FGV MBA: Business Analytics & Big Data

July, 2019

Contents

1 Presentation

1.1 Exploratory data analysis

This website intends to present the work analysis for the *"Análise Exploratória de Dados"* class.

The ideia is Practicing R using a real anonymized Czech bank transactions, account info, and loan records released for PKDD'99 Discovery Challenge.

Use the menu above to navigate and see the final report.

1.1.1 Professor

- Gustavo Mirapalheta (gustavo.mirapalheta@fgv.br)

1.1.2 Authors / students

- Daniel Campos (A57635769 / daniel.ferraz.campos@gmail.com)
- Leandro Daniel (A57622988 / contato@leandrodaniel.com)
- Rodrigo Goncalves (A57566093 / rodrigo.goncalves@me.com)
- Ygor Lima (A57549661 / ygor_redesocial@hotmail.com)

1.1.3 Where to find the source code of this project?

This project can be found and downloaded on GitHub: https://github.com/ldaniel/R_Bank_Berka

Valar Morghulis! :)

1.2 About the data

Data from a real Czech bank. From 1999.

The data about the clients and their accounts consist of following relations:

- relation account (4500 objects in the file ACCOUNT.ASC) - each record describes static characteristics of an account,
- relation client (5369 objects in the file CLIENT.ASC) - each record describes characteristics of a client,
- relation disposition (5369 objects in the file DISP.ASC) - each record relates together a client with an account i.e. this relation describes the rights of clients to operate accounts,
- relation permanent order (6471 objects in the file ORDER.ASC) - each record describes characteristics of a payment order,
- relation transaction (1056320 objects in the file TRANS.ASC) - each record describes one transaction on an account,
- relation loan (682 objects in the file LOAN.ASC) - each record describes a loan granted for a given account,

- relation credit card (892 objects in the file CARD.ASC) - each record describes a credit card issued to an account,
- relation demographic data (77 objects in the file DISTRICT.ASC) - each record describes demographic characteristics of a district.

Each account has both static characteristics (e.g. date of creation, address of the branch) given in relation "account" and dynamic characteristics (e.g. payments debited or credited, balances) given in relations "permanent order" and "transaction". Relation "client" describes characteristics of persons who can manipulate with the accounts. One client can have more accounts, more clients can manipulate with single account; clients and accounts are related together in relation "disposition". Relations "loan" and "credit card" describe some services which the bank offers to its clients; more credit cards can be issued to an account, at most one loan can be granted for an account. Relation "demographic data" gives some publicly available information about the districts (e.g. the unemployment rate); additional information about the clients can be deduced from this.

Source: *This database was prepared by Petr Berka and Marta Sochorova.*

---

2 The Exploration's Report

2.1 Setting the scene

2.1.1 Introduction

Onde upon a time, there was a bank offering services to private persons. The services include managing of accounts, offerings loans, etc.

2.1.2 Task descripion

The bank wants to improve their services. For instance, the bank managers have only vague idea, who is a good client (whom to offer some additional services) and who is a bad client (whom to watch carefully to monimize the bank loses).

Fortunately, the bank stores data about their clients, the accounts (transactions within several months), the loans already granted, the credit cards issued.

The bank managers hope to improve their undestanding of customers and seed specific actions to improve services. A mere application of discovery tool will not be convincing for them.

2.1.3 Data description

This database was prepared by Petr Berka and Marta Sochorova.

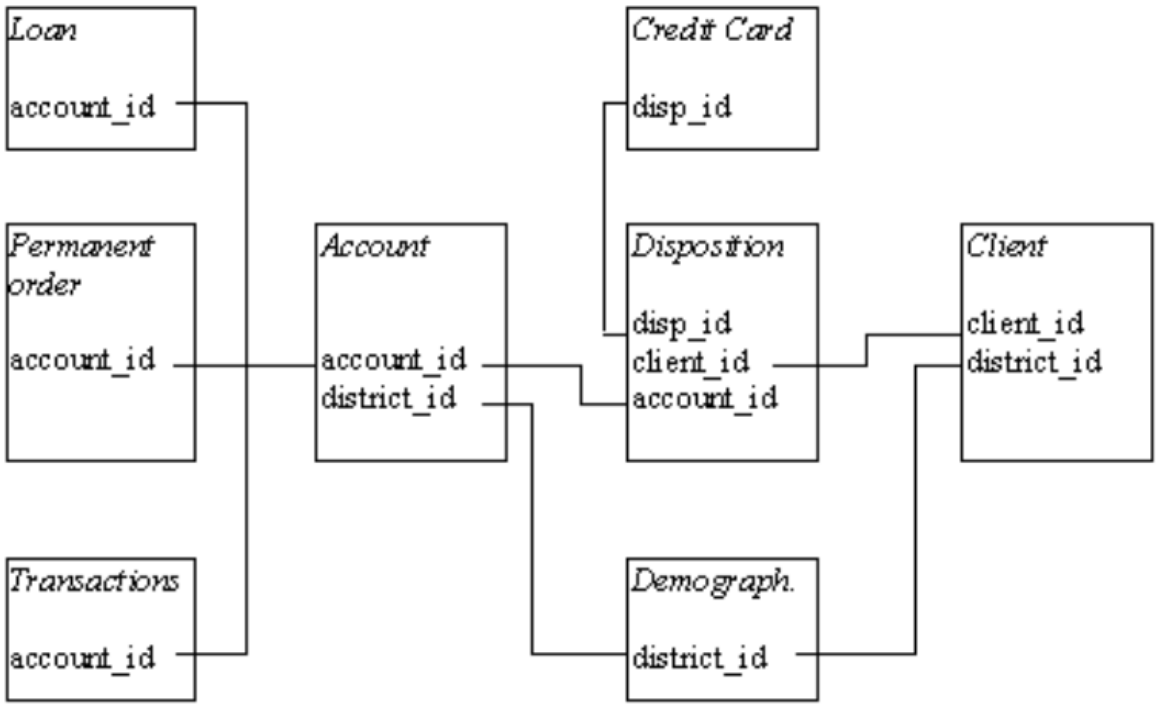


Figure 1: Simplified logical data model of Berka Bank.

2.2 Data ingestion, cleaning, translation and enhancement

Before starting the Berka Analysis, a few important steps were taken in order to prepare the source data files. These steps are listed below:

- **Step 01**: Create Functions;
- **Step 02**: Data Ingestion;
- **Step 03**: Data Cleaning;
- **Step 04**: Label Translation;
- **Step 05**: Data Enhancement.

2.2.1 Create Functions

This step create functions to be used in the next steps. Following, all functions created are described.

2.2.1.1 GetGenderFromBirthnumber

The birth_number column is given in the form of YYMMDD for men, and YYMM+50DD for women. The objective of this function is to return the gender of the client via the birth_number.

```
GetGenderFromBirthnumber <- function(var_birth_number) {
  month <- substr(var_birth_number, 3, 4)
  result <- ifelse(as.integer(month) > 50, "female", "male")
```

```
  return(as.factor(result))
}
```

2.2.1.2 GetBirthdateFromBirthnumber

The birth_number column is given in the form of YYMMDD for men, # and YYMM+50DD for women. The objective of this function is to return the final birthday as Date.

```
GetBirthdateFromBirthnumber <- function(var_birth_number, var_gender) {
  year <- paste("19", substr(var_birth_number, 1, 2), sep="")
  month <- ifelse(var_gender == "male", substr(var_birth_number, 3, 4),
  as.integer(substr(var_birth_number, 3, 4)) - 50)
  day <- substr(var_birth_number, 5, 6)
  result <- as.Date(paste(year, "-", month, "-", day, sep=""), format = "%Y-%m-%d")

  return(result)
}
```

2.2.1.3 ConvertToDate

The objective of this function is to convert the strange bank date style to the regular R Date datatype.

```
ConvertToDate <- function(var_date) {
  year <- paste("19", substr(var_date, 1, 2), sep="")
  month <- substr(var_date, 3, 4)
  day <- substr(var_date, 5, 6)
  result <- as.Date(paste(year, "-", month, "-", day, sep=""), format = "%Y-%m-%d")

  return(result)
}
```

2.2.1.4 GetAgeFromBirthnumber

The objective of this function is to get age given the birth_number.

```
GetAgeFromBirthnumber <- function(var_birth_number) {
  base_year <- 99 # considering 1999 as the base year for this exercise
  year <- substr(var_birth_number, 1, 2)
  result <- base_year - as.integer(year)

  return(result)
}
```

2.2.2 Data Ingestion

The process of data ingestion — preparing data for analysis — usually includes steps called extract (taking the data from its current location), transform (cleansing and normalizing the data), and load (placing the data in a database where it can be analyzed).

During this step, in addition to the loading data processes, it was performed data casting, column renaming and small touch-ups. The list below describe each table adjustment taken:

- **District**: renaming columns and casting columns with decimal or "?" values;
- **Credit Card**: casting column issued in creditcard table from string to datetime data type;
- **Account**: casting column date in account table from string to datetime data type;
- **Loan**: casting columns in table loan to the right data types;
- **Permanent Order**: casting columns with decimal values;
- **Transaction**: casting columns in table transaction to the right data types.

2.2.3 Data Cleaning

The objective of this step is analysing missing values and other strange conditions. In order to accomplish this task, a few R functions were used to quickly discover missing values, like NA and empty fields.

First thing done, was fixing observations in k_symbol transaction table with ' ' (one space) to empty string (''), using the following command.

```
transaction$k_symbol = trimws(transaction$k_symbol)
```

Then, the command below was used to find out any NA values in each table.

```
sapply(TableName, function(x) sum(is.na(x)))
```

Solely the **transaction** table has NA values, in the following columns:

	x
trans_id	0
account_id	0
date	0
type	0
operation	0
amount	0
balance	0
k_symbol	0
bank	0
account	760931

Finally, the following command was used in each table to find out where empty values was hidden.

```
sapply(TableName, function(x) table(as.character(x) =="")["TRUE"])
```

Again, only the **transaction** table had empty values, according to the table below:

	x
trans_id.NA	NA
account_id.NA	NA
date.NA	NA
type.NA	NA
operation.TRUE	183114
amount.NA	NA
balance.NA	NA
k_symbol.TRUE	535314

	x
bank.TRUE	782812
account.NA	NA

For the exploration analysis report, we did not take any additional action, since the missing values was not relevant.

2.2.4 Label Translation

In order to make the data information more understandable, it was translated some relevant labels and domains from Czech to English.

```
# Translating relevant labels and domains to english ------------------------------------------

disposition$type <- plyr::mapvalues(disposition$type, c('OWNER', 'DISPONENT'),
                                    c('Owner', 'User'))

account$frequency <- plyr::mapvalues(account$frequency,
                                     c('POPLATEK MESICNE', 'POPLATEK TYDNE',
                                       'POPLATEK PO OBRATU'),
                                     c('Monthly', 'Weekly', 'On Transaction'))

permanent_order$k_symbol <- plyr::mapvalues(permanent_order$k_symbol,
                                            c('POJISTNE', 'SIPO', 'LEASING', 'UVER'),
                                            c('insurrance payment', 'household',
                                              'leasing', 'loan payment'))

transaction$type <- plyr::mapvalues(transaction$type,
                                    c('PRIJEM', 'VYDAJ', 'VYBER'),
                                    c('credit', 'withdrawal', 'withdrawal in cash'))

transaction$operation <- plyr::mapvalues(transaction$operation,
                                         c('VYBER KARTOU', 'VKLAD', 'PREVOD Z UCTU',
                                           'VYBER', 'PREVOD NA UCET'),
                                         c('credit card withdrawal', 'credit in cash',
                                           'collection from another bank',
                                           'withdrawal in cash', 'remittance to nother bank'))

transaction$k_symbol <- plyr::mapvalues(transaction$k_symbol,
                                        c('POJISTNE', 'SLUZBY', 'UROK', 'SANKC. UROK',
                                          'SIPO', 'DUCHOD', 'UVER'),
                                        c('insurance payment', 'statement',
                                          'interest credited', 'sanction interest',
                                          'household', 'old age pension', 'loan payment'))
```

2.2.5 Data Enhancement

This step aims to improve the analysis by adding auxiliary information. Data enhancement is all about making sure any data that is coming into the business is being looked at with a critical eye and is being filtered down to maximize its value.

