

(Loan Data from Prosper)

by (Ahmed Tarek)

Introduction

On November 24, 2008, the SEC found Prosper to be in violation of the Securities Act of 1933. As a result of these findings, the SEC imposed a cease and desist order on Prosper ... In July 2009, Prosper reopened their website for lending ("investing") and borrowing after having obtained SEC registration for its loans ("notes"). After the relaunch, bidding on loans was restricted to residents of 28 U.S. states and the District of Columbia. Borrowers may reside in any of 47 states, with residents of three states (Iowa, Maine, and North Dakota) not permitted to borrow through Prosper

Investigation Overview

In this presentation I will attempt to answer some questions:

- What attributes affect the borrower's annual percentage rate?

This question is helpful for borrowers to decide on taking loans.

- What attributes affect a loan's outcome status?

This is very critical for banks to minimize the risk and to set the right interest rate.

- insight on increasing Prosper profits from loans.

Dataset Overview

This data set contains 113,937 loans with 81 variables on each loan, including loan amount, borrower rate (or interest rate), current loan status, borrower income, and many others.

```
In [1]: # import all packages and set plots to be embedded inline
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb
import warnings

warnings.filterwarnings('ignore')
%matplotlib inline
```

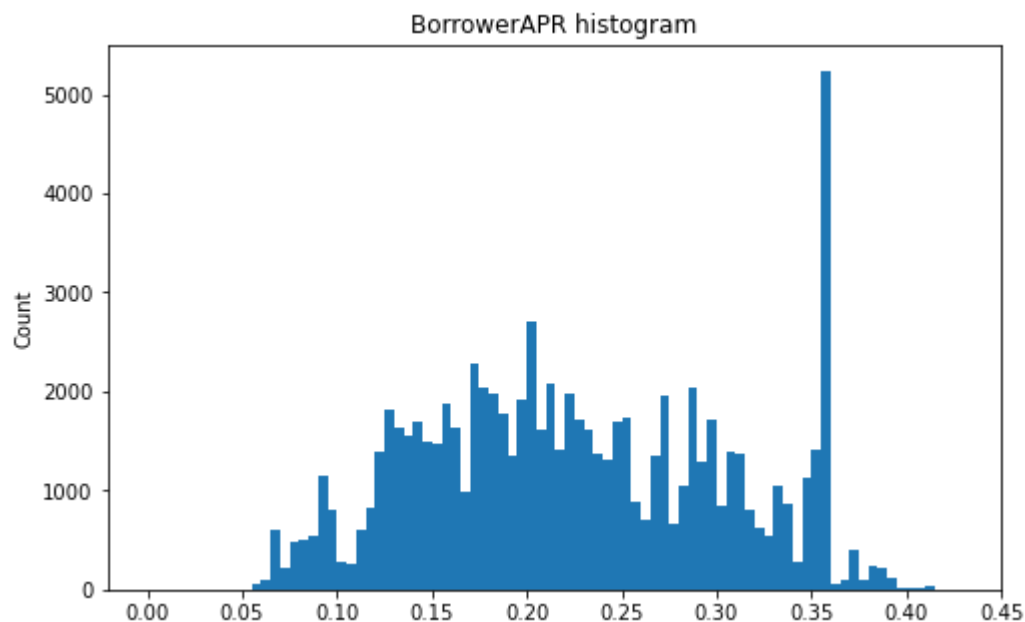
Univariate Exploration

```
In [2]: # reading data
df_loans = pd.read_csv('prosperLoanData_cleaned.csv')
```

Lets take a look at the BorrowerAPR

It appears that this distribution is multimodal with several peaks. A peak at 0.08, 0.28, 0.3, and an exceptionally high peak at 0.36.

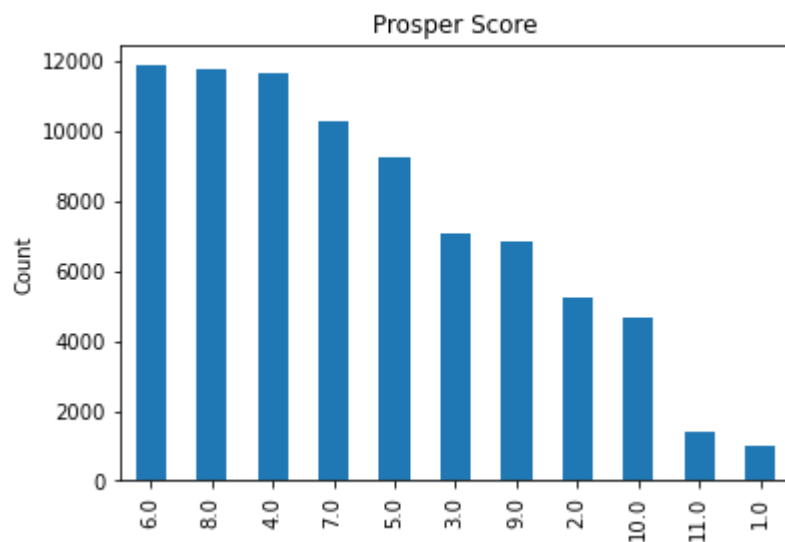
```
In [3]: bins = np.arange(0, df_loans.BorrowerAPR.max(), 0.005)
plt.figure(figsize=[8, 5])
plt.hist(data = df_loans, x = 'BorrowerAPR', bins = bins)
plt.xticks(np.arange(0, df_loans.BorrowerAPR.max()+0.05, 0.05))
plt.ylabel('Count')
plt.title('BorrowerAPR histogram');
```



Let's look at Prosper Score which is a custom risk score built using historical Prosper data.

Most of the borrower got Prosper Score of 6 that means they are safe to loan. Notice that even customers with a low risks score of 1 or 2 did get a loan. Not many borrowers received the highest score of 10.

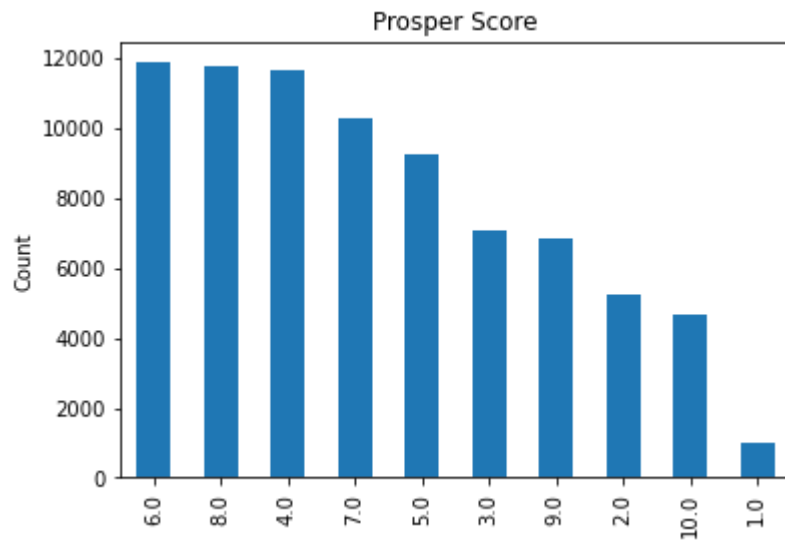
```
In [4]: def plot1():  
        df_loans.ProspersScore.value_counts().plot(kind='bar')  
        plt.ylabel('Count')  
        plt.title('Prospers Score');  
  
plot1()
```



This shows that there are scores above 10 which is not possible since the score is from 1-10. This data must be removed.

```
In [5]: df_loans = df_loans[df_loans.ProspersScore != 11]
```

In [6]: plot1()



Now lets look at ProsperRating (Alpha) and Occupation

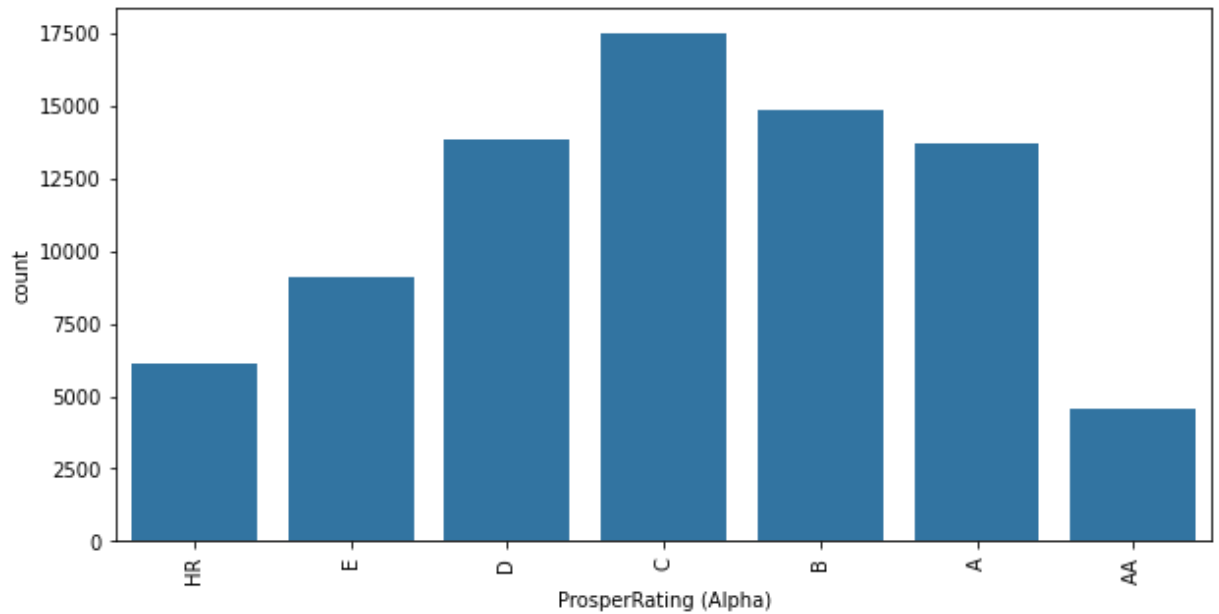
It appears that most borrowers were rated from C to A, and students are the least to take loans.

