Exploração de dados - Banco Czech

Bruno Santos Wance de Souza

Lucas de Jesus Matias
Luiz Cesar Costa Raymundo

Contents

Objetivo

O objetivo do nosso trabalho é identificar quais fatores que podem impactar no atraso e não pagamento das dívidas, a partir da análise dos dados do Banco Berka, definindo os clientes os tipos de clientes apartir da análise de cluster.

Funções utilitárias

Criamos a função prepararNA para simplificar o tratamento dos dados indisponíveis, bastando passar a tablea, nome da coluna a ser tratado e o valor a ser substituído caso encontre o valor NA, vai retornar a tabela com os dados substituídos. A função prepararData auxilia o preparo da data para esta análise, que segue o padrão de ano sem os 2 primeiros dígitos, mês e dia, retorna uma data válida para o R.

```
prepararNA <- function (tabela, nomeColuna, valorSeNA) {
  indiceDados <- which(colnames(tabela)==nomeColuna)
  tabela[is.na(tabela[,indiceDados]),indiceDados] <- valorSeNA
  return(tabela)
}

prepararData <- function (vetorDataYYMMDD) {
  as.Date(paste0("19", vetorDataYYMMDD), "%Y%m%d")
}
```

Carga dos dados

Utilizamos a função read.csv2 que nos permite carregar os dados disponíveis em texto, no formato CSV.

```
read.csv2("./dados/account.asc", stringsAsFactors = FALSE) -> account
read.csv2("./dados/client.asc", stringsAsFactors = FALSE) -> client
read.csv2("./dados/card.asc", stringsAsFactors = FALSE) -> card
read.csv2("./dados/disp.asc", stringsAsFactors = FALSE) -> disp
read.csv2("./dados/district.asc", stringsAsFactors = FALSE) -> district
read.csv2("./dados/trans.asc", stringsAsFactors = FALSE) -> trans
read.csv2("./dados/loan.asc", stringsAsFactors = FALSE) -> loan
read.csv2("./dados/order.asc", stringsAsFactors = FALSE) -> order
```

Tratamento dos dados

Cliente

O tratamento inicial da relação cliente envolve a separação do campo data de nascimento e gênero, que será tratado por M e F. E a transformação do campo birth_number em uma data válida para o R. Para isso transformamos o birth_number em um campo númerico, obtemos 4 dígitos da 3ª à 4ª posição, sendo este valor superior a 50 consideramos como feminino, pois o mês de nascimento das mulheres está com uma soma de 50 unidades. Subtraímos 5000 do mês das mulheres, pois como um único campo numérico, implica em diminuir 50 do campo mensal. Após isso concatenamos o número 19 ao começo do birth_number, no intuito de deixar melhor preparado para a formatação da data, que ocorre logo em seguida. Logo após, calculamos a idade e selecionamos apenas os campos que nos serão úteis para o nosso estudo. Utilizamos a data de referência como 31/12/1998, devido a referência dos dados, para calcular a idade dos clientes.

```
currentdate <- as.Date("1998/12/31", format="%Y/%m/%d")
client <- client %>%
  mutate(mesajustado = as.numeric(stringr::str_sub(birth_number,3,4))) %>%
  mutate(gender = ifelse(mesajustado > 50, "F", "M")) %>%
```

```
  mutate(birth_number = ifelse(gender=="F", birth_number - 5000, birth_number)) %>%
  mutate(birth_number = paste0("19", birth_number)) %>%
  mutate(birth_number = as.Date(birth_number, "%Y%m%d")) %>%
  mutate(age = year(currentdate) - year(birth_number)) %>%
  select(client_id, age, district_id, gender)
```

Conta

O tratamento da relação conta inclui renomear as frequências com que a conta cria extratos para o correntista, pode ser mensal, semanal ou a cada transação, convertemos para fator, pois existem 3 fatores e selecionamos os campos importantes ao estudo.

```
#Tradução
account$frequency <- gsub("POPLATEK MESICNE", "mensal", account$frequency)
account$frequency <- gsub("POPLATEK TYDNE", "semanal", account$frequency)
account$frequency <- gsub("POPLATEK PO OBRATU", "acadatransacao", account$frequency)
account$frequency <- as.factor(account$frequency)
account$account_date <- prepararData(account$date)

account <- account %>%
          mutate(account_age = year(currentdate) - year(account_date)) %>%
          select(account_id, frequency, account_date, account_age)
```

Distrito

O tratamento inicial da relação distrito começa na renomeação dos campos para melhor entendimento, pois as colunas estão codificadas. Conversão dos campos de unemp_95 e unemp_96 para numéricos. Limpeza dos valores NA. Cálculo da taxa de desemprego entre os anos 95 e 96. E seleção dos valores que serão usados neste estudo.

```
#Renomear campos para melhor entendimento
colnames(district)[1] <- 'district_id'
colnames(district)[2] <- 'district_name'
colnames(district)[3] <- 'region'
colnames(district)[11] <- 'avg_sal'
colnames(district)[12] <- 'unemp_95'
colnames(district)[13] <- 'unemp_96'
colnames(district)[14] <- 'numb_enter'

#Converter campos para numérico
district$unemp_95 = as.numeric(district$unemp_95)
```

```
## Warning: NAs introduzidos por coerção
```

```
district$unemp_96 = as.numeric(district$unemp_96)
district$numb_enter = as.numeric(district$numb_enter)

#Limpeza de NA
district = prepararNA(district, "unemp_95", 1)

#Cálculo da taxa de desemprego e seleção de valores
district <- district %>%
            mutate(unemp_r =
                     ifelse(unemp_95 == 0 | unemp_96 == 0, 1, unemp_96/unemp_95)) %>%
            select(district_id, region, district_name, avg_sal, unemp_r, numb_enter)
```

Cartão

O tratamento inicial da relação cartão seleciona os campos que serão úteis para este estudo.

```
colnames(card)[3] <- 'card_type'
card <- card %>%
        select(card_id, disp_id, card_type)
```

Transação

Na relação de transação, identificamos alguns valores como "VYBER" que não estão descritos na documentação e não pareceu ser pertinente aos nossos estudos, por isso foi filtrado. O tipo de transação foi traduzido para facilitar o entendimento. E foi feito um agrupamento pela conta para facilitar o relacionamento de 1 para 1 com os valores que nos interessam. Filtramos também os dados no ano de 1998 pra cima, para focar os estudos nos dados mais atuais.

```
unique(trans$type)
```

```
## [1] "PRIJEM" "VYDAJ"  "VYBER"
```

```
trans$type <- gsub("PRIJEM", "credito", trans$type)
trans$type <- gsub("VYDAJ", "debito", trans$type)
trans$date <- prepararData(trans$date)


trans <- trans %>%
        dplyr::filter(type != "VYBER") %>%
        mutate(amount = as.numeric(amount)) %>%
        mutate(balance = as.numeric(balance)) %>%
        select(account_id, type, amount, balance) %>%
        group_by(account_id) %>%
        summarise(
          credito = sum(ifelse(type == "credito", amount, 0)),
          debito = sum(ifelse(type == "debito", amount, 0)),
          trans_amount = sum(amount),
          saldo_medio_em_conta = mean(balance),
          mediana_saldo = median(balance),
          min_saldo = summary(balance)[1],
          fq_saldo = summary(balance)[2],
          quant_trans = n()) %>%
        mutate(withdraw_rate = debito / credito) %>%
        select(account_id, withdraw_rate, quant_trans, saldo_medio_em_conta,
               mediana_saldo, min_saldo, fq_saldo, trans_amount)
```

