Assignment 2

Abhinav Ram Bhatta, Prajwal Prashanth, Anviksha Gupta

2022-10-31


```
#Import Data
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.2 ──
## ✔ ggplot2 3.3.6      ✔ purrr   0.3.4
## ✔ tibble  3.1.8      ✔ dplyr   1.0.10
## ✔ tidyr   1.2.0      ✔ stringr 1.4.1
## ✔ readr   2.1.2      ✔ forcats 0.5.2
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
```

```
library(lubridate)
```

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

```
library(data.table)
```

```
##
## Attaching package: 'data.table'
##
## The following objects are masked from 'package:lubridate':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year
##
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
##
## The following object is masked from 'package:purrr':
##
##     transpose
```

```
library(broom)
options(dplyr.summarise.inform = FALSE)

#Importing data and saving it in a variable name.
LC_Data <- read_csv('/Users/abhinavram/Downloads/lcDataSampleFall22.csv')
```

```
## Warning: One or more parsing issues, see `problems()` for details
```

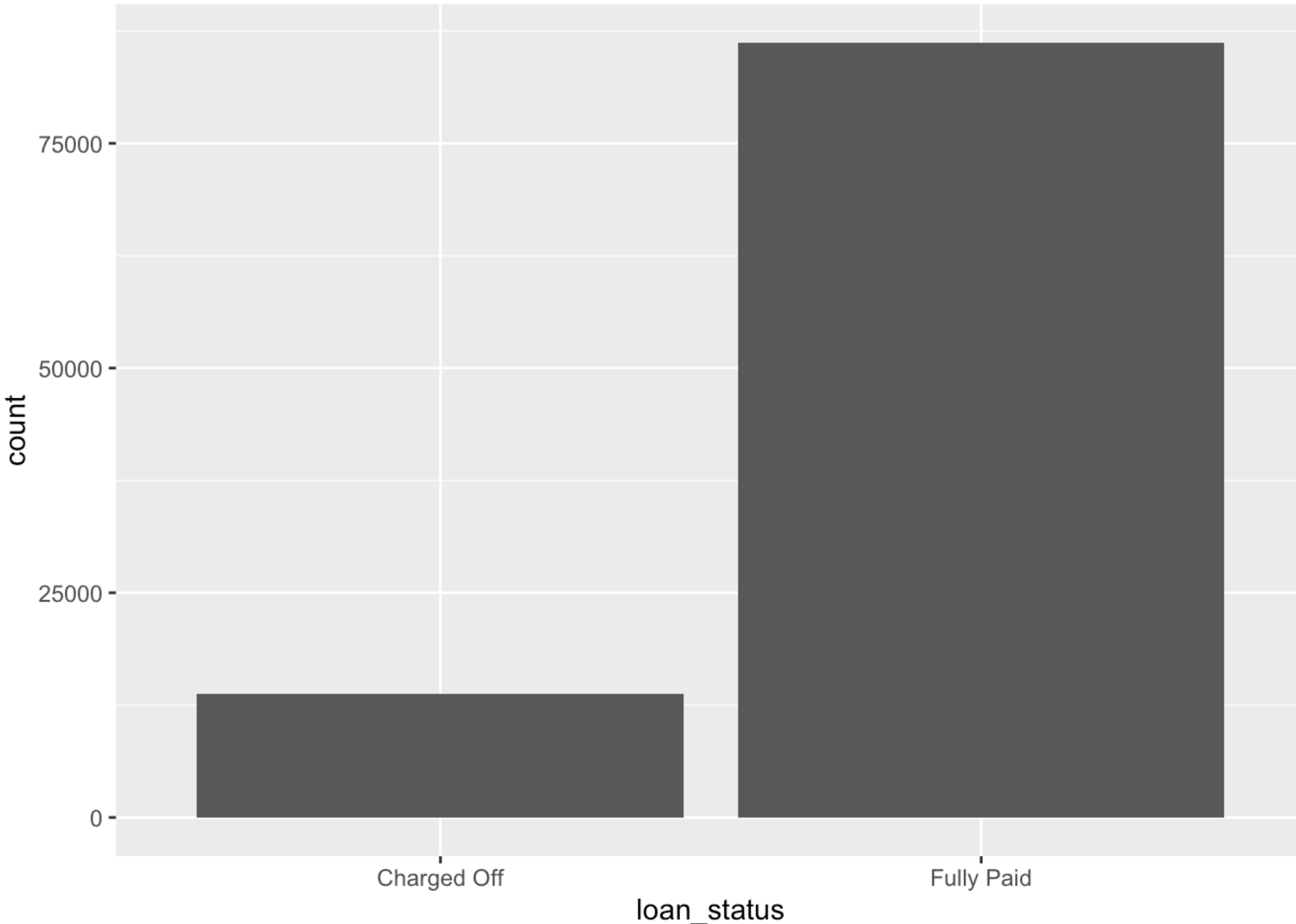
```
## Rows: 100000 Columns: 145
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr  (21): term, grade, sub_grade, emp_title, emp_length, home_ownership, ve...
## dbl  (84): loan_amnt, funded_amnt, funded_amnt_inv, int_rate, installment, a...
## lgl  (39): id, member_id, url, desc, next_pymnt_d, annual_inc_joint, dti_joi...
## dttm  (1): issue_d
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Question 2-a-i

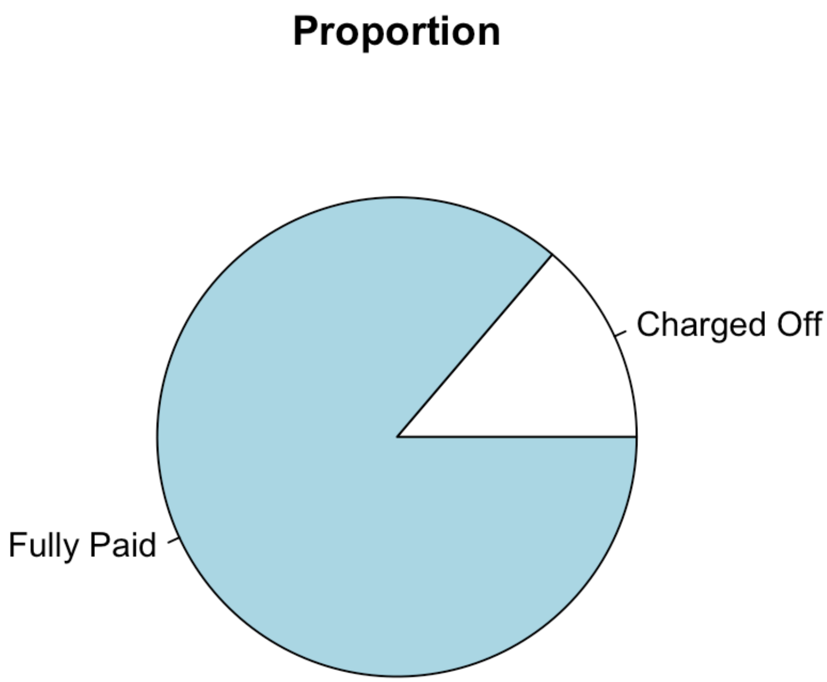
```
#Question 2 - Data Exploration
#Question 2(a) - (i)

#What is the proportion of defaults ('charged off' vs 'fully paid' loans) in the data
?
Prop_of_defaults <- LC_Data %>% group_by(loan_status)%>%summarise(n=n())%>%mutate(fre
q=n/sum(n)*100)
setnames(Prop_of_defaults, old = c('loan_status','n'), new = c('loanStatus','totalCou
nt'))
print(Prop_of_defaults)
```

```
## # A tibble: 2 × 3
##   loanStatus  totalCount  freq
##   <chr>            <int> <dbl>
## 1 Charged Off      13785  13.8
## 2 Fully Paid       86215  86.2
```

```
#Bar graph to visualize the proportion.
ggplot(LC_Data,aes(x=loan_status)) + geom_bar()
```



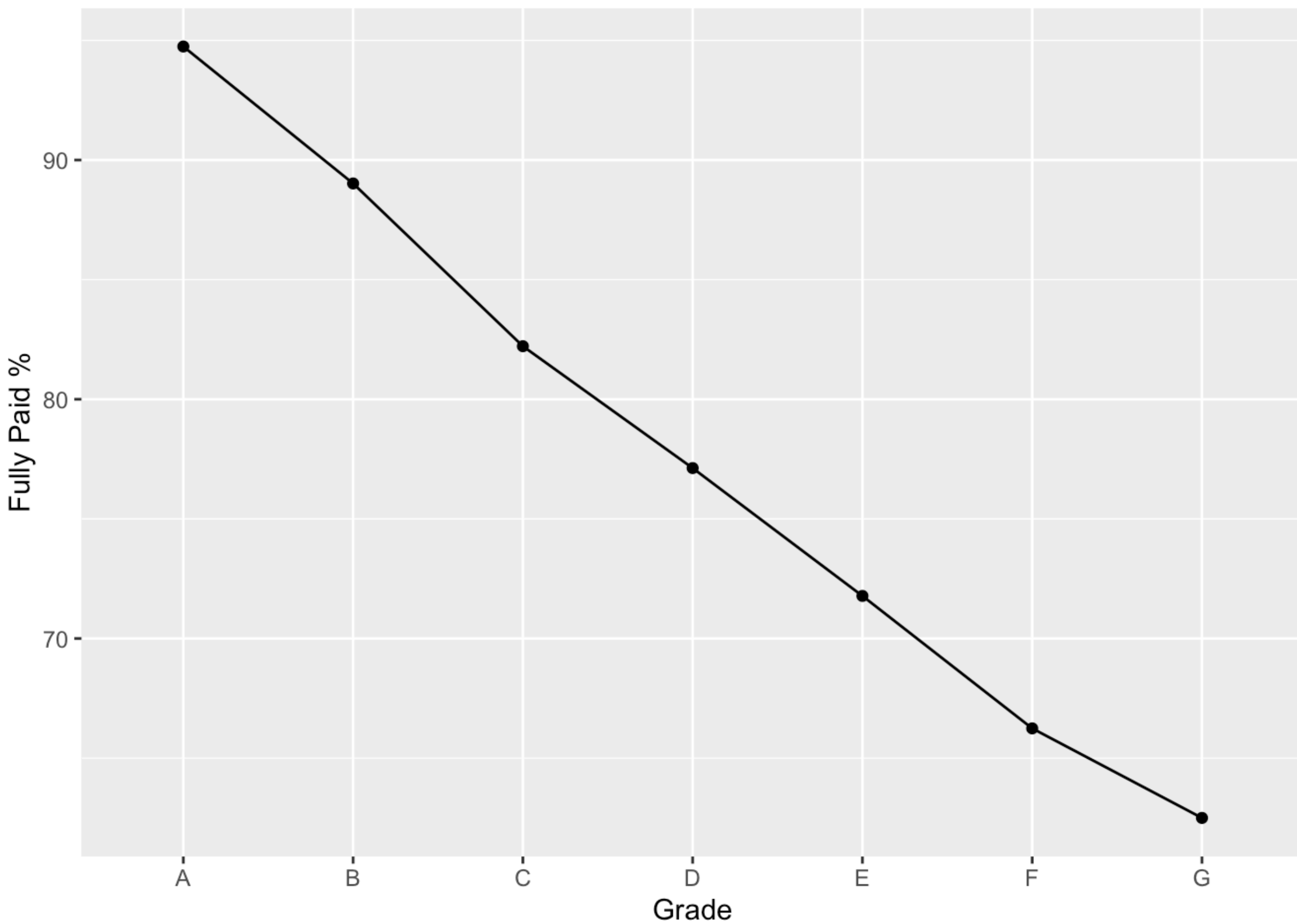

```r
#Pie chart representaion of the proportion of defaults.
lbls <- Prop_of_defaults$'loanStatus'
slices <- Prop_of_defaults$totalCount
pie(slices, labels = lbls, main="Proportion")
```




```
#Proportion of default rate by Grade:

defaultBygrade<-LC_Data%>%group_by(grade,loan_status)%>%summarise(n=n())%>%mutate(fre
q=n/sum(n)*100)
setnames(defaultBygrade, old = c('loan_status','n'), new = c('loanStatus','totalCount
'))
print(defaultBygrade)
```

```
## # A tibble: 14 × 4
## # Groups:   grade [7]
##    grade loanStatus  totalCount  freq
##    <chr> <chr>            <int> <dbl>
##  1 A     Charged Off       1187  5.26
##  2 A     Fully Paid       21401 94.7
##  3 B     Charged Off       3723 11.0
##  4 B     Fully Paid       30184 89.0
##  5 C     Charged Off       4738 17.8
##  6 C     Fully Paid       21907 82.2
##  7 D     Charged Off       2858 22.9
##  8 D     Fully Paid        9635 77.1
##  9 E     Charged Off       1010 28.2
## 10 E     Fully Paid        2569 71.8
## 11 F     Charged Off        239 33.8
## 12 F     Fully Paid         469 66.2
## 13 G     Charged Off         30 37.5
## 14 G     Fully Paid          50 62.5
```

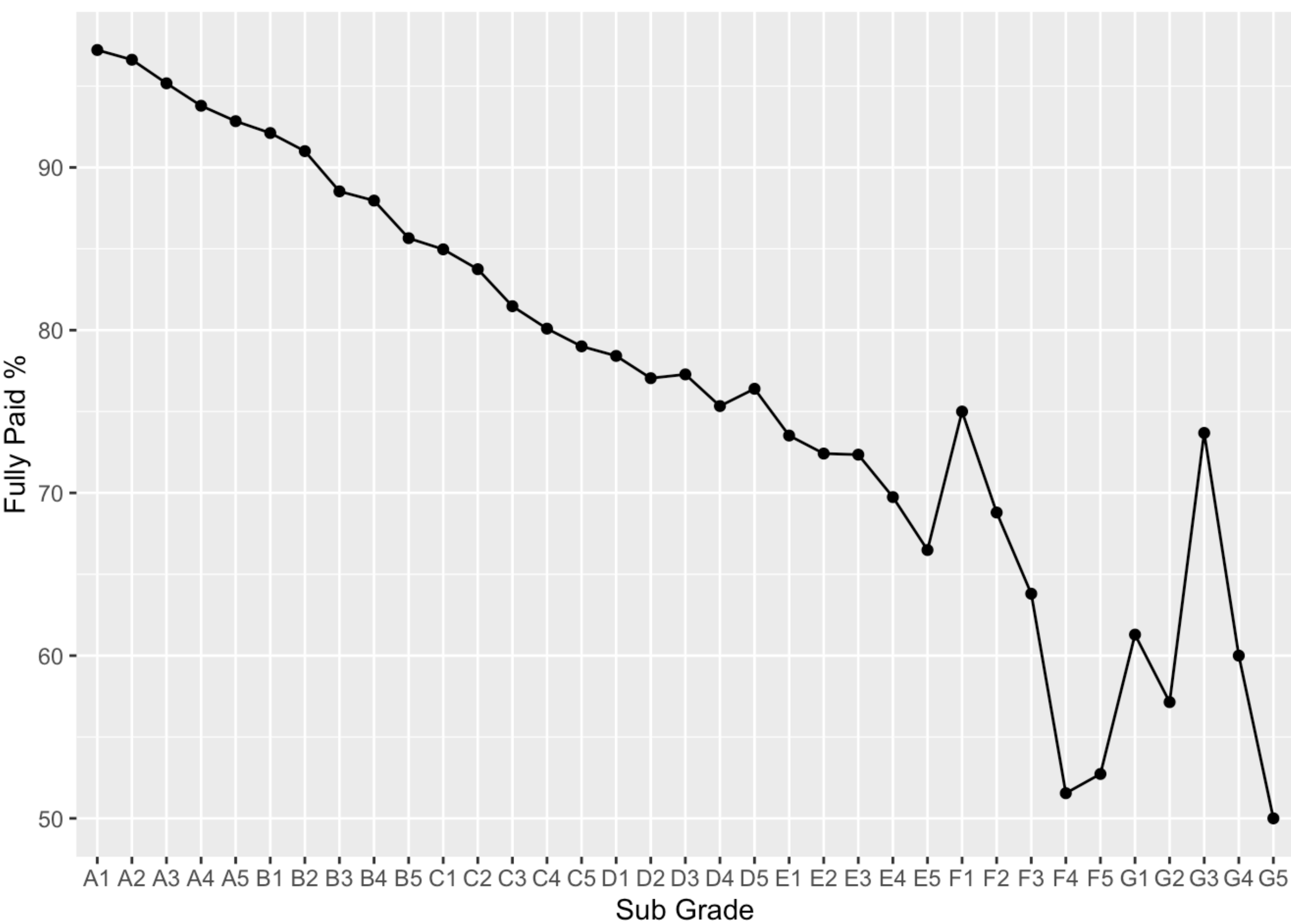
```
#Line graph representation of Fully paid% with Grade.
defaultBygrade=filter(defaultBygrade, loanStatus=="Fully Paid")
ggplot(data=defaultBygrade, aes(x=grade, y=freq, group=1)) +
  geom_line()+
  geom_point()+labs(y="Fully Paid %", x = "Grade")
```




```
#Proportion of default rate by SubGrade:

defaultBysubgrade<-LC_Data%>%group_by(sub_grade,loan_status)%>%summarise(n=n())%>%mut
ate(freq=n/sum(n)*100)
setnames(defaultBysubgrade, old = c('loan_status','n'), new = c('loanStatus','totalCo
unt'))
print(defaultBysubgrade)
```

```
## # A tibble: 70 × 4
## # Groups:   sub_grade [35]
##    sub_grade loanStatus  totalCount  freq
##    <chr>     <chr>            <int> <dbl>
##  1 A1        Charged Off        105  2.78
##  2 A1        Fully Paid        3669 97.2
##  3 A2        Charged Off        116  3.38
##  4 A2        Fully Paid        3315 96.6
##  5 A3        Charged Off        179  4.83
##  6 A3        Fully Paid        3527 95.2
##  7 A4        Charged Off        319  6.21
##  8 A4        Fully Paid        4819 93.8
##  9 A5        Charged Off        468  7.16
## 10 A5        Fully Paid        6071 92.8
## # … with 60 more rows
```

```
#Line graph representation of Fully paid% with Grade.
defaultBysubgrade=filter(defaultBysubgrade, loanStatus=="Fully Paid")
ggplot(data=defaultBysubgrade, aes(x=sub_grade, y=freq, group=1)) +
  geom_line()+
  geom_point()+labs(y="Fully Paid %", x = "Sub Grade")
```



Question 2-a-i

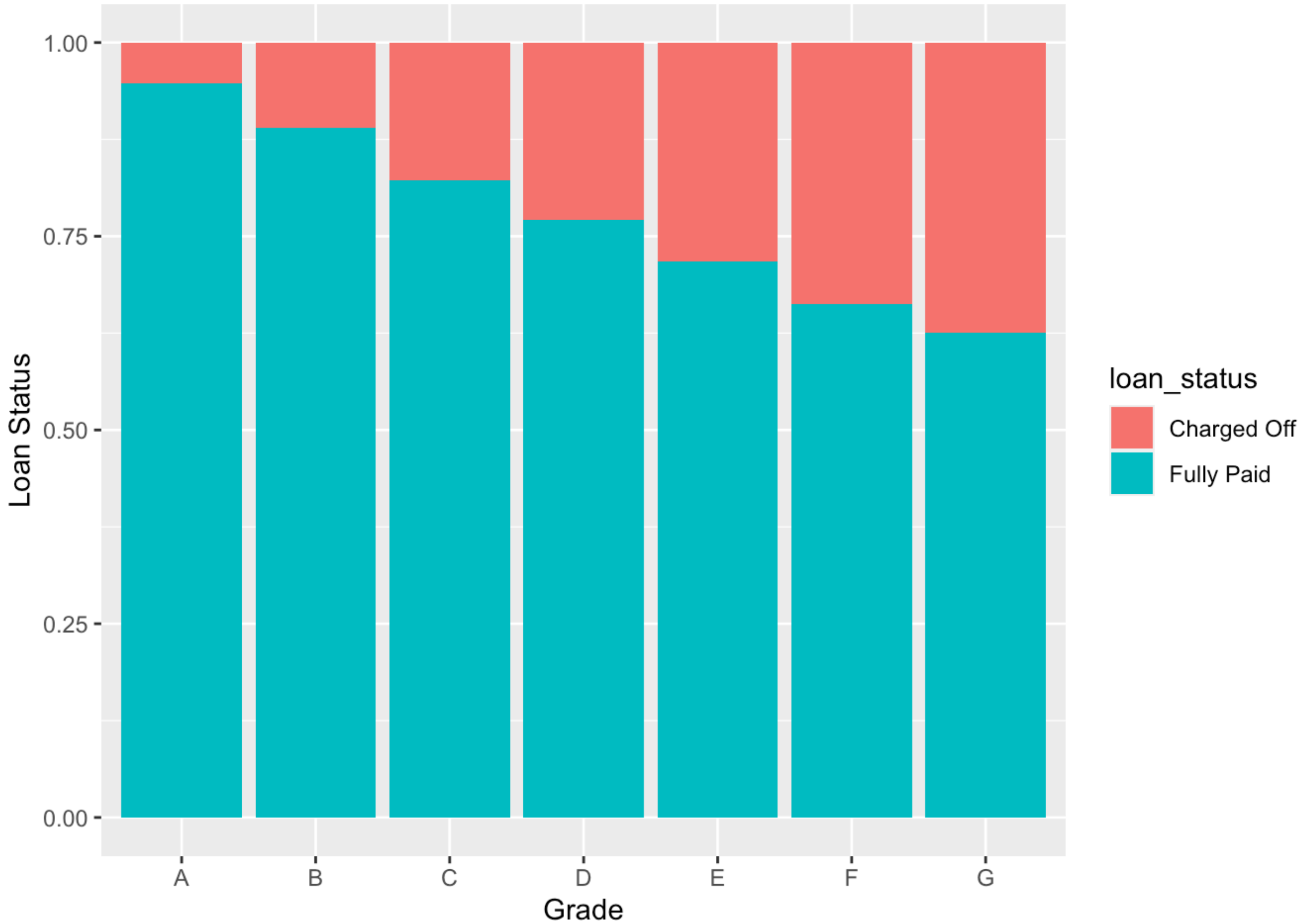
```
#How does default rate vary with loan grade? Does it vary with sub-grade? And is this
what you would expect, and why?


Defaultrate_LoanGrade <- LC_Data %>% group_by(grade) %>% tally()
setnames(Defaultrate_LoanGrade, old = c('grade','n'), new = c('Grade','Default Rate')
)
print(Defaultrate_LoanGrade)
```

```
## # A tibble: 7 × 2
##   Grade `Default Rate`
##   <chr>          <int>
## 1 A              22588
## 2 B              33907
## 3 C              26645
## 4 D              12493
## 5 E               3579
## 6 F                708
## 7 G                 80
```

```
Defaultrate_LoanSubGrade <- LC_Data %>% group_by(sub_grade) %>% tally()

setnames(Defaultrate_LoanSubGrade, old = c('sub_grade','n'), new = c('Sub Grade','Def
ault Rate'))
print(Defaultrate_LoanSubGrade)
```

```
## # A tibble: 35 × 2
##    `Sub Grade` `Default Rate`
##    <chr>                <int>
##  1 A1                    3774
##  2 A2                    3431
##  3 A3                    3706
##  4 A4                    5138
##  5 A5                    6539
##  6 B1                    6228
##  7 B2                    6880
##  8 B3                    7193
##  9 B4                    7103
## 10 B5                    6503
## # … with 25 more rows
```

```
#Bar graph showing the distribution of grades with loan status.
ggplot(LC_Data,aes(x= grade, fill = loan_status)) + geom_bar(position = "fill")+ labs
(y="Loan Status", x = "Grade")
```




```
#Bar graph showing the distribution of sub-grades with loan status.
ggplot(LC_Data,aes(x= sub_grade, fill = loan_status)) + geom_bar(position = "fill") +
labs(y="Loan Status", x = "Sub Grade")
```

Question 2-a-ii

```
#How many loans are there in each grade? And do loan amounts vary by grade?

#Loans in each grade.

LoansCount_EachGrade <- LC_Data %>% group_by(grade) %>% tally()
setnames(LoansCount_EachGrade, old = c('grade','n'), new = c('Grade','Count'))
print(LoansCount_EachGrade)
```

```
## # A tibble: 7 × 2
##   Grade Count
##   <chr> <int>
## 1 A     22588
## 2 B     33907
## 3 C     26645
## 4 D     12493
## 5 E      3579
## 6 F       708
## 7 G        80
```

```
#Loans variation by grade.

Loans_EachGrade <- LC_Data %>% group_by(grade) %>% summarise(sum(loan_amnt))
setnames(Loans_EachGrade, old = c('grade','sum(loan_amnt)'), new = c('Grade','Sum of
amounts'))
print(Loans_EachGrade)
```

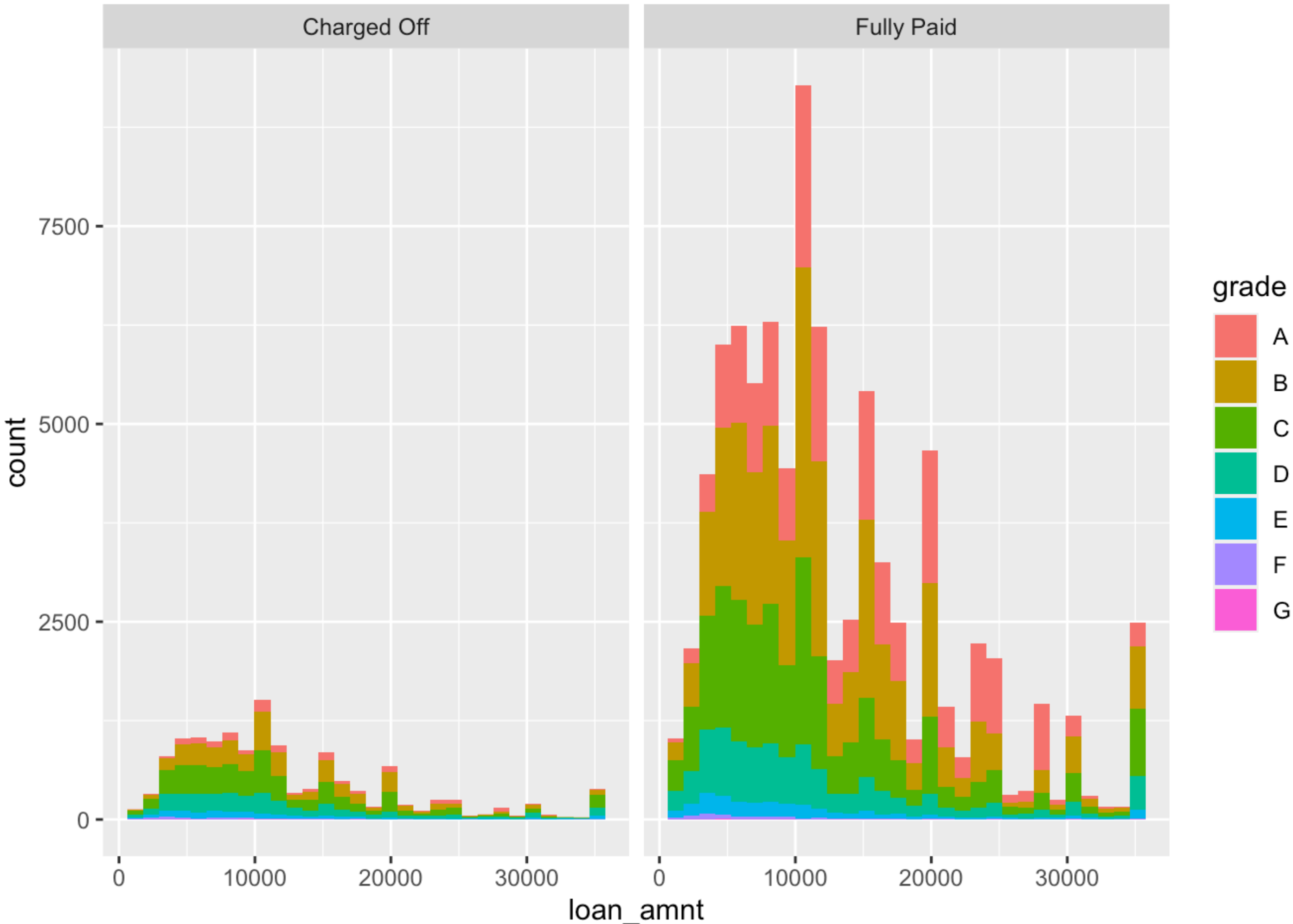
```
## # A tibble: 7 × 2
##   Grade `Sum of amounts`
##   <chr>            <dbl>
## 1 A            327649125
## 2 B            428494575
## 3 C            319762050
## 4 D            148590825
## 5 E             41583800
## 6 F              6564925
## 7 G               946075
```

```
#Loans variation by sub-grade.
Loans_EachSubGrade <- LC_Data %>% group_by(sub_grade) %>% summarise(sum(loan_amnt))
setnames(Loans_EachSubGrade, old = c('sub_grade','sum(loan_amnt)'), new = c('Sub Grad
e','Sum of amounts'))
print(Loans_EachSubGrade)
```

```
## # A tibble: 35 × 2
##    `Sub Grade` `Sum of amounts`
##    <chr>                  <dbl>
##  1 A1                  54621675
##  2 A2                  48499650
##  3 A3                  53865600
##  4 A4                  75401500
##  5 A5                  95260700
##  6 B1                  80444900
##  7 B2                  89162825
##  8 B3                  91849325
##  9 B4                  87767175
## 10 B5                  79270350
## # … with 25 more rows
```

```
#Graph view- segregating Charged off vs Fully Paid
ggplot(LC_Data, aes( x = loan_amnt)) + geom_histogram(aes(fill=grade)) + facet_wrap(~
loan_status)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



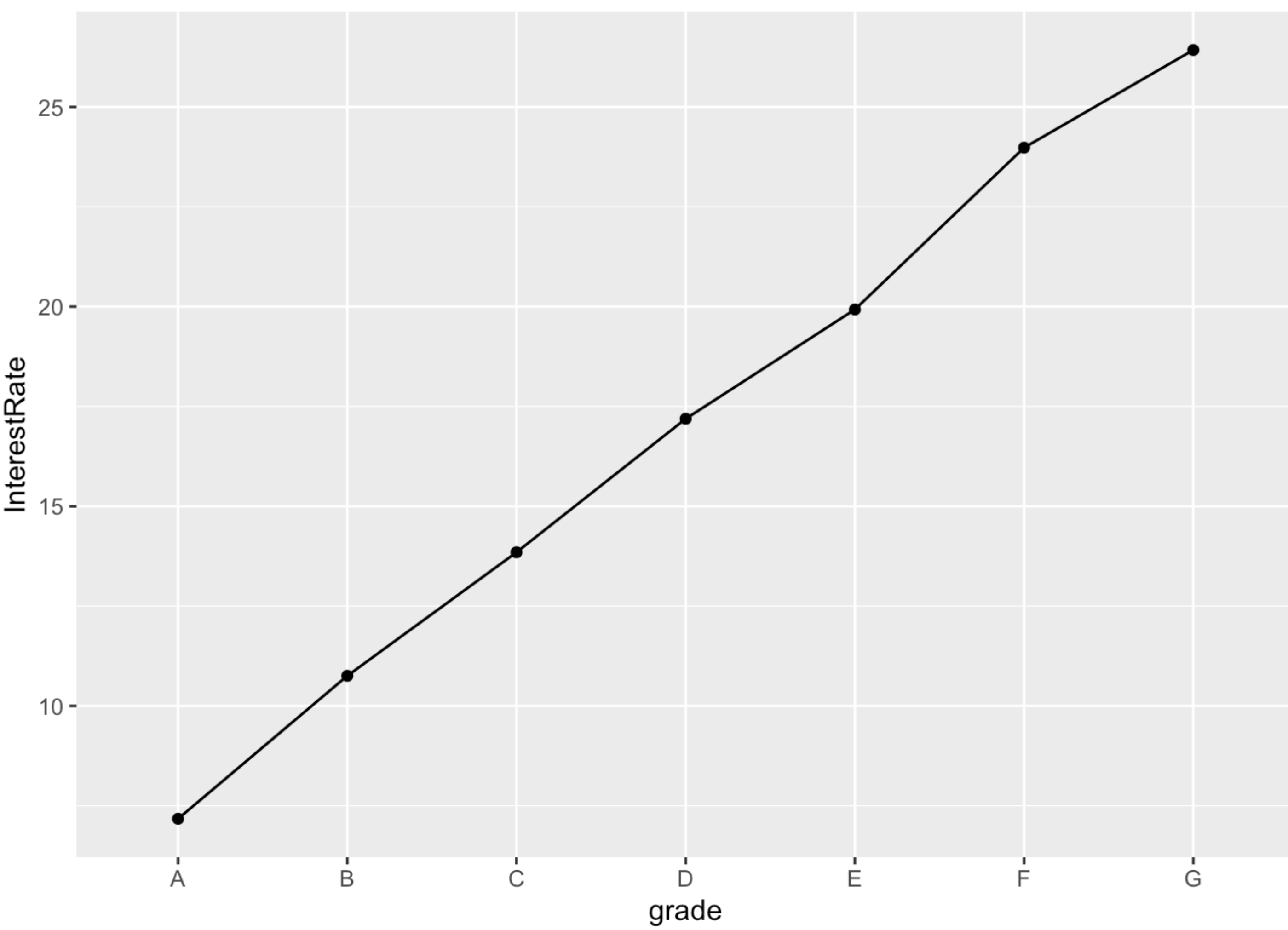

```
#Calculating mean interest to compare with grade and subgrade.
#Comparison with Grade
int_bygrade <- LC_Data %>% group_by(grade) %>% summarise(InterestRate = mean(int_rate
))
print(int_bygrade)
```

```
## # A tibble: 7 × 2
##   grade InterestRate
##   <chr>        <dbl>
## 1 A             7.17
## 2 B            10.8
## 3 C            13.8
## 4 D            17.2
## 5 E            19.9
## 6 F            24.0
## 7 G            26.4
```

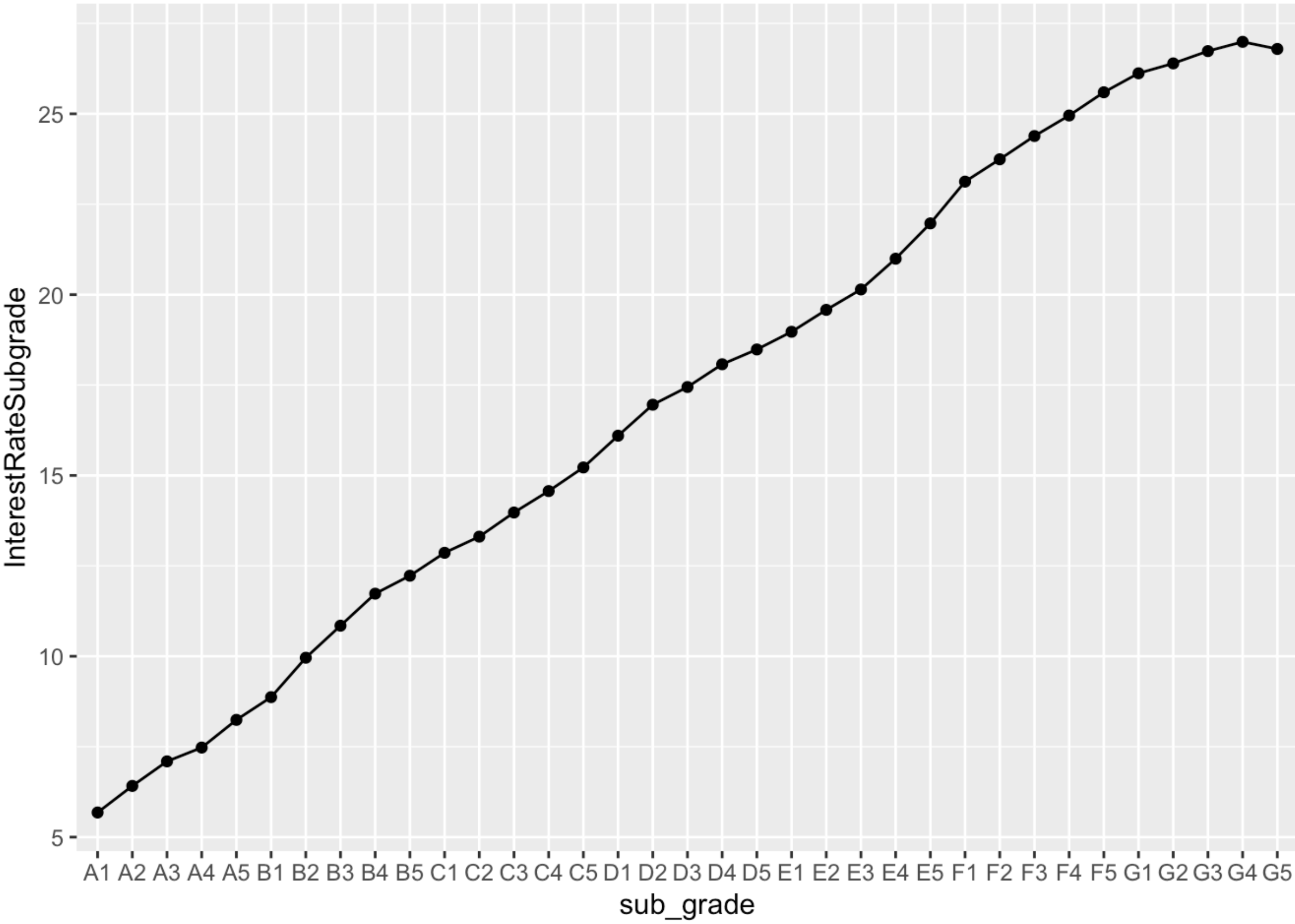
```
#Comparison with Subgrade
int_bysubgrade <- LC_Data %>% group_by(sub_grade) %>% summarise(InterestRateSubgrade
= mean(int_rate))
print(int_bysubgrade)
```

```
## # A tibble: 35 × 2
##    sub_grade InterestRateSubgrade
##    <chr>                    <dbl>
##  1 A1                        5.68
##  2 A2                        6.42
##  3 A3                        7.09
##  4 A4                        7.48
##  5 A5                        8.24
##  6 B1                        8.87
##  7 B2                        9.96
##  8 B3                       10.8
##  9 B4                       11.7
## 10 B5                       12.2
## # … with 25 more rows
```

```
#Plot for mean Interest rate with grade.
ggplot(int_bygrade,aes(x=grade, y =InterestRate, group =1)) + geom_line() + geom_poin
t()
```



```
#Plot for mean Interest rate with sub grade.
ggplot(int_bysubgrade,aes(x=sub_grade, y =InterestRateSubgrade, group =1)) + geom_line() + geom_point()
```



Question 2-a-ii

```
#Summary for Average and standard-deviation of Interest rate by grade and subgrade.

characteristics_intRate_grade <- LC_Data %>% group_by(grade) %>% summarise(numLoans=n(), avgInterest = mean(int_rate), std_dev_Interest = sd(int_rate))
print(characteristics_intRate_grade)
```

```
## # A tibble: 7 × 4
##   grade numLoans avgInterest std_dev_Interest
##   <chr>    <int>       <dbl>            <dbl>
## 1 A        22588        7.17            0.967
## 2 B        33907       10.8             1.44
## 3 C        26645       13.8             1.19
## 4 D        12493       17.2             1.22
## 5 E         3579       19.9             1.38
## 6 F          708       24.0             0.916
## 7 G           80       26.4             0.849
```

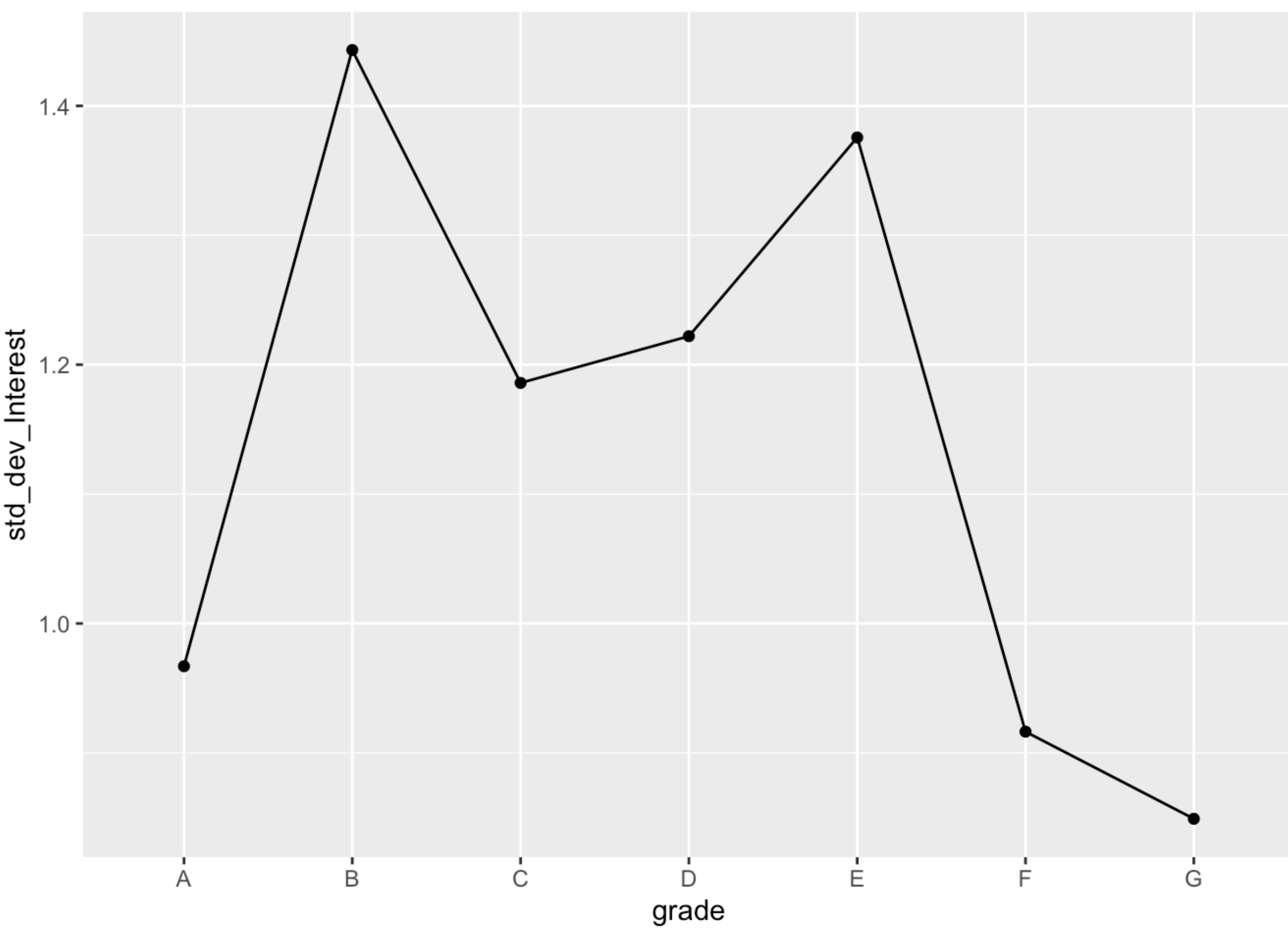
```
characteristics_intRate_subgrade <- LC_Data %>% group_by(sub_grade) %>% summarise(num
Loans=n(), avgInterest = mean(int_rate), std_dev_Interest = sd(int_rate))
print(characteristics_intRate_subgrade)
```

```
## # A tibble: 35 × 4
##    sub_grade numLoans avgInterest std_dev_Interest
##    <chr>        <int>       <dbl>            <dbl>
##  1 A1            3774        5.68            0.347
##  2 A2            3431        6.42            0.166
##  3 A3            3706        7.09            0.325
##  4 A4            5138        7.48            0.357
##  5 A5            6539        8.24            0.424
##  6 B1            6228        8.87            0.722
##  7 B2            6880        9.96            0.816
##  8 B3            7193       10.8             0.887
##  9 B4            7103       11.7             0.840
## 10 B5            6503       12.2             0.851
## # … with 25 more rows
```

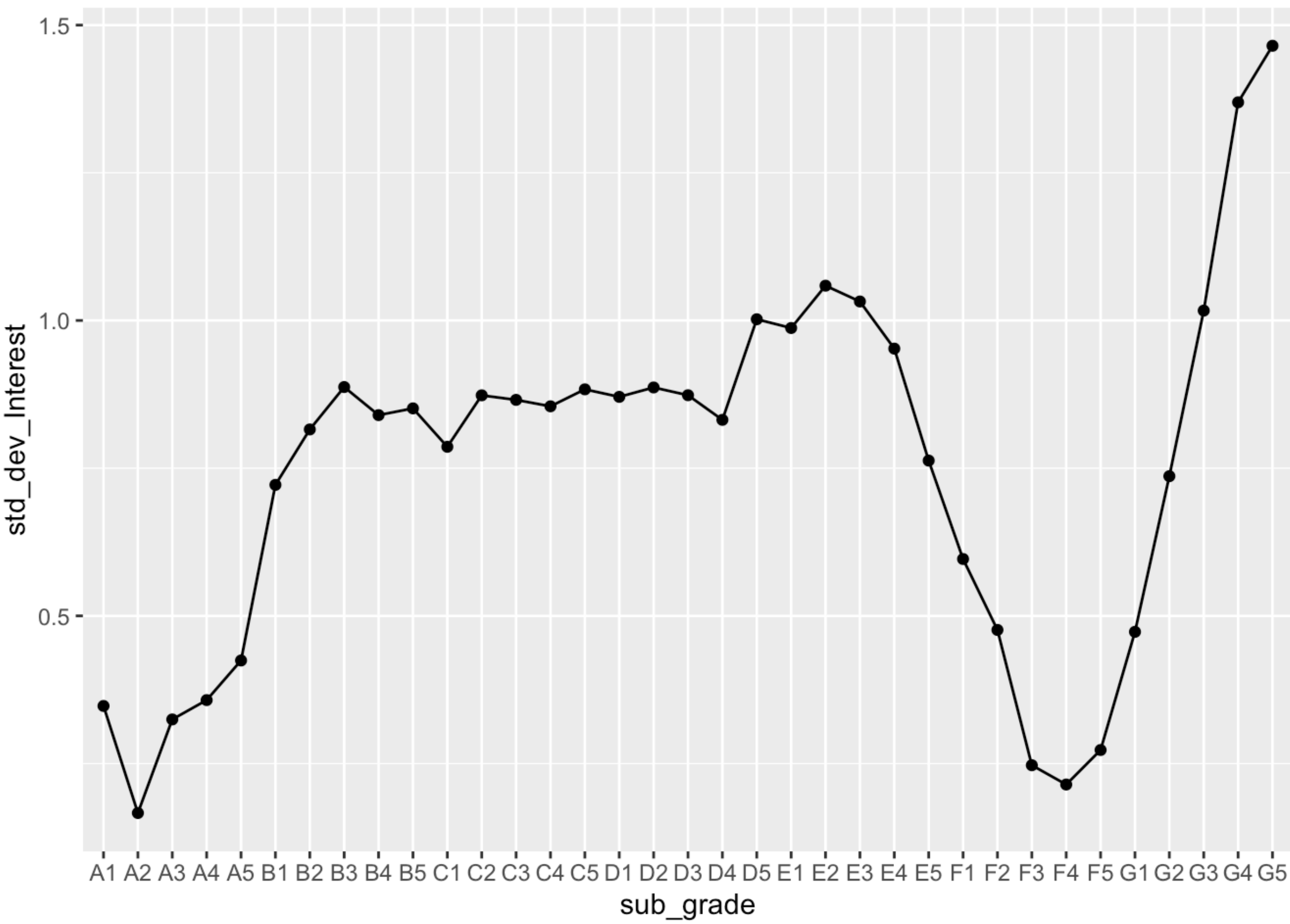
```
mean_int <- LC_Data %>% group_by(grade,sub_grade) %>% summarise(mean_intRate = mean(i
nt_rate))
print(mean_int)
```

```
## # A tibble: 35 × 3
## # Groups:   grade [7]
##    grade sub_grade mean_intRate
##    <chr> <chr>            <dbl>
##  1 A     A1                5.68
##  2 A     A2                6.42
##  3 A     A3                7.09
##  4 A     A4                7.48
##  5 A     A5                8.24
##  6 B     B1                8.87
##  7 B     B2                9.96
##  8 B     B3               10.8 
##  9 B     B4               11.7 
## 10 B     B5               12.2 
## # … with 25 more rows
```

```
#Line plot for Standard dev of int rate versus grades of loans.
ggplot(characteristics_intRate_grade,aes(x=grade, y =std_dev_Interest, group =1)) + g
eom_line() + geom_point()
```



```r
#Line plot for Standard dev of int rate versus sub grades of loans.
ggplot(characteristics_intRate_subgrade,aes(x=sub_grade, y =std_dev_Interest, group =
1)) + geom_line() + geom_point()
```




```r
#Minimum interest rates for each grades and subgrades
min_intRate_grade <-LC_Data %>% group_by(grade) %>% summarize(min(int_rate))
print(min_intRate_grade)
```

```
## # A tibble: 7 × 2
##   grade `min(int_rate)`
##   <chr>           <dbl>
## 1 A                5.32
## 2 B                6
## 3 C                6
## 4 D                6
## 5 E                6
## 6 F               22.0
## 7 G               25.8
```

```r
min_intRate_subgrade <-LC_Data %>% group_by(sub_grade) %>% summarize(min(int_rate))
print(min_intRate_subgrade)
```

```
## # A tibble: 35 × 2
##    sub_grade `min(int_rate)`
##    <chr>               <dbl>
##  1 A1                   5.32
##  2 A2                   6.24
##  3 A3                   6.68
##  4 A4                   6.92
##  5 A5                   6
##  6 B1                   6
##  7 B2                   6
##  8 B3                   6
##  9 B4                   6
## 10 B5                   6
## # … with 25 more rows
```

```r
#Maximum interest rates for each grades and subgrades
max_intRate_grade <-LC_Data %>% group_by(grade) %>% summarize(max(int_rate))
print(max_intRate_grade)
```

```
## # A tibble: 7 × 2
##   grade `max(int_rate)`
##   <chr>           <dbl>
## 1 A                9.25
## 2 B               14.1
## 3 C               17.3
## 4 D               20.3
## 5 E               23.4
## 6 F               26.1
## 7 G               29.0
```

```r
max_intRate_subgrade <-LC_Data %>% group_by(sub_grade) %>% summarize(max(int_rate))
print(max_intRate_subgrade)
```

```
## # A tibble: 35 × 2
##    sub_grade `max(int_rate)`
##    <chr>                <dbl>
##  1 A1                    6.03
##  2 A2                    6.97
##  3 A3                    7.62
##  4 A4                    8.6
##  5 A5                    9.25
##  6 B1                   10.2
##  7 B2                   11.1
##  8 B3                   12.1
##  9 B4                   13.1
## 10 B5                   14.1
## # … with 25 more rows
```

Question 2-a-iii

```
#Data For loans fully paid - time-to-payoff

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

```
## # A tibble: 6 × 2
##   last_pymnt_d issue_d
##   <chr>        <dttm>
## 1 Jul-2016     2015-05-01 00:00:00
## 2 Jun-2017     2015-07-01 00:00:00
## 3 Nov-2017     2014-11-01 00:00:00
## 4 Aug-2015     2014-03-01 00:00:00
## 5 Nov-2017     2015-04-01 00:00:00
## 6 Mar-2016     2014-01-01 00:00:00
```

```
LC_Data$last_pymnt_d<-paste(LC_Data$last_pymnt_d, "-01", sep = "")
#     Then convert this character to a date type variable
LC_Data$last_pymnt_d<-parse_date_time(LC_Data$last_pymnt_d,  "myd")
```

```
## Warning: 64 failed to parse.
```

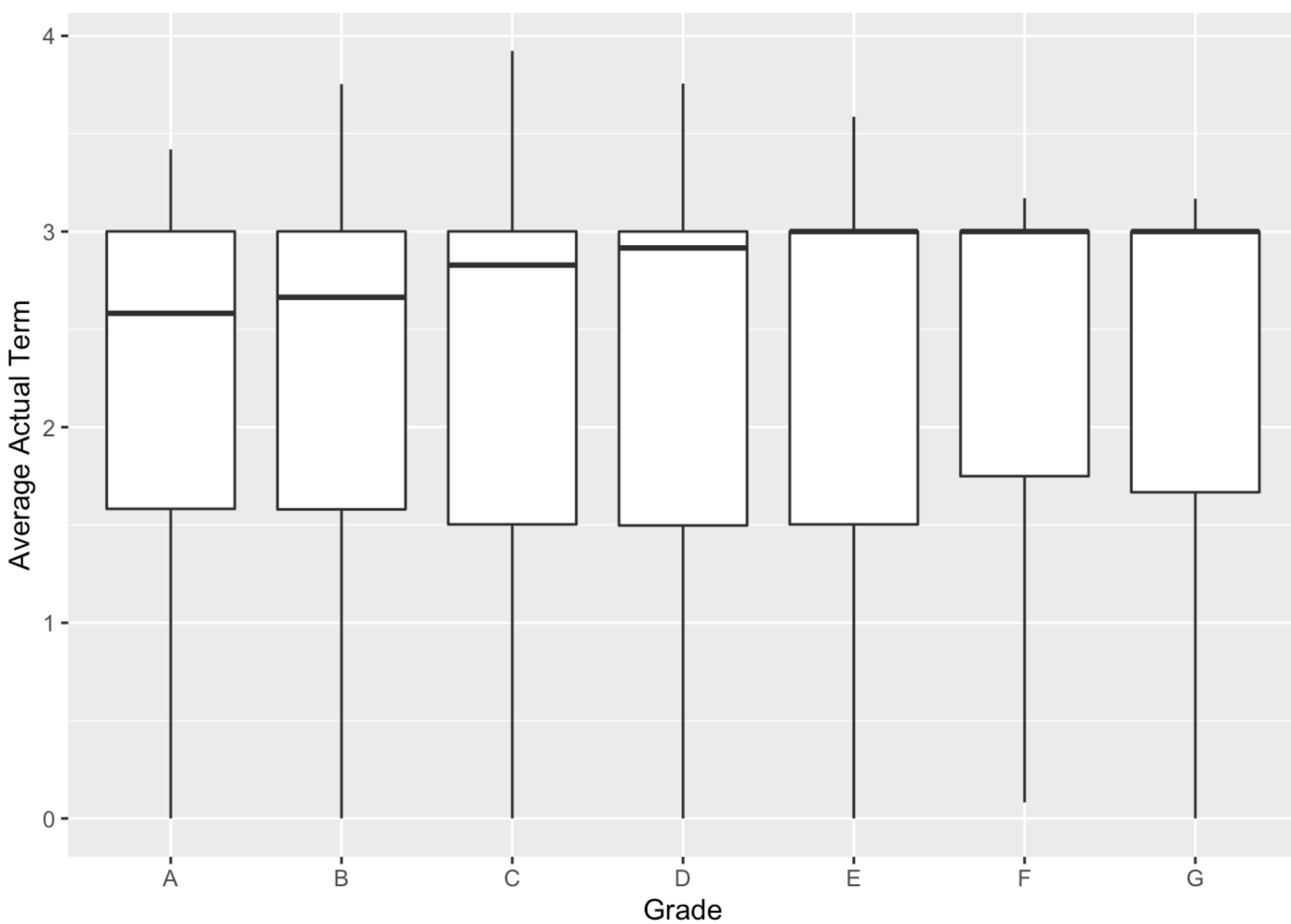
```
head(LC_Data[, c("last_pymnt_d", "issue_d")])
```

```
## # A tibble: 6 × 2
##   last_pymnt_d        issue_d
##   <dttm>              <dttm>
## 1 2016-07-01 00:00:00 2015-05-01 00:00:00
## 2 2017-06-01 00:00:00 2015-07-01 00:00:00
## 3 2017-11-01 00:00:00 2014-11-01 00:00:00
## 4 2015-08-01 00:00:00 2014-03-01 00:00:00
## 5 2017-11-01 00:00:00 2015-04-01 00:00:00
## 6 2016-03-01 00:00:00 2014-01-01 00:00:00
```

```
LC_Data$actualTerm <- ifelse(LC_Data$loan_status=="Fully Paid", as.duration(LC_Data$i
ssue_d  %--% LC_Data$last_pymnt_d)/dyears(1), 3)


head(LC_Data$actualTerm)
```

```
## [1] 1.169062 1.919233 3.000684 1.418207 2.587269 2.162902
```

```
ggplot(LC_Data, aes(x=actualTerm, y=grade)) + geom_boxplot()+coord_flip()+labs(y="Gra
de", x = "Average Actual Term")
```



```
dim(LC_Data)
```

```
## [1] 100000    146
```

Question 2-a-iv

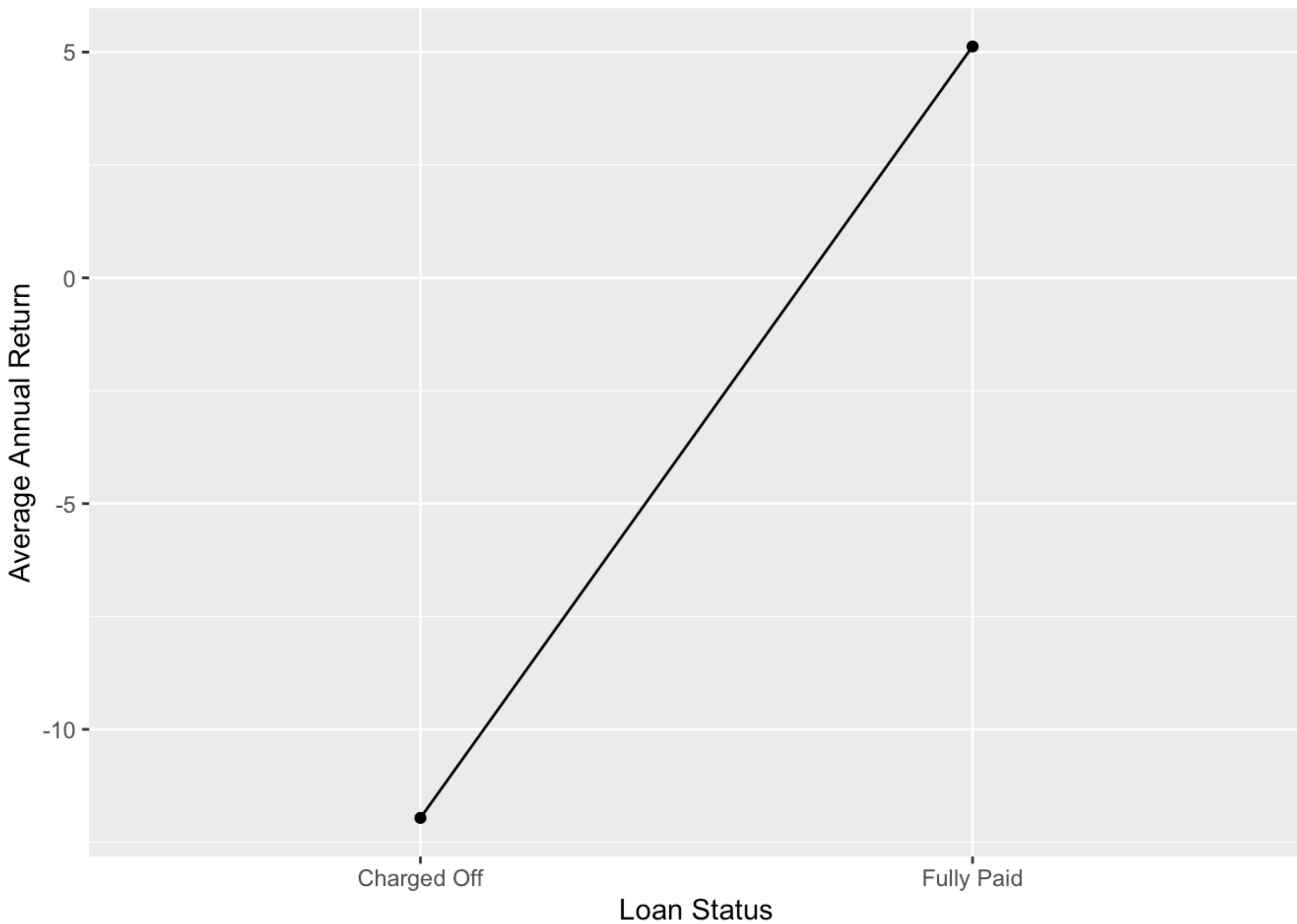
```
#Annualized percent return:
#LC_Data$annRet <- ((LC_Data$total_pymnt
#                   -LC_Data$funded_amnt)/LC_Data$funded_amnt)*(12/36)*100
#print(LC_Data$annRet)

#Actual Annual Return percentage
LC_Data$annRet <- ifelse(LC_Data$actualTerm>0, (((LC_Data$total_pymnt-LC_Data$funded_
amnt)/LC_Data$funded_amnt)/LC_Data$actualTerm)*100,0)


AnnualRetrun <- ((LC_Data$total_pymnt -LC_Data$funded_amnt)/LC_Data$funded_amnt)*(12/
36)*100

#Return from charged off loans vary by loan grades

LC_Data$return = LC_Data$total_pymnt-LC_Data$funded_amnt
LC_Data$returnperyear = (LC_Data$return/LC_Data$funded_amnt)/3*100

#Table for return per year  - grade and loan status.
return_defaults<-LC_Data%>%group_by(grade,loan_status)%>%summarise(return_peryear=mea
n(returnperyear))
print(return_defaults)
```

```
## # A tibble: 14 × 3
## # Groups:   grade [7]
##    grade loan_status return_peryear
##    <chr> <chr>                <dbl>
##  1 A     Charged Off         -11.6
##  2 A     Fully Paid            3.17
##  3 B     Charged Off         -11.5
##  4 B     Fully Paid            4.73
##  5 C     Charged Off         -12.0
##  6 C     Fully Paid            6.03
##  7 D     Charged Off         -12.4
##  8 D     Fully Paid            7.42
##  9 E     Charged Off         -12.7
## 10 E     Fully Paid            8.56
## 11 F     Charged Off         -11.9
## 12 F     Fully Paid           10.6
## 13 G     Charged Off         -14.4
## 14 G     Fully Paid           10.6
```

```
#Table for returns per year from default loans  - grade.
retdef=filter(LC_Data, loan_status=="Charged Off")
returns_defaults<-retdef%>%group_by(grade)%>%summarise(mean_returnperyear=mean(return
peryear),sd_returnper=sd(returnperyear),min_returnperyear=min(returnperyear),max_retu
rnperyear=max(returnperyear))
print(returns_defaults)
```

```
## # A tibble: 7 × 5
##   grade mean_returnperyear sd_returnper min_returnperyear max_returnperyear
##   <chr>              <dbl>        <dbl>             <dbl>             <dbl>
## 1 A                  -11.6         8.49             -32.3              5.17
## 2 B                  -11.5         8.78             -32.5              7.00
## 3 C                  -12.0         9.24             -33.3             13.6 
## 4 D                  -12.4         9.79             -33.3             12.3 
## 5 E                  -12.7        10.1              -33.3             12.4 
## 6 F                  -11.9        10.7              -32.1             13.7 
## 7 G                  -14.4         9.14             -30.7              9.02
```

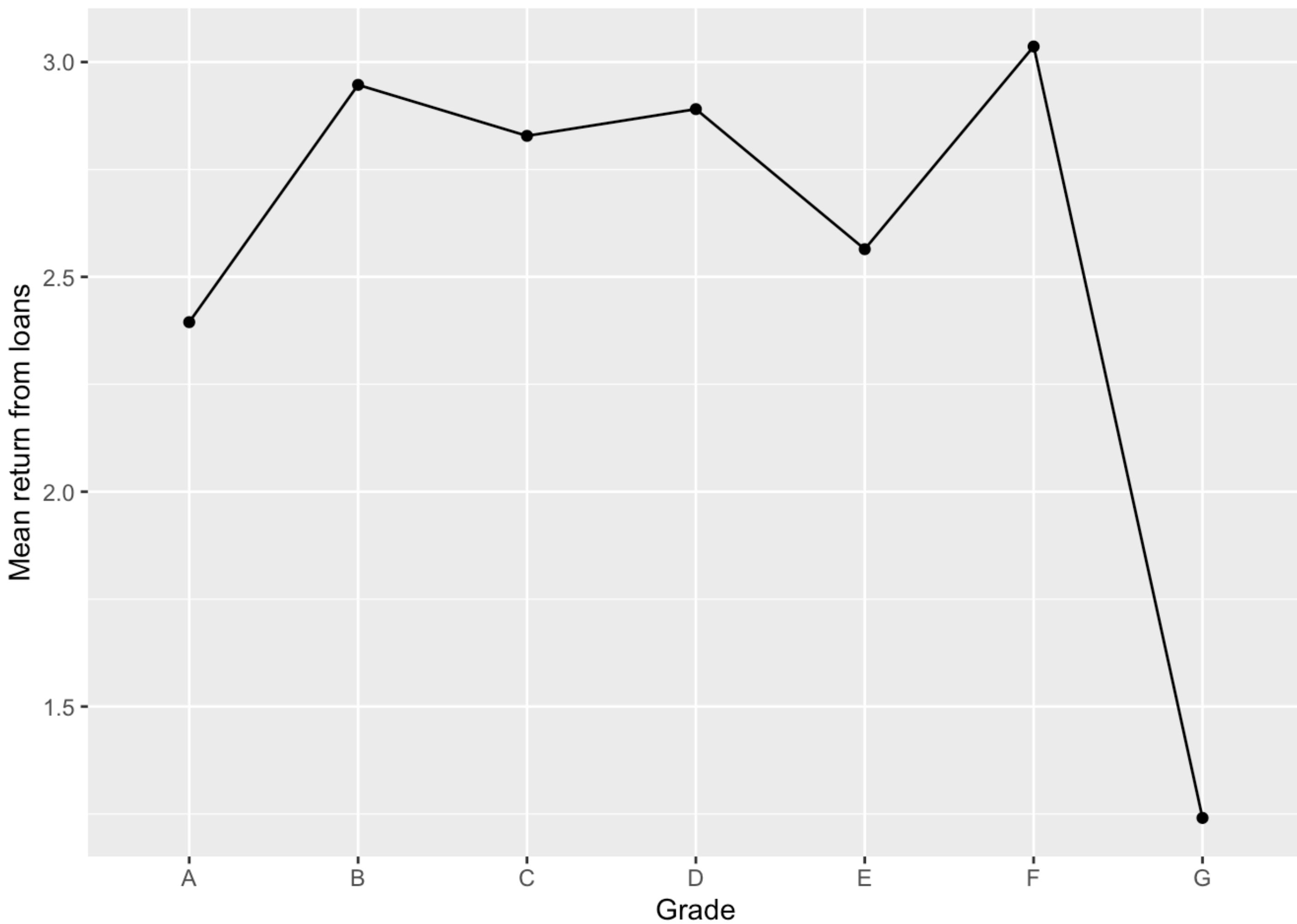
```
ret_loan_status<-LC_Data%>%group_by(loan_status)%>%summarise(mean_returnperyear=mean(
returnperyear),sd_returnperyear=sd(returnperyear),min_returnperyear=min(returnperyear
),
                                                   max_returnper=max(returnperyear))

ggplot(data=ret_loan_status, aes(x=loan_status, y=mean_returnperyear, group=1)) +
  geom_line()+
  geom_point()+labs(y="Average Annual Return", x = "Loan Status")
```