Before plotting, the ProsperRating (Alpha) should be ordered from low to high so there won't be any misleading visualization about the rating order

```
In [7]: rate_order = ['HR', 'E', 'D', 'C', 'B', 'A', 'AA']
ordered_var = pd.api.types.CategoricalDtype(ordered = True, categories = rate_order)
df_loans['ProsperRating (Alpha)'] = df_loans['ProsperRating (Alpha)'].astype(ordered_var)
```

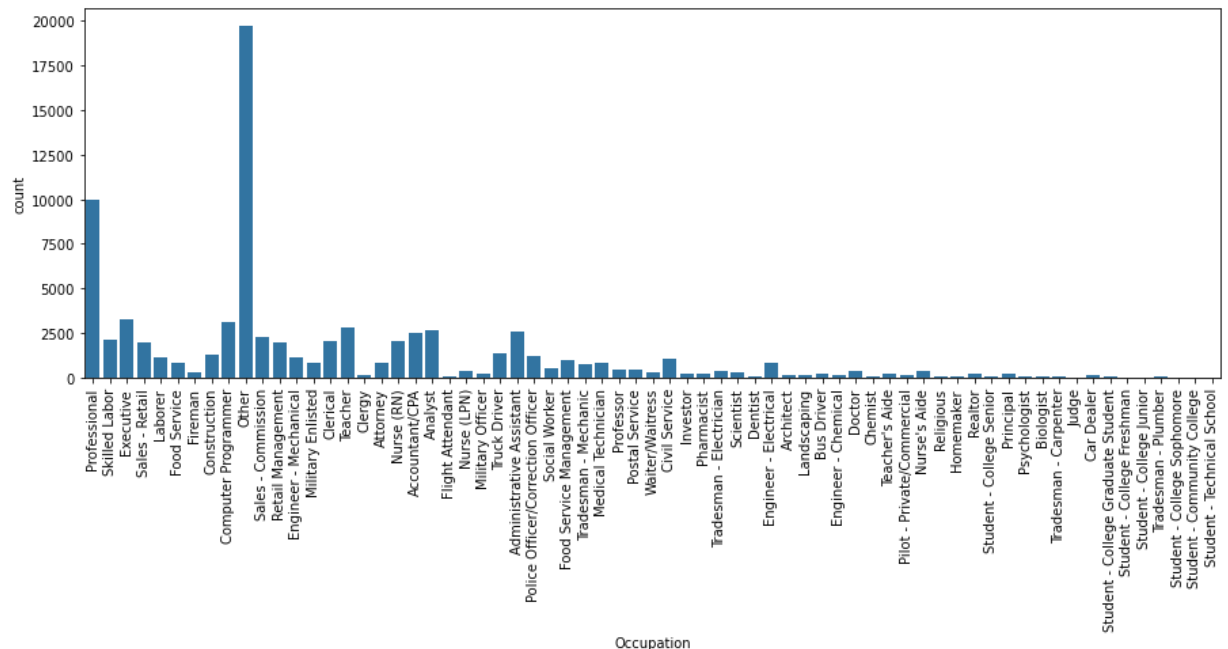
```
In [8]: fig = plt.subplots(figsize = [10, 5])

default_color = sb.color_palette()[0]
sb.countplot(data = df_loans, x = 'ProsperRating (Alpha)', color = default_color)
plt.xticks(rotation=90);
```



```
In [9]: fig = plt.subplots(figsize = [15, 5])

default_color = sb.color_palette()[0]
sb.countplot(data = df_loans, x = 'Occupation', color = default_color)
plt.xticks(rotation=90);
```



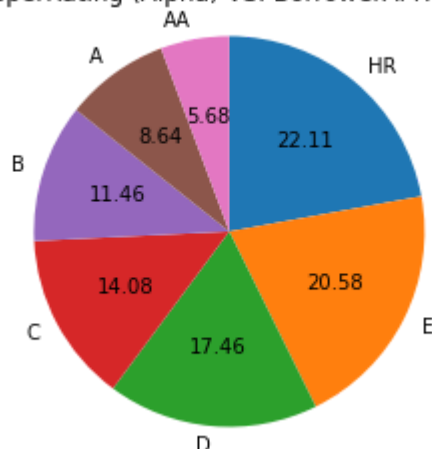
Now, let's compare the Prosper rating (Alpha) mean with the Borrower APR mean

Notice that the highest rating of AA received the lowest BorrowerAPR (5.68), while the lowest rating of HR received the highest BorrowerAPR (22.11). This shows that borrowers with higher ratings received lower BorrowerAPR.

```
In [10]: ProsperRatingAlpha_mean = df_loans.groupby('ProsperRating (Alpha)').BorrowerAPR.mean()

plt.pie(ProsperRatingAlpha_mean, labels = ProsperRatingAlpha_mean.index, startangle=90)
plt.axis('square')
plt.title('ProsperRating (Alpha) VS. BorrowerAPR mean');
```

ProsperRating (Alpha) VS. BorrowerAPR mean



I am interested in knowing more about the LoanOriginalAmount

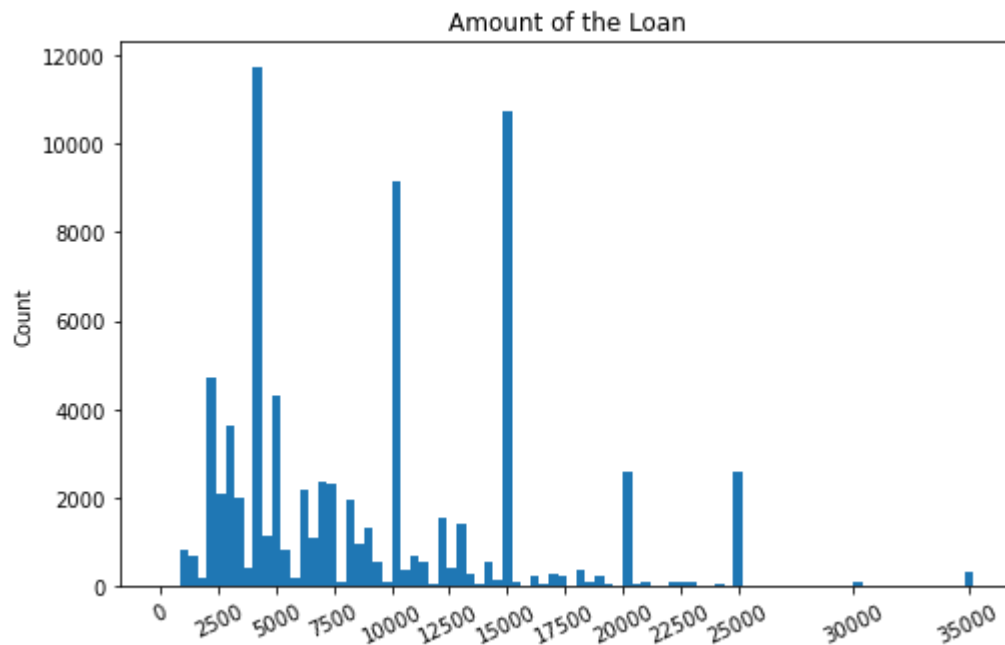
The histogram has several peaks at around 4,000, 10,000, and 15,000. But most of the values are in the lower end between 2500 and 10,000. The most loaned amounts are 4,000 and 15,000. The mean loan amount is between 8200.

```
In [11]: df_loans.LoanOriginalAmount.mean()
```

```
Out[11]: 9021.664034151547
```

```
In [12]: binsize = 400
bins = np.arange(0, df_loans.LoanOriginalAmount.max()+binsize, binsize)

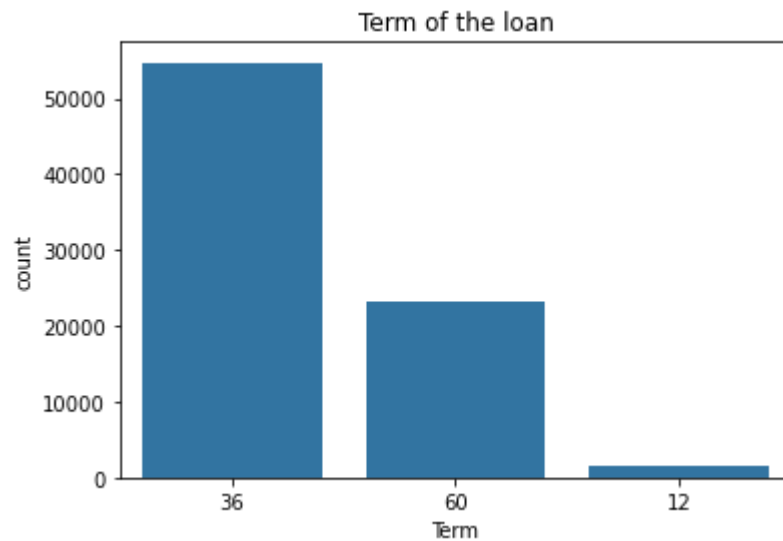
plt.figure(figsize=[8, 5])
plt.hist(data = df_loans, x = 'LoanOriginalAmount', bins = bins)
plt.ylabel('Count')
plt.title('Amount of the Loan')
plt.xticks([0,2500,5000,7500,10000,12500,15000,17500,20000,22500,25000,30000,35000])
```



Now lets see if there is a relation between the LoanOriginalAmount and Term

Most of the loans have a length of 36 months. The fact that most of the loans have a length of just 36 months might be correlated to the fact the average loan amount is around 9000 which can be paid back quickly.

```
In [13]: base_color = sb.color_palette()[0]
sb.countplot(data = df_loans, x='Term', color = base_color, order = df_loans.Term
plt.title('Term of the loan');
```



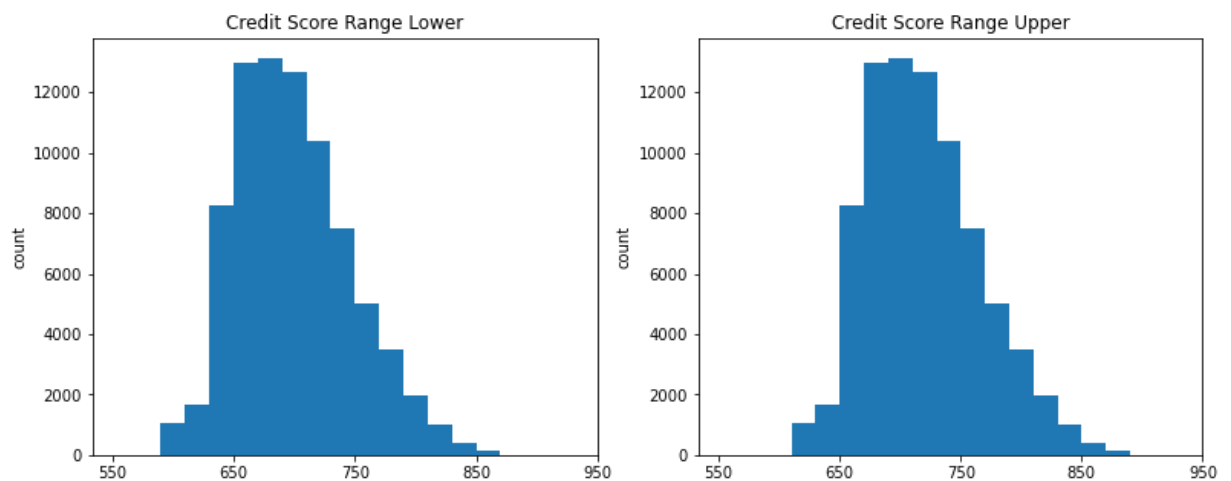
Lets look at the CreditScoreRangeLower and CreditScoreRangeUpper