The code below get gender, birthday and age from birth_number column using *GetGenderFromBirthnumber* and *GetBirthdateFromBirthnumber* functions.

```
client <- client %>%
  mutate(gender = GetGenderFromBirthnumber(birth_number)) %>%
  mutate(birth_date = GetBirthdateFromBirthnumber(birth_number, gender)) %>%
  mutate(age = GetAgeFromBirthnumber(birth_number))
```

The code below improved loan data by having a classification regarding its payment status.

```
loan <- mutate(loan, defaulter =
                as.logical(plyr::mapvalues(status, c ('A','B','C','D'),
                                           c(FALSE,TRUE,FALSE,TRUE))),
                contract_status = plyr::mapvalues(status, c ('A','B','C','D'),
                                  c('finished','finished','running','running')),
                                  type = 'Owner')
```

The code below improved client data by having its age group.

```
client <- mutate(client, age_bin = paste(findInterval(age,
                 c(10, 20, 30, 40, 50, 60, 70, 80, 90, 100)) * 10,'+'))
```

The code below calculate an additional table with current and average account balance for each account.

```
account_balance <- arrange(transaction, desc(date), account_id) %>%
  group_by(account_id) %>%
  mutate(avg_balance = mean(balance)) %>%
  filter(row_number() == 1) %>%
  dplyr::select(account_id, date, balance, avg_balance)

colnames(account_balance) <- c("account_id", "last_transaction_date", 'account_balance', 'avg_balance')
```

The code below calculate an additional table with the proportion of each transaction type (k_symbol) on total transaction amount of each account. That data will be used to fit a logistic model on loan dataset.

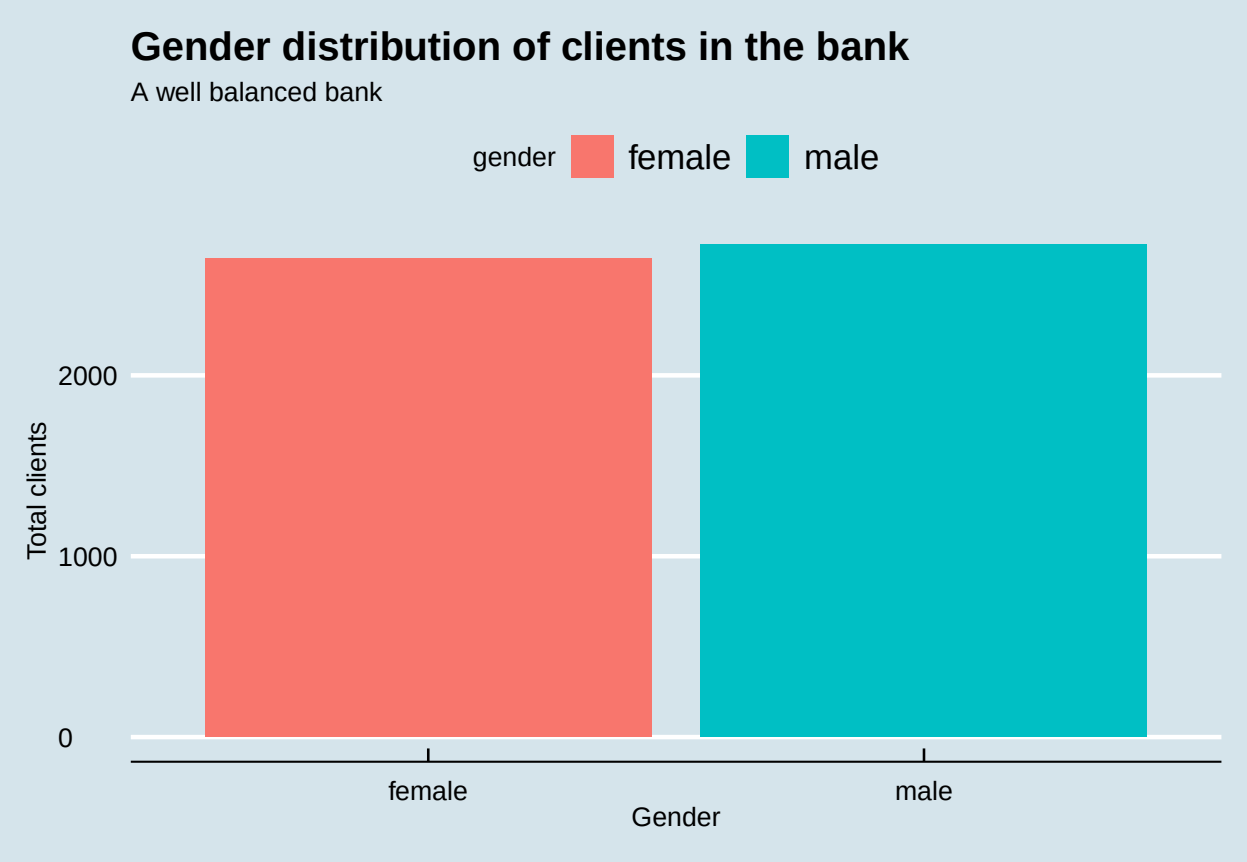
```
account_transaction_pattern <- select(transaction, c(trans_id, account_id, date, amount, k_symbol)) %>%
    mutate(k_symbol = ifelse(k_symbol == '' | is.na(k_symbol), 'other', k_symbol)) %>%
    spread(key = k_symbol, value = amount) %>%
    replace(is.na(.), 0) %>%
    mutate(amount = rowSums(.[4:11])) %>%
    group_by(account_id) %>%
    summarise(transaction_count = n(),
              last_transaction_date = max(date),
              amount = sum(amount),
              prop_household = sum(household) / amount,
              prop_insurance_payment = sum(`insurance payment`) / amount,
              prop_interest_credited = sum(`interest credited`) / amount,
              prop_loan_payment = sum(`loan payment`) / amount,
              prop_old_age_pension = sum(`old age pension`) / amount,
              prop_other = sum(`other`) / amount,
              prop_sanction_interest = sum(`sanction interest`) / amount,
              prop_statement = sum(`statement`) / amount)
```

2.3 The explorations

2.3.1 Gender Exploration

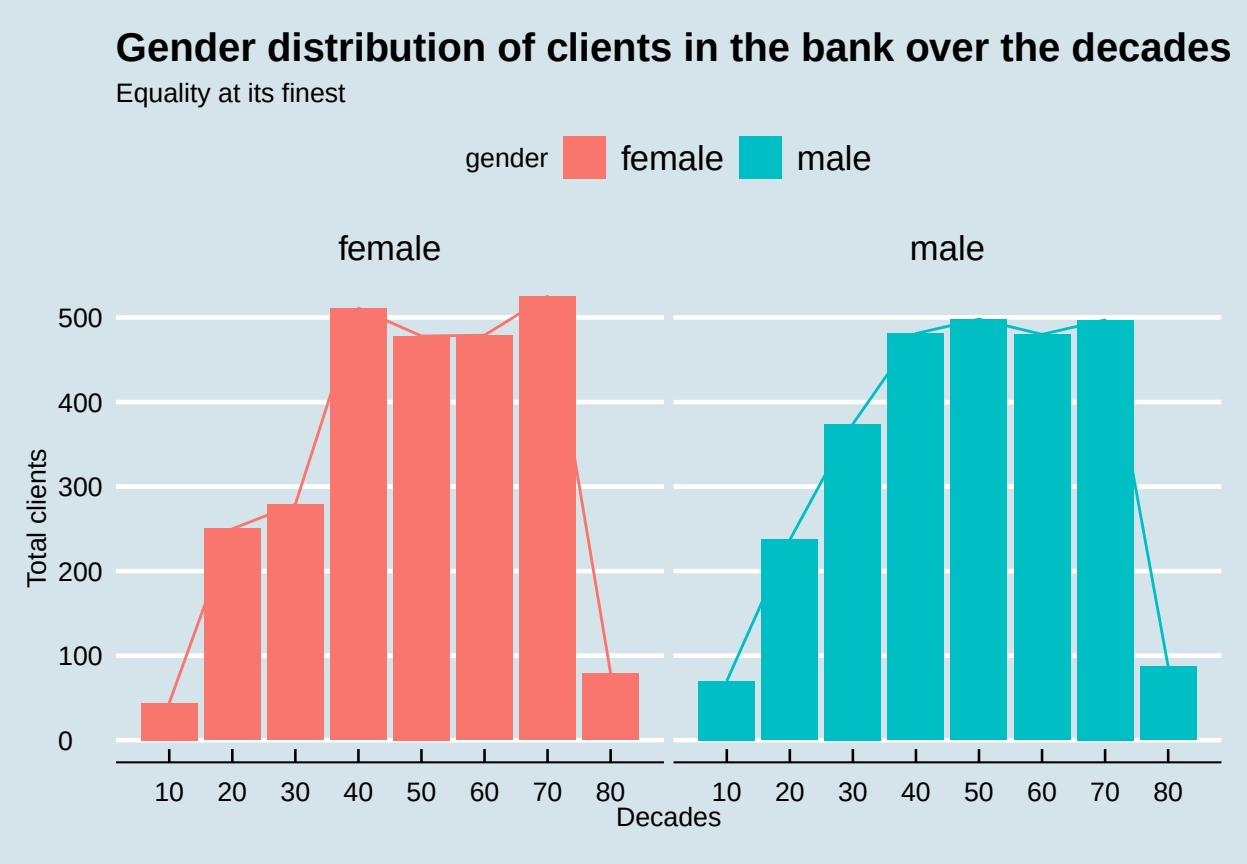
At first glance, gender equality is well balanced in the bank, even when observed over the decades. Even more impressive, gender equality is everywhere in the country.

```
# gender distribution of clients in the bank
ggplot(data = client) +
  aes(x = gender, fill = gender) +
  geom_bar() +
  labs(title = "Gender distribution of clients in the bank",
       subtitle = "A well balanced bank",
       x = "Gender",
       y = "Total clients") +
  theme_economist()
```