Disposição

A relação de disposição é útil para auxiliar nos relacionamentos. Como nosso estudo centraliza no pagamento ou não de dívidas, sendo que somente os donos da conta conseguem pegar empréstimos, agrupamos as disposições para os donos das contas que serão o alvo do estudo.

```
unique(disp$type)
```

```
## [1] "OWNER"     "DISPONENT"
```

```
disp <- disp %>%
        group_by(account_id) %>%
        summarise(
```

```
          client_id = first(client_id[type == "OWNER"]),
          disp_id = first(disp_id[type == "OWNER"]),
          no_account_users = n()
        )
```

Empréstimo

No preparo da relação de empréstimo, traduzimos os status para os significados reais, também preparamos a taxa de pagamento.

```
unique(loan$status)
```

```
## [1] "B" "A" "C" "D"
```

```
loan$status <- gsub("A", "finalizado", loan$status)
loan$status <- gsub("B", "nao pago", loan$status)
loan$status <- gsub("C", "vigente", loan$status)
loan$status <- gsub("D", "em debito", loan$status)
loan$payments = as.numeric(loan$payments)
loan$amount = as.numeric(loan$amount)

loan$loan_date = prepararData(loan$date)

loan <- loan %>%
        mutate(loan_year = year(loan_date)) %>%
        mutate(loan_age = year(currentdate) - loan_year) %>%
        mutate(payments_rate = payments / amount) %>%
        select(loan_id, account_id, amount, duration, status,
               payments_rate, loan_date, loan_age)

colnames(loan)[3] <- 'loan_amount'
colnames(loan)[4] <- 'loan_duration'
colnames(loan)[5] <- 'loan_status'
colnames(loan)[6] <- 'loan_payment_rate'
```

Ordem de Pagamento

No preparo da ordem de pagamento, o texto do tipo de ordem é traduzido, ignoramos os que não possuem nenhum tipo especificado e selecionamos os dados importantes a este estudo através do agrupamento pela conta.

```
unique(order$k_symbol)
```

```
## [1] "SIPO"     "UVER"     " "        "POJISTNE" "LEASING"
```

```
order$k_symbol = gsub("POJISTNE", "seguro", order$k_symbol)
order$k_symbol = gsub("SIPO", "domestico", order$k_symbol)
order$k_symbol = gsub("LEASING", "leasing", order$k_symbol)
order$k_symbol = gsub("UVER", "divida", order$k_symbol)
order$amount = as.numeric(order$amount)

order <- order %>%
         dplyr::filter(k_symbol != " ") %>%
         group_by(account_id) %>%
         summarise(
            total_ordem = sum(amount),
```

```
            quant_ordem = n(),
            paga_divida = any(k_symbol == "divida"),
            paga_leasing = any(k_symbol == "leasing"),
            ja_pagou_seguro = any(k_symbol == "seguro"),
            media_transf = sum(amount) / n()
        )
```

União dos dados

Criamos um dataset "dados" copiando as variaveís que precisamos para inciarmos as nossas análises.

Preparar união

```
dados <- client %>%
         inner_join(disp, by = "client_id") %>%
         left_join(card, by = "disp_id") %>%
         left_join(district, by = "district_id") %>%
         inner_join(account, by = "account_id") %>%
         left_join(order, by= "account_id") %>%
         inner_join(trans, by="account_id") %>%
         left_join(loan, by="account_id")
```

Preparar NAs

Utilizando nossa função, preparamos os dados que possuem NA para que não interfiram na análise de forma negativa.

```
dados <- dados %>%
         prepararNA("card_type", "nenhum") %>%
         prepararNA("total_ordem", 0) %>%
         prepararNA("quant_ordem", 0) %>%
         prepararNA("ja_pagou_seguro", FALSE) %>%
         prepararNA("media_transf", FALSE) %>%
         prepararNA("loan_amount", 0) %>%
         prepararNA("loan_duration", 0) %>%
         prepararNA("loan_status", "nenhum")
```

Preparar factors

```
dados$card_type = as.factor(dados$card_type)
dados$loan_status = as.factor(dados$loan_status)
```

Preparar dados de empréstimo

Fitramos os dados que possuem algum empréstimo válido. Logo após criamos uma nova variável com o nome de "problemas_loan", cujo objetivo é informar se o cliente possui algum problema de pagamento.

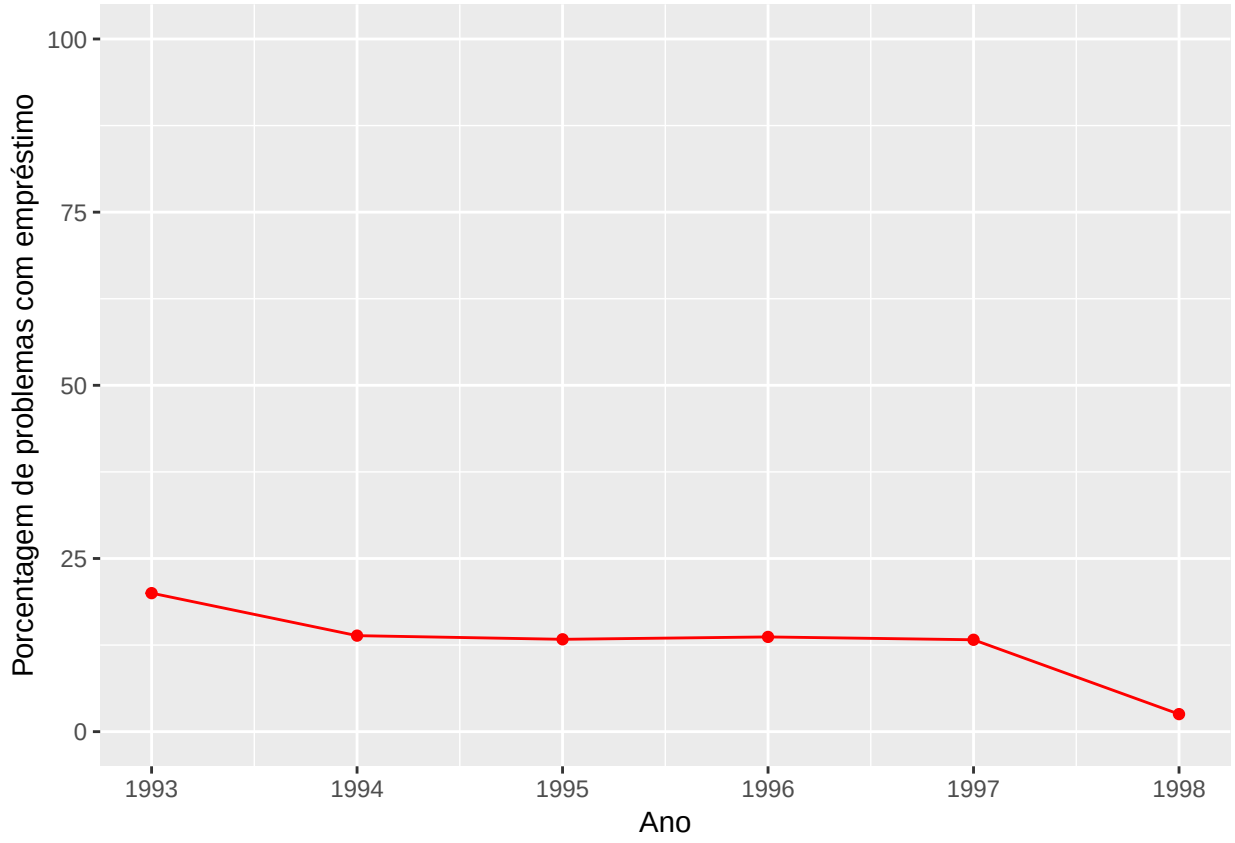
```
dadosLoan <- dados %>%
             dplyr::filter(loan_status != "nenhum") %>%
             mutate(problemas_loan =
                      if_else(loan_status == "em debito" | loan_status == "nao pago",
                    1, 0)) %>%
             mutate(ja_pagou_seguro = ifelse(ja_pagou_seguro == TRUE, 1, 0))
```