The two histograms show similar trends, and there are no outliers that fall out of the range.


```
In [14]: plt.figure(figsize = [13, 5])

plt.subplot(1, 2, 1)
bins = np.arange(550, df_loans.CreditScoreRangeLower.max(), 20)
plt.hist(data = df_loans, x = 'CreditScoreRangeLower', bins = bins)
plt.xticks(np.arange(550, 1000, 100))
plt.title('Credit Score Range Lower')
plt.ylabel('count');

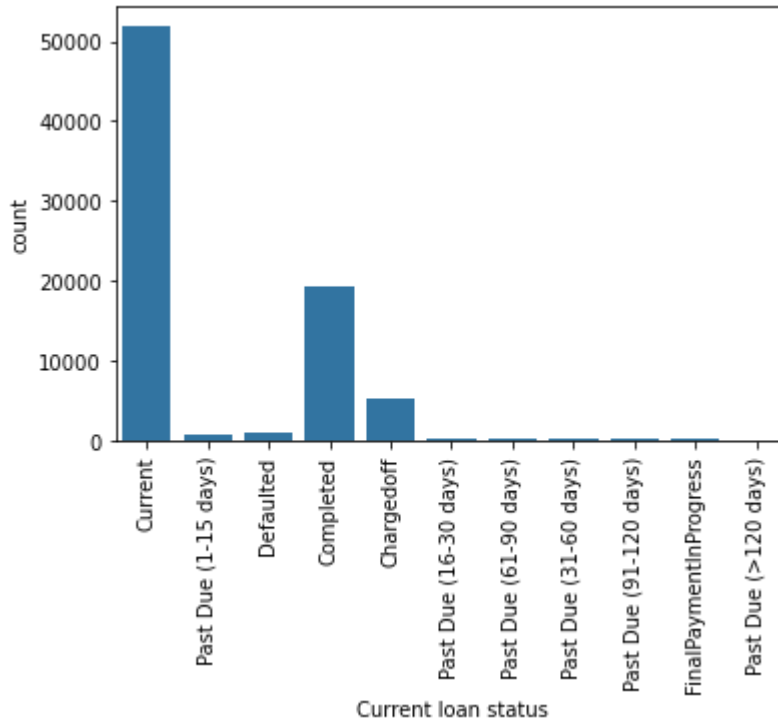
plt.subplot(1, 2, 2)
bins = np.arange(550, df_loans.CreditScoreRangeUpper.max(), 20)
plt.hist(data = df_loans, x = 'CreditScoreRangeUpper', bins = bins)
plt.xticks(np.arange(550, 1000, 100))
plt.title('Credit Score Range Upper')
plt.ylabel('count');
```



Now lets take a look at the loan status

Most of the loans are current or completed.

```
In [15]: def plot2():  
         base_color = sb.color_palette()[0]  
         sb.countplot(data = df_loans, x = 'LoanStatus', color = base_color)  
         plt.xlabel('Current loan status')  
         plt.xticks(rotation = 90);  
  
plot2()
```

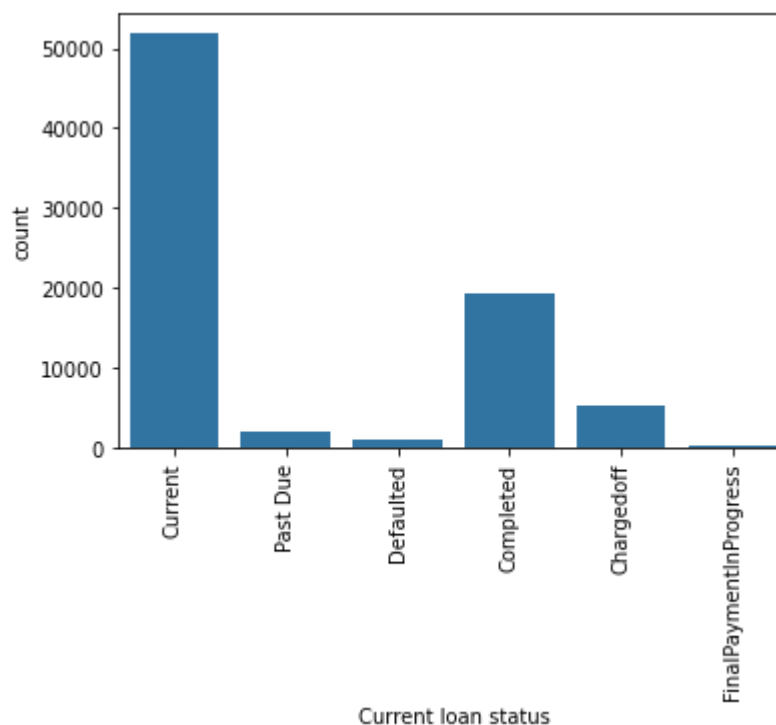


Combining the past dues into one column since it is not important to show that much information

```
In [16]: df_loans.LoanStatus = df_loans.LoanStatus.replace(['Past Due (1-15 days)', 'Past  
            'Past Due (91-120 days)', 'Past  
            'Past Due (>120 days)'], 'Past  
  
df_loans.LoanStatus.value_counts()
```

```
Out[16]: Current          51857  
Completed        19355  
Chargedoff         5191  
Past Due          2043  
Defaulted          999  
FinalPaymentInProgress    200  
Name: LoanStatus, dtype: int64
```

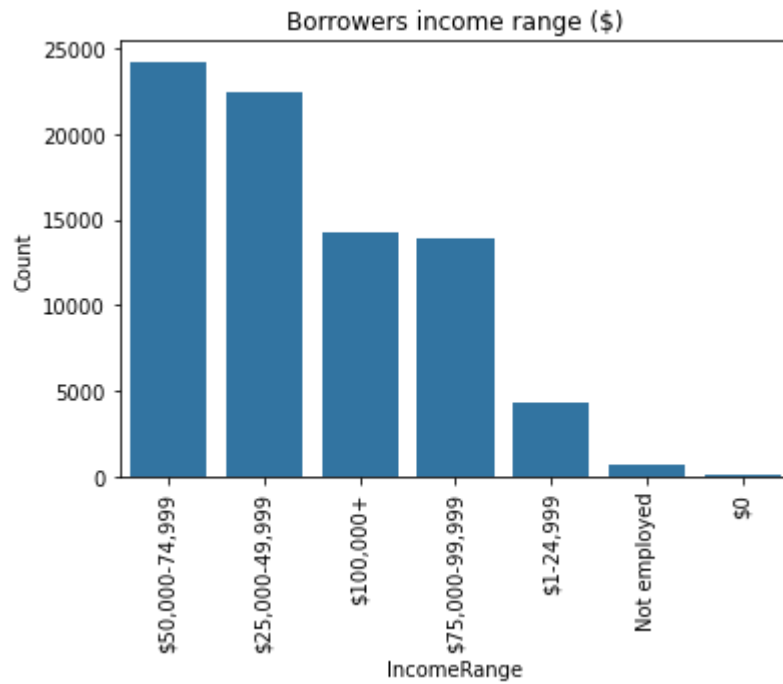
```
In [17]: plot2()
```



Now the IncomeRange

The income range of the borrowers shows that most of the loans were given to customers with an income between 25,000 and 74,999. Notice that people that are not employed, or have an income of 0 received a loan as well. They might be other criteria than the income that qualifies one to get a loan like being a student.

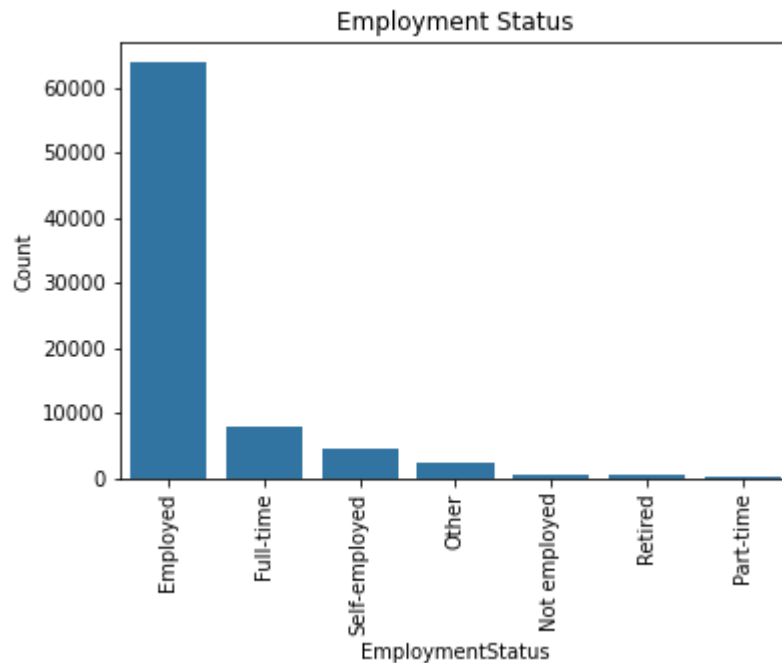
```
In [18]: sb.countplot(data= df_loans, x= 'IncomeRange', color= base_color, order = df_loans['IncomeRange'].value_counts().index)
plt.xticks(rotation=90)
plt.ylabel('Count')
plt.title('Borrowers income range ($)');
```



Lets check the EmploymentStatus to find out

Of course, most of the borrowers are employed, but the data shows that retired persons got a loan too.

```
In [19]: sb.countplot(data= df_loans, x= 'EmploymentStatus', color= base_color, order = df_loans['EmploymentStatus'].value_counts().index)
plt.xticks(rotation=90)
plt.ylabel('Count')
plt.title('Employment Status');
```

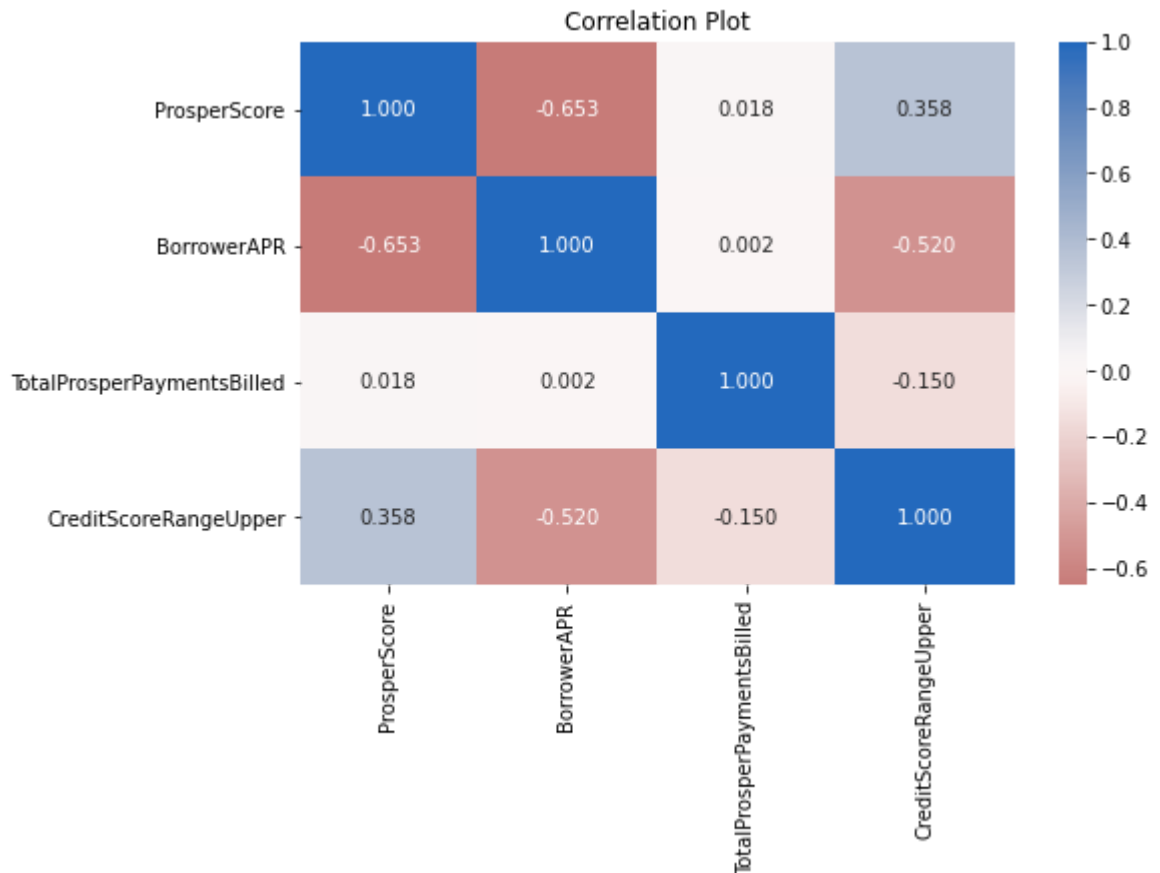


Bivariate Exploration

Lets look at the numeric variables

There are no strong positive relationships between any pairs. BorrowerAPR and ProsperScore are negative because borrowers with a lower score are more likely to pay higher APR. CreditScore and BorrowerAPR are also negative because the higher the borrowers CreditScore the more trustworthy they are, therefore they received lower APR.

```
In [20]: num_var = ['ProsperScore', 'BorrowerAPR', 'TotalProsperPaymentsBilled', 'CreditScoreRangeUpper']  
plt.figure(figsize = [8, 5])  
sb.heatmap(df_loans[num_var].corr(), annot = True, fmt = '.3f', cmap = 'vlag_r',  
plt.title('Correlation Plot');
```

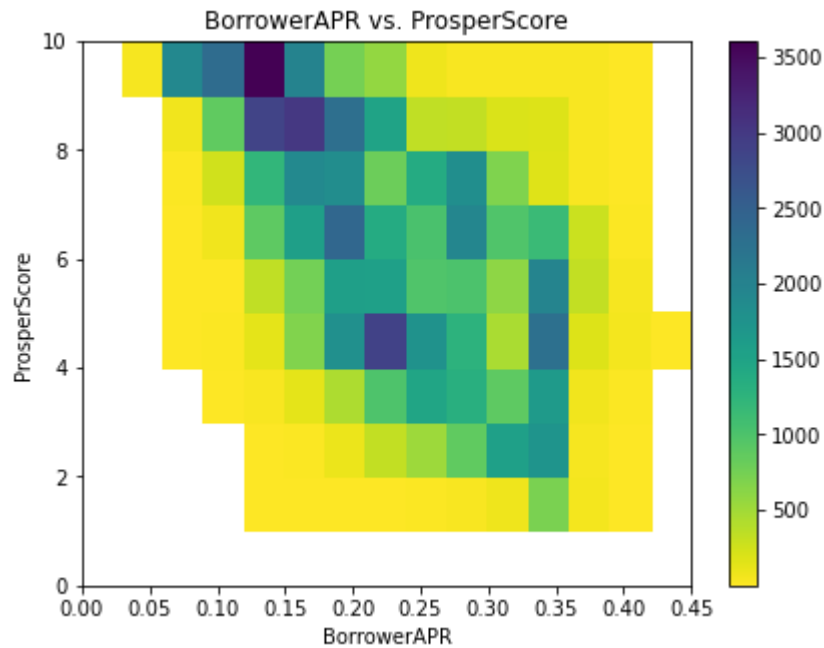


Lets look more closely at ProsperScore vs BorrowerAPR

This also proves that people with higher ratings tend to be more trustworthy and therefore given lower BorrowerAPR.

```
In [21]: plt.figure(figsize = [15, 5])

plt.subplot(1, 2, 2)
bins_x = np.arange(0, df_loans.BorrowerAPR.max()+0.05, 0.03)
bins_y = np.arange(0, df_loans.ProspersScore.max()+1, 1)
plt.hist2d(data = df_loans, x = 'BorrowerAPR', y = 'ProspersScore', bins = [bins_x, bins_y])
plt.colorbar()
plt.title('BorrowerAPR vs. ProspersScore')
plt.xlabel('BorrowerAPR')
plt.ylabel('ProspersScore');
```

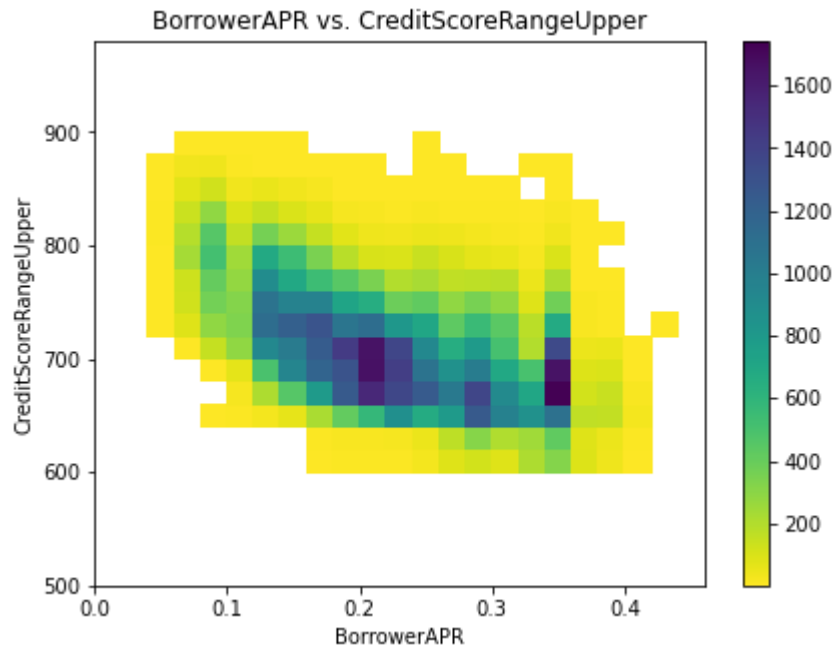


Now BorrowerAPR vs. CreditScoreRangeUpper

We can see the trend that the higher the CreditScore the lower the APR.

```
In [22]: plt.figure(figsize = [15, 5])

plt.subplot(1, 2, 2)
bins_x = np.arange(0, df_loans.BorrowerAPR.max()+0.05, 0.02)
bins_y = np.arange(500, df_loans.CreditScoreRangeUpper.max()+100, 20)
plt.hist2d(data = df_loans, x = 'BorrowerAPR', y = 'CreditScoreRangeUpper', bins
plt.colorbar()
plt.title('BorrowerAPR vs. CreditScoreRangeUpper')
plt.xlabel('BorrowerAPR')
plt.ylabel('CreditScoreRangeUpper');
```

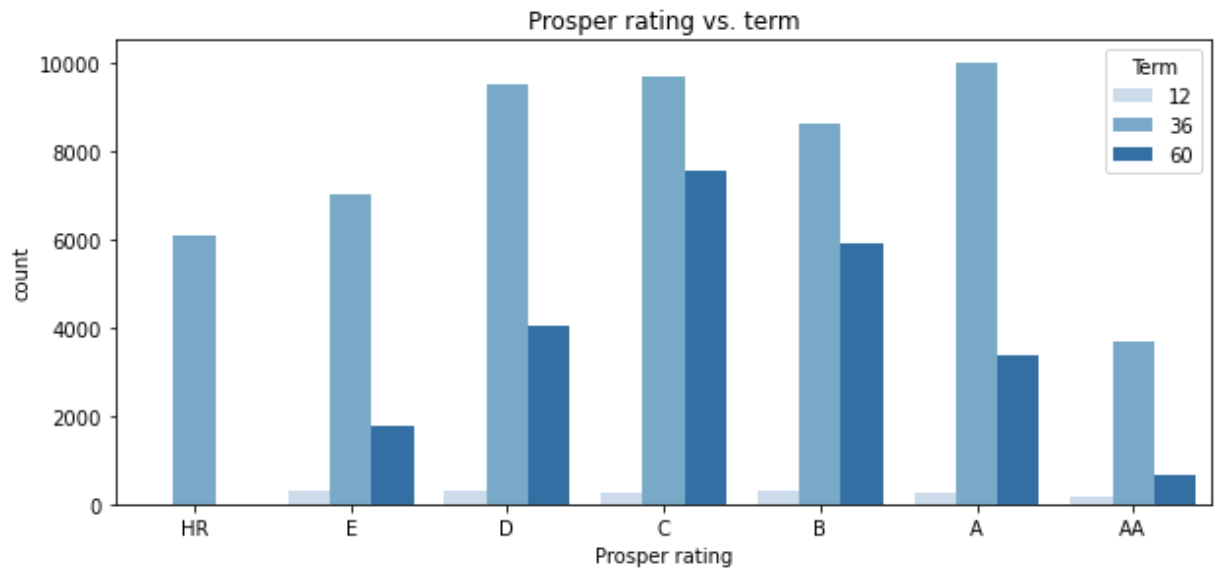


let's look at relationships between the categorical features.

We can see that there is an interaction between term and Prosper rating. The most popular term was 36 and most of the employed especially with full-time jobs took that kind of loan. And of course, the higher proper ratings were given to the employed personnel.

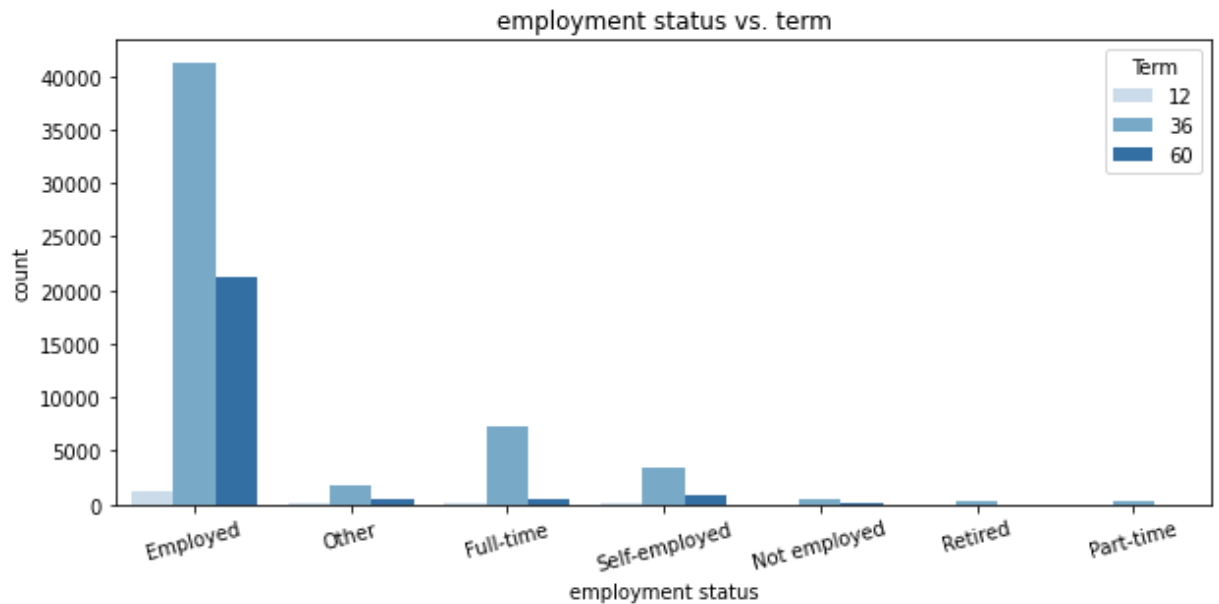

```
In [23]: plt.figure(figsize = [10, 20])

# Prosper rating vs. term
plt.subplot(4, 1, 1)
sb.countplot(data = df_loans, x = 'ProsperRating (Alpha)', hue = 'Term', palette
plt.xlabel('Prosper rating')
plt.title('Prosper rating vs. term');
```

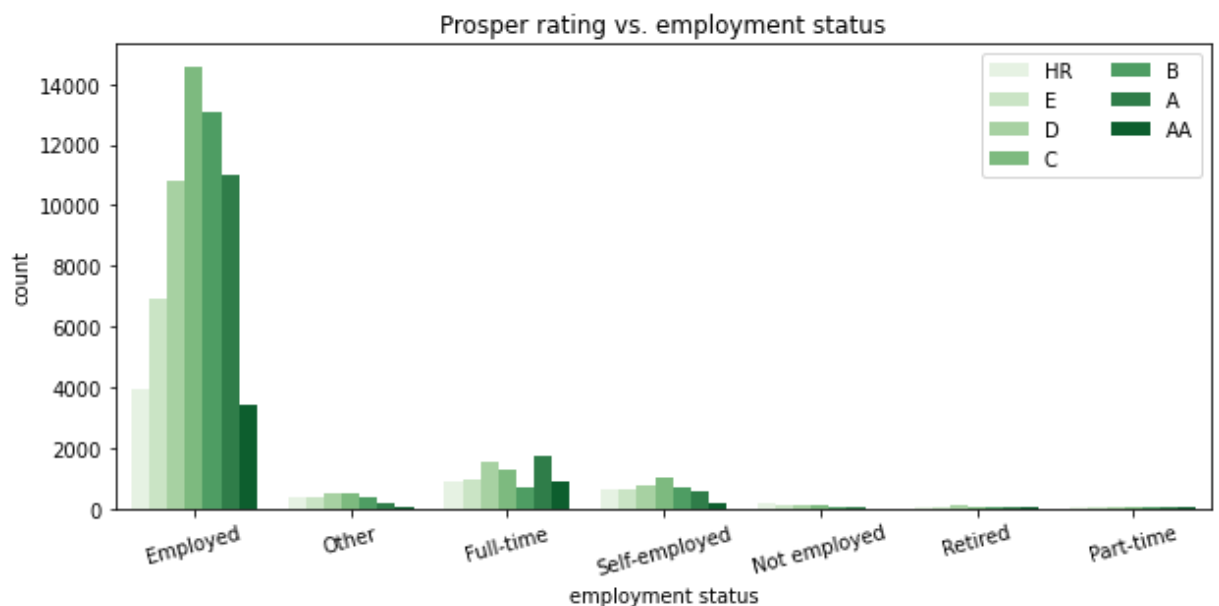


```
In [24]: # employment status vs. term
plt.figure(figsize = [10, 20])

ax = plt.subplot(4, 1, 2)
sb.countplot(data = df_loans, x = 'EmploymentStatus', hue = 'Term', palette = 'Blues')
plt.xticks(rotation = 15)
plt.xlabel('employment status')
plt.title('employment status vs. term');
```



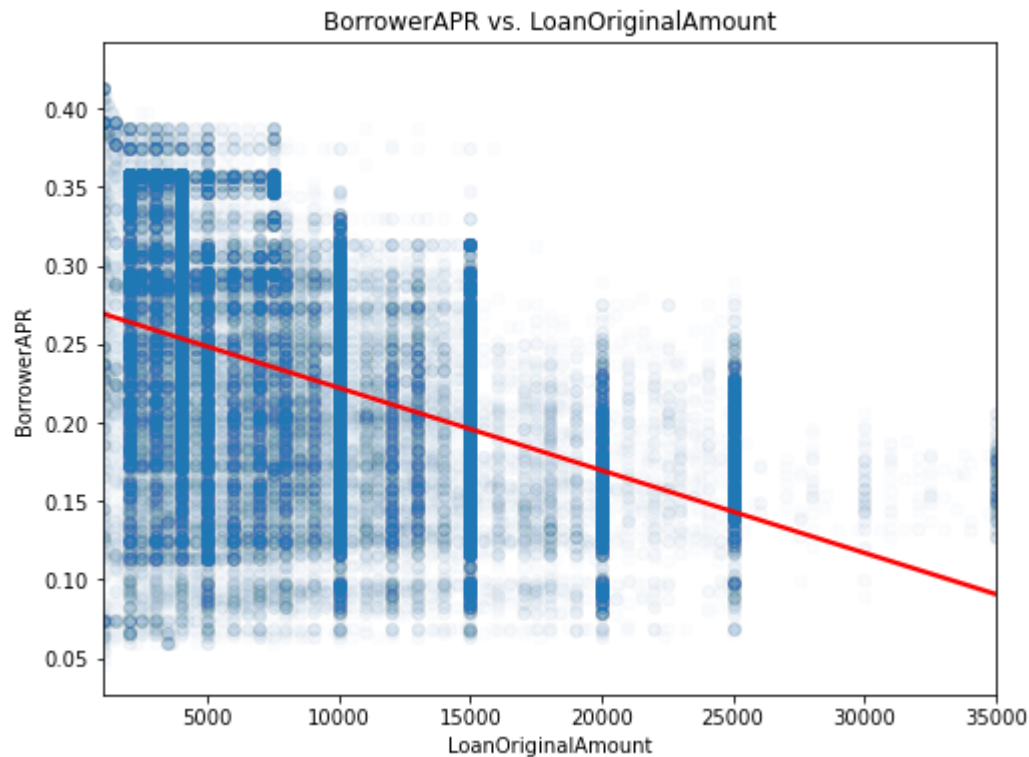
```
In [25]: # Prosper rating vs. employment status
plt.figure(figsize = [10, 20])
ax = plt.subplot(4, 1, 3)
sb.countplot(data = df_loans, x = 'EmploymentStatus', hue = 'ProsperRating (Alpha)', palette = 'Greens')
ax.legend(loc = 1, ncol = 2)
plt.xticks(rotation = 15)
plt.xlabel('employment status')
plt.title('Prosper rating vs. employment status');
```



Lets see how borrower APR and loan original amount are related

This relation shows that the range of APR decreases with the increase in the loan amount. Overall, the borrower's APR is negatively correlated with the loan amount.

```
In [26]: plt.figure(figsize = [8, 6])  
sb.regplot(data = df_loans, x = 'LoanOriginalAmount', y = 'BorrowerAPR', scatter_  
plt.title('BorrowerAPR vs. LoanOriginalAmount');
```

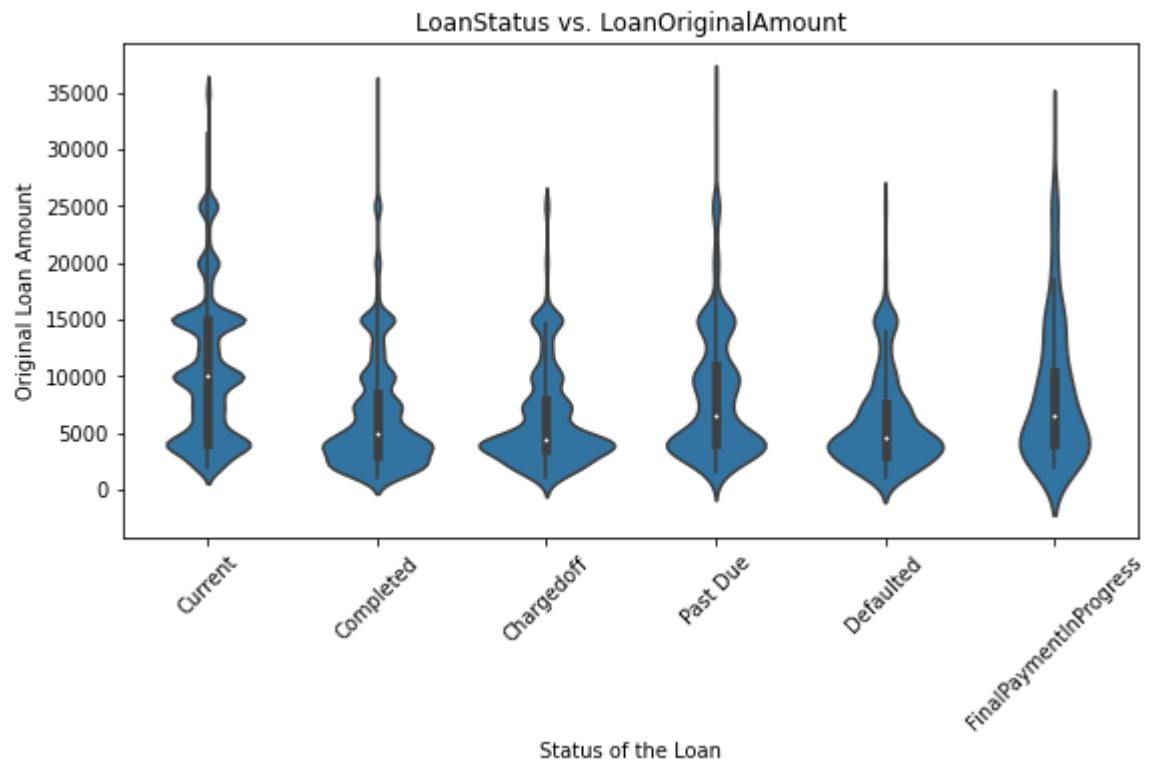


Now the relation between LoanStatus and LoanOriginalAmount

The original loan amount is about the same on average for loans that are completed, current. However, loans with past due payments have on average a higher original loan amount.

```
In [27]: plt.figure(figsize = [20, 10])

plt.subplot(2, 2, 2)
sb.violinplot(data = df_loans, x = 'LoanStatus', y = 'LoanOriginalAmount', color
plt.xticks(rotation=45)
plt.xlabel('Status of the Loan')
plt.ylabel('Original Loan Amount')
plt.title('LoanStatus vs. LoanOriginalAmount');
```

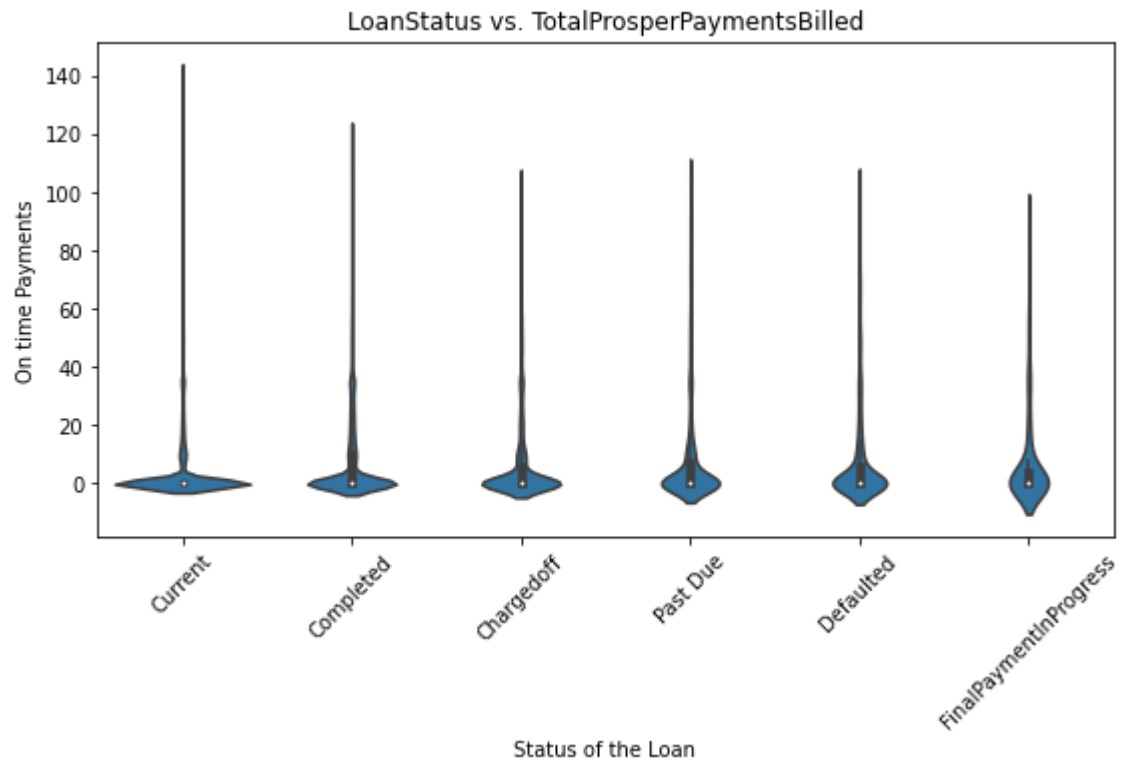


Now the relation between loan status and loan total prosper payments billed

Current loans have on average the highest number of on time payments while loans with the status charged-off and defaulted have the lowest.

```
In [28]: plt.figure(figsize = [20, 10])

plt.subplot(2, 2, 2)
sb.violinplot(data = df_loans, x = 'LoanStatus', y = 'TotalProsperPaymentsBilled')
plt.xticks(rotation=45)
plt.xlabel('Status of the Loan')
plt.ylabel('On time Payments')
plt.title('LoanStatus vs. TotalProsperPaymentsBilled');
```

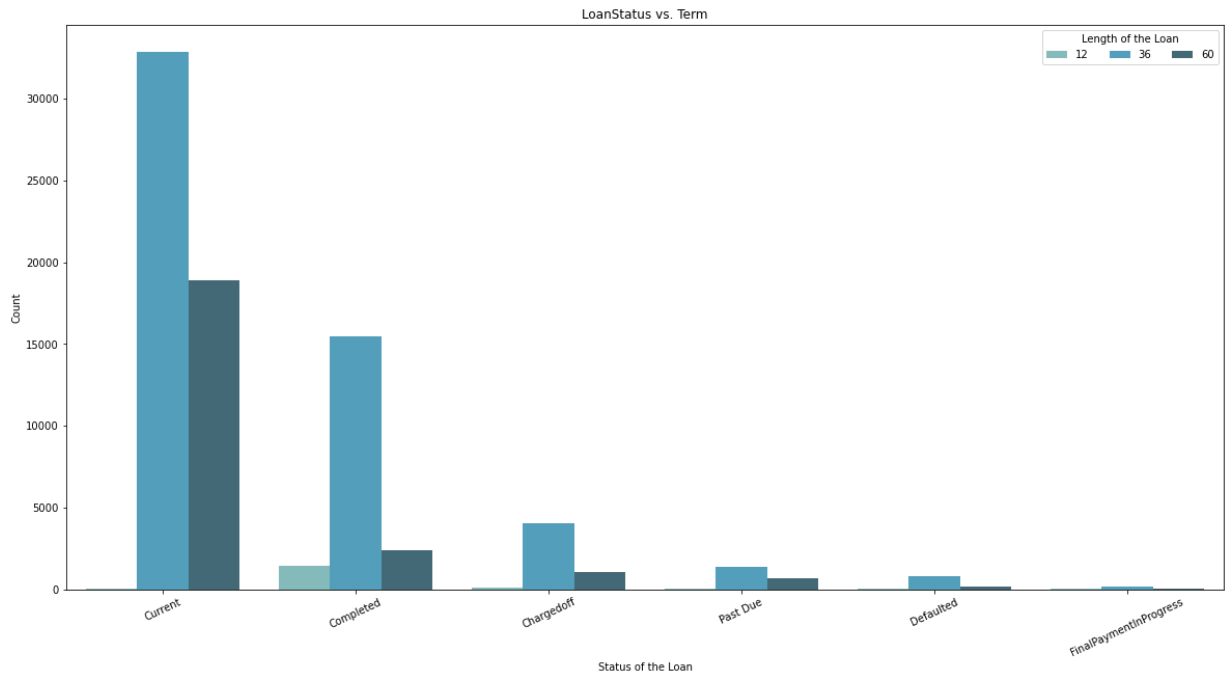


Relation between loan status and term

No matter what status a loan has, the most common length is 36 months and the least common is 60 months.

```
In [29]: plt.figure(figsize = [20,10])

ax = sb.countplot(data = df_loans, x = 'LoanStatus', hue = 'Term', palette = "GnBu")
plt.legend(loc = 1, ncol = 3, title = 'Length of the Loan')
plt.xticks(rotation = 25)
plt.xlabel('Status of the Loan')
ax.set_ylabel('Count')
plt.title('LoanStatus vs. Term');
```

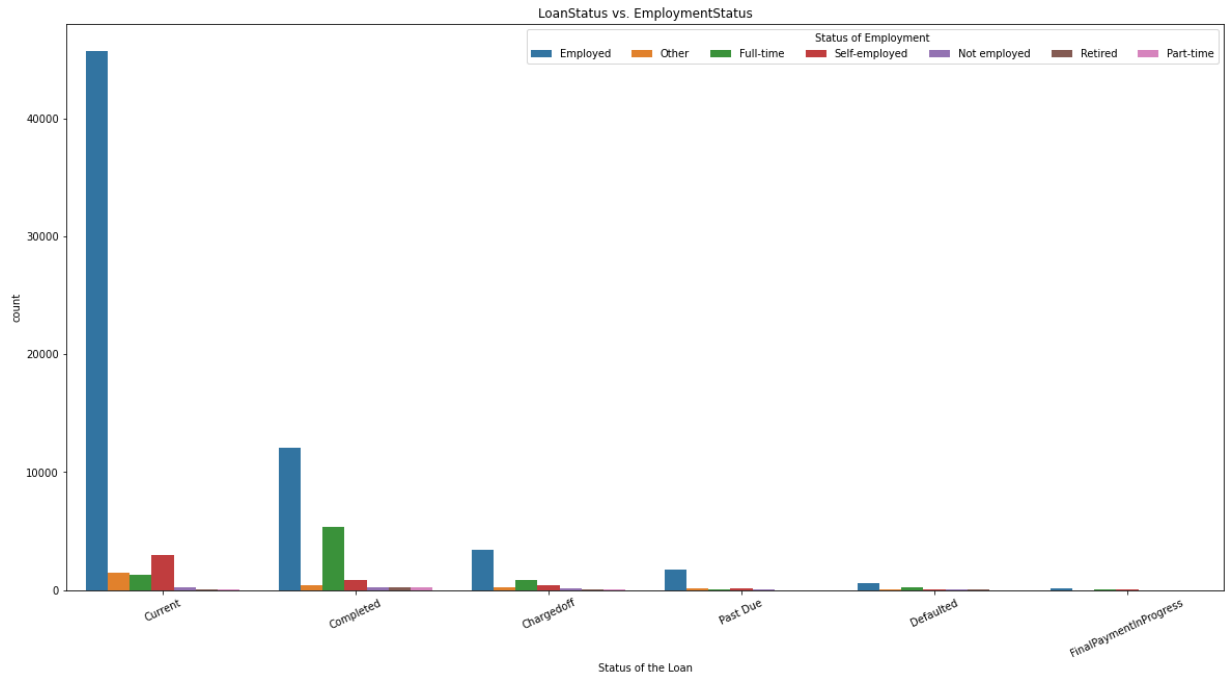


Relation between loan status and employment status

The status of the employment of the borrower seems not to have an impact on the outcome of the loan. but again the ones who take loans mostly are employed as also the ones who completed their loans are the full-time employed.

```
In [30]: plt.figure(figsize = [20,10])

sb.countplot(data = df_loans, x = 'LoanStatus', hue = 'EmploymentStatus', order = 
plt.legend(loc = 1, ncol = 8, title = 'Status of Employment')
plt.xticks(rotation = 25)
plt.xlabel('Status of the Loan')
plt.title('LoanStatus vs. EmploymentStatus');
```

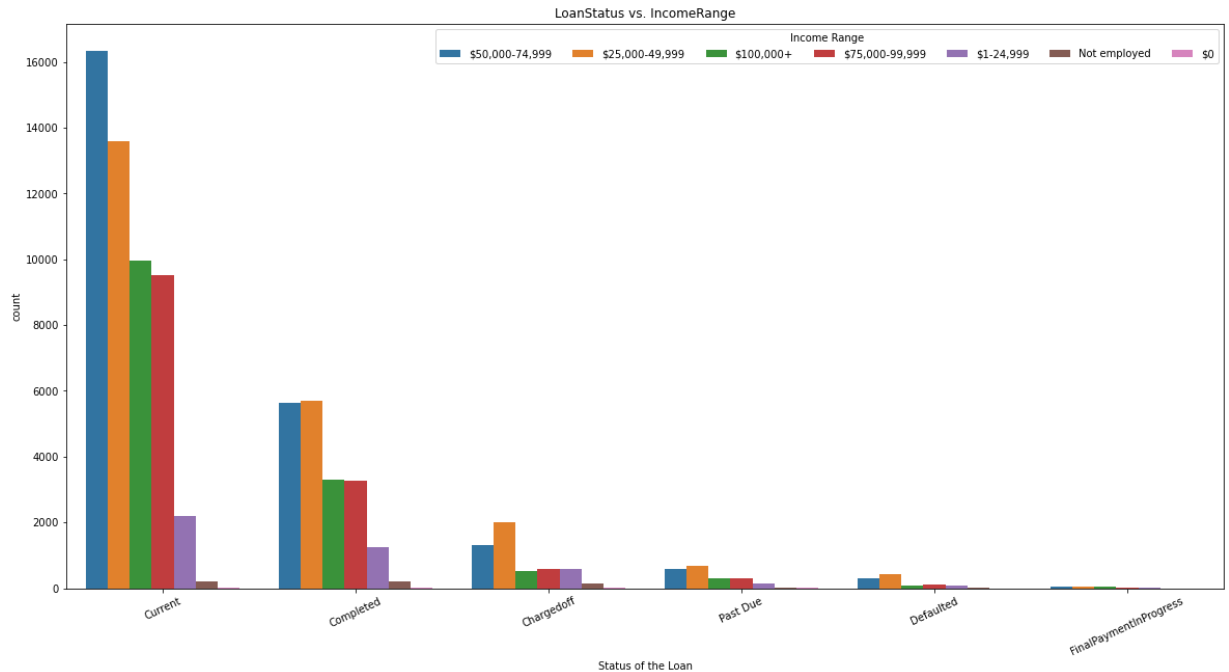


Relation between loan status and income range

The borrowers who have an Income Range of (25000 - 74999) seem to be the ones who get more loans and pay them on time.

```
In [31]: plt.figure(figsize = [20,10])

sb.countplot(data = df_loans, x = 'LoanStatus', hue = 'IncomeRange', order = df_loans['LoanStatus'].value_counts().index)
plt.legend(loc = 1, ncol = 8, title = 'Income Range')
plt.xticks(rotation = 25)
plt.xlabel('Status of the Loan')
plt.title('LoanStatus vs. IncomeRange');
```



Relation between prosper score and loan status

The Prosper Score seems to affect the outcome of the loan. So the highest number of borrowers with completed loans has a prosper score of 8, while the highest number of borrowers with defaulted and charged-off loans have a prosper score of 6. Notice that the most common prosper score for borrowers with loans that are past due payments is 6.

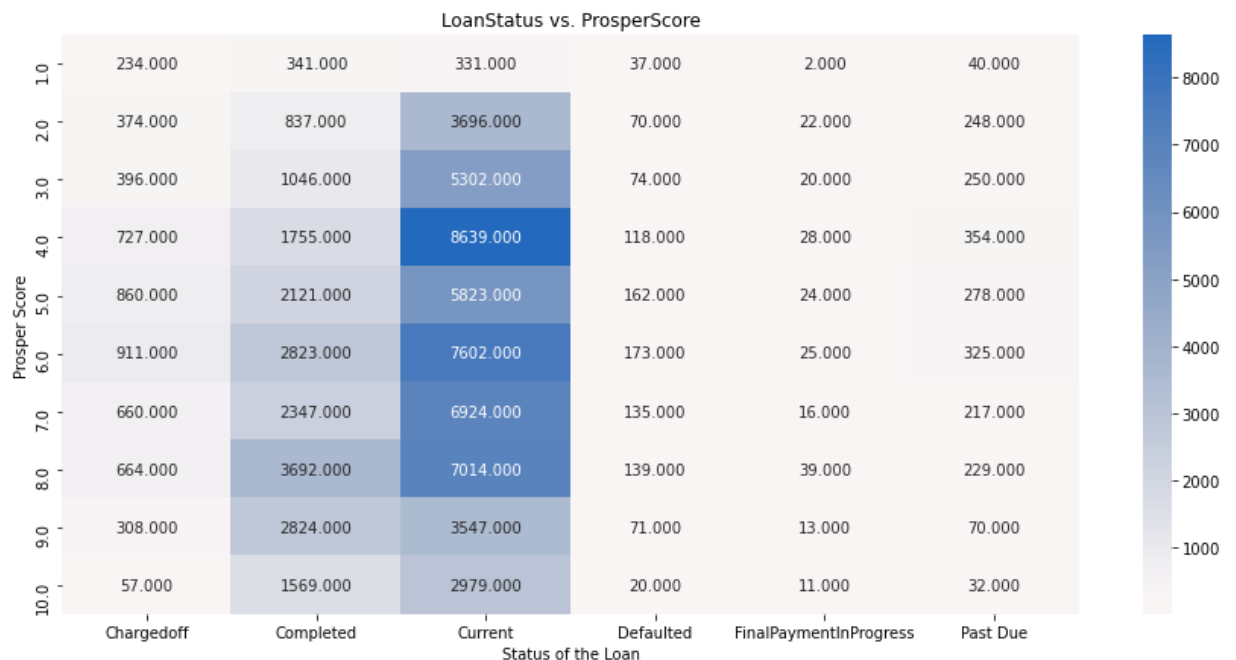

```
In [32]: # first I need to reshape the data
reshape = df_loans.groupby(['LoanStatus', 'ProsperScore']).size()
reshape = reshape.reset_index(name = 'count')
reshape = reshape.pivot(index = 'ProsperScore', columns = 'LoanStatus', values =
reshape
```

Out[32]:

LoanStatus	Chargedoff	Completed	Current	Defaulted	FinalPaymentInProgress	Past Due
ProsperScore						
1.0	234	341	331	37	2	40
2.0	374	837	3696	70	22	248
3.0	396	1046	5302	74	20	250
4.0	727	1755	8639	118	28	354
5.0	860	2121	5823	162	24	278
6.0	911	2823	7602	173	25	325
7.0	660	2347	6924	135	16	217
8.0	664	3692	7014	139	39	229
9.0	308	2824	3547	71	13	70
10.0	57	1569	2979	20	11	32

```
In [33]: plt.figure(figsize = [15,7])

sb.heatmap(reshape, annot = True, fmt = '.3f', cmap = 'vlag_r', center = 0)
plt.xlabel('Status of the Loan')
plt.ylabel('Prosper Score')
plt.title('LoanStatus vs. ProsperScore');
```

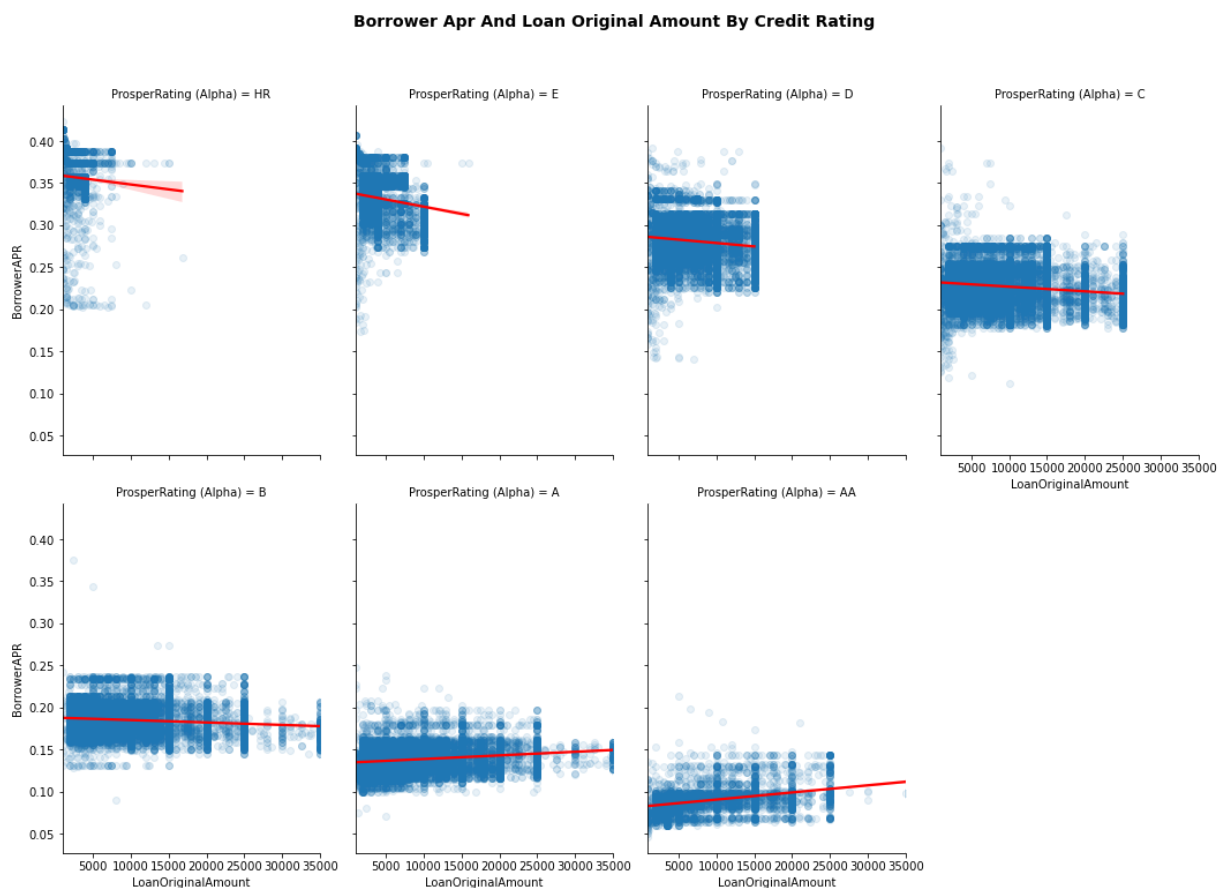


Multivariate Exploration

Prosper rating effect on relationship between borrower APR and loan original amount

The loan amount increases with a better rating, the borrower's APR decreases with a better rating. The relationship between borrower APR and loan amount raises from negative to slightly positive when the prosper ratings are increased from HR to A or better. Maybe because people with A or AA ratings tend to borrow more money, and pay on time.

```
In [34]: # Prosper rating effect on relationship of APR and Loan amount
g=sb.FacetGrid(data = df_loans, aspect = (14.70/4)/(10.27/2), height = 10.27/2, c
g.map(sb.regplot, 'LoanOriginalAmount', 'BorrowerAPR', x_jitter=0.04, scatter_kws
g.add_legend();
plt.suptitle('Borrower APR and loan original amount by credit rating'.title(), y=
plt.tight_layout();
```

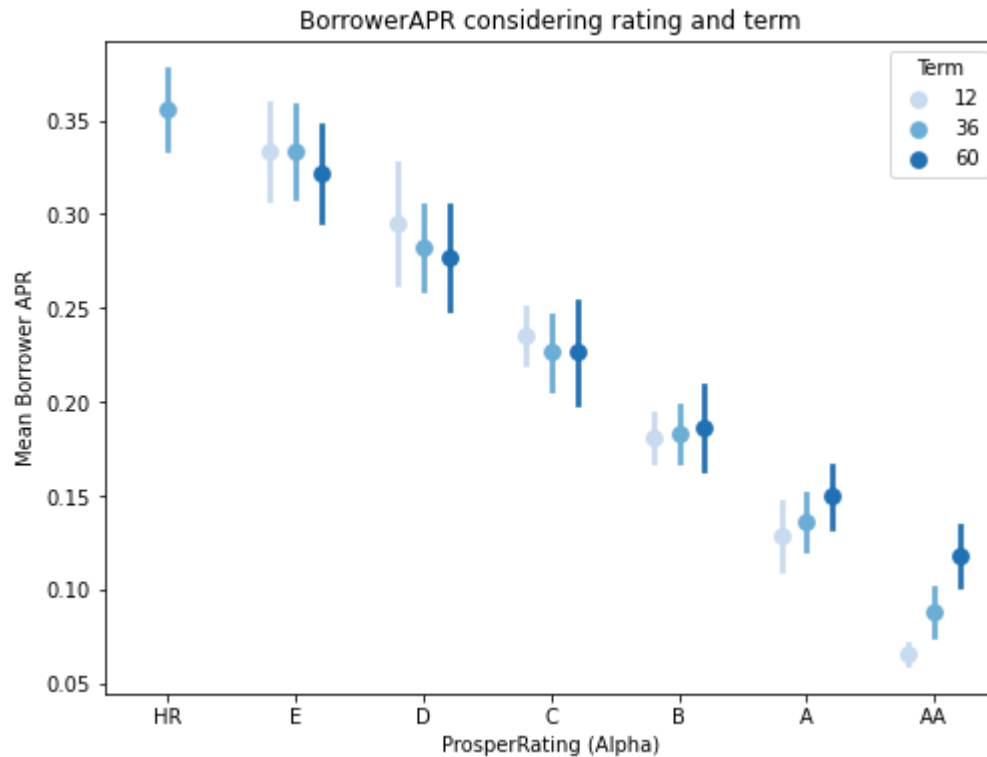


BorrowerAPR considering rating and term

Notice that for prosper rating from HR to D the borrower APR increases with the decrease of borrow term, then it starts to shift from C to AA rating.

```
In [35]: fig = plt.figure(figsize = [8,6])

ax = sb.pointplot(data = df_loans, x = 'ProsperRating (Alpha)', y = 'BorrowerAPR')
plt.title('BorrowerAPR considering rating and term')
plt.ylabel('Mean Borrower APR')
ax.set_yticklabels([],minor = True);
```

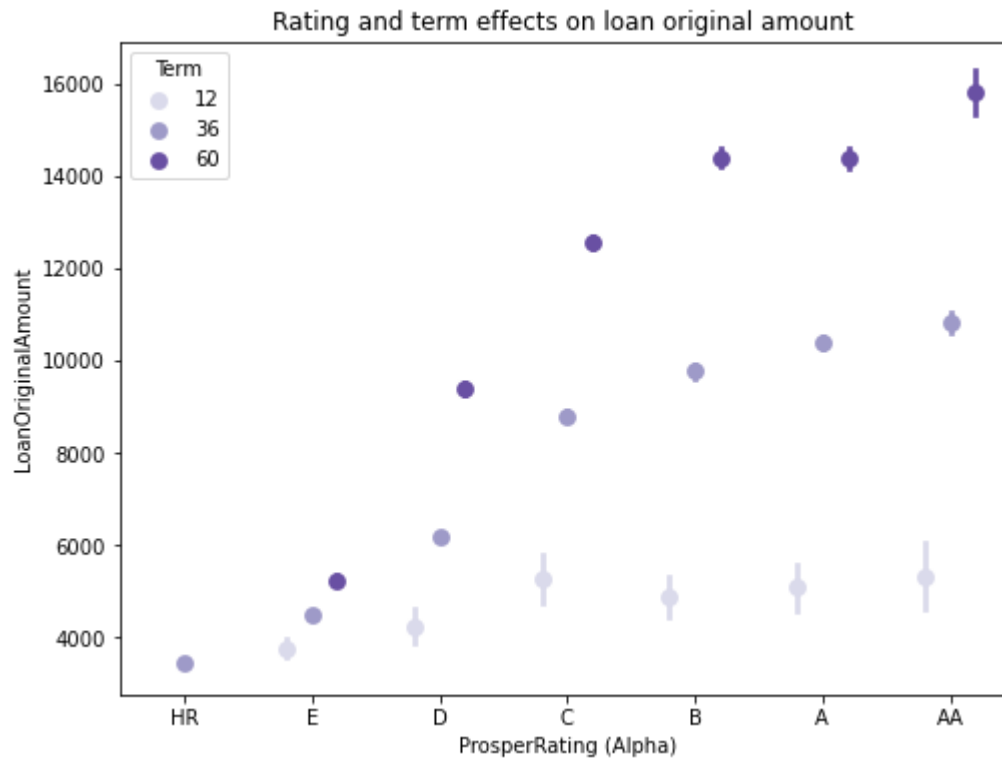


The rating and term effects on loan original amount

Notice that with a better prosper rating, the loan amount of all three terms increases.

```
In [36]: fig, ax = plt.subplots(figsize=[8,6])

sb.pointplot(data = df_loans, x = 'ProsperRating (Alpha)', y = 'LoanOriginalAmount')
plt.title('Rating and term effects on loan original amount');
```

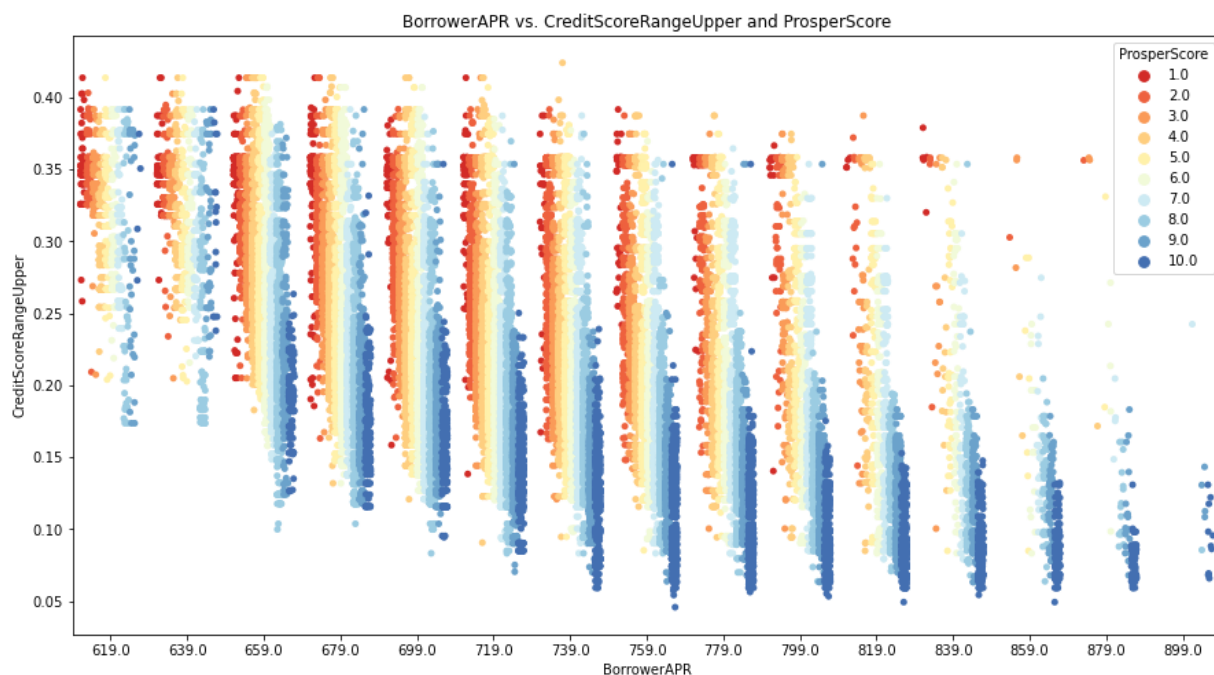


BorrowerAPR considering CreditScoreRangeUpper and ProsperScore

Notice that CreditScoreRangeUpper increase as BorrowerAPR decrease in the plots, this proves that CreditScoreRangeUpper and ProsperScore negatively correlated to BorrowerAPR.

```
In [37]: # BorrowerAPR vs. CreditScoreRangeUpper and ProsperScore
plt.figure(figsize = [15, 8])

sb.stripplot(data = df_loans, x = 'CreditScoreRangeUpper', y = 'BorrowerAPR', hue =
            'ProsperScore', palette = 'RdYlBu',
            dodge = True)
plt.title('BorrowerAPR vs. CreditScoreRangeUpper and ProsperScore')
plt.xlabel('BorrowerAPR')
plt.ylabel('CreditScoreRangeUpper');
```



Conclusion

After the demonstration, we answer the previously answered questions as follows:

- Prosper seem to give loans to all level of borrowers, but more is given to borrowers with a score of (C to A).
- We can see that the higher the borrowers rating the lower the annual rate of interest is charged (APR).
- APR is affected directly with the following attributes:

- Loan amount: when the amount increases the APR decrease.
- Term: as the most popular term is 36, it is also the most completed term which means it gets a higher score rating so as well the APR of the borrower decreases.
- Income: borrowers with an income range of (25000 - 74999) get more loans and also complete their loans on time, therefore gets a prosper rating of 8, so as well the APR is low for them.

- The APR also is low for borrowers who get large loans, but also make more than 100000+ as well as complete their loans on time.
- The most popular loans amount are between 2500 - 10000 as well as the term of 36 months, which is reasonable because small amounts are paid in a short time and preferable between borrowers.
- Borrowers who are employed tend to get more loans and pay on time and complete their loans therefore get higher scores.

Few insights on how Prosper could increase their profits from loans:

- Notice that most borrowers' income is between (25000 - 74000), so I think that prosper should focus their marketing plan on the middle-class segment.
- The most popular term of loans is the short ones, as the average loan amount is between 9000 dollars. I think that a marketing promotion plan should include activities the borrowers do like shopping for example to encourage more clients of that sort of loan.

In [38]: !jupyter nbconvert slide_deck_Loan_Data_from_Prosper.ipynb --to slides --post ser

```
[NbConvertApp] WARNING | Config option `kernel_spec_manager_class` not recogniz
ed by `NbConvertApp`.
[NbConvertApp] Converting notebook slide_deck_Loan_Data_from_Prosper.ipynb to s
lides
Traceback (most recent call last):
  File "C:\Users\Ahmed\anaconda3\Scripts\jupyter-nbconvert-script.py", line 10,
in <module>
    sys.exit(main())
  File "C:\Users\Ahmed\anaconda3\lib\site-packages\jupyter_core\application.p
y", line 270, in launch_instance
    return super(JupyterApp, cls).launch_instance(argv=argv, **kwargs)
  File "C:\Users\Ahmed\anaconda3\lib\site-packages\traitlets\config\application
n.py", line 845, in launch_instance
    app.start()
  File "C:\Users\Ahmed\anaconda3\lib\site-packages\nbconvert\nbconvertapp.py",
line 350, in start
    self.convert_notebooks()
  File "C:\Users\Ahmed\anaconda3\lib\site-packages\nbconvert\nbconvertapp.py",
line 524, in convert_notebooks
    self.convert_single_notebook(notebook_filename)
  File "C:\Users\Ahmed\anaconda3\lib\site-packages\nbconvert\nbconvertapp.py",
line 489, in convert_single_notebook
    output, resources = self.export_single_notebook(notebook_filename, resource
s, input_buffer=input_buffer)
  File "C:\Users\Ahmed\anaconda3\lib\site-packages\nbconvert\nbconvertapp.py",
line 418, in export_single_notebook
    output, resources = self.exporter.from_filename(notebook_filename, resource
s=resources)
  File "C:\Users\Ahmed\anaconda3\lib\site-packages\nbconvert\exporters\exporte
r.py", line 181, in from_filename
    return self.from_file(f, resources=resources, **kw)
  File "C:\Users\Ahmed\anaconda3\lib\site-packages\nbconvert\exporters\exporte
r.py", line 199, in from_file
    return self.from_notebook_node(nbformat.read(file_stream, as_version=4), re
sources=resources, **kw)
  File "C:\Users\Ahmed\anaconda3\lib\site-packages\nbconvert\exporters\html.p
y", line 119, in from_notebook_node
    return super().from_notebook_node(nb, resources, **kw)
  File "C:\Users\Ahmed\anaconda3\lib\site-packages\nbconvert\exporters\template
exporter.py", line 384, in from_notebook_node
    output = self.template.render(nb=nb_copy, resources=resources)
  File "C:\Users\Ahmed\anaconda3\lib\site-packages\jinja2\environment.py", line
1090, in render
    self.environment.handle_exception()
  File "C:\Users\Ahmed\anaconda3\lib\site-packages\jinja2\environment.py", line
832, in handle_exception
    reraise(*rewrite_traceback_stack(source=source))
  File "C:\Users\Ahmed\anaconda3\lib\site-packages\jinja2_compat.py", line 28,
in reraise
    raise value.with_traceback(tb)
  File "C:\Users\Ahmed\Udacity visualization project\output_toggle.tpl", line
5, in top-level template code
    {% extends 'slides_reveal.tpl' %}
jinja2.exceptions.TemplateNotFound: slides_reveal.tpl
```

In []: