EDA_MAIN

Yash Karande

2022-11-28

Lending Club

Assignment 1A and 1B

Loading necessary libraries

```
# Loading the libraries

library(tidyverse)

library(lubridate)

library(stringr)

library(pROC)

library(rpart)

library(ROCR)

library(C50)

library(caret)


library(ranger)

library(glmnet)
```

Loading the data

```
lcdf <- read_csv('/Users/sthaka3/Downloads/Archive (2)/Assignment
1/lcDataSample.csv')

# Checking number of rows and columns in the lc dataframe

paste0('The number of rows are = ', nrow(lcdf))

## [1] "The number of rows are = 110000"

paste0('The number of columns are = ',ncol(lcdf))
```

```
## [1] "The number of columns are = 145"
```

How many differnt types of loan status exist in the data?

```
lcdf %>% group_by(loan_status) %>% tally()
```

```
## # A tibble: 6 × 2
##   loan_status            n
##   <chr>              <int>
## 1 Charged Off        15377
## 2 Current               17
## 3 Fully Paid         94567
## 4 In Grace Period        2
## 5 Late (16-30 days)      1
## 6 Late (31-120 days)    36
```

```
paste0("Since there are values apart from the target - fullly paid and
charged off we will keep only fully paid and charged off loans from the
target variable.
#Filtering the dataframe and updating it to the same dataframe")
```

```
## [1] "Since there are values apart from the target - fullly paid and
charged off we will keep only fully paid and charged off loans from the
target variable.\n#Filtering the dataframe and updating it to the same
dataframe"
```

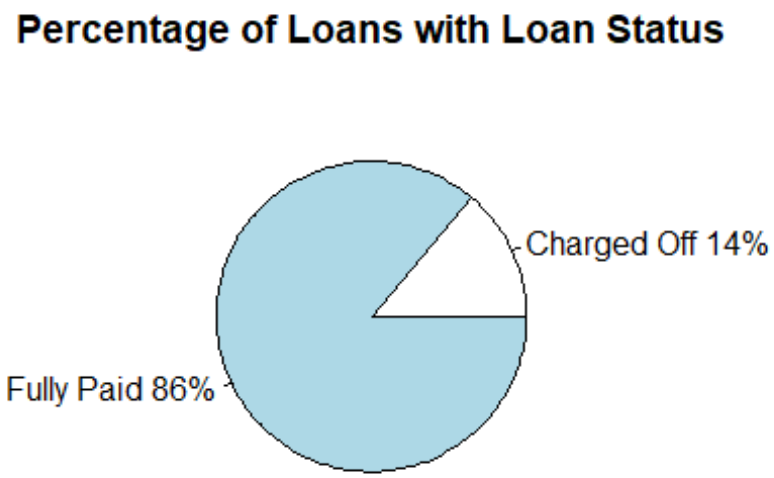
Filtering for Charged off and Fully Paid

```
### Since there are values apart from the target - fullly paid and charged
off we will keep only fully paid and charged off loans from the target
variable.
#Filtering the dataframe and updating it to the same dataframe

lcdf <- lcdf %>% filter(loan_status == "Fully Paid" | loan_status == "Charged
Off")

lcdf %>% group_by(loan_status) %>% tally()
```

```
## # A tibble: 2 × 2
##   loan_status     n
##   <chr>       <int>
## 1 Charged Off 15377
## 2 Fully Paid  94567
```

Distribution of Loan Status

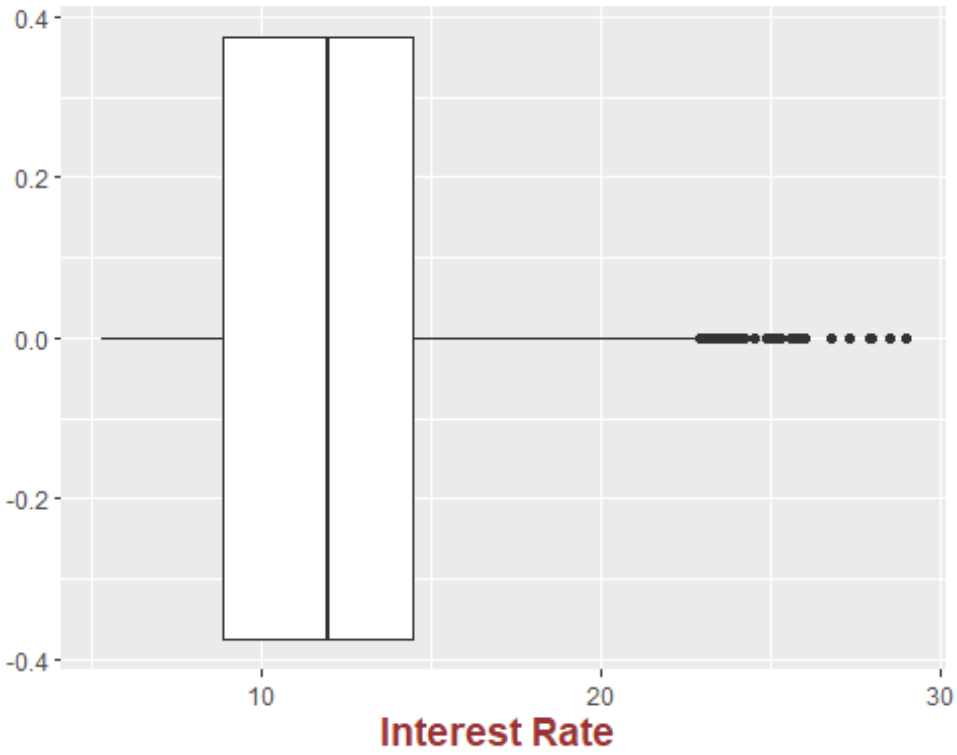
```
loan_status_count <- lcdf %>% group_by(loan_status) %>% count()
pct <- round(loan_status_count$n/sum(loan_status_count$n)*100)
lbls <- paste(loan_status_count$loan_status, pct) # add percents to labels
lbls <- paste(lbls,"%",sep="") # ad % to labels


pie(loan_status_count$n, labels = lbls, main="Percentage of Loans with Loan
Status")
```



Analzing Interest Rate

We will create a box plot to visualize the spread of the interest rate

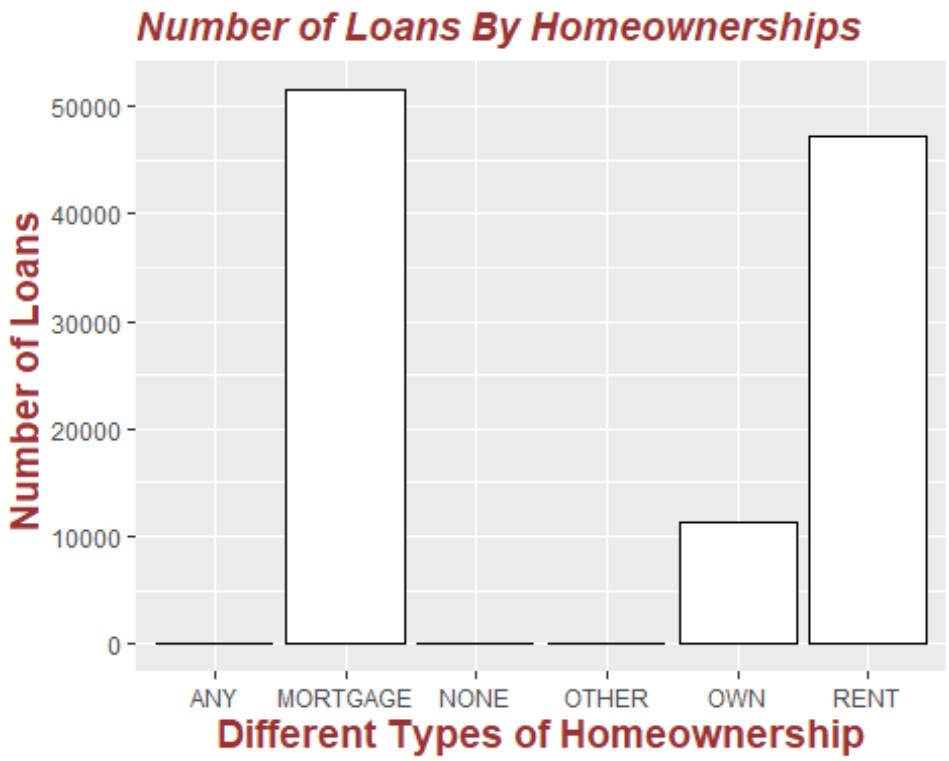
```
summary(lcdf$int_rate)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    5.32    8.90   11.99   12.05   14.48   28.99

ggplot(lcdf, aes( x = int_rate)) + geom_boxplot() +
xlab("Interest Rate ")  + theme(plot.title = element_text(color="#993333",
size=14, face="bold.italic"), axis.title.x = element_text(color="#993333",
size=14, face="bold"), axis.title.y = element_text(color="#993333", size=14,
face="bold"))
```



25 Percentile of loans give less than 8.9% interest rate. Median of the interest rate of all loans in 11.99%. The interest rate can go as high as 28.99 % in some case.The interest rate when higher can be a high risk loan. This interest seems really active to invest in. Very few investment products give an interest of 12%.

Home Ownership

```
ggplot(lcdf, aes( x = home_ownership)) + geom_bar(colour="black",
fill="white") +ggtitle("Number of Loans By Homeowners hips") + xlab("Different
Types of Homeownership") + ylab("Number of Loans ") + theme(plot.title =
element_text(color="#993333", size=14, face="bold.italic"), axis.title.x =
element_text(color="#993333", size=14, face="bold"), axis.title.y =
element_text(color="#993333", size=14, face="bold"))
```



Most borrowers are not owning a home. Most of loans were given to people who have mortgaged and rented house.

Loan Grades

Loans also have different grade and we would want to see how many of them are present in each grade along with loan status

```
lcdf %>% group_by(grade) %>% tally()
```

```
## # A tibble: 7 × 2
##   grade     n
##   <chr> <int>
## 1 A     24854
## 2 B     37865
## 3 C     29145
## 4 D     13455
## 5 E      3790
## 6 F       753
## 7 G        82
```

Adding the loan status to check on loan status and grade together

```
table(lcdf$loan_status, lcdf$grade)
```

```
##
##                  A     B     C     D     E     F     G
```

```
##   Charged Off  1369  4264  5206  3165  1090   252    31
##   Fully Paid  23485 33601 23939 10290  2700   501    51
```

Some loans have been charged off in the grade 'A' Some loans in grade 'G' have been fully paid. Let us look at the default percentage of each grade to get a better picture.

```
lcdf %>% group_by(grade) %>% summarise(TotalLoans=n(),
FullyPaid=sum(loan_status=="Fully Paid"),
ChargedOff=sum(loan_status=="Charged Off"), default_percentage =
ChargedOff/TotalLoans*100)
```

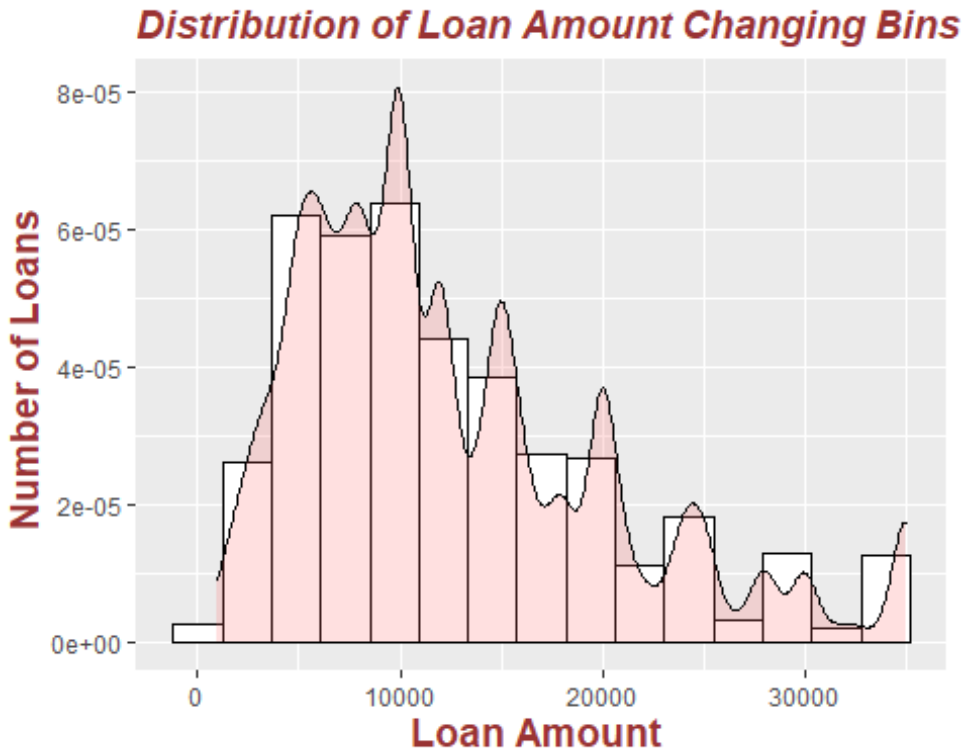
```
## # A tibble: 7 × 5
##   grade TotalLoans FullyPaid ChargedOff default_percentage
##   <chr>      <int>     <int>      <int>              <dbl>
## 1 A          24854     23485       1369               5.51
## 2 B          37865     33601       4264              11.3
## 3 C          29145     23939       5206              17.9
## 4 D          13455     10290       3165              23.5
## 5 E           3790      2700       1090              28.8
## 6 F            753       501        252              33.5
## 7 G             82        51         31              37.8
```

How does number of loans, loan amount, interest rate vary by grade?

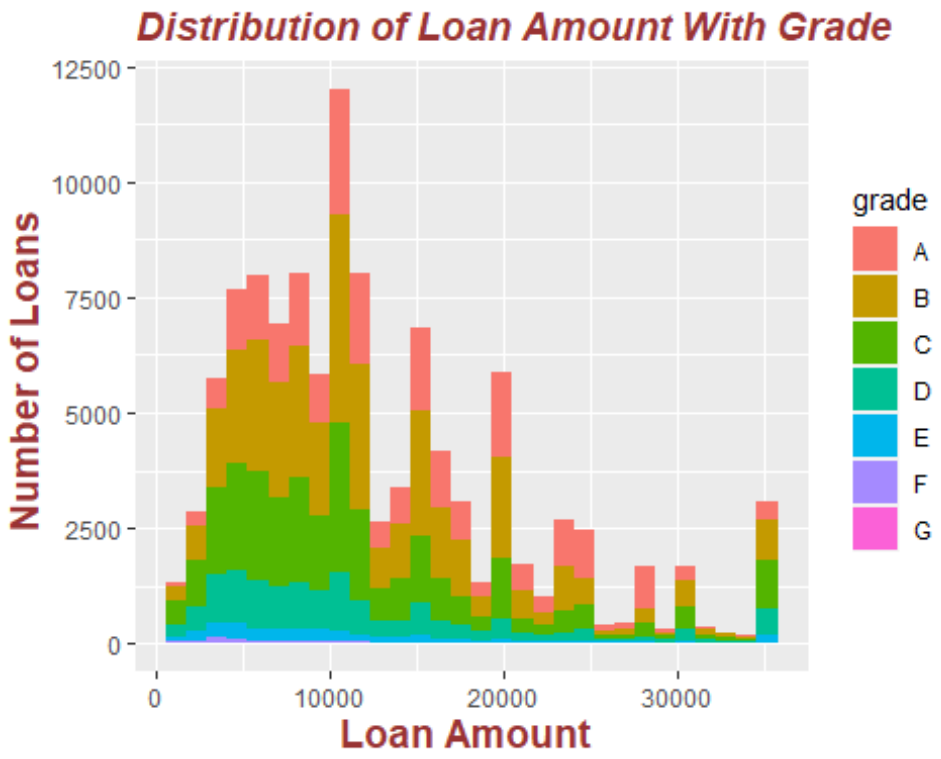
```
# Number of Loans, Sum of Loan Amout, Mean Loan Amount Mean Int Rate by Grade
lcdf %>% group_by(grade) %>% summarise(numberOfLoans=n(),
TotLoanAmt=sum(loan_amnt),MeanLoanAmt=mean(loan_amnt),defaults=sum(loan_statu
s=="Charged Off"), defaultRate=defaults/numberOfLoans, default_percentage =
defaultRate*100,MeanIntRate=mean(int_rate),stdInterest=sd(int_rate), minInt =
min(int_rate),maxInt=max(int_rate),avgLoanAMt=mean(loan_amnt),
sumPmnt=sum(total_pymnt),avgPmnt=mean(total_pymnt))
```

```
## # A tibble: 7 × 14
##   grade numberO…¹ TotLo…² MeanL…³ defau…⁴ defau…⁵ defau…⁶ MeanI…⁷ stdIn…⁸
minInt
##   <chr>     <int>   <dbl>   <dbl>   <int>   <dbl>   <dbl>   <dbl>   <dbl>
<dbl>
## 1 A         24854  3.57e8  14349.    1369  0.0551    5.51    7.21   0.973
5.32
## 2 B         37865  4.74e8  12506.    4264  0.113    11.3    10.9    1.48
6
## 3 C         29145  3.51e8  12048.    5206  0.179    17.9    13.9    1.23
6
## 4 D         13455  1.60e8  11896.    3165  0.235    23.5    17.3    1.22
6
## 5 E          3790  4.52e7  11924.    1090  0.288    28.8    20.0    1.40
6
## 6 F           753  7.10e6   9435.     252  0.335    33.5    23.9    0.955
20.9
## 7 G            82  9.47e5  11550.      31  0.378    37.8    26.5    0.958
24.9
## # … with 4 more variables: maxInt <dbl>, avgLoanAMt <dbl>, sumPmnt <dbl>,
```