Estudo dos dados

Empréstimos

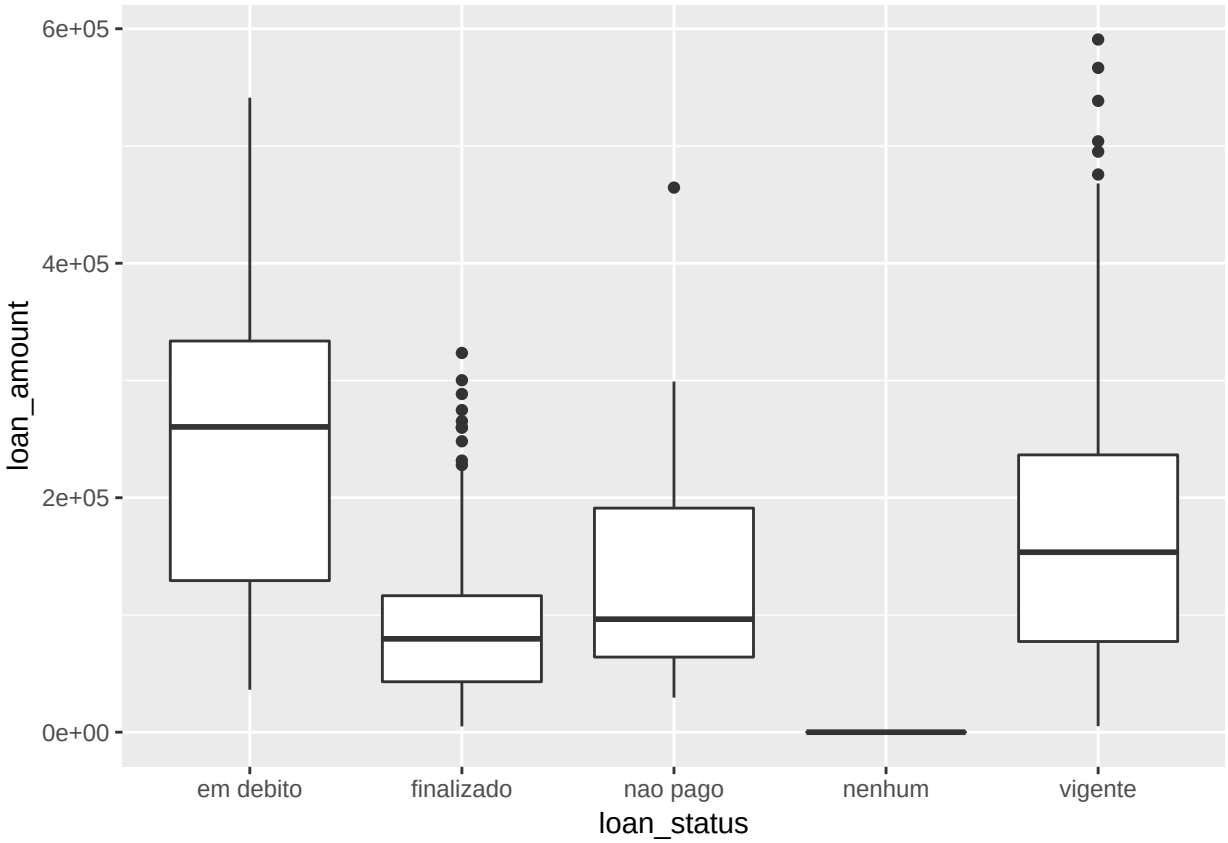
Observamos que contratos mais novos apresentam menos problemas com pagamentos, mas isso não significam que não possam vir a apresentar problemas no futuro.

```
problemasPorAno <- dadosLoan %>%
  mutate(ano = year(loan_date)) %>%
  select(ano, problemas_loan) %>%
  group_by(ano, problemas_loan) %>%
  summarise(
    count = n()
  ) %>%
  mutate(perc =  count/sum(count)) %>%
  dplyr::filter(problemas_loan == TRUE)

ggplot(problemasPorAno, aes(x = ano, y = perc*100), group = 1) +
  geom_line(color="red") +
  geom_point(color="red") +
  labs(x = "Ano", y = "Porcentagem de problemas com empréstimo") +
  expand_limits(y=c(0, 100))
```



E também observamos maiores problemas nos empréstimos quando a medida que o valor é maior

```
emprestimo<-ggplot(dados,aes(x=loan_status,y=loan_amount))
emprestimo<-emprestimo+geom_boxplot()
emprestimo
```



