Cleaning Lending Club Dataset

### About the data

Lending Club is an online P2P lending platform that offers unsecured personal loans by connecting borrowers with investors who wish to back their loans. Data for personal loans issued through this service is available going back to 2011 on the following website: <https://www.lendingclub.com/info/download-data.action>.

For this project, I chose to analyze loans issued in calendar years 2014 and 2015 for two reasons:

* Loans are issued with 36-month or 60-month payment terms, so data from several years ago will offer a more clear picture on each borrower's ability to repay.
* Going any further back in time may cause data to be of lesser quality. Furthermore, there may be macroeconomic trends in recent years that would cause older data to produce different results during analysis.

### Load packages and import data

I used the dplyr and tidyr packages to clean this data. The data was stored in two separate .csv files for 2014 and 2015.

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

loans14 <- read.csv("LoanStats3c.csv")  
loans15 <- read.csv("LoanStats3d.csv")

### Importing variable definitions into R

As a supplement to the loan data, Lending Club offers a dictionary that explains the meaning of each variable. Rather than change the names of any variable, I loaded this dictionary and created a function to search the definition of variables as needed. In the below example, I researched the definition of loan\_status.

definitions <- read.csv("LCDataDictionary.csv")  
definitions$LoanStatNew <- as.character(definitions$LoanStatNew)  
definitions$Description <- as.character(definitions$Description)  
define.var <- function(x) {  
 definitions[definitions$LoanStatNew == x,]  
}  
define.var("loan\_status")

## LoanStatNew Description  
## 43 loan\_status Current status of the loan

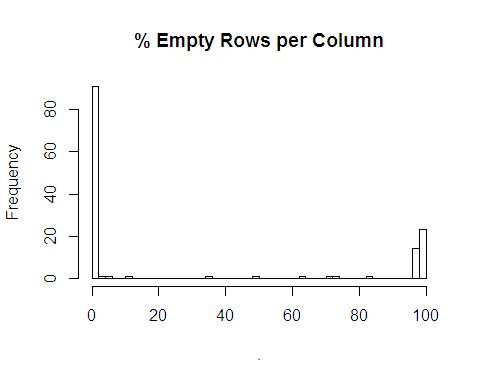
### Combine 2014 and 2015 data.

I combined the data into one dataset with around 650,000 rows and 137 columns. Then I called a summary to review the columns.

loans <- rbind(loans14, loans15)

summary(loans)

At this point, the quickest thing I uncovered was that there were many columns that had no data, most likely because the information was removed to protect borrowers' identities. The following histogram shows the count of columns binned by their percentage of rows that were NA.



After careful review, I decided to clear out any columns that were more than 85% empty, which eliminated 37 columns. This left me with numerous columns that still had empty rows.

emptycols <- c()  
for(i in 1:ncol(loans)) {  
 if(mean(is.na(loans[,i])) > 0.85) {  
 emptycols <- c(emptycols, i)  
 }  
}  
loans[,emptycols] <- NULL

### Wrangling Missing Data

For many of the remaining columns with missing data, I decided to create new variables as factors indicating whether or not the original variable had a value. Below is an example:

summary(loans$mths\_since\_last\_delinq)

This variable is the number of months since the borrower was last delinquent on any payment. Roughly half of the rows in this column were missing values. Since a lower value most likely corresponds to a higher potential for defaulting, we don't want to set the NA's to zero, nor do we want to assume these borrowers would fall in the middle of the distribution. In all likelihood, a missing value would indicate that the borrower has never had a delinquency. Imputing values for this column is a highly complex task, so the workaround is to create a new column reflecting whether or not the borrower has a delinquency in their history.

loans$ever\_delinq <- ifelse(is.na(loans$mths\_since\_last\_delinq), 0, 1)

Similar logic was applied to the following variables:

* mths\_since\_last\_major\_derog, Months since most recent 90-day or worse rating
* 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\_tl\_120dpd\_2m, Number of accounts currently 120 days past due (updated in past 2 months)
* 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. - Additional data manipulation will be done to this column later.

There were several columns where I elected to impute the missing values using the replace\_na function from tidyr. For the following columns, missing values were imputed with the median:

* 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.
* mths\_since\_recent\_bc, Months since most recent bankcard account opened.

The variable "mo\_sin\_old\_il\_acct", number of months since the borrower opened their oldest bank installment account, was imputed using the history on a typical credit report, which looks back 7 years. The NA's for this column would indicate that the borrower's oldest bank installment account was opened at least 7 years or *84 months ago*.

loans <- loans %>% replace\_na(list(mo\_sin\_old\_il\_acct = 84), mo\_sin\_old\_il\_acct)

For the percentage of bankcard accounts where spending has passed 75% of the card's limit, a missing value most likely indicates that the borrower does not have a bankcard. I replaced NA's for percent\_bc\_gt\_75 with zero's. The same logic applies to number of revolving accounts on the borrower's credit report (num\_rev\_accts).

**Employment Length (emp\_length)** - This column consisted of 11 levels that indicated employment length ("< 1 year" all the way up to "10+ years"), plus a 12th factor called "n/a". To resolve this, I collapsed the column to the 11 available factors, leaving the remaining rows as true NA's. As mentioned above, I also created a new variable indicating whether any employment was listed for the borrower.

loans$emp\_length <- factor(loans$emp\_length, levels = c("< 1 year", "1 year",  
 "2 years", "3 years",  
 "4 years", "5 years",  
 "6 years", "7 years",  
 "8 years", "9 years",  
 "10+ years"))  
loans$employment\_listed <- ifelse(is.na(loans$emp\_length), 0, 1)

### Re-formatting and Converting Data Types

Home ownership was a factor with 4 levels, but one of the levels ("ANY") only had 3 records. This column was collapsed to 3 factors: "RENT"", "MORTGAGE"" and "OWN". Each loan's issue date was recorded as an abbreviated month and two-digit year, where the 2014 loans had month first and the 2015 loans had year first. Using regular expressions, I created separate columns for the issue year of the loan and the issue month of the loan and converted issue month to a 12-level factor. Below is a snippet of the code that I used...

loans$issue\_year <- ifelse(grepl("14", loans$issue\_d), 2014, 2015)  
loans$issue\_month <- loans$issue\_d  
loans$issue\_month <- gsub(".\*Jan.\*", "January", loans$issue\_month)  
loans$issue\_month <- gsub(".\*Feb.\*", "February", loans$issue\_month)

...and the same goes for every month, of course.

Some columns simply needed to be converted to different types:

* Since read.csv() was called with stringsAsfactors defaulting to TRUE, employment title and loan description needed to be converted to character strings.
* Interest rate was imported as a factor, most likely because of the '%' symbol in the .csv. I converted this column to characters and used gsub() to remove the symbol, then converted to numeric.

### Completion

Finally, I created a column called "defaulted", a 2-level factor that converted loan status into a TRUE for defaults and FALSE for non-defaults.

loans <- loans %>%  
 mutate(defaulted = as.factor(loan\_status == "Default" | loan\_status == "Charged Off"))

Now the dataset is ready to be analyzed.

write.csv(loans, file = "loans\_clean.csv")