IDS 572 - Assignment 1 Part A

Joseline Luvena Tanujaya, Sweta Bansal, Vibhanshu

Assignment 1 - Part A

2ai Code Chunk What is the proportion of defaults (‘charged off’ vs ‘fully paid’ loans) in the data? How does default rate vary with loan grade? Does it vary with sub-grade? And is this what you would expect, and why?

```
#Tallying loans  that are fully-paid and charged-off
lcdf %>% group_by(loan_status) %>% tally()
```

```
## # A tibble: 3 x 2
##   loan_status     n
##   <chr>       <int>
## 1 Charged Off 11827
## 2 Current         1
## 3 Fully Paid  69195
```

```
loanstatus_summary <- table(lcdf$loan_status)

# Remove there other values for loan_status other than "Fully Paid' and "Charged Off"?
lcdf <- lcdf %>% filter(loan_status == "Fully Paid" | loan_status == "Charged Off")

#Group loan status by loan grade to see variation within grade
lcdf %>% group_by(grade, loan_status) %>% tally()
```

```
## # A tibble: 14 x 3
## # Groups:   grade [7]
##    grade loan_status     n
##    <chr> <chr>       <int>
##  1 A     Charged Off  1108
```

```
##  2 A     Fully Paid  19294
##  3 B     Charged Off  2682
##  4 B     Fully Paid  20717
##  5 C     Charged Off  4116
##  6 C     Fully Paid  18461
##  7 D     Charged Off  2647
##  8 D     Fully Paid   8155
##  9 E     Charged Off  1045
## 10 E     Fully Paid   2146
## 11 F     Charged Off   191
## 12 F     Fully Paid    369
## 13 G     Charged Off    38
## 14 G     Fully Paid     53
```

```
#Percentage within grade and sub grade:
prop.table(table(lcdf$loan_status, lcdf$grade),margin = 2) #proportion by grade
```

```
##
##                        A         B         C         D         E         F
##   Charged Off 0.0543084 0.1146203 0.1823094 0.2450472 0.3274835 0.3410714
##   Fully Paid  0.9456916 0.8853797 0.8176906 0.7549528 0.6725165 0.6589286
##
##                        G
##   Charged Off 0.4175824
##   Fully Paid  0.5824176
```

```
prop.table(table(lcdf$loan_status, lcdf$sub_grade),margin = 2) #proportion by sub_grade
```

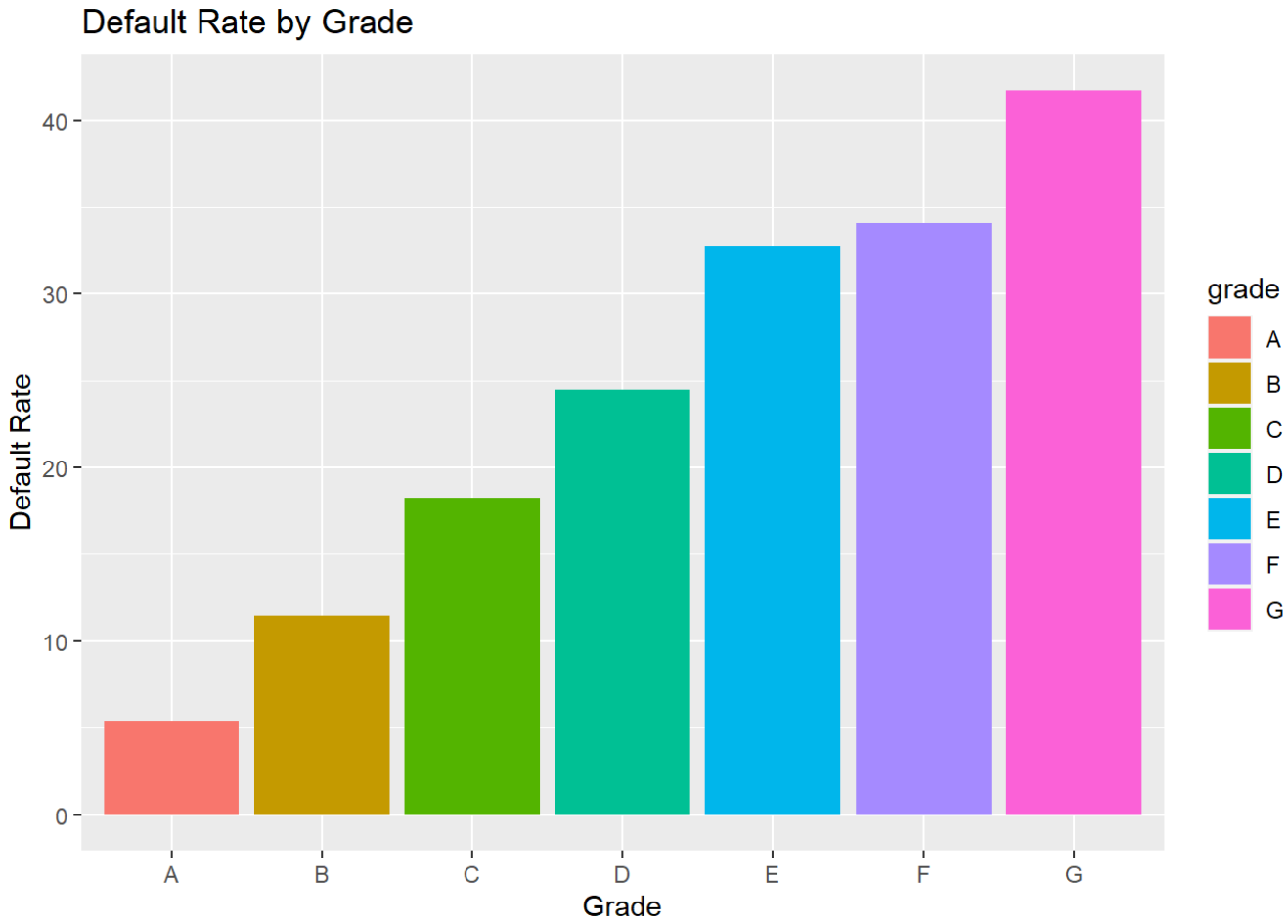
```
##
##                        A1         A2         A3         A4         A5         B1
##   Charged Off 0.03143614 0.04605080 0.04340474 0.05806985 0.07634011 0.09046811
##   Fully Paid  0.96856386 0.95394920 0.95659526 0.94193015 0.92365989 0.90953189
##
##                        B2         B3         B4         B5         C1         C2
```

```
##   Charged Off 0.10403997 0.11535955 0.11797753 0.13839368 0.15582255 0.17738142
##   Fully Paid  0.89596003 0.88464045 0.88202247 0.86160632 0.84417745 0.82261858
##
##                       C3         C4         C5         D1         D2         D3
##   Charged Off 0.19431593 0.19685243 0.19808219 0.23010323 0.24949860 0.23402299
##   Fully Paid  0.80568407 0.80314757 0.80191781 0.76989677 0.75050140 0.76597701
##
##                       D4         D5         E1         E2         E3         E4
##   Charged Off 0.25432526 0.27487473 0.31613508 0.32094176 0.32474227 0.34515366
##   Fully Paid  0.74567474 0.72512527 0.68386492 0.67905824 0.67525773 0.65484634
##
##                       E5         F1         F2         F3         F4         F5
##   Charged Off 0.36421725 0.32178218 0.37190083 0.31428571 0.32467532 0.41818182
##   Fully Paid  0.63578275 0.67821782 0.62809917 0.68571429 0.67532468 0.58181818
##
##                       G1         G2         G3         G4         G5
##   Charged Off 0.42857143 0.40740741 0.26666667 0.80000000 0.50000000
##   Fully Paid  0.57142857 0.59259259 0.73333333 0.20000000 0.50000000
```

```
#The default rate by graph
graph <- lcdf %>% group_by(grade) %>% summarise(Count = n(), DefaultRate = (sum(loan_status
== "Charged Off")/Count)*100)
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
ggplot(graph) + aes(x = grade, y = DefaultRate, fill = grade) + geom_bar(stat =
"identity") + xlab("Grade") + ylab("Default Rate") + ggtitle("Default Rate by Grade")
```



#2a i Analysis The default rate increases as grade gets worse from A to G. The same pattern holds true for sub-grade. With the exception of G3 that breaks the pattern (outlier), where the Charged Off rate is similar to those of D4 or D5. This aligns with our expectations, because Grades are assigned based on the borrower's likelihood to pay their debt.

2a ii Code Chunk How many loans are there in each grade? And do loan amounts vary by grade? Does interest rate for loans vary with grade, subgrade? Look at the average, standard-deviation, min and max of interest rate by grade and subgrade. Is this what you expect, and why?

```
# By grade:
summarise_by_grade <- lcdf %>% group_by(grade) %>% summarise(nLoans=n(), defaults=sum(loan_status=="Charged Off"
```

```
), avgInterest= mean(int_rate), stdInterest=sd(int_rate),minInterest=min(int_rate), maxInterest=max(int_rate), av
gLoanAMt=mean(loan_amnt))
```

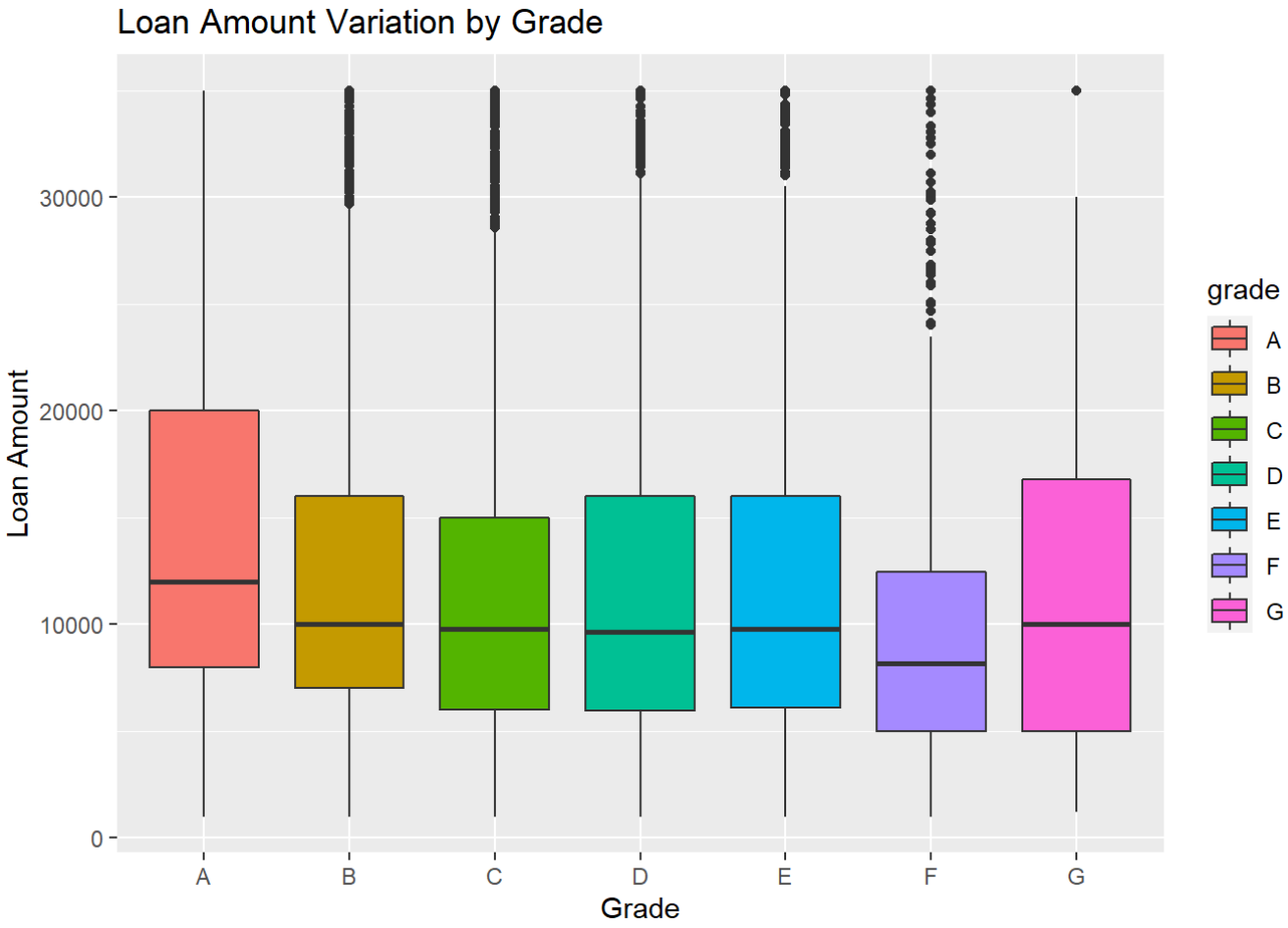
```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
# By sub-grade:
lcdf %>% group_by(sub_grade) %>% summarise(nLoans=n(), defaults=sum(loan_status=="Charged Off"), avgInterest= mea
n(int_rate), stdInterest=sd(int_rate),minInterest=min(int_rate), maxInterest=max(int_rate), avgLoanAMt=mean(loan_
amnt))
```

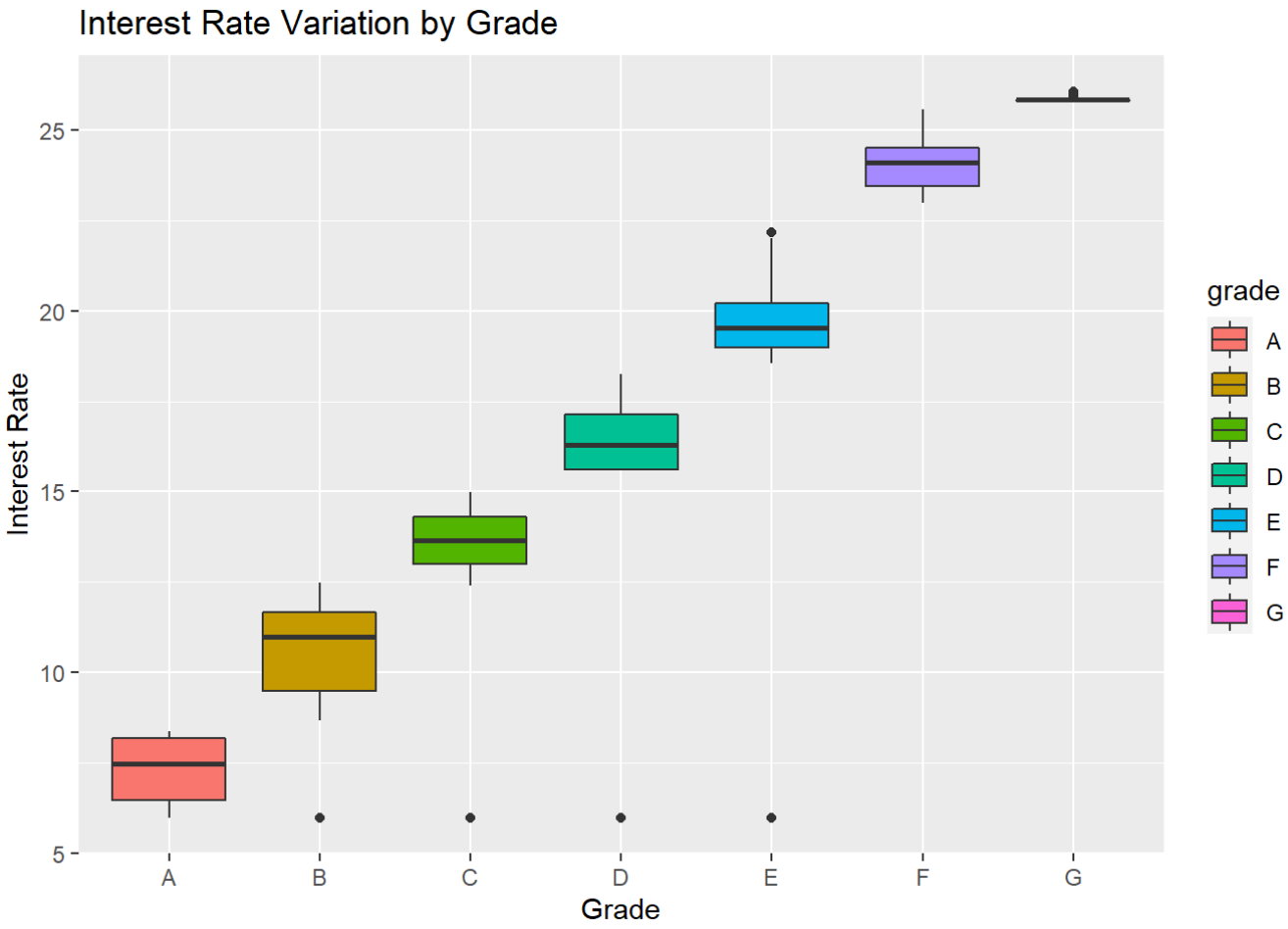
```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 35 x 8
##    sub_grade nLoans defaults avgInterest stdInterest minInterest maxInterest
##    <chr>      <int>    <int>       <dbl>       <dbl>       <dbl>       <dbl>
##  1 A1          3022       95        6.03    0.000546        6           6.03
##  2 A2          3583      165        6.49    0               6.49        6.49
##  3 A3          3548      154        7.05    0.0650          6.99        7.12
##  4 A4          4839      281        7.58    0.0996          7.49        7.69
##  5 A5          5410      413        8.27    0.0971          8.19        8.39
##  6 B1          4123      373        8.88    0.251           6           9.17
##  7 B2          4604      479        9.77    0.326           9.49       10.2
##  8 B3          4603      531       10.7     0.248          10.5        11.0
##  9 B4          4628      546       11.5     0.162           6          11.7
## 10 B5          5441      753       12.2     0.260           6          12.5
## # ... with 25 more rows, and 1 more variable: avgLoanAMt <dbl>
```