Cartões

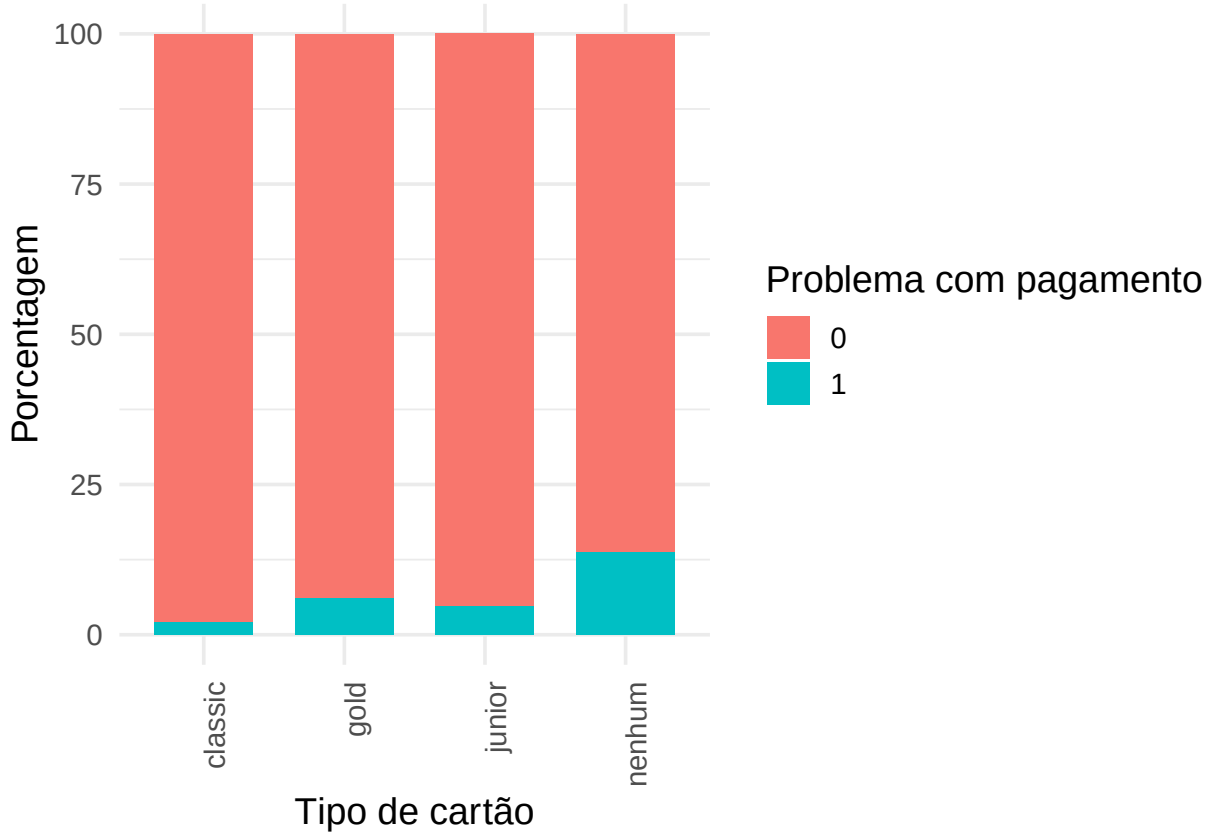
Identificamos também que pessoas que possuem cartões tem menores chances de ter problemas com os empréstimos. O valor P do qui-quadrado confirma a relação.

```
dadosLoan <- dadosLoan %>%
  mutate(has_card = ifelse(card_type=="nenhum", FALSE, TRUE))

chisq.test(table(dadosLoan$has_card, dadosLoan$problemas_loan))
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  table(dadosLoan$has_card, dadosLoan$problemas_loan)
## X-squared = 14.303, df = 1, p-value = 0.0001556
```

```
analiseCartao <- dadosLoan %>%
  select(card_type, problemas_loan) %>%
  group_by(card_type, problemas_loan) %>%
  summarise(
    count = n()
  ) %>%
  mutate(perc =  count/sum(count))
```

```
ggplot(analiseCartao,
       aes(x = factor(card_type), y = perc*100, fill = factor(problemas_loan))) +
  geom_bar(stat="identity", width = 0.7) +
  labs(x = "Tipo de cartão", y = "Porcentagem", fill = "Problema com pagamento") +
  theme_minimal(base_size = 14) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```

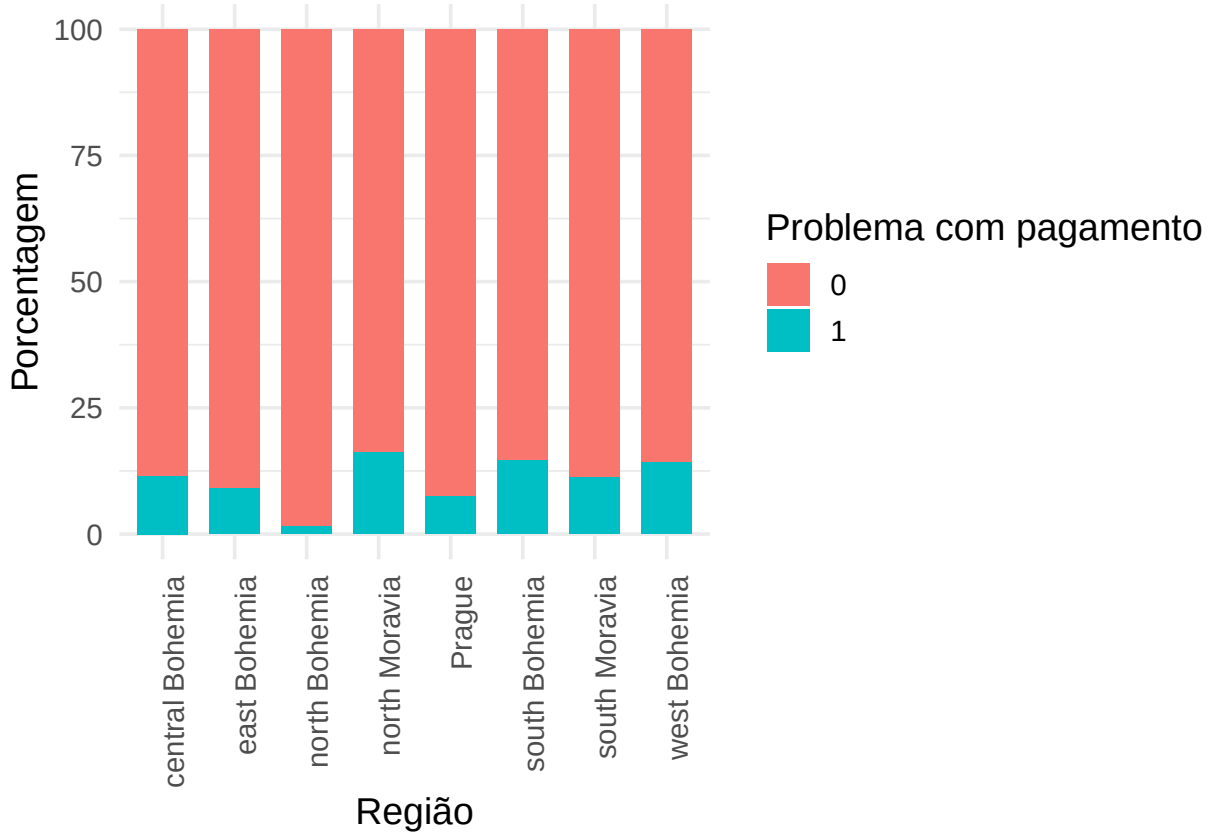


Distritos

Analisando os problemas com empréstimos por distrito, percebemos que clientes da região "North Bohemia" tem baixo índice de inadimplência. e "North Moravia"" tem o maior indíce de inadimplência.