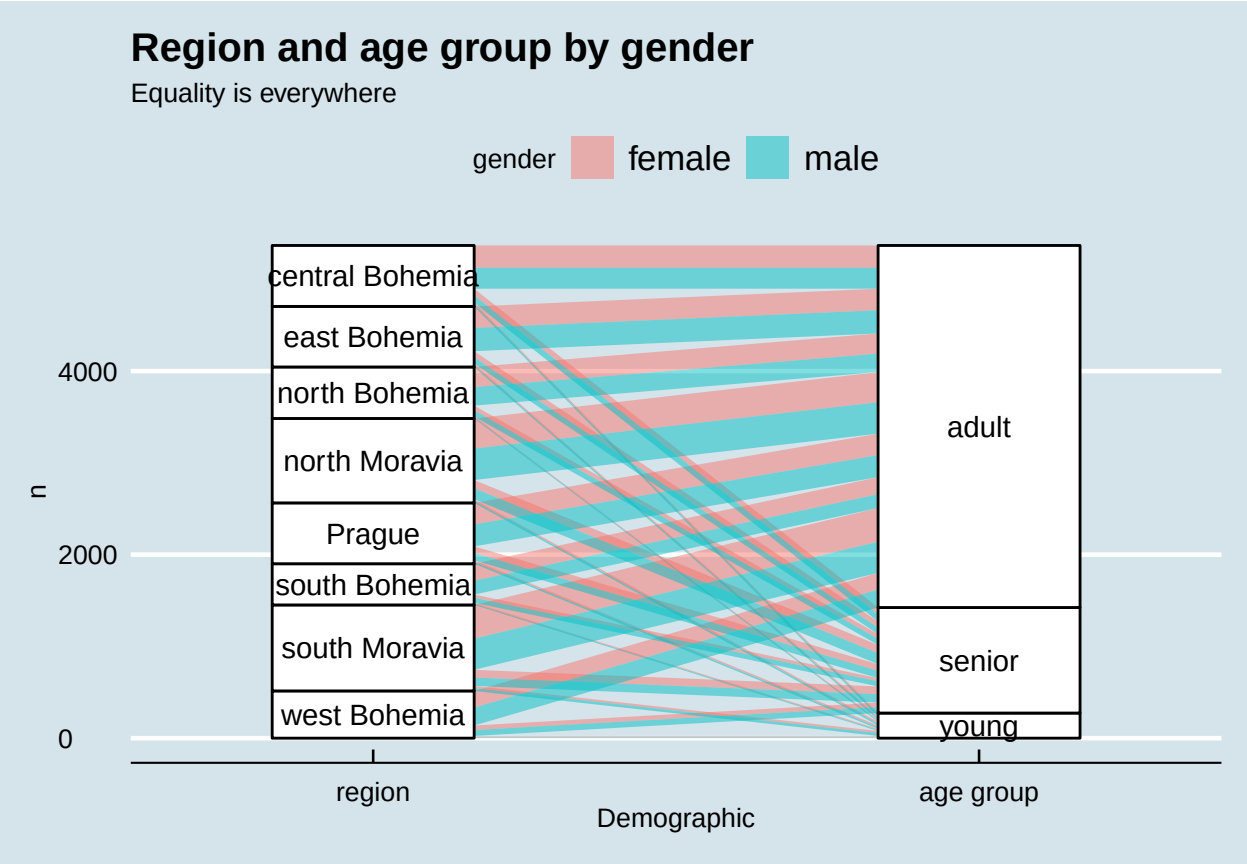

```
clientGenderOverDecades <- client %>%
  group_by(decade = as.integer(substr(client$birth_number, 1,1)) * 10,
           gender = client$gender) %>%
  count()

# gender distribution of clients in the bank over the decades
ggplot(data = clientGenderOverDecades) +
  aes(x = decade, fill = gender, weight = n) +
```

```
  scale_x_continuous(breaks = c(0, 10, 20, 30, 40, 50, 60, 70, 80)) +
  geom_bar() +
  geom_line(aes(y = n, color = gender)) +
  labs(title = "Gender distribution of clients in the bank over the decades",
       subtitle = "Equality at its finest",
       x = "Decades",
       y = "Total clients") +
  theme_economist() +
  facet_wrap(vars(gender))
```




```
# alluvial diagram representation of gender, age group and region
clientGenderAgeGroupByRegion <- client %>%
  mutate(age_group = ifelse(age < 21, "young",
                     ifelse(age >= 21 & age <= 60, "adult", "senior"))) %>%
  inner_join(district, by = "district_id") %>%
  group_by(age_group, gender, region) %>%
  count()

ggplot(data = clientGenderAgeGroupByRegion,
       aes(axis1 = region, axis2 = age_group, y = n)) +
  scale_x_discrete(limits = c("region", "age group"), expand = c(.1, .1)) +
  xlab("Demographic") +
  geom_alluvium(aes(fill = gender), knot.pos = 0) +
  geom_stratum() +
  geom_text(stat = "stratum", label.strata = TRUE) +
```

```
  theme_economist() +
  ggtitle("Region and age group by gender", "Equality is everywhere")
```

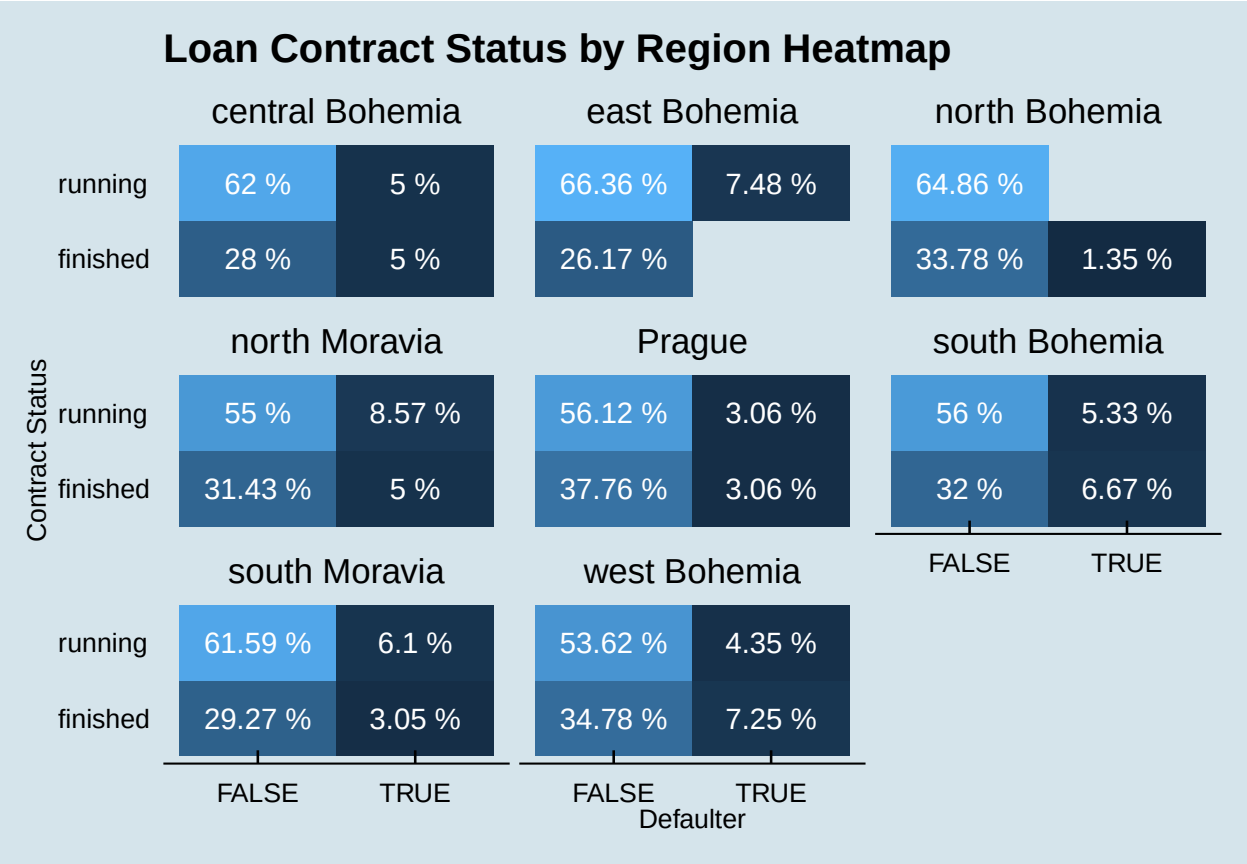


2.3.2 Loan Exploration

Here we investigate if there is any association between the region and the likelihood of default in the 682 loan observations in the dataset.

```
left_join(loan, disposition, by = 'account_id') %>%
  left_join(client, by = 'client_id') %>%
  left_join(district, by = 'district_id') %>%
  group_by(region, contract_status, defaulter) %>%
  summarise(count = n(),
            amount = sum(amount)) %>%
  group_by(region, contract_status) %>%
  mutate(count_contract_status = sum(count),
         amount_contract_status = sum(amount)) %>%
  group_by(region) %>%
  mutate(count_region = sum(count),
         amount_region = sum(amount)) %>%
  ggplot(aes(x = defaulter, y = contract_status, fill = count / count_region)) +
  geom_bin2d(stat = 'identity') +
  geom_text(aes(label = paste(round(count / count_region * 100, 2), '%')),
            color = 'white') +
```

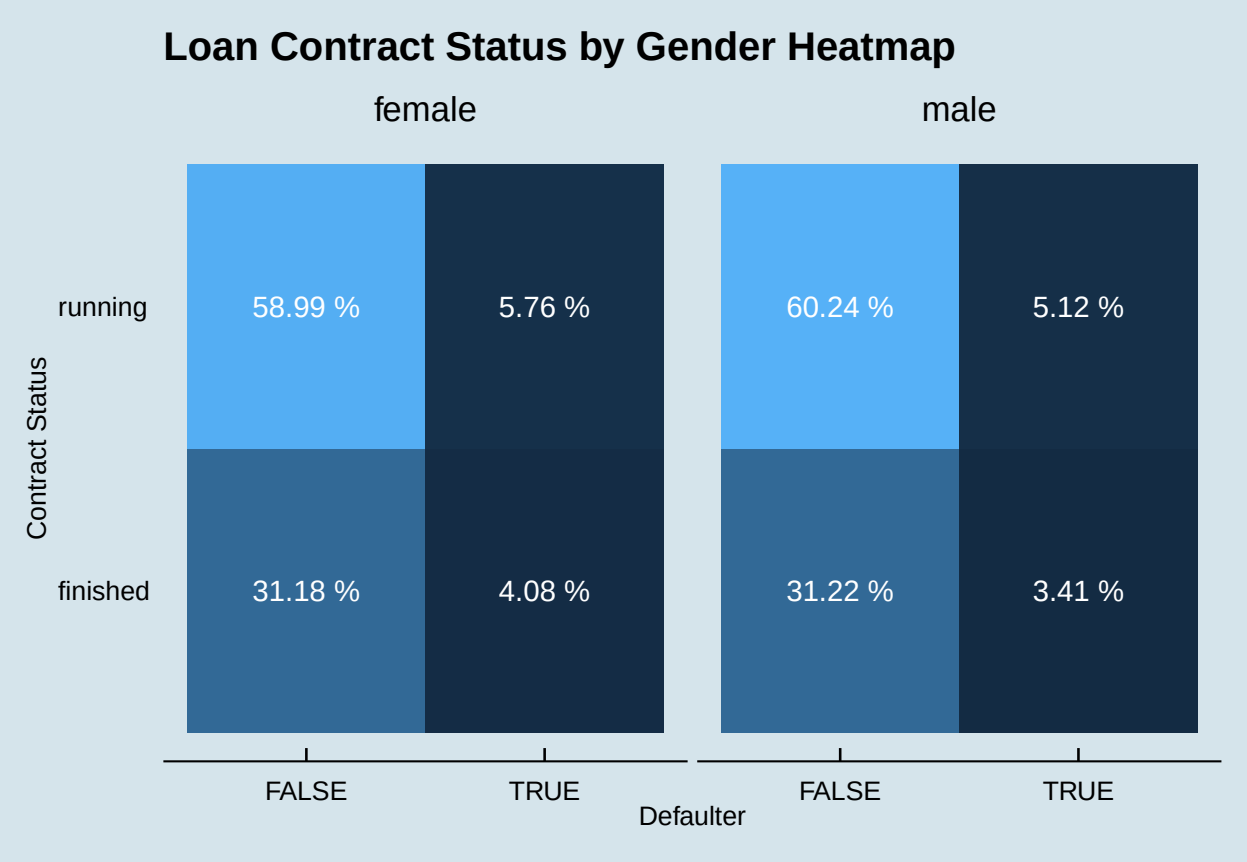
```
  facet_wrap(~region) +
  theme_economist() +
  theme(legend.position = 'none', panel.grid.major = element_blank(),
        panel.grid.minor = element_blank()) +
  labs(x = 'Defaulter',
       y = 'Contract Status',
       title = 'Loan Contract Status by Region Heatmap')
```



