LendingClub Loans Pricing

HarvardX - PH125.9x Data Science Capstone

Emmanuel Rialland - https://github.com/Emmanuel_R8

December 04, 2019

Contents

List of Tables

List of Figures

Introduction

Lending Club (LC) is an American company listed on the New York stock exchange that provides a platform for peer-to-peer lending. Unlike banks, it does not take deposits and invest them. It is purely a matching system. Each loan is split into $25 that multiple investors can invest in. LC is remunerated by fees received from both sides. LC states that they have intermediated more than $50bln since they started operations. Further description of the company is easily available online numerous sources.

In order for investors to make the best investment decisions, LC make a historical dataset publicly available to all registered investors. This dataset is the subject of this report. It was downloaded from the Kaggle data science website[1].

The size of the dataset is rich enough that it could be used to answer many different questions. We decided for a focused approach. Following Chapter 5 of (Peng, 2012), we will first formulate the question we want to answer to guide our analysis.

The business model of LC is to match borrowers and investors. Naturally, more people want to receive money than part with it. An important limiting factor to LC's growth is the ability to attract investors, build a trusting relationship where, as a minimum first step, investors trust LC to provide accurate, transparent and reliable information of the borrowers. For this purpose, LC decided not only to provide extensive information about potential borrowers' profile, but also historical information about past borrowers' performance. This is, as we understand, one of the key purposes of this dataset. We decided to use the dataset for this very purpose. Essentially, the questions are: **given a borrower profile, is his/her rating appropriate in terms of risk of default? And if a default occurs, what is the expected recovery? The summary question is: given a borrower profile, is the risk/reward balance appropriate to commit funds?** In answering this question, we understand that LC allows investment of very granular amounts. Therefore, even an individual investor can diversify his/her loan and risk portfolio. It is not necessary to 'gamble' funds on a single borrower. This is exactly what institutional investors achieve through syndication (although on a very different scale, typically $10-25mln for a medium-size bank).

For this exercise, we made two simplifying (hopefully not simplistic) assumptions:

- In determining the risk/return balance, we have not accounted for LC's cost of intermediation. By ignoring fees paid by both sides, we obviously overestimate the returns to the investors. But in first approximation, **we will assume that the risk/reward balance, from the investors' point of view, across ratings is independent from fees.** This is a simplification. Real-world fees are higher the lower the investment grade and push the investors to receive, and the borrowers to pay, higher interest margin.
- All-in interest rates paid by borrowers are fixed. This is highly desirable for borrowers to be able to manage their cashflow. However, an investor should always consider an investment

[1] https://www.kaggle.com/wendykan/lending-club-loan-data/data

return as a margin above a risk-free return. Banks would look at LIBOR; bond investors (e.g. life insurers) would look at government bonds. Those risk-free rates can change very quickly, whereas we understand that LC sets those rates on a less frequent basis. In other word, the risk premium will vary rapidly. **We assume that individual investors are 'in-elastic' to change in implied risk premia.** But we recognise this as a limitation of our work.

This report is organised as follows:

- [XXXX]

Chapter 1

Internal Rate of Return, Credit Margins and Net Present Values

In this section, we describe the response variables that we will generate for each loan and that will be used in the rest of this report.

As indicated in the introduction, our purpose is to test a model that predicts the financial risk of a loan.

1.1 Important warning

The calculations presented here are simplistic, although they bear some resemblance to what financial institutions (*FIs*) do. The literature on credit assessment and pricing is very rich and very complex. Finding the optimal capital allocation to particular risks while at the same time satisfying internal risk policies and regulatory requirements is a problem that financial institutions have yet to solve in full. Investing in a loan is not only a matter of assessing the risk of a particular borrower, but also assessing systemic risks (which exist across all borrowers), risks associated with funding the loan (interest, currency and liquidity markets), each requiring a risk assessment and pricing.

In other words, nobody would, let alone should, make any investment decision based on the calculations below.

1.2 Background

This subsection can be skipped by anybody with basic financial knowledge.

A bird in hand or two in the bush; a penny today or a pound tomorrow. What is the price of delaying obtaining and owning something? This is what pricing a loan is about. A lender could keep his/her cash in hand, or lend it and have it in hand later. He/she would accept this in exchange for receiving a bit more: this is the rate of interest. A lender wants to be compensated for delaying the possibility of using the cash, but also for taking the risk of not receiving it, partially or in full, when repayment is due.

There are borrowers that one can see as (almost) completely safe or risk-free such as central banks or governments of strong economies. A lender always has the possibility to lend to them instead of more risky borrowers. Therefore, a lender would require a higher interest rate than risk-free. The additional interest that a lender requires is commensurate with the risk of the borrower not repaying (called *credit worthiness*) and is called the *credit margin*.

For each individual borrower, an FI would assess information provided by the borrower and massive amounts of historical data to answer the question: considering historical borrowers with a profile similar to the applicant's, what is the probability of not getting principal and interest back (*Probability of Default* or *PD*)? And, in case the borrower stops paying and using additional courses of action (such as seizing and selling assets), what is the total loss that could be expected on average (*Loss given Default* or *LGD*)?

Making that assessment, the FI would require an interest rate which would roughly be the sum of:

- the risk-free rate;
- a margin to cover the average loss of similar individual borrowers[1];
- a margin to cover all the operational costs of running their operations; and,
- a margin to remunerate the capital allocated by the FI (banking regulations require all banks to allocate an amount of capital against any risk taken; those are stipulated in a number of complex rules).

Said crudely, this total is the amount for the FI to get out of bed and look at a loan. Although this sounds like an exact science (for some definition of the word), it is not. At the end of the day, the FI will also have to contend with the competition from other FIs or non banking lenders, market liquidity (if there is a lot of money available to be lent, it brings prices down) and, critically, whether the borrower would at all be interested in accepting that cost.

Note that the dataset is distorted by this additional survival effect: the application information of many loans does not appear merely because the rate of interest was considered too high (this is not dissimilar to *survival effects* where some data did not survive through the history of a dataset[2]).

1.3 Internal Rate of Return

For the purpose of this report, we will simplify things enormously: we will only consider the first two components of the interest rate. The risk-free rate and the credit margin that would cover the cost of default/losses of individual borrowers.

The modeling exercise will focus on trying to approximate *on average* the credit margin given the information provided by the borrower.

With respect to a given loan and its cash flow, two calculations are important here: the Net Present Value (NPV) and the Internal Rate of Return (IRR). If we remember that an FI is indifferent to holding a principal P today or receiving it with an annual interest a year later (i.e. $P \times (1+r)$ where r is the annual rate of interest), we can say that any amount CF_1 received in a year is equivalent to $CF_0 = \frac{CF_1}{1+r}$ today. More generally, a steam of future cash receipts is worth:

$$NPV(r) = \sum_{Yeari=1}^{Year=n} \frac{CF_i}{(1+r)^i}$$

[1] It is important to realise that the average margin only brings the borrower back to having earned the risk-free rate average: the additional income from the credit margin will be spent to cover average losses. In addition, we present the credit margin as income against borrower-specific losses. It does not address a lot of other risks such as correlation risks: a borrower might default because the economy as a whole gets worse, in which case many borrowers will default. This is a cyclical *systemic* risk similar to the 2007 US real estate crisis.

[2] A well-known example is historical stock prices which disappear when companies are de-listed or go bankrupt.

The amount $NPV(r)$ is called the *Net Present Value* of the cash flow discounted at the rate r. Given that the LendingClub repayments are monthly, the formula becomes:

$$NPV(r) = \sum_{Monthi=1}^{Month=12\times n} \frac{CF_i}{(1+\frac{r}{12})^i}$$

If we now have a day 1 cash flow CF_0, we can calculate:

$$CF_0 - \sum_{Yeari=1}^{Year=n} \frac{CF_i}{(1+r)^i}$$

However, for any given CF_0, there is no reason that it would equal the NPV of the future cash flow (i.e no reason why the difference would be equal to zero). But this is an equation depending on r. If we can find a value of r that zeroes this formula, it is called the internal rate of return of the cash flow:

$$CF_0 - \sum_{Yeari=1}^{Year=n} \frac{CF_i}{(1+IRR(CF))^i} = 0$$

or for monthly cash flow:

$$CF_0 - \sum_{Monthi=1}^{Month=12\times n} \frac{CF_i}{(1+\frac{IRR(r)}{12})^i} = 0$$

1.4 Dataset calculation

For each loan. we calculated the IRR, credit margin and NPV. The calculations were performed in Julia due to R's slow performance on such a large dataset. For this reason, the resulting datasets are available on the GitHub repository as gzipped CSV files.

1.4.1 IRR

We used the dataset to calculate the IRR of each loan. We used the following information for the dataset: `funded_amnt`(loan amount funded), `int_rate` (all-in interest rate), `term` (tenor of the loan in months), `total_pymnt` (total cumulative amount received from the borrower), `total_rec_prncp` (amount repaid allocated to principal repayment), `total_rec_int` (amount repaid allocated to interest payment), `recoveries` (any amount recovered later from the borrower) and `total_rec_late_fee` (any late payments fees paid by the borrower).

From that information, we recreated a cash flow for each loan. The R code is presented in appendix. Unfortunately, this code takes close to a full day to run on the entire dataset of completed loans (i.e. excluding all ongoing loans). This is just impractical for anybody to run to check this report and the resulting IRR results dataset is included in the Github repository. To make things practical, the dataset was actually created using code in Julia[3]. It is a direct translation of the R code, with a similar syntax (therefore very easy to follow). The Julia code runs about 500 times quicker (this is not a typo), or about 3 minutes. We appreciate that this is the departure from the assignment description.

Similarly, we calculate the credit margin required by each loan noting that:

[3] https://julialang.org/

$$\text{Risk-free} + \text{Credit Margin} = IRR(loan)$$

1.4.2 Credit Margins

As noted in the previous section, risk-free rates change over time. When solving for the credit margin, we use the relevant risk-free rate.

Again, this was coded in Julia. The Julia code here takes about 1h20min to run. On the assumptions that the equivalent R code would take therefore almost 30 days to run through the full dataset, we did not write any R code for this calculation. The code is in appendix, and we believe easy to follow.

1.4.3 NPV

We also calculated the NPV of each loan. Again, this was coded in Julia. The code is in appendix.

We calculate both the NPV as an absolute dollar amount and as a portion of the original

Chapter 2

Dataset

The data is sourced as a *SQLite* database that downloaded from (Kan, 2019) and imported as a `tibble dataframe` with the `RSQLite` package. The variables were reformatted according to their respective types.

[We also sourced US zip and FIPS codes, and macroeconomical data for possible geographical statistics. The source code for the data import and reformatting is given in appendix.]

2.1 Preamble

The LendingClub dataset, although rich, is difficult to interpret. The only explanation of what the variables mean comes from a spreadsheet attached to the dataset. The explanations are not precise and/or subject to conflicting interpretation. Despite serching the LendingClub website, no further original information was found. We collected a number of reasonable assumptions in Appendix (see (List of Assumptions)[list-assumptions]).

The dataset has been used a number of times in the past by various people. One paper (Kim and Cho, 2019) mentions they used a dataset that included 110 variables, which is less than ours with 145 variables. The dataset has changed over time in ways we do not know. For example, have loans been excluded because the full 145 veriables were not available?

2.2 General presentation

The original dataset is large: it includes 2260668 loan samples, each containing 145 variables (after the identification variables filled with null values). The loans were issued from 2007-06-01 to 2018-12-01.

2.2.1 Business volume

The dataset represents a total of ca.$34bln in loan principals, which is a substantial share of the total amount stated to have been intermediated to date by LC (publicly reported to be $50bln+). About 55%/60% of the portfolio is fully repaid. See Table **??**.

Figure **??** plots the number, volume (cumulative principal amount) and average principal per loan. It shows that the business grew exponentially (in the common sense of the word) from inception until 2016. At this point, according to Wikipedia [1]:

[1] source: https://en.wikipedia.org/wiki/LendingClub - Retrieval date 15 September 2019

Table 2.1: Number of loans per status

Loan status	Count	Proportion (%)
Charged Off	261655	11.574
Current	919695	40.682
Default	31	0.001
Does not meet the credit policy. Status:Charged Off	761	0.034
Does not meet the credit policy. Status:Fully Paid	1988	0.088
Fully Paid	1041952	46.090
In Grace Period	8952	0.396
Late (16-30 days)	3737	0.165
Late (31-120 days)	21897	0.969

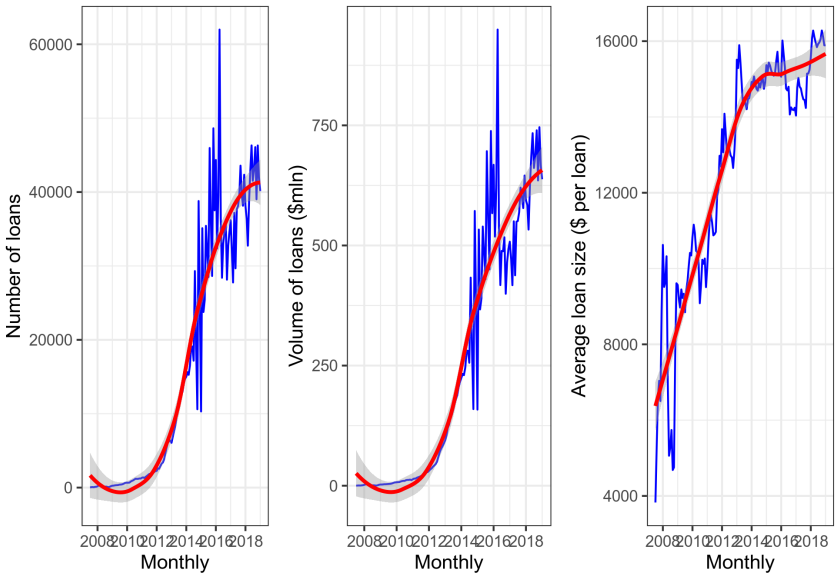


Figure 2.1: Business volume written per month

" *Like other peer-to-peer lenders including Prosper, Sofi and Khutzpa.com, LendingClub experienced increasing difficulty attracting investors during early 2016. This led the firm to increase the interest rate it charges borrowers on three occasions during the first months of the year. The increase in interest rates and concerns over the impact of the slowing United States economy caused a large drop in LendingClub's share price.*"

The number and volume of loans plotted have been aggregated by month. The growth is very smooth in the early years, and suddenly very volatile. As far as the first part of the dataset is concerned, a starting business could expect to be volatile and could witness a yearly cycle (expected from economic consumption figures) superimposed on the growth trend. This is not the case.

An interesting metric is that the average principal of loans has increased (see RHS Figure **??**, on a sample of 100,000 loans). Partly, the increase in the early years could be interpreted success in improving marketing, distribution capabilities and confidence building. This metric plateau-ed in 2016 and decreased afterwards, but to a much lesser extent than the gross volume metrics. However, it is more volatile than the two previous metrics in the early years.

By the end of the dataset, those metrics have essentially recovered to their 2016 level.

2.2.2 Loan lifecyle and status

In the dataset, less loans are still outstanding than matured or "*charged off*" (term that LC use to mean partially or fully written off, i.e. there are no possibilty for LC and/or the investors to receive further payments). The share of outstanding loans is:

```
## Share of current loans =  42.214 %
```

The dataset describes the life cycle of a loan. In the typical (ideal) case, we understand it to be:

Loan is approved → Full amount funded by investors → Loan marked as Current → Fully Paid

In the worst case, it is:

Loan is approved → Full amount funded by investors → Loan marked as Current →

→ Grace period (missed payments under 2 weeks) → Late 15 to 31 days →

→ Late 31 to 120 days → Default → Charged Off

Note that *Default* precedes and is distinct from *Charged Off* [2]. A couple of things could happen to a loan in default:

- LC and the borrower restructure the loan with a new repayment schedule, where the borrower may repay a lesser amount over a longer period; or,
- the claim could be sold to a debt recovery company that would buy the claim from LC/investors. This would be the final payment (if any) received by LC and the investors.

[2] See LendingClub FAQ at and help page

The dataset also describes situations where a borrower negotiated a restructuring of the repayment schedule in case of unexpected hardship (e.g. disaster, sudden unemployment).

Note that this progression of distinguishing default (event in time) from actual financial loss mirrors what banks and rating agencies do. The former is called the *Probability of Default* (PD), the latter *Loss Given Default* (LGD). Ratings change over time (in a process resembling Markov Chains). LGD show some correlations with ratings. The dataset, although detailed, does not include the full life of each loan to conduct this sort of analysis (change of loan quality over time). This is an important reason why we decided to focus on the loan approval and expected return.

> CHANGE: Debt-Settlement companies [3] that explains that debt settlement companies can step in, buy the debt and pay to LC.
>
> CHANGE: What is the difference between a loan that is in "default" and a loan that has been "charged off"? [4]

2.2.3 Loan application

Before a loan is approved, the borrower undergoes a review process that assess his/her capacity to repay. This includes:

- employment situation and income, as well whether this income and possibly its source has been independently verified;
- whether the application is made jointly (likely with a partner or a spouse, but there are no details);
- housing situation (owner, owner with current mortgage, rental) and in which county he/she lives (that piece of information is partially anonymised by removing the last 2 digits of the borrower's zipcode);
- the amount sought, its tenor and the purpose of the loan; and,
- what seems to be previous credit history (number of previous deliquencies). The dataset is very confusing in that regard: it is clear that such information relates to before the loan is approved in the case of the joint applicant. In the case of the principal borrower however, the variable descriptions could be read as pre-approval information, or information gathered during the life of the loan. We have assumed that the information related to the principal borrower is also pre-approval. We also used *Sales Supplements* from the LC website[5] that describe some of the information provided to investors. LendingClub also provides a summary description of its approval process in its regulatory filings with the Securities Exchange Commission (California, 2019).

2.3 Rates

2.3.1 IRR and required credit margins

Figure **??** shows the evolution of credit margins over time grouped by ratings. The plots are made with a random sample of 300,000 loans.

We notice long periods where certain margins remain very stable which indicate that *both* the initial pricing was constant *and* that the proportion of default remains very low.

The graphs show considerations that are relevant to the modeling":

[3] https://help.lendingclub.com/hc/en-us/articles/115011819087-Debt-settlement-companies
[4] https://help.lendingclub.com/hc/en-us/articles/216127747
[5] See https://www.lendingclub.com/legal/prospectus

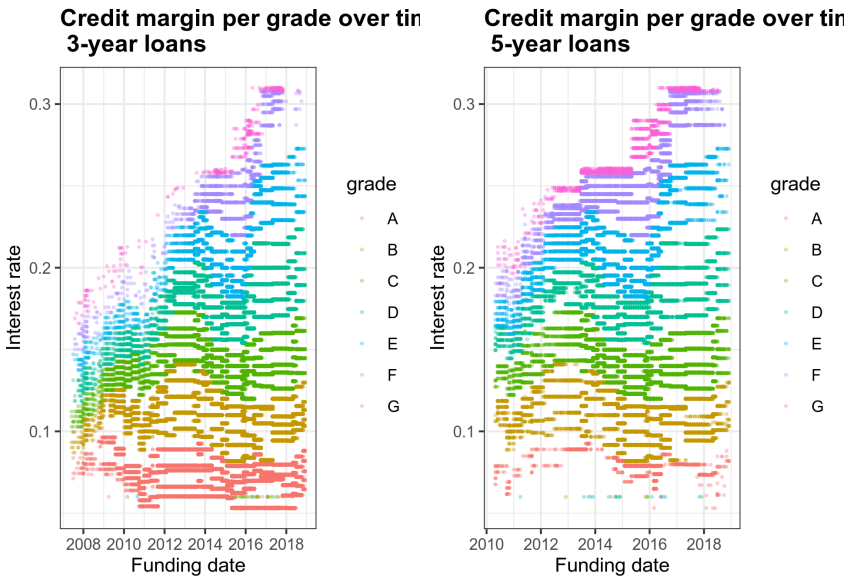


Figure 2.2: Credit margins per grade over time

- The margins clearly change over time. Predictions will require to account for time [probably in a non-linear fashion].
- For a given rating, it widens and narrows over time. The changes happen in multiples that depends on the ratings:
  - For high quality / low margin loans: the changes are multiples of the margin, for example going from roughly 3% to 6/7%.
  - Although the range of change is wide, those changes do not happen very often, especially in the later years.
  - By comparison, for low quality / high margin loans, the range of change is smaller, but more frequent and volatile.
- In other words, the relation between loan quality (its rating) and its pricing (the credit margin) will significantly non-linear.

2.3.2 Dataset

Because we are interested decisions made prior to invest, we will limit the predictors to be used to those that are realistically available prior to funding. We eliminated

2.3.3 Interest rates

Based on this information, the loan is approved or not. Approval includes the final amount (which could be lower than the amount requested), tenor (3 or 5 years) and a rating similar to those given to corporate borrowers. Unlike corporate borrowers however, the rating mechanically determines the rate of interest according to a grid known to the borrower in advance[6]. The rates have changed over time. Those changes where not as frequent as market conditions (e.g. changes in Federal Reserve Bank's rates)[7].

Figure **??** [8] shows the predetermined interest rate depending on the initial rating as of July 2019.

[6] https://www.lendingclub.com/investing/investor-education/interest-rates-and-fees

[7] Corporate borrowers would negociate interest margins on a case-by-case basis despite similar risk profiles.

[8] source: https://www.lendingclub.com/investing/investor-education/interest-rates-and-fees

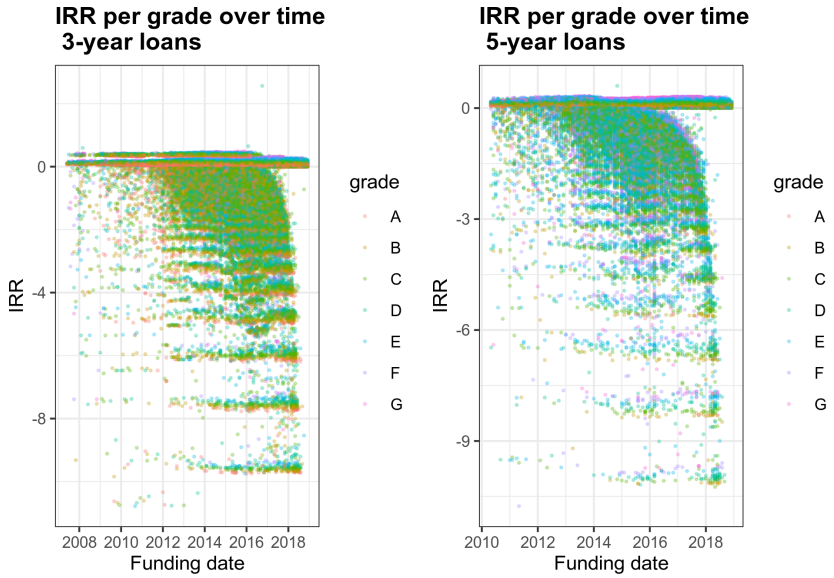


Figure 2.3: Credit margins per grade over time

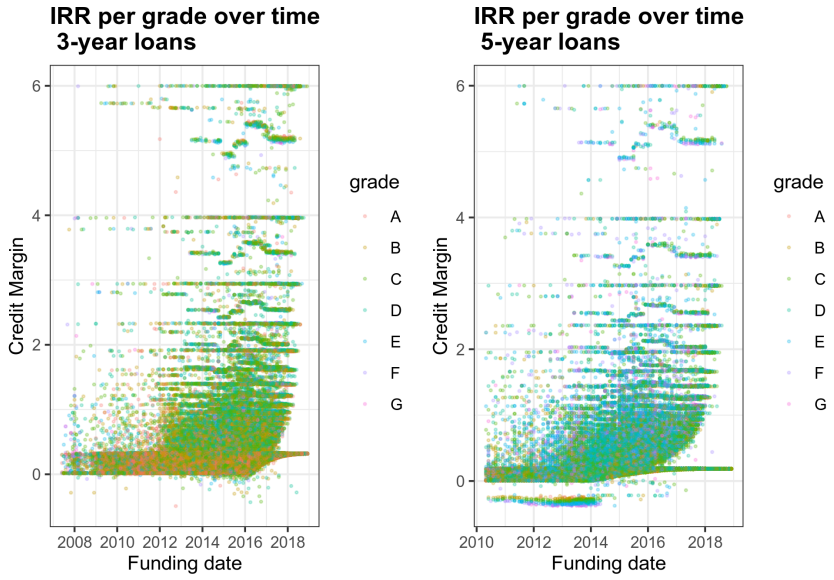


Figure 2.4: Credit margins per grade over time

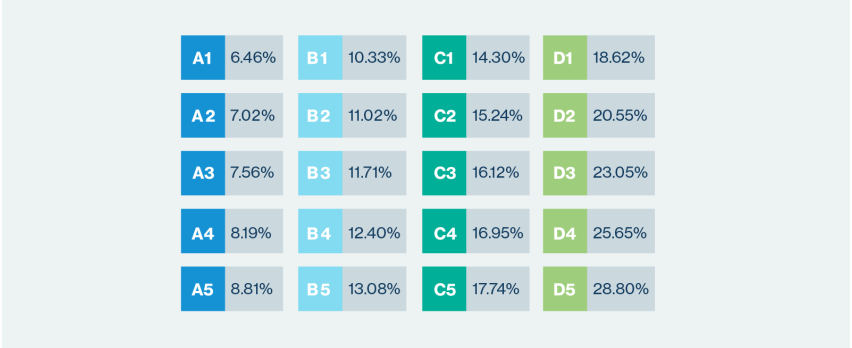


Figure 2.5: Interest rates given rating

At the date of this report, the ratings range from `A` (the best) down to `D`, each split in 5 sub-ratings. However, LC previously also intermediated loans rated F or G (until 6 November 2017) and E (until 30 June 2019) [9]. This explains that such ratings are in the dataset. We will assume that the ratings in the dataset are the rating at the time of approval and that, even if loans are re-rated by LC, the dataset does not reflect it.

Figures **??** shows the change in interest rate over time for different ratings and separated for each tenor. (Each figure is on a sample of 100,000 loans.) For each rating, we can see several parallel lines which correspond to the 5 sub-rating of each rating. We note that the range of interest rates has substantial widened over time. That is, the risk premium necessary to attract potential investors has had to substantially increase. In the most recent years, the highest rates exceed 30% which is higher than many credit cards.3-year loans are naturally considered safer (more A-rated, less G-rated). Identical ratings attract identical rates of interest.

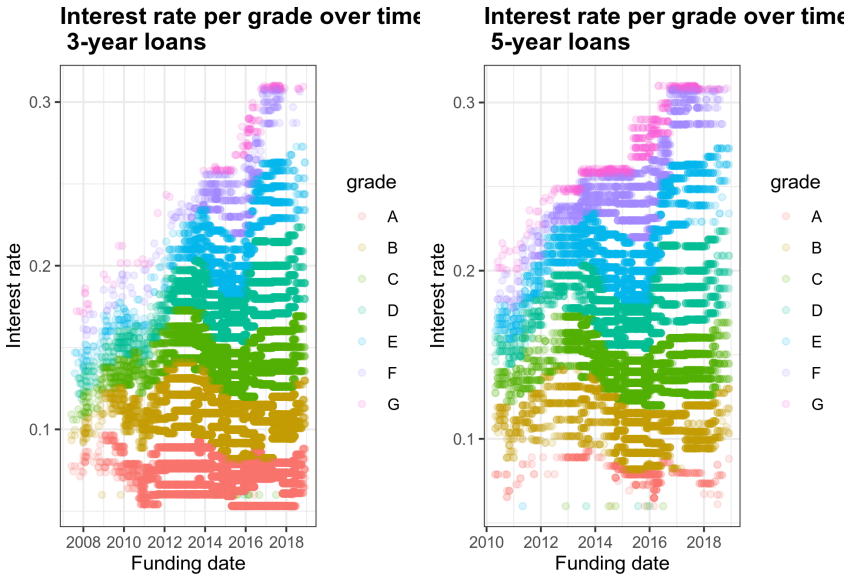


Figure 2.6: Interest rate per grade over time

By comparison, we plot the 3-year (in red) and 5-year (in blue) bank swap rates in Figure **??**. We see that the swap curve has flattened in recent times (3-year and 5-y rates are almost identical). We also can see that in broad terms the interest rates charged reflect those underlying swap rates. It is therefore most relevant to examine the credit margins added to the swap rates.

Figures **??** shows the change in credit margin over time for different ratings and separated for each tenor. (Each figure is on a sample of 100,000 loans.) As above, for each rating, we can see several parallel lines which correspond to the 5 sub-rating of each rating. We note that the range of credit margins has widened over time but less than the interest rates. Identical ratings attract identical credit margins.

2.3.4 Purpose

When applying, a potential borrower must state the purpose of the loan. As shown in table **??**, the main purpose is the consolidation of existing debts.

[9] See https://www.lendingclub.com/info/demand-and-credit-profile.action

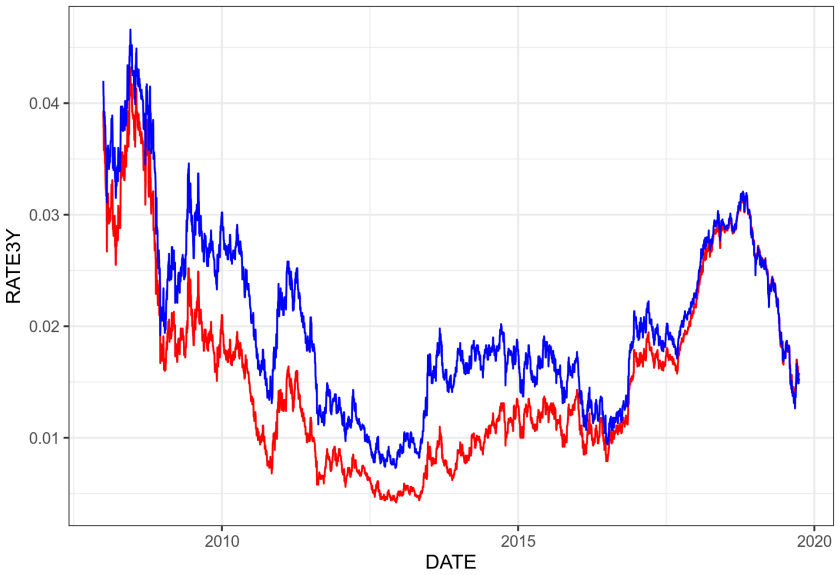


Figure 2.7: Historical Swap Rates

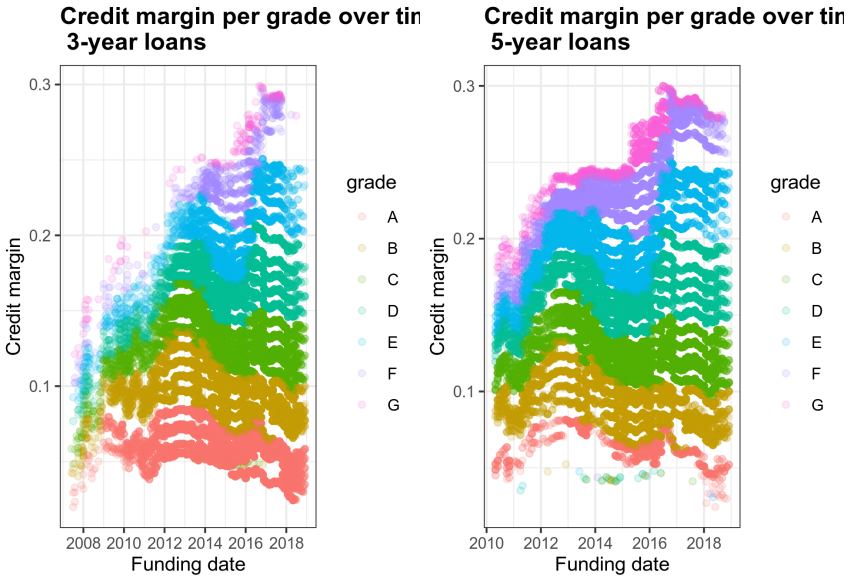


Figure 2.8: Credit margins per grade over time

purpose	Number
debt_consolidation	758691
credit_card	286044
home_improvement	84709
other	75358
major_purchase	28451
small_business	15171
medical	15081
car	14184
moving	9218
vacation	8751
house	7011
wedding	2350
renewable_energy	914
educational	423

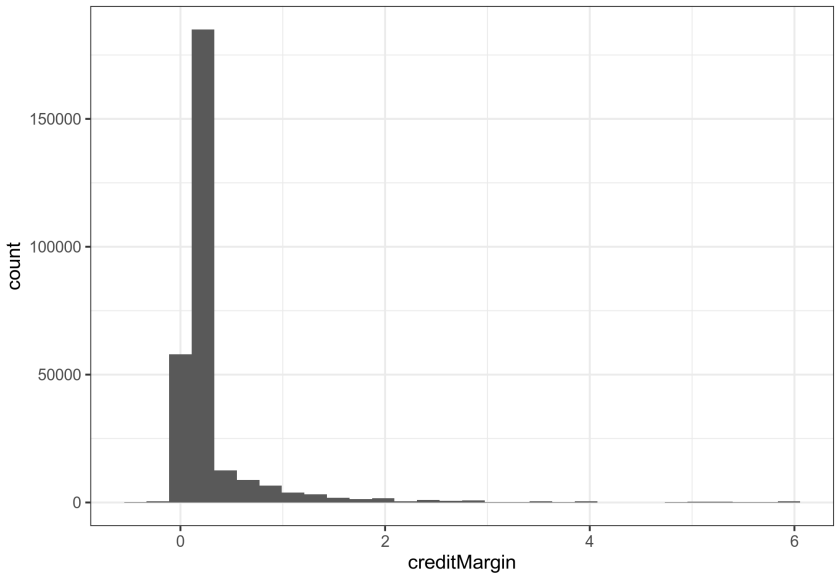


Figure 2.9: Histograms of credit margins per purpose

[TODO: DTI, amount... by grade]

2.3.5 Payments

The loans are approved for only two tenors, 3 and 5 years, with monthly repayments. Installments are calculated easily with the standard formula:

$$Installment = Principal \times rate \times \frac{1}{1 - \frac{1}{(1+rate)^N}}$$

Where $Principal$ is the amount borrowed, $rate = \frac{\text{Quoted Interest Rate}}{12}$ is the monthly interest rate, and N is the number of installments (36 or 60 monthly payments). The following piece of code shows that the average error between this formula and the dataset value is about 2 cents. We therefore precisely understand this variable.

```
local({
  installmentError <- loans %>%
    mutate(
      PMT = round(funded_amnt * int_rate / 12 / (1 - 1 / (1 + int_rate / 12) ^
```

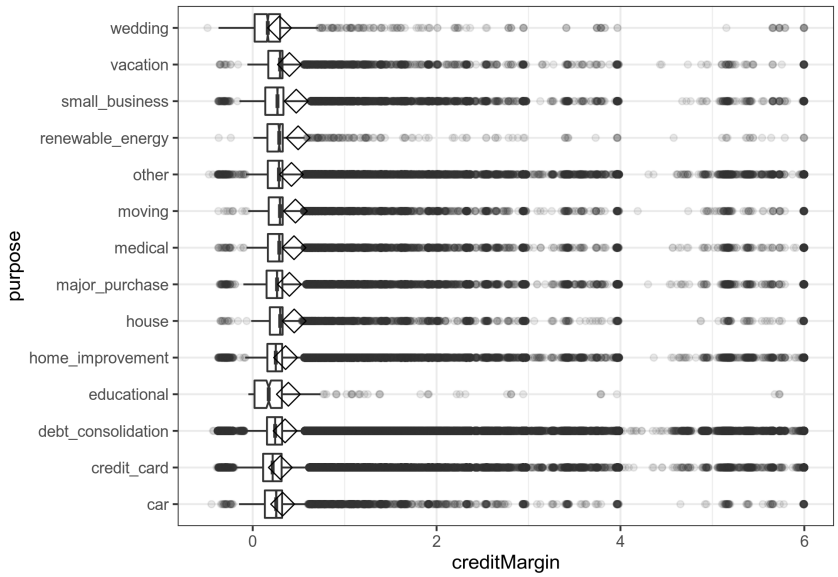


Figure 2.10: Boxplots of credit margins per purpose

```
                                                    term), 2),
      PMT_delta = abs(installment - PMT)
    ) %>%
    select(PMT_delta)

  round(mean(100 * installmentError$PMT_delta), digits = 2)
})
```

2.4 Net present value

The behaviour of the NPV of loan losses is important.

2.4.1 Average NPV and credit margin by subgrade

Figure **??** shows that as ratings worsen, the average NPV[10] expressed as a portion of the funded amount decreases. For the best quality loans, we see that the NPV exceeds 1.00 = 100%: at a risk-free rate, investors receive more than what is necessary to compensate for credit loss and can use the excess to cover additional costs mentioned in the Preamble. As ratings worsen, the NPV drops down to about 50%.