```
distritoAnalise <- dadosLoan %>%
                   select(region, problemas_loan) %>%
                   group_by(region, problemas_loan) %>%
                   summarise(
                     count = n()
                   ) %>%
                   mutate(perc =  count/sum(count))


ggplot(distritoAnalise,
         aes(x = factor(region), y = perc*100, fill = factor(problemas_loan))) +
  geom_bar(stat="identity", width = 0.7) +
  labs(x = "Região", y = "Porcentagem", fill = "Problema com pagamento") +
```

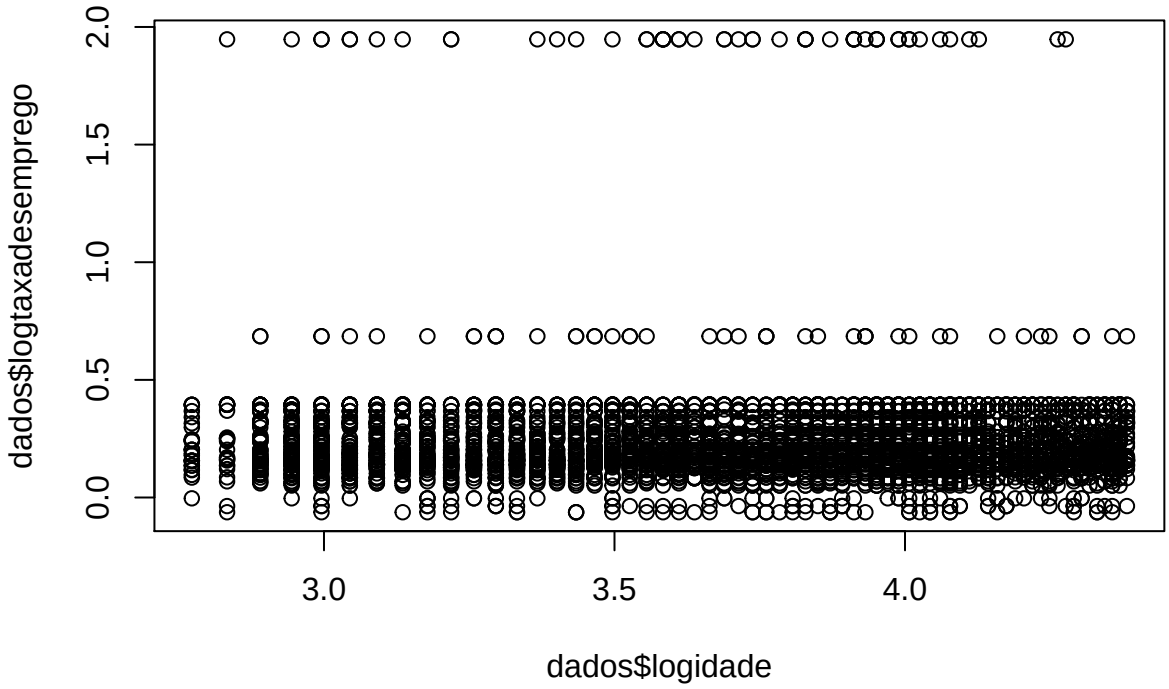
```
  theme_minimal(base_size = 14) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



Idade e taxa de desemprego

A Análise de regressão (log-log) para verificar se há correlação entre idade e taxa de desemprego.

```
dados["logidade"]<-log(dados$age)
dados["logtaxadesemprego"]<-log(dados$unemp_r)

regres<-lm(logidade~logtaxadesemprego, data=dados)

plot (dados$logidade, dados$logtaxadesemprego)

abline(regres)
```




```
summary(regres)
```

```
##
## Call:
## lm(formula = logidade ~ logtaxadesemprego, data = dados)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.93979 -0.33241  0.06149  0.33514  0.69161
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)       3.712264   0.009351 397.008   <2e-16 ***
## logtaxadesemprego -0.031877   0.028533  -1.117    0.264
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4184 on 4498 degrees of freedom
## Multiple R-squared:  0.0002774,  Adjusted R-squared:  5.514e-05
## F-statistic: 1.248 on 1 and 4498 DF,  p-value: 0.264
```

Estudo de cluster

Modelo

Após um estudo de regressão linear múltipla, chegamos a um modelo que explica 50% dos problemas com dívidas no R quadrado.

```
dadosLoanCluster <- dadosLoan %>%
                      select( problemas_loan, ja_pagou_seguro, no_account_users,
                              fq_saldo, media_transf, quant_ordem, min_saldo)

lm(formula = problemas_loan ~ ., data = dadosLoanCluster) -> modelo
summary(modelo)
```

```
##
## Call:
## lm(formula = problemas_loan ~ ., data = dadosLoanCluster)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.46200 -0.12456 -0.05250  0.02771  0.87919
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)       6.148e-01  4.977e-02  12.354  < 2e-16 ***
## ja_pagou_seguro   1.779e-01  3.732e-02   4.767 2.29e-06 ***
## no_account_users -8.648e-02  2.124e-02  -4.071 5.24e-05 ***
## fq_saldo         -7.496e-06  8.226e-07  -9.112  < 2e-16 ***
## media_transf      1.281e-05  4.305e-06   2.976  0.00303 **
## quant_ordem      -1.323e-01  1.978e-02  -6.690 4.69e-11 ***
## min_saldo        -5.817e-05  3.629e-06 -16.031  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2234 on 675 degrees of freedom
## Multiple R-squared:  0.5012, Adjusted R-squared:  0.4967
## F-statistic:   113 on 6 and 675 DF,  p-value: < 2.2e-16
```