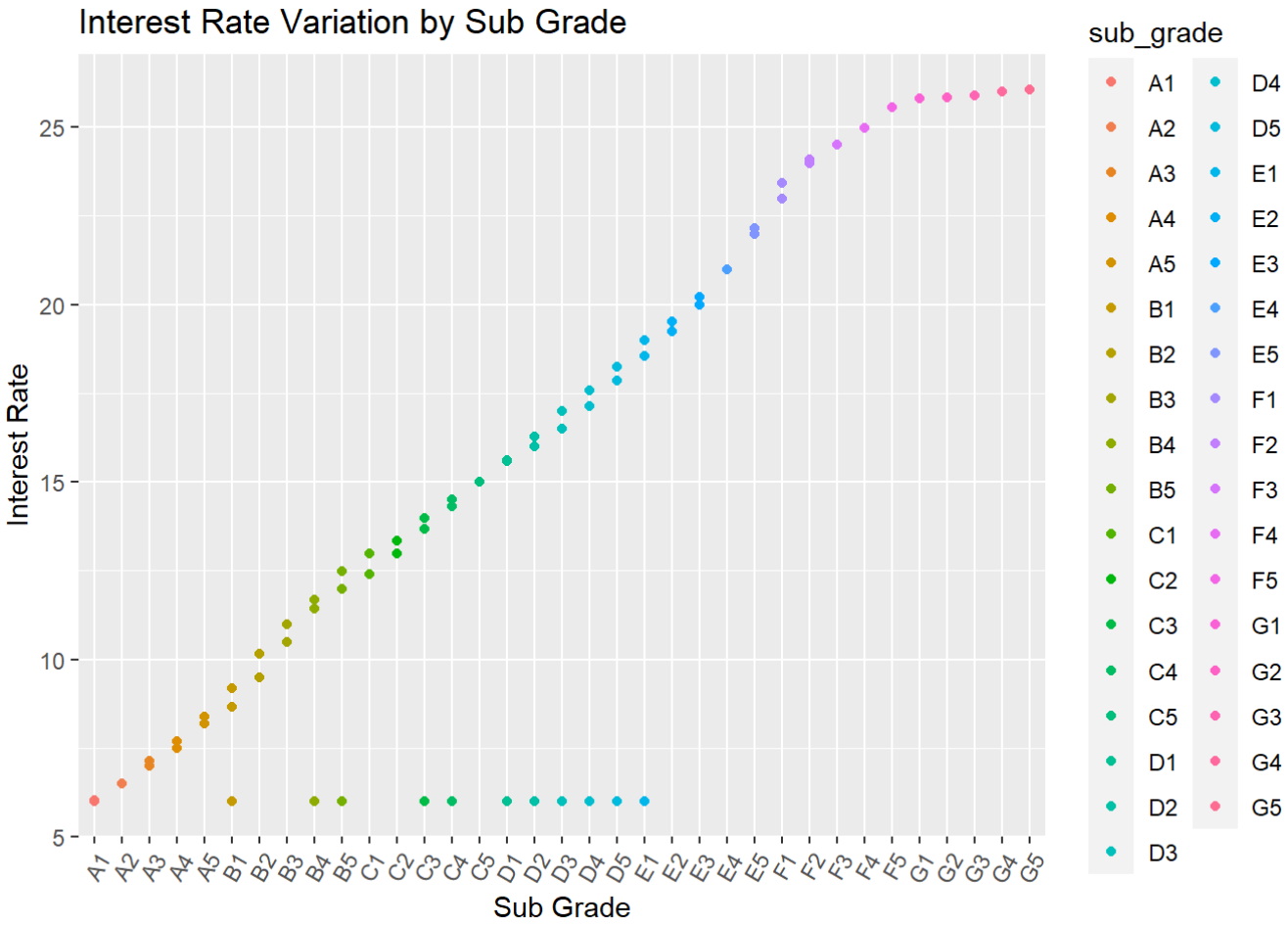
```
ggplot(lcdf) + aes(x = grade, y = loan_amnt, fill = grade) + geom_boxplot() + xlab("Grade") + ylab("Loan Amount")
+ ggtitle("Loan Amount Variation by Grade")
```




```
ggplot(lcdf) + aes(x = grade, y = int_rate, fill = grade) + geom_boxplot() + xlab("Grade") + ylab("Interest Rate"
) + ggtitle("Interest Rate Variation by Grade")
```




```
ggplot(lcdf) + aes(x = sub_grade, y = int_rate, colour= sub_grade) + geom_point() + xlab("Sub Grade") + ylab("Int
erest Rate") + ggtitle("Interest Rate Variation by Sub Grade") + theme(axis.text.x = element_text(angle = 60, hju
st = 1))
```



2a ii Analysis: Loan amounts vary by grade but there is no pattern. The difference between neighboring grades is within 20%. As grade/sub grade goes worse from A1 to G5, which aligns with our expectations, interest rate increases with higher default rates (riskier investments). Looking at the min and max values Within sub grades, we can see that there are loans that break this pattern (for example loans in B4 have smaller interest rate than A4).

2a iii Code Chunk: What are people borrowing money for (purpose)? Examine how many loans, average amounts, etc. by purpose? And within grade? Do defaults vary by purpose?

```
lcdf %>% group_by(purpose) %>% summarise(nLoans=n(), avgLoanAMt=mean(loan_amnt))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 13 x 3
##    purpose            nLoans avgLoanAMt
##    <chr>               <int>      <dbl>
##  1 car                   719      7820.
##  2 credit_card         18780     13501.
##  3 debt_consolidation  48647     13008.
##  4 home_improvement     3942     11707.
##  5 house                 254     12505.
##  6 major_purchase       1402     10172.
##  7 medical               900      6981.
##  8 moving                604      6542.
##  9 other                4455      8271.
## 10 renewable_energy       65      9829.
## 11 small_business        759     14425.
## 12 vacation              492      5872.
## 13 wedding                 3      7867.
```

```
lcdf_purpose_1 <- lcdf %>% group_by(grade) %>% summarise(nLoans=n(), avgLoanAMt=mean(loan_amnt))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
lcdf %>% group_by(grade, purpose) %>% summarise(nLoans=n(), avgLoanAMt=mean(loan_amnt))
```

```
## `summarise()` regrouping output by 'grade' (override with `.groups` argument)
```

```
## # A tibble: 84 x 4
## # Groups:   grade [7]
##    grade purpose            nLoans avgLoanAMt
##    <chr> <chr>               <int>      <dbl>
##  1 A     car                   194      8364.
```

```
##  2 A     credit_card          7487     14683.
##  3 A     debt_consolidation  10824     14294.
##  4 A     home_improvement     1043     12712.
##  5 A     house                   8     20075
##  6 A     major_purchase        361     10577.
##  7 A     medical                73      9566.
##  8 A     moving                  8     10262.
##  9 A     other                 338     10756.
## 10 A     renewable_energy        2      5250
## # ... with 74 more rows
```

```
# find the distribution of grades within purposes:
table(lcdf$purpose, lcdf$loan_status)
```

```
##
##                     Charged Off Fully Paid
##   car                        75        644
##   credit_card              2326      16454
##   debt_consolidation       7423      41224
##   home_improvement          503       3439
##   house                      48        206
##   major_purchase            220       1182
##   medical                   141        759
##   moving                    132        472
##   other                     702       3753
##   renewable_energy           16         49
##   small_business            169        590
##   vacation                   72        420
##   wedding                     0          3
```

```
prop.table(table(lcdf$purpose, lcdf$loan_status), 1) # % by purpose
```

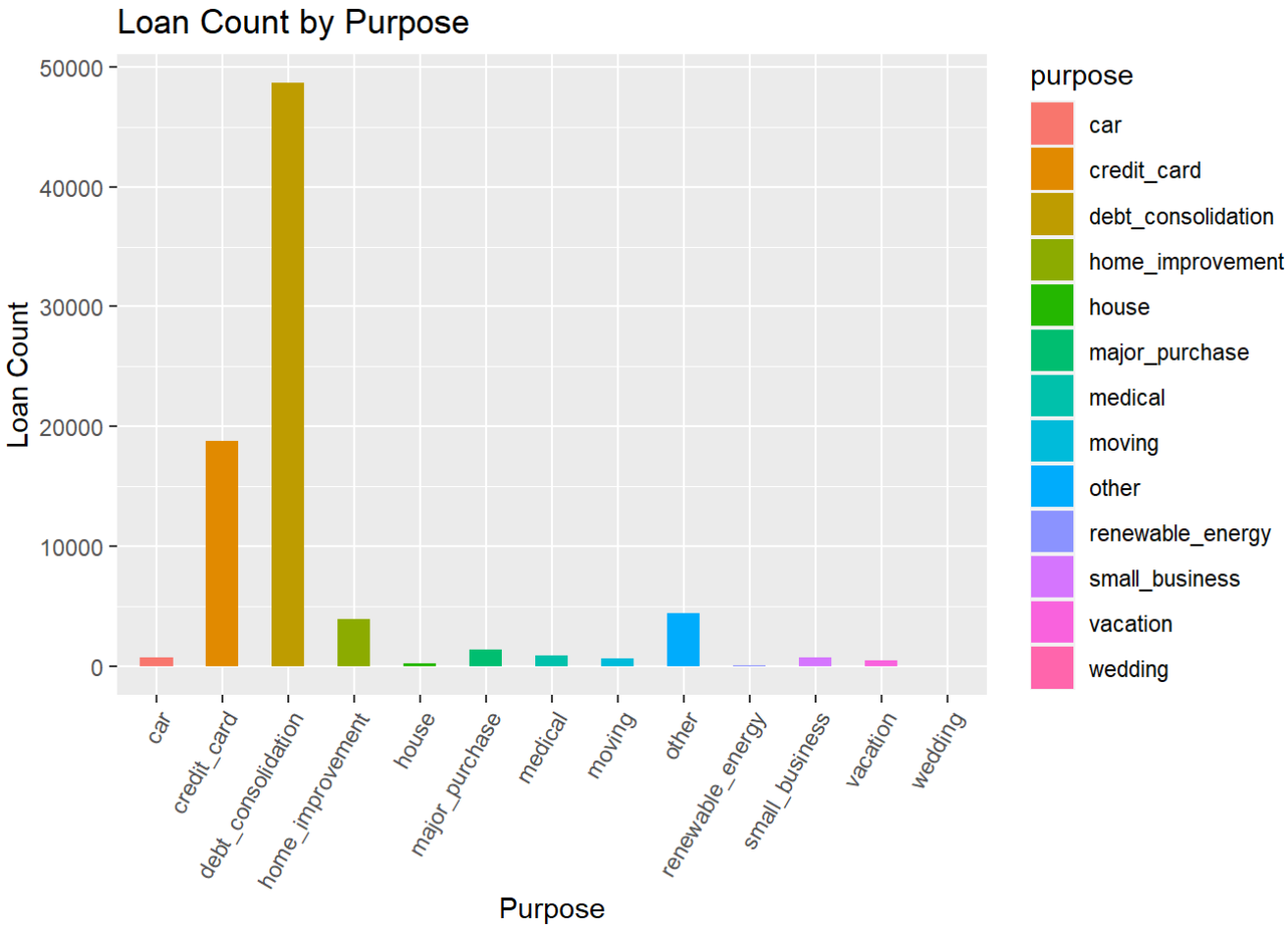
```
##
##                     Charged Off Fully Paid
```

```
##   car                  0.1043115  0.8956885
##   credit_card          0.1238552  0.8761448
##   debt_consolidation   0.1525891  0.8474109
##   home_improvement     0.1276002  0.8723998
##   house                0.1889764  0.8110236
##   major_purchase       0.1569187  0.8430813
##   medical              0.1566667  0.8433333
##   moving               0.2185430  0.7814570
##   other                0.1575758  0.8424242
##   renewable_energy     0.2461538  0.7538462
##   small_business       0.2226614  0.7773386
##   vacation             0.1463415  0.8536585
##   wedding              0.0000000  1.0000000
```

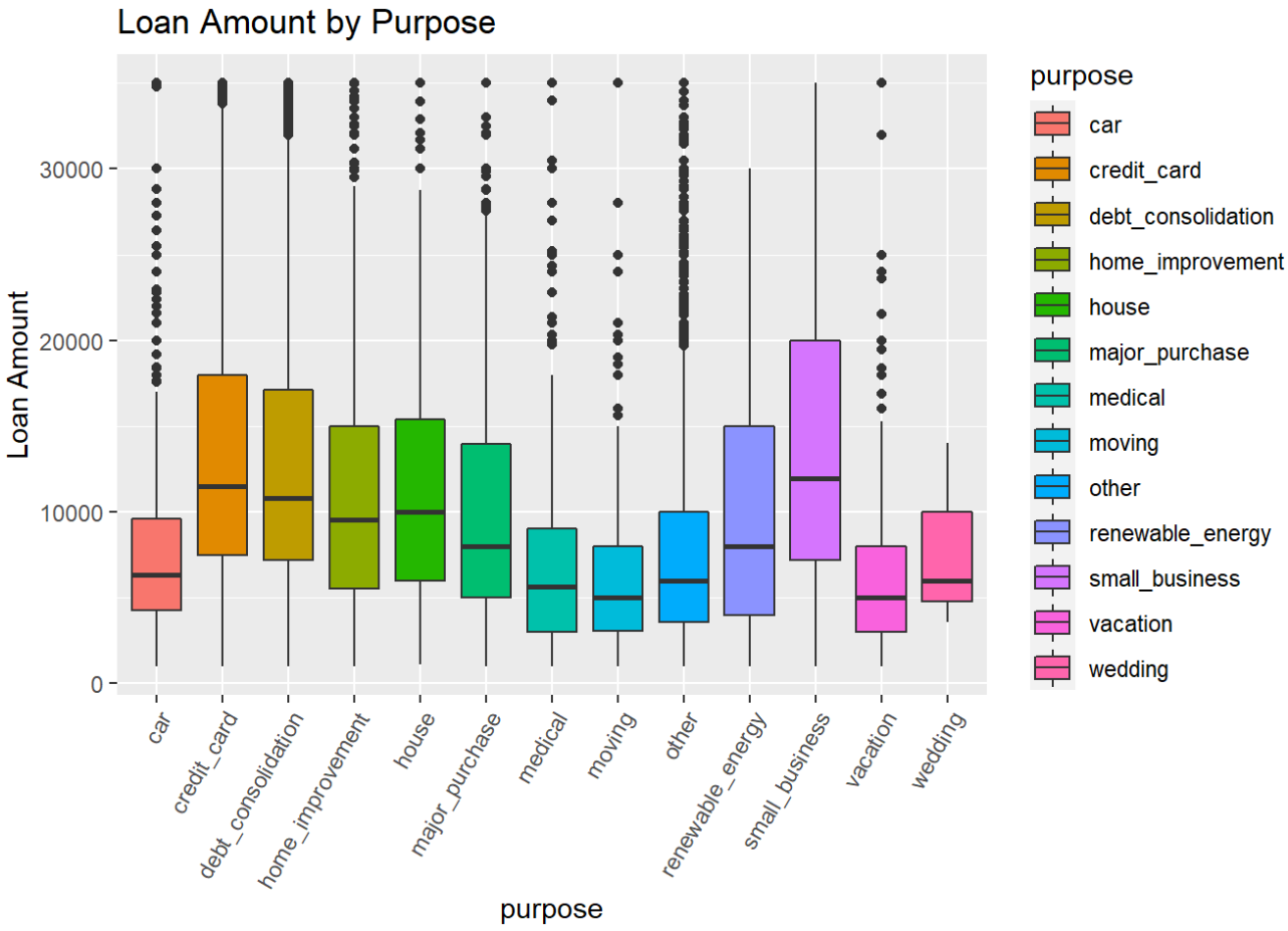
```
prop.table(table(lcdf$purpose, lcdf$loan_status), 2) # % by loan status
```

```
##
##                       Charged Off   Fully Paid
##   car                6.341422e-03 9.307031e-03
##   credit_card        1.966686e-01 2.377917e-01
##   debt_consolidation 6.276317e-01 5.957656e-01
##   home_improvement   4.252980e-02 4.970012e-02
##   house              4.058510e-03 2.977094e-03
##   major_purchase     1.860151e-02 1.708216e-02
##   medical            1.192187e-02 1.096900e-02
##   moving             1.116090e-02 6.821302e-03
##   other              5.935571e-02 5.423802e-02
##   renewable_energy   1.352837e-03 7.081437e-04
##   small_business     1.428934e-02 8.526628e-03
##   vacation           6.087765e-03 6.069803e-03
##   wedding            0.000000e+00 4.335573e-05
```