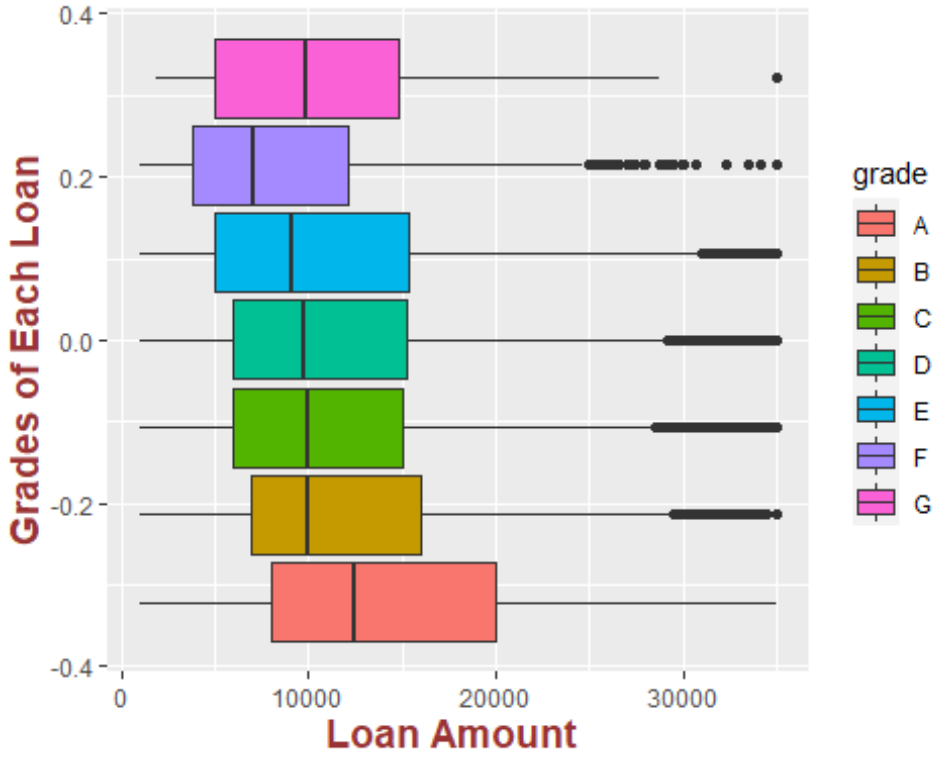
```
## #   avgPmnt <dbl>, and abbreviated variable names ¹numberOfLoans, ²
TotLoanAmt,
## #   ³MeanLoanAmt, ⁴defaults, ⁵defaultRate, ⁶default_percentage, ⁷
MeanIntRate,
## #   ⁸stdInterest

# Loan Amount Distribution
ggplot(lcdf, aes( x = loan_amnt)) + geom_histogram(aes(y=..density..),
colour="black", fill="white", bins=15)+ geom_density(alpha=.2,
fill="#FF6666") +  ggtitle("Distribution of Loan Amount Changing Bins ") +
xlab("Loan Amount ") + ylab("Number of Loans ") + theme(plot.title =
element_text(color="#993333", size=14, face="bold.italic"), axis.title.x =
element_text(color="#993333", size=14, face="bold"), axis.title.y =
element_text(color="#993333", size=14, face="bold"))
```



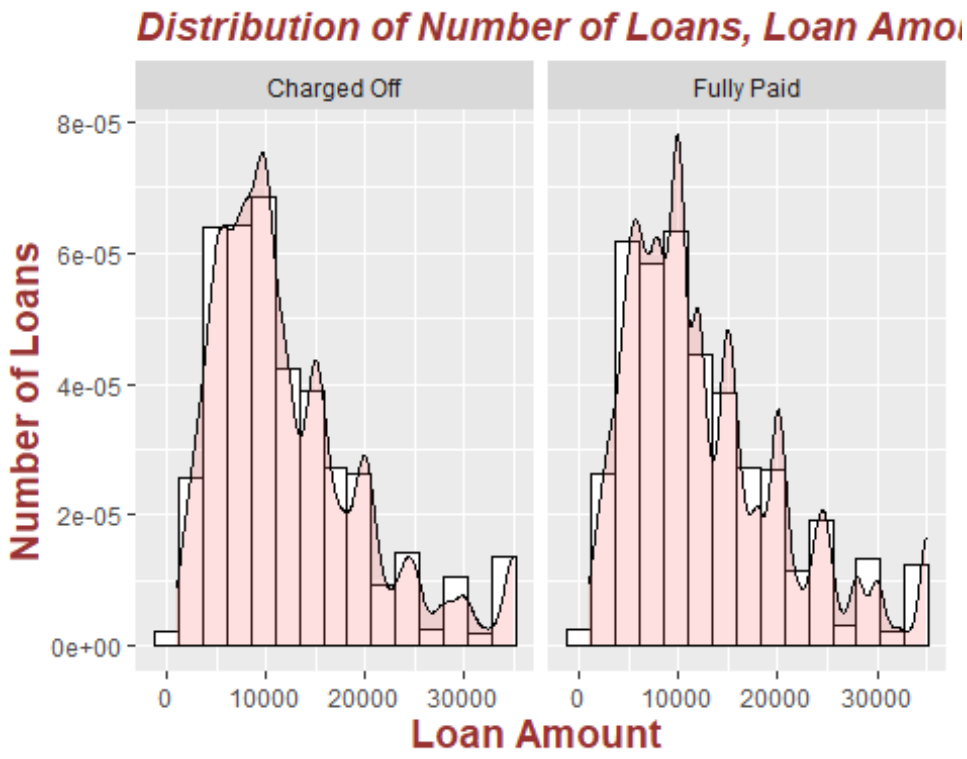

```
# Loan Amount Distribution by Grade

ggplot(lcdf, aes( x = loan_amnt)) + geom_histogram(aes(fill=grade)) +
ggtitle("Distribution of Loan Amount With Grade") + xlab("Loan Amount ") +
ylab("Number of Loans ") + theme(plot.title = element_text(color="#993333",
size=14, face="bold.italic"), axis.title.x = element_text(color="#993333",
size=14, face="bold"), axis.title.y = element_text(color="#993333", size=14,
face="bold"))
```




```
# Let us look at the distribution

ggplot(lcdf, aes( x = loan_amnt)) + geom_boxplot(aes(fill=grade)) +
xlab("Loan Amount ") + ylab("Grades of Each Loan  ") + theme(plot.title =
element_text(color="#993333", size=14, face="bold.italic"), axis.title.x =
element_text(color="#993333", size=14, face="bold"), axis.title.y =
element_text(color="#993333", size=14, face="bold"))
```

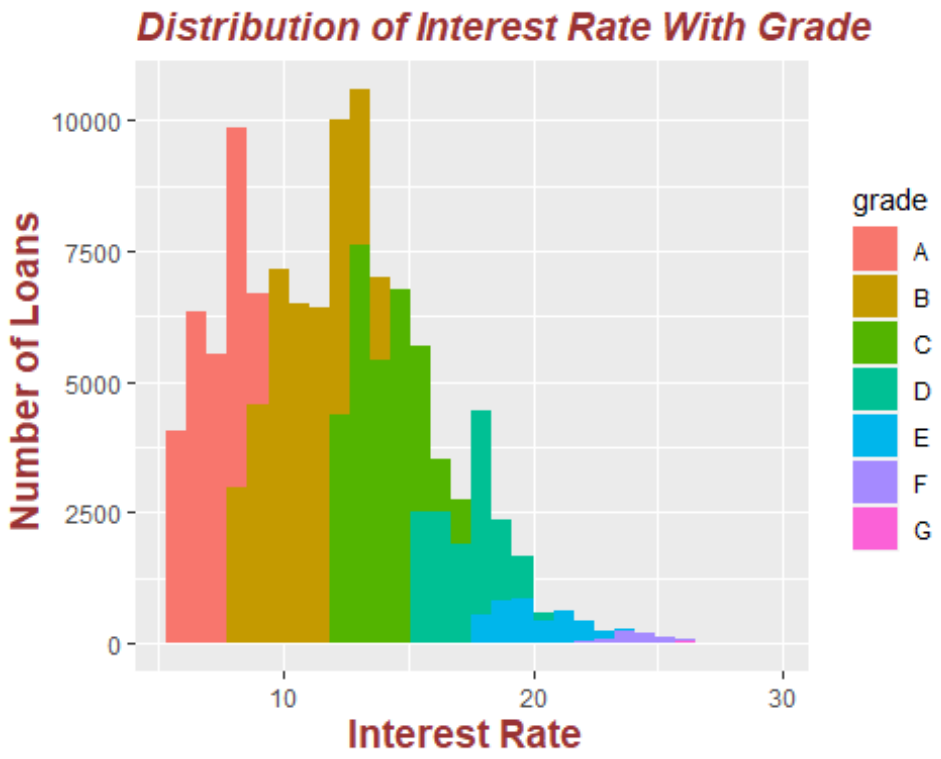



```
# Let us look at the loan amount along with loan status

ggplot(lcdf, aes( x = loan_amnt)) + geom_histogram(aes(y=..density..),
colour="black", fill="white", bins=15)+ geom_density(alpha=.2,
fill="#FF6666") +  ggtitle("Distribution of Number of Loans, Loan Amount with
Status ") + facet_wrap(~loan_status) + xlab("Loan Amount ") + ylab("Number of
Loans  ") + theme(plot.title = element_text(color="#993333", size=14,
face="bold.italic"), axis.title.x = element_text(color="#993333", size=14,
face="bold"), axis.title.y = element_text(color="#993333", size=14,
face="bold"))
```




```
# Let us look at the Interest Rate



ggplot(lcdf, aes( x = int_rate)) + geom_histogram(aes(y=..density..),
colour="black", fill="white")+ geom_density(alpha=.2, fill="#FF6666")
+ggtitle("Distribution of Interest Rate") + xlab("Interest Rate ") +
ylab("Number of Loans ") + theme(plot.title = element_text(color="#993333",
size=14, face="bold.italic"), axis.title.x = element_text(color="#993333",
size=14, face="bold"), axis.title.y = element_text(color="#993333", size=14,
face="bold"))
```


```
# Interest Rate with Grade
ggplot(lcdf, aes( x = int_rate)) + geom_histogram(aes(fill=grade)) +
ggtitle("Distribution of Interest Rate With Grade") + xlab("Interest Rate ")
+ ylab("Number of Loans ") + theme(plot.title = element_text(color="#993333",
size=14, face="bold.italic"), axis.title.x = element_text(color="#993333",
size=14, face="bold"), axis.title.y = element_text(color="#993333", size=14,
face="bold"))
```



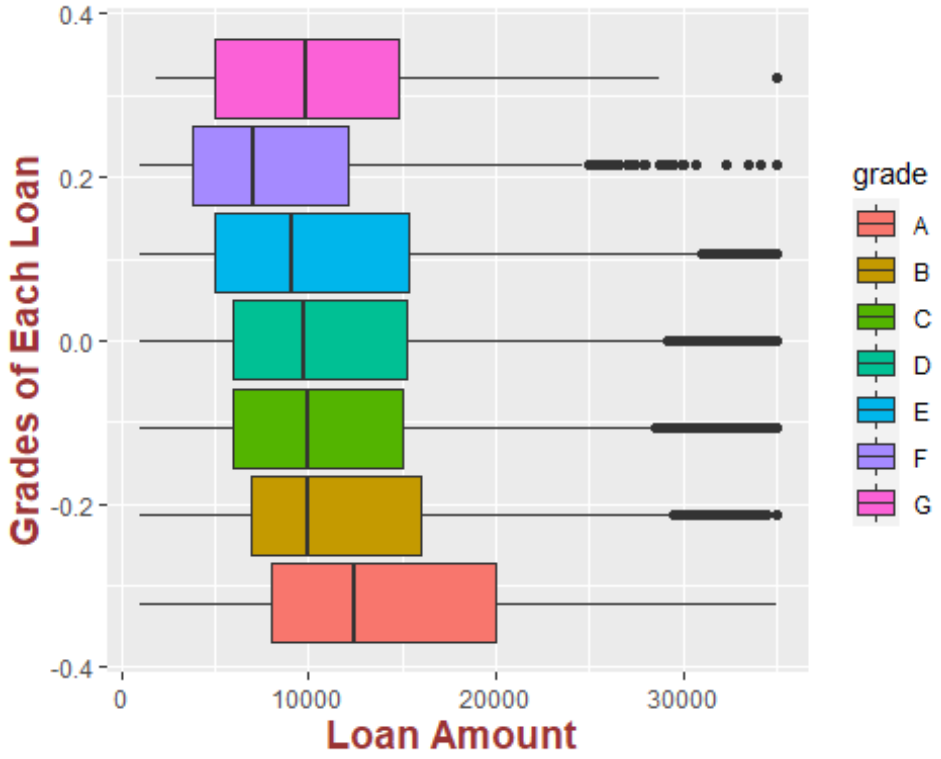
The default rate percentage increases from Grade A to H. Average Payments are more than average average loan amount in each grade. Yes these numbers surprise us-> when compare the returns of the different grade with NASDAQ for last 16 years (2007-2022 Current Year) which yields about 16.7 percent average - there are some grades which are not able to beat the market. Considering both NASDAQ and P2P market are highly volatile and even further risks in P2P we would expect them to give more average returns. If we had to invest in only one grade - depending on the risk apetite we would have chosen # grade C. Although it has a low average interest rate compared to other higher risk grades(D,E,F), it has an average interest rate of 14% which is sufficient to double the money in 5 years time.