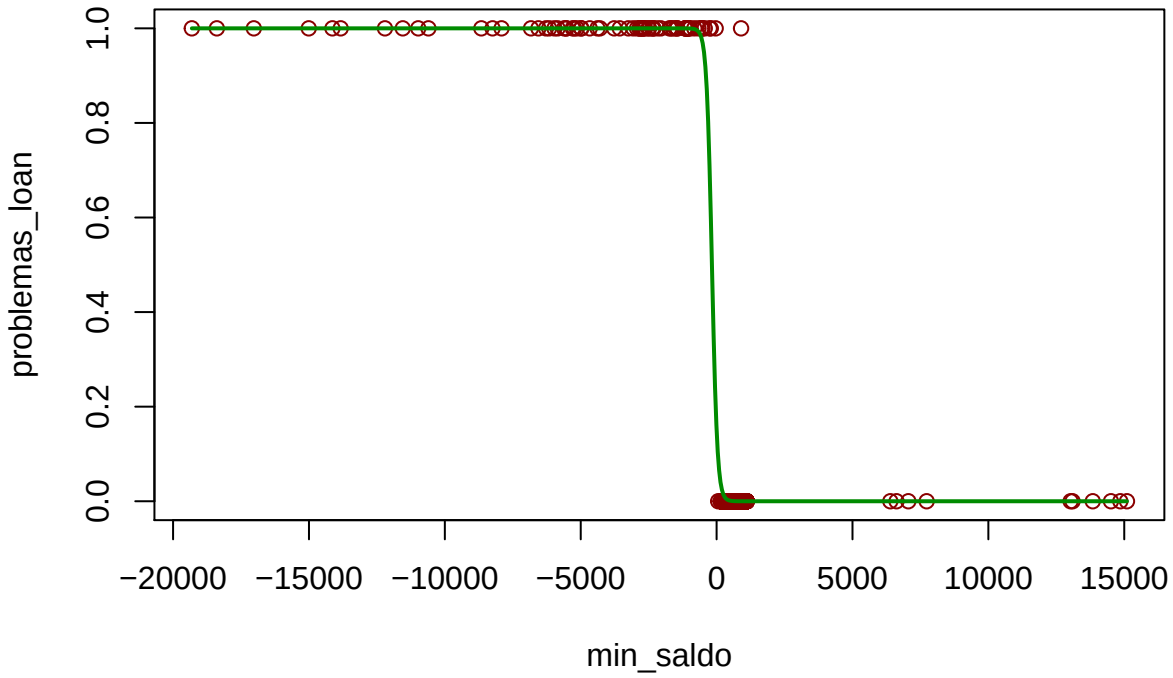
Regressão logística

Ao montar o modelo logístico podemos observer uma relação logística bem forte entre o saldo mínimo em conta e problemas com dívidas.

```
dadosLoan$min_saldo <- as.vector(dadosLoan$min_saldo)
mylogit <- glm(problemas_loan ~ min_saldo, data = dadosLoan, family = "binomial")
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

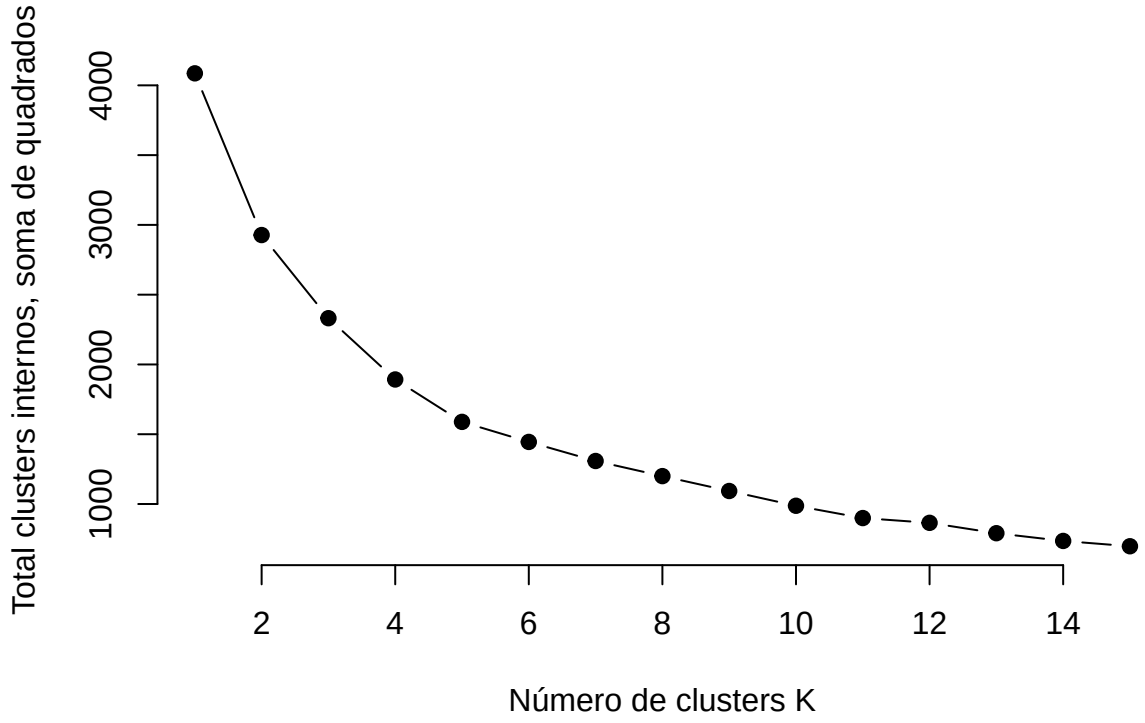
```
newdat <- data.frame(min_saldo=seq(min(dadosLoan$min_saldo), max(dadosLoan$min_saldo),len=682))
newdat$problemas_loan = predict(mylogit, newdata=newdat, type="response")
plot(problemas_loan~min_saldo, data=dadosLoan, col="red4")
lines(problemas_loan ~ min_saldo, newdat, col="green4", lwd=2)
```



Número de clusters

Utilizaremos a metolodologia de clusters k-means, primeiro vamos definir o quantitativo de clusters. Com análise no gráfico utilizando o Elbow Method, consideramos o uso de 3 ou 4 clusters interessante, pois é onde que começa a se perder o benefício de utilização de mais clusters. Decidimos por 3 para classificarmos os consumidores em alto, médio e baixo risco.

```
#achar a quantidade ideal de clusters
dadosLoanCluster_std <- scale(dadosLoanCluster[-1])
scaled_loan_data = as.matrix(dadosLoanCluster_std)

set.seed(123)
k.max <- 15
data <- scaled_loan_data
wss <- sapply(1:k.max,
              function(k){kmeans(data, k, nstart=50,iter.max = 15 )$tot.withinss})

plot(1:k.max, wss,
     type="b", pch = 19, frame = FALSE,
     xlab="Número de clusters K",
     ylab="Total clusters internos, soma de quadrados")
```



Criação dos clusters

Utilizamos o k-means para criar os clusters.

