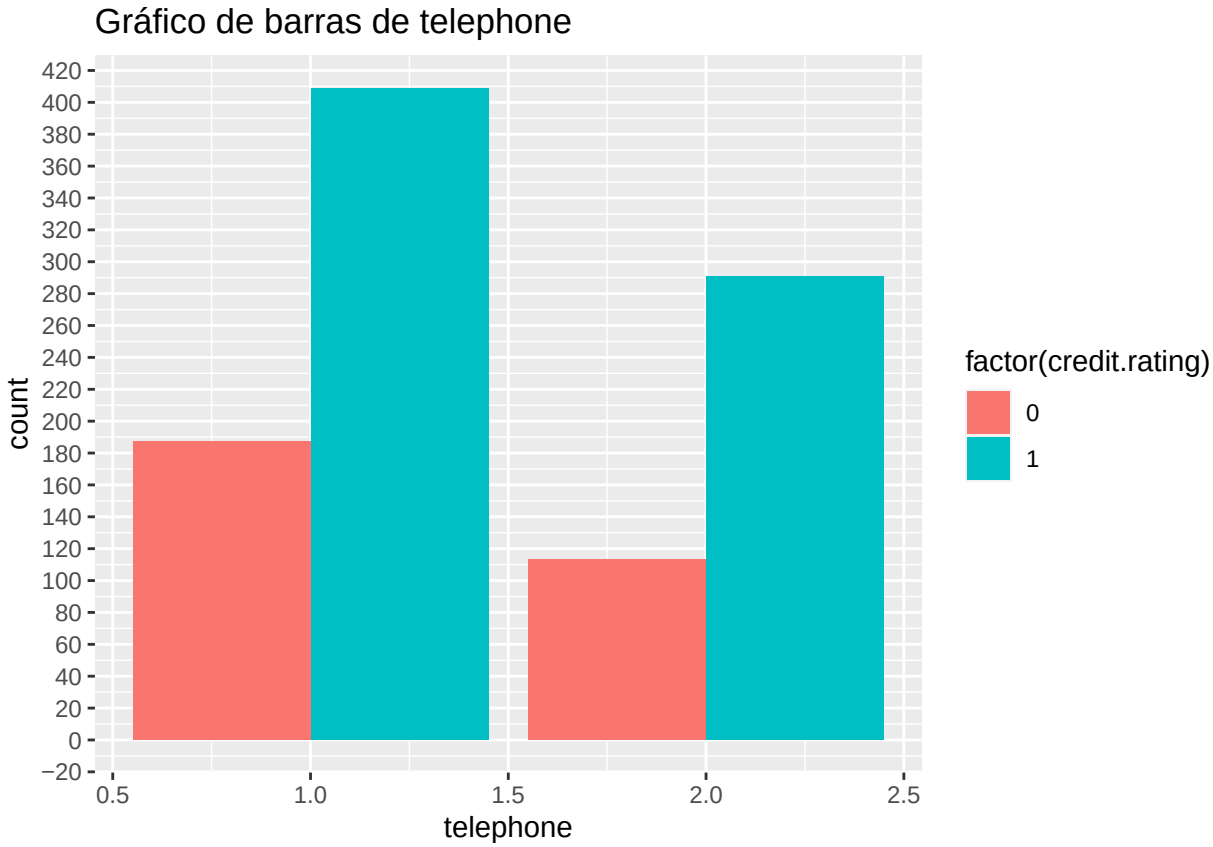