If loans were adequately priced, the excess returns (thanks to higher interest) should on average offset credit losses, that is an NPV average should be at least 100%. This seems to be the case down to ratings of about `D4`. Further down, credit losses become too frequent and/or too substantial to be covered on average. We posit that this justified rejecting loans applications rated `E1` and below.

[10]The averages are *not* weighted by loan amount since an investor can invest in $25 parcels. Weighting would have been appropriate if investors were instead forced to invest in the whole amount.

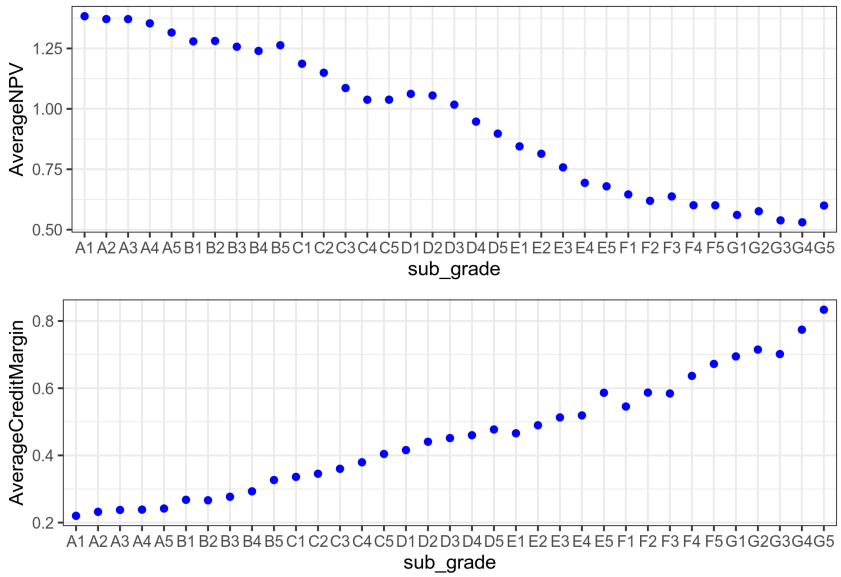


Figure 2.11: Average NPV et credit margin (%) depending on sub-rating

2.4.2 Principal losses

2.4.3 Distribution of principal losses by rating

Figure **??** shows that for a given grade, the losses are very widely spread. The loans are group by ratings and loans that have been fully repaid are removed.

Setting aside the loans rated "A" or "B", the distributions seem log-normal. Unsurprisingly, the worse the rating the larger the principal loss.

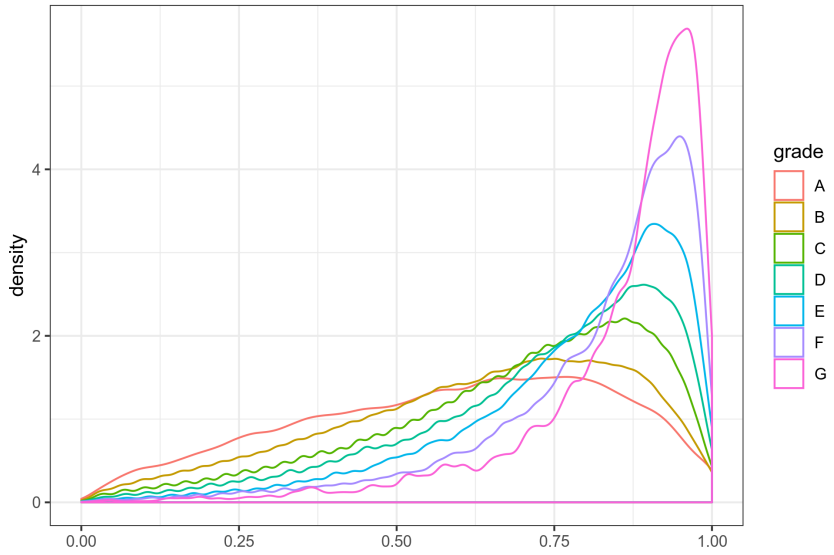


Figure 2.12: Distribution of NPV (%) depending on rating (y-axis square-root scaling)

2.4.4 NPV distribution by rating

Principal loss does not reflect the timing of that loss: a loss now is worse than a loss later. This subsection looks at the NPVs of actual loan cashflow (principal and interest) discounted the risk-free rate.

Figure **??** shows that for a given grade, the NPVs are very widely spread. From top to bottom, loans are group by ratings: from quality ratings of `A` and `B`, average ratings of `C` and `D`, to poor ratings of `E` and below. From left to right, we focus on different parts of how NPVs are distributed. Note that each graph is based on a random sample of 100,000 loans (about 1/12th of the original set) and therefore the NPV densities are comparable from graph to graph.

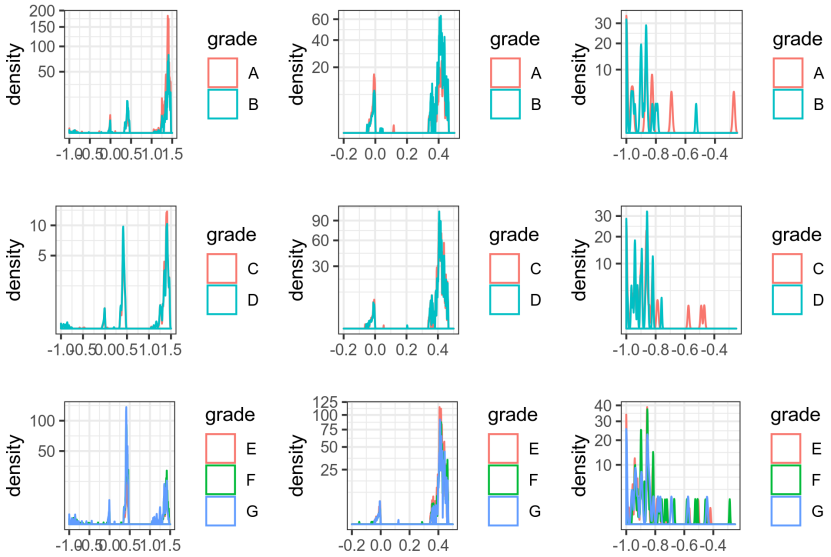


Figure 2.13: Distribution of NPV (%) depending on rating (y-axis square-root scaling)

At the outset, column by column (where NPVs are on the same scale), the NPV distribution show several modes on the same location. The modes are made more apparent by zooming on where the modes are present: the leftmost column basically shows the entire range of the NPVs (as portion of the loan). The middle graph zooms on the -20% / 50% range. The rightmost column zooms on the -100% / -25% section. Looking at the left hand scale, we can see that the lower NPVs overall gain in importance as the loan rating worsen.

Zooming without scaling the y-axis and grouping all the ratings available for investment on a single plot gives more details.

- Figure **??** shows a mode with a maximum around 1.25 / 1.5 being loans seemingly repaid in full (the mode is above 100% given the repayment of principal *and* interest);
- Figure **??** and figure **??** show a second and third mode around 41% and -1%;
- Finally, figure **??** one last very diffuse mode around -100%.

The overall trend is what we should expect. What is surprising is the existence of (1) very clearly defined modes which (2) are common to all types of borrowers. They roughly look log-normal, apart from the mode around 41% which look Gaussian.

2.5 Variables

We here present other aspects of the dataset. The full list of variable is given in appendix (see Table 4.1). This dataset will be reduced as we focused on our core question: *Are LC's loans priced appropriately?*.

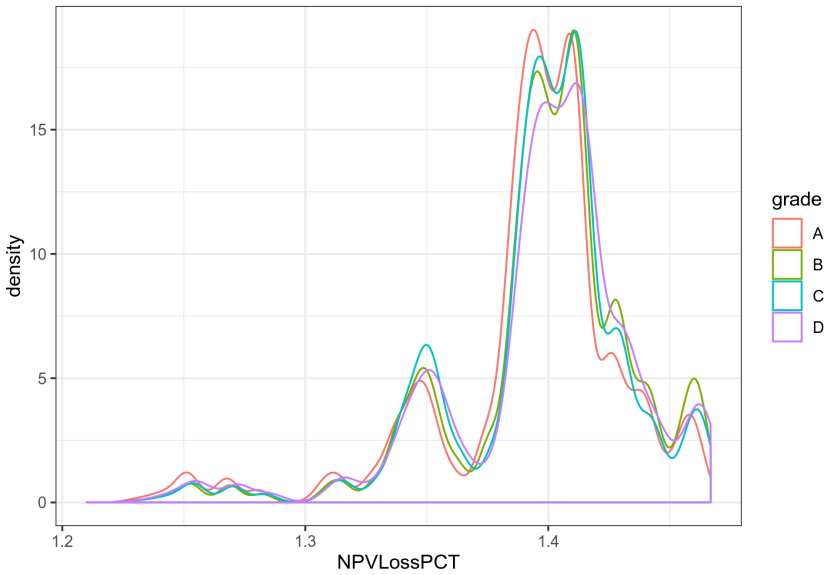


Figure 2.14: NPV % over 120% (no y-axis scaling)

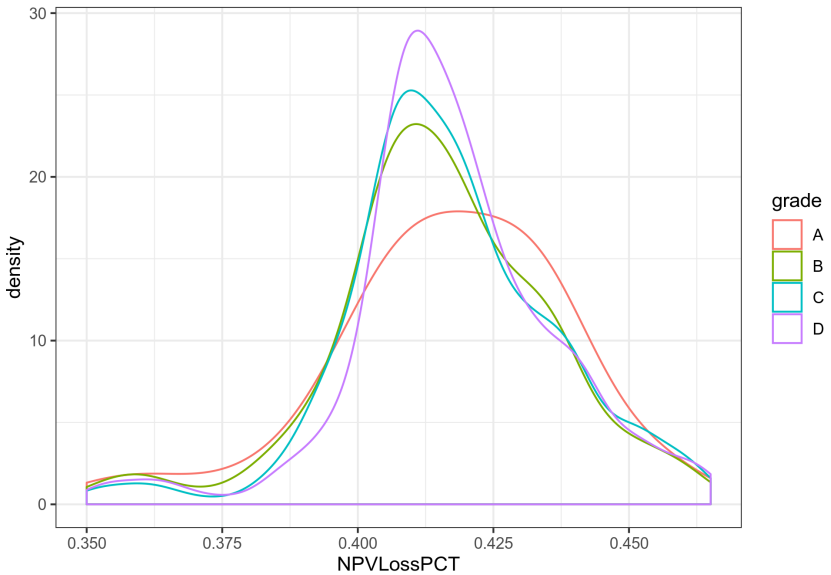


Figure 2.15: NPV % around 41% (no y-axis scaling)

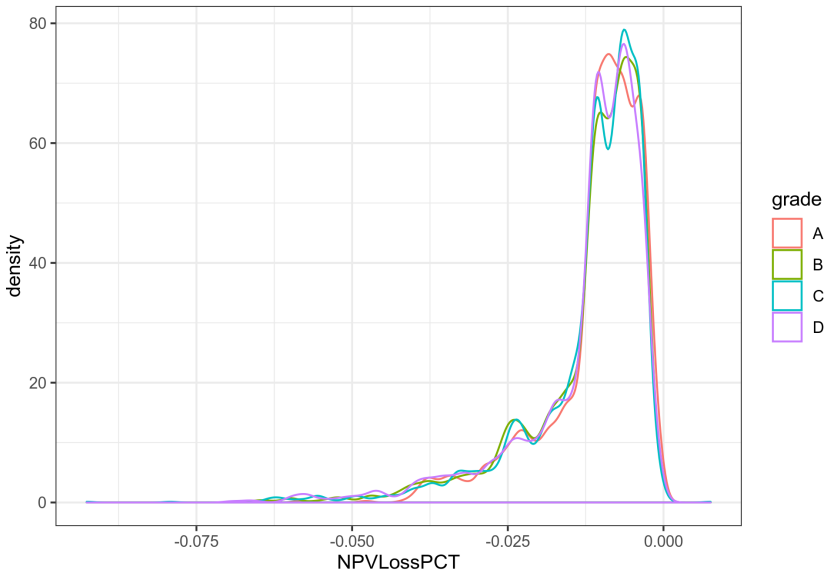


Figure 2.16: NPV % around -1% (no y-axis scaling)

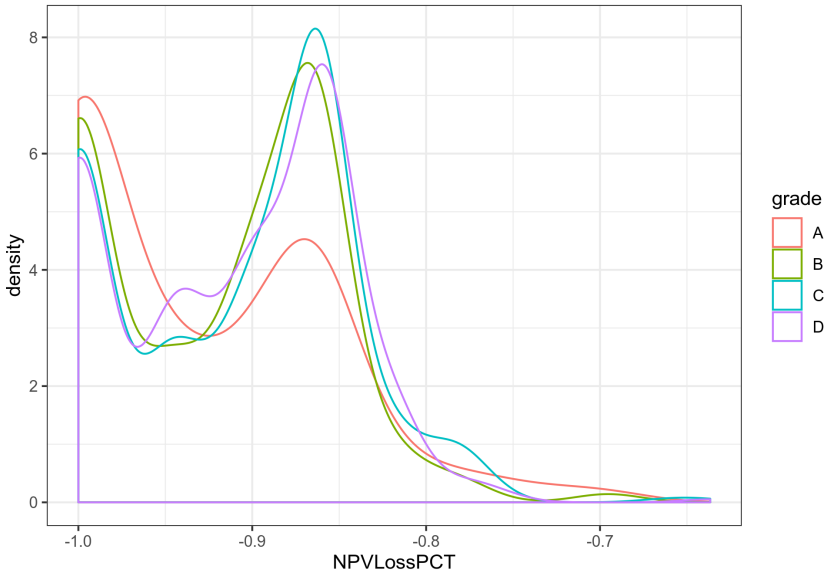


Figure 2.17: NPV % for close to total loss % (no y-axis scaling)

Table 2.2: Matured loans per status

Loan status	Count	Proportion (%)
Fully Paid	964057	24101425
Charged Off	207546	5188650
Does not meet the credit policy. Status:Fully Paid	1334	33350
Does not meet the credit policy. Status:Charged Off	438	10950

2.5.1 Identification

The dataset is anonymised (all identifying ID numbers are deleted) and we therefore removed those columns from the dataset. Since the identification `IDs` have been removed to anonymise the dataset, we cannot see if a borrower borrowed several times.

2.6 Loan decision

As indicated in the introduction, our focus is on loans that have gone through their entire life cycle to consider their respective pricing, risk and profitability. To that effect, we will remove all loans which are still current (either performing or not), and we will only retain loans which currently available (rated `A1` to `D5`). Another reason for limiting ourselves to this subset is that the previous subsection graphing NPVs strongly suggests that the lowest quality loans were very underpriced yielding average losses. From here on, everything will be based on this reduced dataset.

In this reduced dataset, we focus on loans that have matured or been terminated. It contains 1306356 samples. Most of the loans (ca.80%) have been repaid in full. See Table **??**.

When grouped by grade (Figure **??**), we see a clear correlation between grade and default: the lower the grade the higher the portion defaults (note the limited scale with a minimum at about 50%). In addition, most of the business is written in the B- or C-rating range.

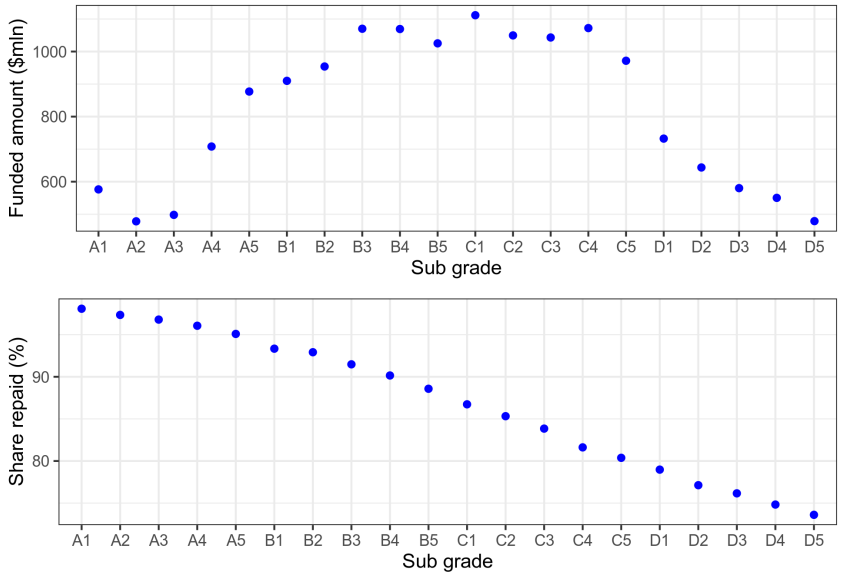


Figure 2.18: Funding and Write-offs by Sub-grades

2.7 Payment-related information

As mentioned previously, the descriptions of the dataset variables is at times incomplete or confusing. For the purpose of determining the cash flow of each individual loans, we have attempted to reconstruct the variables.

We have verfied that:

- installments are calculated as per the formula shown above, rounded to the next cent.
- total_pymnt = total_rec_prncp + total_rec_int + total_rec_late_fee + recoveries

Chapter 3

Logistic Regression Model and Credit Scorecard

At the outset, the dataset presents a number of challenges:

- There is a mix of continuous and categorical data.
- The number of observations is very large.
- The possible number of predictors is large (partially due to one-hot encoding of categorical values).
- As shown in the previous section, credit margins have changed over time. This is clearly related to the wider US economic environment. Financial hardship is a key driver for some of the loans. Availability of disposable income is important to assess the ability to repay. Therefore, the cost of living, which varies from state to state, seems relevant.

[TODO: Bias/Complexity trade-off] [TODO: AoC? or AuC?]

3.1 Introduction

3.1.1 Split

The dataset was randomly split into training and validation sets (80/20 ratio).

We initiated the exploration of potential models with Principal Component Analysis, Linear Regression, Extreme Boosting and Random Forest. No model could be trained on the full set. We therefore came to limit the training set on a random sample of 0.1% (1 thousandth) of the initial full set.

Any model will require training in batch (e.g. stochastic gradient descent) or online. Our untested intuition is that online methods are not adapted to an unbalanced dataset: the number of defaults/write-offs is fairly low for high quality ratings. However, the dataset is evidently a time series which points to online training. We will not explore that line of investigation, although exploring that tension would be an interesting subject.

3.1.2 The modeling task

Two step process: - Given dataset (+/- rating?) find a score between -5 and +5 - Given a score between -5 and +5 given the right parameters for the modes - Given a density curve extrapolate the required margin

- Make money or not?

3.2 Logistic Regression

Logistic models (also called **logit model**) are used to model binary events. Examples would be passing or failing an exam, a newborn being a boy or a girl, a voter choosing a particular political party, or – relevant to us – a borrower defaulting or not on a loan. If the binary variable is modeled as a 0/1 outcome, the model will yield a value between 0 and 1 which can be used as a probabilty.

We are interested in using a number of variables (being continuous and/or categorical) to model the binary response. A natural model is a linear combination of the variables. Since the predicted value would be continuous and not be bounded by 0/1, the outcome is transformed. A commonly used transformation of the logodds (logarithm of the odds given a particular probability) $\log\left(\frac{p}{1-p}\right)$. This expression has a few advantages: it converts any value (between $-\infty$ and $+\infty$ produced by the linear regression), and it is symmetrical around $x = 0$ and $y = 1/2$. That is, using the odds instead of the probability avoids infinity; it behaves identically when p approaches 0 or 1 (this would not be the case using p). The reciprocal of the logodds is $p = \frac{1}{1-e^{-x}}$.

For a number of X_i variables, the model to fit is then:

$$p = \frac{1}{1 - e^{-\sum_i \alpha_i X_i}}$$

The commonly used method to evaluate the creditworthiness of a borrower is to create scorecards whereby particular characteristics are segmented into intervals and attributed discrete scores. In plain English, a continuous variable (say the age of the applicant) is segmented into intervals (e.g. 0-18 year-olds, 18-26 year-olds,...), and then given a score. Those different segments become categorical variables. The task of the model is to:

- identify the best way to segment a continuous variable to maximise the information value of the different segments (called bins): intuitively empty or quasi-empty bins (either no or few applicants in the bin) are not informative; bins for which the response is completely random are not predictive, whereas a bin where the response always has the same response is informative (i.e. anybody with income of 0 and 10 dolars per year will default, anybody with a salary of $1 million per month will repay).
- use a generalised linear model using the new categorical variables;
- transforms the linear coefficients estimated for each category into numerical scores.

  WARNING: To run this model, you will need at least 32GB of memory (real plus virtual/swap space)

3.2.1 Data preparation

WARNING: Execute `CleanLoad.Rmd` first

```
##            used  (Mb) gc trigger   (Mb)  max used   (Mb)
## Ncells   987883  52.8    2568358  137.2   2568358  137.2
## Vcells 79881975 609.5  528502232 4032.2 660021436 5035.6
```

3.3 Binning and Weight of Evidence

This subsection owes a lot influence to the `smbinning` package source code from which we reimplemented some aspects using the **`tidyverse` style** (the original source code uses SQL statements to access dataframes), and the `Information` package vignette [1].

To identify the various bins (and number and location)

Let's say that we have a binary dependent variable Y and a set of predictive variables X1„Xp. As mentioned above, Y can capture a wide range of outcomes, such as defaulting on a loan, clicking on an ad, or terminating a subscription.

WOE and IV play two distinct roles when analyzing data:

`WOE describes the relationship between a predictive variable and a binary target variable.`
`IV measures the strength of that relationship.`

3.3.1 Loop through all variables

```
## Variable:  loanID is numeric, Number of missing/NA values:  0
##      IV :  8.3854
## Variable:  loan_amnt is numeric, Number of missing/NA values:  0
##      IV :  8.076185
## Variable:  term is numeric, Number of missing/NA values:  0
##      IV :  1.372439
## Variable:  int_rate is numeric, Number of missing/NA values:  0
##      IV :  17.48022
## Variable:  grade is a factor,      IV :  4.896228
## Variable:  sub_grade is a factor,      IV :  25.22348
## Variable:  emp_length is a factor,      IV :  7.563685
## Variable:  home_ownership is a factor,      IV :  5.209782
## Variable:  verification_status is a factor,      IV :  2.601848
## Variable:  issue_d is numeric, Number of missing/NA values:  0
##      IV :  5.108584
## Variable:  purpose is a factor,      IV :  12.00144
## Variable:  addr_state is a factor,      IV :  44.35039
## Variable:  dti is numeric, Number of missing/NA values:  247
##      IV :  10.57304
## Variable:  delinq_2yrs is numeric, Number of missing/NA values:  23
##      IV :  4.231142
## Variable:  earliest_cr_line is numeric, Number of missing/NA values:  23
##      IV :  10.28436
## Variable:  inq_last_6mths is numeric, Number of missing/NA values:  24
##      IV :  4.767064
## Variable:  mths_since_last_delinq is numeric, Number of missing/NA values:  527124
##      IV :  3.188741
## Variable:  mths_since_last_record is numeric, Number of missing/NA values:  866836
##      IV :  2.197638
## Variable:  open_acc is numeric, Number of missing/NA values:  23
##      IV :  6.81435
## Variable:  pub_rec is numeric, Number of missing/NA values:  23
##      IV :  3.467995
## Variable:  revol_bal is numeric, Number of missing/NA values:  0
##      IV :  1.835544
## Variable:  revol_util is numeric, Number of missing/NA values:  687
##      IV :  8.817536
```

[1]

```
## Variable:  total_acc is numeric, Number of missing/NA values:  23
##      IV :  4.325905
## Variable:  application_type is a factor,      IV :  1.389787
## Variable:  annual_inc_joint is numeric, Variable skipped.
## Variable:  dti_joint is numeric, Variable skipped.
## Variable:  verification_status_joint is a factor,      IV :  2.33501
## Variable:  acc_now_delinq is numeric, Variable skipped.
## Variable:  open_rv_12m is numeric, Number of missing/NA values:  645401
##      IV :  3.507841
## Variable:  open_rv_24m is numeric, Number of missing/NA values:  645401
##      IV :  5.036733
## Variable:  max_bal_bc is numeric, Number of missing/NA values:  645401
##      IV :  3.076012
## Variable:  inq_fi is numeric, Number of missing/NA values:  645401
##      IV :  2.820658
## Variable:  avg_cur_bal is numeric, Number of missing/NA values:  56032
##      IV :  8.240672
## Variable:  bc_open_to_buy is numeric, Number of missing/NA values:  51160
##      IV :  8.26883
## Variable:  bc_util is numeric, Number of missing/NA values:  51160
##      IV :  8.26883
## Variable:  mo_sin_old_il_acct is numeric, Number of missing/NA values:  85391
##      IV :  5.512796
## Variable:  mo_sin_old_rev_tl_op is numeric, Number of missing/NA values:  56016
##      IV :  6.366685
## Variable:  mo_sin_rcnt_rev_tl_op is numeric, Number of missing/NA values:  56016
##      IV :  9.361446
## Variable:  mo_sin_rcnt_tl is numeric, Number of missing/NA values:  56015
##      IV :  10.66616
## Variable:  mort_acc is numeric, Number of missing/NA values:  39839
##      IV :  6.984269
## Variable:  mths_since_recent_bc is numeric, Number of missing/NA values:  49826
##      IV :  8.715511
## Variable:  mths_since_recent_bc_dlq is numeric, Number of missing/NA values:  797767
##      IV :  2.357887
## Variable:  mths_since_recent_inq is numeric, Number of missing/NA values:  137406
##      IV :  9.039433
## Variable:  mths_since_recent_revol_delinq is numeric, Number of missing/NA values:  696281
##      IV :  3.158398
## Variable:  num_accts_ever_120_pd is numeric, Number of missing/NA values:  56015
##      IV :  3.456826
## Variable:  num_actv_bc_tl is numeric, Number of missing/NA values:  56015
##      IV :  6.586561
## Variable:  num_actv_rev_tl is numeric, Number of missing/NA values:  56015
##      IV :  8.200172
## Variable:  num_bc_sats is numeric, Number of missing/NA values:  46723
##      IV :  5.09235
## Variable:  num_bc_tl is numeric, Number of missing/NA values:  56015
##      IV :  5.826587
## Variable:  num_il_tl is numeric, Number of missing/NA values:  56015
##      IV :  3.430035
## Variable:  num_op_rev_tl is numeric, Number of missing/NA values:  56015
##      IV :  6.597519
## Variable:  num_rev_accts is numeric, Number of missing/NA values:  56015
##      IV :  3.486005
## Variable:  num_rev_tl_bal_gt_0 is numeric, Number of missing/NA values:  56015
##      IV :  8.183789
## Variable:  num_sats is numeric, Number of missing/NA values:  46723
```

```
##      IV :  5.970552
## Variable:  num_tl_120dpd_2m is numeric, Variable skipped.
## Variable:  num_tl_30dpd is numeric, Variable skipped.
## Variable:  num_tl_90g_dpd_24m is numeric, Number of missing/NA values:  56015
##      IV :  2.626433
## Variable:  num_tl_op_past_12m is numeric, Number of missing/NA values:  56015
##      IV :  5.797221
## Variable:  pct_tl_nvr_dlq is numeric, Number of missing/NA values:  56139
##      IV :  2.75026
## Variable:  percent_bc_gt_75 is numeric, Number of missing/NA values:  50862
##      IV :  6.670696
## Variable:  pub_rec_bankruptcies is numeric, Number of missing/NA values:  1114
##      IV :  2.126956
## Variable:  tax_liens is numeric, Variable skipped.
## Variable:  tot_hi_cred_lim is numeric, Number of missing/NA values:  56015
##      IV :  10.05288
## Variable:  total_bal_ex_mort is numeric, Number of missing/NA values:  39839
##      IV :  5.490049
## Variable:  total_bc_limit is numeric, Number of missing/NA values:  39839
##      IV :  10.69037
## Variable:  total_il_high_credit_limit is numeric, Variable skipped.
## Variable:  revol_bal_joint is numeric, Variable skipped.
## Variable:  disbursement_method is a factor,      IV :  1.556783
## Variable:  issue_d2 is numeric, Number of missing/NA values:  0
##      IV :  6.233118
## Variable:  issue_d3 is numeric, Number of missing/NA values:  0
##      IV :  6.233118
## Variable:  earliest_cr_line2 is numeric, Number of missing/NA values:  23
##      IV :  4.594184
## Variable:  earliest_cr_line3 is numeric, Number of missing/NA values:  23
##      IV :  4.885613
## Variable:  isGoodLoan
```

```
##    user  system elapsed
## 510.521   0.563 511.435
```

3.3.2 Select relevant variables

Ignore the variables that would not be available at the time the credit scoring is performed.

```
##   grade nGood  nBad Count cumCount cumGood cumBad    pctCount     pctGood     pctBad pctGoodBin pc
## 1     F 13925 11556 25481  1037695  831541 206154 0.024381772 0.016671975 0.05506791  0.5464856 0
## 2     G  3693  3696  7389  1045084  835234 209850 0.007070245 0.004421515 0.01761258  0.4997970 0
##             WoE           IV
## 1  0.1864810032 1.733737e-02
## 2 -0.0008120179 3.296865e-07
##   sub_grade nGood nBad Count cumCount cumGood cumBad     pctCount      pctGood      pctBad pctGooc
## 1        F3  2663 2189  4852  1024929  824821 200108 0.0046426890 0.0031883281 0.010431260  0.5488
## 2        G2   878  835  1713  1026642  825699 200943 0.0016391027 0.0010512024 0.003979033  0.5125
## 3        G4   512  547  1059  1027701  826211 201490 0.0010133157 0.0006130019 0.002606624  0.4834
## 4        F2  3111 2628  5739  1033440  829322 204118 0.0054914246 0.0037247047 0.012523231  0.5420
## 5        G1  1226 1149  2375  1035815  830548 205267 0.0022725446 0.0014678521 0.005475340  0.5161
## 6        F4  2011 1867  3878  1039693  832559 207134 0.0037107065 0.0024077085 0.008896831  0.5185
## 7        F5  1598 1551  3149  1042842  834157 208685 0.0030131549 0.0019132363 0.007390994  0.5074
## 8        G3   627  671  1298  1044140  834784 209356 0.0012420054 0.0007506878 0.003197522  0.4830
## 9        G5   450  494   944  1045084  835234 209850 0.0009032767 0.0005387712 0.002354062  0.4760
##        LnOdds         WoE           IV
## 1  0.19600849  0.19600849 0.0191483972
## 2  0.05021487  0.05021487 0.0012605017
```

```
## 3 -0.06612418 -0.06612418 0.0021854072
## 4  0.16872112  0.16872112 0.0141997386
## 5  0.06486484  0.06486484 0.0021029863
## 6  0.07429925  0.07429925 0.0027589199
## 7  0.02985296  0.02985296 0.0004455666
## 8 -0.06782260 -0.06782260 0.0022990711
## 9 -0.09328793 -0.09328793 0.0043481664
```

variable	type	IV	WoE
addr_state	categorical	44.350394	list(addr_state = c(11, 39, 5, 28, 4, 23, 37, 2, 24, 44, 3,
purpose	categorical	12.001442	list(purpose = c(3, 1, 2, 6, 5, 10, 7, 8, 12, 14, 9, 13, 11,
total_bc_limit	numeric	10.690371	list(CutNumber = c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12),
mo_sin_rcnt_tl	numeric	10.666158	list(CutNumber = c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12),
dti	numeric	10.573037	list(CutNumber = c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 1
earliest_cr_line	numeric	10.284359	list(CutNumber = c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11), Cut
tot_hi_cred_lim	numeric	10.052881	list(CutNumber = c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10), CutPoin
mo_sin_rcnt_rev_tl_op	numeric	9.361446	list(CutNumber = c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11), Cut
mths_since_recent_inq	numeric	9.039433	list(CutNumber = c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11), Cut
revol_util	numeric	8.817536	list(CutNumber = c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10), CutPoin
mths_since_recent_bc	numeric	8.715511	list(CutNumber = c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10), CutPoin
bc_open_to_buy	numeric	8.268830	list(CutNumber = c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10), CutPoin
bc_util	numeric	8.268830	list(CutNumber = c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10), CutPoin
avg_cur_bal	numeric	8.240672	list(CutNumber = c(1, 2, 3, 4, 5, 6, 7, 8), CutPoint = c(
num_actv_rev_tl	numeric	8.200172	list(CutNumber = c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10), CutPoin
num_rev_tl_bal_gt_0	numeric	8.183789	list(CutNumber = c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10), CutPoin
loan_amnt	numeric	8.076185	list(CutNumber = c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10), CutPoin
mort_acc	numeric	6.984269	list(CutNumber = c(1, 2, 3, 4, 5, 6, 7), CutPoint = c(0,
open_acc	numeric	6.814350	list(CutNumber = c(1, 2, 3, 4, 5, 6, 7), CutPoint = c(7,
percent_bc_gt_75	numeric	6.670696	list(CutNumber = c(1, 2, 3, 4, 5, 6, 7, 8), CutPoint = c(
num_op_rev_tl	numeric	6.597519	list(CutNumber = c(1, 2, 3, 4, 5, 6, 7, 8), CutPoint = c(
num_actv_bc_tl	numeric	6.586561	list(CutNumber = c(1, 2, 3, 4, 5, 6, 7, 8), CutPoint = c(
mo_sin_old_rev_tl_op	numeric	6.366685	list(CutNumber = c(1, 2, 3, 4, 5, 6, 7, 8), CutPoint = c(
issue_d2	numeric	6.233118	list(CutNumber = c(1, 2, 3, 4, 5, 6, 7), CutPoint = c(18
issue_d3	numeric	6.233118	list(CutNumber = c(1, 2, 3, 4, 5, 6, 7), CutPoint = c(25
num_sats	numeric	5.970552	list(CutNumber = c(1, 2, 3, 4, 5, 6, 7), CutPoint = c(7,
num_bc_tl	numeric	5.826587	list(CutNumber = c(1, 2, 3, 4, 5, 6, 7), CutPoint = c(2,
num_tl_op_past_12m	numeric	5.797221	list(CutNumber = c(1, 2, 3, 4, 5, 6, 7), CutPoint = c(0,
mo_sin_old_il_acct	numeric	5.512796	list(CutNumber = c(1, 2, 3, 4, 5, 6, 7), CutPoint = c(35
total_bal_ex_mort	numeric	5.490049	list(CutNumber = c(1, 2, 3, 4, 5, 6), CutPoint = c(8506,
issue_d	numeric	5.108584	list(CutNumber = c(1, 2, 3, 4, 5, 6), CutPoint = c(14.16
num_bc_sats	numeric	5.092350	list(CutNumber = c(1, 2, 3, 4, 5, 6), CutPoint = c(1, 4,
open_rv_24m	numeric	5.036733	list(CutNumber = c(1, 2, 3, 4, 5, 6, 7), CutPoint = c(0,
earliest_cr_line3	numeric	4.885613	list(CutNumber = c(1, 2, 3, 4, 5), CutPoint = c(5.35937
inq_last_6mths	numeric	4.767064	list(CutNumber = c(1, 2, 3, 4, 5), CutPoint = c(0, 1, 2,
earliest_cr_line2	numeric	4.594184	list(CutNumber = c(1, 2, 3, 4), CutPoint = c(3.3611111
total_acc	numeric	4.325905	list(CutNumber = c(1, 2, 3, 4), CutPoint = c(13, 16, 176
delinq_2yrs	numeric	4.231142	list(CutNumber = c(1, 2, 3, 4), CutPoint = c(0, 1, 39, N
open_rv_12m	numeric	3.507841	list(CutNumber = c(1, 2, 3, 4, 5), CutPoint = c(0, 1, 2,
num_rev_accts	numeric	3.486006	list(CutNumber = c(1, 2, 3, 4), CutPoint = c(5, 8, 128,
pub_rec	numeric	3.467995	list(CutNumber = c(1, 2, 3), CutPoint = c(0, 63, NA), N
num_accts_ever_120_pd	numeric	3.456826	list(CutNumber = c(1, 2, 3, 4), CutPoint = c(0, 2, 51, N
num_il_tl	numeric	3.430035	list(CutNumber = c(1, 2, 3, 4), CutPoint = c(1, 22, 150,
mths_since_last_delinq	numeric	3.188741	list(CutNumber = c(1, 2, 3, 4), CutPoint = c(15, 32, 226
mths_since_recent_revol_delinq	numeric	3.158398	list(CutNumber = c(1, 2, 3, 4), CutPoint = c(15, 32, 202
max_bal_bc	numeric	3.076012	list(CutNumber = c(1, 2, 3, 4), CutPoint = c(5122, 7905
inq_fi	numeric	2.820658	list(CutNumber = c(1, 2, 3, 4), CutPoint = c(0, 2, 48, N
pct_tl_nvr_dlq	numeric	2.750260	list(CutNumber = c(1, 2, 3), CutPoint = c(0.959, 1, NA
num_tl_90g_dpd_24m	numeric	2.626434	list(CutNumber = c(1, 2, 3), CutPoint = c(0, 39, NA), N
verification_status	categorical	2.601848	list(verification_status = c(2, 1, 3), nGood = c(318103,
mths_since_recent_bc_dlq	numeric	2.357887	list(CutNumber = c(1, 2, 3), CutPoint = c(29, 202, NA)
verification_status_joint	categorical	2.335010	list(verification_status_joint = c(1, 2, 4, 3), nGood = c
mths_since_last_record	numeric	2.197637	list(CutNumber = c(1, 2, 3), CutPoint = c(67, 129, NA)
pub_rec_bankruptcies	numeric	2.126956	list(CutNumber = c(1, 2, 3), CutPoint = c(0, 12, NA), N
revol_bal	numeric	1.835544	list(CutNumber = c(1, 2), CutPoint = c(36178, 2568995
disbursement_method	categorical	1.556783	list(disbursement_method = 1:2, nGood = c(831588, 36
application_type	categorical	1.389787	list(application_type = 1:2, nGood = c(821317, 13917),

3.3.3 Create data table with only binary variables (transform every bin to a 0/1 value)

For each variable, create new variables for each bin in the WoE table of that variable.

```
## -------------------- 1 -- addr_state
## -------------------- 2 -- purpose
## -------------------- 3 -- total_bc_limit
## -------------------- 4 -- mo_sin_rcnt_tl
## -------------------- 5 -- dti
## -------------------- 6 -- earliest_cr_line
## -------------------- 7 -- tot_hi_cred_lim
## -------------------- 8 -- mo_sin_rcnt_rev_tl_op
## -------------------- 9 -- mths_since_recent_inq
## -------------------- 10 -- revol_util
```

3.3.4 Comparison of individual characteristics

[TODO] Present data by order of IV Sort WoE buildup for key IV

- re-train secures the lowest Akaike criterion and will be chosen.
- On this test, the best variables are: SORT BY IV. Then WoE for the first few ones.