```
#achar padrões

kmeans <- kmeans(dadosLoanCluster,3)


#sumario
kmeans
```

```
## K-means clustering with 3 clusters of sizes 166, 316, 200
##
## Cluster means:
##   problemas_loan ja_pagou_seguro no_account_users fq_saldo media_transf
## 1      0.3192771       0.2650602         1.186747 16857.08     4061.614
## 2      0.0664557       0.1424051         1.218354 30090.45     4545.824
## 3      0.0100000       0.1250000         1.225000 44946.04     5062.291
##   quant_ordem  min_saldo
## 1    1.915663 -1283.8988
## 2    1.829114   715.9772
## 3    1.705000   683.9030
##
## Clustering vector:
##   [1] 2 1 2 1 2 3 2 2 1 2 2 2 2 3 3 2 3 1 2 3 2 3 2 2 1 3 1 2 1 3 3 2 3 3 2
##  [36] 2 2 1 2 1 2 3 1 2 3 1 1 2 2 3 3 2 3 3 2 1 2 1 2 1 2 2 1 2 1 2 2 1 3 1
```

```
##  [71] 1 3 3 2 3 2 2 2 1 2 2 1 2 2 2 3 3 1 1 3 2 1 1 3 2 2 2 3 3 3 1 1 2 2 2
## [106] 3 1 3 1 3 2 1 2 1 2 3 1 2 2 2 2 1 1 3 1 3 3 3 2 1 3 1 1 3 3 3 2 2 1 2
## [141] 2 2 3 1 2 3 2 2 2 1 2 1 3 3 2 2 3 2 3 3 2 2 2 3 2 1 2 3 2 1 3 3 3 2 3
## [176] 2 1 1 3 2 1 3 2 2 2 2 2 1 2 2 2 1 3 2 1 1 3 3 2 1 2 2 2 2 3 2 1 3 3 3
## [211] 1 2 1 1 3 2 2 1 1 3 3 3 1 1 2 3 2 2 2 2 1 2 2 1 2 2 3 1 2 3 2 3 3 2 3
## [246] 2 2 3 1 1 2 3 3 1 2 3 1 2 3 1 2 2 2 3 1 1 3 1 2 2 3 1 3 3 2 3 1 1 2 2
## [281] 1 1 3 3 1 2 2 3 2 2 3 2 2 2 1 2 3 2 2 1 2 1 2 2 3 2 2 3 2 2 1 1 3 3 2
## [316] 2 1 3 1 2 2 2 1 1 2 3 3 2 2 3 3 3 2 3 2 2 2 3 1 1 2 2 2 1 2 3 2 1 3 3
## [351] 2 3 1 2 1 2 3 1 1 1 2 1 3 3 2 3 2 2 3 3 2 3 2 1 1 2 2 1 3 3 2 2 1 2 2
## [386] 3 1 2 3 2 2 2 3 3 1 2 2 1 1 2 2 3 2 3 3 3 1 1 2 2 1 2 2 2 2 3 2 2 1 2
## [421] 1 2 1 2 2 1 1 2 2 2 2 2 3 1 2 2 2 3 2 2 3 2 2 2 3 2 2 1 3 3 2 1 1 2 3
## [456] 3 3 1 3 2 2 2 2 2 3 2 3 3 3 2 3 2 2 1 1 3 2 3 2 3 2 1 3 1 3 1 2 3 1 2
## [491] 2 2 1 2 2 2 2 2 2 2 2 3 3 2 2 3 1 3 1 1 1 3 2 1 1 3 2 3 1 2 2 2 3 2 1
## [526] 1 3 1 2 2 1 3 3 1 1 2 1 2 2 2 2 1 3 1 2 1 2 2 2 2 1 3 2 2 2 3 2 2 1 2
## [561] 3 2 3 3 1 2 2 1 3 2 3 1 3 1 2 2 2 1 3 2 2 1 1 2 2 2 2 3 3 3 1 3 3 2 2
## [596] 3 1 2 3 3 2 2 3 1 2 3 2 3 2 3 2 1 3 2 2 1 1 2 2 2 2 2 3 3 2 3 1 2 2 2
## [631] 2 1 3 3 3 2 2 3 3 2 2 2 2 1 3 2 3 1 1 2 3 2 1 3 3 2 3 2 3 3 2 3 2 2 2
## [666] 3 2 2 2 3 3 3 2 2 1 1 3 3 2 2 2 1
##
## Within cluster sum of squares by cluster:
## [1] 8275067127 7301365869 7741187312
##  (between_SS / total_SS =  75.8 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

```
#VEctor da soma dos quadrados, um componete por cluster
kmeans$withinss
```

```
## [1] 8275067127 7301365869 7741187312
```

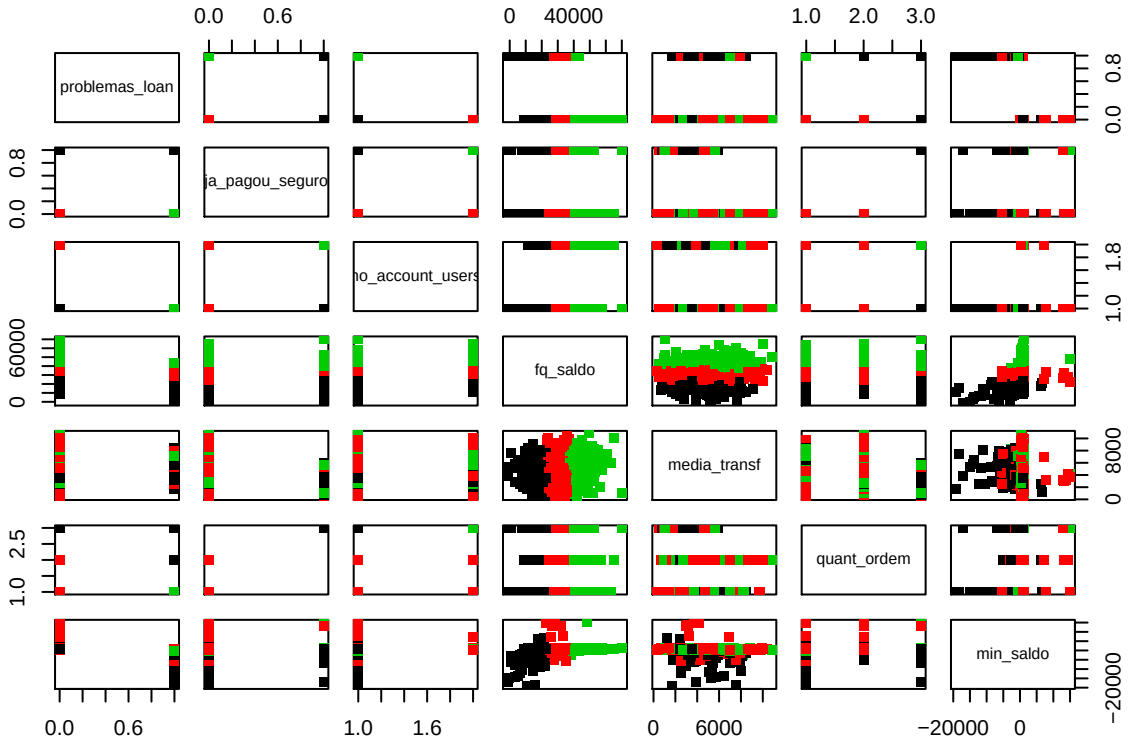
```
#Distancia - Soma dos quadrados entres os clusters
kmeans$betweenss
```

```
## [1] 72916452403
```