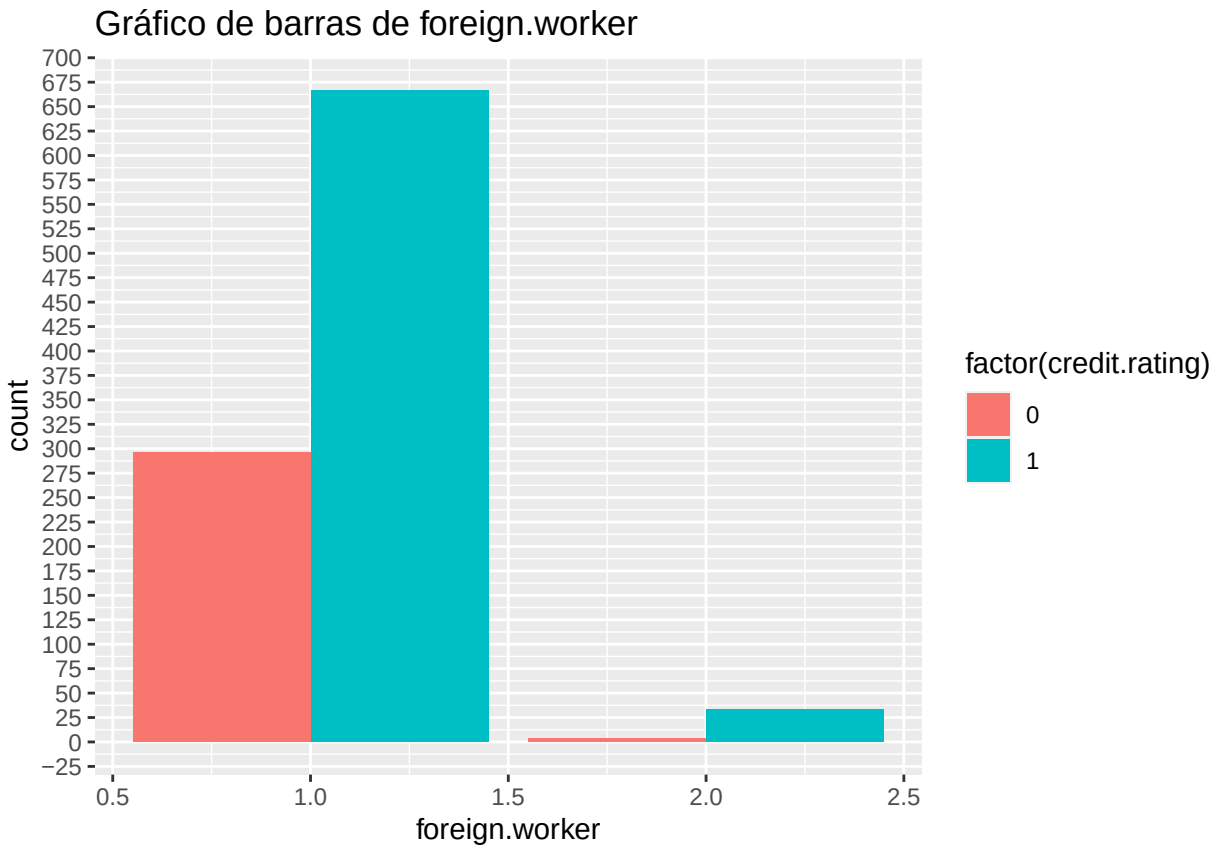


Gráfico de barras de telephone
count
telephone
factor(credit.rating)
0
1
420
400
380
360
340
320
300
280
260
240
220
200
180
160
140
120
100
80
60
40
20
0
−20
0.5
1.0
1.5
2.0
2.5

```
##
## [[18]]
```

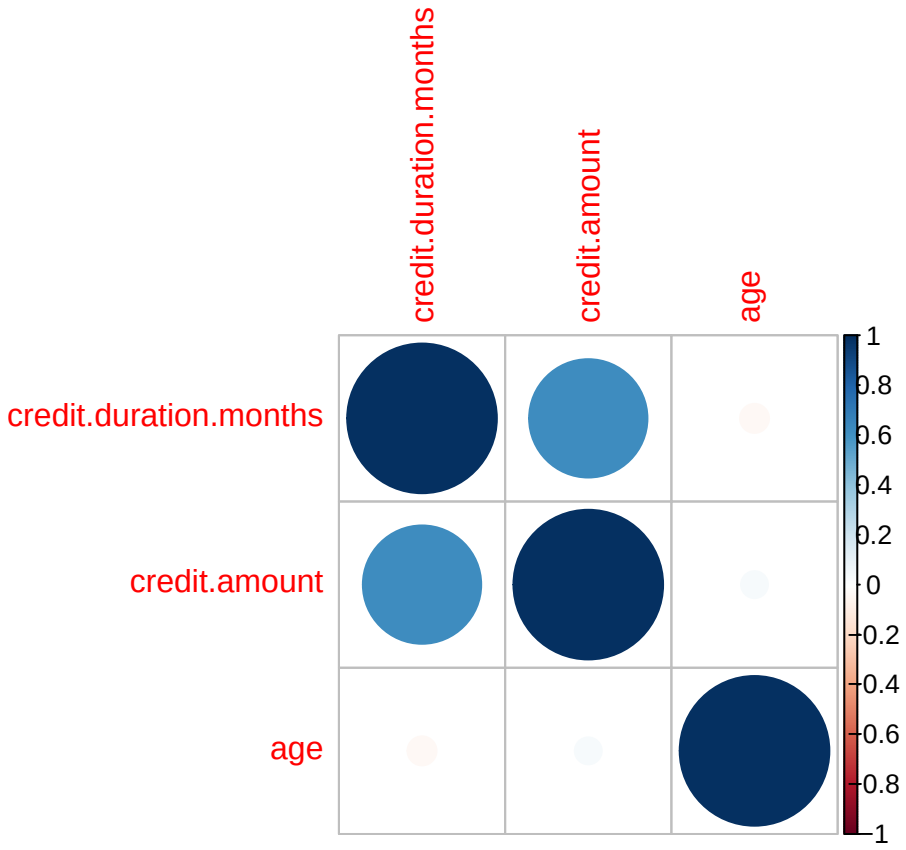


Etapa 7 - Calculando correlação entre variáveis

```
# Correlação - correlação baixa entre as variáveis
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
corrplot(cor(risk[,quantitativas]))
```



