Data Challenge

Zongyan Wang

March 15, 2016

How to use

To use the .Rmd file, it will send the user a request during the compiling to choose the data loaded. For the first request, please upload "LC_biz_all.csv". Please also place "importanceRank.txt" and "state_latlon.csv" in your work directory. Thanks
Please notice, this .Rmd file would also require clean_f.R, plot_f.R, analysis_f.R.

Upload packages for data clean and analysis

```
require(plyr)  #data clean
```

```
## Loading required package: plyr
```

```
require(dplyr) #data clean
```

```
## Loading required package: dplyr

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:plyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize

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

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

```
require(tidyr) #data clean
```

```
## Loading required package: tidyr
```

```
require(ggplot2)  # visualzation
```

```
## Loading required package: ggplot2
```

```
require(XML)   # web scriping
```

```
## Loading required package: XML
```

```
require(testthat) #test model
```

```
## Loading required package: testthat
```

```
require(kernlab) # kernel and SVM
```

```
## Loading required package: kernlab
```

```
##
## Attaching package: 'kernlab'
```

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

```
require(reshape2)
```

```
## Loading required package: reshape2
```

```
require(caret) # Cross - Validation k fold
```

```
## Loading required package: caret
```

```
## Loading required package: lattice
```

```
require(maps) # heatmap: map_data
```

```
## Loading required package: maps
```

```
##
##  # maps v3.1: updated 'world': all lakes moved to separate new #
##  # 'lakes' database. Type '?world' or 'news(package="maps")'.  #
```

```
##
## Attaching package: 'maps'
```

```
## The following object is masked from 'package:plyr':
##
##     ozone
```

```
require(ggmap) # for heatmap
```

```
## Loading required package: ggmap
```

Read file

```
# Read Data and function files
LC <- read.csv(file.choose(), header = T) #Please read LC_biz_all.csv
source("clean_f.R") # functions for cleaning the data
source("plot_f.R") # functions for plot(heatmap)
source("analysis_f.R") # functions for plot and data analysis
```

Data clean

```
lc <- perToN(LC)  # transfer percentage to numeric
lc <- replaceBlank_all(lc) # replace the blank
lc <- dateToNum(lc) # transform earliest_cr_line from date to numeric
```

Data Summary

Data overview

```
dim(lc)               #variable size
```

```
## [1] 5641   56
```

```
hasNA_all(lc)            # variables contains missing values
```

```
##      mths_since_last_delinq      mths_since_last_record
##                        2421                        4302
##                 revol_util mths_since_last_major_derog
##                           3                        3674
##                    bc_util       mths_since_recent_bc
##                          77                          68
##    mths_since_recent_bc_dlq      mths_since_recent_inq
##                        3932                         381
##           num_tl_120dpd_2m           percent_bc_gt_75
##                         275                          77
```

```
length(hasNA_all(lc))
```

```
## [1] 10
```

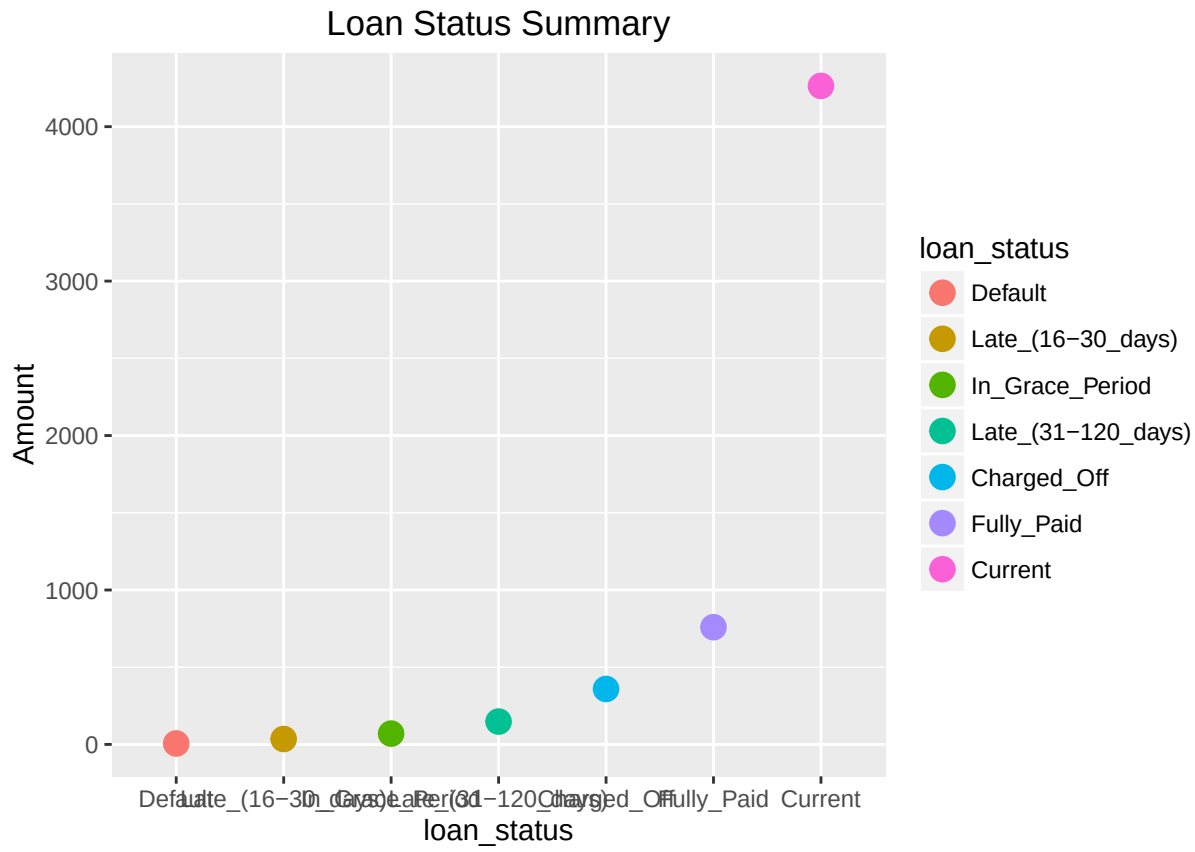
```
n.factor_all(lc)       # factor levels in variables among the data
```