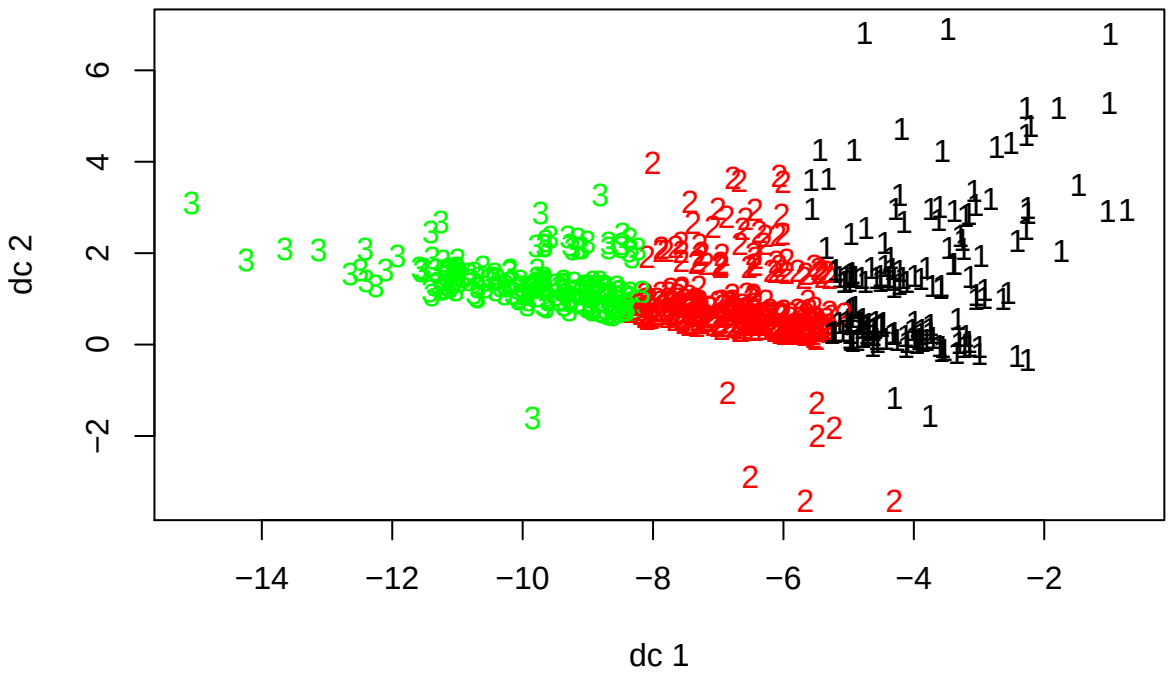
```
#Numero de pontos para cada cluster
kmeans$size
```

```
## [1] 166 316 200
```

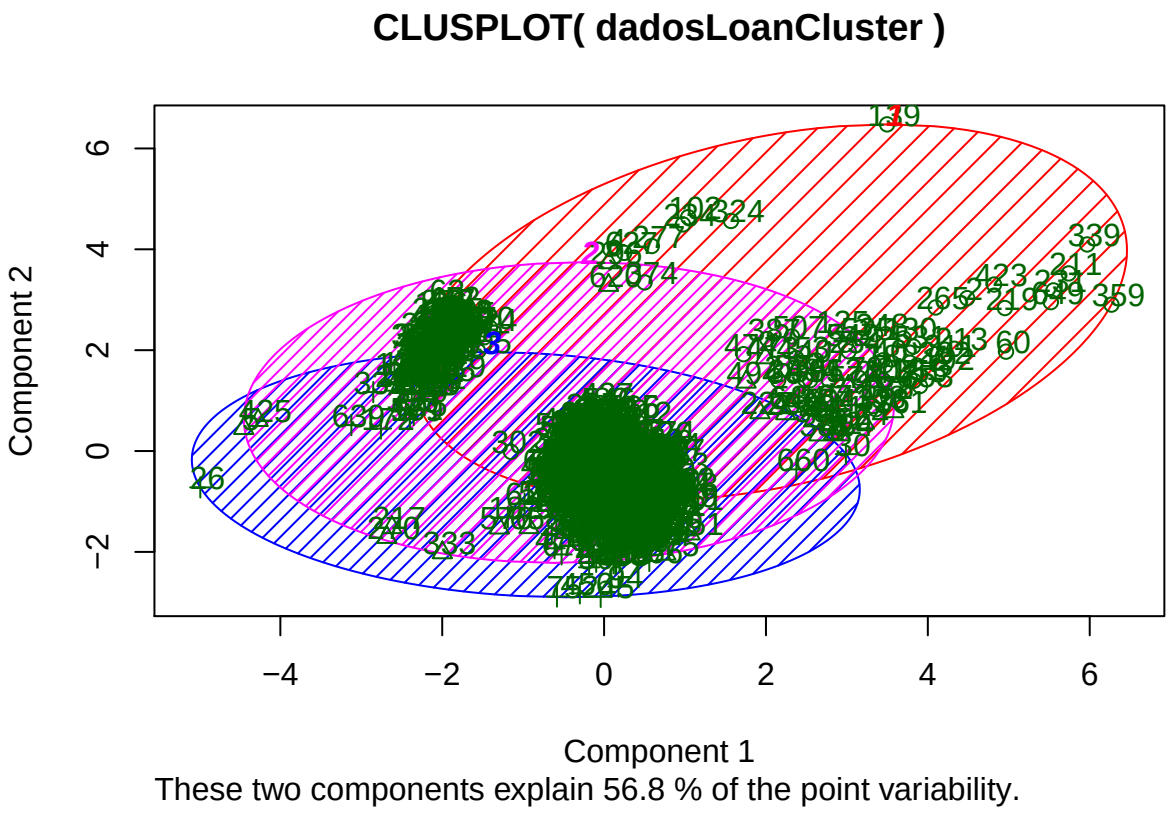
```
#Verificar Padrões

plot(dadosLoanCluster,col=kmeans$cluster,pch=15)
points(kmeans$centers,col=1:8,pch=3)
```




```
plotcluster(dadosLoanCluster,kmeans$cluster)
points(kmeans$centers,col=1:8,pch=16)
```




```
clusplot(dadosLoanCluster, kmeans$cluster, color=TRUE, shade=TRUE, labels=2, lines=0)
```



Ao analisar o gráfico “tabela”, podemos perceber a formação de cluster apenas entre FQ_saldo e Media_transf e quant_ordem e Min_saldo. Indicando que quanto maior o saldo maior, maior a media de transferencias e quantitativo de de ordem.

Conclusão

Os fatores que podem levar um cliente a ser um bom ou mal pagador são diversos, principalmente quando as informações que a base de dados tem sobre o cliente são limitadas, mas podemos identificar alguns sinais de que o cliente possa ser um mal pagador, ou possa parar de pagar uma dívida já contratada. Uma variável que consideramos bem forte para este sinal é a de saldo mínimo em conta, percebemos que pessoas com problemas de pagamento costumam atingir os menores saldos em conta, pode parecer uma informação óbvia, mas também é algo que nos permite identificar clientes que logo se tornarão mal pagadores e permitir que o banco crie estratégias para ajudar estes clientes. Por fim, após um análise de diversas variáveis em torno de clientes que possuem empréstimos, foi criado o modelo e clusters.