We perform the same investigation on the association between the client gender and the likelihood of default in the 682 loan observations in the dataset.

```
left_join(loan, disposition, by = 'account_id') %>%
  left_join(client, by = 'client_id') %>%
  left_join(district, by = 'district_id') %>%
  group_by(gender, contract_status, defaulter) %>%
  summarise(count = n(),
            amount = sum(amount)) %>%
  group_by(gender, contract_status) %>%
  mutate(count_contract_status = sum(count),
         amount_contract_status = sum(amount)) %>%
  group_by(gender) %>%
  mutate(count_gender = sum(count),
         amount_gender = sum(amount)) %>%
  ggplot(aes(x = defaulter, y = contract_status,
             fill = count / count_gender)) +
    geom_bin2d(stat = 'identity') +
```

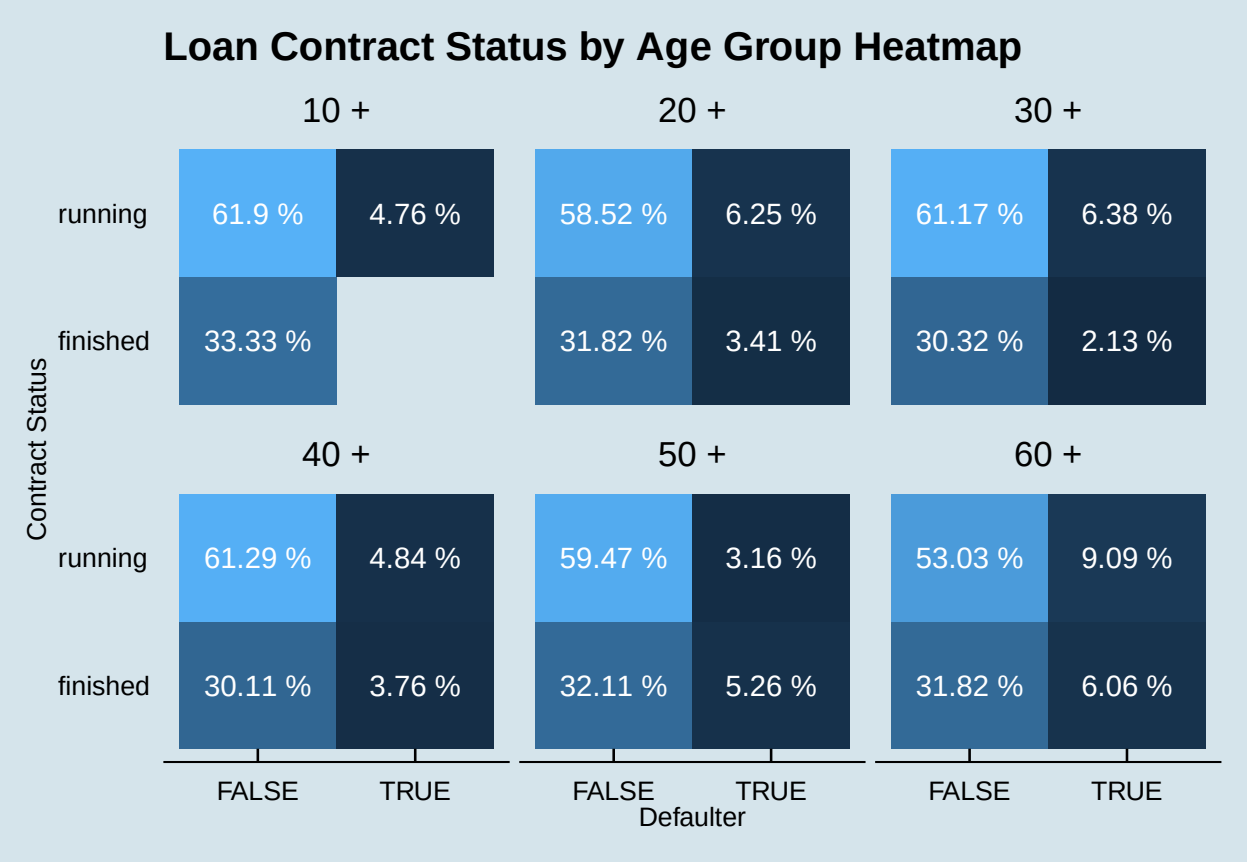
```
    geom_text(aes(label = paste(round(count / count_gender * 100, 2), '%')),
              color = 'white') +
    facet_wrap(~gender) +
    theme_economist() +
    theme(legend.position = 'none', panel.grid.major = element_blank(),
          panel.grid.minor = element_blank()) +
    labs(x = 'Defaulter',
         y = 'Contract Status',
         title = 'Loan Contract Status by Gender Heatmap')
```



We finnaly do the same investigation on the association between the client age and the likelihood of default in the 682 loan observations in the dataset.

```
left_join(loan, disposition, by = 'account_id') %>%
  left_join(client, by = 'client_id') %>%
  left_join(district, by = 'district_id') %>%
  group_by(age_bin, contract_status, defaulter) %>%
  summarise(count = n(),
            amount = sum(amount)) %>%
  group_by(age_bin, contract_status) %>%
  mutate(count_contract_status = sum(count),
         amount_contract_status = sum(amount)) %>%
  group_by(age_bin) %>%
  mutate(count_age_bin = sum(count),
         amount_age_bin = sum(amount)) %>%
  ggplot(aes(x = defaulter,
```

```
          y = contract_status, fill = count / count_age_bin)) +
  geom_bin2d(stat = 'identity') +
  geom_text(aes(label = paste(round(count / count_age_bin * 100, 2), '%')),
            color = 'white') +
  facet_wrap(~age_bin) +
  theme_economist() +
  theme(legend.position = 'none', panel.grid.major = element_blank(),
        panel.grid.minor = element_blank()) +
  labs(x = 'Defaulter',
       y = 'Contract Status',
       title = 'Loan Contract Status by Age Group Heatmap')
```



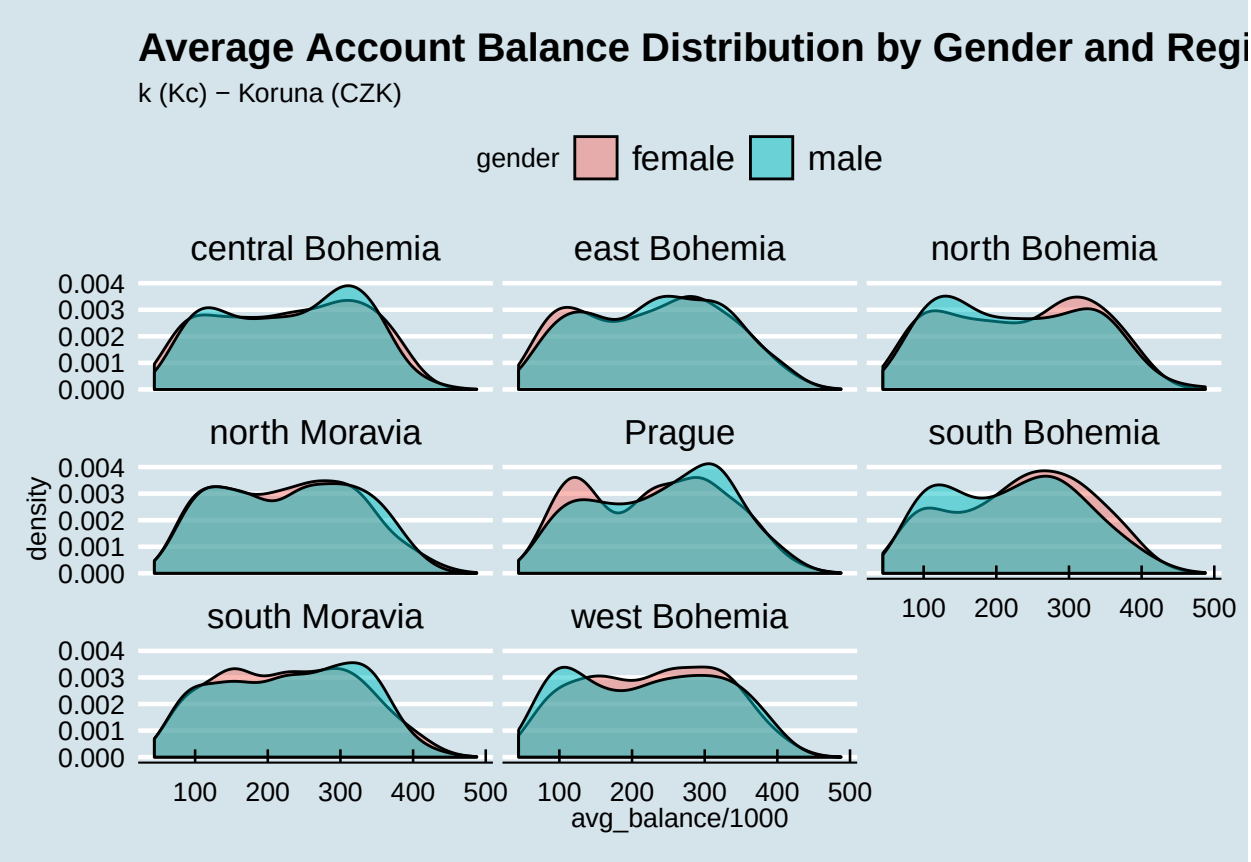
It seems that none of these features, alone, are determinant on the odds of a client default.

2.3.3 Account Balance Exploration

As we do not have the individual income of each client, we tried to perform a proxy of the client's wealth by calculating the average account balance of each account and investigating the distribution by region and age.

```
left_join(account_balance, disposition, by = 'account_id') %>%
  left_join(client, by = 'client_id') %>%
  left_join(district, by = 'district_id') %>%
  filter(type == 'Owner') %>%
  ggplot(aes(avg_balance / 1000)) +
```

```
    geom_density(alpha = 0.5, aes(fill = gender)) +
    scale_x_continuous(labels = scales::comma) +
    labs(title = 'Average Account Balance Distribution by Gender and Region',
         subtitle = 'k (Kč) - Koruna (CZK)') +
    theme_economist() +
    facet_wrap(~region)
```



One interest pattern shows up, males in prague region tend to have a bigger average account balance them females, and the oposite trend in South, North and West Bohemia.

