STA 160: Report

Thomas Munduchira, Sulli Vo, Dannie Vo, Ricardo Rendon Reynoso, Sarah Rahman

Introduction

LendingClub is a peer to peer lending company in which their product offers allows consumers to both invest and borrow loans. They offer multiple kinds of loans like student loans, personal loans, auto refinancing loans and even business loans. The borrowers who are interested in obtaining a loan will get a loan grade assigned to them which affects their interest rate and amount of money they can borrow. A lot of the LendingClub data leads to insightful conclusions about the borrowing and investing patterns of all kinds of individuals. Through our project, we will attempt to explain patterns and similarities of the behaviors of borrowers and investors. We will also create a classifier to predict the likelihood of paying off a loan or defaulting on a loan.

Dataset

The dataset we are using is a compilation of data on loans issued by LendingClub from the period 2007 to 2015. The data includes information on the current loan status (how much has been funded so far, how much has been paid off, etc) as well as information about the borrower (occupation, income, credit score, etc). This data lends itself to a variety of interesting financial analysis, notably time series analysis since the data is date stamped.

More information about the dataset can be found here: [https://www.kaggle.com/wendykan/lending-club-loan-data (https://www.kaggle.com/wendykan/lending-club-loan-data)](https://www.kaggle.com/wendykan/lending-club-loan-data).

Data Exploration

In [3]:
```
%load_ext rpy2.ipython
```

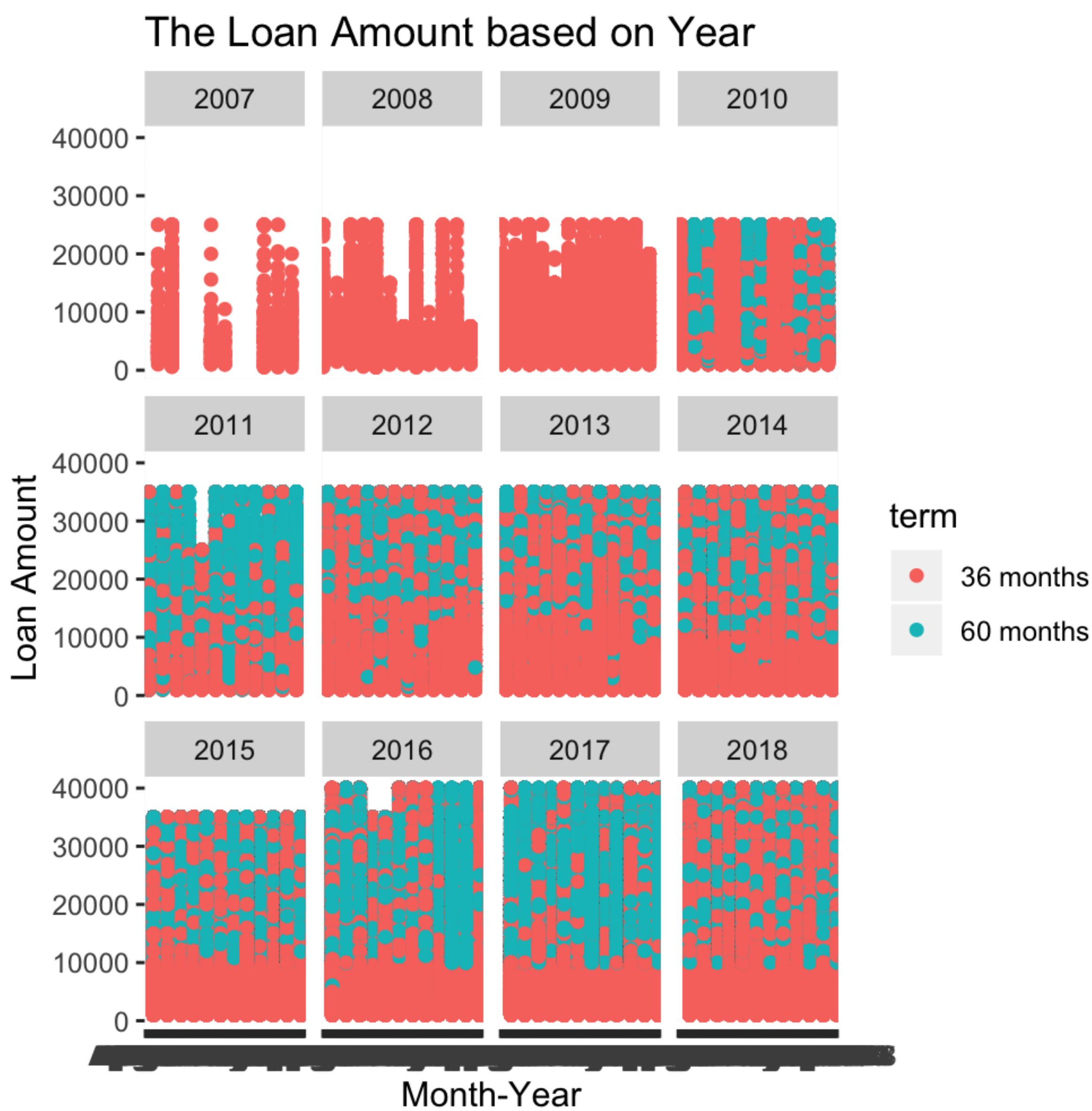
In [22]:
```
%%R -w 5 -h 5 --units in -r 200

set.seed(50)

library(data.table)
library(ggplot2)
library(zoo)
library(plyr)
library(tidyr)
library(maps)
library(magrittr)
library(gapminder)
library(plotly)
```

In [23]:

```
%%R -w 5 -h 5 --units in -r 200

mydata <- fread("data/loan.csv", sep = ",", fill = FALSE, select = c(3,
4, 5, 6,
    7, 8, 9, 11, 12, 13, 14, 16, 17, 21, 24, 27, 31, 39, 46, 47, 48, 53,
59, 65,
    106, 115, 138))
mydata$issue_year = year(as.yearmon(mydata$issue_d, "%b-%Y"))
mydata$issue_month = month(as.yearmon(mydata$issue_d, "%b-%Y"))
# mydata$issue_d = format(as.Date(as.yearmon(mydata$issue_d, '%b-%
Y')),'%m-%Y')
```

In [24]:

```
%%R -w 5 -h 5 --units in -r 200

ggplot(mydata, aes(x = issue_d, y = loan_amnt, color = term)) + xlab("Mo
nth-Year") + ylab("Loan Amount") + ggtitle("The Loan Amount based on Yea
r") + geom_point() + facet_wrap(~issue_year)
```

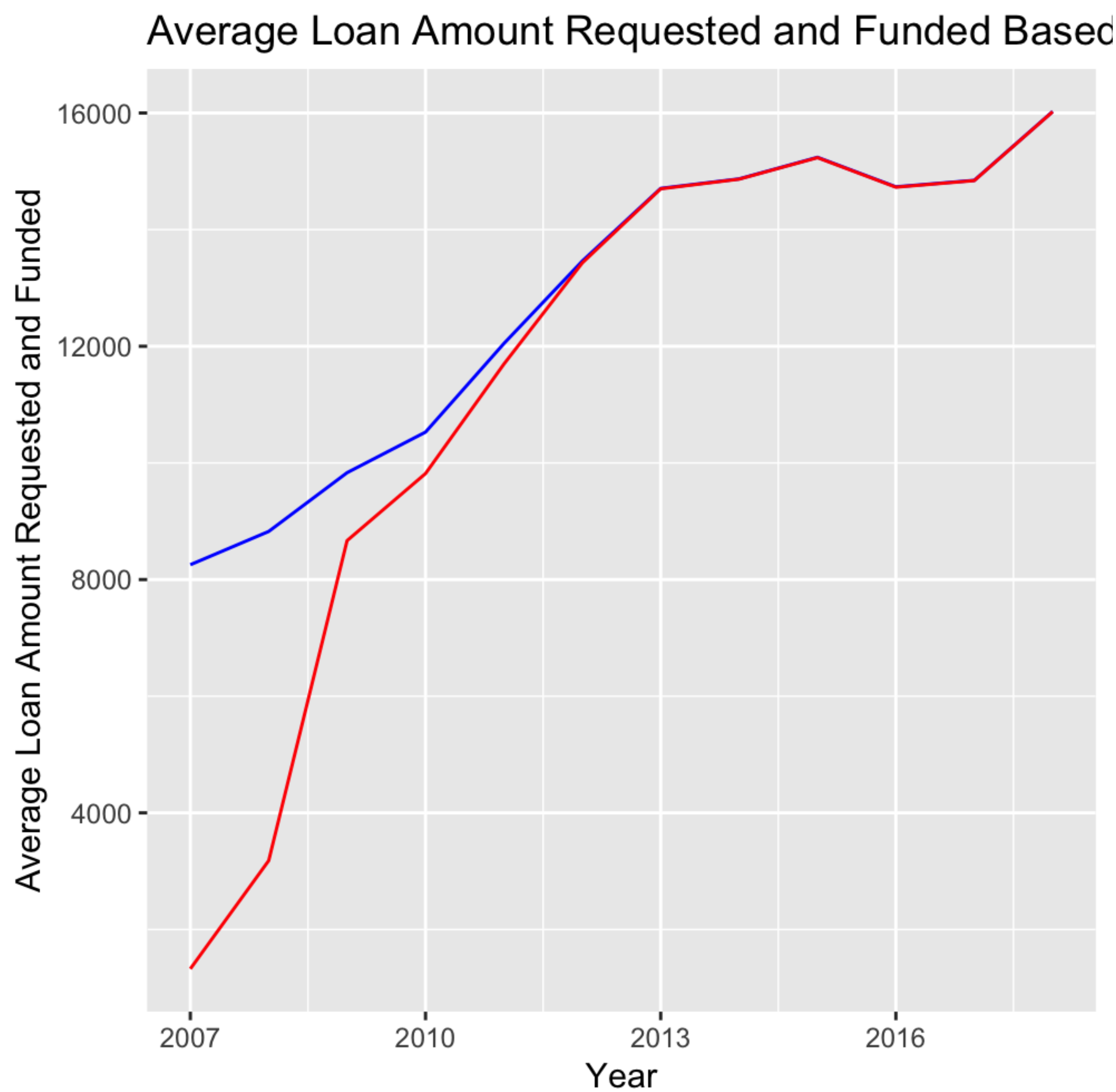


Based on the plot above, we can see that loan amounts are usually below $25, 000until2011, whenitgoesashighas$35,000. This remains the case until 2016, at which point requests went beyond $40,000.