```
##   term emp_title emp_length home_ownership verification_status issue_d
## 1    2      2660         12              3                   3      24
##   loan_status purpose zip_code addr_state last_pymnt_d next_pymnt_d
## 1           7       1      697         48           25            4
##   last_credit_pull_d
## 1                 24
```

```
length(n.factor_all(lc))
```

```
## [1] 13
```

```
loan <- lc %>%
  ddply(~loan_status, function(x) dim(x)[1]) %>%
  mutate(loan_status = reorder(x = loan_status,X = V1))

loanplot <- ggplot(loan, aes(x = loan_status, y = V1, fill = loan_status)) +
  geom_point(aes(color = loan_status),size = 4) +
  labs(y = "Amount", title = "Loan Status Summary")
print(loanplot)
```




```
ggsave(filename = "loanplot.png", plot = loanplot, path = ".",
       width = 10, height = 6, dpi = 400)
```

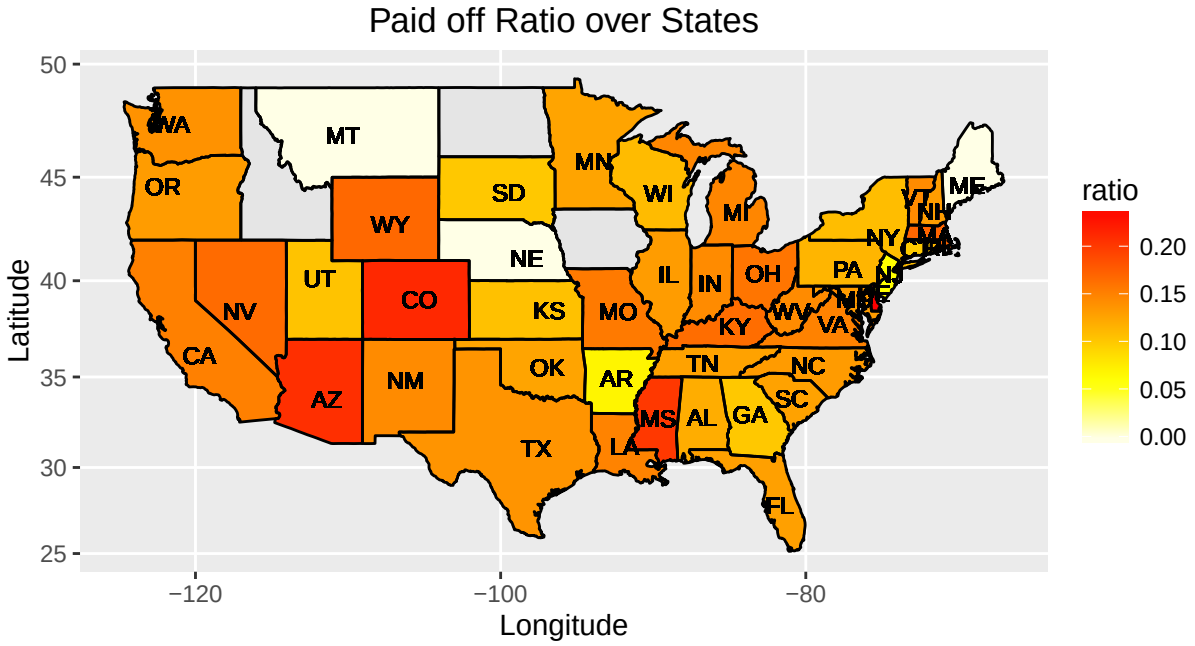
Delete uninteresting variables

```
#exclude loan_status, id, last payment day, next payment day, zip_code, issue_d
lc <- lc %>% select(-id,  -emp_title,
                    -last_pymnt_d, -next_pymnt_d,
                    -zip_code,-issue_d, -last_credit_pull_d)
```

Paid off ratio over United States

```
lc_fullypaid <- lc_ratio(lc, "Fully_Paid")
plot_fullypaid <- lc_heatmap(lc_fullypaid, state, location,
                             "Paid off Ratio over States")
print(plot_fullypaid)
```

```
## Warning: Removed 604 rows containing missing values (geom_text).
```



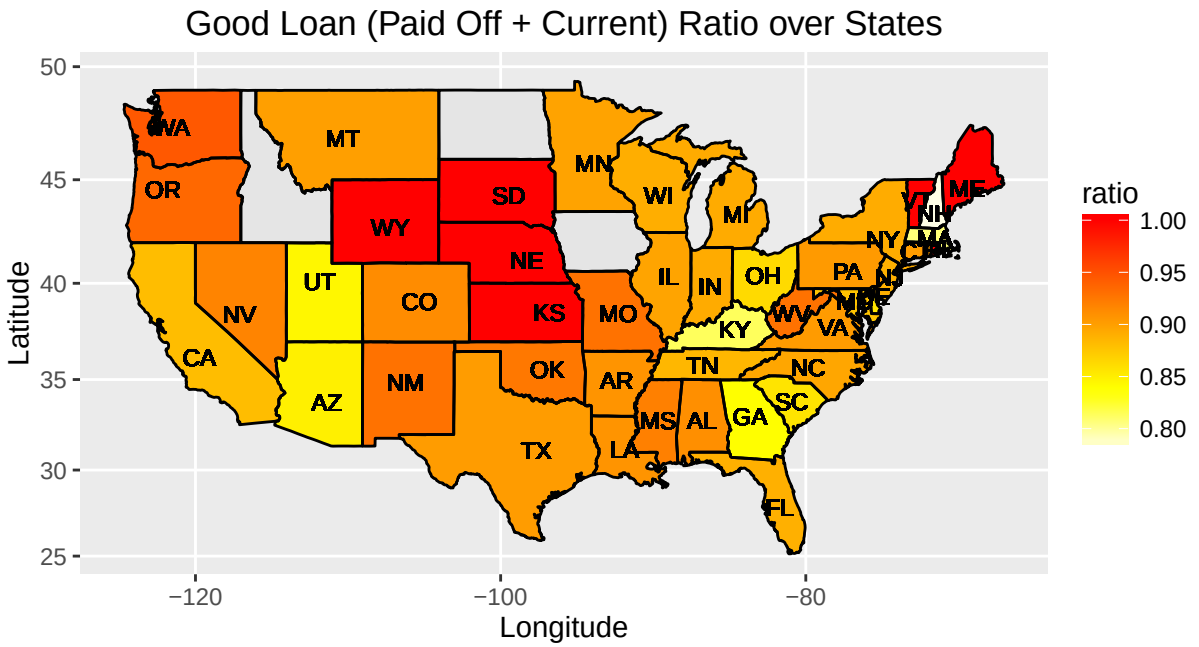

```
ggsave(filename = "plot_fullypaid.png", plot = plot_fullypaid, path = ".",
       width = 6, height = 6, dpi = 400)
```

```
## Warning: Removed 604 rows containing missing values (geom_text).
```

paid off and current (Good loan) ratio over United Status

```
lc_current <- lc_ratio(lc, "Current")
lc_goodLoan <- merge(lc_current, lc_fullypaid, by = "addr_state")
lc_goodLoan <- lc_goodLoan %>% mutate(ratio = ratio.x + ratio.y) %>%
  select(-c(ratio.x,ratio.y))
plot_goodLoan <- lc_heatmap(lc_goodLoan, state, location,
                            "Good Loan (Paid Off + Current) Ratio over States")
print(plot_goodLoan)
```

```
## Warning: Removed 604 rows containing missing values (geom_text).
```



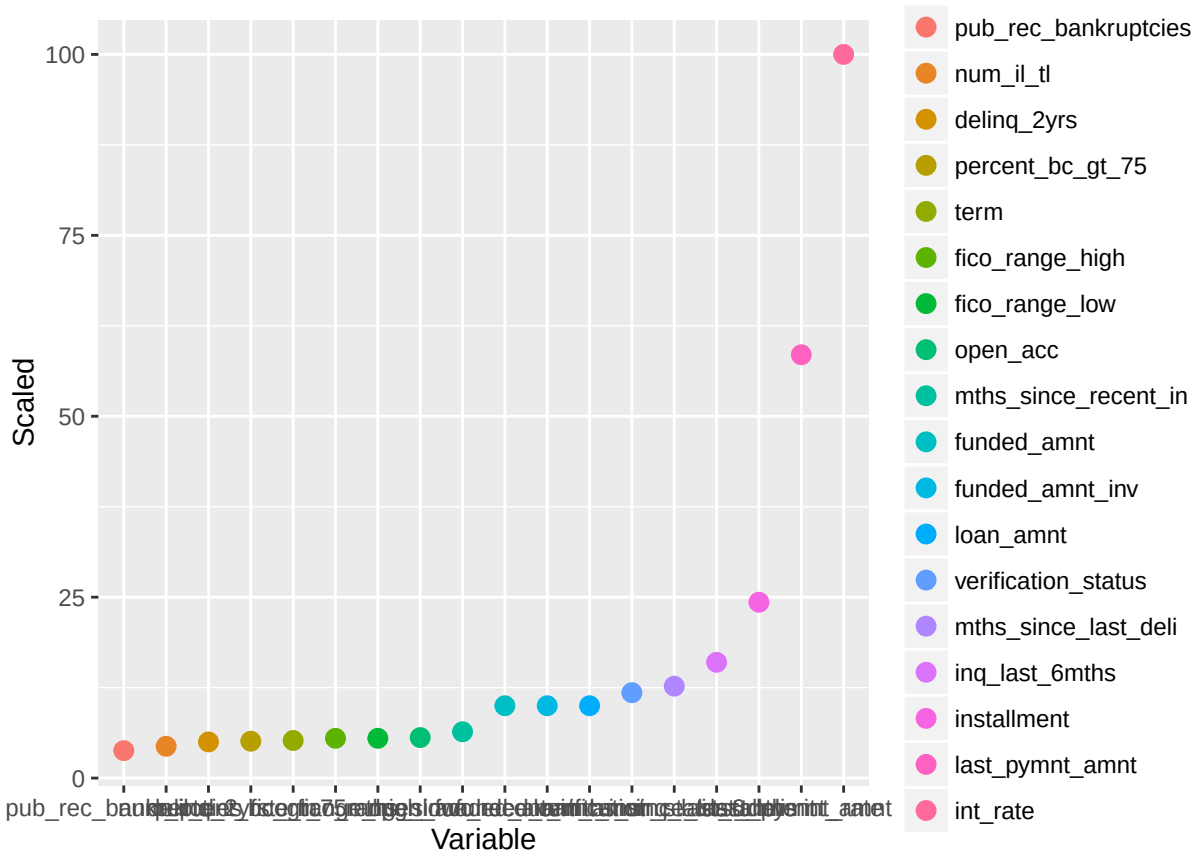

```
ggsave(filename = "plot_goodLoan.png", plot = plot_goodLoan, path = ".",
       width = 6, height = 6, dpi = 400)
```

```
## Warning: Removed 604 rows containing missing values (geom_text).
```

Data Analysis

variable selection

```
rank <- read.table("importanceRank.txt", header = T) # The list of important variables(importanceRank.t
rank <- rank %>% mutate(Variable = reorder(x = Variable, X = Scaled))
## plot rank
rankPlot <- rank %>% ggplot(aes(x = Variable, y = Scaled)) +
  geom_point(aes(color = Variable), size = 3)
print(rankPlot)
```




```
ggsave(filename = "rankPlot.png", plot = rankPlot, path = ".",
       width = 10, height = 6, dpi = 400)
## Select the important data in original data
lc_i <- lc[,names(lc)%in%as.character(rank$Variable)]
lc_i$loan_status <- lc$loan_status
## Summary of new data
hasNA_all(lc_i)  # Variable has missing data
```

```
## percent_bc_gt_75
##               77
```

```
n.factor_all(lc_i)  # Levels of Variable which are categorical data
```

```
##   term verification_status loan_status
## 1    2                  3           7
```

logistic regression

```
# resample the data with delinquency events
lc.sample <- sample_n(lc_i %>% subset(!loan_status %in% c("Current", "Fully_Paid")), 4000, replace = T)
lc_ii <- rbind(lc_i, lc.sample)

# weight other 10 times higher than good loan
```

```r
lc_iii <- lc_i
lc_iii$weight <- 1
lc_iii$weight[!lc_i$loan_status %in% c("Current", "Fully_Paid")] <- 10

# Remove all the rows with NA
lc_i <- lc_i[-which(is.na(lc$percent_bc_gt_75)),]
# Data analysis
status <- as.character(levels(lc_i$loan_status))
# Set "Charged Off" to 1
log_charge <- log.loan(lc_i, status[1])
#model(log_charge, status[1])
# Set "Current" to 1
log_current <- log.loan(lc_i, status[2])
```

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

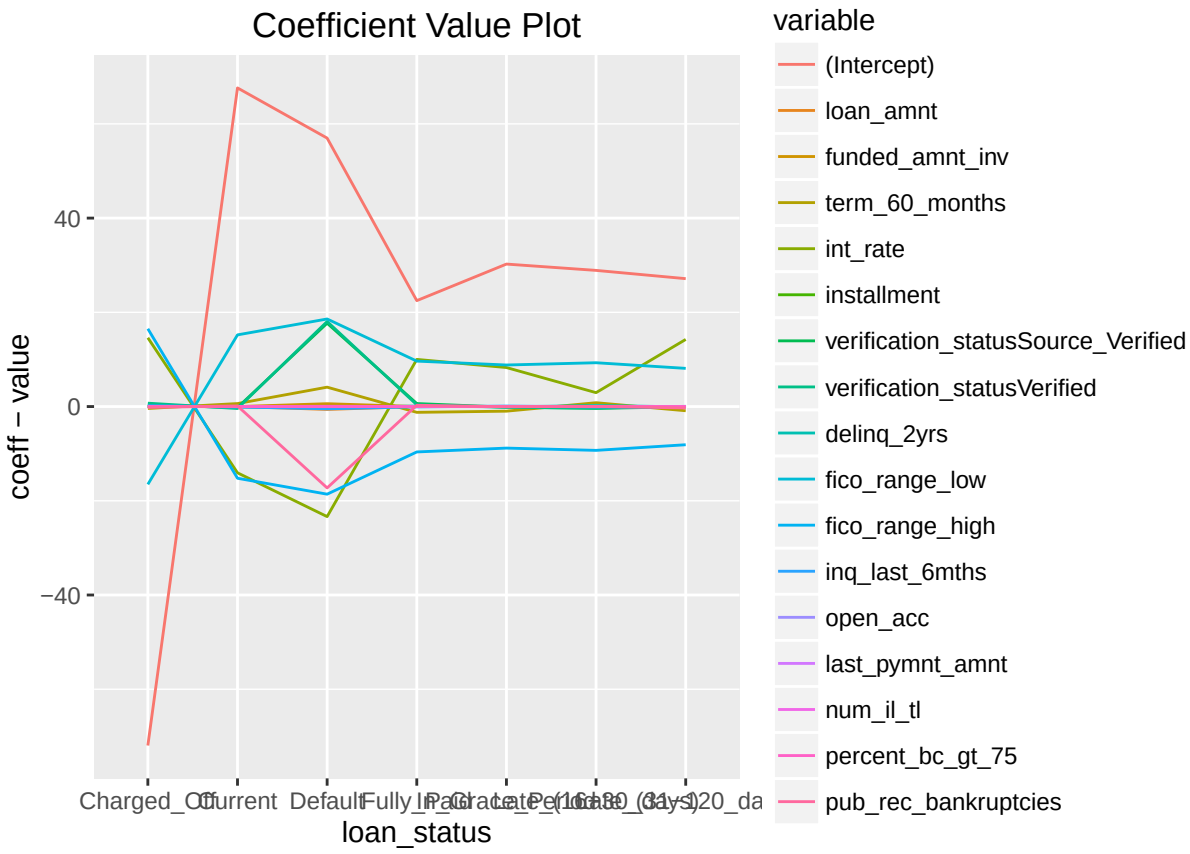
```r
# Set "Default" to 1
log_default <- log.loan(lc_i, status[3])
```

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

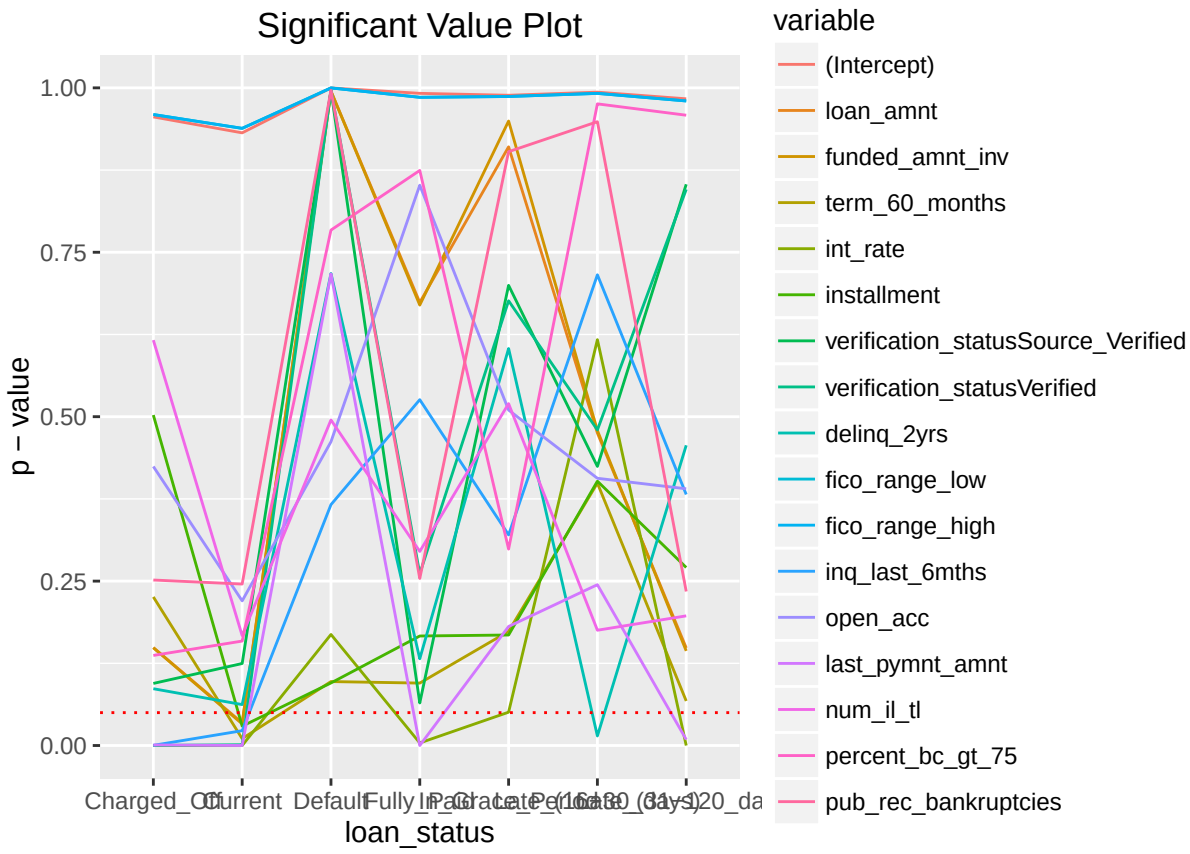
```r
# Set "Fully Paid" to 1
log_fd <- log.loan(lc_i, status[4])
```

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

```r
# Set "In_Grace_Period" to 1
log_grace <- log.loan(lc_i, status[5])
# Set "Late_(16-30 days)" to 1
log_late1 <- log.loan(lc_i, status[6])
# Set "Late_(31 - 120 days)" to 1
log_late2 <- log.loan(lc_i, status[7])

# Plot the significant value
l.list <- list(log_charge, log_current, log_default, log_fd, log_grace, log_late1, log_late2)
plot_coeff <- coeff_plot(l.list,status, sig = F)
plot_significant <- coeff_plot(l.list, status, sig = T)
print(plot_coeff)
```




```
ggsave(filename = "plot_coeff.png", plot = plot_coeff, path = ".",
       width = 10, height = 6, dpi = 400)
print(plot_significant)
```



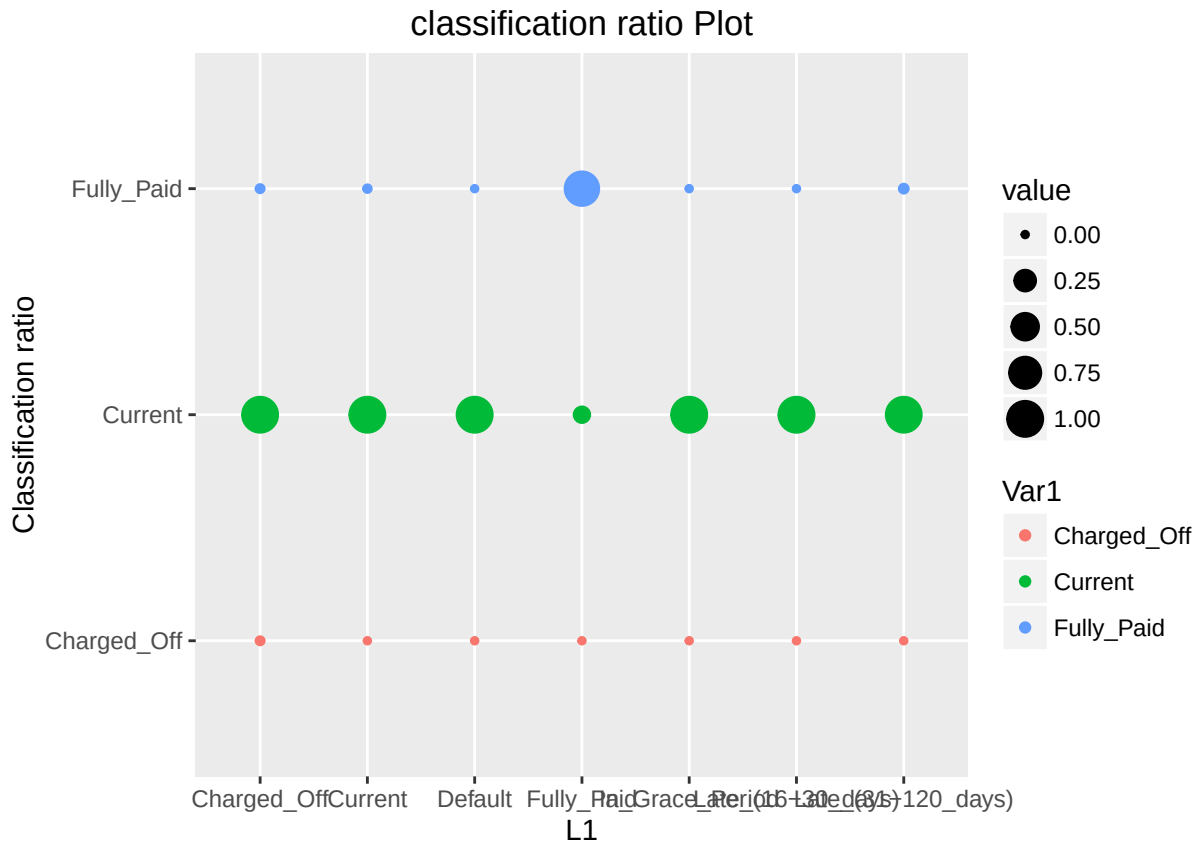

```
ggsave(filename = "plot_significant.png", plot = plot_significant, path = ".",
       width = 10, height = 6, dpi = 400)
```

multi-level Logistic regression model accuracy(without weight)

```
## in-sample error
l.list <- glm_list(lc_i, status)
in_sample_pred <- predict_glm(l.list, lc_i, status)
in_sample_error <- 1 - sum(in_sample_pred == lc_i$loan_status)/length(lc_i[,1])
in_sample_error
```

```
## [1] 0.1234723
```

```
# plot classification result
plot_mis1 <- mis_plot(true = lc_i$loan_status, pred = in_sample_pred)
print(plot_mis1)
```




```
ggsave(filename = "plot_mis1.png", plot = plot_mis1, path = ".",
       width = 10, height = 6, dpi = 400)
```

multi-level Logistic regression model accuracy(resample)

```
## in-sample error
newdata <- lc_i
l.list <- glm_list(lc_ii, status)
```

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

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

```
in_sample_pred <- predict_glm(l.list, newdata, status)
```

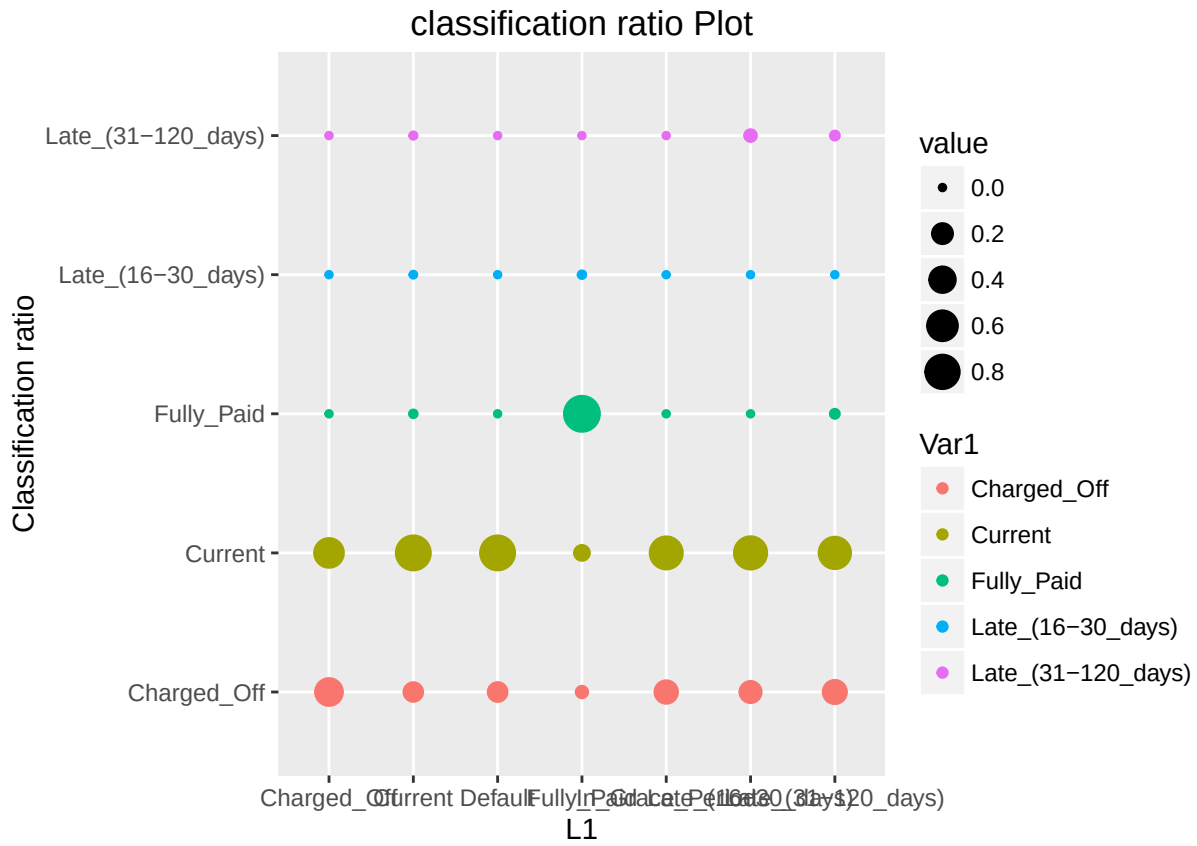
```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
```

```
## ifelse(type == : prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
```

```
in_sample_error <- 1 - sum(in_sample_pred == lc_i$loan_status)/length(lc_i[,1])
in_sample_error
```

```
## [1] 0.2185478
```

```
# plot classification result
plot_mis2 <- mis_plot(true = lc_i$loan_status, pred = in_sample_pred)
print(plot_mis2)
```



```r
ggsave(filename = "plot_mis2.png", plot = plot_mis2, path = ".",
       width = 10, height = 6, dpi = 400)
```

multi-level Logistic regression model accuracy(weight)

```r
## in-sample error
newdata <- lc_i
newdata$weight <- 1
l.list <- glm_list(lc_iii, status)
```

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

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

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

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

```r
in_sample_pred <- predict_glm(l.list, newdata, status)
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
```

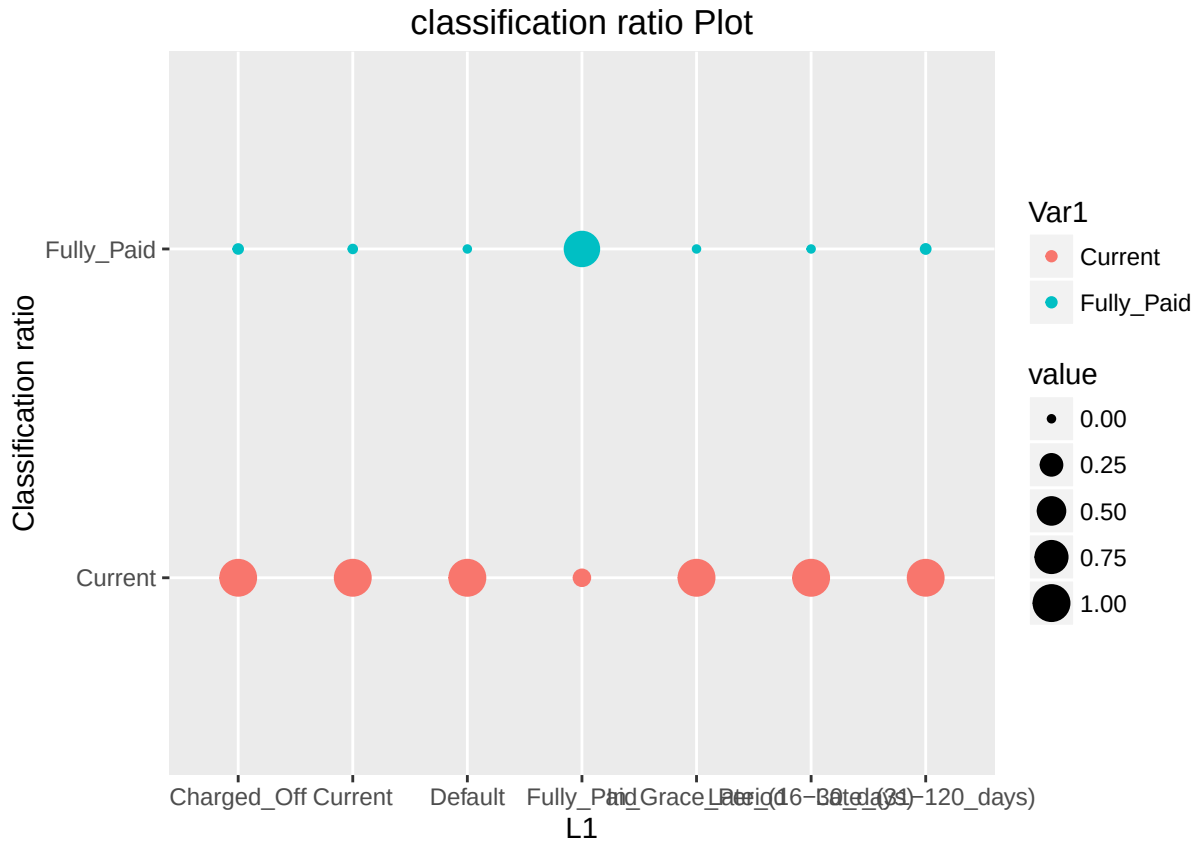
```r
in_sample_error <- 1 - sum(in_sample_pred == lc_iii$loan_status)/length(lc_iii[,1])
```

```
## Warning in is.na(e1) | is.na(e2): longer object length is not a multiple of
## shorter object length

## Warning in `==.default`(in_sample_pred, lc_iii$loan_status): longer object
## length is not a multiple of shorter object length
```

```
in_sample_error
```

```
## [1] 0.3006559
```

```
# plot classification result
plot_mis3 <- mis_plot(true = lc_i$loan_status, pred = in_sample_pred)
print(plot_mis3)
```




```
ggsave(filename = "plot_mis3.png", plot = plot_mis3, path = ".",
       width = 10, height = 6, dpi = 400)
```

Kernel methods

```
lc_svm_i <- prep_svm(lc_i)
# resample data for svm
lc_svm_ii <- prep_svm(lc_ii)
# sample SVM model"
svmmodel <- svm(lc_svm_i,"Fully_Paid")
```

```
## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.
```

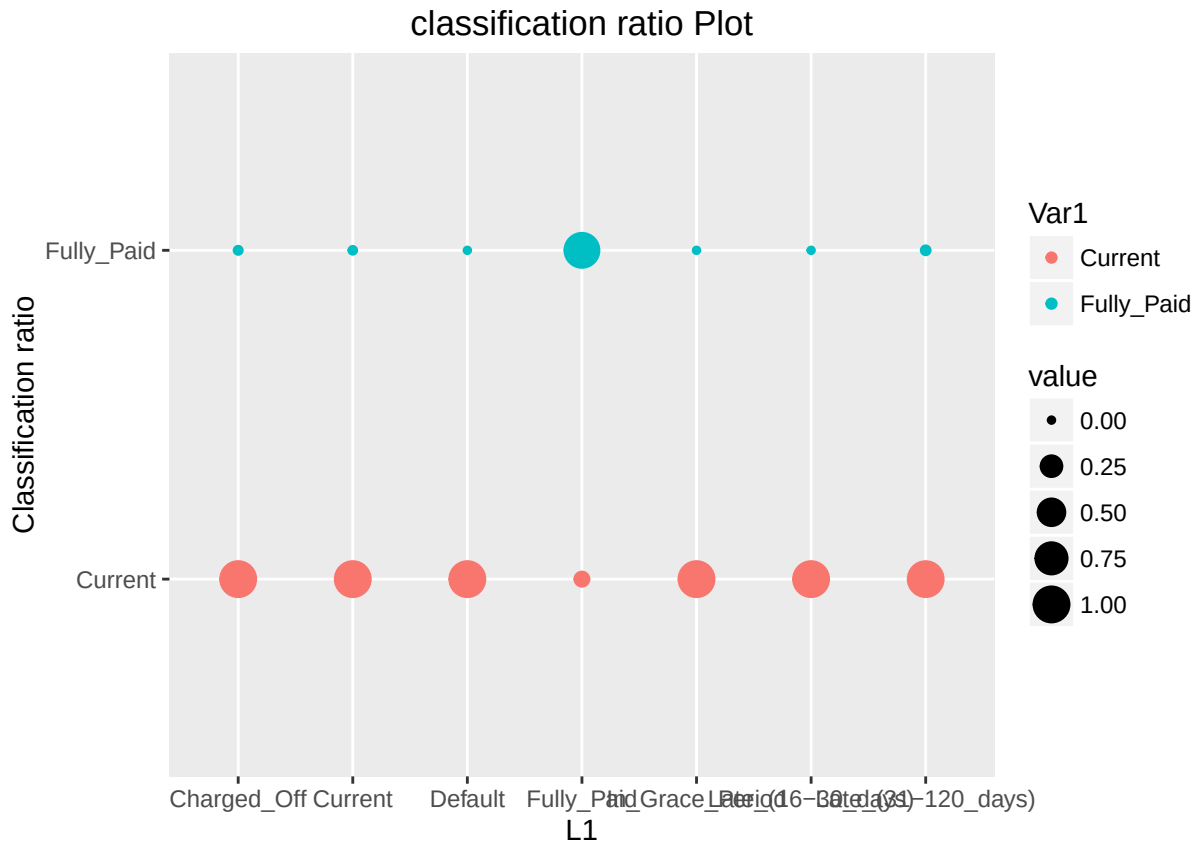
```
  #our SVM model
newdata <- as.matrix(lc_svm_i%>% select(-loan_status))
kern.list <- kern_list(lc_svm_i)
```

```
## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.

## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.

## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.

## maximum number of iterations reached -0.0009416007 -0.0009517965

## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.

## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.

## maximum number of iterations reached 0.001780675 0.001710587

## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.

## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.

## maximum number of iterations reached 8.717039e-05 8.669129e-05
```

```
pred_svm <- pred.svm(kern.list, newdata, status = status)

in_sample_error_kern <- 1 - sum(pred_svm == lc_i$loan_status)/length(lc_i[,1])
in_sample_error_kern
```

```
## [1] 0.1200575
```

```
# plot classification result
plot_mis_kern <- mis_plot(true = lc_i$loan_status, pred = pred_svm)
print(plot_mis_kern)
```




```
ggsave(filename = "plot_mis_kern.png", plot = plot_mis_kern, path = ".",
       width = 10, height = 6, dpi = 400)
```

SVM for data after resampling

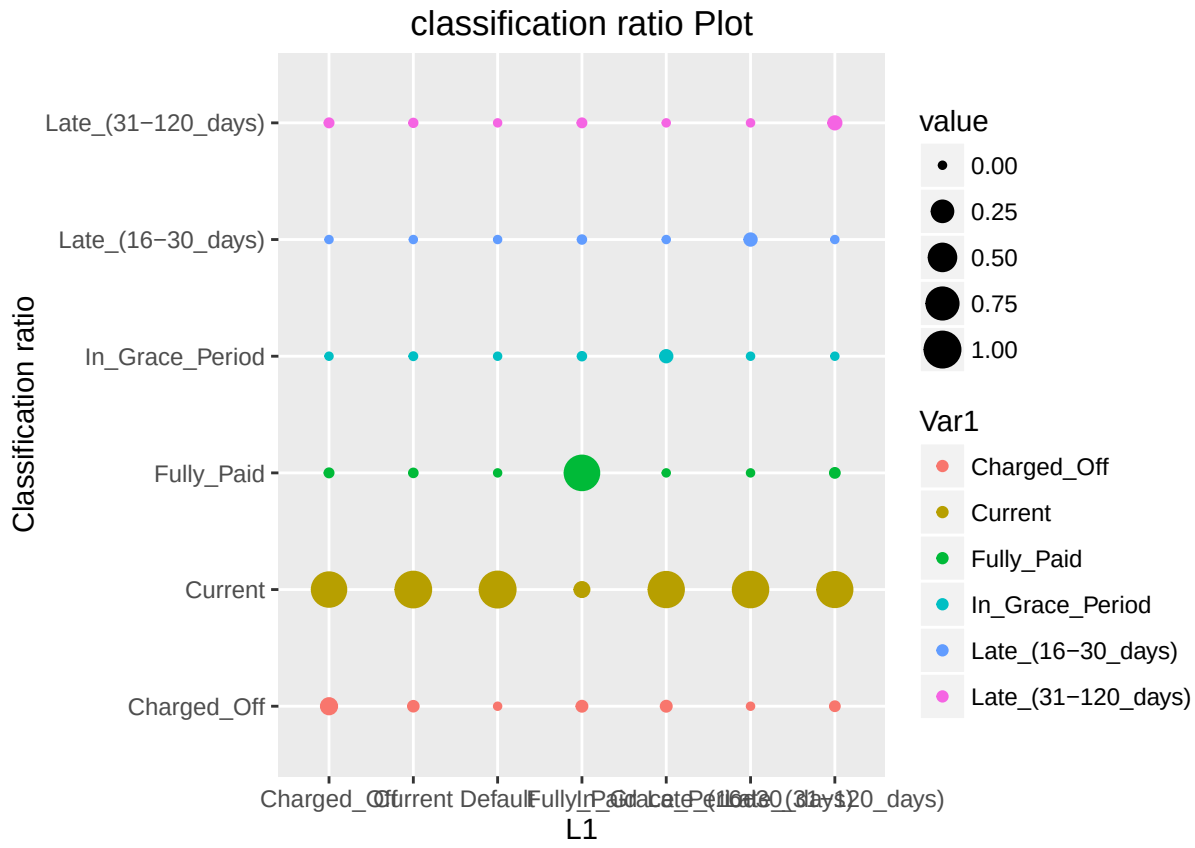
```
#our SVM model
newdata <- as.matrix(lc_svm_i%>% select(-loan_status))
kern.list2 <- kern_list(lc_svm_ii)
```

```
## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.

## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.

## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.

## maximum number of iterations reached 0.0003943027 0.0003842674

## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.

## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.

## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.

## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.
```

```
pred_svm2 <- pred.svm(kern.list2, newdata, status = status)

in_sample_error_kern2 <- 1 - sum(pred_svm == lc_i$loan_status)/length(lc_i[,1])
in_sample_error_kern2
```

```
## [1] 0.1200575
```

```
# plot classification result
plot_mis_kern2 <- mis_plot(true = lc_i$loan_status, pred = pred_svm2)
print(plot_mis_kern2)
```




```
ggsave(filename = "plot_mis_kern2.png", plot = plot_mis_kern2, path = ".",
       width = 10, height = 6, dpi = 400)
```