The loan amount varies from 400 to 38,000. Most number of loans are of the amount approximately 12,000$. Most Grade G loans are of lesser amounts. he number of charged off loans are less in overall number, and it is evident in the graph # Both these distribution seem to be left skewed. In an ideal case these would have been normally distributed.There are loans which are higher than 30,000 and still paid.Also, there are loans of less than 10,000 and charged off

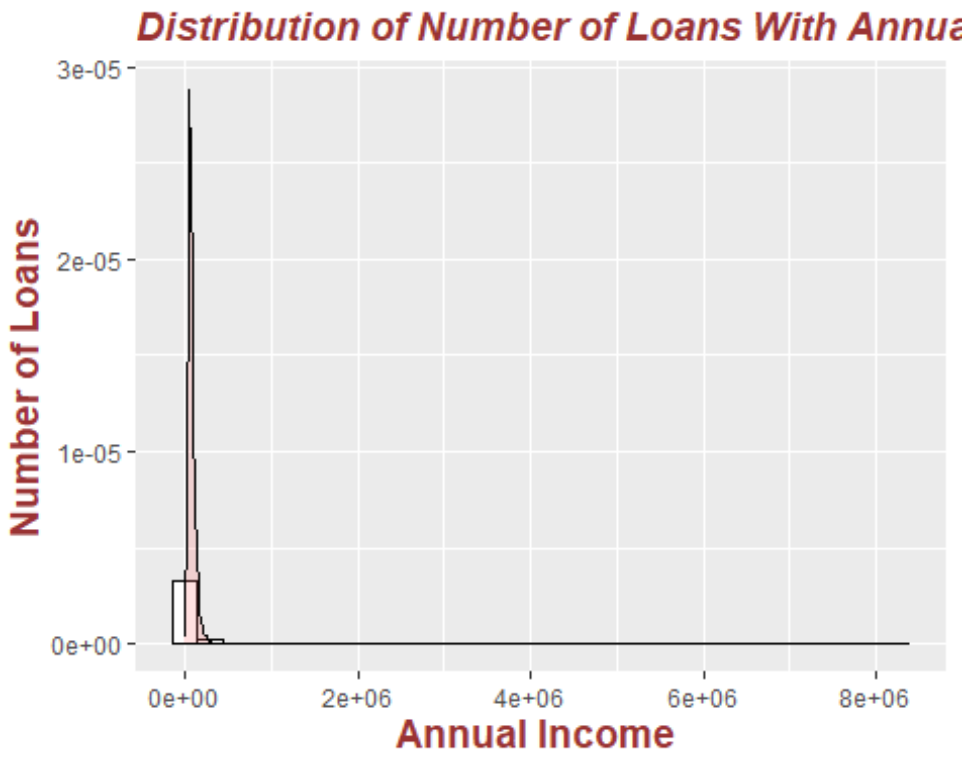
We can see that the average interest rate is higher in higher grades of loans. Intuitively we might be more interested in higher rates, however they come with trade off higher risk. The graph shows that the interest rate varies from 0-28. Most number of loans ~13-14% interest rate. Trend of interest rate with grade - Lower Grade corresponds to lower interest rate. Intutivetly, we should prefer a lower grade loan if both grades give same interest rate. The most common loan is grade B loan with a ~13% interest

Outliers Analysis

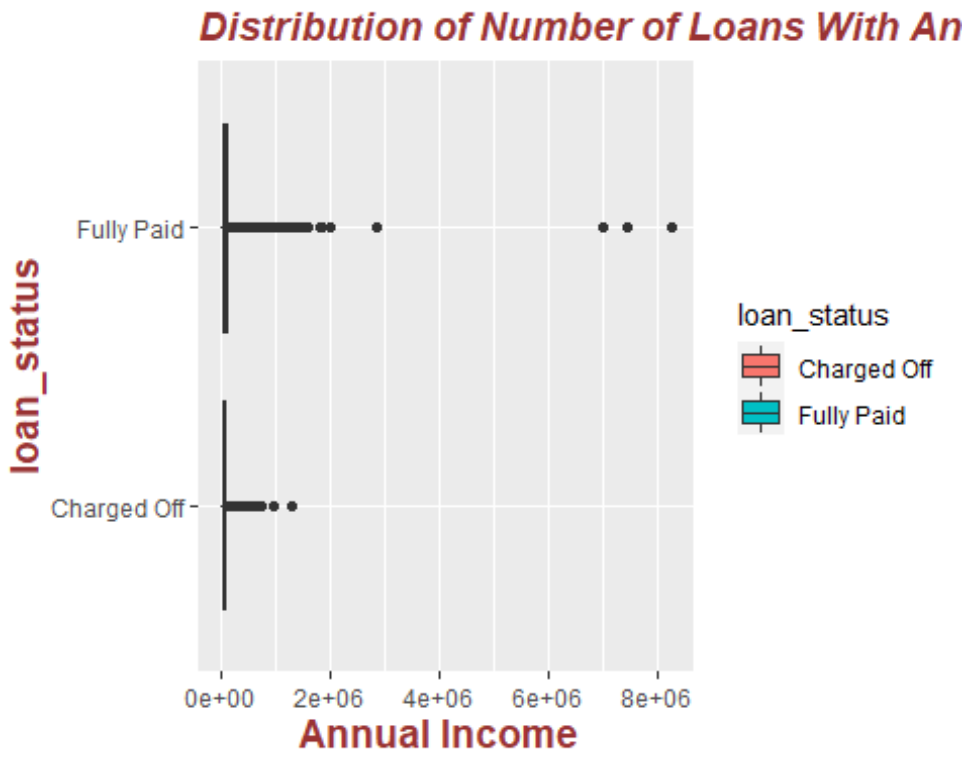
```
#Look at the variable summaries -- focus on a subset of the variables of
interest in your analyses & modeling


#lcdf %>% select_if(is.numeric) %>% summary()

# Let us look at the outliers in loan amount -

ggplot(lcdf, aes( x = loan_amnt)) + geom_boxplot(aes(fill=grade)) +
xlab("Loan Amount ") + ylab("Grades of Each Loan  ") + theme(plot.title =
element_text(color="#993333", size=14, face="bold.italic"), axis.title.x =
element_text(color="#993333", size=14, face="bold"), axis.title.y =
element_text(color="#993333", size=14, face="bold"))
```




```
# Let us look at the annual income

ggplot(lcdf, aes( x = annual_inc)) +  geom_histogram(aes(y=..density..),
colour="black", fill="white")+ geom_density(alpha=.2, fill="#FF6666") +
ggtitle("Distribution of Number of Loans With Annual Income ") + xlab("Annual
Income ") + ylab("Number of Loans  ") + theme(plot.title =
element_text(color="#993333", size=14, face="bold.italic"), axis.title.x =
element_text(color="#993333", size=14, face="bold"), axis.title.y =
element_text(color="#993333", size=14, face="bold"))
```




```
# Let us check how are these very high income associated with loans status

ggplot(lcdf, aes( x = annual_inc, y=loan_status)) +
geom_boxplot(aes(fill=loan_status)) +  ggtitle("Distribution of Number of
Loans With Annual Income By Loan Status - Before Removing Extreme Outliers")
+ xlab("Annual Income ") +  theme(plot.title = element_text(color="#993333",
size=14, face="bold.italic"), axis.title.x = element_text(color="#993333",
size=14, face="bold"), axis.title.y = element_text(color="#993333", size=14,
face="bold"))
```



Yes, there are outliers. However to remove them we should check the frequency and also see the business use case these outliers might be justified given the fact that loan amount can vary.

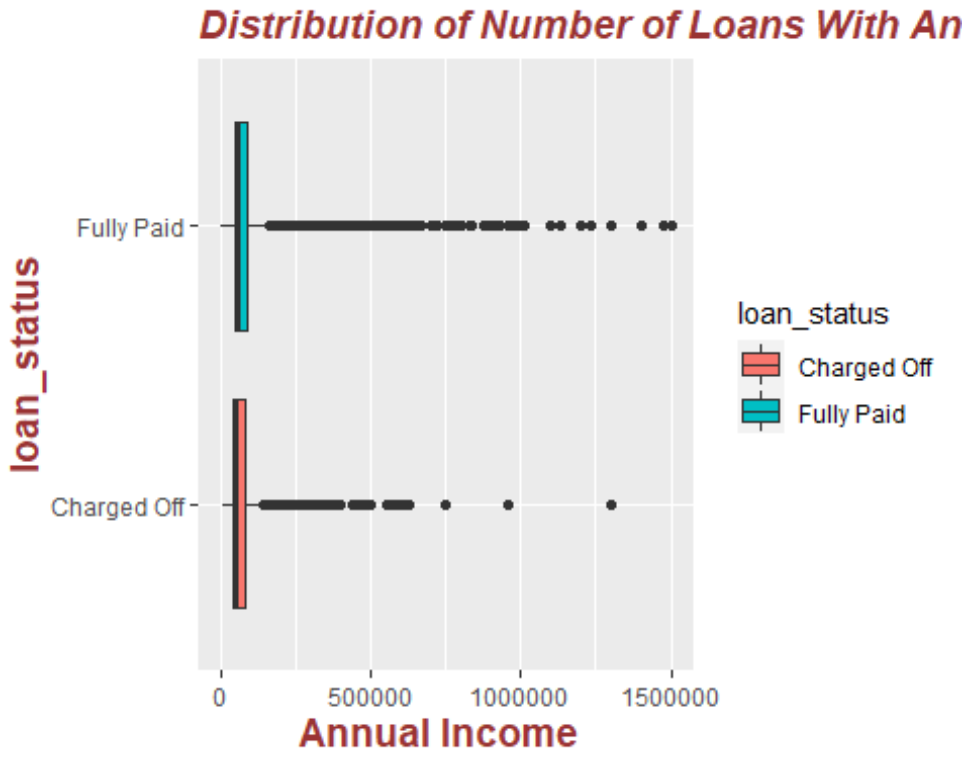
For annual income the data seems to really skewed towards the left, very few loans have the income more than 1.5 Milliion. A person coming to lending club for loan with income more than 1.5 million might be suspicious. It is logical to think about why would a person need a loan with 1.5 million income. Hence we will remove thes 9 observation. We could alternatively assignment a maximum value, since we have 110k data point we can remove 9 rows

The very high income cases are for paid-off loans. # We can exclude them, however we do so we might not have a decision tree model which predicts the hypothesis that high income people pay off the loan in most cases.Going with the use case we will discard and keep them in a separate dataframe.We shall observe what difeerence it makes to out models in the later part. Compared to the 110k data size the number looks really small, hence we will remove these

Removing the outliers for annual income

```
## Chunk 12 <For knitting of .rmd file>


lcdf <- lcdf %>% filter(annual_inc <= 1500000)

# Let us look at the new distribution of annual income after outlier removal

ggplot(lcdf, aes( x = annual_inc, y=loan_status)) +
```

```
geom_boxplot(aes(fill=loan_status)) +  ggtitle("Distribution of Number of
Loans With Annual Income By Loan Status - After Removing Extreme Outliers ")
+ xlab("Annual Income ") +  theme(plot.title = element_text(color="#993333",
size=14, face="bold.italic"), axis.title.x = element_text(color="#993333",
size=14, face="bold"), axis.title.y = element_text(color="#993333", size=14,
face="bold"))
```

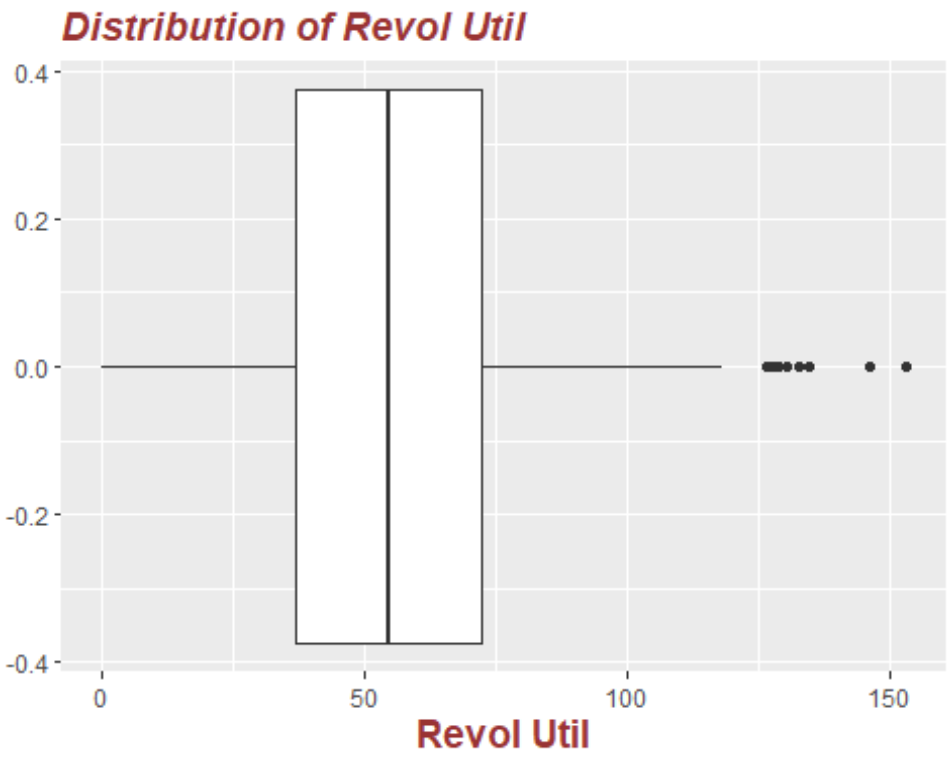


The plot looks much cleaner, and inference can be drawn from the the above as we have removed those outliers.We might argue to the fact that data still has outliers, but removing the ones above 1.5 IQR now we might lose essential information. However this was the case when we removed observations above 1.5 million, but they were just 9 observations in the 109k observations.

Revol util

Ratio of current balance/ high credit limit.

```
### Chunk 13

ggplot(lcdf, aes( x = revol_util)) + geom_boxplot() +  ggtitle("Distribution
of Revol Util ") + xlab("Revol Util") +  theme(plot.title =
element_text(color="#993333", size=14, face="bold.italic"), axis.title.x =
element_text(color="#993333", size=14, face="bold"), axis.title.y =
element_text(color="#993333", size=14, face="bold"))
```




```
# Identified outliers by boxplot
out_ru <- boxplot(lcdf$revol_util, plot=FALSE)$out
#Let us look at these examples
out_ru_i <-which(lcdf$revol_util %in% out_ru)
lcdf[out_ru_i,]
```

```
## # A tibble: 9 × 145
##   id    member_id loan_amnt funded…¹ funde…² term  int_r…³ insta…⁴ grade
sub_g…⁵
##   <lgl> <lgl>         <dbl>    <dbl>   <dbl> <chr>   <dbl>   <dbl> <chr>
<chr>
## 1 NA    NA             3475     3475    3475 36 m…   18.9     127. D
D4
## 2 NA    NA            12600    12600   12600 36 m…    8.39    397. A
A5
## 3 NA    NA            15000    15000   15000 36 m…   12.0     498. C
C1
## 4 NA    NA            20000    20000   20000 36 m…   11.7     661. B
B4
## 5 NA    NA            35000    35000   35000 36 m…   19.5    1292. D
D4
## 6 NA    NA             5000     5000    5000 36 m…    9.49    160. B
B2
## 7 NA    NA            35000    35000   35000 36 m…   25.8    1407. G
G2
## 8 NA    NA            25000    25000   25000 36 m…   12.6     837. C
C2
```

```
## 9 NA    NA            10000    10000   10000 36 m…    9.99    323. B
B3
## # … with 135 more variables: emp_title <chr>, emp_length <chr>,
## #   home_ownership <chr>, annual_inc <dbl>, verification_status <chr>,
## #   issue_d <dttm>, loan_status <chr>, pymnt_plan <chr>, url <lgl>, desc
<lgl>,
## #   purpose <chr>, title <chr>, zip_code <chr>, addr_state <chr>, dti
<dbl>,
## #   delinq_2yrs <dbl>, earliest_cr_line <chr>, inq_last_6mths <dbl>,
## #   mths_since_last_delinq <dbl>, mths_since_last_record <dbl>, open_acc
<dbl>,
## #   pub_rec <dbl>, revol_bal <dbl>, revol_util <dbl>, total_acc <dbl>, …
```

```
# We will remove these 9 outliers

lcdf <- lcdf [-out_ru_i, ]
```

Recoveries and Total Payment Analysis

```
# Recoveries - post a loan charged off gross amount recovered
 # Checking if recoveries are only for charged off loans

lcdf %>% group_by(loan_status) %>%summarise(Rec=sum(recoveries))
```

```
## # A tibble: 2 × 2
##   loan_status       Rec
##   <chr>           <dbl>
## 1 Charged Off 14231328.
## 2 Fully Paid         0
```

```
lcdf %>% group_by(loan_status) %>%summarise(Sum_Rec=sum(recoveries),
TotPmnt=sum(total_pymnt), total_rec_prncp=sum(total_rec_prncp),
total_rec_int=sum(total_rec_int), total_rec_late_fee=sum(total_rec_late_fee))
```

```
## # A tibble: 2 × 6
##   loan_status   Sum_Rec     TotPmnt total_rec_prncp total_rec_int
total_rec_la…¹
##   <chr>           <dbl>       <dbl>           <dbl>         <dbl>
<dbl>
## 1 Charged Off 14231328.  121142712.       79860072.     26993115.
58197.
## 2 Fully Paid         0  1387729348.     1204556474.    183100781.
72092.
## # … with abbreviated variable name ¹total_rec_late_fee
```

Hence recoveries are only for charged off loans, this also goes with the general idea of recovery of credit for any loan it will only be for the charged off if it has to be there. Sometimes recovery might not be present for the charged off loans as well. This is the case where there has been a loss.

he way to calulate recovered amount in terms of charged loans ='total_pymnt'= 'total_rec_prncp'+'total_rec_int'+'total_rec_late_fee'+'recoveries'

Actual Return

```
# Let us look at some columns

lcdf %>% select(loan_status, int_rate, funded_amnt, total_pymnt) %>% head()
```

```
## # A tibble: 6 × 4
##   loan_status int_rate funded_amnt total_pymnt
##   <chr>          <dbl>       <dbl>       <dbl>
## 1 Fully Paid     23.0         4400       6120.
## 2 Fully Paid     22.0         5850       6377.
## 3 Fully Paid      6.24        5000       5496.
## 4 Fully Paid     15.0         1600       1840.
## 5 Fully Paid      9.17       16000      18128.
## 6 Fully Paid      8.18        3000       3394.
```

```
# We will use the following to calculate annualized return
#annReturn = [(Total Payment  - funded amount)/funded amount]*12/36*100

lcdf$annRet <- ((lcdf$total_pymnt -
lcdf$funded_amnt)/lcdf$funded_amnt)*(12/36)*100

# Returns for charged off and fully paid loans
lcdf  %>% group_by(loan_status) %>% summarise(avgRet=mean(annRet),
stdRet=sd(annRet), minRet=min(annRet), maxRet=max(annRet))
```

```
## # A tibble: 2 × 5
##   loan_status avgRet stdRet minRet maxRet
##   <chr>        <dbl>  <dbl>  <dbl>  <dbl>
## 1 Charged Off -12.0    9.35  -33.3   14.4
## 2 Fully Paid    5.16   2.43    0     18.0
```

```
# Do charged off loans have negative returns -

lcdf %>% select(loan_status, int_rate, funded_amnt, total_pymnt, annRet) %>%
filter(annRet < 0) %>% count(loan_status)
```

```
## # A tibble: 1 × 2
##   loan_status     n
##   <chr>       <int>
## 1 Charged Off 13539
```

What is surprising here is the fact that the avg return rate differ significantly fro average interest rate . The minimum return rate for some loans which are fully paid can go as minimum as 0. This might be because some loans which are paid off are paid off earlier than the expected date.