2.3.4 District exploration

Regarding the Czech regions, we can notice in the exploration below which regions are more likely to have defaulters based on the finished loans operations so far.

```
loan_amount_by_region <- select(loan, account_id, amount, defaulter, contract_status) %>%
  filter(defaulter == TRUE) %>%
  inner_join(account) %>%
  inner_join(district) %>%
  group_by(region) %>%
  summarise(transaction_count = n(),
            amount = sum(amount)) %>%
  inner_join(czech_regions_coords)

jsonMapFile <- "../map/czech-republic-regions.json"
```

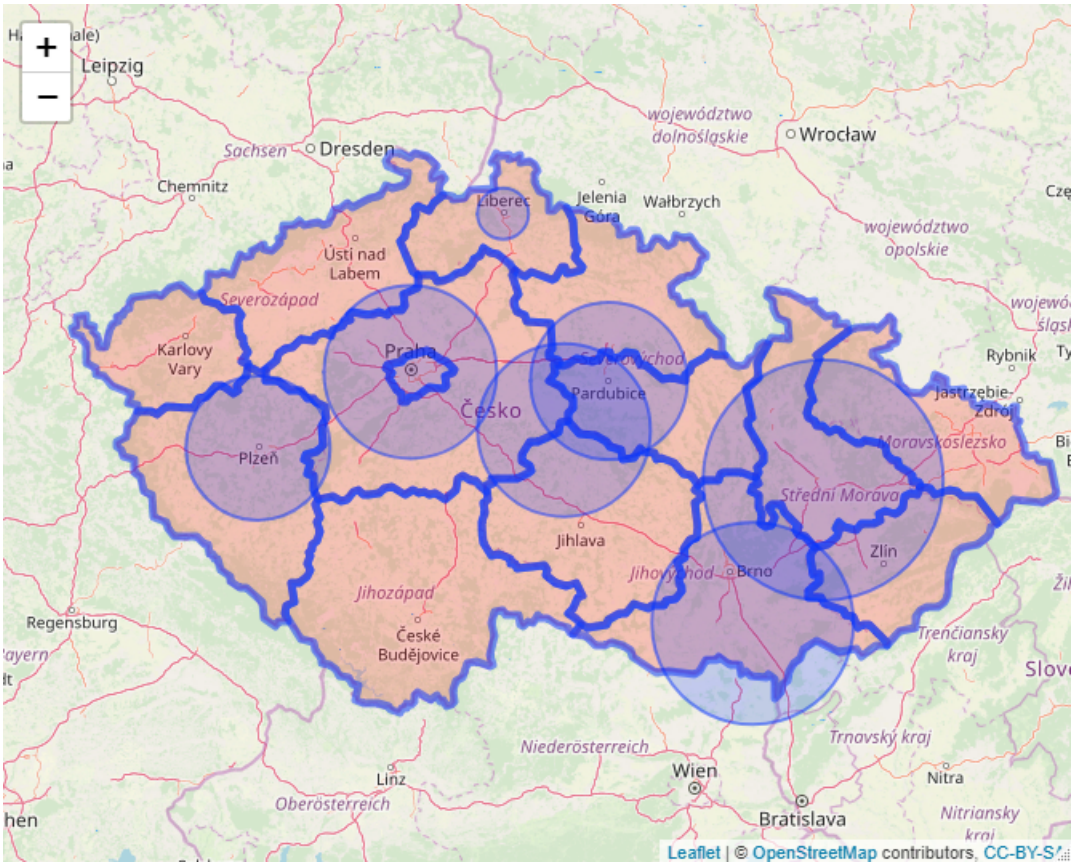


Figure 2: Regions more likely to have defaulters.

```
czech_regions <- as.json(geojson_read(jsonMapFile))

leaflet(loan_amount_by_region) %>%
  addTiles() %>%
  setView(lng = 15.3, lat = 49.8, zoom = 7) %>%
  addGeoJSON(czech_regions, fillColor = "red", stroke = "#555555") %>%
  addCircles(lng = ~long,
             lat = ~lat,
             weight = 2,
             radius = ~sqrt(amount) * 30,
             popup = ~region)
```

---

3 The logistic regression on loan's report

3.1 Objective

The goal of this report is trying to fit a logistic regression model on Loan data aiming to predict the probability of delinquency for each contract.

3.2 Task description

3.2.1 Dataset preparation

Using the vanilla transaction dataset, we calculated several derived variables for each account.

First, we calculated an additional table with the current account balance and average account balance of each account.

Later on, we calculated another auxiliary table that contains the proportion of each kind of transaction (k_symbol) for each account. The idea of these variable is to capture the spend pattern of each client.

Finally, we combine the 682 Loan Contracts observations with client, district, credit card and the auxiliary tables we calculated early.

```
temp <- left_join(loan, disposition, by = c('account_id', 'type')) %>%
  left_join(client, by = 'client_id') %>%
  left_join(district, by = 'district_id') %>%
  left_join(creditcard, by = 'disp_id') %>%
  left_join(account_balance, by = 'account_id') %>%
  left_join(account_transaction_pattern, by = 'account_id') %>%
  mutate(card_age_month = (issued %--%
                             make_date(1998, 12, 31)) / months(1),
         last_transaction_age_days = ((last_transaction_date.y %--%
                                         make_date(1998, 12, 31)) / days(1)),
         has_card = ifelse(type.y == 'no card', 0, 1)) %>%
  dplyr::select(c("amount.x", "duration", "payments", "status", "defaulter",
                  "contract_status", "gender", "age", "district_name",
                  "region", "no_of_inhabitants",
                  "no_of_municip_inhabitants_less_499",
                  "no_of_municip_500_to_1999", "no_of_municip_2000_to_9999",
                  "no_of_municip_greater_10000", "no_of_cities",
                  "ratio_of_urban_inhabitants",
                  "average_salary", "unemploymant_rate_1995",
                  "unemploymant_rate_1996",
                  "no_of_enterpreneurs_per_1000_inhabitants",
                  "no_of_commited_crimes_1995",
                  "no_of_commited_crimes_1996", "type.y",
                  "card_age_month","account_balance",
                  "avg_balance","transaction_count", "amount.y",
                  "last_transaction_age_days", "prop_old_age_pension",
                  "prop_insurance_payment",
                  "prop_sanction_interest","prop_household",
                  "prop_statement", "prop_interest_credited",
                  "prop_loan_payment", "prop_other", "has_card"))

colnames(temp) <- c("x_loan_amount", "x_loan_duration", "x_loan_payments",
                    "x_loan_status", "y_loan_defaulter", "x_loan_contract_status",
                    "x_client_gender", "x_client_age",
                    "x_district_name", "x_region",
                    "x_no_of_inhabitants", "x_no_of_municip_inhabitants_less_499",
                    "x_no_of_municip_500_to_1999", "x_no_of_municip_2000_to_9999",
                    "x_no_of_municip_greater_10000", "x_no_of_cities",
```

```
                    "x_ratio_of_urban_inhabitants",
                    "x_average_salary", "x_unemploymant_rate_1995",
                    "x_unemploymant_rate_1996",
                    "x_no_of_enterpreneurs_per_1000_inhabitants",
                    "x_no_of_commited_crimes_1995",
                    "x_no_of_commited_crimes_1996", "x_card_type",
                    "x_card_age_month","x_account_balance",
                    "x_avg_account_balance","x_transaction_count",
                    "x_transaction_amount", "x_last_transaction_age_days",
                    "x_prop_old_age_pension", "x_prop_insurance_payment",
                    "x_prop_sanction_interest","x_prop_household","x_prop_statement",
                    "x_prop_interest_credited", "x_prop_loan_payment", "x_prop_other",
                    "x_has_card")

temp <- dplyr::select(temp, y_loan_defaulter, everything())

temp$x_card_type = ifelse(is.na(temp$x_card_type), 'no card',
                          as.character(temp$x_card_type))

temp$x_has_card = ifelse(temp$x_card_type == 'no card', 0, 1)

temp$x_card_age_month = ifelse(is.na(temp$x_card_age_month), 0,
                               temp$x_card_age_month)

temp$y_loan_defaulter = as.numeric(temp$y_loan_defaulter)
```

We ended up having a data set with 39 variables.

variables
y_loan_defaulter
x_loan_amount
x_loan_duration
x_loan_payments
x_loan_status
x_loan_contract_status
x_client_gender
x_client_age
x_district_name
x_region
x_no_of_inhabitants
x_no_of_municip_inhabitants_less_499
x_no_of_municip_500_to_1999
x_no_of_municip_2000_to_9999
x_no_of_municip_greater_10000
x_no_of_cities
x_ratio_of_urban_inhabitants
x_average_salary
x_unemploymant_rate_1995
x_unemploymant_rate_1996
x_no_of_enterpreneurs_per_1000_inhabitants
x_no_of_commited_crimes_1995
x_no_of_commited_crimes_1996
x_card_type

variables
x_card_age_month
x_account_balance
x_avg_account_balance
x_transaction_count
x_transaction_amount
x_last_transaction_age_days
x_prop_old_age_pension
x_prop_insurance_payment
x_prop_sanction_interest
x_prop_household
x_prop_statement
x_prop_interest_credited
x_prop_loan_payment
x_prop_other
x_has_card

3.2.2 Variable selection

From this dataset, we excluded 6 variables that are redundant, shows no variability on the 682 Loan contract observations or have no applicability for the exercise:

- **x_prop_old_age_pension** (no variation on selected sample);
- **x_district_name** (sample have not enough variability to fit a model);
- **x_region** (sample have not enough variability to fit a model);
- **x_loan_status** (redundant);
- **x_loan_contract_status** (redundant);
- **x_prop_sanction_interest** (seems not to be valid for this analysis, as the reason for interest payment in the account is exactly the delinquency in the observation contact itself).

```
temp <- dplyr::select(temp, -c(x_prop_old_age_pension,
                               x_district_name,
                               x_region,
                               x_loan_status,
                               x_loan_contract_status,
                               x_prop_sanction_interest))
```

3.2.3 Investigating Multicollinearity

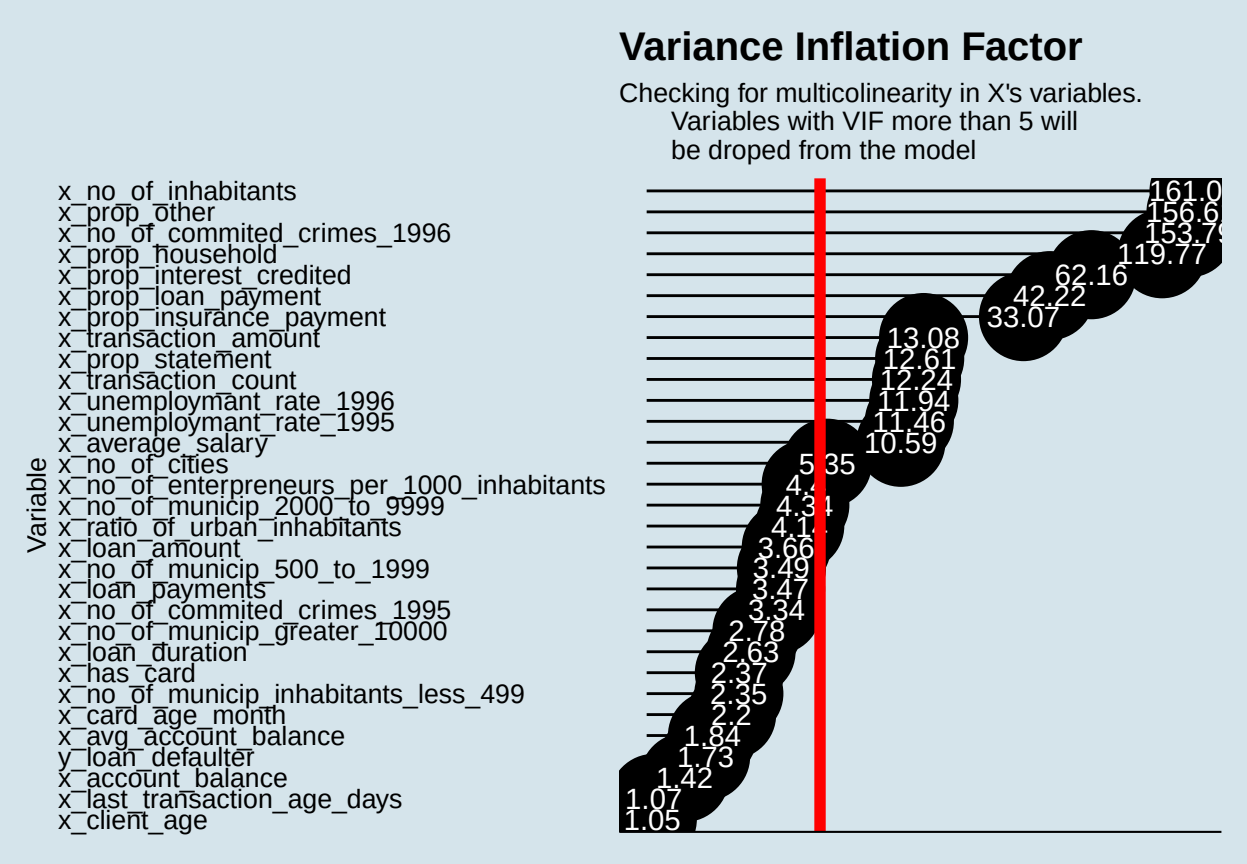
With the remaining variables we ran a multicollinearity test to identify additional variables to drop from the model specification.

```
vars.quant <- select_if(temp, is.numeric)
VIF <- imcdiag(vars.quant, temp$y_loan_defaulter)

VIF_Table_Before <- tibble(variable = names(VIF$idiags[,1]),
                    VIF = VIF$idiags[,1]) %>%
             arrange(desc(VIF))

kable(VIF_Table_Before)
```