During 2007-2009, the loans all had 36 month payment plans, but people started borrowing loans with 60 month payment plans from 2010 onwards. Loans with higher amounts requested are more likely to have 60 month payment plans.

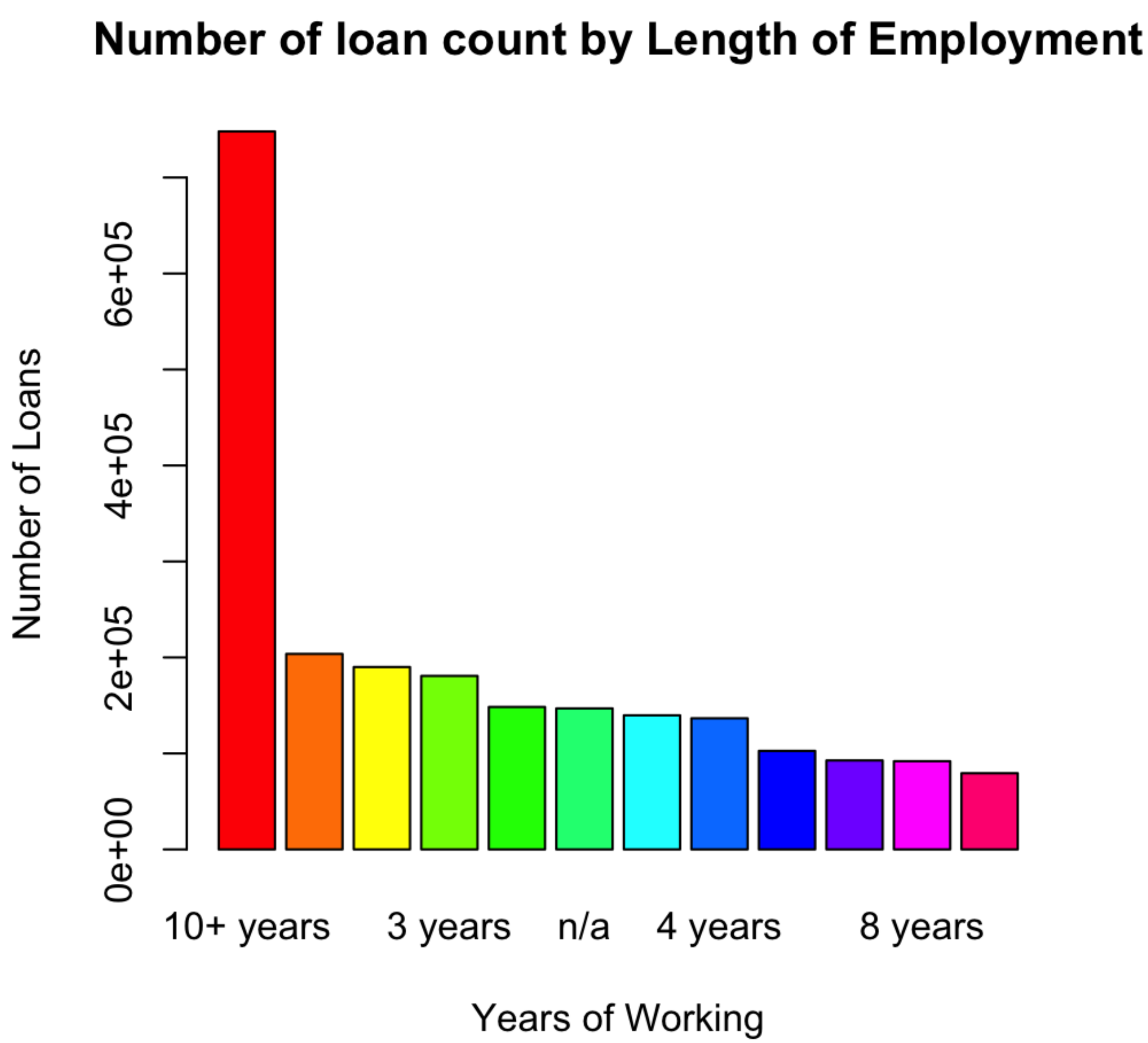
In [25]:

```
%%R -w 5 -h 5 --units in -r 200

ave_loan_amnt = aggregate(loan_amnt ~ issue_year, mydata, FUN = mean, n
a.rm = T)
avg_funded_amt_inv = aggregate(funded_amnt_inv ~ issue_year, mydata, FUN
= mean, na.rm = T)
joinbyyear <- merge(ave_loan_amnt, avg_funded_amt_inv, by = "issue_year"
)
ggplot(joinbyyear, aes(x = issue_year)) + geom_line(aes(y = loan_amnt),
colour = "blue") +
    geom_line(aes(y = funded_amnt_inv), colour = "red") + ylab("Average
 Loan Amount Requested and Funded") +
    xlab("Year") + ggtitle("Average Loan Amount Requested and Funded Bas
ed on Year")
```



The plot above shows the average amount of loan requested and the amount of loan funded by investors each year. The average loan amount is on the rise with the exception of the 2015-2016 period. Moreover, the loan amount funded is less than the requested loan amount from 2007-2012. This shows that it is getting easier to receive loans later on.

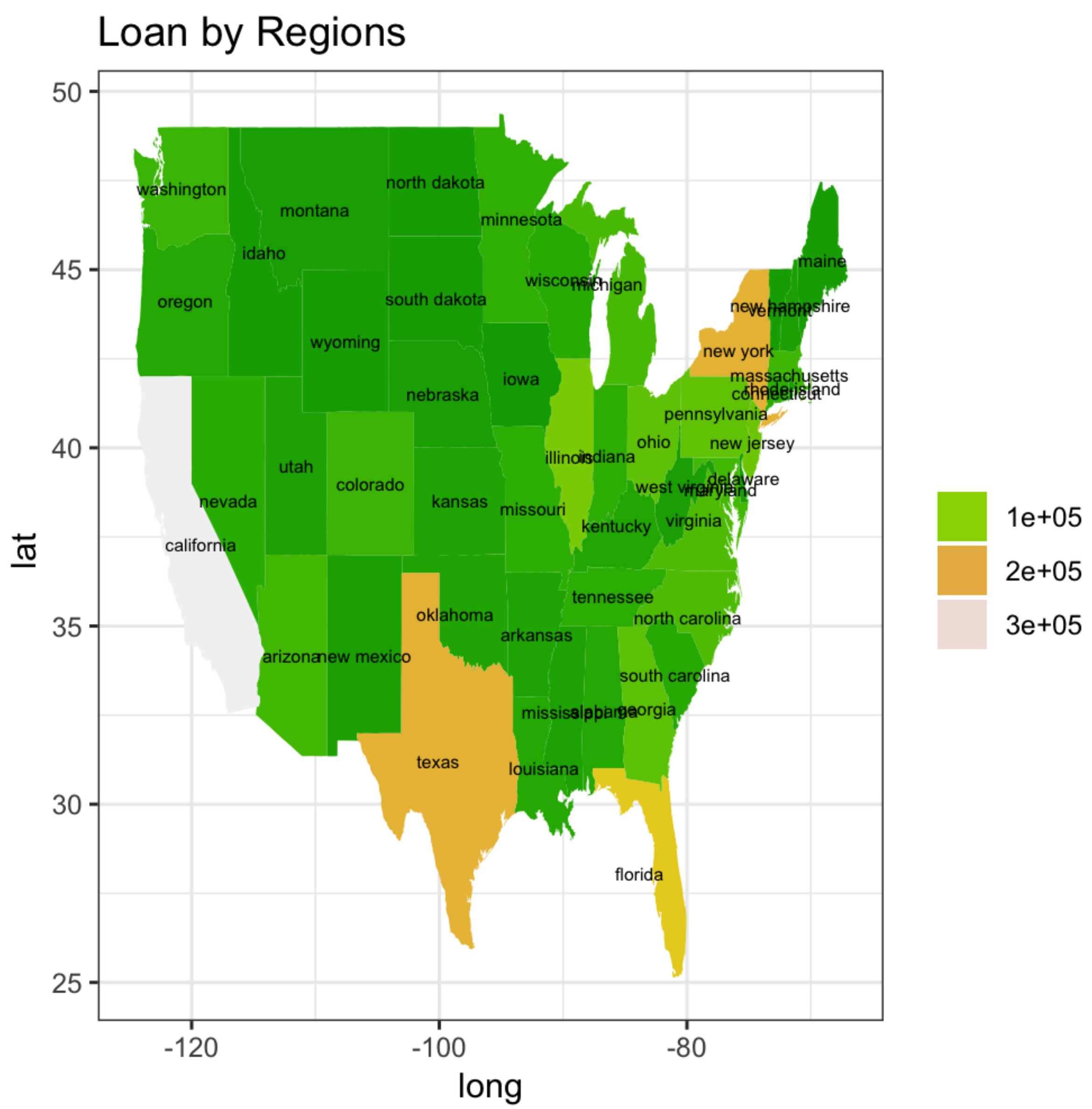
In [26]:
```
%%R -w 5 -h 5 --units in -r 200

levels(factor(mydata$emp_length))
emp_length_count = as.data.frame(mydata %>% dplyr::count(emp_length, sor
t = TRUE))
barplot(emp_length_count$n, main = "Number of loan count by Length of Em
ployment",
    xlab = "Years of Working", names.arg = emp_length_count$emp_length,
ylab = "Number of Loans",
    col = rainbow(12))
```



It can be seen that as work experience increases, people are more likely to request loans. Notably, people with 10+ years of work experience tend to ask for considerably more loans.

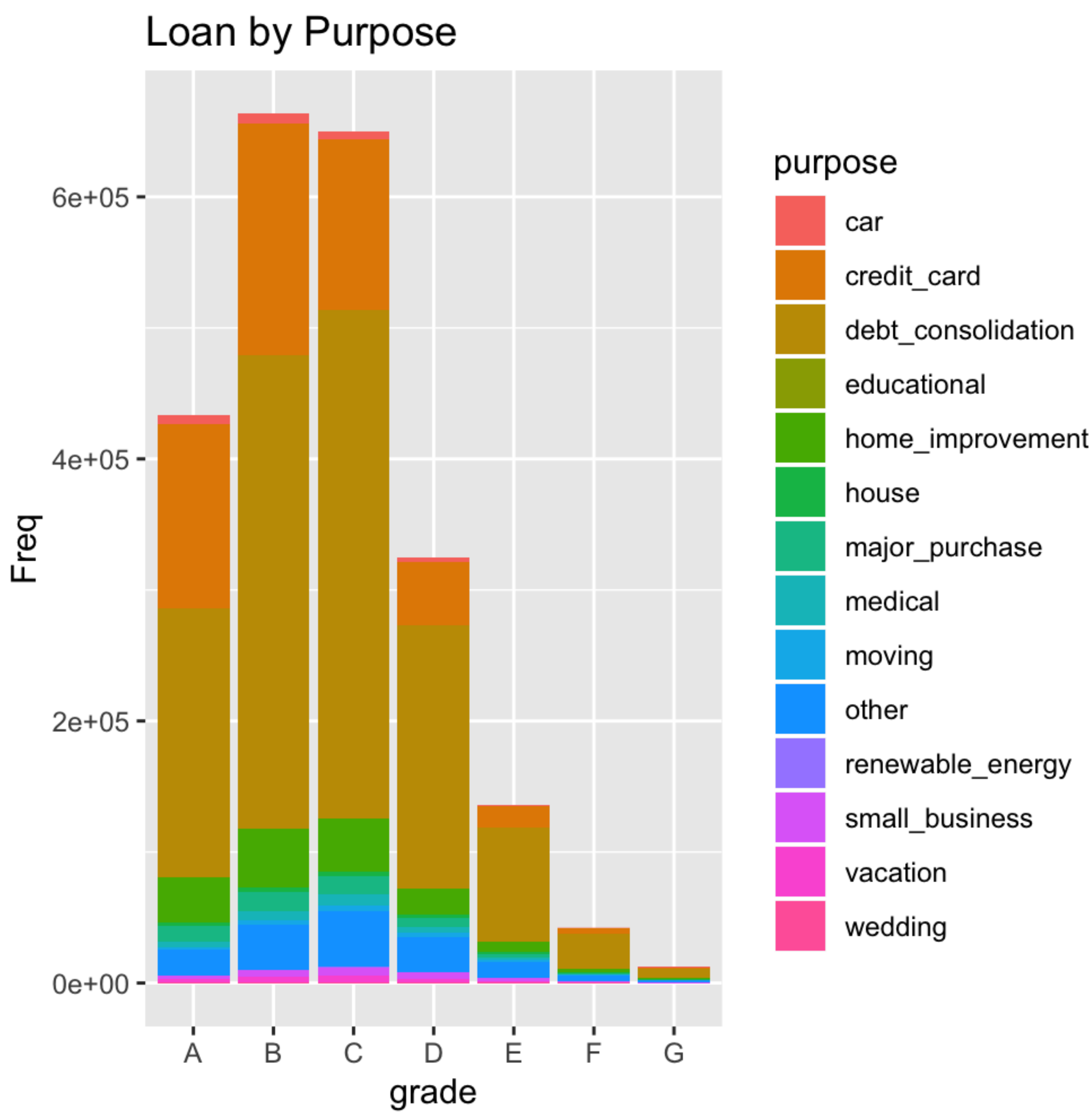
In [27]:

```
%%R -w 5 -h 5 --units in -r 200

a <- group_by(mydata, addr_state) %>% dplyr::count(addr_state) %>% set_colnames(c("region","count"))
a$region = sapply(state.name[match(a$region, state.abb)], tolower)
all_states <- map_data("state")
b <- merge(all_states, a, by = "region")
cnames <- aggregate(cbind(long, lat) ~ region, data = b, FUN = function(x) mean(range(x)))
ggplot(b, aes(x = long, y = lat, map_id = region)) + geom_map(aes(fill = count),
    map = all_states) + labs(title = "Loan by Regions", x = "long", y = "lat ") +
    scale_fill_gradientn("", colours = terrain.colors(10), guide = "legend") + geom_text(data = cnames,
    aes(long, lat, label = region), size = 2) + theme_bw()
# California has the most people that get a loan
```



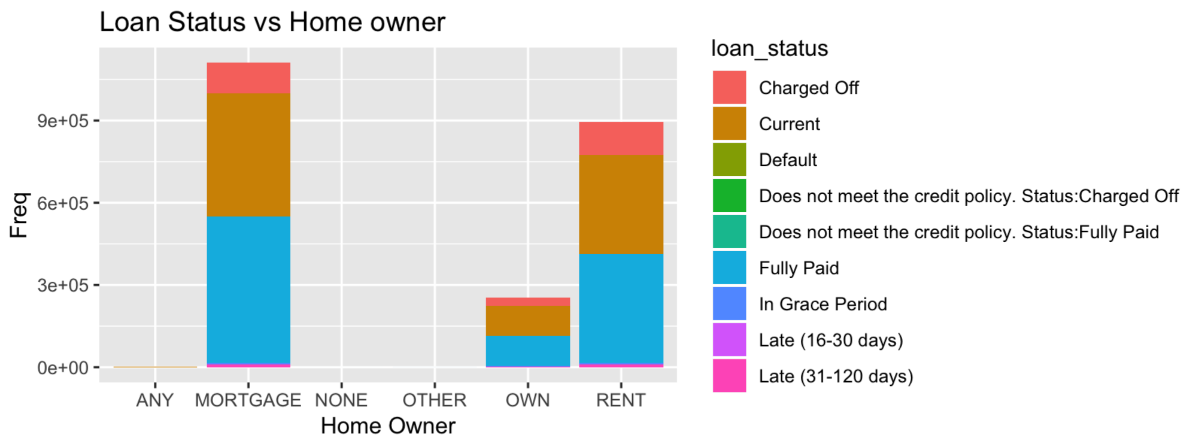
California has the most people asking for a loan. Texas and New York are the next biggest in terms of the number of people requesting loans. These three regions are some of the most populous in the United States, so it makes sense why this is the case.

In [28]:
```
%%R -w 5 -h 5 --units in -r 200

ggplot(data = mydata, aes(x = grade, fill = purpose)) + geom_bar() + ggt
itle("Loan by Purpose") +
    xlab("grade") + ylab("Freq")
## debt-consolidation and credit-card are the most 2 reasons for people
 to get
## loan and wedding and vacation are the least reasons for people to get
loan
```



Debt consolidation and paying off credit cards are the two most popular reasons for people to get loans. On the other hand, weddings and vacations are the least popular reasons for people to get loans.

In [34]:
```
%%R -w 8 -h 3 --units in -r 200

ggplot(data = mydata, aes(x = home_ownership, fill = loan_status)) + geo
m_bar() +
    ggtitle("Loan Status vs Home owner") + xlab("Home Owner") + ylab("Fr
eq")
# percentage of people own a home with a mostly paid off loan vs current
ly having
# a loan are almost the same
```



Homeowners with mortgages and renters have a similar breakdown in terms of the average status of their loans.

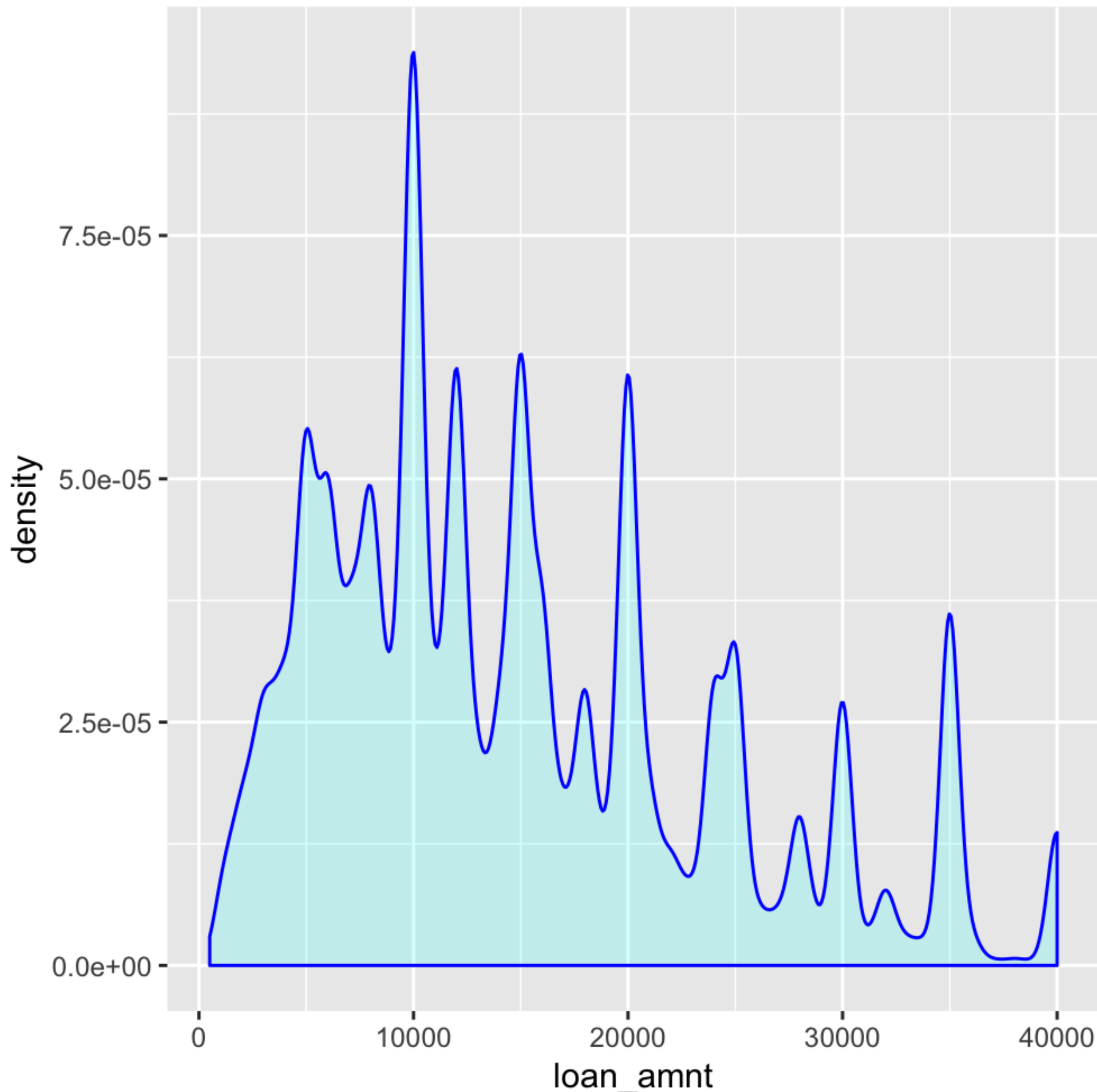
In [30]:
```
%%R -w 5 -h 5 --units in -r 200

c <- group_by(mydata, int_rate, grade) %>% dplyr::count(int_rate, grade)
%>% set_colnames(c("interest",
    "grade", "count"))
d <- ggplot(data = c, aes(x = interest, y = count)) + geom_smooth(aes(co
lour = grade,
    fill = grade, method = "loess")) + facet_wrap(~grade) + labs(x = "In
terest Rate",
    y = "Frequency")
ggplotly(d)
# we can see the interest rate in G and F grade is the most distributed
 which the
# interest rate fall at 6.63 and 6.62
```

```
R[write to console]: Warning:
R[write to console]:  Ignoring unknown aesthetics: method

R[write to console]: `geom_smooth()` using method = 'loess' and formula
'y ~ x'
```

Here, we see a correlation plot between annual income and loan amount, separated on cash and direct pay. Loans tend to be paid with direct pay more compared to cash.

In [31]:

```
%%R -w 5 -h 5 --units in -r 200

ggplot(data = mydata, aes(loan_amnt)) + geom_density(color = "blue", fil
l = "cyan",
    alpha = 0.2)
# based on the plot, the most loan amount is 10k
```



Based on the plot, the highest loan amount requested is $40,000. The shape of the graph is skewed right, signalling most of the loans requested are relatively small.

In [32]:

```
%%R -w 5 -h 5 --units in -r 200

mydata <- fread("data/loan.csv")

plot(mydata$loan_amnt, mydata$annual_inc)

loan_amnt = mydata$loan_amt[mydata$loan_amnt < 6000]
income = mydata$annual_inc[mydata$loan_amnt < 6000]

plot(loan_amnt[1:900], income[1:900],main="Income v.s. Loan amount",  xl
ab="Loan Amount", ylab="Income",col="purple")
#Annual Income v.s. Loan Amount

paid = mydata$loan_status=="Fully Paid"
grade = mydata$grade[mydata$loan_status=="Fully Paid"]

counts <- table(grade[1:900])
barplot(counts, main="Grade Distribution", col=c("lightblue","tan", "lig
htblue",  "tan","lightblue"),
       xlab="Grades of the loans that were paid off")
```

```
R[write to console]: Error in xy.coords(x, y, xlabel, ylabel, log) :
  'x' and 'y' lengths differ
Calls: <Anonymous> ... <Anonymous> -> withVisible -> plot -> plot.defau
lt -> xy.coords

R[write to console]: Error in dev.off() :
  QuartzBitmap_Output - unable to open file '/var/folders/98/yq68j11n04
s6_t2qdvm_4v9m0000gn/T/tmpryki02pb/Rplots001.png'
Calls: <Anonymous> -> <Anonymous> -> dev.off


Error in xy.coords(x, y, xlabel, ylabel, log) :
  'x' and 'y' lengths differ
Calls: <Anonymous> ... <Anonymous> -> withVisible -> plot -> plot.defau
lt -> xy.coords
```

```
---------------------------------------------------------------------------
----
RRuntimeError                             Traceback (most recent call l
ast)
<ipython-input-32-1cb4358ae80e> in <module>
----> 1 get_ipython().run_cell_magic('R', '-w 5 -h 5 --units in -r 200'
, '\nmydata <- fread("data/loan.csv")\n\nplot(mydata$loan_amnt, mydata
$annual_inc)\n\nloan_amnt = mydata$loan_amt[mydata$loan_amnt < 6000]\ni
ncome = mydata$annual_inc[mydata$loan_amnt < 6000]\n\nplot(loan_amnt[1:
900], income[1:900],main="Income v.s. Loan amount",  xlab="Loan Amoun
t", ylab="Income",col="purple")\n#Annual Income v.s. Loan Amount\n\npai
d = mydata$loan_status=="Fully Paid"\ngrade = mydata$grade[mydata$loan_
status=="Fully Paid"]\n\ncounts <- table(grade[1:900])\nbarplot(counts,
main="Grade Distribution", col=c("lightblue","tan", "lightblue",  "ta
n","lightblue"),\n       xlab="Grades of the loans that were paid off")
\n')

/usr/local/lib/python3.7/site-packages/IPython/core/interactiveshell.py
in run_cell_magic(self, magic_name, line, cell)
   2350                 with self.builtin_trap:
   2351                     args = (magic_arg_s, cell)
-> 2352                     result = fn(*args, **kwargs)
   2353                 return result
   2354 

</usr/local/lib/python3.7/site-packages/decorator.py:decorator-gen-130>
in R(self, line, cell, local_ns)

/usr/local/lib/python3.7/site-packages/IPython/core/magic.py in <lambda
>(f, *a, **k)
    185     # but it's overkill for just that one bit of state.
    186     def magic_deco(arg):
--> 187         call = lambda f, *a, **k: f(*a, **k)
    188 
    189         if callable(arg):

/usr/local/lib/python3.7/site-packages/rpy2/ipython/rmagic.py in R(sel
f, line, cell, local_ns)
    750         finally:
    751             if self.device in ['png', 'svg']:
--> 752                 ro.r('dev.off()')
    753 
    754         if text_output:

/usr/local/lib/python3.7/site-packages/rpy2/robjects/__init__.py in __c
all__(self, string)
    387     def __call__(self, string):
    388         p = _rparse(text=StrSexpVector((string,)))
--> 389         res = self.eval(p)
    390         return conversion.rpy2py(res)
    391 

/usr/local/lib/python3.7/site-packages/rpy2/robjects/functions.py in __
call__(self, *args, **kwargs)
    190                 kwargs[r_k] = v
    191         return (super(SignatureTranslatedFunction, self)
--> 192                 .__call__(*args, **kwargs))
```

```
    193
    194

/usr/local/lib/python3.7/site-packages/rpy2/robjects/functions.py in __
call__(self, *args, **kwargs)
    119             else:
    120                 new_kwargs[k] = conversion.py2rpy(v)
--> 121         res = super(Function, self).__call__(*new_args, **new_k
wargs)
    122         res = conversion.rpy2py(res)
    123         return res

/usr/local/lib/python3.7/site-packages/rpy2/rinterface_lib/conversion.p
y in _(*args, **kwargs)
     26 def _cdata_res_to_rinterface(function):
     27     def _(*args, **kwargs):
---> 28         cdata = function(*args, **kwargs)
     29         # TODO: test cdata is of the expected CType
     30         return _cdata_to_rinterface(cdata)

/usr/local/lib/python3.7/site-packages/rpy2/rinterface.py in __call__(s
elf, *args, **kwargs)
    771                     error_occured))
    772             if error_occured[0]:
--> 773                 raise embedded.RRuntimeError(_rinterface._geter
rmessage())
    774         return res
    775

RRuntimeError: Error in dev.off() :
  QuartzBitmap_Output - unable to open file '/var/folders/98/yq68j11n04
s6_t2qdvm_4v9m0000gn/T/tmpryki02pb/Rplots001.png'
Calls: <Anonymous> -> <Anonymous> -> dev.off
```

Data exploration

In [38]:

```
import pandas as pd
import sklearn as sk
```

In [39]:

```
data = pd.read_csv("data/loan.csv")
```

```
/usr/local/lib/python3.7/site-packages/IPython/core/interactiveshell.p
y:3049: DtypeWarning: Columns (19,47,55,112,123,124,125,128,129,130,13
3,139,140,141) have mixed types. Specify dtype option on import or set
low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

In [7]:
```
print(data.columns)
data.shape
```

```
Index(['id', 'member_id', 'loan_amnt', 'funded_amnt', 'funded_amnt_in
v',
       'term', 'int_rate', 'installment', 'grade', 'sub_grade',
       ...
       'hardship_payoff_balance_amount', 'hardship_last_payment_amoun
t',
       'disbursement_method', 'debt_settlement_flag',
       'debt_settlement_flag_date', 'settlement_status', 'settlement_da
te',
       'settlement_amount', 'settlement_percentage', 'settlement_ter
m'],
      dtype='object', length=145)
```

Out[7]:
```
(2260668, 145)
```

If a person requires a loan, build a model to predict whether the guy is trustworthy enough to get fully funded based on his credit score, salary, state of residence, etc? For simplicity, you can set "1" if the guy can get full money, otherwise set "0". So this becomes a 2-class classification problem

To be able to understand if the loan was a success" (fully payed off on time) or not ,we have to understand what each label means:

Charge off means that the original creditor has given up on being repaid according to the original terms of the loan. It considers the remaining balance to be bad debt, but that doesn't mean you no longer owe the amount that has not been repaid. In grace period:Still in time to pay but late Late:havent payed full amount on time Current:in process Fully paid: Paid on time

For the purpose of creating a model we decided to remove the current loans(the ones that are still in procces) since we dont know if this will end in "fail to pay" or "fully payed" so we will create a model using all rows besideds thew current ones.

To be able to categorize this model we will treat the status of loan of the remaining rows as:

If is fully paid we will assign the category of 1 and if the loan is in a status of anything else we will treat it as the category 0. Which will be the metric that determine if we should lean to his indivvual or not.

We will be using a diferent tyopes of modles to be able to create a model that can categorizre if a new costumer will pay off the loan. Based on that information we should determine if we should loan the amount of not.

First we will create dummy variables for the nominal categories:

In [4]:
```
data=data[data['loan_status']!="Current"]
data_dummis=pd.get_dummies(data[["issue_month","emp_length","term","grade","home_ownership","purpose","addr_state","application_type","disbursement_method"]])
print(data_dummis.shape)
data_dummis.columns
```

```
(1340973, 96)
```

```
Out[4]: Index(['issue_month', 'emp_length_1 year', 'emp_length_10+ years',
               'emp_length_2 years', 'emp_length_3 years', 'emp_length_4 year
        s',
               'emp_length_5 years', 'emp_length_6 years', 'emp_length_7 year
        s',
               'emp_length_8 years', 'emp_length_9 years', 'emp_length_< 1 yea
        r',
               'term_36 months', 'term_60 months', 'grade_A', 'grade_B', 'grade
        _C',
               'grade_D', 'grade_E', 'grade_F', 'grade_G', 'home_ownership_AN
        Y',
               'home_ownership_MORTGAGE', 'home_ownership_NONE',
               'home_ownership_OTHER', 'home_ownership_OWN', 'home_ownership_RE
        NT',
               'purpose_car', 'purpose_credit_card', 'purpose_debt_consolidatio
        n',
               'purpose_educational', 'purpose_home_improvement', 'purpose_hous
        e',
               'purpose_major_purchase', 'purpose_medical', 'purpose_moving',
               'purpose_other', 'purpose_renewable_energy', 'purpose_small_busi
        ness',
               'purpose_vacation', 'purpose_wedding', 'addr_state_AK', 'addr_st
        ate_AL',
               'addr_state_AR', 'addr_state_AZ', 'addr_state_CA', 'addr_state_C
        O',
               'addr_state_CT', 'addr_state_DC', 'addr_state_DE', 'addr_state_F
        L',
               'addr_state_GA', 'addr_state_HI', 'addr_state_IA', 'addr_state_I
        D',
               'addr_state_IL', 'addr_state_IN', 'addr_state_KS', 'addr_state_K
        Y',
               'addr_state_LA', 'addr_state_MA', 'addr_state_MD', 'addr_state_M
        E',
               'addr_state_MI', 'addr_state_MN', 'addr_state_MO', 'addr_state_M
        S',
               'addr_state_MT', 'addr_state_NC', 'addr_state_ND', 'addr_state_N
        E',
               'addr_state_NH', 'addr_state_NJ', 'addr_state_NM', 'addr_state_N
        V',
               'addr_state_NY', 'addr_state_OH', 'addr_state_OK', 'addr_state_O
        R',
               'addr_state_PA', 'addr_state_RI', 'addr_state_SC', 'addr_state_S
        D',
               'addr_state_TN', 'addr_state_TX', 'addr_state_UT', 'addr_state_V
        A',
               'addr_state_VT', 'addr_state_WA', 'addr_state_WI', 'addr_state_W
        V',
               'addr_state_WY', 'application_type_Individual',
               'application_type_Joint App', 'disbursement_method_Cash',
               'disbursement_method_DirectPay'],
              dtype='object')
```

In [5]:
```python
data1=pd.concat([data, data_dummis], axis=1)
data1.shape
```

Out[5]: (1340973, 127)

In [65]:
```python
pd.set_option('display.max_rows', 500)
pd.set_option('display.max_columns', 500)
pd.set_option('display.width', 1000)
display(data1.sort_values(by=['ID']).head())
```

	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	grade	emp_length
100	30000	30000	30000.0	36 months	22.35	1151.16	D	5 years
152	40000	40000	40000.0	60 months	16.14	975.71	C	< 1 year
170	20000	20000	20000.0	36 months	7.56	622.68	A	10+ years
186	4500	4500	4500.0	36 months	11.31	147.99	B	10+ years
215	8425	8425	8425.0	36 months	27.27	345.18	E	3 years

In [7]:
```python
"we drop the title since there are to many categories, also since we hav
e income that should have a direct relation the the position"

data1=data1.drop(["emp_title"], axis=1)
data1=data1.drop(["Unnamed: 0"], axis=1)
print(data1.shape)
data1.columns
```

```
(1340973, 125)
```

Out[7]:
```
Index(['loan_amnt', 'funded_amnt', 'funded_amnt_inv', 'term', 'int_rat
e', 'installment', 'grade', 'emp_length', 'home_ownership', 'annual_in
c',
       ...
       'addr_state_VA', 'addr_state_VT', 'addr_state_WA', 'addr_state_W
I', 'addr_state_WV', 'addr_state_WY', 'application_type_Individual', 'a
pplication_type_Joint App', 'disbursement_method_Cash', 'disbursement_m
ethod_DirectPay'], dtype='object', length=125)
```

In [64]:
```
display(data1.head())
data1.shape
```

	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	grade	emp_length
100	30000	30000	30000.0	36 months	22.35	1151.16	D	5 years
152	40000	40000	40000.0	60 months	16.14	975.71	C	< 1 year
170	20000	20000	20000.0	36 months	7.56	622.68	A	10+ years
186	4500	4500	4500.0	36 months	11.31	147.99	B	10+ years
215	8425	8425	8425.0	36 months	27.27	345.18	E	3 years

Out[64]:
```
(1340973, 122)
```

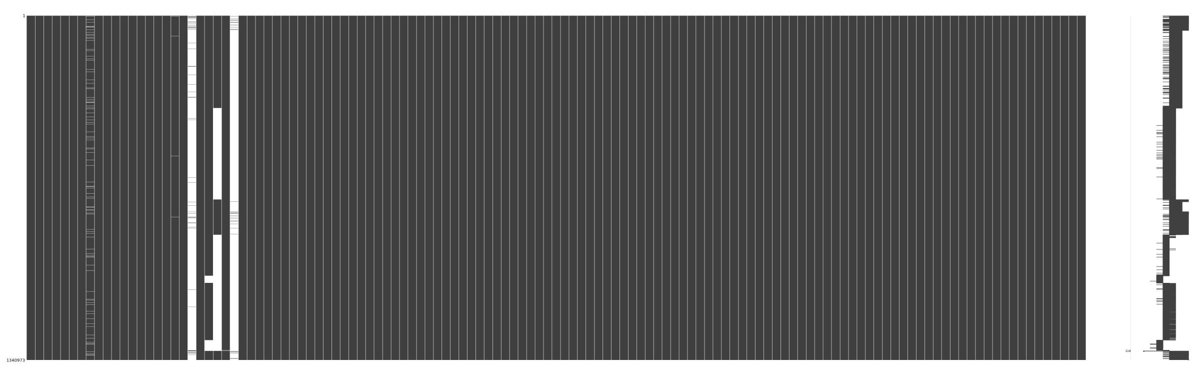
In [9]:
```
"if we delete all rows that have any soprt of NaN our datasa et becomes
 to small and to fit a machine learning model would be to hard"
"so we need to find another way to fix NAN"

data1_noNA = data1.dropna()
data1.shape
```

Out[9]:
```
(1340973, 125)
```

In [10]:
```
import missingno
```

In [11]:
```
missingno.matrix(data1, figsize=(100,30))
```

Out[11]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1998f9fbf60>
```



In [12]:

```
data1.isna().sum()
```

```
Out[12]: loan_amnt                          0
         funded_amnt                        0
         funded_amnt_inv                    0
         term                               0
         int_rate                           0
         installment                        0
         grade                              0
         emp_length                     78428
         home_ownership                     0
         annual_inc                         4
         issue_d                            0
         loan_status                        0
         purpose                            0
         addr_state                         0
         earliest_cr_line                  29
         open_acc                          29
         total_pymnt                        0
         last_pymnt_d                    2426
         last_pymnt_amnt                    0
         next_pymnt_d                 1303607
         application_type                   0
         tot_cur_bal                    70276
         total_bal_il                  809794
         pub_rec_bankruptcies            1365
         sec_app_mort_acc             1321075
         disbursement_method                0
         issue_year                         0
         issue_month                        0
         ID                                 0
         issue_month                        0
         emp_length_1 year                  0
         emp_length_10+ years               0
         emp_length_2 years                 0
         emp_length_3 years                 0
         emp_length_4 years                 0
         emp_length_5 years                 0
         emp_length_6 years                 0
         emp_length_7 years                 0
         emp_length_8 years                 0
         emp_length_9 years                 0
         emp_length_< 1 year                0
         term_36 months                     0
         term_60 months                     0
         grade_A                            0
         grade_B                            0
         grade_C                            0
         grade_D                            0
         grade_E                            0
         grade_F                            0
         grade_G                            0
         home_ownership_ANY                 0
         home_ownership_MORTGAGE            0
         home_ownership_NONE                0
         home_ownership_OTHER               0
         home_ownership_OWN                 0
         home_ownership_RENT                0
         purpose_car                        0
```

purpose_credit_card	0
purpose_debt_consolidation	0
purpose_educational	0
purpose_home_improvement	0
purpose_house	0
purpose_major_purchase	0
purpose_medical	0
purpose_moving	0
purpose_other	0
purpose_renewable_energy	0
purpose_small_business	0
purpose_vacation	0
purpose_wedding	0
addr_state_AK	0
addr_state_AL	0
addr_state_AR	0
addr_state_AZ	0
addr_state_CA	0
addr_state_CO	0
addr_state_CT	0
addr_state_DC	0
addr_state_DE	0
addr_state_FL	0
addr_state_GA	0
addr_state_HI	0
addr_state_IA	0
addr_state_ID	0
addr_state_IL	0
addr_state_IN	0
addr_state_KS	0
addr_state_KY	0
addr_state_LA	0
addr_state_MA	0
addr_state_MD	0
addr_state_ME	0
addr_state_MI	0
addr_state_MN	0
addr_state_MO	0
addr_state_MS	0
addr_state_MT	0
addr_state_NC	0
addr_state_ND	0
addr_state_NE	0
addr_state_NH	0
addr_state_NJ	0
addr_state_NM	0
addr_state_NV	0
addr_state_NY	0
addr_state_OH	0
addr_state_OK	0
addr_state_OR	0
addr_state_PA	0
addr_state_RI	0
addr_state_SC	0
addr_state_SD	0
addr_state_TN	0
addr_state_TX	0

```
addr_state_UT                          0
addr_state_VA                          0
addr_state_VT                          0
addr_state_WA                          0
addr_state_WI                          0
addr_state_WV                          0
addr_state_WY                          0
application_type_Individual            0
application_type_Joint App             0
disbursement_method_Cash               0
disbursement_method_DirectPay          0
dtype: int64
```

So we can see how we should get rid completly of atleast 3 columns since we dont have nought data

In [13]:
```
data1=data1.drop(["next_pymnt_d","total_bal_il","sec_app_mort_acc"], axi
s=1)
```

In [14]:
```
data1.isna().sum()
```

```
Out[14]: loan_amnt                          0
         funded_amnt                        0
         funded_amnt_inv                    0
         term                               0
         int_rate                           0
         installment                        0
         grade                              0
         emp_length                     78428
         home_ownership                     0
         annual_inc                         4
         issue_d                            0
         loan_status                        0
         purpose                            0
         addr_state                         0
         earliest_cr_line                  29
         open_acc                          29
         total_pymnt                        0
         last_pymnt_d                    2426
         last_pymnt_amnt                    0
         application_type                   0
         tot_cur_bal                    70276
         pub_rec_bankruptcies            1365
         disbursement_method                0
         issue_year                         0
         issue_month                        0
         ID                                 0
         issue_month                        0
         emp_length_1 year                  0
         emp_length_10+ years               0
         emp_length_2 years                 0
         emp_length_3 years                 0
         emp_length_4 years                 0
         emp_length_5 years                 0
         emp_length_6 years                 0
         emp_length_7 years                 0
         emp_length_8 years                 0
         emp_length_9 years                 0
         emp_length_< 1 year                0
         term_36 months                     0
         term_60 months                     0
         grade_A                            0
         grade_B                            0
         grade_C                            0
         grade_D                            0
         grade_E                            0
         grade_F                            0
         grade_G                            0
         home_ownership_ANY                 0
         home_ownership_MORTGAGE            0
         home_ownership_NONE                0
         home_ownership_OTHER               0
         home_ownership_OWN                 0
         home_ownership_RENT                0
         purpose_car                        0
         purpose_credit_card                0
         purpose_debt_consolidation         0
         purpose_educational                0
```

```
purpose_home_improvement           0
purpose_house                      0
purpose_major_purchase             0
purpose_medical                    0
purpose_moving                     0
purpose_other                      0
purpose_renewable_energy           0
purpose_small_business             0
purpose_vacation                   0
purpose_wedding                    0
addr_state_AK                      0
addr_state_AL                      0
addr_state_AR                      0
addr_state_AZ                      0
addr_state_CA                      0
addr_state_CO                      0
addr_state_CT                      0
addr_state_DC                      0
addr_state_DE                      0
addr_state_FL                      0
addr_state_GA                      0
addr_state_HI                      0
addr_state_IA                      0
addr_state_ID                      0
addr_state_IL                      0
addr_state_IN                      0
addr_state_KS                      0
addr_state_KY                      0
addr_state_LA                      0
addr_state_MA                      0
addr_state_MD                      0
addr_state_ME                      0
addr_state_MI                      0
addr_state_MN                      0
addr_state_MO                      0
addr_state_MS                      0
addr_state_MT                      0
addr_state_NC                      0
addr_state_ND                      0
addr_state_NE                      0
addr_state_NH                      0
addr_state_NJ                      0
addr_state_NM                      0
addr_state_NV                      0
addr_state_NY                      0
addr_state_OH                      0
addr_state_OK                      0
addr_state_OR                      0
addr_state_PA                      0
addr_state_RI                      0
addr_state_SC                      0
addr_state_SD                      0
addr_state_TN                      0
addr_state_TX                      0
addr_state_UT                      0
addr_state_VA                      0
addr_state_VT                      0
```

```
addr_state_WA                          0
addr_state_WI                          0
addr_state_WV                          0
addr_state_WY                          0
application_type_Individual            0
application_type_Joint App             0
disbursement_method_Cash               0
disbursement_method_DirectPay          0
dtype: int64
```

In [15]:
```
data2_noNA=data1.dropna()
data2_noNA.shape
```

Out[15]:
```
(1192258, 122)
```

In [16]:
```
"we only less than10% of our data set so its fine let keep going"
```

Out[16]:
```
'we only less than10% of our data set so its fine let keep going'
```

In [17]:
```
data2_noNA.head()
```

Out[17]:

	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	grade	emp_length
100	30000	30000	30000.0	36 months	22.35	1151.16	D	5 years
152	40000	40000	40000.0	60 months	16.14	975.71	C	< 1 year
170	20000	20000	20000.0	36 months	7.56	622.68	A	10+ years
186	4500	4500	4500.0	36 months	11.31	147.99	B	10+ years
215	8425	8425	8425.0	36 months	27.27	345.18	E	3 years

In [18]:
```
"already creted dumy variable so it is fine to drop this columns"
data2_noNA=data2_noNA.drop(["last_pymnt_amnt","issue_month","term","grad
e","home_ownership","issue_d","purpose","addr_state","earliest_cr_line",
"application_type","disbursement_method","issue_month"], axis=1)
```

In [19]:
```
data2_noNA.shape
```

Out[19]:
```
(1192258, 110)
```

In [20]:
```
"convert fully payed to 1 otherwise to 0"
data2_noNA['y'] = np.where(data2_noNA['loan_status']=='Fully Paid', 1, 0
)
```

total_pymnt has a direct relation with what we are trying to estimte and is somtehing we wont know with new custumers so we get rid of it. last_pymnt_d is something we will not know from new custumers emp_length already creted dumy variables

In [21]:
```
data2_noNA = data2_noNA.drop('loan_status', axis=1) # data
data2_noNA = data2_noNA.drop('last_pymnt_d', axis=1) # data
data2_noNA = data2_noNA.drop('emp_length', axis=1) # data
data2_noNA = data2_noNA.drop('total_pymnt', axis=1) # data


data2_noNA.head()
```

Out[21]:

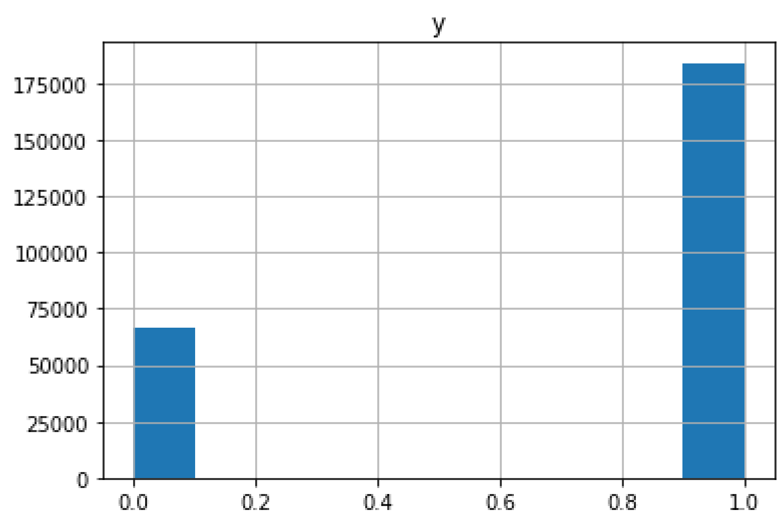
	loan_amnt	funded_amnt	funded_amnt_inv	int_rate	installment	annual_inc	open_acc	tot_(
100	30000	30000	30000.0	22.35	1151.16	100000.0	11.0	47
152	40000	40000	40000.0	16.14	975.71	45000.0	18.0	27
170	20000	20000	20000.0	7.56	622.68	100000.0	9.0	51
186	4500	4500	4500.0	11.31	147.99	38500.0	12.0	2
215	8425	8425	8425.0	27.27	345.18	450000.0	21.0	69

In [22]:
```
data2_noNA.shape
data2_noNA.index = range(1192258)
```

Reduce data size since pc will crash if we try to run model with this amount of data

In [23]:
```
data3_noNA=data2_noNA.loc[1:250000,:]
data3_noNA.shape
data3_noNA
data3_noNA.hist(column='y')
```

Out[23]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x000001998E62
A048>]],
      dtype=object)
```

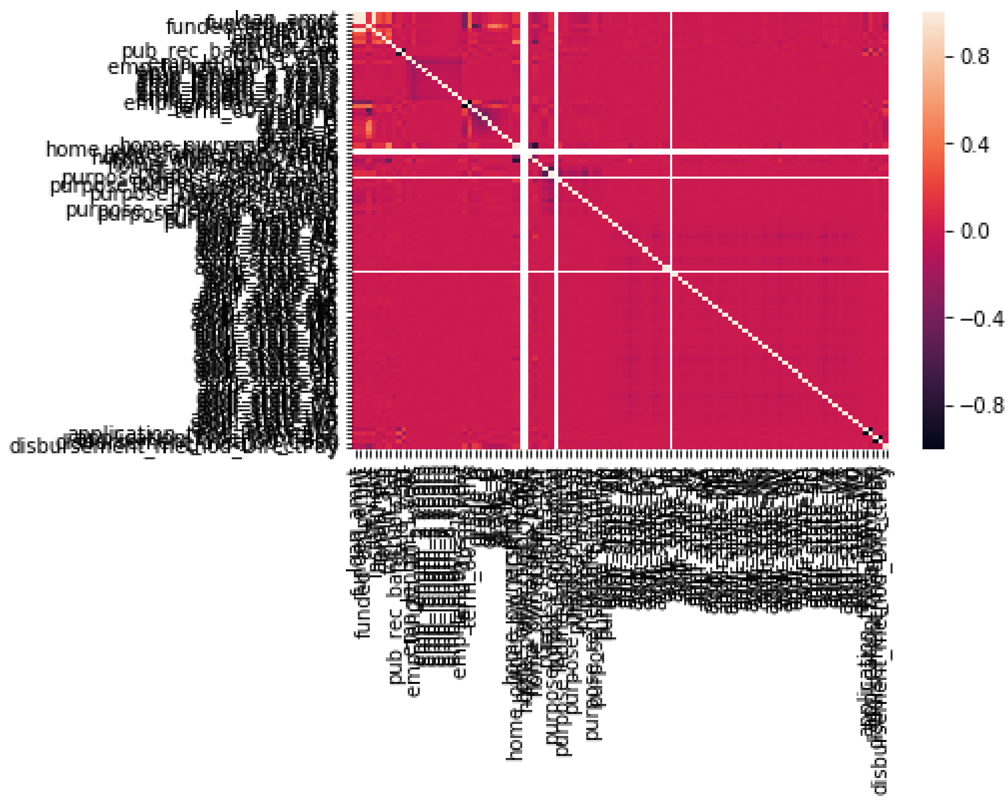


In [46]:
```
data3_noNA.shape
```

Out[46]:
```
(250000, 107)
```

In [40]:
```
import seaborn as sns
corr = data3_noNA.corr()
sns.heatmap(corr, xticklabels = corr.columns.values ,yticklabels = corr.
columns.values)
```

Out[40]: <matplotlib.axes._subplots.AxesSubplot at 0x19990f22a90>

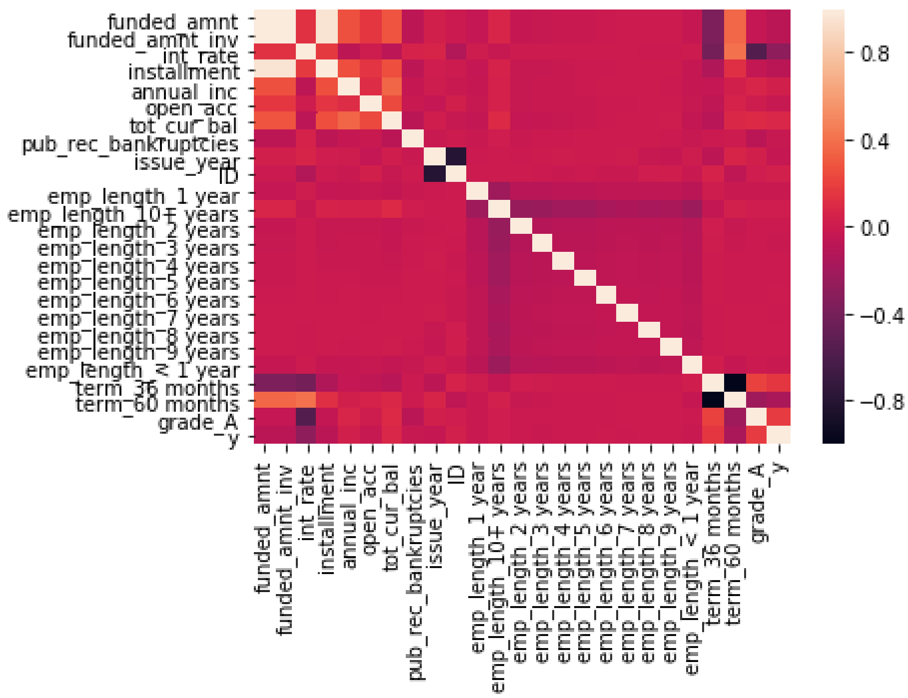


In [45]:

```
import numpy as np
from sklearn.linear_model import LinearRegression
import statsmodels.formula.api as sm

list1=[]
list1=list(range(1,25))
list1.append(106)

data3_noNA1=data3_noNA.iloc[:,list1]

corr = data3_noNA1.corr()
sns.heatmap(corr, xticklabels = corr.columns.values ,yticklabels = corr.
columns.values)
```

Out[45]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x199ce451e10>
```

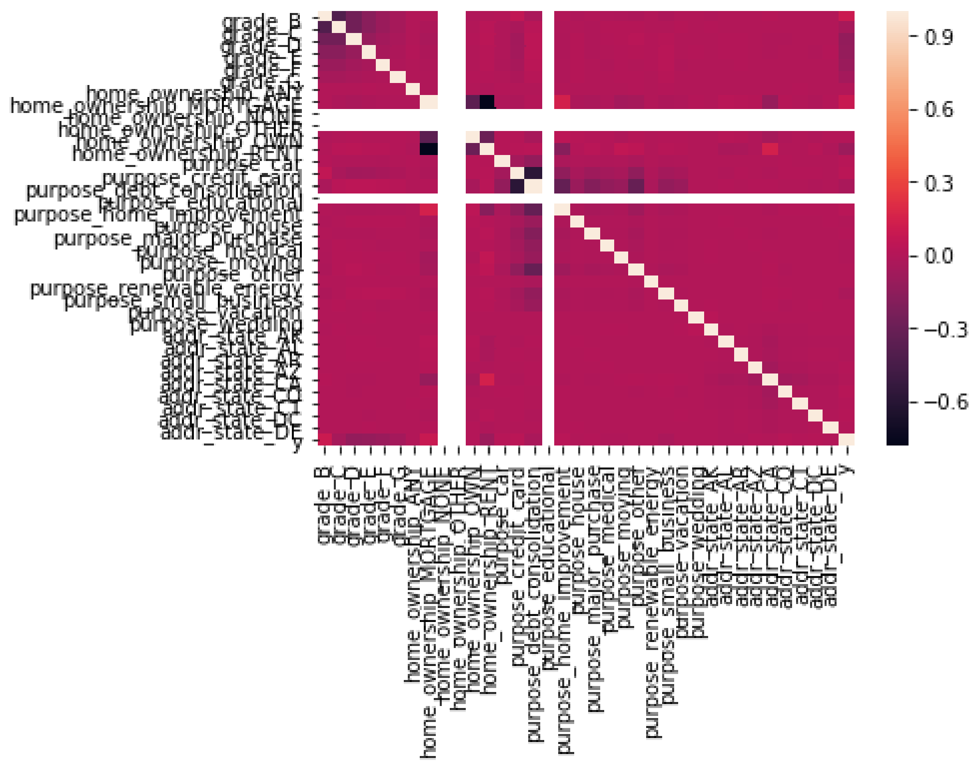


In [48]:

```
list1=[]
list1=list(range(25,60))
list1.append(106)

data3_noNA1=data3_noNA.iloc[:,list1]

corr = data3_noNA1.corr()
sns.heatmap(corr, xticklabels = corr.columns.values ,yticklabels = corr.
columns.values)
```

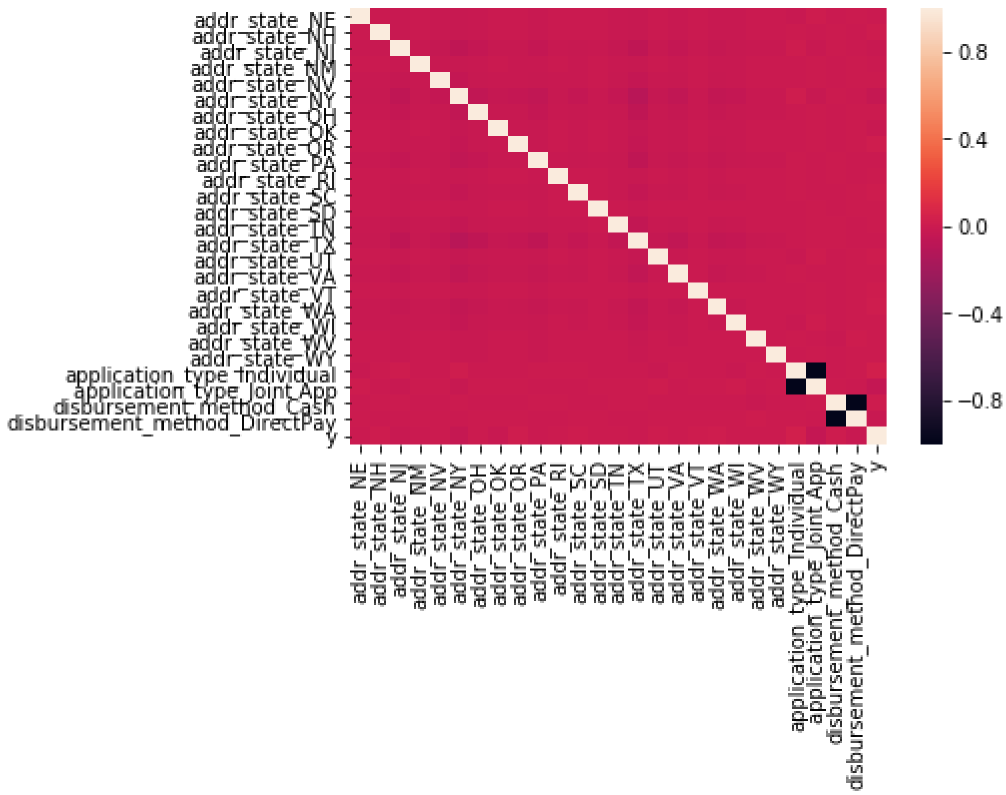
Out[48]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x199cef8a898>
```



In [47]:
```
list1=[]
list1=list(range(80,106))
list1.append(106)

data3_noNA1=data3_noNA.iloc[:,list1]

corr = data3_noNA1.corr()
sns.heatmap(corr, xticklabels = corr.columns.values ,yticklabels = corr.
columns.values)
```

Out[47]: <matplotlib.axes._subplots.AxesSubplot at 0x199ce47b198>



In [49]:
```
y_train = data3_noNA.y
X_train = data3_noNA.drop('y', axis=1)
```

In [50]:
```
X_train.head()
```

Out[50]:

	loan_amnt	funded_amnt	funded_amnt_inv	int_rate	installment	annual_inc	open_acc	tot_cur
1	40000	40000	40000.0	16.14	975.71	45000.0	18.0	2710
2	20000	20000	20000.0	7.56	622.68	100000.0	9.0	5157
3	4500	4500	4500.0	11.31	147.99	38500.0	12.0	291
4	8425	8425	8425.0	27.27	345.18	450000.0	21.0	6903
5	20000	20000	20000.0	17.97	507.55	57000.0	10.0	333

In [51]:
```
print(y_train.shape)
X_train.shape
```

```
(250000,)
```

Out[51]:
```
(250000, 106)
```

X_train is our data set and the test data it should be proiveded by kaggle so we will divide this train data to shceck our accuracy

In [52]:
```
import math, time, random, datetime

# Data Manipulation
import numpy as np
import pandas as pd

# Visualization
import matplotlib.pyplot as plt
import missingno
import seaborn as sns
plt.style.use('seaborn-whitegrid')

# Preprocessing
from sklearn.preprocessing import OneHotEncoder, LabelEncoder, label_bin
arize

# Machine learning
from sklearn.model_selection import train_test_split
from sklearn import model_selection, tree, preprocessing, metrics, linea
r_model
from sklearn.svm import LinearSVC
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LinearRegression, LogisticRegression, S
GDClassifier
from sklearn.tree import DecisionTreeClassifier

# Let's be rebels and ignore warnings for now
import warnings
warnings.filterwarnings('ignore')
```

In [53]:
```
def fit_ml_algo(algo, X_train, y_train, cv):

    # One Pass
    model = algo.fit(X_train, y_train)
    acc = round(model.score(X_train, y_train) * 100, 2)

    # Cross Validation
    train_pred = model_selection.cross_val_predict(algo,
                                                   X_train,
                                                   y_train,
                                                   cv=cv,
                                                   n_jobs = -1)
    # Cross-validation accuracy metric
    acc_cv = round(metrics.accuracy_score(y_train, train_pred) * 100, 2)

    return train_pred, acc, acc_cv
```

In [54]:
```
# Logistic Regression
train_pred_log, acc_log, acc_cv_log = fit_ml_algo(LogisticRegression(),
                                                  X_train,
                                                  y_train,
                                                  10)


print("Accuracy: %s" % acc_log)
print("Accuracy CV 10-Fold: %s" % acc_cv_log)
```

```
Accuracy: 73.56
Accuracy CV 10-Fold: 73.22
```

In [55]:
```
train_pred_knn, acc_knn, acc_cv_knn = fit_ml_algo(KNeighborsClassifier
(),
                                                  X_train,
                                                  y_train,
                                                  10)
print("Accuracy: %s" % acc_knn)
print("Accuracy CV 10-Fold: %s" % acc_cv_knn)
```

```
Accuracy: 77.68
Accuracy CV 10-Fold: 40.72
```

In [56]:
```
# Stochastic Gradient Descent
train_pred_sgd, acc_sgd, acc_cv_sgd = fit_ml_algo(SGDClassifier(),
                                                  X_train,
                                                  y_train,
                                                  10)
print("Accuracy: %s" % acc_sgd)
print("Accuracy CV 10-Fold: %s" % acc_cv_sgd)
```

```
Accuracy: 73.33
Accuracy CV 10-Fold: 70.12
```

In [57]:
```
 X_train, X_test, y_train, y_test = train_test_split(X_train, y_train, t
est_size=0.33, random_state=42)
```

In [63]:
```
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_mldata
from sklearn.neural_network import MLPClassifier
mlp = MLPClassifier(hidden_layer_sizes=(10, 10), max_iter=1000, alpha=1e
-4,
                    solver='sgd', verbose=10, tol=1e-4, random_state=1,
                    learning_rate_init=.1,activation='logistic' )
mlp.fit(X_train, y_train)
print("Training set score: %f" % mlp.score(X_train, y_train))
print("Test set score: %f" % mlp.score(X_test, y_test))
mlp.classes_
```

```
Iteration 1, loss = 0.57859091
Iteration 2, loss = 0.57843190
Iteration 3, loss = 0.57855596
Iteration 4, loss = 0.57845613
Iteration 5, loss = 0.57833466
Iteration 6, loss = 0.57841682
Iteration 7, loss = 0.57832603
Iteration 8, loss = 0.57838674
Iteration 9, loss = 0.57828509
Iteration 10, loss = 0.57837514
Iteration 11, loss = 0.57830721
Iteration 12, loss = 0.57832793
Iteration 13, loss = 0.57828029
Training loss did not improve more than tol=0.000100 for 10 consecutive
epochs. Stopping.
Training set score: 0.735313
Test set score: 0.737200
```

Out[63]:
```
array([0, 1])
```

We can see that the best model is our n# Logistic Regression that gives us a 73% accuracy (cvv) to determine if the custumer will completely pay the loan on time or not.