3.3.4.1 dti

```
## Number of missing/NA values:  47
```

CutNumber	CutPoint	Min	Max	Count	nGood	nBad	cumCount	cumGood	cumBad	pctCount
1	0.0506	-Inf	0.0506	10765	9162	1603	10765	9162	1603	0.0515032
2	0.0739	0.0506	0.0739	10745	9265	1480	21510	18427	3083	0.0514075
3	0.1119	0.0739	0.1119	25635	21608	4027	47145	40035	7110	0.1226461
4	0.1500	0.1119	0.1500	33275	27702	5573	80420	67737	12683	0.1591983
5	0.1709	0.1500	0.1709	19093	15547	3546	99513	83284	16229	0.0913471
6	0.1996	0.1709	0.1996	25562	20483	5079	125075	103767	21308	0.1222969
7	0.2429	0.1996	0.2429	33232	25855	7377	158307	129622	28685	0.1589926
8	0.2871	0.2429	0.2871	25019	19025	5994	183326	148647	34679	0.1196990
9	9.9900	0.2871	9.9900	25643	18290	7353	208969	166937	42032	0.1226844
10	NA	NA	NA	47	36	11	209016	166973	42043	0.0002249

```
## [1] 8.859178
```

3.3.4.2 Home ownership

```
## $IV
## [1] 5.611856
##
## $type
## [1] "categorical"
##
## $table
## # A tibble: 6 x 16
##   home_ownership  nBad nGood  Count cumCount cumGood cumBad pctCount pctGood  pctBad pctGoodBin pc
##   <fct>          <int> <int>  <int>    <int>   <int>  <int>    <dbl>   <dbl>   <dbl>      <dbl>
## 1 RENT           19564 63839  83403    83403   63839  19564  3.99e-1 3.82e-1 4.65e-1      0.765
## 2 MORTGAGE       17959 85422 103381   186784  149261  37523  4.95e-1 5.12e-1 4.27e-1      0.826
## 3 OWN             4507 17646  22153   208937  166907  42030  1.06e-1 1.06e-1 1.07e-1      0.797
## 4 OTHER              5    20     25   208962  166927  42035  1.20e-4 1.20e-4 1.19e-4      0.8
```

```
## 5 ANY                7    41     48   209010  166968  42042  2.30e-4 2.46e-4 1.66e-4      0.854
## 6 NONE               1     5      6   209016  166973  42043  2.87e-5 2.99e-5 2.38e-5      0.833
```

```
## [1] 5.611856
```

3.3.4.3 Verification Status

```
## $IV
## [1] 2.599975
##
## $type
## [1] "categorical"
##
## $table
## # A tibble: 3 x 16
##   verification_st~  nBad nGood Count cumCount cumGood cumBad pctCount pctGood pctBad pctGoodBin pc
##   <fct>            <int> <int> <int>    <int>   <int>  <int>    <dbl>   <dbl>  <dbl>      <dbl>
## 1 Verified         15740 49519 65259    65259   49519  15740    0.312   0.297  0.374      0.759
## 2 Not Verified      9208 53692 62900   128159  103211  24948    0.301   0.322  0.219      0.854
## 3 Source Verified  17095 63762 80857   209016  166973  42043    0.387   0.382  0.407      0.789
```

```
## [1] 2.599975
```

3.3.4.4 Purpose

```
## $IV
## [1] 12.01481
##
## $type
## [1] "categorical"
##
## $table
## # A tibble: 14 x 16
##    purpose        nBad nGood  Count cumCount cumGood cumBad pctCount pctGood  pctBad pctGoodBin pc
##    <fct>         <int> <int>  <int>    <int>   <int>  <int>    <dbl>   <dbl>   <dbl>      <dbl>
##  1 debt_consoli~ 25797 95832 121629   121629   95832  25797 0.582    5.74e-1 6.14e-1      0.788
##  2 major_purcha~   841  3664   4505   126134   99496  26638 0.0216   2.19e-2 2.00e-2      0.813
##  3 small_busine~   686  1688   2374   128508  101184  27324 0.0114   1.01e-2 1.63e-2      0.711
##  4 other          2576  9425  12001   140509  110609  29900 0.0574   5.64e-2 6.13e-2      0.785
##  5 medical         533  1872   2405   142914  112481  30433 0.0115   1.12e-2 1.27e-2      0.778
##  6 credit_card    7810 37704  45514   188428  150185  38243 0.218    2.26e-1 1.86e-1      0.828
##  7 vacation        275  1137   1412   189840  151322  38518 0.00676  6.81e-3 6.54e-3      0.805
##  8 home_improve~  2495 11216  13711   203551  162538  41013 0.0656   6.72e-2 5.93e-2      0.818
##  9 car             328  1917   2245   205796  164455  41341 0.0107   1.15e-2 7.80e-3      0.854
## 10 moving          376  1129   1505   207301  165584  41717 0.00720  6.76e-3 8.94e-3      0.750
## 11 house           237   904   1141   208442  166488  41954 0.00546  5.41e-3 5.64e-3      0.792
## 12 wedding          48   321    369   208811  166809  42002 0.00177  1.92e-3 1.14e-3      0.870
## 13 renewable_en~    27   107    134   208945  166916  42029 0.000641 6.41e-4 6.42e-4      0.799
## 14 educational      14    57     71   209016  166973  42043 0.000340 3.41e-4 3.33e-4      0.803
```

```
## [1] 12.01481
```

3.3.4.5 Sub grade

```
## $IV
## [1] 25.29456
##
```

```
## $type
## [1] "categorical"
##
## $table
## # A tibble: 35 x 16
##    sub_grade  nBad nGood Count cumCount cumGood cumBad pctCount pctGood  pctBad pctGoodBin pctBadl
##    <fct>     <int> <int> <int>    <int>   <int>  <int>    <dbl>   <dbl>   <dbl>      <dbl>     <dl
##  1 D2         2141  4924  7065     7065    4924   2141   0.0338  0.0295 0.0509       0.697    0.30
##  2 A5          849  9171 10020    17085   14095   2990   0.0479  0.0549 0.0202       0.915    0.08
##  3 B2         1244 10101 11345    28430   24196   4234   0.0543  0.0605 0.0296       0.890    0.11
##  4 C3         2653  9019 11672    40102   33215   6887   0.0558  0.0540 0.0631       0.773    0.22
##  5 D4         1807  3768  5575    45677   36983   8694   0.0267  0.0226 0.0430       0.676    0.32
##  6 C5         2731  7602 10333    56010   44585  11425   0.0494  0.0455 0.0650       0.736    0.26
##  7 B3         1652 11186 12838    68848   55771  13077   0.0614  0.0670 0.0393       0.871    0.12
##  8 D1         2287  5659  7946    76794   61430  15364   0.0380  0.0339 0.0544       0.712    0.28
##  9 A3          307  5582  5889    82683   67012  15671   0.0282  0.0334 0.00730      0.948    0.05
## 10 B5         2133 10749 12882    95565   77761  17804   0.0616  0.0644 0.0507       0.834    0.16
## # ... with 25 more rows
```

```
## [1] 25.29456
```

3.3.4.6 Employment years

```
## $IV
## [1] 7.56036
##
## $type
## [1] "categorical"
##
## $table
## # A tibble: 9 x 16
##   emp_length  nBad nGood  Count cumCount cumGood cumBad pctCount pctGood pctBad pctGoodBin pctBadl
##   <fct>      <int> <int>  <int>    <int>   <int>  <int>    <dbl>   <dbl>  <dbl>      <dbl>     <dl
## 1 <NA>       22646 88650 111296   111296   88650  22646   0.532   0.531  0.539       0.797     0.2
## 2 7           1886  7409   9295   120591   96059  24532   0.0445  0.0444 0.0449      0.797     0.2
## 3 5           2606 10426  13032   133623  106485  27138   0.0623  0.0624 0.0620      0.800     0.2
## 4 8           1920  7567   9487   143110  114052  29058   0.0454  0.0453 0.0457      0.798     0.2
## 5 3           3423 13222  16645   159755  127274  32481   0.0796  0.0792 0.0814      0.794     0.2
## 6 2           3772 15398  19170   178925  142672  36253   0.0917  0.0922 0.0897      0.803     0.2
## 7 9           1503  6172   7675   186600  148844  37756   0.0367  0.0370 0.0357      0.804     0.2
## 8 6           1871  7944   9815   196415  156788  39627   0.0470  0.0476 0.0445      0.809     0.2
## 9 4           2416 10185  12601   209016  166973  42043   0.0603  0.0610 0.0575      0.808     0.2
```

```
## [1] 7.56036
```

3.3.4.7 Bank card utilisation

```
## Number of missing/NA values:  10370
```

```
## $IV
## [1] 5.776365
##
## $type
## [1] "numeric"
##
## $table
## # A tibble: 7 x 19
```

```
##   CutNumber CutPoint    Min   Max Count nGood  nBad cumCount cumGood cumBad pctCount pctGood pctBa
##       <dbl>    <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl>    <dbl>   <dbl>  <dbl>    <dbl>   <dbl>  <dbl
## 1         1     31.5 -Inf    31.5 37880 31697  6183    37880   31697   6183   0.181   0.190  0.147
## 2         2     52.2   31.5  52.2 38014 30989  7025    75894   62686  13208   0.182   0.186  0.167
## 3         3     76.1   52.2  76.1 52106 41337 10769   128000  104023  23977   0.249   0.248  0.256
## 4         4     90.8   76.1  90.8 36497 28541  7956   164497  132564  31933   0.175   0.171  0.189
## 5         5     95.4   90.8  95.4 14346 10999  3347   178843  143563  35280   0.0686  0.0659 0.079
## 6         6    203.    95.4 203.  19803 14780  5023   198646  158343  40303   0.0947  0.0885 0.119
## 7         7     NA     NA    NA   10370  8630  1740   209016  166973  42043   0.0496  0.0517 0.041
## # ... with 2 more variables: pctGoodBin <dbl>, pctBadBin <dbl>
```

```
## [1] 5.776365
```

3.3.4.8 State

```
## $IV
## [1] 43.96563
##
## $type
## [1] "categorical"
##
## $table
## # A tibble: 51 x 16
##    addr_state  nBad nGood Count cumCount cumGood cumBad pctCount pctGood  pctBad pctGoodBin pctBad
##    <fct>      <int> <int> <int>    <int>   <int>  <int>    <dbl>   <dbl>   <dbl>      <dbl>     <
##  1 CO           725  3908  4633     4633    3908    725  0.0222  0.0234  0.0172       0.844     0
##  2 MD          1074  3683  4757     9390    7591   1799  0.0228  0.0221  0.0255       0.774     0
##  3 WY            73   387   460     9850    7978   1872  0.00220 0.00232 0.00174      0.841     0
##  4 CA          6050 24682 30732    40582   32660   7922  0.147   0.148   0.144        0.803     0
##  5 NY          3839 13300 17139    57721   45960  11761  0.0820  0.0797  0.0913       0.776     0
##  6 MA           923  3797  4720    62441   49757  12684  0.0226  0.0227  0.0220       0.804     0
##  7 GA          1260  5563  6823    69264   55320  13944  0.0326  0.0333  0.0300       0.815     0
##  8 NJ          1589  5815  7404    76668   61135  15533  0.0354  0.0348  0.0378       0.785     0
##  9 TN           662  2458  3120    79788   63593  16195  0.0149  0.0147  0.0157       0.788     0
## 10 IL          1498  6480  7978    87766   70073  17693  0.0382  0.0388  0.0356       0.812     0
## # ... with 41 more rows
```

```
## [1] 43.96563
```

3.3.4.9 Annual income

```
## [1] 25
```

```
## $IV
## [1] 12.01481
##
## $type
## [1] "categorical"
##
## $table
## # A tibble: 14 x 16
##    purpose        nBad nGood  Count cumCount cumGood cumBad pctCount pctGood  pctBad pctGoodBin pc
##    <fct>         <int> <int>  <int>    <int>   <int>  <int>    <dbl>   <dbl>   <dbl>      <dbl>
##  1 debt_consoli~ 25797 95832 121629   121629   95832  25797 0.582    5.74e-1 6.14e-1      0.788
##  2 major_purcha~   841  3664   4505   126134   99496  26638 0.0216   2.19e-2 2.00e-2      0.813
##  3 small_busine~   686  1688   2374   128508  101184  27324 0.0114   1.01e-2 1.63e-2      0.711
##  4 other          2576  9425  12001   140509  110609  29900 0.0574   5.64e-2 6.13e-2      0.785
```

```
##  5 medical         533  1872   2405   142914  112481  30433 0.0115   1.12e-2 1.27e-2      0.778
##  6 credit_card    7810 37704  45514   188428  150185  38243 0.218    2.26e-1 1.86e-1      0.828
##  7 vacation        275  1137   1412   189840  151322  38518 0.00676  6.81e-3 6.54e-3      0.805
##  8 home_improve~  2495 11216  13711   203551  162538  41013 0.0656   6.72e-2 5.93e-2      0.818
##  9 car             328  1917   2245   205796  164455  41341 0.0107   1.15e-2 7.80e-3      0.854
## 10 moving          376  1129   1505   207301  165584  41717 0.00720  6.76e-3 8.94e-3      0.750
## 11 house           237   904   1141   208442  166488  41954 0.00546  5.41e-3 5.64e-3      0.792
## 12 wedding          48   321    369   208811  166809  42002 0.00177  1.92e-3 1.14e-3      0.870
## 13 renewable_en~    27   107    134   208945  166916  42029 0.000641 6.41e-4 6.42e-4      0.799
## 14 educational      14    57     71   209016  166973  42043 0.000340 3.41e-4 3.33e-4      0.803
```

```
## NULL
```

3.3.4.10 Loan amount

```
## Number of missing/NA values:  0
```

```
## $IV
## [1] 6.066515
##
## $type
## [1] "numeric"
##
## $table
## # A tibble: 7 x 19
##   CutNumber CutPoint   Min   Max Count nGood  nBad cumCount cumGood cumBad pctCount pctGood pctBa
##       <dbl>    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>    <dbl>   <dbl>  <dbl>    <dbl>   <dbl>  <dbl
## 1         1     3500  -Inf  3500 12109 10325  1784    12109   10325   1784   0.0579  0.0618 0.042
## 2         2     9000  3500  9000 54266 45162  9104    66375   55487  10888   0.260   0.270  0.217
## 3         3    10000  9000 10000 20022 16170  3852    86397   71657  14740   0.0958  0.0968 0.091
## 4         4    14975 10000 14975 33242 26250  6992   119639   97907  21732   0.159   0.157  0.166
## 5         5    15000 14975 15000 10817  8743  2074   130456  106650  23806   0.0518  0.0524 0.049
## 6         6    28000 15000 28000 60284 46570 13714   190740  153220  37520   0.288   0.279  0.326
## 7         7    40000 28000 40000 18276 13753  4523   209016  166973  42043   0.0874  0.0824 0.108
## # ... with 2 more variables: pctGoodBin <dbl>, pctBadBin <dbl>
```

```
## [1] 6.066515
```

3.4 Logistic Regression

3.4.1 Logistic regression using the `SpeedGLM` package

NOTE: `speedglm` has a `select()` function which would shadow `dplyr::select()`, so it is only used fully qualified to avoid any collision.

```
##            used  (Mb) gc trigger   (Mb)  max used   (Mb)
## Ncells  1960612 104.8    3946045  210.8   2570668  137.3
## Vcells 89022495 679.2  422802609 3225.8 660022508 5035.6
```

```
## [1] 78
```

```
## [1] 4
```

3.4.2 Remove identical bins

```
## character(0)
```

3.4.3 First training on variables previously selected on their Information Value

We use `speedglm` being quick. Note that alternatives were also tried: `glm` crashed on even small extracts of the dataset. `glmnet` returns errors that were not understandable or documented on the internet.

3.4.3.1 GLM on characteristics

```
##           used  (Mb) gc trigger   (Mb)  max used   (Mb)
## Ncells 1948309 104.1    3910885  208.9   2570682  137.3
## Vcells 3820520  29.2  422802612 3225.8 660022511 5035.6
```

```
## 7.784 1.915 9.709 0 0
```

```
## Generalized Linear Model of class 'speedglm':
##
## Call:  speedglm::speedglm(formula = isGoodLoan ~ ., data = loansData,      family = binomial())
##
## Coefficients:
##  ---------------------------------------------------------------------
##                                              Estimate Std. Error  z value  Pr(>|z|)
## (Intercept)                                  1.512339   0.057223  26.4287 6.42e-154 ***
## addr_stateAL                                -0.160005   0.054417  -2.9404  3.28e-03 **
## addr_stateAR                                -0.191424   0.056944  -3.3616  7.75e-04 ***
## addr_stateAZ                                 0.049890   0.052727   0.9462  3.44e-01
## addr_stateCA                                 0.046384   0.050714   0.9146  3.60e-01
## addr_stateCO                                 0.325793   0.053481   6.0918  1.12e-09 ***
## addr_stateCT                                 0.166316   0.054659   3.0428  2.34e-03 **
## addr_stateDC                                 0.462728   0.075653   6.1165  9.57e-10 ***
## addr_stateDE                                -0.014748   0.068081  -0.2166  8.28e-01
## addr_stateFL                                -0.062163   0.051091  -1.2167  2.24e-01
## addr_stateGA                                 0.131973   0.052247   2.5260  1.15e-02 *
## addr_stateHI                                 0.041368   0.060991   0.6783  4.98e-01
## addr_stateIA                                -0.577661   0.680965  -0.8483  3.96e-01
## addr_stateID                                 0.098402   0.087127   1.1294  2.59e-01
## addr_stateIL                                 0.111874   0.051953   2.1534  3.13e-02 *
## addr_stateIN                                -0.048944   0.053676  -0.9118  3.62e-01
## addr_stateKS                                 0.258245   0.057876   4.4620  8.12e-06 ***
## addr_stateKY                                -0.024841   0.055994  -0.4436  6.57e-01
## addr_stateLA                                -0.147344   0.054776  -2.6899  7.15e-03 **
## addr_stateMA                                 0.033515   0.052906   0.6335  5.26e-01
## addr_stateMD                                -0.063710   0.052701  -1.2089  2.27e-01
## addr_stateME                                 0.457986   0.088977   5.1473  2.64e-07 ***
## addr_stateMI                                 0.035013   0.052517   0.6667  5.05e-01
## addr_stateMN                                 0.046273   0.053612   0.8631  3.88e-01
## addr_stateMO                                -0.042303   0.053809  -0.7862  4.32e-01
## addr_stateMS                                -0.266313   0.059750  -4.4571  8.31e-06 ***
## addr_stateMT                                 0.253715   0.070534   3.5971  3.22e-04 ***
## addr_stateNC                                -0.012232   0.052329  -0.2337  8.15e-01
## addr_stateND                                 0.004955   0.086315   0.0574  9.54e-01
## addr_stateNE                                -0.217888   0.067005  -3.2518  1.15e-03 **
## addr_stateNH                                 0.401021   0.064363   6.2306  4.65e-10 ***
## addr_stateNJ                                -0.082524   0.051879  -1.5907  1.12e-01
## addr_stateNM                                -0.025392   0.059929  -0.4237  6.72e-01
## addr_stateNV                                -0.081066   0.053880  -1.5046  1.32e-01
## addr_stateNY                                -0.113834   0.050979  -2.2329  2.56e-02 *
## addr_stateOH                                -0.008856   0.052063  -0.1701  8.65e-01
```

```
## addr_stateOK                                    -0.165074   0.055871  -2.9546  3.13e-03 **
## addr_stateOR                                     0.449863   0.056254   7.9970  1.27e-15 ***
## addr_statePA                                    -0.033098   0.051984  -0.6367  5.24e-01
## addr_stateRI                                     0.152929   0.063483   2.4090  1.60e-02 *
## addr_stateSC                                     0.299216   0.055909   5.3519  8.71e-08 ***
## addr_stateSD                                    -0.031462   0.072900  -0.4316  6.66e-01
## addr_stateTN                                    -0.041325   0.053916  -0.7665  4.43e-01
## addr_stateTX                                     0.031023   0.051034   0.6079  5.43e-01
## addr_stateUT                                     0.249062   0.058592   4.2508  2.13e-05 ***
## addr_stateVA                                     0.027578   0.052379   0.5265  5.99e-01
## addr_stateVT                                     0.441671   0.080840   5.4635  4.67e-08 ***
## addr_stateWA                                     0.329987   0.053505   6.1674  6.94e-10 ***
## addr_stateWI                                     0.129925   0.054949   2.3645  1.81e-02 *
## addr_stateWV                                     0.331058   0.067281   4.9205  8.63e-07 ***
## addr_stateWY                                     0.225475   0.075061   3.0039  2.67e-03 **
## purposecredit_card                              -0.201001   0.027258  -7.3742  1.65e-13 ***
## purposedebt_consolidation                       -0.437401   0.026843 -16.2947  1.08e-59 ***
## purposeeducational                              -0.591799   0.144932  -4.0833  4.44e-05 ***
## purposehome_improvement                         -0.216494   0.028498  -7.5969  3.03e-14 ***
## purposehouse                                    -0.447089   0.042143 -10.6088  2.71e-26 ***
## purposemajor_purchase                           -0.277936   0.031649  -8.7818  1.61e-18 ***
## purposemedical                                  -0.474676   0.034547 -13.7401  5.84e-43 ***
## purposemoving                                   -0.520396   0.038427 -13.5426  8.76e-42 ***
## purposeother                                    -0.397706   0.028457 -13.9758  2.19e-44 ***
## purposerenewable_energy                         -0.558710   0.091336  -6.1171  9.53e-10 ***
## purposesmall_business                           -0.934878   0.033261 -28.1075 7.92e-174 ***
## purposevacation                                 -0.254516   0.040554  -6.2760  3.47e-10 ***
## purposewedding                                   0.077209   0.074694   1.0337  3.01e-01
## total_bc_limit20950<total_bc_limit<=24806        0.243912   0.011882  20.5279  1.21e-93 ***
## total_bc_limit4478<total_bc_limit<=9405          0.053960   0.008662   6.2292  4.69e-10 ***
## total_bc_limit34756<total_bc_limit<=47175        0.451143   0.011800  38.2337  0.00e+00 ***
## total_bc_limit13970<total_bc_limit<=18174        0.171821   0.010055  17.0875  1.84e-65 ***
## total_bc_limit9405<total_bc_limit<=11360         0.093380   0.011400   8.1909  2.59e-16 ***
## total_bc_limittotal_bc_limit=NA                  0.552768   0.015653  35.3129 3.72e-273 ***
## total_bc_limit29471<total_bc_limit<=34756        0.351202   0.013159  26.6885 6.40e-157 ***
## total_bc_limit47175<total_bc_limit<=1105500      0.658259   0.011510  57.1918  0.00e+00 ***
## total_bc_limit11360<total_bc_limit<=13970        0.128140   0.010867  11.7916  4.31e-32 ***
## total_bc_limit24806<total_bc_limit<=29471        0.307783   0.012432  24.7567 2.63e-135 ***
## total_bc_limit18174<total_bc_limit<=20950        0.207128   0.012358  16.7610  4.70e-63 ***
##
## ---------------------------------------------------------------------
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ---
## null df: 1045083; null deviance: 1048231;
## residuals df: 1045009; residuals deviance: 1036097;
## # obs.: 1045084; # non-zero weighted obs.: 1045084;
## AIC: 1036247; log Likelihood: -518048.5;
## RSS: 1044955; dispersion: 1; iterations: 4;
## rank: 75; max tolerance: 1.07e-12; convergence: TRUE.
```

```
## [1] 0
```

3.4.3.2 GLM on bins

```
##           used  (Mb) gc trigger   (Mb)  max used   (Mb)
## Ncells 2046580 109.3    3910885  208.9   3277428  175.1
```

```
## Vcells 6645410  50.8  338242090 2580.6 660022511 5035.6
```

```
## 11.301 2.609 13.929 0 0
```

```
## Generalized Linear Model of class 'speedglm':
##
## Call:  speedglm::speedglm(formula = isGoodLoan ~ ., data = loansData,      family = binomial())
##
## Coefficients:
##  ---------------------------------------------------------------------
##                                                 Estimate Std. Error  z value  Pr(>|z|)
## (Intercept)                                      0.89565    0.69377   1.2910  1.97e-01
## `(addr_state) addr_state=GA`                     0.70963    0.67925   1.0447  2.96e-01
## `(addr_state) addr_state=PA`                     0.54456    0.67923   0.8017  4.23e-01
## `(addr_state) addr_state=CA`                     0.62405    0.67913   0.9189  3.58e-01
## `(addr_state) addr_state=NC`                     0.56543    0.67926   0.8324  4.05e-01
## `(addr_state) addr_state=AZ`                     0.62755    0.67929   0.9238  3.56e-01
## `(addr_state) addr_state=MI`                     0.61267    0.67927   0.9020  3.67e-01
## `(addr_state) addr_state=OK`                     0.41259    0.67954   0.6072  5.44e-01
## `(addr_state) addr_state=AL`                     0.41766    0.67942   0.6147  5.39e-01
## `(addr_state) addr_state=MN`                     0.62393    0.67936   0.9184  3.58e-01
## `(addr_state) addr_state=TX`                     0.60868    0.67916   0.8962  3.70e-01
## `(addr_state) addr_state=AR`                     0.38624    0.67963   0.5683  5.70e-01
## `(addr_state) addr_state=VA`                     0.60524    0.67926   0.8910  3.73e-01
## `(addr_state) addr_state=CO`                     0.90345    0.67935   1.3299  1.84e-01
## `(addr_state) addr_state=FL`                     0.51550    0.67916   0.7590  4.48e-01
## `(addr_state) addr_state=IL`                     0.68954    0.67923   1.0152  3.10e-01
## `(addr_state) addr_state=UT`                     0.82672    0.67977   1.2162  2.24e-01
## `(addr_state) addr_state=NV`                     0.49659    0.67938   0.7310  4.65e-01
## `(addr_state) addr_state=NY`                     0.46383    0.67915   0.6829  4.95e-01
## `(addr_state) addr_state=SC`                     0.87688    0.67954   1.2904  1.97e-01
## `(addr_state) addr_state=NM`                     0.55227    0.67989   0.8123  4.17e-01
## `(addr_state) addr_state=MA`                     0.61118    0.67930   0.8997  3.68e-01
## `(addr_state) addr_state=CT`                     0.74398    0.67944   1.0950  2.74e-01
## `(addr_state) addr_state=NJ`                     0.49514    0.67922   0.7290  4.66e-01
## `(addr_state) addr_state=OR`                     1.02752    0.67957   1.5120  1.31e-01
## `(addr_state) addr_state=AK`                     0.57766    0.68097   0.8483  3.96e-01
## `(addr_state) addr_state=VT`                     1.01933    0.68205   1.4945  1.35e-01
## `(addr_state) addr_state=WA`                     0.90765    0.67935   1.3361  1.82e-01
## `(addr_state) addr_state=MD`                     0.51395    0.67929   0.7566  4.49e-01
## `(addr_state) addr_state=DE`                     0.56291    0.68065   0.8270  4.08e-01
## `(addr_state) addr_state=MO`                     0.53536    0.67937   0.7880  4.31e-01
## `(addr_state) addr_state=KY`                     0.55282    0.67955   0.8135  4.16e-01
## `(addr_state) addr_state=WI`                     0.70759    0.67947   1.0414  2.98e-01
## `(addr_state) addr_state=OH`                     0.56881    0.67924   0.8374  4.02e-01
## `(addr_state) addr_state=NE`                     0.35977    0.68055   0.5286  5.97e-01
## `(addr_state) addr_state=MS`                     0.31135    0.67988   0.4579  6.47e-01
## `(addr_state) addr_state=TN`                     0.53634    0.67939   0.7894  4.30e-01
## `(addr_state) addr_state=WV`                     0.90872    0.68057   1.3352  1.82e-01
## `(addr_state) addr_state=RI`                     0.73059    0.68021   1.0741  2.83e-01
## `(addr_state) addr_state=HI`                     0.61903    0.67998   0.9104  3.63e-01
## `(addr_state) addr_state=KS`                     0.83591    0.67971   1.2298  2.19e-01
## `(addr_state) addr_state=WY`                     0.80314    0.68139   1.1787  2.39e-01
## `(addr_state) addr_state=IN`                     0.52872    0.67937   0.7782  4.36e-01
## `(addr_state) addr_state=MT`                     0.83138    0.68090   1.2210  2.22e-01
## `(addr_state) addr_state=LA`                     0.43032    0.67945   0.6333  5.27e-01
## `(addr_state) addr_state=NH`                     0.97868    0.68029   1.4386  1.50e-01
## `(addr_state) addr_state=ID`                     0.67606    0.68283   0.9901  3.22e-01
## `(addr_state) addr_state=DC`                     1.04039    0.68145   1.5267  1.27e-01
```

```
## `(addr_state) addr_state=ME`                        1.03565    0.68307   1.5162  1.29e-01
## `(addr_state) addr_state=ND`                        0.58262    0.68272   0.8534  3.93e-01
## `(addr_state) addr_state=SD`                        0.54620    0.68115   0.8019  4.23e-01
## `(addr_state) addr_state=IA`                             NA         NA       NA        NA
## `(purpose) purpose=debt_consolidation`              0.15440    0.14260   1.0827  2.79e-01
## `(purpose) purpose=car`                             0.59180    0.14493   4.0833  4.44e-05 ***
## `(purpose) purpose=credit_card`                     0.39080    0.14269   2.7389  6.17e-03 **
## `(purpose) purpose=house`                           0.14471    0.14621   0.9897  3.22e-01
## `(purpose) purpose=home_improvement`                0.37531    0.14291   2.6261  8.64e-03 **
## `(purpose) purpose=other`                           0.19409    0.14288   1.3584  1.74e-01
## `(purpose) purpose=major_purchase`                  0.31386    0.14353   2.1868  2.88e-02 *
## `(purpose) purpose=medical`                         0.11712    0.14423   0.8121  4.17e-01
## `(purpose) purpose=small_business`                 -0.34308    0.14381  -2.3856  1.70e-02 *
## `(purpose) purpose=wedding`                         0.66901    0.15831   4.2259  2.38e-05 ***
## `(purpose) purpose=moving`                          0.07140    0.14519   0.4918  6.23e-01
## `(purpose) purpose=vacation`                        0.33728    0.14579   2.3134  2.07e-02 *
## `(purpose) purpose=renewable_energy`                0.03309    0.16712   0.1980  8.43e-01
## `(purpose) purpose=educational`                          NA         NA       NA        NA
## `(total_bc_limit) -Inf<total_bc_limit<=4478`       -0.55277    0.01565 -35.3129 3.72e-273 ***
## `(total_bc_limit) 4478<total_bc_limit<=9405`       -0.49881    0.01512 -32.9870 1.25e-238 ***
## `(total_bc_limit) 9405<total_bc_limit<=11360`      -0.45939    0.01684 -27.2866 6.11e-164 ***
## `(total_bc_limit) 11360<total_bc_limit<=13970`     -0.42463    0.01648 -25.7725 1.80e-146 ***
## `(total_bc_limit) 13970<total_bc_limit<=18174`     -0.38095    0.01595 -23.8805 4.88e-126 ***
## `(total_bc_limit) 18174<total_bc_limit<=20950`     -0.34564    0.01749 -19.7590  6.71e-87 ***
## `(total_bc_limit) 20950<total_bc_limit<=24806`     -0.30886    0.01716 -17.9998  1.95e-72 ***
## `(total_bc_limit) 24806<total_bc_limit<=29471`     -0.24498    0.01754 -13.9655  2.53e-44 ***
## `(total_bc_limit) 29471<total_bc_limit<=34756`     -0.20157    0.01806 -11.1604  6.37e-29 ***
## `(total_bc_limit) 34756<total_bc_limit<=47175`     -0.10162    0.01709  -5.9457  2.75e-09 ***
## `(total_bc_limit) 47175<total_bc_limit<=1105500`    0.10549    0.01688   6.2481  4.16e-10 ***
## `(total_bc_limit) total_bc_limit=NA`                     NA         NA       NA        NA
##
## ---------------------------------------------------------------------
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ---
## null df: 1045083; null deviance: 1048231;
## residuals df: 1045009; residuals deviance: 1036097;
## # obs.: 1045084; # non-zero weighted obs.: 1045084;
## AIC: 1036247; log Likelihood: -518048.5;
## RSS: 1044955; dispersion: 1; iterations: 4;
## rank: 75; max tolerance: 1.07e-12; convergence: TRUE.
```

```
## [1] 3
```

3.4.4 Second training

```
##            used  (Mb) gc trigger   (Mb)  max used   (Mb)
## Ncells  2047743 109.4    6052487  323.3   3277428  175.1
## Vcells 88173456 672.8  479835922 3660.9 660022511 5035.6
```

```
## 8.936 2.059 10.998 0 0
```

```
## Generalized Linear Model of class 'speedglm':
##
## Call:  speedglm::speedglm(formula = isGoodLoan ~ ., data = loansData,      family = binomial())
##
## Coefficients:
##  ---------------------------------------------------------------------
```

```
##                                                  Estimate Std. Error  z value  Pr(>|z|)
## (Intercept)                                       0.89565    0.69377   1.2910  1.97e-01
## `(addr_state) addr_state=GA`                      0.70963    0.67925   1.0447  2.96e-01
## `(addr_state) addr_state=PA`                      0.54456    0.67923   0.8017  4.23e-01
## `(addr_state) addr_state=CA`                      0.62405    0.67913   0.9189  3.58e-01
## `(addr_state) addr_state=NC`                      0.56543    0.67926   0.8324  4.05e-01
## `(addr_state) addr_state=AZ`                      0.62755    0.67929   0.9238  3.56e-01
## `(addr_state) addr_state=MI`                      0.61267    0.67927   0.9020  3.67e-01
## `(addr_state) addr_state=OK`                      0.41259    0.67954   0.6072  5.44e-01
## `(addr_state) addr_state=AL`                      0.41766    0.67942   0.6147  5.39e-01
## `(addr_state) addr_state=MN`                      0.62393    0.67936   0.9184  3.58e-01
## `(addr_state) addr_state=TX`                      0.60868    0.67916   0.8962  3.70e-01
## `(addr_state) addr_state=AR`                      0.38624    0.67963   0.5683  5.70e-01
## `(addr_state) addr_state=VA`                      0.60524    0.67926   0.8910  3.73e-01
## `(addr_state) addr_state=CO`                      0.90345    0.67935   1.3299  1.84e-01
## `(addr_state) addr_state=FL`                      0.51550    0.67916   0.7590  4.48e-01
## `(addr_state) addr_state=IL`                      0.68954    0.67923   1.0152  3.10e-01
## `(addr_state) addr_state=UT`                      0.82672    0.67977   1.2162  2.24e-01
## `(addr_state) addr_state=NV`                      0.49659    0.67938   0.7310  4.65e-01
## `(addr_state) addr_state=NY`                      0.46383    0.67915   0.6829  4.95e-01
## `(addr_state) addr_state=SC`                      0.87688    0.67954   1.2904  1.97e-01
## `(addr_state) addr_state=NM`                      0.55227    0.67989   0.8123  4.17e-01
## `(addr_state) addr_state=MA`                      0.61118    0.67930   0.8997  3.68e-01
## `(addr_state) addr_state=CT`                      0.74398    0.67944   1.0950  2.74e-01
## `(addr_state) addr_state=NJ`                      0.49514    0.67922   0.7290  4.66e-01
## `(addr_state) addr_state=OR`                      1.02752    0.67957   1.5120  1.31e-01
## `(addr_state) addr_state=AK`                      0.57766    0.68097   0.8483  3.96e-01
## `(addr_state) addr_state=VT`                      1.01933    0.68205   1.4945  1.35e-01
## `(addr_state) addr_state=WA`                      0.90765    0.67935   1.3361  1.82e-01
## `(addr_state) addr_state=MD`                      0.51395    0.67929   0.7566  4.49e-01
## `(addr_state) addr_state=DE`                      0.56291    0.68065   0.8270  4.08e-01
## `(addr_state) addr_state=MO`                      0.53536    0.67937   0.7880  4.31e-01
## `(addr_state) addr_state=KY`                      0.55282    0.67955   0.8135  4.16e-01
## `(addr_state) addr_state=WI`                      0.70759    0.67947   1.0414  2.98e-01
## `(addr_state) addr_state=OH`                      0.56881    0.67924   0.8374  4.02e-01
## `(addr_state) addr_state=NE`                      0.35977    0.68055   0.5286  5.97e-01
## `(addr_state) addr_state=MS`                      0.31135    0.67988   0.4579  6.47e-01
## `(addr_state) addr_state=TN`                      0.53634    0.67939   0.7894  4.30e-01
## `(addr_state) addr_state=WV`                      0.90872    0.68057   1.3352  1.82e-01
## `(addr_state) addr_state=RI`                      0.73059    0.68021   1.0741  2.83e-01
## `(addr_state) addr_state=HI`                      0.61903    0.67998   0.9104  3.63e-01
## `(addr_state) addr_state=KS`                      0.83591    0.67971   1.2298  2.19e-01
## `(addr_state) addr_state=WY`                      0.80314    0.68139   1.1787  2.39e-01
## `(addr_state) addr_state=IN`                      0.52872    0.67937   0.7782  4.36e-01
## `(addr_state) addr_state=MT`                      0.83138    0.68090   1.2210  2.22e-01
## `(addr_state) addr_state=LA`                      0.43032    0.67945   0.6333  5.27e-01
## `(addr_state) addr_state=NH`                      0.97868    0.68029   1.4386  1.50e-01
## `(addr_state) addr_state=ID`                      0.67606    0.68283   0.9901  3.22e-01
## `(addr_state) addr_state=DC`                      1.04039    0.68145   1.5267  1.27e-01
## `(addr_state) addr_state=ME`                      1.03565    0.68307   1.5162  1.29e-01
## `(addr_state) addr_state=ND`                      0.58262    0.68272   0.8534  3.93e-01
## `(addr_state) addr_state=SD`                      0.54620    0.68115   0.8019  4.23e-01
## `(purpose) purpose=debt_consolidation`            0.15440    0.14260   1.0827  2.79e-01
## `(purpose) purpose=car`                           0.59180    0.14493   4.0833  4.44e-05 ***
## `(purpose) purpose=credit_card`                   0.39080    0.14269   2.7389  6.17e-03 **
## `(purpose) purpose=house`                         0.14471    0.14621   0.9897  3.22e-01
## `(purpose) purpose=home_improvement`              0.37531    0.14291   2.6261  8.64e-03 **
## `(purpose) purpose=other`                         0.19409    0.14288   1.3584  1.74e-01
```

```
## `(purpose) purpose=major_purchase`                0.31386    0.14353   2.1868  2.88e-02 *
## `(purpose) purpose=medical`                       0.11712    0.14423   0.8121  4.17e-01
## `(purpose) purpose=small_business`               -0.34308    0.14381  -2.3856  1.70e-02 *
## `(purpose) purpose=wedding`                       0.66901    0.15831   4.2259  2.38e-05 ***
## `(purpose) purpose=moving`                        0.07140    0.14519   0.4918  6.23e-01
## `(purpose) purpose=vacation`                      0.33728    0.14579   2.3134  2.07e-02 *
## `(purpose) purpose=renewable_energy`              0.03309    0.16712   0.1980  8.43e-01
## `(total_bc_limit) -Inf<total_bc_limit<=4478`     -0.55277    0.01565 -35.3129 3.72e-273 ***
## `(total_bc_limit) 4478<total_bc_limit<=9405`     -0.49881    0.01512 -32.9870 1.25e-238 ***
## `(total_bc_limit) 9405<total_bc_limit<=11360`    -0.45939    0.01684 -27.2866 6.11e-164 ***
## `(total_bc_limit) 11360<total_bc_limit<=13970`   -0.42463    0.01648 -25.7725 1.80e-146 ***
## `(total_bc_limit) 13970<total_bc_limit<=18174`   -0.38095    0.01595 -23.8805 4.88e-126 ***
## `(total_bc_limit) 18174<total_bc_limit<=20950`   -0.34564    0.01749 -19.7590  6.71e-87 ***
## `(total_bc_limit) 20950<total_bc_limit<=24806`   -0.30886    0.01716 -17.9998  1.95e-72 ***
## `(total_bc_limit) 24806<total_bc_limit<=29471`   -0.24498    0.01754 -13.9655  2.53e-44 ***
## `(total_bc_limit) 29471<total_bc_limit<=34756`   -0.20157    0.01806 -11.1604  6.37e-29 ***
## `(total_bc_limit) 34756<total_bc_limit<=47175`   -0.10162    0.01709  -5.9457  2.75e-09 ***
## `(total_bc_limit) 47175<total_bc_limit<=1105500`  0.10549    0.01688   6.2481  4.16e-10 ***
##
## ---------------------------------------------------------------------
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ---
## null df: 1045083; null deviance: 1048231;
## residuals df: 1045009; residuals deviance: 1036097;
## # obs.: 1045084; # non-zero weighted obs.: 1045084;
## AIC: 1036247; log Likelihood: -518048.5;
## RSS: 1044955; dispersion: 1; iterations: 4;
## rank: 75; max tolerance: 1.07e-12; convergence: TRUE.
```

3.4.5 Third training

```
##            used  (Mb) gc trigger   (Mb)  max used   (Mb)
## Ncells  2048471 109.5    6052487  323.3   3277428  175.1
## Vcells 88183951 672.8  460706485 3515.0 660022511 5035.6
```

```
## 3.207 0.067 3.278 0 0
```

```
## Generalized Linear Model of class 'speedglm':
##
## Call:  speedglm::speedglm(formula = isGoodLoan ~ ., data = loansData,      family = binomial())
##
## Coefficients:
##  ---------------------------------------------------------------------
##                                                  Estimate Std. Error z value  Pr(>|z|)
## (Intercept)                                        1.6547   0.014134 117.072  0.00e+00 ***
## `(purpose) purpose=car`                            0.4364   0.026787  16.293  1.11e-59 ***
## `(purpose) purpose=credit_card`                    0.2357   0.006346  37.147 4.95e-302 ***
## `(purpose) purpose=home_improvement`               0.2156   0.010500  20.534  1.07e-93 ***
## `(purpose) purpose=major_purchase`                 0.1517   0.017336   8.752  2.09e-18 ***
## `(purpose) purpose=small_business`                -0.4975   0.020156 -24.684 1.59e-134 ***
## `(purpose) purpose=wedding`                        0.4997   0.069864   7.153  8.49e-13 ***
## `(purpose) purpose=vacation`                       0.1745   0.030677   5.689  1.28e-08 ***
## `(total_bc_limit) -Inf<total_bc_limit<=4478`      -0.5522   0.015543 -35.525 2.01e-276 ***
## `(total_bc_limit) 4478<total_bc_limit<=9405`      -0.4994   0.015009 -33.275 8.82e-243 ***
## `(total_bc_limit) 9405<total_bc_limit<=11360`     -0.4602   0.016730 -27.511 1.32e-166 ***
## `(total_bc_limit) 11360<total_bc_limit<=13970`    -0.4251   0.016369 -25.970 1.09e-148 ***
## `(total_bc_limit) 13970<total_bc_limit<=18174`    -0.3828   0.015843 -24.160 5.94e-129 ***
```

```
## `(total_bc_limit) 18174<total_bc_limit<=20950`    -0.3462   0.017390 -19.910  3.31e-88 ***
## `(total_bc_limit) 20950<total_bc_limit<=24806`    -0.3085   0.017054 -18.088  3.99e-73 ***
## `(total_bc_limit) 24806<total_bc_limit<=29471`    -0.2445   0.017440 -14.017  1.23e-44 ***
## `(total_bc_limit) 29471<total_bc_limit<=34756`    -0.2010   0.017961 -11.192  4.48e-29 ***
## `(total_bc_limit) 34756<total_bc_limit<=47175`    -0.1005   0.016990  -5.914  3.33e-09 ***
## `(total_bc_limit) 47175<total_bc_limit<=1105500`   0.1055   0.016782   6.289  3.20e-10 ***
##
## ---------------------------------------------------------------------
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ---
## null df: 1045083; null deviance: 1048231;
## residuals df: 1045065; residuals deviance: 1038773;
## # obs.: 1045084; # non-zero weighted obs.: 1045084;
## AIC: 1038811; log Likelihood: -519386.5;
## RSS: 1045033; dispersion: 1; iterations: 4;
## rank: 19; max tolerance: 7.97e-13; convergence: TRUE.
```

3.5 Model result

3.5.1 Final list of variables

3.5.2 Scoring

Scoring expresses the coefficients that were estimated during the logistic regression into points on a scale. The conversion is done using three parameters that are chosen somewhat arbitrarily.

- the number of points increase / decrease that would reflect halving / doubling the odds of defaulting;
- an *anchoring* score reflecting a particular odd.

For our purpose, we will choose ($Score_{anchor}$) 5,000 points being equivalent to 1 in a 20 to default ($Odds_{anchor}$), i.e. 5,000 points <=> $Odds_{Anchor} = \frac{1/20}{1-1/20}$. We will also choose 100 to reflect *times 2* change in odds ($DoubleOdds$). Those choices are completely arbitrary. Basically, the score is a linear representation of the odds. The score is defined by a point (the anchor) and the slope of the line going through that point.

Those values will reflect the total estimated score for a borrower (i.e. loan sample). The number of characteristics (information points gathered in a credit application), or number of bins, have to be irrelevant in calculating this score. In other words, if LendingClub were to gather 5 more information points, the score should, **mutatis mutandis**, be unchanged. (However, we would hope that the quality of the estimated score would improve.)

The score per variable needs to be adjusted using the number of information points, that is the number of characteristics. Here, the model has been trained on the number of bins. We first need to determine how many characteristics are used in the model.

```
## [1] 3
```

We can now perform the scoring calculation.

Then we apportion across characteristics.

VERY IMPORTANT:

The intercept points have been allocated across all characteristics. Therefore the regression coefficient estimated for the intercept becomes redundant and needs removing.

Using the model is a simple matrix multiplication: Loan matrix × Scorecard weights

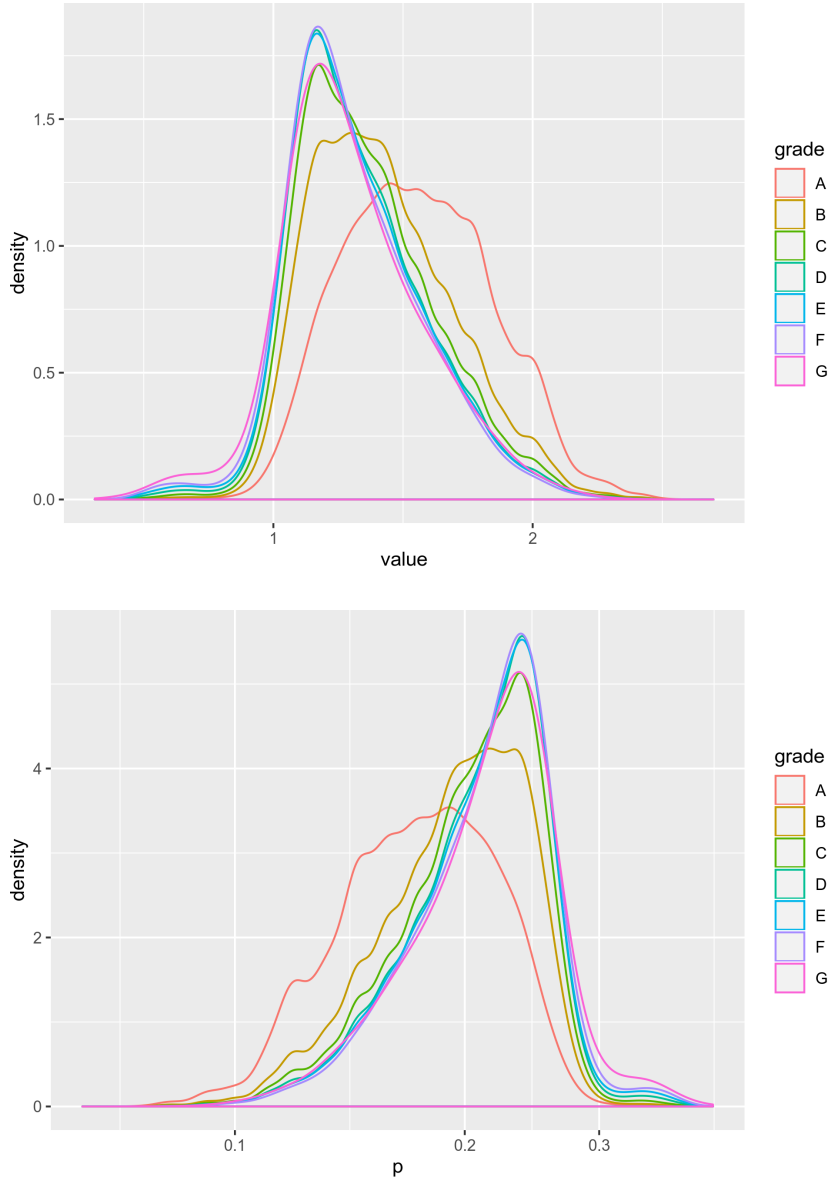
3.6 Training set

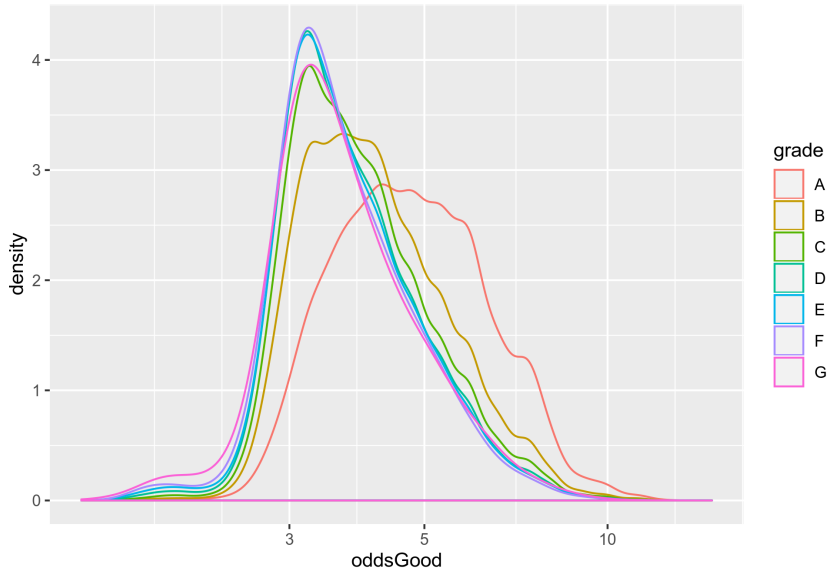
```
##             used   (Mb) gc trigger   (Mb)  max used   (Mb)
## Ncells   2029310  108.4    4142207  221.3   2570864  137.3
## Vcells 151622863 1156.8  422802665 3225.8 660022580 5035.6
```

```
## [1] 1045084      75
```

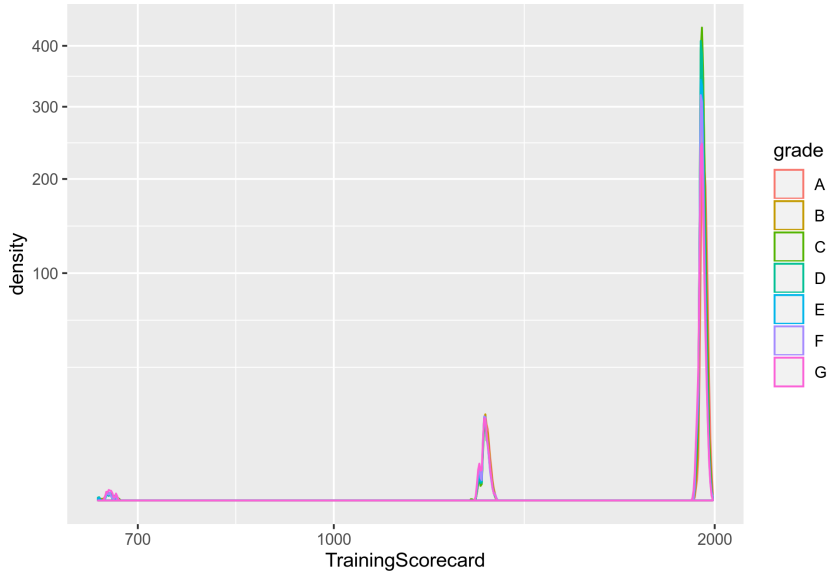
```
##             used  (Mb) gc trigger   (Mb)  max used   (Mb)
## Ncells   2009202 107.4    4085841  218.3   2569072  137.3
## Vcells 82341313 628.3  422802732 3225.8 660022667 5035.6
```

3.6.1 Histogram of the scores





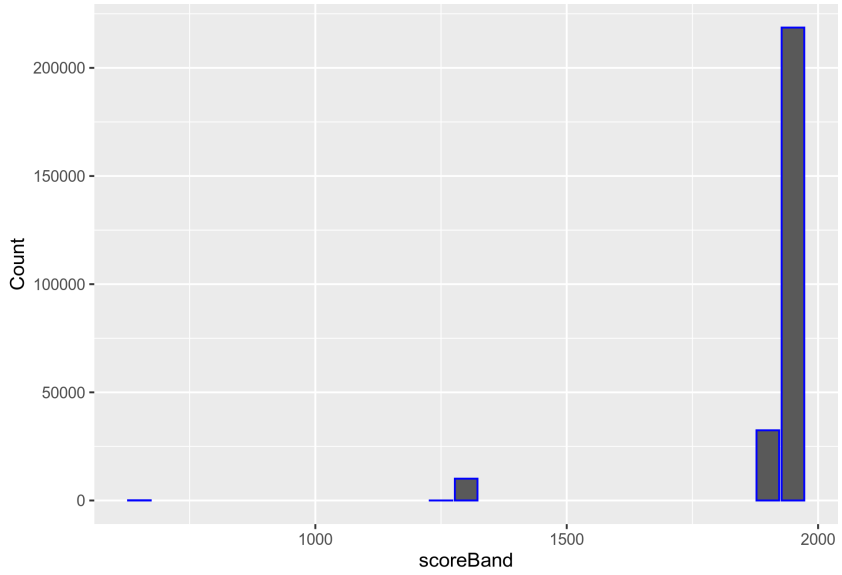
The following histogram shows the density plot of the scorecards. Note the log-scale for the score, and the square-root scale for the density. It mirrors the multimodal density distributions of previous visualisations.



3.7 Test set

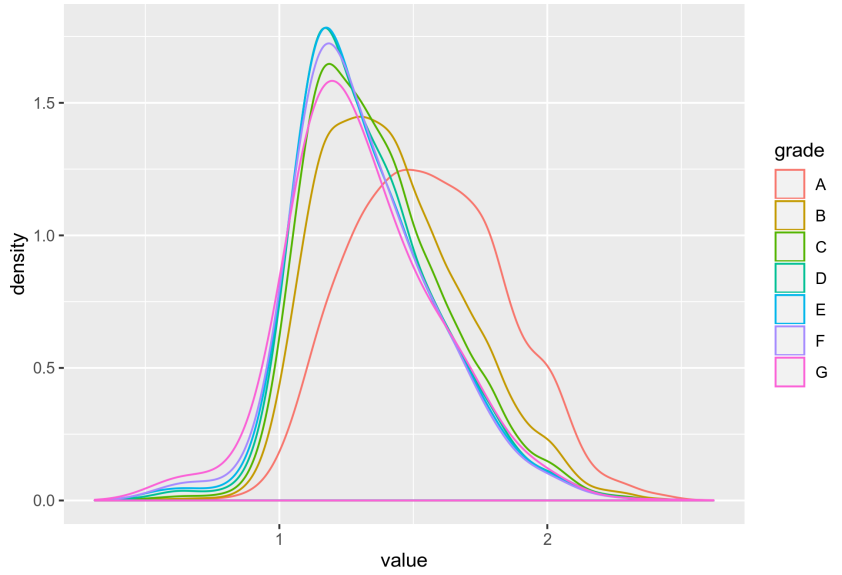
```
## [1] 261272    423
```

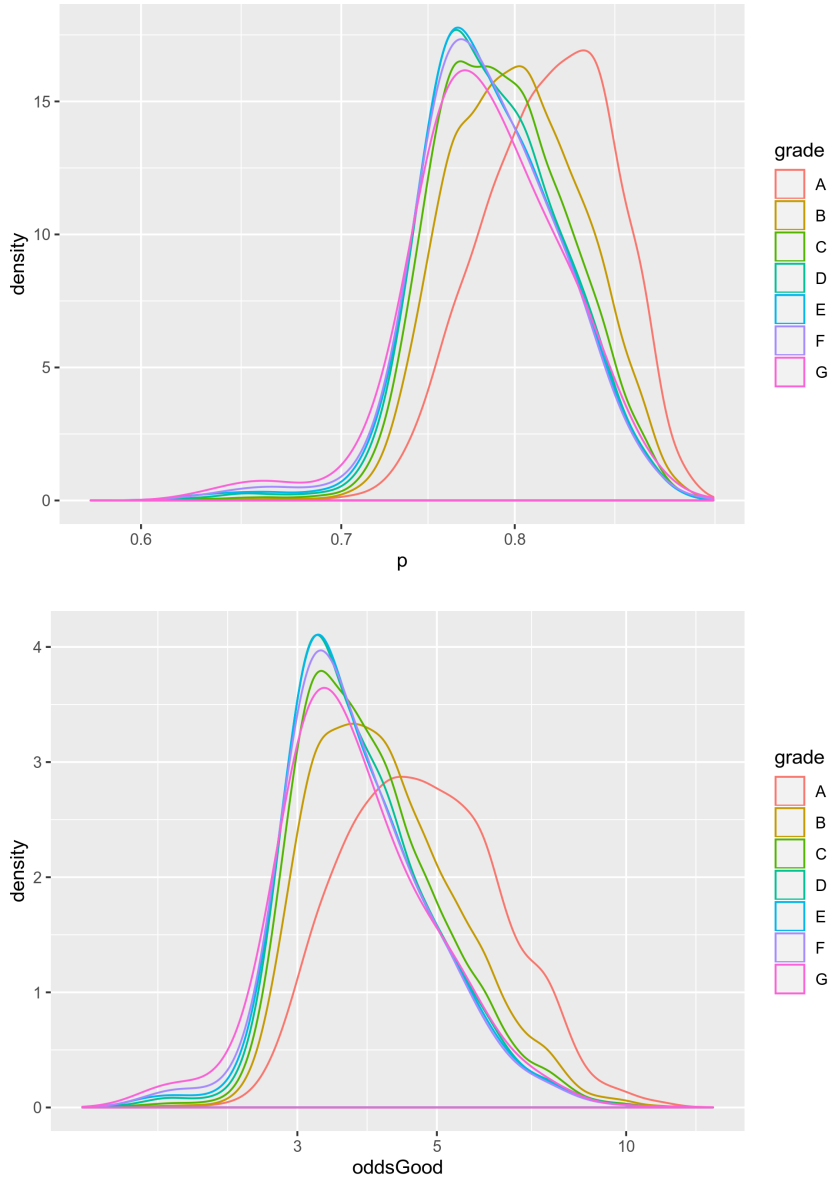
```
## [1] 261272     75
```




```
## [1] 1956
```

Same downward dynamics as training set

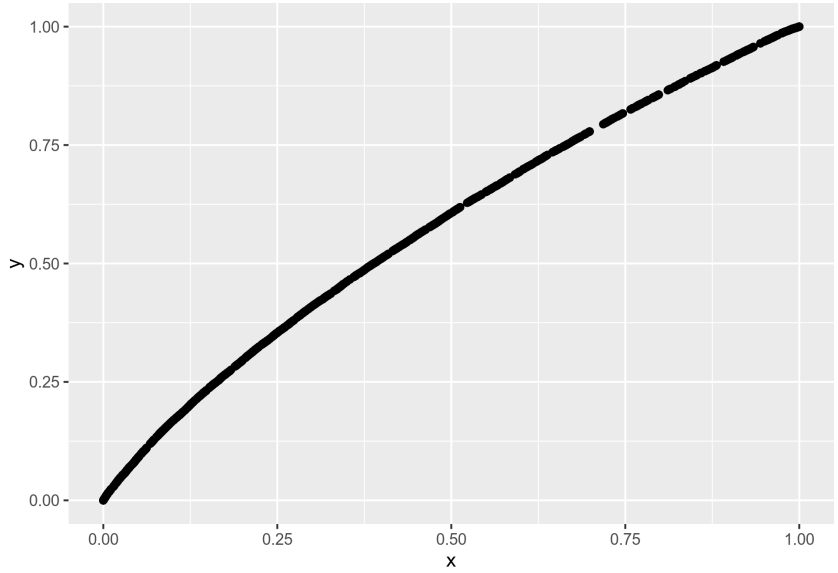




3.8 Correlation matrix

[TODO]

3.9 ROC Curve




```
##
##  Two-sample Kolmogorov-Smirnov test
##
## data:  ROCRPlot$x and ROCRPlot$y
## D = 0.12134, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

```
## Formal class 'performance' [package "ROCR"] with 6 slots
##   ..@ x.name      : chr "False positive rate"
##   ..@ y.name      : chr "True positive rate"
##   ..@ alpha.name  : chr "Cutoff"
##   ..@ x.values    :List of 1
##   .. ..$ : num [1:5637] 0 0 0 0 0 0 0 0 0 0 ...
##   ..@ y.values    :List of 1
##   .. ..$ : num [1:5637] 0.00 4.79e-06 1.44e-05 1.92e-05 2.40e-05 ...
##   ..@ alpha.values:List of 1
##   .. ..$ : num [1:5637] Inf 2.62 2.61 2.59 2.58 ...
```

```
## [[1]]
##    [1] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##   [10] 0.000000e+00 1.904000e-05 1.904000e-05 1.904000e-05 3.808001e-05 7.616001e-05 9.520002e-05
##   [19] 1.332800e-04 1.523200e-04 1.523200e-04 1.713600e-04 1.713600e-04 2.094400e-04 2.094400e-04
##   [28] 2.094400e-04 2.284800e-04 2.284800e-04 2.665600e-04 2.665600e-04 2.665600e-04 2.665600e-04
##   [37] 3.236801e-04 3.427201e-04 3.427201e-04 4.569601e-04 4.569601e-04 4.569601e-04 4.569601e-04
##   [46] 4.760001e-04 4.950401e-04 4.950401e-04 5.331201e-04 5.331201e-04 5.331201e-04 5.331201e-04
##   [55] 6.092801e-04 6.092801e-04 6.092801e-04 6.473601e-04 6.473601e-04 6.473601e-04 9.329601e-04
##   [64] 1.237600e-03 1.237600e-03 1.237600e-03 1.237600e-03 1.275680e-03 1.275680e-03 1.351840e-03
##   [73] 1.428000e-03 1.447040e-03 1.466080e-03 1.466080e-03 1.466080e-03 1.466080e-03 1.656480e-03
##   [82] 1.770720e-03 1.770720e-03 1.770720e-03 1.770720e-03 1.770720e-03 1.770720e-03 1.770720e-03
##   [91] 1.808800e-03 1.904000e-03 1.904000e-03 1.904000e-03 1.942080e-03 1.942080e-03 1.942080e-03
##  [100] 1.961120e-03 2.056320e-03 2.113440e-03 2.113440e-03 2.151520e-03 2.151520e-03 2.151520e-03
##  [109] 2.189600e-03 2.360960e-03 2.360960e-03 2.437120e-03 2.437120e-03 2.437120e-03 2.456160e-03
##  [118] 2.494240e-03 2.494240e-03 2.494240e-03 2.494240e-03 2.589440e-03 2.589440e-03 2.627520e-03
##  [127] 2.665600e-03 2.665600e-03 2.760800e-03 2.760800e-03 2.760800e-03 2.779840e-03 2.836960e-03
##  [136] 2.894080e-03 3.027360e-03 3.046400e-03 3.065440e-03 3.065440e-03 3.065440e-03 3.236801e-03
##  [145] 3.255841e-03 3.274881e-03 3.293921e-03 3.312961e-03 3.351041e-03 3.389121e-03 3.408161e-03
##  [154] 3.427201e-03 3.427201e-03 3.427201e-03 3.446241e-03 3.484321e-03 3.484321e-03 3.503361e-03
##  [163] 3.522401e-03 3.522401e-03 3.522401e-03 3.522401e-03 3.522401e-03 3.522401e-03 3.560481e-03
```

```
##   [172] 3.598561e-03 3.598561e-03 3.598561e-03 3.598561e-03 3.598561e-03 3.598561e-03 3.598561e-03
##   [181] 3.598561e-03 3.598561e-03 3.617601e-03 3.884161e-03 3.884161e-03 3.998401e-03 3.998401e-03
##   [190] 4.017441e-03 4.036481e-03 4.036481e-03 4.036481e-03 4.036481e-03 4.036481e-03 4.036481e-03
##   [199] 4.169761e-03 4.169761e-03 4.341121e-03 4.341121e-03 4.341121e-03 4.341121e-03 4.341121e-03
##   [208] 4.341121e-03 4.341121e-03 4.341121e-03 4.341121e-03 4.379201e-03 4.379201e-03 4.512481e-03
##   [217] 4.569601e-03 4.569601e-03 4.588641e-03 4.607681e-03 4.626721e-03 4.683841e-03 4.683841e-03
##   [226] 4.740961e-03 4.798081e-03 4.798081e-03 5.216961e-03 5.236001e-03 5.236001e-03 5.236001e-03
##   [235] 5.483521e-03 5.502561e-03 5.502561e-03 5.521601e-03 5.521601e-03 5.597761e-03 5.616801e-03
##   [244] 5.826241e-03 5.864321e-03 5.864321e-03 5.864321e-03 5.883361e-03 5.883361e-03 5.883361e-03
##   [253] 6.188001e-03 6.188001e-03 6.188001e-03 6.207041e-03 6.321281e-03 6.321281e-03 6.340321e-03
##   [262] 6.340321e-03 6.359361e-03 7.159041e-03 7.159041e-03 7.197121e-03 7.216161e-03 7.216161e-03
##   [271] 7.501761e-03 7.616001e-03 7.616001e-03 7.616001e-03 7.635041e-03 7.673121e-03 7.711201e-03
##   [280] 7.730241e-03 7.882561e-03 7.882561e-03 7.882561e-03 7.901601e-03 7.939681e-03 7.958721e-03
##   [289] 8.187201e-03 8.187201e-03 8.206241e-03 8.815521e-03 8.815521e-03 8.948801e-03 8.948801e-03
##   [298] 8.967841e-03 8.986881e-03 9.672322e-03 9.729442e-03 9.729442e-03 9.767522e-03 9.786562e-03
##   [307] 9.824642e-03 9.824642e-03 9.824642e-03 9.824642e-03 9.843682e-03 9.843682e-03 9.843682e-03
##   [316] 9.843682e-03 9.843682e-03 9.862722e-03 9.862722e-03 9.862722e-03 9.862722e-03 9.862722e-03
##   [325] 9.881762e-03 9.881762e-03 9.900802e-03 9.900802e-03 9.900802e-03 9.900802e-03 9.900802e-03
##   [334] 1.028160e-02 1.030064e-02 1.037680e-02 1.043392e-02 1.045296e-02 1.047200e-02 1.047200e-02
##   [343] 1.051008e-02 1.051008e-02 1.051008e-02 1.051008e-02 1.054816e-02 1.054816e-02 1.056720e-02
##   [352] 1.064336e-02 1.066240e-02 1.106224e-02 1.106224e-02 1.110032e-02 1.110032e-02 1.110032e-02
##   [361] 1.115744e-02 1.399440e-02 1.443232e-02 1.445136e-02 1.447040e-02 1.448944e-02 1.448944e-02
##   [370] 1.460368e-02 1.460368e-02 1.460368e-02 1.466080e-02 1.466080e-02 1.475600e-02 1.475600e-02
##   [379] 1.481312e-02 1.481312e-02 1.481312e-02 1.481312e-02 1.481312e-02 1.481312e-02 1.481312e-02
##   [388] 1.494640e-02 1.496544e-02 1.496544e-02 1.500352e-02 1.534624e-02 1.538432e-02 1.547952e-02
##   [397] 1.591744e-02 1.591744e-02 1.591744e-02 1.591744e-02 1.726928e-02 1.784048e-02 1.785952e-02
##   [406] 1.787856e-02 1.787856e-02 1.793568e-02 1.848784e-02 1.848784e-02 1.852592e-02 1.860208e-02
##   [415] 1.865920e-02 1.865920e-02 1.865920e-02 1.865920e-02 1.867824e-02 1.877344e-02 1.881152e-02
##   [424] 1.911616e-02 1.911616e-02 1.934464e-02 1.938272e-02 1.938272e-02 1.945888e-02 1.947792e-02
##   [433] 1.972544e-02 1.972544e-02 1.978256e-02 1.978256e-02 2.027760e-02 2.029664e-02 2.031568e-02
##   [442] 2.035376e-02 2.037280e-02 2.060128e-02 2.067744e-02 2.067744e-02 2.067744e-02 2.079168e-02
##   [451] 2.081072e-02 2.081072e-02 2.081072e-02 2.081072e-02 2.081072e-02 2.081072e-02 2.081072e-02
##   [460] 2.111536e-02 2.111536e-02 2.113440e-02 2.115344e-02 2.115344e-02 2.115344e-02 2.115344e-02
##   [469] 2.119152e-02 2.119152e-02 2.119152e-02 2.119152e-02 2.119152e-02 2.128672e-02 2.130576e-02
##   [478] 2.132480e-02 2.132480e-02 2.132480e-02 2.132480e-02 2.132480e-02 2.138192e-02 2.138192e-02
##   [487] 2.151520e-02 2.201024e-02 2.204832e-02 2.204832e-02 2.208640e-02 2.212448e-02 2.212448e-02
##   [496] 2.298128e-02 2.303840e-02 2.303840e-02 2.307648e-02 2.309552e-02 2.313360e-02 2.313360e-02
##   [505] 2.341920e-02 2.343824e-02 2.347632e-02 2.351440e-02 2.351440e-02 2.357152e-02 2.357152e-02
##   [514] 2.370480e-02 2.370480e-02 2.372384e-02 2.374288e-02 2.376192e-02 2.378096e-02 2.389520e-02
##   [523] 2.421888e-02 2.429504e-02 2.429504e-02 2.442832e-02 2.442832e-02 2.442832e-02 2.442832e-02
##   [532] 2.442832e-02 2.444736e-02 2.448544e-02 2.458064e-02 2.458064e-02 2.458064e-02 2.458064e-02
##   [541] 2.461872e-02 2.467584e-02 2.492336e-02 2.492336e-02 2.492336e-02 2.492336e-02 2.492336e-02
##   [550] 2.549456e-02 2.553264e-02 2.555168e-02 2.555168e-02 2.555168e-02 2.562784e-02 2.564688e-02
##   [559] 2.583728e-02 2.583728e-02 2.583728e-02 2.583728e-02 2.585632e-02 2.589440e-02 2.591344e-02
##   [568] 2.619904e-02 2.637040e-02 2.637040e-02 2.640848e-02 2.671312e-02 2.680832e-02 2.705584e-02
##   [577] 2.705584e-02 2.711296e-02 2.713200e-02 2.718912e-02 2.718912e-02 2.718912e-02 2.718912e-02
##   [586] 2.718912e-02 2.718912e-02 2.720816e-02 2.737952e-02 2.747472e-02 2.768416e-02 2.770320e-02
##   [595] 2.776032e-02 2.777936e-02 2.795072e-02 2.796976e-02 2.800784e-02 2.800784e-02 2.800784e-02
##   [604] 2.804592e-02 2.806496e-02 2.810304e-02 2.812208e-02 2.819824e-02 2.819824e-02 2.819824e-02
##   [613] 2.829344e-02 2.829344e-02 2.836960e-02 2.846480e-02 2.856000e-02 2.869328e-02 2.873136e-02
##   [622] 2.964528e-02 2.968336e-02 2.968336e-02 2.968336e-02 2.968336e-02 3.086384e-02 3.086384e-02
##   [631] 3.118752e-02 3.124464e-02 3.124464e-02 3.126369e-02 3.128273e-02 3.130177e-02 3.132081e-02
##   [640] 3.132081e-02 3.135889e-02 3.137793e-02 3.139697e-02 3.139697e-02 3.143505e-02 3.143505e-02
##   [649] 3.208241e-02 3.210145e-02 3.212049e-02 3.212049e-02 3.212049e-02 3.213953e-02 3.223473e-02
##   [658] 3.227281e-02 3.227281e-02 3.229185e-02 3.229185e-02 3.231089e-02 3.231089e-02 3.255841e-02
##   [667] 3.282497e-02 3.314865e-02 3.314865e-02 3.314865e-02 3.314865e-02 3.314865e-02 3.314865e-02
##   [676] 3.366273e-02 3.370081e-02 3.373889e-02 3.373889e-02 3.375793e-02 3.392929e-02 3.392929e-02
##   [685] 3.394833e-02 3.408161e-02 3.410065e-02 3.410065e-02 3.436721e-02 3.436721e-02 3.438625e-02
```