variable	VIF
x_no_of_inhabitants	161.078688
x_prop_other	156.614374
x_no_of_commited_crimes_1996	153.788642
x_prop_household	119.766648
x_prop_interest_credited	62.164451
x_prop_loan_payment	42.215533
x_prop_insurance_payment	33.065760
x_transaction_amount	13.077575
x_prop_statement	12.611010
x_transaction_count	12.243487
x_unemploymant_rate_1996	11.942010
x_unemploymant_rate_1995	11.457609
x_average_salary	10.592530
x_no_of_cities	5.348665
x_no_of_enterpreneurs_per_1000_inhabitants	4.398114
x_no_of_municip_2000_to_9999	4.342691
x_ratio_of_urban_inhabitants	4.141633
x_loan_amount	3.656975
x_no_of_municip_500_to_1999	3.494354
x_loan_payments	3.467355
x_no_of_commited_crimes_1995	3.343160
x_no_of_municip_greater_10000	2.784531
x_loan_duration	2.634738
x_has_card	2.365295
x_no_of_municip_inhabitants_less_499	2.353720
x_card_age_month	2.200021
x_avg_account_balance	1.842499
y_loan_defaulter	1.728511
x_account_balance	1.422176
x_last_transaction_age_days	1.068578
x_client_age	1.053711

```
ggplot(VIF_Table_Before, aes(x = fct_reorder(variable, VIF), y = log(VIF), label = round(VIF, 2))) +
  geom_point(stat='identity', fill="black", size=15)  +
  geom_segment(aes(y = 0,
                   yend = log(VIF),
                   xend = variable),
               color = "black") +
  geom_text(color="white", size=4) +
  geom_hline(aes(yintercept = log(5)), color = 'red', size = 2) +
  scale_y_continuous(labels = NULL, breaks = NULL) +
  coord_flip() +
  theme_economist() +
  theme(legend.position = 'none',
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank()) +
  labs(x = 'Variable',
       y = NULL,
       title = 'Variance Inflation Factor',
       subtitle="Checking for multicolinearity in X's variables.
       Variables with VIF more than 5 will
```

```
      be droped from the model")
```



We decided to exclude any variable with a VIF greater than 5.

Below variables were excluded based on the multicollinear presence on them.

- **x_prop_insurance_payment**;
- **x_prop_household**;
- **x_prop_statement** ;
- **x_prop_loan_payment**;
- **x_prop_other**;
- **x_no_of_inhabitants**;
- **x_no_of_commited_crimes_1996**;
- **x_transaction_amount**;
- **x_transaction_count**;
- **x_unemploymant_rate_1996**;
- **x_unemploymant_rate_1995**;
- **x_average_salary**;
- **x_no_of_cities**.

```
temp <- dplyr::select(temp, -c(x_prop_insurance_payment,
                               x_prop_household,
                               x_prop_statement,
                               x_prop_loan_payment,
                               x_prop_other,
```

```
                                x_no_of_inhabitants,
                                x_no_of_commited_crimes_1996,
                                x_transaction_amount,
                                x_transaction_count,
                                x_unemploymant_rate_1996,
                                x_unemploymant_rate_1995,
                                x_average_salary,
                                x_no_of_cities))

loan_reg_dataset <- temp
```

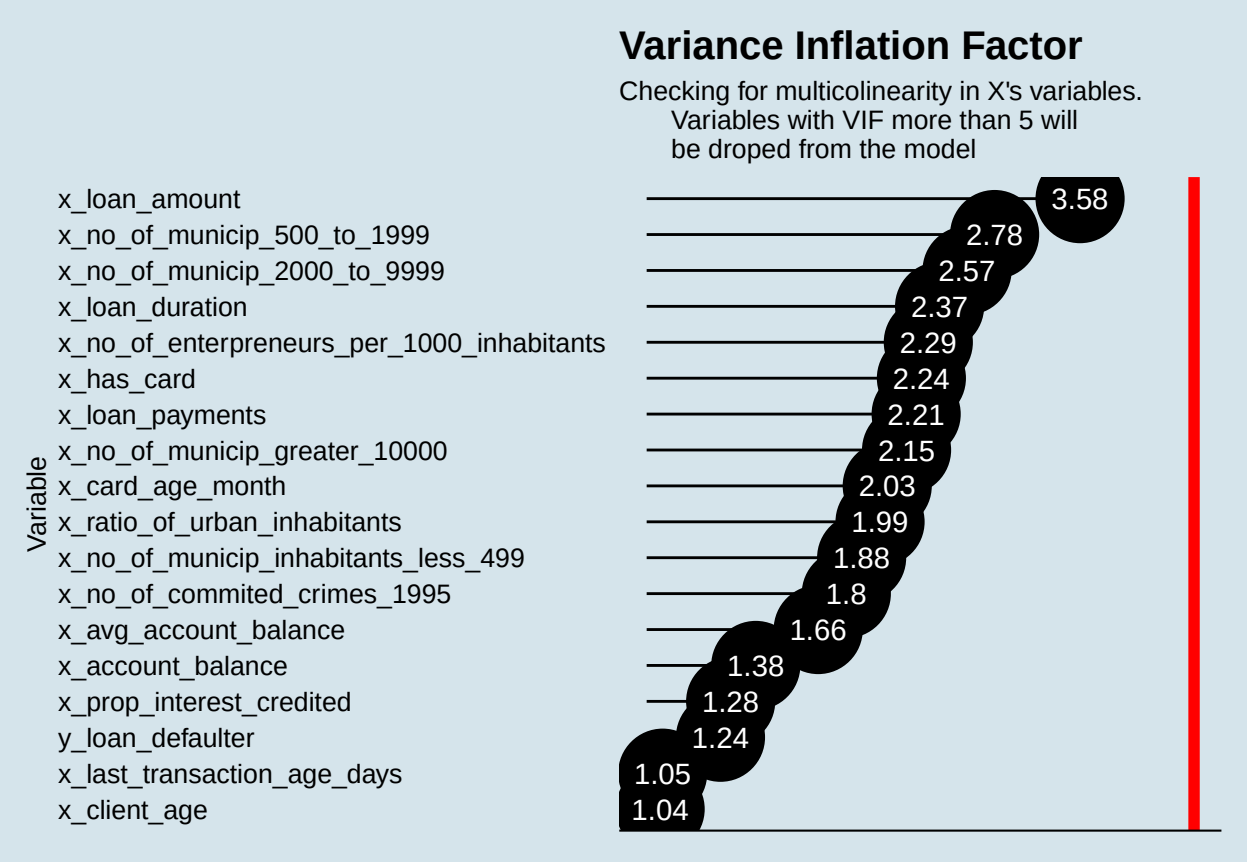
Here is the final correlation matrix we got:

variable	VIF
x_loan_amount	3.576281
x_no_of_municip_500_to_1999	2.781569
x_no_of_municip_2000_to_9999	2.565167
x_loan_duration	2.366083
x_no_of_enterpreneurs_per_1000_inhabitants	2.289033
x_has_card	2.242915
x_loan_payments	2.208547
x_no_of_municip_greater_10000	2.147182
x_card_age_month	2.028568
x_ratio_of_urban_inhabitants	1.986178
x_no_of_municip_inhabitants_less_499	1.881482
x_no_of_commited_crimes_1995	1.798576
x_avg_account_balance	1.656011
x_account_balance	1.378326
x_prop_interest_credited	1.280158
y_loan_defaulter	1.242373
x_last_transaction_age_days	1.048220
x_client_age	1.040390

Plot of VIF values:

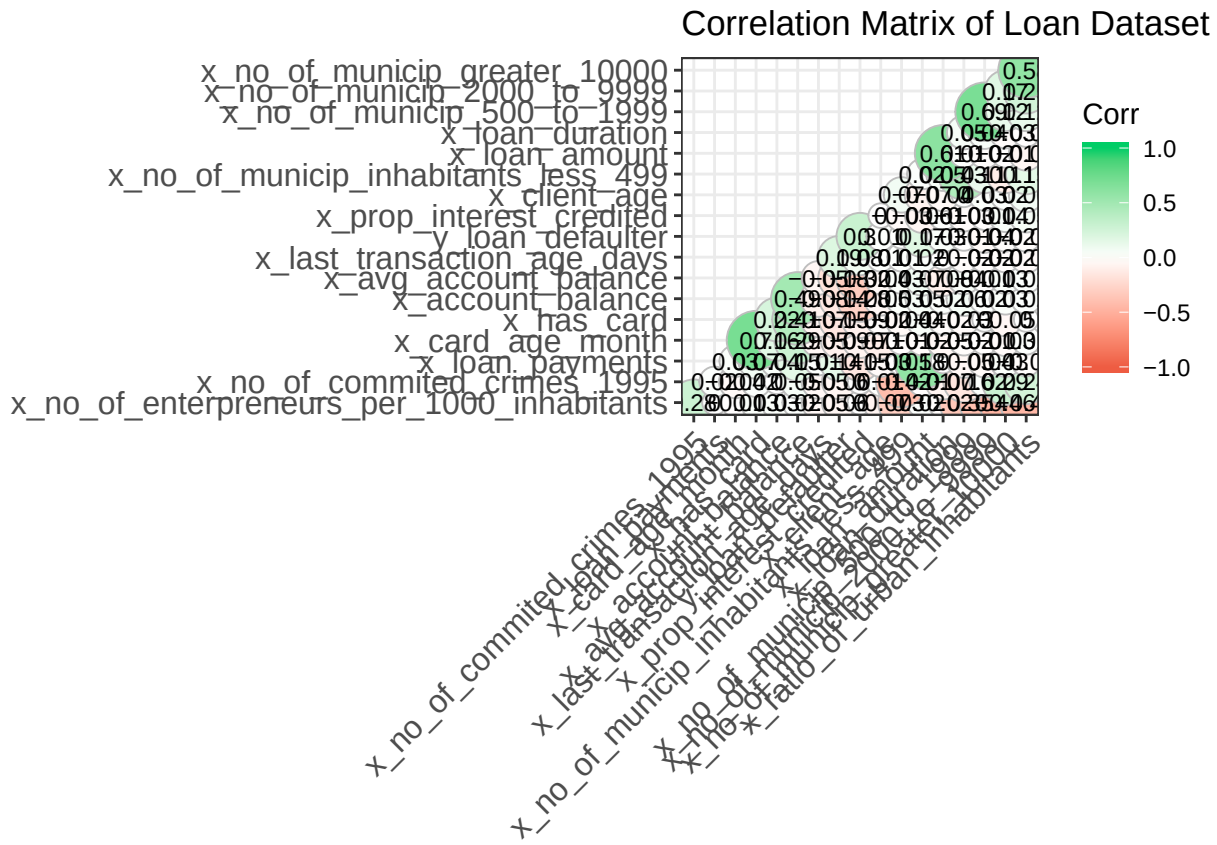
```
ggplot(VIF_Table_After, aes(x = fct_reorder(variable, VIF), y = log(VIF), label = round(VIF, 2))) +
  geom_point(stat='identity', fill="black", size=15)  +
  geom_segment(aes(y = 0,
                   yend = log(VIF),
                   xend = variable),
               color = "black") +
  geom_text(color="white", size=4) +
  geom_hline(aes(yintercept = log(5)), color = 'red', size = 2) +
  scale_y_continuous(labels = NULL, breaks = NULL) +
  coord_flip() +
  theme_economist() +
  theme(legend.position = 'none',
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank()) +
  labs(x = 'Variable',
       y = NULL,
       title = 'Variance Inflation Factor',
```

```
       subtitle="Checking for multicolinearity in X's variables.
       Variables with VIF more than 5 will
       be droped from the model")
```



Correlation Matrix Plot: no Variable with correlation more than 0.6.

```
cor_mtx <- cor(vars.quant)

ggcorrplot(cor_mtx, hc.order = TRUE,
           type = "lower",
           lab = TRUE,
           lab_size = 3,
           method="circle",
           colors = c("tomato2", "white", "springgreen3"),
           title="Correlation Matrix of Loan Dataset",
           ggtheme=theme_bw)
```



3.2.4 Sample split into Test and Training Data

The available data in Loan Dataset is split into Train and Testing data on the following proportion:

- **Train Dataset** (70% 478 obs);
- **Test Dataset** (30% 204 obs).

```
set.seed(12345)
index <- caret::createDataPartition(loan_reg_dataset$y_loan_defaulter,
                                    p= 0.7,list = FALSE)

data.train <- loan_reg_dataset[index, ]
data.test  <- loan_reg_dataset[-index,]


event_proportion <- bind_rows(prop.table(table(loan_reg_dataset$y_loan_defaulter)),
                              prop.table(table(data.train$y_loan_defaulter)),
                              prop.table(table(data.test$y_loan_defaulter)))

event_proportion$scope = ''
event_proportion$scope[1] = 'full dataset'
event_proportion$scope[2] = 'train dataset'
event_proportion$scope[3] = 'test dataset'

event_proportion <- select(event_proportion, scope, everything())
```

```
kable(event_proportion)
```

scope	0	1
full dataset	0.8885630	0.1114370
train dataset	0.8933054	0.1066946
test dataset	0.8774510	0.1225490

Both datasets keep the same proportion for the explained variable around 11%.

3.2.5 Fit the logistic model

With the final cleaned dataset, we got from above steps we fit our Logistic Regression Y_loan_defaulter on all x variables.

```
model_1 <- glm(data = data.train, formula = y_loan_defaulter ~ .,
                family= binomial(link='logit'))

# jut truncating variable names to 30 char to fit nicely in the screen.
names(model_1$coefficients) <- stringr::str_sub(names(model_1$coefficients), 1, 30)

summary(model_1)
```

```
##
## Call:
## glm(formula = y_loan_defaulter ~ ., family = binomial(link = "logit"),
##     data = data.train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.5308  -0.4369  -0.2692  -0.1609   2.9273
##
## Coefficients: (1 not defined because of singularities)
##                                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)                   -4.405e+00  2.351e+00  -1.874   0.0610 .
## x_loan_amount                  7.239e-06  2.978e-06   2.431   0.0151 *
## x_loan_duration               -2.860e-02  1.867e-02  -1.532   0.1255
## x_loan_payments                2.306e-03  1.839e-03   1.254   0.2099
## x_client_gendermale            2.084e-02  3.594e-01   0.058   0.9537
## x_client_age                   1.801e-02  1.414e-02   1.274   0.2028
## x_no_of_municip_inhabitants_le -1.823e-03  6.599e-03  -0.276   0.7824
## x_no_of_municip_500_to_1999    8.119e-04  1.957e-02   0.041   0.9669
## x_no_of_municip_2000_to_9999  -1.178e-02  6.436e-02  -0.183   0.8547
## x_no_of_municip_greater_10000 -3.642e-01  2.321e-01  -1.569   0.1167
## x_ratio_of_urban_inhabitants   9.512e-03  1.150e-02   0.827   0.4080
## x_no_of_enterpreneurs_per_1000 -1.078e-02  1.162e-02  -0.927   0.3538
## x_no_of_commited_crimes_1995  -1.362e-02  9.617e-03  -1.416   0.1566
## x_card_typegold                5.329e-01  1.315e+00   0.405   0.6852
## x_card_typejunior              3.728e-01  1.311e+00   0.284   0.7762
## x_card_typeno card             1.580e+00  8.938e-01   1.768   0.0770 .
## x_card_age_month               2.938e-02  3.072e-02   0.956   0.3390
```

```
## x_account_balance               2.386e-06  1.383e-06   1.725   0.0844 .
## x_avg_account_balance          -4.174e-06  2.969e-06  -1.406   0.1598
## x_last_transaction_age_days     2.641e-02  4.276e-02   0.618   0.5368
## x_prop_interest_credited        1.156e+01  2.241e+00   5.158  2.5e-07 ***
## x_has_card                             NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 324.61  on 477  degrees of freedom
## Residual deviance: 246.59  on 457  degrees of freedom
## AIC: 288.59
##
## Number of Fisher Scoring iterations: 6
```

Alternatively we fit a second model only with variables statistically significant p-value less than 10%.

```
model_2 <- glm(data = data.train, formula = y_loan_defaulter ~ x_loan_amount +
                x_loan_duration + x_has_card + x_prop_interest_credited,
                family= binomial(link='logit'))

# jut truncating variable names to 30 char to fit nicely in the screen.
names(model_2$coefficients) <- stringr::str_sub(names(model_2$coefficients), 1, 30)

summary(model_2)
```

```
##
## Call:
## glm(formula = y_loan_defaulter ~ x_loan_amount + x_loan_duration +
##     x_has_card + x_prop_interest_credited, family = binomial(link = "logit"),
##     data = data.train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.5790  -0.4375  -0.3290  -0.2165   2.6686
##
## Coefficients:
##                            Estimate Std. Error z value Pr(>|z|)
## (Intercept)              -3.632e+00  4.984e-01  -7.286 3.19e-13 ***
## x_loan_amount             9.023e-06  1.888e-06   4.780 1.75e-06 ***
## x_loan_duration          -3.864e-02  1.458e-02  -2.651  0.00803 **
## x_has_card               -1.064e+00  5.009e-01  -2.124  0.03364 *
## x_prop_interest_credited  9.715e+00  1.750e+00   5.551 2.85e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 324.61  on 477  degrees of freedom
## Residual deviance: 264.00  on 473  degrees of freedom
## AIC: 274
##
## Number of Fisher Scoring iterations: 6
```

In the next step we will compare how each model performed.

3.2.6 Evaluating the model performance

We started this step by making predictions using our model on the X's variables in our Train and Test datasets.

```
glm.prob.train.1 <- predict(model_1, type = "response")
glm.prob.test.1 <- predict(model_1, newdata = data.test, type= "response")

glm.prob.train.2 <- predict(model_2, type = "response")
glm.prob.test.2 <- predict(model_2, newdata = data.test, type= "response")
```

We then evaluate the metrics in the each model for Train and Test data:

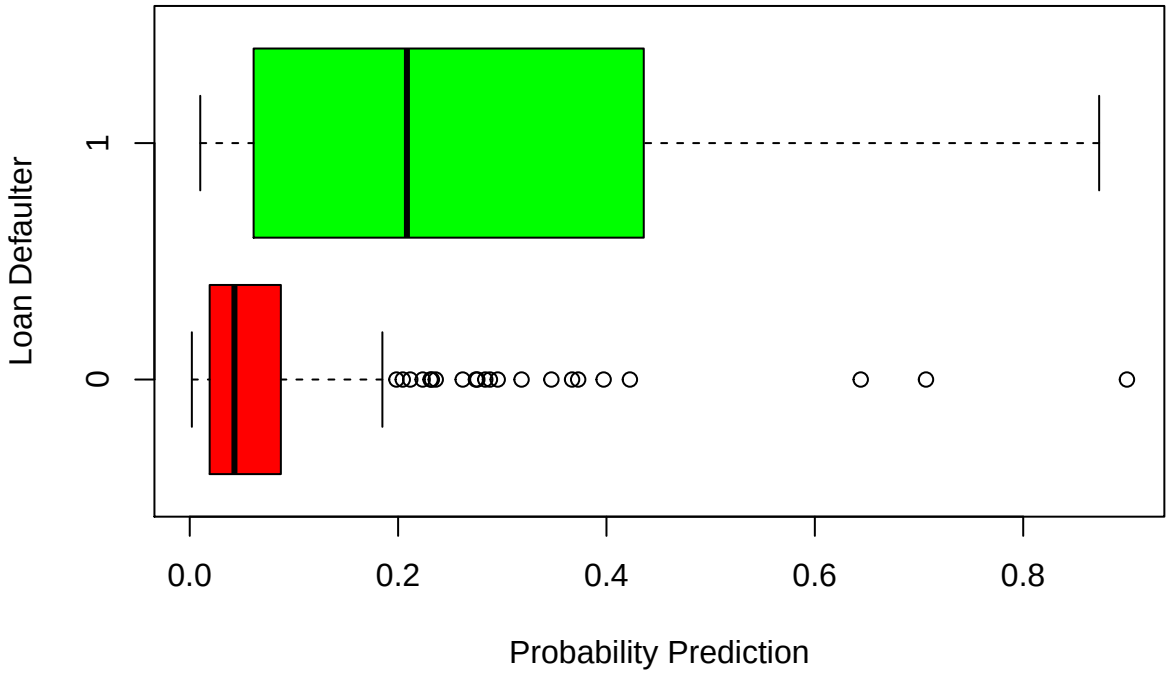
```
glm.train.1 <- HMeasure(data.train$y_loan_defaulter, glm.prob.train.1, threshold = 0.5)
glm.test.1  <- HMeasure(data.test$y_loan_defaulter, glm.prob.test.1, threshold = 0.5)

glm.train.2 <- HMeasure(data.train$y_loan_defaulter, glm.prob.train.2, threshold = 0.5)
glm.test.2 <- HMeasure(data.test$y_loan_defaulter, glm.prob.test.2, threshold = 0.5)

measures <- t(bind_rows(glm.train.1$metrics,
                        glm.test.1$metrics,
                        glm.train.2$metrics,
                        glm.test.2$metrics)) %>% as_tibble(., rownames = NA)

colnames(measures) <- c('model 1 - train','model 1 - test',
                        'model 2 - train','model 2 - test')

measures$metric = rownames(measures)

measures <- select(measures, metric, everything())

kable(measures, row.names = FALSE)
```

metric	model 1 - train	model 1 - test	model 2 - train	model 2 - test
H	0.3782094	0.3436917	0.3259694	0.4267060
Gini	0.6830601	0.5642458	0.6040777	0.6420112
AUC	0.8415301	0.7821229	0.8020388	0.8210056
AUCH	0.8567066	0.8179888	0.8227029	0.8521788
KS	0.5430959	0.4563128	0.5100335	0.5450279
MER	0.0962343	0.0980392	0.0941423	0.1078431
MWL	0.0870958	0.1169262	0.0933982	0.0978470
Spec.Sens95	0.5035129	0.3016760	0.3161593	0.3743017
Sens.Spec95	0.3921569	0.4000000	0.3333333	0.4000000
ER	0.1025105	0.1176471	0.0983264	0.1176471
Sens	0.1764706	0.1600000	0.1568627	0.1600000
Spec	0.9836066	0.9832402	0.9906323	0.9832402
Precision	0.5625000	0.5714286	0.6666667	0.5714286

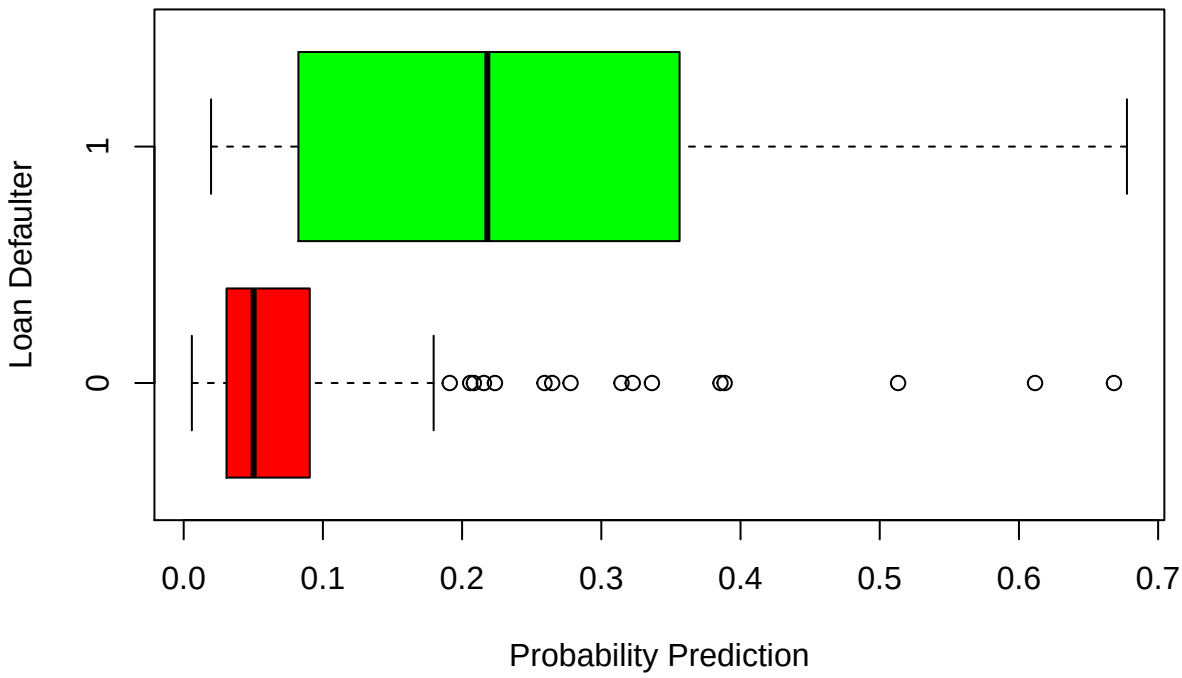
metric	model 1 - train	model 1 - test	model 2 - train	model 2 - test
Recall	0.1764706	0.1600000	0.1568627	0.1600000
TPR	0.1764706	0.1600000	0.1568627	0.1600000
FPR	0.0163934	0.0167598	0.0093677	0.0167598
F	0.2686567	0.2500000	0.2539683	0.2500000
Youden	0.1600771	0.1432402	0.1474951	0.1432402
TP	9.0000000	4.0000000	8.0000000	4.0000000
FP	7.0000000	3.0000000	4.0000000	3.0000000
TN	420.0000000	176.0000000	423.0000000	176.0000000
FN	42.0000000	21.0000000	43.0000000	21.0000000

Then, we look a boxplot chart to see how well our model split the observation into our explained variable:

-**Model 1:**

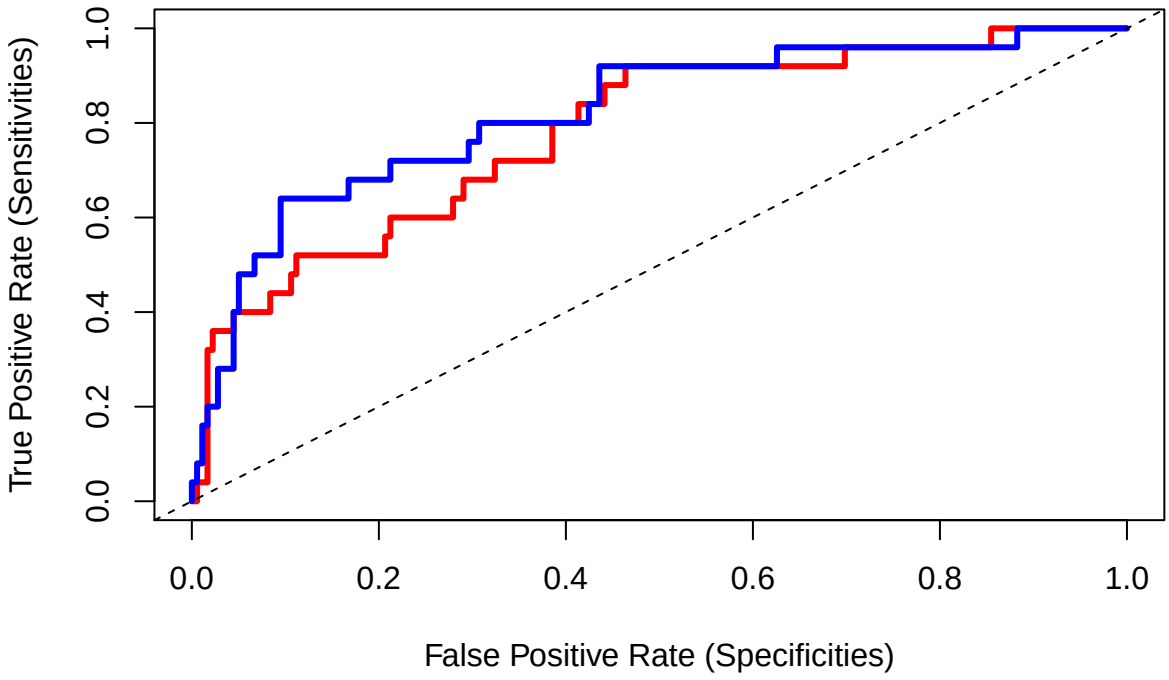


-**Model 2:**



Then we plot the ROC(Receiver Operator Characteristic Curve) of the model:

```
roc_1 <- roc(data.test$y_loan_defaulter, glm.prob.test.1)
roc_2 <- roc(data.test$y_loan_defaulter, glm.prob.test.2)

y1 <- roc_1$sensitivities
x1 <- 1-roc_1$specificities

y2 <- roc_2$sensitivities
x2 <- 1-roc_2$specificities

plot(x1, y1,  type="n",
     xlab = "False Positive Rate (Specificities)",
     ylab= "True Positive Rate (Sensitivities)")

lines(x1, y1,lwd=3,lty=1, col="red")
lines(x2, y2,lwd=3,lty=1, col="blue")

abline(0,1, lty=2)
```



By the ROC curve of each model we see that model 2 is a better model. It only consider the statistic significant variables.

Finally we look more closely to the model 2 accuracy:

-**Model 2:**

To perform this task, we start by defining a threshold to assign the observation to each class, and them calculate the General Accuracy and the True Positive Rate. We start our analysis with a threshold of 0.5

```
threshold <- 0.5

fitted.results.2 <- ifelse(glm.prob.test.2 > threshold ,1 ,0)

misClasificError <- mean(fitted.results.2 != data.test$y_loan_defaulter)

misClassCount <- misclassCounts(fitted.results.2, data.test$y_loan_defaulter)

paste('Model General Accuracy of: ', round((1 - misClassCount$metrics['ER']) * 100, 2), '%', sep = '')
```

```
## [1] "Model General Accuracy of: 88.24%"
```

```
paste('True Positive Rate of    : ', round(misClassCount$metrics['TPR'] * 100, 2), '%', sep = '')
```

```
## [1] "True Positive Rate of    : 16%"
```

```
kable(misClassCount$conf.matrix)
```

	pred.1	pred.0
actual.1	4	21
actual.0	3	176

At a first glance we may see this model as an excellent predictor of default as it can predict the correct class 88.24% of the cases.

But looking more closely at the confusion matrix we can see that this model can predict only 16% of the true defaulters.

One way to improve the performance of the model is to decrease the threshold to 0.2.

See results below:

```
threshold <- 0.2

fitted.results.2 <- ifelse(glm.prob.test.2 > threshold ,1 ,0)

misClasificError <- mean(fitted.results.2 != data.test$y_loan_defaulter)

misClassCount <- misclassCounts(fitted.results.2, data.test$y_loan_defaulter)

paste('Model General Accuracy of: ', round((1 - misClassCount$metrics['ER']) * 100, 2), '%', sep = '')
```

```
## [1] "Model General Accuracy of: 86.76%"
```

```
paste('True Positive Rate of    : ', round(misClassCount$metrics['TPR'] * 100, 2), '%', sep = '')
```

```
## [1] "True Positive Rate of    : 52%"
```

```
kable(misClassCount$conf.matrix)
```

	pred.1	pred.0
actual.1	13	12
actual.0	15	164

Now we see that the True Positive accuracy increased to 52% and the general accuracy drop just a bit to 86.76%.

We may be tempted to reduce even more our threshold.

Let's see what happen if we use 0.1 as threshold.

```
## [1] "Model General Accuracy of: 76.47%"
```

```
## [1] "True Positive Rate of    : 72%"
```

	pred.1	pred.0
actual.1	18	7
actual.0	41	138

Now our TPR jumped to 72% and our general accuracy drop to 76.47%.

The final threshold will depend on the real word use case, by how much we value each class.

In this sample we see that we can improve our True Positive prediction rate at expense of classifying an increasing number as defaulter.

This balance must be decided by the managers and the bank should keep seeking and experimenting with more variables to find a better model.

---

4 Conclusion

The final model can accurately segregate the loan contracts likely to default with test set accuracy of 76.47% using 0.1 as the threshold to assign the observation to defaulter class.

Model was also able to predict 72% of real defaulters in our test dataset.

In practice, the probabilities returned by the logistic regression model can be used to decide on grant or not the loan and help on correctly price the interest rate based on the likelihood of default.

Model 2 shows that below variables are associated with the probability of default in a loan contract:

- **x_loan_amount**;
- **x_loan_duration**;
- **x_has_card**;
- **x_prop_interest_credited**.

The threshold definition depends on the tradeoff we accept between correctly identifying the true defaulters and incorrectly set a non-defaulter as a defaulter.

This balance must be decided by the managers and the bank should keep seeking and experimenting with more variables to find a better model.

This exercise was in no way a definitive answer to Berka Bank on how to perform their client credit analysis, but it served very well to exercise the new techniques learned in this class.