Returns from loans - Fully Paid and Charged off

```
## Chunk 16
# Fully Paid
```

```
lcdf %>% filter( loan_status == "Fully Paid") %>% group_by(grade) %>%
summarise(nLoans=n(),  avgInterest= mean(int_rate),
avgLoanAmt=mean(loan_amnt), avgPmnt=mean(total_pymnt), avgRet=mean(annRet),
minRet=min(annRet), maxRet=max(annRet))
```

```
## # A tibble: 7 × 8
##   grade nLoans avgInterest avgLoanAmt avgPmnt avgRet   minRet maxRet
##   <chr>  <int>       <dbl>      <dbl>   <dbl>  <dbl>    <dbl>  <dbl>
## 1 A      23480        7.19     14383.  15772.   3.19 0          5.19
## 2 B      33596       10.8      12546.  14357.   4.80 0.000333   8.18
## 3 C      23935       13.9      12038.  14244.   6.06 0.0134    10.5
## 4 D      10288       17.3      11745.  14401.   7.53 0         11.9
## 5 E       2700       20.0      11602.  14549.   8.64 0.0194    14.0
## 6 F        501       23.9       9134.  11975.  10.6  0.0255    18.0
## 7 G         50       26.5      10512   14144.  11.7  0.422     17.0
```

```
# Adding subgrade
```

```
lcdf %>% filter( loan_status == "Fully Paid") %>% group_by(sub_grade) %>%
summarise(nLoans=n(),  avgInterest= mean(int_rate),
avgLoanAmt=mean(loan_amnt), avgPmnt=mean(total_pymnt), avgRet=mean(annRet),
minRet=min(annRet), maxRet=max(annRet))
```

```
## # A tibble: 35 × 8
##    sub_grade nLoans avgInterest avgLoanAmt avgPmnt avgRet    minRet maxRet
##    <chr>      <int>       <dbl>      <dbl>   <dbl>  <dbl>     <dbl>  <dbl>
##  1 A1          3934        5.70     14098.  15167.   2.50 0.00518     3.34
##  2 A2          3745        6.43     13958.  15143.   2.82 0.0000476   3.72
##  3 A3          3851        7.13     14476.  15862.   3.16 0.0000208   4.20
##  4 A4          5388        7.52     14749.  16239.   3.35 0           4.70
##  5 A5          6562        8.28     14441.  16056.   3.69 0.00840     5.19
##  6 B1          6285        8.96     12935.  14480.   3.97 0.00909     6.62
##  7 B2          6922       10.0      12912.  14643.   4.44 0.0102      6.52
##  8 B3          7324       11.0      12545.  14387.   4.89 0.0296      7.01
##  9 B4          6829       11.9      12263.  14219.   5.26 0.000333    8.00
## 10 B5          6236       12.4      12058.  14033.   5.44 0.0132      8.18
## # … with 25 more rows
```

```
# Charged Off
```

```
lcdf %>% filter( loan_status == "Charged Off") %>% group_by(grade) %>%
summarise(nLoans=n(), avgInterest= mean(int_rate),
avgLoanAmt=mean(loan_amnt), avgPmnt=mean(total_pymnt), avgRet=mean(annRet),
minRet=min(annRet), maxRet=max(annRet))
```

```
## # A tibble: 7 × 8
##   grade nLoans avgInterest avgLoanAmt avgPmnt avgRet minRet maxRet
##   <chr>  <int>       <dbl>      <dbl>   <dbl>  <dbl>  <dbl>  <dbl>
## 1 A       1369        7.49     13747.   8781.  -12.1  -32.3   5.80
## 2 B       4264       11.0      12195.   7939.  -11.7  -33.3  13.8
```

```
## 3 C       5206       14.0      12085.   7792.  -11.9  -33.3   9.54
## 4 D       3164       17.2      12376.   7719.  -12.6  -33.3  11.3
## 5 E       1090       20.0      12722.   7858.  -12.6  -33.3  11.7
## 6 F        252       23.9      10032.   5931.  -12.2  -32.0  14.4
## 7 G         31       26.4      12469.   7056.  -15.5  -28.6   4.86
```

```
# Adding Subgrade

lcdf %>% filter( loan_status == "Charged Off") %>% group_by(sub_grade) %>%
summarise(nLoans=n(), avgInterest= mean(int_rate),
avgLoanAmt=mean(loan_amnt), avgPmnt=mean(total_pymnt), avgRet=mean(annRet),
minRet=min(annRet), maxRet=max(annRet))
```

```
## # A tibble: 34 × 8
##    sub_grade nLoans avgInterest avgLoanAmt avgPmnt avgRet minRet maxRet
##    <chr>      <int>       <dbl>      <dbl>   <dbl>  <dbl>  <dbl>  <dbl>
##  1 A1           104        5.69     13727.   8584.  -12.5  -32.3   2.00
##  2 A2           161        6.43     13333.   8150.  -12.9  -31.4   2.74
##  3 A3           193        7.14     13754.   8798.  -12.3  -31.3   4.05
##  4 A4           385        7.51     13878.   8969.  -11.6  -32.3   4.79
##  5 A5           526        8.27     13778.   8869.  -12.0  -32.3   5.80
##  6 B1           575        8.95     11918.   7733.  -11.7  -31.4   5.35
##  7 B2           775        9.98     12405.   8080.  -11.5  -33.3   6.19
##  8 B3           958       11.0      12448.   8058.  -11.8  -33.3   5.72
##  9 B4           928       11.8      11966.   7751.  -11.7  -33.3   6.74
## 10 B5          1028       12.4      12163.   8007.  -11.6  -33.3  13.8
## # … with 24 more rows
```

Checking if loans paid paid early -

```
# Chunk 17
# 2 dates we will use - payment and issue date

head(lcdf[, c("last_pymnt_d", "issue_d")])
```

```
## # A tibble: 6 × 2
##   last_pymnt_d issue_d
##   <chr>        <dttm>
## 1 Mar-2018     2015-03-01 00:00:00
## 2 Mar-2015     2014-05-01 00:00:00
## 3 Sep-2018     2015-09-01 00:00:00
## 4 Jun-2015     2014-05-01 00:00:00
## 5 Jun-2017     2015-05-01 00:00:00
## 6 Oct-2018     2015-11-01 00:00:00
```

```
# Bringing them to a consistent format
lcdf$last_pymnt_d<-paste(lcdf$last_pymnt_d, "-01", sep = "")
lcdf$last_pymnt_d<-parse_date_time(lcdf$last_pymnt_d,  "myd")
```

```r
#Check their format now
head(lcdf[, c("last_pymnt_d", "issue_d")])
```

```
## # A tibble: 6 × 2
##   last_pymnt_d        issue_d
##   <dttm>              <dttm>
## 1 2018-03-01 00:00:00 2015-03-01 00:00:00
## 2 2015-03-01 00:00:00 2014-05-01 00:00:00
## 3 2018-09-01 00:00:00 2015-09-01 00:00:00
## 4 2015-06-01 00:00:00 2014-05-01 00:00:00
## 5 2017-06-01 00:00:00 2015-05-01 00:00:00
## 6 2018-10-01 00:00:00 2015-11-01 00:00:00
```

```r
# Creating actual term column - If loan is charged off by default - 3 years
lcdf$actualTerm <- ifelse(lcdf$loan_status=="Fully Paid",
as.duration(lcdf$issue_d  %--% lcdf$last_pymnt_d)/dyears(1), 3)

# We know using simple interest Total =  principle + pnr/100
# Hence r = (Total - principle)/principle * 100/n

# Then, considering this actual term, the actual annual return is

lcdf$actualReturn <- ifelse(lcdf$actualTerm>0, ((lcdf$total_pymnt -
lcdf$funded_amnt)/lcdf$funded_amnt)*(1/lcdf$actualTerm)*100, 0)

lcdf %>% select(loan_status, int_rate, funded_amnt, total_pymnt, annRet,
actualTerm, issue_d,last_pymnt_d) %>%  head()
```

```
## # A tibble: 6 × 8
##   loan_status int_rate funded_amnt total_py…¹ annRet actua…² issue_d
##   <chr>          <dbl>       <dbl>      <dbl>  <dbl>   <dbl> <dttm>
## 1 Fully Paid     23.0         4400      6120.  13.0    3.00  2015-03-01
00:00:00
## 2 Fully Paid     22.0         5850      6377.   3.00   0.832 2014-05-01
00:00:00
## 3 Fully Paid      6.24        5000      5496.   3.31   3.00  2015-09-01
00:00:00
## 4 Fully Paid     15.0         1600      1840.   4.99   1.08  2014-05-01
00:00:00
## 5 Fully Paid      9.17       16000     18128.   4.43   2.09  2015-05-01
00:00:00
## 6 Fully Paid      8.18        3000      3394.   4.37   2.92  2015-11-01
00:00:00
## # … with 1 more variable: last_pymnt_d <dttm>, and abbreviated variable
names
## #   ¹total_pymnt, ²actualTerm
```

```r
# Checking the same for charged off loans
lcdf %>% select(loan_status, int_rate, funded_amnt, total_pymnt, annRet,
actualTerm, actualReturn) %>% filter(loan_status=="Charged Off") %>% head()
```

```
## # A tibble: 6 × 7
##   loan_status int_rate funded_amnt total_pymnt annRet actualTerm
actualReturn
##   <chr>          <dbl>       <dbl>       <dbl>  <dbl>      <dbl>
<dbl>
## 1 Charged Off     13.4        6500       2701. -19.5           3       -
19.5
## 2 Charged Off     13.4       15000       9898. -11.3           3       -
11.3
## 3 Charged Off     14.0        9000       6765.  -8.28          3        -
8.28
## 4 Charged Off     10.2        5000       3013. -13.2           3       -
13.2
## 5 Charged Off     17.9       10575       5295. -16.6           3       -
16.6
## 6 Charged Off      7.9       27000       3971. -28.4           3       -
28.4
```

Additional Analysis on returns

```
# Chunk 17

# For cost-based performance, we may want to see the average interest rate,
and the average of proportion of loan amount paid back, grouped by
loan_status

lcdf%>% group_by(loan_status) %>% summarise(  meanintRate=mean(int_rate),
meanRet=mean((total_pymnt-
funded_amnt)/funded_amnt),meanRetPer=mean((total_pymnt-
funded_amnt)/funded_amnt)*100, sumTotalpymt = sum(total_pymnt), sumFundedamnt
= sum(funded_amnt), term=mean(actualTerm)  )

## # A tibble: 2 × 7
##   loan_status meanintRate meanRet meanRetPer sumTotalpymt sumFundedamnt
term
##   <chr>             <dbl>   <dbl>      <dbl>        <dbl>         <dbl>
<dbl>
## 1 Charged Off        13.9  -0.361      -36.1   121142712.     189668875  3
## 2 Fully Paid         11.8   0.155       15.5  1387729348.    1204556475
2.14

# Checking the same by grade along with loan status

lcdf%>% group_by(loan_status, grade) %>% summarise(
intRate=mean(int_rate),meanRet=mean((total_pymnt-funded_amnt)/funded_amnt),
meanRetPer=mean((total_pymnt-funded_amnt)/funded_amnt)*100,sumTotalpymt =
sum(total_pymnt), sumFundedamnt = sum(funded_amnt), term=mean(actualTerm)   )

## # A tibble: 14 × 8
## # Groups:   loan_status [2]
```

```
##    loan_status grade intRate meanRet meanRetPer sumTotalpymt sumFundedamnt
term
##    <chr>       <chr>   <dbl>   <dbl>      <dbl>        <dbl>         <dbl>
<dbl>
##  1 Charged Off A        7.49 -0.362      -36.2     12021258.      18819050
3
##  2 Charged Off B       11.0  -0.350      -35.0     33852516.      51996225
3
##  3 Charged Off C       14.0  -0.357      -35.7     40567130.      62913550
3
##  4 Charged Off D       17.2  -0.377      -37.7     24423334.      39158675
3
##  5 Charged Off E       20.0  -0.378      -37.8      8565088.      13866675
3
##  6 Charged Off F       23.9  -0.365      -36.5      1494657.       2528175
3
##  7 Charged Off G       26.4  -0.466      -46.6       218729.        386525
3
##  8 Fully Paid  A        7.19  0.0957       9.57   370317452.     337701500
2.21
##  9 Fully Paid  B       10.8   0.144       14.4    482339194.     421484600
2.16
## 10 Fully Paid  C       13.9   0.182       18.2    340925023.     288108225
2.08
## 11 Fully Paid  D       17.3   0.226       22.6    148159630.     120834850
2.06
## 12 Fully Paid  E       20.0   0.259       25.9     39281121.      31325525
2.03
## 13 Fully Paid  F       23.9   0.317       31.7      5999721.       4576175
2.10
## 14 Fully Paid  G       26.5   0.350       35.0       707207.        525600
2.12
```

```
# For Fully Paid loans, is the average value of totRet what you'd expect,
considering the average value for intRate?
```

```
lcdf %>% group_by(loan_status) %>% summarise(avgInt=mean(int_rate),
avgRet=mean(actualReturn),avgTerm=mean(actualTerm))
```

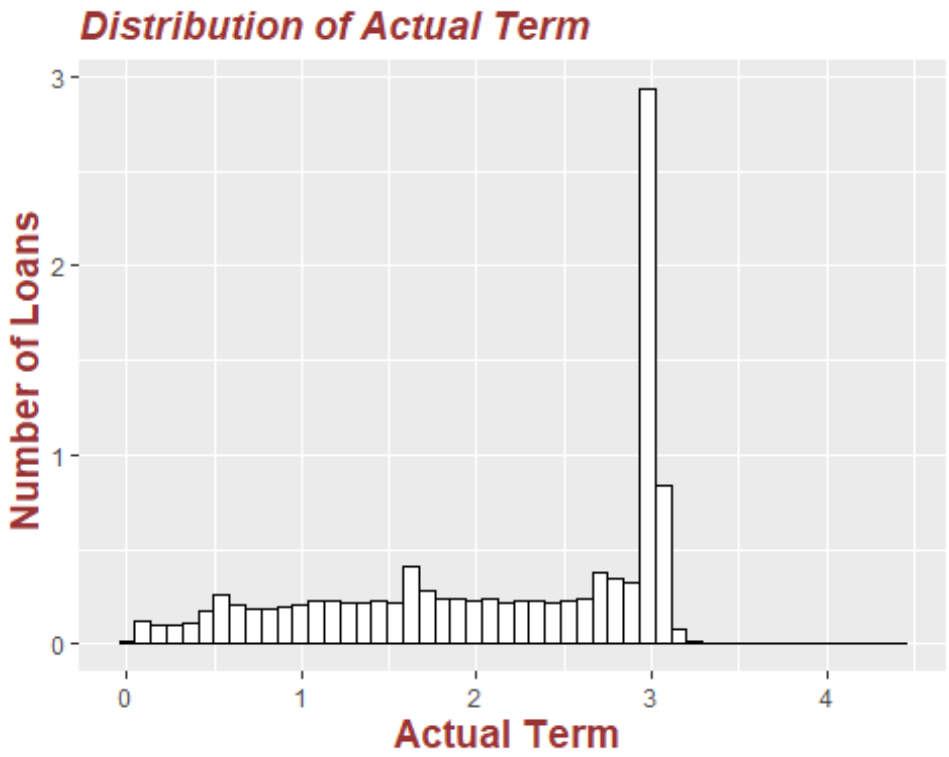
```
## # A tibble: 2 × 4
##   loan_status avgInt avgRet avgTerm
##   <chr>        <dbl>  <dbl>   <dbl>
## 1 Charged Off   13.9 -12.0     3
## 2 Fully Paid    11.8   8.02    2.14
```