```
returns_grade<-LC_Data%>%group_by(grade)%>%summarise(mean_returnperyear=mean(returnpe
ryear),sd_returnperyear=sd(returnperyear),min_returnperyear=min(returnperyear),max_re
turnperyear=max(returnperyear))
print(returns_grade)
```

```
## # A tibble: 7 × 5
##   grade mean_returnperyear sd_returnperyear min_returnperyear max_returnperyear
##   <chr>              <dbl>            <dbl>             <dbl>             <dbl>
## 1 A                   2.39             3.94             -32.3              5.17
## 2 B                   2.95             6.05             -32.5              7.90
## 3 C                   2.83             8.14             -33.3             13.6 
## 4 D                   2.89             9.84             -33.3             12.3 
## 5 E                   2.56            11.3              -33.3             14.6 
## 6 F                   3.04            12.8              -32.1             15.2 
## 7 G                   1.24            14.1              -30.7             16.5
```

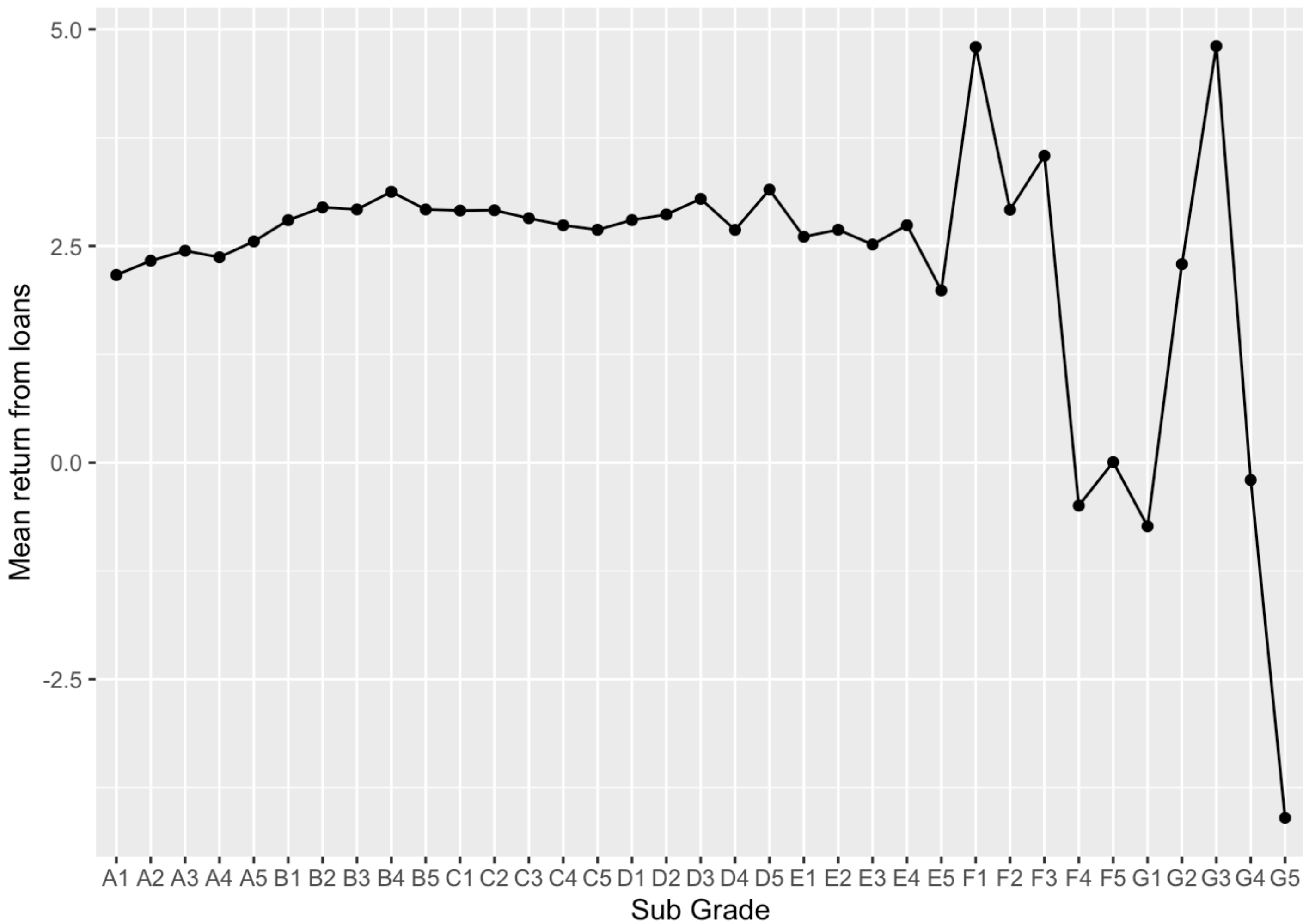
```
#Line plot for return from loans versus grades.
ggplot(returns_grade, aes(x=grade, y=mean_returnperyear,group =1)) + geom_line()+geom
_point() +labs(y="Mean return from loans", x = "Grade")
```




```
#Return from loans vary by loan sub grades
return_subgrade<-LC_Data%>%group_by(sub_grade)%>%summarise(mean_returnperyear=mean(re
turnperyear),sd_returnperyear=sd(returnperyear),min_returnperyear=min(returnperyear),
max_returnperyear=max(returnperyear))
print(return_subgrade)
```

```
## # A tibble: 35 × 5
##    sub_grade mean_returnperyear sd_returnperyear min_returnperyear max_returnp…¹
##    <chr>                  <dbl>            <dbl>             <dbl>         <dbl>
##  1 A1                      2.17             2.49             -27.6          3.34
##  2 A2                      2.33             3.15             -32.3          3.70
##  3 A3                      2.44             3.75             -31.3          4.25
##  4 A4                      2.37             4.34             -32.3          4.62
##  5 A5                      2.55             4.69             -31.3          5.17
##  6 B1                      2.80             4.81             -31.3          5.56
##  7 B2                      2.95             5.48             -32.3          6.15
##  8 B3                      2.92             6.18             -32.3          6.86
##  9 B4                      3.13             6.45             -32.5          7.59
## 10 B5                      2.92             7.01             -32.3          7.90
## # … with 25 more rows, and abbreviated variable name ¹max_returnperyear
```

```
#Line plot for return from loans versus sub grades.
ggplot(return_subgrade, aes(x=sub_grade, y=mean_returnperyear,group =1)) + geom_line(
)+geom_point() +labs(y="Mean return from loans", x = "Sub Grade")
```




```
#Average returns versus Average interest rate:

returns_intRate_grade<-LC_Data%>%group_by(grade)%>%summarise(mean_returnperyear=mean(
returnperyear),avgIntrate=mean(int_rate))
print(returns_intRate_grade)
```

```
## # A tibble: 7 × 3
##   grade mean_returnperyear avgIntrate
##   <chr>              <dbl>      <dbl>
## 1 A                   2.39       7.17
## 2 B                   2.95      10.8
## 3 C                   2.83      13.8
## 4 D                   2.89      17.2
## 5 E                   2.56      19.9
## 6 F                   3.04      24.0
## 7 G                   1.24      26.4
```

```
returns_intRate_subgrade<-LC_Data%>%group_by(sub_grade)%>%summarise(mean_returnperyea
r=mean(returnperyear),avgIntrate=mean(int_rate))
print(returns_intRate_subgrade)
```

```
## # A tibble: 35 × 3
##    sub_grade mean_returnperyear avgIntrate
##    <chr>                  <dbl>      <dbl>
##  1 A1                      2.17       5.68
##  2 A2                      2.33       6.42
##  3 A3                      2.44       7.09
##  4 A4                      2.37       7.48
##  5 A5                      2.55       8.24
##  6 B1                      2.80       8.87
##  7 B2                      2.95       9.96
##  8 B3                      2.92      10.8 
##  9 B4                      3.13      11.7 
## 10 B5                      2.92      12.2 
## # … with 25 more rows
```

```
dim(LC_Data)
```

```
## [1] 100000    149
```

Question 2-a-v

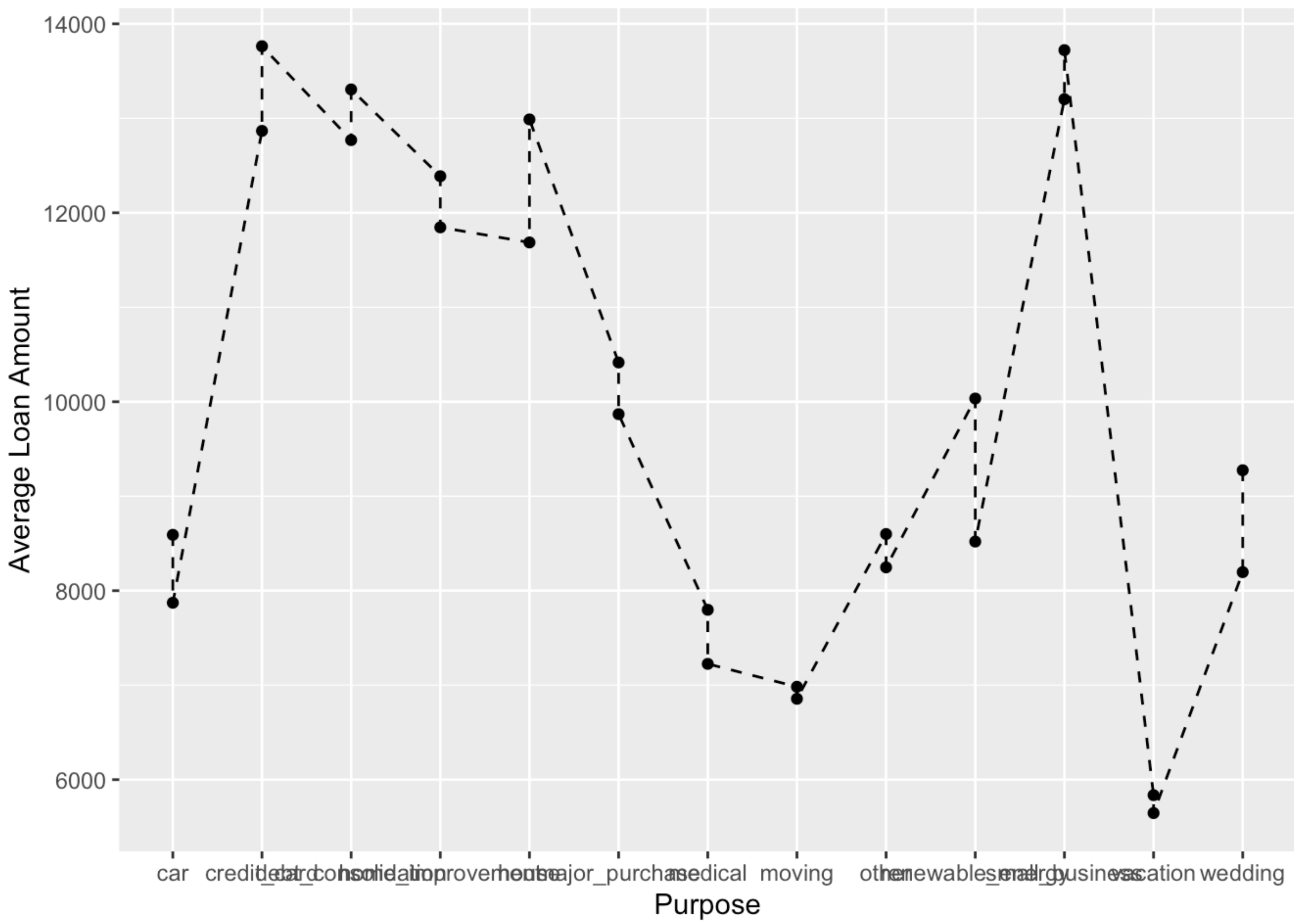
```
#Loans granted versus purpose.
purpose_loan<-LC_Data%>%group_by(purpose)%>%summarise(n=n(),mean_loan=mean(loan_amnt)
)%>%mutate(freq=n/sum(n)*100)
setnames(purpose_loan, old = c('purpose','n'), new = c('Purpose','totalCount'))
print(purpose_loan)
```

```
## # A tibble: 13 × 4
##    Purpose            totalCount mean_loan   freq
##    <chr>                   <int>     <dbl>  <dbl>
##  1 car                       928     7955.  0.928
##  2 credit_card             24989    13660. 25.0
##  3 debt_consolidation      57622    13228. 57.6
##  4 home_improvement         5654    11911.  5.65
##  5 house                     354    12757.  0.354
##  6 major_purchase           1823     9948.  1.82
##  7 medical                  1119     7313.  1.12
##  8 moving                    691     6882.  0.691
##  9 other                    5091     8305.  5.09
## 10 renewable_energy           58     8807.  0.058
## 11 small_business            893    13603.  0.893
## 12 vacation                  678     5674.  0.678
## 13 wedding                   100     9124.  0.1
```

```
#Loan status versus purpose.
purpose_loan_status<-LC_Data%>%group_by(purpose,loan_status)%>%summarise(n=n(),mean_l
oan=mean(loan_amnt))%>%mutate(freq=n/sum(n)*100)
print(purpose_loan_status)
```

```
## # A tibble: 26 × 5
## # Groups:   purpose [13]
##    purpose            loan_status     n mean_loan  freq
##    <chr>              <chr>       <int>     <dbl> <dbl>
##  1 car                Charged Off   107     8591.  11.5
##  2 car                Fully Paid    821     7872.  88.5
##  3 credit_card        Charged Off  2865    12867.  11.5
##  4 credit_card        Fully Paid  22124    13763.  88.5
##  5 debt_consolidation Charged Off  8319    12769.  14.4
##  6 debt_consolidation Fully Paid  49303    13305.  85.6
##  7 home_improvement   Charged Off   682    12387.  12.1
##  8 home_improvement   Fully Paid   4972    11846.  87.9
##  9 house              Charged Off    63    11686.  17.8
## 10 house              Fully Paid    291    12988.  82.2
## # … with 16 more rows
```

```
ggplot(data=purpose_loan_status, aes(x=purpose, y=mean_loan, group=1)) +
  geom_line(linetype = "dashed")+
  geom_point()+labs(y="Average Loan Amount", x = "Purpose")
```




```
#Loan grade versus purpose.
purpose_loan_grade<-LC_Data%>%group_by(purpose,grade)%>%summarise(n=n(),mean_loan=mea
n(loan_amnt))%>%mutate(freq=n/sum(n)*100)
print(purpose_loan_grade)
```

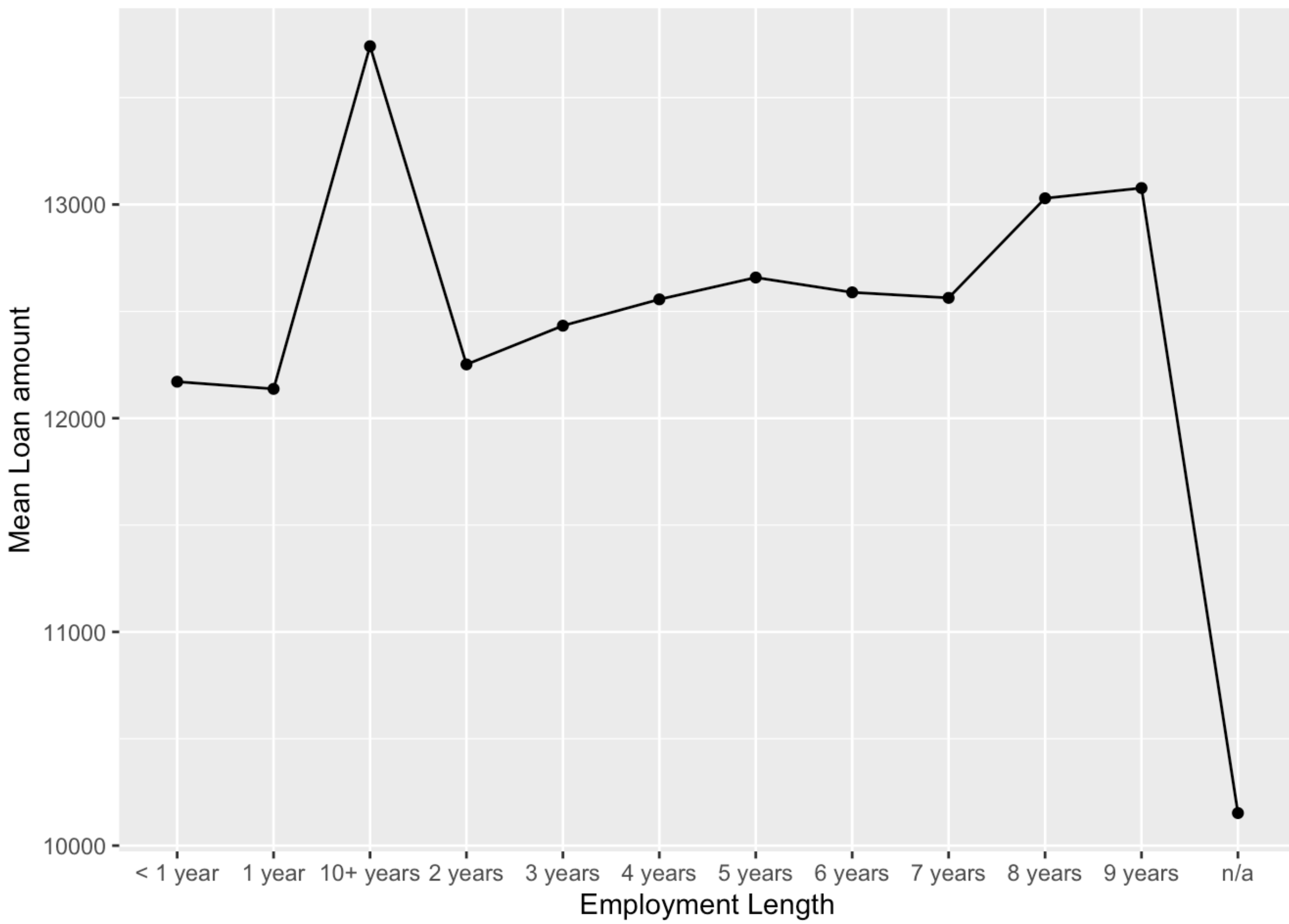
```
## # A tibble: 87 × 5
## # Groups:   purpose [13]
##    purpose     grade     n mean_loan   freq
##    <chr>       <chr> <int>     <dbl>  <dbl>
##  1 car         A       253     8591. 27.3
##  2 car         B       306     7691. 33.0
##  3 car         C       238     7476. 25.6
##  4 car         D        92     8910.  9.91
##  5 car         E        27     7587.  2.91
##  6 car         F         8     4272.  0.862
##  7 car         G         4     4325   0.431
##  8 credit_card A      8349    14972. 33.4
##  9 credit_card B      9809    13006. 39.3
## 10 credit_card C      5008    13150. 20.0
## # … with 77 more rows
```

Question 2-a-vi

```r
#Employment period versus Mean Loan amount:
emply_loanamt<-LC_Data%>%group_by(emp_length)%>%summarise(n=n(),mean_loan=mean(loan_a
mnt))
print(emply_loanamt)
```

```
## # A tibble: 12 × 3
##    emp_length     n mean_loan
##    <chr>      <int>     <dbl>
##  1 < 1 year    8104    12171.
##  2 1 year      6649    12137.
##  3 10+ years  31394    13741.
##  4 2 years     8987    12252.
##  5 3 years     8046    12433.
##  6 4 years     5892    12556.
##  7 5 years     6046    12658.
##  8 6 years     4712    12589.
##  9 7 years     5124    12563.
## 10 8 years     4990    13029.
## 11 9 years     3908    13077.
## 12 n/a         6148    10152.
```

```r
ggplot(data=emply_loanamt, aes(x=emp_length, y=mean_loan, group=1)) +
  geom_line()+geom_point() + labs(y="Mean Loan amount", x = "Employment Length")
```




```
#Employment length versus grade:
emply_grade<-LC_Data%>%group_by(emp_length,grade)%>%summarise(n=n(),mean_loan=mean(lo
an_amnt))
print(emply_grade)
```

```
## # A tibble: 84 × 4
## # Groups:   emp_length [12]
##    emp_length grade     n mean_loan
##    <chr>      <chr> <int>     <dbl>
##  1 < 1 year   A      1786    14020.
##  2 < 1 year   B      2664    12233.
##  3 < 1 year   C      2164    11344.
##  4 < 1 year   D      1076    11319.
##  5 < 1 year   E       342    10787.
##  6 < 1 year   F        60     8145.
##  7 < 1 year   G        12     8094.
##  8 1 year     A      1395    14305.
##  9 1 year     B      2229    11981.
## 10 1 year     C      1846    11465.
## # … with 74 more rows
```

```r
#Employment length versus purpose
emply_purpose<-LC_Data%>%group_by(emp_length,purpose)%>%summarise(n=n(),mean_loan=mea
n(loan_amnt))
print(emply_purpose)
```

```
## # A tibble: 156 × 4
## # Groups:   emp_length [12]
##    emp_length purpose                n mean_loan
##    <chr>      <chr>              <int>     <dbl>
##  1 < 1 year   car                  104     9246.
##  2 < 1 year   credit_card         2260    12890.
##  3 < 1 year   debt_consolidation  4489    12721.
##  4 < 1 year   home_improvement     302    11799.
##  5 < 1 year   house                 41    12498.
##  6 < 1 year   major_purchase       149     8653.
##  7 < 1 year   medical               87     8391.
##  8 < 1 year   moving               148     7046.
##  9 < 1 year   other                422     7782.
## 10 < 1 year   renewable_energy       6     9496.
## # … with 146 more rows
```

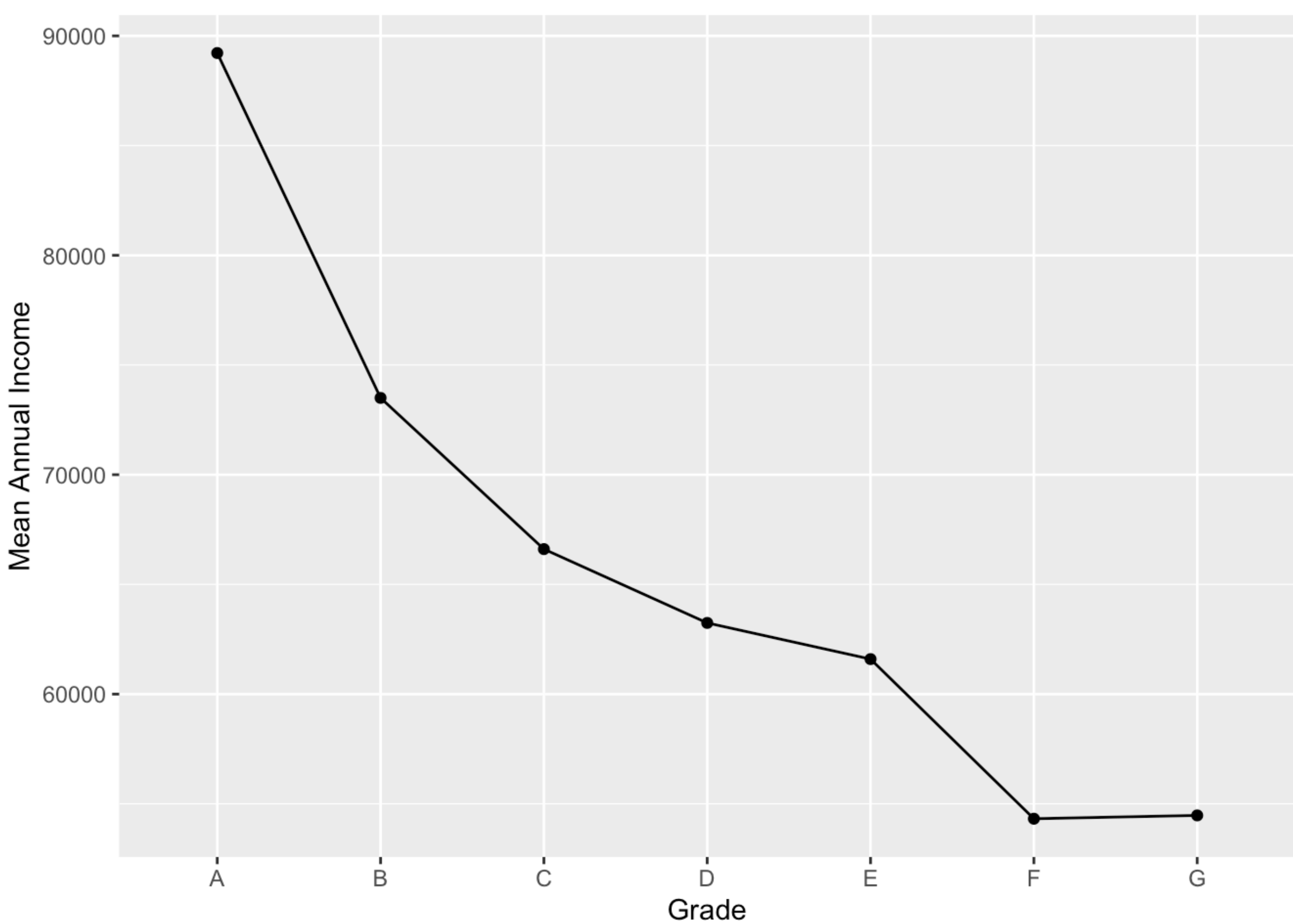
```r
#Annual income versus purpose
annInc_pur<-LC_Data%>%group_by(purpose)%>%summarise(n=n(),mean_anninc=mean(annual_inc
))
print(annInc_pur)
```

```
## # A tibble: 13 × 3
##    purpose                n mean_anninc
##    <chr>              <int>       <dbl>
##  1 car                  928      72182.
##  2 credit_card        24989      74014.
##  3 debt_consolidation 57622      72071.
##  4 home_improvement    5654      87438.
##  5 house                354      78575.
##  6 major_purchase      1823      73380.
##  7 medical             1119      76724.
##  8 moving               691      63991.
##  9 other               5091      67347.
## 10 renewable_energy      58      69656.
## 11 small_business       893      88868.
## 12 vacation             678      70037.
## 13 wedding              100      64378.
```

```
#Annual income versus Grade:
annInc_grade<-LC_Data%>%group_by(grade)%>%summarise(n=n(),mean_anninc=mean(annual_inc
))
print(annInc_grade)
```

```
## # A tibble: 7 × 3
##   grade     n mean_anninc
##   <chr> <int>       <dbl>
## 1 A     22588      89218.
## 2 B     33907      73500.
## 3 C     26645      66609.
## 4 D     12493      63245.
## 5 E      3579      61596.
## 6 F       708      54321.
## 7 G        80      54473.
```

```
ggplot(data=annInc_grade, aes(x=grade, y=mean_anninc, group=1)) +
  geom_line()+geom_point() + labs(y="Mean Annual Income", x = "Grade")
```




```
dim(LC_Data)
```

```
## [1] 100000    149
```

Question 2-a-vii

```
#Generate some (at least 3) new derived attributes which you think may be useful for
predicting default., and explain what these are. For these, do an analyses as in the
questions above (as reasonable based on the derived variables).

#Derived attribute-1: proportion of satisfactory bankcard accounts

LC_Data$satisBankcardAccts_prop <- ifelse(LC_Data$num_bc_tl>0, LC_Data$num_bc_sats/LC
_Data$num_bc_tl, 0)
head(LC_Data$satisBankcardAccts_prop)
```

```
## [1] 0.9090909 0.3333333 0.6666667 0.3333333 0.5000000 0.1818182
```

```
#Derived Attribute-2: length of borrower's history with LC

LC_Data$earliest_cr_line<-paste(LC_Data$earliest_cr_line, "-01", sep = "")
LC_Data$earliest_cr_line<-parse_date_time(LC_Data$earliest_cr_line, "myd")
LC_Data$borrHistory <- as.duration(LC_Data$earliest_cr_line %--% LC_Data$issue_d) /dy
ears(1)
head(LC_Data$borrHistory)
```

```
## [1] 20.413415 27.247091 26.250513 30.078029 14.748802  8.999316
```

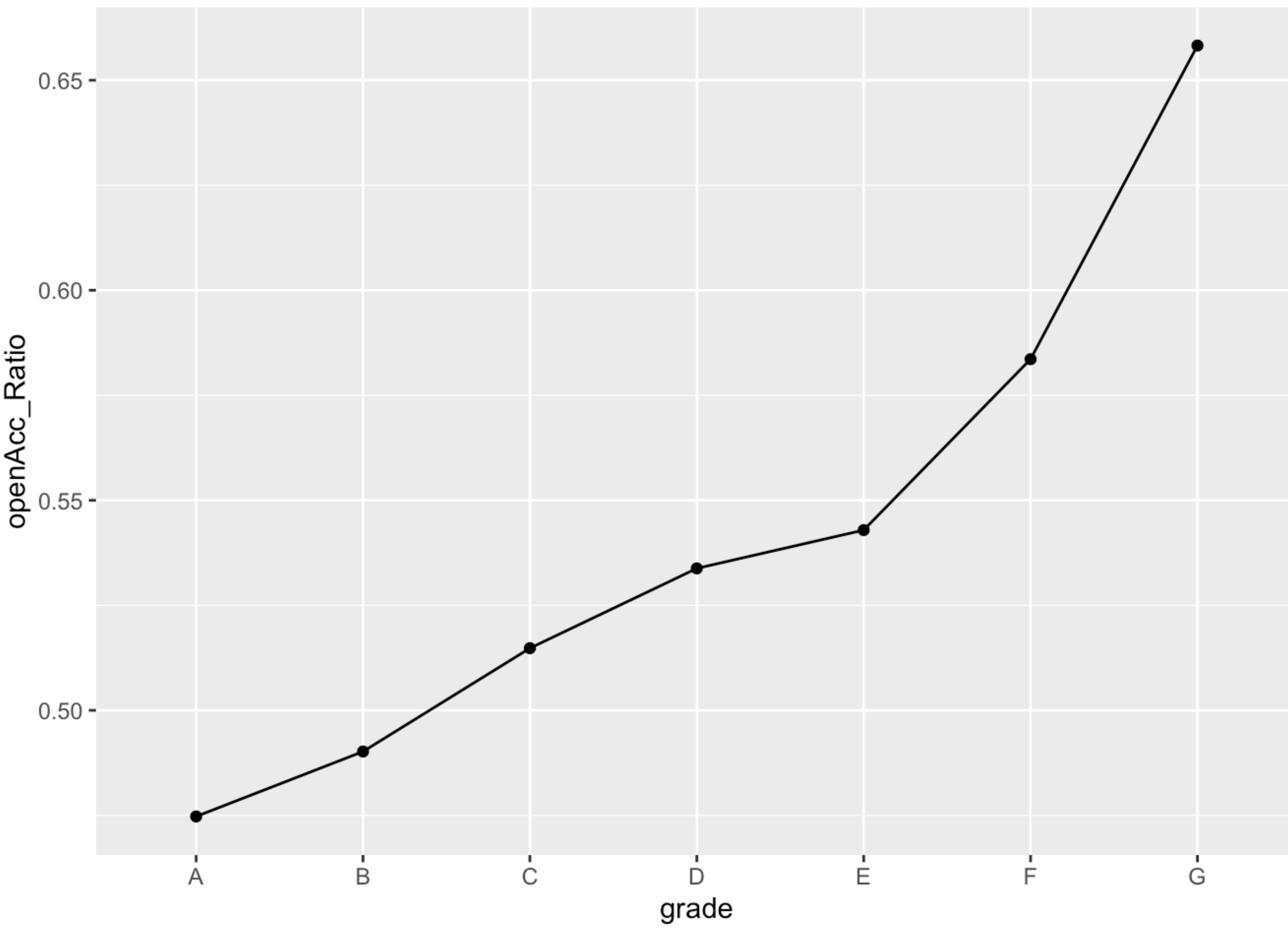
```
#Derived attribute-3: ratio of open Accounts to total Accounts
LC_Data$openAccRatio <- ifelse(LC_Data$total_acc>0, LC_Data$open_acc/LC_Data$total_ac
c, 0)
head(LC_Data$openAccRatio)
```

```
## [1] 0.5161290 0.4285714 0.3888889 0.3846154 0.3000000 0.3636364
```

```
#Summary with line plot for openAccRatio with Grade
openAcc_Grade <- LC_Data %>% group_by(grade) %>% summarise(openAcc_Ratio=mean(openAcc
Ratio))
print(openAcc_Grade)
```

```
## # A tibble: 7 × 2
##   grade openAcc_Ratio
##   <chr>         <dbl>
## 1 A             0.475
## 2 B             0.490
## 3 C             0.515
## 4 D             0.534
## 5 E             0.543
## 6 F             0.584
## 7 G             0.658
```

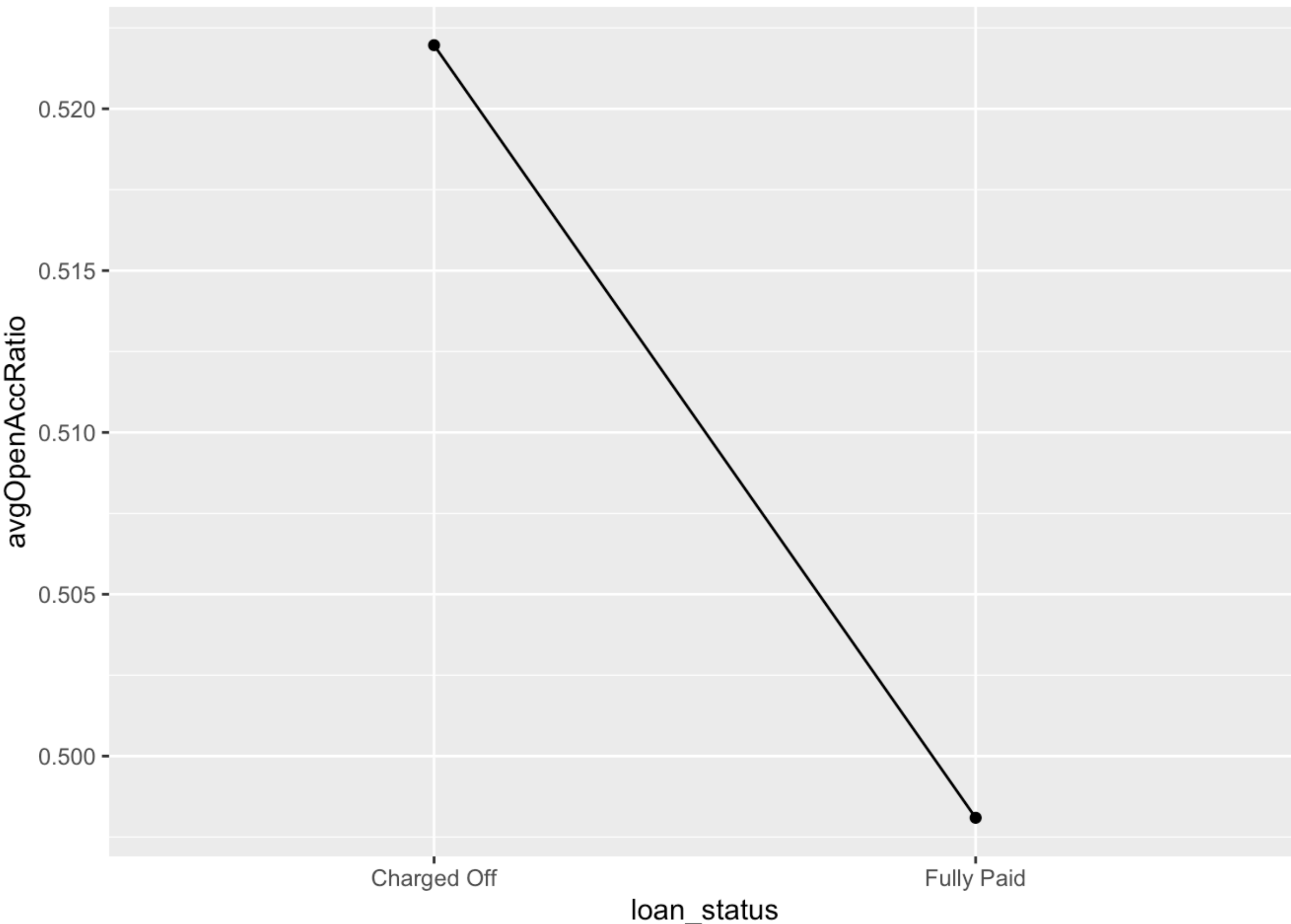
```
ggplot(openAcc_Grade, aes(x=grade, y=openAcc_Ratio,group=1)) + geom_line() + geom_poi
nt()
```




```
#Summary with line plot for openAccRatio with loan status.
openAcc_loanstat<-LC_Data %>% group_by(loan_status) %>% summarise(avgOpenAccRatio=mea
n(openAccRatio))
print(openAcc_loanstat)
```

```
## # A tibble: 2 × 2
##   loan_status avgOpenAccRatio
##   <chr>                 <dbl>
## 1 Charged Off           0.522
## 2 Fully Paid            0.498
```

```
ggplot(openAcc_loanstat, aes(x=loan_status, y=avgOpenAccRatio,group=1)) + geom_line()
+ geom_point()
```




```
#Derived attribute-4: Balance amount to pay

LC_Data$balance_to_pay <- LC_Data$funded_amnt - LC_Data$total_pymnt
glimpse(LC_Data$balance_to_pay)
```

```
##  num [1:100000] -1437 -1161 -1772 -861 -1009 ...
```

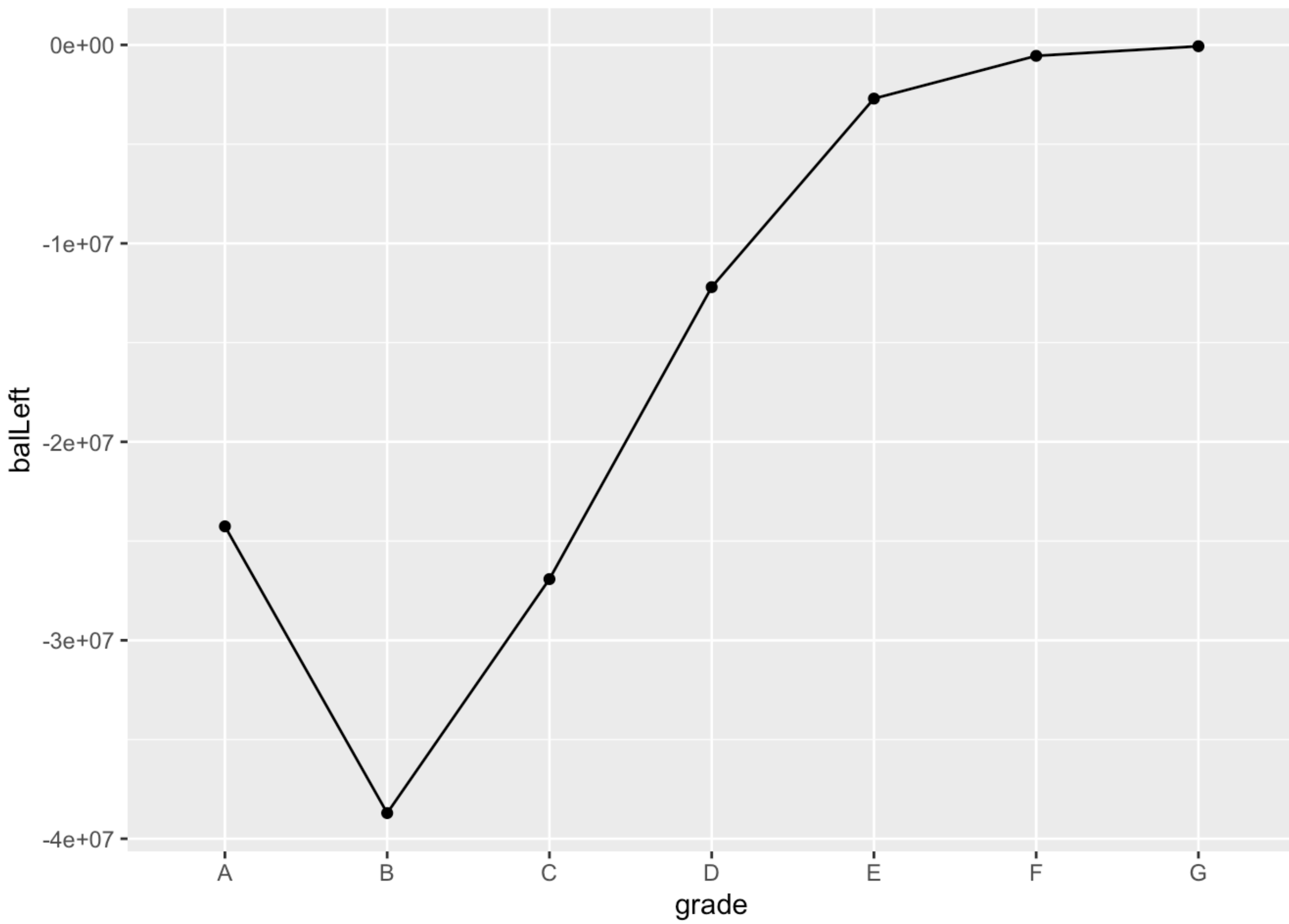
```
bal_to_paygrade<-LC_Data%>%group_by(grade)%>%summarise(balLeft=sum(balance_to_pay))
print(bal_to_paygrade)
```

```
## # A tibble: 7 × 2
##   grade    balLeft
##   <chr>      <dbl>
## 1 A     -24258884.
## 2 B     -38706016.
## 3 C     -26916360.
## 4 D     -12204367.
## 5 E      -2704059.
## 6 F       -550575.
## 7 G        -65546.
```

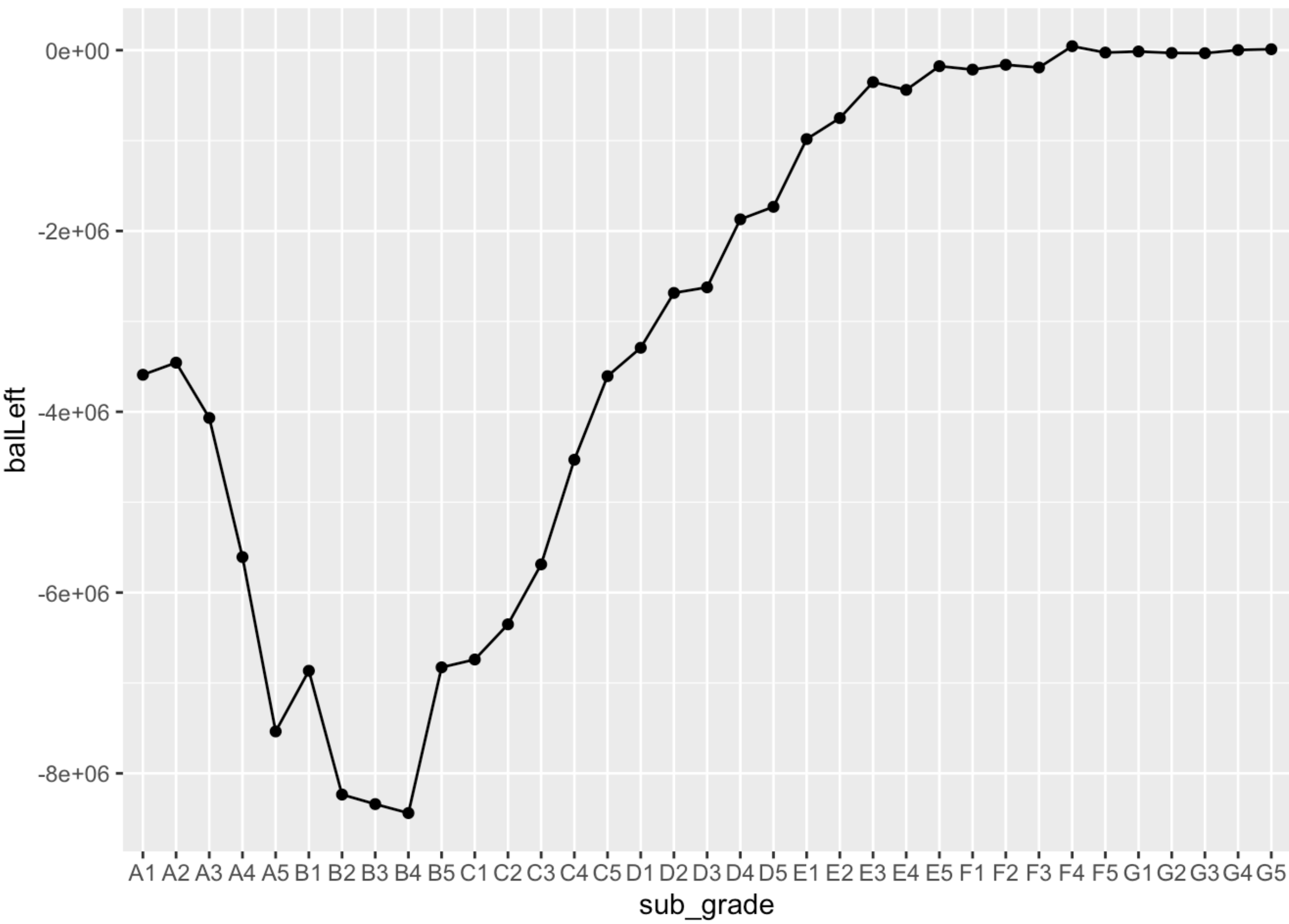
```
bal_to_paysubgrade<-LC_Data%>%group_by(sub_grade)%>%summarise(balLeft=sum(balance_to_
pay))
print(bal_to_paysubgrade)
```

```
## # A tibble: 35 × 2
##    sub_grade   balLeft
##    <chr>         <dbl>
##  1 A1        -3590584.
##  2 A2        -3456706.
##  3 A3        -4068151.
##  4 A4        -5607195.
##  5 A5        -7536248.
##  6 B1        -6865049.
##  7 B2        -8234440.
##  8 B3        -8340132.
##  9 B4        -8439203.
## 10 B5        -6827192.
## # … with 25 more rows
```

```
#Line plot for Balance left by grades.
ggplot(bal_to_paygrade, aes(x=grade, y=balLeft,group=1)) + geom_line() + geom_point()
```



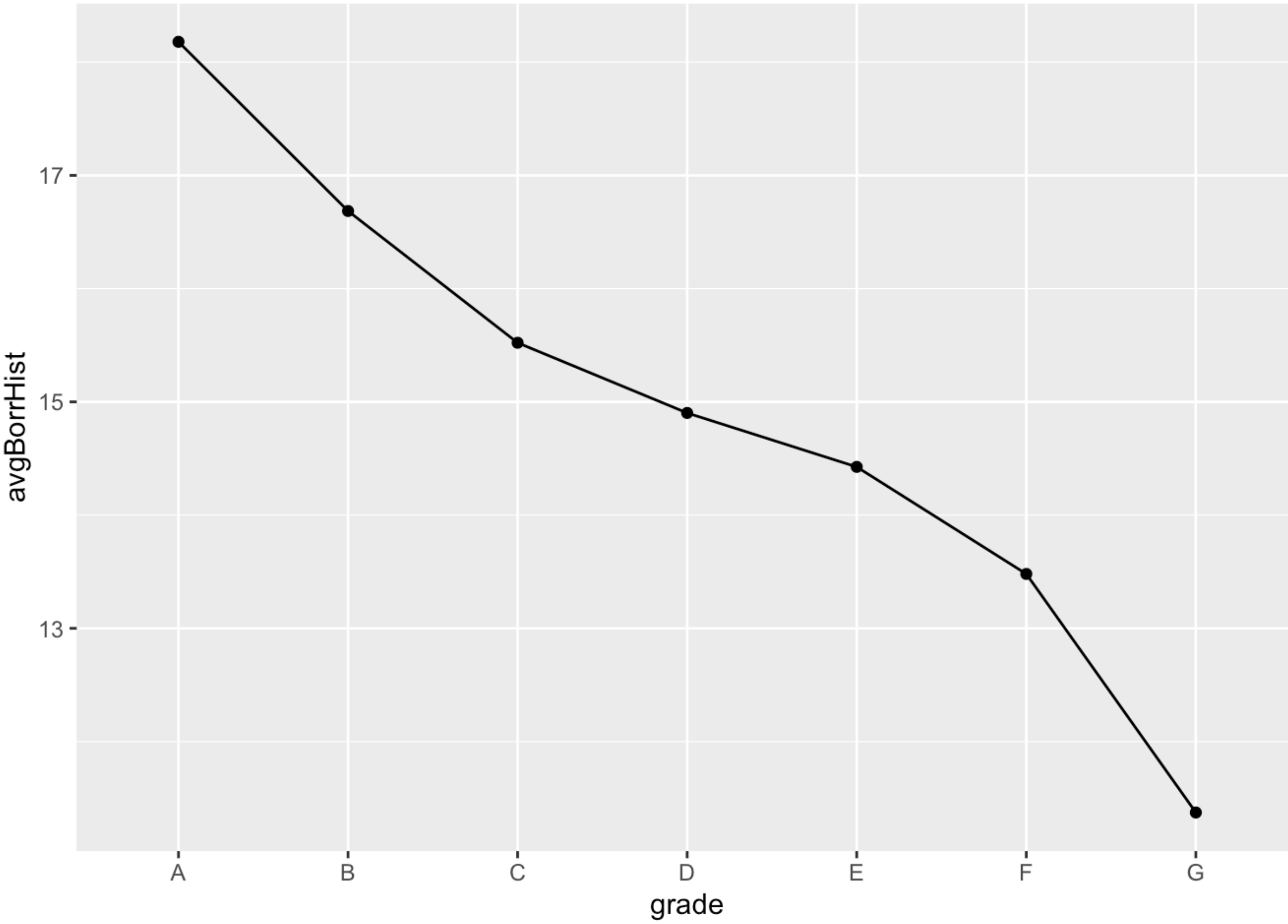

```
#Line plot for Balance left by subgrades.
ggplot(bal_to_paysubgrade, aes(x=sub_grade, y=balLeft,group=1)) + geom_
point()
```




```
# negative values indicate most of the loans are paid off and with some interest rate
. Thta's why total payment exceeds the funded amount
#for positive values the loan is charged off


#LC assigned Grade variation by borrow History
loan_v_borrow <- LC_Data %>% group_by(grade) %>% summarise(avgBorrHist=mean(borrHisto
ry))
loan_v_borrow
```

```
## # A tibble: 7 × 2
##   grade avgBorrHist
##   <chr>       <dbl>
## 1 A            18.2
## 2 B            16.7
## 3 C            15.5
## 4 D            14.9
## 5 E            14.4
## 6 F            13.5
## 7 G            11.4
```

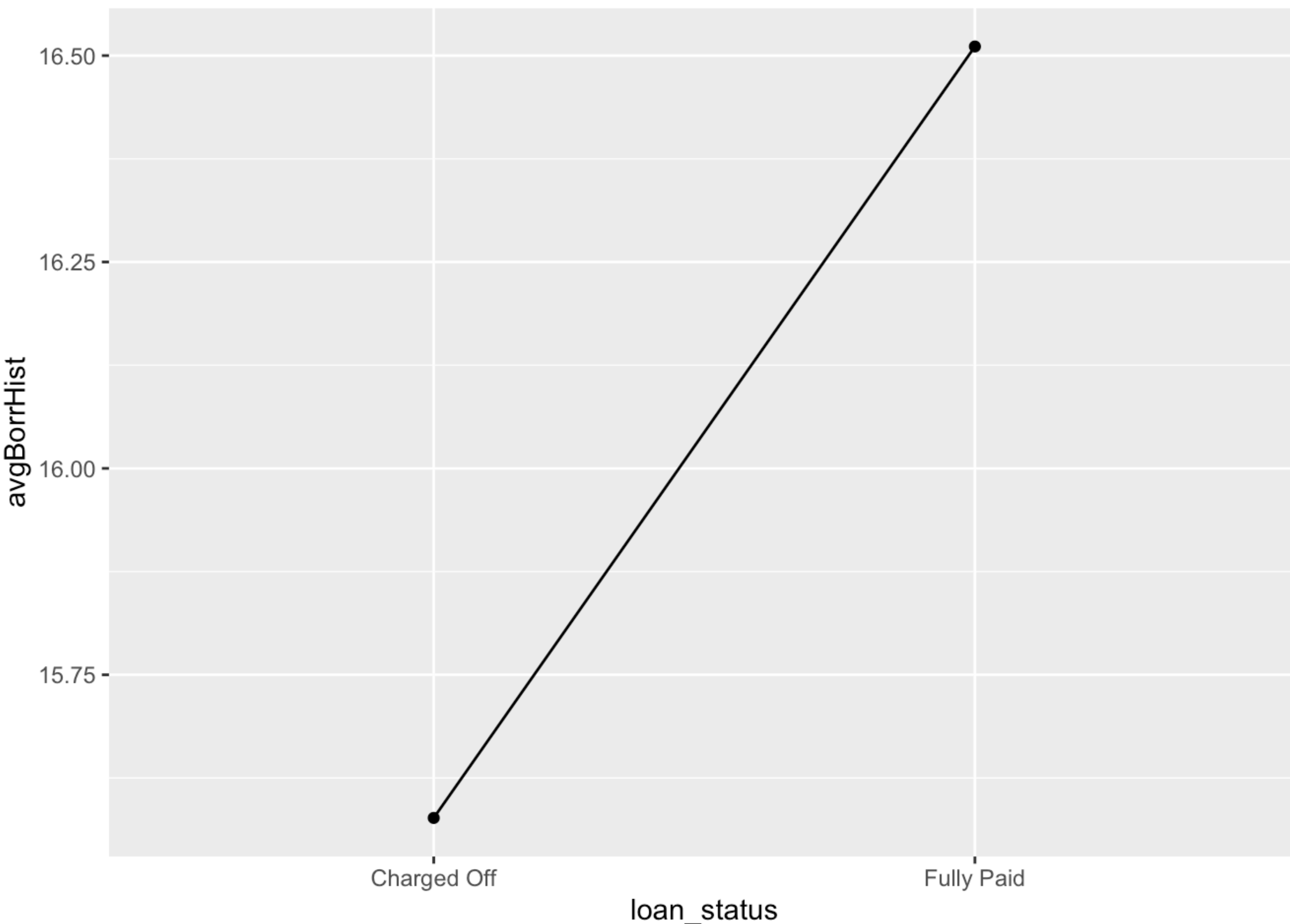
```
#plot to understand variation between borrow history and grade
ggplot(loan_v_borrow, aes(x=grade, y=avgBorrHist,group=1)) + geom_line() + geom_point
()
```




```
#Summary with line plot for mean borrHistory with Loan status.


borrHis_grade<-LC_Data %>% group_by(loan_status) %>% summarise(avgBorrHist=mean(borrH
istory))
print(borrHis_grade)
```

```
## # A tibble: 2 × 2
##   loan_status avgBorrHist
##   <chr>             <dbl>
## 1 Charged Off        15.6
## 2 Fully Paid         16.5
```

```
ggplot(borrHis_grade, aes(x=loan_status, y=avgBorrHist,group=1)) + geom_line() + geom
_point()
```




```
LC_Data %>% group_by(grade) %>%summarise(avgSatisBankCard_prop=mean(satisBankcardAccts_prop))
```

```
## # A tibble: 7 × 2
##   grade avgSatisBankCard_prop
##   <chr>                 <dbl>
## 1 A                     0.596
## 2 B                     0.602
## 3 C                     0.633
## 4 D                     0.645
## 5 E                     0.642
## 6 F                     0.663
## 7 G                     0.745
```

```
dim(LC_Data)
```

```
## [1] 100000    153
```

Question 2-a-vii-b

```r
#(c) Are there missing values? What is the proportion of missing values in different variables?

dim(LC_Data)
```

```
## [1] 100000    153
```

```r
#Drop col's with all empty values into new data frame -lcdf
lcdf <- LC_Data %>% select_if(function(x){!all(is.na(x))})

dim(lcdf)
```

```
## [1] 100000    116
```

```r
#columns where there are missing values
colMeans(is.na(lcdf))[colMeans(is.na(lcdf))>0]
```

```
##                      emp_title                          title
##                        0.06705                        0.00012
##          mths_since_last_delinq         mths_since_last_record
##                        0.49919                        0.82423
##                     revol_util                    last_pymnt_d
##                        0.00041                        0.00064
##              last_credit_pull_d    mths_since_last_major_derog
##                        0.00004                        0.71995
##                    open_acc_6m                    open_act_il
##                        0.97313                        0.97313
##                    open_il_12m                    open_il_24m
##                        0.97313                        0.97313
##              mths_since_rcnt_il                   total_bal_il
##                        0.97393                        0.97313
##                        il_util                    open_rv_12m
##                        0.97694                        0.97313
##                    open_rv_24m                     max_bal_bc
##                        0.97313                        0.97313
##                       all_util                         inq_fi
##                        0.97313                        0.97313
##                    total_cu_tl                   inq_last_12m
##                        0.97313                        0.97313
##                    avg_cur_bal                 bc_open_to_buy
##                        0.00002                        0.00964
##                        bc_util              mo_sin_old_il_acct
##                        0.01044                        0.03620
##            mths_since_recent_bc        mths_since_recent_bc_dlq
##                        0.00911                        0.74329
##           mths_since_recent_inq mths_since_recent_revol_delinq
##                        0.10612                        0.64746
##                  num_rev_accts               num_tl_120dpd_2m
##                        0.00001                        0.03824
##                 pct_tl_nvr_dlq               percent_bc_gt_75
##                        0.00016                        0.01034
##                   hardship_dpd                 settlement_term
##                        0.99955                        0.99535
```

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

```
## [1] 100000     96
```

```
#Check where the missing values are present
names(colMeans(is.na(lcdf))[colMeans(is.na(lcdf))>0])
```

```
##  [1] "emp_title"               "title"                   "mths_since_last_delinq"
##  [4] "revol_util"              "last_pymnt_d"            "last_credit_pull_d"
##  [7] "avg_cur_bal"             "bc_open_to_buy"          "bc_util"
## [10] "mo_sin_old_il_acct"      "mths_since_recent_bc"    "mths_since_recent_inq"
## [13] "num_rev_accts"           "num_tl_120dpd_2m"        "pct_tl_nvr_dlq"
## [16] "percent_bc_gt_75"
```

```
#variable imputation

lcdf<- lcdf %>% replace_na(list(bc_open_to_buy=median(lcdf$bc_open_to_buy, na.rm=TRUE
), num_tl_120dpd_2m = median(lcdf$num_tl_120dpd_2m, na.rm=TRUE),percent_bc_gt_75 = me
dian(lcdf$percent_bc_gt_75, na.rm=TRUE), bc_util=median(lcdf$bc_util, na.rm=TRUE) ))


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

```
##  [1] "emp_title"               "title"                   "mths_since_last_delinq"
##  [4] "revol_util"              "last_pymnt_d"            "last_credit_pull_d"
##  [7] "avg_cur_bal"             "mo_sin_old_il_acct"      "mths_since_recent_bc"
## [10] "mths_since_recent_inq"   "num_rev_accts"           "pct_tl_nvr_dlq"
```

```
dim(lcdf)
```

```
## [1] 100000     96
```

Question 2-a-vii-c

```
#3. Consider the potential for data leakage. You do not want to include variables in
your model which may not be available when applying the model; that is, some data may
not be available for new loans before they are funded. Leakage may also arise from va
riables in the data which may have been updated during the loan period (ie., after th
e loan is funded). Identify and explain which variables will you exclude from the mod
el.

# new data after considering for leakage
new_data <- lcdf %>% select(-c(funded_amnt_inv, term, emp_title, pymnt_plan, title, z
ip_code, addr_state, out_prncp, out_prncp_inv, total_pymnt_inv, total_rec_prncp, tota
l_rec_int,total_rec_late_fee,recoveries, collection_recovery_fee, last_credit_pull_d,
policy_code, disbursement_method, debt_settlement_flag, hardship_flag, application_ty
pe))

#removing additional variables which are not present in the
new_data <- new_data %>% select(-c(last_pymnt_d, last_pymnt_amnt))

dim(new_data)
```

```
## [1] 100000     73
```

```
names(colMeans(is.na(new_data))[colMeans(is.na(new_data))>0])
```

```
## [1] "mths_since_last_delinq" "revol_util"             "avg_cur_bal"
## [4] "mo_sin_old_il_acct"     "mths_since_recent_bc"   "mths_since_recent_inq"
## [7] "num_rev_accts"          "pct_tl_nvr_dlq"
```

```
new_data <- new_data %>% select(-c(return,returnperyear))
```

Question 3

```
#Do a univariate analyses to determine which variables (from amongst those you decide
to consider for the next stage prediction task) will be #individually useful for pred
icting the dependent variable (loan_status). For this, you need a measure of relation
ship between the dependent #variable and each of the potential predictor variables. G
iven loan-status as a binary dependent variable, which measure will you use? From you
r #analyses using this measure, which variables do you think will be useful for predi
cting loan_status? (Note – if certain variables on their own #are highly predictive o
f the outcome, it is good to ask if this variable has a leakage issue).


library(pROC) #importing the package which has AUC(..) function
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
#Using sapply function to apply AUC curve on the variables
#considered both numeric and factor variables.
# we need numeric variables to calculate the area under the curve

head(new_data$earliest_cr_line)
```

```
## [1] "1994-12-01 UTC" "1988-04-01 UTC" "1988-08-01 UTC" "1984-02-01 UTC"
## [5] "2000-07-01 UTC" "2005-01-01 UTC"
```

```r
new_data$earliest_cr_line <- as.Date(new_data$earliest_cr_line)
new_data$issue_d <- as.Date(new_data$issue_d)
new_data <- mutate_if(new_data, is.character, as.factor)


#dropping the loan status variable
ds_train <- new_data %>% select(-c(loan_status))

aucAll<- sapply(ds_train %>% mutate_if(is.factor, as.numeric) %>% select_if(is.numeri
c), auc, response = new_data$loan_status)
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls < cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls < cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls > cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls < cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls > cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls > cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls < cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls > cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls < cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls < cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls < cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls > cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls < cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls > cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls < cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls > cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls < cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls < cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls > cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls < cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls < cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls < cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls < cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls < cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls < cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls < cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls < cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls > cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls < cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls < cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls > cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls < cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls < cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls < cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls < cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls < cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls < cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls < cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls < cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls < cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls < cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls < cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls < cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls < cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls < cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls < cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls > cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls < cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls < cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls > cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls < cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls < cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls < cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls < cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls < cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls < cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls < cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls < cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls < cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls < cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls < cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls < cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls > cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls < cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls > cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls < cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls > cases
```

```
## Setting levels: control = Charged Off, case = Fully Paid
```

```
## Setting direction: controls > cases
```

```
#To determine which variables have AUC > 0.5
length(aucAll[aucAll>0.5])
```

```
## [1] 47
```

```r
selected_col<-names(aucAll[aucAll>0.5])

selected_col <- append(selected_col,"loan_status")

# adding the loan status variable
new_data <- new_data %>% select((selected_col))
```

```
## Note: Using an external vector in selections is ambiguous.
## ℹ Use `all_of(selected_col)` instead of `selected_col` to silence this message.
## ℹ See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.
```

```r
library(broom)

#view a table output
tidy(aucAll[aucAll > 0.5]) %>% view()
```

```
## Warning: 'tidy.numeric' is deprecated.
## See help("Deprecated")
```

```r
#arranging auc curve values in descending order
tidy(aucAll) %>% arrange(desc(aucAll))
```

```
## Warning: 'tidy.numeric' is deprecated.
## See help("Deprecated")
```

```
## # A tibble: 68 × 2
##    names                    x
##    <chr>                <dbl>
##  1 annRet               0.986
##  2 balance_to_pay       0.963
##  3 total_pymnt          0.756
##  4 sub_grade            0.666
##  5 actualTerm           0.664
##  6 int_rate             0.658
##  7 grade                0.654
##  8 acc_open_past_24mths 0.583
##  9 annual_inc           0.577
## 10 bc_open_to_buy       0.574
## # … with 58 more rows
```

Question 4-a

```r
new_dt=new_data

glimpse(new_data)
```

```
## Rows: 100,000
## Columns: 48
## $ loan_amnt                  <dbl> 28000, 6150, 7200, 4750, 5000, 9600, 1600, …
## $ funded_amnt                <dbl> 28000, 6150, 7200, 4750, 5000, 9600, 1600, …
## $ int_rate                   <dbl> 5.32, 13.33, 14.99, 15.31, 12.69, 14.98, 13…
## $ installment                <dbl> 843.22, 208.20, 249.56, 165.39, 167.73, 332…
## $ grade                      <fct> A, C, C, C, C, C, C, B, C, C, C, A, B, C, A…
## $ sub_grade                  <fct> A1, C3, C5, C4, C2, C3, C3, B2, C2, C2, C1,…
## $ home_ownership             <fct> MORTGAGE, RENT, RENT, MORTGAGE, MORTGAGE, M…
## $ annual_inc                 <dbl> 140000, 40000, 20000, 30000, 27000, 48500, …
## $ dti                        <dbl> 12.83, 22.19, 23.44, 3.40, 12.27, 21.70, 11…
## $ inq_last_6mths             <dbl> 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 2…
## $ mths_since_last_delinq     <dbl> NA, 43, 31, NA, NA, NA, NA, 15, 13, NA, 57,…
## $ open_acc                   <dbl> 16, 6, 7, 5, 9, 12, 7, 11, 14, 6, 12, 16, 2…
## $ revol_bal                  <dbl> 74178, 428, 11907, 2797, 8345, 14076, 7434,…
## $ revol_util                 <dbl> 41.5, 17.1, 55.6, 20.0, 71.0, 66.7, 67.0, 5…
## $ total_acc                  <dbl> 31, 14, 18, 13, 30, 33, 9, 35, 35, 13, 35, …
## $ initial_list_status        <fct> w, f, f, f, f, w, w, f, f, f, f, f, f, w, w…
## $ total_pymnt                <dbl> 29436.890, 7311.160, 8971.949, 5611.070, 60…
## $ tot_cur_bal                <dbl> 364466, 21157, 11907, 72539, 163964, 213998…
## $ total_rev_hi_lim           <dbl> 136800, 2500, 21400, 13750, 11800, 21100, 1…
## $ acc_open_past_24mths       <dbl> 3, 2, 1, 3, 8, 7, 5, 5, 4, 7, 6, 4, 5, 0, 2…
## $ avg_cur_bal                <dbl> 26033, 3526, 1701, 18134, 20495, 21400, 237…
## $ bc_open_to_buy             <dbl> 54787, 409, 2329, 500, 2500, 132, 0, 7772, …
## $ bc_util                    <dbl> 43.2, 41.6, 82.9, 76.0, 54.0, 99.0, 100.7, …
## $ mo_sin_old_il_acct         <dbl> 194, 127, NA, 128, 156, 107, 31, 146, 195, …
## $ mo_sin_old_rev_tl_op       <dbl> 245, 326, 315, 361, 177, 108, 97, 175, 229,…
## $ mo_sin_rcnt_rev_tl_op      <dbl> 16, 9, 39, 13, 4, 2, 1, 14, 8, 11, 9, 3, 1,…
## $ mo_sin_rcnt_tl             <dbl> 16, 9, 20, 13, 4, 2, 1, 4, 8, 11, 9, 3, 1, …
## $ mort_acc                   <dbl> 7, 1, 3, 2, 3, 3, 0, 7, 1, 1, 1, 0, 1, 3, 0…
## $ mths_since_recent_bc       <dbl> 16, 25, 47, 43, 13, 8, 45, 14, 34, 11, 9, 3…
## $ mths_since_recent_inq      <dbl> 15, 9, NA, 13, 7, 7, 16, 5, 8, 19, 4, 3, 4,…
## $ num_bc_tl                  <dbl> 11, 6, 6, 3, 6, 11, 3, 11, 8, 6, 8, 12, 17,…
## $ num_il_tl                  <dbl> 7, 4, 0, 3, 4, 9, 3, 15, 15, 4, 22, 20, 12,…
## $ num_op_rev_tl              <dbl> 14, 4, 7, 3, 6, 10, 5, 8, 11, 4, 5, 9, 16, …
## $ num_rev_accts              <dbl> 17, 9, 15, 8, 23, 21, 6, 13, 19, 8, 11, 21,…
## $ num_sats                   <dbl> 16, 6, 7, 4, 8, 12, 7, 11, 14, 6, 12, 16, 2…
## $ num_tl_120dpd_2m           <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…
## $ pct_tl_nvr_dlq             <dbl> 100.0, 85.7, 94.4, 54.0, 63.0, 100.0, 100.0…
## $ tot_hi_cred_lim            <dbl> 457375, 33161, 21400, 85964, 171142, 226976…
## $ total_bal_ex_mort          <dbl> 94529, 21157, 11907, 2797, 8345, 41089, 166…
## $ total_bc_limit             <dbl> 94900, 700, 13600, 500, 2500, 13000, 5600, …
## $ total_il_high_credit_limit <dbl> 33575, 30661, 0, 0, 0, 31911, 17800, 37191,…
## $ actualTerm                 <dbl> 1.1690623, 1.9192334, 3.0006845, 1.4182067,…
## $ annRet                     <dbl> 4.389629, 9.837598, 8.201598, 12.782191, 7.…
```

```
## $ satisBankcardAccts_prop    <dbl> 0.9090909, 0.3333333, 0.6666667, 0.3333333,…
## $ borrHistory                <dbl> 20.413415, 27.247091, 26.250513, 30.078029,…
## $ openAccRatio               <dbl> 0.5161290, 0.4285714, 0.3888889, 0.3846154,…
## $ balance_to_pay             <dbl> -1436.8900, -1161.1598, -1771.9493, -861.07…
## $ loan_status                <fct> Fully Paid, Fully Paid, Fully Paid, Fully P…
```

```
##pre-preprocessing data steps

#removing variables like actualTerm, actualRetrun

# excluding certain elements from the dataset because of data leakage issue.
new_data2=new_data
new_data1 <- new_data%>%select(-c(annRet,total_pymnt, balance_to_pay))

new_data1 <- new_data1%>%select(-c(grade))

new_data1 <- new_data1%>%select(-c(actualTerm,funded_amnt))

names(colMeans(is.na(new_data1)))[colMeans(is.na(new_data1))>0]
```

```
## [1] "mths_since_last_delinq" "revol_util"             "avg_cur_bal"
## [4] "mo_sin_old_il_acct"     "mths_since_recent_bc"   "mths_since_recent_inq"
## [7] "num_rev_accts"          "pct_tl_nvr_dlq"
```

```
#replacing some of the missing NA values in the columns by median values

new_data1<- new_data1 %>% replace_na(list(mths_since_last_delinq=median(new_data1$mth
s_since_last_delinq, na.rm=TRUE),
                                     revol_util = median(new_data1$revol_util, na.rm=
TRUE),
                                     avg_cur_bal = median(new_data1$avg_cur_bal, na.r
m=TRUE),
                                     mths_since_recent_bc = median(new_data1$mths_sin
ce_recent_bc, na.rm=TRUE),
                                     mths_since_recent_inq = median(new_data1$mths_si
nce_recent_inq, na.rm=TRUE),
                                     num_rev_accts = median(new_data1$num_rev_accts,
na.rm=TRUE),
                                     pct_tl_nvr_dlq = median(new_data1$pct_tl_nvr_dlq
, na.rm=TRUE),
                                     mo_sin_old_il_acct=median(new_data1$mo_sin_old_i
l_acct, na.rm=TRUE) ))

names(colMeans(is.na(new_data1)))[colMeans(is.na(new_data1))>0]
```

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

```r
new_data1$loan_status <- factor(new_data1$loan_status)#, levels=c("Fully Paid", "Char
ged Off"))

dim(new_data1)
```

```
## [1] 100000     42
```

```r
library(rpart)
library(rpart.plot)
library(ranger)


#Splitting data into 70% training and 30%  testing ratio.

PROP = 0.7  #proportion of examples in the training sample
nr<-nrow(new_data1)
trnIndex<- sample(1:nr, size = round(PROP * nr), replace=FALSE)

final_dataTrn <- new_data1[trnIndex, ]
final_dataTst <- new_data1[-trnIndex, ]

names(new_data1)
```

```
##  [1] "loan_amnt"                  "int_rate"
##  [3] "installment"                "sub_grade"
##  [5] "home_ownership"             "annual_inc"
##  [7] "dti"                        "inq_last_6mths"
##  [9] "mths_since_last_delinq"     "open_acc"
## [11] "revol_bal"                  "revol_util"
## [13] "total_acc"                  "initial_list_status"
## [15] "tot_cur_bal"                "total_rev_hi_lim"
## [17] "acc_open_past_24mths"       "avg_cur_bal"
## [19] "bc_open_to_buy"             "bc_util"
## [21] "mo_sin_old_il_acct"         "mo_sin_old_rev_tl_op"
## [23] "mo_sin_rcnt_rev_tl_op"      "mo_sin_rcnt_tl"
## [25] "mort_acc"                   "mths_since_recent_bc"
## [27] "mths_since_recent_inq"      "num_bc_tl"
## [29] "num_il_tl"                  "num_op_rev_tl"
## [31] "num_rev_accts"              "num_sats"
## [33] "num_tl_120dpd_2m"           "pct_tl_nvr_dlq"
## [35] "tot_hi_cred_lim"            "total_bal_ex_mort"
## [37] "total_bc_limit"             "total_il_high_credit_limit"
## [39] "satisBankcardAccts_prop"    "borrHistory"
## [41] "openAccRatio"               "loan_status"
```

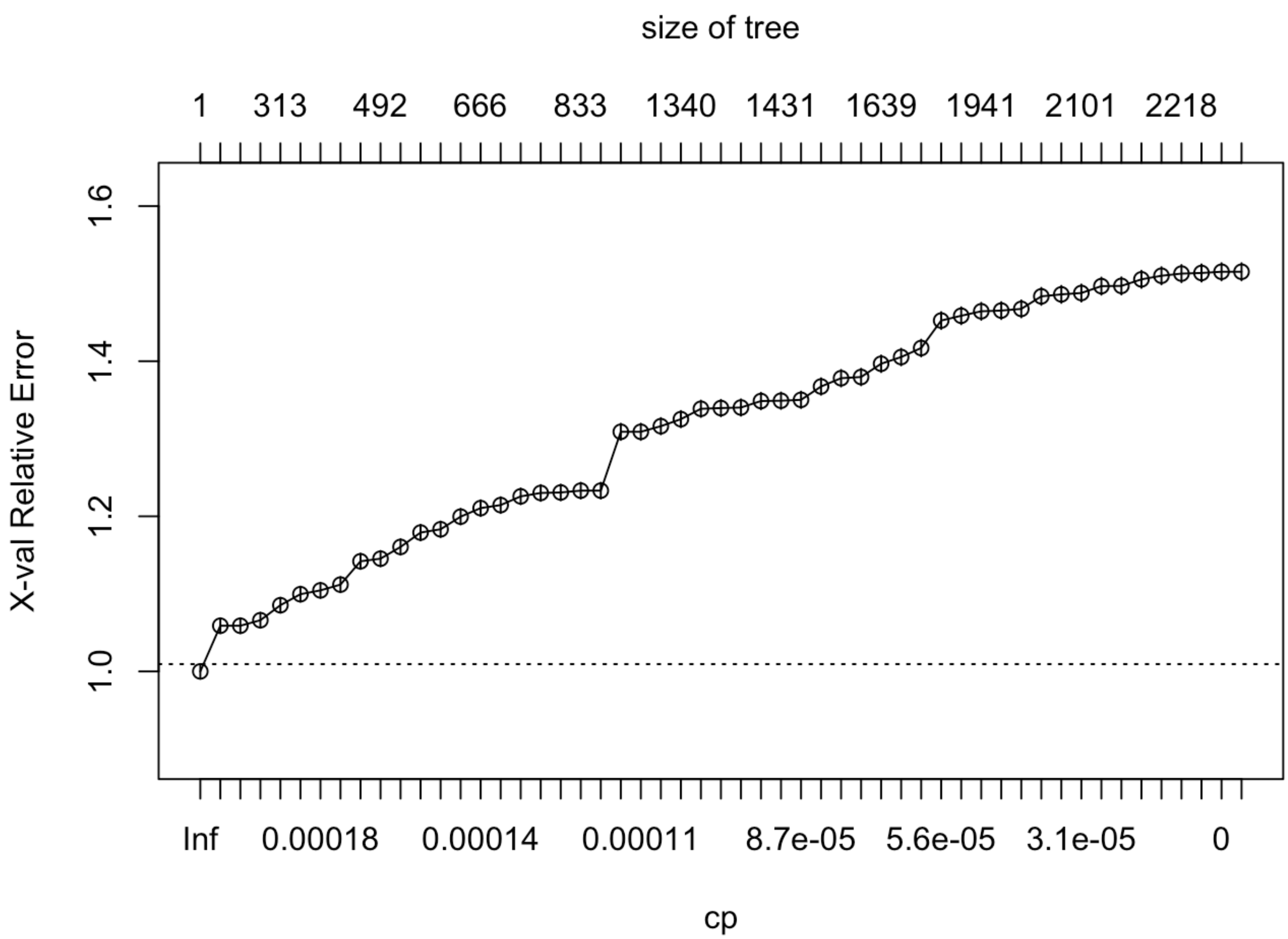
Question 5

```
set.seed(673)

lcDT1 <- rpart(loan_status ~., data=final_dataTrn, method="class", parms = list(split
= "information"), control = rpart.control(cp=-1))
printcp(lcDT1)
```

```
##
## Classification tree:
## rpart(formula = loan_status ~ ., data = final_dataTrn, method = "class",
##     parms = list(split = "information"), control = rpart.control(cp = -1))
##
## Variables actually used in tree construction:
##  [1] acc_open_past_24mths       annual_inc
##  [3] avg_cur_bal                bc_open_to_buy
##  [5] bc_util                    borrHistory
##  [7] dti                        home_ownership
##  [9] initial_list_status        inq_last_6mths
## [11] installment                int_rate
## [13] loan_amnt                  mo_sin_old_il_acct
## [15] mo_sin_old_rev_tl_op       mo_sin_rcnt_rev_tl_op
## [17] mo_sin_rcnt_tl             mort_acc
## [19] mths_since_last_delinq     mths_since_recent_bc
## [21] mths_since_recent_inq      num_bc_tl
## [23] num_il_tl                  num_op_rev_tl
## [25] num_rev_accts              num_sats
## [27] open_acc                   openAccRatio
## [29] pct_tl_nvr_dlq             revol_bal
## [31] revol_util                 satisBankcardAccts_prop
## [33] sub_grade                  tot_cur_bal
## [35] tot_hi_cred_lim            total_acc
## [37] total_bal_ex_mort          total_bc_limit
## [39] total_il_high_credit_limit total_rev_hi_lim
##
## Root node error: 9712/70000 = 0.13874
##
## n= 70000
##
##             CP nsplit rel error xerror      xstd
## 1   2.2309e-04      0   1.00000 1.0000 0.0094170
## 2   2.2064e-04    220   0.92164 1.0589 0.0096443
## 3   2.1880e-04    233   0.91742 1.0589 0.0096443
## 4   2.0593e-04    248   0.91320 1.0660 0.0096710
## 5   1.9306e-04    312   0.89734 1.0854 0.0097430
## 6   1.8534e-04    329   0.89322 1.0996 0.0097952
## 7   1.8019e-04    352   0.88828 1.1045 0.0098132
## 8   1.7161e-04    364   0.88612 1.1119 0.0098401
## 9   1.6732e-04    399   0.87891 1.1420 0.0099476
## 10  1.6474e-04    491   0.85358 1.1454 0.0099596
## 11  1.5445e-04    514   0.84874 1.1605 0.0100127
```

```
## 12  1.4977e-04    612   0.82877 1.1791 0.0100768
## 13  1.4415e-04    632   0.82444 1.1833 0.0100913
## 14  1.3729e-04    648   0.82187 1.1995 0.0101467
## 15  1.3386e-04    665   0.81909 1.2106 0.0101839
## 16  1.2871e-04    695   0.81291 1.2145 0.0101970
## 17  1.2356e-04    713   0.81054 1.2257 0.0102344
## 18  1.2169e-04    753   0.80385 1.2301 0.0102491
## 19  1.2013e-04    806   0.79500 1.2307 0.0102511
## 20  1.1584e-04    832   0.79088 1.2331 0.0102589
## 21  1.1326e-04    869   0.78449 1.2331 0.0102589
## 22  1.1155e-04    912   0.77790 1.3090 0.0105025
## 23  1.0787e-04    944   0.77358 1.3090 0.0105025
## 24  1.0297e-04    974   0.76987 1.3162 0.0105250
## 25  9.6530e-05   1339   0.71973 1.3253 0.0105530
## 26  9.3605e-05   1355   0.71818 1.3387 0.0105941
## 27  9.2669e-05   1375   0.71602 1.3396 0.0105969
## 28  9.1525e-05   1401   0.71304 1.3403 0.0105991
## 29  9.0095e-05   1414   0.71170 1.3486 0.0106245
## 30  8.8256e-05   1430   0.71026 1.3491 0.0106258
## 31  8.5805e-05   1485   0.70140 1.3502 0.0106292
## 32  8.2372e-05   1501   0.69944 1.3674 0.0106809
## 33  8.0084e-05   1546   0.69409 1.3781 0.0107128
## 34  7.7224e-05   1580   0.69028 1.3796 0.0107174
## 35  7.2076e-05   1638   0.68523 1.3967 0.0107678
## 36  6.8644e-05   1652   0.68421 1.4054 0.0107930
## 37  6.1779e-05   1742   0.67566 1.4170 0.0108267
## 38  5.1483e-05   1772   0.67288 1.4524 0.0109277
## 39  4.5762e-05   1931   0.66330 1.4586 0.0109450
## 40  4.5047e-05   1940   0.66289 1.4644 0.0109611
## 41  4.1186e-05   1967   0.66155 1.4653 0.0109637
## 42  3.7442e-05   2000   0.65970 1.4675 0.0109697
## 43  3.4322e-05   2020   0.65888 1.4836 0.0110144
## 44  3.1682e-05   2082   0.65661 1.4862 0.0110215
## 45  2.9419e-05   2100   0.65548 1.4880 0.0110263
## 46  2.5741e-05   2107   0.65527 1.4968 0.0110505
## 47  2.0593e-05   2163   0.65383 1.4971 0.0110513
## 48  1.8721e-05   2183   0.65342 1.5056 0.0110743
## 49  1.7161e-05   2194   0.65321 1.5102 0.0110868
## 50  1.4709e-05   2217   0.65280 1.5130 0.0110943
## 51  1.2871e-05   2244   0.65239 1.5138 0.0110965
## 52  0.0000e+00   2252   0.65229 1.5153 0.0111006
## 53 -1.0000e+00   3410   0.65229 1.5153 0.0111006
```

```
plotcp(lcDT1)
```



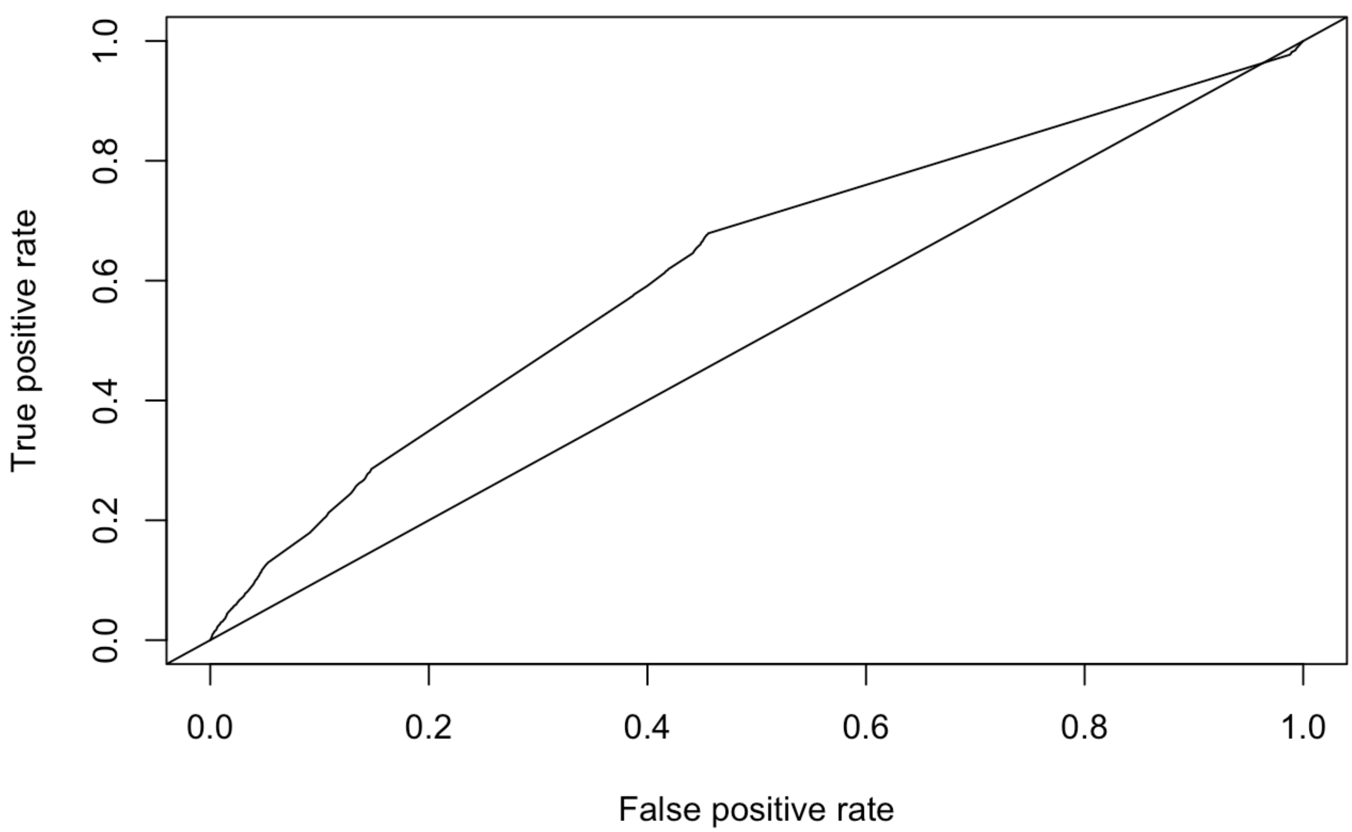

```r
library(ROCR)
library(caret)
```

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

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

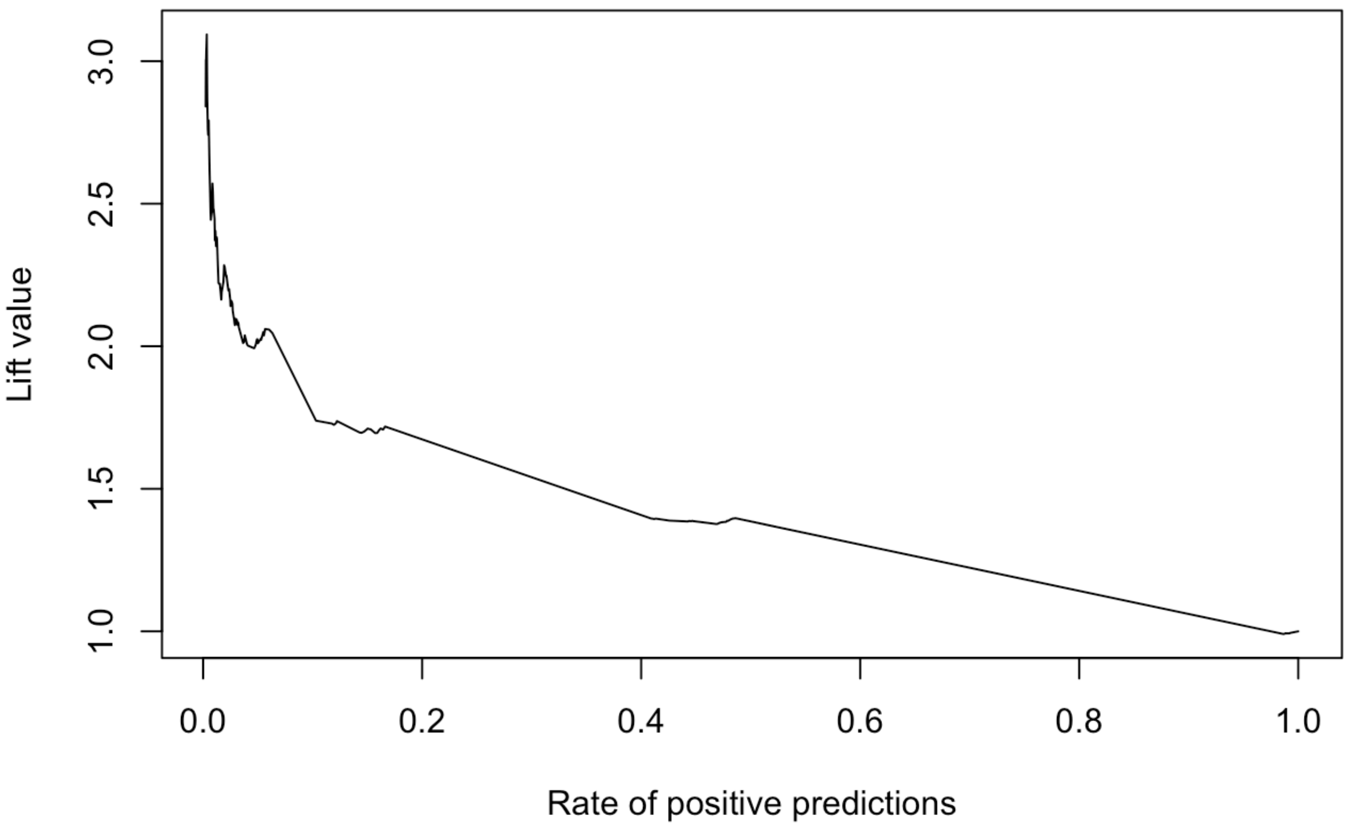
```
#model 1
lcDT1_1 <- rpart(loan_status ~., data=final_dataTrn, method="class", parms = list(spl
it = "information"), control = rpart.control(cp=0.00019279))


#ROC plot
score=predict(lcDT1_1,final_dataTst, type="prob")[,"Charged Off"]
pred=prediction(score, final_dataTst$loan_status, label.ordering = c("Fully Paid", "C
harged Off"))
    #label.ordering here specifies the 'negative', 'positive' class labels

#ROC curve
aucPerf <-performance(pred, "tpr", "fpr")
plot(aucPerf)
abline(a=0, b= 1)
```




```
#AUC value
aucPerf=performance(pred, "auc")
aucPerf@y.values
```

```
## [[1]]
## [1] 0.6208069
```

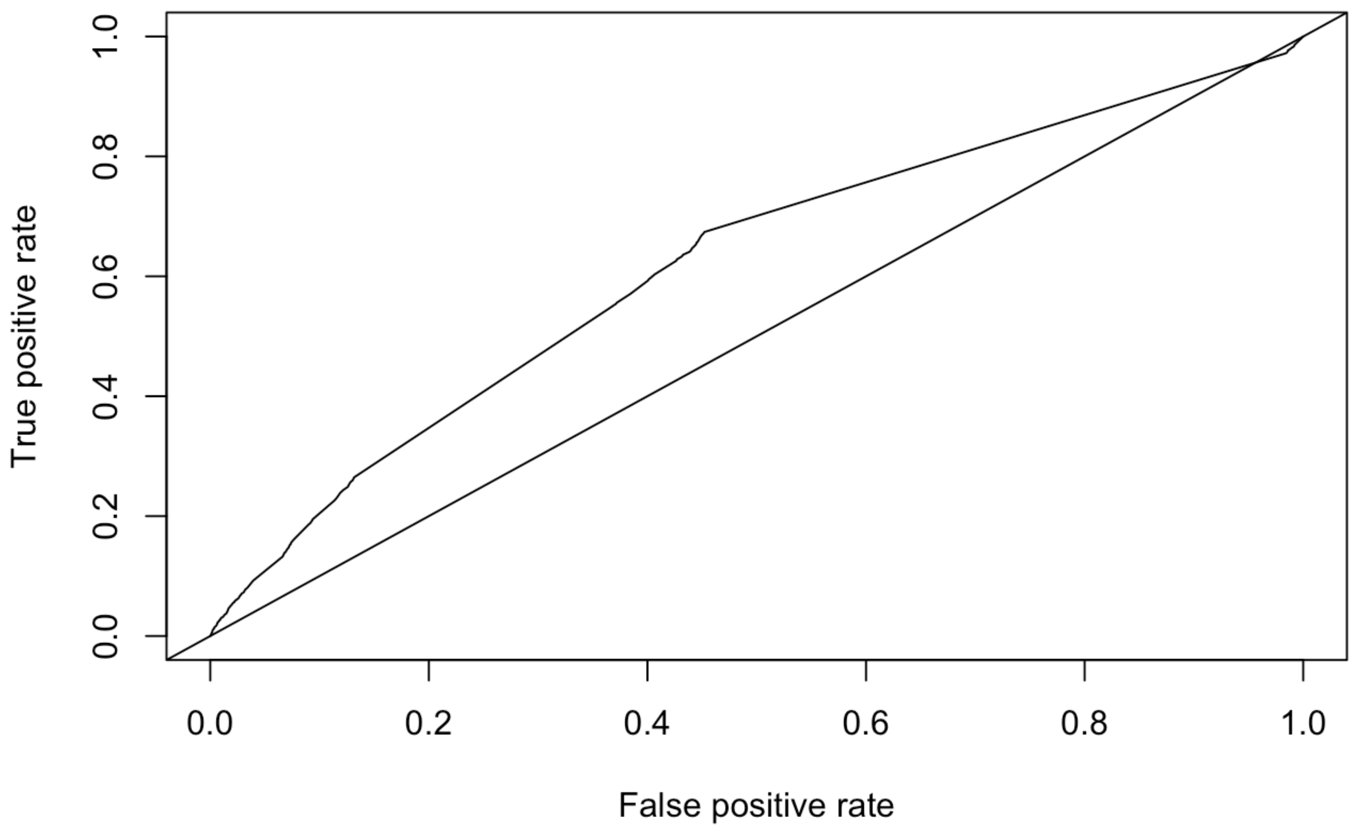
```
#Lift curve
liftPerf <-performance(pred, "lift", "rpp")
plot(liftPerf)
```




```
test_preds = predict(lcDT1_1,final_dataTst, type="prob")
thrsh = 0.5
test_preds <- ifelse(test_preds[,1] > thrsh, "Charged Off", "Fully Paid")
confusionMatrix(factor(test_preds,levels=c('Charged Off','Fully Paid')),final_dataTst
$loan_status,positive = "Charged Off")
```

```
## Confusion Matrix and Statistics
##
##               Reference
## Prediction    Charged Off Fully Paid
##   Charged Off         231        559
##   Fully Paid         3842      25368
##
##                Accuracy : 0.8533
##                  95% CI : (0.8492, 0.8573)
##     No Information Rate : 0.8642
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0532
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.05671
##             Specificity : 0.97844
##          Pos Pred Value : 0.29241
##          Neg Pred Value : 0.86847
##              Prevalence : 0.13577
##          Detection Rate : 0.00770
##    Detection Prevalence : 0.02633
##       Balanced Accuracy : 0.51758
##
##        'Positive' Class : Charged Off
##
```

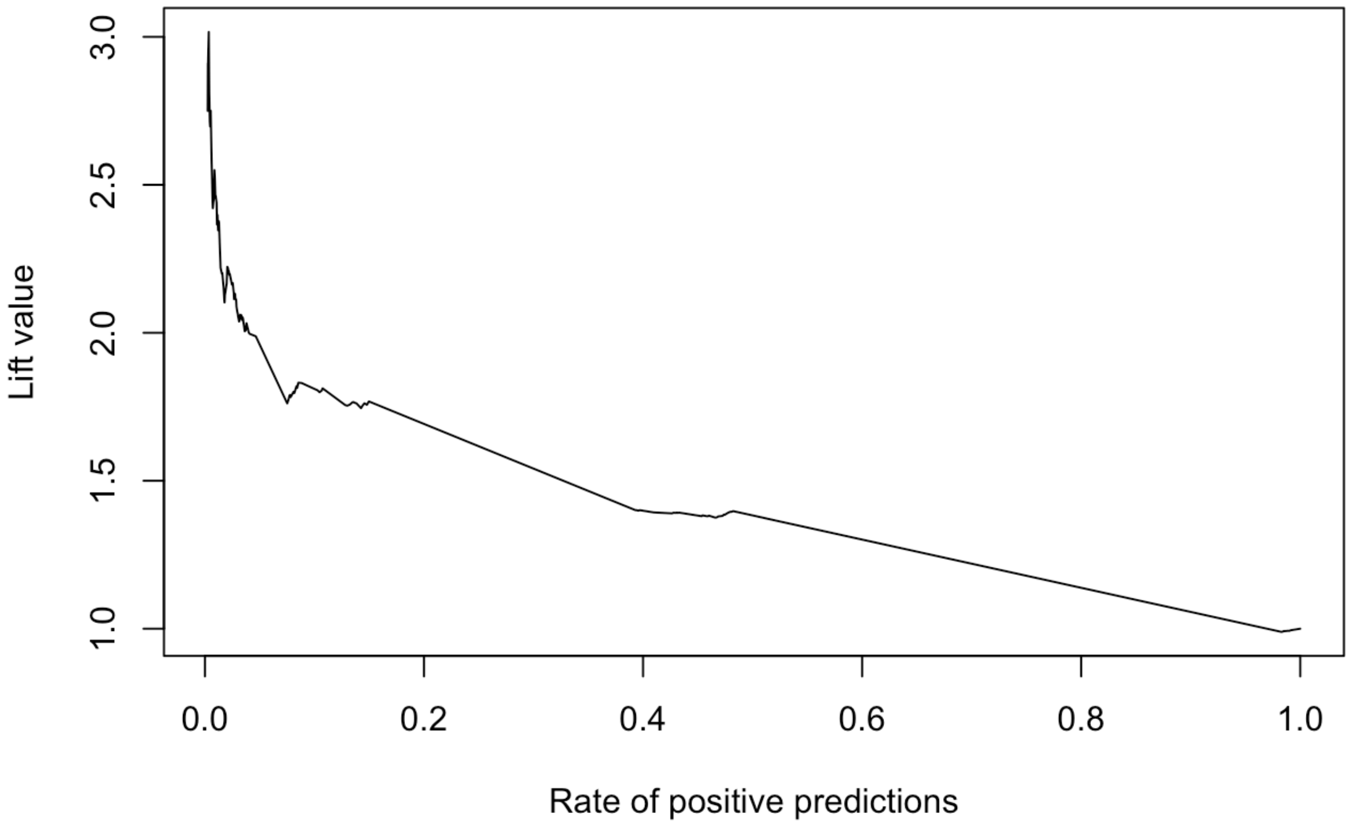
```
##model 2

lcDT1_2 <- rpart(loan_status ~., data=final_dataTrn, method="class", parms = list(spl
it = "information"), control = rpart.control(cp= 0.00017302))


#ROC plot
score=predict(lcDT1_2,final_dataTst, type="prob")[,"Charged Off"]
pred=prediction(score, final_dataTst$loan_status, label.ordering = c("Fully Paid", "C
harged Off"))
    #label.ordering here specifies the 'negative', 'positive' class labels

#ROC curve
aucPerf <-performance(pred, "tpr", "fpr")
plot(aucPerf)
abline(a=0, b= 1)
```




```
#AUC value
aucPerf=performance(pred, "auc")
aucPerf@y.values
```

```
## [[1]]
## [1] 0.6189293
```

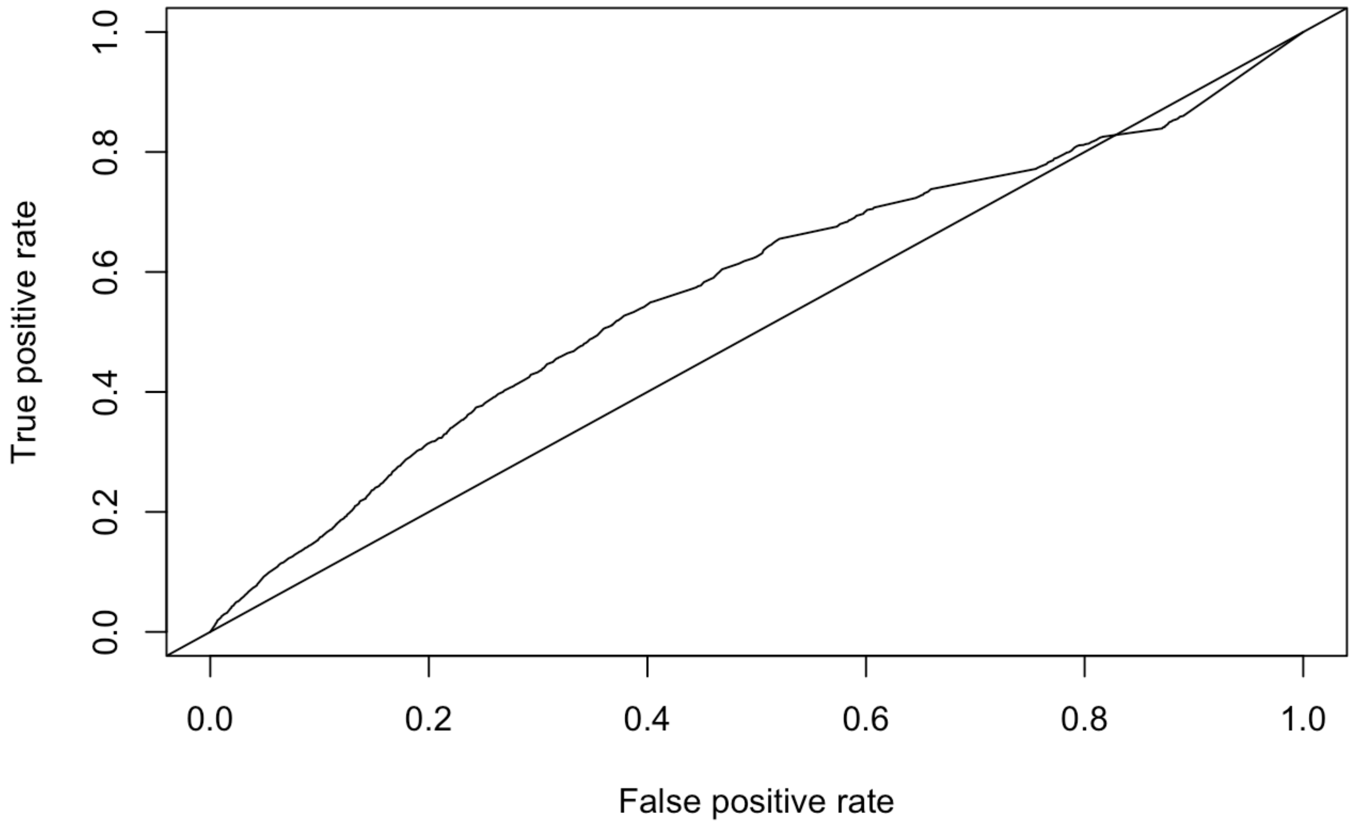
```
#Lift curve
liftPerf <-performance(pred, "lift", "rpp")
plot(liftPerf)
```




```
test_preds = predict(lcDT1_2,final_dataTst, type="prob")
thrsh = 0.5
test_preds <- ifelse(test_preds[,1] > thrsh, "Charged Off", "Fully Paid")
confusionMatrix(factor(test_preds,levels=c('Charged Off','Fully Paid')),final_dataTst
$loan_status,positive = "Charged Off")
```

```
## Confusion Matrix and Statistics
##
##               Reference
## Prediction    Charged Off Fully Paid
##   Charged Off         242        601
##   Fully Paid         3831      25326
##
##                Accuracy : 0.8523
##                  95% CI : (0.8482, 0.8563)
##     No Information Rate : 0.8642
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0544
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.059416
##             Specificity : 0.976820
##          Pos Pred Value : 0.287070
##          Neg Pred Value : 0.868608
##              Prevalence : 0.135767
##          Detection Rate : 0.008067
##    Detection Prevalence : 0.028100
##       Balanced Accuracy : 0.518118
##
##        'Positive' Class : Charged Off
##
```

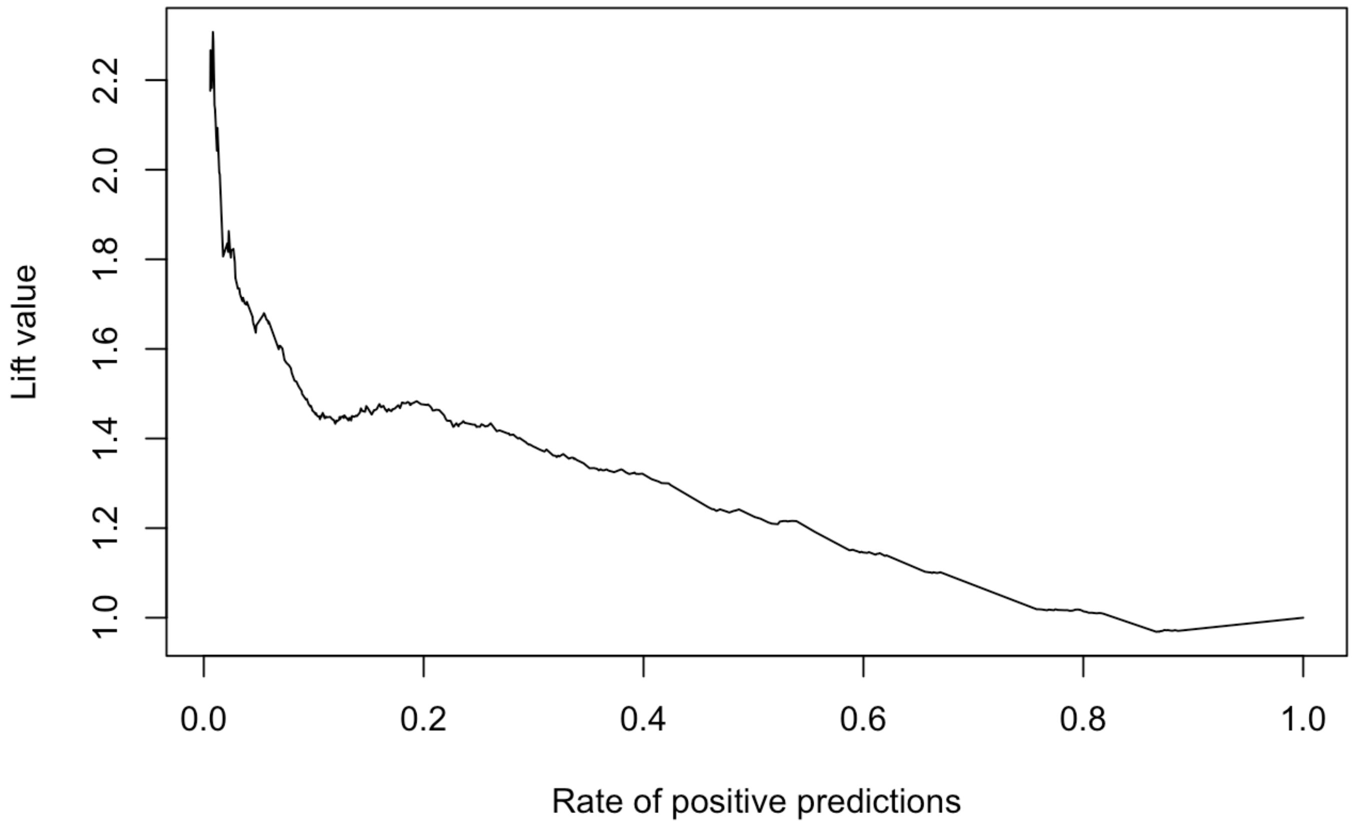
```
##model 3

lcDT1_3 <- rpart(loan_status ~., data=final_dataTrn, method="class", parms = list(spl
it = "information"), control = rpart.control(cp= 0.000057672))


#ROC plot
score=predict(lcDT1_3,final_dataTst, type="prob")[,"Charged Off"]
pred=prediction(score, final_dataTst$loan_status, label.ordering = c("Fully Paid", "C
harged Off"))
    #label.ordering here specifies the 'negative', 'positive' class labels

#ROC curve
aucPerf <-performance(pred, "tpr", "fpr")
plot(aucPerf)
abline(a=0, b= 1)
```




```
#AUC value
aucPerf=performance(pred, "auc")
aucPerf@y.values
```

```
## [[1]]
## [1] 0.5720689
```

```
#Lift curve
liftPerf <-performance(pred, "lift", "rpp")
plot(liftPerf)
```




```
test_preds = predict(lcDT1_3,final_dataTst, type="prob")
thrsh = 0.5
test_preds <- ifelse(test_preds[,1] > thrsh, "Charged Off", "Fully Paid")
confusionMatrix(factor(test_preds,levels=c('Charged Off','Fully Paid')),final_dataTst
$loan_status,positive = "Charged Off")
```

```
## Confusion Matrix and Statistics
##
##               Reference
## Prediction    Charged Off Fully Paid
##   Charged Off         588       2373
##   Fully Paid         3485      23554
##
##                Accuracy : 0.8047
##                  95% CI : (0.8002, 0.8092)
##     No Information Rate : 0.8642
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0597
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.1444
##             Specificity : 0.9085
##          Pos Pred Value : 0.1986
##          Neg Pred Value : 0.8711
##              Prevalence : 0.1358
##          Detection Rate : 0.0196
##    Detection Prevalence : 0.0987
##       Balanced Accuracy : 0.5264
##
##        'Positive' Class : Charged Off
##
```

Question 6

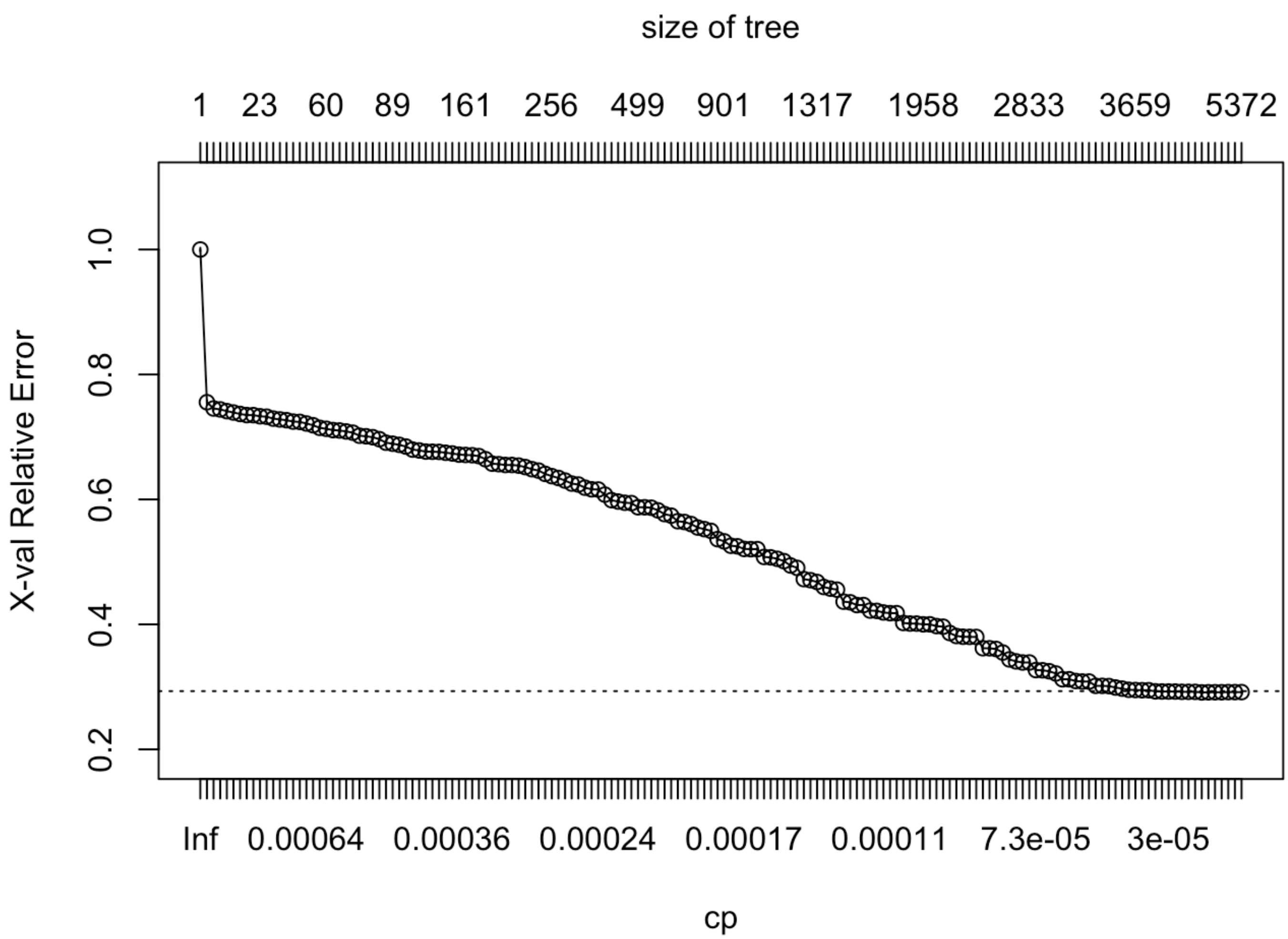
```
set.seed(673)

library(ROSE)
```

```
## Loaded ROSE 0.0-4
```

```
balanced_train <- ovun.sample(loan_status ~ ., data = final_dataTrn, method = "over",
N = 120000)$data
round(100*prop.table(table(balanced_train$loan_status)),digits=2)
```

```
##
##  Fully Paid Charged Off
##       50.24       49.76
```

```
#Now that we have over sampled the charged off data. We can use the above balanced tr
ain set to train our model and test our specivity and accuracy again.

dt_model <- rpart(loan_status ~., data=balanced_train, method="class", parms = list(s
plit = "information"), control = rpart.control(cp=-1))

plotcp(dt_model)
```




```
printcp(dt_model)
```

```
##
## Classification tree:
## rpart(formula = loan_status ~ ., data = balanced_train, method = "class",
##     parms = list(split = "information"), control = rpart.control(cp = -1))
##
## Variables actually used in tree construction:
##  [1] acc_open_past_24mths       annual_inc
##  [3] avg_cur_bal                bc_open_to_buy
##  [5] bc_util                    borrHistory
##  [7] dti                        home_ownership
##  [9] initial_list_status        inq_last_6mths
```

```
## [11] installment                 int_rate
## [13] loan_amnt                   mo_sin_old_il_acct
## [15] mo_sin_old_rev_tl_op        mo_sin_rcnt_rev_tl_op
## [17] mo_sin_rcnt_tl              mort_acc
## [19] mths_since_last_delinq      mths_since_recent_bc
## [21] mths_since_recent_inq       num_bc_tl
## [23] num_il_tl                   num_op_rev_tl
## [25] num_rev_accts               num_sats
## [27] open_acc                    openAccRatio
## [29] pct_tl_nvr_dlq              revol_bal
## [31] revol_util                  satisBankcardAccts_prop
## [33] sub_grade                   tot_cur_bal
## [35] tot_hi_cred_lim             total_acc
## [37] total_bal_ex_mort           total_bc_limit
## [39] total_il_high_credit_limit total_rev_hi_lim
##
## Root node error: 59712/120000 = 0.4976
##
## n= 120000
##
##              CP nsplit rel error  xerror      xstd
## 1    2.4446e-01      0   1.00000 1.00000 0.0029006
## 2    4.9990e-03      1   0.75554 0.75554 0.0028100
## 3    1.8673e-03      3   0.74555 0.74555 0.0028024
## 4    1.5407e-03      5   0.74181 0.74405 0.0028013
## 5    1.4989e-03      6   0.74027 0.74129 0.0027991
## 6    1.3900e-03      8   0.73727 0.73910 0.0027974
## 7    9.2946e-04     13   0.72957 0.73679 0.0027956
## 8    9.0434e-04     15   0.72771 0.73498 0.0027941
## 9    8.8759e-04     21   0.72181 0.73491 0.0027941
## 10   8.5410e-04     22   0.72093 0.73310 0.0027926
## 11   7.7874e-04     24   0.71922 0.73240 0.0027920
## 12   7.2012e-04     26   0.71766 0.72935 0.0027895
## 13   7.0338e-04     27   0.71694 0.72799 0.0027884
## 14   6.8663e-04     29   0.71553 0.72681 0.0027874
## 15   6.6988e-04     30   0.71485 0.72458 0.0027856
## 16   6.5314e-04     41   0.70664 0.72411 0.0027852
## 17   6.1964e-04     44   0.70468 0.72153 0.0027830
## 18   6.0708e-04     45   0.70406 0.71873 0.0027806
## 19   5.8280e-04     49   0.70163 0.71458 0.0027770
## 20   5.6940e-04     59   0.69502 0.71306 0.0027757
## 21   5.6103e-04     61   0.69388 0.71100 0.0027739
## 22   5.5824e-04     65   0.69164 0.71034 0.0027733
## 23   5.5265e-04     70   0.68859 0.70877 0.0027719
## 24   5.1079e-04     77   0.68423 0.70684 0.0027702
## 25   5.0241e-04     79   0.68321 0.70199 0.0027658
## 26   4.8566e-04     81   0.68221 0.70107 0.0027650
## 27   4.7729e-04     83   0.68124 0.69959 0.0027636
## 28   4.5217e-04     86   0.67966 0.69644 0.0027607
## 29   4.3542e-04     87   0.67921 0.69080 0.0027554
```

```
## 30   4.2984e-04     88   0.67877 0.68917 0.0027538
## 31   4.1868e-04     98   0.67442 0.68737 0.0027521
## 32   3.8518e-04    100   0.67358 0.68464 0.0027495
## 33   3.7513e-04    101   0.67320 0.67978 0.0027447
## 34   3.7402e-04    106   0.67132 0.67842 0.0027434
## 35   3.6844e-04    115   0.66704 0.67656 0.0027415
## 36   3.6509e-04    116   0.66667 0.67633 0.0027413
## 37   3.6365e-04    121   0.66484 0.67610 0.0027411
## 38   3.6006e-04    149   0.65175 0.67467 0.0027396
## 39   3.5169e-04    153   0.65030 0.67358 0.0027385
## 40   3.4611e-04    157   0.64890 0.67172 0.0027367
## 41   3.4331e-04    160   0.64786 0.67115 0.0027361
## 42   3.4052e-04    162   0.64717 0.67072 0.0027356
## 43   3.3494e-04    165   0.64615 0.66935 0.0027342
## 44   3.1819e-04    169   0.64469 0.66439 0.0027291
## 45   3.1261e-04    179   0.64151 0.65727 0.0027216
## 46   3.0982e-04    185   0.63964 0.65640 0.0027207
## 47   3.0815e-04    190   0.63798 0.65511 0.0027193
## 48   3.0703e-04    195   0.63644 0.65511 0.0027193
## 49   3.0145e-04    199   0.63520 0.65414 0.0027183
## 50   2.9586e-04    203   0.63399 0.65176 0.0027157
## 51   2.9307e-04    209   0.63190 0.64838 0.0027120
## 52   2.8470e-04    224   0.62656 0.64613 0.0027096
## 53   2.7912e-04    237   0.62239 0.64089 0.0027037
## 54   2.7633e-04    255   0.61562 0.63724 0.0026996
## 55   2.6795e-04    278   0.60725 0.63431 0.0026963
## 56   2.6237e-04    299   0.60125 0.63017 0.0026915
## 57   2.5679e-04    302   0.60047 0.62537 0.0026859
## 58   2.5121e-04    314   0.59681 0.62386 0.0026841
## 59   2.4702e-04    355   0.58484 0.61882 0.0026781
## 60   2.4562e-04    364   0.58233 0.61621 0.0026750
## 61   2.4283e-04    379   0.57789 0.61587 0.0026746
## 62   2.3446e-04    405   0.57104 0.60772 0.0026645
## 63   2.2888e-04    438   0.56268 0.59886 0.0026534
## 64   2.2776e-04    463   0.55424 0.59656 0.0026505
## 65   2.2609e-04    490   0.54595 0.59467 0.0026480
## 66   2.2329e-04    495   0.54441 0.59412 0.0026473
## 67   2.2190e-04    498   0.54374 0.58745 0.0026386
## 68   2.2010e-04    514   0.53957 0.58745 0.0026386
## 69   2.1771e-04    528   0.53562 0.58672 0.0026376
## 70   2.1213e-04    598   0.51752 0.58255 0.0026321
## 71   2.0934e-04    608   0.51506 0.57652 0.0026240
## 72   2.0655e-04    614   0.51377 0.57402 0.0026206
## 73   2.0096e-04    617   0.51315 0.56506 0.0026081
## 74   1.9678e-04    732   0.48704 0.56386 0.0026064
## 75   1.9538e-04    738   0.48553 0.56057 0.0026018
## 76   1.9259e-04    766   0.47902 0.55501 0.0025938
## 77   1.8980e-04    791   0.47366 0.55247 0.0025901
## 78   1.8422e-04    819   0.46701 0.54967 0.0025860
## 79   1.8003e-04    896   0.45140 0.53642 0.0025662
```

```
## 80   1.7752e-04    900   0.45068 0.53287 0.0025608
## 81   1.7584e-04    909   0.44864 0.52584 0.0025499
## 82   1.7305e-04    935   0.44339 0.52492 0.0025485
## 83   1.7082e-04    954   0.43956 0.52077 0.0025419
## 84   1.7026e-04    983   0.43323 0.52040 0.0025413
## 85   1.6747e-04    992   0.43154 0.52040 0.0025413
## 86   1.6412e-04   1095   0.41163 0.50792 0.0025212
## 87   1.6328e-04   1100   0.41081 0.50725 0.0025201
## 88   1.6189e-04   1106   0.40975 0.50497 0.0025163
## 89   1.5910e-04   1123   0.40687 0.50124 0.0025101
## 90   1.5631e-04   1136   0.40479 0.49424 0.0024983
## 91   1.5072e-04   1159   0.40010 0.49042 0.0024918
## 92   1.4514e-04   1257   0.38475 0.47242 0.0024600
## 93   1.4235e-04   1266   0.38284 0.47073 0.0024570
## 94   1.3956e-04   1316   0.37508 0.46786 0.0024518
## 95   1.3816e-04   1325   0.37383 0.45984 0.0024370
## 96   1.3398e-04   1329   0.37328 0.45730 0.0024322
## 97   1.2839e-04   1471   0.35335 0.45559 0.0024290
## 98   1.2560e-04   1478   0.35227 0.43634 0.0023918
## 99   1.2281e-04   1563   0.33846 0.43532 0.0023898
## 100  1.2142e-04   1597   0.33374 0.43094 0.0023810
## 101  1.2058e-04   1604   0.33268 0.43094 0.0023810
## 102  1.1962e-04   1609   0.33208 0.42213 0.0023631
## 103  1.1723e-04   1639   0.32747 0.42161 0.0023621
## 104  1.1388e-04   1831   0.30383 0.41909 0.0023569
## 105  1.1304e-04   1846   0.30208 0.41792 0.0023545
## 106  1.1165e-04   1867   0.29848 0.41792 0.0023545
## 107  1.0886e-04   1870   0.29815 0.40225 0.0023212
## 108  1.0809e-04   1923   0.29205 0.40134 0.0023193
## 109  1.0718e-04   1951   0.28827 0.40128 0.0023191
## 110  1.0606e-04   1957   0.28761 0.40007 0.0023165
## 111  1.0551e-04   2072   0.27005 0.40007 0.0023165
## 112  1.0467e-04   2095   0.26686 0.39729 0.0023104
## 113  1.0327e-04   2107   0.26541 0.39630 0.0023083
## 114  1.0048e-04   2115   0.26454 0.38615 0.0022857
## 115  9.7133e-05   2284   0.24446 0.38140 0.0022749
## 116  9.6296e-05   2289   0.24397 0.38024 0.0022722
## 117  9.4900e-05   2305   0.24144 0.38002 0.0022717
## 118  9.2109e-05   2318   0.24005 0.37989 0.0022714
## 119  9.0434e-05   2420   0.23015 0.36182 0.0022290
## 120  8.9318e-05   2428   0.22903 0.36172 0.0022288
## 121  8.7922e-05   2452   0.22660 0.36060 0.0022261
## 122  8.3735e-05   2460   0.22590 0.35495 0.0022124
## 123  8.0944e-05   2663   0.20728 0.34392 0.0021849
## 124  7.8153e-05   2675   0.20582 0.34077 0.0021770
## 125  7.5362e-05   2702   0.20359 0.33901 0.0021725
## 126  7.3687e-05   2832   0.19231 0.33901 0.0021725
## 127  7.2571e-05   2843   0.19100 0.32685 0.0021409
## 128  7.1175e-05   2868   0.18916 0.32653 0.0021401
## 129  6.6988e-05   2893   0.18641 0.32496 0.0021359
```

```
## 130  6.3639e-05   3111   0.16945 0.32143 0.0021265
## 131  6.2801e-05   3120   0.16856 0.31215 0.0021013
## 132  6.1406e-05   3142   0.16667 0.31215 0.0021013
## 133  6.0289e-05   3173   0.16471 0.30937 0.0020937
## 134  5.8615e-05   3178   0.16441 0.30853 0.0020913
## 135  5.5824e-05   3362   0.15216 0.30853 0.0020913
## 136  5.4428e-05   3409   0.14923 0.30161 0.0020720
## 137  5.3591e-05   3415   0.14876 0.30161 0.0020720
## 138  5.0241e-05   3420   0.14850 0.30125 0.0020709
## 139  4.6054e-05   3571   0.14029 0.29893 0.0020644
## 140  4.4659e-05   3577   0.13996 0.29718 0.0020593
## 141  4.1868e-05   3640   0.13676 0.29532 0.0020540
## 142  4.0193e-05   3658   0.13600 0.29493 0.0020529
## 143  3.9076e-05   3672   0.13543 0.29460 0.0020519
## 144  3.7681e-05   3712   0.13351 0.29460 0.0020519
## 145  3.3494e-05   3720   0.13321 0.29282 0.0020468
## 146  3.0145e-05   3833   0.12934 0.29257 0.0020461
## 147  2.9307e-05   3838   0.12919 0.29254 0.0020460
## 148  2.7912e-05   3854   0.12872 0.29250 0.0020459
## 149  2.5121e-05   3861   0.12845 0.29209 0.0020447
## 150  2.2329e-05   3884   0.12785 0.29207 0.0020446
## 151  1.6747e-05   3892   0.12766 0.29198 0.0020444
## 152  1.1165e-05   4050   0.12502 0.29111 0.0020418
## 153  1.0048e-05   4065   0.12485 0.29133 0.0020425
## 154  8.3735e-06   4070   0.12480 0.29138 0.0020426
## 155  5.5824e-06   4122   0.12436 0.29140 0.0020427
## 156  3.3494e-06   4142   0.12425 0.29162 0.0020433
## 157  0.0000e+00   4147   0.12423 0.29158 0.0020432
## 158 -1.0000e+00   5371   0.12423 0.29158 0.0020432
```

```
#Decision Tree
dt_model <- rpart(loan_status ~., data=balanced_train, method="class", parms = list(s
plit = "infomration"), control = rpart.control(minsplit = 50,minbucket = 35, cp=0.000
0083846))

printcp(dt_model)
```

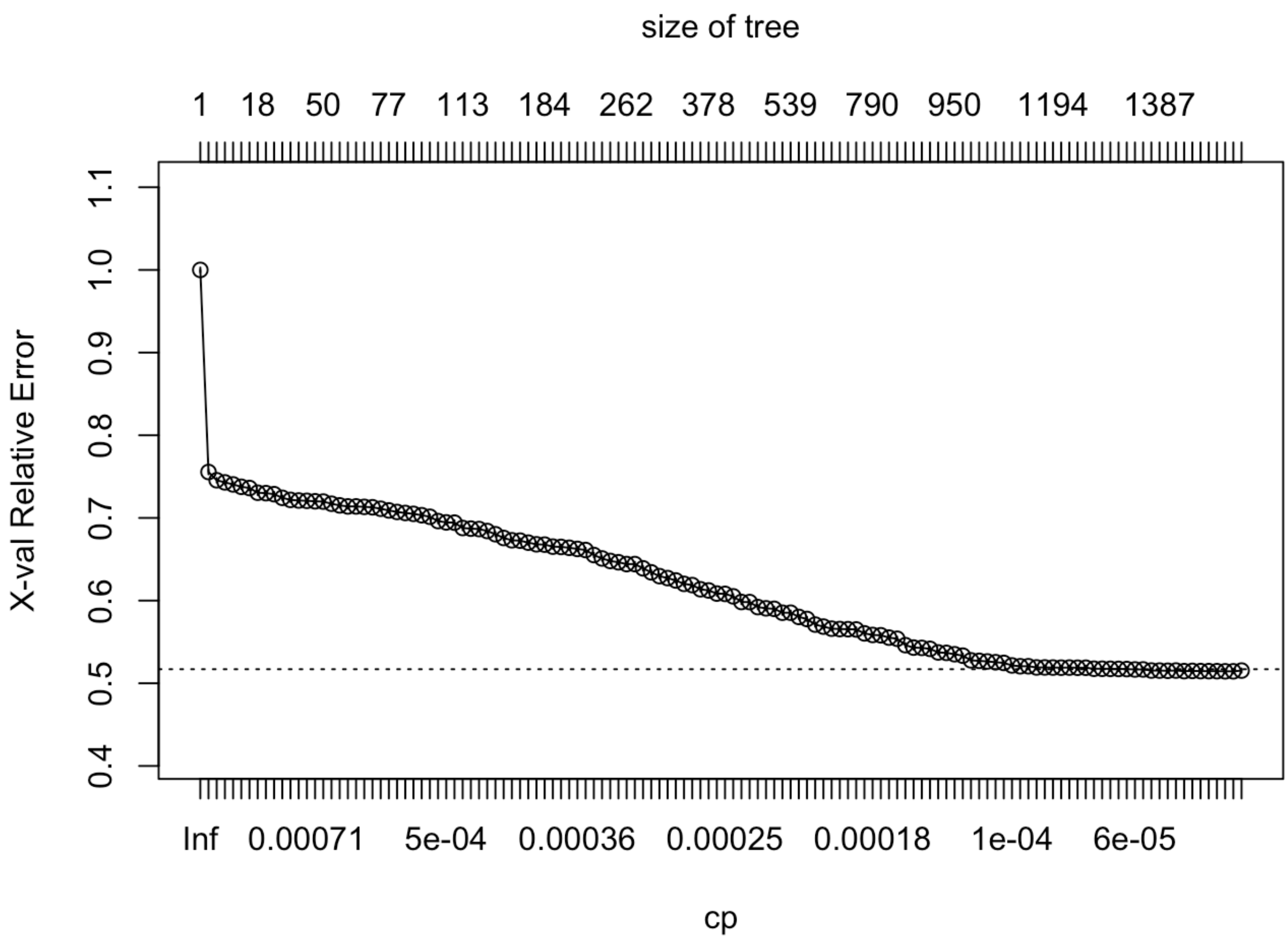
```
##
## Classification tree:
## rpart(formula = loan_status ~ ., data = balanced_train, method = "class",
##     parms = list(split = "infomration"), control = rpart.control(minsplit = 50,
##         minbucket = 35, cp = 8.3846e-06))
##
## Variables actually used in tree construction:
##  [1] acc_open_past_24mths       annual_inc
##  [3] avg_cur_bal                bc_open_to_buy
##  [5] bc_util                    borrHistory
##  [7] dti                        home_ownership
##  [9] initial_list_status        inq_last_6mths
```

```
## [11] installment                int_rate
## [13] loan_amnt                  mo_sin_old_il_acct
## [15] mo_sin_old_rev_tl_op       mo_sin_rcnt_rev_tl_op
## [17] mo_sin_rcnt_tl             mort_acc
## [19] mths_since_last_delinq     mths_since_recent_bc
## [21] mths_since_recent_inq      num_bc_tl
## [23] num_il_tl                  num_op_rev_tl
## [25] num_rev_accts              num_sats
## [27] open_acc                   openAccRatio
## [29] pct_tl_nvr_dlq             revol_bal
## [31] revol_util                 satisBankcardAccts_prop
## [33] sub_grade                  tot_cur_bal
## [35] tot_hi_cred_lim            total_acc
## [37] total_bal_ex_mort          total_bc_limit
## [39] total_il_high_credit_limit total_rev_hi_lim
##
## Root node error: 59712/120000 = 0.4976
##
## n= 120000
##
##              CP nsplit rel error  xerror      xstd
## 1   2.4446e-01      0   1.00000 1.00000 0.0029006
## 2   4.9990e-03      1   0.75554 0.75554 0.0028100
## 3   1.8673e-03      3   0.74555 0.74555 0.0028024
## 4   1.7417e-03      5   0.74181 0.74298 0.0028005
## 5   1.5407e-03      9   0.73444 0.74052 0.0027985
## 6   1.4989e-03     10   0.73290 0.73766 0.0027963
## 7   9.2946e-04     12   0.72990 0.73617 0.0027951
## 8   9.0434e-04     17   0.72525 0.73047 0.0027905
## 9   8.8759e-04     23   0.71962 0.72997 0.0027901
## 10  8.5410e-04     24   0.71873 0.72873 0.0027890
## 11  7.9269e-04     26   0.71703 0.72431 0.0027854
## 12  7.5362e-04     30   0.71366 0.72183 0.0027833
## 13  7.2012e-04     31   0.71291 0.72109 0.0027826
## 14  7.0338e-04     33   0.71147 0.72069 0.0027823
## 15  6.9500e-04     38   0.70788 0.72012 0.0027818
## 16  6.8663e-04     49   0.69976 0.71977 0.0027815
## 17  6.6988e-04     50   0.69907 0.71718 0.0027793
## 18  6.5314e-04     52   0.69773 0.71517 0.0027775
## 19  6.4755e-04     60   0.69128 0.71391 0.0027764
## 20  6.4476e-04     63   0.68934 0.71391 0.0027764
## 21  6.3639e-04     65   0.68805 0.71339 0.0027760
## 22  6.1964e-04     69   0.68551 0.71289 0.0027756
## 23  6.0289e-04     72   0.68365 0.71111 0.0027740
## 24  5.8615e-04     76   0.68115 0.70907 0.0027722
## 25  5.6940e-04     77   0.68057 0.70723 0.0027705
## 26  5.6103e-04     85   0.67549 0.70599 0.0027694
## 27  5.5265e-04     87   0.67437 0.70493 0.0027685
## 28  5.3172e-04     89   0.67327 0.70329 0.0027670
## 29  5.1079e-04     96   0.66903 0.70142 0.0027653
```

```
## 30  5.0241e-04     98   0.66801 0.69617 0.0027604
## 31  4.9683e-04    102   0.66600 0.69460 0.0027590
## 32  4.7227e-04    106   0.66395 0.69430 0.0027587
## 33  4.6892e-04    112   0.66104 0.68782 0.0027525
## 34  4.5217e-04    113   0.66057 0.68705 0.0027518
## 35  4.4380e-04    117   0.65876 0.68655 0.0027513
## 36  4.4101e-04    122   0.65652 0.68417 0.0027490
## 37  4.1868e-04    125   0.65519 0.68028 0.0027452
## 38  4.1449e-04    129   0.65352 0.67574 0.0027407
## 39  4.0193e-04    148   0.64198 0.67295 0.0027379
## 40  4.0041e-04    150   0.64118 0.67251 0.0027375
## 41  3.8853e-04    170   0.63247 0.67010 0.0027350
## 42  3.8518e-04    178   0.62899 0.66809 0.0027329
## 43  3.7960e-04    183   0.62706 0.66770 0.0027326
## 44  3.7681e-04    190   0.62440 0.66528 0.0027300
## 45  3.7402e-04    202   0.61899 0.66492 0.0027297
## 46  3.6844e-04    209   0.61512 0.66387 0.0027286
## 47  3.6006e-04    212   0.61401 0.66250 0.0027272
## 48  3.5169e-04    214   0.61329 0.66117 0.0027258
## 49  3.3494e-04    229   0.60802 0.65513 0.0027193
## 50  3.2936e-04    239   0.60467 0.65097 0.0027149
## 51  3.2657e-04    253   0.59959 0.64796 0.0027116
## 52  3.2378e-04    255   0.59894 0.64637 0.0027098
## 53  3.1819e-04    261   0.59700 0.64418 0.0027074
## 54  3.1484e-04    271   0.59373 0.64418 0.0027074
## 55  3.0982e-04    276   0.59216 0.63903 0.0027016
## 56  3.0145e-04    282   0.59030 0.63419 0.0026961
## 57  2.9307e-04    300   0.58437 0.62959 0.0026908
## 58  2.8470e-04    316   0.57968 0.62723 0.0026881
## 59  2.7633e-04    326   0.57668 0.62435 0.0026847
## 60  2.7214e-04    340   0.57255 0.62024 0.0026798
## 61  2.6795e-04    345   0.57109 0.61875 0.0026780
## 62  2.6237e-04    371   0.56349 0.61371 0.0026719
## 63  2.5958e-04    377   0.56171 0.61227 0.0026702
## 64  2.5539e-04    397   0.55503 0.60852 0.0026655
## 65  2.5121e-04    401   0.55401 0.60812 0.0026650
## 66  2.4283e-04    430   0.54622 0.60514 0.0026613
## 67  2.4004e-04    440   0.54339 0.59831 0.0026527
## 68  2.3446e-04    444   0.54230 0.59820 0.0026526
## 69  2.2776e-04    481   0.53328 0.59214 0.0026447
## 70  2.2609e-04    488   0.53152 0.59065 0.0026428
## 71  2.2190e-04    513   0.52472 0.59003 0.0026420
## 72  2.2010e-04    519   0.52313 0.58534 0.0026358
## 73  2.1771e-04    538   0.51785 0.58534 0.0026358
## 74  2.0934e-04    566   0.51147 0.58024 0.0026290
## 75  2.0655e-04    599   0.50399 0.57776 0.0026256
## 76  2.0096e-04    631   0.49488 0.57097 0.0026164
## 77  1.9538e-04    656   0.48947 0.56856 0.0026130
## 78  1.9259e-04    659   0.48888 0.56607 0.0026095
## 79  1.9092e-04    688   0.48310 0.56560 0.0026089
```

```
## 80  1.8980e-04    693   0.48215 0.56541 0.0026086
## 81  1.8422e-04    730   0.47354 0.56510 0.0026082
## 82  1.8003e-04    780   0.46272 0.56037 0.0026015
## 83  1.7864e-04    789   0.46098 0.55845 0.0025987
## 84  1.7584e-04    796   0.45967 0.55806 0.0025982
## 85  1.7305e-04    828   0.45398 0.55523 0.0025941
## 86  1.6747e-04    843   0.45101 0.55384 0.0025921
## 87  1.5910e-04    865   0.44698 0.54609 0.0025807
## 88  1.5631e-04    871   0.44602 0.54316 0.0025764
## 89  1.5491e-04    874   0.44556 0.54279 0.0025758
## 90  1.5072e-04    878   0.44494 0.54163 0.0025741
## 91  1.4514e-04    928   0.43708 0.53730 0.0025676
## 92  1.4235e-04    935   0.43591 0.53676 0.0025668
## 93  1.3956e-04    949   0.43372 0.53510 0.0025642
## 94  1.3398e-04    953   0.43306 0.53338 0.0025616
## 95  1.2560e-04   1001   0.42509 0.52772 0.0025528
## 96  1.2351e-04   1009   0.42409 0.52713 0.0025519
## 97  1.2281e-04   1022   0.42216 0.52638 0.0025507
## 98  1.1723e-04   1031   0.42105 0.52549 0.0025494
## 99  1.0886e-04   1069   0.41660 0.52462 0.0025480
## 100 1.0048e-04   1083   0.41498 0.52157 0.0025432
## 101 9.4900e-05   1138   0.40747 0.52058 0.0025416
## 102 9.3783e-05   1157   0.40566 0.52058 0.0025416
## 103 9.2109e-05   1174   0.40315 0.51914 0.0025393
## 104 9.0434e-05   1188   0.40186 0.51914 0.0025393
## 105 8.9318e-05   1193   0.40141 0.51901 0.0025391
## 106 8.3735e-05   1202   0.40061 0.51886 0.0025389
## 107 8.0386e-05   1230   0.39778 0.51897 0.0025391
## 108 7.8153e-05   1236   0.39727 0.51877 0.0025387
## 109 7.5362e-05   1241   0.39680 0.51877 0.0025387
## 110 7.2571e-05   1263   0.39505 0.51758 0.0025368
## 111 6.9779e-05   1266   0.39483 0.51755 0.0025368
## 112 6.6988e-05   1272   0.39441 0.51747 0.0025367
## 113 6.4197e-05   1309   0.39188 0.51735 0.0025365
## 114 6.1406e-05   1315   0.39150 0.51718 0.0025362
## 115 5.8615e-05   1318   0.39131 0.51670 0.0025354
## 116 5.4428e-05   1331   0.39049 0.51673 0.0025355
## 117 5.0241e-05   1346   0.38935 0.51541 0.0025333
## 118 4.6054e-05   1386   0.38734 0.51534 0.0025332
## 119 4.1868e-05   1390   0.38716 0.51511 0.0025329
## 120 3.9076e-05   1404   0.38651 0.51539 0.0025333
## 121 3.3494e-05   1410   0.38627 0.51469 0.0025322
## 122 2.7912e-05   1443   0.38517 0.51496 0.0025326
## 123 2.6795e-05   1446   0.38508 0.51472 0.0025322
## 124 2.5121e-05   1451   0.38495 0.51469 0.0025322
## 125 2.2329e-05   1471   0.38445 0.51464 0.0025321
## 126 2.0934e-05   1477   0.38431 0.51447 0.0025318
## 127 1.6747e-05   1481   0.38423 0.51447 0.0025318
## 128 8.3846e-06   1514   0.38367 0.51536 0.0025333
```

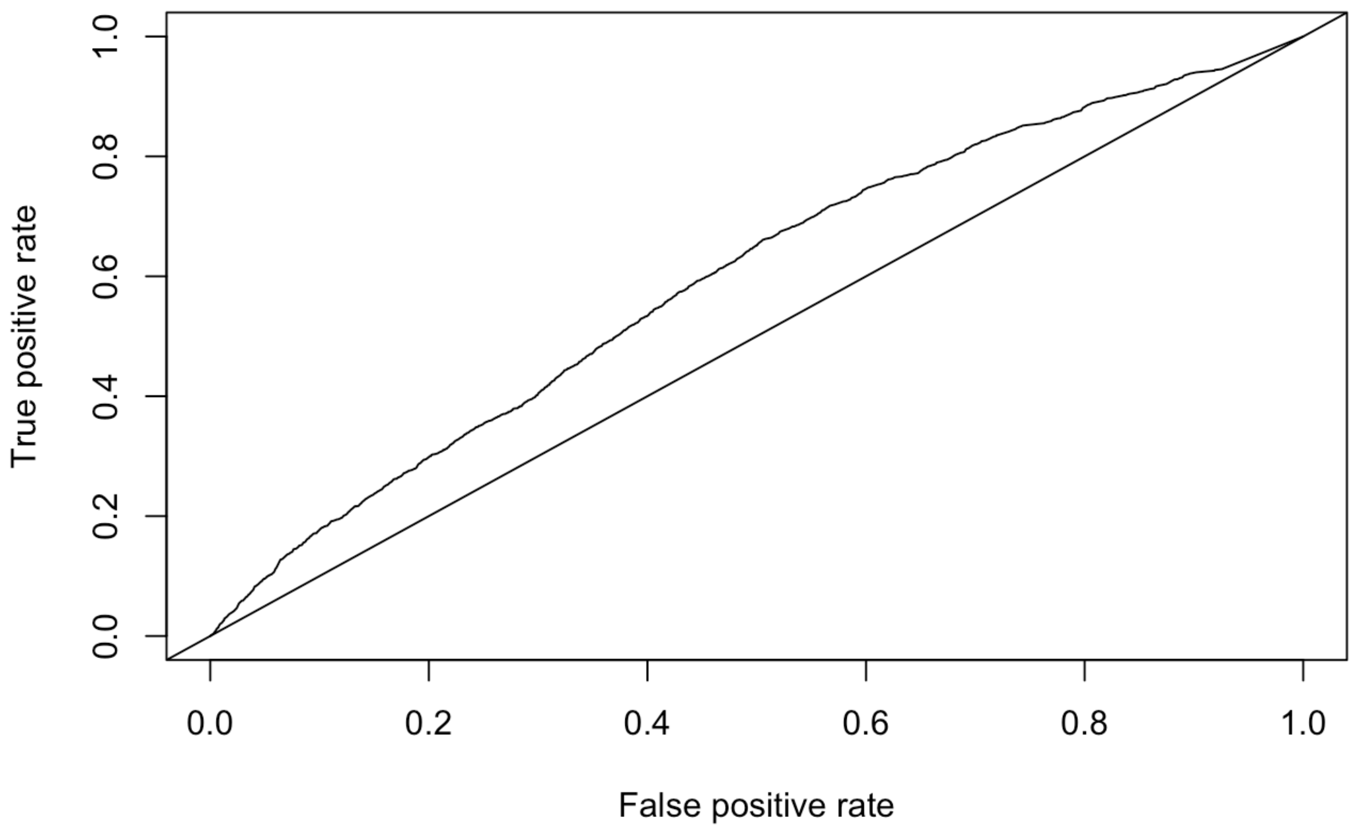
```
plotcp(dt_model)
```




```
#Classification method
test_preds<-predict(dt_model,final_dataTst, type='class')
confusionMatrix(factor(test_preds,levels=c('Charged Off','Fully Paid')),final_dataTst
$loan_status,positive = "Charged Off")
```

```
## Confusion Matrix and Statistics
##
##               Reference
## Prediction    Charged Off Fully Paid
##   Charged Off        1564       7392
##   Fully Paid         2509      18535
##
##                Accuracy : 0.67
##                  95% CI : (0.6646, 0.6753)
##     No Information Rate : 0.8642
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0657
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.38399
##             Specificity : 0.71489
##          Pos Pred Value : 0.17463
##          Neg Pred Value : 0.88077
##              Prevalence : 0.13577
##          Detection Rate : 0.05213
##    Detection Prevalence : 0.29853
##       Balanced Accuracy : 0.54944
##
##        'Positive' Class : Charged Off
##
```

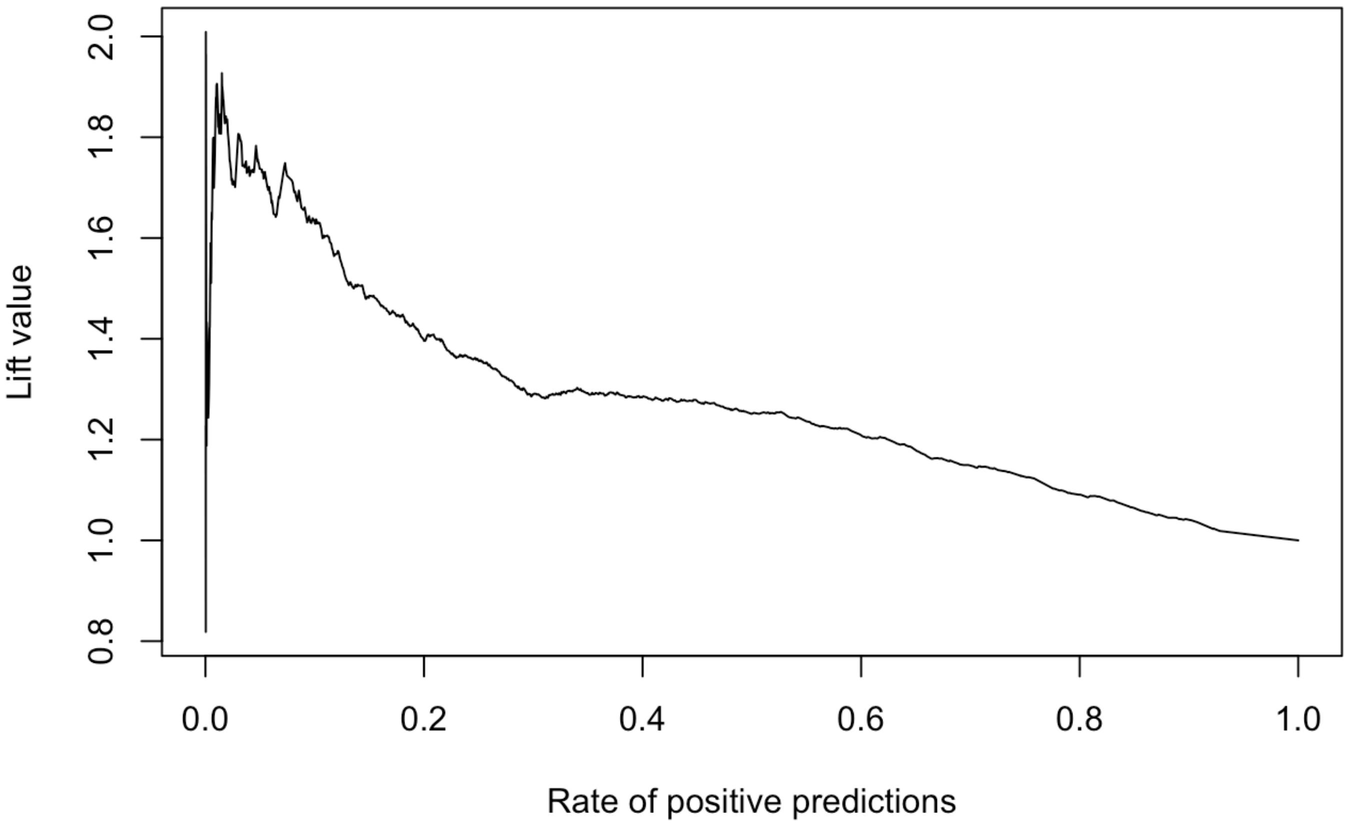
```
#Probability Method
test_preds<-predict(dt_model,final_dataTst, type='prob')[,'Charged Off']
pred=prediction(test_preds, final_dataTst$loan_status, label.ordering = c("Fully Paid
", "Charged Off"))  #label.ordering = (negative class, positive class)


#ROC curve
roc_curve <-performance(pred, "tpr", "fpr")
plot(roc_curve)
abline(a=0, b= 1)
```




```
#AUC value
auc_score<-performance(pred, "auc")
auc_score@y.values
```

```
## [[1]]
## [1] 0.5953038
```

```
#Lift curve
liftPerf <-performance(pred, "lift", "rpp")
plot(liftPerf)
```

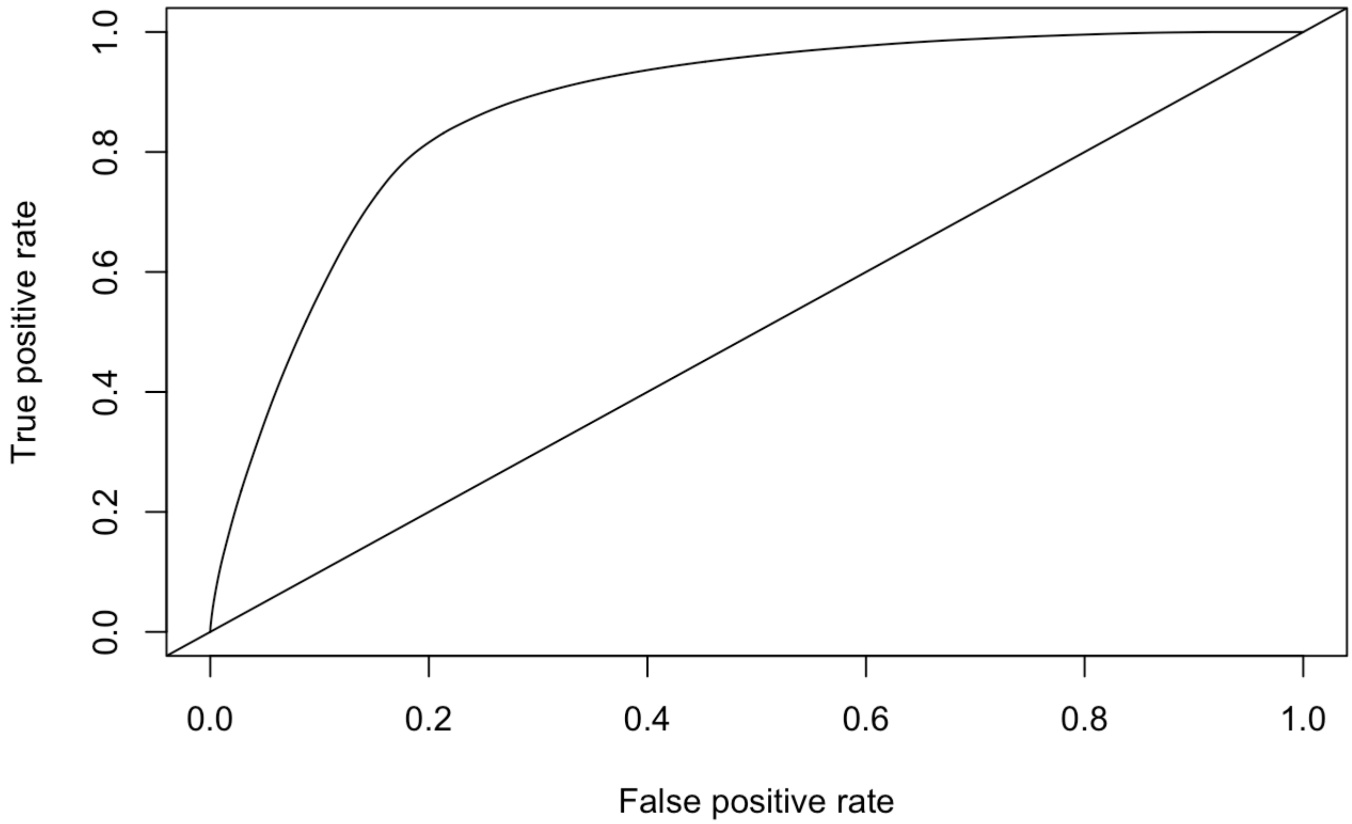



```
## performing tests on training data

#Classification method
test_preds<-predict(dt_model,balanced_train, type='class')
confusionMatrix(factor(test_preds,levels=c('Charged Off','Fully Paid')),balanced_trai
n$loan_status,positive = "Charged Off")
```

```
## Warning in confusionMatrix.default(factor(test_preds, levels = c("Charged
## Off", : Levels are not in the same order for reference and data. Refactoring
## data to match.
```

```
## Confusion Matrix and Statistics
##
##               Reference
## Prediction    Fully Paid Charged Off
##   Fully Paid       47168        9790
##   Charged Off      13120       49922
##
##                Accuracy : 0.8091
##                  95% CI : (0.8068, 0.8113)
##     No Information Rate : 0.5024
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6183
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.8360
##             Specificity : 0.7824
##          Pos Pred Value : 0.7919
##          Neg Pred Value : 0.8281
##              Prevalence : 0.4976
##          Detection Rate : 0.4160
##    Detection Prevalence : 0.5253
##       Balanced Accuracy : 0.8092
##
##        'Positive' Class : Charged Off
##
```

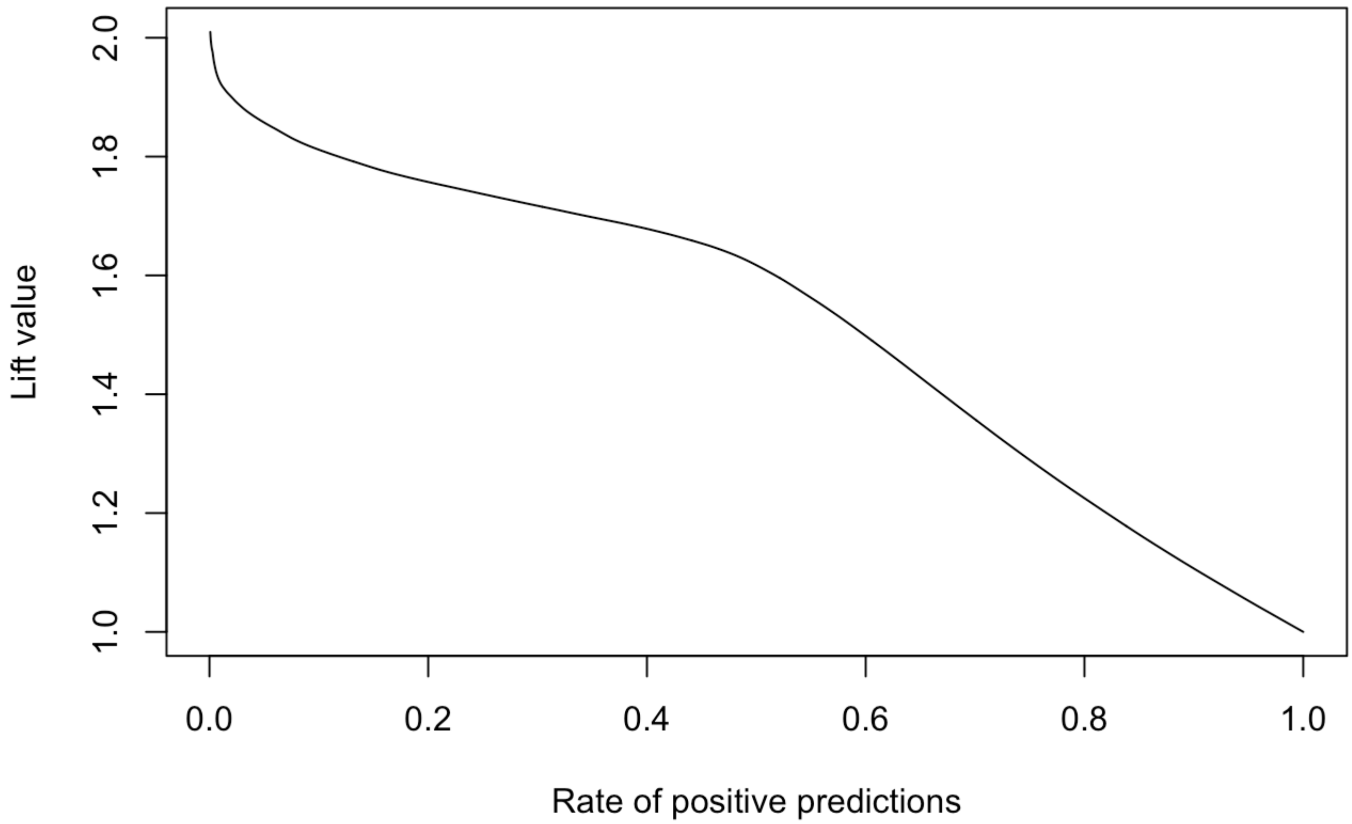
```
#Probability Method
test_preds<-predict(dt_model,balanced_train, type='prob')[,'Charged Off']
pred=prediction(test_preds, balanced_train$loan_status, label.ordering = c("Fully Pai
d", "Charged Off"))  #label.ordering = (negative class, positive class)


#ROC curve
roc_curve <-performance(pred, "tpr", "fpr")
plot(roc_curve)
abline(a=0, b= 1)
```




```
#AUC value
auc_score<-performance(pred, "auc")
auc_score@y.values
```

```
## [[1]]
## [1] 0.8719911
```

```
#Lift curve
liftPerf <-performance(pred, "lift", "rpp")
plot(liftPerf)
```



Question 7-a&b

```
# excluding certain elements from the dataset because of data leakage issue.

new_data1 <- new_dt%>%select(-c(annRet,actualTerm,total_pymnt, balance_to_pay))

names(colMeans(is.na(new_data1)))[colMeans(is.na(new_data1))>0]
```

```
## [1] "mths_since_last_delinq" "revol_util"             "avg_cur_bal"
## [4] "mo_sin_old_il_acct"     "mths_since_recent_bc"   "mths_since_recent_inq"
## [7] "num_rev_accts"          "pct_tl_nvr_dlq"
```

```r
#cc<-table( new_data1$loan_status, replace_na(list( new_data1$mths_since_recent_inq , "missing")) )

#cc[1,]/(cc[2,]+cc[1,])

#replacing some of the missing NA values in the columns by median values

new_data1<- new_data1 %>% replace_na(list(mths_since_last_delinq=median(new_data1$mths_since_last_delinq, na.rm=TRUE),
                                     revol_util = median(new_data1$revol_util, na.rm=TRUE),
                                     avg_cur_bal = median(new_data1$avg_cur_bal, na.rm=TRUE),
                                     mths_since_recent_bc = median(new_data1$mths_since_recent_bc, na.rm=TRUE),
                                     mths_since_recent_inq = median(new_data1$mths_since_recent_inq, na.rm=TRUE),
                                     num_rev_accts = median(new_data1$num_rev_accts, na.rm=TRUE),
                                     pct_tl_nvr_dlq = median(new_data1$pct_tl_nvr_dlq, na.rm=TRUE),
                                     mo_sin_old_il_acct=median(new_data1$mo_sin_old_il_acct, na.rm=TRUE) ))

names(colMeans(is.na(new_data1)))[colMeans(is.na(new_data1))>0]
```

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

```r
library(ranger)

#Splitting data into 70% training and 30%  testing ratio.

TRNPROP = 0.7  #proportion of examples in the training sample
nr<-nrow(new_data1)
trnIndex<- sample(1:nr, size = round(TRNPROP * nr), replace=FALSE)

new_data1Trn <- new_data1[trnIndex, ]
new_data1Tst <- new_data1[-trnIndex, ]

#ran a  random forest based using ranger, splitrule is gini
new_data1T1<- ranger(loan_status ~., data=new_data1Trn, classification = TRUE,
                     num.trees =200, importance='permutation', probability = TRUE)

sort(new_data1T1$variable.importance, decreasing = TRUE)
```

```
##            tot_hi_cred_lim                 tot_cur_bal
##               1.780473e-02                1.576666e-02
##                avg_cur_bal             total_bc_limit
##               1.283776e-02                1.057233e-02
##           total_rev_hi_lim             bc_open_to_buy
##               9.783938e-03                9.415023e-03
##                   int_rate                installment
##               9.172740e-03                8.399520e-03
##          total_bal_ex_mort                  sub_grade
##               7.718983e-03                7.462327e-03
##                funded_amnt                 annual_inc
##               7.353994e-03                7.097650e-03
##                  revol_bal                  loan_amnt
##               6.815858e-03                6.598715e-03
##                    bc_util                      grade
##               6.056259e-03                5.500468e-03
##       acc_open_past_24mths                 revol_util
##               5.347104e-03                4.871431e-03
## total_il_high_credit_limit                  total_acc
##               4.665214e-03                4.515857e-03
##                borrHistory       mo_sin_old_rev_tl_op
##               4.481912e-03                4.455603e-03
##              num_op_rev_tl               num_rev_accts
##               4.022295e-03                3.830656e-03
##                        dti                   open_acc
##               3.474265e-03                3.026190e-03
##                  num_bc_tl                openAccRatio
##               2.984882e-03                2.978213e-03
##                   num_sats                  num_il_tl
##               2.847697e-03                2.530706e-03
##        mo_sin_rcnt_rev_tl_op       mths_since_recent_bc
##               2.520935e-03                2.240789e-03
##               mo_sin_rcnt_tl    satisBankcardAccts_prop
##               2.227337e-03                1.957132e-03
##                   mort_acc          mo_sin_old_il_acct
##               1.883026e-03                1.588350e-03
##             home_ownership              pct_tl_nvr_dlq
##               1.261741e-03                1.142704e-03
##              inq_last_6mths      mths_since_recent_inq
##               6.491880e-04                5.624445e-04
##      mths_since_last_delinq        initial_list_status
##               5.426665e-04                1.432945e-06
##            num_tl_120dpd_2m
##              -9.411572e-07
```

```
#Making predictions and evaluating performance of the model
#training data
#predicting values in training data

predTrn<-predict(new_data1T1,new_data1Trn, type='response') # type response as a clas
sification

# we get the predictions of charged off and fully paid loans in the form of probabili
ties.
#Next we compare if probability of charged off is greated than fully charged(thatis 5
0% threshold value) then loan is
#charged off else fully paid

predictions<- ifelse (predTrn$predictions[,"Charged Off"]>predTrn$predictions[,"Fully
Paid"],"Charged Off","Fully Paid")
#Performance Evaluation
#creating a confusion matrix

CM<-table(pred = predictions, true=new_data1Trn$loan_status)
CM
```

```
##              true
## pred           Charged Off Fully Paid
##   Charged Off        7707          0
##   Fully Paid         1936      60357
```

```
mean(predictions == new_data1Trn$loan_status)
```

```
## [1] 0.9723429
```

```
#accuracy is around 98.8%

# Calculating F1score

precision <- CM[1,1]/(CM[1,1]+CM[1,2])
recall <- CM[1,1]/(CM[1,1]+CM[2,1])
F1 <- (2 * precision * recall) / (precision + recall)
F1
```

```
## [1] 0.888415
```

```r
#testing

predTst=predict(new_data1T1,new_data1Tst, type='response') # type response as a class
ification

#when threshold of charged off >50%
predictions<- ifelse (predTst$predictions[,"Charged Off"]>predTst$predictions[,"Fully
Paid"],"Charged Off","Fully Paid")
CM<-table(pred = predictions, true=new_data1Tst$loan_status)
CM
```

```
##               true
## pred           Charged Off Fully Paid
##   Charged Off           7          6
##   Fully Paid         4135      25852
```

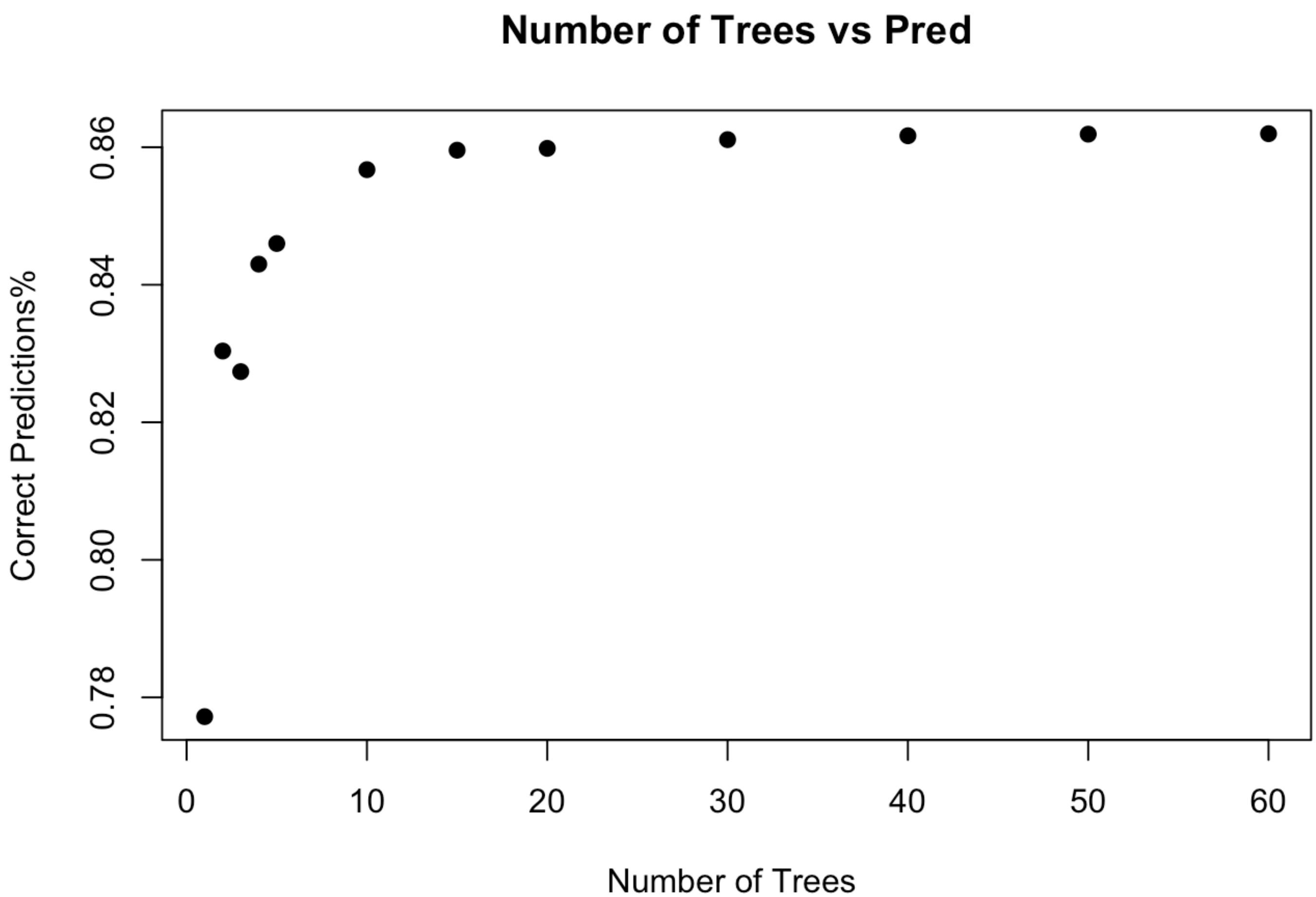
```r
mean(predictions == new_data1Tst$loan_status)
```

```
## [1] 0.8619667
```

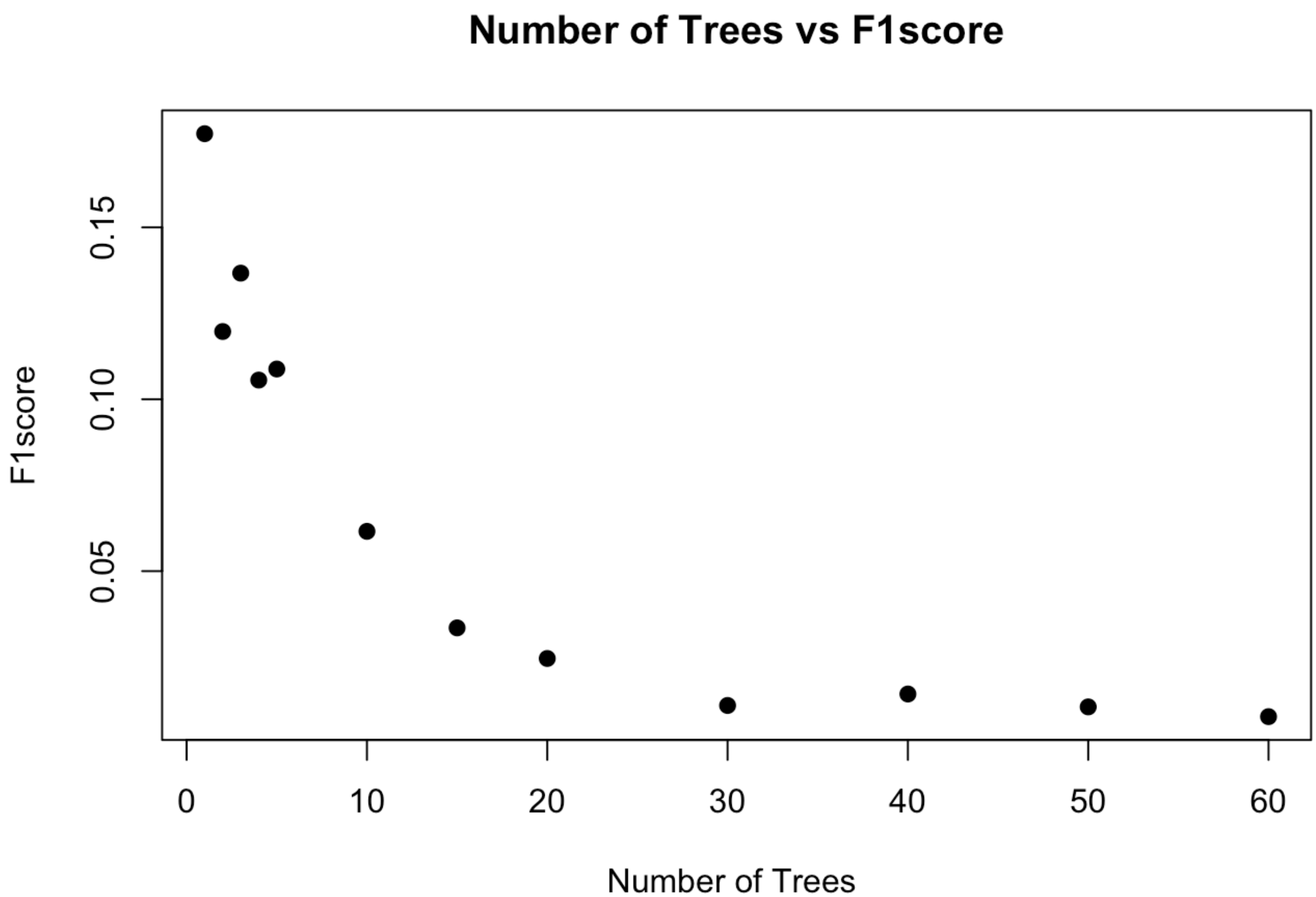
```r
precision <- CM[1,1]/(CM[1,1]+CM[1,2])
recall <- CM[1,1]/(CM[1,1]+CM[2,1])
F1 <- (2 * precision * recall) / (precision + recall)
F1
```

```
## [1] 0.003369434
```

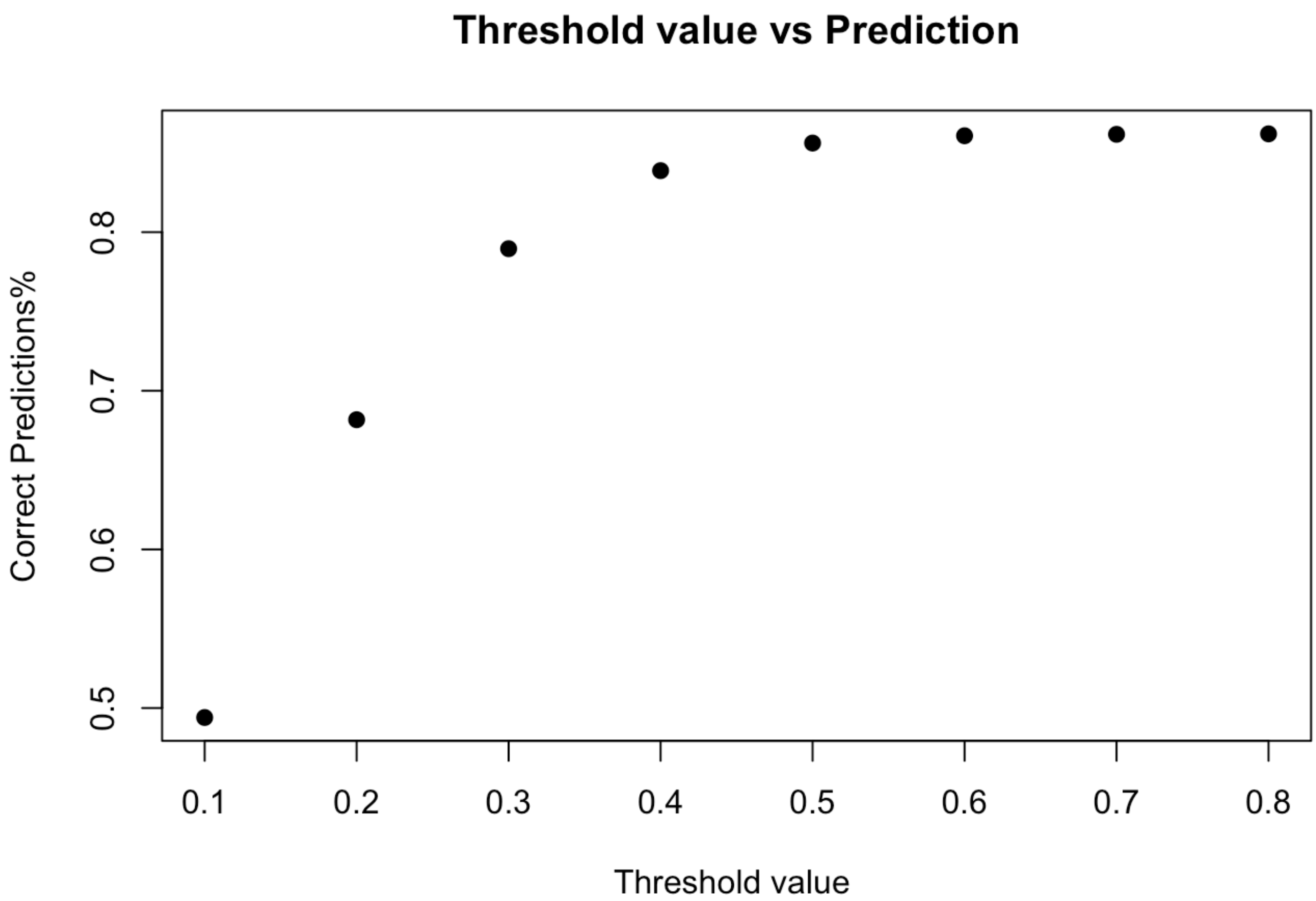
```r
#accuracy for test data is around 93.3%

#altering number of trees in random forest for better precision using a loop
trees.no<- c(1,2,3,4,5,10,15,20,30,40,50,60)
pred<-c()
F1score<-c()
for (i in trees.no)
  {
  trial1<- ranger(loan_status ~., data=new_data1Trn, classification = TRUE,
                     num.trees =i, importance='permutation', probability = TRUE)
  predTst=predict(trial1,new_data1Tst, type='response') # type response as a classifi
cation
  predictions<- ifelse (predTst$predictions[,"Charged Off"]>predTst$predictions[,"Ful
ly Paid"],"Charged Off","Fully  Paid")
  CM<-table(pred = predictions, true=new_data1Tst$loan_status)
  precision <- CM[1,1]/(CM[1,1]+CM[1,2])
  recall <- CM[1,1]/(CM[1,1]+CM[2,1])
  F1 <- (2 * precision * recall) / (precision + recall)
  pred<-append(pred,(CM[1,1]+CM[2,2])/length(predictions))
  F1score<- append(F1score,F1)
}


plot(trees.no, pred, main="Number of Trees vs Pred",
   xlab="Number of Trees ", ylab="Correct Predictions%", pch=19)
```



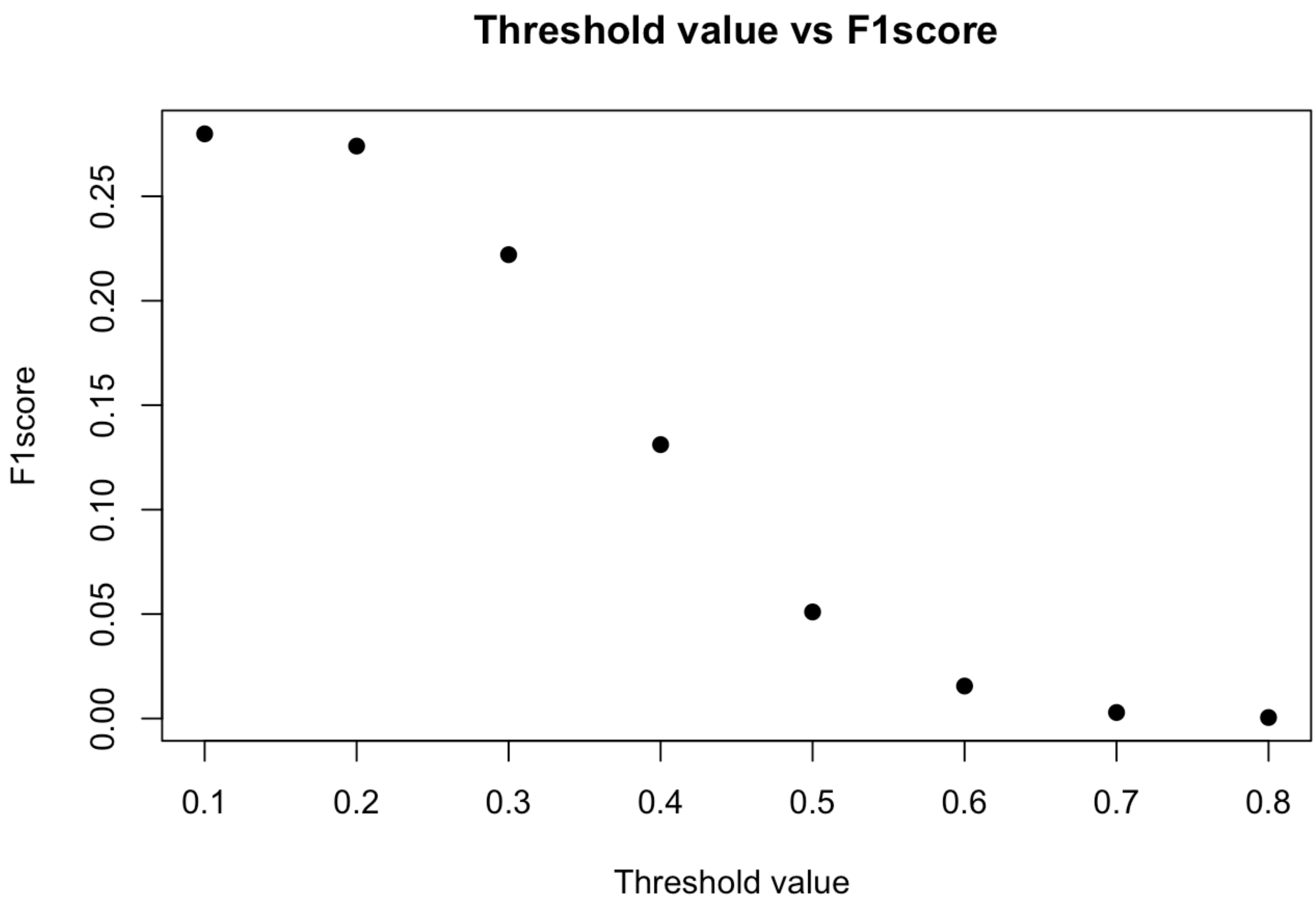

```
plot(trees.no, F1score, main="Number of Trees vs F1score",
   xlab="Number of Trees ", ylab="F1score", pch=19)
```


Number of Trees vs F1score
F1score
0.05
0.10
0.15
0
10
20
30
40
50
60
Number of Trees

```
#we observe that the model has highest accuracy and F1 score when number of trees is
10

new_data1T1<- ranger(loan_status ~., data=new_data1Trn, classification = TRUE,
                     num.trees =10, importance='permutation', probability = TRUE)
#testing
#Checking for different threshold values
predTst=predict(new_data1T1,new_data1Tst, type='response') # type response as a class
ification
perc<-c(0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8) #array of threshold values
pred1<-c()
F1score1<-c()
for (i in perc){
predictions<- ifelse (predTst$predictions[,"Charged Off"]>i,"Charged Off","Fully Paid
")
CM<-table(pred = predictions, true=new_data1Tst$loan_status)
precision <- CM[1,1]/(CM[1,1]+CM[1,2])
recall <- CM[1,1]/(CM[1,1]+CM[2,1])
F1 <- (2 * precision * recall) / (precision + recall)
  pred1<-append(pred1,(CM[1,1]+CM[2,2])/length(predictions))
  F1score1<- append(F1score1,F1)

}
plot(perc, pred1, main="Threshold value vs Prediction",
   xlab="Threshold value ", ylab="Correct Predictions%", pch=19)
```




```
plot(perc, F1score1, main="Threshold value vs F1score",
   xlab="Threshold value ", ylab="F1score", pch=19)
```



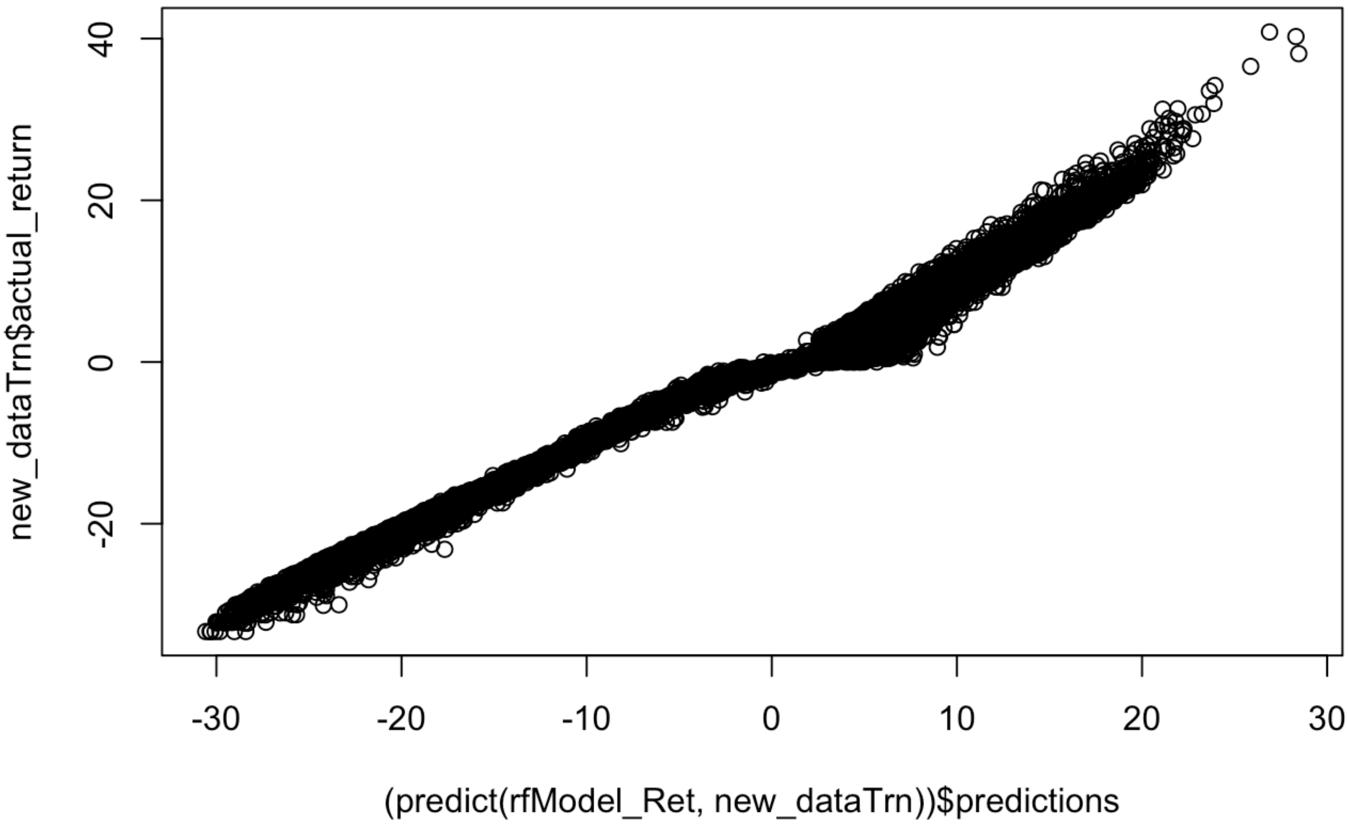

```
# we observe that at threshold value=0.4 we get the maximum accuracy and F1 score
```

Question 8

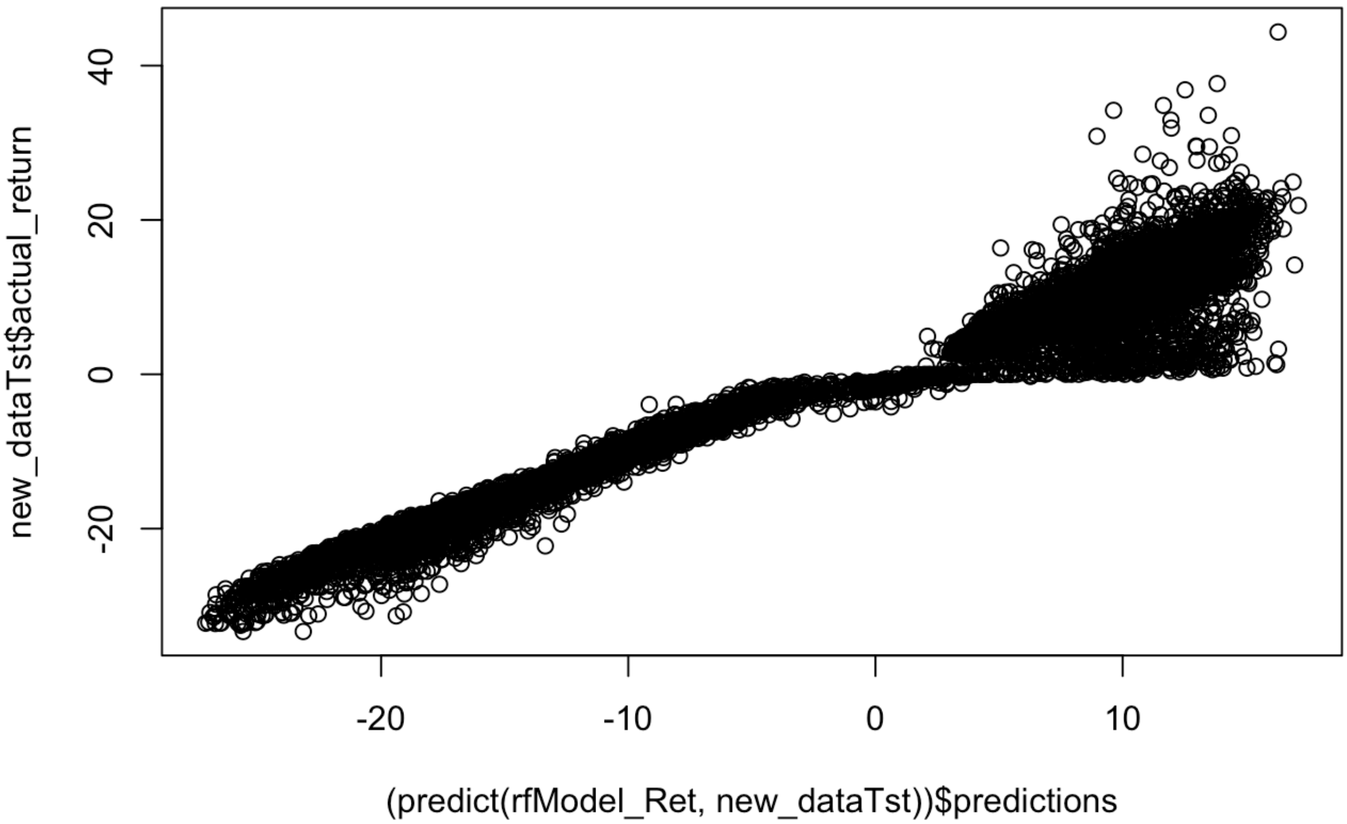
```
#Predict ActualReturn
new_data$actual_return <- ifelse(new_data$actualTerm >0, ((new_data$total_pymnt-new_d
ata$funded_amnt)/new_data$funded_amnt)*(1/new_data$actualTerm)*100,0)
TRNPROP = 0.7  #proportion of examples in the training sample
nr<-nrow(new_data)
trnIndex<- sample(1:nr, size = round(TRNPROP * nr), replace=FALSE)

new_data$mths_since_last_delinq[is.na(new_data$mths_since_last_delinq)]<-median(new_d
ata$mths_since_last_delinq,na.rm=TRUE)
new_data$revol_util[is.na(new_data$revol_util)]<-median(new_data$revol_util,na.rm=TRU
E)
new_data$avg_cur_bal[is.na(new_data$avg_cur_bal)]<-median(new_data$avg_cur_bal,na.rm=
TRUE)
new_data$mo_sin_old_il_acct[is.na(new_data$mo_sin_old_il_acct)]<-median(new_data$mo_s
in_old_il_acct,na.rm=TRUE)
new_data$mths_since_recent_bc[is.na(new_data$mths_since_recent_bc)]<-median(new_data$
mths_since_recent_bc,na.rm=TRUE)
new_data$mths_since_recent_inq[is.na(new_data$mths_since_recent_inq)]<-median(new_dat
a$mths_since_recent_inq,na.rm=TRUE)
new_data$num_rev_accts[is.na(new_data$num_rev_accts)]<-median(new_data$num_rev_accts,
na.rm=TRUE)
new_data$pct_tl_nvr_dlq[is.na(new_data$pct_tl_nvr_dlq)]<-median(new_data$pct_tl_nvr_d
lq,na.rm=TRUE)


new_dataTrn <- new_data[trnIndex, ]
new_dataTst <- new_data[-trnIndex, ]


rfModel_Ret <- ranger(actual_return ~., data=subset(new_dataTrn, select=-c(annRet, ac
tualTerm, loan_status)), num.trees =200, importance='permutation')
rfPredRet_trn<- predict(rfModel_Ret, new_dataTrn)
sqrt( mean( (rfPredRet_trn$predictions - new_dataTrn$actual_return)^2) )
```

```
## [1] 0.8674697
```

```
plot ( (predict(rfModel_Ret, new_dataTrn))$predictions, new_dataTrn$actual_return)
```




```
plot ( (predict(rfModel_Ret, new_dataTst))$predictions, new_dataTst$actual_return)
```



40
20
0
-20
new_dataTst$actual_return
-20
-10
0
10
(predict(rfModel_Ret, new_dataTst))$predictions