Etapa 8 - Feature Selection

```
library(randomForest)
```

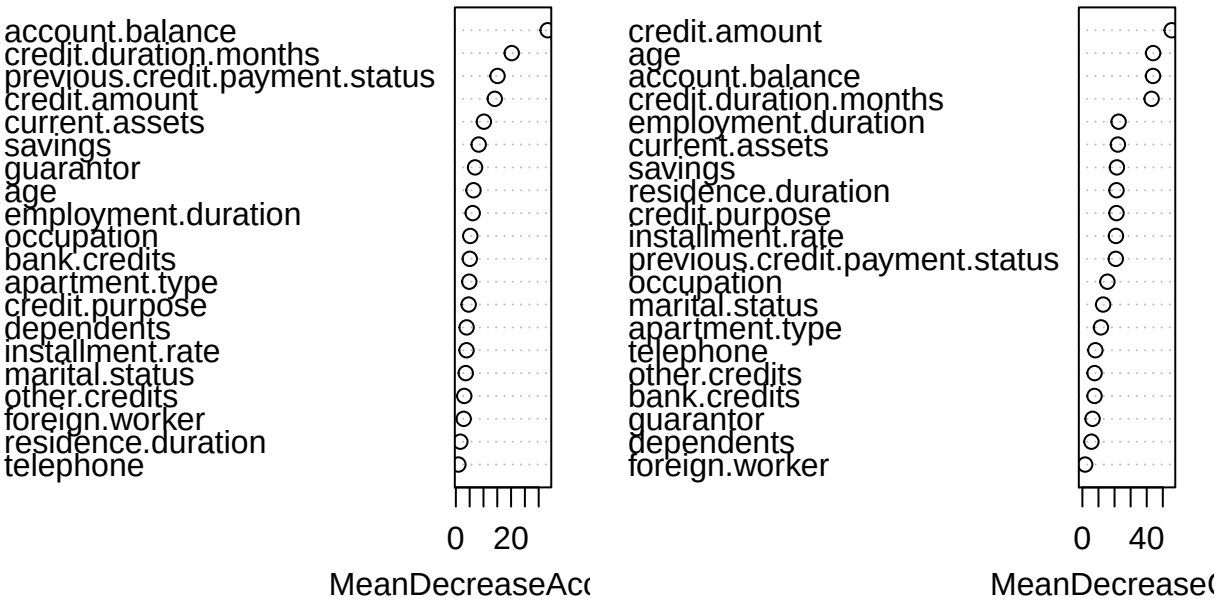
```
## randomForest 4.6-14
```

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

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
select.feature <- randomForest(data = risK[,-22], credit.rating ~ .,
                               importance = TRUE)
varImpPlot(select.feature)
```



Etapa 9 - Balanceamento de dados e feature selection

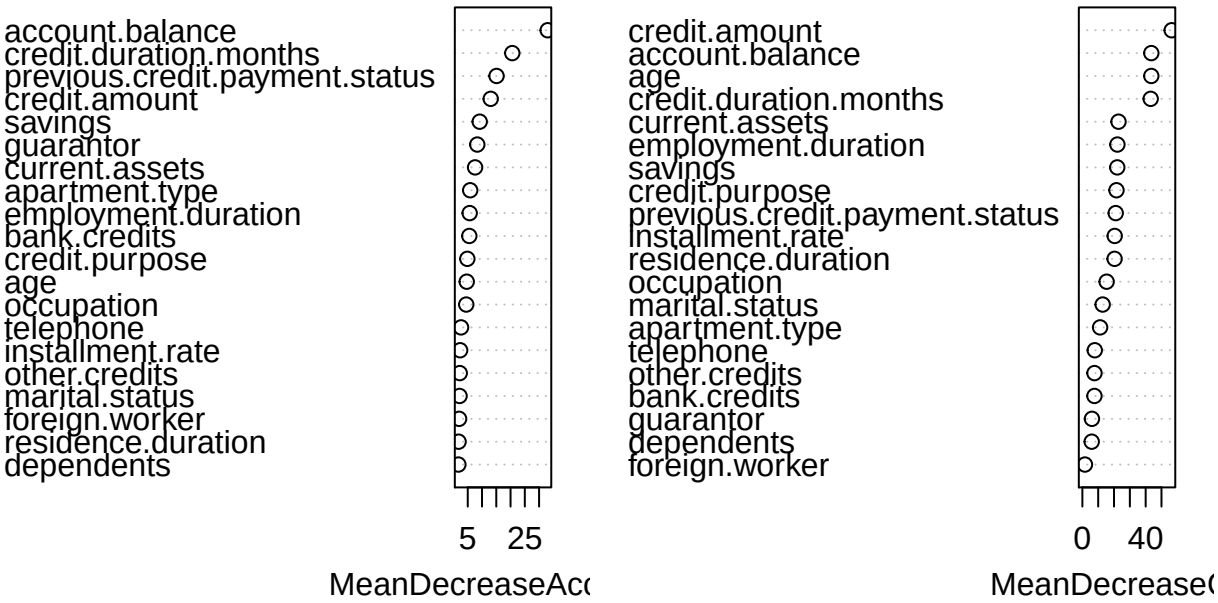
```
library(ROSE)
```

```
## Loaded ROSE 0.0-3
```

```
creditRiskRose <- ROSE(data = risK[-22], credit.rating ~ ., seed = 1)$data
table(creditRiskRose$credit.rating)
```

```
##
##   1   0
## 520 480
```

```
select.feature1 <- randomForest(data = risK[-22], credit.rating ~ .,
                                importance = TRUE)
varImpPlot(select.feature1)
```



Etapa 10 - Criando modelos

```
# Feature Selection
# as variáveis importantes:
# account.balance, credit.duration.months, previous.credit.payment.status,
# credit.amount, savings, guarantor, current.assets, credit.purpose, age.
colunas <- c("account.balance","credit.duration.months","previous.credit.payment.status",
             "credit.amount","savings","guarantor","current.assets","credit.purpose",
             "age")

paste.formula <- function(resposta, preditora) {
  form <- paste(resposta, sep = " ", "~ ")
  for(i in 1:length(preditora)) {
    if(i == 1) {
      form <- paste(form, sep = "", preditora[i])
    }
    else {
      form <- paste(form, sep = " + ", preditora[i])
    }
  }
  return(as.formula(form))
}

# Função cria vários modelos com dados balanceados de formas diferentes - índice 60, performance 0.823
modelsROSE <- function(quantity) {
  indices <- c()
  acuracia <- c()
  for(i in 1:quantity) {
```

```
    dfROSE <- ROSE(data = risK, credit.rating ~ ., seed = i)$data
    modelo <- randomForest(data = dfROSE, paste.formula("credit.rating", colunas), importance = TRUE)
    indices <- c(indices,i)
    acuracia <- c(acuracia,
                  (modelo$confusion[1,1] + modelo$confusion[2,2])/(modelo$confusion[1,1] + modelo$confus
  }
  return(list(acuracia, indices))
}

modelsROSE(100)
```

```
## [[1]]
##   [1] 0.768 0.789 0.778 0.793 0.805 0.784 0.777 0.795 0.796 0.786 0.777 0.777
##  [13] 0.781 0.793 0.773 0.805 0.805 0.794 0.784 0.799 0.793 0.805 0.798 0.792
##  [25] 0.790 0.806 0.794 0.792 0.818 0.794 0.788 0.799 0.795 0.792 0.786 0.785
##  [37] 0.768 0.781 0.779 0.798 0.776 0.779 0.765 0.820 0.776 0.790 0.779 0.809
##  [49] 0.785 0.809 0.797 0.781 0.771 0.816 0.817 0.768 0.791 0.775 0.785 0.823
##  [61] 0.817 0.772 0.771 0.790 0.753 0.789 0.791 0.769 0.766 0.786 0.786 0.774
##  [73] 0.790 0.794 0.792 0.797 0.806 0.782 0.786 0.789 0.794 0.795 0.777 0.789
##  [85] 0.802 0.797 0.795 0.806 0.790 0.775 0.780 0.812 0.787 0.782 0.773 0.808
##  [97] 0.818 0.770 0.805 0.787
##
## [[2]]
##   [1]   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
##  [19]  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36
##  [37]  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54
##  [55]  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72
##  [73]  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90
##  [91]  91  92  93  94  95  96  97  98  99 100
```

Dividindo dados de treino e teste para ver a performance do modelo

```
# Dividindo dados de treino e teste random forest - indice 26, performance 0.89
trainTest <- function(n) {
  indices <- c()
  acuracia <- c()
  dfROSE <- ROSE(data = risK, credit.rating ~ ., seed = 60)$data
  for(i in 1:n) {
    set.seed(i)
    rows <- sample(1:nrow(risK), 0.8*nrow(risK), replace = FALSE)
    trainDataSet <- dfROSE[rows,]
    testDataSet <- dfROSE[-rows,]
    modelo <- randomForest(data = trainDataSet, paste.formula("credit.rating", colunas), importance = TR
    pred <- predict(modelo, testDataSet[,-1])
    indices <- c(indices,i)
    acuracia <- c(acuracia, mean(testDataSet[,1] == pred))
  }
  return(list(acuracia, indices))
}

trainTest(100)
```

```
## [[1]]
##   [1] 0.865 0.760 0.795 0.780 0.835 0.825 0.800 0.800 0.835 0.765 0.775 0.835
```

```
##  [13] 0.785 0.830 0.805 0.800 0.830 0.820 0.765 0.765 0.800 0.860 0.775 0.825
##  [25] 0.830 0.890 0.860 0.855 0.805 0.785 0.810 0.810 0.790 0.810 0.810 0.795
##  [37] 0.820 0.815 0.800 0.805 0.850 0.850 0.815 0.810 0.785 0.810 0.780 0.830
##  [49] 0.770 0.790 0.795 0.825 0.795 0.835 0.835 0.815 0.835 0.815 0.780 0.820
##  [61] 0.825 0.790 0.870 0.820 0.785 0.805 0.840 0.790 0.840 0.870 0.785 0.820
##  [73] 0.810 0.825 0.855 0.825 0.790 0.770 0.820 0.805 0.805 0.810 0.800 0.810
##  [85] 0.840 0.790 0.830 0.830 0.785 0.785 0.855 0.840 0.805 0.750 0.785 0.825
##  [97] 0.760 0.815 0.780 0.840
##
## [[2]]
##   [1]   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
##  [19]  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36
##  [37]  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54
##  [55]  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72
##  [73]  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90
##  [91]  91  92  93  94  95  96  97  98  99 100
```