We also observe the actual term for loan is not 3 years in case of fully paid loans. Indeed some loans are fully paid earlier than 3 years. # Charged off loans are expected to have negative return irrespective of the grade. Higher graded have higher loss / negative mean return rate. But the distribution of the return is only between -0.36 - -0.466 in case of

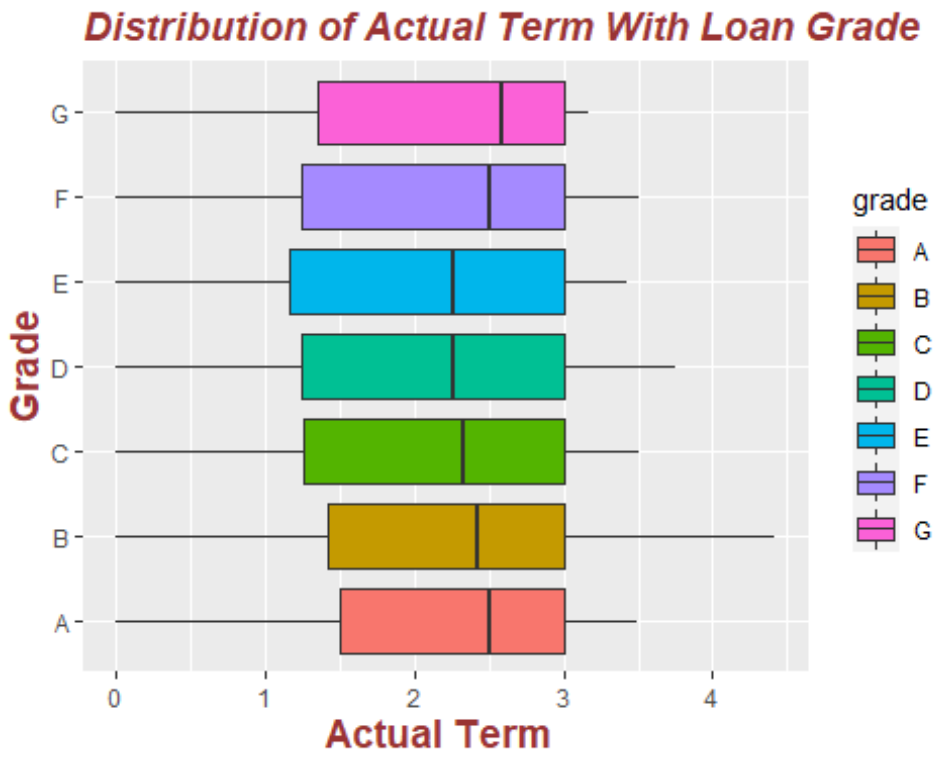
charged off loans.In case of fully paid loans, higher grades give higher average return. The range of return is higher 0.09 to 0.349. We would want our investor to get the best returns and minimize losses at the same time.

Distribution of actual term

```
# Chunk 21


ggplot(lcdf %>% filter(loan_status=='Fully Paid'), aes( x = actualTerm)) +
geom_histogram(aes(y=..density..), colour="black", fill="white", bins=50)
+ggtitle("Distribution of Actual Term ") + xlab("Actual Term ") +
ylab("Number of Loans ") + theme(plot.title = element_text(color="#993333",
size=14, face="bold.italic"), axis.title.x = element_text(color="#993333",
size=14, face="bold"), axis.title.y = element_text(color="#993333", size=14,
face="bold"))
```

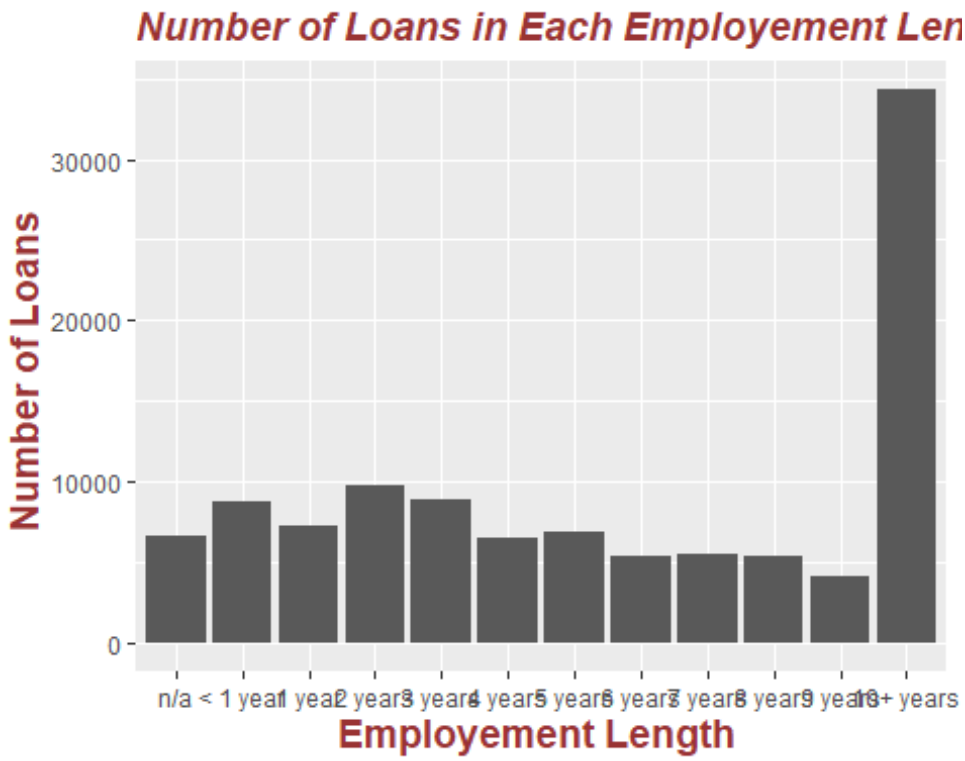



```
ggplot(lcdf %>% filter(loan_status=='Fully Paid'), aes( x = actualTerm,
y=grade)) + geom_boxplot(aes(fill=grade)) + ggtitle("Distribution of Actual
Term With Loan Grade ")+
xlab("Actual Term ") + ylab("Grade") + theme(plot.title =
element_text(color="#993333", size=14, face="bold.italic"), axis.title.x =
element_text(color="#993333", size=14, face="bold"), axis.title.y =
element_text(color="#993333", size=14, face="bold"))
```



Employment Length

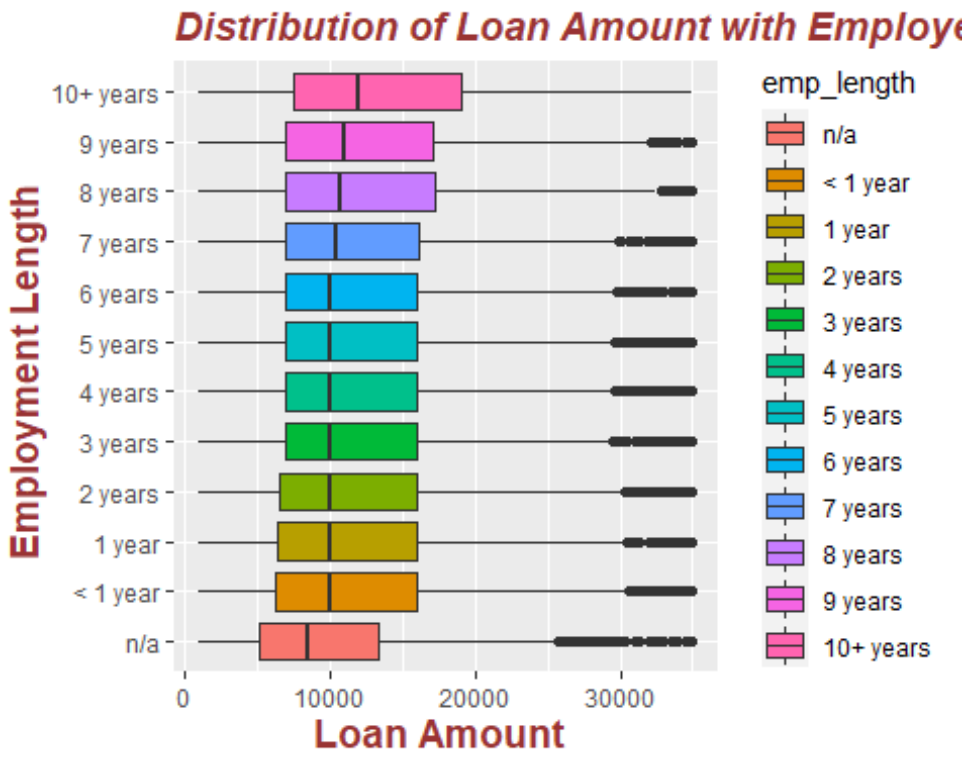
```
# Arranging them since they
lcdf$emp_length <- factor(lcdf$emp_length, levels=c("n/a", "< 1 year","1
year","2 years", "3 years" ,  "4 years",   "5 years",   "6 years",   "7
years" ,  "8 years", "9 years", "10+ years" ))

# Number of loans in each employment length
ggplot(data = lcdf, aes(x = emp_length)) + geom_bar() + ggtitle("Number of
Loans in Each Employement Length ") + xlab("Employement Length ") +
ylab("Number of Loans ")+ theme(plot.title = element_text(color="#993333",
size=14, face="bold.italic"), axis.title.x = element_text(color="#993333",
size=14, face="bold"), axis.title.y = element_text(color="#993333", size=14,
face="bold"))
```




```
# Results in a table

table(lcdf$loan_status, lcdf$emp_length)

##
##                 n/a < 1 year 1 year 2 years 3 years 4 years 5 years 6
years
##   Charged Off  1345     1268   1097    1327    1265     895     983
757
##   Fully Paid   5300     7515   6237    8471    7622    5607    5989
4692
##
##               7 years 8 years 9 years 10+ years
##   Charged Off     737     724     598      4380
##   Fully Paid     4837    4731    3589     29960

# Calculating the proportion of defaults across employment length


lcdf %>% group_by(emp_length) %>% summarise(nLoans=n(),
defaults=sum(loan_status=="Charged Off"),
defaultPercentage=defaults/nLoans*100, avgIntRate=mean(int_rate),
avgLoanAmt=mean(loan_amnt),  avgActRet = mean(actualReturn),
avgActTerm=mean(actualTerm))

## # A tibble: 12 × 8
##    emp_length nLoans defaults defaultPercentage avgInt…¹ avgLo…² avgAc…³
```

```
avgAc…⁴
##    <fct>       <int>    <int>             <dbl>    <dbl>   <dbl>   <dbl>
<dbl>
##  1 n/a          6645     1345              20.2     12.5  10251.    3.94
2.42
##  2 < 1 year     8783     1268              14.4     12.1  12108.    5.01
2.25
##  3 1 year       7334     1097              15.0     12.2  12080.    5.11
2.25
##  4 2 years      9798     1327              13.5     12.1  12183.    5.39
2.23
##  5 3 years      8887     1265              14.2     12.1  12344.    5.22
2.26
##  6 4 years      6502      895              13.8     12.1  12661.    5.26
2.25
##  7 5 years      6972      983              14.1     12.1  12513.    5.19
2.26
##  8 6 years      5449      757              13.9     12.2  12475.    5.25
2.26
##  9 7 years      5574      737              13.2     12.1  12656.    5.40
2.25
## 10 8 years      5455      724              13.3     12.0  12935.    5.43
2.24
## 11 9 years      4187      598              14.3     12.1  12974.    5.17
2.23
## 12 10+ years   34340     4380              12.8     11.8  13662.    5.41
2.25
## # … with abbreviated variable names ¹avgIntRate, ²avgLoanAmt, ³avgActRet,
## #   ⁴avgActTerm
```

```
# Plot for Distribution of Loan Amount with Employment Length

ggplot(lcdf, aes( x = loan_amnt, y=emp_length)) +
geom_boxplot(aes(fill=emp_length)) +
xlab("Loan Amount ") + ylab("Employment Length ")+ggtitle("Distribution of
Loan Amount with Employement Length") + theme(plot.title =
element_text(color="#993333", size=14, face="bold.italic"), axis.title.x =
element_text(color="#993333", size=14, face="bold"), axis.title.y =
element_text(color="#993333", size=14, face="bold"))
```



The internal percentage of default within a grade differ by emp length. This is a good factor to understand how employment length plays a role in defaults and returns

Loan Purpose

```
# Checking number of loans by purpose


lcdf %>% group_by(purpose) %>% tally()
```

```
## # A tibble: 13 × 2
##    purpose                n
##    <chr>              <int>
##  1 car                 1083
##  2 credit_card        27091
##  3 debt_consolidation 63277
##  4 home_improvement    6190
##  5 house                432
##  6 major_purchase      2111
##  7 medical             1170
##  8 moving               755
##  9 other               5684
## 10 renewable_energy      65
## 11 small_business      1117
## 12 vacation             753
## 13 wedding              198
```

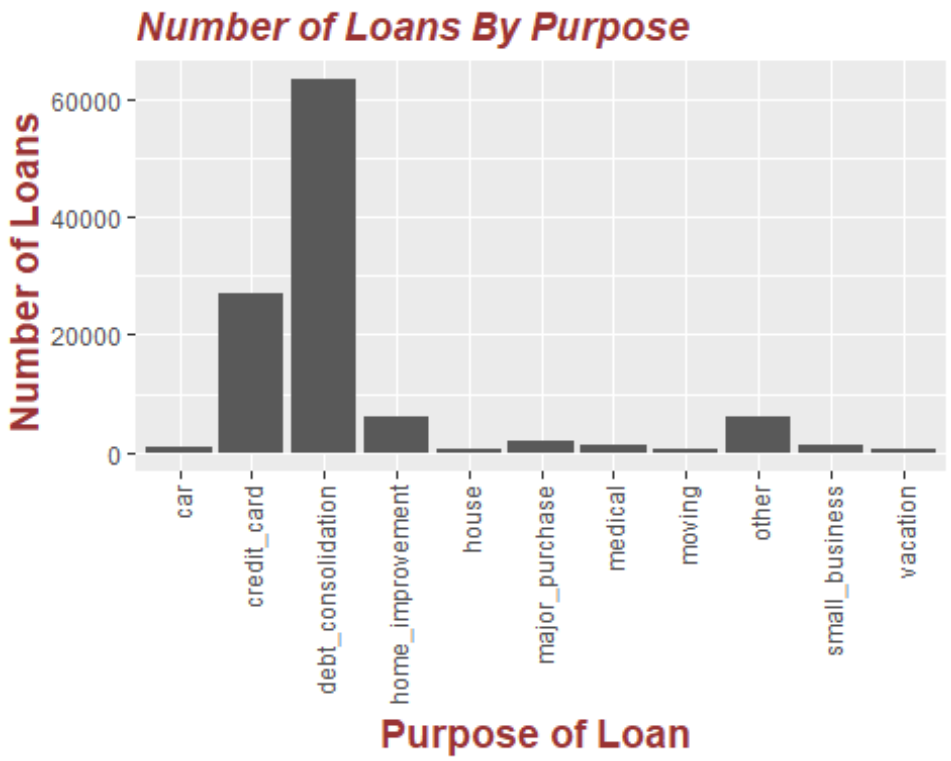
```r
lcdf$purpose <- as.character(lcdf$purpose )
lcdf$purpose  <- str_trim(lcdf$purpose )
lcdf$purpose  <- as.factor(lcdf$purpose )
```

```r
lcdf$purpose <- fct_collapse(lcdf$purpose, other =
c("wedding","renewable_energy", "other"),NULL = "H")
```

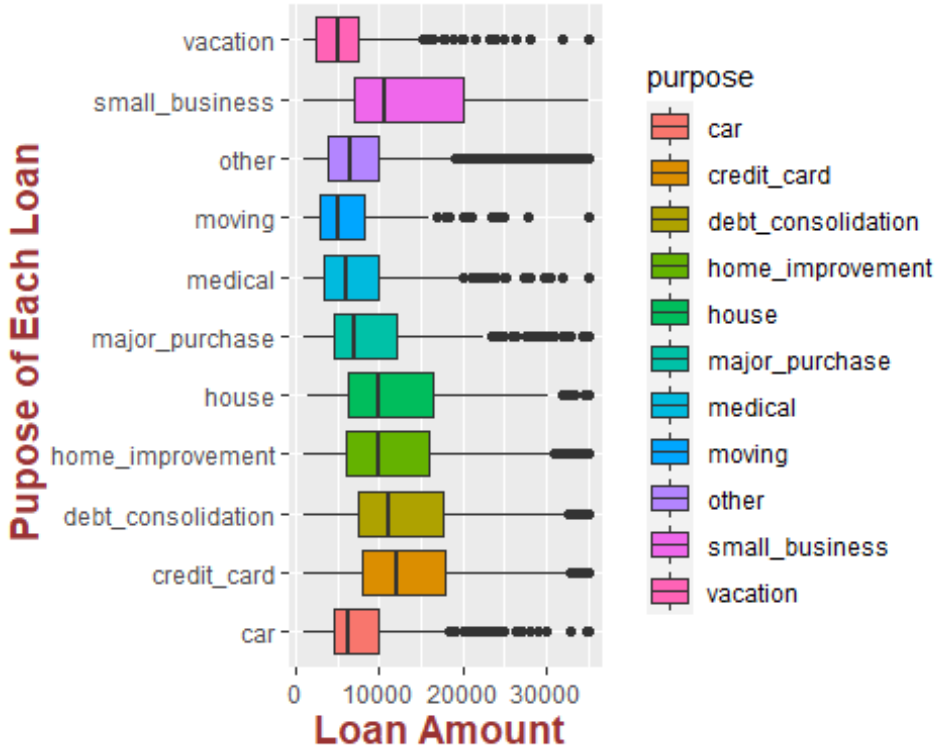
```r
lcdf %>% group_by(purpose) %>% tally()
```

```
## # A tibble: 11 × 2
##    purpose               n
##    <fct>             <int>
##  1 car                1083
##  2 credit_card       27091
##  3 debt_consolidation 63277
##  4 home_improvement   6190
##  5 house               432
##  6 major_purchase     2111
##  7 medical            1170
##  8 moving              755
##  9 other              5947
## 10 small_business     1117
## 11 vacation            753
```

```r
# Get the number of loans by loan purpose

ggplot(data = lcdf, aes(x = purpose)) + geom_bar() + ggtitle("Number of Loans
By Purpose") + xlab("Purpose of Loan ") + ylab("Number of Loans ")+
theme(plot.title = element_text(color="#993333", size=14,
face="bold.italic"), axis.title.x = element_text(color="#993333", size=14,
face="bold"), axis.title.y = element_text(color="#993333", size=14,
face="bold")) + theme(axis.text.x = element_text(angle = 90, vjust = 0.5,
hjust=1))
```




```
#Plot of loan amount by purpose


ggplot(lcdf, aes( x = loan_amnt, y=purpose)) +
geom_boxplot(aes(fill=purpose)) +
xlab("Loan Amount ") + ylab("Pupose of Each Loan ") + theme(plot.title =
element_text(color="#993333", size=14, face="bold.italic"), axis.title.x =
element_text(color="#993333", size=14, face="bold"), axis.title.y =
element_text(color="#993333", size=14, face="bold"))
```




```
# Percentages

lcdf %>% group_by(purpose) %>% summarise(nLoans=n(),
defaults=sum(loan_status=="Charged Off"), Default_per = defaults/nLoans)
```

```
## # A tibble: 11 × 4
##    purpose            nLoans defaults Default_per
##    <fct>               <int>    <int>       <dbl>
##  1 car                  1083      125       0.115
##  2 credit_card         27091     3169       0.117
##  3 debt_consolidation  63277     9253       0.146
##  4 home_improvement     6190      780       0.126
##  5 house                 432       70       0.162
##  6 major_purchase       2111      281       0.133
##  7 medical              1170      202       0.173
##  8 moving                755      137       0.181
##  9 other                5947      975       0.164
## 10 small_business       1117      259       0.232
## 11 vacation              753      125       0.166
```

```
#Does loan-grade vary by purpose? Which pupose the loan grade fall in?

table(lcdf$purpose, lcdf$grade)
```

```
##
##                        A     B     C     D     E     F     G
##   car                310   333   278   117    33    11     1
```

```
##   credit_card          8852 10795  5464  1686   267    24     3
##   debt_consolidation 12641 21988 18031  8128  2121   341    27
##   home_improvement    1600  2033  1584   699   226    43     5
##   house                 51    99   114    92    49    23     4
##   major_purchase       571   622   555   272    77    11     3
##   medical              105   292   391   239   104    34     5
##   moving                39   104   249   218   101    42     2
##   other                522  1279  1907  1487   585   151    16
##   small_business        92   166   301   324   164    57    13
##   vacation              66   149   267   190    63    16     2
```

```
#Bivariate analysis of employment length and purpose.

table(lcdf$purpose, lcdf$emp_length)
```

```
##
##                       n/a < 1 year 1 year 2 years 3 years 4 years 5 years
##   car                  57       90     92     136      87      67      74
##   credit_card        1731     2415   1857    2533    2203    1618    1714
##   debt_consolidation 3573     4975   4239    5501    5132    3705    3975
##   home_improvement    513      330    273     435     425     359     413
##   house                14       44     34      48      43      38      33
##   major_purchase      135      162    144     215     191     140     161
##   medical              88       79     89     115      92      61      75
##   moving               50      122     88      86      62      47      60
##   other               406      443    403     541     492     334     353
##   small_business       29       78     67     122     114      82      73
##   vacation             49       45     48      66      46      51      41
##
##                      6 years 7 years 8 years 9 years 10+ years
##   car                     55      57      37      36       295
##   credit_card           1346    1314    1247     968      8145
##   debt_consolidation    3103    3262    3257    2544     20011
##   home_improvement       289     343     310     219      2281
##   house                   23      19      20      17        99
##   major_purchase         123      93      88      68       591
##   medical                 55      58      66      44       348
##   moving                  36      23      31      17       133
##   other                  314     293     297     205      1866
##   small_business          69      76      61      38       308
##   vacation                36      36      41      31       263
```

```
#do those with home-improvement loans own or rent a home? Checking because
loan improvement should be with the people who own a home. Very rarely tenant
would take a loan for home improvement


table(lcdf$purpose,lcdf$home_ownership)
```

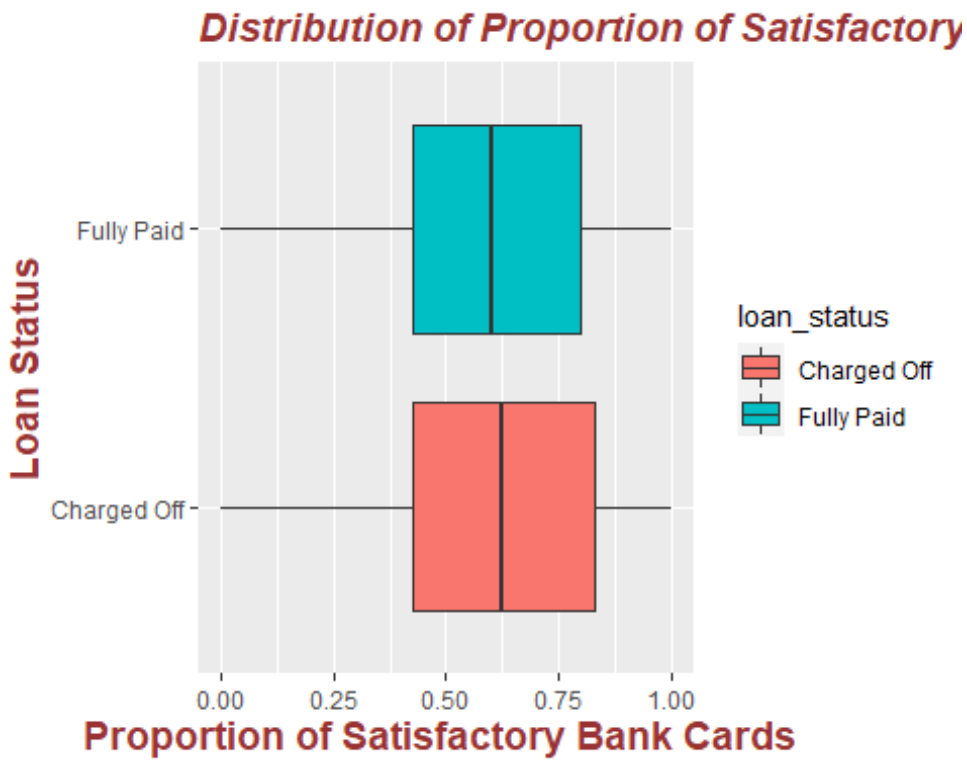
```
##
##                         ANY MORTGAGE  NONE OTHER   OWN  RENT
##   car                     0      459     0     0   124   500
##   credit_card             0    12239     7     1  2800 12044
##   debt_consolidation      0    29508     2     2  6052 27713
##   home_improvement        0     4682     1     1   968   538
##   house                   0      121     0     0    54   257
##   major_purchase          0      868     0     0   246   997
##   medical                 0      497     0     0   130   543
##   moving                  0      128     0     0    38   589
##   other                   1     2272     0     0   630  3044
##   small_business          0      499     0     0   121   497
##   vacation                0      263     0     0    81   409
```

More than half (58 %) of loans were taken for debt consolidation. This follows the the Pareto principle of 80:20 rule, as the top 3 puposes are more than 80% of loan purposes.mall business has higher default percentage. Loan borrowed for smaller business is defaulted the most in terms of percentage.It is also indicative of the fact that borrowers are coming to lending club for small business as they might have been already declined for a loan by bank.he loans show a very similar pattern irrespective of the purpose. Most number of loans in B for some cases , C followed by A grade loans. People with 10 + years of experience are the most common borrower of loan for credit card and debt consolidation. Home improvement loans are more common with 10+ years of experience.car loans are more common with people having 2 years of experience. Which might be reflective of the fact that once people are in job for 2 years they would want to keep a car for which they come to the lending club. We can see that home improvement loans were not the all with those owning the house. This might be because they were doing home improvement in rented house. Also more than 60% home were mortgaged. Also it can happen i might not directly own the house, owned by the partner while person borrowing the loan is doing home improvement over it.We can see the distribution of loan amount by various purposes.We can see that small business loans have significant distribution, fairly wide spred. Implying scales might be diffeerent for the small business.

Derived Attributes - proportion of satisfactory bankcard accounts, length of borrower's history, ratio of openAccounts to totalAccounts

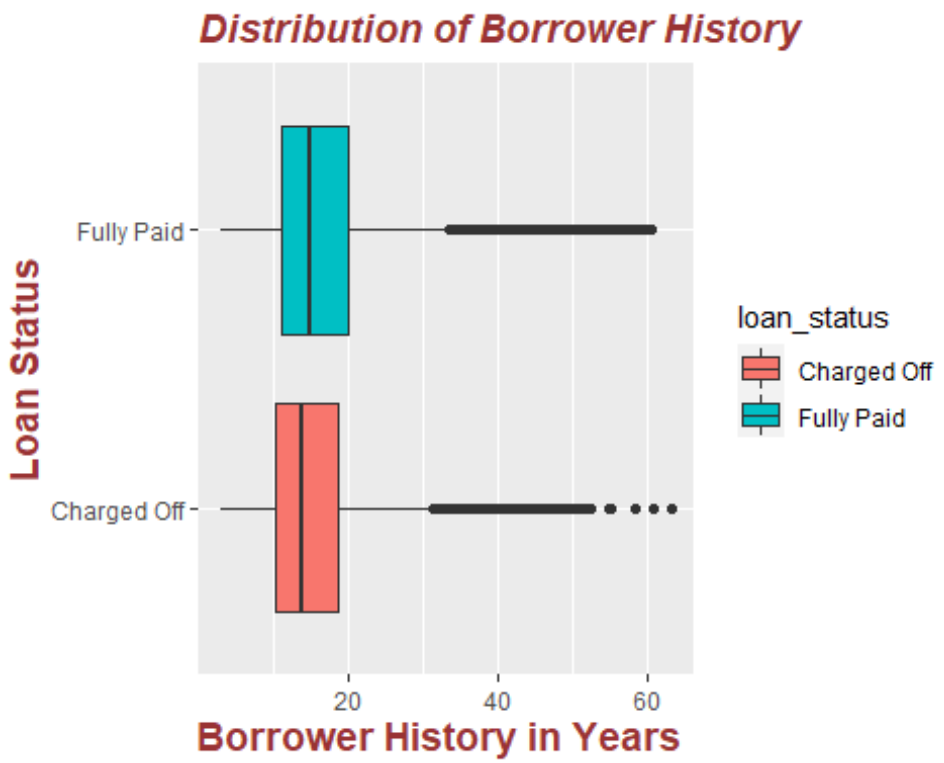
```
#  num_bc_tl - number  number of card
# and num_bc_sats satisfactory card


lcdf$propSatisBankcardAccts <- ifelse(lcdf$num_bc_tl>0,
lcdf$num_bc_sats/lcdf$num_bc_tl, 0)

# Let us look at the column created

summary(lcdf$propSatisBankcardAccts)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   0.000   0.429   0.600   0.614   0.800   1.000    4094
```

```
# Plot
ggplot(lcdf, aes( x = propSatisBankcardAccts, y=loan_status)) +
geom_boxplot(aes(fill=loan_status)) + ggtitle("Distribution of Proportion of
Satisfactory Bank Cards") +
xlab("Proportion of Satisfactory Bank Cards ") + ylab(" Loan Status ") +
theme(plot.title = element_text(color="#993333", size=14,
face="bold.italic"), axis.title.x = element_text(color="#993333", size=14,
face="bold"), axis.title.y = element_text(color="#993333", size=14,
face="bold"))
```



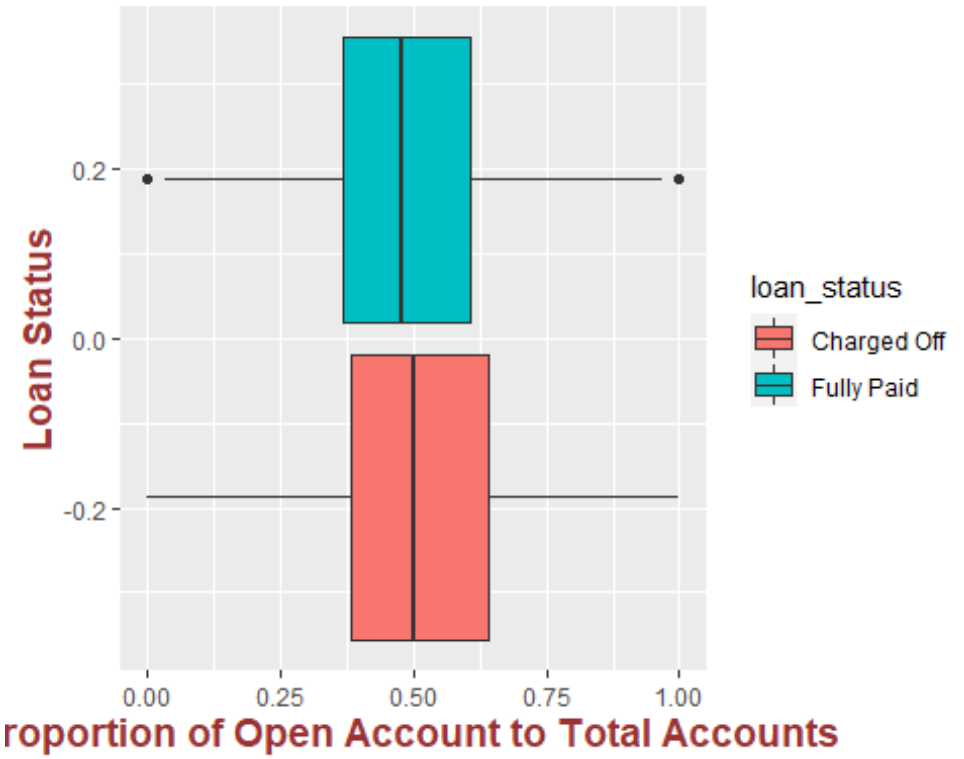

```
#Another one - lets calculate the length of borrower's history

#  i.e time between earliest_cr_line - open of current credit line. The month
the borrowers earliers
# issue_d

# Correcting the date format
lcdf$earliest_cr_line<-paste(lcdf$earliest_cr_line, "-01", sep = "")

lcdf$earliest_cr_line<-parse_date_time(lcdf$earliest_cr_line, "myd")

lcdf$earliest_cr_line %>% head()

## [1] "2011-08-01 UTC" "2006-12-01 UTC" "1995-02-01 UTC" "1995-11-01 UTC"
## [5] "2001-02-01 UTC" "2003-01-01 UTC"
```

```
lcdf$borrHistory <- as.duration(lcdf$earliest_cr_line %--% lcdf$issue_d  ) /
dyears(1)


ggplot(lcdf, aes( x = borrHistory, y=loan_status)) +
geom_boxplot(aes(fill=loan_status)) +
xlab("Borrower History in Years ") + ylab("Loan
Status")+ggtitle("Distribution of Borrower History") + theme(plot.title =
element_text(color="#993333", size=14, face="bold.italic"), axis.title.x =
element_text(color="#993333", size=14, face="bold"), axis.title.y =
element_text(color="#993333", size=14, face="bold"))
```

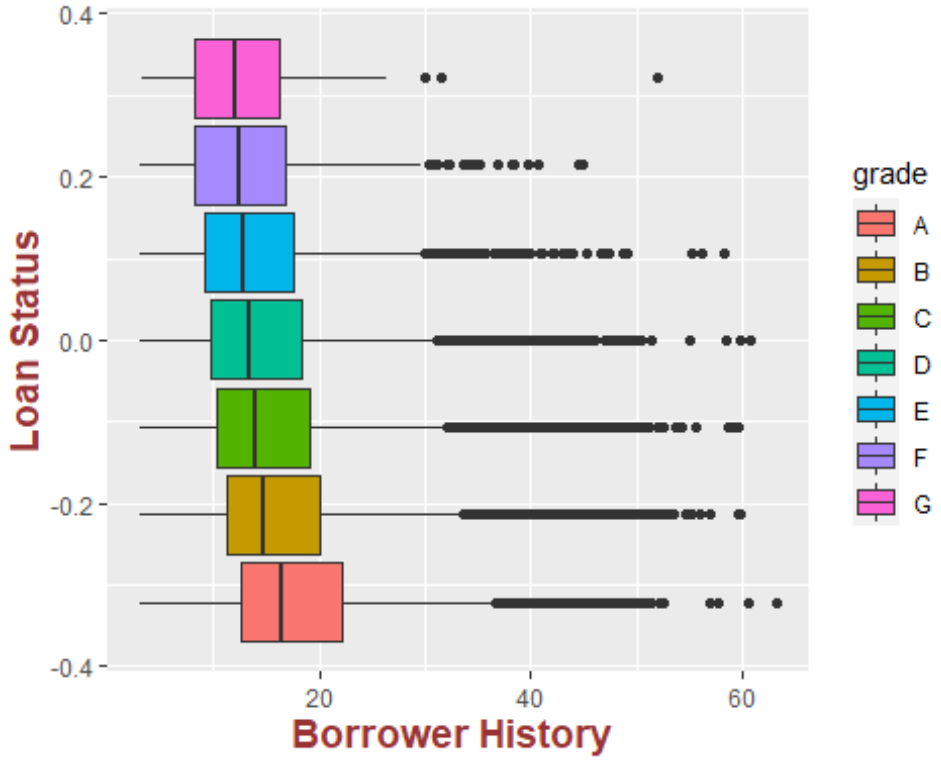



```
#Another new attribute: ratio of openAccounts to totalAccounts


lcdf$openAccRatio <- ifelse(lcdf$total_acc>0, lcdf$open_acc/lcdf$total_acc,
0)


summary(lcdf$openAccRatio)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.3704  0.4815  0.5018  0.6154  1.0000

 #   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 # 0.0000  0.3704  0.4815  0.5017  0.6154  1.0000
```

```
ggplot(lcdf, aes( x = openAccRatio)) + geom_boxplot(aes(fill=loan_status)) +
xlab("Proportion of Open Account to Total Accounts ") + ylab(" Loan Status ")
+ theme(plot.title = element_text(color="#993333", size=14,
face="bold.italic"), axis.title.x = element_text(color="#993333", size=14,
face="bold"), axis.title.y = element_text(color="#993333", size=14,
face="bold"))
```




```
#does LC-assigned loan grade vary by borrHistory?

lcdf %>% group_by(grade) %>% summarise(avgBorrHist=mean(borrHistory))

## # A tibble: 7 × 2
##   grade avgBorrHist
##   <chr>       <dbl>
## 1 A            18.1
## 2 B            16.4
## 3 C            15.4
## 4 D            14.8
## 5 E            14.2
## 6 F            13.5
## 7 G            13.3

ggplot(lcdf, aes( x = borrHistory)) + geom_boxplot(aes(fill=grade)) +
xlab("Borrower History ") + ylab(" Loan Status ") + theme(plot.title =
element_text(color="#993333", size=14, face="bold.italic"), axis.title.x =
element_text(color="#993333", size=14, face="bold"), axis.title.y =
element_text(color="#993333", size=14, face="bold"))
```




```
lcdf %>% group_by(grade) %>% summarise(avgBorrHist=mean(borrHistory),
minBorrHist=min(borrHistory), maxBorrHist = max(borrHistory),
medianBorrHist=median(borrHistory))
```

```
## # A tibble: 7 × 5
##   grade avgBorrHist minBorrHist maxBorrHist medianBorrHist
##   <chr>       <dbl>       <dbl>       <dbl>          <dbl>
## 1 A            18.1        3.08        63.2           16.4
## 2 B            16.4        3.08        59.8           14.8
## 3 C            15.4        3.08        59.6           14.0
## 4 D            14.8        3.08        60.7           13.4
## 5 E            14.2        3.16        58.2           12.8
## 6 F            13.5        3.08        44.8           12.4
## 7 G            13.3        3.25        52             12
```

Yes, assigned loan grade varies, significantly with the borrower history. We can also check the min, max median in the below box plot

Converting character variables

```
#glimpse(lcdf)

#  there are a few character type variables - grade, sub_grade,
verification_status,....
#   We can  convert all of these to factor

lcdf <- lcdf %>% mutate_if(is.character, as.factor)
```

```
#Checking the datatype after conversion

#glimpse(lcdf)
```

Leakage variables

Concept of leakage - In statistics and machine learning, leakage (also known as data leakage or target leakage) is the use of information in the model training process which would not be expected to be available at prediction time, causing the predictive scores (metrics) to overestimate the model's utility when run in a production environment.Reference - https://en.wikipedia.org/wiki/Leakage_(machine_learning)#:~:text=In%20statistics%20and%20machine%20learning,when%20run%20in%20a%20production

```
#Identified the variables you want to remove

varsToRemove = c('funded_amnt_inv', 'term', 'emp_title', 'pymnt_plan',
'earliest_cr_line', 'title', 'zip_code', 'addr_state', 'out_prncp',
'out_prncp_inv', 'total_pymnt_inv', 'total_rec_prncp', 'total_rec_int',
'total_rec_late_fee', 'recoveries', 'collection_recovery_fee',
'last_credit_pull_d', 'policy_code', 'disbursement_method',
'debt_settlement_flag',  'settlement_term', 'application_type')

lcdf <- lcdf %>% select(-all_of(varsToRemove))

#Drop all the variables with names starting with "hardship" -- as they can
cause leakage, unknown at the time when the loan was given.

#First checking before dropping

lcdf %>% select(starts_with("hardship"))
```

```
## # A tibble: 109,926 × 12
##    hardship_flag hards…¹ hards…² hards…³ hards…⁴ hards…⁵ hards…⁶ hards…⁷
hards…⁸
##    <fct>         <lgl>   <lgl>   <lgl>   <lgl>   <lgl>   <lgl>   <lgl>
<lgl>
##  1 N             NA      NA      NA      NA      NA      NA      NA
NA
##  2 N             NA      NA      NA      NA      NA      NA      NA
NA
##  3 N             NA      NA      NA      NA      NA      NA      NA
NA
##  4 N             NA      NA      NA      NA      NA      NA      NA
NA
##  5 N             NA      NA      NA      NA      NA      NA      NA
NA
##  6 N             NA      NA      NA      NA      NA      NA      NA
NA
##  7 N             NA      NA      NA      NA      NA      NA      NA
```

```
NA
##  8 N             NA      NA      NA      NA      NA      NA      NA
NA
##  9 N             NA      NA      NA      NA      NA      NA      NA
NA
## 10 N             NA      NA      NA      NA      NA      NA      NA
NA
## # … with 109,916 more rows, 3 more variables: hardship_loan_status <lgl>,
## #   hardship_payoff_balance_amount <lgl>, hardship_last_payment_amount
<lgl>,
## #   and abbreviated variable names ¹hardship_type, ²hardship_reason,
## #   ³hardship_status, ⁴hardship_amount, ⁵hardship_start_date,
## #   ⁶hardship_end_date, ⁷hardship_length, ⁸hardship_dpd

# Dropping

lcdf <- lcdf %>% select(-starts_with("hardship"))

#similarly, all variable starting with "settlement", these are happening
after disbursement

lcdf %>% select(starts_with('settlement'))

## # A tibble: 109,926 × 4
##    settlement_status settlement_date settlement_amount
settlement_percentage
##    <lgl>             <lgl>           <lgl>             <lgl>
##  1 NA                NA              NA                NA
##  2 NA                NA              NA                NA
##  3 NA                NA              NA                NA
##  4 NA                NA              NA                NA
##  5 NA                NA              NA                NA
##  6 NA                NA              NA                NA
##  7 NA                NA              NA                NA
##  8 NA                NA              NA                NA
##  9 NA                NA              NA                NA
## 10 NA                NA              NA                NA
## # … with 109,916 more rows

# 4 columns

#Dropping them

lcdf <- lcdf %>% select(-starts_with("settlement"))

# Additional Leakage variables - based on our understanding


varsToRemove2 <- c("last_pymnt_d", "last_pymnt_amnt",
"issue_d",'next_pymnt_d', 'deferral_term', 'payment_plan_start_date',
```

```
'debt_settlement_flag_date'  )


# last_pymnt_d, last_pymnt_amnt, next_pymnt_d, deferral_term,
payment_plan_start_date, debt_settlement_flag_date

lcdf <- lcdf %>% select(-all_of(varsToRemove2))
```

Understanding the leakage is very important in the concept of Data Mining where we will be going ahead to predict models based on the training data. The models will be well trained if we use the leakage variable, however when we get unseen set of data the prediction will be poor as they wont be having values of these variables

Missing Values

Potential reasons for missing values in different variables? Are some of the missing values actually 'zeros' which are not recorded in the data? Is missing-ness informative in some way? Are there, for example, more/less defaults for cases where values on the attribute are missing ?

```
# Dropping columns with all n/a

lcdf %>% select_if(function(x){  all(is.na(x)) } ) # Checking what are those
columns
```

```
## # A tibble: 109,926 × 19
##    id    member_id url   desc  annual_…¹ dti_j…² verif…³ revol…⁴ sec_a…⁵
sec_a…⁶
##    <lgl> <lgl>     <lgl> <lgl> <lgl>     <lgl>   <lgl>   <lgl>   <lgl>
<lgl>
##  1 NA    NA        NA    NA    NA        NA      NA      NA      NA
NA
##  2 NA    NA        NA    NA    NA        NA      NA      NA      NA
NA
##  3 NA    NA        NA    NA    NA        NA      NA      NA      NA
NA
##  4 NA    NA        NA    NA    NA        NA      NA      NA      NA
NA
##  5 NA    NA        NA    NA    NA        NA      NA      NA      NA
NA
##  6 NA    NA        NA    NA    NA        NA      NA      NA      NA
NA
##  7 NA    NA        NA    NA    NA        NA      NA      NA      NA
NA
##  8 NA    NA        NA    NA    NA        NA      NA      NA      NA
NA
##  9 NA    NA        NA    NA    NA        NA      NA      NA      NA
NA
## 10 NA    NA        NA    NA    NA        NA      NA      NA      NA
NA
```

```
## # … with 109,916 more rows, 9 more variables: sec_app_mort_acc <lgl>,
## #   sec_app_open_acc <lgl>, sec_app_revol_util <lgl>,
## #   sec_app_open_act_il <lgl>, sec_app_num_rev_accts <lgl>,
## #   sec_app_chargeoff_within_12_mths <lgl>,
## #   sec_app_collections_12_mths_ex_med <lgl>,
## #   sec_app_mths_since_last_major_derog <lgl>,
## #   orig_projected_additional_accrued_interest <lgl>, and abbreviated …
```

```
lcdf <- lcdf %>% select_if(function(x){ ! all(is.na(x)) } ) # Dropping

# Finding names of columns which has atleast 1 missing values


names(lcdf)[colSums(is.na(lcdf)) > 0]
```

```
##  [1] "dti"                              "mths_since_last_delinq"
##  [3] "mths_since_last_record"           "revol_util"
##  [5] "mths_since_last_major_derog"      "tot_coll_amt"
##  [7] "tot_cur_bal"                      "open_acc_6m"
##  [9] "open_act_il"                      "open_il_12m"
## [11] "open_il_24m"                      "mths_since_rcnt_il"
## [13] "total_bal_il"                     "il_util"
## [15] "open_rv_12m"                      "open_rv_24m"
## [17] "max_bal_bc"                       "all_util"
## [19] "total_rev_hi_lim"                 "inq_fi"
## [21] "total_cu_tl"                      "inq_last_12m"
## [23] "acc_open_past_24mths"             "avg_cur_bal"
## [25] "bc_open_to_buy"                   "bc_util"
## [27] "mo_sin_old_il_acct"               "mo_sin_old_rev_tl_op"
## [29] "mo_sin_rcnt_rev_tl_op"            "mo_sin_rcnt_tl"
## [31] "mort_acc"                         "mths_since_recent_bc"
## [33] "mths_since_recent_bc_dlq"         "mths_since_recent_inq"
## [35] "mths_since_recent_revol_delinq" "num_accts_ever_120_pd"
## [37] "num_actv_bc_tl"                   "num_actv_rev_tl"
## [39] "num_bc_sats"                      "num_bc_tl"
## [41] "num_il_tl"                        "num_op_rev_tl"
## [43] "num_rev_accts"                    "num_rev_tl_bal_gt_0"
## [45] "num_sats"                         "num_tl_120dpd_2m"
## [47] "num_tl_30dpd"                     "num_tl_90g_dpd_24m"
## [49] "num_tl_op_past_12m"               "pct_tl_nvr_dlq"
## [51] "percent_bc_gt_75"                 "tot_hi_cred_lim"
## [53] "total_bal_ex_mort"                "total_bc_limit"
## [55] "total_il_high_credit_limit"       "propSatisBankcardAccts"
```

```
# Finding proportion

options(scipen=999) # To not use scientific notation

colMeans(is.na(lcdf))[colMeans(is.na(lcdf))>0]
```

```
##                           dti         mths_since_last_delinq
##                 0.000009097029                 0.505649254953
##         mths_since_last_record                     revol_util
##                 0.835034477740                 0.000418463330
##    mths_since_last_major_derog                   tot_coll_amt
##                 0.733393373724                 0.037243236359
##                    tot_cur_bal                    open_acc_6m
##                 0.037243236359                 0.974482833906
##                    open_act_il                    open_il_12m
##                 0.974482833906                 0.974482833906
##                    open_il_24m              mths_since_rcnt_il
##                 0.974482833906                 0.975274275422
##                   total_bal_il                         il_util
##                 0.974482833906                 0.978094354384
##                    open_rv_12m                    open_rv_24m
##                 0.974482833906                 0.974482833906
##                     max_bal_bc                       all_util
##                 0.974482833906                 0.974482833906
##               total_rev_hi_lim                         inq_fi
##                 0.037243236359                 0.974482833906
##                    total_cu_tl                   inq_last_12m
##                 0.974482833906                 0.974482833906
##           acc_open_past_24mths                    avg_cur_bal
##                 0.009879373397                 0.037270527446
##                 bc_open_to_buy                         bc_util
##                 0.019513127013                 0.020140822008
##             mo_sin_old_il_acct           mo_sin_old_rev_tl_op
##                 0.073558575769                 0.037243236359
##          mo_sin_rcnt_rev_tl_op                mo_sin_rcnt_tl
##                 0.037243236359                 0.037243236359
##                       mort_acc           mths_since_recent_bc
##                 0.009879373397                 0.018812655787
##       mths_since_recent_bc_dlq          mths_since_recent_inq
##                 0.752924694795                 0.116905918527
## mths_since_recent_revol_delinq          num_accts_ever_120_pd
##                 0.652256972873                 0.037243236359
##                 num_actv_bc_tl                num_actv_rev_tl
##                 0.037243236359                 0.037243236359
##                    num_bc_sats                     num_bc_tl
##                 0.021532667431                 0.037243236359
##                      num_il_tl                    num_op_rev_tl
##                 0.037243236359                 0.037243236359
##                  num_rev_accts            num_rev_tl_bal_gt_0
##                 0.037243236359                 0.037243236359
##                       num_sats               num_tl_120dpd_2m
##                 0.021532667431                 0.072103051143
##                   num_tl_30dpd              num_tl_90g_dpd_24m
##                 0.037243236359                 0.037243236359
##             num_tl_op_past_12m                 pct_tl_nvr_dlq
##                 0.037243236359                 0.037452468024
```

```
##               percent_bc_gt_75                 tot_hi_cred_lim
##                  0.020195404181                  0.037243236359
##              total_bal_ex_mort                  total_bc_limit
##                  0.009879373397                  0.009879373397
##     total_il_high_credit_limit          propSatisBankcardAccts
##                  0.037243236359                  0.037243236359
```

```
# Finding the columns which have more than 60% missing values

names(lcdf)[colMeans(is.na(lcdf))>0.6]
```

```
##  [1] "mths_since_last_record"         "mths_since_last_major_derog"
##  [3] "open_acc_6m"                    "open_act_il"
##  [5] "open_il_12m"                    "open_il_24m"
##  [7] "mths_since_rcnt_il"             "total_bal_il"
##  [9] "il_util"                        "open_rv_12m"
## [11] "open_rv_24m"                    "max_bal_bc"
## [13] "all_util"                       "inq_fi"
## [15] "total_cu_tl"                    "inq_last_12m"
## [17] "mths_since_recent_bc_dlq"       "mths_since_recent_revol_delinq"
```

```
nm<-names(lcdf)[colMeans(is.na(lcdf))>0.6]
lcdf <- lcdf %>% select(-all_of(nm))

#Impute missing values for remaining variables which have missing values
# - first get the columns with missing values

colMeans(is.na(lcdf))[colMeans(is.na(lcdf))>0]
```

```
##                          dti      mths_since_last_delinq
##              0.000009097029                0.505649254953
##                  revol_util                  tot_coll_amt
##              0.000418463330                0.037243236359
##                 tot_cur_bal             total_rev_hi_lim
##              0.037243236359                0.037243236359
##        acc_open_past_24mths                   avg_cur_bal
##              0.009879373397                0.037270527446
##              bc_open_to_buy                         bc_util
##              0.019513127013                0.020140822008
##          mo_sin_old_il_acct        mo_sin_old_rev_tl_op
##              0.073558575769                0.037243236359
##       mo_sin_rcnt_rev_tl_op              mo_sin_rcnt_tl
##              0.037243236359                0.037243236359
##                    mort_acc        mths_since_recent_bc
##              0.009879373397                0.018812655787
##       mths_since_recent_inq       num_accts_ever_120_pd
##              0.116905918527                0.037243236359
##              num_actv_bc_tl             num_actv_rev_tl
##              0.037243236359                0.037243236359
##                 num_bc_sats                      num_bc_tl
```

```
##             0.021532667431             0.037243236359
##                  num_il_tl              num_op_rev_tl
##             0.037243236359             0.037243236359
##              num_rev_accts        num_rev_tl_bal_gt_0
##             0.037243236359             0.037243236359
##                   num_sats           num_tl_120dpd_2m
##             0.021532667431             0.072103051143
##               num_tl_30dpd         num_tl_90g_dpd_24m
##             0.037243236359             0.037243236359
##         num_tl_op_past_12m             pct_tl_nvr_dlq
##             0.037243236359             0.037452468024
##           percent_bc_gt_75            tot_hi_cred_lim
##             0.020195404181             0.037243236359
##          total_bal_ex_mort             total_bc_limit
##             0.009879373397             0.009879373397
## total_il_high_credit_limit     propSatisBankcardAccts
##             0.037243236359             0.037243236359

nm<- names(lcdf)[colSums(is.na(lcdf))>0]

summary(lcdf[, nm])

##       dti         mths_since_last_delinq   revol_util      tot_coll_amt
##  Min.   :  0.00   Min.   :  0.00         Min.   :  0.00   Min.   :     0.0
##  1st Qu.: 11.58   1st Qu.: 16.00         1st Qu.: 36.70   1st Qu.:     0.0
##  Median : 17.28   Median : 31.00         Median : 54.50   Median :     0.0
##  Mean   : 17.82   Mean   : 34.19         Mean   : 54.07   Mean   :   224.1
##  3rd Qu.: 23.63   3rd Qu.: 50.00         3rd Qu.: 72.20   3rd Qu.:     0.0
##  Max.   :137.40   Max.   :188.00         Max.   :117.90   Max.   :143558.0
##  NA's   :1        NA's   :55584          NA's   :46       NA's   :4094
##   tot_cur_bal      total_rev_hi_lim  acc_open_past_24mths  avg_cur_bal
##  Min.   :      0   Min.   :      0   Min.   : 0.000       Min.   :     0
##  1st Qu.:  25541   1st Qu.:  12800   1st Qu.: 2.000       1st Qu.:  2771
##  Median :  64569   Median :  22000   Median : 4.000       Median :  6273
##  Mean   : 128436   Mean   :  30504   Mean   : 4.356       Mean   : 12441
##  3rd Qu.: 190600   3rd Qu.:  37600   3rd Qu.: 6.000       3rd Qu.: 17107
##  Max.   :3370799   Max.   :1046900   Max.   :40.000       Max.   :312125
##  NA's   :4094      NA's   :4094      NA's   :1086         NA's   :4097
##  bc_open_to_buy      bc_util       mo_sin_old_il_acct mo_sin_old_rev_tl_op
##  Min.   :     0   Min.   :  0.00   Min.   :  0.0      Min.   :  5.0
##  1st Qu.:  1173   1st Qu.: 42.70   1st Qu.: 95.0      1st Qu.:115.0
##  Median :  3878   Median : 66.50   Median :128.0      Median :164.0
##  Mean   :  9010   Mean   : 62.68   Mean   :124.8      Mean   :181.7
##  3rd Qu.: 10600   3rd Qu.: 86.30   3rd Qu.:152.0      3rd Qu.:230.0
##  Max.   :278899   Max.   :255.20   Max.   :519.0      Max.   :757.0
##  NA's   :2145     NA's   :2214     NA's   :8086       NA's   :4094
##  mo_sin_rcnt_rev_tl_op mo_sin_rcnt_tl       mort_acc
mths_since_recent_bc
##  Min.   :  0.00        Min.   :  0.000   Min.   : 0.000   Min.   :  0.00
##  1st Qu.:  4.00        1st Qu.:  3.000   1st Qu.: 0.000   1st Qu.:  6.00
```

```
##  Median :  8.00        Median :  6.000   Median : 1.000   Median : 14.00
##  Mean   : 13.27        Mean   :  8.263   Mean   : 1.627   Mean   : 24.53
##  3rd Qu.: 16.00        3rd Qu.: 10.000   3rd Qu.: 3.000   3rd Qu.: 30.00
##  Max.   :372.00        Max.   :197.000   Max.   :34.000   Max.   :555.00
##  NA's   :4094          NA's   :4094      NA's   :1086     NA's   :2068
##  mths_since_recent_inq num_accts_ever_120_pd num_actv_bc_tl
num_actv_rev_tl
##  Min.   : 0.000        Min.   : 0.000        Min.   : 0.000   Min.   :
0.000
##  1st Qu.: 2.000        1st Qu.: 0.000        1st Qu.: 2.000   1st Qu.:
3.000
##  Median : 5.000        Median : 0.000        Median : 3.000   Median :
5.000
##  Mean   : 6.928        Mean   : 0.498        Mean   : 3.652   Mean   :
5.676
##  3rd Qu.:10.000        3rd Qu.: 0.000        3rd Qu.: 5.000   3rd Qu.:
7.000
##  Max.   :25.000        Max.   :35.000        Max.   :30.000   Max.
:38.000
##  NA's   :12851         NA's   :4094          NA's   :4094     NA's   :4094
##   num_bc_sats       num_bc_tl        num_il_tl      num_op_rev_tl
##  Min.   : 0.000   Min.   : 0.000   Min.   : 0.000   Min.   : 0.000
##  1st Qu.: 3.000   1st Qu.: 5.000   1st Qu.: 3.000   1st Qu.: 5.000
##  Median : 4.000   Median : 7.000   Median : 6.000   Median : 7.000
##  Mean   : 4.652   Mean   : 8.324   Mean   : 8.076   Mean   : 8.183
##  3rd Qu.: 6.000   3rd Qu.:11.000   3rd Qu.:11.000   3rd Qu.:10.000
##  Max.   :46.000   Max.   :60.000   Max.   :97.000   Max.   :58.000
##  NA's   :2367     NA's   :4094     NA's   :4094     NA's   :4094
##  num_rev_accts   num_rev_tl_bal_gt_0    num_sats     num_tl_120dpd_2m
##  Min.   : 1.00   Min.   : 0.000      Min.   : 0.00   Min.   :0.000
##  1st Qu.: 9.00   1st Qu.: 3.000      1st Qu.: 8.00   1st Qu.:0.000
##  Median :13.00   Median : 5.000      Median :10.00   Median :0.000
##  Mean   :14.78   Mean   : 5.642      Mean   :11.38   Mean   :0.001
##  3rd Qu.:19.00   3rd Qu.: 7.000      3rd Qu.:14.00   3rd Qu.:0.000
##  Max.   :92.00   Max.   :38.000      Max.   :62.00   Max.   :2.000
##  NA's   :4094    NA's   :4094        NA's   :2367    NA's   :7926
##   num_tl_30dpd   num_tl_90g_dpd_24m num_tl_op_past_12m pct_tl_nvr_dlq
##  Min.   :0.000   Min.   : 0.000     Min.   : 0.000     Min.   : 14.80
##  1st Qu.:0.000   1st Qu.: 0.000     1st Qu.: 1.000     1st Qu.: 91.00
##  Median :0.000   Median : 0.000     Median : 2.000     Median : 97.80
##  Mean   :0.003   Mean   : 0.093     Mean   : 2.036     Mean   : 94.07
##  3rd Qu.:0.000   3rd Qu.: 0.000     3rd Qu.: 3.000     3rd Qu.:100.00
##  Max.   :4.000   Max.   :20.000     Max.   :25.000     Max.   :100.00
##  NA's   :4094    NA's   :4094       NA's   :4094       NA's   :4117
##  percent_bc_gt_75 tot_hi_cred_lim   total_bal_ex_mort total_bc_limit
##  Min.   :  0.00   Min.   :      0   Min.   :      0   Min.   :     0
##  1st Qu.: 16.70   1st Qu.:  43057   1st Qu.:  18781   1st Qu.:  7000
##  Median : 50.00   Median :  93220   Median :  33795   Median : 13600
##  Mean   : 48.31   Mean   : 158917   Mean   :  45639   Mean   : 20163
##  3rd Qu.: 75.00   3rd Qu.: 228957   3rd Qu.:  57068   3rd Qu.: 26000
```

```
##  Max.   :100.00   Max.   :8700253   Max.   :1043860   Max.   :456200
##  NA's   :2220     NA's   :4094      NA's   :1086      NA's   :1086
##  total_il_high_credit_limit propSatisBankcardAccts
##  Min.   :     0             Min.   :0.000
##  1st Qu.: 12000             1st Qu.:0.429
##  Median : 28170             Median :0.600
##  Mean   : 38108             Mean   :0.614
##  3rd Qu.: 51194             3rd Qu.:0.800
##  Max.   :975560             Max.   :1.000
##  NA's   :4094               NA's   :4094
```

```
# Replacing values - adding median values

lcdf<- lcdf %>%
replace_na(list(mths_since_last_delinq=median(lcdf$mths_since_last_delinq,
na.rm=TRUE), bc_open_to_buy=median(lcdf$bc_open_to_buy, na.rm=TRUE),
mo_sin_old_il_acct=median(lcdf$mo_sin_old_il_acct,na.rm=TRUE),
mths_since_recent_bc=median(lcdf$mths_since_recent_bc, na.rm=TRUE),
mths_since_recent_inq=5, num_tl_120dpd_2m = median(lcdf$num_tl_120dpd_2m,
na.rm=TRUE),percent_bc_gt_75 = median(lcdf$percent_bc_gt_75, na.rm=TRUE),
bc_util=median(lcdf$bc_util, na.rm=TRUE) ))



lcdf<- lcdf %>% mutate_if(is.numeric,  ~ifelse(is.na(.x), median(.x, na.rm =
TRUE), .x))


dim(lcdf)
```

```
## [1] 109926     69
```

Yes, some columns have same percentage of missing values. This could be because they are dependent columns. Information source of a column is also the source of other columns could be the reason. These missing values can be because of the following - 1. Missing Completely at Random 2. Missing at Random 3. Missing Not At Random. We could use various techniques taught in class to impute these missing values. 1. Imputing values 2. Leaving those rows. However approach for each column can be different. We could use various techniques taught in class to impute these missing values. 1. Imputing values 2. Leaving those rows. However approach for each column can be different. If they do not relate well to larger values, than we should not assume that missings are for values higher than the max.We will remove columns with more than 60% missing values, this is taken as a trial and test way - However when it comes to removing columns with NA approach could be different in each case. This could also mean loss of very important variable. We can tune our model based on the results

Univariate Analysis - AUC

which variables are individually predictive of the outcome ? Considering a single variable model to predict loan_status, what could be a measure of performance? AUC? For a univariate model with a variable, say, x1, what should we consider as the model 'score' for predicting loan_status? Can we take the values of x1 as the score for a model y_hat=f(x1) ? Using this approximate approach, we can then compute the AUC for each variable. AUC of a classifier is equivalent to the probability that the classifier will rank a randomly chosen positive instance higher than a randomly chosen negative instance.Reference - https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7695228/

```
#We will use the function auc(response, prediction) which returns the AUC
value for the specified predictor variable, and considering the response
variable as the dependent.


aucAll<- sapply(lcdf %>% mutate_if(is.factor, as.numeric) %>%
select_if(is.numeric), auc, response=lcdf$loan_status)


library(broom)

tidy(aucAll[aucAll > 0.5])
```

```
## # A tibble: 46 × 2
##    names              x
##    <chr>          <dbl>
##  1 loan_amnt      0.515
##  2 funded_amnt    0.515
##  3 int_rate       0.656
##  4 installment    0.501
##  5 grade          0.652
##  6 sub_grade      0.663
##  7 emp_length     0.529
##  8 home_ownership 0.552
##  9 annual_inc     0.575
## 10 loan_status    1
## # … with 36 more rows
```

```
tidy(aucAll) %>% arrange(desc(aucAll))
```

```
## # A tibble: 69 × 2
##    names                    x
##    <chr>                <dbl>
##  1 loan_status          1
##  2 actualReturn         0.987
##  3 annRet               0.968
##  4 total_pymnt          0.752
##  5 actualTerm           0.679
##  6 sub_grade            0.663
```

```
##  7 int_rate             0.656
##  8 grade                0.652
##  9 acc_open_past_24mths 0.579
## 10 annual_inc           0.575
## # … with 59 more rows
```

Example, actualReturn, actualTerm are in the data - we have kept these because they will be useful for evaluating performance of models. High AUC effect on model It tells how much the model is capable of distinguishing between classes. Higher the AUC, the better the model is at predicting 0 classes as 0 and 1 classes as 1. By analogy, the Higher the AUC, the better the model is at distinguishing between patients with the disease and no disease. Will need to make sure these are not included in building the models