```
##  [694] 3.530017e-02 3.531921e-02 3.531921e-02 3.537633e-02 3.541441e-02 3.541441e-02 3.564289e-02
##  [703] 3.577617e-02 3.579521e-02 3.579521e-02 3.583329e-02 3.592849e-02 3.592849e-02 3.592849e-02
##  [712] 3.604273e-02 3.606177e-02 3.636641e-02 3.638545e-02 3.640449e-02 3.640449e-02 3.640449e-02
##  [721] 3.651873e-02 3.651873e-02 3.661393e-02 3.667105e-02 3.674721e-02 3.682337e-02 3.682337e-02
##  [730] 3.712801e-02 3.712801e-02 3.712801e-02 3.712801e-02 3.712801e-02 3.712801e-02 3.773729e-02
##  [739] 3.773729e-02 3.773729e-02 3.779441e-02 3.785153e-02 3.785153e-02 3.785153e-02 3.790865e-02
##  [748] 3.808001e-02 3.808001e-02 3.809905e-02 3.809905e-02 3.809905e-02 3.817521e-02 3.817521e-02
##  [757] 3.828945e-02 3.828945e-02 3.832753e-02 3.832753e-02 3.832753e-02 3.832753e-02 3.832753e-02
##  [766] 3.847985e-02 3.847985e-02 3.847985e-02 3.849889e-02 3.851793e-02 3.973649e-02 3.975553e-02
##  [775] 3.977457e-02 3.977457e-02 3.981265e-02 4.026961e-02 4.026961e-02 4.030769e-02 4.030769e-02
##  [784] 4.036481e-02 4.036481e-02 4.040289e-02 4.040289e-02 4.057425e-02 4.059329e-02 4.059329e-02
##  [793] 4.078369e-02 4.080273e-02 4.158337e-02 4.322081e-02 4.322081e-02 4.325889e-02 4.325889e-02
##  [802] 4.343025e-02 4.343025e-02 4.343025e-02 4.343025e-02 4.343025e-02 4.344929e-02 4.346833e-02
##  [811] 4.415377e-02 4.417281e-02 4.422993e-02 4.422993e-02 4.424897e-02 4.426801e-02 4.428705e-02
##  [820] 4.438225e-02 4.438225e-02 4.438225e-02 4.438225e-02 4.455361e-02 4.459169e-02 4.462977e-02
##  [829] 4.466785e-02 4.466785e-02 4.470593e-02 4.598161e-02 4.601969e-02 4.624817e-02 4.624817e-02
##  [838] 4.624817e-02 4.624817e-02 4.624817e-02 4.624817e-02 4.624817e-02 4.626721e-02 4.628625e-02
##  [847] 4.630529e-02 4.636241e-02 4.676225e-02 4.681937e-02 4.681937e-02 4.681937e-02 4.681937e-02
##  [856] 4.683841e-02 4.685745e-02 4.685745e-02 4.685745e-02 4.687649e-02 4.689553e-02 4.691457e-02
##  [865] 4.697169e-02 4.697169e-02 4.699073e-02 4.702881e-02 4.708593e-02 4.708593e-02 4.718113e-02
##  [874] 4.727633e-02 4.754289e-02 4.754289e-02 4.756193e-02 4.756193e-02 4.758097e-02 4.758097e-02
##  [883] 4.769521e-02 4.780945e-02 4.780945e-02 4.780945e-02 4.780945e-02 4.786657e-02 4.805697e-02
##  [892] 4.815217e-02 4.815217e-02 4.815217e-02 4.826641e-02 4.826641e-02 4.866625e-02 4.866625e-02
##  [901] 4.878049e-02 4.878049e-02 4.879953e-02 4.879953e-02 4.879953e-02 4.883761e-02 4.883761e-02
##  [910] 4.891377e-02 4.900897e-02 4.908513e-02 4.914225e-02 4.914225e-02 4.914225e-02 4.918033e-02
##  [919] 4.919937e-02 4.919937e-02 4.919937e-02 4.919937e-02 4.954209e-02 4.961825e-02 4.961825e-02
##  [928] 4.992289e-02 4.992289e-02 4.994193e-02 4.994193e-02 4.994193e-02 4.994193e-02 4.994193e-02
##  [937] 5.001809e-02 5.009425e-02 5.009425e-02 5.037985e-02 5.037985e-02 5.037985e-02 5.037985e-02
##  [946] 5.043697e-02 5.049409e-02 5.049409e-02 5.049409e-02 5.057025e-02 5.058929e-02 5.097009e-02
##  [955] 5.138897e-02 5.138897e-02 5.142705e-02 5.142705e-02 5.146513e-02 5.165553e-02 5.167457e-02
##  [964] 5.388321e-02 5.416881e-02 5.416881e-02 5.418785e-02 5.422593e-02 5.422593e-02 5.426401e-02
##  [973] 5.432113e-02 5.435921e-02 5.445441e-02 5.449249e-02 5.454961e-02 5.456865e-02 5.458769e-02
##  [982] 5.483521e-02 5.508273e-02 5.512081e-02 5.512081e-02 5.512081e-02 5.513985e-02 5.513985e-02
##  [991] 5.521601e-02 5.523505e-02 5.536833e-02 5.538737e-02 5.538737e-02 5.538737e-02 5.540641e-02
## [1000] 5.552065e-02 5.552065e-02 5.563489e-02 5.576817e-02 5.578721e-02 5.580625e-02 5.580625e-02
## [1009] 5.590145e-02 5.590145e-02 5.639649e-02 5.643457e-02 5.643457e-02 5.643457e-02 5.647265e-02
## [1018] 5.685345e-02 5.687249e-02 5.687249e-02 5.687249e-02 5.687249e-02 5.689153e-02 5.704385e-02
## [1027] 5.736753e-02 5.736753e-02 5.894785e-02 5.896689e-02 5.902401e-02 5.930961e-02 5.936673e-02
## [1036] 5.953809e-02 5.955713e-02 5.957617e-02 5.959521e-02 5.961425e-02 5.963329e-02 5.963329e-02
## [1045] 6.003313e-02 6.005217e-02 6.005217e-02 6.009025e-02 6.010929e-02 6.010929e-02 6.012833e-02
## [1054] 6.037585e-02 6.050913e-02 6.056625e-02 6.058529e-02 6.058529e-02 6.066145e-02 6.066145e-02
## [1063] 6.165153e-02 6.165153e-02 6.184193e-02 6.184193e-02 6.191809e-02 6.199425e-02 6.203233e-02
## [1072] 6.210849e-02 6.216561e-02 6.216561e-02 6.220369e-02 6.696369e-02 6.761105e-02 6.772529e-02
## [1081] 6.801089e-02 6.806801e-02 6.806801e-02 6.820129e-02 6.867729e-02 6.867729e-02 6.867729e-02
## [1090] 6.875345e-02 6.875345e-02 6.877249e-02 6.896289e-02 6.896289e-02 6.905809e-02 6.909617e-02
## [1099] 6.932465e-02 6.934369e-02 6.972449e-02 6.983873e-02 6.983873e-02 6.983873e-02 6.983873e-02
## [1108] 6.985777e-02 6.989585e-02 6.991489e-02 6.991489e-02 6.993393e-02 6.993393e-02 6.995297e-02
## [1117] 6.995297e-02 6.999105e-02 7.002913e-02 7.008625e-02 7.008625e-02 7.008625e-02 7.010529e-02
## [1126] 7.050513e-02 7.052417e-02 7.052417e-02 7.052417e-02 7.071457e-02 7.134289e-02 7.134289e-02
## [1135] 7.153329e-02 7.155233e-02 7.157137e-02 7.252337e-02 7.258049e-02 7.259953e-02 7.259953e-02
## [1144] 7.269473e-02 7.301841e-02 7.303745e-02 7.305649e-02 7.305649e-02 7.636945e-02 7.638849e-02
## [1153] 7.650273e-02 7.650273e-02 7.652177e-02 7.654081e-02 7.657889e-02 7.659793e-02 7.659793e-02
## [1162] 7.686449e-02 7.688353e-02 7.688353e-02 7.796881e-02 7.804497e-02 7.815921e-02 7.815921e-02
## [1171] 7.874945e-02 7.874945e-02 7.878753e-02 7.880657e-02 7.880657e-02 7.882561e-02 7.882561e-02
## [1180] 7.903505e-02 7.905409e-02 7.905409e-02 7.907313e-02 7.935873e-02 7.935873e-02 7.939681e-02
## [1189] 7.945393e-02 7.947297e-02 7.949201e-02 7.954913e-02 7.954913e-02 7.994897e-02 7.994897e-02
## [1198] 8.006321e-02 8.008225e-02 8.008225e-02 8.050113e-02 8.050113e-02 8.053921e-02 8.067249e-02
## [1207] 8.101521e-02 8.103425e-02 8.103425e-02 8.114849e-02 8.114849e-02 8.114849e-02 8.116753e-02
```

```
## [1216] 8.185297e-02 8.194817e-02 8.202433e-02 8.204337e-02 8.210049e-02 8.210049e-02 8.210049e-02
## [1225] 8.215761e-02 8.219569e-02 8.223377e-02 8.227185e-02 8.227185e-02 8.227185e-02 8.229089e-02
## [1234] 8.229089e-02 8.230993e-02 8.240513e-02 8.240513e-02 8.240513e-02 8.242417e-02 8.248129e-02
## [1243] 8.251937e-02 8.253841e-02 8.255745e-02 8.255745e-02 8.259553e-02 8.261457e-02 8.278593e-02
## [1252] 8.280497e-02 8.282401e-02 8.284305e-02 8.299537e-02 8.299537e-02 8.299537e-02 8.301441e-02
## [1261] 8.354753e-02 8.356657e-02 8.371889e-02 8.375697e-02 8.375697e-02 8.390929e-02 8.390929e-02
## [1270] 8.406161e-02 8.406161e-02 8.409969e-02 8.409969e-02 8.409969e-02 8.419489e-02 8.427105e-02
## [1279] 8.430913e-02 8.432817e-02 8.436625e-02 8.440433e-02 8.440433e-02 8.451857e-02 8.453761e-02
## [1288] 8.488033e-02 8.488033e-02 8.489937e-02 8.489937e-02 8.495649e-02 8.495649e-02 8.495649e-02
## [1297] 8.505169e-02 8.514689e-02 8.522305e-02 8.537537e-02 8.581329e-02 8.581329e-02 8.583233e-02
## [1306] 8.585137e-02 8.585137e-02 8.585137e-02 8.653681e-02 8.653681e-02 8.655585e-02 8.676529e-02
## [1315] 8.708897e-02 8.710801e-02 8.712705e-02 8.745073e-02 8.745073e-02 8.745073e-02 8.745073e-02
## [1324] 8.746977e-02 8.748881e-02 8.756497e-02 8.756497e-02 8.762209e-02 8.764113e-02 8.783153e-02
## [1333] 8.903105e-02 8.906913e-02 8.912625e-02 8.916433e-02 8.922145e-02 8.927857e-02 8.929761e-02
## [1342] 8.941185e-02 8.941185e-02 8.941185e-02 8.944993e-02 8.969745e-02 8.969745e-02 8.971649e-02
## [1351] 9.087793e-02 9.087793e-02 9.087793e-02 9.087793e-02 9.122065e-02 9.141105e-02 9.141105e-02
## [1360] 9.143009e-02 9.143009e-02 9.143009e-02 9.144913e-02 9.146817e-02 9.158241e-02 9.160145e-02
## [1369] 9.165857e-02 9.171569e-02 9.173473e-02 9.177281e-02 9.181089e-02 9.181089e-02 9.181089e-02
## [1378] 9.192513e-02 9.198225e-02 9.207745e-02 9.207745e-02 9.207745e-02 9.207745e-02 9.209649e-02
## [1387] 9.209649e-02 9.209649e-02 9.219169e-02 9.278193e-02 9.291521e-02 9.295329e-02 9.310561e-02
## [1396] 9.360065e-02 9.390530e-02 9.394338e-02 9.394338e-02 9.396242e-02 9.417186e-02 9.419090e-02
## [1405] 9.438130e-02 9.438130e-02 9.569506e-02 9.582834e-02 9.584738e-02 9.588546e-02 9.647570e-02
## [1414] 9.695170e-02 9.714210e-02 9.716114e-02 9.716114e-02 9.716114e-02 9.718018e-02 9.718018e-02
## [1423] 9.725634e-02 9.725634e-02 9.725634e-02 9.737058e-02 9.737058e-02 9.742770e-02 9.750386e-02
## [1432] 9.759906e-02 9.767522e-02 9.769426e-02 9.784658e-02 9.784658e-02 9.790370e-02 9.790370e-02
## [1441] 9.794178e-02 9.796082e-02 9.801794e-02 9.801794e-02 9.809410e-02 9.811314e-02 9.811314e-02
## [1450] 1.006074e-01 1.006264e-01 1.006264e-01 1.006264e-01 1.006264e-01 1.006264e-01 1.009882e-01
## [1459] 1.012547e-01 1.013499e-01 1.014261e-01 1.015784e-01 1.015975e-01 1.016165e-01 1.017688e-01
## [1468] 1.019783e-01 1.020163e-01 1.020163e-01 1.020544e-01 1.020544e-01 1.027208e-01 1.027779e-01
## [1477] 1.029303e-01 1.031778e-01 1.031968e-01 1.031968e-01 1.037299e-01 1.038061e-01 1.038442e-01
## [1486] 1.038823e-01 1.038823e-01 1.049485e-01 1.053103e-01 1.053103e-01 1.053674e-01 1.053864e-01
## [1495] 1.054816e-01 1.054816e-01 1.055197e-01 1.062051e-01 1.062051e-01 1.062051e-01 1.065098e-01
## [1504] 1.071381e-01 1.077093e-01 1.077093e-01 1.077664e-01 1.078235e-01 1.079759e-01 1.082805e-01
## [1513] 1.083567e-01 1.083567e-01 1.083567e-01 1.083947e-01 1.086042e-01 1.086423e-01 1.086613e-01
## [1522] 1.088517e-01 1.088517e-01 1.088517e-01 1.088517e-01 1.090802e-01 1.091373e-01 1.091373e-01
## [1531] 1.092896e-01 1.093087e-01 1.093277e-01 1.093277e-01 1.093277e-01 1.093277e-01 1.095371e-01
## [1540] 1.099751e-01 1.099941e-01 1.099941e-01 1.101845e-01 1.102416e-01 1.102416e-01 1.109651e-01
## [1549] 1.109842e-01 1.109842e-01 1.111365e-01 1.112127e-01 1.112127e-01 1.112127e-01 1.112507e-01
## [1558] 1.112698e-01 1.113079e-01 1.116125e-01 1.116125e-01 1.118219e-01 1.118410e-01 1.118410e-01
## [1567] 1.118981e-01 1.119171e-01 1.119362e-01 1.125645e-01 1.126407e-01 1.127739e-01 1.129263e-01
## [1576] 1.129834e-01 1.129834e-01 1.130024e-01 1.130215e-01 1.132880e-01 1.132880e-01 1.132880e-01
## [1585] 1.137450e-01 1.137640e-01 1.137831e-01 1.138021e-01 1.138211e-01 1.138211e-01 1.140115e-01
## [1594] 1.140496e-01 1.141639e-01 1.141829e-01 1.141829e-01 1.141829e-01 1.143162e-01 1.143162e-01
## [1603] 1.170770e-01 1.171151e-01 1.172864e-01 1.172864e-01 1.172864e-01 1.173816e-01 1.174007e-01
## [1612] 1.174578e-01 1.174578e-01 1.174578e-01 1.174578e-01 1.174578e-01 1.174768e-01 1.175720e-01
## [1621] 1.176863e-01 1.177053e-01 1.201805e-01 1.201805e-01 1.201805e-01 1.202376e-01 1.211896e-01
## [1630] 1.215133e-01 1.215323e-01 1.215704e-01 1.215704e-01 1.215895e-01 1.215895e-01 1.216085e-01
## [1639] 1.221416e-01 1.222368e-01 1.222749e-01 1.240837e-01 1.242360e-01 1.242360e-01 1.242360e-01
## [1648] 1.244264e-01 1.244645e-01 1.245026e-01 1.245026e-01 1.245026e-01 1.245216e-01 1.245978e-01
## [1657] 1.249595e-01 1.249786e-01 1.250167e-01 1.253784e-01 1.254165e-01 1.254165e-01 1.254927e-01
## [1666] 1.254927e-01 1.254927e-01 1.254927e-01 1.257021e-01 1.257211e-01 1.257211e-01 1.257402e-01
## [1675] 1.266351e-01 1.266922e-01 1.266922e-01 1.278155e-01 1.278346e-01 1.278727e-01 1.278727e-01
## [1684] 1.279488e-01 1.284058e-01 1.284629e-01 1.284629e-01 1.284819e-01 1.284819e-01 1.285391e-01
## [1693] 1.289960e-01 1.289960e-01 1.289960e-01 1.292626e-01 1.293197e-01 1.293197e-01 1.293768e-01
## [1702] 1.298719e-01 1.298909e-01 1.299099e-01 1.299099e-01 1.303859e-01 1.303859e-01 1.304050e-01
## [1711] 1.311666e-01 1.312618e-01 1.312618e-01 1.312808e-01 1.313379e-01 1.314331e-01 1.314712e-01
## [1720] 1.315093e-01 1.338703e-01 1.342511e-01 1.342511e-01 1.344224e-01 1.344224e-01 1.352031e-01
## [1729] 1.357171e-01 1.357171e-01 1.359266e-01 1.359456e-01 1.359647e-01 1.362122e-01 1.369928e-01
```

```
## [1738] 1.386493e-01 1.386493e-01 1.386683e-01 1.390111e-01 1.390111e-01 1.390301e-01 1.390491e-01
## [1747] 1.390872e-01 1.391063e-01 1.395061e-01 1.395061e-01 1.395061e-01 1.395251e-01 1.395251e-01
## [1756] 1.402106e-01 1.402106e-01 1.402487e-01 1.402487e-01 1.402677e-01 1.402867e-01 1.403058e-01
## [1765] 1.403439e-01 1.403439e-01 1.403439e-01 1.403629e-01 1.403819e-01 1.404010e-01 1.404391e-01
## [1774] 1.404962e-01 1.405343e-01 1.405723e-01 1.405723e-01 1.405914e-01 1.408199e-01 1.408579e-01
## [1783] 1.411626e-01 1.411626e-01 1.411626e-01 1.411626e-01 1.411626e-01 1.411626e-01 1.411816e-01
## [1792] 1.412959e-01 1.414672e-01 1.414672e-01 1.414672e-01 1.428762e-01 1.428762e-01 1.430285e-01
## [1801] 1.431618e-01 1.433903e-01 1.433903e-01 1.433903e-01 1.434474e-01 1.435045e-01 1.435045e-01
## [1810] 1.435235e-01 1.435235e-01 1.435235e-01 1.436568e-01 1.437330e-01 1.437520e-01 1.437901e-01
## [1819] 1.438282e-01 1.438282e-01 1.438472e-01 1.438472e-01 1.439995e-01 1.440376e-01 1.440376e-01
## [1828] 1.444184e-01 1.444755e-01 1.444946e-01 1.449515e-01 1.449896e-01 1.450087e-01 1.450277e-01
## [1837] 1.451800e-01 1.453895e-01 1.454085e-01 1.454466e-01 1.454466e-01 1.454466e-01 1.454466e-01
## [1846] 1.455418e-01 1.455608e-01 1.455608e-01 1.460178e-01 1.460368e-01 1.460559e-01 1.465890e-01
## [1855] 1.466651e-01 1.466842e-01 1.466842e-01 1.466842e-01 1.466842e-01 1.467413e-01 1.467794e-01
## [1864] 1.468365e-01 1.468365e-01 1.468365e-01 1.470459e-01 1.471031e-01 1.471031e-01 1.471221e-01
## [1873] 1.471602e-01 1.471792e-01 1.471792e-01 1.471792e-01 1.474839e-01 1.475981e-01 1.475981e-01
## [1882] 1.476933e-01 1.477695e-01 1.478075e-01 1.479027e-01 1.479599e-01 1.480360e-01 1.480551e-01
## [1891] 1.480931e-01 1.480931e-01 1.481122e-01 1.481122e-01 1.482645e-01 1.482835e-01 1.519202e-01
## [1900] 1.520725e-01 1.525295e-01 1.525295e-01 1.525295e-01 1.525866e-01 1.526627e-01 1.530435e-01
## [1909] 1.532720e-01 1.532720e-01 1.532720e-01 1.532720e-01 1.532720e-01 1.532720e-01 1.533672e-01
## [1918] 1.535767e-01 1.537099e-01 1.537290e-01 1.537290e-01 1.540146e-01 1.542240e-01 1.549666e-01
## [1927] 1.551760e-01 1.551760e-01 1.554426e-01 1.555378e-01 1.556711e-01 1.556711e-01 1.556901e-01
## [1936] 1.559757e-01 1.560899e-01 1.562803e-01 1.562803e-01 1.562803e-01 1.565469e-01 1.565469e-01
## [1945] 1.566992e-01 1.571371e-01 1.577845e-01 1.578035e-01 1.578416e-01 1.578416e-01 1.578416e-01
## [1954] 1.578797e-01 1.579749e-01 1.579939e-01 1.579939e-01 1.580511e-01 1.584890e-01 1.585080e-01
## [1963] 1.585651e-01 1.585842e-01 1.585842e-01 1.585842e-01 1.585842e-01 1.586032e-01 1.586032e-01
## [1972] 1.588507e-01 1.588698e-01 1.595171e-01 1.595743e-01 1.595743e-01 1.595933e-01 1.596123e-01
## [1981] 1.596314e-01 1.596504e-01 1.599360e-01 1.601645e-01 1.601645e-01 1.602026e-01 1.602787e-01
## [1990] 1.609261e-01 1.609832e-01 1.609832e-01 1.611165e-01 1.613259e-01 1.613259e-01 1.614021e-01
## [1999] 1.614211e-01 1.614211e-01 1.614211e-01 1.614783e-01 1.616115e-01 1.619162e-01 1.619543e-01
## [2008] 1.619923e-01 1.641819e-01 1.643914e-01 1.643914e-01 1.643914e-01 1.644485e-01 1.645247e-01
## [2017] 1.645437e-01 1.653243e-01 1.654005e-01 1.654767e-01 1.657432e-01 1.657623e-01 1.657623e-01
## [2026] 1.660479e-01 1.660479e-01 1.660669e-01 1.661431e-01 1.661811e-01 1.662002e-01 1.663906e-01
## [2035] 1.665048e-01 1.665048e-01 1.666952e-01 1.666952e-01 1.667143e-01 1.667523e-01 1.667523e-01
## [2044] 1.668285e-01 1.668285e-01 1.668666e-01 1.669427e-01 1.677615e-01 1.677805e-01 1.678376e-01
## [2053] 1.682565e-01 1.682946e-01 1.683136e-01 1.683136e-01 1.683898e-01 1.683898e-01 1.684088e-01
## [2062] 1.688467e-01 1.689229e-01 1.689229e-01 1.690181e-01 1.690371e-01 1.691514e-01 1.691514e-01
## [2071] 1.692656e-01 1.693418e-01 1.693608e-01 1.693608e-01 1.695131e-01 1.699511e-01 1.699701e-01
## [2080] 1.699891e-01 1.703319e-01 1.703890e-01 1.705413e-01 1.705413e-01 1.706175e-01 1.706365e-01
## [2089] 1.706936e-01 1.708269e-01 1.708269e-01 1.708459e-01 1.709031e-01 1.720645e-01 1.725024e-01
## [2098] 1.734163e-01 1.734925e-01 1.734925e-01 1.738543e-01 1.738543e-01 1.738923e-01 1.742160e-01
## [2107] 1.744064e-01 1.744826e-01 1.747491e-01 1.747491e-01 1.747491e-01 1.747491e-01 1.747682e-01
## [2116] 1.748063e-01 1.748063e-01 1.749967e-01 1.750347e-01 1.750538e-01 1.753203e-01 1.753394e-01
## [2125] 1.753965e-01 1.753965e-01 1.753965e-01 1.754155e-01 1.761391e-01 1.761391e-01 1.761391e-01
## [2134] 1.765389e-01 1.765389e-01 1.765770e-01 1.779288e-01 1.779669e-01 1.779859e-01 1.780050e-01
## [2143] 1.784048e-01 1.785191e-01 1.785952e-01 1.786143e-01 1.786143e-01 1.789189e-01 1.789379e-01
## [2152] 1.792807e-01 1.793378e-01 1.805563e-01 1.805563e-01 1.805563e-01 1.815845e-01 1.815845e-01
## [2161] 1.816035e-01 1.816035e-01 1.816035e-01 1.816226e-01 1.816416e-01 1.817749e-01 1.817749e-01
## [2170] 1.818891e-01 1.819653e-01 1.821176e-01 1.821938e-01 1.822509e-01 1.822509e-01 1.823271e-01
## [2179] 1.887245e-01 1.895432e-01 1.895623e-01 1.895813e-01 1.895813e-01 1.896765e-01 1.896765e-01
## [2188] 1.902858e-01 1.904191e-01 1.904191e-01 1.904381e-01 1.908570e-01 1.908570e-01 1.910093e-01
## [2197] 1.913520e-01 1.913520e-01 1.913520e-01 1.913901e-01 1.914472e-01 1.915044e-01 1.916567e-01
## [2206] 1.919042e-01 1.919994e-01 1.923231e-01 1.923231e-01 1.925325e-01 1.927991e-01 1.928181e-01
## [2215] 1.930466e-01 1.930847e-01 1.931037e-01 1.931037e-01 1.931037e-01 1.931037e-01 1.931037e-01
## [2224] 1.931418e-01 1.931799e-01 1.942842e-01 1.942842e-01 1.942842e-01 1.943223e-01 1.943223e-01
## [2233] 1.961882e-01 1.963976e-01 1.966261e-01 1.968165e-01 1.968165e-01 1.968356e-01 1.968356e-01
## [2242] 1.971212e-01 1.971402e-01 1.971592e-01 1.971592e-01 1.971973e-01 1.972164e-01 1.972164e-01
## [2251] 1.972544e-01 1.973116e-01 1.982826e-01 1.982826e-01 1.982826e-01 1.982826e-01 1.983397e-01
```

```
## [2260] 1.985301e-01 1.985301e-01 1.993108e-01 1.993108e-01 1.993108e-01 1.993298e-01 1.994440e-01
## [2269] 1.995012e-01 2.001295e-01 2.001866e-01 2.002628e-01 2.003199e-01 2.003770e-01 2.036900e-01
## [2278] 2.045658e-01 2.045658e-01 2.045658e-01 2.045658e-01 2.046039e-01 2.046039e-01 2.049085e-01
## [2287] 2.050228e-01 2.050418e-01 2.053084e-01 2.053274e-01 2.053464e-01 2.053464e-01 2.054226e-01
## [2296] 2.054988e-01 2.055178e-01 2.055940e-01 2.056130e-01 2.057082e-01 2.057272e-01 2.057653e-01
## [2305] 2.071172e-01 2.074408e-01 2.074408e-01 2.074980e-01 2.075170e-01 2.076693e-01 2.076884e-01
## [2314] 2.076884e-01 2.080120e-01 2.080120e-01 2.080311e-01 2.080501e-01 2.086213e-01 2.086213e-01
## [2323] 2.096685e-01 2.099351e-01 2.099922e-01 2.104492e-01 2.104682e-01 2.115725e-01 2.115725e-01
## [2332] 2.117248e-01 2.118010e-01 2.118010e-01 2.118200e-01 2.120485e-01 2.120676e-01 2.125245e-01
## [2341] 2.134575e-01 2.134575e-01 2.140858e-01 2.141810e-01 2.141810e-01 2.142191e-01 2.142191e-01
## [2350] 2.142381e-01 2.142381e-01 2.142572e-01 2.143524e-01 2.147903e-01 2.147903e-01 2.148284e-01
## [2359] 2.148855e-01 2.151140e-01 2.151520e-01 2.152092e-01 2.155709e-01 2.155709e-01 2.155900e-01
## [2368] 2.156471e-01 2.158756e-01 2.158946e-01 2.160088e-01 2.160469e-01 2.160469e-01 2.160660e-01
## [2377] 2.160850e-01 2.160850e-01 2.160850e-01 2.160850e-01 2.161040e-01 2.162754e-01 2.174178e-01
## [2386] 2.175130e-01 2.176463e-01 2.177034e-01 2.177034e-01 2.177034e-01 2.177415e-01 2.177605e-01
## [2395] 2.181604e-01 2.181604e-01 2.186173e-01 2.186744e-01 2.187316e-01 2.187316e-01 2.187696e-01
## [2404] 2.189220e-01 2.189981e-01 2.191695e-01 2.191695e-01 2.192456e-01 2.192647e-01 2.192647e-01
## [2413] 2.194741e-01 2.195884e-01 2.195884e-01 2.196645e-01 2.198740e-01 2.201596e-01 2.202357e-01
## [2422] 2.202928e-01 2.204261e-01 2.204261e-01 2.213972e-01 2.219493e-01 2.220255e-01 2.221016e-01
## [2431] 2.222540e-01 2.222540e-01 2.222540e-01 2.225015e-01 2.225205e-01 2.227680e-01 2.228061e-01
## [2440] 2.229204e-01 2.229394e-01 2.229394e-01 2.229775e-01 2.230156e-01 2.234916e-01 2.235677e-01
## [2449] 2.273948e-01 2.273948e-01 2.273948e-01 2.274328e-01 2.274519e-01 2.284800e-01 2.284800e-01
## [2458] 2.285181e-01 2.285181e-01 2.285181e-01 2.285181e-01 2.285752e-01 2.286324e-01 2.286514e-01
## [2467] 2.288799e-01 2.288799e-01 2.288799e-01 2.288799e-01 2.288989e-01 2.289560e-01 2.289941e-01
## [2476] 2.300794e-01 2.300794e-01 2.300794e-01 2.300794e-01 2.302127e-01 2.302317e-01 2.302317e-01
## [2485] 2.302888e-01 2.302888e-01 2.303650e-01 2.303840e-01 2.304602e-01 2.304792e-01 2.304792e-01
## [2494] 2.305173e-01 2.305173e-01 2.305744e-01 2.305744e-01 2.305935e-01 2.312408e-01 2.312980e-01
## [2503] 2.314312e-01 2.314503e-01 2.315836e-01 2.316216e-01 2.316407e-01 2.316597e-01 2.316597e-01
## [2512] 2.318882e-01 2.319072e-01 2.319072e-01 2.319072e-01 2.319263e-01 2.321357e-01 2.327640e-01
## [2521] 2.329354e-01 2.329354e-01 2.330116e-01 2.330116e-01 2.341159e-01 2.341540e-01 2.347823e-01
## [2530] 2.354106e-01 2.354106e-01 2.356391e-01 2.356772e-01 2.357343e-01 2.357533e-01 2.358104e-01
## [2539] 2.360008e-01 2.360008e-01 2.368576e-01 2.368576e-01 2.369338e-01 2.387997e-01 2.393138e-01
## [2548] 2.411988e-01 2.421888e-01 2.421888e-01 2.422079e-01 2.423983e-01 2.427029e-01 2.429885e-01
## [2557] 2.430076e-01 2.430266e-01 2.430456e-01 2.430647e-01 2.431408e-01 2.432551e-01 2.432551e-01
## [2566] 2.443023e-01 2.443594e-01 2.443594e-01 2.443784e-01 2.443975e-01 2.444165e-01 2.444356e-01
## [2575] 2.444736e-01 2.445308e-01 2.445308e-01 2.445308e-01 2.445688e-01 2.445688e-01 2.445688e-01
## [2584] 2.445688e-01 2.446260e-01 2.446260e-01 2.448354e-01 2.451400e-01 2.451591e-01 2.451591e-01
## [2593] 2.466632e-01 2.467013e-01 2.467013e-01 2.467204e-01 2.467394e-01 2.467394e-01 2.469108e-01
## [2602] 2.471392e-01 2.471964e-01 2.471964e-01 2.472154e-01 2.472344e-01 2.472344e-01 2.472344e-01
## [2611] 2.474058e-01 2.474820e-01 2.475581e-01 2.475772e-01 2.475962e-01 2.482626e-01 2.483007e-01
## [2620] 2.486434e-01 2.487576e-01 2.488338e-01 2.488719e-01 2.489290e-01 2.489290e-01 2.489671e-01
## [2629] 2.493669e-01 2.493669e-01 2.493860e-01 2.493860e-01 2.494050e-01 2.495002e-01 2.495383e-01
## [2638] 2.501476e-01 2.504903e-01 2.505093e-01 2.505284e-01 2.505284e-01 2.514042e-01 2.514042e-01
## [2647] 2.516136e-01 2.522991e-01 2.522991e-01 2.524324e-01 2.524324e-01 2.524704e-01 2.524704e-01
## [2656] 2.529084e-01 2.530036e-01 2.531368e-01 2.536700e-01 2.537080e-01 2.538223e-01 2.552122e-01
## [2665] 2.554788e-01 2.555168e-01 2.556501e-01 2.566402e-01 2.566402e-01 2.566402e-01 2.566592e-01
## [2674] 2.568496e-01 2.568496e-01 2.577445e-01 2.578207e-01 2.592296e-01 2.592487e-01 2.592487e-01
## [2683] 2.593248e-01 2.593248e-01 2.594010e-01 2.594010e-01 2.594391e-01 2.594581e-01 2.598770e-01
## [2692] 2.599532e-01 2.599532e-01 2.600293e-01 2.601055e-01 2.603720e-01 2.603720e-01 2.603720e-01
## [2701] 2.604292e-01 2.609432e-01 2.609432e-01 2.609623e-01 2.610004e-01 2.610194e-01 2.610575e-01
## [2710] 2.612288e-01 2.612479e-01 2.612479e-01 2.612669e-01 2.612860e-01 2.613240e-01 2.613240e-01
## [2719] 2.637421e-01 2.641039e-01 2.646370e-01 2.646370e-01 2.651320e-01 2.651320e-01 2.651320e-01
## [2728] 2.657223e-01 2.657413e-01 2.657604e-01 2.664077e-01 2.664648e-01 2.664648e-01 2.664648e-01
## [2737] 2.665981e-01 2.666552e-01 2.666552e-01 2.666933e-01 2.666933e-01 2.671312e-01 2.671312e-01
## [2746] 2.671693e-01 2.671693e-01 2.671693e-01 2.671693e-01 2.671693e-01 2.673026e-01 2.673788e-01
## [2755] 2.675120e-01 2.675120e-01 2.675311e-01 2.675311e-01 2.675311e-01 2.675311e-01 2.675311e-01
## [2764] 2.676263e-01 2.679880e-01 2.680452e-01 2.680642e-01 2.692066e-01 2.693970e-01 2.699492e-01
## [2773] 2.700063e-01 2.700253e-01 2.700253e-01 2.703300e-01 2.703871e-01 2.712629e-01 2.714533e-01
```

```
## [2782] 2.716056e-01 2.716056e-01 2.716056e-01 2.716247e-01 2.716437e-01 2.716628e-01 2.716628e-01
## [2791] 2.731479e-01 2.734144e-01 2.734144e-01 2.734144e-01 2.734716e-01 2.734716e-01 2.734716e-01
## [2800] 2.743093e-01 2.744426e-01 2.780983e-01 2.785172e-01 2.785743e-01 2.786504e-01 2.786885e-01
## [2809] 2.789170e-01 2.789551e-01 2.789932e-01 2.796024e-01 2.796024e-01 2.796024e-01 2.808972e-01
## [2818] 2.816588e-01 2.816778e-01 2.817730e-01 2.817730e-01 2.818872e-01 2.820396e-01 2.821157e-01
## [2827] 2.825536e-01 2.825727e-01 2.826108e-01 2.827440e-01 2.827440e-01 2.828964e-01 2.829154e-01
## [2836] 2.834866e-01 2.834866e-01 2.838674e-01 2.840197e-01 2.840197e-01 2.840197e-01 2.840197e-01
## [2845] 2.841340e-01 2.841530e-01 2.841530e-01 2.841530e-01 2.841720e-01 2.841720e-01 2.842482e-01
## [2854] 2.867234e-01 2.867234e-01 2.867424e-01 2.867615e-01 2.867805e-01 2.867805e-01 2.867996e-01
## [2863] 2.868948e-01 2.879420e-01 2.879800e-01 2.880752e-01 2.880752e-01 2.880752e-01 2.885703e-01
## [2872] 2.890082e-01 2.896365e-01 2.896556e-01 2.896746e-01 2.896746e-01 2.896746e-01 2.897698e-01
## [2881] 2.903600e-01 2.903600e-01 2.903791e-01 2.904362e-01 2.911407e-01 2.911407e-01 2.912930e-01
## [2890] 2.940348e-01 2.941109e-01 2.941300e-01 2.941300e-01 2.943965e-01 2.946250e-01 2.946440e-01
## [2899] 2.948916e-01 2.948916e-01 2.950439e-01 2.950439e-01 2.972906e-01 2.973287e-01 2.973477e-01
## [2908] 2.983949e-01 2.983949e-01 2.984140e-01 2.984330e-01 2.985092e-01 2.985282e-01 2.985282e-01
## [2917] 2.991565e-01 2.995373e-01 3.016698e-01 3.016888e-01 3.018792e-01 3.019364e-01 3.020696e-01
## [2926] 3.031359e-01 3.031740e-01 3.031740e-01 3.033644e-01 3.033834e-01 3.033834e-01 3.033834e-01
## [2935] 3.034786e-01 3.034786e-01 3.038213e-01 3.038594e-01 3.040308e-01 3.043164e-01 3.046972e-01
## [2944] 3.047543e-01 3.047543e-01 3.049066e-01 3.049066e-01 3.049256e-01 3.067725e-01 3.067916e-01
## [2953] 3.081815e-01 3.083909e-01 3.084290e-01 3.094000e-01 3.095143e-01 3.095143e-01 3.095714e-01
## [2962] 3.100855e-01 3.101045e-01 3.105996e-01 3.105996e-01 3.105996e-01 3.106186e-01 3.106186e-01
## [2971] 3.106567e-01 3.106567e-01 3.107328e-01 3.107328e-01 3.107519e-01 3.107709e-01 3.107709e-01
## [2980] 3.115135e-01 3.116848e-01 3.117610e-01 3.117800e-01 3.117800e-01 3.117800e-01 3.117800e-01
## [2989] 3.149026e-01 3.149026e-01 3.149597e-01 3.149788e-01 3.150740e-01 3.150930e-01 3.153025e-01
## [2998] 3.166924e-01 3.167114e-01 3.168257e-01 3.169209e-01 3.169209e-01 3.169399e-01 3.173588e-01
## [3007] 3.176634e-01 3.178538e-01 3.178538e-01 3.178538e-01 3.178538e-01 3.182727e-01 3.183869e-01
## [3016] 3.185202e-01 3.185773e-01 3.186345e-01 3.186535e-01 3.186725e-01 3.186916e-01 3.187106e-01
## [3025] 3.190153e-01 3.191676e-01 3.195484e-01 3.196055e-01 3.196436e-01 3.196436e-01 3.200815e-01
## [3034] 3.211858e-01 3.215095e-01 3.215857e-01 3.216618e-01 3.216809e-01 3.217380e-01 3.217570e-01
## [3043] 3.220617e-01 3.220807e-01 3.226329e-01 3.226329e-01 3.226900e-01 3.228994e-01 3.228994e-01
## [3052] 3.229946e-01 3.237753e-01 3.239466e-01 3.241561e-01 3.241751e-01 3.241751e-01 3.241941e-01
## [3061] 3.242322e-01 3.243274e-01 3.243465e-01 3.260791e-01 3.266693e-01 3.266693e-01 3.266884e-01
## [3070] 3.268026e-01 3.268026e-01 3.268026e-01 3.268407e-01 3.268597e-01 3.268788e-01 3.268978e-01
## [3079] 3.319815e-01 3.321148e-01 3.322290e-01 3.339617e-01 3.342282e-01 3.344948e-01 3.345138e-01
## [3088] 3.350089e-01 3.350089e-01 3.354468e-01 3.354658e-01 3.354658e-01 3.355039e-01 3.357705e-01
## [3097] 3.359418e-01 3.359418e-01 3.359799e-01 3.359989e-01 3.360180e-01 3.360370e-01 3.361132e-01
## [3106] 3.362845e-01 3.362845e-01 3.363036e-01 3.364369e-01 3.364369e-01 3.364369e-01 3.368367e-01
## [3115] 3.369319e-01 3.369509e-01 3.369509e-01 3.370081e-01 3.370652e-01 3.370652e-01 3.371413e-01
## [3124] 3.388359e-01 3.388359e-01 3.388359e-01 3.388359e-01 3.388549e-01 3.389501e-01 3.391786e-01
## [3133] 3.395594e-01 3.395594e-01 3.399212e-01 3.399212e-01 3.399402e-01 3.399402e-01 3.399783e-01
## [3142] 3.401497e-01 3.402068e-01 3.402068e-01 3.402258e-01 3.402449e-01 3.402639e-01 3.410065e-01
## [3151] 3.411207e-01 3.411588e-01 3.411588e-01 3.414444e-01 3.422250e-01 3.422631e-01 3.422631e-01
## [3160] 3.424725e-01 3.424916e-01 3.424916e-01 3.425106e-01 3.432532e-01 3.433484e-01 3.433674e-01
## [3169] 3.443004e-01 3.443004e-01 3.443956e-01 3.444527e-01 3.444527e-01 3.445860e-01 3.454999e-01
## [3178] 3.493841e-01 3.494221e-01 3.494221e-01 3.494412e-01 3.501647e-01 3.502028e-01 3.538965e-01
## [3187] 3.546201e-01 3.596276e-01 3.596276e-01 3.605034e-01 3.605605e-01 3.605605e-01 3.606177e-01
## [3196] 3.616458e-01 3.616839e-01 3.616839e-01 3.616839e-01 3.617029e-01 3.617220e-01 3.617410e-01
## [3205] 3.619885e-01 3.620076e-01 3.620647e-01 3.621028e-01 3.621028e-01 3.621409e-01 3.623122e-01
## [3214] 3.625407e-01 3.625407e-01 3.626930e-01 3.626930e-01 3.631690e-01 3.631881e-01 3.632261e-01
## [3223] 3.636450e-01 3.640829e-01 3.645209e-01 3.664439e-01 3.665391e-01 3.665772e-01 3.678909e-01
## [3232] 3.679290e-01 3.679861e-01 3.681956e-01 3.682146e-01 3.684812e-01 3.684812e-01 3.684812e-01
## [3241] 3.685764e-01 3.686525e-01 3.689953e-01 3.689953e-01 3.689953e-01 3.689953e-01 3.691857e-01
## [3250] 3.692999e-01 3.692999e-01 3.693380e-01 3.694332e-01 3.708041e-01 3.708231e-01 3.708612e-01
## [3259] 3.727271e-01 3.728794e-01 3.728985e-01 3.729175e-01 3.733745e-01 3.736601e-01 3.736601e-01
## [3268] 3.744217e-01 3.744217e-01 3.745169e-01 3.746692e-01 3.749548e-01 3.750119e-01 3.750119e-01
## [3277] 3.750690e-01 3.750881e-01 3.750881e-01 3.754308e-01 3.754308e-01 3.754498e-01 3.776204e-01
## [3286] 3.789913e-01 3.792007e-01 3.792388e-01 3.797148e-01 3.797719e-01 3.797909e-01 3.797909e-01
## [3295] 3.798861e-01 3.799433e-01 3.799433e-01 3.799433e-01 3.799813e-01 3.800765e-01 3.815236e-01
```

```
## [3304] 3.817140e-01 3.817711e-01 3.817711e-01 3.828564e-01 3.828564e-01 3.828754e-01 3.837132e-01
## [3313] 3.838084e-01 3.838084e-01 3.840178e-01 3.840178e-01 3.840940e-01 3.842082e-01 3.842463e-01
## [3322] 3.842653e-01 3.842844e-01 3.842844e-01 3.842844e-01 3.844938e-01 3.845509e-01 3.845700e-01
## [3331] 3.851031e-01 3.851031e-01 3.851602e-01 3.852173e-01 3.854077e-01 3.854458e-01 3.855601e-01
## [3340] 3.859409e-01 3.859409e-01 3.859599e-01 3.859789e-01 3.859980e-01 3.860170e-01 3.860741e-01
## [3349] 3.869881e-01 3.870452e-01 3.870452e-01 3.871404e-01 3.871594e-01 3.873117e-01 3.873117e-01
## [3358] 3.883018e-01 3.883209e-01 3.883209e-01 3.883209e-01 3.883209e-01 3.883589e-01 3.883970e-01
## [3367] 3.888159e-01 3.891586e-01 3.892729e-01 3.893490e-01 3.894252e-01 3.899583e-01 3.899773e-01
## [3376] 3.901106e-01 3.901106e-01 3.901487e-01 3.901677e-01 3.901868e-01 3.915957e-01 3.915957e-01
## [3385] 3.916529e-01 3.917671e-01 3.918052e-01 3.918052e-01 3.918052e-01 3.918052e-01 3.919575e-01
## [3394] 3.922431e-01 3.923002e-01 3.923002e-01 3.923002e-01 3.926810e-01 3.946041e-01 3.946041e-01
## [3403] 3.956132e-01 3.956322e-01 3.956322e-01 3.960511e-01 3.974220e-01 3.977837e-01 3.978028e-01
## [3412] 3.979932e-01 3.988881e-01 3.988881e-01 3.989261e-01 3.989452e-01 3.990023e-01 3.990023e-01
## [3421] 3.994021e-01 3.994593e-01 3.994593e-01 3.994593e-01 3.996306e-01 3.996306e-01 3.996687e-01
## [3430] 3.997258e-01 3.997258e-01 3.999353e-01 4.010777e-01 4.010967e-01 4.021439e-01 4.021629e-01
## [3439] 4.022201e-01 4.022581e-01 4.023914e-01 4.023914e-01 4.024105e-01 4.024105e-01 4.024676e-01
## [3448] 4.026009e-01 4.030197e-01 4.032101e-01 4.032673e-01 4.041241e-01 4.041621e-01 4.041621e-01
## [3457] 4.075322e-01 4.075322e-01 4.075513e-01 4.082367e-01 4.082748e-01 4.088460e-01 4.088841e-01
## [3466] 4.090554e-01 4.090935e-01 4.090935e-01 4.090935e-01 4.090935e-01 4.092268e-01 4.097599e-01
## [3475] 4.097789e-01 4.098741e-01 4.153386e-01 4.161573e-01 4.161764e-01 4.162145e-01 4.162525e-01
## [3484] 4.163287e-01 4.163477e-01 4.163477e-01 4.183660e-01 4.183660e-01 4.183850e-01 4.185945e-01
## [3493] 4.187849e-01 4.187849e-01 4.187849e-01 4.189753e-01 4.189943e-01 4.190324e-01 4.190324e-01
## [3502] 4.199463e-01 4.199463e-01 4.203461e-01 4.204223e-01 4.204413e-01 4.207841e-01 4.207841e-01
## [3511] 4.211458e-01 4.211458e-01 4.211649e-01 4.212029e-01 4.213553e-01 4.216218e-01 4.217932e-01
## [3520] 4.218503e-01 4.219074e-01 4.220788e-01 4.221169e-01 4.246682e-01 4.246873e-01 4.246873e-01
## [3529] 4.253727e-01 4.254108e-01 4.256773e-01 4.256773e-01 4.279431e-01 4.282287e-01 4.285143e-01
## [3538] 4.291807e-01 4.291997e-01 4.291997e-01 4.306468e-01 4.306468e-01 4.307991e-01 4.310847e-01
## [3547] 4.314655e-01 4.315036e-01 4.315036e-01 4.315417e-01 4.316178e-01 4.316178e-01 4.318273e-01
## [3556] 4.320748e-01 4.321890e-01 4.330839e-01 4.335980e-01 4.336361e-01 4.337503e-01 4.337693e-01
## [3565] 4.349498e-01 4.351402e-01 4.351402e-01 4.351783e-01 4.359399e-01 4.359399e-01 4.362065e-01
## [3574] 4.363017e-01 4.363017e-01 4.363778e-01 4.364540e-01 4.365301e-01 4.395194e-01 4.395575e-01
## [3583] 4.410617e-01 4.410617e-01 4.410807e-01 4.411378e-01 4.411378e-01 4.411378e-01 4.411950e-01
## [3592] 4.412711e-01 4.412902e-01 4.413282e-01 4.413663e-01 4.413663e-01 4.413854e-01 4.414044e-01
## [3601] 4.415186e-01 4.426420e-01 4.432322e-01 4.432894e-01 4.433084e-01 4.433846e-01 4.434036e-01
## [3610] 4.437082e-01 4.437082e-01 4.437463e-01 4.438986e-01 4.438986e-01 4.443556e-01 4.445270e-01
## [3619] 4.445460e-01 4.447364e-01 4.447554e-01 4.460121e-01 4.460121e-01 4.460121e-01 4.461073e-01
## [3628] 4.463358e-01 4.471735e-01 4.472116e-01 4.480113e-01 4.480113e-01 4.487919e-01 4.488871e-01
## [3637] 4.491156e-01 4.491727e-01 4.498010e-01 4.498391e-01 4.498582e-01 4.498582e-01 4.500676e-01
## [3646] 4.508102e-01 4.514956e-01 4.515718e-01 4.516289e-01 4.516289e-01 4.517431e-01 4.517431e-01
## [3655] 4.521049e-01 4.521049e-01 4.521239e-01 4.521620e-01 4.525047e-01 4.525047e-01 4.525238e-01
## [3664] 4.527142e-01 4.527142e-01 4.531902e-01 4.532092e-01 4.532282e-01 4.532282e-01 4.532473e-01
## [3673] 4.533234e-01 4.533425e-01 4.534567e-01 4.534567e-01 4.538375e-01 4.538375e-01 4.538375e-01
## [3682] 4.556844e-01 4.561414e-01 4.561414e-01 4.563889e-01 4.564841e-01 4.569410e-01 4.569410e-01
## [3691] 4.569601e-01 4.570553e-01 4.576265e-01 4.576836e-01 4.577026e-01 4.577026e-01 4.582548e-01
## [3700] 4.601398e-01 4.601398e-01 4.601969e-01 4.614726e-01 4.621961e-01 4.622151e-01 4.622151e-01
## [3709] 4.626721e-01 4.627102e-01 4.627102e-01 4.627102e-01 4.627863e-01 4.628054e-01 4.628054e-01
## [3718] 4.629958e-01 4.630148e-01 4.631862e-01 4.683460e-01 4.683460e-01 4.691076e-01 4.693170e-01
## [3727] 4.693551e-01 4.693742e-01 4.694313e-01 4.694503e-01 4.694503e-01 4.698502e-01 4.699454e-01
## [3736] 4.710687e-01 4.710687e-01 4.727062e-01 4.728204e-01 4.728585e-01 4.728585e-01 4.729156e-01
## [3745] 4.731250e-01 4.731250e-01 4.733154e-01 4.733154e-01 4.739438e-01 4.744578e-01 4.744959e-01
## [3754] 4.746102e-01 4.748958e-01 4.751623e-01 4.755241e-01 4.756574e-01 4.758858e-01 4.766474e-01
## [3763] 4.771234e-01 4.771806e-01 4.771996e-01 4.772186e-01 4.772567e-01 4.773710e-01 4.774090e-01
## [3772] 4.776185e-01 4.776375e-01 4.799604e-01 4.801318e-01 4.801318e-01 4.801318e-01 4.802841e-01
## [3781] 4.808553e-01 4.815026e-01 4.815026e-01 4.816930e-01 4.816930e-01 4.816930e-01 4.816930e-01
## [3790] 4.817692e-01 4.818073e-01 4.818073e-01 4.820738e-01 4.820929e-01 4.820929e-01 4.826260e-01
## [3799] 4.832162e-01 4.832162e-01 4.838446e-01 4.838446e-01 4.845110e-01 4.845110e-01 4.845490e-01
## [3808] 4.855582e-01 4.856153e-01 4.856153e-01 4.873479e-01 4.874812e-01 4.874812e-01 4.875002e-01
## [3817] 4.904134e-01 4.904134e-01 4.905466e-01 4.905657e-01 4.905847e-01 4.931361e-01 4.931361e-01
```

```
## [3826] 4.932313e-01 4.932313e-01 4.932503e-01 4.932503e-01 4.935550e-01 4.948306e-01 4.948306e-01
## [3835] 4.955922e-01 4.956113e-01 4.956494e-01 4.959921e-01 4.961825e-01 4.961825e-01 4.962396e-01
## [3844] 4.963158e-01 4.963158e-01 4.963348e-01 4.963729e-01 4.965442e-01 4.965442e-01 4.965633e-01
## [3853] 4.978770e-01 4.978770e-01 4.979342e-01 4.979532e-01 4.981626e-01 4.981817e-01 4.981817e-01
## [3862] 4.984482e-01 4.984863e-01 4.984863e-01 5.014566e-01 5.014566e-01 5.015327e-01 5.015518e-01
## [3871] 5.021610e-01 5.021610e-01 5.026180e-01 5.031130e-01 5.031511e-01 5.033986e-01 5.033986e-01
## [3880] 5.038556e-01 5.040650e-01 5.041983e-01 5.041983e-01 5.054169e-01 5.070162e-01 5.070162e-01
## [3889] 5.070543e-01 5.070543e-01 5.070734e-01 5.070924e-01 5.074732e-01 5.075684e-01 5.078350e-01
## [3898] 5.083110e-01 5.083110e-01 5.083110e-01 5.083110e-01 5.083110e-01 5.083110e-01 5.101007e-01
## [3907] 5.102150e-01 5.103863e-01 5.103863e-01 5.103863e-01 5.103863e-01 5.103863e-01 5.104434e-01
## [3916] 5.119286e-01 5.120047e-01 5.120618e-01 5.120809e-01 5.120809e-01 5.120809e-01 5.120809e-01
## [3925] 5.127282e-01 5.127282e-01 5.227052e-01 5.241522e-01 5.241713e-01 5.242474e-01 5.244188e-01
## [3934] 5.249138e-01 5.249138e-01 5.249138e-01 5.249519e-01 5.250471e-01 5.250471e-01 5.250471e-01
## [3943] 5.263228e-01 5.263228e-01 5.263799e-01 5.271225e-01 5.272177e-01 5.272748e-01 5.273510e-01
## [3952] 5.275604e-01 5.275604e-01 5.280174e-01 5.290074e-01 5.290074e-01 5.290074e-01 5.291217e-01
## [3961] 5.296548e-01 5.296929e-01 5.297310e-01 5.297310e-01 5.297500e-01 5.299785e-01 5.299975e-01
## [3970] 5.300546e-01 5.300546e-01 5.300737e-01 5.303783e-01 5.304354e-01 5.318254e-01 5.318444e-01
## [3979] 5.319777e-01 5.319967e-01 5.320348e-01 5.320348e-01 5.320538e-01 5.320919e-01 5.326250e-01
## [3988] 5.326822e-01 5.331772e-01 5.332724e-01 5.332914e-01 5.332914e-01 5.332914e-01 5.341863e-01
## [3997] 5.344910e-01 5.345862e-01 5.345862e-01 5.346052e-01 5.351574e-01 5.374612e-01 5.374993e-01
## [4006] 5.375564e-01 5.375754e-01 5.376326e-01 5.378230e-01 5.382799e-01 5.382990e-01 5.388130e-01
## [4015] 5.406980e-01 5.421641e-01 5.422212e-01 5.422212e-01 5.422974e-01 5.423926e-01 5.426591e-01
## [4024] 5.427353e-01 5.431351e-01 5.431732e-01 5.431732e-01 5.436302e-01 5.436873e-01 5.493802e-01
## [4033] 5.495897e-01 5.503513e-01 5.504465e-01 5.504655e-01 5.504655e-01 5.505036e-01 5.511510e-01
## [4042] 5.513414e-01 5.515127e-01 5.515318e-01 5.516079e-01 5.516079e-01 5.516650e-01 5.517031e-01
## [4051] 5.521791e-01 5.521982e-01 5.522172e-01 5.524266e-01 5.545782e-01 5.547305e-01 5.551684e-01
## [4060] 5.563870e-01 5.563870e-01 5.564060e-01 5.564250e-01 5.564631e-01 5.564631e-01 5.565774e-01
## [4069] 5.566535e-01 5.567487e-01 5.567487e-01 5.567678e-01 5.567868e-01 5.567868e-01 5.568439e-01
## [4078] 5.570343e-01 5.573580e-01 5.580434e-01 5.589764e-01 5.589764e-01 5.590145e-01 5.591668e-01
## [4087] 5.592239e-01 5.595095e-01 5.626511e-01 5.628606e-01 5.628606e-01 5.628606e-01 5.628796e-01
## [4096] 5.629558e-01 5.630129e-01 5.634318e-01 5.634508e-01 5.634699e-01 5.635841e-01 5.636222e-01
## [4105] 5.641743e-01 5.645551e-01 5.646884e-01 5.646884e-01 5.646884e-01 5.647075e-01 5.647265e-01
## [4114] 5.649359e-01 5.649359e-01 5.649550e-01 5.649931e-01 5.650121e-01 5.650121e-01 5.650121e-01
## [4123] 5.654881e-01 5.655452e-01 5.659260e-01 5.660212e-01 5.660212e-01 5.661545e-01 5.662116e-01
## [4132] 5.666686e-01 5.709335e-01 5.709526e-01 5.709526e-01 5.709526e-01 5.709526e-01 5.711620e-01
## [4141] 5.713905e-01 5.714476e-01 5.715047e-01 5.731803e-01 5.731803e-01 5.732755e-01 5.732755e-01
## [4150] 5.734087e-01 5.735420e-01 5.735991e-01 5.738276e-01 5.738276e-01 5.738467e-01 5.765123e-01
## [4159] 5.768550e-01 5.768550e-01 5.770263e-01 5.778260e-01 5.780735e-01 5.780735e-01 5.786447e-01
## [4168] 5.791588e-01 5.803583e-01 5.803583e-01 5.803964e-01 5.804535e-01 5.804535e-01 5.804726e-01
## [4177] 5.837665e-01 5.838807e-01 5.838998e-01 5.839188e-01 5.839569e-01 5.841473e-01 5.914587e-01
## [4186] 5.941243e-01 5.941623e-01 5.950572e-01 5.951143e-01 5.952286e-01 5.952667e-01 5.953238e-01
## [4195] 5.959711e-01 5.959711e-01 5.960473e-01 5.960663e-01 5.960663e-01 5.961425e-01 5.962758e-01
## [4204] 5.966185e-01 5.966566e-01 5.966756e-01 5.966947e-01 5.967137e-01 5.967899e-01 5.968089e-01
## [4213] 5.969612e-01 5.970945e-01 5.973039e-01 5.975134e-01 5.975134e-01 5.975705e-01 5.978371e-01
## [4222] 5.979513e-01 5.980084e-01 5.980084e-01 5.980275e-01 5.980465e-01 5.980465e-01 5.981227e-01
## [4231] 5.983321e-01 5.983321e-01 5.990747e-01 5.990937e-01 6.028636e-01 6.033015e-01 6.033206e-01
## [4240] 6.034158e-01 6.034158e-01 6.040631e-01 6.040822e-01 6.042535e-01 6.043297e-01 6.043487e-01
## [4249] 6.049580e-01 6.049580e-01 6.049771e-01 6.062147e-01 6.086899e-01 6.092230e-01 6.092420e-01
## [4258] 6.094515e-01 6.094515e-01 6.094515e-01 6.109747e-01 6.126883e-01 6.138497e-01 6.138687e-01
## [4267] 6.146684e-01 6.167247e-01 6.167438e-01 6.168009e-01 6.210468e-01 6.210659e-01 6.211611e-01
## [4276] 6.223987e-01 6.224177e-01 6.224177e-01 6.224177e-01 6.224367e-01 6.224367e-01 6.224558e-01
## [4285] 6.230079e-01 6.230460e-01 6.230460e-01 6.230460e-01 6.230651e-01 6.231031e-01 6.247025e-01
## [4294] 6.251023e-01 6.251214e-01 6.251404e-01 6.251595e-01 6.251595e-01 6.254260e-01 6.254260e-01
## [4303] 6.254831e-01 6.254831e-01 6.257497e-01 6.260734e-01 6.260924e-01 6.260924e-01 6.261495e-01
## [4312] 6.265684e-01 6.270254e-01 6.270444e-01 6.272158e-01 6.272158e-01 6.272348e-01 6.272539e-01
## [4321] 6.280155e-01 6.280726e-01 6.280916e-01 6.280916e-01 6.282249e-01 6.282630e-01 6.282630e-01
## [4330] 6.288151e-01 6.289484e-01 6.298243e-01 6.299575e-01 6.299766e-01 6.303764e-01 6.303955e-01
## [4339] 6.326041e-01 6.326803e-01 6.328897e-01 6.336513e-01 6.346604e-01 6.350603e-01 6.350603e-01
```

```
## [4348] 6.352507e-01 6.352697e-01 6.352697e-01 6.353649e-01 6.354411e-01 6.354791e-01 6.354982e-01
## [4357] 6.357647e-01 6.378211e-01 6.379543e-01 6.379543e-01 6.381067e-01 6.451324e-01 6.460654e-01
## [4366] 6.488643e-01 6.488643e-01 6.491308e-01 6.492260e-01 6.492641e-01 6.492641e-01 6.492641e-01
## [4375] 6.495307e-01 6.495497e-01 6.495687e-01 6.495878e-01 6.496068e-01 6.496259e-01 6.496830e-01
## [4384] 6.497211e-01 6.497211e-01 6.498353e-01 6.498353e-01 6.499115e-01 6.501590e-01 6.503303e-01
## [4393] 6.504446e-01 6.504446e-01 6.505779e-01 6.506540e-01 6.506731e-01 6.510348e-01 6.511871e-01
## [4402] 6.512062e-01 6.541955e-01 6.541955e-01 6.543478e-01 6.551284e-01 6.552998e-01 6.558329e-01
## [4411] 6.598884e-01 6.610118e-01 6.610118e-01 6.621161e-01 6.621161e-01 6.621923e-01 6.621923e-01
## [4420] 6.628967e-01 6.630110e-01 6.630110e-01 6.643438e-01 6.644390e-01 6.644390e-01 6.644390e-01
## [4429] 6.648007e-01 6.649340e-01 6.660193e-01 6.662097e-01 6.662478e-01 6.663620e-01 6.669523e-01
## [4438] 6.670094e-01 6.670665e-01 6.674663e-01 6.674663e-01 6.688753e-01 6.703414e-01 6.703414e-01
## [4447] 6.735211e-01 6.735782e-01 6.735972e-01 6.736163e-01 6.736163e-01 6.736543e-01 6.736734e-01
## [4456] 6.737876e-01 6.751585e-01 6.752537e-01 6.762819e-01 6.763009e-01 6.763009e-01 6.763009e-01
## [4465] 6.764151e-01 6.765103e-01 6.767388e-01 6.780907e-01 6.781097e-01 6.782049e-01 6.782239e-01
## [4474] 6.783382e-01 6.783572e-01 6.783953e-01 6.783953e-01 6.783953e-01 6.784143e-01 6.784524e-01
## [4483] 6.802612e-01 6.803183e-01 6.803374e-01 6.803755e-01 6.803755e-01 6.803755e-01 6.804135e-01
## [4492] 6.805087e-01 6.805849e-01 6.806039e-01 6.806039e-01 6.808324e-01 6.808324e-01 6.825079e-01
## [4501] 6.825270e-01 6.825841e-01 6.829078e-01 6.829268e-01 6.829268e-01 6.830411e-01 6.830411e-01
## [4510] 6.830601e-01 6.830791e-01 6.855163e-01 6.855163e-01 6.855353e-01 6.855353e-01 6.855353e-01
## [4519] 6.858019e-01 6.858209e-01 6.909427e-01 6.909427e-01 6.909427e-01 6.909617e-01 6.924088e-01
## [4528] 6.932656e-01 6.932656e-01 6.932656e-01 6.932846e-01 6.933036e-01 6.933417e-01 6.936844e-01
## [4537] 6.937225e-01 6.938748e-01 6.971497e-01 6.977590e-01 6.977590e-01 6.977590e-01 6.977971e-01
## [4546] 6.981398e-01 6.981588e-01 6.982350e-01 6.982540e-01 6.987110e-01 6.987110e-01 6.987300e-01
## [4555] 6.987681e-01 7.183412e-01 7.183412e-01 7.208736e-01 7.209307e-01 7.209497e-01 7.212924e-01
## [4564] 7.218446e-01 7.218636e-01 7.219779e-01 7.219779e-01 7.219779e-01 7.220540e-01 7.221112e-01
## [4573] 7.230060e-01 7.233107e-01 7.233488e-01 7.233678e-01 7.233678e-01 7.233868e-01 7.234630e-01
## [4582] 7.239390e-01 7.239390e-01 7.239580e-01 7.239580e-01 7.240723e-01 7.247387e-01 7.266427e-01
## [4591] 7.266998e-01 7.266998e-01 7.267188e-01 7.267188e-01 7.267379e-01 7.296510e-01 7.296700e-01
## [4600] 7.298414e-01 7.298604e-01 7.299176e-01 7.299366e-01 7.299937e-01 7.300128e-01 7.300699e-01
## [4609] 7.311552e-01 7.313265e-01 7.313646e-01 7.314027e-01 7.315550e-01 7.315740e-01 7.318406e-01
## [4618] 7.348680e-01 7.390187e-01 7.390377e-01 7.391329e-01 7.391329e-01 7.391520e-01 7.391710e-01
## [4627] 7.419508e-01 7.419889e-01 7.426744e-01 7.427505e-01 7.429028e-01 7.450544e-01 7.453590e-01
## [4636] 7.462729e-01 7.463110e-01 7.463110e-01 7.464252e-01 7.575256e-01 7.576208e-01 7.576398e-01
## [4645] 7.597913e-01 7.602483e-01 7.602864e-01 7.603244e-01 7.605339e-01 7.605529e-01 7.606291e-01
## [4654] 7.606862e-01 7.607052e-01 7.607052e-01 7.647036e-01 7.647036e-01 7.651416e-01 7.651606e-01
## [4663] 7.666838e-01 7.666838e-01 7.666838e-01 7.667028e-01 7.667219e-01 7.667980e-01 7.676739e-01
## [4672] 7.681308e-01 7.681880e-01 7.682260e-01 7.683212e-01 7.683974e-01 7.684355e-01 7.684926e-01
## [4681] 7.685497e-01 7.685878e-01 7.686068e-01 7.686259e-01 7.686259e-01 7.688734e-01 7.689115e-01
## [4690] 7.690448e-01 7.690448e-01 7.690828e-01 7.693494e-01 7.693684e-01 7.693875e-01 7.695779e-01
## [4699] 7.698635e-01 7.699206e-01 7.699206e-01 7.699206e-01 7.699587e-01 7.699777e-01 7.701491e-01
## [4708] 7.716342e-01 7.716532e-01 7.717865e-01 7.719388e-01 7.719769e-01 7.721102e-01 7.721292e-01
## [4717] 7.732336e-01 7.736334e-01 7.737096e-01 7.737667e-01 7.737857e-01 7.778222e-01 7.778412e-01
## [4726] 7.779174e-01 7.792312e-01 7.793264e-01 7.793264e-01 7.793454e-01 7.793644e-01 7.797072e-01
## [4735] 7.798595e-01 7.798595e-01 7.800308e-01 7.800308e-01 7.800308e-01 7.803164e-01 7.806972e-01
## [4744] 7.823537e-01 7.828678e-01 7.828678e-01 7.828678e-01 7.830201e-01 7.830392e-01 7.830392e-01
## [4753] 7.892843e-01 7.892843e-01 7.893033e-01 7.894176e-01 7.894176e-01 7.894556e-01 7.895318e-01
## [4762] 7.896651e-01 7.896651e-01 7.896841e-01 7.903696e-01 7.906932e-01 7.907504e-01 7.907884e-01
## [4771] 7.911692e-01 7.919499e-01 7.919499e-01 7.920451e-01 7.920641e-01 7.921593e-01 7.922736e-01
## [4780] 7.932827e-01 7.934921e-01 7.935492e-01 7.935492e-01 7.957579e-01 7.957579e-01 7.958340e-01
## [4789] 7.960435e-01 7.960816e-01 7.962529e-01 7.962529e-01 7.962529e-01 7.962910e-01 7.968051e-01
## [4798] 7.979475e-01 7.979665e-01 7.979856e-01 7.979856e-01 7.980617e-01 7.980998e-01 8.109518e-01
## [4807] 8.121894e-01 8.125702e-01 8.149693e-01 8.149883e-01 8.150073e-01 8.201481e-01 8.203385e-01
## [4816] 8.204337e-01 8.215952e-01 8.215952e-01 8.216523e-01 8.216523e-01 8.216713e-01 8.217475e-01
## [4825] 8.221283e-01 8.262029e-01 8.262219e-01 8.262600e-01 8.262790e-01 8.263933e-01 8.264313e-01
## [4834] 8.270216e-01 8.270406e-01 8.270406e-01 8.270787e-01 8.270787e-01 8.274214e-01 8.274214e-01
## [4843] 8.275547e-01 8.275928e-01 8.276118e-01 8.276309e-01 8.276689e-01 8.276880e-01 8.278022e-01
## [4852] 8.313817e-01 8.314198e-01 8.314198e-01 8.314389e-01 8.314769e-01 8.326765e-01 8.326955e-01
## [4861] 8.434721e-01 8.434721e-01 8.434912e-01 8.434912e-01 8.439101e-01 8.439291e-01 8.451667e-01
```

```
## [4870] 8.467661e-01 8.467851e-01 8.468613e-01 8.478894e-01 8.507264e-01 8.510881e-01 8.511072e-01
## [4879] 8.512976e-01 8.513357e-01 8.513357e-01 8.513357e-01 8.520592e-01 8.523638e-01 8.523638e-01
## [4888] 8.524590e-01 8.524590e-01 8.526304e-01 8.526875e-01 8.527637e-01 8.528398e-01 8.530112e-01
## [4897] 8.577141e-01 8.578093e-01 8.578473e-01 8.578473e-01 8.579045e-01 8.579045e-01 8.579045e-01
## [4906] 8.584947e-01 8.586280e-01 8.586470e-01 8.586851e-01 8.586851e-01 8.586851e-01 8.591421e-01
## [4915] 8.591992e-01 8.595609e-01 8.595800e-01 8.597133e-01 8.597133e-01 8.597133e-01 8.643971e-01
## [4924] 8.644733e-01 8.644923e-01 8.645494e-01 8.645494e-01 8.674245e-01 8.674625e-01 8.676339e-01
## [4933] 8.684907e-01 8.685097e-01 8.696331e-01 8.721083e-01 8.721464e-01 8.727176e-01 8.746025e-01
## [4942] 8.746597e-01 8.746597e-01 8.750405e-01 8.750785e-01 8.750785e-01 8.751737e-01 8.751737e-01
## [4951] 8.753832e-01 8.753832e-01 8.753832e-01 8.754022e-01 8.760115e-01 8.765256e-01 8.765256e-01
## [4960] 8.765637e-01 8.789817e-01 8.790960e-01 8.790960e-01 8.791531e-01 8.791531e-01 8.792293e-01
## [4969] 8.792673e-01 8.792673e-01 8.793054e-01 8.793245e-01 8.793245e-01 8.793435e-01 8.793435e-01
## [4978] 8.794577e-01 8.794577e-01 8.794768e-01 8.794958e-01 8.795149e-01 8.795339e-01 8.795529e-01
## [4987] 8.799528e-01 8.799718e-01 8.799909e-01 8.800480e-01 8.801813e-01 8.801813e-01 8.801813e-01
## [4996] 8.804097e-01 8.804097e-01 8.804097e-01 8.804097e-01 8.804097e-01 8.804097e-01 8.804478e-01
## [5005] 8.804669e-01 8.805049e-01 8.806763e-01 8.807334e-01 8.810761e-01 8.810952e-01 8.811333e-01
## [5014] 8.914720e-01 8.914910e-01 8.948040e-01 8.948992e-01 8.948992e-01 8.948992e-01 8.949944e-01
## [5023] 8.984977e-01 8.984977e-01 9.017917e-01 9.017917e-01 9.018107e-01 9.018107e-01 9.019249e-01
## [5032] 9.021725e-01 9.021725e-01 9.022105e-01 9.022296e-01 9.023248e-01 9.023629e-01 9.023819e-01
## [5041] 9.025913e-01 9.026104e-01 9.033529e-01 9.034481e-01 9.034481e-01 9.039051e-01 9.054664e-01
## [5050] 9.094648e-01 9.095029e-01 9.097123e-01 9.107595e-01 9.107976e-01 9.108166e-01 9.108166e-01
## [5059] 9.111403e-01 9.111593e-01 9.121304e-01 9.121304e-01 9.121685e-01 9.123398e-01 9.127587e-01
## [5068] 9.129301e-01 9.129301e-01 9.129681e-01 9.129872e-01 9.130253e-01 9.154624e-01 9.155957e-01
## [5077] 9.198225e-01 9.198416e-01 9.199558e-01 9.200129e-01 9.200320e-01 9.200320e-01 9.200510e-01
## [5086] 9.209840e-01 9.210411e-01 9.210792e-01 9.210792e-01 9.210982e-01 9.210982e-01 9.211363e-01
## [5095] 9.242017e-01 9.242017e-01 9.242398e-01 9.242398e-01 9.242398e-01 9.242398e-01 9.242398e-01
## [5104] 9.242779e-01 9.246016e-01 9.246206e-01 9.246206e-01 9.246587e-01 9.251537e-01 9.251537e-01
## [5113] 9.252299e-01 9.252299e-01 9.252299e-01 9.252489e-01 9.252489e-01 9.252489e-01 9.253251e-01
## [5122] 9.276861e-01 9.277051e-01 9.277241e-01 9.277241e-01 9.294187e-01 9.294187e-01 9.294187e-01
## [5131] 9.297233e-01 9.297424e-01 9.298947e-01 9.300470e-01 9.302945e-01 9.303326e-01 9.303897e-01
## [5140] 9.305040e-01 9.322176e-01 9.322366e-01 9.323128e-01 9.323128e-01 9.323699e-01 9.325032e-01
## [5149] 9.325984e-01 9.326365e-01 9.326365e-01 9.328078e-01 9.328078e-01 9.334933e-01 9.334933e-01
## [5158] 9.335123e-01 9.335504e-01 9.335504e-01 9.335694e-01 9.336646e-01 9.336646e-01 9.337217e-01
## [5167] 9.338741e-01 9.338741e-01 9.341787e-01 9.341787e-01 9.341977e-01 9.441366e-01 9.441366e-01
## [5176] 9.497534e-01 9.497725e-01 9.501723e-01 9.501914e-01 9.524571e-01 9.525714e-01 9.526475e-01
## [5185] 9.536757e-01 9.537518e-01 9.537709e-01 9.538661e-01 9.538661e-01 9.538851e-01 9.539422e-01
## [5194] 9.539803e-01 9.540755e-01 9.547990e-01 9.549133e-01 9.549133e-01 9.549323e-01 9.549704e-01
## [5203] 9.550085e-01 9.563413e-01 9.563413e-01 9.563603e-01 9.563603e-01 9.563603e-01 9.563603e-01
## [5212] 9.563603e-01 9.563984e-01 9.564555e-01 9.564936e-01 9.564936e-01 9.580168e-01 9.602254e-01
## [5221] 9.602445e-01 9.602445e-01 9.602445e-01 9.602445e-01 9.602635e-01 9.604349e-01 9.604349e-01
## [5230] 9.608347e-01 9.608538e-01 9.608538e-01 9.608918e-01 9.608918e-01 9.608918e-01 9.609680e-01
## [5239] 9.633099e-01 9.633861e-01 9.633861e-01 9.633861e-01 9.633861e-01 9.635384e-01 9.635574e-01
## [5248] 9.642429e-01 9.642429e-01 9.643190e-01 9.644904e-01 9.644904e-01 9.644904e-01 9.644904e-01
## [5257] 9.645475e-01 9.645475e-01 9.646808e-01 9.646998e-01 9.646998e-01 9.665467e-01 9.665848e-01
## [5266] 9.668514e-01 9.669846e-01 9.669846e-01 9.670227e-01 9.670798e-01 9.670798e-01 9.689267e-01
## [5275] 9.690219e-01 9.690410e-01 9.743722e-01 9.743912e-01 9.744864e-01 9.745054e-01 9.745245e-01
## [5284] 9.745816e-01 9.746006e-01 9.746387e-01 9.746387e-01 9.747910e-01 9.747910e-01 9.748291e-01
## [5293] 9.749624e-01 9.749814e-01 9.749814e-01 9.751718e-01 9.751909e-01 9.754003e-01 9.754574e-01
## [5302] 9.755146e-01 9.755146e-01 9.755336e-01 9.755526e-01 9.755717e-01 9.756098e-01 9.756288e-01
## [5311] 9.756859e-01 9.756859e-01 9.770187e-01 9.773614e-01 9.773805e-01 9.774566e-01 9.774566e-01
## [5320] 9.779517e-01 9.779707e-01 9.781992e-01 9.782182e-01 9.782182e-01 9.782182e-01 9.782754e-01
## [5329] 9.796272e-01 9.796653e-01 9.796843e-01 9.798176e-01 9.799318e-01 9.799509e-01 9.799509e-01
## [5338] 9.799890e-01 9.800270e-01 9.800270e-01 9.800270e-01 9.800461e-01 9.823499e-01 9.823690e-01
## [5347] 9.830163e-01 9.830163e-01 9.843682e-01 9.844062e-01 9.844253e-01 9.844634e-01 9.845205e-01
## [5356] 9.846157e-01 9.846157e-01 9.849394e-01 9.849774e-01 9.849965e-01 9.849965e-01 9.850346e-01
## [5365] 9.850917e-01 9.851107e-01 9.851678e-01 9.851869e-01 9.851869e-01 9.852630e-01 9.852630e-01
## [5374] 9.853582e-01 9.853963e-01 9.853963e-01 9.853963e-01 9.862722e-01 9.862912e-01 9.863864e-01
## [5383] 9.865006e-01 9.866149e-01 9.866339e-01 9.866530e-01 9.866530e-01 9.866720e-01 9.867101e-01
```

```
## [5392] 9.867672e-01 9.867672e-01 9.867862e-01 9.867862e-01 9.869386e-01 9.869386e-01 9.869386e-01
## [5401] 9.869576e-01 9.881000e-01 9.882142e-01 9.882142e-01 9.882142e-01 9.883666e-01 9.883856e-01
## [5410] 9.884237e-01 9.884618e-01 9.884998e-01 9.885189e-01 9.885760e-01 9.885760e-01 9.885760e-01
## [5419] 9.886902e-01 9.886902e-01 9.887093e-01 9.887093e-01 9.887854e-01 9.888045e-01 9.888235e-01
## [5428] 9.888806e-01 9.888806e-01 9.888806e-01 9.890520e-01 9.890710e-01 9.891091e-01 9.891091e-01
## [5437] 9.892234e-01 9.892424e-01 9.892424e-01 9.892995e-01 9.892995e-01 9.893947e-01 9.894709e-01
## [5446] 9.895470e-01 9.895661e-01 9.895661e-01 9.904990e-01 9.904990e-01 9.905371e-01 9.905562e-01
## [5455] 9.905942e-01 9.905942e-01 9.906133e-01 9.906704e-01 9.907466e-01 9.907656e-01 9.910702e-01
## [5464] 9.911083e-01 9.911464e-01 9.911464e-01 9.911654e-01 9.911845e-01 9.912035e-01 9.913749e-01
## [5473] 9.914320e-01 9.914891e-01 9.915082e-01 9.915082e-01 9.915272e-01 9.916414e-01 9.916605e-01
## [5482] 9.918699e-01 9.918699e-01 9.919461e-01 9.920032e-01 9.920222e-01 9.920413e-01 9.921936e-01
## [5491] 9.923459e-01 9.924602e-01 9.924792e-01 9.924982e-01 9.925363e-01 9.925363e-01 9.925554e-01
## [5500] 9.927077e-01 9.927838e-01 9.929552e-01 9.929742e-01 9.929742e-01 9.930123e-01 9.930314e-01
## [5509] 9.930694e-01 9.931075e-01 9.931266e-01 9.931646e-01 9.933550e-01 9.933931e-01 9.934312e-01
## [5518] 9.935454e-01 9.935835e-01 9.936597e-01 9.936597e-01 9.936978e-01 9.937549e-01 9.937549e-01
## [5527] 9.939643e-01 9.939643e-01 9.939643e-01 9.940405e-01 9.940595e-01 9.940786e-01 9.940786e-01
## [5536] 9.943642e-01 9.944594e-01 9.944594e-01 9.948211e-01 9.948782e-01 9.948973e-01 9.949163e-01
## [5545] 9.949354e-01 9.950306e-01 9.950306e-01 9.950306e-01 9.950496e-01 9.953923e-01 9.954114e-01
## [5554] 9.955446e-01 9.955827e-01 9.955827e-01 9.956018e-01 9.956779e-01 9.956970e-01 9.957160e-01
## [5563] 9.960016e-01 9.960016e-01 9.960397e-01 9.960587e-01 9.960587e-01 9.964966e-01 9.965538e-01
## [5572] 9.966680e-01 9.966680e-01 9.967822e-01 9.967822e-01 9.967822e-01 9.967822e-01 9.968965e-01
## [5581] 9.973154e-01 9.973534e-01 9.973725e-01 9.973915e-01 9.974677e-01 9.974677e-01 9.974677e-01
## [5590] 9.976581e-01 9.976962e-01 9.977152e-01 9.977152e-01 9.977152e-01 9.977723e-01 9.977914e-01
## [5599] 9.980960e-01 9.981531e-01 9.982864e-01 9.983054e-01 9.983626e-01 9.984768e-01 9.984958e-01
## [5608] 9.986482e-01 9.986482e-01 9.986862e-01 9.987053e-01 9.987814e-01 9.988005e-01 9.988195e-01
## [5617] 9.992003e-01 9.993717e-01 9.993717e-01 9.994098e-01 9.994288e-01 9.994669e-01 9.995050e-01
## [5626] 9.995430e-01 9.995811e-01 9.997715e-01 9.997715e-01 9.998477e-01 9.999048e-01 9.999238e-01
## [5635] 9.999429e-01 9.999619e-01 1.000000e+00
```

3.10 Bin test set by prediction

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.2373  0.5427  0.5772  0.5786  0.6122  0.7238
```

Chapter 4

Conclusion

Is the interest rate proposed by LC high enough?

This report was one of those "it-builds-character" endeavour. Facing such a large dataset, it took a very long time before settling to address a particular question. Exploring the data lead to many blind alleys with little interest.

As a practical tool, this approach would not work in real life: we have no data on rejected loans. Using this model to accept/reject loans would need more work (using Reject Inference).

Currently only look at PD. Extend to LGD with Good Dollars and Bad Dollars instead of Good Loans / Bad Loans

Further possible explorations:

- Address the unbalanced dataset by sampling a number of training samples whose size is identical to the test dataset.
- Perform PCA beforehand
- Improve the regularisation of the model parameters.
- Explore economic cycles/situations as additional entry.
- We use the entire dataset to estimate the impact of time as a polynomial curve. To assess future loans, this would not be acceptable. Only an online algorithm should be used. For example ARMA/ARIMA, Kalman filtering of the time-trend trajectory.
- Loss amount where good loan/bad loan proportion is replaced by a good dollar (repaid) / bad dollars (lost) fraction. See SAS Global Forum 2012.
- The size of the dataset is an issue to apply other techinques. But with stochastics methods, possible other models could be:
  - Tree models which have numerous variations (CART generally, and simple or aggregated boosted decision trees specifically)
  - Neural network

Appendix

4.1 List of assumptions / limitations regarding the dataset

As mentioned during this report, we had to make numerous assumptions given the lack of clarity of the variable descriptions.

- *Dataset quality*: Aside from cents rounding issues, the dataset does not contain any flagrant errors that we could see (e.g. minor error of amount or rate, zipcode). Quality of the variable description is a different matter altogether.
- *Ratings*: The day-1 rating is between A1 and (and no lower than) G5. No note is rated lower than E5 after 6 November 2017, and lower than D5 after 30 June 2019.
- *Credit history*: Credit history information for the principal borrower relates to pre-approval and not post-funding. This is clear for the joint applicants, but simply an assumption for the principal borrower.
- *Recoveries*: Recoveries (if any) are assumed to be paid 3 months after the last scheduled payment date (variable `last_pymnt_d`)
- *Survival effect*: The dataset does not include applications that were rejected by the lender (for whatever reason) or by the borrower (for example because the interest rate quote is too high). It may also be the case that some actual loans were excluded as and when the dataset changed over the years.
- LIBOR funding rate: we use the 3-year and 5-year swap rates. In reality, we should have used average tenor-weighted swap rates (i.e. ca. 1.5 Y and 2.5 Y). This requires a full swap curve and more calculation than necessary for our purpose. The principles of this report should not be significantly affted by this approximation.

We do hope that LendingClub investors receive information of much better quality!

4.2 Data preparation and formatting

We used different sources of information:

- The LendingClub dataset made available on Kaggle;
- US geographical data about zip and FIPS codes;
- Market interest rates from the Saint Louis Federal Reserve Bank; and,
- Macro data from the same source.

We here show the code used to prepare the data. It was automatically formatted by *RStudio*.

4.2.1 LendinClub dataset

```
local({
  #
  # STEP 1: Download the dataset
  #
  #   Got to https://www.kaggle.com/wendykan/lending-club-loan-data
  #
  #   Download into the 'datasets' subdirectory
  #   Unzip the file.
  #   WARNING: The unzipping will be about 2.4GB
  #
  #   Name the sql database "datasets/lending_club.sqlite"
  #

  #
  # STEP 2: Prepare the dabase as a tibble
  #

  ##
  ## WARNING: THIS ASSUMES A 'datasets' DIRECTORY WAS CREATED
  ##
  library(RSQLite)
  db_conn <-
    dbConnect(RSQLite::SQLite(), "datasets/lending_club.sqlite")
  dbListTables(db_conn)

  # Returns a 2.96GB data frame
  lending_club <- dbGetQuery(db_conn, "SELECT * FROM loan")
  lending_club <- as_tibble(lending_club)

  # Close the database
  dbDisconnect(db_conn)

  # Compressed to ca.285MB on disk
  saveRDS(lending_club, "datasets/lending_club.rds")


  library(tidyverse)
  library(lubridate)
  library(hablar)

  # Before reformat in case the previous step was already done
  # lending_club <- readRDS("datasets/lending_club.rds")
  #
  # str(lending_club)
  #

  # Leave the original dataset untouched and work with a copy.
  lc <- lending_club

  lc <- lc %>%

    # Add loan identification number to track the loans across calculations
    mutate(loanID = row_number()) %>%

    # Remove useless strings
    mutate(
```

```
    term       = str_remove(term, " months"),
    emp_length = str_replace(emp_length, "<1", "0"),
    emp_length = str_replace(emp_length, "10+", "10"),
    emp_length = str_remove(emp_length, "years")
  ) %>%

  # Creates dates out of strings - Parse errors will be raised when no dates.
  mutate(
    debt_settlement_flag_date = as_date(dmy(
      str_c("1-", debt_settlement_flag_date)
    )),
    earliest_cr_line          = as_date(dmy(str_c(
      "1-", earliest_cr_line
    ))),
    hardship_start_date       = as_date(dmy(str_c(
      "1-", hardship_start_date
    ))),
    hardship_end_date         = as_date(dmy(str_c(
      "1-", hardship_end_date
    ))),
    issue_d                   = as_date(dmy(str_c("1-", issue_d))),
    last_credit_pull_d        = as_date(dmy(str_c(
      "1-", last_credit_pull_d
    ))),
    last_pymnt_d              = as_date(dmy(str_c(
      "1-", last_pymnt_d
    ))),
    next_pymnt_d              = as_date(dmy(str_c(
      "1-", next_pymnt_d
    ))),
    payment_plan_start_date   = as_date(dmy(str_c(
      "1-", payment_plan_start_date
    ))),
    sec_app_earliest_cr_line  = as_date(dmy(str_c(
      "1-", sec_app_earliest_cr_line
    ))),
    settlement_date           = as_date(dmy(str_c(
      "1-", settlement_date
    )))
  ) %>%

  # Bulk type conversion with convert from the `hablar` package
  convert(
    # Strings
    chr(emp_title, title, url, zip_code),

    # Factors
    fct(
      addr_state,
      application_type,
      debt_settlement_flag,
      desc,
      disbursement_method,
      grade,
      hardship_flag,
      hardship_loan_status,
      hardship_reason,
      hardship_status,
```

```
        hardship_type,
        home_ownership,
        id,
        initial_list_status,
        loan_status,
        member_id,
        policy_code,
        purpose,
        pymnt_plan,
        settlement_status,
        sub_grade,
        verification_status,
        verification_status_joint
      ),

      # Integers
      int(
        acc_now_delinq,
        acc_open_past_24mths,
        chargeoff_within_12_mths,
        collections_12_mths_ex_med,
        deferral_term,
        delinq_2yrs,
        emp_length,
        hardship_dpd,
        hardship_length,
        inq_fi,
        inq_last_12m,
        inq_last_6mths,
        mo_sin_old_il_acct,
        mo_sin_old_rev_tl_op,
        mo_sin_rcnt_rev_tl_op,
        mo_sin_rcnt_tl,
        mort_acc,
        mths_since_last_delinq,
        mths_since_last_major_derog,
        mths_since_last_record,
        mths_since_rcnt_il,
        mths_since_recent_bc,
        mths_since_recent_bc_dlq,
        mths_since_recent_inq,
        mths_since_recent_revol_delinq,
        num_accts_ever_120_pd,
        num_actv_bc_tl,
        num_actv_rev_tl,
        num_bc_sats,
        num_bc_tl,
        num_il_tl,
        num_op_rev_tl,
        num_rev_accts,
        num_rev_tl_bal_gt_0,
        num_sats,
        num_tl_120dpd_2m,
        num_tl_30dpd,
        num_tl_90g_dpd_24m,
        num_tl_op_past_12m,
        open_acc,
        open_acc_6m,
```

```
        open_act_il,
        open_il_12m,
        open_il_24m,
        open_rv_12m,
        open_rv_24m,
        sec_app_chargeoff_within_12_mths,
        sec_app_collections_12_mths_ex_med,
        sec_app_inq_last_6mths,
        sec_app_mort_acc,
        sec_app_mths_since_last_major_derog,
        sec_app_num_rev_accts,
        sec_app_open_acc,
        sec_app_open_act_il,
        term
      ),

      # Floating point
      dbl(
        all_util,
        annual_inc,
        annual_inc_joint,
        avg_cur_bal,
        bc_open_to_buy,
        bc_util,
        collection_recovery_fee,
        delinq_amnt,
        dti,
        dti_joint,
        funded_amnt,
        funded_amnt_inv,
        hardship_amount,
        hardship_last_payment_amount,
        hardship_payoff_balance_amount,
        il_util,
        installment,
        int_rate,
        last_pymnt_amnt,
        loan_amnt,
        max_bal_bc,
        orig_projected_additional_accrued_interest,
        out_prncp,
        out_prncp_inv,
        pct_tl_nvr_dlq,
        percent_bc_gt_75,
        pub_rec,
        pub_rec_bankruptcies,
        recoveries,
        revol_bal,
        revol_bal_joint,
        revol_util,
        sec_app_revol_util,
        settlement_amount,
        settlement_percentage,
        tax_liens,
        tot_coll_amt,
        tot_cur_bal,
        tot_hi_cred_lim,
        total_acc,
```

```
        total_bal_ex_mort,
        total_bal_il,
        total_bc_limit,
        total_cu_tl,
        total_il_high_credit_limit,
        total_pymnt,
        total_pymnt_inv,
        total_rec_int,
        total_rec_late_fee,
        total_rec_prncp,
        total_rev_hi_lim
      )
    ) %>%

    # Converts some values to 1/-1 (instead of Boolean)
    mutate(
      pymnt_plan =           if_else(pymnt_plan == "y",           1, -1),
      hardship_flag =        if_else(hardship_flag == "Y",        1, -1),
      debt_settlement_flag = if_else(debt_settlement_flag == "Y", 1, -1)
    ) %>%

    # Some values are percentages
    mutate(
      int_rate = int_rate / 100,
      dti = dti / 100,
      dti_joint = dti_joint / 100,
      revol_util = revol_util / 100,
      il_util = il_util / 100,
      all_util = all_util / 100,
      bc_open_to_buy = bc_util / 100,
      pct_tl_nvr_dlq = pct_tl_nvr_dlq / 100,
      percent_bc_gt_75 = percent_bc_gt_75 / 100,
      sec_app_revol_util = sec_app_revol_util / 100
    ) %>%

    # Create quasi-centered numerical grades out of grade factors with "A" = 3 down to "G" = -3
    mutate(grade_num = 4 - as.integer(grade)) %>%

    # Ditto with sub_grades. "A1" = +3.4, "A3" = +3.0, down to "G3" = -3.0, "G5" = -3.4
    mutate(sub_grade_num = 3.6 - as.integer(sub_grade) / 5) %>%

    # Keep the first 3 digits of the zipcode as numbers
    mutate(zip_code = as.integer(str_sub(zip_code, 1, 3))) %>%

    # order by date
    arrange(issue_d) %>%

    # Remove empty columns
    select(-id, -member_id, -url)

  saveRDS(lc, "datasets/lending_club_reformatted.rds")


  # Select loans which have matured or been terminated
  past_loans <- lc %>%
    filter(
      loan_status %in% c(
        "Charged Off",
```

```
        "Does not meet the credit policy. Status:Charged Off",
        "Does not meet the credit policy. Status:Fully Paid",
        "Fully Paid"
      )
    )

  saveRDS(past_loans, "datasets/lending_club_reformatted_paid.rds")
})
```

4.2.2 Zip codes and FIPS codes

The R package `zipcode` was installed.

```
#
# ZIPCodes dataset.
#

library(zipcode)
data(zipcode)
zips <- zipcode %>%
  as_tibble() %>%
  mutate(zip = as.integer(str_sub(zip, 1, 3)))

saveRDS(zips, "datasets/zips.rds")
```

A csv file containing zip codes, FIPS codes and population information was downloaded from the *Simple Maps* [1] website.

```
local({
  kaggleCodes <- read.csv("datasets/csv/ZIP-COUNTY-FIPS_2017-06.csv")

  kaggleCodes <-
    kaggleCodes %>%
    as_tibble() %>%
    mutate(zip = floor(ZIP/100),
           FIPS = STCOUNTYFP,
           COUNTYNAME = str_replace(COUNTYNAME, pattern = "County", replacement = ""),
           COUNTYNAME = str_replace(COUNTYNAME, pattern = "Borough", replacement = ""),
           COUNTYNAME = str_replace(COUNTYNAME, pattern = "Municipio", replacement = ""),
           COUNTYNAME = str_replace(COUNTYNAME, pattern = "Parish", replacement = ""),
           COUNTYNAME = str_replace(COUNTYNAME, pattern = "Census Area", replacement = "")) %>%
    rename(county = COUNTYNAME) %>%
    select(zip, county, FIPS) %>%
    arrange(zip)

  saveRDS(zipfips, "datasets/kaggleCodes.rds")
})
```

4.2.3 Market interest rates

Market interest rates (3-year and 5-year swap rates) were download from the Saint Louis Federal Reserve Bank. Datasets are split between before and after the LIBOR fixing scandal. The datasets are merged with disctinct dates.

Download sources are:

- Pre-LIBOR 3-y swap https://fred.stlouisfed.org/series/DSWP3

[1] https://simplemaps.com/data/us-zips

- Post-LIBOR 3-y swap https://fred.stlouisfed.org/series/ICERATES1100USD3Y
- Pre-LIBOR 5-y swap https://fred.stlouisfed.org/series/MSWP5
- Post-LIBOR 5-y swap https://fred.stlouisfed.org/series/ICERATES1100USD5Y

```
local({
  LIBOR3Y <- read.csv("datasets/csv/DSWP3.csv") %>%
    as_tibble() %>%
    filter(DSWP3 != ".") %>%
    mutate(DATE = as_date(DATE),
           RATE3Y = as.numeric(as.character(DSWP3)) / 100) %>%
    select(DATE, RATE3Y)

  ICE3Y <- read.csv("datasets/csv/ICERATES1100USD3Y.csv") %>%
    as_tibble() %>%
    filter(ICERATES1100USD3Y != ".") %>%
    mutate(DATE = as_date(DATE),
           RATE3Y = as.numeric(as.character(ICERATES1100USD3Y)) / 100) %>%
    select(DATE, RATE3Y)


  LIBOR5Y <- read.csv("datasets/csv/DSWP5.csv") %>%
    as_tibble() %>%
    filter(DSWP5 != ".") %>%
    mutate(DATE = as_date(DATE),
           RATE5Y = as.numeric(as.character(DSWP5)) / 100) %>%
    select(DATE, RATE5Y)

  ICE5Y <- read.csv("datasets/csv/ICERATES1100USD5Y.csv") %>%
    as_tibble() %>%
    filter(ICERATES1100USD5Y != ".") %>%
    mutate(DATE = as_date(DATE),
           RATE5Y = as.numeric(as.character(ICERATES1100USD5Y)) / 100) %>%
    select(DATE, RATE5Y)

  RATES3Y <- LIBOR3Y %>% rbind(ICE3Y) %>%
    arrange(DATE) %>% distinct(DATE, .keep_all = TRUE)

  RATES5Y <- LIBOR5Y %>% rbind(ICE5Y) %>%
    arrange(DATE) %>% distinct(DATE, .keep_all = TRUE)

  saveRDS(RATES3Y, "datasets/rates3Y.rds")
  saveRDS(RATES5Y, "datasets/rates5Y.rds")


  # Note there are 7212 days from 1 Jan 2000 to 30 Sep 2019
  #
  # (ymd("2000-01-01") %--% ymd("2019-09-30")) %/% days(1)
  RATES <- tibble(n = seq(0, 7212)) %>%

    # Create a column with all dates
    mutate(DATE = ymd("2000-01-01") + days(n)) %>%
    select(-n) %>%

    # Add all daily 3- then 5-year rates and fill missing down
    left_join(RATES3Y) %>%
    fill(RATE3Y, .direction = "down") %>%

```

```
    left_join(RATES5Y) %>%
    fill(RATE5Y, .direction = "down")

  saveRDS(RATES, "datasets/rates.rds")
})
```

4.2.4 Macro-economical data

Macro-economical datasets were sourced from the same website as Microsoft Excel files. They were converted as-is to tab-separated csv files with LibreOffice.

- Median income per household: https://geofred.stlouisfed.org/map/?th=pubugn&cc=5&rc=false&im=fractile&sb&lng=-112.41&lat=44.31&zm=4&sl&sv&am=Average&at=Not%20Seasonally%20Adjusted,%20Annual,%20Dollars&sti=2022&fq=Annual&rt=county&un=lin&dt=2017-01-01
- Per capita personal income: https://geofred.stlouisfed.org/map/?th=pubugn&cc=5&rc=false&im=fractile&sb&lng=-112.41&lat=44.31&zm=4&sl&sv&am=Average&at=Not%20Seasonally%20Adjusted,%20Annual,%20Dollars&sti=882&fq=Annual&rt=county&un=lin&dt=2017-01-01
- Unemployment: https://geofred.stlouisfed.org/map/?th=rdpu&cc=5&rc=false&im=fractile&sb&lng=-90&lat=40&zm=4&sl&sv&am=Average&at=Not%20Seasonally%20Adjusted,%20Monthly,%20Percent&sti=1224&fq=Monthly&rt=county&un=lin&dt=2019-08-01

```
local({
  ##############################################################################################################
  ##
  ## Median income per household by FIPS from 2002 to 2017
  ##
  # Prepare median income
  medianIncome <-
    # Load the dataset after dropping the first line
    read.csv(
      "datasets/csv/GeoFRED_Estimate_of_Median_Household_Income_by_County_Dollars.csv",
      sep = "\t",
      skip = 1,
      stringsAsFactors = FALSE
    ) %>%

    # Drops columnsn containing a unique identifier and the FIPS name
    select(-"Series.ID", -"Region.Name") %>%

    # Rename the relevant column to 'FIPS'
    rename(FIPS = "Region.Code") %>%

    # Order by FIPS
    arrange(FIPS) %>%

    # Convert to a 'long' table, i.e. one column for FIPS, one for date, one for income
    pivot_longer(cols = starts_with("X"),
                 names_to = "Date",
                 values_to = "medianIncome") %>%

    # Create actual dates
    mutate(Date = str_replace(Date, "[X]", ""),
```

```
          Date = ymd(str_c(Date, "-12-31")))

saveRDS(medianIncome, "datasets/medianincome.rds")



################################################################################################################
##
## Per capita income by FIPS from 2002 to 2017
##
personalIncome <-
  # Load the dataset after dropping the first line
  read.csv(
    "datasets/csv/GeoFRED_Per_Capita_Personal_Income_by_County_Dollars.csv",
    sep = "\t",
    skip = 1,
    stringsAsFactors = FALSE
  ) %>%

  # Drops columnsn containing a unique identifier and the FIPS name
  select(-"Series.ID", -"Region.Name") %>%

  # Rename the relevant column to 'FIPS'
  rename(FIPS = "Region.Code") %>%

  # Order by FIPS
  arrange(FIPS) %>%

  # Convert to a 'long' table, i.e. one column for FIPS, one for date, one for income
  pivot_longer(cols = starts_with("X"),
               names_to = "Date",
               values_to = "personalIncome") %>%

  # Create actual dates
  mutate(Date = str_replace(Date, "[X]", ""),
         Date = ymd(str_c(Date, "-12-31")))

saveRDS(personalIncome, "datasets/personalincome.rds")


################################################################################################################
##
## Unemplyment rate monthly by FIPS from January 2000 to August 2019
##
unemploymentRate <-
  # Load the dataset after dropping the first line
  read.csv(
    "datasets/csv/GeoFRED_Unemployment_Rate_by_County_Percent.csv",
    sep = "\t",
    skip = 1,
    stringsAsFactors = FALSE
  ) %>%

  # Drops columnsn containing a unique identifier and the FIPS name
  select(-"Series.ID", -"Region.Name") %>%

  # Rename the relevant column to 'FIPS'
  rename(FIPS = "Region.Code") %>%
```

```
    # Order by FIPS
    arrange(FIPS) %>%

    mutate_all(as.double) %>%

    # Convert to a 'long' table, i.e. one column for FIPS, one for date, one for income
    pivot_longer(cols = starts_with("X"),
                 names_to = "Date",
                 values_to = "unemploymentRate",
                 values_ptypes = c("unemploymentRate", numeric)) %>%

    # Converts the content of the Year column to an actual date
    mutate(
      Date = str_replace(Date, "[X]", ""),
      Date = str_replace(Date, "[.]", "-"),
      Date = ymd(str_c(Date, "-1"))
    )

  saveRDS(unemploymentRate, "datasets/unemployment.rds")
})
```

4.3 List of variables

This table presents the list of variables provided in the original dataset. The descriptions come from a spreadsheet attached with the dataset and, unfortunately, are not extremely precise and subject to interpretation. We added comments and/or particular interpretations in *CAPITAL LETTERS*.

Table 4.1: Description of the dataset variables as provided in the dataset downloaded from Kaggle

Variable Name	Description
loanID	NOTE THIS IS NOT AN ORIGINAL VARIABLE. IT WAS ADDED FOR THE PURPOSE OF TRACKING LOANS INDIVIDUALLY AS AND WHEN NEEDED.
loan_amnt	The listed amount of the loan applied for by the borrower. If at some point in time, the credit department reduces the loan amount, then it will be reflected in this value.
funded_amnt	The total amount committed to that loan at that point in time.
funded_amnt_inv	The total amount committed by investors for that loan at that point in time.
term	The number of payments on the loan. Values are in months and can be either 36 or 60.
int_rate	Interest Rate on the loan
installment	The monthly payment owed by the borrower if the loan originates.
grade	LC assigned loan grade
sub_grade	LC assigned loan subgrade
emp_title	The job title supplied by the Borrower when applying for the loan.

Table 4.1: Description of the dataset variables as provided in the dataset downloaded from Kaggle *(continued)*

Variable Name	Description
emp_length	Employment length in years. Possible values are between 0 and 10 where 0 means less than one year and 10 means ten or more years.
home_ownership	The home ownership status provided by the borrower during registration or obtained from the credit report. Our values are: RENT, OWN, MORTGAGE, OTHER, NONE
annual_inc	The self-reported annual income provided by the borrower during registration. NOT USED AS A VARIABLE SINCE JOINT INCOME ALREADY INCLUDES IT.
verification_status	Indicates if income was verified by LC, not verified, or if the income source was verified
issue_d	The month which the loan was funded
loan_status	Current status of the loan
pymnt_plan	Indicates if a payment plan has been put in place for the loan
url	URL for the LC page with listing data.
desc	Loan description provided by the borrower
purpose	A category provided by the borrower for the loan request.
title	The loan title provided by the borrower
zip_code	The first 3 numbers of the zip code provided by the borrower in the loan application.
addr_state	The state provided by the borrower in the loan application
dti	A ratio calculated using the borrower s total monthly debt payments on the total debt obligations, excluding mortgage and the requested LC loan, divided by the borrower s self-reported monthly income. NOT USED AS A VARIABLE. ONLY USE JOINT DTI.
delinq_2yrs	The number of 30+ days past-due incidences of delinquency in the borrower s credit file for the past 2 years
earliest_cr_line	The month the borrower s earliest reported credit line was opened
inq_last_6mths	The number of inquiries in past 6 months (excluding auto and mortgage inquiries)
mths_since_last_delinq	The number of months since the borrower s last delinquency.
mths_since_last_record	The number of months since the last public record.
open_acc	The number of open credit lines in the borrower s credit file.
pub_rec	Number of derogatory public records
revol_bal	Total credit revolving balance
revol_util	Revolving line utilization rate, or the amount of credit the borrower is using relative to all available revolving credit.

Table 4.1: Description of the dataset variables as provided in the dataset downloaded from Kaggle *(continued)*

Variable Name	Description
total_acc	The total number of credit lines currently in the borrower s credit file
initial_list_status	The initial listing status of the loan. Possible values are – W, F
out_prncp	Remaining outstanding principal for total amount funded. NOTE ONCE A LOAN IS REPAID OR CHARGED OFF, THIS AMOUNT BECOMES 0.
out_prncp_inv	Remaining outstanding principal for portion of total amount funded by investors. NOTE ONCE A LOAN IS REPAID OR CHARGED OFF, THIS AMOUNT BECOMES 0.
total_pymnt	Payments received to date for total amount funded
total_pymnt_inv	Payments received to date for portion of total amount funded by investors
total_rec_prncp	Principal received to date. NOTE THIS AMOUNT WILL SHOW WHETHER A BORROWER DID NOT REPAY IN FULL
total_rec_int	Interest received to date
total_rec_late_fee	Late fees received to date
recoveries	Post charge off gross recovery. NOTE IF A LOAN IS REPAID, THIS AMOUNT IS 0.
collection_recovery_fee	Post charge off collection fee
last_pymnt_d	Last month payment was received
last_pymnt_amnt	Last total payment amount received
next_pymnt_d	Next scheduled payment date
last_credit_pull_d	The most recent month LC pulled credit for this loan
collections_12_mths_ex_med	Number of collections in 12 months excluding medical collections
mths_since_last_major_derog	Months since most recent 90-day or worse rating
policy_code	Publicly available policy_code=1 / New products not publicly available policy_code=2
application_type	Indicates whether the loan is an individual application or a joint application with two coborrowers
annual_inc_joint	The combined self-reported annual income provided by the coborrowers during registration
dti_joint	A ratio calculated using the coborrowers total monthly payments on the total debt obligations, excluding mortgages and the requested LC loan, divided by the coborrowers combined self-reported monthly income
verification_status_joint	Indicates if income was verified by LC, not verified, or if the income source was verified
acc_now_delinq	The number of accounts on which the borrower is now delinquent.
tot_coll_amt	Total collection amounts ever owed

Table 4.1: Description of the dataset variables as provided in the dataset downloaded from Kaggle *(continued)*

Variable Name	Description
tot_cur_bal	Total current balance of all accounts
open_acc_6m	Number of open trades in last 6 months
open_act_il	Number of currently active installment trades
open_il_12m	Number of installment accounts opened in past 12 months
open_il_24m	Number of installment accounts opened in past 24 months
mths_since_rcnt_il	Months since most recent instalment accounts opened
total_bal_il	Total current balance of all installment accounts
il_util	Ratio of total current balance to high credit/credit limit on all install acct
open_rv_12m	Number of revolving trades opened in past 12 months
open_rv_24m	Number of revolving trades opened in past 24 months
max_bal_bc	Maximum current balance owed on all revolving accounts
all_util	Balance to credit limit on all trades
total_rev_hi_lim	Total revolving high credit/credit limit
inq_fi	Number of personal finance inquiries
total_cu_tl	Number of finance trades
inq_last_12m	Number of credit inquiries in past 12 months
acc_open_past_24mths	Number of trades opened in past 24 months.
avg_cur_bal	Average current balance of all accounts
bc_open_to_buy	Total open to buy on revolving bankcards.
bc_util	Ratio of total current balance to high credit/credit limit for all bankcard accounts.
chargeoff_within_12_mths	Number of charge-offs within 12 months
delinq_amnt	The past-due amount owed for the accounts on which the borrower is now delinquent.
mo_sin_old_il_acct	Months since oldest bank instalment account opened
mo_sin_old_rev_tl_op	Months since oldest revolving account opened
mo_sin_rcnt_rev_tl_op	Months since most recent revolving account opened
mo_sin_rcnt_tl	Months since most recent account opened
mort_acc	Number of mortgage accounts.
mths_since_recent_bc	Months since most recent bankcard account opened.
mths_since_recent_bc_dlq	Months since most recent bankcard delinquency
mths_since_recent_inq	Months since most recent inquiry.
mths_since_recent_revol_delinq	Months since most recent revolving delinquency.
num_accts_ever_120_pd	Number of accounts ever 120 or more days past due
num_actv_bc_tl	Number of currently active bankcard accounts
num_actv_rev_tl	Number of currently active revolving trades

Table 4.1: Description of the dataset variables as provided in the dataset downloaded from Kaggle *(continued)*

Variable Name	Description
num_bc_sats	Number of satisfactory bankcard accounts
num_bc_tl	Number of bankcard accounts
num_il_tl	Number of installment accounts
num_op_rev_tl	Number of open revolving accounts
num_rev_accts	Number of revolving accounts
num_rev_tl_bal_gt_0	Number of revolving trades with balance >0
num_sats	Number of satisfactory accounts
num_tl_120dpd_2m	Number of accounts currently 120 days past due (updated in past 2 months)
num_tl_30dpd	Number of accounts currently 30 days past due (updated in past 2 months)
num_tl_90g_dpd_24m	Number of accounts 90 or more days past due in last 24 months
num_tl_op_past_12m	Number of accounts opened in past 12 months
pct_tl_nvr_dlq	Percent of trades never delinquent
percent_bc_gt_75	Percentage of all bankcard accounts > 75% of limit.
pub_rec_bankruptcies	Number of public record bankruptcies
tax_liens	Number of tax liens
tot_hi_cred_lim	Total high credit/credit limit
total_bal_ex_mort	Total credit balance excluding mortgage
total_bc_limit	Total bankcard high credit/credit limit
total_il_high_credit_limit	Total installment high credit/credit limit
revol_bal_joint	Total credit revolving balance
sec_app_earliest_cr_line	Earliest credit line at time of application for the secondary applicant. VARIABLE NOT USED. WE RELY ON THE MAIN BORROWER IN THE FIRST INSTANCE.
sec_app_inq_last_6mths	Credit inquiries in the last 6 months at time of application for the secondary applicant. VARIABLE NOT USED. WE RELY ON THE MAIN BORROWER IN THE FIRST INSTANCE.
sec_app_mort_acc	Number of mortgage accounts at time of application for the secondary applicant. VARIABLE NOT USED. WE RELY ON THE MAIN BORROWER IN THE FIRST INSTANCE.
sec_app_open_acc	Number of open trades at time of application for the secondary applicant. VARIABLE NOT USED. WE RELY ON THE MAIN BORROWER IN THE FIRST INSTANCE.
sec_app_revol_util	Ratio of total current balance to high credit/credit limit for all revolving accounts. VARIABLE NOT USED. WE RELY ON THE MAIN BORROWER IN THE FIRST INSTANCE.

Table 4.1: Description of the dataset variables as provided in the dataset downloaded from Kaggle *(continued)*

Variable Name	Description
sec_app_open_act_il	Number of currently active installment trades at time of application for the secondary applicant. VARIABLE NOT USED. WE RELY ON THE MAIN BORROWER IN THE FIRST INSTANCE.
sec_app_num_rev_accts	Number of revolving accounts at time of application for the secondary applicant. VARIABLE NOT USED. WE RELY ON THE MAIN BORROWER IN THE FIRST INSTANCE.
sec_app_chargeoff_within_12_mths	Number of charge-offs within last 12 months at time of application for the secondary applicant. VARIABLE NOT USED. WE RELY ON THE MAIN BORROWER IN THE FIRST INSTANCE.
sec_app_collections_12_mths_ex_med	Number of collections within last 12 months excluding medical collections at time of application for the secondary applicant. VARIABLE NOT USED. WE RELY ON THE MAIN BORROWER IN THE FIRST INSTANCE.
sec_app_mths_since_last_major_derog	Months since most recent 90-day or worse rating at time of application for the secondary applicant. VARIABLE NOT USED. WE RELY ON THE MAIN BORROWER IN THE FIRST INSTANCE.
hardship_flag	Flags whether or not the borrower is on a hardship plan
hardship_type	Describes the hardship plan offering
hardship_reason	Describes the reason the hardship plan was offered
hardship_status	Describes if the hardship plan is active, pending, cancelled, completed, or broken
deferral_term	Amount of months that the borrower is expected to pay less than the contractual monthly payment amount due to a hardship plan
hardship_amount	The interest payment that the borrower has committed to make each month while they are on a hardship plan
hardship_start_date	The start date of the hardship plan period
hardship_end_date	The end date of the hardship plan period
payment_plan_start_date	The day the first hardship plan payment is due. For example, if a borrower has a hardship plan period of 3 months, the start date is the start of the three-month period in which the borrower is allowed to make interest-only payments.
hardship_length	The number of months the borrower will make smaller payments than normally obligated due to a hardship plan

Table 4.1: Description of the dataset variables as provided in the dataset downloaded from Kaggle *(continued)*

Variable Name	Description
hardship_dpd	Account days past due as of the hardship plan start date
hardship_loan_status	Loan Status as of the hardship plan start date
orig_projected_additional_accrued_interest	The original projected additional interest amount that will accrue for the given hardship payment plan as of the Hardship Start Date. This field will be null if the borrower has broken their hardship payment plan.
hardship_payoff_balance_amount	The payoff balance amount as of the hardship plan start date
hardship_last_payment_amount	The last payment amount as of the hardship plan start date
disbursement_method	The method by which the borrower receives their loan. Possible values are: CASH, DIRECT_PAY
debt_settlement_flag	Flags whether or not the borrower, who has charged-off, is working with a debt-settlement company.
debt_settlement_flag_date	The most recent date that the Debt_Settlement_Flag has been set
settlement_status	The status of the borrower's settlement plan. Possible values are: COMPLETE, ACTIVE, BROKEN, CANCELLED, DENIED, DRAFT
settlement_date	The date that the borrower agrees to the settlement plan
settlement_amount	The loan amount that the borrower has agreed to settle for
settlement_percentage	The settlement amount as a percentage of the payoff balance amount on the loan
settlement_term	The number of months that the borrower will be on the settlement plan

4.4 Calculations of the internal rate of returns and month-to-default

The following shows two versions of the same code. The R version is provided because of the description of the assignment. However, the R version takes just under a full day to run. A Julia version, which is a direct translation of the R code, runs in about 150s (ca. 500x faster).

4.4.1 R code

```
#############################################################################################################
#
# Given some numerical parameters describing a loan in the dataset, returns its Internal Rate
# of Return.
#
# In the first instance, the function creates a schedule of payments.
# In many cases, the schedule will be extremely simple: a series of 36 or 60 equal instalements.
#
# But in some cases, a loan repayment are accelerated. Therefore the total amount of interest will
```

```r
# be lower than expected (but this is good for the investor because highe interest rate over
# shorter tenor.).
#
# In other cases, the borrower defaults. Overall payments are less than expected.
#
# Based on the limited information of the dataset, the function makes educated guesses on the exact
#
# WARNING: THIS IS NOT OPTIMISED. RUNNING THIS FOR ALL LOANS (1.3 MLN OF THEM) TAKES CA.20 HOURS !!!
#
calculateIRR <- function(loanNumber = 1,
                         loan = 1000,
                         intRate = 0.02,
                         term = 36,
                         totalPaid = 1000,
                         totalPrincipalPaid,
                         totalInterestPaid,
                         recoveries = 0,
                         lateFees = 0,
                         showSchedule = FALSE) {
  require(tidyverse)

  # number of monthly payments.
  # It exceeds 60 months in case recoveries on a 60-month loan takes the schedule after 60 months.
  nMonths <- 90

  # Months after which a loan defaults (normal tenor if no default or early prepayment)
  monthDefault = term

  # Note: *100 /100 to calculate in cent because ceiling cannot specify significant digits.
  installment <-
    ceiling(100 * loan * intRate / 12 / (1 - 1 / (1 + intRate / 12) ^ term)) / 100

  # We create a schedule
  schedule <- tibble(
    month = 0:nMonths,
    monthlyPayment = 0.0,
    totalPandI = 0.0,
    totalI = 0.0,
    totalP = 0.0
  )

  for (i in 2:(nMonths + 2)) {
    # Get situation at the end of previous month
    previousTotalPandI <- as.numeric(schedule[i - 1, "totalPandI"])
    previousTotalP     <- as.numeric(schedule[i - 1, "totalP"])
    previousTotalI     <- as.numeric(schedule[i - 1, "totalI"])

    # This is the beginning of a new month. First and foremost, the borrower is expected to pay the
    # accrued interest on amount of principal outstanding.
    # ceiling doesn't seem accept to accept significative digits.
    accruedInterest <-
      ceiling(100 * (loan - previousTotalP) * intRate / 12) / 100

    # If that amount takes the schedule above the total amount of interest shown in the data set,
    # we should stop the schedule at this point
    if (previousTotalI + accruedInterest > totalInterestPaid) {
      # We stop the normal schedule at this date.
      # Interest is paid (although less than scheduled)
```

```
      schedule[i, "monthlyPayment"] <-
        totalInterestPaid - previousTotalI

      # As well as whatever principal is left as per the dataset
      schedule[i, "monthlyPayment"] <-
        schedule[i, "monthlyPayment"] + totalPrincipalPaid - previousTotalP

      # Then 3-month after the last payment date, recoveries and and late fees are paid
      schedule[i + 3, "monthlyPayment"] <-
        schedule[i + 3, "monthlyPayment"] + recoveries + lateFees

      # Not really useful, but for completeness
      schedule[i, "totalPandI"] <- totalPaid
      schedule[i, "totalI"]     <- totalInterestPaid
      schedule[i, "totalP"]     <- totalPrincipalPaid

      # If total principal paid is less than borrower, then it is a default, and the monthDefault
      # is adjusted.
      if (totalPrincipalPaid < loan) {
        monthDefault = i
      }

      # No more payments to add to the schedule
      break()

    } else {
      # Deal with normal schedule
      schedule[i, "monthlyPayment"] <- installment
      schedule[i, "totalPandI"] <-
        schedule[i - 1, "totalPandI"] + installment
      schedule[i, "totalI"]     <-
        schedule[i - 1, "totalI"]   + accruedInterest
      schedule[i, "totalP"]     <-
        schedule[i - 1, "totalP"]   + installment - accruedInterest
    }
  }

  # At this point schedule[, "monthlyPayment"] contains the schedule of all payments, but needs to
  # include the initial loan.
  schedule[1, "monthlyPayment"] <- -loan

  if (showSchedule) {
    schedule %>% view()
  }

  NPV <- function(interest, cashFlow) {
    t = 0:(length(cashFlow) - 1)
    sum(cashFlow / (1 + interest) ^ t)
  }

  IRR <- function(CF) {
    res <- NA
    try({
      res <- uniroot(NPV, c(-0.9, 1), cashFlow = CF)$root
    },
    silent = TRUE)
    return(res)
  }
```

```

  return(tibble(
    loanID = loanNumber,
    IRR = round(as.numeric(IRR(
      schedule$monthlyPayment
    ) * 12), digits = 4),
    monthDefault = monthDefault
  ))
}

loanNumberIRR <- function(loanNumber) {
  require(tidyverse)

  l <- loans %>% filter(loanID == loanNumber)
  calculateIRR(
    loanNumber = l$loanID,
    loan = l$funded_amnt, intRate = l$int_rate, term = l$term,
    totalPaid = l$total_pymnt, totalPrincipalPaid = l$total_rec_prncp, totalInterestPaid = l$total_r
    recoveries = l$recoveries, lateFees = l$total_rec_late_fee,
    showSchedule = TRUE
  )
}
```

```
#
# Calculate all the IRRs and month of default for all the loans.
# WARNING: This takes around a full day to run!!!!
#
# The actual data was generated by the Julia version, with cross-checks.
# Julia version takes about 150 sec on the same unoptimised code.
#
local({
  loansIRR <-
    loans %>%
    rowwise() %>%
    do(
      calculateIRR(
        loanNumber = .$loanID,
        loan = .$funded_amnt,
        intRate = .$int_rate,
        term = .$term,
        totalPaid = .$total_pymnt,
        totalPrincipalPaid = .$total_rec_prncp,
        totalInterestPaid = .$total_rec_int,
        recoveries = .$recoveries,
        lateFees = .$total_rec_late_fee
      )
    )

  saveRDS(loansIRR, "datasets/lending_club_IRRs.rds")

})
```

4.4.2 Julia code

4.4.2.1 Internal Rate of Return

```
##############################################################################################
##
```

```
## Prepare datasets
##

## Previously saved from R with:
##     lending_club <- readRDS("lending_club.rds"); write.csv(lending_club, "lending_club.rds")
##
## WARNING: 1.7GB on disk
##
using CSV
lendingClub = CSV.read("datasets/lending_club.csv"; delim = ",")



#############################################################################################################
##
## IRR calculations
##
## Given some numerical parameters describing a loan in the dataset, returns its Internal Rate
## of Return.
##
## In the first instance, the function creates a schedule of payments.
## In many cases, the schedule will be extremely simple: a series of 36 or 60 equal instalements.
##
## But in some cases, a loan repayment are accelerated. Therefore the total amount of interest will
## be lower than expected (but this is good for the investor because highe interest rate over
## shorter tenor.).
##
## In other cases, the borrower defaults. Overall payments are less than expected.
##
## Based on the limited information of the dataset, the function makes educated guesses on the exact
## schedule.
##
using DataFrames, Roots

function calculateIRR(; loanNumber = 1, loan = 0.0, intRate = 0.0, term = 36,
  totalPaid = 0.0, totalPrincipalPaid = 0.0, totalInterestPaid = 0.0,
  recoveries = 0.0, lateFees = 0.0,
  showSchedule = false)

  # number of monthly payments.
  # It exceeds 60 months in case recoveries on a 60-month loan takes the schedule after 60 months.
  nMonths = 90

  # Months after which a loan defaults (normal tenor if no default or early prepayment)
  monthDefault = term

  # Note: *100 /100 to calculate in cent because ceiling cannot specify significant digits.
  installment = ceil(loan * intRate / 12 / (1 - 1 / (1 + intRate / 12) ^ term), digits = 2)

  # We create a schedule
  schedule = DataFrame(month = 0:nMonths, monthlyPayment = 0.0,
                       totalPandI = 0.0, totalI = 0.0, totalP = 0.0)

  for i in 2:(nMonths + 1)
    # Get situation at the end of previous month
    previousTotalPandI = schedule[i - 1, :totalPandI]
    previousTotalP     = schedule[i - 1, :totalP]
    previousTotalI     = schedule[i - 1, :totalI]
```

```
61
62      # This is the beginning of a new month. First and foremost, the borrower is expected to pay the
63      # accrued interest on amount of principal outstanding.
64      # The installment is expected to cover that amount of interest and the rest goes to
65      # reducing the principal due outstanding.
66      accruedInterest = ceil((loan - previousTotalP) * intRate / 12; digits = 2)
67      decreasePrincipal = installment - accruedInterest
68
69      # If that amount takes the schedule above the total amount of interest shown in the data set,
70      # we should stop the schedule at this point
71      # This is a shortcut since we could have a payment higher than the interest due, but not enough
72      # to cover the expected principal repayment. However, it works well in practice.
73      if previousTotalI + accruedInterest > totalInterestPaid
74
75        # We stop the normal schedule at this date.
76        # Interest is paid (although less than scheduled)
77        schedule[i, :monthlyPayment] = totalInterestPaid - previousTotalI
78
79        # As well as whatever principal is left as per the dataset
80        schedule[i, :monthlyPayment] = schedule[i, :monthlyPayment] + totalPrincipalPaid - previousTota
81
82        # Then 3-month after the last payment date, recoveries and and later fees are paid
83        schedule[i + 3, :monthlyPayment] = schedule[i + 3, :monthlyPayment] + recoveries + lateFees
84
85        # Not really useful, but for completeness
86        schedule[i, :totalPandI] = totalPaid
87        schedule[i, :totalI]     = totalInterestPaid
88        schedule[i, :totalP]     = totalPrincipalPaid
89
90        # If total principal paid is less than borrower, then it is a default, and the monthDefault
91        # is adjusted.
92        if (totalPrincipalPaid < loan)
93          monthDefault = i
94        end
95
96        # No more payments to add to the schedule
97        break
98
99      else
100       # Deal with normal schedule
101       schedule[i, :monthlyPayment] = installment
102       schedule[i, :totalPandI]     = schedule[i - 1, :totalPandI] + installment
103       schedule[i, :totalI]         = schedule[i - 1, :totalI]     + accruedInterest
104       schedule[i, :totalP]         = schedule[i - 1, :totalP]     + installment - accruedInterest
105     end
106   end
107
108   # At this point schedule[, :monthlyPayment] contains the schedule of all payments, but needs to
109   # include the initial loan.
110   schedule[1, :monthlyPayment] = -loan
111
112   if (showSchedule)
113     print(schedule)
114   end
115
116   cashFlow = schedule[:,:monthlyPayment]
117
118   ##
```

```
  ## Finding the IRR is equivalent to finding the root such that the NPV of the cash flow is zero.
  ## Julia has a function (see below) called `find_zero` to do that which requires a function to
  ## be zeroed. This helper function is defined as NPV.
  ##
  function NPV(interest)
    t = 0:(length(cashFlow) - 1)

    ## If you are new to Julia, note the dot before the operation. This indicates that the
    ## operation has to be done element-wise (called `broadcasting` in Julia-ese).
    ## Otherwise, Julia would try to divide one vector by another vector, which makes no sense.
    ## This is also exactly the approach taken in Matlab/Octave.
    return sum(cashFlow ./ (1 + interest) .^ t)
  end

  ## Finds the root, catching any problems which would instead return the R equivalent of NA
  rootInterest = try
                   round(12 * find_zero(NPV, (-0.9, 1.0), Bisection(); xatol = 0.000001); digits = 4)
                 catch e
                   NaN
                 end

  return((
    loanID = loanNumber,
    IRR = rootInterest,
    monthDefault = monthDefault
  ))
end


##
## Calculate the IRR and repayment schedule of a particular loan identified by its loanID
function loanNumberIRR(loanNumber)
  l = lc[ lc[:, :Column1] .== loanNumber, :]
  global lc
  calculateIRR(loanNumber = l[1, :Column1],
               loan = l[1, :funded_amnt], intRate = l[1, :int_rate], term =l[1, :tenor],
               totalPaid = l[1, :total_pymnt], totalPrincipalPaid = l[1, :total_rec_prncp],
               totalInterestPaid = l[1, :total_rec_int],
               recoveries = l[1, :recoveries], lateFees = l[1, :total_rec_late_fee],
               showSchedule = true)
end


#############################################################################################################
##
## Quick check
##
calculateIRR(loanNumber = 1, loan = 5600, intRate = 0.1299, term = 36,
             totalPaid = 6791.72, totalPrincipalPaid = 5600, totalInterestPaid = 1191.72,
             recoveries = 0, lateFees = 0,
             showSchedule = true)

calculateIRR(loanNumber = 1, loan = 35000, intRate = 0.1820, term = 60,
             totalPaid = 26600.1, totalPrincipalPaid = 3874.72, totalInterestPaid = 5225.38,
             recoveries = 17500, lateFees = 0.0,
             showSchedule = false)

calculateIRR(loanNumber = 1734666, loan = 35000, intRate = 0.0797, term = 36,
```

```
            totalPaid = 1057.04, totalPrincipalPaid = 863.83, totalInterestPaid = 193.72,
            recoveries = 0, lateFees = 0,
            showSchedule = false)



##
## Look for the loans which have gone to their end
##
indextmp = (lendingClub.loan_status .== "Fully Paid") .|
           (lendingClub.loan_status .== "Charged Off") .|
           (lendingClub.loan_status .== "Does not meet the credit policy. Status:Charged Off") .|
           (lendingClub.loan_status .== "Does not meet the credit policy. Status:Fully Paid")

## Create the dataset we will use - Should be the same as lending_club_reformatted_paid.rds
lc = lendingClub[indextmp, :]

## Select relevant variables to calculate profitability
## Column1 contains the loanID's
cols = [:Column1, :funded_amnt, :int_rate, :term,
        :total_pymnt, :total_rec_prncp, :total_rec_int,
        :recoveries, :total_rec_late_fee]

lc = select(lc, cols)

## Interest rates as percentage
lc[:, :int_rate] = lc[:, :int_rate] ./ 100

## Create a new column
lc[:tenor] = 0

## that will record the official loan tenor as a number (instead of string)
lc[startswith.( lc[:, :term], " 36"), :tenor] .= 36
lc[startswith.( lc[:, :term], " 60"), :tenor] .= 60

## New data frame to store the results
IRR_Result = DataFrame(loanID = zeros(Int64, nrow(lc)),
                       IRR = zeros(Float64, nrow(lc)),
                       monthDefault = zeros(Int64, nrow(lc)))


# ~150 sec. to do the whole dataset
@time for i in 1:nrow(lc)
  global IRR_Result

  # Use multiple-return-value
  (IRR_Result[i, :loanID], IRR_Result[i, :IRR], IRR_Result[i, :monthDefault]) =
      calculateIRR(
          loanNumber = lc[i, :Column1],
          loan = lc[i, :funded_amnt], intRate = lc[i, :int_rate], term =lc[i, :tenor],
          totalPaid = lc[i, :total_pymnt], totalPrincipalPaid = lc[i, :total_rec_prncp],
          totalInterestPaid = lc[i, :total_rec_int],
          recoveries = lc[i, :recoveries], lateFees = lc[i, :total_rec_late_fee],
          showSchedule = false)
end


```

```
IRR_Result[1:10,:]
# Check
loanNumberIRR(171)

CSV.write("datasets/loanIRR.csv", IRR_Result)
```

4.4.2.2 Credit margins

```
#############################################################################################################
##
## Prepare datasets
##

## Previously saved from R with:
##     lending_club <- readRDS("lending_club.rds"); write.csv(lending_club, "lending_club.rds")
##
## WARNING: 1.7GB on disk
##
using CSV
lendingClub = CSV.read("datasets/lending_club.csv"; delim = ",")



#############################################################################################################
##
## CREDIT MARGIN
##
## This method of approximating the credit margin is far less sophisticated than what FI's do.
##
## We need to calculate what the credit margin should be on a defaulted loan to get a nil NPV.
##
## On a risk free loan the CF will be P+I at risk-free on _both_ borrowing and lending sides.
##
##  On a defaulted loan, the CF will be:
##      borrowing unchanged = P&I at risk-free
##    and
##      lending P&I at (risk-free + credit margin) until before default, then recoveries+fees.
##
## The principal amortisation profile depends on the credit margin used. We will arbitrarily use
## 20% which is a conservative assumption.
##
## credit risk on that CF should be nil with the right margin when discounted at risk-free.
##
## The function is very similar to the IRR calculation.
##

using DataFrames, Roots

# number of monthly payments to model
# It exceeds 60 months in case recoveries on a 60-month loan takes the schedule after 60 months.
const nMonths = 90



# Sculpt credit foncier profiles over 36 and 60 months at 20% per annum.
# The profile is expressed as percentage of loan amount
function CreateCreditFoncier(;n = 36, riskFree = 0.0)
```

```
50
51    instalment = 1 * riskFree/12 * 1 / (1 - 1 / (1 + riskFree/12) ^ n)
52
53    # We create a schedule
54    schedule = DataFrame(month = 0:nMonths, payment = 0.0)
55
56    # Add the day 1 principal outlay
57    schedule[1,       :payment] = -1
58    schedule[2:(n+1), :payment] = instalment
59
60    return(schedule)
61  end
62
63
64  # Solve for the credit margin
65  function CreditMargin(; loanNumber = 1, loan = 1000.0, intRate = 0.05, term = 36,
66    totalPaid = 1000.0, totalPrincipalPaid = 700.0, totalInterestPaid = 50.0,
67    recoveries = 0.0, lateFees = 0.0,
68    riskFree = 0.01,
69    showSchedule = false)
70
71    # Months after which a loan defaults (normal tenor if no default or early prepayment)
72    monthDefault = term
73
74    # Monthly instlment
75    instalment = ceil(loan * intRate/12 / (1 - 1 / (1 + intRate/12) ^ term), digits = 2)
76
77    # Create a blank schedule
78    schedule = DataFrame(month = 0:nMonths, monthlyPayment = 0.0,
79                         principalPayment = 0.0,
80                         totalPandI = 0.0, totalI = 0.0, totalP = 0.0)
81
82    for i in 2:(nMonths + 1)
83      # Get situation at the end of previous month
84      previousTotalPandI = schedule[i - 1, :totalPandI]
85      previousTotalP     = schedule[i - 1, :totalP]
86      previousTotalI     = schedule[i - 1, :totalI]
87
88      # This is the beginning of a new month. First and foremost, the borrower is expected to pay the
89      # accrued interest on amount of principal outstanding.
90      # The instalment is expected to cover that amount of interest and the rest goes to
91      # reducing the principal due outstanding.
92      accruedInterest = ceil((loan - previousTotalP) * intRate/12; digits = 2)
93      decreasePrincipal = instalment - accruedInterest
94
95      # If that amount takes the schedule above the total amount of interest shown in the data set,
96      # we should stop the schedule at this point
97      # This is a shortcut since we could have a payment higher than the interest due, but not enough
98      # to cover the expected principal repayment. However, it works well in practice.
99      if previousTotalI + accruedInterest > totalInterestPaid
100
101       # We stop the normal schedule at this date.
102       # Interest is paid (although less than scheduled)
103       schedule[i, :monthlyPayment] = totalInterestPaid - previousTotalI
104
105       # Whatever principal is left as per the dataset
106       schedule[i, :monthlyPayment] += totalPrincipalPaid - previousTotalP
107
```

```
      # Then 3-month after the last payment date, recoveries and and later fees are paid
      schedule[i + 3, :principalPayment] += recoveries + lateFees

      # Not really useful, but for completeness
      schedule[i, :totalPandI] = totalPaid
      schedule[i, :totalI]     = totalInterestPaid
      schedule[i, :totalP]     = totalPrincipalPaid

      # If total principal paid is less than borrower, then it is a default, and the monthDefault
      # is adjusted.
      if (totalPrincipalPaid < loan)
        monthDefault = i
      end

      # No more payments to add to the schedule
      break

    else
      # Deal with normal schedule
      schedule[i, :monthlyPayment]   = instalment
      schedule[i, :principalPayment] = decreasePrincipal
      schedule[i, :totalPandI]       = schedule[i-1, :totalPandI] + instalment
      schedule[i, :totalI]           = schedule[i-1, :totalI]     + accruedInterest
      schedule[i, :totalP]           = schedule[i-1, :totalP]     + decreasePrincipal
    end
  end

  # At this point schedule[, :monthlyPayment] contains the schedule of all payments, but needs to
  # include the initial loan.
  schedule[1, :principalPayment] = -loan

  if (showSchedule)
    println("Payments")
    println(schedule)
  end

  # Principal profile on the borrowing side
  creditFoncier = round.(CreateCreditFoncier(n = term, riskFree = riskFree)[:, :payment] .* loan;
                                             digits = 2)

  # For a given margin, calculate the _net_ NPV between the what is borrowed at the risk-free rate
  # and what is earned on the loan principal profile (possibly shortned because of default)
  # carrying an interest of risk-free + credit margin
  function NetNPV(margin)
    # We need to store the calculated interest
    interestSchedule = DataFrame(month = 0:nMonths, interestPayment = 0.0)

    # For each month, calculate the amount of interest with the credit margin
    for i in 2:(monthDefault + 1)
      outstandingPrincipal = sum(schedule[1:(i - 1), :principalPayment])
      interestSchedule[i, :interestPayment] =
            -round((riskFree + margin)/12 * outstandingPrincipal; digits = 2)
    end

    # Net final cashflow is:
    #     total principal and interest cashflow on the lending side
    # less
    #     borrowing profile
```

```
    cashFlow = schedule[:, :principalPayment] .+ interestSchedule[:, :interestPayment]
    cashFlow = cashFlow .- creditFoncier

    return sum(cashFlow ./ (1 + riskFree/12) .^ (0:nMonths))
  end

  # rootInterest = round(find_zero(NetNPV, (-0.5, 10), Bisection()); digits = 6)
  rootInterest = try
                   rootInterest = round(find_zero(NetNPV, (-0.5, 10), Bisection()); digits = 6)
                 catch e
                   NaN
                 end


  return((
    loanID = loanNumber,
    creditMargin = rootInterest,
    monthDefault = monthDefault
  ))
end


#########################################################################################################
##
## Quick check
##
CreditMargin(loanNumber = 1, loan = 5600, intRate = 0.1299, term = 36,
             totalPaid = 6791.72, totalPrincipalPaid = 5600, totalInterestPaid = 1191.72,
             recoveries = 0, lateFees = 0,
             riskFree = 0.02, showSchedule = false)

CreditMargin(loanNumber = 1, loan = 35000, intRate = 0.1820, term = 60,
             totalPaid = 26600.1, totalPrincipalPaid = 3874.72, totalInterestPaid = 5225.38,
             recoveries = 17500, lateFees = 0.0,
             riskFree = 0.02, showSchedule = true)

CreditMargin(loanNumber = 1734666, loan = 35000, intRate = 0.0797, term = 36,
             totalPaid = 1057.04, totalPrincipalPaid = 863.83, totalInterestPaid = 193.72,
             recoveries = 0, lateFees = 0,
             riskFree = 0.02, showSchedule = true)


##
## Look for the loans which have gone to their end
##
indextmp = (lendingClub.loan_status .== "Fully Paid") .|
           (lendingClub.loan_status .== "Charged Off") .|
           (lendingClub.loan_status .== "Does not meet the credit policy. Status:Charged Off") .|
           (lendingClub.loan_status .== "Does not meet the credit policy. Status:Fully Paid")

## Create the dataset we will use - Should be the same as lending_club_reformatted_paid.rds
lc = lendingClub[indextmp, :]

## Select relevant variables to calculate profitability
## Column1 contains the loanID's
cols = [:Column1, :funded_amnt, :int_rate, :term,
        :total_pymnt, :total_rec_prncp, :total_rec_int,
        :recoveries, :total_rec_late_fee]
```

```
lc = select(lc, cols)

## Interest rates as percentage
lc[!, :int_rate] = lc[!, :int_rate] ./ 100

## Create a new column
lc[:tenor] = 0

## that will record the official loan tenor as a number (instead of string)
lc[startswith.( lc[!, :term], " 36"), :tenor] .= 36
lc[startswith.( lc[!, :term], " 60"), :tenor] .= 60

## New data frame to store the results
creditMargin_Result = DataFrame(loanID = zeros(Int64, nrow(lc)),
                                creditMargin = zeros(Float64, nrow(lc)),
                                monthDefault = zeros(Int64, nrow(lc)))


# ~4,700 sec. to do the whole dataset
@time for i in 1:nrow(lc)
  global creditMargin_Result

  # Use multiple-return-value
  # 1h17m runtime
  (creditMargin_Result[i, :loanID],
   creditMargin_Result[i, :creditMargin],
   creditMargin_Result[i, :monthDefault]) =
      CreditMargin(
          loanNumber = lc[i, :Column1],
          loan = lc[i, :funded_amnt], intRate = lc[i, :int_rate], term =lc[i, :tenor],
          totalPaid = lc[i, :total_pymnt], totalPrincipalPaid = lc[i, :total_rec_prncp],
          totalInterestPaid = lc[i, :total_rec_int],
          recoveries = lc[i, :recoveries], lateFees = lc[i, :total_rec_late_fee],
          riskFree = 0.02,
          showSchedule = false)
end


creditMargin_Result[1:10,:]

CSV.write("datasets/CreditMargins.csv", creditMargin_Result)
```

4.4.3 Maxima derivation of the cost function

```
PDF1(x, Q) := alpha1( Q) * sqrt( 1 / ( 2 * pi)) *
              exp( - 1 / 2*(( log( -( x - m1( Q)) / m1( Q)) + sigma1( Q) ^ 2) / sigma1( Q))
              ( -( x - m1(Q)) * sigma1( Q)) ;

PDF2(x, Q) := alpha2( Q) * sqrt( 1 / ( 2 * pi)) *
              exp( - 1 / 2*(( log( -( x - m2( Q)) / m2( Q)) + sigma2( Q) ^ 2) / sigma2( Q))
              ( -( x - m2( Q)) * sigma2( Q)) ;

PDF3(x, Q) := alpha3( Q) * sqrt( 1 / ( 2 (* pi)) *
              exp( - 1 / 2*(( log( -( x - m3( Q)) / m3( Q)) + sigma3( Q) ^ 2) / sigma3( Q))
              ( -( x - m3( Q)) * sigma3( Q)) ;
```

```
PDF4(x, Q) := alpha4( Q) * sqrt( 1 / ( 2 * pi)) *
              exp( - 1 / 2*(( log( ( x - m4( Q)) / m4( Q)) + sigma4( Q) ^ 2) / sigma4( Q)) ^
              ( x - m4( Q)) * sigma4( Q)) ;


alpha1(Q) := am1* Q + an1 ;
alpha2(Q) := am2* Q + an2 ;
alpha3(Q) := am3* Q + an3 ;
alpha4(Q) := am4* Q + an4 ;

m1(Q) := mm1* Q + mn1 ;
m2(Q) := mm2* Q + mn2 ;
m3(Q) := mm3* Q + mn3 ;
m4(Q) := mm4* Q + mn4 ;

sigma1(Q) := sm1* Q + sn1 ;
sigma2(Q) := sm2* Q + sn2 ;
sigma3(Q) := sm3* Q + sn3 ;
sigma4(Q) := sm4* Q + sn4 ;

J(x, Q): = -( x - ( PDF1( x, Q) + PDF2( x, Q) + PDF3( x, Q) + PDF4( x, Q))) ^ 2 ;

diff( PDF1(x, Q), Q) ;
```

4.5 System version

```
##                                       sysname                                       release
##                                       "Linux"                           "5.3.0-24-generic"
##                                       version                                      nodename
## "#26-Ubuntu SMP Thu Nov 14 01:33:18 UTC 2019"                                        "x260"
##                                       machine                                         login
##                                      "x86_64"                                     "unknown"
##                                          user                                effective_user
##                                     "emmanuel"                                    "emmanuel"
```

Bibliography

California, L. C. S. F. (2019). Prospectus Regulatory Filing S3-ASR for Member Payment Dependent Notes. https://www.sec.gov/Archives/edgar/data/1409970/000140997019000988/0001409970-19-000988-index.htm. [Note: Accessed 31 October 2019].

Kan, L. C. P. W. (2019). Kaggle - LendingClub dataset. https://www.kaggle.com/wendykan/lending-club-loan-data. [Note: Accessed 31 August 2019].

Kim, A. and Cho, S.-B. (2019). An ensemble semi-supervised learning method for predicting defaults in social lending. *Engineering Applications of Artificial Intelligence*, 81:193–199.

Peng, R. (2012). *Exploratory data analysis with R*. Lulu. com.