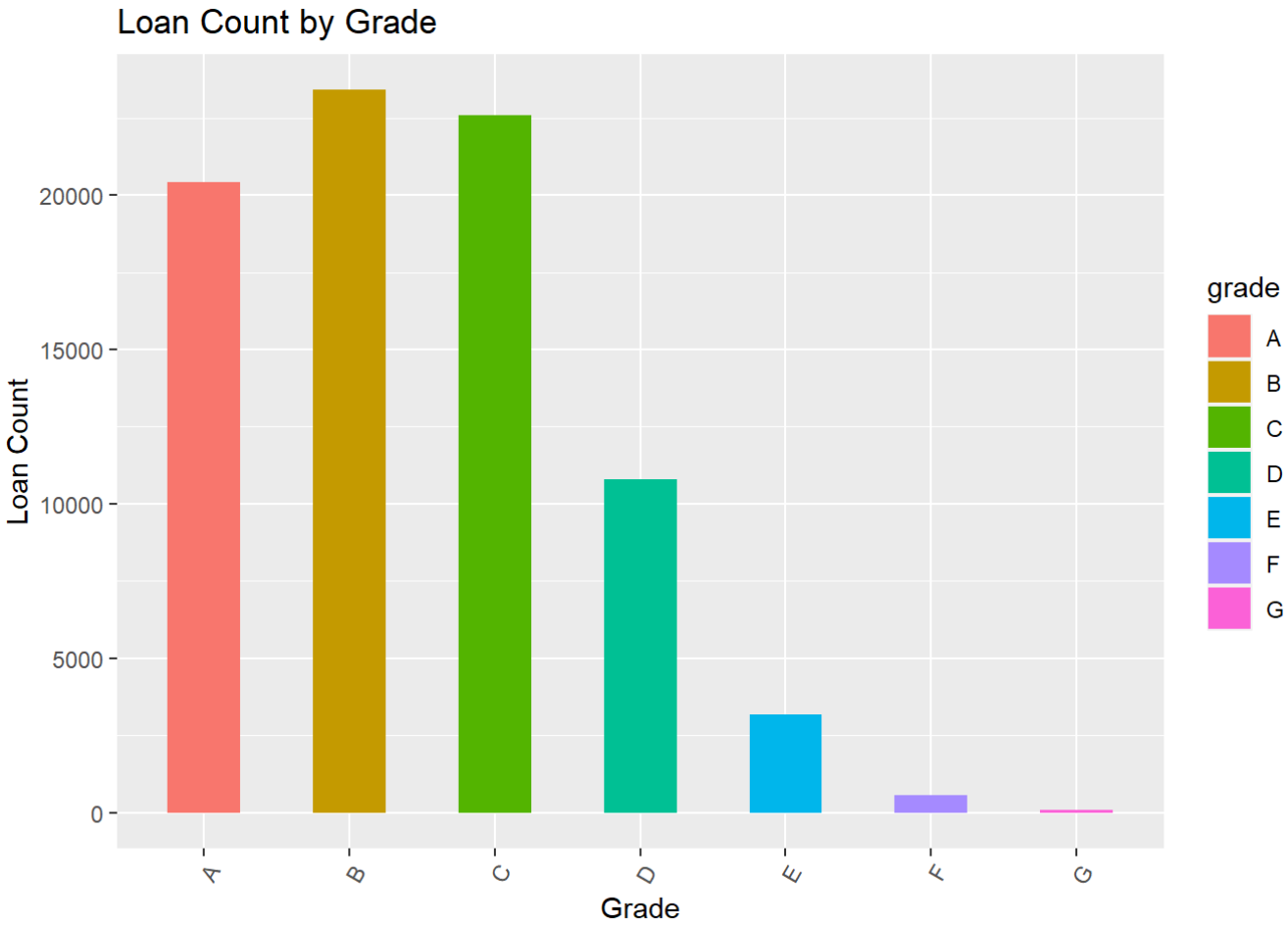
```
#The loan count by purpose
ggplot(lcdf, aes(x = purpose, fill = purpose)) + geom_bar(width = 0.5) + xlab("Purpose") +
ylab("Loan Count") + theme(axis.text.x = element_text(angle = 60, hjust = 1)) + ggtitle("Loan Count by Purpose")
```



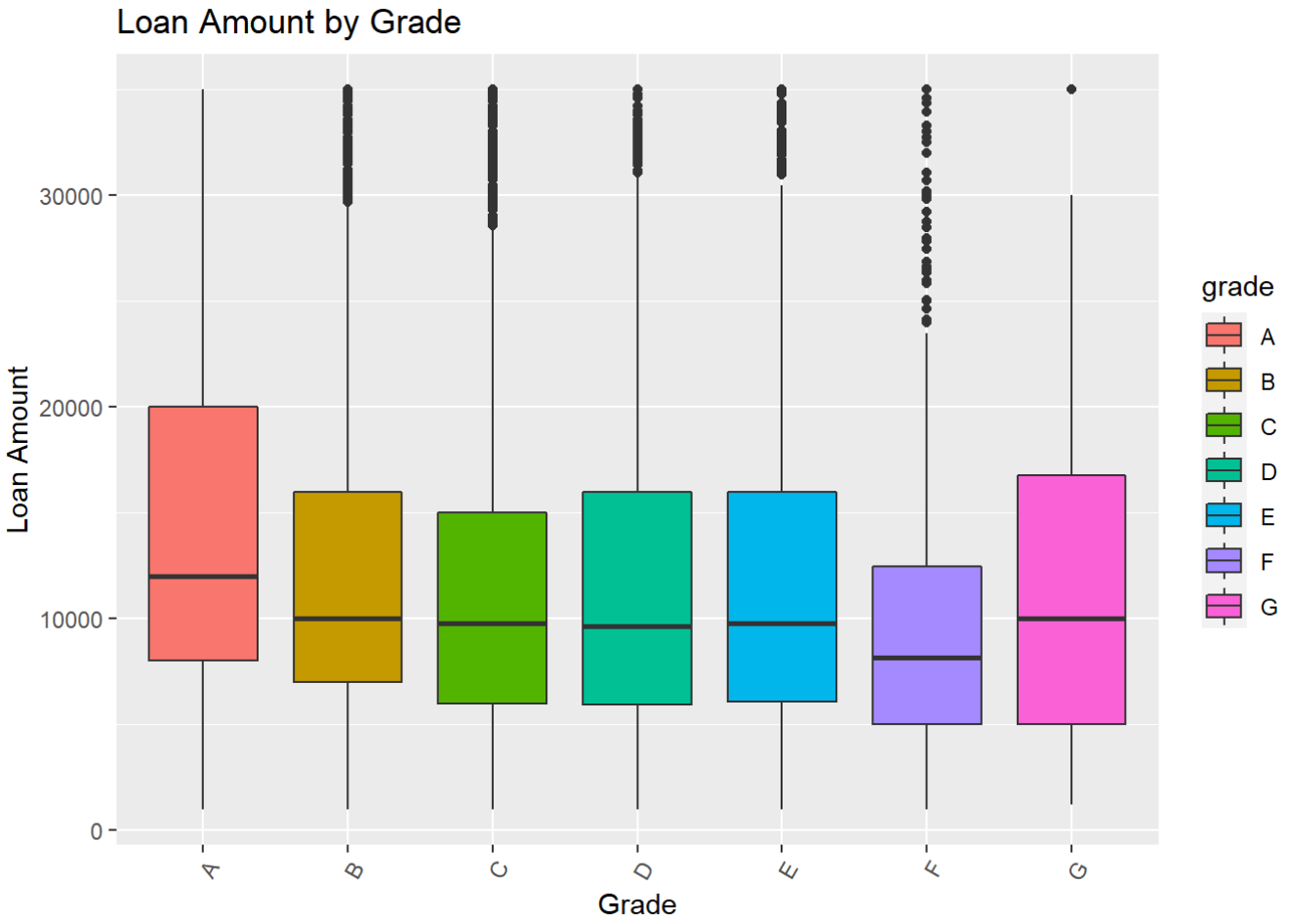

```
#The loan amount by purpose
ggplot(lcdf) + aes(x = purpose, y = loan_amnt, fill = purpose) + geom_boxplot() + xlab("purpose") + ylab("Loan Am
ount") + ggtitle("Loan Amount by Purpose") + theme(axis.text.x = element_text(angle = 60, hjust = 1))
```




```
#The loan count by grade
ggplot(lcdf, aes(x = grade, fill = grade)) + geom_bar(width = 0.5) + xlab("Grade") +
ylab("Loan Count") + theme(axis.text.x = element_text(angle = 60, hjust = 1)) + ggtitle("Loan Count by Grade")
```




```
#The loan amount by grade
ggplot(lcdf) + aes(x = grade, y = loan_amnt, fill = grade) + geom_boxplot() + xlab("Grade") + ylab("Loan Amount")
+ ggtitle("Loan Amount by Grade") + theme(axis.text.x = element_text(angle = 60, hjust = 1))
```

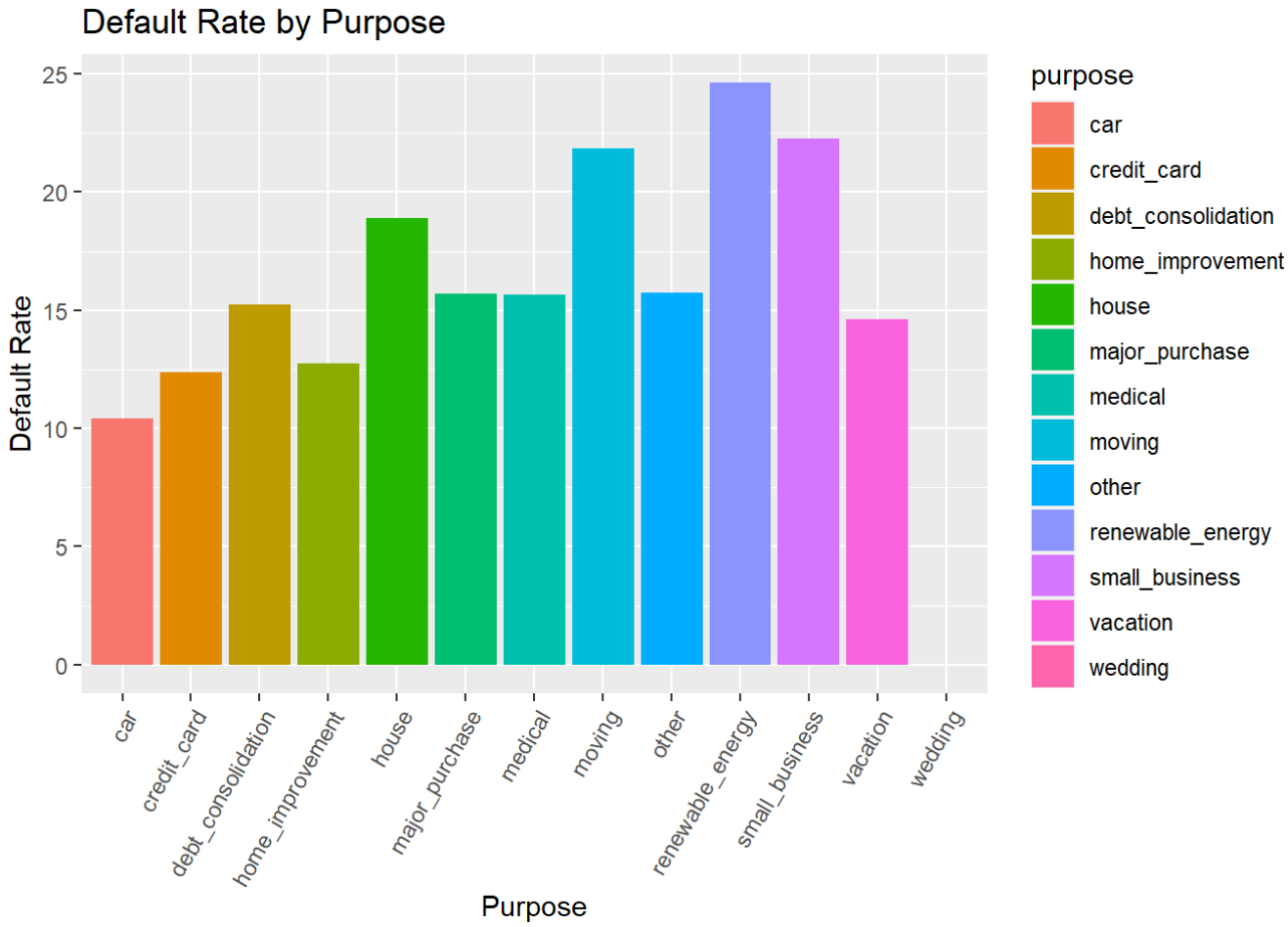



```
#The default rate by purpose
lcdf_by_purpose <- lcdf %>% group_by(purpose) %>% summarise(Count = n(), DefaultRate = (sum(loan_status
== "Charged Off")/Count)*100)
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
ggplot(lcdf_by_purpose) + aes(x = purpose, y = DefaultRate, fill = purpose) + geom_bar(stat =
"identity") + xlab("Purpose") + ylab("Default Rate") + ggtitle("Default Rate by Purpose") +theme(axis.text.x = el
```

```
ement_text(angle = 60, hjust = 1))
```



2a iii Analysis People are mostly borrowing for debt consolidation. Default rates do vary by purpose. The highest default rate is on renewable energy, small business, and moving.

2a iv Code Chunk: For loans which are fully paid back, how does the time-to-full-payoff vary? For this, calculate the 'actual term' (issue-date to last-payment-date) for all loans. How does this actual-term vary by loan grade (a box-plot can help visualize this)

```
#Term of the loan is the duration between the last-payment-date and the loan issue-date.
# check the format of these two columns with date values
```

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

```
## # A tibble: 6 x 2
##   last_pymnt_d issue_d
##   <chr>        <date>
## 1 Dec-2015     2015-01-01
## 2 Mar-2018     2015-01-01
## 3 Feb-2018     2015-01-01
## 4 Jan-2018     2015-01-01
## 5 Nov-2017     2015-01-01
## 6 Nov-2015     2015-01-01
```

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

```
## Warning: 45 failed to parse.
```

```
# for defaulted loans, set the actual-term at 3 years.
lcdf$actualTerm <- ifelse(lcdf$loan_status=="Fully Paid", as.duration(lcdf$issue_d  %--% lcdf$last_pymnt_d)/dyear
s(1), 3)

lcdf_paid <- subset(lcdf, loan_status == "Fully Paid")
str(lcdf_paid)
```

```
## tibble [69,195 x 147] (S3: tbl_df/tbl/data.frame)
##  $ X1                                        : num [1:69195] 1 2 3 4 5 6 7 9 11 12 ...
##  $ id                                        : logi [1:69195] NA NA NA NA NA NA ...
##  $ member_id                                 : logi [1:69195] NA NA NA NA NA NA ...
##  $ loan_amnt                                 : num [1:69195] 5000 17000 3500 14000 1400 9000 19200 12000 3000
14000 ...
##  $ funded_amnt                               : num [1:69195] 5000 17000 3500 14000 1400 9000 19200 12000 3000
14000 ...
```

```
##  $ funded_amnt_inv                       : num [1:69195] 5000 17000 3500 14000 1400 9000 19200 12000 3000
14000 ...
##  $ term                                  : chr [1:69195] "36 months" "36 months" "36 months" "36 months"
...
##  $ int_rate                              : num [1:69195] 12.39 12.39 7.49 11.99 12.99 ...
##  $ installment                           : num [1:69195] 167 567.8 108.9 464.9 47.2 ...
##  $ grade                                 : chr [1:69195] "C" "C" "A" "B" ...
##  $ sub_grade                             : chr [1:69195] "C1" "C1" "A4" "B5" ...
##  $ emp_title                             : chr [1:69195] "Trooper" "export agent" "Associate Director" "fi
nancial counselor" ...
##  $ emp_length                            : chr [1:69195] "< 1 year" "1 year" "10+ years" "6 years" ...
##  $ home_ownership                        : chr [1:69195] "RENT" "RENT" "RENT" "RENT" ...
##  $ annual_inc                            : num [1:69195] 48000 53000 72000 44000 23000 39000 93500 40000 4
0000 64000 ...
##  $ verification_status                   : chr [1:69195] "Not Verified" "Not Verified" "Not Verified" "Not
Verified" ...
##  $ issue_d                               : Date[1:69195], format: "2015-01-01" "2015-01-01" ...
##  $ loan_status                           : chr [1:69195] "Fully Paid" "Fully Paid" "Fully Paid" "Fully Pai
d" ...
##  $ pymnt_plan                            : chr [1:69195] "n" "n" "n" "n" ...
##  $ url                                   : logi [1:69195] NA NA NA NA NA NA ...
##  $ desc                                  : logi [1:69195] NA NA NA NA NA NA ...
##  $ purpose                               : chr [1:69195] "debt_consolidation" "debt_consolidation" "debt_c
onsolidation" "debt_consolidation" ...
##  $ title                                 : chr [1:69195] "Debt consolidation" "Debt consolidation" "Debt c
onsolidation" "Debt consolidation" ...
##  $ zip_code                              : chr [1:69195] "437xx" "902xx" "100xx" "606xx" ...
##  $ addr_state                            : chr [1:69195] "OH" "CA" "NY" "IL" ...
##  $ dti                                   : num [1:69195] 14.2 21.3 14.3 18.4 17.8 ...
##  $ delinq_2yrs                           : num [1:69195] 0 0 0 0 0 0 0 0 0 0 ...
##  $ earliest_cr_line                      : chr [1:69195] "Jul-2010" "Sep-2006" "Apr-1995" "Apr-2001" ...
##  $ inq_last_6mths                        : num [1:69195] 0 0 1 0 0 1 0 0 1 2 ...
##  $ mths_since_last_delinq                : num [1:69195] NA NA NA 48 NA NA NA 34 34 NA ...
##  $ mths_since_last_record                : num [1:69195] NA NA NA NA NA NA NA NA NA NA ...
##  $ open_acc                              : num [1:69195] 7 12 13 11 8 10 14 10 7 8 ...
##  $ pub_rec                               : num [1:69195] 0 0 0 0 0 0 0 0 0 0 ...
##  $ revol_bal                             : num [1:69195] 5994 14690 30063 14330 6981 ...
```

```
##  $ revol_util                    : num [1:69195] 44.4 73.1 86.1 34.8 74.3 77.7 43.4 42 43.2 89 ...
##  $ total_acc                     : num [1:69195] 10 22 22 15 12 11 23 15 8 14 ...
##  $ initial_list_status           : chr [1:69195] "f" "f" "f" "f" ...
##  $ out_prncp                     : num [1:69195] 0 0 0 0 0 0 0 0 0 0 ...
##  $ out_prncp_inv                 : num [1:69195] 0 0 0 0 0 0 0 0 0 0 ...
##  $ total_pymnt                   : num [1:69195] 5475 20452 3916 16593 1694 ...
##  $ total_pymnt_inv               : num [1:69195] 5475 20452 3916 16593 1694 ...
##  $ total_rec_prncp               : num [1:69195] 5000 17000 3500 14000 1400 9000 19200 12000 3000
14000 ...
##  $ total_rec_int                 : num [1:69195] 475 3424 416 2593 294 ...
##  $ total_rec_late_fee            : num [1:69195] 0 28.4 0 0 0 ...
##  $ recoveries                    : num [1:69195] 0 0 0 0 0 0 0 0 0 0 ...
##  $ collection_recovery_fee       : num [1:69195] 0 0 0 0 0 0 0 0 0 0 ...
##  $ last_pymnt_d                  : POSIXct[1:69195], format: "2015-12-01" "2018-03-01" ...
##  $ last_pymnt_amnt               : num [1:69195] 3978.93 5.59 108.69 338.66 186.55 ...
##  $ next_pymnt_d                  : logi [1:69195] NA NA NA NA NA NA ...
##  $ last_credit_pull_d            : chr [1:69195] "Feb-2019" "Mar-2018" "Feb-2019" "Dec-2017" ...
##  $ collections_12_mths_ex_med    : num [1:69195] 0 0 0 0 0 0 0 0 0 0 ...
##  $ mths_since_last_major_derog   : num [1:69195] NA NA NA NA NA NA NA NA 34 NA ...
##  $ policy_code                   : num [1:69195] 1 1 1 1 1 1 1 1 1 1 ...
##  $ application_type              : chr [1:69195] "Individual" "Individual" "Individual" "Individua
l" ...
##  $ annual_inc_joint              : logi [1:69195] NA NA NA NA NA NA ...
##  $ dti_joint                     : logi [1:69195] NA NA NA NA NA NA ...
##  $ verification_status_joint     : logi [1:69195] NA NA NA NA NA NA ...
##  $ acc_now_delinq                : num [1:69195] 0 0 0 0 0 0 0 0 0 0 ...
##  $ tot_coll_amt                  : num [1:69195] 0 0 0 455 0 0 0 100 0 0 ...
##  $ tot_cur_bal                   : num [1:69195] 19022 39917 30063 79584 28063 ...
##  $ open_acc_6m                   : logi [1:69195] NA NA NA NA NA NA ...
##  $ open_act_il                   : logi [1:69195] NA NA NA NA NA NA ...
##  $ open_il_12m                   : logi [1:69195] NA NA NA NA NA NA ...
##  $ open_il_24m                   : logi [1:69195] NA NA NA NA NA NA ...
##  $ mths_since_rcnt_il            : logi [1:69195] NA NA NA NA NA NA ...
##  $ total_bal_il                  : logi [1:69195] NA NA NA NA NA NA ...
##  $ il_util                       : logi [1:69195] NA NA NA NA NA NA ...
##  $ open_rv_12m                   : logi [1:69195] NA NA NA NA NA NA ...
##  $ open_rv_24m                   : logi [1:69195] NA NA NA NA NA NA ...
```

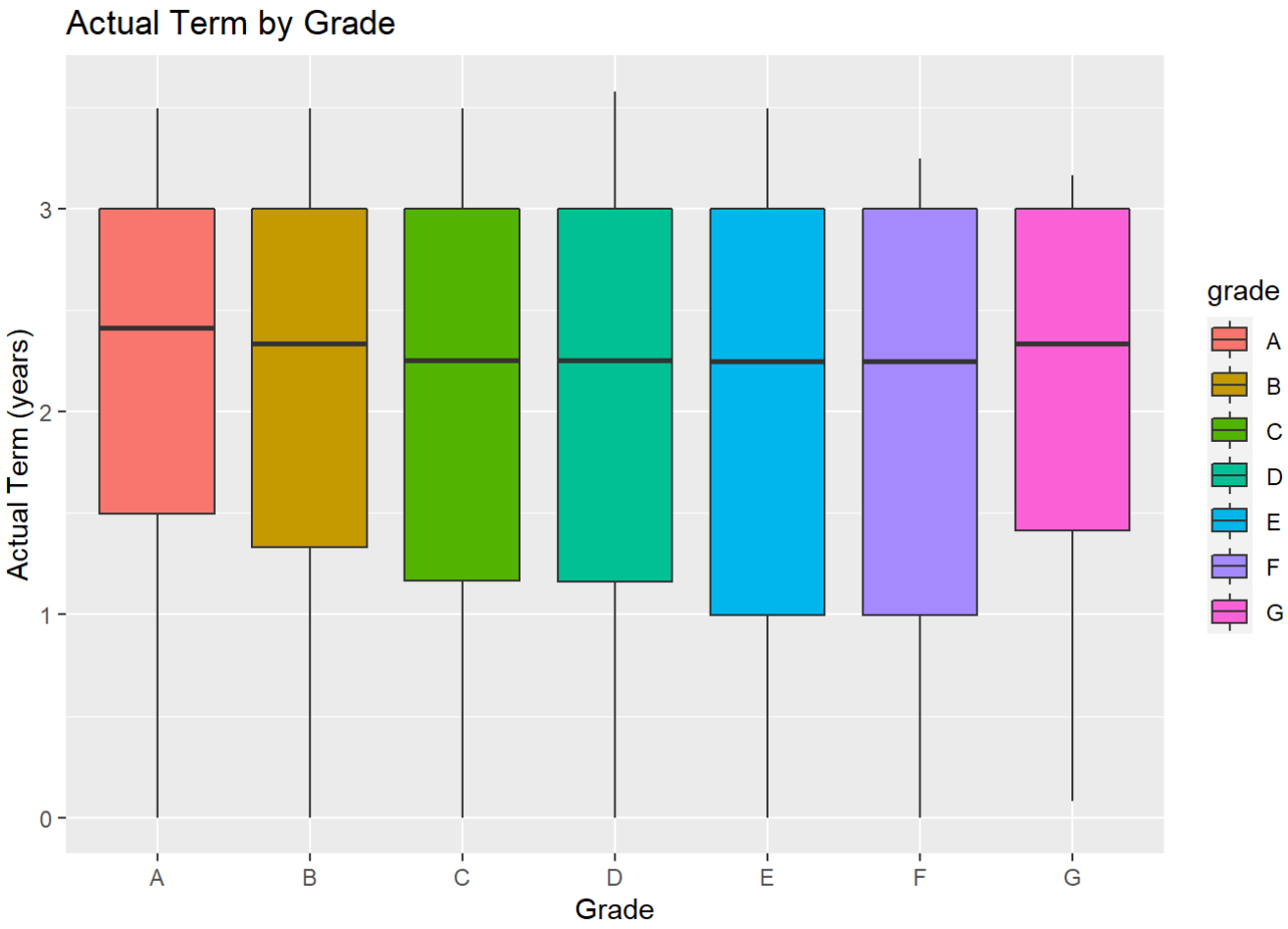
```
##  $ max_bal_bc                                : logi [1:69195] NA NA NA NA NA NA ...
##  $ all_util                                  : logi [1:69195] NA NA NA NA NA NA ...
##  $ total_rev_hi_lim                          : num [1:69195] 13500 20100 34900 41200 9400 33700 66400 16200 54
00 16300 ...
##  $ inq_fi                                    : logi [1:69195] NA NA NA NA NA NA ...
##  $ total_cu_tl                               : logi [1:69195] NA NA NA NA NA NA ...
##  $ inq_last_12m                              : logi [1:69195] NA NA NA NA NA NA ...
##  $ acc_open_past_24mths                      : num [1:69195] 7 2 4 2 4 7 6 4 6 4 ...
##  $ avg_cur_bal                               : num [1:69195] 3170 3629 2313 7235 3508 ...
##  $ bc_open_to_buy                            : num [1:69195] 610 3279 359 23870 359 ...
##  $ bc_util                                   : num [1:69195] 81.5 80.9 98.7 37.5 91 86.7 47.6 75.2 56.1 87.9
...
##  $ chargeoff_within_12_mths                  : num [1:69195] 0 0 0 0 0 0 0 0 0 0 ...
##  $ delinq_amnt                               : num [1:69195] 0 0 0 0 0 0 0 0 0 0 ...
##  $ mo_sin_old_il_acct                        : num [1:69195] 29 100 44 100 49 10 190 64 32 147 ...
##  $ mo_sin_old_rev_tl_op                      : num [1:69195] 54 84 237 165 54 126 226 67 20 143 ...
##  $ mo_sin_rcnt_rev_tl_op                     : num [1:69195] 2 23 6 15 10 2 1 1 2 4 ...
##  $ mo_sin_rcnt_tl                            : num [1:69195] 2 3 6 15 10 2 1 1 2 4 ...
##  $ mort_acc                                  : num [1:69195] 0 0 0 0 0 0 0 0 0 3 ...
##  $ mths_since_recent_bc                      : num [1:69195] 18 23 36 27 39 2 14 17 2 4 ...
##  $ mths_since_recent_bc_dlq                  : num [1:69195] NA NA NA NA NA NA NA NA NA NA ...
##  $ mths_since_recent_inq                     : num [1:69195] 4 NA 6 15 11 6 4 10 2 4 ...
##  $ mths_since_recent_revol_delinq            : num [1:69195] NA NA NA NA NA NA NA 40 NA NA ...
##  $ num_accts_ever_120_pd                     : num [1:69195] 0 0 0 0 0 0 0 0 1 0 ...
##  $ num_actv_bc_tl                            : num [1:69195] 2 7 8 3 2 4 2 5 2 4 ...
##  $ num_actv_rev_tl                           : num [1:69195] 4 9 12 3 6 4 5 7 5 5 ...
##  $ num_bc_sats                               : num [1:69195] 2 8 8 4 2 6 4 5 2 4 ...
##  $ num_bc_tl                                 : num [1:69195] 2 13 15 4 3 6 7 6 2 5 ...
##  $ num_il_tl                                 : num [1:69195] 4 6 1 6 3 1 5 6 2 5 ...
##  $ num_op_rev_tl                             : num [1:69195] 6 10 13 5 6 8 11 8 5 5 ...
##  $ num_rev_accts                             : num [1:69195] 6 16 21 5 9 9 18 9 5 6 ...
##  $ num_rev_tl_bal_gt_0                       : num [1:69195] 4 9 12 3 6 4 5 7 5 5 ...
##   [list output truncated]
##  - attr(*, "problems")= tibble [1 x 5] (S3: tbl_df/tbl/data.frame)
##   ..$ row     : int 15013
##   ..$ col     : chr "next_pymnt_d"
##   ..$ expected: chr "1/0/T/F/TRUE/FALSE"
```

```
##   ..$ actual  : chr "Feb-2018"
##   ..$ file    : chr "'lcDataSample5m.csv'"
```

```
lcdf_paid_days <- lcdf_paid %>% group_by(grade) %>% summarise(nLoans=n(), actTerm= (mean(actualTerm)*365))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
# sd(lcdf_paid$actTerm)
library(ggplot2)
ggplot(lcdf_paid, aes(x=grade, y=actualTerm, fill=grade)) +
    geom_boxplot() + xlab("Grade") + ylab("Actual Term (years)") + ggtitle("Actual Term by Grade")
```




```
lcdf_paid %>% group_by(grade) %>% summarise(avgActTerm= mean(actualTerm), minActTerm=min(actualTerm), maxActTerm=
max(actualTerm))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 7 x 4
##   grade avgActTerm minActTerm maxActTerm
##   <chr>      <dbl>      <dbl>      <dbl>
```

```
## 1 A           2.18     0           3.50
## 2 B           2.11     0           3.50
## 3 C           2.04     0           3.50
## 4 D           2.02     0           3.58
## 5 E           1.97     0           3.50
## 6 F           1.99     0           3.25
## 7 G           2.07     0.0821      3.17
```

2a iv Analysis: For fully paid back loans, the average loans are paid within 2.5 years.

There are no significant differences of Actual Terms between grades. However, on average people pay back all of the amount below 800 days. We can conclude that if a loan is not paid after 800 days, it is less likely to be paid.

2a v Code Chunk: Calculate the annual return. Show how you calculate the percentage annual return. Is there any return from loans which are 'charged off'? Explain. How does return from charged off loans vary by loan grade? Compare the average return values with the average interest_rate on loans – do you notice any differences, and how do you explain this? How do returns vary by grade, and by sub-grade. If you wanted to invest in loans based on this data exploration, which loans would you invest in?

```
#Based on actual term, the actual annual return is
lcdf$actualReturn <- ifelse(lcdf$actualTerm>0, ((lcdf$total_pymnt -lcdf$funded_amnt)/lcdf$funded_amnt)*(1/lcdf$ac
tualTerm), 0)

#return from loans which are 'charged off'? How does return from charged-off loans vary by loan grade?
lcdf_charged_off <- subset(lcdf, loan_status == "Charged Off")
# profit for charged off loans (if any)
lcdf_charged_off %>% summarise(retValue=total_pymnt-funded_amnt) %>% filter(retValue > 0)
```

```
## # A tibble: 1,461 x 1
##    retValue
##       <dbl>
##  1     840.
##  2     539.
##  3     444.
##  4    1446.
##  5     578.
##  6    1136.
```

```
##  7    1761.
##  8     608.
##  9    2427.
## 10     609.
## # ... with 1,451 more rows
```

```
# any return from charged off loans
lcdf_charged_off %>% filter(total_pymnt > 0) %>% group_by(grade) %>% summarise(totPayment=mean(total_pymnt))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 7 x 2
##   grade totPayment
##   <chr>      <dbl>
## 1 A          8847.
## 2 B          8273.
## 3 C          7565.
## 4 D          7923.
## 5 E          7641.
## 6 F          6614.
## 7 G          7560.
```

```
lcdf_charged_off %>% group_by(grade) %>% summarise(nLoans=n(), actRet= (mean(actualReturn)))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 7 x 3
##   grade nLoans actRet
##   <chr>  <int>  <dbl>
## 1 A       1108 -0.112
## 2 B       2682 -0.110
## 3 C       4116 -0.116
## 4 D       2647 -0.122
```

```
## 5 E       1045 -0.128
## 6 F        191 -0.125
## 7 G         38 -0.110
```

```
lcdf %>% group_by(grade) %>% summarise(avgReturn= mean(actualReturn), avgInt= mean(int_rate)/100)
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 7 x 3
##   grade avgReturn avgInt
##   <chr>     <dbl>  <dbl>
## 1 A        0.0394 0.0725
## 2 B        0.0522 0.107
## 3 C        0.0573 0.137
## 4 D        0.0589 0.165
## 5 E        0.0545 0.198
## 6 F        0.0729 0.241
## 7 G        0.0633 0.258
```

```
lcdf %>% group_by(sub_grade) %>% summarise(avgReturn= mean(actualReturn), avgInt= mean(int_rate)/100)
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 35 x 3
##    sub_grade avgReturn avgInt
##    <chr>         <dbl>  <dbl>
##  1 A1           0.0355 0.0603
##  2 A2           0.0353 0.0649
##  3 A3           0.0392 0.0705
##  4 A4           0.0411 0.0758
##  5 A5           0.0428 0.0827
##  6 B1           0.0445 0.0888
##  7 B2           0.0477 0.0977
```

```
##  8 B3           0.0525 0.107
##  9 B4           0.0572 0.115
## 10 B5           0.0572 0.122
## # ... with 25 more rows
```

```
# Calculate total return
lcdf$totReturn <- ifelse(lcdf$actualTerm>0, ((lcdf$total_pymnt -lcdf$funded_amnt)/lcdf$funded_amnt), 0) #not annu
al return

(lcdf_chargedoff_return<-lcdf_charged_off %>% group_by(grade) %>% summarise(nLoans=n(), actRet= (mean(actualRetur
n)), intRate=mean(int_rate)))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 7 x 4
##   grade nLoans actRet intRate
##   <chr>  <int>  <dbl>   <dbl>
## 1 A       1108 -0.112    7.47
## 2 B       2682 -0.110   10.9
## 3 C       4116 -0.116   13.7
## 4 D       2647 -0.122   16.6
## 5 E       1045 -0.128   19.8
## 6 F        191 -0.125   24.2
## 7 G         38 -0.110   25.8
```

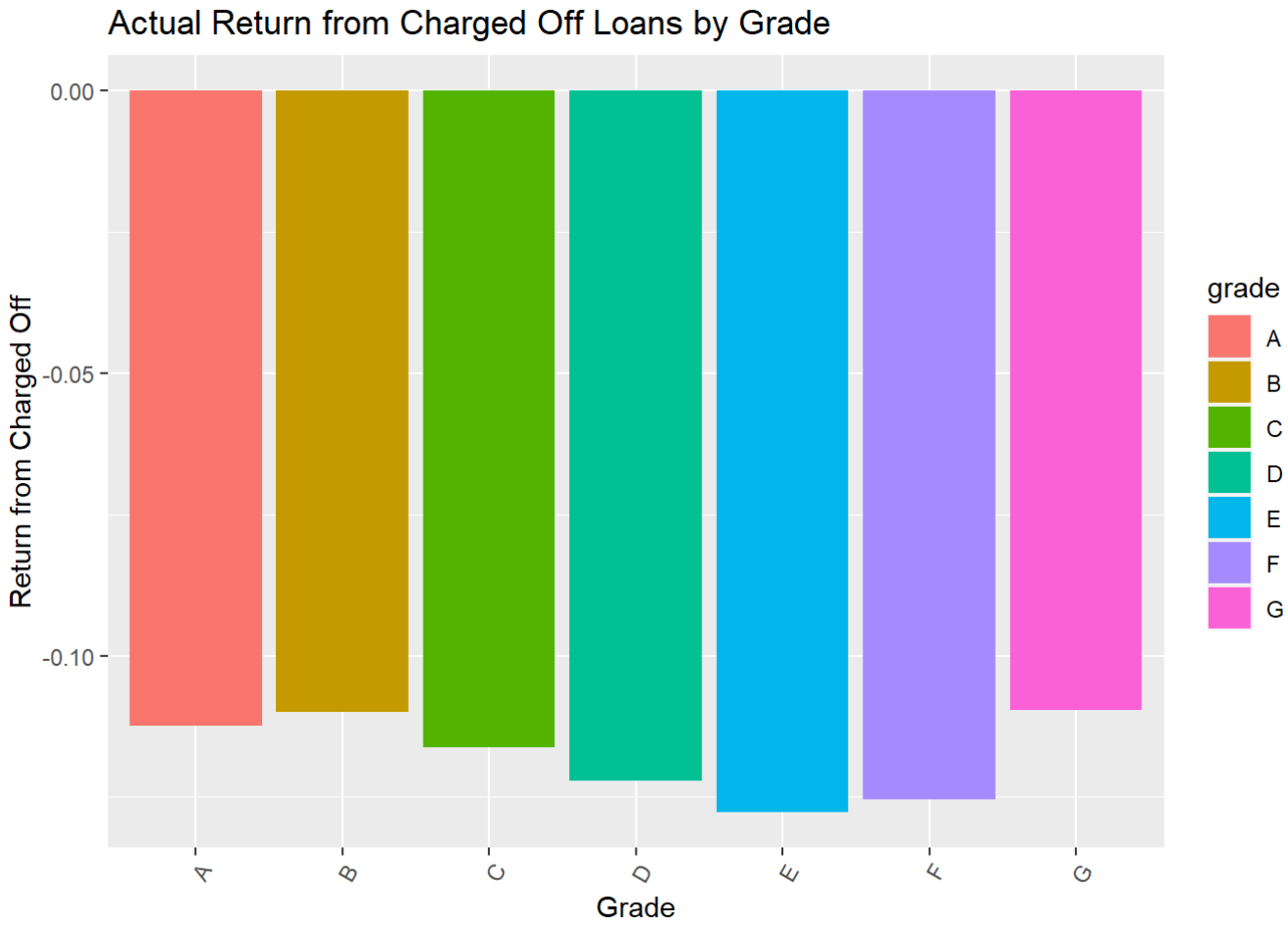
```
lcdf %>% group_by(sub_grade) %>% summarise(nLoans=n(), annRet=mean(actualReturn), TotRet=mean(totReturn), default
s=sum(loan_status=="Charged Off"), defRate=(sum(loan_status=="Charged Off")/n()), return_to_risk_ratio=(mean(actu
alReturn)/(sum(loan_status=="Charged Off")/n())))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 35 x 7
##    sub_grade nLoans annRet TotRet defaults defRate return_to_risk_ratio
```

```
##    <chr>      <int>  <dbl>  <dbl>    <int>   <dbl>                <dbl>
##  1 A1          3022 0.0355 0.0670       95  0.0314                1.13
##  2 A2          3583 0.0353 0.0653      165  0.0461                0.766
##  3 A3          3548 0.0392 0.0735      154  0.0434                0.903
##  4 A4          4839 0.0411 0.0756      281  0.0581                0.707
##  5 A5          5410 0.0428 0.0745      413  0.0763                0.561
##  6 B1          4123 0.0445 0.0769      373  0.0905                0.492
##  7 B2          4604 0.0477 0.0787      479  0.104                 0.458
##  8 B3          4603 0.0525 0.0865      531  0.115                 0.455
##  9 B4          4628 0.0572 0.0930      546  0.118                 0.485
## 10 B5          5441 0.0572 0.0904      753  0.138                 0.414
## # ... with 25 more rows
```

```
#graphing code
#Actual Return from Charged Off Loans by Grade
ggplot(lcdf_chargedoff_return) + aes(x = grade, y = actRet, fill = grade) + geom_bar(stat ="identity") + xlab("Gr
ade") + ylab("Return from Charged Off") + ggtitle("Actual Return from Charged Off Loans by Grade") +theme(axis.te
xt.x = element_text(angle = 60, hjust = 1))
```

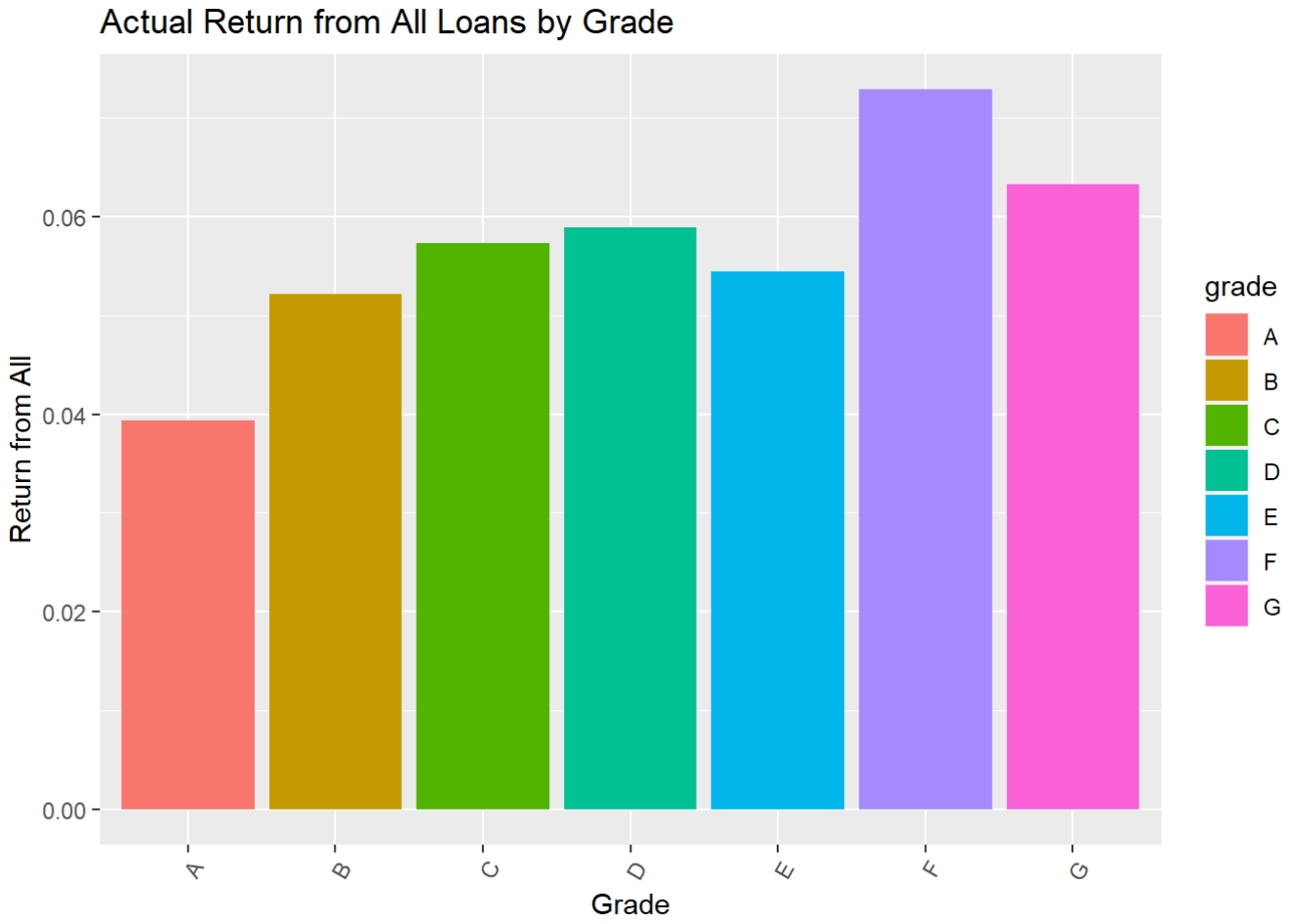



```
#Actual Return by Grade (All Status)
lcdf_all_return_grade<-lcdf %>% group_by(grade) %>% summarise(nLoans=n(), actRet= (mean(actualReturn)), intRate=m
ean(int_rate))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
ggplot(lcdf_all_return_grade) + aes(x = grade, y = actRet, fill = grade) + geom_bar(stat ="identity") + xlab("Gra
de") + ylab("Return from All") + ggtitle("Actual Return from All Loans by Grade") +theme(axis.text.x = element_te
```

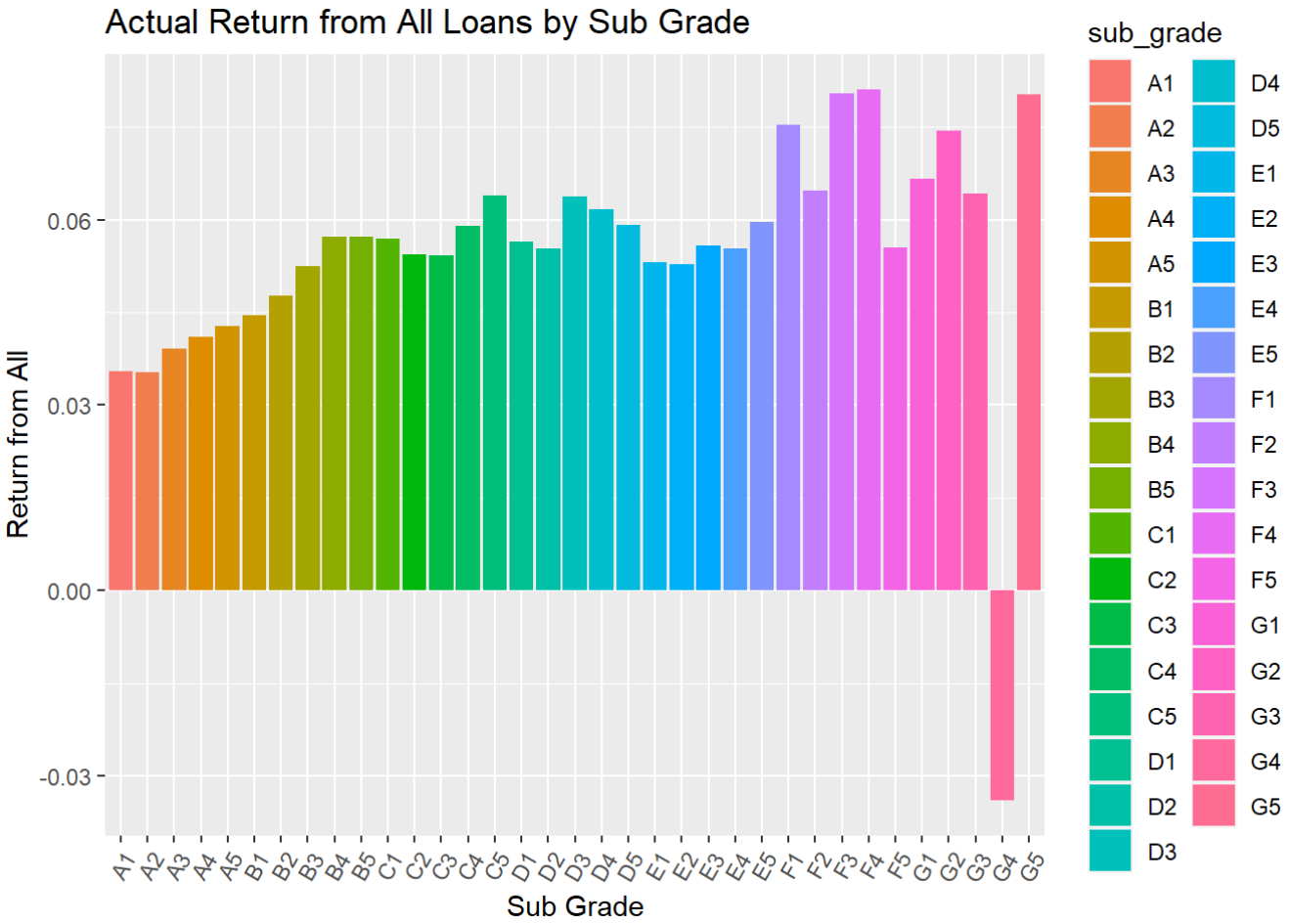
```
xt(angle = 60, hjust = 1))
```




```
#Actual Return by Sub Grade (All Status)
lcdf_all_return_subgrade<-lcdf %>% group_by(sub_grade) %>% summarise(nLoans=n(), actRet= (mean(actualReturn)), in
tRate=mean(int_rate))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
ggplot(lcdf_all_return_subgrade) + aes(x = sub_grade, y = actRet, fill = sub_grade) + geom_bar(stat ="identity")
 + xlab("Sub Grade") + ylab("Return from All") + ggtitle("Actual Return from All Loans by Sub Grade") +theme(axi
s.text.x = element_text(angle = 60, hjust = 1))
```



2a v Analysis: Some charged off loans are paid back partially. Some charged off loans also make profit (interest rates can be so high that the investors end up making money out of defaulted loans).

The losses vary a little bit by grade, which is roughly normally distributed (less losses on grade A and grade G). They typically range between 10-12%.

The highest average return is seen on grade F and G, because even though the default rate is high, the interest rate is high enough that the investors make more money on these loans than of other grades. By sub grade, return also increase as sub-grade goes worse from A1-G5. The highest average return is found on F4.

Suggestions which loans to make: 1) For highest return, we suggest to take F4, which average return is 8.1% but with default risk of 32%.

2. If safety is the priority, we suggest to take A1 with default risk of 3.1%, the lowest of all, and annual return of 3.6%. The return to risk ratio is 113%, the most optimum return considering the default rate.
3. Our personal suggestion is to try loans at sub-grade G3, where the default rate is an outlier (much lower than its peers), and offers the second-highest total return of all.

2a vi Code Chunk

```
#Generate some (at least 3) new derived attributes which you think may be useful for predicting default., and exp
lain what these are.
lcdf %>% group_by(sub_grade) %>% summarise(nLoans=n(), annRet=mean(actualReturn), TotRet=mean(totReturn), default
s=sum(loan_status=="Charged Off"), defRate=((sum(loan_status=="Charged Off")/n())))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 35 x 6
##    sub_grade nLoans annRet TotRet defaults defRate
##    <chr>      <int>  <dbl>  <dbl>    <int>   <dbl>
##  1 A1          3022 0.0355 0.0670       95  0.0314
##  2 A2          3583 0.0353 0.0653      165  0.0461
##  3 A3          3548 0.0392 0.0735      154  0.0434
##  4 A4          4839 0.0411 0.0756      281  0.0581
##  5 A5          5410 0.0428 0.0745      413  0.0763
##  6 B1          4123 0.0445 0.0769      373  0.0905
##  7 B2          4604 0.0477 0.0787      479  0.104
##  8 B3          4603 0.0525 0.0865      531  0.115
##  9 B4          4628 0.0572 0.0930      546  0.118
## 10 B5          5441 0.0572 0.0904      753  0.138
## # ... with 25 more rows
```

```
# emp_length, change to ordinal factor
#Consider emp_length - what are the different values, and how many examples are there for each value
lcdf %>% group_by(emp_length) %>% tally()
```

```
## # A tibble: 12 x 2
##    emp_length     n
##    <chr>      <int>
##  1 < 1 year    6911
##  2 1 year      5342
##  3 10+ years  25365
##  4 2 years     7426
##  5 3 years     6546
##  6 4 years     4967
##  7 5 years     4409
##  8 6 years     3700
##  9 7 years     4155
## 10 8 years     4022
## 11 9 years     3075
## 12 n/a         5104
```

```
#convert emp_length to factor -- with factor levels ordered in a meaningful way
lcdf$emp_length <- factor(lcdf$emp_length, levels=c("n/a", "< 1 year","1 year","2 years", "3 years" ,  "4 years",
"5 years",   "6 years",   "7 years" ,  "8 years", "9 years", "10+ years" ))
# Note: we could have converted to factor by simply using x<-as.factor(lcdf$emp_length), but here the factor leve
ls would be randomly arranged

#in purpose, merge some categories together if they are too small by themselves
#Look at loan purpose
lcdf %>% group_by(purpose) %>% tally()
```

```
## # A tibble: 13 x 2
##    purpose               n
##    <chr>             <int>
##  1 car                 719
##  2 credit_card       18780
```

```
##  3 debt_consolidation 48647
##  4 home_improvement    3942
##  5 house                254
##  6 major_purchase      1402
##  7 medical              900
##  8 moving               604
##  9 other               4455
## 10 renewable_energy      65
## 11 small_business       759
## 12 vacation             492
## 13 wedding                3
```

```
# do you want to recode some categories with very few cases to "other"
lcdf$purpose <- fct_recode(lcdf$purpose, other="wedding", other="educational", other="renewable_energy")
```

```
## Warning: Unknown levels in `f`: educational
```

```
#Derived attribute 1: proportion of satisfactory bankcard accounts
lcdf$propSatisBankcardAccts <- ifelse(lcdf$num_bc_tl>0, lcdf$num_bc_sats/lcdf$num_bc_tl, 0)

#Derived attribute 2: the length of borrower's history with LC
lcdf$earliest_cr_line<-paste(lcdf$earliest_cr_line, "-01", sep = "")
lcdf$earliest_cr_line<-parse_date_time(lcdf$earliest_cr_line, "myd")

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

#Derived attribute 3: ratio of openAccounts to totalAccounts
lcdf$prop_OpAccts_to_TotAccts <- ifelse(lcdf$open_acc >0, lcdf$open_acc/lcdf$total_acc, 0)

#Derived attribute 4: ratio of loan amount to annual income
lcdf$propLoanAmt_to_AnnInc <- lcdf$loan_amnt/lcdf$annual_inc

#Derived attribute 5: ratio of total current balance to annual income
lcdf$prop_CurBal_to_AnnIc <- lcdf$tot_cur_bal/lcdf$annual_inc
```

```
#Check if the loan is fully paid after 800 days
#lcdf$paid_after_800 <- ifelse(lcdf$actualTerm > 800 & lcdf$total_pymnt < ,,, , 0)

glimpse(lcdf)
```

```
## Rows: 81,022
## Columns: 154
## $ X1                                    <dbl> 1, 2, 3, 4, 5, 6, 7, 8, ...
## $ id                                    <lgl> NA, NA, NA, NA, NA, NA, ...
## $ member_id                             <lgl> NA, NA, NA, NA, NA, NA, ...
## $ loan_amnt                             <dbl> 5000, 17000, 3500, 14000...
## $ funded_amnt                           <dbl> 5000, 17000, 3500, 14000...
## $ funded_amnt_inv                       <dbl> 5000, 17000, 3500, 14000...
## $ term                                  <chr> "36 months", "36 months"...
## $ int_rate                              <dbl> 12.39, 12.39, 7.49, 11.9...
## $ installment                           <dbl> 167.01, 567.82, 108.86, ...
## $ grade                                 <chr> "C", "C", "A", "B", "C",...
## $ sub_grade                             <chr> "C1", "C1", "A4", "B5", ...
## $ emp_title                             <chr> "Trooper", "export agent...
## $ emp_length                            <fct> < 1 year, 1 year, 10+ ye...
## $ home_ownership                        <chr> "RENT", "RENT", "RENT", ...
## $ annual_inc                            <dbl> 48000, 53000, 72000, 440...
## $ verification_status                   <chr> "Not Verified", "Not Ver...
## $ issue_d                               <date> 2015-01-01, 2015-01-01,...
## $ loan_status                           <chr> "Fully Paid", "Fully Pai...
## $ pymnt_plan                            <chr> "n", "n", "n", "n", "n",...
## $ url                                   <lgl> NA, NA, NA, NA, NA, NA, ...
## $ desc                                  <lgl> NA, NA, NA, NA, NA, NA, ...
## $ purpose                               <fct> debt_consolidation, debt...
## $ title                                 <chr> "Debt consolidation", "D...
## $ zip_code                              <chr> "437xx", "902xx", "100xx...
## $ addr_state                            <chr> "OH", "CA", "NY", "IL", ...
## $ dti                                   <dbl> 14.25, 21.31, 14.34, 18....
## $ delinq_2yrs                           <dbl> 0, 0, 0, 0, 0, 0, 0, 6, ...
## $ earliest_cr_line                      <dttm> 2010-07-01, 2006-09-01,...
## $ inq_last_6mths                        <dbl> 0, 0, 1, 0, 0, 1, 0, 1, ...
## $ mths_since_last_delinq                <dbl> NA, NA, NA, 48, NA, NA, ...
```

```
## $ mths_since_last_record                <dbl> NA, NA, NA, NA, NA, NA, ...
## $ open_acc                              <dbl> 7, 12, 13, 11, 8, 10, 14...
## $ pub_rec                               <dbl> 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ revol_bal                             <dbl> 5994, 14690, 30063, 1433...
## $ revol_util                            <dbl> 44.4, 73.1, 86.1, 34.8, ...
## $ total_acc                             <dbl> 10, 22, 22, 15, 12, 11, ...
## $ initial_list_status                   <chr> "f", "f", "f", "f", "w",...
## $ out_prncp                             <dbl> 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ out_prncp_inv                         <dbl> 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ total_pymnt                           <dbl> 5475.140, 20452.099, 391...
## $ total_pymnt_inv                       <dbl> 5475.14, 20452.10, 3915....
## $ total_rec_prncp                       <dbl> 5000.00, 17000.00, 3500....
## $ total_rec_int                         <dbl> 475.14, 3423.71, 415.88,...
## $ total_rec_late_fee                    <dbl> 0.00, 28.39, 0.00, 0.00,...
## $ recoveries                            <dbl> 0.00, 0.00, 0.00, 0.00, ...
## $ collection_recovery_fee               <dbl> 0.0000, 0.0000, 0.0000, ...
## $ last_pymnt_d                          <dttm> 2015-12-01, 2018-03-01,...
## $ last_pymnt_amnt                       <dbl> 3978.93, 5.59, 108.69, 3...
## $ next_pymnt_d                          <lgl> NA, NA, NA, NA, NA, NA, ...
## $ last_credit_pull_d                    <chr> "Feb-2019", "Mar-2018", ...
## $ collections_12_mths_ex_med            <dbl> 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ mths_since_last_major_derog           <dbl> NA, NA, NA, NA, NA, NA, ...
## $ policy_code                           <dbl> 1, 1, 1, 1, 1, 1, 1, 1, ...
## $ application_type                      <chr> "Individual", "Individua...
## $ annual_inc_joint                      <lgl> NA, NA, NA, NA, NA, NA, ...
## $ dti_joint                             <lgl> NA, NA, NA, NA, NA, NA, ...
## $ verification_status_joint             <lgl> NA, NA, NA, NA, NA, NA, ...
## $ acc_now_delinq                        <dbl> 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ tot_coll_amt                          <dbl> 0, 0, 0, 455, 0, 0, 0, 0...
## $ tot_cur_bal                           <dbl> 19022, 39917, 30063, 795...
## $ open_acc_6m                           <lgl> NA, NA, NA, NA, NA, NA, ...
## $ open_act_il                           <lgl> NA, NA, NA, NA, NA, NA, ...
## $ open_il_12m                           <lgl> NA, NA, NA, NA, NA, NA, ...
## $ open_il_24m                           <lgl> NA, NA, NA, NA, NA, NA, ...
## $ mths_since_rcnt_il                    <lgl> NA, NA, NA, NA, NA, NA, ...
## $ total_bal_il                          <lgl> NA, NA, NA, NA, NA, NA, ...
## $ il_util                               <lgl> NA, NA, NA, NA, NA, NA, ...
```

```
## $ open_rv_12m                    <lgl> NA, NA, NA, NA, NA, NA, ...
## $ open_rv_24m                    <lgl> NA, NA, NA, NA, NA, NA, ...
## $ max_bal_bc                     <lgl> NA, NA, NA, NA, NA, NA, ...
## $ all_util                       <lgl> NA, NA, NA, NA, NA, NA, ...
## $ total_rev_hi_lim               <dbl> 13500, 20100, 34900, 412...
## $ inq_fi                         <lgl> NA, NA, NA, NA, NA, NA, ...
## $ total_cu_tl                    <lgl> NA, NA, NA, NA, NA, NA, ...
## $ inq_last_12m                   <lgl> NA, NA, NA, NA, NA, NA, ...
## $ acc_open_past_24mths           <dbl> 7, 2, 4, 2, 4, 7, 6, 3, ...
## $ avg_cur_bal                    <dbl> 3170, 3629, 2313, 7235, ...
## $ bc_open_to_buy                 <dbl> 610, 3279, 359, 23870, 3...
## $ bc_util                        <dbl> 81.5, 80.9, 98.7, 37.5, ...
## $ chargeoff_within_12_mths       <dbl> 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ delinq_amnt                    <dbl> 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ mo_sin_old_il_acct             <dbl> 29, 100, 44, 100, 49, 10...
## $ mo_sin_old_rev_tl_op           <dbl> 54, 84, 237, 165, 54, 12...
## $ mo_sin_rcnt_rev_tl_op          <dbl> 2, 23, 6, 15, 10, 2, 1, ...
## $ mo_sin_rcnt_tl                 <dbl> 2, 3, 6, 15, 10, 2, 1, 7...
## $ mort_acc                       <dbl> 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ mths_since_recent_bc           <dbl> 18, 23, 36, 27, 39, 2, 1...
## $ mths_since_recent_bc_dlq       <dbl> NA, NA, NA, NA, NA, NA, ...
## $ mths_since_recent_inq          <dbl> 4, NA, 6, 15, 11, 6, 4, ...
## $ mths_since_recent_revol_delinq <dbl> NA, NA, NA, NA, NA, NA, ...
## $ num_accts_ever_120_pd          <dbl> 0, 0, 0, 0, 0, 0, 0, 5, ...
## $ num_actv_bc_tl                 <dbl> 2, 7, 8, 3, 2, 4, 2, 3, ...
## $ num_actv_rev_tl                <dbl> 4, 9, 12, 3, 6, 4, 5, 4,...
## $ num_bc_sats                    <dbl> 2, 8, 8, 4, 2, 6, 4, 3, ...
## $ num_bc_tl                      <dbl> 2, 13, 15, 4, 3, 6, 7, 1...
## $ num_il_tl                      <dbl> 4, 6, 1, 6, 3, 1, 5, 5, ...
## $ num_op_rev_tl                  <dbl> 6, 10, 13, 5, 6, 8, 11, ...
## $ num_rev_accts                  <dbl> 6, 16, 21, 5, 9, 9, 18, ...
## $ num_rev_tl_bal_gt_0            <dbl> 4, 9, 12, 3, 6, 4, 5, 4,...
## $ num_sats                       <dbl> 7, 12, 13, 11, 8, 10, 14...
## $ num_tl_120dpd_2m               <dbl> 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ num_tl_30dpd                   <dbl> 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ num_tl_90g_dpd_24m             <dbl> 0, 0, 0, 0, 0, 0, 0, 5, ...
## $ num_tl_op_past_12m             <dbl> 3, 1, 2, 0, 1, 4, 4, 2, ...
```

```
## $ pct_tl_nvr_dlq                             <dbl> 100.0, 100.0, 100.0, 86....
## $ percent_bc_gt_75                           <dbl> 100.0, 57.1, 100.0, 75.0...
## $ pub_rec_bankruptcies                       <dbl> 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ tax_liens                                  <dbl> 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ tot_hi_cred_lim                            <dbl> 26852, 45660, 34900, 909...
## $ total_bal_ex_mort                          <dbl> 19022, 39917, 30063, 795...
## $ total_bc_limit                             <dbl> 3300, 17200, 28000, 3820...
## $ total_il_high_credit_limit                 <dbl> 13352, 25560, 0, 49511, ...
## $ revol_bal_joint                            <lgl> NA, NA, NA, NA, NA, NA, ...
## $ sec_app_earliest_cr_line                   <lgl> NA, NA, NA, NA, NA, NA, ...
## $ sec_app_inq_last_6mths                     <lgl> NA, NA, NA, NA, NA, NA, ...
## $ sec_app_mort_acc                           <lgl> NA, NA, NA, NA, NA, NA, ...
## $ sec_app_open_acc                           <lgl> NA, NA, NA, NA, NA, NA, ...
## $ sec_app_revol_util                         <lgl> NA, NA, NA, NA, NA, NA, ...
## $ sec_app_open_act_il                        <lgl> NA, NA, NA, NA, NA, NA, ...
## $ sec_app_num_rev_accts                      <lgl> NA, NA, NA, NA, NA, NA, ...
## $ sec_app_chargeoff_within_12_mths           <lgl> NA, NA, NA, NA, NA, NA, ...
## $ sec_app_collections_12_mths_ex_med         <lgl> NA, NA, NA, NA, NA, NA, ...
## $ sec_app_mths_since_last_major_derog        <lgl> NA, NA, NA, NA, NA, NA, ...
## $ hardship_flag                              <chr> "N", "N", "N", "N", "N",...
## $ hardship_type                              <lgl> NA, NA, NA, NA, NA, NA, ...
## $ hardship_reason                            <lgl> NA, NA, NA, NA, NA, NA, ...
## $ hardship_status                            <lgl> NA, NA, NA, NA, NA, NA, ...
## $ deferral_term                              <lgl> NA, NA, NA, NA, NA, NA, ...
## $ hardship_amount                            <lgl> NA, NA, NA, NA, NA, NA, ...
## $ hardship_start_date                        <lgl> NA, NA, NA, NA, NA, NA, ...
## $ hardship_end_date                          <lgl> NA, NA, NA, NA, NA, NA, ...
## $ payment_plan_start_date                    <lgl> NA, NA, NA, NA, NA, NA, ...
## $ hardship_length                            <lgl> NA, NA, NA, NA, NA, NA, ...
## $ hardship_dpd                               <lgl> NA, NA, NA, NA, NA, NA, ...
## $ hardship_loan_status                       <lgl> NA, NA, NA, NA, NA, NA, ...
## $ orig_projected_additional_accrued_interest <lgl> NA, NA, NA, NA, NA, NA, ...
## $ hardship_payoff_balance_amount             <lgl> NA, NA, NA, NA, NA, NA, ...
## $ hardship_last_payment_amount               <lgl> NA, NA, NA, NA, NA, NA, ...
## $ disbursement_method                        <chr> "Cash", "Cash", "Cash", ...
## $ debt_settlement_flag                       <chr> "N", "N", "N", "N", "N",...
## $ debt_settlement_flag_date                  <lgl> NA, NA, NA, NA, NA, NA, ...
```

```
## $ settlement_status                    <lgl> NA, NA, NA, NA, NA, NA, ...
## $ settlement_date                      <lgl> NA, NA, NA, NA, NA, NA, ...
## $ settlement_amount                    <lgl> NA, NA, NA, NA, NA, NA, ...
## $ settlement_percentage                <lgl> NA, NA, NA, NA, NA, NA, ...
## $ settlement_term                      <lgl> NA, NA, NA, NA, NA, NA, ...
## $ actualTerm                           <dbl> 0.9144422, 3.1622177, 3....
## $ actualReturn                         <dbl> 0.10391909, 0.06421590, ...
## $ totReturn                            <dbl> 0.09502800, 0.20306465, ...
## $ propSatisBankcardAccts               <dbl> 1.0000000, 0.6153846, 0....
## $ borrHistory                          <dbl> 4.503765, 8.334018, 19.7...
## $ prop_OpAccts_to_TotAccts             <dbl> 0.7000000, 0.5454545, 0....
## $ propLoanAmt_to_AnnInc                <dbl> 0.10416667, 0.32075472, ...
## $ prop_CurBal_to_AnnIc                 <dbl> 0.39629167, 0.75315094, ...
```

```
# convert all of these to factor
lcdf <- lcdf %>% mutate_if(is.character, as.factor)
```

Derived attribute 1: proportion of satisfactory bankcard accounts. The ratio describes how much the person has been able to pay out of all his debts so far. The higher this number, the more likely he is going to pay out his debts.

Derived attribute 2: the length of borrower's history with LC.The longer his history is, the more likely this person is going to pay for his debts (credible history), because this person has a proven track record.

Derived attribute 3: ratio of openAccounts to totalAccounts. This shows how many accounts this person has managed to close off (pay back). If he has only a few account remaining, he is more likely to pay back the loan.

Derived attribute 4: ratio of loan amount to annual income. This is the proportion of loan compared to his annual income. The smaller the ratio is, the more likely he is able to pay off the loan.

Derived attribute 5: ratio of total current balance to annual income.

2b Missing Values Are there missing values? What is the proportion of missing values in different variables? Explain how you will handle missing values for different variables. You should consider what he variable is about, and what missing values may arise from – for example, a variable monthsSinceLastDeliquency may have no value for someone who has not yet had a delinquency; what is a sensible value to replace the missing values in this case? Are there some variables you will exclude from your model due to missing values?

```
#Check NA before removal:
colMeans(is.na(lcdf))
```

```
##                                     X1
##                           0.0000000000
##                                     id
##                           1.0000000000
##                              member_id
##                           1.0000000000
##                              loan_amnt
##                           0.0000000000
##                            funded_amnt
##                           0.0000000000
##                          funded_amnt_inv
##                           0.0000000000
##                                   term
##                           0.0000000000
##                               int_rate
##                           0.0000000000
##                            installment
##                           0.0000000000
##                                  grade
##                           0.0000000000
##                              sub_grade
##                           0.0000000000
##                              emp_title
##                           0.0630939745
##                             emp_length
##                           0.0000000000
##                         home_ownership
##                           0.0000000000
##                             annual_inc
##                           0.0000000000
##                    verification_status
##                           0.0000000000
##                                issue_d
```

```
##                             0.0000000000
##                              loan_status
##                             0.0000000000
##                               pymnt_plan
##                             0.0000000000
##                                      url
##                             1.0000000000
##                                     desc
##                             1.0000000000
##                                  purpose
##                             0.0000000000
##                                    title
##                             0.0000000000
##                                 zip_code
##                             0.0000000000
##                               addr_state
##                             0.0000000000
##                                      dti
##                             0.0000000000
##                              delinq_2yrs
##                             0.0000000000
##                          earliest_cr_line
##                             0.0000000000
##                            inq_last_6mths
##                             0.0000000000
##                      mths_since_last_delinq
##                             0.4782898472
##                      mths_since_last_record
##                             0.8178642838
##                                 open_acc
##                             0.0000000000
##                                  pub_rec
##                             0.0000000000
##                                revol_bal
##                             0.0000000000
##                               revol_util
##                             0.0004196391
```

```
##                              total_acc
##                           0.0000000000
##                  initial_list_status
##                           0.0000000000
##                              out_prncp
##                           0.0000000000
##                          out_prncp_inv
##                           0.0000000000
##                            total_pymnt
##                           0.0000000000
##                        total_pymnt_inv
##                           0.0000000000
##                        total_rec_prncp
##                           0.0000000000
##                          total_rec_int
##                           0.0000000000
##                      total_rec_late_fee
##                           0.0000000000
##                                recoveries
##                           0.0000000000
##                    collection_recovery_fee
##                           0.0000000000
##                              last_pymnt_d
##                           0.0005554047
##                         last_pymnt_amnt
##                           0.0000000000
##                              next_pymnt_d
##                           1.0000000000
##                       last_credit_pull_d
##                           0.0001481079
##                  collections_12_mths_ex_med
##                           0.0000000000
##                mths_since_last_major_derog
##                           0.7033521760
##                             policy_code
##                           0.0000000000
##                        application_type
```

```
##                              0.0000000000
##                         annual_inc_joint
##                              1.0000000000
##                                 dti_joint
##                              1.0000000000
##                   verification_status_joint
##                              1.0000000000
##                             acc_now_delinq
##                              0.0000000000
##                              tot_coll_amt
##                              0.0000000000
##                               tot_cur_bal
##                              0.0000000000
##                               open_acc_6m
##                              1.0000000000
##                               open_act_il
##                              1.0000000000
##                               open_il_12m
##                              1.0000000000
##                               open_il_24m
##                              1.0000000000
##                         mths_since_rcnt_il
##                              1.0000000000
##                              total_bal_il
##                              1.0000000000
##                                   il_util
##                              1.0000000000
##                               open_rv_12m
##                              1.0000000000
##                               open_rv_24m
##                              1.0000000000
##                                 max_bal_bc
##                              1.0000000000
##                                  all_util
##                              1.0000000000
##                            total_rev_hi_lim
##                              0.0000000000
```

```
##                                 inq_fi
##                           1.0000000000
##                            total_cu_tl
##                           1.0000000000
##                           inq_last_12m
##                           1.0000000000
##                   acc_open_past_24mths
##                           0.0000000000
##                            avg_cur_bal
##                           0.0000000000
##                         bc_open_to_buy
##                           0.0120584533
##                                bc_util
##                           0.0126879119
##                chargeoff_within_12_mths
##                           0.0000000000
##                             delinq_amnt
##                           0.0000000000
##                     mo_sin_old_il_acct
##                           0.0375330157
##                   mo_sin_old_rev_tl_op
##                           0.0000000000
##                  mo_sin_rcnt_rev_tl_op
##                           0.0000000000
##                         mo_sin_rcnt_tl
##                           0.0000000000
##                             mort_acc
##                           0.0000000000
##                    mths_since_recent_bc
##                           0.0112068327
##                mths_since_recent_bc_dlq
##                           0.7287526845
##                   mths_since_recent_inq
##                           0.1014786107
##            mths_since_recent_revol_delinq
##                           0.6293352423
##                   num_accts_ever_120_pd
```

```
##                              0.0000000000
##                           num_actv_bc_tl
##                              0.0000000000
##                          num_actv_rev_tl
##                              0.0000000000
##                               num_bc_sats
##                              0.0000000000
##                                 num_bc_tl
##                              0.0000000000
##                                 num_il_tl
##                              0.0000000000
##                           num_op_rev_tl
##                              0.0000000000
##                          num_rev_accts
##                              0.0000000000
##                       num_rev_tl_bal_gt_0
##                              0.0000000000
##                                   num_sats
##                              0.0000000000
##                         num_tl_120dpd_2m
##                              0.0257831206
##                              num_tl_30dpd
##                              0.0000000000
##                        num_tl_90g_dpd_24m
##                              0.0000000000
##                        num_tl_op_past_12m
##                              0.0000000000
##                           pct_tl_nvr_dlq
##                              0.0000000000
##                         percent_bc_gt_75
##                              0.0124534077
##                       pub_rec_bankruptcies
##                              0.0000000000
##                                 tax_liens
##                              0.0000000000
##                           tot_hi_cred_lim
##                              0.0000000000
```

```
##                          total_bal_ex_mort
##                                 0.0000000000
##                             total_bc_limit
##                                 0.0000000000
##                   total_il_high_credit_limit
##                                 0.0000000000
##                               revol_bal_joint
##                                 1.0000000000
##                     sec_app_earliest_cr_line
##                                 1.0000000000
##                       sec_app_inq_last_6mths
##                                 1.0000000000
##                             sec_app_mort_acc
##                                 1.0000000000
##                             sec_app_open_acc
##                                 1.0000000000
##                          sec_app_revol_util
##                                 1.0000000000
##                           sec_app_open_act_il
##                                 1.0000000000
##                        sec_app_num_rev_accts
##                                 1.0000000000
##             sec_app_chargeoff_within_12_mths
##                                 1.0000000000
##           sec_app_collections_12_mths_ex_med
##                                 1.0000000000
##          sec_app_mths_since_last_major_derog
##                                 1.0000000000
##                               hardship_flag
##                                 0.0000000000
##                               hardship_type
##                                 1.0000000000
##                             hardship_reason
##                                 1.0000000000
##                             hardship_status
##                                 1.0000000000
##                              deferral_term
```

```
##                              1.0000000000
##                           hardship_amount
##                              1.0000000000
##                       hardship_start_date
##                              1.0000000000
##                         hardship_end_date
##                              1.0000000000
##                   payment_plan_start_date
##                              1.0000000000
##                            hardship_length
##                              1.0000000000
##                              hardship_dpd
##                              0.9996790995
##                       hardship_loan_status
##                              1.0000000000
## orig_projected_additional_accrued_interest
##                              1.0000000000
##             hardship_payoff_balance_amount
##                              1.0000000000
##               hardship_last_payment_amount
##                              0.9999876577
##                        disbursement_method
##                              0.0000000000
##                      debt_settlement_flag
##                              0.0000000000
##                 debt_settlement_flag_date
##                              1.0000000000
##                         settlement_status
##                              1.0000000000
##                           settlement_date
##                              1.0000000000
##                         settlement_amount
##                              1.0000000000
##                     settlement_percentage
##                              1.0000000000
##                           settlement_term
##                              0.9950137000
```

```
##                                actualTerm
##                             0.0000000000
##                             actualReturn
##                             0.0000000000
##                                totReturn
##                             0.0000000000
##                   propSatisBankcardAccts
##                             0.0000000000
##                              borrHistory
##                             0.0000000000
##                 prop_OpAccts_to_TotAccts
##                             0.0000000000
##                    propLoanAmt_to_AnnInc
##                             0.0000000000
##                     prop_CurBal_to_AnnIc
##                             0.0000000000
```

```
dim(lcdf)
```

```
## [1] 81022   154
```

```
#149
#drop variables with all NAs:
lcdf <- lcdf %>% select_if(function(x){!all(is.na(x))})

dim(lcdf)
```

```
## [1] 81022   104
```

```
#99 cols remaining, we dropped 50 cols

#Of the columns remaining, names of columns with missing values
names(lcdf)[colSums(is.na(lcdf))>0]
```

```
##  [1] "emp_title"                     "mths_since_last_delinq"
##  [3] "mths_since_last_record"        "revol_util"
##  [5] "last_pymnt_d"                  "last_credit_pull_d"
##  [7] "mths_since_last_major_derog"   "bc_open_to_buy"
##  [9] "bc_util"                       "mo_sin_old_il_acct"
## [11] "mths_since_recent_bc"          "mths_since_recent_bc_dlq"
## [13] "mths_since_recent_inq"         "mths_since_recent_revol_delinq"
## [15] "num_tl_120dpd_2m"              "percent_bc_gt_75"
## [17] "hardship_dpd"                  "hardship_last_payment_amount"
## [19] "settlement_term"
```

```
#missing value proportions in each column
colMeans(is.na(lcdf))
```

```
##                          X1                   loan_amnt
##                0.0000000000                0.0000000000
##                 funded_amnt               funded_amnt_inv
##                0.0000000000                0.0000000000
##                        term                    int_rate
##                0.0000000000                0.0000000000
##                 installment                       grade
##                0.0000000000                0.0000000000
##                   sub_grade                   emp_title
##                0.0000000000                0.0630939745
##                  emp_length               home_ownership
##                0.0000000000                0.0000000000
##                  annual_inc          verification_status
##                0.0000000000                0.0000000000
##                     issue_d                 loan_status
##                0.0000000000                0.0000000000
##                  pymnt_plan                     purpose
##                0.0000000000                0.0000000000
##                       title                    zip_code
##                0.0000000000                0.0000000000
##                  addr_state                         dti
```

```
##                   0.0000000000                   0.0000000000
##                    delinq_2yrs               earliest_cr_line
##                   0.0000000000                   0.0000000000
##                inq_last_6mths        mths_since_last_delinq
##                   0.0000000000                   0.4782898472
##        mths_since_last_record                       open_acc
##                   0.8178642838                   0.0000000000
##                        pub_rec                      revol_bal
##                   0.0000000000                   0.0000000000
##                     revol_util                      total_acc
##                   0.0004196391                   0.0000000000
##            initial_list_status                      out_prncp
##                   0.0000000000                   0.0000000000
##                  out_prncp_inv                    total_pymnt
##                   0.0000000000                   0.0000000000
##                total_pymnt_inv                total_rec_prncp
##                   0.0000000000                   0.0000000000
##                  total_rec_int              total_rec_late_fee
##                   0.0000000000                   0.0000000000
##                     recoveries         collection_recovery_fee
##                   0.0000000000                   0.0000000000
##                   last_pymnt_d                last_pymnt_amnt
##                   0.0005554047                   0.0000000000
##             last_credit_pull_d     collections_12_mths_ex_med
##                   0.0001481079                   0.0000000000
##    mths_since_last_major_derog                    policy_code
##                   0.7033521760                   0.0000000000
##                application_type                acc_now_delinq
##                   0.0000000000                   0.0000000000
##                   tot_coll_amt                    tot_cur_bal
##                   0.0000000000                   0.0000000000
##                total_rev_hi_lim           acc_open_past_24mths
##                   0.0000000000                   0.0000000000
##                    avg_cur_bal                bc_open_to_buy
##                   0.0000000000                   0.0120584533
##                        bc_util       chargeoff_within_12_mths
##                   0.0126879119                   0.0000000000
```

```
##                   delinq_amnt             mo_sin_old_il_acct
##                  0.0000000000                  0.0375330157
##           mo_sin_old_rev_tl_op          mo_sin_rcnt_rev_tl_op
##                  0.0000000000                  0.0000000000
##                mo_sin_rcnt_tl                      mort_acc
##                  0.0000000000                  0.0000000000
##           mths_since_recent_bc       mths_since_recent_bc_dlq
##                  0.0112068327                  0.7287526845
##         mths_since_recent_inq mths_since_recent_revol_delinq
##                  0.1014786107                  0.6293352423
##         num_accts_ever_120_pd                 num_actv_bc_tl
##                  0.0000000000                  0.0000000000
##               num_actv_rev_tl                  num_bc_sats
##                  0.0000000000                  0.0000000000
##                     num_bc_tl                    num_il_tl
##                  0.0000000000                  0.0000000000
##                num_op_rev_tl                 num_rev_accts
##                  0.0000000000                  0.0000000000
##            num_rev_tl_bal_gt_0                      num_sats
##                  0.0000000000                  0.0000000000
##               num_tl_120dpd_2m                  num_tl_30dpd
##                  0.0257831206                  0.0000000000
##             num_tl_90g_dpd_24m             num_tl_op_past_12m
##                  0.0000000000                  0.0000000000
##                pct_tl_nvr_dlq               percent_bc_gt_75
##                  0.0000000000                  0.0124534077
##           pub_rec_bankruptcies                     tax_liens
##                  0.0000000000                  0.0000000000
##                tot_hi_cred_lim              total_bal_ex_mort
##                  0.0000000000                  0.0000000000
##                total_bc_limit     total_il_high_credit_limit
##                  0.0000000000                  0.0000000000
##                 hardship_flag                   hardship_dpd
##                  0.0000000000                  0.9996790995
##   hardship_last_payment_amount            disbursement_method
##                  0.9999876577                  0.0000000000
##           debt_settlement_flag                settlement_term
```

```
##                   0.0000000000                   0.9950137000
##                     actualTerm                   actualReturn
##                   0.0000000000                   0.0000000000
##                      totReturn         propSatisBankcardAccts
##                   0.0000000000                   0.0000000000
##                    borrHistory       prop_OpAccts_to_TotAccts
##                   0.0000000000                   0.0000000000
##          propLoanAmt_to_AnnInc            prop_CurBal_to_AnnIc
##                   0.0000000000                   0.0000000000
```

```
# or, get only those columns where there are missing values
colMeans(is.na(lcdf))[colMeans(is.na(lcdf))>0]
```

```
##                      emp_title         mths_since_last_delinq
##                   0.0630939745                   0.4782898472
##         mths_since_last_record                     revol_util
##                   0.8178642838                   0.0004196391
##                   last_pymnt_d             last_credit_pull_d
##                   0.0005554047                   0.0001481079
##    mths_since_last_major_derog                 bc_open_to_buy
##                   0.7033521760                   0.0120584533
##                        bc_util             mo_sin_old_il_acct
##                   0.0126879119                   0.0375330157
##           mths_since_recent_bc       mths_since_recent_bc_dlq
##                   0.0112068327                   0.7287526845
##          mths_since_recent_inq mths_since_recent_revol_delinq
##                   0.1014786107                   0.6293352423
##               num_tl_120dpd_2m                percent_bc_gt_75
##                   0.0257831206                   0.0124534077
##                   hardship_dpd   hardship_last_payment_amount
##                   0.9996790995                   0.9999876577
##                settlement_term
##                   0.9950137000
```

```
#remove variables which have more than 60% missing values, because the data available is insufficient to predict
 missing values.
nm<-names(lcdf)[colMeans(is.na(lcdf))>0.6]
lcdf <- lcdf %>% select(-nm)
```

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

```
#Impute missing values - first get the columns with missing values
colMeans(is.na(lcdf))[colMeans(is.na(lcdf))>0]
```

```
##              emp_title mths_since_last_delinq             revol_util
##           0.0630939745           0.4782898472           0.0004196391
##           last_pymnt_d     last_credit_pull_d         bc_open_to_buy
##           0.0005554047           0.0001481079           0.0120584533
##                bc_util     mo_sin_old_il_acct   mths_since_recent_bc
##           0.0126879119           0.0375330157           0.0112068327
##  mths_since_recent_inq       num_tl_120dpd_2m       percent_bc_gt_75
##           0.1014786107           0.0257831206           0.0124534077
```

```
#summary of data in these columns
nm<- names(lcdf)[colSums(is.na(lcdf))>0]
summary(lcdf[, nm])
```

```
##             emp_title     mths_since_last_delinq   revol_util
##  Teacher          : 1524   Min.   :  0.00         Min.   :  0.00
##  Manager          : 1323   1st Qu.: 15.00         1st Qu.: 36.50
##  Owner            :  732   Median : 30.00         Median : 54.35
##  Registered Nurse:  637   Mean   : 33.63         Mean   : 54.22
##  Supervisor       :  596   3rd Qu.: 49.00         3rd Qu.: 72.20
##  (Other)          :71098   Max.   :133.00         Max.   :184.60
```

```
##  NA's            : 5112   NA's   :38752          NA's   :34
##   last_pymnt_d                last_credit_pull_d bc_open_to_buy
##  Min.   :2014-09-01 00:00:00   Feb-2019:28164     Min.   :     0
##  1st Qu.:2016-01-01 00:00:00   Oct-2016: 4110     1st Qu.:  1025
##  Median :2017-01-01 00:00:00   Jan-2019: 3238     Median :  3656
##  Mean   :2016-11-05 04:39:00   Oct-2017: 2712     Mean   :  8854
##  3rd Qu.:2017-10-01 00:00:00   Jan-2018: 2604     3rd Qu.: 10377
##  Max.   :2018-08-01 00:00:00   (Other) :40182     Max.   :264424
##  NA's   :45                    NA's    :   12     NA's   :977
##     bc_util       mo_sin_old_il_acct mths_since_recent_bc mths_since_recent_inq
##  Min.   :  0.00   Min.   :  1.0      Min.   :  0.00       Min.   : 0.000
##  1st Qu.: 43.30   1st Qu.: 96.0      1st Qu.:  6.00       1st Qu.: 2.000
##  Median : 67.30   Median :128.0      Median : 13.00       Median : 5.000
##  Mean   : 63.46   Mean   :124.8      Mean   : 23.92       Mean   : 6.731
##  3rd Qu.: 87.40   3rd Qu.:152.0      3rd Qu.: 29.00       3rd Qu.:10.000
##  Max.   :318.20   Max.   :545.0      Max.   :451.00       Max.   :25.000
##  NA's   :1028     NA's   :3041       NA's   :908          NA's   :8222
##  num_tl_120dpd_2m percent_bc_gt_75
##  Min.   :0.000    Min.   :  0.00
##  1st Qu.:0.000    1st Qu.: 20.00
##  Median :0.000    Median : 50.00
##  Mean   :0.001    Mean   : 49.43
##  3rd Qu.:0.000    3rd Qu.: 80.00
##  Max.   :3.000    Max.   :100.00
##  NA's   :2089     NA's   :1009
```

```
#mths_since_last_delinq: has 48% missings, these pertain to no delinquincy, so replace by a value higher than the max (500) -- we will try this out and put results in a temporary dataset lcx, with the attributes that have missn g values
lcx<-lcdf[, c(nm)]
colMeans(is.na(lcx))[colMeans(is.na(lcx))>0]
```

```
##             emp_title mths_since_last_delinq            revol_util
##          0.0630939745           0.4782898472          0.0004196391
##          last_pymnt_d     last_credit_pull_d         bc_open_to_buy
##          0.0005554047           0.0001481079          0.0120584533
```

```
##                bc_util     mo_sin_old_il_acct   mths_since_recent_bc
##           0.0126879119           0.0375330157           0.0112068327
##  mths_since_recent_inq       num_tl_120dpd_2m       percent_bc_gt_75
##           0.1014786107           0.0257831206           0.0124534077
```

```
lcx<- lcx %>% replace_na(list(mths_since_last_delinq = 500))

#bc_open_to_buy, use mean because number of NA is really low (1.2%)
lcx<- lcx %>% replace_na(list(bc_open_to_buy=mean(lcdf$bc_open_to_buy, na.rm=TRUE)))

#Replace na in last_credit_pull_d with a date older than 1 year(valid period). In this case we chose 1 Jan 2015.
lcx<- lcx %>% replace_na(list(last_credit_pull_d='01-01-2015'))
```

```
## Warning in `[<-.factor`(`*tmp*`, !is_complete(data[[var]]), value =
## "01-01-2015"): invalid factor level, NA generated
```

```
sum(is.na(lcx$last_credit_pull_d)>0) #prove we replaced the NAs
```

```
## [1] 12
```

```
#mo_sin_old_il_acct, use mean because number of NA is really low (3.8%)
lcx<- lcx %>% replace_na(list(mo_sin_old_il_acct=max(lcdf$mo_sin_old_il_acct, na.rm=TRUE)))

#mths_since_recent_bc, use max because NA because it means the person has never opened a bankcard acc before. So
 we assign a number that is the longest, or way above the max.
lcx<- lcx %>% replace_na(list(mths_since_recent_bc=max(lcdf$mths_since_recent_bc, na.rm=TRUE)))

#mths_since_recent_inq, use max because NA because it means no inquiry has been made. So we assign a number that
 is the longest, or way above the max.
lcx<- lcx %>% replace_na(list(mths_since_recent_inq=max(lcdf$mths_since_recent_inq, na.rm=TRUE)))

#bc_util, use mean because number of NA is really low (1.2%)
lcx<- lcx %>% replace_na(list(bc_util=mean(lcdf$bc_util, na.rm=TRUE)))
```

```
#listnum_tl_120dpd_2m, use mean because number of NA is really low (2.6%)
lcx<- lcx %>% replace_na(list(num_tl_120dpd_2m = mean(lcdf$num_tl_120dpd_2m, na.rm=TRUE)))

#percent_bc_gt_75, use mean because number of NA is really low (1.2%)
lcx<- lcx %>% replace_na(list(percent_bc_gt_75 = mean(lcdf$percent_bc_gt_75, na.rm=TRUE)))

#revol_util, use mean because number of NA is really low (.04%)
lcx<- lcx %>% replace_na(list(revol_util = mean(lcdf$revol_util, na.rm = TRUE)))

#emp_length, NA means 0 experience, so replace it with < 1 year
lcx<- lcx %>% replace_na(list(emp_length= "< 1 year"))

#After trying this out on the temporary dataframe lcx, if we are sure this is what we want, we can now  replace t
he missing values on the lcdf dataset
lcdf<- lcdf %>% replace_na(list(mths_since_last_delinq=500, bc_open_to_buy=mean(lcdf$bc_open_to_buy, na.rm=TRUE),
last_credit_pull_d='01-01-2015', mo_sin_old_il_acct=max(lcdf$mo_sin_old_il_acct, na.rm=TRUE), mths_since_recent_b
c=max(lcdf$mths_since_recent_bc, na.rm=TRUE), mths_since_recent_inq=max(lcdf$mths_since_recent_inq, na.rm=TRUE),
 bc_util=mean(lcdf$bc_util, na.rm=TRUE), num_tl_120dpd_2m = mean(lcdf$num_tl_120dpd_2m, na.rm=TRUE),percent_bc_gt
_75 = mean(lcdf$percent_bc_gt_75, na.rm=TRUE), revol_util = mean(lcdf$revol_util, na.rm = TRUE), emp_length= "< 1
year"))
```

```
## Warning in `[<-.factor`(`*tmp*`, !is_complete(data[[var]]), value =
## "01-01-2015"): invalid factor level, NA generated
```

```
#CHECK FOR NAs AGAIN
colMeans(is.na(lcdf))[colMeans(is.na(lcdf))>0]
```

```
##          emp_title       last_pymnt_d last_credit_pull_d
##       0.0630939745       0.0005554047       0.0001481079
```

```
# The last payment date missing are from 'Charger Off' where they didn't pay at all.
lcdf %>% filter(is.na(lcdf$last_pymnt_d)) %>% group_by(loan_status) %>% tally()
```

```
## # A tibble: 1 x 2
##   loan_status     n
##   <fct>       <int>
## 1 Charged Off    45
```

```
#replace the NA in last payment date with 3 years after issue date
#lcdf<- lcdf %>% replace_na(list(lcdf$last_pymnt_d=issue_d+years(3)))
```

2b comments/analysis:

3 Data Leakage 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 variables in the data which may have been updated during the loan period (ie., after the loan is funded). Identify and explain which variables will you exclude from the model

```
#Drop some other columns which are not useful and those which will cause 'leakage'
lcdf <- lcdf %>% select(-c(funded_amnt_inv, term, emp_title, pymnt_plan, title, zip_code, addr_state, out_prncp,
 out_prncp_inv, total_pymnt_inv, total_rec_prncp, total_rec_int, last_credit_pull_d, policy_code, disbursement_me
thod, debt_settlement_flag, hardship_flag, application_type))

#Drop some other variables
#varsToRemove <- c("last_pymnt_d", "last_pymnt_amnt","annRet")
varsToRemove <- c("inq_last_6mths",
                  "acc_open_past_24mths",
                  "actualReturn",
                  "actualTerm",
                  "bc_open_to_buy",
                  "collection_recovery_fee",
                  "collections_12_mths_ex_med",
                  "earliest_cr_line",
                  "funded_amnt",
                  "initial_list_status",
                  "issue_d",
                  "last_pymnt_amnt",
                  "last_pymnt_d",
                  "mo_sin_old_il_acct",
```

```
                    "mo_sin_old_rev_tl_op",
                    "mo_sin_rcnt_rev_tl_op",
                    "mo_sin_rcnt_tl",
                    "mths_since_recent_bc",
                    "mths_since_recent_inq",
                    "num_actv_bc_tl",
                    "num_actv_rev_tl",
                    "num_bc_sats",
                    "num_bc_tl",
                    "num_il_tl",
                    "num_tl_30dpd",
                    "num_tl_90g_dpd_24m",
                    "open_acc",
                    "pct_tl_nvr_dlq",
                    "percent_bc_gt_75",
                    "recoveries",
                    "tax_liens",
                    "tot_coll_amt",
                    "total_il_high_credit_limit",
                    "total_pymnt",
                    "total_rec_late_fee",
                    "total_rev_hi_lim",
                    "totReturn",
                    "X1",
                    "actualTerm",
                    "actualReturn",
                    "totReturn"
)
lcdf <- lcdf %>% select(-varsToRemove)
```

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

3 Analysis:

4. Uni-variate Code Chunk Do a uni-variate analyses to determine which variables (from amongst those you decide to consider for the next stage prediction task) will be individually useful for predicting the dependent variable (loan_status). For this, you need a measure of relationship between the dependent variable and each of the potential predictor variables. Given loan-status as a binary dependent variable, which measure will you use? From your analyses using this measure, which variables do you think will be useful for predicting loan_status? (Note – if certain variables on their own are highly predictive of the outcome, it is good to ask if this variable has a leakage issue).

```
#split the data into trn, tst subsets
TRNFRACTION = 0.5 #or use other values
nr<-nrow(lcdf)

trnIndex<- sample(1:nr, size = round(TRNFRACTION * nr), replace=FALSE)
lcdfTrn <- lcdf[trnIndex, ]
lcdfTst <- lcdf[-trnIndex, ]

library(pROC) #for AUC function

#Consider factor variable as numbers:
auc(response=lcdfTrn$loan_status, as.numeric(lcdfTrn$emp_length))
```

```
## Area under the curve: 0.5317
```

```
# For the numeric variables:
aucsNum<-sapply(lcdfTrn %>% select_if(is.numeric), auc, response=lcdfTrn$loan_status)

#Or considering both numeric and factor variables:
aucAll<- sapply(lcdfTrn %>% mutate_if(is.factor, as.numeric) %>% select_if(is.numeric), auc, response=lcdfTrn$loa
n_status)

# determine which variables have auc > 0.5 (have predicting capabilities)
aucAll[aucAll>0.5]
```

```
##                loan_amnt                 int_rate                    grade
##                0.5129270                0.6731617                0.6651966
##                sub_grade               emp_length           home_ownership
```

```
##                0.6731885                0.5316691                0.5565789
##               annual_inc              loan_status                      dti
##                0.5739763                1.0000000                0.5738406
##   mths_since_last_delinq               revol_bal               revol_util
##                0.5004903                0.5350469                0.5398486
##                total_acc          acc_now_delinq              tot_cur_bal
##                0.5196796                0.5002435                0.5618977
##              avg_cur_bal                  bc_util chargeoff_within_12_mths
##                0.5707452                0.5568959                0.5001080
##             delinq_amnt                 mort_acc            num_op_rev_tl
##                0.5003133                0.5644889                0.5167550
##            num_rev_accts                 num_sats         num_tl_120dpd_2m
##                0.5101946                0.5105660                0.5002780
##          tot_hi_cred_lim       total_bal_ex_mort           total_bc_limit
##                0.5746151                0.5155682                0.5770967
##   propSatisBankcardAccts              borrHistory prop_OpAccts_to_TotAccts
##                0.5205875                0.5439987                0.5402673
##    propLoanAmt_to_AnnInc     prop_CurBal_to_AnnIc
##                0.5570459                0.5359701
```

```
#convert to tibble, and set the threshold to 0.53 to see the most influential variables:
library(broom)
tidy(aucAll[aucAll > 0.53]) %>% view()

# in sorted order
tidy(aucAll) %>% arrange(desc(aucAll)) %>% view()
```

Analysis: The measurement we use is AUC, because it properly measures the variables importance and quality of the model's predictions. The threshold we are using is AUC score >= 0.55, which means that the predictive capability is better than random chance (50%). Using the AUC score, we can see that the 3 most influential variables are: int_rate, grade, sub_grade. The complete list can be find below: (EXCEL TABLE)

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this: